Politechnika Krakowska
Wydział Inżynierii
Elektrycznej i Komputerowej
Katedra Automatyki i Informatyki E–1

PRACA DOKTORSKA

SYNTEZA ZAAWANSOWANYCH TECHNIK WIZYJNYCH
W INTELIGENTNYM SYSTEMIE NADZORU
OBIEKTÓW UŻYTKU PUBLICZNEGO

mgr inż. Adam Surówka

Dyscyplina: Automatyka, elektronika, elektrotechnika
i technologie kosmiczne

Promotor: prof. dr hab. inż. Janusz Gołdasz

Promotor pomocniczy: dr inż. Grzegorz Litawa

Kraków 2024

Składam serdeczne podziękowania mojemu promotorowi prof. dr hab. inż. Januszowi Gołdaszowi, za wszystkie sugestie, pracę włożoną w konsultowanie rozprawy oraz za stworzenie warunków sprzyjających jej powstawaniu.

Wyrazy wdzięczności kieruję również do ks. proboszcza Eugeniusza Mrożka, za umożliwienie realizacji badań empirycznych na terenie obiektów sakralnych w Parafii Najświętszej Maryi Panny Częstochowskiej w Nowym Sączu.

Dziękuję również moim kolegom i koleżankom za liczny udział i zaangażowanie w tworzenie baz danych wideo, przeznaczonych do testowania opracowanego systemu IVSS.

Pracę tę dedykuję mojej mamie Janinie oraz siostrze Annie, dziękując im za wyrozumiałość i wsparcie okazywane podczas przygotowywania rozprawy.

Spis treści

Wykaz oznaczeń i skrótów I
Wykaz zmiennych IV
1 Wprowadzenie 1
2 Zaawansowane techniki analizy sygnału wizyjnego 4
2.1 Wstępne przetwarzanie obrazów 4
2.2 Kamery monitoringowe 7
2.3 Kalibracja kamer 9
2.4 Detekcja i klasyfikacja obiektów 10
2.4.1 Adnotacja obiektów 11
2.4.2 Detektory obiektów w ujęciu historycznym 11
2.4.3 Detektory tradycyjne oparte na technikach analitycznych 12
2.4.4 Detektory dwustopniowe oparte na DL 12
2.4.5 Detektory jednostopniowe oparte na DL 13
2.4.6 Bazy danych uczących 16
2.4.7 Podsumowanie 18
2.5 Techniki optycznego rozpoznawania znaków alfanumerycznych 20
2.5.1 Techniki ANPR 20
2.5.2 Silniki OCR 21
2.5.3 Podsumowanie 22
2.6 Estymacja sylwetki człowieka 23
2.6.1 Graficzna reprezentacja sylwetki człowieka 23
2.6.2 Klasyfikacja technik HPE 24
2.6.3 Przegląd wybranych bibliotek HPE 25
2.6.4 Podsumowanie 27
2.7 Śledzenie obiektów 29
2.7.1 Wzorce cech 29
2.7.2 Klasyfikacja technik śledzenia 30
2.7.3 Przetwarzanie danych 32
2.7.4 Uczenie maszynowe 32
2.7.5 Podsumowanie 33
2.8 Techniki analizy aktywności człowieka 33
2.8.1 Typy rozpoznawanych aktywności 34
2.8.2 Dynamika aktywności 34
2.8.3 Klasyfikacja technik HAR 35
2.8.4 Podsumowanie 37
2.9 Sprzętowe jednostki obliczeniowe 38
2.10 Wskaźniki oceny jakości 39
2.11 Podsumowanie 43

3 Inteligentne systemy nadzoru wizyjnego 45
3.1 Obszary zastosowań 45
3.2 Ogólna klasyfikacja systemów IVSS 46
3.3 Architektury systemów IVSS 47
3.4 Przegląd rozwiązań w obszarze IVSS 48
3.4.1 Analiza stosowanych technik 48
3.4.2 Klasyfikacja zdarzeń alarmowych 58
3.5 Podsumowanie 61
3.6 Luki badawcze 63
4 Opis projektowanego systemu IVSS 65
4.1 Koncepcja przepływu informacji wizyjnej 65
4.2 GUI 67
4.3 Stanowisko badawcze 69
4.4 Źródła sygnału wizyjnego 70
4.4.1 Testowe nagrania wideo 70
4.4.2 Symulacje 3D 74
4.5 Podsumowanie 75
5 Wstępne przetwarzanie obrazu 77
5.1 Kalibracja kamery 77
5.2 Korekta jasności obrazu 80
5.3 Detekcja obiektów 84
5.3.1 OpenPose 84
5.3.2 YOLO v4 oraz YOLO v4 tiny 87
5.3.3 Tesseract OCR 92
5.4 Śledzenie obiektów 92
5.5 Kalibracja sceny 96
5.6 Podsumowanie 108
6 Algorytmy detekcji zdarzeń alarmowych 111
6.1 Identyfikacja pojazdów 111
6.2 Obecność zwierząt 118
6.3 Sabotaż kamery 121
6.4 Nielegalne parkowanie 129
6.5 Wejście (wyjście) ze strefy 133
6.6 Naruszenie wirtualnej granicy 137
6.7 Przeludnienie 140
6.8 Bieg oraz analiza aktywności tłumu 143
6.9 Włóczęgostwo 148
6.10 Naruszenie wirtualnej strefy 151
6.11 Pożar 155

6.12 Upadek 156
6.13 Kradzież 164
6.14 Przemoc 168
6.15 Synteza zaawansowanych technik wizyjnych 175
6.16 Podsumowanie 179
7 Podsumowanie 185
Dorobek naukowy 189
Spis ilustracji 190
Spis tabel 196
Literatura 198
Streszczenie 209
Summary 210

Wykaz oznaczeń i skrótów

2D – Two Dimensional image (obraz dwuwymiarowy),

3D – Three Dimensional image (obraz trójwymiarowy),

AB – Anchor Box (czworokątna, ortogonalna ramka, stanowiąca adnotację propozycji detekcji obiektów na obrazie),

AI – Artificial Inteligence (sztuczna inteligencja),

ANN – Artificial Neural Network (sztuczna sieć neuronowa),

ANPR – Automatic Number Plate Recognition (automatyczne rozpoznawanie numerów tablic rejestracyjnych),

API – Application Programming Interface (interfejs programistyczny aplikacji),

ASIC –Application-Specific Integrated Circuit (specjalizowany układ scalony),

BB – Bounding Box (czworokątna, ortogonalna ramka, stanowiąca jedną z metod adnotacji prawdziwych detekcji obiektów na obrazie),

BSD – Berkeley Software Distribution Licenses (jedna z popularnych licencji programistycznych typu open-source),

CCTV – Closed Circuit Television (monitoring przemysłowy),

CIoU – Complete Intersection over Union (domyślna funkcja strat w używana procesie uczenia modelu YOLO v4 tiny),

CNN – Convolutional Neural Network (konwolucyjna sieć neuronowa),

CPU – Central Processing Unit (procesor główny),

CSP – Cross Stage Partial connections (technika projektowania sieci neuronowych),

CUDA – Compute Unified Device Architecture (uniwersalna architektura procesorów wielordzeniowych GPU),

cuDNN – CUDA Deep Neural Network (biblioteka programistyczna zaprojektowana w celu przyspieszania operacji związanych z użyciem głębokich sieci neuronowych na procesorach GPU),

CV – Computer Vision (widzenie komputerowe),

DL – Deep Learning (uczenie maszynowe głębokie),

DNN – Deep Neural Network (głęboka sieć neuronowa),

DPM – Deformable Part-based Model (detektor obiektów na obrazie),

EPTZ – Electronic Pan-Tilt-Zoom (typ kamery monitoringowej umożliwiający programową zmianę kąta widzenia i zbliżenia),

FFN – Feed Forward Neural Network,

FHD – Full High Definition (rozdzielczość obrazu 1920 x 1080 px),

FPGA – Field-Programmable Gate Array (programowalny układ logiczny),

FPN – Feature Pyramid Network (sztuczna sieć piramidy cech),

GAN – Generative Adversarial Networks (generatywne sieci przeciwstawne),

GNU GPL – Gnu General Public License (licencja wolnego i otwartego oprogramowania),

GPU – Graphics Processing Unit (procesor graficzny),

GUI – Graphical User Interface (graficzny interfejs użytkownika),

HAR – Human Activity Recognition (rozpoznawanie aktywności człowieka),

HOG – Histogram of Oriented Gradients (histogram gradientów zorientowanych),

HPE – Human Pose Estimation (estymacja sylwetki człowieka),

ILSVRC – ImageNet Large Scale Visual Recognition Challenge (popularny konkurs z dziedziny przetwarzania obrazów),

IP – Internet Protocol address (adres protokołu internetowego),

IPT – Image Processing Toolbox (moduł przetwarzania obrazów dostępny w środowisku MATLAB),

IR – Infrared (światło podczerwone),

IVSS – Intelligent Video Surveillance System (inteligentny system nadzoru wideo),

LAB – nazwa przestrzeni barw,

LSTM – Long Short-Term Memory (typ rekurencyjnej sztucznej sieci neuronowej),

LUT – Look-Up Table (struktura danych zawierająca zdefiniowane pary wejść i wyjść w celu przyspieszenia operacji obliczeniowych),

MDA – Mosaic Data Augmentation (technika augmentacji zbioru danych uczących),

ML – Machine Learning (uczenie maszynowe),

MOT – Multiple Object Tracking (śledzenie wielu obiektów),

MPPE – Multi Person Pose Estimation (estymacja wielu sylwetek człowieka),

MS-COCO – Microsoft Common Objects in Context (popularna baza danych z dziedziny CV),

MSIM – Multi Stage Identity Matching (moduł wielostopniowego dopasowania identyfikacji),

NMS – Non Maximum Suppression (mechanizm eliminacji redundantnych detekcji),

OCR – Optical Character Recognition (optyczna klasyfikacja znaków alfanumerycznych),

OS – Operating System (system operacyjny),

PGA – Pose Guided Alignment (moduł dopasowania sylwetek),

P-NMS – Pose Non-Maximum-Suppression (mechanizm eliminacji redundantnych detekcji sylwetek osób),

PSM – Page Segmentation Mode (lista trybów segmentacji strony silnika Tesseract OCR),

PTZ – Pan-Tilt-Zoom (typ kamery monitoringowej umożliwiający mechaniczną zmianę kąta widzenia i zbliżenia),

RCNN – Regjon-based Convolutional Neural Network (konwolucyjna sieć neuronowa oparta na rejonach),

RNN – Recurent Neural Network (rekurencyjna sieć neuronowa),

ROI – Region Of Interest (obszar obrazu objęty analizą),

RPN – Region Proposal Network (sztuczna sieć obszaru propozycji),

SAT – Self Adversarial Training (technika augmentacji zbioru danych uczących),

SIKR – Symmetric Integral Keypoint Regression (regresja symetryczna punktu kluczowego),

SPE – Spatial Pyramid Pooling (nazwa warstwy sieci neuronowej),

SPPE – Single Person Pose Estimation (estymacja pojedynczej sylwetki człowieka),

SPPNET – Spatial Pyramid Pooling Network (dwustopniowy detektor obiektów oparty na DL),

SSD –Single Shot MultiBox Detector (jednostopniowy detektor obiektów oparty na DL),

SVM – Support Vector Machine (maszyna wektorów nośnych),

TL – Transfer Learning (uczenie transferowe),

TPU – Tensor Processing Unit (procesor tensorowy),

VAE – Variational Autoencoder (autoenkodery wariacyjne),

VJ – Viola Jones Detector (detektor obiektów opracowany przez Viola Jones'a),

VOC – Visual Object Classes (popularny zbiór danych w dziedzinie CV),

VSS – Video Surveillance System (system nadzoru wideo),

Wi-Fi – Wireless Fidelity (grupa standardów radiowej sieci bezprzewodowej),

YOLO – You Only Look Once (rodzina jednostopniowych detektorów obiektów).

Wykaz zmiennych

A – Accuracy (wskaźnik dokładności),

a, b, c, d – współczynniki funkcji automatycznej korekty jasności obrazu,

AP – Average Precision (średnia precyzja dla jednej klasy obiektów),

A_r – rzeczywisty obszar lokalizacji BB obiektu na obrazie,

AR – stosunek szerokość i wysokość obrazu,

AUC – *Area Under the Curve* (pole powierzchni pod krzywą *ROC*),

BB_h – wysokość BB,

BB_w – szerokość BB,

B_p – obszar BB generowanego w procesie predykcji,

C – punkt główny kamery,

C_h – wysokość montażu kamery nad płaszczyzną podłoża,

c_x, c_y – współrzędne punktu głównego obrazu,

D – liczba wszystkich śledzonych osób i pojazdów ($D_o + D_p$),

d_1 – odległości od punktu P_1' do punktu C,

d_2 – odległości od punktu P_2' do punktu C,

d_3 – odległości od punktu P_3' do punktu C,

d_{calc} – obliczona odległość w 3D pomiędzy punktami P_{d1} oraz P_{d2},

D_f – liczba wszystkich detekcji ognia,

$dist_1$ – odległość analizowanej osoby od kamery,

$dist_2$ – odległość pomiędzy dwoma osobami,

D_k – liczba wszystkich śledzonych osób z zapisanymi modelami szkieletowymi,

D_{mm} – przekątna sensora optycznego kamery w mm,

D_o – liczba wszystkich śledzonych osób,

D_p – liczba wszystkich detekcji pojazdów,

d_{real} – rzeczywista odległość w przestrzeni 3D pomiędzy punktami P_{d1} oraz P_{d2},

D_t – liczba wszystkich detekcji tablic rejestracyjnych,

D_z – liczba wszystkich detekcji zwierząt,

EOF – znacznik końca wektora,

f – wartość ogniskowej kamery,

F_1 – średnia harmoniczna Pr oraz R,

F_a – flaga aktywności funkcji analizy tłumu,

F_b – flaga aktywności funkcji detekcji biegu,

F_g – flaga przekroczenia linii wirtualnej,

FN – False Negative (klasyfikacja fałszywie negatywna),

FP – False Positive (klasyfikacja fałszywie pozytywna),

FPR – False Positive Rate (wskaźnik negatywnych detekcji),

FPS – Frames Per Second (szybkość przetwarzania danych wizyjnych, wyrażana w obrazach na sekundę),

F_w – flaga wejścia (wyjścia) osoby ze strefy,

f_x, f_y – wartości ogniskowej kamery w osiach *x* oraz *y* macierzy obrazu,

H – liczba elementów wektora historii śledzonej osoby,

height – wysokość obrazu,

H_{mm} – wysokości czujnika optycznego kamery,

i – licznik pętli iteracyjnych,

IoU – Intersection over Union (współczynnik podobieństwa Jaccarda),

j – wartość średniej jasności obrazu w danym ujęciu,

j_m – wartość wzorcowa średniej jasności obrazu,

K – macierz kamery,

k – współczynnik korekty D_{mm},

k_1, k_2, k_3 – współczynniki zniekształceń radialnych kamery,

K_p – liczba punktorów szkieletu śledzonej osoby,

k_r – długość kręgosłupa,

L_1 – licznik kolejnych negatywnych detekcji sabotażu,

L_2 – licznik pozytywnych detekcji pojazdu w ROI,

L_3 – licznik pozytywnych detekcji wejścia (wyjścia) osoby ze strefy,

L_4 – licznik biegnących osób,

L_5 – licznik pozytywnych detekcji włóczęgostwa,

L_6 – licznik pozytywnych detekcji upadku,

L_7 – licznik pozytywnych detekcji kradzieży,

L_8 – licznik pozytywnych detekcji przemocy,

L_k – liczba pikseli niezerowych maski tła obiektu zainteresowania (algorytm detekcji kradzieży),

lp – długość lewego podudzia,

L_p – liczba pikseli niezerowych maski tła (algorytm detekcji sabotażu kamery),

lu – długość lewego uda,

mAP – Mean Average Precision (średnia precyzja dla wszystkich klas obiektów),

n – liczba klas,

N – Negatives (klasyfikacja negatywna),

O – obszar przemieszczania się osoby na obrazie,

O' – obszar przemieszczania się osoby w przestrzeni rzeczywistej,

P – Positives (klasyfikacja pozytywna),

P_0 – punkt początkowy globalnego układu współrzędnych (algorytm kalibracji sceny),

P_1, P_2, P_3 – punkty referencyjne oznaczane na obrazie,

p_1, p_2, – współczynniki zniekształceń stycznych kamery,

P_1', P_2', P_3' – rzuty punktów referencyjnych P_1, P_2 oraz P_3 na podłożu obserwowanej sceny,

P_1'', P_2'', P_3'' – współrzędne globalne punktów końców linii rzutowania,

p_a – liczba wszystkich pikseli składających się na zaznaczoną strefę,

P_{d1}, P_{d2} – punkty testowe oznaczane na obrazie,

P_f – punkt przecięcia ogniskowej z płaszczyzną obrazowania,

P_g –punkt uzyskany w procesie rzutowania,

p_h –wysokość pojedynczego piksela obrazu,

P_l –punkt w lokalnym układzie płaszczyzny obrazowania,

pp – długość prawego podudzia,

PPV –Positive Predictive Value (wartość predykcyjna dodatnia),

Pr – Precision (wskaźnik precyzji),

P_r – punkt przecięcia linii rzutu z nową płaszczyzną podłoża (R_z),

P_s – punkt główny obrazu (środek geometryczny),

pu – długość prawego uda,

P_V – średnia szybkość zmian zestawów kątów,

p_w – szerokość pojedynczego piksela obrazu,

R – Recall (wskaźnik czułości),

R_1 – licznik stref wirtualnych (algorytm identyfikacji pojazdów),

R_2 – licznik stref wirtualnych (algorytm detekcji nielegalnego parkowania),

R_3 – licznik granic wirtualnych (algorytm detekcji wejścia (wyjścia) ze strefy),

R_4 – licznik granic wirtualnych (algorytm detekcji naruszenia wirtualnej granicy),

R_5 – licznik stref wirtualnych (algorytm detekcji naruszenia wirtualnej strefy),

R_6 – licznik stref wirtualnych (algorytm detekcji kradzieży),

ROC – *Receiver Operating Characteristic* (krzywa ilustrująca działanie klasyfikatora),

R_z – równanie płaszczyzny rzutowania,

s_1, s_2, s_3, s_4 – współczynniki zniekształceń pryzmatycznych kamery,

sabotaż – flaga detekcji zdarzenia sabotażu,

sensor_size – przekątna sensora optycznego kamery w calach,

sn – średnia długość nóg,

T – Total measurements (całkowita liczba pomiarów rzeczywistych),

T_h – maksymalna odległość tablicy rejestracyjnej od kamery w poziomie,

TN – True Negative (klasyfikacja fałszywie negatywna),

TP – True Positive (klasyfikacja prawdziwie pozytywna),

TPR – True Positive Rate (wskaźnik klasyfikacji pozytywnie prawdziwych),

t_r – maksymalna wartość procentowa zmian w zaznaczonej strefie,

T_v – maksymalna odległość tablicy rejestracyjnej od kamery w pionie,

T_θ – wektor miar kątów,

V – szybkość przemieszczania się osoby,

V_θ – wektor szybkości zmian kątów,

W_1 – wartość progowa maksymalnej liczby okluzji obiektu,

W_{10} – wartość progowa obszaru przemieszczania się osoby,

W_{11} – wartość progowa szybkości przemieszczania się osoby (algorytm detekcji włóczęgostwa),

W_{12} – wartość progowa liczby elementów wektora historii śledzonej osoby,

W_{13} – wartość progowa pozytywnych detekcji włóczęgostwa,

W_{14} – wartość progowa stosunku długości kręgosłupa do długości nóg,

W_{15} – wartość progowa liczby pozytywnych detekcji upadku osoby,

W_{16} – wartość progowa szybkości przemieszczania się osoby (algorytm detekcji upadku),

W_{17} – wartość progowa liczby pikseli niezerowych (algorytm detekcji kradzieży),

W_{18} – wartość progowa liczby pozytywnych detekcji kradzieży,

W_{19} – wartość progowa odległości osoby od kamery,

W_2 – wartość progowa zmiany jasności obrazu,

W_{20} – wartość progowa szybkości zmiany kątów,

W_{21} – wartość progowa dystansu pomiędzy osobami,

W_{22} – wartość progowa liczby pozytywnych detekcji przemocy,

W_3 – wartość progowa liczby pikseli niezerowych (algorytm detekcji sabotażu kamery),

W_4 – wartość progowa liczby negatywnych detekcji sabotażu,

W_5 – wartość progowa *IoU*,

W_6 – wartość progowa liczby pozytywnych detekcji pojazdu w określonym obszarze ROI,

W_7 – wartość progowa liczby osób,

W_8 – wartość progowa szybkości przemieszczania się osoby (algorytm detekcji biegu),

W_9 – wartość progowa dopuszczalnej liczby biegnących osób,

W_{BB} – wektor nowych detekcji BB,

width – szerokość obrazu,

W_{mm} – szerokość czujnika optycznego kamery,

W_s – wektor śledzonych obiektów,

Z_n – wektor wskaźników zniekształceń kamery,

α, β, γ – współczynniki korekty jasności obrazu,

δ – kąt nutacji (nachylenia kamery do płaszczyzny podłoża),

λ – maksymalny kąt widzenia tablicy rejestracyjnej w osi pionowej,

μ – maksymalny kąt widzenia tablicy rejestracyjnej w osi poziomej,

σ_1, σ_2 – wyznaczniki oceny jakości algorytmu kalibracji sceny,

τ_x, τ_y – współczynniki zniekształceń perspektywicznych kamery,

ψ – kąt obrotu własnego kamery (względem głównej osi optycznej),

Δj – wartość zmiany średniej jasności obrazu w danym ujęciu.

1 Wprowadzenie

Systemy nadzoru wizyjnego VSS [1] ewoluowały technicznie przez cztery generacje. Początki pierwszej generacji datowane są na późne lata 40. ubiegłego wieku, kiedy to wprowadzono analogowe systemy telewizji przemysłowej CCTV. Stan ówczesnej technologii ograniczał pewne możliwości, zwłaszcza w zakresie procesów nagrywania i dystrybucji obrazu. Problemy te zostały rozwiązane w drugiej generacji, kiedy to w latach 70. zaczęto realizować nagrywanie filmów z monitoringu, głównie przy użyciu kaset magnetycznych. Z końcem lat 80. nastąpiła ewolucja obrazowania cyfrowego, przekształcając systemy VSS do poziomu trzeciej generacji i oferując dwie główne innowacje. Po pierwsze, kompresja i dystrybucja obrazu stały się bardziej efektywne i ekonomiczne. Po drugie, do systemów monitoringu wprowadzono algorytmy widzenia komputerowego, oferując funkcje półautomatyczne, takie jak rozpoznawanie i śledzenie obiektów. Efektywność systemów VSS jest silnie skorelowana ze zdolnością analizy obrazów w czasie rzeczywistym, co było trudne do osiągnięcia z uwagi na ciągle rosnącą liczbę instalowanych kamer. Wyniki badań [2] wskazują, że po zaledwie dwudziestu minutach obserwacji obrazu z pojedynczej kamery, przeciętny operator monitoringu traci 95% zdolności koncentracji. Jest to spowodowane naturalnym znużeniem i zmęczeniem operatora, które prowadzi do obniżenia jego czujności, wymaganej do prawidłowej klasyfikacji aktywności na scenie. W celu przetwarzania danych ze wszystkich kamer w czasie rzeczywistym oraz wyeliminowania wpływu zmęczenia operatora, rozpoczęto implementowanie w pełni automatycznych algorytmów analizy obrazu, które są oparte na technikach ML. W rezultacie, od roku 2010, mówi się o rozwoju czwartej generacji systemów IVSS [1]. W tym wydaniu systemy automatycznie realizują detekcję zdarzeń potencjalnie niebezpiecznych i sygnalizują je operatorowi, celem podjęcia ostatecznej decyzji i działań.

Przedmiotem badań niniejszej rozprawy jest synteza istniejących, zaawansowanych technik przetwarzania i analizy obrazu w celu opracowania inteligentnej aplikacji komputerowej, realizującej automatyczne wykrywanie wybranych zdarzeń alarmowych. Poprzez definicje zdarzeń alarmowych należy rozumieć szczególne przypadki akcji i/lub interakcji osób oraz wybranych klas zwierząt i obiektów, mogących stanowić potencjalne zagrożenie bezpieczeństwa ludzi i dóbr materialnych, znajdujących się w monitorowanej przestrzeni publicznej. Projektowany system IVSS będzie przeznaczony do detekcji piętnastu zdarzeń alarmowych: identyfikacja pojazdów (ANPR), sabotaż kamery, niepożądana obecność zwierząt, nielegalne parkowanie, analiza wejścia (wyjścia) ze strefy, naruszenie wirtualnej granicy, przeludnienie, bieg, analiza aktywności tłumu, włóczęgostwo, naruszenie wirtualnej strefy, pożar, upadek, kradzież oraz przemoc. Osiągnięcie pożądanych rezultatów będzie możliwe poprzez dobór, parametryzację, dopasowanie i integrację algorytmów realizujących zadania z zakresu detekcji, klasyfikacji,

śledzenia oraz analizy zachowań osób, a także wybranych klas zwierząt i obiektów, znajdujących się w polu widzenia kamery. Wdrożony system ma umożliwić komunikację z operatorem poprzez intuicyjny w obsłudze GUI. Praktyczne zastosowanie opracowanej aplikacji, będzie możliwe dzięki osiągnięciu przetwarzania danych wizyjnych w czasie rzeczywistym. Skuteczność opracowanego algorytmu zostanie zweryfikowana przy użyciu symulacji 3D, jak również w procesie badań empirycznych przeprowadzonych w warunkach ustalonych.

Powyższy zakres prac wiąże się z postawionymi w rozprawie tezami głównymi:

1. **Możliwe jest opracowanie metody syntezy zaawansowanych technik widzenia komputerowego opartych na modelach uczenia maszynowego, w celu stworzenia całościowego i skutecznego systemu nadzoru przestrzeni publicznych, zdolnego do automatycznej detekcji i klasyfikacji wybranych zdarzeń alarmowych w środowisku rzeczywistym, z wykorzystaniem sygnału z konwencjonalnych kamer monitoringu wizyjnego.**
2. **Opracowane algorytmy przetwarzania danych wizyjnych, ekstrakcji cech, identyfikacji, klasyfikacji oraz śledzenia umożliwiają efektywne wykrywanie i raportowanie kluczowych z punktu widzenia bezpieczeństwa publicznego aktywności związanych z zachowaniem i interakcjami osób oraz wybranych klas zwierząt i obiektów.**

Układ treści rozprawy jest następujący: przegląd zaawansowanych technik analizy sygnału wizyjnego przedstawiono w rozdziale drugim. Rozdział ten ukierunkowany jest na zagadnienia związane z pozyskiwaniem i wstępną analizą obrazu, a także zawiera przegląd rozwiązań stosowanych do detekcji i klasyfikacji obiektów, estymacji sylwetki człowieka, śledzenia obiektów w polu widzenia kamery oraz analizy zachowań osób, a także odniesienie do najczęściej stosowanych kryteriów oceny jakości w dziedzinie CV. Rozdział trzeci zawiera odniesienie do istoty i problematyki inteligentnych systemów nadzoru wizyjnego oraz szczegółowy przegląd istniejących systemów IVSS, nazywanych dalej systemami referencyjnymi. W rozdziale czwartym omówiono główne aspekty projektowanego systemu, które dotyczą schematu przepływu informacji w warstwie aplikacji, parametryzacji stanowiska badawczego pod kątem sprzętowym oraz programowym, a także typów źródeł sygnału wizyjnego, stosowanych w celu realizacji badań. Rozdział piąty przedstawia szczegółowy opis implementacji algorytmów wstępnego przetwarzania obrazu, opracowanych przez autora, w celu realizacji zadań z zakresu kalibracji kamery, kalibracji sceny, korekty jasności obrazu, detekcji, klasyfikacji oraz śledzenia obiektów. W oparciu o zestawione elementy wiedzy, w rozdziale szóstym przedstawiono szczegółowy opis piętnastu algorytmów detekcji zdarzeń alarmowych opracowanych przez autora, w tym przyjęte cele i założenia, zastosowaną logikę operacyjną, testy oraz wyniki

przeprowadzonych badań, jak również odniesienie do istniejących tego typu rozwiązań w literaturze. Ponadto rozdział ten zawiera szczegółowe informacje dotyczące metody syntezy algorytmów w warstwie aplikacji, z uwzględnieniem aspektów wielowątkowości oraz wynikami testów wydajnościowych systemu pracującego w konfiguracji wielozadaniowej. Rozdział siódmy zawiera podsumowanie niniejszej rozprawy, w którym autor określa stopień realizacji zadań badawczych, przedstawia osiągnięte rezultaty, odnosi się do tez głównych, jak również wskazuje wprowadzone rozwiązania oryginalne.

2 Zaawansowane techniki analizy sygnału wizyjnego

Istota dziedziny CV [3] polega na dążeniu do realizacji procesu inżynierii odwrotnej. Proces ten polega na opisie istniejącego świata materialnego, poprzez odtworzenie jego istotnych właściwości wizualnych (np. kształty, kolory) w postaci cyfrowej. Zastosowane do tego celu algorytmy programistyczne oraz narzędzia i architektury sprzętowe umożliwiają wizualną percepcję rzeczywistości w sposób zbliżony do działania zmysłu wzroku człowieka.

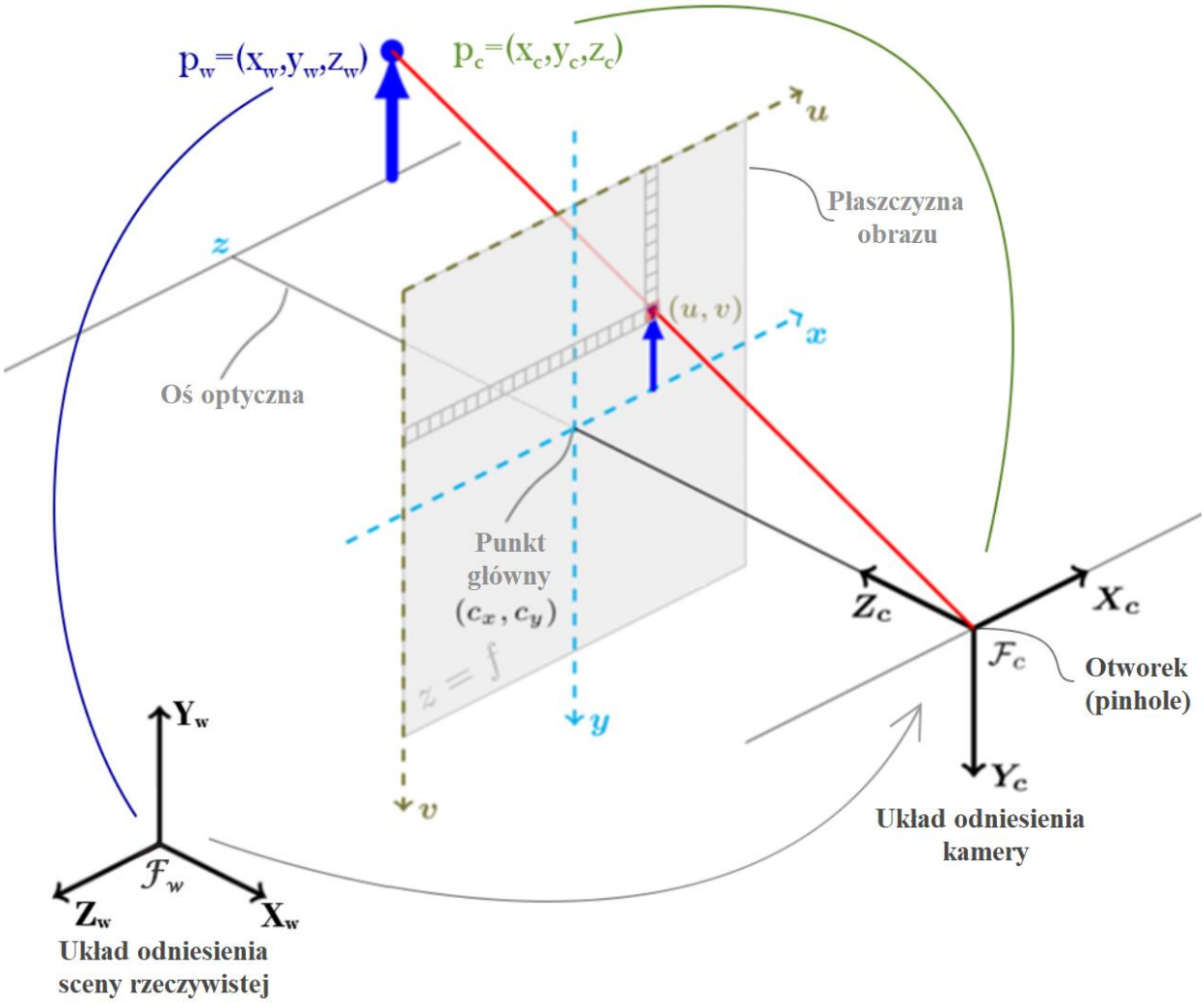
Geneza dziedziny CV [4] stanowi złożony proces ewolucji, w którego centrum znajdują się interdyscyplinarne połączenia między naukami matematycznymi, informatycznymi oraz naukami neurobiologicznymi, odnoszącymi się do definicji zmysłu wzroku. Początki tej dziedziny sięgają lat 60. ubiegłego wieku, kiedy to pojawiły się próby automatycznego analizowania i interpretowania obrazów cyfrowych przy użyciu komputerów. Jednym z kamieni milowych w tym procesie było opracowanie pierwszej technologii komputerowego skanera obrazu, umożliwiającego digitalizację zdjęć. W kolejnych latach rozwój CV znacząco przyspieszył dzięki postępowi w obszarze uczenia maszynowego, które obecnie stanowi integralną część szeregu aplikacji, począwszy od wizyjnych systemów identyfikacji biometrycznej po inteligentne systemy stosowane w pojazdach autonomicznych.

Z uwagi na przyjęte cele badawcze oraz przeznaczenie projektowanego systemu (zob. rozdział 1), autor rozprawy postanawia ograniczyć rozpoznanie literaturowe tylko do obszarów zastosowań związanych z przetwarzaniem danych wizyjnych, rejestrowanych przy użyciu stacjonarnych, konwencjonalnych kamer monitoringowych 2D.

2.1 Wstępne przetwarzanie obrazów

Pozyskiwanie danych wizyjnych przy użyciu cyfrowych rejestratorów obrazów 2D odgrywa kluczową rolę w dziedzinie CV. Proces pozyskiwania cyfrowego obrazu 2D na podstawie obserwowanej sceny rzeczywistej 3D często opisywany jest przy użyciu modelu perspektywicznego kamery, znanego również modelem kamery otworkowej, którego graficzną reprezentację przedstawiono na Rys. 2.1. Model otworkowy kamery [5,6] opiera się na zasadzie rzutowania perspektywicznego, gdzie wszystkie linie projekcji (wirtualne linie łączące punkty sceny z punktem obserwacji na obrazie) przecinają się w jednym punkcie zwanym „*otworkiem kamery*”. W modelu tym, obiekty znajdujące się dalej od kamery są reprezentowane jako mniejsze, co odzwierciedla zjawisko perspektywy, występujące w rzeczywistym świecie. Punkty sceny 3D są transformowane do 2D za pomocą macierzy projekcji, która uwzględnia takie parametry jak: ogniskowa obiektywu, orientacja przestrzenna kamery, kąt widzenia oraz proporcje obrazu.

Dane pozyskane z kamery cyfrowej najczęściej są dostarczane w postaci zapisanych wcześniej plików wideo lub streamingu wideo w czasie rzeczywistym. Takie formy danych



Rys. 2.1 Klasyczny model kamery otworkowej [5]

wymagają przeprowadzenia procesu wstępnego przetwarzania obrazów [7]. Proces ten odnosi się przede wszystkim do dekompozycji wideo na obrazy poklatkowe. Ponadto obejmuje szereg operacji mających na celu poprawę interpretowalności obrazu przy zastosowaniu narzędzi: filtracji, usuwania szumów i zakłóceń, normalizacji, transformacji oraz selekcji ROI. Analiza technik wstępnego przetwarzania obrazów pozwala wskazać kilka popularnych bibliotek programistycznych, w których narzędzia te zostały zaimplementowane. Można tu wyróżnić m.in.: Intel Integrated Performance Primitives [8], Vision-something-Libraries [9], Numerical Algorithms Group's Library of Technical Intelligence [10], VLFeat [11] czy CImg [12]. Obecnie, do najbardziej popularnych bibliotek programistycznych zalicza się OpenCV, Scikit-image i MATLAB IPT.

Biblioteka OpenCV [13] została zaimplementowana w kilku językach programowania, w tym C++, Python oraz Java, co sprawia, że jest uniwersalna i dostępna dla wielu użytkowników. Posiada wsparcie dla technologii CUDA i OpenCL umożliwiając użycie akceleracji GPU i efektywne przyspieszenie przetwarzania obrazów. Zaimplementowany moduł DNN zawiera narzędzia do realizacji głębokiego uczenia, co czyni bibliotekę atrakcyjną dla projektów z obszaru DL. OpenCV oferuje szereg funkcji, począwszy od podstawowych narzędzi edycji obrazu (np.

skalowanie, rotacja, operacje morfologiczne) po zaawansowane techniki analityczne (np. detekcja krawędzi, śledzenie obiektów). Biblioteka cieszy się dużą społecznością, co przekłada się na obszerną dokumentację oraz aktywne fora użytkowników, ułatwiające rozwiązywanie problemów i wymianę doświadczeń. Biblioteka OpenCV jest rozpowszechniana na licencji BSD, co sprawia, że można ją swobodnie używać w projektach komercyjnych i niekomercyjnych.

Biblioteka Scikit-image [14] jest zaimplementowana w języku Python, co sprawia, że jest łatwa w integracji z ekosystemem tego języka. Nie posiada ona wbudowanego wsparcia dla akceleracji GPU, co może być ograniczeniem w przypadku projektów wymagających dużych zasobów obliczeniowych. Scikit-image skupia się głównie na tradycyjnych algorytmach przetwarzania obrazów, przez co jej użyteczność w obszarze DL jest ograniczona. Posiada dobrą dokumentację, choć społeczność użytkowników może być mniejsza niż w przypadku OpenCV. Biblioteka Scikit-image jest rozpowszechniana na licencji BSD umożliwiając swobodne korzystanie i modyfikowanie opracowanego algorytmu w projektach komercyjnych i niekomercyjnych.

Moduł MATLAB IPT [15] został zaimplementowany w języku M, ponadto zawiera narzędzia umożliwiające tłumaczenie kodu do języków C/C++. MATLAB IPT oferuje ograniczone wsparcie dla GPU, co może wpływać na wydajność w projektach wymagających intensywnego przetwarzania obrazów. Posiada narzędzia do pracy z DL, choć to nie jest główny obszar funkcjonalności modułu IPT. Zapewnia korzystanie z zaawansowanych funkcji przetwarzania obrazu i dobrą dokumentację, natomiast wsparcie i aktualizacje zależą od firmy MathWorks. Społeczność użytkowników skupiona wokół IPT jest mniejsza w porównaniu do bibliotek otwartych. Jest to oprogramowanie komercyjne.

Każda z tych bibliotek posiada unikalne cechy, które wpływają na wydajność, skalowalność i elastyczność procesów przetwarzania. W Tab. 2.1 zamieszczono opracowane przez autora porównanie bibliotek OpenCV, Scikit-image oraz MATLAB IPT ze względu na ich przydatność w projektowanym systemie IVSS.

Tab. 2.1 Porównanie wybranych bibliotek wstępnego przetwarzania danych wizyjnych

Biblioteka	**Język implementacji**	**Akceleracja GPU**	**Wsparcie DL**	**Zakres dostępnych funkcjonalności (skala 1–5)**	**Dokumentacje i wsparcie społeczności (skala 1–5)**	**Licencja**
OpenCV	C++, Python, Java	Tak	Tak	5	5	Otwarta BSD
Scikit-image	Python	Nie	Nie	3	4	Otwarta BSD
MATLAB IPT	M	Tak	Tak	4	4	Komercyjna

Porównanie przeprowadzono z użyciem sześciu kryteriów. Pierwsze dotyczy możliwości implementacji bibliotek w wybranym języku programowania, jakim jest C++. Spośród wybranych

bibliotek tylko OpenCV spełnia to kryterium. Drugie kryterium dotyczy zdolności bibliotek do korzystania z akceleracji sprzętowej z użyciem GPU, natomiast trzecie kryterium dotyczy implementacji funkcji, umożliwiających realizację procesów związanych z DL. Kryteria 2 oraz 3 spełniają biblioteki OpenCV oraz MATLAB IPT. Czwarte oraz piąte kryterium dotyczą zakresu dostępnych funkcjonalności oraz dostępności dokumentacji technicznych i możliwości uzyskania wsparcia wśród społeczności użytkowników. Kryteria 4 oraz 5 zostały określone na podstawie subiektywnej oceny autora, opartej na jego doświadczeniach związanych z użyciem analizowanych bibliotek i wskazują na przewagę OpenCV. Ostatnim kryterium porównawczym jest typ licencji, na której warunkach biblioteki są udostępniane. W tym przypadku zarówno OpenCV jak i Scikit-image udostępniane są na warunkach licencji BSD, co stanowi o ich kluczowym atucie ekonomicznym.

Z przedstawionej analizy wynika, iż OpenCV należy do najlepszych bibliotek wstępnego przetwarzania obrazów w przedmiotowym systemie IVSS. Oferuje wszechstronność, wsparcie społeczności użytkowników, oraz rozbudowany zestaw funkcji, w tym wsparcie dla DL i akceleracji obliczeń realizowanych przy użyciu GPU. OpenCV umożliwia implementację kodu w języku C++, co korzystnie wpływa na wydajność opracowanej aplikacji. Licencja BSD dodatkowo umożliwia elastyczne zastosowanie biblioteki w projektach komercyjnych i niekomercyjnych.

2.2 Kamery monitoringowe

Obecnie, w systemach IVSS używa się różnego rodzaju kamer, co umożliwia przeprowadzenie wielostopniowej kategoryzacji [16]. Jednym z kryteriów podziału jest czułość na światło, które definiuje możliwość pracy kamery w trybie dziennym lub dziennym i nocnym (praca w sztucznym świetle podczerwieni). Innym kryterium jest odporność kamery na warunki atmosferyczne definiujące przeznaczenie kamery do pracy wewnątrz lub na zewnątrz budynku. Kolejnym kryterium jest odporność kamery na akty wandalizmu, definiujące urządzenia o zwiększonej odporności na uszkodzenia mechaniczne. Kryterium typu transmisji danych wizyjnych dzieli kamery na przewodowe oraz bezprzewodowe (kamery IP).

Na Rys. 2.2 przedstawiono przykłady kamer najczęściej stosowanych w systemach IVSS [16]. Kamery statyczne oraz tubowe umożliwiają obserwację stałego obszaru sceny. Kamery PTZ zapewniają możliwość fizycznego obracania się (Pan), pochylania (Tilt) oraz zoomowania (Zoom) obiektywu, pozwalając na dynamiczną zmianę kąta widzenia i zbliżenia na interesujące obiekty. Innym przykładem są kamery EPTZ umożliwiające symulowane (programowe) „*przesuwanie*” i „*zbliżanie*” obrazu, bazując na danych wizyjnych o wysokiej rozdzielczości, bez użycia fizycznych mechanizmów kontroli pozycjonowania kadru. Kamery kopułowe i wieżyczkowe są podobne konstrukcyjnie oraz funkcjonalnie. Umożliwiają zdalną regulację kątów widzenia oraz montaż zarówno w pionie (na ścianie) jak i poziomie (na suficie). Główną różnicą jest

Rys. 2.2 Typy kamer stosowane w systemach IVSS: a) statyczna, b) PTZ (zewnętrzna), c) PTZ (wewnętrzna), d) tubowa, e) kopułowa, f) wieżyczkowa [16]

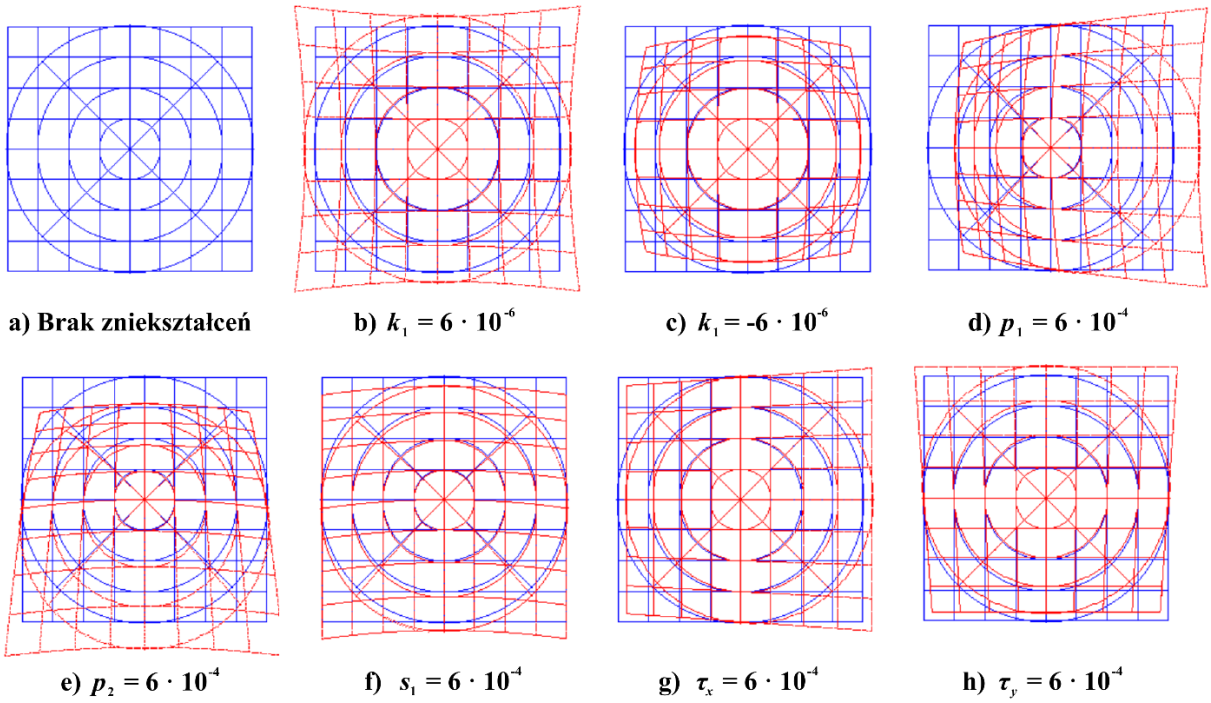
zastosowanie kopuły, utrudniającej rozpoznanie aktualnego ustawienia obiektywu, uniemożliwiając potencjalnym przestępcom oszacowanie tzw. martwych stref monitoringu. Niestety, konstrukcja kopułowa może powodować odbijanie się światła podczerwonego emitowanego w warunkach nocnych, zbytnio naświetlając obiektyw. Ten problem nie występuje w kamerach wieżyczkowych, gdyż ich konstrukcja jest pozbawiona kopuły.

W celu realizacji testów systemu, autor postanowił użyć kamerę wieżyczkową TP-Link Tapo C500 [17]. Wybór tej kamery uzasadnia kompromis ceny do jakości pozyskiwanego sygnału wizyjnego zarówno wewnątrz budynku, jak i na zewnątrz, a także w trudnych warunkach oświetleniowych (automatyczny tryb doświetlania IR). Kamera umożliwia zapis oraz transmisję sygnału wideo w rozdzielczości FHD przy maksymalnej częstotliwości klatkowania 15 *FPS* w popularnym formacie MP4. Ponadto, kamera umożliwia komunikację bezprzewodową Wi-Fi, oraz zdalnie sterowany mechanizm obrotu przetwornika, umożliwiający precyzyjne kadrowanie obrazu.

Realizacja testów w różnych środowiskach docelowych i przy różnych konfiguracjach montażu kamery stanowi poważny problem, gdyż wymaga odpowiedniego przygotowania logistycznego i przeprowadzenia prac monterskich, często o charakterze inwazyjnym. Autor rozprawy postanowił rozwiązać ten problem używając mobilnego statywu studyjnego z amortyzacją powietrzną W809 [18]. Teleskopowy mechanizm rozkładania statywu umożliwia płynną regulację wysokości montażu kamery w zakresie 1.18–4.40 m, ponadto lekka i zwarta konstrukcja zapewnia łatwe umiejscowienie i zorientowanie kamery w dowolnej aranżacji sceny.

2.3 Kalibracja kamer

Pomimo zastosowania zaawansowanych· technologicznie procesów wytwarzania urządzeń z zakresu optyki cyfrowej, niezwykle trudne jest całkowite wyeliminowanie błędów i niedokładności produkcyjnych oraz montażowych. Powstałe w konsekwencji zniekształcenia wpływają niekorzystnie na pozyskiwane dane wizyjne. W konwencjonalnych kamerach 2D występują trzy główne rodzaje zniekształceń [5]: radialne, styczne, pryzmatyczne i perspektywiczne. Zniekształcenia radialne [5] są jednym z najczęściej występujących rodzajów zniekształceń kamer. Powodują zakrzywienie obrazu, szczególnie w pobliżu brzegów obrazu wskutek nieliniowego kształtu soczewek kamery. Zniekształcenia styczne [5] wynikają z niedoskonałości montażu soczewek kamery, które powodują, że osie optyczne soczewek nie są idealnie prostopadłe do płaszczyzny obrazu. Zniekształcenia pryzmatyczne [5] powstają, gdy soczewki kamery nie ogniskują różnych długości fal światła w tym samym punkcie, co powoduje wystąpienie efektu rozszczepienia kolorów. Nieco innym przykładem zniekształceń geometrycznych (nie związanych z konstrukcją kamery) są zniekształcenia perspektywiczne [5] wynikające z wzajemnej orientacji kamery i obserwowanej sceny lub obiektu. Zniekształcenia te mogą powodować nierzeczywiste odwzorowanie przedmiotów oddalonych od kamery zarówno pod względem kształtu, jak i rozmiaru. Zniekształcenia perspektywiczne mogą również wprowadzać niejednorodność rozkładu głębi ostrości w przypadku gdy obserwowana płaszczyzna obiektu lub sceny jest nierównoległa do płaszczyzny obrazowania, co szczegółowo opisuje reguła Scheimpfluga [5]. Na Rys. 2.3 przedstawiono graficzną reprezentację różnych rodzajów zniekształceń kamery.



Rys. 2.3 Przykłady zniekształceń kamery: a) wzorzec idealny, b)–c) zniekształcenia radialne, d)–e) zniekształcenia styczne, f) zniekształcenia pryzmatyczne, g)–h) zniekształcenia perspektywiczne [5]

Kolorem niebieskim oznaczono wzorce idealne (bez zniekształceń), natomiast kolorem czerwonym efekty wpływu zniekształceń na geometrię wzorca idealnego. Do opisu parametrów zniekształceń stosuje się następujące współczynniki [5]: dla zniekształceń radialnych: k_1, k_2 oraz k_3, dla zniekształceń stycznych: p_1 oraz p_2, dla zniekształceń pryzmatycznych: s_1, s_2, s_3 oraz s_4 i dla zniekształceń perspektywicznych: τ_x oraz τ_y.

Kalibracja kamery [5] ma na celu zidentyfikowanie i skorygowanie zniekształceń przy użyciu dedykowanych metod analitycznych. Proces kalibracji wymaga zestawu obrazów wzorcowych (np. szachownicy), na podstawie których optymalizowane są parametry modelu kalibracyjnego, minimalizując różnicę między obserwowanymi a przewidywanymi położeniami punktów na obrazie. W celu przeprowadzenia procesu kalibracji kamery w projektowanym systemie IVSS, autor postanowił użyć narzędzi analitycznych zaimplementowanych w bibliotece OpenCV. Proces korekty zostanie zrealizowany z użyciem dedykowanej funkcji *cv::CalibrateCamera()*.

2.4 Detekcja i klasyfikacja obiektów

Detekcja i klasyfikacja obiektów pełnią kluczową rolę w dziedzinie CV. Detekcja obiektów [19] to proces umożliwiający określenie obecności i lokalizacji obiektu na analizowanym obrazie w odniesieniu do współrzędnych macierzy pikseli. Klasyfikacja obiektów [19] jest procesem nadawania tzw. etykiet, które stanowią o przynależności obiektu do konkretnych kategorii lub klas. Na Rys.2.4 przedstawiono różne przykłady detekcji i klasyfikacji obiektów na obrazie.

Rys. 2.4 Przykłady detekcji i klasyfikacji obiektów na obrazie [19]

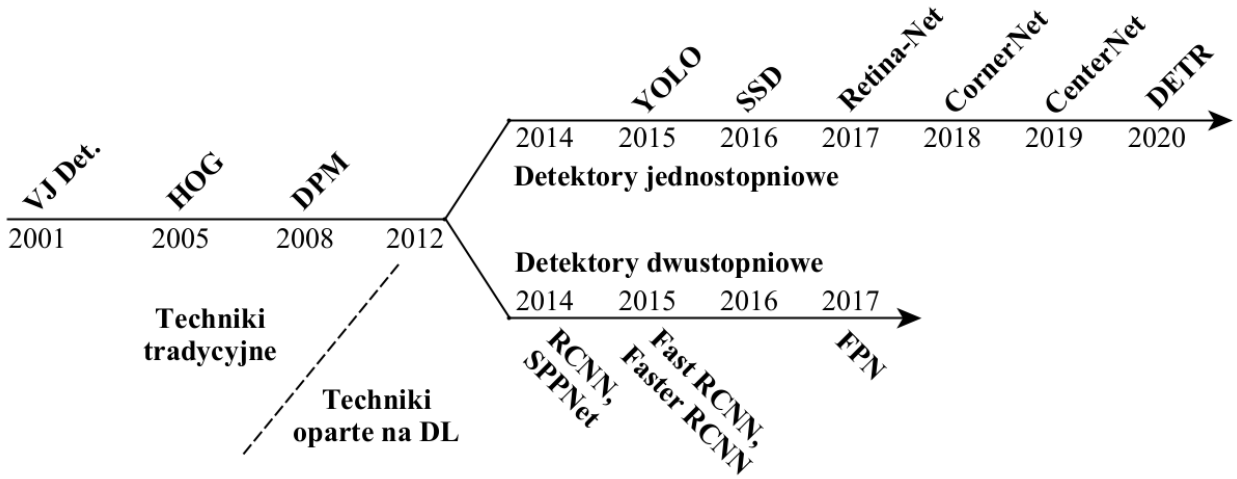
W praktyce, detekcja i klasyfikacja są często łączone w jedną operację. Przy takim podejściu, najpierw identyfikuje się obiekty na obrazie, a następnie przypisuje się je do odpowiednich kategorii lub klas.

2.4.1 Adnotacja obiektów

W celu graficznego oznaczenia obiektu, na obrazie używane są różne typy adnotacji [19]. Ramka BB jest jednym z najczęściej używanych typów adnotacji. Graficzna reprezentacja BB to czworokątna, ortogonalna ramka, wewnątrz której zawiera się cały oznaczany obiekt. Segmentacja semantyczna to adnotacja pikselowa, w której każdy piksel składający się na obiekt tworzy obszar oznaczenia. Linie i splajny to kolejne typy adnotacji, które umożliwiają precyzyjne oznaczanie struktur ciągłych, takich jak kontury, krawędzie czy granice obiektów.

2.4.2 Detektory obiektów w ujęciu historycznym

Powszechnie przyjmuje się, że rozwój technik detekcji i klasyfikacji obiektów dzieli się na dwa okresy historyczne [20]: rozwój technik tradycyjnych (do 2012 r.) i rozwój technik opartych na DL (od 2012 r.). Rozwój technik detekcji i klasyfikacji obiektów w ujęciu historycznym przedstawiono na Rys.2.5.



Rys. 2.5 Rozwój technik detekcji i klasyfikacji obiektów w ujęciu historycznym [20]

W zakresie tradycyjnych technik detekcji i klasyfikacji obiektów stosowano głównie metody analityczne. Ze względu na brak skutecznych technik reprezentacji obrazu w tamtym czasie, autorzy projektowali wyrafinowane reprezentacje cech, łącząc je z różnorodnymi technikami optymalizacji. Tradycyjne techniki detekcji i klasyfikacji obiektów posiadały pewne ograniczenia, takie jak duża liczba generowanych propozycji, z których wiele było zbędnych, a które często dawały wiele wyników fałszywie pozytywnych.

Rozwój technik opartych na głębokim uczeniu maszynowym zaowocował opracowaniem konwolucyjnych sztucznych sieci neuronowych (CNN). Architektura CNN umożliwiła realizację ekstrakcji cech obrazu na poziomie nieosiągalnym przy użyciu technik tradycyjnych, zwiększając jednocześnie szybkość i skuteczność detekcji obiektów. W rezultacie, od 2012 r. metody wykrywania obiektów zaczęły ewoluować z niespotykaną dotąd szybkością. W erze metod DL wyróżnia się dwie podgrupy detektorów [20]: dwustopniowe i jednostopniowe.

2.4.3 Detektory tradycyjne oparte na technikach analitycznych

W klasie detektorów tradycyjnych opartych na technikach analitycznych, można wyróżnić kilka głównych rozwiązań. Jednym z nich jest detektor VJ [21,22], którego działanie polega na kaskadowym skanowaniu obrazu przy użyciu techniki „*przesuwne okna*” (ang. *sliding windows*). Kaskada przesuwnych klasyfikatorów („*okien*”) bazuje na cechach Haara, reprezentujących zmiany intensywności pikseli. Ponadto, zastosowanie techniki kaskadowego odrzucania umożliwia szybką eliminację obszarów niezawierających obiektu.

Innym przykładem jest detektor HOG [23]. W pierwszym etapie detektor HOG realizuje normalizację obrazu wejściowego pod kątem redukcji wpływu zmian oświetlenia. Następnie, na podstawie zmiany intensywności lub koloru pikseli obliczane są gradienty obrazu, które dostarczają informacji o kierunku i sile potencjalnych krawędzi obiektu. W kolejnym kroku realizowane jest głosowanie na histogramy orientacji gradientów. Głosy są ważone wielkością gradientu, co wzmacnia wpływ pikseli z wyraźnymi krawędziami. Histogramy, opracowane w procesie głosowania, są normalizowane i kompilowane do postaci deskryptorów. Deskryptory są następnie używane jako wektory cech w procesie uczenia klasyfikatorów obiektów. W detektorze HOG często stosowanymi klasyfikatorami są maszyny wektorów nośnych (SVM).

Detektor DPM [24] bazuje na piramidzie cech HOG. Na wyższych poziomach piramidy reprezentowane są bardziej ogólne cechy, podczas gdy na niższych reprezentowane są cechy o wyższej rozdzielczości. Detektor DPM składa się z filtru głównego (ang. *root filter*) obejmującego cały obiekt oraz dodatkowych filtrów części, które reprezentują mniejsze części obiektu. Zastosowany model deformacji pozwala na elastyczne dopasowanie filtru głównego i filtrów części, uwzględniając możliwe odkształcenia obiektów. Detektor DPM stosuje podejście kaskadowego skanowania obrazu przy użyciu techniki „*przesuwne okna*”, gdzie każde okno analizuje obraz pod kątem obecności obiektu. Wynik detekcji dla każdego okna jest sumą wyniku filtru głównego i maksymalnych wyników filtrów części, pomniejszonych o koszty deformacji. Trening detektora DPM realizowany jest przy użyciu zmodyfikowanego modelu maszyny wektorów nośnych (SVM), w którym lokalizacje części obiektów są traktowane jako ukryte zmienne. W procesie treningowym dopasowywanie możliwych lokalizacji części realizowane jest w sposób iteracyjny.

2.4.4 Detektory dwustopniowe oparte na DL

Detekcja i klasyfikacja obiektów na obrazie przy użyciu dwustopniowych detektorów opartych na DL, realizowana jest dwuetapowo [20]. Pierwszy etap odpowiada za generowanie propozycji obszarów obrazu zawierających potencjalne obiekty. Drugi polega na dokładnej lokalizacji oraz klasyfikacji obiektów w wygenerowanych obszarach. Detektory te z reguły zapewniają osiągnięcie lepszej dokładności, ale ze uwagi na konieczność realizacji dwóch etapów,

cechują się mniejszą prędkością wnioskowania w porównaniu z detektorami jednostopniowymi. W klasie detektorów dwustopniowych wyróżnia się kilka głównych rozwiązań.

RCNN [25] to pionierski, dwustopniowy detektor obiektów. W pierwszym etapie detektor RCNN analizuje obraz w poszukiwaniu propozycji obszarów mogących zawierać obiekty. W tym celu korzysta z zewnętrznych metod generowania propozycji obszarów zainteresowania (np. metody Selective Search). W drugim etapie wyodrębnione obszary stanowią wejście sieci konwolucyjnej, realizującej ekstrakcję cech oraz klasyfikację obiektu. Pomimo osiąganej dokładności, RCNN charakteryzuje się niską wydajnością wynikającą z konieczności wielokrotnego przetwarzania każdego obszaru zainteresowania przez sieć konwolucyjną. Problem ten rozwiązano w detektorze Fast RCNN [26]. Zaimplementowanie warstwy ROI Pooling umożliwiło wydajne przetwarzanie wielu obszarów zainteresowania w jednym przebiegu sieci konwolucyjnej. Detektor Faster RCNN [27] wprowadza kolejne usprawnienie, które eliminuje konieczność stosowania zewnętrznych metod generowania propozycji obszarów zainteresowania. Cel ten osiągnięto poprzez integrację sieci RPN bezpośrednio w strukturze modelu RCNN.

Innym przykładem dwustopniowego detektora obiektów jest SPPNET [28]. Główną innowacją tego detektora jest implementacja warstwy SPP w tradycyjnej architekturze CNN. Warstwa SPP umieszczona jest bezpośrednio po ostatniej warstwie konwolucyjnej umożliwiając generowanie wektorów cech o stałej długości. Ta modyfikacja pozwala osiągnąć wysoką dokładność detekcji obiektów niezależnie od rozmiaru i skali analizowanego obrazu.

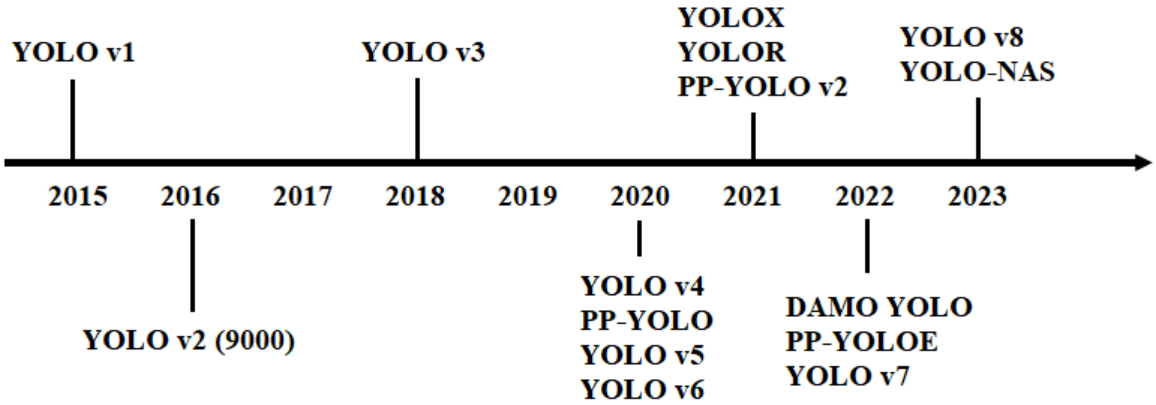
Detektor FPN [29] wprowadza piramidalną hierarchię warstw cech, która generowana jest w dwuetapowym procesie. Etap pierwszy, tzw. „*Bottom-Up Pathway*”, to standardowa ścieżka obliczeniowa sieci konwolucyjnej, która generuje piramidalną hierarchię warstw. Każda kolejna warstwa piramidy wyznacza mapy cech o mniejszej rozdzielczości, ale z wyższym poziomem reprezentacji semantycznej. Drugi etap tzw. „*Top-Down Pathway & Lateral Connections*”, to proces przekazywania informacji zarówno w górę, jak i w dół w hierarchii cech, realizowany przy pomocy połączeń bocznych. Dzięki temu, cechy z wyższych poziomów piramidy (o silnej reprezentacji semantycznej, lecz niskiej rozdzielczości) są wzbogacane mapami cech z dolnych warstw piramidy (o wyższej rozdzielczości, lecz słabszej reprezentacji semantycznej). Umożliwia to osiągnięcie wysokiej dokładności detekcji obiektów, niezależnie od ich rozmiaru na obrazie.

2.4.5 Detektory jednostopniowe oparte na DL

Detektory jednostopniowe oparte na DL [20] umożliwiają realizację zadań detekcji i klasyfikacji obiektów na obrazie w procesie jednoetapowym. Detektory te są najczęściej stosowane w aplikacjach wymagających przetwarzania w czasie rzeczywistym, ponieważ dzięki uproszczonej architekturze zapewniają pożądany wynik znacznie szybciej niż detektory

dwustopniowe. Wśród klasy detektorów jednostopniowych można wyróżnić kilka głównych rozwiązań.

YOLO jest detektorem jednostopniowym, służącym do lokalizacji i klasyfikacji obiektów wielu kategorii w jednym przejściu danych przez sieć neuronową. Aktualnie istnieje osiem oficjalnych wersji YOLO [30–37] oraz kilka wersji bazujących na oryginałach takich jak PP-YOLO, YOLOR czy YOLO-NAS [38]. Rozwój detektorów YOLO w ujęciu historycznym przedstawiono na Rys.2.6.



Rys. 2.6 Rozwój detektorów YOLO w ujęciu historycznym [38]

Ogólna zasada działania detektora YOLO sprowadza się do podziału obrazu na siatkę komórek. W każdej komórce wyznaczane jest prawdopodobieństwo wystąpienia obiektów, określenie ich przynależności do określonej kategorii oraz adnotacje (współrzędne BB). W każdej wersji YOLO, autorzy wprowadzali kolejne usprawnienia, mające na celu przede wszystkim optymalizację szybkości przetwarzania danych, jak również zwiększenie dokładności rozpoznawania obiektów o różnych skalach [38–40]. Modele pierwszych czterech wersji oparte są na architekturze DARKNET [41] zaimplementowanej w języku C++, natomiast kolejne bazują na tłumaczeniu frameworka DARKNET do frameworka PyTorch celem implementacji w języku Python. Wersja YOLO v4 [33] stanowi istotne odniesienie w niniejszej rozprawie, ponieważ w chwili rozpoczęcia prac przez autora, była to najnowsza wersja tej biblioteki, zaimplementowana w języku programowania C++. W czwartej wersji detektora YOLO wprowadzono szereg innowacji, w tym zastosowanie technik CSP, MDA oraz SAT. CSP to technika projektowania sieci, która ma na celu zmniejszenie obciążenia obliczeniowego modelu poprzez rozdzielenie warstw sieci na dwie części. Każda część przetwarza dane oddzielnie, a ich wyniki są łączone na końcu. MDA to technika augmentacji danych, polegająca na łączeniu czterech różnych obrazów treningowych w jedną mozaikę. Dzięki temu, trening modelu może zostać zrealizowany dla wielu obiektów występujących w różnych kontekstach i skalach. SAT to kolejna technika augmentacji danych, polegająca na tworzeniu nowych przykładów uczących poprzez wprowadzanie pewnych modyfikacji w oryginalnych obrazach treningowych. Ma to na celu osiągnięcie wysokiej

dokładności predykcji modelu, testowanego na danych, które nie były używane podczas procesu uczenia.

SSD [42] to kolejny detektor jednostopniowy, opracowany do detekcji wielu kategorii obiektów. Architekturę SSD można podzielić na dwie główne części: model bazowy i model warstw konwolucyjnych. Pierwszą część stanowi wstępnie wytrenowana sieć klasyfikacyjna, która działa jako ekstraktor map cech. Sieć ta pozbawiona jest ostatniego fragmentu odpowiadającego za klasyfikację i w to miejsce zaimplementowano drugą część – warstwy cech modelu konwolucyjnego. Warstwy te umożliwiają detekcję obiektów o różnych skalach. Ponadto, w architekturze SSD zastosowano model tłumienia niejednoznacznych detekcji NMS, który eliminuje redundantne wykrycia obiektów, pozostawiając jedynie najbardziej prawdopodobne wyniki. W celu oznaczania wykrytych obiektów na obrazie używana jest adnotacja typu BB.

W detektorze Retina-Net [43] autorzy wprowadzają nowy typ funkcji strat o nazwie „*focal loss*”, której głównym celem jest dążenie do rozwiązania problemu nierównowagi ilościowej przykładów uczących pomiędzy różnymi klasami obiektów. Zastosowanie funkcji „*focal loss*” umożliwia zmniejszenie wartości wag dla klas stanowiących znaczną większość zbioru danych i zwiększenie dla klas stanowiących mniejszość. Architektura detektora Retina-Net składa się z czterech głównych części: sieci ResNet, sieci FPN, podsieci klasyfikacyjnej i podsieci regresyjnej. Sieć ResNet [44] stanowi wejście architektury Retina-Net i służy do obliczania map cech obrazów o różnych rozmiarach. Mapy cech trafiają następnie do sieci FPN [29], która generuje wieloskalową, konwolucyjną piramidę cech. Podsieć klasyfikacyjna określa prawdopodobieństwo przynależności wykrytych AB do możliwych kategorii obiektów. Podsieć regresyjna realizuje obliczenia związane z przekształceniem wynikowych AB do adnotacji typu BB dla każdego wykrytego obiektu na obrazie.

Detektor CornerNet [45] wprowadza nowe jednostopniowe podejście do detekcji obiektów, które, w przeciwieństwie do poprzednich rozwiązań tego typu, eliminuje konieczność używania AB. Detektor CornerNet analizuje obraz przy użyciu pojedynczej sieci konwolucyjnej, generującej mapę ciepła (ang. *heatmaps*). Wartości szczytowe mapy ciepła reprezentują predykcje lokalizacji punktów charakterystycznych – lewych górnych i prawych dolnych narożników obiektów tej samej klasy. Współrzędne narożników są następnie grupowane, celem przypisania ich do pojedynczego obiektu i określenia wynikowej adnotacji BB. W architekturze CornerNet autorzy wprowadzają nowy rodzaj warstwy poolingowej typu „*corner pooling*”, która wspomaga działanie sieci konwolucyjnej w celu dokładnego lokalizowania narożników BB.

Detektor CenterNet [46], podobnie jak CornerNet, odchodzi od koncepcji stosowania tradycyjnych metod detekcji opartych na wyznaczaniu AB. Koncepcja CenterNet odnosi się do reprezentacji obiektów przy użyciu punktu środka ciężkości BB. Detekcja obiektów sprowadza się więc do standardowego problemu estymacji punktów charakterystycznych. Obraz wejściowy

przesyłany jest do konwolucyjnej sieci neuronowej, generującej mapę ciepła. Wartości szczytowe generowanej mapy ciepła reprezentują predykcje lokalizacji punktów, które stanowią środki ciężkości obiektów. Cechy obrazu zorientowane wokół wartości szczytowych mapy, umożliwiają predykcję wysokości i szerokość wynikowych adnotacji BB.

DETR [47] to kolejny detektor jednostopniowy, w którym proces przetwarzania danych obejmuje kilka etapów. W pierwszej kolejności realizowany jest proces przetworzenia obrazu wejściowego za pomocą CNN, tworząc mapę cech o niższej rozdzielczości. Następnie mapa cech jest przekazywana do modelu transformatora, który składa się z enkodera i dekodera. Enkoder analizuje i koduje cechy obrazu, tworząc kontekst uwzględniający zależności między cechami. Dekoder z kolei przetwarza zakodowany kontekst cech w celu predykcji obiektów na obrazie. Ostateczna detekcja obiektów realizowana jest za pomocą prostego modelu sieci FFN, która przetwarza wyniki z dekodera, generuje etykiety klasyfikujące oraz wyznacza adnotacje BB obiektów.

2.4.6 Bazy danych uczących

Baza danych uczących [19,20] jest kluczowym elementem procesu uczenia maszynowego. Zawiera reprezentatywne próbki przykładów umożliwiające trenowanie algorytmów ML, przeznaczonych do rozpoznawania wzorców, podejmowania decyzji oraz predykcji bez konieczności jawnego programowania. W klasycznym procesie uczenia maszynowego baza danych jest zwykle dzielona na trzy główne zbiory: uczący (treningowy), walidacyjny oraz testowy. Zbiór uczący umożliwia dostosowywanie parametrów modelu w celu osiągnięcia jak najlepszej wydajności w procesie uczenia. Zbiór walidacyjny służy do dostrajania hiperparametrów modelu oraz oceny, jak dobrze działa na danych, które nie były używane podczas procesu trenowania. Zbiór testowy jest używany do oceny modelu po zakończeniu procesu trenowania i dostrajania. Aby zapewnić obiektywną ocenę modelu, zbiór testowy powinien być całkowicie niezależny od danych treningowych i walidacyjnych.

Efektywność wytrenowanego modelu jest ściśle związana z jakością, dokładnością, objętością i zróżnicowaniem próbek składających się na bazę danych. Tworzenie własnej bazy danych jest procesem wymagającym dużego nakładu pracy, szczególnie jeśli baza ma zostać użyta w uczeniu nadzorowanym. W tym rodzaju uczenia maszynowego konieczne jest dostarczenie stosownych adnotacji dla każdej próbki, co sprawia, że opracowywanie własnych baz danych jest zasadne głównie w przypadkach specyficznych zastosowań. Alternatywę mogą stanowić gotowe, powszechnie dostępne zbiory bazy danych, dedykowane do celów detekcji i klasyfikacji obiektów na obrazach. Do popularnych zbiorów tego typu należą: PASCAL VOC, ImageNet, MS-COCO, Open Images, YouTube-8M, RoboFlow oraz Kaggle.

PASCAL VOC [19] w latach 2005–2012 był popularnym zbiorem danych w dziedzinie CV. Co roku tworzone były różne wersje zbiorów, których obrazy pochodziły z istniejących ówcześnie baz danych oraz z serwisu Flickr. PASCAL VOC, jako ogólnodostępny zbiór danych, był używany do organizowania tzw. „*wyzwań*”, które umożliwiały rozwijanie oraz dokładne porównywanie efektywności modeli dedykowanych do detekcji obiektów, klasyfikacji i segmentacji obrazów.

ImageNet [19,48] to jedna z najpopularniejszych baz danych uczących z dziedziny CV. ILSVRC to konkurs z dziedziny przetwarzania obrazów, który odbywał się co roku w latach 2010–2017. ILSVRC obejmował publicznie dostępne zbiory danych ImageNet przeznaczone do realizacji „*wyzwań*” związanych z oceną efektywności algorytmów detekcji, klasyfikacji i segmentacji. W obecnej formie baza danych ImageNet składa się z ponad 14 mln. obrazów i ponad 20 tys. kategorii obiektów.

MS-COCO [19,49] to kolejna popularna baza danych, obecnie składająca się z 328 tys. obrazów, na których oznaczono 2.5 mln. obiektów. Obiekty te zawierają się w zbiorze 91 kategorii, z czego około 80 z nich ma więcej niż 5 tys. oznaczeń na obrazach. MS-COCO zawiera różne sceny z codziennego życia, wraz z dokładnymi adnotacjami odnoszącymi się do detekcji, segmentacji i klasyfikacji obiektów. Konkursy oparte na zbiorze danych MS-COCO odbywają się co roku od 2015 roku. W przeciwieństwie do powyższych baz danych, MS-COCO zawiera mniejszą liczbę klas obiektów, ale większe ich zagęszczenie na obrazach. Dzięki temu modele trenowane przy użyciu MS-COCO wykazują lepszą zdolność detekcji obiektów najczęściej spotykanych w środowisku rzeczywistym.

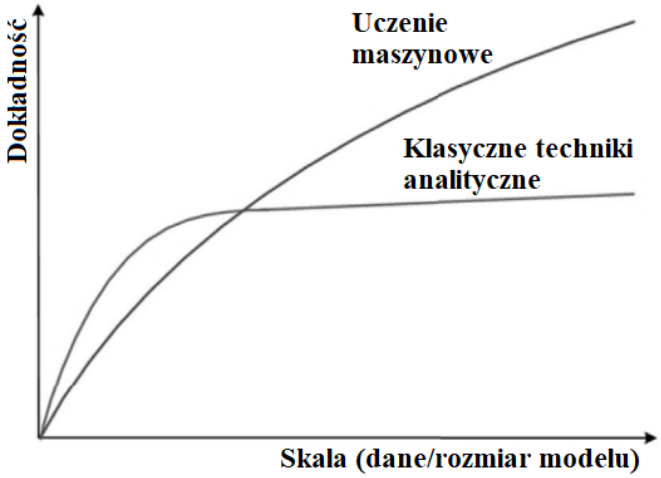
Open Images [19,50] to baza danych zawierająca aktualnie 9.2 mln. obrazów z jednolitymi adnotacjami umożliwiającymi między innymi klasyfikację obrazów, detekcję obiektów oraz rozpoznawanie relacji wizualnych. Obrazy zostały zaczerpnięte z serwisu Flickr i posiadają licencję Creative Commons Attribution, co pozwala na ich przetwarzanie i udostępnianie. Obrazy często przedstawiają skomplikowane sceny z wieloma obiektami (średnio 8 oznaczonych obiektów na jednym obrazie). Open Images oferuje ogromny zbiór danych pod wieloma względami: 30.1 mln etykiet dla 19800 kategorii klasyfikacji obrazów, 15.4 mln. BB dla 600 klas obiektów oraz 375 tys. adnotacji dotyczących relacji wizualnych obejmujących 57 klas.

YouTube-8M [51] to baza danych opracowana przez zespół badawczy Google. Baza zawiera ponad 8 mln. filmów o łącznej długości ok. 500 tys. godzin, które zostały pobrane z popularnej platformy YouTube. Na podstawie średniej długości filmów szacuje się, że baza zawiera ok. 1.9 mld. ujęć poklatkowych, na których oznaczono 4800 klas obiektów. Baza zawiera filmy o różnych długościach, z różnymi rodzajami treści i w różnych językach. Dzięki temu YouTube-8M jest chętnie używana do realizacji zadań związanych z klasyfikacją wideo, rozpoznawaniem obiektów, generowaniem podtytułów i rekomendacji wideo oraz wiele innych.

RoboFlow [52] oraz Kaggle [53] to platformy internetowe udostępniające cenne narzędzia dla naukowców i deweloperów pracujących nad rozwiązaniami w różnych obszarach badawczych, także CV. Obie platformy koncentrują się na przyspieszaniu i upraszczaniu procesów związanych z uczeniem maszynowym, jednak Kaggle charakteryzuje szerszy obszar zastosowań (analiza danych, uczenie maszynowe w różnych dziedzinach), podczas gdy Roboflow specjalizuje się głównie w dziedzinie CV. Dzięki aktywnej społeczności użytkowników platformy oferują także ogólnodostępne zbiory baz danych, wśród których można wyróżnić rzadko występujące klasy obiektów takie jak „*ogień*" czy „*tablice rejestracyjne*".

2.4.7 Podsumowanie

Na podstawie przeprowadzonej analizy można wyróżnić dwie grupy detektorów: tradycyjne analityczne oraz detektory DL. Wybór właściwej grupy detektorów, spełniającej założenia projektowanego systemu IVSS, został poprzedzony analizą zależności dokładności i skali dla technik opartych na uczeniu maszynowym i klasycznych technik analitycznych, której wykres przedstawiono na Rys. 2.7.



Rys. 2.7 Wykres zależności dokładności i skali modelu dla technik opartych na uczeniu maszynowym i klasycznych technik analitycznych [3]

Analiza tego wykresu wskazuje przewagę klasycznych technik analitycznych w aspekcie dokładności tylko przy implementacjach realizujących przetwarzanie niewielkiej liczby danych. Sytuacja zmienia się wraz ze wzrostem rozmiaru danych i wówczas techniki oparte na ML wykazują znacznie większą dokładność. Kolejnym czynnikiem decyzyjnym jest łatwość implementacji. Detektory tradycyjne oparte na technikach analitycznych wymagają ręcznego opracowywania skomplikowanych zestawów reguł dotyczących ekstrakcji cech. Taka konieczność nie występuje w przypadku ML. Co więcej większość omawianych detektorów opartych na DL jest dostępna w formie już wytrenowanych modeli, gotowych do implementacji programowej [3]. Biorąc pod uwagę powyższe czynniki, detekcja i klasyfikacja obiektów w ramach niniejszych badań zostanie zrealizowana przy użyciu technik DL.

Techniki DL dzielą się na detektory dwustopniowe oraz jednostopniowe, przy czym istotną różnicą jest stosunek szybkości przetwarzania danych do uzyskiwanej dokładności detekcji i klasyfikacji [20]. Detektory dwustopniowe wykazują lepszą dokładność kosztem szybkości, co związane jest ze złożonością ich architektury. Detektory jednostopniowe cechuje nieco mniejsza dokładność, ale większa szybkość przetwarzania danych. Z uwagi na fakt, iż jednym z głównych założeń koncepcyjnych przedmiotowej rozprawy jest osiągnięcie przetwarzania dużej liczby danych w czasie rzeczywistym, autor rozprawy zdecydował o wyborze podgrupy detektorów jednostopniowych.

Istnieje wiele rozwiązań z zakresu jednostopniowych detektorów DL. W Tab. 2.2 umieszczono opracowane przez autora porównanie wybranych detektorów ze względu na ich przydatność w projektowanym IVSS.

Tab. 2.2 Porównanie wybranych detektorów jednostopniowych opartych na DL

Detektor	**Szybkość przetwarzania [*FPS*]**	***AP* [%]**	**Język programowania**
YOLO v4	23^{M}	43.5	C++
SSD	22^{M}	28.8	Python
Retina-Net	5^{M}	38	Python
CornerNet	4^{M}	40.5	Python
CenterNet	8^{P}	45.1	Python
DETR	10^{V}	44.9	Python

^{M} – testy przeprowadzone z użyciem architektury Nvidia Maxwell
^{P} – testy przeprowadzone z użyciem architektury Nvidia Pascal
^{V} – testy przeprowadzone z użyciem architektury Nvidia Volta

Porównanie zostało opracowane na podstawie trzech kryteriów: szybkości przetwarzania obrazów, średniej precyzji modelu oraz języka implementacji. Dane reprezentujące dwa pierwsze kryteria zostały przedstawione w odniesieniu do testów przeprowadzonych na popularnym zbiorze danych MS-COCO [49]. W aktualnej literaturze nie znaleziono oficjalnych porównań szybkości działania wybranych detektorów przeprowadzonych przy użyciu jednej konfiguracji sprzętowej. Aby zapewnić wiarygodne porównanie analizowanych detektorów, wartości szybkości przetwarzania detektorów YOLO v4, SSD, Retina-Net oraz CornerNet zaczerpnięto z pracy [33], gdzie pomiary realizowane zostały przy użyciu stanowiska badawczego wyposażonego w procesory graficzne Nvidia Maxwell, natomiast dane dotyczące szybkości przetwarzania CenterNet pochodzą z pracy [46], gdzie zastosowano architekturę Nvidia Pascal, a wartość wskaźnika szybkości działania detektora DETR pochodzi z pracy [47], gdzie zastosowano architekturę Nvidia Volta. Porównanie wskaźników szybkości przetwarzania danych poszczególnych detektorów wskazuje na przewagę detektora YOLO v4, pomimo użycia najstarszej (w tym porównaniu) architektury GPU (Nvidia Maxwell). Wartość *AP* detektora YOLO v4 jest również wysoka, tracąc niecałe dwa punkty procentowe do najdokładniejszego (w

tym porównaniu) detektora CenterNet. Ostatnim kluczowym aspektem analizy porównawczej jest możliwość implementacji modelu detektora w wybranym języku programowania C++. W tym aspekcie tylko YOLO v4 zapewnia taką możliwość.

Na podstawie przyjętych kryteriów porównawczych oraz wyników przeprowadzonej analizy, do realizacji zadań detekcji i klasyfikacji obiektów w projektowanym systemie IVSS autor postanowił wybrać detektor YOLO v4.

2.5 Techniki optycznego rozpoznawania znaków alfanumerycznych

Techniki optycznego rozpoznawania znaków alfanumerycznych (OCR) [54] od niespełna stu lat stanowią jeden z kluczowych aspektów badań prowadzonych w obszarze CV. Proces OCR obejmuje dwa podstawowe elementy: lokalizację obszarów występowania tekstu na obrazie i translację graficznych znaków alfanumerycznych do interpretowalnej postaci cyfrowej. Szerokie spektrum zastosowań technik OCR wskazuje na zadania związane z cyfryzacją pisma odręcznego, interpretacją dokumentów drukowanych, automatycznym tłumaczeniem tekstów graficznych, rozpoznawaniem numerów tablic rejestracyjnych, jak również zastosowaniem w aplikacjach wspierających osoby niewidome i słabowidzące.

2.5.1 Techniki ANPR

Automatyczne rozpoznawanie numerów tablic rejestracyjnych (ANPR) [55,56] to jedno z głównych zastosowań OCR w obszarze inteligentnych systemów nadzoru wideo. Implementacja techniki ANPR umożliwia m.in. identyfikację właścicieli pojazdów, nadzór nad wjazdem do stref zastrzeżonych, obliczanie zajętości miejsc parkingowych oraz wyznaczanie czasu parkowania (np. w systemach automatycznych parkometrów). Na Rys.2.8 przedstawiono przykładowy rezultat analizy obrazu techniką ANPR.

Rys. 2.8 Rezultat analizy obrazu techniką ANPR [56]

Jedną z najbardziej efektywnych metod realizacji ANPR jest wykorzystanie zjawiska retrorefleksji światła podczerwonego od powierzchni tablic rejestracyjnych [56]. Tablice rejestracyjne są często pokryte specjalnymi materiałami retrorefleksyjnymi, zawierającymi mikropryzmaty, które umożliwiają odbicie wiązki światła podczerwonego do jego źródła z bardzo

małym rozproszeniem. Zjawisko to pozwala na wyraźne odczytanie numerów rejestracyjnych pojazdów nawet w trudnych warunkach oświetleniowych, takich jak noc. Ponieważ jednak projektowany system IVSS będzie głównie przeznaczony do pracy z kamerami rejestrującymi barwy w spektrum światła widzialnego, zastosowanie zjawiska retrorefleksji nie będzie szczegółowiej omawiane.

W literaturze można spotkać dwa najczęściej stosowane rodzaje implementacji technik ANPR: manualnie trenowane modele ML [55,56] lub gotowe silniki OCR [57]. Różnice między nimi dotyczą przede wszystkim elastyczności i skomplikowania implementacji.

Implementacja manualnie trenowanych modeli ML [55,56] polega na indywidualnym tworzeniu i trenowaniu modeli przeznaczonych do realizacji detekcji tablic rejestracyjnych, a następnie procesu OCR, przy użyciu zaawansowanych technik uczenia maszynowego (głównie architektur sieci CNN, RNN oraz LSTM). Pozwala to na bardziej szczegółową kontrolę procesu rozpoznawania i dostosowanie modelu do specyficznych wymagań, natomiast jest bardziej czasochłonne oraz wymaga zaawansowanej wiedzy eksperckiej i dostępu do dedykowanych zbiorów danych uczących.

Gotowe silniki OCR [57] to w pełni funkcjonalne algorytmy rozwijane przez lata i wspierane przez rozległe społeczności programistów i użytkowników. Implementacja silników OCR umożliwia osiągnięcie wysokiej dokładności rozpoznawania, a także elastyczność oraz łatwość integracji w różnego rodzaju projektach. Dzięki szerokiemu zakresowi obsługiwanych alfabetów, języków i zaawansowanym algorytmom przetwarzania obrazu, gotowe silniki OCR skutecznie realizują wymagające scenariusze rozpoznawania tekstu. Główną wadą implementacji silników OCR w zadaniach ANPR jest konieczność użycia zewnętrznych metod detekcji obszarów występowania tablic rejestracyjnych.

Z uwagi na łatwość implementacji, wsparcie szerokiej społeczności użytkowników oraz możliwą do osiągnięcia dokładność, autor postanowił zrealizować zadanie ANPR używając implementacji gotowych silników OCR.

2.5.2 Silniki OCR

Analiza literatury [57–60] wskazuje dwa najczęściej cytowane silniki OCR, udostępniane przez autorów w trybie otwartej licencji: Tesseract OCR oraz Easy OCR.

Silnik Tesseract [61] to rozbudowane narzędzie OCR udostępniane w trybie otwartej licencji Apache 2.0. Silnik ten umożliwia rozpoznawanie tekstu w ponad stu różnych językach (w tym polski) zapisanych przy pomocy 35 różnych alfabetów (w tym łaciński, arabski oraz chiński). Tesseract OCR jest przeznaczony do pracy na platformach systemowych Linux, Windows oraz Mac OS, umożliwiając rozpoznawanie tekstów z obrazów przy użyciu wiersza poleceń. Dodatkowo autorzy udostępniają dedykowane API, umożliwiające implementację programową

silnika w aplikacjach C, C++ oraz Python. Autorzy wprowadzają 14 metod segmentacji strony (PSM), które umożliwiają dostosowanie procesu detekcji do różnych typów geometrii, orientacji oraz zagęszczenia tekstu na obrazie.

Silnik Easy OCR [62] to narzędzie OCR, które podobnie jak Tesseract OCR udostępniane jest w trybie otwartej licencji Apache 2.0. Silnik ten umożliwia rozpoznawanie tekstu w ponad osiemdziesięciu różnych językach (w tym polski) zapisanych przy pomocy różnych alfabetów (w tym łaciński, arabski oraz chiński). Easy OCR przeznaczony jest do pracy w aplikacjach Python.

Na podstawie przeprowadzonej analizy, autor rozprawy postanawia wybrać silnik Tesseract OCR do celu realizacji zadań związanych z ANPR. Uzasadnienie tego wyboru motywowane jest głownie wysoką skutecznością opisywaną w pracach przeglądowych [57–60], możliwością implementacji silnika w wybranym języku C++, dostępnością bogatej funkcjonalności oraz ciągłym rozwojem technologii przez twórców i szerokie grono użytkowników.

2.5.3 Podsumowanie

Proces realizacji zadania ANPR w projektowanym systemie IVSS będzie obejmował trzy kluczowe etapy: detekcja tablic rejestracyjnych, ekstrakcja grafiki tekstu z obrazu oraz rozpoznanie tekstu przy użyciu silnika OCR.

Pierwszy etap dotyczy precyzyjnej lokalizacji tablic rejestracyjnych na obrazie. W tym celu autor użyje oryginalnego modelu YOLO v4 umożliwiającego detekcję obiektów klasy „*samochód osobowy”*, „*samochód ciężarowy*” oraz „*autobus*”, które będą wspólnie traktowane jako jedna klasa obiektu „*pojazd*”. Następnie, dodatkowy, specjalnie wytrenowany model YOLO v4 tiny umożliwi detekcję tablic rejestracyjnych znajdujących się wyłącznie w obszarach obrazu oznaczonych uprzednio jako „*pojazd*”. Takie podejście pozwoli osiągnąć wymaganą precyzję detekcji oraz ograniczy możliwość wystąpienia nieprawdziwych detekcji mogących wystąpić poza obrysem pojazdu.

Drugi etap będzie obejmował ekstrakcję grafiki tekstu z obszaru obrazu oznaczonego jako tablice rejestracyjne poprzez usunięcie wszystkich elementów graficznych nie będących tekstem. W tym celu autor rozprawy użyje operacji takich jak zmiana przestrzeni barw, skalowanie, korekta perspektywy i operacje morfologiczne, zgodnie wytycznymi opracowanymi przez autorów silnika Tesseract OCR [63] oraz wynikami badań zawartymi w pracy [59].

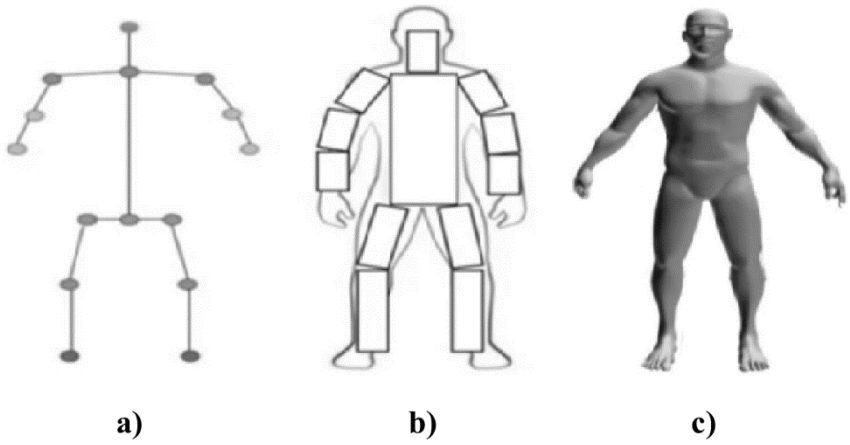
Ostatni etap będzie obejmował użycie silnika Tesseract OCR w celu rozpoznania numerów tablic rejestracyjnych. Uzyskany w ten sposób tekst cyfrowy zostanie porównany z danymi pojazdów zapisanymi w bazie, aby automatycznie zidentyfikować właściciela pojazdu. Głównym przeznaczeniem projektowanej funkcji ANPR będzie realizacja zadania dotyczącego nadzoru wjazdu do stref zastrzeżonych. Funkcja ta może zostać rozwinięta w celach obliczania zajętości miejsc parkingowych oraz wyznaczenia czasu parkowania.

2.6 Estymacja sylwetki człowieka

Estymacja sylwetki człowieka (HPE) [64] w kontekście dziedziny CV jest obszarem badawczym, odnoszącym się do analizy obrazów lub sekwencji wideo celem precyzyjnego określenia położenia i kształtu ludzkiego ciała na podstawie danych wizyjnych. Wybrany w rozdziale 2.4.7 detektor YOLO v4 umożliwia detekcję osób na podstawie analizy danych wizyjnych 2D. Niestety, typ adnotacji sylwetki w postaci BB nie pozwala na szczegółową analizę ciała człowieka. Problem ten może zostać rozwiązany poprzez implementację dodatkowej techniki, zorientowanej na bardziej precyzyjną metodę adnotacji sylwetki człowieka, której metodyka wyboru i uzasadnienie zostaną omówione w tym rozdziale.

2.6.1 Graficzna reprezentacja sylwetki człowieka

W literaturze [65] można spotkać trzy najczęściej stosowane rodzaje modeli sylwetek, których reprezentację graficzną przedstawiono na Rys. 2.9.



Rys. 2.9 Graficzna reprezentacja modeli sylwetek człowieka stosowanych w systemach CV: a) model kinematyczny, b) model planarny, c) model wolumetryczny [65]

Model kinematyczny (szkieletowy) [65] przedstawiony na Rys. 2.9a używany jest do szacowania postawy ciała zarówno w przestrzeni 2D jak i 3D. W tym modelu reprezentacja sylwetki dotyczy wyznaczania określonych punktów charakterystycznych części ciała (tzw. keypoints), np. głowa, dłonie, stopy oraz punktów połączeń tych części. Dzięki temu model ten może być używany do reprezentowania relacji między różnymi częściami ciała. Model kinematyczny ma jednak ograniczenia związane z brakiem informacji o teksturze czy ogólnym kształcie sylwetki.

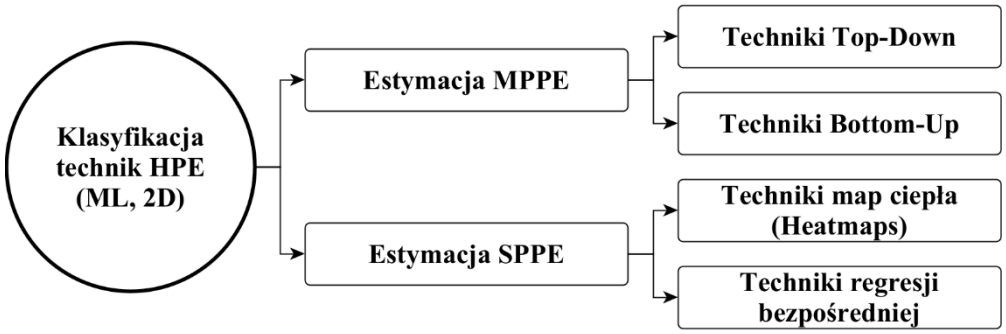
Model planarny (konturowy) [65] przedstawiony na Rys. 2.9b stosowany jest głównie do reprezentowania konturów ciała w przestrzeni 2D. Ten model służy do ilustrowania ogólnego kształtu ciała ludzkiego, widzianego z konkretnej perspektywy. Zazwyczaj części ciała są reprezentowane przy użyciu prostokątów przybliżających kontury sylwetki człowieka.

Model wolumetryczny [65] przedstawiony na Rys. 2.9c stosowany jest do szacowania pozycji w przestrzeni 3D. Model ten tworzony jest przy pomocy geometrycznych siatek o różnych kształtach i rozmiarach, co umożliwia precyzyjną reprezentację sylwetki człowieka, uwzględniając tekstury i topografię powierzchni ciała.

2.6.2 Klasyfikacja technik HPE

Na wczesnym etapie badań z zakresu estymacji sylwetki człowieka stosowano tradycyjne metody analityczne takie jak HOG [66], łańcuchy krawędzi Edgelet [67] i metoda Pictorial Structure Model [68]. Aplikacje oparte na metodach tradycyjnych charakteryzowały się jednak niską dokładnością i wymagały stosowania bardzo skomplikowanych algorytmów, przez co nie spełniały pożądanych oczekiwań. Ograniczenia metod tradycyjnych zostały w dużym stopniu wyeliminowane poprzez zastosowanie modeli ML, używanych po dziś dzień [65].

Klasyfikacja technik HPE, typu ML, stosowanych w obrazowaniu 2D, została zaprezentowana na Rys. 2.10. Główna linia podziału dotyczy estymacji jednoosobowej (SPPE) i wieloosobowej (MPPE) [69].



Rys. 2.10 Klasyfikacja technik HPE typu ML stosowanych w obrazowaniu 2D [69]

Techniki estymacji SPPE [70] odnoszą się do detekcji pojedynczej sylwetki człowieka i są stosowane wówczas, gdy na obrazie wejściowym jest widoczna tylko jedna osoba. W większości przypadków zadanie estymacji SPPE można rozwiązać stosując techniki regresji bezpośredniej lub techniki oparte na mapach ciepła. Techniki oparte na regresji bezpośredniej realizują bezpośrednią regresję punktów kluczowych sylwetki z obrazu wejściowego. W drugiej grupie technik, punkty kluczowe sylwetki są estymowane na podstawie analizy mapy ciepła wygenerowanej z obrazu wejściowego.

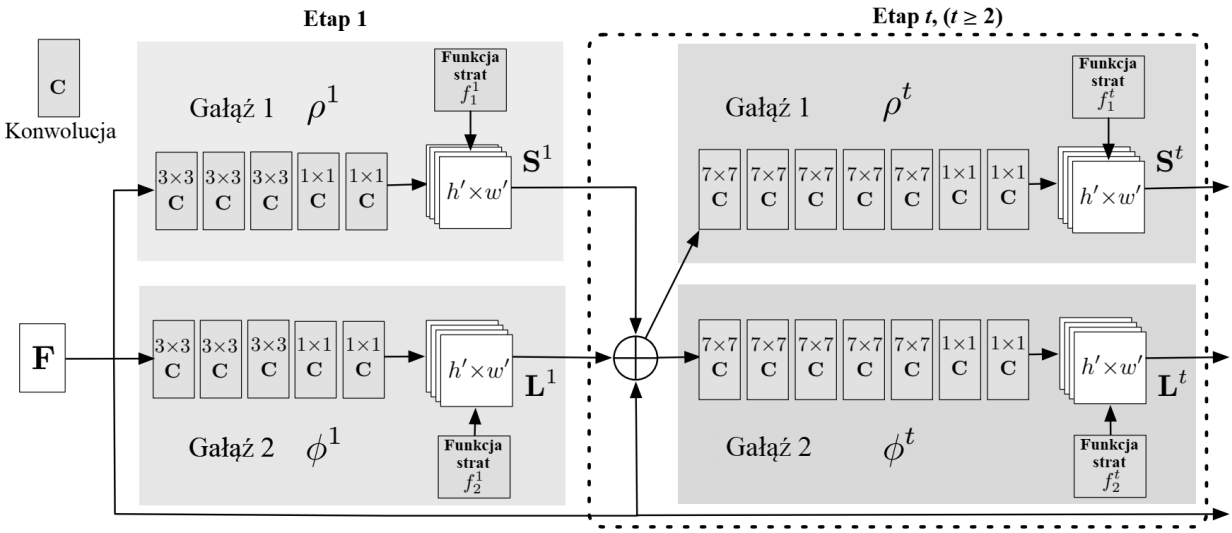
Techniki estymacji MPPE [70] są bardziej skomplikowane z uwagi na konieczność detekcji i klasyfikacji wielu osób jednocześnie. W większości przypadków zadanie estymacji MPPE można rozwiązać dwojako. Pierwsze podejście typu Top-Down skupia się najpierw na detekcji wszystkich postaci na całym obrazie, a następnie szacowaniu punktów kluczowych każdej z rozpoznanych sylwetek. W tym podejściu czas przetwarzania danych zależy głównie od liczby osób znajdujących się na analizowanym obrazie. Drugie podejście typu Bottom-Up najpierw

identyfikuje wszystkie potencjalne punkty kluczowe sylwetek na całym obrazie, a następnie realizuje ich grupowanie celem utworzenia sylwetek poszczególnych postaci. Podejście Bottom-Up jest zazwyczaj szybsze w porównaniu do Top-Down, ponieważ nie wymaga dodatkowego etapu estymowania pozycji dla każdej wykrytej osoby.

2.6.3 Przegląd wybranych bibliotek HPE

Projektowany system IVSS docelowo przeznaczony będzie do pracy w monitoringu publicznym, gdzie liczba obserwowanych osób jest zmienna. Zatem, z uwagi na brak zastosowania w niniejszej rozprawie, techniki z grupy SPPE nie będą one szczegółowiej omawiane. Przegląd wybranych technik HPE zostanie zatem zorientowany tylko na grupę technik MPPE.

Jedną z pierwszych otwarto źródłowych bibliotek realizujących zadanie estymacji MPPE, zapewniającą działanie w czasie rzeczywistym, jest OpenPose [71] (Bottom-Up). Biblioteka implementuje dwugałęziową, wieloetapową architekturę sieci konwolucyjnej (CNN), której graficzną reprezentację przedstawiono na Rys. 2.11.



Rys. 2.11 Graficzna reprezentacja dwugałęziowej architektury sieci konwolucyjnej modelu OpenPose [71]

W pierwszym kroku z obrazu wejściowego generowany jest wektor map cech (F). Wektor ten stanowi wejście do pierwszego etapu każdej z dwóch równoległych gałęzi sieci (S^1 i L^1). Pierwsza gałąź (S) realizuje predykcję map ufności (ang. *confidence maps*), które kodują prawdopodobieństwo lokalizacji poszczególnych części ciała na obrazie. Druga gałąź (L) realizuje predykcję tzw. Part Affinity Fields, które kodują stopień powiązania między częściami ciała. Każda gałąź sieci została opracowana jako struktura wieloetapowa, przy czym minimalna liczba etapów to 2 ($t \geq 2$). Każdy kolejny etap przyjmuje na wejście dane otrzymane w wyniku połączenia oryginalnego wektora map cech obrazu (F) z predykcjami uzyskanymi w poprzednim etapie z obu gałęzi (ρ^{t-1} oraz $\emptyset^{t-1}$). Opisany proces iteracyjnego wzmacniania dokładności predykcji przekłada się na wysoką efektywność OpenPose w trudnych zadaniach estymacji jak np.

okluzje czy skomplikowana postawa ciała. Kod źródłowy OpenPose [72] jest dostępny zarówno w języku C++ jak i w języku Python.

DeeperCut [73] to kolejna otwarto źródłowa biblioteka typu MPPE (Bottom-Up), która jest ulepszoną wersją biblioteki DeepCut [74]. Koncepcja estymacji sylwetek w modelu DeeperCut obejmuje trzy kluczowe komponenty: zaawansowany detektor części ciała, innowacyjny model parowania części ciała oraz strategia optymalizacji inkrementalnej. Zaawansowany detektor części ciała to model DL oparty na architekturze ResNet [44] o głębokości 152 warstw. Zastosowanie tak głębokiej architektury sieci pozwoliło osiągnąć wysoką dokładność detekcji. Innowacyjny model parowania części ciała został użyty do predykcji wzajemnych położeń różnych części ciała względem siebie. Cel ten osiągnięto poprzez implementację dedykowanej sieci CNN, zdolnej do szacowania prawdopodobieństwa korelacji pojedynczych części ciała z częściami mogącymi tworzyć parę (np. znając lokalizację ramienia, model może przewidzieć prawdopodobne położenie łokcia). Strategia optymalizacji inkrementalnej sprowadza się do określenia priorytetu klas części ciała w procesie ich grupowania do postaci spójnych sylwetek. Klasy części ciała dzielone są na dwa zestawy: części opisujące sylwetkę w sposób bardziej niezawodny (np. głowa, barki) i części określające sylwetkę mniej niezawodnie (np. nadgarstki, łokcie). Zadanie grupowania rozpoczyna się od najbardziej niezawodnych klas części ciała, następnie stopniowo dołącza mniej niezawodne klasy. Finalnie, detektor DeeperCut dzięki metodzie określania priorytetu klas części ciała umożliwia efektywne przeszukiwanie przestrzeni rozwiązań, prowadząc do znacznego przyspieszenia procesu estymacji i poprawy dokładności wykrywania pozycji ciała względem poprzedniej wersji (DeepCut). Kod źródłowy biblioteki został zaimplementowany w środowisku MATLAB i przetestowany przez autorów tylko na platformie systemowej Linux [75].

Najnowsza wersja popularnej biblioteki YOLO – YOLO v8 [37] (Top-Down) zawiera szereg modeli wytrenowanych w celu detekcji i klasyfikacji obiektów (przy użyciu adnotacji typu klasyczny BB, zorientowany BB oraz segmentacja semantyczna), klasyfikacji obrazów, a także estymacji i śledzenia trajektorii ruchu sylwetek ciała. Implementacja środowiska YOLO v8 jest możliwa w języku Python [76], co sprawia, iż wydajność obliczeniowa może być niższa z uwagi na interpretację kodu w czasie rzeczywistym. Obecnie, brak rzetelnych publikacji autorskich dotyczących najnowszej wersji YOLO v8, co utrudnia ewentualne analizy i porównania. Analiza prac przeglądowych [38–40] dostarcza kilku informacji dotyczących sposobu przepływu informacji i architektury modelu. Architektura modelu YOLO v8 jest inspirowana architekturą modelu YOLO v5 [34] z pewnymi modyfikacjami. Kluczową zmianą jest wprowadzenie modułu tzw. „*cross-stage partial bottleneck with two convolutions*”, łączącego cechy wysokiego poziomu z informacjami kontekstowymi w celu poprawy dokładności detekcji obiektów. Ponadto, skrócono czas pracy mechanizmu NMS i zredukowano liczbę niepewnych obszarów predykcji eliminując estymację AB. Estymacja AB została zastąpiona metodą tzw. „*decoupled head*”, w której zadania

detekcji, klasyfikacji i regresji obiektów realizowane są w oddzielnych gałęziach sieci zwiększając dokładność i wydajność modelu. Model YOLO v8 używa funkcji aktywacji typu „*sigmoid*" do reprezentowania wyniku prawdopodobieństwa detekcji obiektu oraz funkcji typu „*softmax*" do reprezentowania prawdopodobieństw przynależności obiektów do każdej z możliwych klas.

AlphaPose [77] to kolejna otwarto źródłowa biblioteka typu MPPE (Top-Down), której działanie oparte jest na czterech głównych etapach: detekcji osób, regresji punktów kluczowych sylwetek, mechanizmie eliminacji redundantnych detekcji, oraz śledzeniu trajektorii wykrytych osób. Detekcja osób na obrazie realizowana jest przy użyciu detektorów YOLO v3 [32] lub EfficientDet [78], umożliwiających oznaczenie obszarów predykcji w adnotacji typu BB. Obszary detekcji postaci są następnie kadrowane i skalowane, dostosowując ich rozmiar i proporcje w celu umożliwienia realizacji kolejnych etapów przetwarzania. Etap regresji punktów kluczowych sylwetek zrealizowano z użyciem innowacyjnej techniki SIKR. Technika ta polega na analizowaniu symetrycznych właściwości rozkładu prawdopodobieństwa wokół przewidywanej lokalizacji punktu kluczowego, co pozwala na dokładniejsze i bardziej niezawodne określenie jego położenia na obrazie. AlphaPose stosuje technikę P-NMS, polegającą na eliminacji redundantnych estymacji pozycji, co jest szczególnie ważne w scenariuszach z tłumem ludzi, gdzie wiele postaci może zostać wykrytych wielokrotnie. Ostatnim kluczowym etapem działania algorytmu AlphaPose jest śledzenie trajektorii wielu osób realizowane z użyciem modułów PGA oraz MSIM. Moduł PGA używany jest do generowania cech reidentyfikacyjnych każdej wykrytej sylwetki, natomiast moduł MSIM dokonuje syntezy danych (sylwetek osób, cech reidentyfikacyjnych oraz adnotacji BB) umożliwiając generowanie finalnych predykcji śledzenia. Biblioteka AlphaPose została zaimplementowana w języku Python [79], przez co jej wydajność obliczeniowa może być niższa np. w porównaniu z OpenPose.

2.6.4 Podsumowanie

Na podstawie przeprowadzonych badań i analiz autor zdecydował o wyborze modelu kinematycznego do graficznej reprezentacji sylwetek człowieka. Wybór ten uzasadnia możliwość przeprowadzenia dokładnej analizy ruchu wybranych części ciała, co z kolei pozwoli na osiągnięcie wysokiej precyzji w zadaniach związanych z analizą aktywności osób na obrazie. Co więcej, zdecydowano o wyborze technik MPPE. Wynika to z docelowego przeznaczenia projektowanego systemu w monitoringu publicznym, gdzie liczba obserwowanych osób jest zmienna. Powyższe kryteria spełniają cztery wybrane biblioteki, których zasada działania została opisana w rozdziale 2.6.3. W Tab. 2.3 umieszczono opracowane przez autora porównanie wybranych bibliotek z uwagi na ich przydatność w projektowanym IVSS. Porównanie zostało opracowane na podstawie pięciu kryteriów: typ podejścia (Top-Down, Bottom-Up), średnia

Tab. 2.3 Porównanie wybranych bibliotek MPPE

Biblioteka	**Typ podejścia**	***AP* [%]**	**Zintegrowany moduł śledzenia sylwetek**	**Język programowania**	**Dostępność wytrenowanych modeli**
OpenPose	Bottom-Up	75.6	Tak, ale tylko w trybie SPPE	C++, Python	Publiczne repozytorium [72]
DeeperCut	Bottom-Up	59.4	Nie	M	Publiczne repozytorium [75]
YOLO v8	Top-Down	brak danych	Tak	Python	Publiczne repozytorium [76]
AlphaPose	Top-Down	82.1	Tak	Python	Publiczne repozytorium [79]

wartości precyzji estymacji, integracja modułu śledzenia sylwetek, język implementacji oraz dostępność wytrenowanych modeli.

Na podstawie analizy pracy przeglądowej [70] można stwierdzić, iż biblioteki HPE działające w podejściu typu Bottom-Up z reguły osiągają wyższą prędkość przetwarzania danych w przeciwieństwie do bibliotek typu Top-Down. Wobec tego, kryterium typu podejścia zawęża wybór bibliotek do OpenPose i DeeperCut.

Następne kryterium porównawcze odnosi się do wartości *AP* estymacji. Wartości tychże wskaźników podane w Tab. 2.3 pochodzą ze źródeł literaturowych [71,73,79] i odnoszą się do testów przeprowadzonych na pełnym zbiorze danych MPII [80]. Niestety, w aktualnej literaturze brak danych dotyczących wartości dokładności dla zbioru danych MPII i biblioteki YOLO v8. Pozostałe dane wskazują na przewagę biblioteki AlphaPose.

Kolejne kryterium porównawcze dotyczy zintegrowanego modułu śledzenia estymowanych sylwetek. Tę funkcjonalność posiadają YOLO v8, AlphaPose i OpenPose, przy czym ta ostatnia realizuje funkcje śledzenia tylko w przypadku obecności pojedynczej osoby na obrazie.

Następne kryterium odnosi się do możliwości implementacji biblioteki w wybranym języku programowania C++. Obecnie, tylko OpenPose zapewnia taką możliwość.

Ostatnie kryterium wskazuje, iż autorzy każdej z analizowanych bibliotek udostępniają publiczne repozytoria zawierające wytrenowane modele.

Na podstawie przeprowadzonej analizy, autor rozprawy zdecydował o wyborze biblioteki OpenPose z uwagi na wydajność wynikającą z typu podejścia Bottom-Up i implementację programową w wybranym języku C++. Autorzy biblioteki OpenPose udostępniają publicznie [72] pełen kod źródłowy, wytrenowane modele jak również oferują wsparcie dla użytkowników poprzez forum internetowe. Udostępnione modele umożliwiają detekcję całego ciała (modele składających się z 15, 18 i 25 punktów kluczowych), stóp (2 × 6 punktów kluczowych), dłoni (2 × 21 punktów kluczowych) oraz twarzy (70 punktów kluczowych). Ponadto, autorzy udostępniają pełną dokumentację programistyczną biblioteki dostępną online [81].

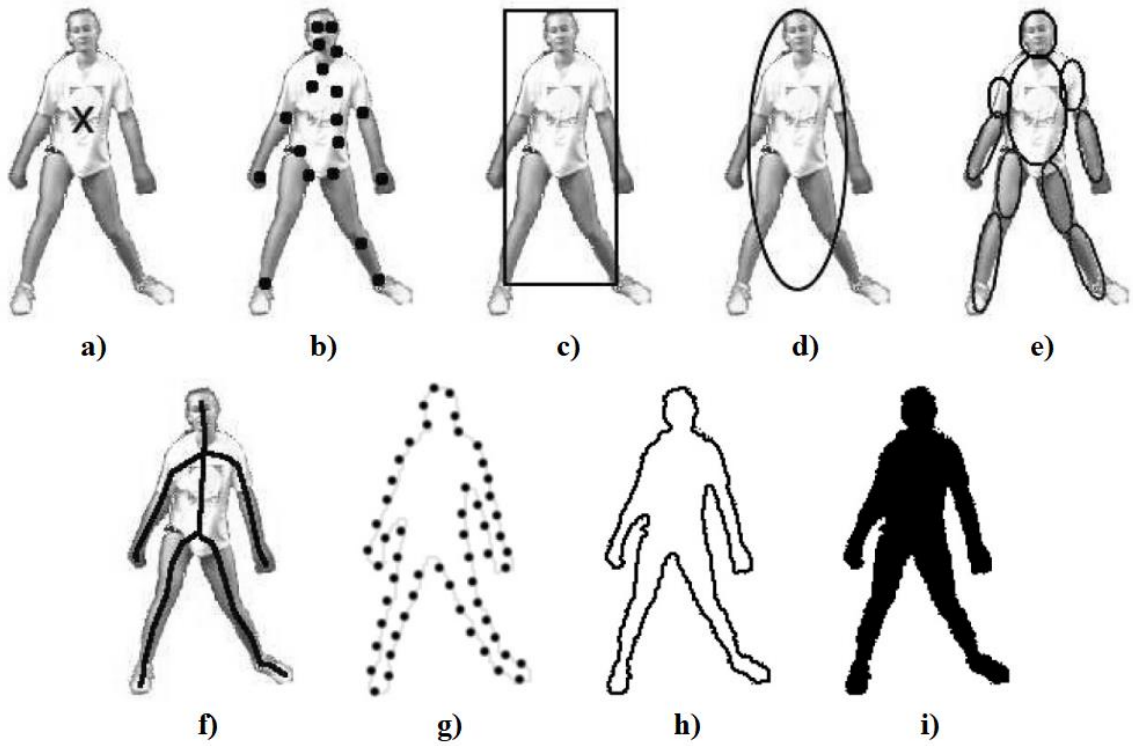
2.7 Śledzenie obiektów

Śledzenie wielu obiektów (MOT) [82] odgrywa kluczową rolę w obszarze systemów IVSS. Zdolność systemu do identyfikowania tych samych obiektów w sekwencji wideo umożliwia gromadzenie danych historycznych, reprezentujących zmiany ich lokalizacji w czasie. Zgromadzone dane można następnie użyć w kolejnych etapach analizy obrazu związanych z wyznaczaniem kierunku ruchu, trajektorii, prędkości przemieszczenia, aktywności oraz interakcji, a nawet predykcji lokalizacji w przypadku wystąpienia okluzji.

2.7.1 Wzorce cech

Jednym z głównych problemów występujących algorytmach MOT jest detekcja unikalnych, dokładnych i niezawodnych cech wizualnych, na podstawie których możliwe będzie odróżnienie docelowego obiektu od pozostałych na każdym kolejnym ujęciu. W pracy [83] dokonano następującej klasyfikacji cech wizualnych: cechy koloru definiujące obiekt na podstawie histogramu barw, cechy kształtu definiujące obiekt na podstawie geometrii obrysu, cechy tekstury definiujące obiekt poprzez detekcję wzorców geometrycznych oraz cechy ruchu definiujące obiekt poprzez detekcję zmian kształtu obiektu w czasie (np. Optical Flow).

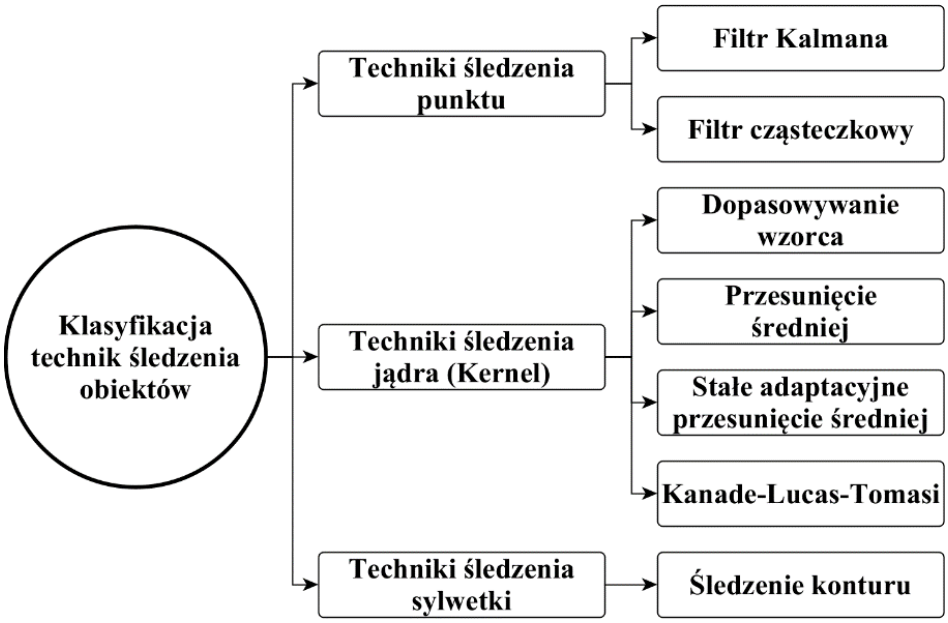
Adnotacja obiektów stosowana w algorytmach MOT zazwyczaj sprowadza się do użycia wzorców typu [83]: punkt, centroida, BB, elipsa, a także model przegubowy, szkieletowy, obrysowy oraz pełna segmentacja. Na Rys. 2.12 przedstawiono przykłady adnotacji sylwetek osób stosowane w algorytmach MOT.



Rys. 2.12 Przykłady adnotacji sylwetek osób stosowane w algorytmach MOT: a) centroida, b) wielopunktowy, c) ramka (BB), d) elipsa, e) przegubowy, f) szkieletowy, g) punkty kontrolne obrysu, h) pełny obrys, i) pełny obszar kształtu (segmentacja) [83]

2.7.2 Klasyfikacja technik śledzenia

Analiza pracy przeglądowej [83] pozwala zorientować klasyfikację technik śledzenia na trzy grupy: techniki śledzenia punktu, jądra i sylwetki. Na Rys. 2.13 przedstawiono ogólną klasyfikację technik śledzenia obiektów na obrazie.



Rys. 2.13 Ogólna klasyfikacja technik śledzenia obiektów [83]

Techniki śledzenia punktu

W technikach śledzenia punktu proces śledzenia wymaga aktualizowania cech punktu na każdym nowym ujęciu. Głównymi metodami stosowanymi w tej podgrupie są [83] filtr Kalmana i filtr cząsteczkowy.

Filtr Kalmana [83] jest rekurencyjnym estymatorem stanu, umożliwiającym estymację stanu systemu liniowego w obecności zaszumionych i niepełnych danych. Jego działanie obejmuje fazy predykcji i korekcji. W fazie predykcji filtr dokonuje estymacji stanu obiektu na nowym, nieznanym dotąd ujęciu. Następnie, w fazie korekcji, aktualizuje przewidywany stan na podstawie danych z aktualnego ujęcia w celu zmniejszenia niepewności estymacji. Filtr Kalmana jest szczególnie skuteczny w przypadku systemów liniowych i gaussowskich rozkładów błędów, oferując analityczne rozwiązanie problemu estymacji.

Filtr cząsteczkowy [83], znany również jako sekwencyjna metoda Monte Carlo, jest techniką estymacji stanu systemów nieliniowych i niegaussowskich. W celu reprezentacji rozkładu prawdopodobieństwa stanu filtr cząsteczkowy używa zestawu próbek (cząstek). Każda cząstka reprezentuje możliwy stan systemu, a jej waga odzwierciedla stopień, w jakim dany stan pasuje do rzeczywistych detekcji. Filtr aktualizuje stan systemu poprzez selekcję, propagację i ponowną

aktualizację wag cząstek na podstawie nowych detekcji, co pozwala na aproksymację rozkładu prawdopodobieństwa stanu systemu.

Techniki śledzenia jadra

Techniki śledzenia jądra realizują proces śledzenia bazując na cechach wyznaczonych jednokrotnie w momencie pierwszej detekcji obiektu. Głównymi technikami stosowanymi w tej podgrupie są [83] dopasowywanie wzorca, przesunięcie średniej, stałe adaptacyjne przesunięcie średniej oraz Kanade-Lucas-Tomasi.

Technika dopasowywania wzorca [83] polega na przesuwaniu szablonu (fragmentu obrazu zawierającego wzór śledzonego obiektu) po całym obrazie, obliczając w każdej pozycji podobieństwo między szablonem a bieżącym fragmentem obrazu. Najlepsze dopasowanie wskazuje położenie obiektu. Metoda ta jest relatywnie prosta w implementacji, lecz może generować błędne wyniki w przypadku zmiany skali, rotacji lub innych deformacji obiektu.

Technika przesunięcia średniej [83] umożliwia generowanie wyników na podstawie analizy gęstości rozkładu cech obrazu. Algorytm iteracyjnie przesuwa okno (jądro) w kierunku większej gęstości punktów cech na obrazie, aż osiągnie lokalne maksimum gęstości, które odpowiada przewidywanemu położeniu obiektu. Technika ta jest szczególnie użyteczna w śledzeniu obiektów na podstawie ich koloru lub tekstury, lecz może generować błędne wyniki w przypadku znaczących zmian tła lub pojawiających się okluzji obiektów.

Technika stałego adaptacyjnego przesunięcia średniej [83] jest rozszerzeniem techniki przesunięcia średniej umożliwiającym dostosowywanie parametrów jądra w każdej iteracji. Przyczynia się to do wzrostu efektywności śledzenia w przypadku wystąpienia zmian rozkładu cech obiektu.

Technika Kanade-Lucas-Tomasi [83] to popularna metoda śledzenia punktów charakterystycznych (np. narożników obiektów) w sekwencji obrazów. Opiera się na założeniu, że ruch punktów charakterystycznych między kolejnymi ujęciami jest relatywnie mały i można go aproksymować na podstawie różnicy odległości tych punktów. Algorytm może używać gradientów obrazu do obliczenia przepływu optycznego i estymacji ruchu punktów charakterystycznych pozwalając na efektywne śledzenie ruchu obiektów. Technika ta jest wydajna obliczeniowo, lecz może generować błędne wyniki w przypadku dużej dynamiki ruchów lub znacznych deformacji obiektu.

Techniki śledzenia sylwetki

Techniki śledzenia sylwetki, podobnie jak techniki śledzenia jądra, realizują proces śledzenia bazując na cechach wyznaczonych jednokrotnie w momencie pierwszej detekcji obiektu. W tej

podgrupie wyróżnia się technikę śledzenia konturu [83] umożliwiającą efektywne reprezentowanie i śledzenie obiektów o nieregularnych kształtach, takich jak palce czy dłonie, które trudno jest przedstawić za pomocą prostych kształtów geometrycznych. Kontury mogą być reprezentowane na różne sposoby, przy czym często stosuje się funkcję wskaźnikową, która obszary nie będące obiektem oznacza zerami, a obszar obiektu jedynkami. Analiza stopnia nakładania się części obiektu w bieżącym ujęciu na obiekt w poprzednim obrazie stanowi podstawę decyzyjną.

2.7.3 Przetwarzanie danych

W aspekcie sposobu przetwarzania danych algorytmy śledzenia dzielimy na dwie grupy [84]: stochastyczne i deterministyczne.

Ogólna struktura problemu estymacji w stochastycznych algorytmach śledzenia [84] dotyczy realizacji zadania predykcji lokalizacji obiektu na kolejnym, nieznanym dotąd ujęciu. Proces ten obejmuje porównanie obserwacji bieżącego ujęcia z wynikami poprzedniej predykcji, umożliwiając obliczenie parametrów korekty dla następnej predykcji. W tym celu stosuje się głównie metody takie jak filtr Kalmana oraz filtr cząsteczkowy. Użycie algorytmów stochastycznych uniemożliwia osiągnięcie stabilności i powtarzalności wyników, nawet przy próbie ponownej analizy tego samego nagrania wideo. Algorytmy te jednak lepiej radzą sobie z niepewnością, zwłaszcza gdy obiekty są częściowo przysłonięte, a ich ruch jest trudny do przewidzenia.

Działanie algorytmu śledzenia deterministycznego [84] polega na uzyskaniu stałego, powtarzalnego wyniku na podstawie danych historycznych oraz aktualnej obserwacji, z wykluczeniem etapu predykcji. Oznacza to, że wynik śledzenia będzie taki sam przy wielokrotnym uruchomieniu algorytmu dlatego samego nagrania. W podejściu deterministycznym głównym problemem jest precyzyjne przyporządkowanie tych samych obiektów występujących na nowych ujęciach. W celu rozwiązania tego problemu opracowano szereg metod optymalizacji deterministycznej, wśród których można wyróżnić np. algorytm węgierski [84].

2.7.4 Uczenie maszynowe

Techniki śledzenia typu ML dzieli się na trzy kategorie [82]: dyskryminacyjne, uczenie ze wzmocnieniem i generatywne.

Podejście dyskryminacyjne [82] odnosi się do użycia modeli uczenia płytkiego lub głębokiego jako klasyfikatorów. Różnica polega na tym, że modele uczenia głębokiego są zdolne do samodzielnej ekstrakcji cech obiektu, natomiast modele uczenia płytkiego wymagają zastosowania zewnętrznych modeli ekstrakcji cech. Jednym z popularnych rozwiązań z grupy uczenia głębokiego są modele syjamskie. Architektura tych modeli zawiera dwa wejścia i jedno wyjście. Na oba wejścia podawane są obrazy z bieżącego i poprzedniego ujęcia, w celu analizy

podobieństwa między nimi, oraz ustalenia obecności określonych obiektów na obu klatkach. Inne modele z tej grupy to modele uczenia ścieżek i modele grafów. W tych rozwiązaniach zastosowano teorię ścieżek i teorię grafów w celu przedstawienia relacji przestrzennych między obiektami na obrazach.

Techniki śledzenia opartego na uczeniu ze wzmocnieniem [82] odnoszą się do paradygmatu, w którym tzw. „*agent*”, a więc model ML, uczy się podejmować decyzje poprzez interakcję z otoczeniem. Głównym celem jest zoptymalizowanie sekwencji działań „*agenta*” w celu maksymalizacji kumulatywnej nagrody, którą w tym przypadku jest prawidłowe identyfikowanie obiektów na kolejnych obrazach.

Techniki generatywne [82] odnoszą się do przeszukiwania i znajdywania tych obszarów obrazu, które są najbardziej podobne do analizowanego obiektu według określonego kryterium. Jednym z przykładów tych technik są metody oparte na filtrze korelacji.

2.7.5 Podsumowanie

W projektowanym systemie IVSS, śledzenie obiektów będzie realizowane dla wyników detekcji osób na obrazie, których typ adnotacji to BB. Dzięki zastosowanym detektorom OpenPose oraz YOLO v4 wyniki detekcji BB będą aktualizowane z każdym nowym ujęciem. Postanowiono zatem, aby wzorcem cechy śledzonego obiektu był pojedynczy punkt środka górnej linii BB. Takie podejście zupełnie wyklucza wpływ zmian wysokości BB oraz uśrednia jego szerokość wskazując na domniemaną oś symetrii śledzonej osoby.

W celu ograniczenia konsumpcyjności obliczeniowej projektowanego systemu, związanej z użyciem zbyt wielu modeli ML, algorytm śledzenia zostanie opracowany przy użyciu klasycznych technik analitycznych opartych na śledzeniu punktu. Proponowana metoda zostanie opracowana w sposób umożliwiający dokonanie syntezy deterministycznego i stochastycznego procesu przetwarzania danych. W podejściu deterministycznym, dopasowanie obiektów będzie realizowane na podstawie wyznaczania minimalnej odległości pomiędzy BB w sekwencji ujęć. W podejściu stochastycznym zostanie użyty filtr Kalmana, generujący predykcję lokalizacji osób, których detekcja na nowym ujęciu nie była możliwa np. z powodu wystąpienia chwilowej okluzji.

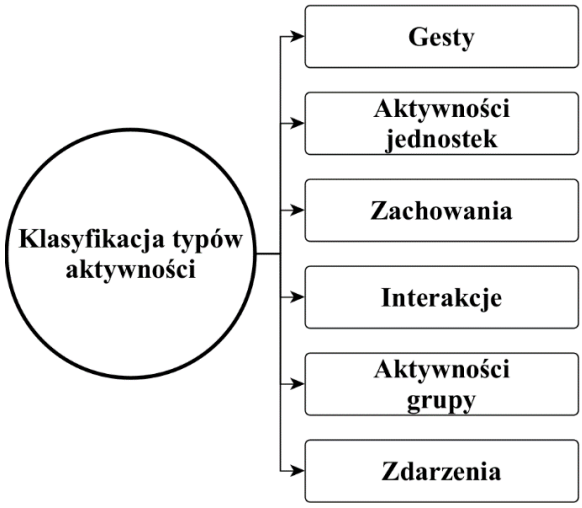
2.8 Techniki analizy aktywności człowieka

Techniki wizyjnej analizy aktywności człowieka (HAR) [85–87] dotyczą automatycznej interpretacji danych wizyjnych, mającej na celu zrozumienie i klasyfikację działań podejmowanych przez ludzi. Obecnie techniki HAR znajdują wszechstronne zastosowanie, począwszy od interaktywnych systemów rozrywkowych, po systemy wsparcia dla osób starszych i niepełnosprawnych. W niniejszej dysertacji zastosowanie technik HAR zostanie zorientowane na możliwość opracowania efektywnych i wydajnych obliczeniowo metod, umożliwiających

detekcję i klasyfikację aktywności niepożądanych, mogących stanowić podstawę identyfikacji sytuacji zagrażających bezpieczeństwu osób lub/i dóbr materialnych, znajdujących się w obszarze monitoringu.

2.8.1 Typy rozpoznawanych aktywności

W literaturze przeglądowej [87] zaprezentowano ogólną klasyfikację typów aktywności, odnoszącej się do sześciu grup: gestów, aktywności jednostek, zachowania, interakcji, aktywności grupy oraz zdarzenia (zob. Rys. 2.14).



Rys. 2.14 Ogólna klasyfikacja typów aktywności człowieka [87]

Gesty [87] to ruchy pojedynczych części ciała, które są często stosowane w celu podkreślenia ekspresji komunikacji niewerbalnej (np. gestykulacja, potrząsanie głową).

Aktywności jednostek [87] odnoszą się do zbiorowego ruchu różnych części ciała pojedynczej osoby (np. bieg, upadek czy zmiana pozycji).

Zachowania [87] definiowane są jako połączenie gestów i aktywności. Przykładem może być zachowanie agresywne, obserwowane jako połączenie chaotycznej gestykulacji z przyśpieszonym chodem.

Interakcje [87] definiowane są jako zachowania, które mogą wystąpić w relacji osoba–osoba (np. taniec, przywitanie się) lub osoba–obiekt (np. gotowanie, jazda na rowerze).

Aktywności grupy [87] definiowane są jako działania wielu osób i mogą obejmować zbiory gestów, akcji oraz interakcji (np. mecz piłki nożnej).

Zdarzenia [87] definiowane są jako podsumowanie zależności przyczynowo-skutkowej zaobserwowanych zbiorów gestów, akcji i interakcji (np. włamanie, kradzież przedmiotu).

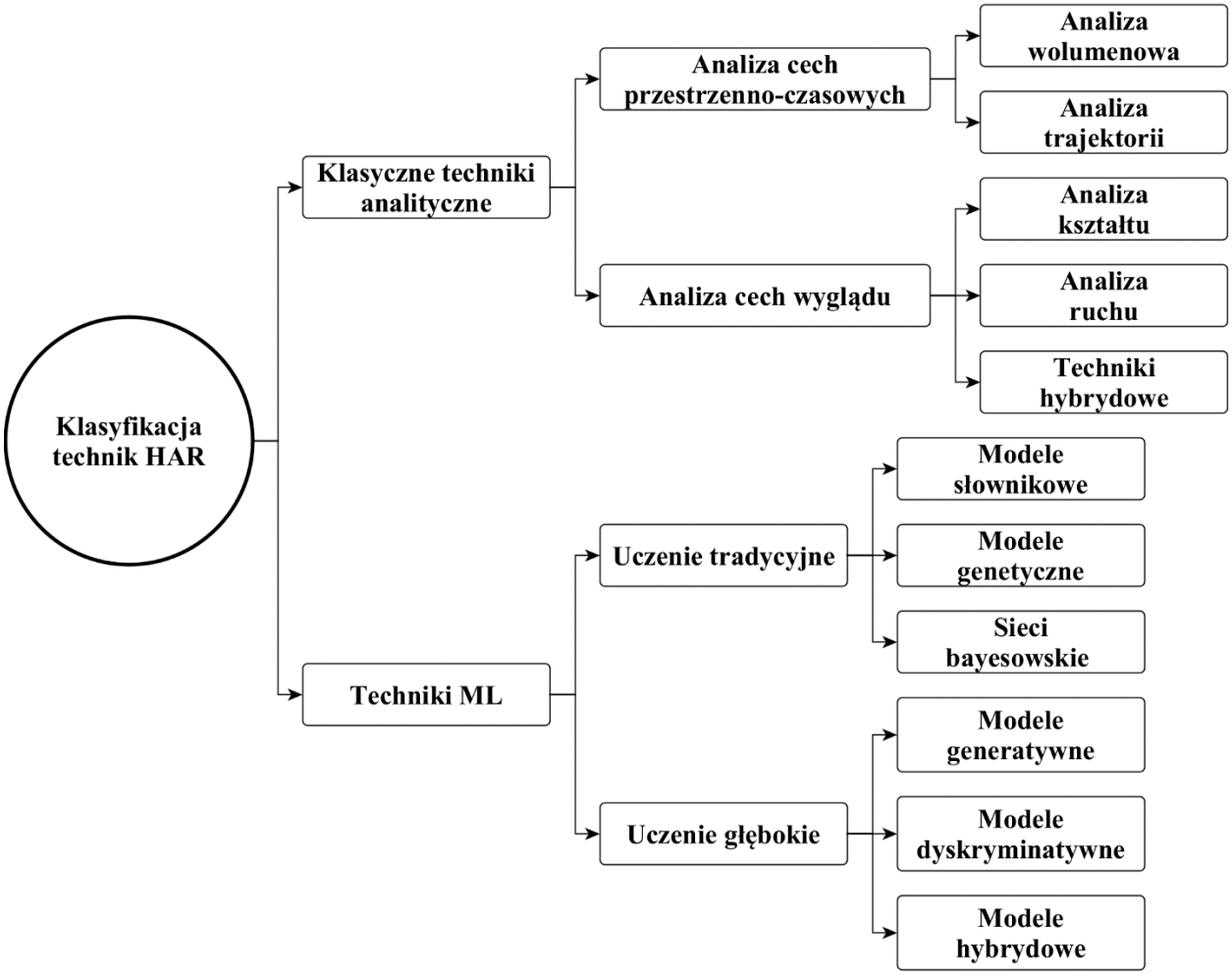
2.8.2 Dynamika aktywności

Klasyfikacja dynamiki aktywności człowieka obejmuje dwie grupy [88]: aktywności statyczne i dynamiczne. Do opisu aktywności statycznych korzysta się z analizy przestrzennej modelu ludzkiego ciała, a więc orientacji i lokalizacji kończyn w przestrzeni. Aktywności

dynamiczne są opisywane jako sekwencje aktywności statycznych w funkcji czasu. Stąd rozpoznawanie aktywności dokonuje się w oparciu zarówno o dane przestrzenne (orientacja, położenie), jak i czasowe (zmienność analizowanej aktywności w czasie).

2.8.3 Klasyfikacja technik HAR

W pracy przeglądowej [88] przeprowadzono ogólną klasyfikację technik HAR, rozróżniając dwie główne grupy: klasyczne techniki analityczne i techniki oparte na uczeniu maszynowym. Klasyfikacja technik HAR w ujęciu ogólnym została przedstawiona na Rys.2.15.



Rys. 2.15 Ogólna klasyfikacja technik HAR [88]

Klasyczne techniki analityczne

Klasyczne techniki analityczne [88] koncentrują się na możliwości opracowania funkcji decyzyjnych w oparciu o intuicję, pomysłowość i wiedzę ekspercką z zakresu biomechaniki oraz psychologii behawioralnej. Techniki te są efektywne obliczeniowo, uniwersalne w zastosowaniu i nie wymagają dużych zestawów danych uczących. Wyróżniamy tu podejścia oparte na analizie przestrzenno-czasowej i analizie wyglądu obiektu.

Wśród technik przestrzenno-czasowych [88] wyróżnia się analizę wolumenową i trajektorii punktów. W analizie wolumenowej aktywności rozpoznawane są poprzez wyznaczanie podobieństwa między dwoma objętościami, definiowanymi przy pomocy cech, takich jak tekstura, kolor lub HOG. Analiza wolumenowa umożliwia osiągnięcie pożądanej efektywności w przypadku klasyfikacji prostych aktywności lub gestów na niezatłoczonej scenie. W analizie trajektorii, model ciała reprezentowany jest za pomocą zbioru punktów charakterystycznych. Śledzenie zmian lokalizacji tych punktów w czasie umożliwia wyznaczenie trajektorii stanowiących podstawę klasyfikacji aktywności. Analiza trajektorii umożliwia uzyskanie pożądanej efektywności, niezależnie od warunków oświetleniowych i orientacji sylwetki względem kamery, co sprawia, iż jest często używana w rozpoznawaniu skomplikowanych aktywności.

Wśród technik opartych na wyglądzie [88], wyróżnia się analizę kształtu (punktów konturu, obwiedni i innych cech geometrycznych sylwetki), ruchu (np. histogramu przepływu optycznego i analizy historii ruchu) lub analizę hybrydową stanowiącą syntezę obu tych metod.

Techniki ML

Techniki ML [88] umożliwiają rozpoznawanie i klasyfikowanie aktywności przy użyciu modeli wytrenowanych w procesie uczenia tradycyjnego lub głębokiego. W odróżnieniu od klasycznych technik analitycznych, techniki ML nie wymagają opracowania skomplikowanych detektorów cech umożliwiając przeprowadzenie analizy jednoetapowej.

Wśród modeli uczenia tradycyjnego [88] wyróżnia się głównie modele słownikowe, genetyczne i sieć bayesowską. Modele słownikowe są generowane w procesie uczenia nienadzorowanego. Metoda ta polega na opracowaniu reprezentacji charakterystycznych cech ruchu poprzez liniową kombinację atomów, czyli elementów słownika. Iteracyjna optymalizacja kodowania cech, realizowana w procesie uczenia, ma na celu minimalizację błędu rekonstrukcji danych wejściowych, co przekłada się na możliwość zwiększenia wskaźnika dokładności w dalszych zadaniach związanych z klasyfikacją aktywności. Modele genetyczne należą do grupy metod ewolucyjnych, inspirowanych podstawowym mechanizmem procesu ewolucji biologicznej. W początkowym etapie model inicjowany jest z losową populacją rozwiązań, które w kolejnych etapach poddawane są procesom ewaluacji i selekcji na podstawie ich efektywności w klasyfikowaniu aktywności. Operacje genetyczne, takie jak krzyżowanie i mutacja, służą nie tylko do generowania nowych pokoleń rozwiązań, ale również do eksploracji przestrzeni rozwiązań, w celu identyfikacji i promowania rozwiązań zbliżających się do globalnego maksimum funkcji przystosowania. Sieć bayesowska używa graficznych modeli probabilistycznych w celu reprezentacji cech aktywności i ich zależności warunkowych za pomocą skierowanego grafu acyklicznego. Proces tworzenia modelu rozpoczyna się od zdefiniowania struktury sieci,

odzwierciedlającej zrozumienie relacji przyczynowo-skutkowych między czynnikami wpływającymi na daną aktywność oraz od określenia rozkładów prawdopodobieństwa dla każdej cechy. Model stosuje wnioskowanie bayesowskie, pozwalające na aktualizację prawdopodobieństw, umożliwiając efektywną klasyfikację aktywności.

Wśród modeli uczenia głębokiego [88] wyróżnia się głównie modele generatywne, dyskryminacyjne i hybrydowe. Modele generatywne powstają w procesie uczenia nienadzorowanego i stosują zasadę inspirowaną słynnym cytatem Richarda Feynmana: „*Czego nie potrafię stworzyć, tego nie rozumiem*”. Głównym celem modeli generatywnych jest zrozumienie rozkładu danych wejściowych, w tym identyfikacja cech charakterystycznych dla każdej klasy, aby umożliwić replikację oryginalnego rozkładu danych treningowych. Do najbardziej znanych metod generatywnych należą autoenkodery, autoenkodery wariacyjne (VAE) i generatywne sieci przeciwstawne (GAN). Metody dyskryminatywne powstają w procesie uczenia nadzorowanego i używają hierarchicznego modelu uczenia, składającego się z różnych ukrytych warstw sieci, przeznaczonych do klasyfikacji danych wejściowych na różne kategorie wyjściowe. Do najbardziej znanych architektur modeli generatywnych należą sieci DNN, RNN oraz CNN. Modele hybrydowe łączą cechy modeli generatywnych i dyskryminatywnych w celu osiągnięcia korzyści obu tych podejść.

2.8.4 Podsumowanie

Jedno z głównych założeń niniejszej rozprawy dotyczy opracowania systemu IVSS, który mógłby być wdrożony w istniejących instalacjach monitoringowych. Takie założenie z góry narzuca konieczność adaptacji systemu do sposobu rozmieszczenia kamer, a więc wysokości ich montażu, orientacji przestrzennej oraz pola widzenia. Większość kamer monitoringu montuje się na wysokościach powyżej przeciętnej wysokość człowieka. Ponadto, w celu ograniczenia pojemności magazynu danych, konwencjonalne systemy monitoringowe mogą być skonfigurowane w sposób ograniczający rozdzielczości i częstotliwość rejestrowanych obrazów. Ograniczenia te utrudniają lub wręcz uniemożliwiają analizę aktywności na podstawie niewielkich detali (rozpoznawanie gestów, mimika twarzy). W związku z tym autor postanawia opracować rozpoznawanie aktywności na podstawie całego modelu kinematycznego człowieka, generowanego przy użyciu wybranego detektora OpenPose (zob. rozdział 2.6.4).

Modele kinematyczne generowane przy użyciu detektora OpenPose umożliwiają wyznaczenie lokalizacji charakterystycznych punktów układu szkieletowego człowieka w odniesieniu do układu współrzędnych obrazu 2D. W celu realizacji analizy aktywności w odniesieniu do rzeczywistego układu współrzędnych 3D, autor opracuje dodatkowy algorytm, zwany dalej algorytmem kalibracji sceny. Algorytm ten będzie przeznaczony do generowania macierzy projekcji 3D, której dane umożliwią rzutowanie każdego piksela obrazu do

współrzędnych rzeczywistych podłoża obserwowanej sceny. Dzięki temu, projektowany system zyska zdolność automatycznego określania prędkości oraz dystansu przemieszczania wszystkich zidentyfikowanych obiektów w polu widzenia kamery.

Opracowane przez autora rozprawy algorytmy rozpoznawania aktywności, mają obejmować detekcję aktywności o charakterze zarówno statycznym jak i dynamicznym. W ujęciu ogólnej klasyfikacji typów aktywności (zob. Rys. 2.14), projektowany system umożliwi klasyfikowanie: aktywności jednostek, interakcji, aktywności grupy oraz zdarzeń. Szerokie spektrum klasyfikowanych typów aktywności zapewni uniwersalność projektowanego systemu IVSS.

W ujęciu ogólnej klasyfikacji technik HAR (zob. Rys. 2.15), projektowany algorytm zostanie zrealizowany głównie przy użyciu klasycznych technik analitycznych w podejściu stanowiącym syntezę analizy przestrzenno-czasowej (opartej na trajektorii) oraz analizy cech wyglądu (opartej na wzorcach ruchu). W tym przypadku rezygnacja z metod opartych na uczeniu maszynowym jest zasadna, ponieważ proces klasyfikacji rozpatrywanych typów aktywności może zostać opisany przy użyciu nieskomplikowanych zestawów reguł i funkcji decyzyjnych. Ponadto, przyjęte podejście nie wymaga stosowania dużych zbiorów danych uczących.

2.9 Sprzętowe jednostki obliczeniowe

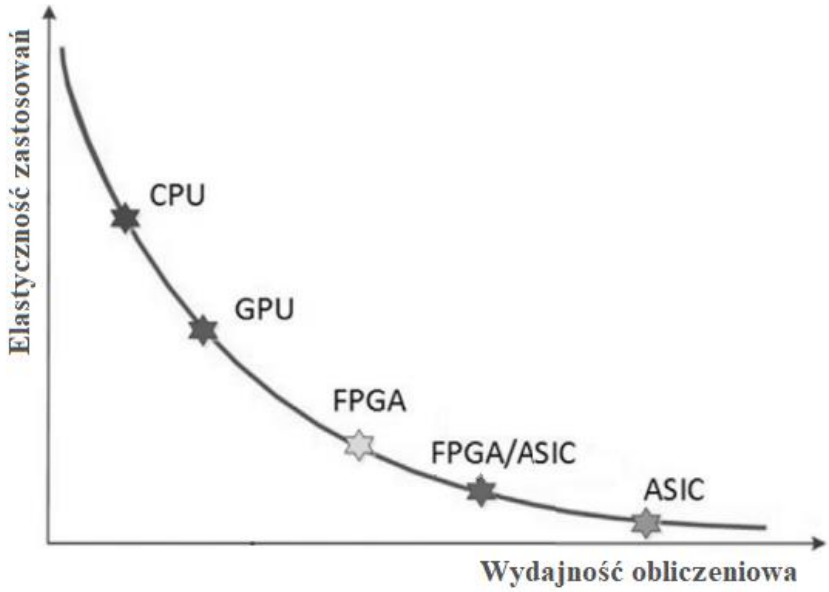
W większości przypadków proces uczenia maszynowego, jak i również uruchomienie opracowanych modeli ML, wymaga użycia jednostek sprzętowych o wysokiej wydajności obliczeniowej. Jest to szczególnie istotne, gdy projektowane modele docelowo mają analizować dane wejściowe o dużej nośności informacji w czasie rzeczywistym.

Obecnie, najczęściej stosowanymi rozwiązaniami sprzętowymi w kontekście ML są układy CPU, GPU oraz TPU [89]. CPU jest podstawową jednostką logiczną, stosowaną do realizacji większości ogólnych zadań obliczeniowych. Charakteryzuje się wysoką elastycznością, umożliwiającą wykonywanie różnorodnych zadań, lecz zawiera stosunkowo niewielką liczbę rdzeni, co może ograniczać wydajność w przypadku zadań o dużym zapotrzebowaniu obliczeniowym. GPU to jednostka pierwotnie dedykowana do przetwarzania grafiki komputerowej. Posiada stosunkowo dużą liczbę rdzeni, co pozwala na równoległe przetwarzanie wielu operacji. Wydajnie obsługuje obliczenia macierzowe, często spotykane w algorytmach ML. Układy TPU to specjalnie zaprojektowane jednostki dla zadań związanych z ML, zwłaszcza do operacji na tensorach, oferujące bardzo wysoką wydajność przy operacjach macierzowych.

Wśród innych rozwiązań sprzętowych w tym obszarze, można również wskazać układy FPGA [90], które zawierają reprogramowalne bloki logiczne i połączenia. Elastyczna konfiguracja tych układów umożliwia akcelerację niektórych etapów ML poprzez sprzętową optymalizację operacji. Układy ASIC [91] to kolejna grupa rozwiązań sprzętowych oferująca układy specjalnie zaprojektowane i zoptymalizowane pod kątem konkretnych zastosowań. Ze względu na

stosunkowo wysoki koszt ich produkcji stosowane są głównie w systemach, gdzie wymagane są bardzo wysokie prędkości obliczeniowe i niski pobór mocy.

Na Rys. 2.16 przedstawiono wykres zależności elastyczności spektrum zastosowań od wydajności obliczeniowej w przypadku różnych implementacji sprzętowych.



Rys. 2.16 Zależność elastyczności od wydajności obliczeniowej dla różnych jednostek obliczeniowych [3]

Analiza tego wykresu pozwala stwierdzić, iż wybór jednostki obliczeniowej zależy głównie od ustalenia pożądanego optimum elastyczności i wydajności w korelacji z ostatecznym przeznaczeniem opracowywanego systemu, złożoności realizowanych funkcji oraz uwarunkowań ekonomicznych.

W celu opracowania i przetestowania projektowanego systemu IVSS, autor postanowił użyć jednostkę GPU – NVIDIA RTX 3080 z kilku kluczowych powodów. Przede wszystkim, była to najnowsza komercyjna karta graficzna, dostępna w momencie rozpoczęcia prac badawczych. Dodatkowo, NVIDIA RTX 3080 oferuje atrakcyjny stosunek ceny do mocy obliczeniowej, co przełożyło się na optymalizację kosztów przy zapewnieniu pożądanej wydajności systemu. Ponadto, wybrana jednostka GPU jest w pełni kompatybilna z wybranymi bibliotekami oraz modelami ML, takimi jak OpenCV, YOLO v4, OpenPose oraz Tesseract OCR, umożliwiając ich akcelerację obliczeniową.

2.10 Wskaźniki oceny jakości

W dziedzinie CV przyjęto stosowanie wskaźników oceny jakości, które pozwalają na kompleksową i zunifikowaną ocenę wydajności i efektywności opracowanych algorytmów analizy i przetwarzania obrazów. Wśród podstawowych wskaźników wyróżniamy [92]: *TP* oznaczające poprawne wykrycie obiektu, *FP* oznaczające wykrycie obiektu w nieprawidłowym miejscu, *FN* oznaczające brak wykrycia obiektu, mimo że istnieje na obrazie, *TN* oznaczające prawidłowe wykrycie braku obiektu (pojęcie rzadko stosowane w kontekście detekcji obiektów),

P oznaczające całkowitą liczbę rzeczywistych pozytywów (np. całkowita liczba obiektów do wykrycia na obrazie), *N* oznaczające całkowitą liczbę rzeczywistych negatywów (np. całkowita liczba obszarów obrazu, w których nie powinno być wykrytych obiektów) oraz *T* oznaczające całkowitą liczbę wszystkich pomiarów.

Do oceny algorytmów pod kątem szybkości przetwarzania danych wizyjnych stosuje się wskaźnik *FPS* [19], informujący o liczbie przetwarzanych ujęć/obrazów w czasie jednej sekundy. W dziedzinie CV dąży się do osiągnięcia najwyższej możliwej wartości *FPS*, w celu uzyskania przetwarzania obrazu w czasie rzeczywistym, przy jednoczesnej minimalizacji wymaganych zasobów obliczeniowych.

Dokładność (*A*) [92]

$$A = \frac{TP + TN}{T} = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

oznacza stosunek liczby poprawnych detekcji (zarówno pozytywnych, jak i negatywnych) do całkowitej liczby pomiarów. Zawiera się w przedziale od 0 do 1, przy czym im wyższa wartość, tym lepsza skuteczność modelu w realizacji zadania.

Precyzja (*Pr*), określana również jako wartość predykcyjna dodatnia (*PPV*) [92]

$$Pr = \frac{TP}{TP + FP} \tag{2.2}$$

oznacza stosunek liczby poprawnych pozytywnych detekcji do całkowitej liczby pozytywnych detekcji, dokonanych przez model (zarówno poprawnych, jak i błędnych). Precyzja koncentruje się na tym, jak duża część detekcji pozytywnych jest prawdziwie pozytywna. Zawiera się w przedziale od 0 do 1, przy czym im wyższa wartość, tym lepsza skuteczność modelu w realizacji zadania.

Czułość, określana również jako wskaźnik recall (*R*), wskaźnik prawdziwych detekcji (*TPR*) lub wskaźnik trafień [92]

$$R = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{2.3}$$

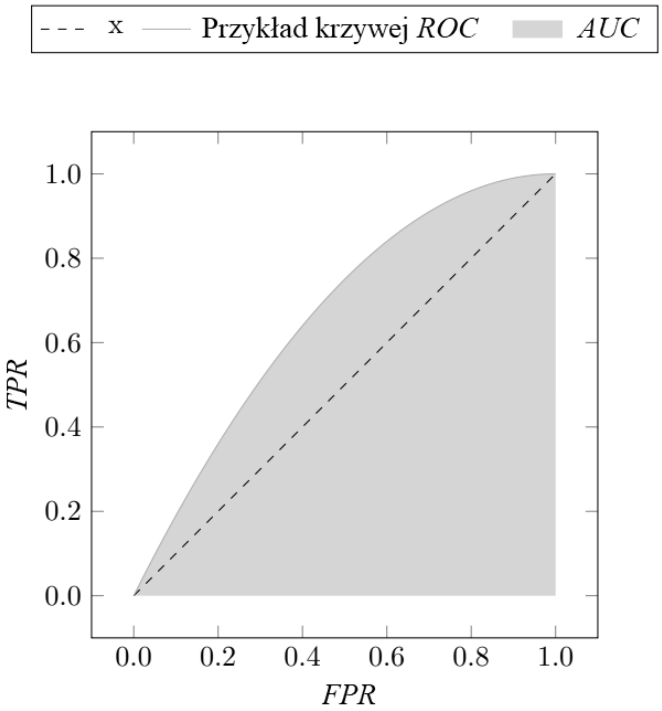
oznacza stosunek liczby poprawnych pozytywnych detekcji do całkowitej liczby rzeczywistych pozytywów. Wskaźnik czułości jest szczególnie ważny w aplikacjach, gdzie pominięcie pozytywnego przypadku (tj. niezidentyfikowanie istotnego obiektu lub cechy) może mieć poważne konsekwencje. Zawiera się w przedziale od 0 do 1, przy czym im wyższa wartość, tym lepsza skuteczność modelu w realizacji zadania.

Wskaźnik negatywnych detekcji (*FPR*) [92]

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{2.4}$$

oznacza stosunek liczby detekcji negatywnych, które zostały błędnie zaklasyfikowane jako pozytywne, do całkowitej liczby rzeczywiście negatywnych detekcji. Wskaźnik *FPR* informuje o generowaniu fałszywych alarmów przez model. Na przykład, w systemach IVSS, niska wartość *FPR* oznacza, że system rzadko zgłasza fałszywe alarmy, poprawnie ignorując elementy, które nie stanowią zagrożenia. Zawiera się w przedziale od 0 do 1, przy czym im wyższa wartość, tym lepsza skuteczność modelu w realizacji zadania.

Wykres zależności *TPR* od *FPR* przedstawia krzywą *ROC*, której przykład przedstawiono na Rys. 2.17.



Rys. 2.17 Krzywa *ROC* oraz obszar *AUC* [92]

Krzywa *ROC* umożliwia wyznaczanie wskaźnika *AUC*, wartości progu decyzyjnego klasyfikacji oraz porównanie różnych modeli klasyfikatorów. Wskaźnik *AUC*

$$AUC = \int_{x=0}^{1} TPR\left(FPR^{-1}(x)\right)dx = \frac{1}{n} \cdot \sum_{i=1}^{n} TPR(FPR_i) \tag{2.5}$$

odnosi się do pola powierzchni pod krzywą *ROC*, gdzie *n* oznacza liczbę klas. Wartość *AUC* zawiera się w przedziale wartości od 0 do 1, przy czym im wyższa wartość, tym lepsza zdolność modelu do klasyfikacji, natomiast wartość 0.5 oznacza brak zdolności dyskryminacyjnej, równoważną losowemu zgadywaniu. Wartość *AUC* poniżej 0.5 sugeruje, że model działa gorzej

niż losowe zgadywanie, co może być wynikiem niewłaściwej implementacji lub potrzeby dalszej optymalizacji modelu. Wskaźnik *AUC* jest szczególnie istotny w sytuacjach, gdy zbiory klas są nierównomierne, tj. gdy jedna klasa jest znacznie liczniejsza od drugiej, co jest częste w wielu rzeczywistych problemach klasyfikacyjnych.

Średnia precyzja (*AP*) [92]

$$AP = \int_{x=0}^{1} Pr\left(R^{-1}(x)\right)dx = \frac{1}{n} \cdot \sum_{i=1}^{n} Pr(R_i) \tag{2.6}$$

jest średnią wartością precyzji (*Pr*) przy różnych poziomach czułości (*R*), uzyskaną poprzez obliczenie obszaru pod krzywą precyzja–czułość, gdzie *n* oznacza liczbę klas. Wskaźnik *AP* informuje, w jakim stopniu model jest w stanie łączyć precyzję (jak dobrze model unika fałszywych alarmów) i czułość (jak dobrze model wykrywa wszystkie prawdziwe pozytywy). Wskaźnik *AP* (w kontekście detekcji obiektów) obliczany jest przy uwzględnieniu różnych progów wskaźnika *IoU*, zwanego również indeksem Jaccarda [92]

$$IoU = \left|\frac{A_r \cap B_p}{A_r \cup B_p}\right| \tag{2.7}$$

który mierzy stopień nakładania się obszaru BB generowanego w procesie predykcji (B_p) i rzeczywistego obszaru lokalizacji obiektu na obrazie (A_r). Rozszerzeniem wskaźnika *AP* jest wskaźnik *mAP* [92]

$$mAP = \frac{1}{n} \cdot \sum_{i=1}^{n} AP_i \tag{2.8}$$

będący średnią wartością *AP* dla wszystkich klas obiektów rozpoznawanych przez model. Wskaźnik *mAP* obliczany jest często przy *IoU* = 0.5 (*mAP*@0.5), ale również przy użyciu średniej z różnych progów *IoU* (np. *mAP*@0.5:0.95), co daje bardziej kompleksową ocenę zdolności modelu do dokładnego lokalizowania i klasyfikowania obiektów. Wskaźniki *AP*, *IoU* oraz *mAP* zawierają się w przedziale wartości od 0 do 1, przy czym im wyższa wartość, tym lepsza zdolność modelu do realizacji zadania.

Wskaźnik F_1 [92]

$$F_1 = 2 \cdot \frac{Pr \cdot R}{Pr + R} \tag{2.9}$$

jest średnią harmoniczną z dwóch wartości *Pr* oraz *R*, co umożliwia równomierne uwzględnienie obu tych wartości i jest szczególnie pożądane w przypadkach znacznej dysproporcji rozkładu klas.

Zawiera się w przedziale wartości od 0 do 1, przy czym im wyższa wartość, tym lepsza zdolność modelu do realizacji zadania.

2.11 Podsumowanie

W celu realizacji zadań związanych ze wstępnym przetwarzaniem obrazu, wybrano bibliotekę OpenCV. Wybór ten motywowany jest rozbudowanym zestawem dostępnych funkcji, wsparciem dla DL, akceleracją obliczeń realizowanych przy użyciu GPU, możliwością implementacji w wybranym języku C++, darmową licencją BSD oraz wsparciem społeczności użytkowników.

W celu realizacji testów systemu, autor postanowił użyć kamerę wieżyczkową TP-Link Tapo C500 wraz z mobilnym statywem studyjnym. Zestaw ten umożliwi bezinwazyjne wykonanie nagrań wideo w pożądanej jakości, niezależnie od wybranej lokalizacji.

Kalibracja kamery zostanie przeprowadzona z użyciem funkcji dostępnych w bibliotece OpenCV. Wybór ten jest zasadny, ponieważ zwalnia z konieczności stosowania dodatkowych bibliotek lub algorytmów, jednocześnie umożliwiając przeprowadzenie procesu kalibracji w sposób precyzyjny i łatwy do wykonania.

Detekcja i klasyfikacja obiektów na obrazach zostanie zrealizowana z użyciem detektora YOLO v4. Wybór ten motywowany jest dostępnością szeregu modeli wytrenowanych na popularnym zbiorze danych MS-COCO, co umożliwia detekcję i klasyfikację osiemdziesięciu klas typowych obiektów, przy zastosowaniu adnotacji typu BB. Ponadto, autorzy detektora YOLO v4 udostępniają skrypty programowe, umożliwiające przeprowadzenie uczenia maszynowego, zorientowanego na własny zestaw danych treningowych, a wytrenowane modele są możliwe do implementacji w wybranym języku C++.

W procesie uczenia maszynowego modelu detektora YOLO v4 tiny, autor dokona syntezy gotowych baz danych, zawierających próbki pożądanych klas obiektów, które zostały zaczerpnięte z kilku platform internetowych, takich jak: Roboflow, ImageNet oraz Keegle. Uzasadnieniem wyboru tych źródeł jest publiczna dostępność dużej liczby próbek dla specyficznych klas obiektów wraz z adnotacjami.

W celu realizacji zadań związanych z ANPR, autor postanowił użyć silnik Tesseract OCR. Uzasadnienie tego wyboru motywowane jest głównie wysoką skutecznością opisywaną w pracach przeglądowych, możliwością implementacji w wybranym języku C++, dostępnością bogatej funkcjonalności oraz ciągłym rozwojem technologii przez twórców i szerokie grono użytkowników. Wejście silnika będą stanowić fragmenty obrazu, oznaczone przez detektor YOLO v4 tiny jako tablice rejestracyjne.

Estymacja sylwetek osób na obrazie zostanie zrealizowana przy użyciu detektora OpenPose. Wybór ten uzasadnia wysoka wydajność przetwarzania wynikającą z typu podejścia Bottom-Up oraz implementacja programowa biblioteki w wybranym języku C++. Ponadto, autorzy biblioteki

OpenPose udostępniają publicznie pełen kod źródłowy, pełną dokumentację programistyczną dostępną online, wytrenowane modele, jak również oferują wsparcie dla użytkowników poprzez forum internetowe.

Algorytm śledzenia obiektów zostanie opracowany przy użyciu klasycznych technik analitycznych, opartych na analizie punktu. Uzasadnieniem tego wyboru jest chęć ograniczenia konsumpcyjności obliczeniowej projektowanego systemu, związanej z użyciem zbyt wielu modeli ML. Proponowany algorytm śledzenia MOT będzie stanowił syntezę analizy deterministycznej (wyznaczanie minimalnej odległości pomiędzy BB na kolejnych ujęciach) i stochastycznej (filtr Kalmana generujący predykcję lokalizacji obiektu na kolejnym ujęciu).

W celu realizacji zadań związanych z klasyfikacją aktywności człowieka na obrazie, autor opracuje algorytmy, bazujące na klasycznych technikach analitycznych, w podejściu stanowiącym syntezę analizy przestrzenno-czasowej (opartej na trajektorii) oraz analizy cech wyglądu (opartej na wzorcach ruchu). Jako wzorzec sylwetki użyty zostanie model kinematyczny osoby generowany przy pomocy detektora OpenPose. W tym przypadku rezygnacja z metod opartych na uczeniu maszynowym jest zasadna, ponieważ proces klasyfikacji rozpatrywanych typów aktywności może zostać opisany przy użyciu nieskomplikowanych zestawów reguł i funkcji decyzyjnych. Ponadto, przyjęte podejście nie wymaga stosowania dużych zbiorów danych uczących.

Estymacja parametrów ruchu obiektów z obrazu 2D do przestrzeni 3D zostanie zrealizowana przy użyciu dedykowanego algorytmu kalibracji sceny. Algorytm ten będzie przeznaczony do generowania macierzy projekcji 3D, której dane umożliwią rzutowanie pikseli obrazu do współrzędnych rzeczywistych podłoża obserwowanej sceny. Dzięki temu, projektowany system zyska zdolność automatycznego określania prędkości oraz dystansu przemieszczania wszystkich zidentyfikowanych obiektów w polu widzenia kamery.

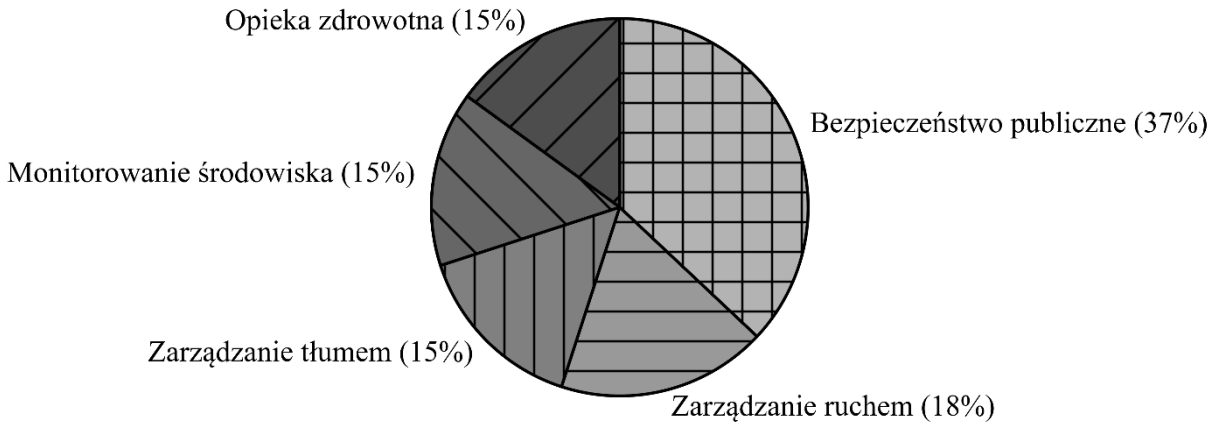
W celu opracowania i przetestowania projektowanego systemu IVSS, autor postanowił użyć jednostkę GPU – NVIDIA RTX 3080, której wybór uzasadnia atrakcyjny stosunek ceny do mocy obliczeniowej, co przekłada się na optymalizację kosztów przy zapewnieniu pożądanej wydajności systemu.

3 Inteligentne systemy nadzoru wizyjnego

Inteligentne systemy nadzoru wizyjnego (IVSS) [93] to kompleksowe rozwiązania technologiczne, umożliwiające realizację analizy i oceny informacji wizyjnej przy użyciu zaawansowanych technik automatycznego przetwarzania obrazów. Głównym celem tych systemów jest zapewnienie możliwości skutecznego nadzoru i zarządzania obszarami objętymi monitoringiem w celu zwiększenia bezpieczeństwa ludzi oraz ochrony mienia. Pomimo znaczącego postępu w dziedzinie CV, istniejące badania [16] wskazują, iż opracowanie precyzyjnego i efektywnego systemu IVSS jest przedsięwzięciem bardzo złożonym. Głównym problemem jest wieloaspektowość ludzkich zachowań, która zależy od takich czynników jak: kontekst sytuacji, czas trwania, miejsce zdarzenia czy liczba osób i ich interakcja z obiektami znajdującymi się na scenie. Inne trudności związane są ze złożonością sceny oraz dynamiką jej zmian w czasie. Należy tu wskazać przede wszystkim na warunki środowiskowe (np. oświetlenie, opady atmosferyczne, mgła), które mają negatywny wpływ na skuteczność analizy. Dodatkowe wyzwania obejmują występowanie zakłóceń, okluzji, a także ograniczenia sprzętowe, wynikające z parametrów kamer oraz urządzeń przetwarzających pozyskane dane wizyjne. Mnogość i złożoność opisanych problemów dowodzą potrzeby dalszego rozwoju w obszarze systemów IVSS.

3.1 Obszary zastosowań

Najczęstsze obszary zastosowań systemów IVSS [93] zostały przedstawione na Rys. 3.1.



Rys. 3.1 Obszary najczęstszych zastosowań systemów IVSS [93]

Sektor bezpieczeństwa publicznego [93] stanowi największy udział (37%) w ogólnej liczbie obszarów zastosowań. Dotyczy on głównie realizacji zadań związanych z nadzorem przestrzeni (chodników, parków, węzłów transportowych) oraz obiektów (szkół, banków, galerii handlowych) użytku publicznego. Celem jest analiza zachowań obywateli umożliwiająca identyfikację działań kryminalnych, wandalizmu lub innych form zakłócania porządku publicznego.

Sektor zarządzania ruchem [93] stanowi 18% udziału. Dotyczy on głównie realizacji zadań związanych z nadzorem dróg, skrzyżowań i przestrzeni parkingowych w celu detekcji kolizji, wypadków, zatorów komunikacyjnych oraz naruszeń przepisów drogowych.

Sektor zarządzania tłumem [93] stanowi 15% udziału. Dotyczy on głównie realizacji zadań związanych z nadzorem dużej liczby osób znajdujących się na monitorowanej scenie, umożliwiając detekcje zdarzeń takich jak zatory komunikacyjne czy przeludnienia.

Sektor monitorowania środowiska [93] stanowi 15% udziału. Dotyczy on głównie realizacji zadań związanych z analizą zmian środowiskowych w aspektach klęsk żywiołowych, takich jak powodzie czy pożary oraz ogólnopojętej meteorologii.

Sektor opieki zdrowotnej [93] stanowi 15% udziału. Dotyczy on głównie realizacji zadań związanych z nadzorem osób starszych, chorych i niepełnosprawnych w celu identyfikacji ich aktualnego stanu zdrowia (zdalne monitorowanie pacjentów) oraz detekcji zdarzeń potencjalnie niebezpiecznych (np. upadek czy leżenie w bezruchu).

3.2 Ogólna klasyfikacja systemów IVSS

W ogólnym ujęciu, klasyfikacje typów systemów IVSS można uprościć do [93]: systemów detekcji zdarzeń alarmowych oraz systemów detekcji aktywności anormalnych.

Systemy detekcji zdarzeń alarmowych [94] umożliwiają precyzyjną detekcję i klasyfikację tzw. zdarzeń krytycznych, zaobserwowanych na nagraniu wideo oraz aktywowanie przypisanej im procedury alarmowej. Istota działania sprowadza się do implementacji technik dyskryminacyjnych, które umożliwiają interpretację wzorców zdarzeń krytycznych, mogących stanowić przesłankę o wystąpieniu konkretnego zdarzenia z grupy rozpatrywanych zdarzeń alarmowych.

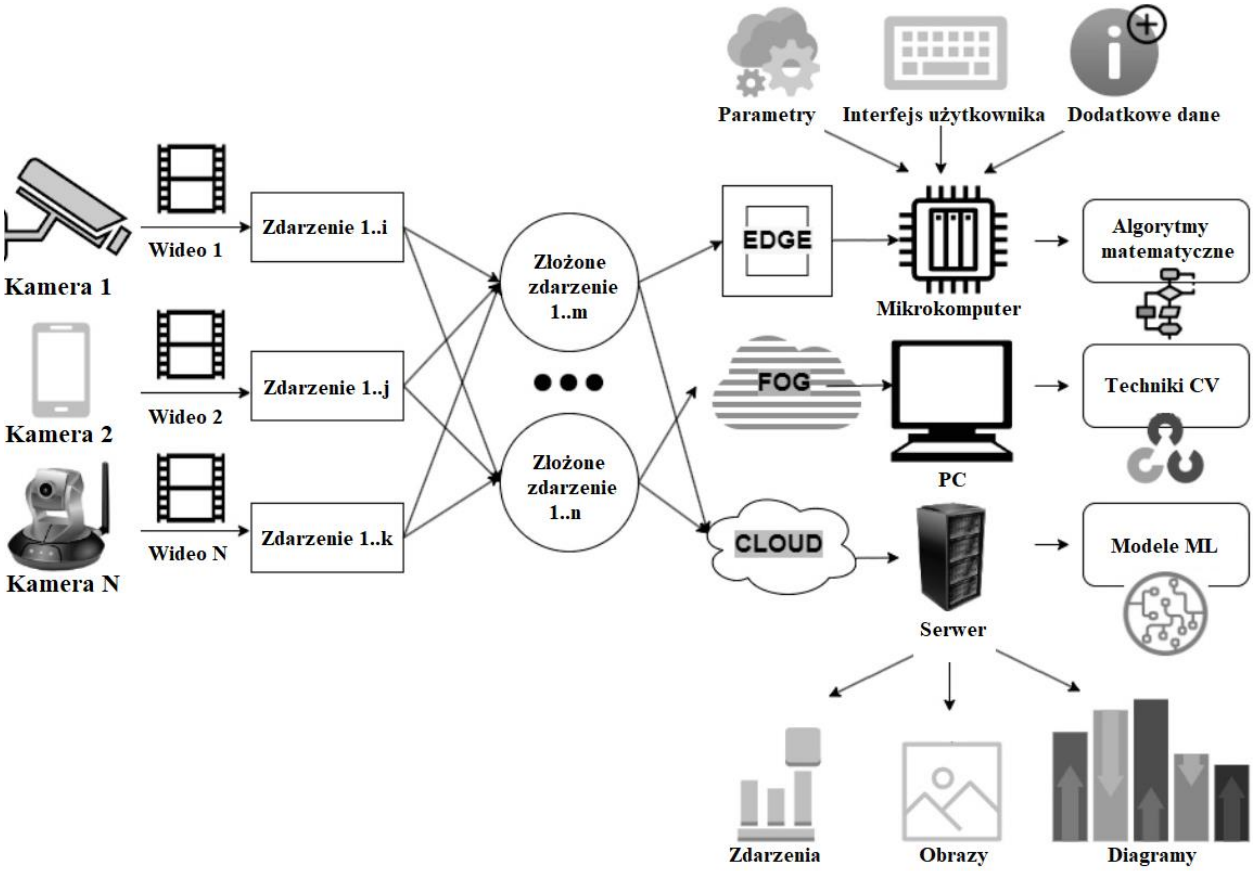
Systemy detekcji aktywności anormalnych [95], w przeciwieństwie do systemów opartych na detekcji zdarzeń, charakteryzuje brak możliwości klasyfikacji konkretnych typów zachowań. Działają na zasadzie klasyfikacji binarnej, wskazując na wystąpienie lub brak wystąpienia potencjalnych zachowań anormalnych w obszarze monitorowanej sceny. Istota działania sprowadza się do możliwości nauczenia systemu wzorców zachowań normalnych, niewzbudzających podejrzeń. Następnie, system realizuje proces analizy porównawczej, polegający na sprawdzaniu stopnia odchyleń pomiędzy zachowaniami obserwowanymi na aktualnej scenie, a zachowaniami wzorcowymi. Tak przygotowany system jest w stanie wykryć sytuacje wykraczające poza ustalone kryterium progowe, w celu aktywowania procedury alarmowej.

Z uwagi na przyjęte cele badawcze oraz przeznaczenie projektowanego systemu, autor postanawia ograniczyć rozpoznanie literaturowe tylko do grupy systemów IVSS, realizujących

detekcję zdarzeń alarmowych. Zatem, z uwagi na brak zastosowania w niniejszej rozprawie, systemy detekcji aktywności anormalnych nie będą szczegółowiej omawiane.

3.3 Architektury systemów IVSS

Systemy IVSS przeznaczone są do monitorowania przestrzeni zarówno prywatnych jak i publicznych przy użyciu różnego rodzaju sprzętu rejestrującego (kamery CCTV) oraz przetwarzającego sekwencje wideo [96]. Na Rys. 3.2 przedstawiono przykład modelu przepływu danych w systemach detekcji zdarzeń alarmowych.



Rys. 3.2 Przykład modelu przepływu danych w systemach IVSS [96]

W pierwszej kolejności sygnał z kamer monitoringu przesyłany jest w formie strumienia danych wideo. W drugim kroku dane ze źródła sygnału wizyjnego podlegają procesowi wstępnego przetwarzania, umożliwiając detekcję pożądanych cech obiektów. Przetworzone dane stanowią wejście dla algorytmów detekcji zdarzeń prostych, a następnie zdarzeń złożonych. W modelu występuje podział na trzy główne architektury systemowe: Cloud, Edge oraz Fog.

W systemach o architekturze typu Cloud [96] przetwarzanie i analiza danych odbywają się w centralnej chmurze obliczeniowej. Wśród zalet tego rozwiązania należy wskazać wysoką wydajność obliczeniową oraz wielkoskalowy magazyn pamięci, co jest korzystne przy realizacji uczenia maszynowego i przechowywania dużej ilości danych z różnych źródeł. Do wad tego rozwiązania należą: ograniczenie prywatności, związane z koniecznością udostępniania obrazu z

monitoringu serwerom zewnętrznym, obciążenie łącza sieciowego oraz opóźnienia komunikacyjne, zależne od stabilności i przepustowości usług sieciowych.

Dla porównania, w systemach o architekturze typu Edge [96] przetwarzanie danych odbywa się lokalnie, tzn. bezpośrednio w urządzeniach rejestrujących obraz lub w urządzeniach znajdujących się relatywnie blisko źródeł sygnału wizyjnego. Zaletami tego rozwiązania są skrócony czas odpowiedzi, mniejsze obciążenie sieci i większa prywatność, gdyż dane nie muszą być przesyłane przy użyciu łącza sieciowego. Główną wadą tego typu architektury jest niska wydajność obliczeniowa.

Dogodną alternatywą, stanowiącą kompromis pomiędzy systemami typu Cloud i Edge, są systemy o architekturze typu Fog [96]. W tym podejściu przetwarzanie danych jest rozszerzone o możliwość użycia infrastruktury sieciowej, lecz wciąż odbywa się blisko lokalizacji źródła sygnału wizyjnego. Kompromis pomiędzy lokalnym a centralnym przetwarzaniem pozwala na osiągnięcie elastyczności i skalowalności systemu. W przeciwieństwie do architektury typu Cloud, architektura typu Fog umożliwia zmniejszenie obciążenia łącza sieciowego oraz zachowanie prywatności poprzez utrzymanie lokalnej kontroli danych wizyjnych. Natomiast, w przeciwieństwie do architektury Edge, architektura typu Fog umożliwia realizację znacznie bardziej rozbudowanych modeli przetwarzania danych dzięki użyciu jednostek obliczeniowych takich jak GPU czy TPU.

3.4 Przegląd rozwiązań w obszarze IVSS

Istnieje wiele wariantów systemów monitoringu wideo, co umożliwia przeprowadzenie wielostopniowej kategoryzacji. Aby zatem osiągnąć wymaganą rzeczowość, autor rozprawy ograniczył zakres przeglądu istniejących rozwiązań, przyjmując kryteria wyboru dostosowane do docelowego przeznaczenia projektowanego systemu IVSS. Wybrane kryteria dotyczą wielozadaniowych systemów inteligentnego nadzoru wizyjnego, zdolnych do analizy danych w czasie rzeczywistym, w których źródłem sygnału wejściowego jest strumień kolorowego obrazu wideo, pozyskiwany przy użyciu pojedynczych, konwencjonalnych, statycznych kamer monitoringowych 2D.

3.4.1 Analiza stosowanych technik

Tab. 3.1 stanowi opracowanie własne autora i zawiera chronologiczny wykaz 24 prac badawczych, wybranych spośród 143 analizowanych w ramach szeroko zakrojonego przeglądu literatury. Prace te szczegółowo opisują implementacje referencyjnych systemów IVSS i zostały uznane za szczególnie istotne w kontekście niniejszej analizy ze względu na ich reprezentatywność oraz zgodność z przyjętymi kryteriami. Dane zawarte w Tab. 3.1 stanowią podstawę do przeprowadzenia analizy porównawczej, będącej podstawą do oceny proponowanych rozwiązań.

Tab. 3.1 Chronologiczne zestawienie porównawcze referencyjnych systemów IVSS

Źródło	**Data**	**Techniki przetwarzania obrazów użyte w celu klasyfikacji zdarzeń alarmowych**	**Algorytmy przestrzennej analizy sceny**	**Algorytmy MOT**	**Wskaźniki oceny jakości**	**Bazy danych**
[97]	1997	Analityczne techniki odejmowania tła, detekcja ruchomych krawędzi, analiza różnic w poziomach szarości, analiza tekstur oraz samoorganizująca się ANN	Brak implementacji	Brak implementacji	Brak informacji	Opracowane przez autorów
[98]	2004	Analityczne techniki segmentacji pierwszoplanowej na podstawie kontrastu luminancji, analiza lokalizacji i trajektorii obiektów pierwszoplanowych	Brak implementacji	Wyznaczanie najmniejszej odległości BB na kolejnych ujęciach	10 *FPS*	CAVIAR
[99]	2005	Kontynuacja pracy [97], analityczne techniki segmentacji pierwszoplanowej przy użyciu algorytmu pełnego dopasowania blokowego (identyfikacji pikseli tła przy użyciu mieszaniny rozkładów prawdopodobieństwa Gaussa)	Interpretacja zniekształceń perspektywicznych na podstawie prędkości przemieszczania się obiektów w różnych odległościach od kamery	Algorytm pełnego dopasowania blokowego w połączeniu z wyznaczaniem najmniejszej odległości BB na kolejnych ujęciach	Dokładność 72%	Opracowane przez autorów
[100]	2005	Analityczne techniki detekcji ruchomych elementów tła przy użyciu pikselowego rozkładu Gaussa, detekcja zdarzeń przy użyciu technik AI (brak szczegółowych informacji)	Brak implementacji	Metoda porównywania trójwymiarowych histogramów barw wykrytych obiektów	Brak informacji	PETS, CREDS oraz opracowane przez autorów

Źródło	Data	Techniki przetwarzania obrazów użyte w celu klasyfikacji zdarzeń alarmowych	Algorytmy przestrzennej analizy sceny	Algorytmy MOT	Wskaźniki oceny jakości	Bazy danych
[101]	2009	Analityczne techniki odejmowania tła na podstawie mieszaniny rozkładów prawdopodobieństwa Gaussa, analiza trajektorii, różnicowanie histogramów kolorów	Brak implementacji	Online appearance model [102] w połączeniu z techniką kwantyzacji kolorów (Bi-Conic Color Quantization)	10 *FPS*	PETS 2001 oraz opracowane przez autorów
[103]	2012	Analityczne techniki odejmowania tła, detekcja zdarzeń oparta na deskryptorach semantycznych wyznaczanych na podstawie cech ruchu obiektów (szybkość, trajektoria, lokalizacja)	Zaimplementowano, lecz brak szczegółowych informacji	Zaimplementowano, lecz brak szczegółowych informacji	Precyzja 92%, czułość 70%, 5 *FPS*	BEHAVE, CAVIAR (PETS 2004) oraz PETS 2006
[104]	2014	Analityczne techniki odejmowania tła, klasyfikator MARTMAP (ang. *Modular Adaptive Resonance Theory Map*) [105], analiza cech wyglądu (tekstura, kolor, rozmiary)	Brak implementacji	Analiza ruchu kształtów obiektów w połączniu z filtrem cząsteczkowym	Dokładność 83.02%; czułość 87.82%, precyzja 92.79%, 11 *FPS*	PETS, CANTATA, UMN, i-LIDS oraz YouTube
[106]	2014	Analityczne techniki odejmowania tła, analiza trajektorii, klasyfikator JRoG (ang. *Joint Ranking of Granules Features*) [107]	Algorytm zaczerpnięty z pracy [108]	Analiza ruchu kształtów obiektów w połączniu z filtrem cząsteczkowym	Czułość 78%, precyzja 100%, 10 *FPS*	Opracowane przez autorów
[109]	2015	Analityczne techniki odejmowania tła, model SVM do wyznaczania różnic cech: GCH (Global Color Histogram), LBP (Local Binary Pattern), oraz HOG	Brak implementacji	Filtr Kalmana w połączniu z LSAP (ang. Linear Sum Assignment Problem)	Dokładność 91%,	CAVIAR oraz opracowane przez autorów

Źródło	Data	Techniki przetwarzania obrazów użyte w celu klasyfikacji zdarzeń alarmowych	Algorytmy przestrzennej analizy sceny	Algorytmy MOT	Wskaźniki oceny jakości	Bazy danych
[110]	2017	Brak implementacji algorytmów detekcji obiektów (autorzy bazują na gotowych detekcjach z baz danych), detekcja zdarzeń przy użyciu hybrydowego modelu IBSTM (ang. *Interval-Based Spatio-Temporal Model*) łączącego modele Threshold Model (przestrzenno-czasowa analiza ruchu), BoA (ang. *Bag of Actions*) (redukcja przestrzeni poszukiwań w fazie wykrywania) oraz MLN (ang. *Markov Logic Network*) (generowanie zrozumiałych i łatwych w zarządzaniu predykcji logicznych)	Brak implementacji (autorzy używają parametrów 3D sceny dostarczonych w bazach danych)	Analiza trajektorii obiektów z ostatnich 15 ujęć	Dokładność dla bazy CAVIAR: 88%, dla bazy BEHAVE: 91%	CAVIAR, BEHAVE oraz opracowane przez autorów
[111]	2018	Detekcja obiektów przy użyciu YOLO v2, estymacja sylwetek przy użyciu modelu CNN, klasyfikacja zdarzeń przy użyciu modelu LSTM	Brak implementacji	Filtr Kalmana	Czułość 82%, precyzja 84%	UT-Interaction-Data
[112]	2019	Analiza lokalizacji i trajektorii punktorów szkieletu OpenPose w obszarach ROI	Brak implementacji	Brak implementacji	Brak informacji	Opracowane przez autorów
[113]	2019	Detektor YOLO v1 w połączeniu z wielowarstwowym modelem reprezentacji wiedzy EDCAR (ang. *Elements and Descriptors of Context and Action Representations*)	Brak implementacji	Tracking-Learning-Detection [114]	Precyzja 98%, czułość 81%, dokładność 86%, F_1 88%	Behave, Caviar, UMN-Crowd, iLids, Pets2006, Pets2012 oraz UT-Interaction, UCF-Crime
[115]	2020	Analiza przepływu optycznego (szybkości i kierunku poruszania się obiektów)	Brak implementacji	Brak implementacji	Dokładność 93.89%, precyzja 96.21%, czułość 94.9%, *AUC* 90%	UMN, Avenue oraz YouTube

Źródło	Data	Techniki przetwarzania obrazów użyte w celu klasyfikacji zdarzeń alarmowych	Algorytmy przestrzennej analizy sceny	Algorytmy MOT	Wskaźniki oceny jakości	Bazy danych
[116]	2021	Ekstrakcja cech przy użyciu I3D (ang. *Inflated 3D Convolutional Network*), klasyfikacja zdarzeń przy użyciu sieci LSTM	Brak implementacji	Brak implementacji	Dokładność 59%, *AUC* 87%	Avenue, UCSD Ped2, ShanghaiTech, UCF-Crime oraz opracowana przez autorów LAD
[117]	2021	Ekstrakcja cech przy użyciu 3DConvNets, klasyfikacja zdarzeń przy użyciu w pełni połączonej sieci NN	Brak implementacji	Brak implementacji	Dokładność 45%, precyzja 45%, czułość 45%, F_1 45%, *AUC* 82%	UCF-Crime
[118]	2021	Ekstrakcja cech przy użyciu sieci CNN, następnie klasyfikacja zdarzeń przy użyciu sieci LSTM	Brak implementacji	Brak implementacji	Dokładność/czułość/ precyzja/F_1/*AUC* dla bazy UCF-Crime 78/78/87/81/96%, dla bazy UMN 98/98/98/98/98%, dla bazy Avenue 99/98/99/99/98%, 114 *FPS*	UMN, Avenue oraz UCF-Crime
[119]	2022	Ekstrakcja cech przy użyciu sieci CNN opartej na architekturze ResNet50, następnie klasyfikacja zdarzeń przy użyciu sieci ConvLSTM	Brak implementacji	Brak implementacji	Precyzja 22.93%, czułość 24.31%, F_1 23.60%, dokładność 22.72%, *AUC* 82%	UCF-Crime
[120]	2022	Zmodyfikowany model OpenPose (zastosowanie sieci ResNet-51 zamiast VGG-19 w etapie ekstrakcji cech), analiza cech morfologicznych i dynamicznych punktorów szkieletów, klasyfikacja zdarzeń przy użyciu SVM	Brak implementacji	Brak implementacji	Dokładność 94.2%, precyzja 97.1%, czułość 90.5%, F_1 93.7%	Opracowane przez autorów

Źródło	Data	Techniki przetwarzania obrazów użyte w celu klasyfikacji zdarzeń alarmowych	Algorytmy przestrzennej analizy sceny	Algorytmy MOT	Wskaźniki oceny jakości	Bazy danych
[121]	2023	Autorski algorytm pomijania mniej znaczących ujęć dla detekcji zdarzeń o różnej długości czasu trwania, detekcja zdarzeń przy użyciu autorskiego modelu sieci Inflated Inception I3D network	Brak implementacji	Brak implementacji	Precyzja 47%, czułość 47%, F_1 47%, dokładność 47%, *AUC* 84%	UCF-Crime
[122]	2023	Ekstrakcja cech przy użyciu sieci C3D (ang. *3D Convolutional Neural Networks*), detekcja i lokalizacja aktywności przy użyciu autoenkodera opartego na CNN, klasyfikacja zdarzeń przy użyciu sieci GAN	Brak implementacji	Brak implementacji	Dokładność dla bazy UT Interaction: 97.5%, HCA: 89.6%, UCF-Crime: 47.34%, *AUC* dla bazy UCF-Crime: 80.6%	UT Interaction, HCA oraz UCF-Crime
[123]	2024	Ekstrakcja cech przy użyciu sieci CNN, następnie klasyfikacja zdarzeń przy użyciu sieci Bi-LSTM	Brak implementacji	Brak implementacji	Precyzja 79%, czułość 79%, F_1 79%, 32 *FPS*	UCF-Crime
[124]	2024	Ekstrakcja cech i klasyfikacja zdarzeń przy użyciu sieci MD-RAN (ang. *Multiscale Dilated assisted Residual Attention Network*), implementacja autorskiego optymalizatora hiper parametrów sieci MRP-ChOA (ang. *Modified Random Parameter-based Chimp Optimization Algorithm*)	Brak implementacji	Śledzenie aktywności przy użyciu zmodyfikowanego modelu YOLO v3 (AM-YOLO V3)	Dokładność/czułość/precyzja/F_1 dla bazy UCF-Crime: 95%/95%/60%73%, dla bazy XD-Violence: 95%/95%/40%/56%	UCF-Crime oraz XD-Violence
[125]	2024	Algorytm pomijania mniej znaczących ujęć dla detekcji zdarzeń o różnej długości czasu trwania, detekcja zdarzeń przy użyciu modelu DL o nazwie AnomalyNet bazującego na cechach przestrzenno-czasowych i przepływie optycznym	Brak implementacji	Brak implementacji	Dla bazy UFC-Crime: precyzja 47%, czułość 42%, F_1 44%, dokładność 51%, *AUC* 86%	UCF-Crime oraz ShanghaiTech

W Tab. 3.1 użyto pięć kryteriów porównawczych:

1) Techniki przetwarzania obrazów użyte w celu klasyfikacji zdarzeń alarmowych – kryterium mające na celu wskazanie kluczowych metod i algorytmów użytych w poszczególnych systemach IVSS do realizacji zadań związanych z detekcją i klasyfikacją obiektów, ekstrakcją cech i klasyfikacją zdarzeń alarmowych.
2) Algorytmy przestrzennej analizy sceny – kryterium mające na celu wskazanie kluczowych metod i algorytmów użytych w poszczególnych systemach IVSS do realizacji zadań związanych z oszacowaniem parametrów ruchu obiektów w środowisku rzeczywistym monitorowanej sceny.
3) Algorytmy MOT– kryterium mające na celu wskazanie kluczowych metod i algorytmów, użytych w poszczególnych systemach IVSS do realizacji zadań związanych z gromadzeniem informacji o zmianach parametrów ruchu oraz zachowania obserwowanych obiektów w funkcji czasu.
4) Wskaźniki oceny jakości – kryterium mające na celu wskazanie, jakich wskaźników oceny jakości używa się najczęściej podczas testów systemów IVSS oraz wskazanie ich wartości liczbowych w celu umożliwienia późniejszego porównania poszczególnych systemów IVSS z systemem projektowanym w niniejszej rozprawie.
5) Bazy danych – kryterium mające na celu wskazanie, jakich baz danych używa się najczęściej w celu ewaluacji opracowanych systemów IVSS pod kątem zdolności do klasyfikowania zdarzeń alarmowych.

Techniki przetwarzania obrazów użyte w celu klasyfikacji zdarzeń alarmowych

Pierwsze kryterium porównawcze umożliwia dokonanie klasyfikacji systemów IVSS na trzy etapy rozwoju.

Pierwszy etap obejmuje rozwiązania z lat 1997–2014. Zastosowane tu metody zostały zorientowane głównie na możliwości użycia analitycznych technik detekcji obiektów pierwszoplanowych, realizowanych głównie poprzez detekcję ruchomych elementów tła, takich jak krawędzie [97], czy też analizę kontrastu luminacji [98]. W kolejnych latach autorzy implementowali bardziej efektywne metody detekcji obiektów oparte na analizie cech złożonych, takich jak cechy wyglądu (kolory [101], tekstury [106]) oraz cech ruchu (szybkość trajektorii i lokalizacja [104]). Analityczne techniki detekcji obiektów pierwszoplanowych, zorientowane głównie na identyfikacji ruchomych elementów tła, często uniemożliwiały realizacje procesu detekcji i śledzenia obiektów pozostających w spoczynku. Bazowanie wyłącznie na prostych cechach wyglądu było częstym powodem braku implementacji metod klasyfikacji obiektów, co również wpływało niekorzystnie na precyzje detekcji zdarzeń alarmowych.

Problemy związane z detekcją i klasyfikacją obiektów zostały w znacznym stopniu rozwiązane w drugim etapie rozwoju systemów IVSS, za którego początek można przyjąć rok 2015, kiedy to rozpoczęto stosowanie zaawansowanych modeli uczenia maszynowego. Implementacje ML obejmowały głównie SVM [109], CNN i LSTM [111], YOLO [111,113] oraz OpenPose [112].

Trzeci etap rozwoju, za którego początek można przyjąć rok 2021, wskazuje na zmianę klasycznej koncepcji projektowania systemów IVSS, w myśl której każde zadanie związane z detekcją, klasyfikacją i śledzeniem obiektów, a następnie identyfikacją zdarzeń alarmowych było realizowane osobno, przy użyciu różnych technik, w formie wieloetapowego przetwarzania obrazu. Najnowsze badania wskazują na utrzymujący się trend stosowania dwuetapowego procesu analizy obrazu przy użyciu dwóch modeli DL. Pierwszy etap odnosi się do realizacji ekstrakcji pożądanych cech z obrazu, najczęściej przy użyciu modeli DL opartych na architekturach CNN [116–119,122,123]. Drugi etap to finalna klasyfikacja zdarzeń alarmowych na podstawie opracowanych cech, najczęściej przy użyciu modeli DL opartych na architekturach LSTM [116,118,119,123]. Prezentowane podejście dwuetapowe w pewnym stopniu upraszcza projekt systemu IVSS, umożliwiając uzyskanie akceptowalnych wyników w zakresie ograniczonego obszaru zastosowań, ściśle związanego z użytą bazą danych uczących. Najczęściej jednak takie podejście utrudnia osiągnięcie wyższego stopnia uniwersalności i skalowalności systemu do pracy w różnych środowiskach. Jest tak, ponieważ uproszczona konstrukcja w wielu przypadkach uniemożliwia realizację zadań związanych ze śledzeniem obiektów, estymacją parametrów ruchu obiektów w rzeczywistym układzie odniesienia, wyznaczaniem indywidualnych stref ROI, czy precyzyjnym dostosowywaniem parametrów czułości detekcji każdego z rozpoznawanych zdarzeń.

Algorytmy analizy przestrzennej sceny

W pracy [99] autorzy opisują algorytm kalibracji sceny na podstawie analizy rozkładu prędkości obiektów oraz ich pozycji na obrazie. Zastosowana tu logika oparta jest na uproszczeniu, iż ludzie zazwyczaj poruszają się z podobną prędkością, niezależnie od odległości od kamery (głębokości sceny). To podejście wymaga nieprzerwanego i precyzyjnego śledzenia obiektów oraz analizy ich ruchu, co może być wyzwaniem w zatłoczonych lub dynamicznie zmieniających się scenach. Autorzy sami wymieniają szereg problemów mogących powodować nieprawidłowe działanie algorytmu takie jak: okluzje osób, brak detekcji osób, stacjonarność osób, a także pionowy kierunek ruchu osób (względem obrazu).

W pracy [103] w celu detekcji upadku, autorzy używają pomiarów rzeczywistej wysokości osoby, której wartość jest wynikiem estymacji przestrzeni 2D obrazu do rzeczywistej przestrzeni

sceny 3D podawanej w metrach. Niestety, autorzy nie podają szczegółów związanych z zastosowaną techniką estymacji wymiarów 2D do 3D.

System prezentowany w pracy [106] został wyposażony w implementację metody kalibracji sceny zaczerpniętą z artykułu [108]. Metoda ta polega na użyciu mobilnego wózka kalibracyjnego oraz zestawu znaczników świetlnych. Przez precyzyjne obliczenie pozycji i orientacji wózka kalibracyjnego w globalnym systemie współrzędnych sceny oraz analizę znanych współrzędnych znaczników świetlnych, algorytm umożliwia estymację obserwowanych punktów 2D do przestrzennego układu 3D. Opisana metoda pozwala uzyskać akceptowalną efektywność procesu kalibracji sceny, natomiast sam proces kalibracji jest utrudniony z powodu konieczności użycia specjalistycznego sprzętu, jakim jest wózek mobilny wraz z zestawem znaczników świetlnych.

W pracy [110] zaproponowano użycie parametrów kalibracyjnych kamery, które są udostępnione w bazach danych, wybranych do celu uczenia maszynowego modeli detekcji zdarzeń alarmowych. Autorzy nie zdradzają szczegółów związanych ze sposobem użycia tych parametrów kalibracyjnych do estymacji punktów 2D obrazu do przestrzennego układu 3D. Ponadto, autorzy nie odnoszą się problemu, w jaki sposób pozyskać wymagane parametry kalibracyjne w przypadku próby implementacji systemu w środowisku rzeczywistym.

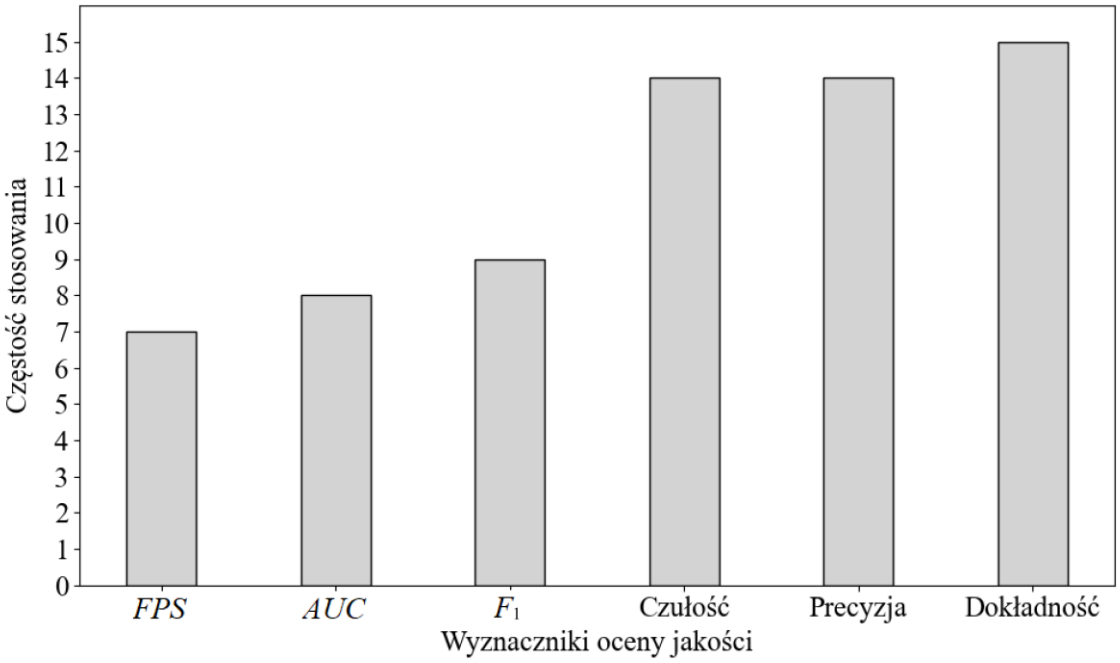
Analiza drugiego kryterium porównawczego wyraźnie wskazuje na obecność luk badawczych z obszaru metod i algorytmów szacowania parametrów ruchu i lokalizacji obiektów w środowisku rzeczywistym monitorowanej sceny. Zaledwie cztery prace [99,103,106,110] spośród analizowanych prac referencyjnych zawierają informacje na ten temat, przy czym tylko dwie prace [99,106] przedstawiają konkretne rozwiązania implementacyjne.

Algorytmy śledzenia

W większości prac zawierających implementację śledzenia obiektów [98,99,104,106,109,110] zadanie to najczęściej realizowane jest przy użyciu analitycznych metod szacowania najmniejszej odległości pomiędzy lokalizacją obiektów na kolejnych ujęciach. Dodatkowo, autorzy posiłkują się implementacją filtru Kalmana [109,111] oraz filtru cząsteczkowego [104,106]. Należy zwrócić uwagę, iż najnowsze opracowania systemów IVSS (z ostatnich 4 lat) są niemal całkowicie pozbawione tej funkcjonalności. Jest to w dużej mierze spowodowane brakiem detekcji i klasyfikacji obiektów, wynikającym z zastosowania uproszczonego, dwuetapowego podejścia opartego na modelach DL. Wyjątek stanowi praca [124], w której śledzenie zostało zrealizowane przy użyciu zmodyfikowanego modelu YOLO v3 (AMYOLO V3).

Wyznaczniki oceny jakości

Na podstawie analizy kolumny „*Wyznaczniki oceny jakości*”, zawartej w Tab. 3.1, opracowano wykres ilustrujący częstość stosowania poszczególnych wskaźników oceny jakości, w celu opisu rezultatów przeprowadzonych testów referencyjnych systemów IVSS (zob. Rys. 3.3).

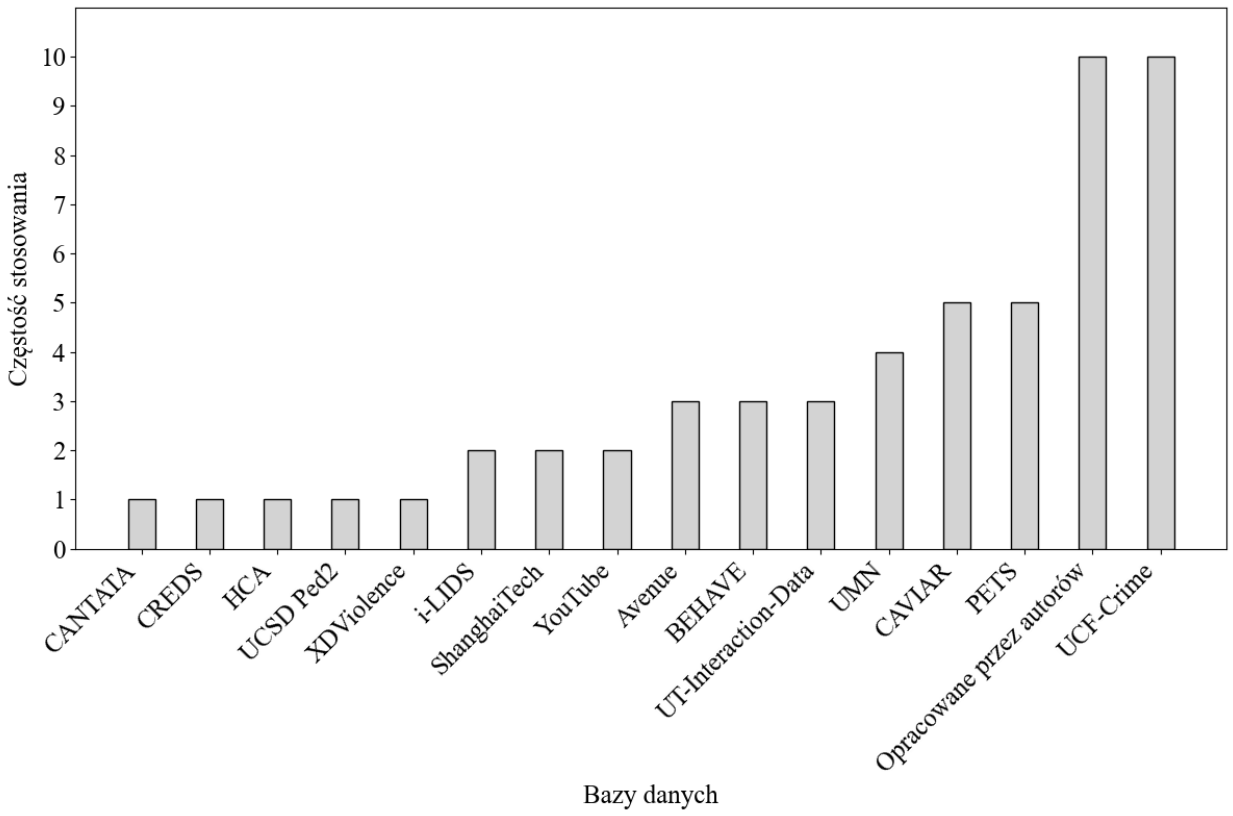


Rys. 3.3 Wykres częstości stosowania poszczególnych wskaźników oceny jakości w celu realizacji testów referencyjnych systemów IVSS

Zgromadzone w ten sposób dane wskazują, że do grupy najczęściej używanych wskaźników oceny jakości systemów IVSS należą: dokładność, precyzja, czułość, F_1, *ACU* oraz *FPS*. Wskaźnik *FPS* jest silnie skorelowany z wydajnością sprzętową stanowiska komputerowego, użytego w celu realizacji testów. Z tego powodu próby porównania systemów IVSS z użyciem wskaźnika *FPS* mogą nie dostarczyć odpowiednio rzetelnych i miarodajnych wyników. Pozostałe wskaźniki umożliwiają przeprowadzenie właściwego porównania z projektowanym w niniejszej rozprawie systemem IVSS tylko przy spełnieniu warunku, jakim jest użycie tych samych zbiorów baz danych.

Bazy danych

Na podstawie analizy kolumny „*Bazy danych*” zawartej w Tab. 3.1 opracowano wykres ilustrujący częstość użycia poszczególnych baz danych, w celu realizacji procesu uczenia maszynowego oraz weryfikacji systemów w analizowanych pracach badawczych (zob. Rys. 3.4). Opracowane zestawienie wskazuje na szerokie spektrum baz danych używanych przez autorów. Wśród zestawu szesnastu propozycji, tylko dwie wyróżniają się największą liczbą cytowań: indywidualne bazy opracowane przez autorów oraz baza UCF-Crime, obie z częstością cytowań równą 10.



Rys. 3.4 Wykres częstości użycia poszczególnych baz danych, stosowanych w celu realizacji uczenia maszynowego referencyjnych systemów IVSS

Pozostałe bazy danych charakteryzują się znacznie mniejszą liczbą zastosowań, nie przekraczającą 5 cytowań. Bazy danych specjalnie opracowane przez autorów nie są udostępniane publicznie, co uniemożliwia użycie ich w testach porównawczych z systemem projektowanym w ramach niniejszej rozprawy. Baza UCF-Crime [126] zawiera łącznie 1900 nagrań wideo, każde nagranie zawiera średnio 7247 ujęć. Baza umożliwia wytrenowanie modeli do celu klasyfikacji trzynastu zdarzeń alarmowych: nadużycie, aresztowanie, podpalenie, napaść, wypadek drogowy, włamanie, eksplozja, bójka, rabunek, strzelanina, kradzież, kradzież w sklepie oraz wandalizm. Pomimo największej liczby cytowań, baza UFC-Crime nie umożliwi przetestowania efektywności dla wszystkich klas zdarzeń alarmowych, do których rozpoznawania projektowany system IVSS będzie przeznaczony. Co więcej, przeprowadzone rozpoznanie literaturowe wskazuje na brak istniejących baz danych, które umożliwiłyby w pełni realizację tego zadania. Z tego powodu, autor rozprawy użyje zbioru nagrań testowych opracowanych we własnym zakresie.

3.4.2 Klasyfikacja zdarzeń alarmowych

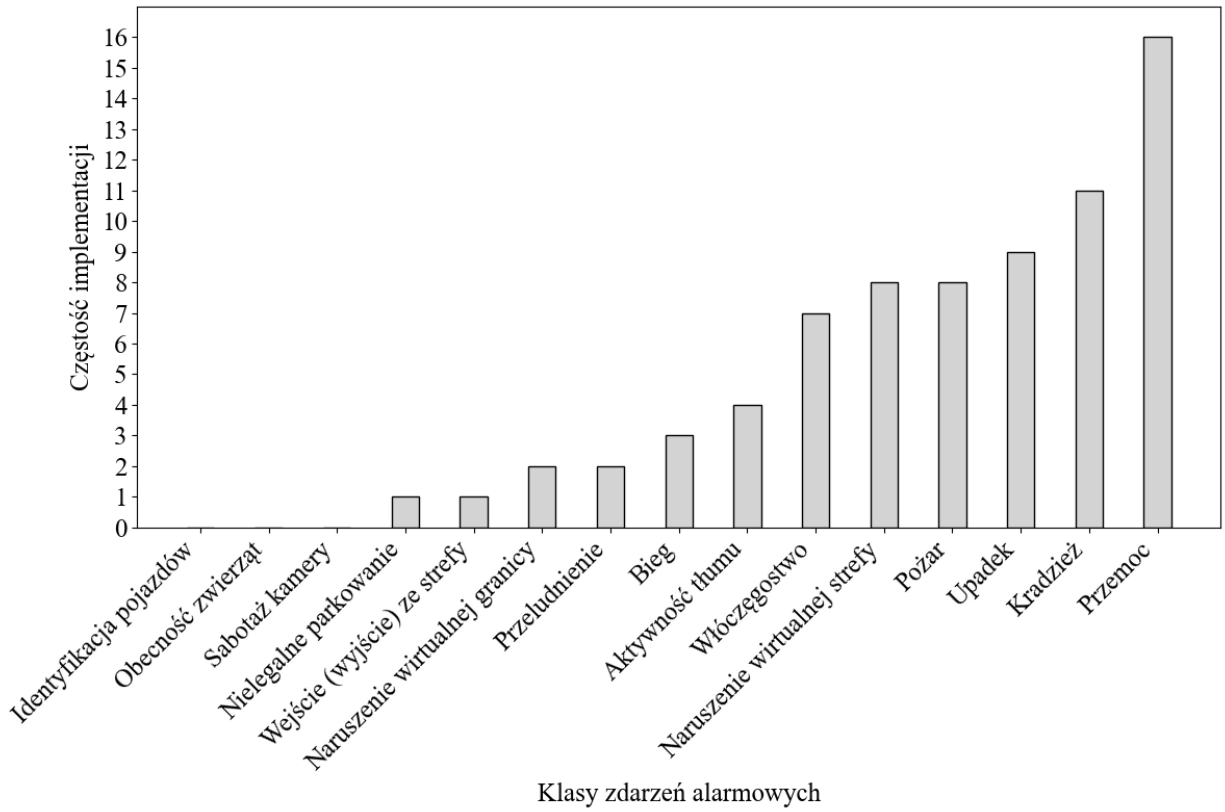
Tab. 3.2 stanowi rozszerzenie Tab. 3.1. Dane w niej zawarte umożliwiają dokonanie porównania zdolności do klasyfikacji wybranych zdarzeń alarmowych pomiędzy systemami referencyjnymi, a projektowanym w ramach rozprawy systemem IVSS. Ponadto, dane te umożliwiają analizę zmian i trendów ewolucji tego typu rozwiązań w układzie chronologicznym.

Tab. 3.2 Wykaz detekcji zdarzeń alarmowych realizowanych przez analizowane systemy referencyjne IVSS

Źródło	**Data**	**Detekcja zdarzeń alarmowych**														
		Identyfikacja pojazdów	Obecność zwierząt	Sabotaż kamery	Nielegalne parkowanie	Wejście (wyjście) ze strefy	Naruszenie wirtualnej granicy	Przeludnienie	Bieg	Aktywność tłumu	Włóczęgostwo	Naruszenie wirtualnej strefy	Pożar	Upadek	Kradzież	Przemoc
[97]	1997							✓			✓	✓		✓		
[98]	2004											✓		✓		✓
[99]	2005							✓			✓	✓				
[100]	2005											✓		✓		
[101]	2009				✓		✓				✓	✓				
[103]	2012										✓			✓	✓	✓
[104]	2014									✓	✓	✓		✓		
[106]	2014											✓		✓		
[109]	2015					✓					✓				✓	
[110]	2017								✓							✓
[111]	2018															✓
[112]	2019						✓					✓				
[113]	2019									✓					✓	✓
[115]	2020								✓	✓				✓		✓
[116]	2021									✓	✓		✓	✓	✓	✓
[117]	2021												✓		✓	✓
[118]	2021															✓
[119]	2022												✓		✓	✓
[120]	2022								✓					✓		✓
[121]	2023												✓		✓	✓
[122]	2023												✓		✓	✓
[123]	2024												✓		✓	✓
[124]	2024												✓		✓	✓
[125]	2024												✓		✓	✓
Praca autorska	**2024**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Duża liczba analizowanych systemów referencyjnych oraz szerokie spektrum stosowanych baz danych i związana z tym mnogość rozpoznawanych klas zdarzeń alarmowych utrudnia realizację porównania. Z tego powodu, autor rozprawy postanowił ograniczyć zakres klas zdarzeń alarmowych użytych w tym porównaniu do piętnastu, których detekcję docelowo będzie umożliwiał projektowany system IVSS. Autor rozprawy pragnie nadmienić, iż w celu usprawnienia kategoryzacji analizowanych zdarzeń alarmowych, zastosowano ujednolicenie adnotacji niektórych typów klas. Jest to zasadne z uwagi na różnorodność nazw poszczególnych klas, które w zależności od użytej bazy danych mogą mieć inną narrację i adnotację, lecz odnosić się do tego samego lub bardzo podobnego zdarzenia alarmowego. Przykładem mogą być adnotacje klas zastosowane w bazie danych UFC-Crime takie jak: „*znęcanie się*” (ang. *abuse*), „*napaść*” (ang. *assault*) oraz „*bójka*” (ang. *fighting*), które autor rozprawy postanowił ujednolicić do postaci jednej klasy o nazwie „*przemoc*”.

Na podstawie danych zawartych w Tab. 3.2 opracowano wykres ilustrujący częstość implementacji funkcji detekcji poszczególnych zdarzeń alarmowych w referencyjnych systemach IVSS (zob. Rys. 3.5).



Rys. 3.5 Wykres częstości implementacji funkcji detekcji zdarzeń alarmowych w referencyjnych systemach IVSS

Wykres ten pozwala na ocenę stopnia innowacyjności oraz kompleksowości projektowanego systemu IVSS w kontekście istniejącego krajobrazu technologicznego. Analiza częstości implementacji funkcji detekcji poszczególnych zdarzeń alarmowych wykazała, że najczęściej implementowaną funkcją jest detekcja przemocy (16 z 24 źródeł), co świadczy o wysokiej

priorytetyzacji tego typu zdarzeń w systemach IVSS różnych autorów. Kolejną często występującą funkcją jest detekcja kradzieży, zaimplementowana w 11 analizowanych systemach. Detekcja upadku również zajmuje wysoką pozycję z 9 implementacjami, co podkreśla zainteresowanie bezpieczeństwem osób w monitorowanych obszarach.

Mniej powszechnymi, lecz nadal istotnymi funkcjami są detekcje pożaru oraz naruszenia wirtualnej strefy, obie z 8 implementacjami. Detekcja włóczęgostwa została uwzględniona w 7 systemach, co może odzwierciedlać rosnące zainteresowanie monitorowaniem obszarów publicznych i prywatnych w kontekście wystąpienia potencjalnych zagrożeń związanych z tym zdarzeniem (np. sprzedaż narkotyków, nielegalne stanowiska handlowe, przebywanie osób bezdomnych lub pod wpływem alkoholu).

Pozostałe 8 funkcji jest rzadziej implementowane, co może wskazywać na wyższy stopień specyfiki zastosowań lub trudności w opracowaniu syntezy tych detekcji w obszarze jednego systemu. Szczególną uwagę należy zwrócić na funkcje detekcji sabotażu kamery, detekcji niepożądanej obecności zwierząt oraz identyfikacji pojazdów (ANPR), które nie zostały zaimplementowane w żadnej z prac referencyjnych, pomimo iż stanowią istotny obszar zainteresowań współczesnych systemów monitoringowych.

3.5 Podsumowanie

W ogólnym ujęciu, klasyfikacje typów systemów IVSS można uprościć do systemów detekcji zdarzeń alarmowych, realizujących precyzyjną klasyfikację wielu typów zdarzeń oraz systemów detekcji aktywności anormalnych, działających na zasadzie klasyfikatora binarnego. Z uwagi na przyjęte cele badawcze oraz docelowe przeznaczenie, projektowany w ramach niniejszej rozprawy system IVSS będzie mógł być sklasyfikowany do grupy systemów detekcji zdarzeń alarmowych.

Projektowany system będzie przeznaczony do pracy z jednostkami GPU, umożliwiającymi akcelerację obliczeniową zastosowanych modeli ML. Ponadto system będzie korzystał z przetwarzania wielowątkowego CPU. Opracowany system IVSS będzie zatem skoncentrowany na architekturze FOG, w której to przetwarzanie danych jest rozszerzone o możliwość użycia infrastruktury sieciowej (łącze: kamera–stacja robocza), lecz wciąż odbywa się blisko lokalizacji źródła sygnału wizyjnego. Kompromis pomiędzy lokalnym a centralnym przetwarzaniem umożliwi osiągnięcie elastyczności i skalowalności systemu.

Autor przeprowadził szczegółową analizę porównawczą referencyjnych systemów IVSS na podstawie 24 pozycji literaturowych. Prace te obejmują szeroki zakres dat (1997–2024), przy czym 67% cytowań obejmuje okres ostatnich dziesięciu lat, natomiast 46% cytowań obejmuje okres ostatnich pięciu lat. Wybrane kryteria porównawcze dotyczą: technik przetwarzania obrazów użytych w celu klasyfikacji zdarzeń alarmowych, implementacji algorytmów analizy przestrzennej sceny oraz śledzenia obiektów, wskaźników oceny jakości stosowanych w procesie

testowania systemów, baz danych stosowanych w procesie uczenia maszynowego, a także klasyfikacji zdarzeń alarmowych rozpoznawanych przez poszczególne systemy.

Analiza technik przetwarzania obrazów i klasyfikacji zdarzeń alarmowych ukazuje ewolucję od prostych technik analitycznych po zaawansowane modele uczenia maszynowego, włączając SVM, CNN, LSTM, YOLO, oraz OpenPose. Pierwszy etap (1997–2014) charakteryzował się głównie zastosowaniem analitycznych technik detekcji, które często nie uwzględniały klasyfikacji obiektów, co wpływało na niską precyzję detekcji zdarzeń alarmowych. Drugi etap (od 2015 r.) wyróżnia się stosowaniem zaawansowanych modeli uczenia maszynowego, które znacząco poprawiły efektywność detekcji i klasyfikacji. Najnowsze badania (od 2021 r.) sugerują przejście na dwuetapowy proces analizy obrazu z zastosowaniem głębokiego uczenia, co upraszcza konstrukcję systemów IVSS, lecz może ograniczać ich uniwersalność i skalowalność.

Analiza metod przestrzennej analizy sceny wskazuje na istnienie poważnych luk badawczych w tej dziedzinie. Zaledwie 4 prace zawierają informacje dotyczące parametryzacji ruchu i lokalizacji obiektów w środowisku rzeczywistym monitorowanej sceny, przy czym tylko dwie prace przedstawiają konkretne rozwiązania implementacyjne. Zastosowane rozwiązania opierają się głównie na uproszczonych założeniach lub danych zewnętrznych, co w wielu przypadkach może wpływać niekorzystnie na dokładność estymacji 3D, natomiast chęć uzyskania wyższej dokładności wymaga użycia specjalistycznego sprzętu kalibracyjnego.

Należy zwrócić uwagę, iż najnowsze opracowania systemów IVSS (z ostatnich 4 lat) są niemal całkowicie pozbawione funkcjonalności MOT. Powodem tego może być skupienie się na dwuetapowej analizie obrazu, opartej na głębokim uczeniu, której zastosowanie ogranicza możliwości systemów IVSS do detekcji i klasyfikacji obiektów.

Analiza wskaźników oceny jakości pokazuje, iż najczęściej stosowanymi są: dokładność, precyzja, czułość, F_1, *ACU* oraz *FPS*.

Wśród najczęściej używanych baz danych dominuje UCF-Crime oraz własne bazy danych opracowane przez autorów. Przeprowadzona analiza wskazuje jednak na brak uniwersalnych baz danych, umożliwiających jednoznaczne porównanie każdego systemu pod kątem różnych scenariuszy zdarzeń alarmowych. Problem ten narzuca konieczność opracowania własnych zbiorów bazodanowych.

Aby usprawnić kategoryzację i porównanie, autor rozprawy wprowadził ujednolicenie adnotacji niektórych typów zdarzeń alarmowych, co jest istotne ze względu na różnorodność nazewnictwa w zależności od użytej bazy danych. Klasyfikacja dotyczy 15 typów zdarzeń, których detekcja będzie docelowo realizowana przez projektowany system IVSS. Są to: identyfikacja pojazdów, sabotaż kamery, niepożądana obecność zwierząt, nielegalne parkowanie, analiza wejścia (wyjścia) ze strefy, naruszenie wirtualnej granicy, przeludnienie, bieg, analiza aktywności tłumu, włóczęgostwo, naruszenie wirtualnej strefy, pożar, upadek, kradzież oraz przemoc.

Najczęściej implementowanymi funkcjami detekcji zdarzeń alarmowych są detekcja przemocy, kradzieży oraz upadku, co świadczy o wysokiej priorytetyzacji tych zdarzeń w kontekście systemów IVSS. Mniej powszechne, ale nadal istotne, są detekcje pożaru i naruszenia wirtualnej strefy zastrzeżonej. Pozostałe 8 funkcji jest rzadziej implementowane, co może wskazywać na wyższy stopień specyfiki zastosowań lub trudności w opracowaniu syntezy tych detekcji w obszarze jednego systemu. Szczególnymi przykładami są detekcja sabotażu kamery, niepożądanej obecności zwierząt oraz identyfikacja pojazdów, gdzie pomimo ich istotności w nowoczesnych systemach IVSS, nie zostały zaimplementowane w żadnej z analizowanych prac referencyjnych.

3.6 Luki badawcze

Na podstawie badań i analiz przeprowadzonych w rozdziale 3, autor dokonał identyfikacji następujących luk badawczych w obszarze inteligentnych systemów nadzoru wizyjnego:

- Brak uniwersalności i skalowalności w najnowszych systemach – systemy opracowane po 2021 roku stosują uproszczony, dwuetapowy proces analizy obrazu, który może ograniczać ich uniwersalność i skalowalność. Wiele z tych prac nie obejmuje zadań związanych detekcją, klasyfikacją oraz śledzeniem obiektów, analizą przestrzenną sceny, czy dostosowywaniem czułości detekcji, co sugeruje na konieczność dalszego rozwoju systemów IVSS zdolnych do działania w różnorodnych środowiskach.
- Ograniczenia w kalibracji i estymacji przestrzeni sceny – analizowane prace rzadko zawierają konkretne, implementacyjne rozwiązania problemu kalibracji sceny i estymacji rzeczywistych parametrów ruchu obiektów. Tylko nieliczne prace przedstawiają metody kalibracji, a te, które to robią, często wymagają użycia specjalistycznego sprzętu lub bazują na parametrach dostępnych w używanych bazach danych. To wskazuje na brak uniwersalnych, łatwo dostępnych metod kalibracji i estymacji przestrzennej sceny.
- Niedostateczne rozwiązania śledzenia MOT w nowoczesnych systemach – najnowsze systemy niemal całkowicie pomijają funkcjonalność śledzenia obiektów. Tylko jedna analizowana praca z ostatnich czterech lat zawiera nowoczesne podejście do śledzenia (oparte na zmodyfikowanym modelu YOLO v3). Wskazuje to na potrzebę dalszych badań w zakresie opracowania efektywnych tego typu metod.
- Rozpoznawanie specyficznych zdarzeń alarmowych – pomimo że niektóre klasy zdarzeń, takie jak przemoc czy kradzież, są często rozpoznawane, inne ważne kategorie, takie jak detekcja sabotażu kamery, niepożądanej obecności zwierząt czy identyfikacji pojazdów, są stosowane rzadko lub w ogóle nie są uwzględniane w analizowanych systemach referencyjnych. Wskazuje to na istotną lukę badawczą, dotyczącą braku zdolności istniejących systemów IVSS do zapewnienia kompleksowej ochrony monitorowanych przestrzeni.

- Ograniczone badania detekcji zdarzeń w zróżnicowanych środowiskach – część analizowanych systemów skupia się na detekcji kilku typów zdarzeń, często w bardzo ograniczonych i specyficznych kontekstach (np. detekcja upadków w domach opieki). W celu osiągnięcia pożądanej uniwersalności należy zatem rozszerzyć badania na detekcję zdarzeń w różnorodnych środowiskach, takich jak miejsca publiczne, obszary miejskie czy środowiska przemysłowe.
- Brak publicznie dostępnych, uniwersalnych baz danych do testowania i porównywania systemów – z przeprowadzonej analizy wynika, że znaczna część badań opiera się na bazach danych opracowanych przez autorów, które nie są dostępne publicznie, ograniczając możliwość weryfikacji i porównania wyników przez inne zespoły badawcze. To podkreśla potrzebę opracowania i udostępnienia baz danych, które umożliwiłyby miarodajne porównanie opracowanych systemów IVSS.
- Potrzeba rozwoju adaptacyjnych systemów IVSS – przeprowadzona analiza wskazuje na niewielką liczbę badań nad systemami zdolnymi do adaptacji do zmieniających się warunków środowiskowych i dynamicznych scenariuszy, takich jak zmienne oświetlenie, różnorodność zachowań ludzi, czy zmienna gęstość tłumu. Rozwój systemów adaptacyjnych, które mogą dostosowywać parametry detekcji i klasyfikacji w czasie rzeczywistym, jest kluczowym kierunkiem przyszłych badań.
- Integracja detekcji zdarzeń z rozwiązaniami reagowania – większość istniejących systemów IVSS skupia się na detekcji i klasyfikacji zdarzeń, z mniejszym naciskiem na integrację z systemami reagowania, takimi jak automatyczne powiadamianie służb ratunkowych lub zewnętrznych systemów zarządzania kryzysowego. Rozwój systemów IVSS oferujących zintegrowane rozwiązania detekcji i reagowania, stanowi istotny obszar potencjalnych innowacji.

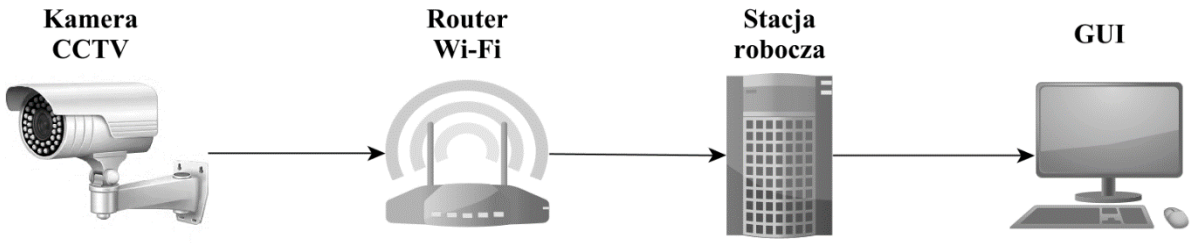
Wnioski te podkreślają konieczność dalszych badań i rozwoju w dziedzinie inteligentnych systemów nadzoru wizyjnego, z naciskiem na zwiększenie uniwersalności, skuteczności i zakresu aplikacji takich systemów. Innowacje w zakresie algorytmów uczenia maszynowego, metod analizy przestrzennej sceny i śledzenia MOT, a także rozwoju publicznych baz danych i integracji systemów z rozwiązaniami wczesnego reagowania mogą przyczynić się do znacznego postępu w tej dziedzinie.

4 Opis projektowanego systemu IVSS

Rozdział ten stanowi wprowadzenie do części implementacyjnej. Przedstawione w nim informacje dotyczą ogólnej koncepcji przepływu informacji w projektowanym systemie IVSS, charakterystyki sprzętowej oraz programowej stanowiska badawczego, a także typów źródeł sygnału wizyjnego wybranych w celu weryfikacji opracowanego systemu.

4.1 Koncepcja przepływu informacji wizyjnej

Na Rys. 4.1 przedstawiono ogólny schemat budowy projektowanego systemu IVSS.



Rys. 4.1 Ogólny schemat budowy projektowanego systemu IVSS

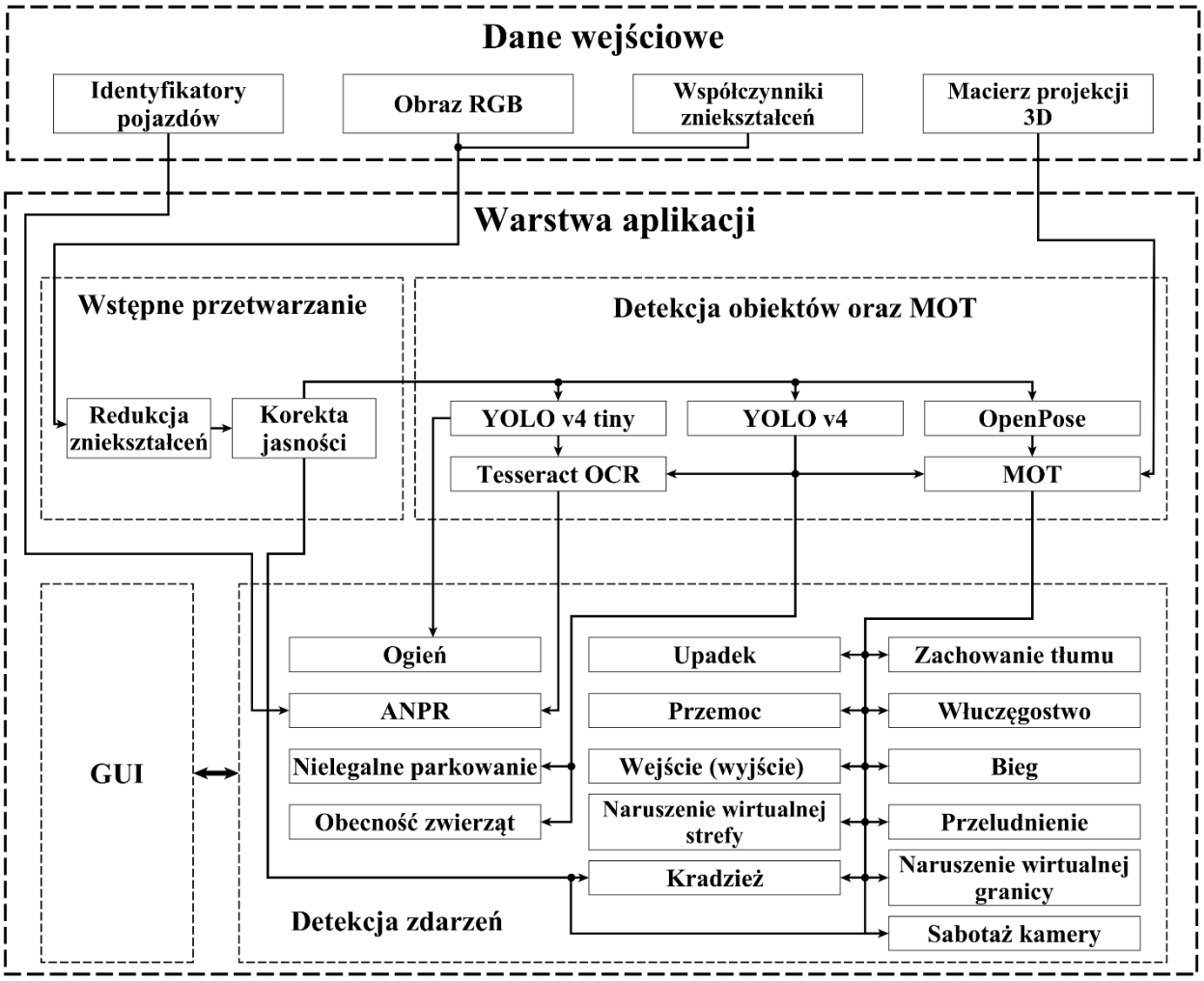
Transmisja sygnału wizyjnego z kamery CCTV do stacji roboczej realizowana jest bezprzewodowo przy użyciu konwencjonalnego routera Wi-Fi, działającego w paśmie 2.4 GHz. Stacja robocza wyposażona jest w aplikację przeznaczoną do analizowania sygnału wizyjnego i raportowania zdarzeń alarmowych przy pomocy GUI.

Schemat przepływu informacji w warstwie aplikacji przedstawiono na Rys. 4.2. Opracowana aplikacja przyjmuje 4 typy danych wejściowych.

Pierwszy typ danych wejściowych dotyczy bazy danych identyfikatorów pojazdów. Jest to plik tekstowy rozdzielony przecinkami (CSV), zawierający listę autoryzowanych numerów tablic rejestracyjnych wraz z odpowiadającymi personaliami właścicieli pojazdów. Plik ten jest wymagany wyłącznie w przypadku aktywowania funkcji systemu realizującej zadanie identyfikacji pojazdów.

Drugi typ to dane RGB (obrazy). Docelowo aplikacja będzie przeznaczona do pracy z obrazami w rozdzielczości nie przekraczającej 1920 ×1080 px. Przekroczenie tej wartości może wpłynąć niekorzystnie na szybkość przetwarzania pojedynczego ujęcia. System będzie wspierać wszystkie formaty sygnału wizyjnego, które są obsługiwane przez bibliotekę OpenCV. Sygnał wizyjny RGB jest wymaganym typem danych wejściowych, niezbędnym do zapewnienia prawidłowej pracy systemu.

Trzeci typ danych wejściowych dotyczy wskaźników dystorsji kamery, a więc parametrów opracowanych w procesie kalibracji kamery (zob. rozdział 2.3). Dane te dostarczane są w formacie XML. Użyta biblioteka OpenCV oferuje wsparcie dla formatu XML, umożliwiając łatwe zapisywanie oraz odczytywanie parametrów, takich jak macierze, wektory, wartości liczbowe czy



Rys. 4.2 Schemat przepływu informacji w warstwie aplikacji

struktury danych pomiędzy różnymi aplikacjami. Kamery monitoringowe montowane są zazwyczaj w trudnodostępnych miejscach, co może uniemożliwiać przeprowadzenie procesu ich kalibracji. Dlatego, ten typ danych wejściowych nie jest wymagany, natomiast zaleca się jego użycie w przypadku zastosowania kamer obarczonych znaczącymi zniekształceniami obrazu.

Czwarty typ danych wejściowych dotyczy macierzy projekcji 3D, opracowanej w procesie kalibracji sceny, która umożliwia rzutowanie pikseli obrazu 2D do współrzędnych 3D obserwowanego podłoża sceny. Dane te również dostarczane są w formacie XML i są wymagane do zapewnienia prawidłowej pracy systemu.

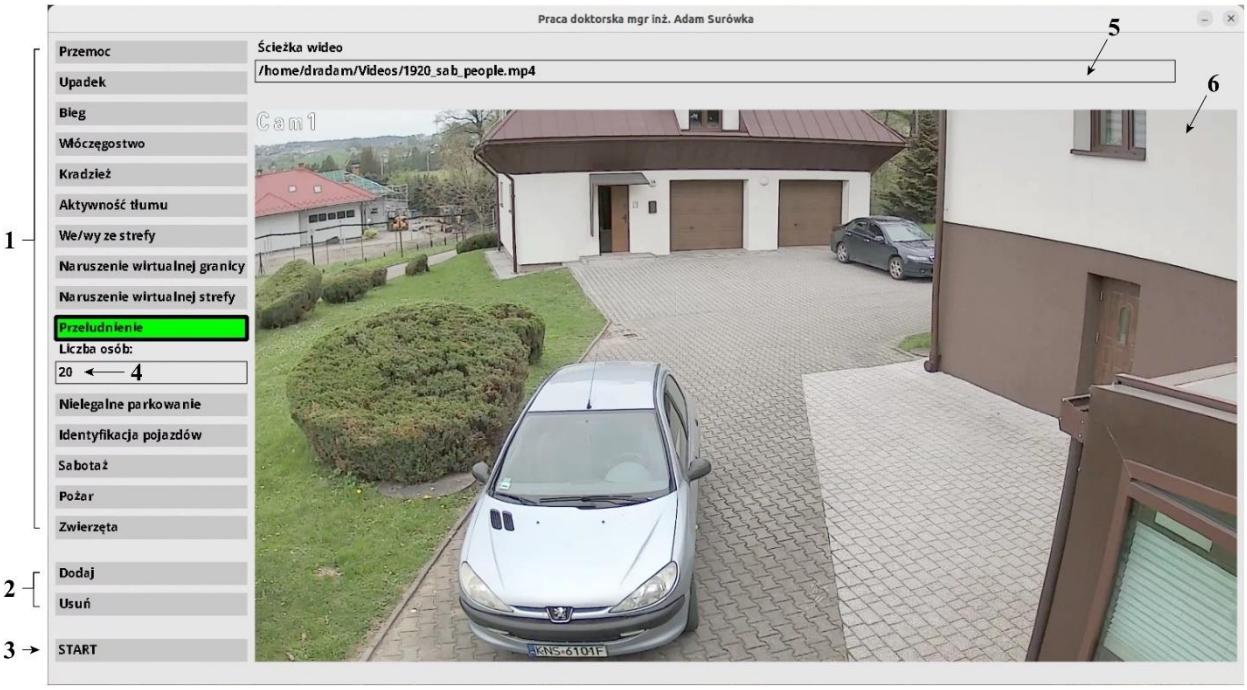
W warstwie aplikacji, jako pierwszy, realizowany jest moduł wstępnego przetwarzania obrazów. Dotyczy on zadań związanych z automatyczną korektą zniekształceń oraz jasności obrazu. W następnym kroku realizowana jest detekcja obiektów wraz z algorytmem śledzenia MOT. Dalej, realizowane są funkcje decyzyjne wszystkich detektorów zdarzeń alarmowych. Dzięki GUI, użytkownik ma możliwość wprowadzania nastaw do systemu oraz podglądu obrazu w czasie rzeczywistym wraz z naniesionymi adnotacjami dotyczącymi wykrytych alarmów.

Implementacja programowa projektowanej aplikacji zostanie zrealizowana przy użyciu programowania obiektowego w języku C++. Decyzja ta opiera się na dogłębnej analizie wymagań

systemowych, zwłaszcza w kontekście balansu pomiędzy łatwością implementacji logiki aplikacji a wydajnością kodu wynikowego. Implementacja projektu IVSS w języku C++ umożliwia również efektywną integrację z licznymi bibliotekami i frameworkami opracowanymi w tym języku. Program aplikacji zostanie opracowany w popularnym, darmowym edytorze kodów źródłowych Visual Studio Code w formie aplikacji konsolowej.

4.2 GUI

Aplikacje konsolowe, oparte na interfejsie tekstowym, są zauważalnie łatwiejsze w implementacji w porównaniu z aplikacjami wyposażonymi w graficzny interfejs użytkownika. Nie zapewniają jednak łatwej i intuicyjnej kontroli dla użytkownika końcowego. W celu osiągnięcia kompromisu pomiędzy złożonością implementacji a wygodą obsługi aplikacji, autor rozprawy opracował specjalny interfejs GUI bazujący wyłącznie na funkcjonalnościach oferowanych przez bibliotekę OpenCV. Na Rys. 4.3 przedstawiono GUI projektowanego systemu IVSS, na którym cyframi 1–6 oznaczono główne elementy menu kontekstowego.

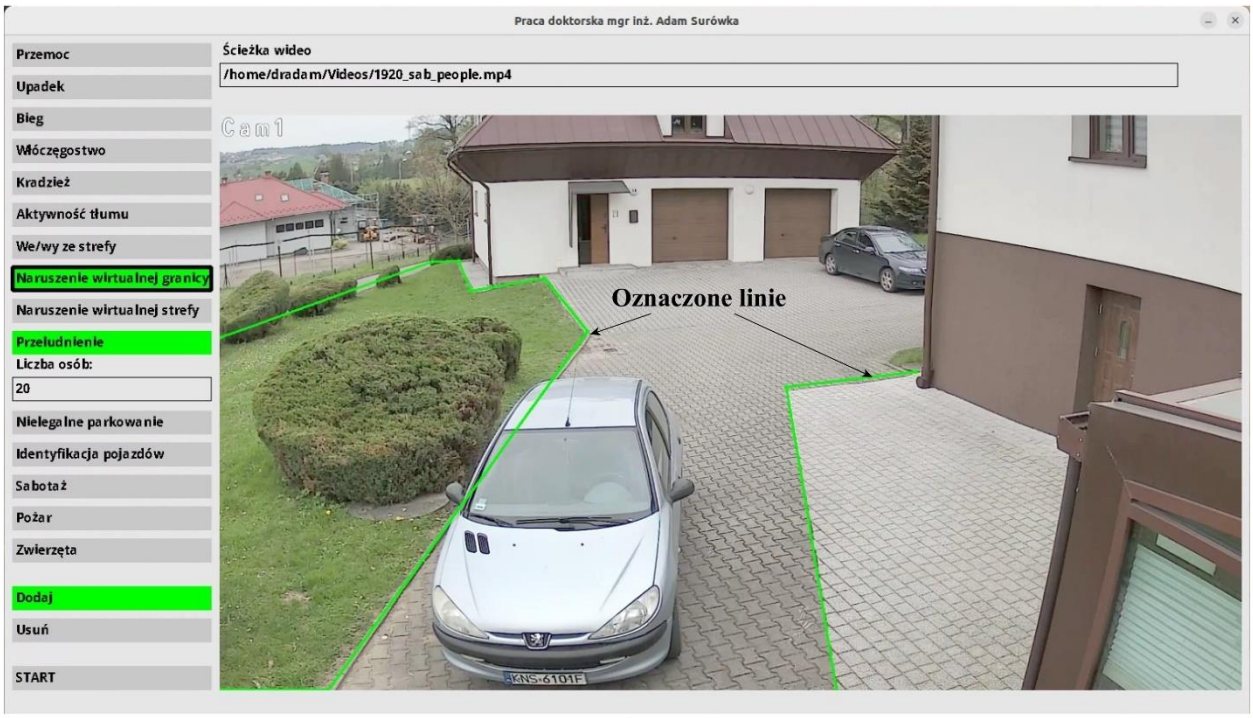


Rys. 4.3 GUI projektowanego systemu IVSS

Poszczególne elementy menu kontekstowego dotyczą:

1. Lista przycisków funkcyjnych służących do aktywowania i dezaktywowania wszystkich funkcjonalności systemu. Kliknięcie danego przycisku powoduje zmianę jego koloru na zielony, co wskazuje na aktywację skorelowanej funkcjonalności.
2. Przyciski pomocnicze, umożliwiające dodawanie i usuwanie wirtualnych stref oraz granic na analizowanej scenie. Część funkcjonalności systemu, obejmująca zadania, takie jak detekcja naruszenia strefy zastrzeżonej, wymaga uprzedniego oznaczenia stosownych

obszarów i granic. Na Rys. 4.4 przedstawiono przykład realizacji oznaczeń wirtualnych linii oraz wirtualnych obszarów w oknie podglądu obrazu.



a)

b)

Rys. 4.4 GUI – przykład realizacji oznaczeń: a) wirtualnych linii, b) wirtualnych obszarów

Użytkownik może wprowadzić te dane do systemu w następujących krokach:

a. Aktywowanie funkcji poprzez kliknięcie skorelowanego przycisku funkcyjnego.
b. Kliknięcie przycisku „*Dodaj*” aktywuje możliwość nanoszenia oznaczeń w obszarze podglądu obrazu, co sygnalizuje zmiana koloru tego przycisku na zielony.

c. Kliknięcie kursorem myszy w obszarze podglądu obrazu skutkuje pojawieniem się graficznego znacznika punktowego. W zależności od wymagań aktywowanej funkcji, dodawanie kolejnych punktów umożliwia automatyczne tworzenie obszarów lub linii. Zarówno obszary jak i linie mogą składać się z dowolnej liczby punktów tworząc pożądany kształt. Linie składają się z połączonych odcinków utworzonych z sekwencji oznaczonych punktów. Obszary tworzone są w ten sam sposób z dodatkowym wypełnieniem wewnątrz zaznaczenia. Na obrazie kształty te są wizualizowane kolorem zielonym.
d. Ponowne kliknięcie przycisku „*Dodaj*” zapisuje wprowadzone oznaczenie do pamięci systemu, co sygnalizuje zmiana koloru tego przycisku na pierwotny.
e. Każdorazowe kliknięcie przycisku „*Usuń*” umożliwia usunięcie ostatniego zapisu obszaru bądź linii.

3. Przycisk umożliwiający uruchomienie programu głównego wraz z podglądem obrazu w czasie rzeczywistym.
4. Dodatkowe okno edycji tekstu pojawiające się w przypadku aktywowania funkcji „*Przeludnienie*”. Element ten umożliwia użytkownikowi wprowadzenie progowej wartości liczby osób znajdujących się na obrazie, powyżej której system będzie alarmował o wystąpieniu przeludnienia.
5. Okno tekstowe zawierające ścieżkę dostępu do źródła sygnału wizyjnego.
6. Okno podglądu sygnału wizyjnego.

Zaprezentowane rozwiązanie oferuje niski stopień skomplikowania implementacji kodu, wysoką responsywnością aplikacji oraz możliwość uruchomienia GUI na dowolnym systemie operacyjnym MS Windows, Mac OS czy Linux.

4.3 Stanowisko badawcze

Jako bazę stanowiska badawczego użyto zmodyfikowaną stację roboczą HP Z620 [127]. Wprowadzone modyfikacje dotyczą następujących komponentów:

- procesor CPU: 2 × Intel® Xeon(R) CPU E5-2660 v2 @ 2.20 GHz,
- pamięć RAM: 4 × 8 GB DIMM DDR3,
- dysk twardy: Samsung 250 GB 2.5” SATA SSD Seria 850 EVO,
- procesor graficzny: NVIDIA® GeForce RTX™ 3080 VENTUS 3X 10 GB OC,
- dodatkowa jednostka zasilająca: SilentiumPC Vero L3 700 W (SPC267).

Podstawowa jednostka zasilająca stację roboczą nie jest wyposażona w złącze 2 × PCI Express 8 pin, które jest wymagane do zasilania GPU. Ponadto, ograniczona liczba nieużywanych złączy zasilających uniemożliwia zastosowanie przejściówek typu SATA–PCI Express 8 pin lub

Molex–PCI Express 8 pin. Z tego powodu postanowiono zastosować dodatkową, zewnętrzną jednostkę zasilającą SilentiumPC Vero L3, obsługującą wyłącznie GPU.

Realizacja testów opisywanych w dalszych częściach rozprawy będzie przeprowadzona z użyciem platformy systemowej Ubuntu 22.04.3 LTS 64-bit. Obecnie jest to najnowsza wersja tej dystrybucji, udostępniana na zasadach licencji GNU GPL, która zapewnia pożądaną stabilność i szybkość działania, a także wsparcie dla realizacji projektów otwarto źródłowych. Autor rozprawy pragnie zaznaczyć, iż projektowany system IVSS będzie mógł być również uruchomiony na platformach systemowych MS Windows oraz Mac OS, gdyż użyte biblioteki oraz modele ML zapewniają taką możliwość.

W kontekście konfiguracji programowej, zastosowanie procesora graficznego NVIDIA RTX 3080 wymaga instalacji dedykowanego sterownika w najnowszej obecnie wersji 550.40.07. W ramach projektowanego systemu IVSS jednym z kluczowych aspektów jest implementacja algorytmów sztucznej inteligencji i uczenia maszynowego. W celu pełnego użycia mocy obliczeniowej GPU, autor korzysta z technologii NVIDIA CUDA Toolkit. Dla zapewnienia optymalnej wydajności i kompatybilności, zdecydowano się na instalację wersji CUDA Toolkit 12.3.107. Dopełnieniem konfiguracji jest biblioteka cuDNN, której wersja 8.9.7 została specjalnie dobrana do współpracy z CUDA Toolkit 12.3.107. cuDNN, jako dedykowana biblioteka przyspieszająca operacje sieci neuronowych, jest niezastąpionym komponentem w projektach implementujących głębokie uczenie.

Dobór systemu operacyjnego oraz omówionych zależności wynika z decyzji autora rozprawy, opartej na potrzebie zapewnienia maksymalnej wydajności, stabilności oraz kompatybilności w ramach projektowanego systemu IVSS. Są to najnowsze wersje oprogramowań dostępne w czasie projektowania systemu, co podkreśla innowacyjność i zaawansowanie technologiczne całego projektu.

4.4 Źródła sygnału wizyjnego

Testy projektowanego systemu IVSS będą realizowane dla dwóch typów źródeł sygnału wizyjnego: nagrania oraz strumienie wideo z kamery CCTV, a także nagrania wideo opracowane przez autora w formie symulacji 3D.

4.4.1 Testowe nagrania wideo

Część baz danych używanych przez badaczy (zob. Rys. 3.4) nie jest obecnie dostępna online, co utrudnia odtworzenie badań oraz porównanie wyników. Ponadto, dostęp do istniejących baz bywa ograniczony i wymaga uzyskania specjalnej zgody od autorów, co, jak pokazało doświadczenie autora niniejszej rozprawy, często kończy się brakiem odpowiedzi. Nawet jeśli dostęp do baz danych jest otwarty, część z nich charakteryzuje się niską jakością nagrań, co

ogranicza ich przydatność do zaawansowanych analiz. Ponadto, aktualnie nie odnaleziono bazy danych zawierającej nagrania dotyczące wszystkich rozpatrywanych w niniejszej rozprawie zdarzeń alarmowych. W obliczu tych wyzwań opracowanie własnej bazy danych stało się nie tylko rozsądnym, ale i koniecznym krokiem, pozwalającym na pełną kontrolę nad jakością oraz zakresem dostępnych materiałów wideo. Taka baza danych stanowi solidną podstawę do dalszych badań jakościowych i ilościowych mających na celu określenie efektywności proponowanych algorytmów detekcji wybranych zdarzeń alarmowych.

Do nagrywania filmów użyto klasyczną kamerę monitoringową TP-Link Tapo C500 zamontowaną na mobilnym statywie (zob. rozdział 2.2), co pozwoliło na elastyczność w wyborze lokalizacji, kątów widzenia oraz wysokości montażu w zakresie od 2 do 3.5 m. Nagrania rejestrowano z maksymalną dostępne rozdzielczością FHD oraz częstotliwością 15 *FPS*. Zapis filmów był realizowany do popularnego formatu MP4. Nagrania realizowano w warunkach naturalnego dziennego oświetlenia, zarówno na zewnątrz, jak i wewnątrz budynków. Wybrano różnorodne lokalizacje, takie jak ulice miejskie, parkingi, korytarze biurowe oraz pomieszczenia mieszkalne, aby jak najlepiej oddać rzeczywiste warunki, w których mogłyby wystąpić wybrane zdarzenia alarmowe. Każde zdarzenie alarmowe zostało starannie zainscenizowane zgodnie z ustalonymi scenariuszami. Scenariusze te obejmowały różne zachowania ludzi, zwierząt i pojazdów, interakcje między nimi, a także symulacje sytuacji kryzysowych, takich jak pożary, akty przemocy, czy sabotaż kamery. W Tab. 4.1 przedstawiono charakterystykę opracowanej bazy danych pod kątem liczby ujęć dla każdego typu zdarzenia.

Tab. 4.1 Charakterystyka ilościowa opracowanej bazy nagrań wideo
*Użyto filmy z serwisu YouTube

Nazwa zestawu	**Sumaryczna liczba ujęć**
Identyfikacja pojazdów	17413
Obecność zwierząt	16071
Sabotaż kamery (zasłonięcie/oślepienie/obrót obiektywu)	16561 (6233/5112/5216)
Nielegalne parkowanie	18655
Wejście (wyjście) ze strefy	15202
Naruszenie wirtualnej granicy	14270
Przeludnienie	16904
Bieg oraz analiza aktywności tłumu	17660 (10052/7608)
Włóczęgostwo	17201
Naruszenie wirtualnej strefy	15339
Pożar *	16933
Upadek	16577
Kradzież	18003
Przemoc	15900
Łącznie	**233786**

Opracowana baza danych zawiera łącznie 233786 ujęć, które zostały podzielone na 15 odrębnych zestawów, z których każdy odpowiada innemu typowi zdarzenia alarmowego.

Zestaw „*Identyfikacja pojazdów*” zawiera 17413 ujęć, w których symulowano różne scenariusze związane z zadaniem ANPR, takie jak wjazd na parking czy wjazd na posesję. Charakterystyka sposobu poruszania się pojazdów w tych miejscach pozwoliła na zredukowanie zakresu badań dotyczących prędkości najazdu do 20 km/h. Opracowane ujęcia charakteryzują się różnorodnością pojazdów (samochody osobowe oraz dostawcze). Z uwagi na brak dostępności pojazdów oznaczonych różnymi typami tablic rejestracyjnych, nagrania odbywały się wyłącznie z użyciem tablic zwykłych (czarne znaki na białym tle). Samochody poruszały się w polu widzenia kamery przy różnym kącie najazdu oraz różnych odległościach od kamery nie przekraczających 10 m.

W skład zestawu „*Obecność zwierząt*” wchodzi 16071 ujęć, w których symulowano pojawienie się zwierząt na monitorowanych obszarach. Scenariusze dotyczyły zarówno obecności psów jak i kotów różnych gatunków, poruszających się po terenach publicznych i prywatnych. Scenariusze obejmowały nagrania realizowane zarówno wewnątrz, jak i na zewnątrz budynków.

Zestaw „*Sabotaż kamery*” zawiera 16561 ujęć, z których 6233 przedstawia próby zasłonięcia kamery przy użyciu przysłon o różnym kolorze, strukturze powierzchni oraz kształcie (np. ręka, materiał tekstylny, tworzywo sztuczne, papier), 5112 – oślepienie obiektywu przy użyciu intensywnej wiązki światła latarki, a 5216 – obrócenie obiektywu. Scenariusze te symulowano w różnych warunkach, aby uwzględnić potencjalne trudności w detekcji sabotażu w realnych sytuacjach zarówno wewnątrz, jak i na zewnątrz budynków.

Zestaw „*Nielegalne parkowanie*” obejmuje 18655 ujęć przedstawiających pojazdy parkujące w miejscach niedozwolonych, takich jak strefy wyłączone z ruchu czy strefy zielone. Ujęcia te charakteryzują się różnorodnością pojazdów, od samochodów osobowych, przez dostawcze po motocykle.

W zestawie „*Wejście (wyjście) ze strefy*” znajduje się 15202 ujęć, które pokazują przypadki wejścia lub wyjścia osób z lub do pomieszczeń zamkniętych. Ujęcia charakteryzuje wysoki stopień różnorodności konfiguracji pola widzenia kamery, co pozwala na dokładne odwzorowanie warunków rzeczywistych, zarówno dla przypadków montażu kamery na zewnętrz jak i wewnątrz monitorowanego pomieszczenia.

Zestaw „*Naruszenie wirtualnej granicy*” zawiera 14270 ujęć, w których symulowano przejścia osób przez ogrodzenie w niedozwolonym miejscu. Scenariusze obejmowały nagrania realizowane dla ogrodzeń o różnym typie konstrukcji, takich jak siatka, panel czy pełna konstrukcja murowana. Takie podejście umożliwi ocenę efektywności algorytmu w obu przypadkach, zarówno gdy osoba będzie widoczna po obu stronach ogrodzenia (np. siatka, panel), lub też nie (np. pełna konstrukcja murowana).

W skład zestawu „*Przeludnienie*” wchodzi 16904 ujęć symulujących sytuacje, w których liczba osób w monitorowanym obszarze przekracza bezpieczne normy. Scenariusze obejmowały nagrania z miejsc publicznych, zarówno wewnątrz, jak i na zewnątrz budynków.

Zestaw „*Bieg oraz analiza aktywności tłumu*” obejmuje 18757 ujęć, z czego 5112 dotyczy biegu, a 7502 – analizy aktywności tłumu. Scenariusze obejmowały różne rodzaje aktywności fizycznej, od spacerów, przez szybki chód, po nagłe przyspieszenia, oraz reakcje tłumu na nagłe wydarzenia dotyczące rozbiegania się bądź ucieczki co najmniej trzech osób jednocześnie.

Zestaw „*Włóczęgostwo*” zawiera 17201 ujęć, w których monitorowano osoby pozostające w spoczynku lub poruszające się bez wyraźnego celu w obrębie określonych stref przez określony czas. Scenariusze obejmowały nagrania realizowane zarówno w miejscach publicznych charakteryzujących się dużym ruchem pieszych typu przystanki komunikacji miejskiej, jak i w miejscach rzadko uczęszczanych typu „*ślepa uliczka*”.

Zestaw „*Naruszenie wirtualnej strefy*” obejmuje 15339 ujęć przedstawiających przypadki, które mogą być interpretowane jako wkroczenie na teren zabroniony, takie jak prywatne posesje, obszary objęte nadzorem czy strefy zakazu wstępu.

Ze względu na wysokie niebezpieczeństwo związane z nagrywaniem zdarzenia pożaru w miejscach publicznych, zestaw „*Pożar*” jako jedyny został opracowany przy użyciu gotowych filmów pobranych z popularnej platformy internetowej YouTube. W celu ujednolicenia parametrów jakościowych nagrań w opracowanej bazie, wszystkie filmy zostały pobrane w rozdzielczości 1920 × 1080 px, a następnie każdy z nich został przekonwertowany w celu uzyskania częstotliwości klatkowania 15 *FPS*.

W zestawie „*Upadek*” znajduje się 16577 ujęć przedstawiających przypadki upadków osób, które zostały zainscenizowane w różnych lokalizacjach, takich jak ulice miejskie, chodniki, czy wnętrza budynków. Scenariusze uwzględniały upadki, które mogą wystąpić w wyniku potknięcia się, nagłego zasłabnięcia czy obrażeń ciała nabytych wskutek ataków napastnika. Nagrania zostały zrealizowane zarówno wewnątrz, jak i na zewnątrz budynków.

Zestaw „*Kradzież*” obejmuje 18003 ujęć, w których symulowano różne scenariusze związane z identyfikacją kradzieży obiektu statycznego. Poprzez pojęcie obiekt statyczny należy rozumieć dowolny przedmiot, który powinien znajdować się przez cały czas obserwacji w tym samym miejscu na obrazie. Scenariusze obejmowały nagrania realizowane zarówno wewnątrz, jak i na zewnątrz budynków.

Zestaw „*Przemoc*” zawiera 15900 ujęć, przedstawiających różne przypadki przemocy fizycznej, takie jak bójki czy napady. Scenariusze symulowano w różnych miejscach publicznych i prywatnych, aby oddać pełne spektrum sytuacji, w których może dojść do eskalacji konfliktu, w aranżacji zarówno wewnątrz, jak i na zewnątrz budynków.

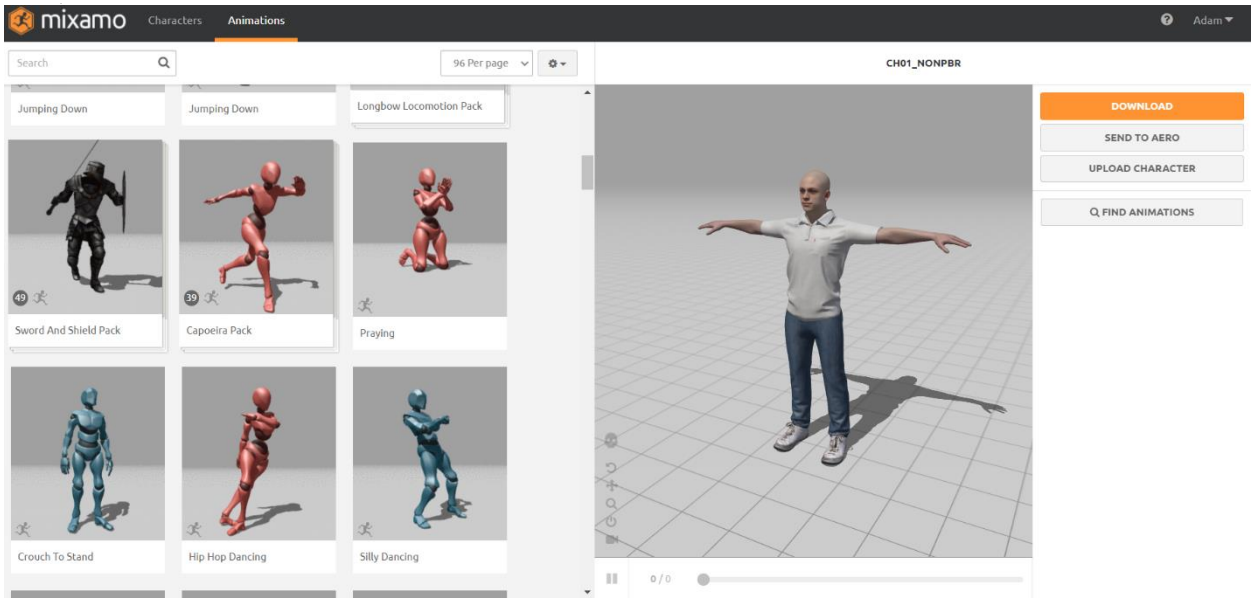
Realizacja nagrań została przeprowadzona z udziałem 25 ochotników (9 kobiet i 16 mężczyzn) w wieku 12–64 lat. Tak zróżnicowana grupa pozwoliła na stworzenie bazy danych testowych, odzwierciedlających szerokie spektrum cech demograficznych, co zwiększa skuteczność oceny opracowywanych algorytmów. Wszystkie nagrane filmy zostały manualnie oznaczone przez autora rozprawy w sposób umożliwiający precyzyjną identyfikację ujęć, na których występują zdarzenia alarmowe. Dzięki starannej symulacji wybranych typów zdarzeń oraz różnorodności warunków nagrywania, opracowana baza pozwala na dokładną i wszechstronną ocenę skuteczności algorytmów w rzeczywistych warunkach operacyjnych.

4.4.2 Symulacje 3D

Sporządzenie nagrań wideo w realnych warunkach często wymaga uzyskania zezwoleń, wynajęcia obiektu, zatrudnienia statystów, personelu, a także montażu sprzętu rejestrującego i nagrywającego. W celu obniżenia kosztów oraz skrócenia czasu przygotowania materiału do testów, autor postanowił użyć symulacji 3D, opracowanych w popularnym programie Blender [128]. Blender jest darmowym oprogramowaniem umożliwiającym opracowanie precyzyjnej konfiguracji symulowanej sceny, jak również parametryzację wirtualnej kamery, w celu renderingu nagrań wideo, charakteryzujących się wysokim stopniem odwzorowania rzeczywistości. Opracowane symulacje 3D będą użyte w celu wstępnego określenia zdolności projektowanego systemu do realizacji trzech kluczowych zadań: detekcji osoby leżącej, identyfikacji ANPR oraz procesu analizy przestrzennej sceny.

W celu osiągnięcia wysokiego stopnia realizmu odwzorowania ruchu i wyglądu człowieka, autor postanowił użyć animowane modele ciał, dostępne w popularnym serwisie internetowym Mixamo [129]. Mixamo dysponuje obszerną biblioteką gotowych do użycia postaci 3D oraz animacji, co pozwala na szybkie tworzenie realistycznych scen z udziałem ludzi, bez konieczności ręcznego modelowania. Postacie i animacje pobrane z Mixamo można modyfikować oraz bezpośrednio importować do sceny w Blenderze, dzięki czemu proces tworzenia materiałów wideo jest szybki i efektywny. Zaimplementowane w projektowanym systemie detektory Yolo v4 oraz OpenPose doskonale radzą sobie z zadaniem rozpoznawania osób w symulacjach 3D, opracowanych przy użyciu modeli pobranych z Mixamo, co zostanie udowodnione w dalszych częściach niniejszej rozprawy. Na Rys. 4.5 przedstawiono widok interfejsu Mixamo zawierający bibliotekę animacji 3D.

W celu osiągnięcia wysokiego stopnia odwzorowania istotnych detali pojazdów mechanicznych, autor postanowił użyć generatora tablic rejestracyjnych online PlatesMania [130]. Narzędzie to oferuje szereg funkcji umożliwiających dostosowanie tablic rejestracyjnych do specyficznych wymagań dotyczących formatu oraz standardu dla wybranego kraju oraz regionu.



Rys. 4.5 Biblioteka animacji Mixamo [129]

Dodatkowo pozwala na personalizację tekstu, wyboru stylu (klasyczne, zabytkowe, specjalne) oraz koloru tła (białe, zielone, żółte). Możliwość podglądu w czasie rzeczywistym, jak również zapisu w różnych formatach graficznych (JPEG czy PNG) umożliwiają łatwą implementację wygenerowanej tablicy w środowisku Blender. Na Rys. 4.6 przedstawiono przykłady czterech typów tablic rejestracyjnych wygenerowanych przy użyciu generatora PlatesMania.

Rys. 4.6 Przykłady tablic rejestracyjnych wygenerowanych przy użyciu PlatesMania [130]: a) zwykłe, b) tymczasowe, c) zabytkowe, d) samochody elektryczne

Jak można zaobserwować na Rys. 4.6, wygenerowane przykłady w stopniu zadawalającym odzwierciedlają rzeczywisty wygląd tablic rejestracyjnych.

4.5 Podsumowanie

Ogólna koncepcja budowy stanowiska badawczego zakłada użycie wybranej kamery CCTV, której sygnał wizyjny transmitowany jest bezprzewodowo do stacji roboczej przy użyciu konwencjonalnego routera Wi-Fi, działającego w paśmie 2.4 GHz. Stacja robocza wyposażona jest w aplikację przeznaczoną do analizowania sygnału wizyjnego i raportowania zdarzeń alarmowych przy pomocy GUI.

Opracowana aplikacja przyjmuje 4 typy danych wejściowych: baza identyfikatorów pojazdów (opcjonalne), obraz RGB (wymagane), wskaźniki zniekształceń kamery (opcjonalne) oraz macierz projekcji 3D (wymagane). Przepływ informacji w warstwie aplikacji zdefiniowano w formie sekwencyjnej pracy czterech modułów. Są to: moduł wstępnego przetwarzania obrazów, moduł detekcji obiektów oraz analiza MOT, moduł funkcji decyzyjnych detektorów zdarzeń alarmowych oraz GUI. Implementacja programowa algorytmów zostanie opracowana w języku C++, typ aplikacji – konsolowa. Projektowany system IVSS będzie przeznaczony do pracy na platformach systemowych MS Windows oraz Mac OS, przy czym testy algorytmów opisane w dalszych częściach rozprawy zostaną przeprowadzone na platformie Ubuntu 22.04.3 LTS 64-bit. W celu osiągnięcia kompromisu pomiędzy złożonością implementacji a wygodą obsługi aplikacji, autor opracował dedykowany interfejs GUI w oparciu o bibliotekę OpenCV. Opracowane GUI umożliwia użytkownikowi końcowemu obsługę panelu startowego w celu parametryzacji pracy systemu, włączając w to aktywację bądź dezaktywację poszczególnych funkcjonalności oraz nanoszenie wirtualnych granic oraz stref na analizowany obraz.

Jako bazę stanowiska badawczego użyto zmodyfikowaną stację roboczą HP Z620. Główne modyfikacje to montaż drugiego procesora CPU Intel® Xeon(R) CPU E5-2660 v2, procesora graficznego GPU NVIDIA® GeForce RTX™ 3080 10 GB oraz zewnętrznej jednostki zasilającej o mocy 700 W przeznaczonej do zasilania GPU.

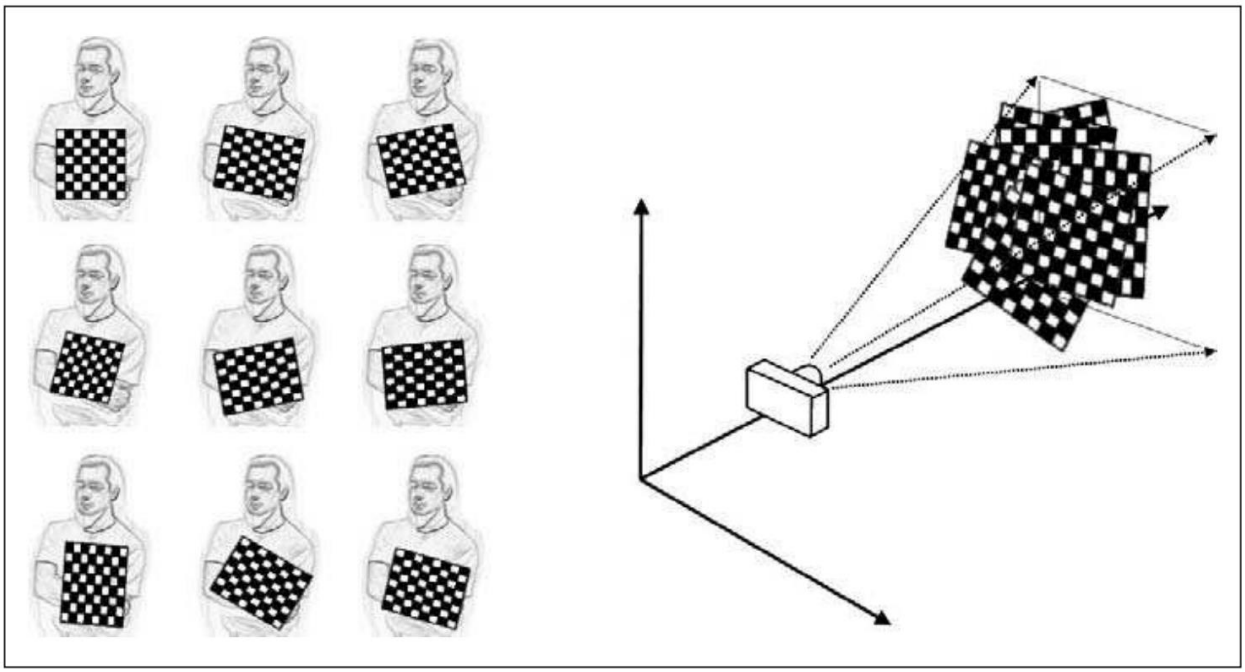
Głównym źródłem sygnału wizyjnego w celach testowych projektowanego systemu IVSS będzie baza nagrań wideo, opracowana przez autora przy użyciu wybranej kamery CCTV. Baza danych zawiera łącznie 233786 ujęć, które zostały podzielone na 15 odrębnych zestawów, z których każdy odpowiada innemu typowi zdarzenia alarmowego. W celach pomocniczych użyte zostaną również nagrania wideo, opracowane przez autora w formie symulacji 3D. Symulacje te będą przeznaczone głównie do celu realizacji wstępnej weryfikacji wybranych algorytmów, opracowanych w ramach projektowanego systemu IVSS.

5 Wstępne przetwarzanie obrazu

W tym rozdziale dokonano opisu technik wstępnego przetwarzania obrazu, realizujących następujące zadania: kalibracja kamery, automatyczna korekta jasności obrazu, detekcja i śledzenie obiektów, a także kalibracja monitorowanej sceny.

5.1 Kalibracja kamery

Zgodnie z założeniami przyjętymi w rozdziale 2.3, proces korekty zniekształceń kamery zostanie zrealizowany z użyciem dedykowanej funkcji *cv::CalibrateCamera()* zaimplementowanej w bibliotece OpenCV. Realizację tego zadania należy poprzedzić doborem wzorca kalibracyjnego, który stanowi graficzną reprezentację pewnej znanej geometrii. Biblioteka OpenCV oferuje trzy rodzaje geometrii wzorca [131]: klasyczna czarno-biała szachownica, tablica ChArUco oraz siatka kółek, występująca w wydaniu asymetrycznym i symetrycznym. Wzorce te dostępne są w formie obrazów w katalogu biblioteki OpenCV: *opencv/doc*. Autor postanowił użyć wzorca czarno-białej szachownicy z uwagi na fakt, iż jest to jeden z najczęściej stosowanych wzorców kalibracyjny w zadaniach związanych z CV. Zgodnie ze wskazówkami zawartymi w [13], autor opracował nagranie wideo z użyciem wybranego wzorca. Wizualizację aranżacji tego procesu przedstawiono na Rys. 5.1.



Rys. 5.1 Przykład aranżacji procedury kalibracji kamery z użyciem biblioteki OpenCV i wzorca szachownicy [13]

Nagranie trwające 2 minuty zostało następnie przetworzone do postaci zdjęć poklatkowych, z których wybrano 37 obrazów charakteryzujących się najlepszą jakością wizualną oraz dużym zróżnicowaniem ustawień względem kamery.

W ostatnim kroku zbiór przygotowanych obrazów został użyty w celu realizacji algorytmu kalibracji kamery. Do tego celu autor użył programu dostępnego w katalogu biblioteki OpenCV [132]. Program ten wyszukuje na obrazach geometrię szachownicy, a następnie przetwarza wyniki detekcji umożliwiając obliczenie parametrów kalibracji kamery, po czym dokonuje ich zapisu w pliku konfiguracyjnym *out_camera_data.xml*. Spośród informacji zawartych w pliku konfiguracyjnym, realizacja redukcji zniekształceń kamery wymaga użycia dwóch parametrów określających [131]: macierz kamery

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.1}$$

oraz wektor wskaźników zniekształceń

$$Z_n = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3] \tag{5.2}$$

Wyniki uzyskane w przeprowadzonym procesie kalibracji kamery przedstawiają następujące wartości:

$$\begin{cases} K = \begin{bmatrix} 1487.4609 & 0 & 929.9073 \\ 0 & 1487.4609 & 537.7375 \\ 0 & 0 & 1 \end{bmatrix} \\ Z_n = [-0.2656 \quad -0.0855 \quad -0.0004 \quad -0.0006 \quad 0.1432] \end{cases} \tag{5.3}$$

Powyższe wartości wyrażone są w pikselach. Analiza tych wartości umożliwia dokonanie oceny parametrów wewnętrznych kamery.

Rozdzielczość obrazów używanych w procesie kalibracji to 1920 × 1080 px, zatem teoretyczny punkt centralny obrazu powinien znajdować się w połowie tych wartości, a więc w punkcie (960,540). Parametry c_x oraz c_y z macierzy K wskazują, iż w procesie kalibracji współrzędne centralnego punkt obrazu zostały oszacowane jako (929.9073, 537.7375). Wartości te wskazują na różnicę o około 31 px w osi poziomej i około 2 px w osi pionowej obrazu.

Wartość ogniskowej f podana w macierzy K jest zapisana w pikselach natomiast producent kamery podaje tą wartość w milimetrach. Aby zatem porównać te dwie wartości należy ustalić jaka wartość w milimetrach przypada na jeden piksel zarówno w osi poziomej jak i osi pionowej obrazu. Znając rozmiar czujnika kamery (1/3”) oraz współczynnik proporcji obrazu (16:9), wynikający z rozdzielczości matrycy, można skorzystać z tabeli nominalnych rozmiarów czujników optycznych (zob. Rys. 5.2), zgodnie z którą autor przyjmuje, iż dla używanej kamery rozmiar sensora wynosi 5.23 mm w osi poziomej i 2.94 mm w osi pionowej.

Nominal Image Sensor Format Type Size										
			4:3 Aspect Ratio		3:2 Aspect Ratio		16:10 Aspect Ratio		16:9 Aspect Ratio	
Sensor Format Type	Real Calc Diag. (mm)	Format Type Diag. (mm)	Width (mm)	Height (mm)	Width (mm)	Height (mm)	Width (mm)	Height (mm)	Width (mm)	Height (mm)
35mm Full Frame	N/A	43.30	34.64	25.98	36.03	24.02	36.72	22.95	37.74	21.23
APS-C	N/A	30.10	24.08	18.06	25.04	16.70	25.52	15.95	26.23	14.76
4/3"	33.87	21.33	17.07	12.80	17.75	11.83	18.09	11.30	18.59	10.46
1.1"	27.94	17.60	14.08	10.56	14.64	9.76	14.92	9.33	15.34	8.63
1"	25.40	16.00	12.80	9.60	13.31	8.88	13.57	8.48	13.95	7.84
1/1.1"	23.09	14.55	11.64	8.73	12.10	8.07	12.33	7.71	12.68	7.13
1/1.2"	21.17	13.33	10.67	8.00	11.09	7.40	11.31	7.07	11.62	6.54
2/3"	16.93	10.67	8.53	6.40	8.88	5.92	9.05	5.66	9.30	5.23
1/1.6"	15.88	10.00	8.00	6.00	8.32	5.55	8.48	5.30	8.72	4.90
1/1.7"	14.94	9.41	7.53	5.65	7.83	5.22	7.98	4.99	8.20	4.61
1/1.8"	14.11	8.89	7.11	5.33	7.40	4.93	7.54	4.71	7.75	4.36
1/1.9"	13.37	8.42	6.74	5.05	7.01	4.67	7.14	4.46	7.34	4.13
1/2.0"	12.70	8.00	6.40	4.80	6.66	4.44	6.78	4.24	6.97	3.92
1/2.3"	11.04	7.83	6.26	4.70	6.51	4.34	6.64	4.15	6.82	3.84
1/2.5"	10.16	7.20	5.76	4.32	5.99	3.99	6.11	3.82	6.28	3.53
1/2.7"	9.41	6.67	5.33	4.00	5.55	3.70	5.65	3.53	5.81	3.27
1/2.8"	9.07	6.43	5.14	3.86	5.35	3.57	5.45	3.41	5.60	3.15
1/2.9"	8.76	6.21	4.97	3.72	5.16	3.44	5.26	3.29	5.41	3.04
1/3.0"	8.47	6.00	4.80	3.60	4.99	3.33	5.09	3.18	5.23	2.94
1/3.2"	7.94	5.63	4.50	3.38	4.68	3.12	4.77	2.98	4.90	2.76
1/3.4"	7.47	5.29	4.24	3.18	4.41	2.94	4.49	2.81	4.61	2.60
1/3.6"	7.06	5.00	4.00	3.00	4.16	2.77	4.24	2.65	4.36	2.45
1/4.0"	6.35	4.50	3.60	2.70	3.74	2.50	3.82	2.39	3.92	2.21
1/5.0"	5.08	3.60	2.88	2.16	3.00	2.00	3.05	1.91	3.14	1.76
1/6.0"	4.23	3.00	2.40	1.80	2.50	1.66	2.54	1.59	2.61	1.47
1/7.5"	3.39	2.40	1.92	1.44	2.00	1.33	2.04	1.27	2.09	1.18
1/9.0"	2.82	2.00	1.60	1.20	1.66	1.11	1.69	1.06	1.74	0.98

Rys. 5.2 Tabela nominalnych rozmiarów czujników optycznych [139]

Następnie korzystając ze wzoru

$$\begin{cases} f_x = \dfrac{1487.4609 \times 5.23}{1920} \approx 4.05\ mm \\ f_y = \dfrac{1487.4609 \times 2.94}{1080} \approx 4.05\ mm \end{cases} \tag{5.4}$$

obliczono wartość ogniskowej w milimetrach. Wyniki obliczeń zrealizowanych zarówno dla wysokości jak i szerokości piksela są w przybliżeniu takie same. Wartość ogniskowej kamery podana przez producenta to 3.89 mm. Różnica pomiędzy obliczoną wartością ogniskowej, a wartością podaną przez producenta wynosi 0.16 mm. Różnica ta jest relatywnie niska. Należy pamiętać, iż obliczona wartość ogniskowej to wartość przybliżona, wyznaczona na podstawie ogólnie przyjętych standardów wymiarowania czujników optycznych. Innym istotnym czynnikiem, mogącym wpływać na otrzymaną różnicę, jest funkcja automatycznego ustawiania ostrości (autofocus), która mogła zostać aktywowana w momencie realizacji nagrania z udziałem wzorca kalibracyjnego.

Wyniki przeprowadzonej kalibracji wskazują na minimalne zniekształcenia styczne (wskaźniki p_1, p_2), natomiast wskaźniki zniekształceń radialnych (k_1, k_2, k_3) przyjmują wartości mogące wprowadzać dystorsje szczególnie zauważalne w obszarach krawędzi obrazu.

Korzystając z opracowanych parametrów kalibracyjnych, przeprowadzono testy redukcji zniekształceń na zbiorze obrazów testowych. Biblioteka OpenCV dysponuje dedykowaną funkcją *cv::undistort()* przeznaczoną do realizacji procesu redukcji zniekształceń. W trakcie testów

zauważono, iż średni czas przetwarzania obrazu z użyciem tej funkcji to 31 ms. W celu skrócenia czasu procesu redukcji zniekształceń, autor postanowił użyć funkcję *cv::remap()*. Funkcja ta jest często używana w aplikacjach wymagających szybkiego czasu przetwarzania, ponieważ używa mapy przekształceń obliczanej tylko raz przy starcie programu, natomiast funkcja *cv::undistort()* powtarza te obliczenia dla każdego przetwarzanego obrazu. Zastosowanie funkcji *cv::remap()* umożliwiło redukcję czasu przetwarzania do ok. 4 ms. Autor rozprawy przyjmuje, iż maksymalna rozdzielczość analizowanego obrazu nie będzie przekraczać wartości 1920 × 10800 px (FHD). Ograniczenie to jest zasadne z uwagi na chęć ograniczenia konsumpcyjności obliczeniowej projektowanego systemu IVSS w celu osiągnięcia rzeczywistego czasu przetwarzania.

Na Rys. 5.3a przedstawiono oryginalny obraz, natomiast Rys. 5.3b przedstawia wynik procesu redukcji zniekształceń. Analiza wizualna Rys. 5.3a i 5.3b wskazuje osiągnięcie pewnego stopnia poprawy, co może zostać zauważone na pionowych i poziomych liniach konstrukcyjnych budynków znajdujących się w prawej części obrazu. Negatywnym skutkiem realizacji redukcji zniekształceń jest zauważalna zmiana geometrii obrazu w postaci charakterystycznego zwężenia zarówno w osi pionowej jak i poziomej obrazu. Z uwagi na chęć zapewnienia komfortu użytkownikowi końcowemu oraz w celu osiągnięcia wysokiej estetyki szaty graficznej projektowanego systemu IVSS, autor proponuje rozwiązanie tego problemu poprzez zachowanie tylko najistotniejszych pikseli obrazu. Jest to możliwe dzięki zmianie wartości (z 1.0 na 0.0) argumentu *alpha* w funkcji *cv::getOptimalNewCameraMatrix()*, używanej w celu obliczeń map przekształceń. Efekt końcowy przedstawiono na Rys. 5.3c.

Podsumowując, przeprowadzony proces kalibracji kamery wykazał, iż zniekształcenia radialne wprowadzają zaoblenia widoczne szczególnie w pobliżu brzegów obrazu. Redukcję zniekształceń zrealizowano z użyciem funkcji *cv::remap()*, co pozwoliło na osiągnięcie akceptowalnego czasu przetwarzania rzędu 4 ms dla pojedynczego obrazu. Końcową korektę geometrii obrazu (po redukcji zniekształceń) zrealizowano poprzez zmianę argumentu *alpha* w funkcji *cv::getOptimalNewCameraMatrix()* do wartości 0.0. Efekt tych działań (zob. Rys. 5.3c) zapewnia wymagany poziom szczegółowości detali w dalszych funkcjach przetwarzania obrazu oraz oferuje użytkownikowi końcowemu klasyczny, estetyczny widok panoramicznego podglądu monitorowanej sceny.

5.2 Korekta jasności obrazu

Jednym z ważniejszych czynników wpływających na dokładność pracy zastosowanych detektorów obiektów (OpenPose oraz YOLO v4) jest jasność obrazu. Analiza obrazów o niskim poziomie ekspozycji światła może prowadzić do rosnącej liczby detekcji typu *FN*, a w konsekwencji pogorszenia skuteczności projektowanego systemu IVSS.

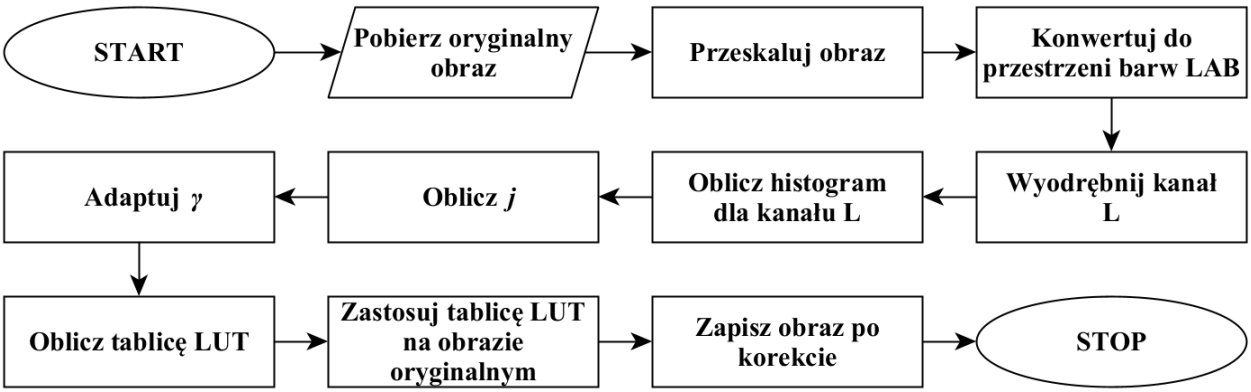
a)

b)

c)

Rys. 5.3 Wizualizacja wpływu redukcji zniekształceń na obraz: a) oryginalny, b) po redukcji zniekształceń (*alpha* = 1.0), c) po redukcji zniekształceń (*alpha* = 0.0)

W celu rozwiązania tego problemu, autor opracował algorytm, mający na celu automatyczne dostosowanie jasności obrazu. Wśród popularnych metod realizacji tego zadania, takich jak korekta współczynnika α czy β, na podstawie porównania opisanego w [133], postanowiono skorzystać z techniki korekty γ. Korekta ta działa na zasadzie nieliniowej modyfikacji wartości pikseli, wprowadzając mniejsze zmiany w jaśniejszych pikselach, a większe w ciemniejszych. Pozwala to na lepsze zachowanie jasnych detali przy jednoczesnym uwydatnieniu detali, znajdujących się ciemniejszych obszarach obrazu. Schemat blokowy algorytmu automatycznej korekty jasności obrazu przedstawiono na Rys. 5.4.

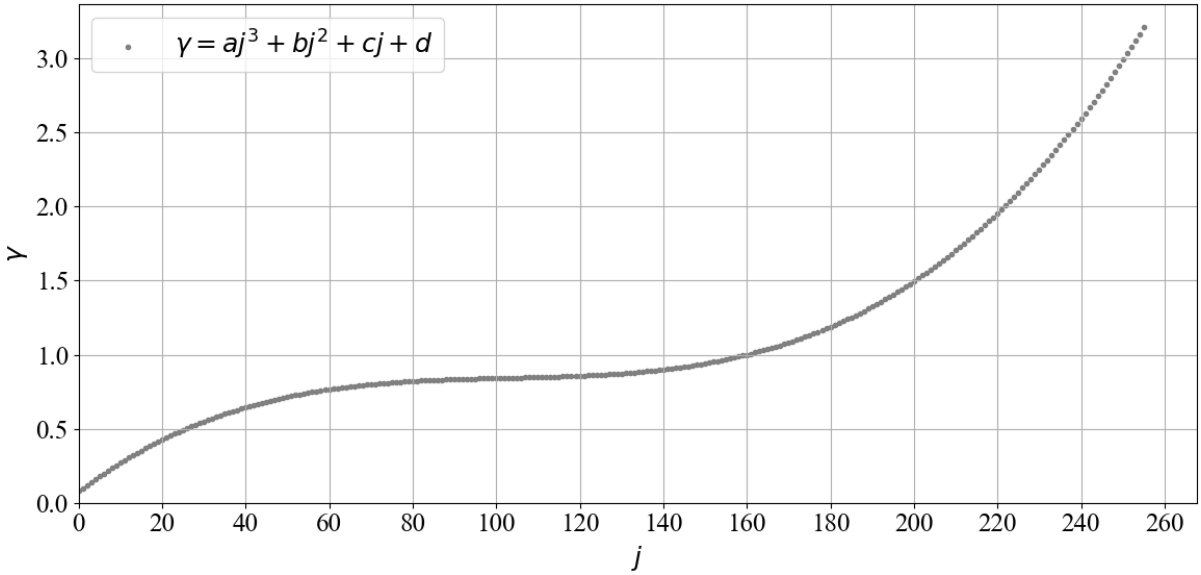


Rys. 5.4 Schemat blokowy algorytmu automatycznej korekty jasności obrazu, gdzie: j – średnia ważona jasność obrazu, γ – wartość korekty

Proces rozpoczyna się od przeskalowania obrazu wejściowego do mniejszej rozdzielczości, w celu redukcji czasu przetwarzania w dalszych krokach. Następnie, przeskalowany obraz zostaje przekonwertowany do przestrzeni barw LAB. Umożliwia to wydobycie precyzyjnych informacji dotyczących natężenia jasności obrazu zapisanej w kanale L. W dalszym kroku dla kanału L obliczany jest histogram. Każdy pojemnik (bin) histogramu odpowiada liczbie pikseli o danej wartości jasności w skali 0–255. Wartości te są ważone i sumowane, a następnie dzielone przez całkowitą liczbę pikseli, aby uzyskać średnią ważoną jasność obrazu j. Do obliczenia nowej wartości korekty użyto wielomianu 3-go stopnia

$$\gamma = aj^3 + bj^2 + cj + d \tag{5.5}$$

gdzie: $a = 6.4396312943 \cdot 10^{-7}$, $b = -1.98584 \cdot 10^{-4}$, $c = 2.10432 \cdot 10^{-2}$, $d = 7.91547 \cdot 10^{-2}$. Zależność 5.5 przedstawiono na Rys. 5.5. Zależność ta pozwala na automatyczną adaptację wartości korekty na podstawie obliczonej uprzednio wartości uśrednionej jasności obrazu. Współczynniki wielomianu zostały przez autora dobrane w sposób empiryczny. Korekta $\gamma = 1$ wyznacza granicę pomiędzy rozjaśnianiem, a przyciemnianiem. Im mniejsza wartość γ poniżej 1 tym intensywniejsze rozjaśnienie, natomiast im większa wartość powyżej 1 tym większe zaciemnienie zostaje zastosowane na obrazie. Analiza wykresu z Rys. 5.5 pozwala zauważyć obecność trzech kluczowych zakresów korekty. W pierwszym zakresie dla wartości $j < 60$



Rys. 5.5 $\gamma = f(j)$

algorytm wprowadza znaczące zmiany w celu rozjaśnienia obrazów skrajnie zaciemnionych. W drugim zakresie dla średnich wartości j (z przedziału 60–160) algorytm wprowadza jedynie subtelne korekty rozjaśniania. W trzecim zakresie dla wartości $j > 160$ algorytm wprowadza znaczące zmiany w celu przyciemnienia obrazów o zbyt wysokiej ekspozycji światła. Dalej, tworzona jest tablica LUT, której zastosowanie na obrazie oryginalnym umożliwia wprowadzenie wyznaczonej korekty poziomu jasności. Po zakończonym procesie rozjaśniania zmodyfikowany obraz zostaje zapisany do wektora o zasięgu globalnym, umożliwiając dostęp dla pozostałych algorytmów systemu.

Na Rys. 5.6 przedstawiono porównanie wyników detektora OpenPose, uzyskanych podczas analizy obrazu zarówno przed, jak i po zastosowaniu opracowanej metody korekty jasności.

a) **b)**

Rys. 5.6 Wyniki detektora OpenPose uzyskane podczas analizy obrazu: a) oryginalnego o słabym oświetleniu $j = 73$, b) tego samego obrazu po zastosowaniu automatycznej korekty jasności $\gamma = 0.808$

Detektor OpenPose nie jest w stanie wykryć osoby stojącej przy prawej krawędzi obrazu (zob. Rys. 5.6a) z powodu zbyt niskiej ekspozycji światła w tym obszarze. Zastosowanie opracowanej korekty spowodowało, że osoba została prawidłowo wykryta przez detektor OpenPose (zob. Rys. 5.6b).

Podsumowując, opracowany algorytm, poprzez adaptacyjny mechanizm korekty współczynnika γ, automatycznie reguluje poziom jasności, wprowadzając stosowną zmianę zarówno w przypadku obrazów zaciemnionych jak i prześwietlonych. Opracowana przez autora zależność 5.5 umożliwia automatyczne dostosowanie intensywności korekty do średniej ważonej jasności obrazu, co przyczynia się do osiągnięcia uniwersalności zastosowania. Niski stopień skomplikowania algorytmu umożliwia szybką realizację obliczeń w czasie ok. 4 ms dla pojedynczego obrazu FHD. Zastosowanie opracowanego algorytmu umożliwia zwiększenie liczby detekcji typu *TP*, co udowodniono na konkretnym przykładzie (zob. Rys. 5.6).

5.3 Detekcja obiektów

W projektowanym systemie IVSS użyto czterech detektorów obiektów: OpenPose, YOLO v4, YOLO v4 tiny oraz Tesseract OCR. W Tab. 5.1 przedstawiono wykaz klas obiektów rozpoznawanych przez zastosowane detektory.

Tab. 5.1 Wykaz klas obiektów rozpoznawanych przez wybrane detektory

Detektor	Klasy obiektów
OpenPose	Człowiek
YOLO v4	Człowiek, pies, kot, samochód osobowy, samochód ciężarowy, autobus, motocykl
YOLO v4 tiny	Ogień, tablice rejestracyjne
Tesseract OCR	Znaki alfanumeryczne

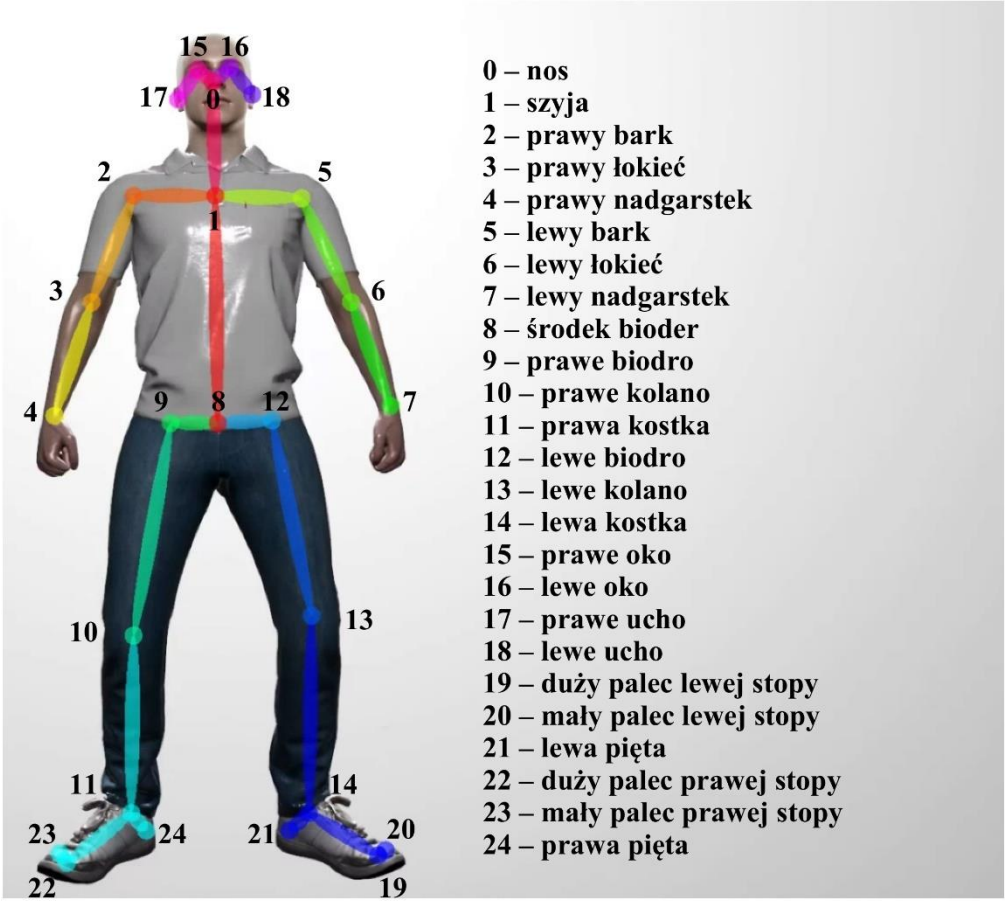
W tym podrozdziale zostaną omówione szczegóły dotyczące parametryzacji i wyboru konkretnych modeli detektorów, a także proces uczenia maszynowego modelu YOLO v4 tiny.

5.3.1 OpenPose

Autor postanowił użyć obecnie najnowszej dostępnej wersji biblioteki OpenPose v1.7.0 [72]. W ramach struktury projektowanego systemu IVSS biblioteka ta stanowi najbardziej rozbudowany i skomplikowany komponent. Złożona architektura biblioteki skupia w sobie obszerny zbiór plików źródłowych i nagłówkowych, definiujących m.in. funkcje przetwarzania wielowątkowego i obsługi akceleracji obliczeń z użyciem GPU. Z tego powodu autor postanowił, aby kod programu OpenPose pełnił funkcję szkieletu docelowej aplikacji projektowanego systemu IVSS, wokół którego konstruowana będzie cała logika operacyjna. Jest to strategiczna decyzja projektowa, podkreślająca dominującą rolę OpenPose w kontekście analizy ruchu i zachowań osób

na obrazie. Implementacja pozostałych bibliotek i opracowanych algorytmów będzie zatem stanowić rozszerzenie funkcjonalności OpenPose, zapewniając elastyczność i skalowalność systemu.

Autorzy biblioteki OpenPose udostępniają publicznie 4 wytrenowane modele, umożliwiające detekcję całego ciała [134]: BODY_25, COCO, MPI oraz MPI_4_layers. W projektowanym systemie IVSS autor rozprawy wybrał model BODY_25, ponieważ (zgodnie z opisem twórców [134,135]) model ten zapewnia najwyższą dokładność detekcji i najwyższą szybkość przetwarzania w przypadku użycia GPU wspieranych technologią CUDA w porównaniu do pozostałych dostępnych modeli. Na Rys. 5.7 przedstawiono graficzną reprezentację szkieletu kinematycznego, wygenerowanego przy użyciu modelu BODY_25, wraz z numeracją i opisem punktorów szkieletu [136].



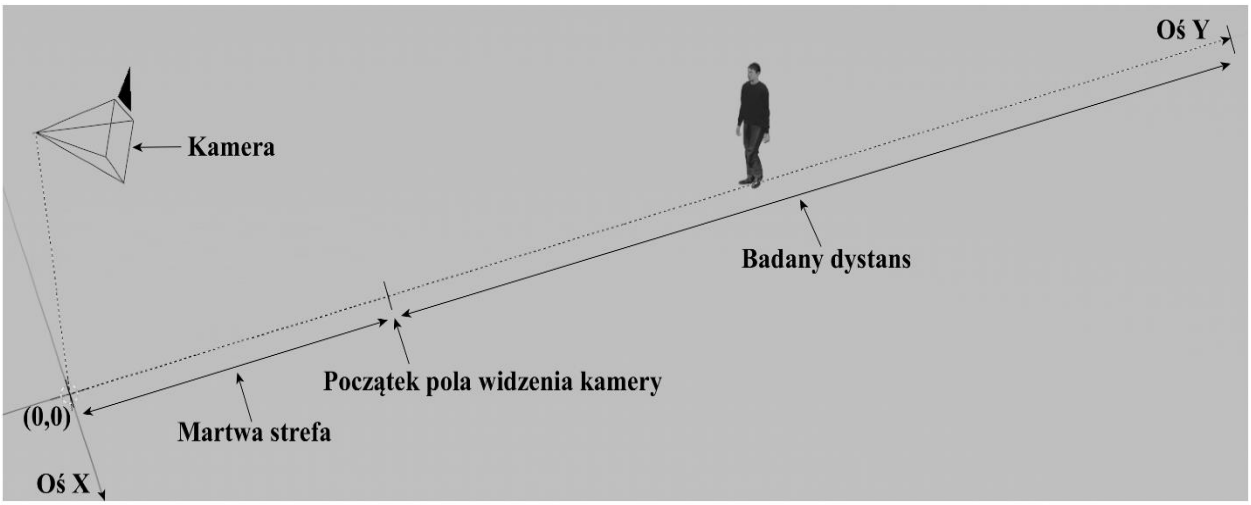
Rys. 5.7 Graficzna reprezentacja szkieletu kinematycznego wygenerowanego przy użyciu modelu BODY_25

Model BODY_25 umożliwia generowanie szkieletów składających się z 25 punktorów. Każdy reprezentuje trzy zmienne: współrzędne x oraz y punktu na obrazie, a także wskaźnik prawdopodobieństwa (pewności) wykrycia tego punktu. W projektowanym systemie IVSS autor postanowił korzystać tylko z 15 punktorów oznaczonych numeracją od 0 do 14. Jest to uzasadnione możliwością zapewnienia wymaganej szczegółowości analizowanego szkieletu przy jednoczesnej redukcji złożoności obliczeniowej. Punktory nie uwzględniane w docelowej analizie

szkieletu (15–24) zostaną użyte do oszacowania pozycji punktorów 0, 11 oraz 14 w przypadku wystąpienia braku ich detekcji.

Autor postanowił deaktywować domyślną funkcjonalność OpenPose odpowiedzialną za generowanie graficznej reprezentacji szkieletu na obrazie końcowym. Dzięki temu możliwy jest wzrost szybkości przetwarzania, a kolorowe znaczniki nie zakrywają osób znajdujących się na scenie, co umożliwia ich łatwiejszą identyfikację na obrazie. Wyłączenie tej funkcjonalności nie ma wpływu na dokładność generowanych detekcji, a ma charakter jedynie ilustracyjny.

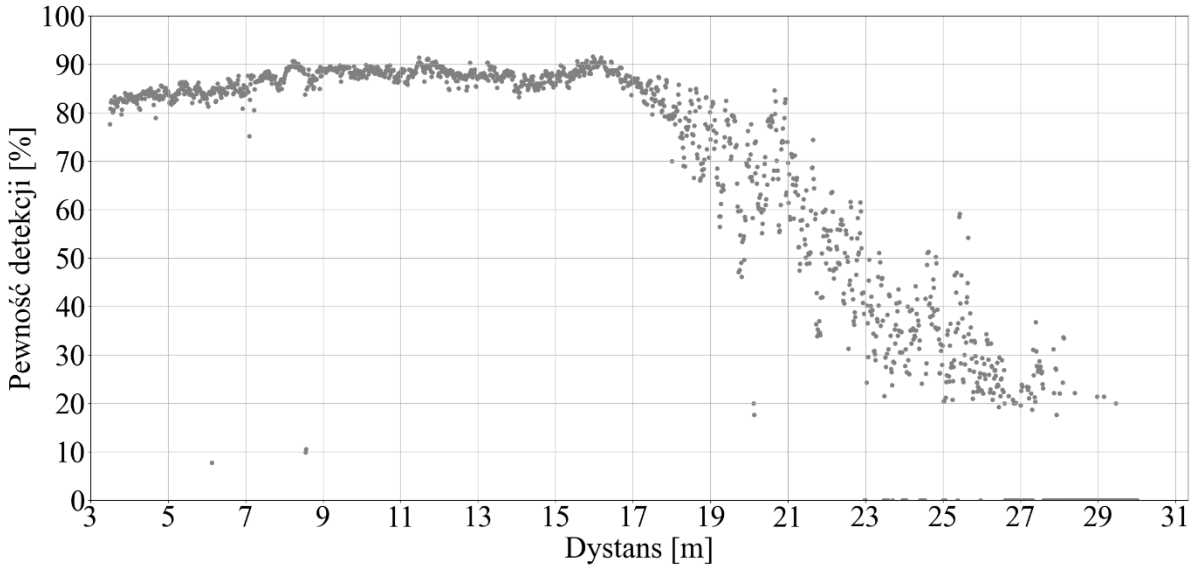
Skuteczność działania detektora OpenPose jest silnie skorelowana z odległością osoby od kamery. W celu określenia maksymalnej odległości, poniżej której pewność detekcji jest zadawalająca (minimum 75%), autor przeprowadził badanie empiryczne. Badanie te polegało na analizie nagrania wideo, na którym uczestnik poruszał się wzdłuż pionowej osi obrazu, oddalając się od kamery na dystansie 40 m. Na Rys. 5.8 przedstawiono symulację zaaranżowanej sceny.



Rys. 5.8 Widok sytuacyjny przebiegu testów detektora OpenPose

Kamera została zamontowana na wysokości 2.5 m nad podłożem, natomiast kąt nachylenia zapewniał pole widzenia umożliwiające realizację badania na dystansie od 3.5 do 40 m. Opracowane nagranie obejmowało 1675 ujęć w rozdzielczości 1920 × 1080 px. Wideo zostało poddane analizie przez detektor OpenPose przy zachowaniu domyślnych ustawień rozdzielczości wejścia sieci, a więc 656 × 368. Na każdym ujęciu testowym mógł zostać wykryty tylko jeden szkielet, dla którego obliczono średnią wartość pewności detekcji na podstawie obliczonych wskaźników pewności każdego z rozpatrywanych punktorów (0–14).

Na Rys. 5.9 przedstawiono wyniki przeprowadzonych badań w funkcji pewności detekcji od odległości filmowanej osoby od kamery. Z przedstawionego wykresu wynika, iż stabilny poziom pewności detekcji przekraczający 75% osiągany jest na dystansie poniżej 18 m. Po przekroczeniu tej odległości zauważalny jest spadek pewności detekcji oraz większy rozrzut wyników. Przeprowadzone badanie potwierdza tezę, iż skuteczność działania detektora OpenPose jest silnie skorelowana z odległością osoby od kamery.



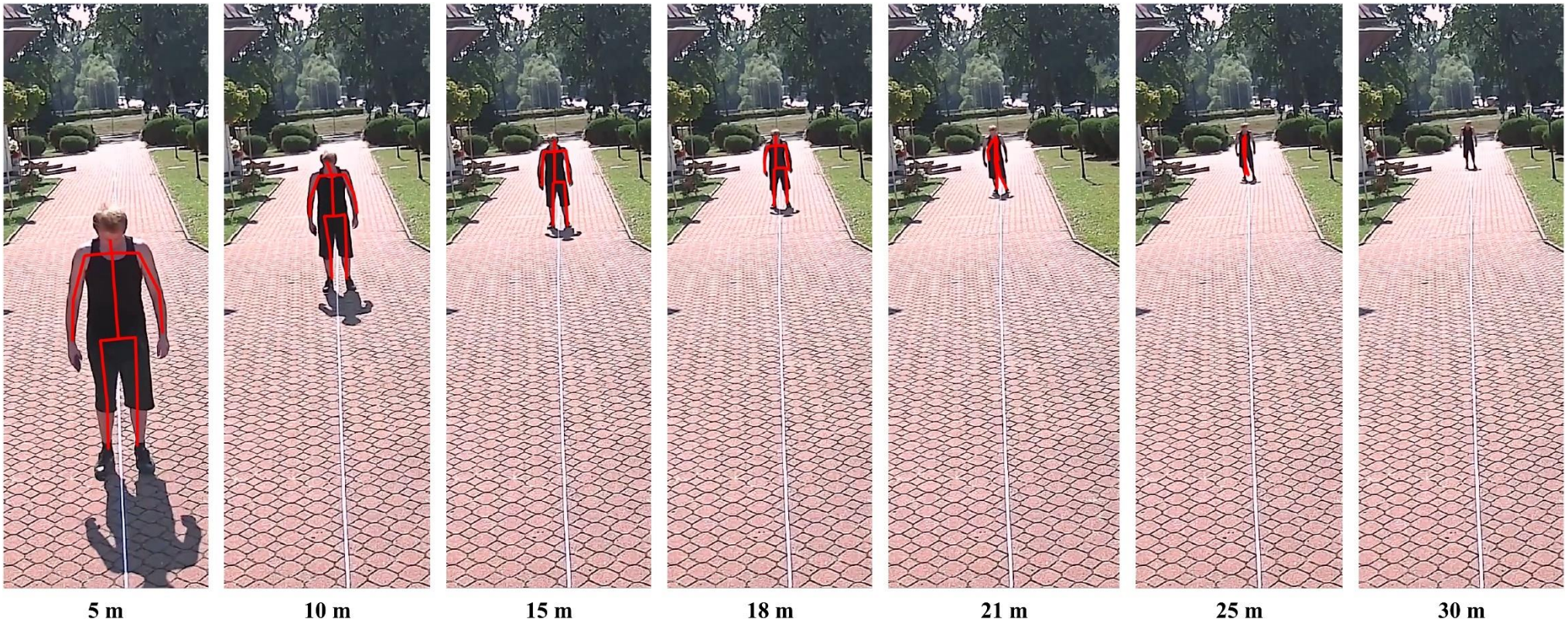
Rys. 5.9 Pewność detektora OpenPose w funkcji odległości osoby od kamery

Na Rys. 5.10 przedstawiono fragmenty ujęć nagrania testowego, które umożliwiają dokonanie wizualnej oceny wygenerowanego szkieletu filmowanej osoby. Zgodnie z informacją zawartą na Rys. 5.10, ujęcia odpowiadają następującym odległościom osoby od kamery: 5, 10, 15, 18, 21, 25, oraz 30 m. W zakresie odległości do 10 m można zauważyć poprawnie wygenerowany szkielet, a w zakresie powyżej 10 m pojawiające się błędy, deformacje szkieletu, nieprawidłowe lokalizacje punktorów lub ich brak. W zakresie pomiędzy 10 a 18 m wynik identyfikacji w dalszym ciągu można uznać za poprawny pomimo nieznacznych deformacji (np. brak symetrii) szkieletu.

Podsumowując, rezultaty przeprowadzonych badań dowodzą, iż efektywność działania detektora OpenPose w naturalnych warunkach oświetlenia dziennego i dobrych warunkach atmosferycznych jest zadowalająca. Pewność detekcji pogarsza się wraz z odległością osoby od kamery. Powyżej 18 m próba analizy sylwetek osób może prowadzić do błędnych wniosków. Problem ten może zostać rozwiązany poprzez programową kontrolę odległości osoby od kamery. Jeżeli osoba zostanie wykryta na dystansie przekraczającym ustaloną wartość progową wówczas system nie będzie jej dalej analizować. Zadanie to może zostać zrealizowane przy użyciu macierzy rzutowania 3D, opisanej w dalszych częściach niniejszej rozprawy.

5.3.2 YOLO v4 oraz YOLO v4 tiny

Autorzy YOLO v4 udostępniają szereg modeli, różniących się rozmiarem i liczbą parametrów sieci. Mniejsze modele (np. *yolov4-tiny*) charakteryzują się mniejszą dokładnością detekcji, natomiast umożliwiają uzyskanie szybszego przetwarzania danych na urządzeniach o mniejszej wydajności obliczeniowej. Większe modele (np. *yolov4*) są dokładniejsze, choć wymagają akceleracji sprzętowej do osiągnięcia przetwarzania danych w czasie rzeczywistym. Wszystkie modele w wersji 4 udostępniane przez autorów, są wytrenowane z użyciem popularnego zbioru danych MS-COCO [49].



Rys. 5.10 Wizualna ilustracja działania detektora OpenPose w funkcji odległości osoby od kamery

Dzięki temu umożliwiają detekcję i klasyfikację osiemdziesięciu klas typowych obiektów, takich jak samochód osobowy, pies, telewizor oraz detekcję osób. Adnotacja obiektów na obrazie realizowana jest za pomocą BB. Ponadto, autorzy udostępniają skrypty programowe, umożliwiające przeprowadzenie uczenia maszynowego, zorientowanego na własny zestaw danych treningowych. Dodatkowo, proces uczenia może zostać zrealizowany przy użyciu metody TL. Metoda ta umożliwia korzystanie z uprzednio wytrenowanych modeli, co sprawia, że osiągnięcie pożądanego stopnia wytrenowania nowego modelu jest znacznie szybsze. W projektowanym systemie IVSS postanowiono użyć zarówno oryginalnego modelu *yolov4*, jak również specjalnie wytrenowanego modelu *yolov4-tiny*.

W celu detekcji i klasyfikacji osób oraz pojazdów w projektowanym systemie IVSS, autor postanowił użyć oryginalnego modelu o nazwie *yolov4*. Model ten został pobrany z publicznego repozytorium [137] i składa się z dwóch plików: *yolov4.cfg* oraz *yolov4.weights*, z czego pierwszy zawiera informacje o architekturze sieci, natomiast drugi zawiera wytrenowane wagi modelu, będące wynikiem procesu uczenia maszynowego. Model *yolov4* został użyty do celu detekcji następujących klas obiektów: „*człowiek*”, „*kot*”, „*pies*”, „*samochód osobowy*”, „*samochód ciężarowy*”, „*autobus*” oraz „*motocykl*”, przy czym finalnie wyniki detekcji czterech ostatnich klas są interpretowane jako pojedyncza klasa o nazwie „*pojazd*”.

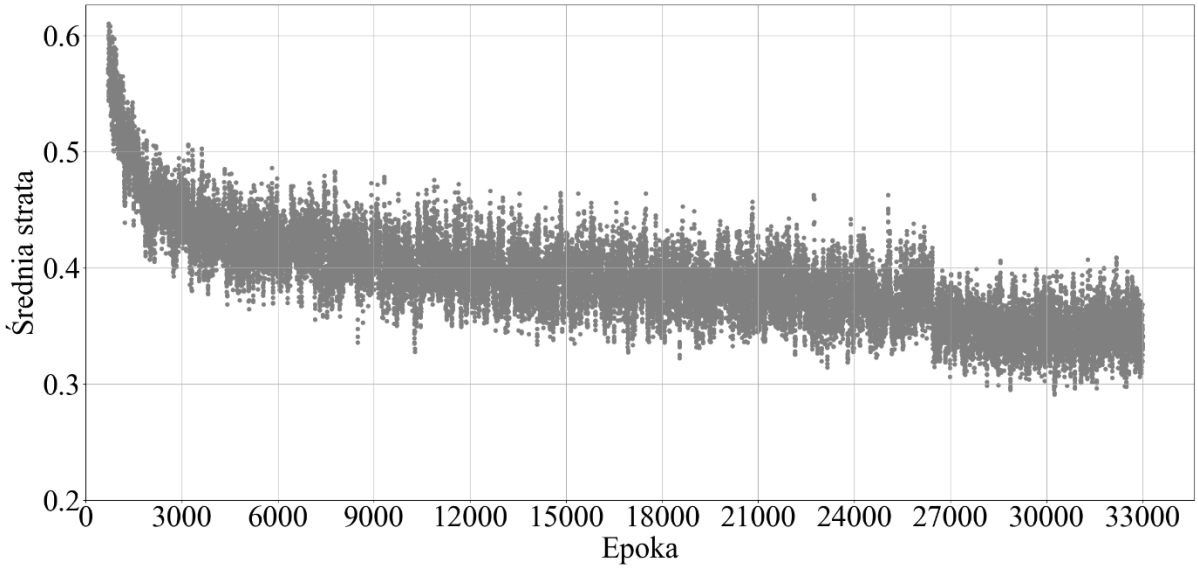
Dostępne wytrenowane modele YOLO v4 nie są zdolne do detekcji obiektów klas „*tablica rejestracyjna*” oraz „*ogień*”, których detekcja jest wymagana w celu osiągnięcia pożądanej funkcjonalności projektowanego systemu IVSS. Dlatego, autor rozprawy przeprowadził proces uczenia maszynowego dodatkowego modelu YOLO v4 o nazwie *yolov4-tiny*. Model ten w porównaniu do *yolov4* charakteryzuje się znacznie mniejszą złożonością architektury sieci, co w konsekwencji przekłada się na znacznie krótszy czas realizacji uczenia maszynowego, jak i przetwarzania obrazu przez wytrenowany model. W procesie uczenia, autor użył gotowe bazy danych, które zostały pobrane z platformy internetowej Roboflow [52]. Charakterystykę ilościową bazy danych uczących przedstawiono w Tab. 5.2.

Tab. 5.2 Charakterystyka ilościowa bazy danych uczących modelu *yolov4-tiny*

Klasa obiektów	**Zbiór uczący [l. obrazów]**	**Zbiór walidacyjny [l. obrazów]**
Tablice rejestracyjne	37056	8824
Ogień	35800	11422

Proces uczenia maszynowego został zrealizowany przy użyciu dedykowanej biblioteki programistycznej udostępnionej wraz z dokładną instrukcją obsługi przez twórców YOLO v4 w publicznym repozytorium [137]. Zgodnie z tą instrukcją, trening został poprzedzony

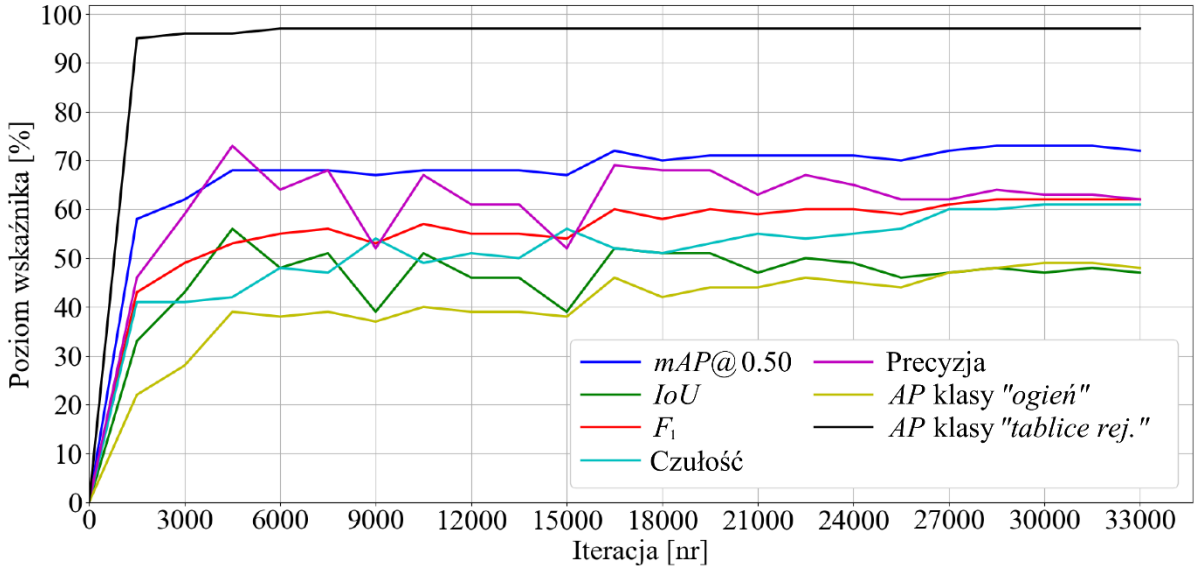
przygotowaniem stosownych plików konfiguracyjnych: *obj.names*, *obj.data*, *train.txt* oraz *valid.txt*. Dalej, w celu dostosowania architektury sieci do realizacji detekcji dwóch klas obiektów w pliku *yolov4-tiny-custom.cfg*, wprowadzono następujące modyfikacje: *batch* = 64, *subdivisions* = 16, *max_batches* = 33000, *steps* = (26400,29700), *width* = 416, *height* = 416, *filters* = 21 oraz *classes* = 2. W celu realizacji procesu uczenia typu TL, autor użył uprzednio wytrenowanego modelu bazowego *yolov4-tiny.conv.29*, pobranego z repozytorium [137]. Dodatkowo, autor rozprawy dokonał modyfikacji kodu źródłowego realizującego proces uczenia maszynowego w celu umożliwienia zapisu wag modelu co 100 kolejnych epok. Trening modelu został zrealizowany dla 33000 epok i trwał ok. 7.5 godziny. Przebieg procesu uczenia modelu YOLO v4 tiny przedstawiono na Rys. 5.11 w formie wykresu, ilustrującego zmiany wartości średniej straty w funkcji liczby epok.



Rys. 5.11 Przebieg procesu uczenia modelu YOLO v4 tiny

W procesie uczenia zastosowano domyślną funkcję strat CIoU [33], która uwzględnia zarówno błędy lokalizacyjne, jak i odległości między prognozowanymi a rzeczywistymi BB obiektów. We wstępnym etapie procesu, wartość średniej straty była stosunkowo wysoka, co wynika z początkowego zainicjowania wag modelu. W miarę postępu uczenia zaobserwowano systematyczny spadek tej wartości, wskazując na poprawę jakości dopasowania wag modelu. Wartość średniej straty osiągnięta po zakończeniu procesu uczenia wynosi ok. 0.34. Zgodnie z zaleceniami twórców YOLO v4 [137], finalna wartość średniej straty w granicach od 0.05 (dla niewielkich modeli i nieskomplikowanych zbiorów danych) do 3.00 (dla dużych modeli i skomplikowanych zbiorów danych) wskazuje na odpowiednią jakość dopasowania wag modelu. Osiągnięta wartość 0.34 mieści się w dolnym zakresie tego przedziału, co świadczy o zadawalającej jakości rezultatów procesu uczenia. Osiągnięty wynik może wskazywać, iż zarówno zbiory danych, jak i parametry modelu zostały dobrane w sposób prawidłowy.

W wyniku przeprowadzonego procesu uczenia otrzymano 330 zestawów wag modelu, zapisywanych co 100 epok. W celu oceny zdolności modelu do generalizacji przeprowadzono badanie z użyciem zbioru walidacyjnego, który nie był wykorzystywany podczas uczenia. Każdy zapisany zestaw wag został poddany ocenie pod kątem pięciu kryteriów: *mAP*@50, *IoU*, F_1, czułość, precyzja, a także *AP* opracowane osobno dla klasy „*tablice rejestracyjne*" oraz „*ogień*". Na Rys. 5.12 przedstawiono charakterystykę przeprowadzonego badania.



Rys. 5.12 Charakterystyka wyników ewaluacji zestawów wag przeprowadzona na zbiorze walidacyjnym

Analiza opracowanych wyników wskazuje, iż model stosunkowo szybko osiągnął wysoką wartość *AP* dla klasy „*tablice rejestracyjne*" równą 95% przy 1500 iteracjach, natomiast klasa „*ogień*" przy tej samej liczbie iteracji osiągnęła wartość *AP* na poziomie zaledwie 22%. Znacząca różnica tych wartości utrzymywała się w czasie całego procesu uczenia. Powodem tego może być wysoki stopień skomplikowania reprezentacji graficznej obiektu, jakim jest ogień na obrazie 2D. W naturalnych warunkach ogień może występować w szerokim spektrum rozmiarów, kształtów, a nawet kolorów, zwłaszcza w przypadku spalania różnych materiałów czy okluzji wywołanych towarzyszącym zadymieniem. Z kolei tablice rejestracyjne charakteryzują się regularnym kształtem i rozmiarem, a często również kolorem, co pozytywnie wpływa na proces uczenia detekcji i klasyfikacji obiektów tego typu. Dla klasy „*tablice rejestracyjne*" osiągnięto maksymalną wartości *AP* równą 97% w epoce 6000, natomiast dla klasy „*ogień*" maksymalną wartość *AP* równą 49% osiągnięto w epoce 30000. W celu uniknięcia zjawiska tzw. „*przetrenowania modelu*", zwłaszcza w odniesieniu do klasy „*tablice rejestracyjne*", autor wybrał zestaw wag zapisanych w epoce 16500. W tym zestawie *AP* wyznaczone dla klasy „*tablice rejestracyjne*" przyjmuje maksymalną wartość (97%), natomiast dla klasy „*ogień*" przyjmuje wartość 46%, a więc zbliżoną do maksymalnej. Wybrany zestaw wag zostanie następnie użyty w celu opracowania algorytmów identyfikacji pojazdów oraz detekcji pożaru.

5.3.3 Tesseract OCR

Detekcja znaków alfanumerycznych przy użyciu Tesseract OCR może być realizowana w przy zastosowaniu jednej z czternastu metod segmentacji obrazu (PSM) [138]. Opis poszczególnych metod zestawiono w Tab. 5.3.

Tab. 5.3 Opis metod PSM detektora Tesseract OCR [138]

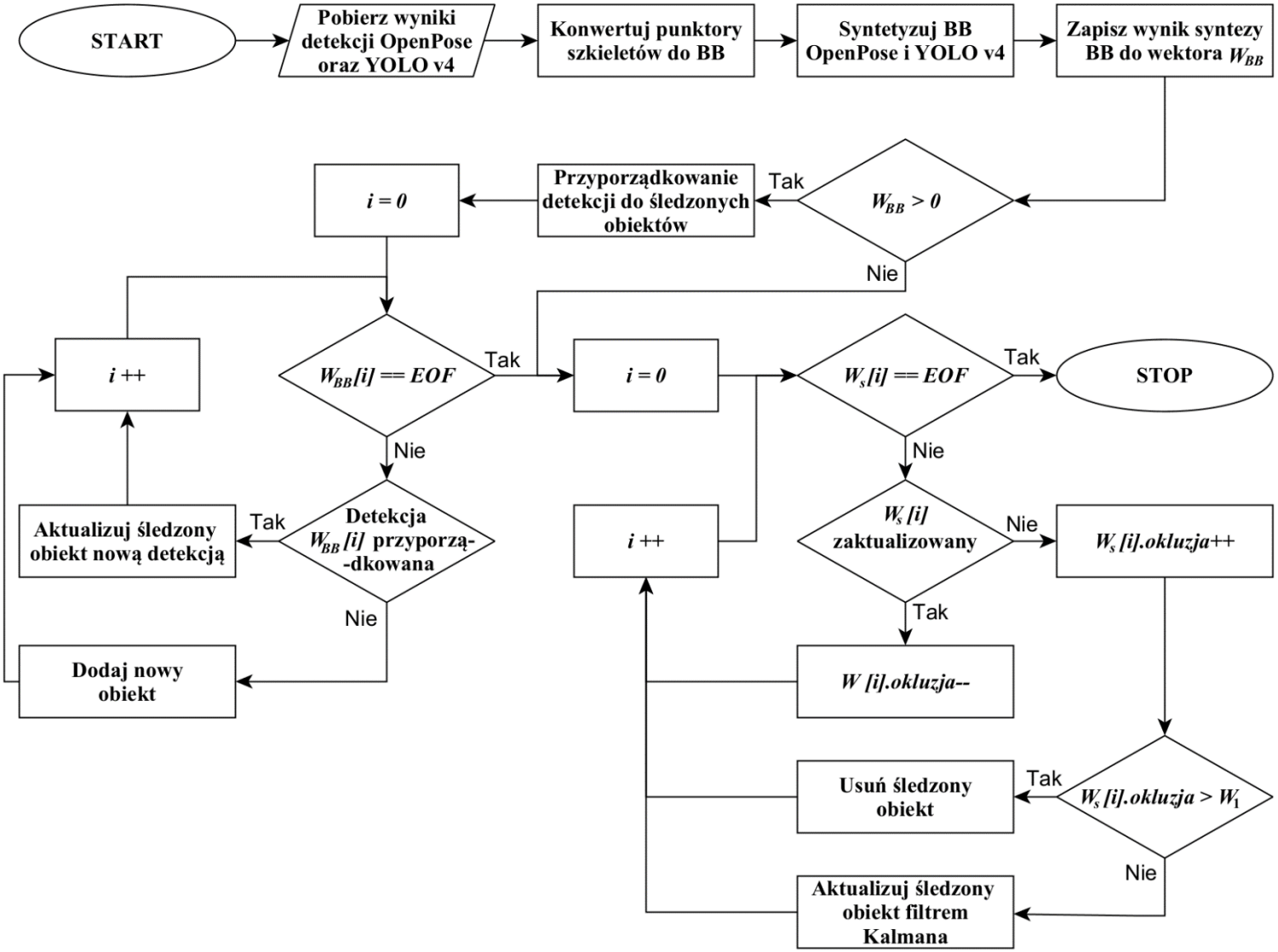
Metoda PSM	Opis
0	Detekcja OSD (bez OCR)
1	Automatyczna segmentacja obrazu na bloki tekstu + OSD (bez OCR)
2	Automatyczna segmentacja obrazu (bez OSD i OCR)
3	Automatyczna segmentacja obrazu + OCR (bez OSD)
4	Interpretacja obrazu jako pojedynczej kolumny tekstu o różnych rozmiarach
5	Interpretacja obrazu jako pojedynczego bloku tekstu wyrównanego pionowo
6	Interpretacja obrazu jako pojedynczego bloku tekstu
7	Interpretacja obrazu jako pojedynczej linii tekstu
8	Interpretacja obrazu jako pojedynczego słowa
9	Interpretacja obrazu jako pojedynczego słowa ułożonego w kształcie koła
10	Interpretacja obrazu jako pojedynczego znaku
11	Detekcja maksymalnej liczby tekstu rozproszonego, bez zachowania konkretnej kolejności
12	Detekcja maksymalnej liczby tekstu rozproszonego, bez zachowania konkretnej kolejności + OSD
13	Interpretacja obrazu jako pojedynczej linii tekstu z pominięciem technik optymalizacyjnych Tesseract

Detektor Tesseract OCR przyjmuje PSM 3 jako metodę domyślną. W projektowanym systemie IVSS rozpoznawanie znaków alfanumerycznych będzie realizowane wyłącznie do analizy tablic rejestracyjnych jednorzędowych, zawierających pojedynczą linię dwuczłonowego tekstu. Na podstawie informacji zawartych w Tab. 5.3 można stwierdzić, iż dla tekstu w takiej formie najbardziej odpowiednią metodą pracy detektora jest PSM 7, dotyczącą interpretacji obrazu jako pojedynczej linii tekstu.

5.4 Śledzenie obiektów

W projektowanym systemie IVSS implementacja algorytmu MOT została realizowana przy użyciu syntezy filtra Kalmana i logiki przyporządkowania detekcji, biorąc pod uwagę kryterium minimalnej zmiany położenia przez identyfikowany obiekt w sekwencji ujęć. Schemat blokowy algorytmu przedstawiono na Rys. 5.13.

W pierwszym kroku pobierane są detekcje OpenPose oraz YOLO v4 dla obiektów klasy „*człowiek*”. Detektor OpenPose dostarcza bardziej szczegółowych informacji o sylwetce człowieka, dlatego pełni funkcję detektora nadrzędnego, natomiast YOLO v4 pomocniczego. Następnie, dla każdego modelu szkieletowego OpenPose generowana jest adnotacja typu BB.



Rys. 5.13 Schemat blokowy algorytmu MOT, gdzie: *EOF* – znacznik końca wektora, *i* – licznik pętli iteracyjnych, W_{BB} – wektor nowych detekcji BB, W_s – wektor śledzonych obiektów, W_1 – wartość progowa maksymalnej liczby okluzji obiektu

Detekcje obu detektorów cechują ramki BB o podobnym rozmiarze i lokalizacji. W celu uniknięcia powtórzeń detekcji, w kolejnym kroku realizowana jest operacja łączenia ramek obu detektorów, biorąc pod uwagę kryteria: minimalnej odległości centroid ramek z obu detektorów oraz stopnia przenikania się ramek. Finalnie, wektor detekcji W_{BB} stanowi kopię ramek wygenerowanych z OpenPose rozszerzoną o detekcje YOLO v4, które nie znalazły dopasowania.

W następnym kroku przeprowadzana jest operacja przyporządkowania nowych detekcji do śledzonych obiektów. Polega ona na porównaniu lokalizacji wszystkich par śledzonych obiektów i nowych detekcji w celu wyznaczenia par o najmniejszej różnicy odległości na obrazie. Lokalizacja odnosi się do współrzędnych środkowego punktu górnej linii ramki. Wybór tego punktu jako referencyjnego pozwala uśrednić szerokość ramki poprzez wskazanie domniemanej osi symetrii osoby na zdjęciu, przy jednoczesnym wykluczeniu wpływ zmian wysokości ramki na kolejnych ujęciach. Zmiany wysokości ramki, o których mowa, to powszechne zjawisko występujące w sytuacji, gdy trajektoria ruchu co najmniej dwóch osób przecina się po rzutowaniu, a scena obserwowana jest z wysokości powyżej przeciętnego wzrostu człowieka. Problem ten zostanie szczegółowiej omówiony w dalszych częściach niniejszej rozprawy.

Opracowany algorytm umożliwia identyfikację czterech przypadków: utworzenie nowego śledzonego obiektu, aktualizację położenia śledzonego obiektu nową detekcją, aktualizację położenia śledzonego obiektu przy użyciu predykatora oraz usunięcie obiektu z procedury śledzenia w przypadku utrzymującego się braku detekcji.

Aktualizacja śledzonego obiektu nową detekcją ma miejsce, gdy nowa detekcja została pozytywnie przyporządkowana do któregoś z uprzednio śledzonych obiektów. Aktualizacja polega wówczas na zapisaniu nowych lokalizacji wzorcowego punktu śledzenia oraz aktualnego BB. Nowy wzorcowy punkt śledzenia jest zatem używany również do aktualizacji filtru Kalmana. Ponadto, przy użyciu macierzy rzutowania 3D, obliczane są dystans oraz nowa szybkość przemieszczania obiektu na scenie. Jeśli nowa detekcja zawiera punktory szkieletu, to również są one zapisywane.

Aktualizacja śledzonego obiektu predykcją ma miejsce, gdy żadna nowa detekcja nie została pozytywnie przyporządkowana do uprzednio śledzonego obiektu. Aktualizacja polega wówczas na zapisaniu nowej lokalizacji wzorcowego punktu śledzenia wyznaczonego przy użyciu predykcji filtru Kalmana. Stan braku przyporządkowania nowej detekcji jest definiowany jako całkowita okluzja i może powtarzać się w sekwencji nie dłuższej, niż zostało to arbitralnie określone wartością progową W_1. Przekroczenie wartości progowej definiowane jest jako opuszczenie obszaru monitoringu przez obiekt i skutkuje trwałym usunięciem obiektu z analizy śledzenia w kolejnych ujęciach.

Utworzenie nowego śledzonego obiektu ma miejsce, gdy nowa detekcja nie została pozytywnie przyporządkowana do żadnego uprzednio śledzonego obiektu. Wówczas zapisana

zostaje nowa lokalizacja wzorcowego punktu śledzenia, nowa adnotacja BB oraz zostaje nadany nowy unikalny identyfikator obiektu. Do każdego śledzonego obiektu tworzony jest indywidualny model filtru Kalmana, mający za zadanie predykcję lokalizacji obiektu w przypadku wystąpienia chwilowej pełnej okluzji, a więc braku przyporządkowania nowych detekcji. W tym przypadku obliczanie dystansu oraz szybkości przemieszczania obiektu na scenie jest niemożliwe, ponieważ istnieje tylko jedna zapisana lokalizacja. Jeśli nowa detekcja zawiera punktory szkieletu, to również są one zapisywane.

W implementacji filtru Kalmana, uwzględniono specyfikę danych wejściowych oraz dynamikę ruchu obiektów w przestrzeni 2D. Filtr ten wykorzystuje wektor stanu, który dotyczy pozycji (x,y) oraz prędkość (v_x,v_y) obiektu, umożliwiając estymację jego trajektorii na podstawie szeregów czasowych danych pomiarowych. Macierz przejścia została zaprojektowana tak, aby uwzględniała ruch prostoliniowy z założeniem stałego kroku czasowego, wyrażonego jako odwrotność liczby klatek na sekundę ($1/FPS$). Taka konstrukcja pozwala na efektywne modelowanie dynamiki ruchu przy założeniu, iż prędkość obiektu pozostaje stała między kolejnymi obrazami. Macierz pomiaru odzwierciedla fakt, iż system pomiarowy dostarcza wyłącznie informacji o pozycji obiektu, co wyrażono poprzez redukcję wymiarowości macierzy do komponentów odpowiadających pozycjom (x,y). W celu uwzględnienia niepewności w modelu dynamiki ruchu, macierz kowariancji szumu procesu została zdefiniowana jako macierz jednostkowa, skalowana wartością 10^{-2}. Parametr ten wprowadza umiarkowaną dyspersję modelu, pozwalając na elastyczne dopasowanie do potencjalnych zakłóceń. Analogicznie, macierz kowariancji szumu pomiaru wyznaczono jako macierz jednostkową, skalowaną wartością 10^{-1}, co odzwierciedla ograniczoną precyzję pomiarów pozycji obiektów, generowanych przez detektory OpenPose oraz YOLO v4. Początkowy wektor stanu został zainicjowany na podstawie pozycji obiektu, przyjmując zerową wartość prędkości w obu osiach ($v_x = v_y = 0$). Wartości macierzy kowariancji błędu początkowego ustawiono jako macierz jednostkową, co stanowi standardowe założenie w sytuacji ograniczonej wiedzy o obiekcie w momencie jego inicjalizacji. Opisany dobór parametrów filtru Kalmana umożliwia estymację trajektorii obiektów w warunkach częściowo zaszumionych lub niekompletnych danych pomiarowych. Zaproponowane rozwiązanie znacząco zwiększa skuteczność śledzenia w przypadkach całkowitej okluzji obiektu.

Na Rys. 5.14 przedstawiono wizualizację rezultatów działania opracowanego algorytmu śledzenia MOT. W analizowanej przestrzeni monitoringowej znajdują się trzy osoby. Opracowany algorytm śledzenia MOT prawidłowo oznacza każdą z nich za pomocą adnotacji typu BB oraz unikatowego identyfikatora liczbowego. W celu ułatwienia identyfikacji osób na obrazie, algorytm przypisuje indywidualne kolory adnotacji dla każdej śledzonej sylwetki. Dodatkowo, wyświetlane są ścieżki przejścia śledzonych osób w formie znaczników punktowych. Znaczniki te dotyczą zapamiętanych lokalizacji wzorcowego punktu śledzenia, a więc środka górnej krawędzi BB.

Rys. 5.14 Wizualizacja rezultatów pracy algorytmu śledzenia MOT

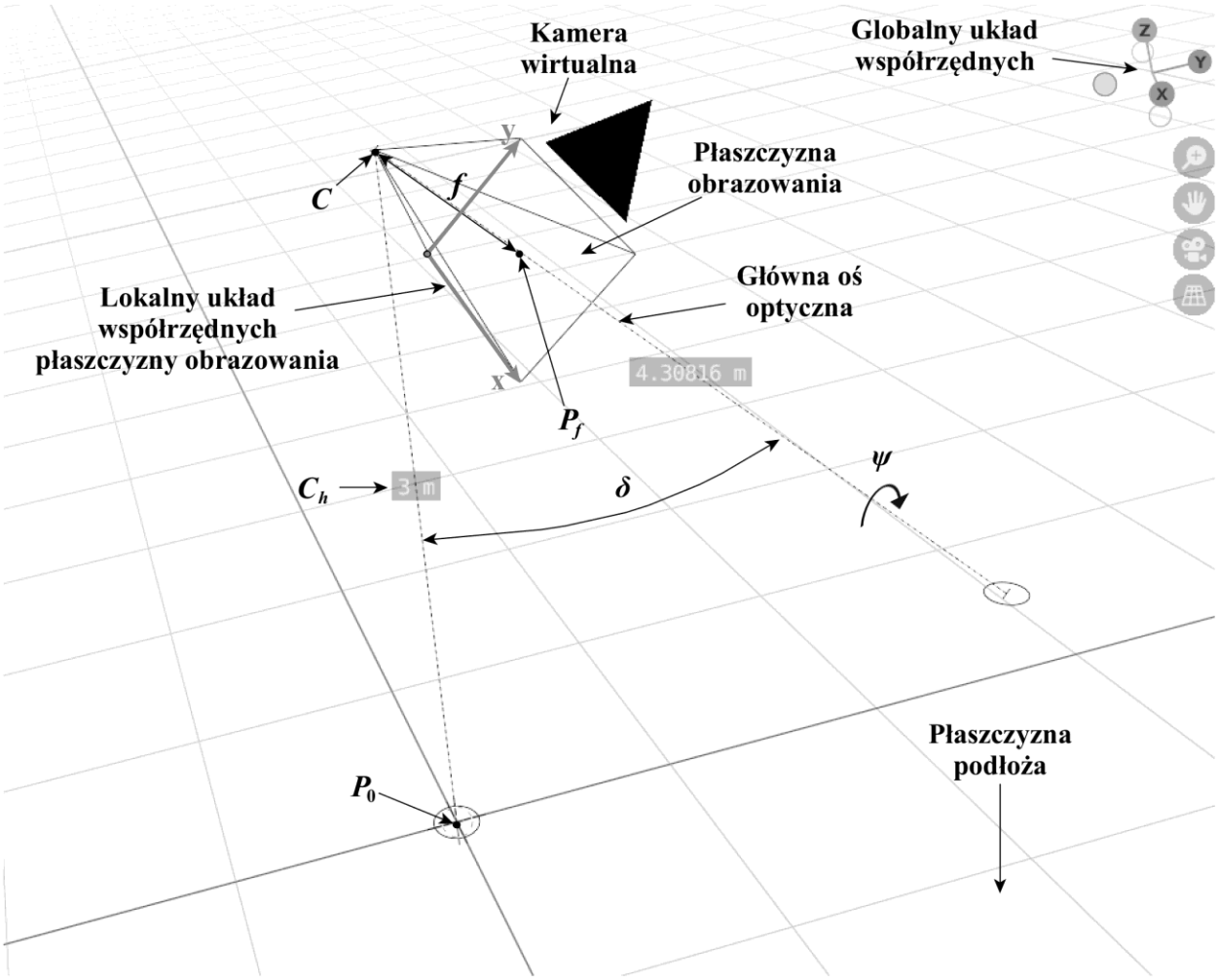
Podsumowując, zaprojektowany algorytm śledzenia MOT opiera się na syntezie wyników filtru Kalmana oraz detekcji najkrótszego dystansu pomiędzy obiektami w sekwencji ujęć. Opracowany algorytm umożliwia identyfikację czterech przypadków: utworzenie nowego śledzonego obiektu, aktualizację położenia śledzonego obiektu nową detekcją, aktualizację położenia śledzonego obiektu przy użyciu predykatora (w przypadku braku dopasowania nowej detekcji) oraz usunięcie obiektu z procedury śledzenia w przypadku utrzymującego się braku dopasowania detekcji. Przeprowadzone testy w większości przypadków wykazują zadawalającą skuteczność pracy algorytmu. Obserwacje błędnej realizacji zadania śledzenia wynikały najczęściej z braku detekcji osób na obrazie lub okluzji, podczas których osoba w sposób znaczący zmienia kierunek ruchu. Zaproponowane rozwiązanie jest wysoce uniwersalne, gdyż umożliwia realizację śledzenie różnych klas obiektów. Jednak w projektowanym systemie IVSS, ze względu na specyficzne wymagania, algorytm ten zostanie użyty wyłącznie do śledzenia osób na obrazie.

5.5 Kalibracja sceny

Detekcje generowane przy użyciu OpenPose oraz YOLO v4 umożliwiają oszacowanie lokalizacji wykrytych obiektów w układzie współrzędnym obrazu 2D. Dane te są jednak niewystarczające do oszacowania rzeczywistych parametrów ruchu wykrytych obiektów przy obserwacji sceny z różnych perspektyw, kątów widzenia czy odległości i wysokości montażu kamery. Estymacja rzeczywistych parametrów ruchu wymaga zatem implementacji metod, umożliwiających transformację współrzędnych obrazu 2D do rzeczywistego układu 3D

analizowanej sceny, uwzględniającej parametryzację kamery i jej orientację w przestrzeni. W celu realizacji tego zadania, autor opracował dedykowany algorytm kalibracji sceny.

W celu wizualizacji logiki działania opracowanego algorytmu kalibracji sceny, autor posłuży się symulacją komputerową stworzoną w programie Blender. Na Rys. 5.15 przedstawiono widok poglądowy symulowanej sceny.



Rys. 5.15 Widok poglądowy symulowanej sceny, gdzie: C – punkt główny kamery, f – ogniskowa kamery, P_f – punkt przecięcia ogniskowej z płaszczyzną obrazowania, C_h – wysokość montażu kamery nad płaszczyzną podłoża, δ – kąt nutacji (nachylenia kamery do płaszczyzny podłoża), ψ – kąt obrotu własnego kamery (względem głównej osi optycznej), P_0 – punkt początkowy globalnego układu współrzędnych

Jednostka globalnego układu współrzędnych to metr. Osie X oraz Y globalnego układu współrzędnych tworzą płaszczyznę podłoża, natomiast oś Z to wysokość nad tą płaszczyzną. Punkt C zawsze zlokalizowany jest w początku globalnego układu współrzędnych na wysokości C_h i dotyczy punktu montażu obiektywu kamery. Proponowana metoda będzie umożliwiać przeprowadzenie procesu kalibracji sceny niezależnie od jej kąta obrotu względem osi optycznej (ψ) oraz kąta nutacji (δ). Model kamery wirtualnej został sparametryzowany w celu odwzorowania parametrów rzeczywistej kamery używanej w testach.

Opracowany algorytm kalibracji sceny, w przeciwieństwie do metod referencyjnych [106], nie wymaga użycia specjalistycznego sprzętu kalibrującego. Wymagane jest natomiast

wprowadzenie do programu podstawowych parametrów używanej kamery, takich jak wartości ogniskowej oraz przekątnej sensora optycznego. Następnie algorytm automatycznie wykrywa wartości wysokości (*height*) oraz szerokości (*width*) analizowanego obrazu w pikselach. W dalszym kroku realizowane są wstępne obliczenia i przekształcenia wprowadzonych wielkości.

Najpierw obliczana jest proporcja analizowanego obrazu (*AR*) stanowiąca stosunek jego szerokość i wysokość

$$AR = \frac{width}{height} \tag{5.6}$$

W następnym kroku realizowana jest konwersja przekątnej czujnika optycznego z cali na milimetry (D_{mm})

$$D_{mm} = 25.4 sensor_size \tag{5.7}$$

W kolejnym kroku, zgodnie z informacjami zawartymi w [139], algorytm dokonuje korekty D_{mm}

$$D_{mm} = \frac{D_{mm}}{k} \tag{5.8}$$

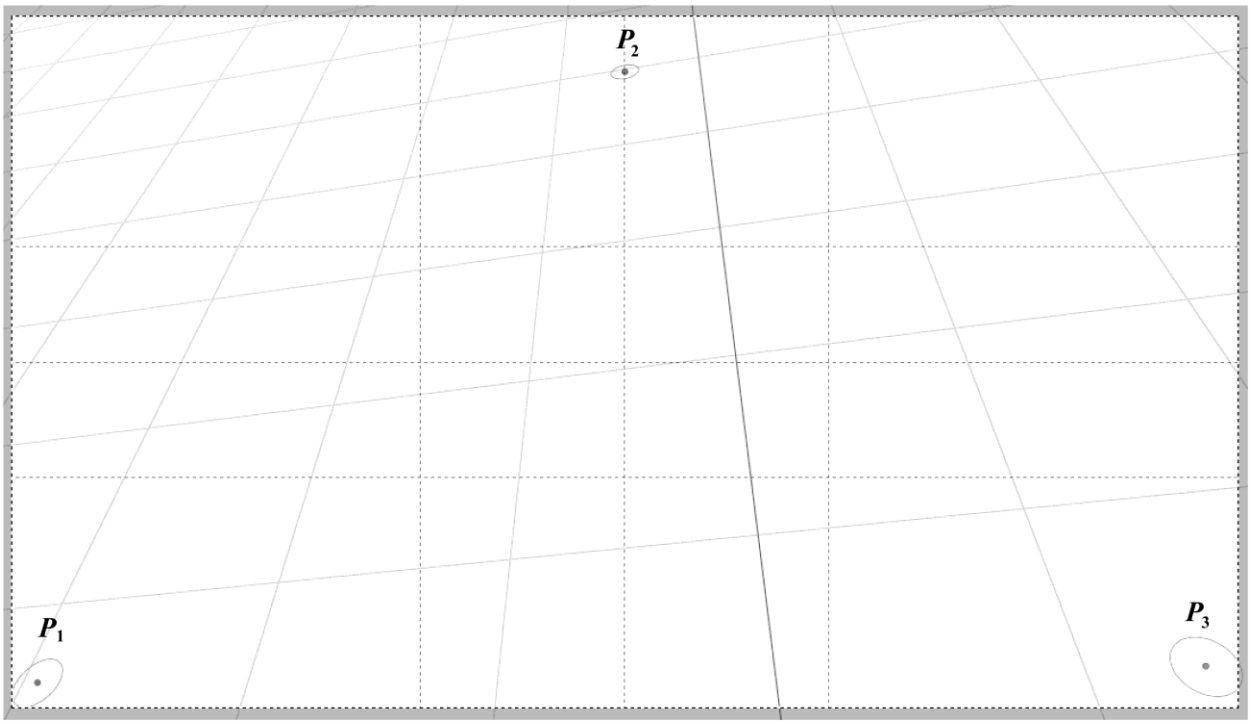
gdzie współczynnik korekty *k* może przyjmować dwie wartości:

- dla *sensor_size* < 0.5 współczynnik *k* = 1.411,
- dla *sensor_size* ≥ 0.5 współczynnik *k* = 1.5875.

Następnie obliczane są wartości wysokości (H_{mm}) i szerokości (W_{mm}) czujnika optycznego kamery w milimetrach. Wartości te wyznaczane są z zależności geometrycznych w trójkącie prostokątnym, którego przekątna D_{mm} oraz proporcje boków są znane

$$\begin{cases} W_{mm} = \sqrt{\dfrac{{D_{mm}}^2}{1 + \dfrac{1}{AR^2}}} \\ H_{mm} = \dfrac{W_{mm}}{AR} \end{cases} \tag{5.9}$$

W kolejnym etapie pracy użytkownik wybiera trzy punkty referencyjne (P_1, P_2 oraz P_3) na obrazie, których rzutami na podłożu obserwowanej sceny są punkty P_1', P_2' oraz P_3'. Wybór punktów jest realizowany manualnie poprzez zastosowanie GUI, w którego oknie wyświetlany jest obraz, stanowiący podgląd obserwowanej sceny w czasie rzeczywistym. Symulację realizacji tego zadania przedstawiono na Rys. 5.16. Wybór lokalizacji punktów referencyjnych jest dowolny, co zapewnia uniwersalność zastosowania algorytmu. W celu uwzględnienia kątów δ



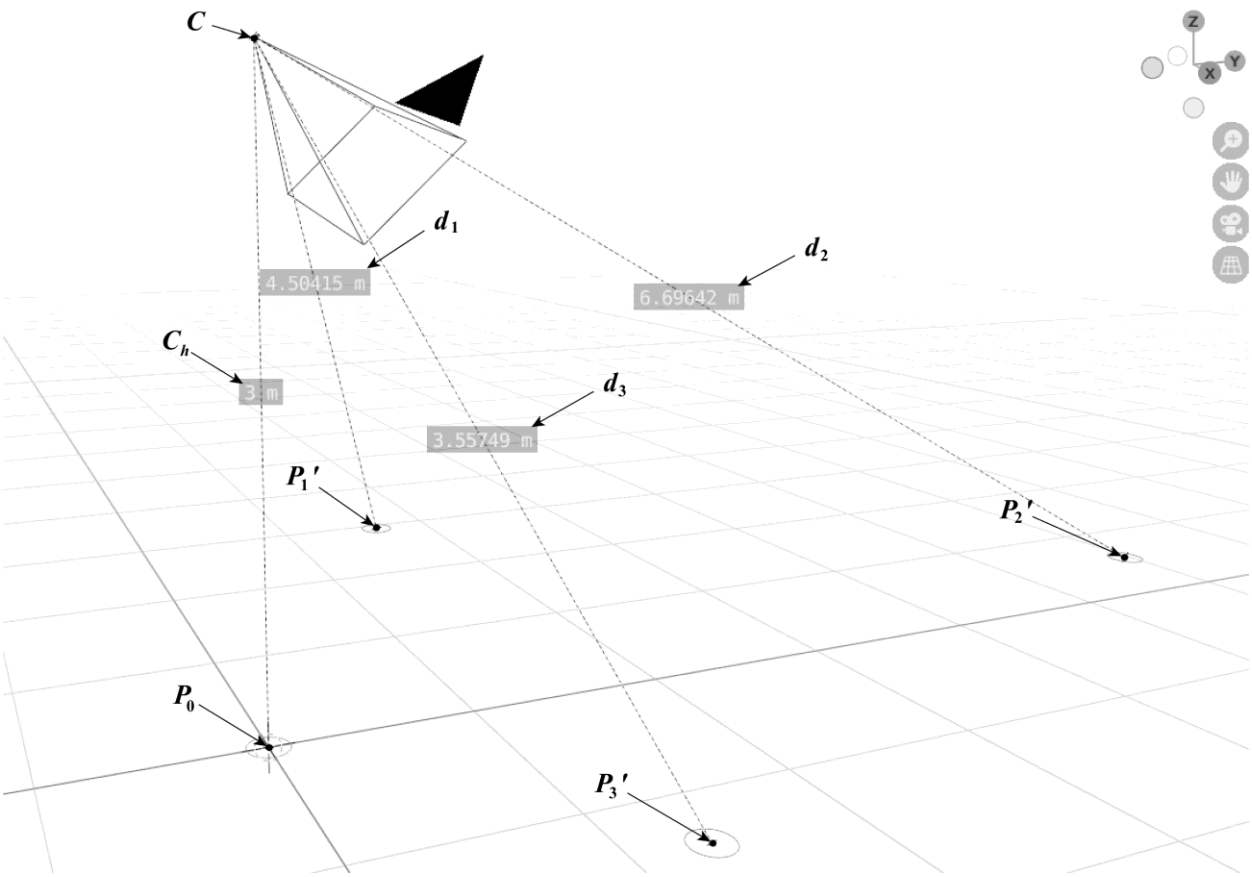
Rys. 5.16 Wybór punktów referencyjnych P_1, P_2 oraz P_3 w widoku obrazu z kamery wirtualnej

oraz ψ w dalszych etapach pracy zaleca się jednak zachowanie możliwie dużej odległości pomiędzy punktami np. poprzez oznaczanie ich w pobliżu krawędzi obrazu (zob. Rys. 5.16).

W kolejnych krokach wyznaczane są odległości od punktu C do punktu P_1' (d_1), P_2' (d_2) oraz punktu P_3' (d_3), a także wysokość kamery względem podłoża (C_h). Pomiary te są realizowane w sposób manualny przez użytkownika przy użyciu dowolnego narzędzia pomiarowego np. uniwersalnego dalmierza laserowego. Zmierzone wartości d_1, d_2, d_3 oaz C_h należy wprowadzić do programu w jednostce metr. Na Rys. 5.17 przedstawiono przykład realizacji pomiarów w symulowanej scenie.

Następnie obliczane są współrzędne globalne punktu P_f. Jest to możliwe po przyjęciu następujących założeń:

- C to punkt początkowy ogniskowej (f), natomiast P_f jest jej punktem końcowym,
- ogniskowa (f) jest linią równoległą do osi Y i prostopadłą do osi X oraz Z globalnego układu współrzędnych,
- ogniskowa (f) jest prostopadła do płaszczyzny obrazowania, a jej punkt końcowy odzwierciedla geometryczny środek macierzy obrazu,
- lokalna oś y płaszczyzny obrazowania jest równoległa do osi Z globalnego układu współrzędnych i przyjmuje jej zwrot,
- lokalna oś x płaszczyzny obrazowania jest równoległa do osi X globalnego układu współrzędnych i przyjmuje jej zwrot,
- punkt C przyjmuje współrzędne (0, 0, C_h).



Rys. 5.17 Symulacja realizacji pomiarów wartości d_1, d_2, d_3 oraz C_h na scenie

Powyższe założenia definiują brakujące zależności geometryczne, niezbędne do lokalizacji kamery w globalnym układzie współrzędnych, których wyznaczenie wymagałoby zastosowania skomplikowanej procedury parametryzacji sceny. Ponadto, orientacja pola widzenia kamery zostaje uproszczona ustalając stałe wartości kątów $\psi = 0°$ i $\delta = 90°$. Takie podejście znacząco upraszcza procedurę kalibracji sceny bez negatywnego wpływu na ostateczne wyniki rzutowania 3D.

Na podstawie przyjętych założeń można wyznaczyć globalne współrzędne punktu P_f

$$P_f\left(0, \frac{f}{1000}, C_h\right) \tag{5.10}$$

W kolejnym kroku algorytm realizuje transformacje punktów P_1, P_2, P_3 oraz punktu środka obrazu $P_s = \left(\frac{width}{2}, \frac{height}{2}\right)$ z układu współrzędnych macierzy pikseli do lokalnego układu współrzędnych płaszczyzny obrazowania. Jest to możliwe poprzez zastosowanie przelicznika szerokości (p_w) i wysokości (p_h) piksela na metry

$$\begin{cases} p_w = \dfrac{W_{mm}}{width} \\ p_h = \dfrac{H_{mm}}{height} \end{cases} \tag{5.11}$$

oraz uwzględnienie przeciwnych zwrotów osi y macierzy obrazu i współrzędnych płaszczyzny obrazowania. Ponieważ punkt P_f przecina płaszczyznę obrazowania w jej centralnym punkcie, zatem współrzędne punktu P_f w lokalnym układzie współrzędnych płaszczyzny obrazowania są równe współrzędnym punktu P_s. Znając tą relację, można wyznaczyć współrzędne globalne dowolnego punktu, opisanego przy użyciu współrzędnych lokalnych płaszczyzny obrazowania. Ogólny wzór matematyczny transformacji współrzędnych punktów z lokalnego do globalnego układu współrzędnych można zatem zapisać w formie

$$\begin{cases} P_{gx} = (P_{lx} - P_{sx}) + P_{fx} \\ P_{gy} = P_{fy} \\ P_{gz} = (\mathrm{P_{ly}} - P_{sy}) + P_{fz} \end{cases} \tag{5.12}$$

gdzie:

- P_g – globalne współrzędne punktu uzyskane w procesie transformacji,
- P_l – współrzędne punktu w lokalnym układzie płaszczyzny obrazowania.

Znając długości linii rzutowania d_1, d_2 oraz d_3, współrzędne globalne punktu początkowego (punt C) oraz współrzędne globalne punktów referencyjnych (P_1, P_2, P_3), przez które linie przechodzą, można wyznaczyć współrzędne globalne punktów końców tych linii (P_1'', P_2'' oraz P_3''). Następnie algorytm wyznacza równanie płaszczyzny rzutowania R_z, przechodzącej przez punkty P_1'', P_2'' oraz P_3''. Płaszczyzna rzutowania R_z stanowi nową płaszczyznę podłoża dla uproszczonej orientacji kamery. Kąty ψ oraz δ płaszczyzny R_z względem globalnego układu współrzędnych można wyznaczyć jako kąty pomiędzy wektorem normalnym płaszczyzny R_z a daną osią układu.

W ostatnim etapie algorytm wyznacza siatkę punktów, zwaną dalej macierzą rzutowania 3D, określającą rzeczywiste współrzędne 3D rzutu każdego piksela obrazu na podłoże obserwowanej sceny. Znając współrzędne punktu początkowego linii rzutu (punkt C) oraz współrzędne globalne dowolnego punktu płaszczyzny obrazowania, przez który przechodzi linia rzutu, algorytm wyznacza punkt przecięcia linii rzutu (P_r) z nową płaszczyzną podłoża (R_z) wyznaczoną w poprzednim etapie. Punkt P_r opisuje współrzędne globalne rzutu punktu opisanego w płaszczyźnie obrazowania.

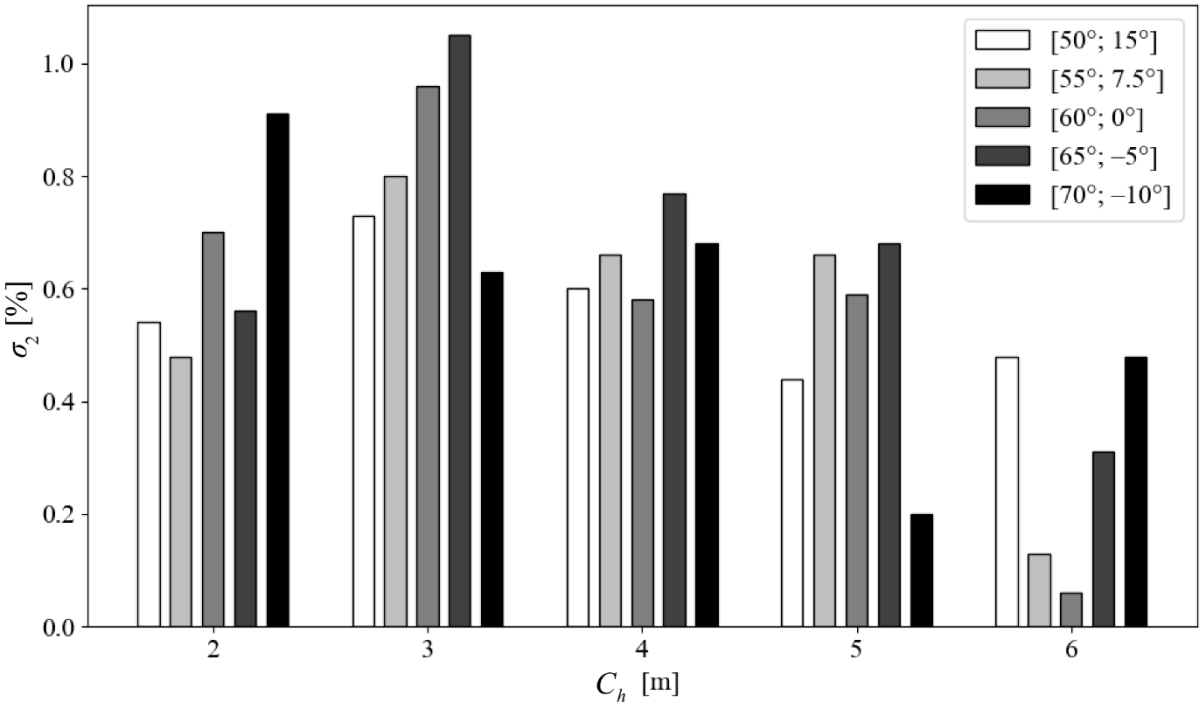
Wstępne badania mające na celu weryfikację algorytmu, przeprowadzono przy użyciu symulacji komputerowej opracowanej w programie Blender. Zasadność tego podejścia wynika z możliwości precyzyjnej parametryzacji kamery wirtualnej, jak również łatwej aranżacji symulowanej sceny w wielu pożądanych wariantach. Przebieg badania obejmuje:

- ustawienie parametrów wewnętrznych kamery wirtualnej w sposób umożliwiający dokładne odwzorowanie fizycznej kamery używanej w testach, a więc: *width* = 1920 px,

height = 1080 px, f = 3.89 mm, *sensor_size* = 0.333 cala, W_{mm} = 5.23 mm oraz H_{mm} = 2.94 mm,

- parametryzację lokalizacji kamery wirtualnej w punkcie $C(0, 0, C_h)$, dla C_h o wartościach: 2.0, 3.0, 4.0, 5.0 oraz 6.0 m,
- parametryzację orientacji kamery wirtualnej względem globalnego układu współrzędnych, a więc kąta δ o wartościach: 50.0, 55.0, 60.0, 65.0 oraz 70.0°, a także kąta ψ o wartościach: 15.0, 7.5, 0.0, –5.0 oraz –10.0°,
- wybór trzech dowolnych punktów referencyjnych P_1, P_2 oraz P_3 na macierzy obrazu,
- realizację pomiaru odległości d_1, d_2, d_3 oraz C_h,
- wybór dwóch dowolnych punktów na macierzy obrazu (P_{d1} oraz P_{d2}), a następnie pomiar rzeczywistej odległości między tymi punktami (d_{real}),
- porównanie obliczonej odległości d_{calc} z rzeczywistą d_{real}.

Wyniki przeprowadzonych badań zamieszczono w Tab. 5.4. Na podstawie Tab. 5.4 opracowano wykres zależności $\sigma_2 = f(C_h)$ (zob. Rys. 5.18) dla badań przeprowadzonych przy różnych kątach δ oraz ψ.



Rys. 5.18 Wykres zależności $\sigma_2 = f(C_h)$ dla zmiennych kątów [δ; ψ]

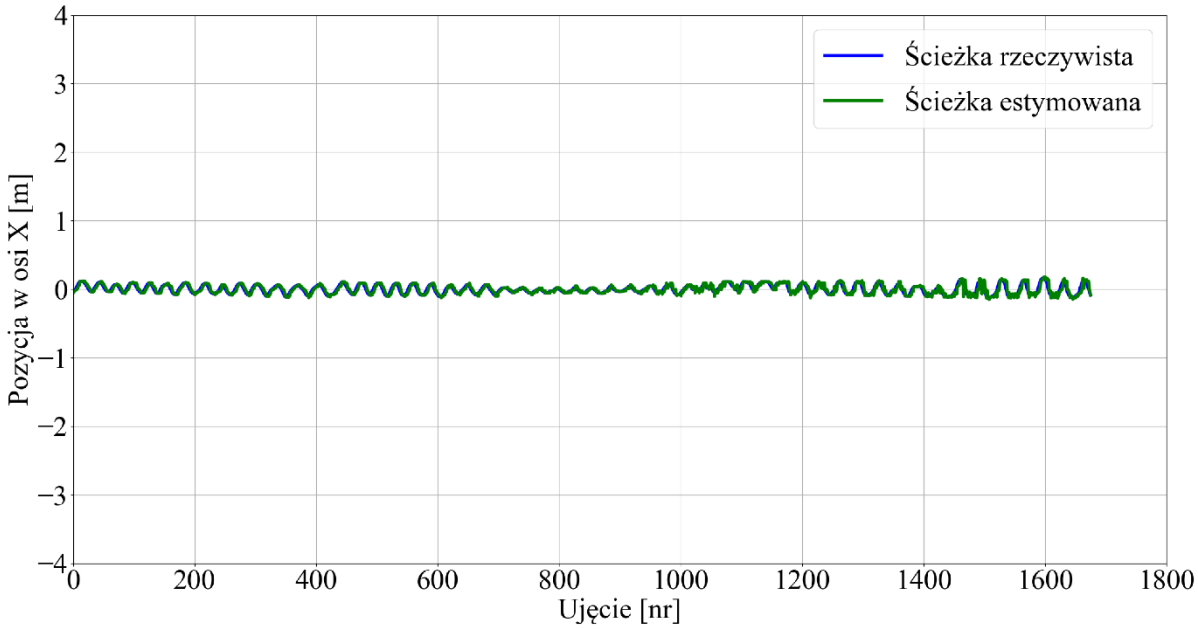
Badania zostały przeprowadzone w sposób umożliwiający weryfikację algorytmu dla różnych konfiguracji lokalizacji i orientacji kamery wirtualnej. Wybrane wartości C_h, ψ oraz δ zawierają się w zakresach często stosowanych konfiguracji monitoringowych. Punkty P_1, P_2 oraz P_3 w każdej próbie zostały wybrane arbitralnie, z zachowaniem możliwie dużych odległości pomiędzy każdym z nich.

Tab. 5.4 Wyniki badań symulacyjnych opracowanego algorytmu kalibracji sceny

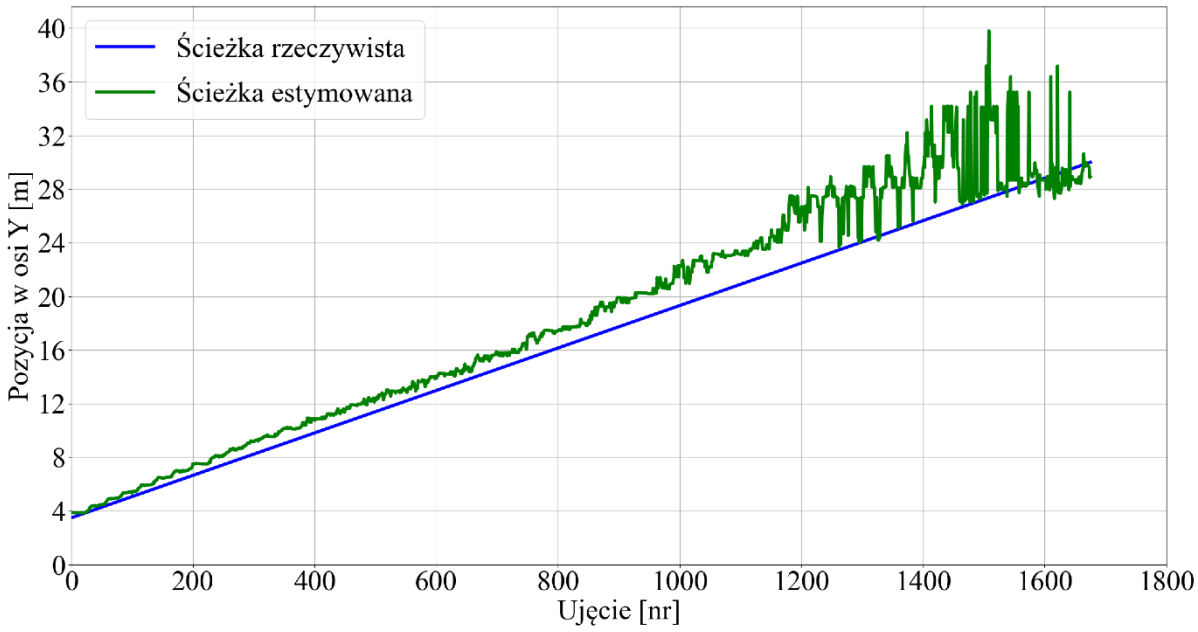
C_h [m]	δ [°]	ψ [°]	P_1 [px]	P_2 [px]	P_3 [px]	d_1 [m]	d_2 [m]	d_3 [m]	P_{d1} [px]	P_{d2} [px]	d_{calc} [m]	d_{real} [m]	σ_1 [m]	σ_2 [%]
2.00	50.00	15.00	(188,205)	(952,1044)	(1819,162)	7.71	2.40	4.15	(1830,1009)	(584,344)	3.68	3.70	0.02	0.54
2.00	55.00	7.50	(87,233)	(961,1040)	(1865,363)	7.71	2.47	4.35	(1583,1044)	(546,304)	4.12	4.14	0.02	0.48
2.00	60.00	0.00	(47,246)	(967,1056)	(1847,278)	7.59	2.60	7.08	(1754,1019)	(465,374)	4.28	4.31	0.03	0.70
2.00	65.00	–5.00	(29,296)	(978,1038)	(1866,364)	7.53	2.87	9.50	(1687,1045)	(303,367)	5.28	5.31	0.03	0.56
2.00	70.00	–10.00	(32,347)	(985,1038)	(1863,498)	7.44	3.20	12.03	(1749,1034)	(262,434)	5.45	5.50	0.05	0.91
3.00	50.00	15.00	(43,282)	(981,1048)	(1886,284)	11.47	3.57	5.53	(1829,984)	(194,464)	6.83	6.88	0.05	0.73
3.00	55.00	7.50	(51,275)	(973,1049)	(1868,285)	10.92	3.69	7.15	(1586,1029)	(193,351)	7.48	7.54	0.06	0.80
3.00	60.00	0.00	(58,284)	(965,1052)	(1867,308)	10.54	3.91	10.09	(1679,1001)	(224,378)	7.19	7.26	0.07	0.96
3.00	65.00	–5.00	(37,334)	(980,1063)	(1853,380)	10.45	4.23	13.60	(1763,1008)	(167,439)	7.53	7.61	0.08	1.05
3.00	70.00	–10.00	(15,300)	(969,1032)	(1902,655)	12.36	4.81	12.23	(1594,998)	(340,406)	7.94	7.99	0.05	0.63
4.00	50.00	15.00	(38,371)	(973,1066)	(1874,230)	13.27	4.74	7.75	(1869,1011)	(370,413)	8.31	8.36	0.05	0.60
4.00	55.00	7.50	(26,367)	(956,1041)	(1833,362)	12.63	4.95	8.66	(1807,1026)	(314,381)	9.09	9.15	0.06	0.66
4.00	60.00	0.00	(16,374)	(956,1061)	(1872,373)	12.18	5.19	12.07	(1853,1067)	(290,351)	10.26	10.32	0.06	0.58
4.00	65.00	–5.00	(22,424)	(951,1058)	(1892,563)	11.91	5.65	12.69	(1825,1035)	(238,418)	10.31	10.39	0.08	0.77
4.00	70.00	–10.00	(28,422)	(964,1059)	(1829,737)	12.84	6.28	13.17	(1733,1060)	(347,387)	11.70	11.78	0.08	0.68
5.00	50.00	15.00	(21,571)	(1000,1040)	(1890,259)	12.95	5.95	9.43	(1891,1001)	(295,415)	11.23	11.28	0.05	0.44
5.00	55.00	7.50	(63,517)	(952,1054)	(1820,411)	12.57	6.15	10.25	(1855,1024)	(287,454)	10.55	10.62	0.07	0.66
5.00	60.00	0.00	(28,532)	(957,1057)	(1873,521)	12.12	6.50	12.21	(1851,1055)	(257,410)	11.82	11.89	0.07	0.59
5.00	65.00	–5.00	(55,502)	(959,1049)	(1850,663)	13.14	7.10	13.33	(1870,1052)	(286,406)	13.09	13.18	0.09	0.68
5.00	70.00	–10.00	(29,555)	(960,1059)	(1892,954)	12.99	7.85	12.49	(1846,1060)	(379,386)	15.21	15.24	0.03	0.20
6.00	50.00	15.00	(40,850)	(967,1036)	(1880,390)	11.86	7.20	10.12	(1830,1009)	(655,343)	10.39	10.34	0.05	0.48
6.00	55.00	7.50	(48,812)	(952,1065)	(1860,490)	11.30	7.35	11.43	(1851,1013)	(302,334)	15.14	15.12	0.02	0.13
6.00	60.00	0.00	(35,732)	(966,1061)	(1884,702)	11.69	7.79	12.02	(1867,1062)	(207,374)	15.62	15.63	0.01	0.06
6.00	65.00	–5.00	(37,729)	(956,1048)	(1871,896)	12.04	8.53	12.32	(1884,1067)	(150,435)	16.07	16.12	0.05	0.31
6.00	70.00	–10.00	(38,754)	(962,1063)	(1856,1071)	12.21	9.40	12.97	(1889,1046)	(279,391)	18.94	18.85	0.09	0.48

Punkty P_{d1} oraz P_{d2} zostały wybrane w taki sposób, aby odległość pomiędzy nimi tworzyła linię zbliżoną do przekątnej obrazu. Ma to na celu ocenę wpływu zarówno kąta ψ jak i δ w procesie kalibracji sceny na końcowy wynik pomiaru odległości d_{calc}. Do oszacowania efektywności działania algorytmu użyto wskaźników $\sigma_1 = |d_{real} - d_{calc}|$ oraz $\sigma_2 = 100\frac{|d_{calc}-d_{real}|}{d_{real}}$. Analiza wyników zestawionych w Tab. 5.4 oraz na Rys. 5.18 pozwala stwierdzić, iż opracowany algorytm kalibracji sceny wprowadza nieznaczne błędy obliczeniowe (maksymalnie 1.05%), których wartość jest w pełni akceptowana pod kątem możliwości implementacji algorytmu w projektowanym systemie IVSS.

Następnie, zaprojektowany algorytm został poddany badaniom empirycznym mającym na celu określenie jego efektywności w warunkach rzeczywistych. W tym celu skorzystano z nagrania wideo opisanego w rozdziale 5.3.1. W pierwszym kroku dokonano wyboru trzech punktów referencyjnych zgodnie z wzorem przedstawionym na Rys. 5.16. Następnie dokonano pomiaru długości C_h = 2.50 m, d_1 = 5.35 m, d_2 = 10.74 m oraz d_3 = 4.58 m (zob. Rys. 5.17) przy użyciu uniwersalnego przymiaru zwijanego. W kolejnym kroku wprowadzono otrzymane dane do programu, co umożliwiło automatyczne wygenerowanie macierzy rzutowania 3D. Następnie przeprowadzono analizę nagrania wideo (zob. rozdział 5.3.1), na którym osoba przemieszczała się wzdłuż pionowej osi obrazu na dystansie 40 m od kamery. Do tego celu użyto programu z zaimplementowanym algorytmem śledzenia osób, opisanym w rozdziale 5.4 oraz zaimportowaną macierzą rzutowania 3D. Program miał za zadanie zapisywać wszystkie wykryte lokalizacje 3D śledzonej osoby na obrazie. Realizacja rzutowania 3D dotyczy wyłącznie globalnego układu współrzędnych płaszczyzny obserwowanej sceny. Z tego powodu postanowiono, aby referencyjny punkt rzutowania dotyczył środka dolnej krawędzi BB monitorowanej osoby. Punkt ten stanowi domniemaną oś symetrii śledzonej osoby, natomiast dolna krawędź BB stanowi domniemany punkt styku osoby z płaszczyzną podłoża. Rezultaty pracy programu przedstawiono na Rys. 5.19. Na wykresie 5.19a przedstawiono porównanie współrzędnych osi X dla trajektorii rzeczywistej ścieżki przejścia oraz ścieżki estymowanej, której wartości zostały obliczone przy użyciu macierzy rzutowania 3D. Obserwacja ta wskazuje zadawalającą skuteczność opracowanego algorytmu, o czym świadczą minimalne wartości odchyłek na całym dystansie. Na wykresie 5.19b przedstawiono porównanie współrzędnych osi *Y* dla trajektorii rzeczywistej ścieżki przejścia oraz ścieżki estymowanej, której wartości zostały obliczone przy użyciu macierzy rzutowania 3D. Wartości odchyłek są w tym przypadku nieco większe i charakteryzują się zachowaniem tendencji wzrostowej wraz ze wzrostem dystansu osoby od kamery. Problem ten może występować w następstwie wielu czynników, takich jak: niepewność pomiaru długości d_1, d_2, d_3 oraz C_h, błędy obliczeniowe programu (zaokrąglanie wartości), niepewność detektora OpenPose (zob. Rys. 5.9), występująca szczególnie na większych dystansach czy niska rozdzielczość obrazu, wymuszająca



a)



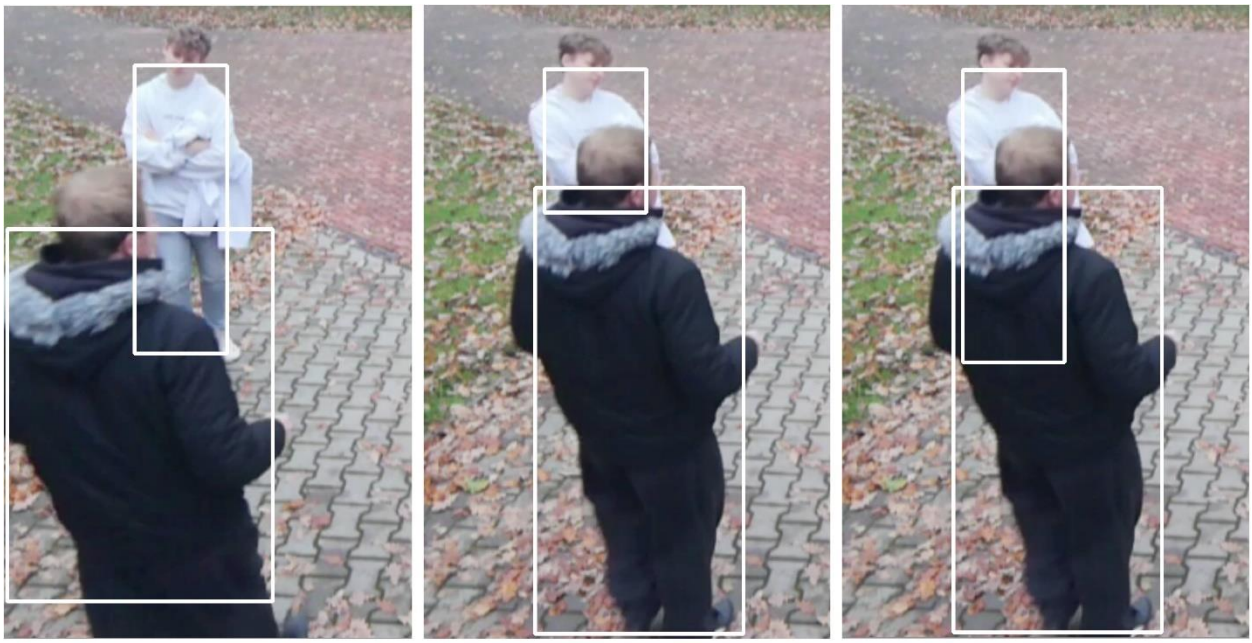
b)

Rys. 5.19 Wyniki badań empirycznych algorytmu kalibracji sceny dla współrzędnych osi: a) X, b) Y

skokową procedurę rzutowania (brak możliwości wyznaczenia rzutu 3D pomiędzy dwoma sąsiadującymi pikselami obrazu). Mając na uwadze powyższe ograniczenia, można przyjąć, iż skuteczność proponowanego algorytmu jest zadawalająca dla dystansu nie przekraczającego 18 m, gdyż dla tych wartości odchyłka pomiaru nie przekracza 2 m.

Dalsze badania empiryczne, dotyczące weryfikacji opracowanej metody, ujawniły jedną z najczęstszych przyczyn występowania błędnych wyników rzutowania. Zmiany wysokości BB, o których mowa, to powszechne zjawisko występujące podczas częściowej okluzji dolnej części śledzonego obiektu przy obserwacji sceny z wysokości powyżej przeciętnego wzrostu człowieka.

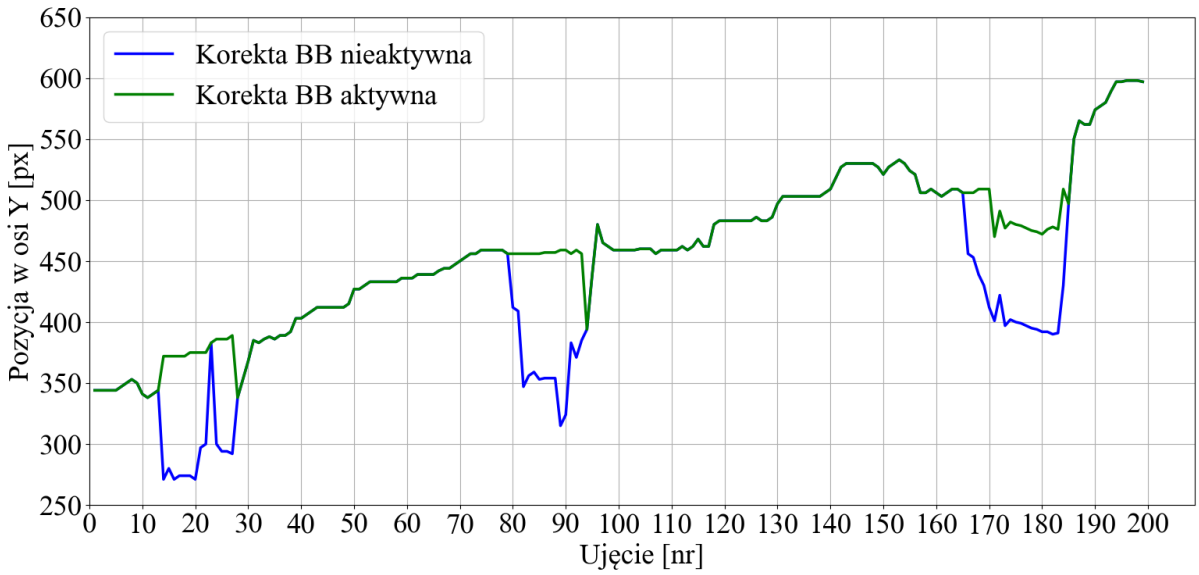
Na Rys. 5.20 przedstawiono omawiany przykład dla analizy wybranych ujęć (83 oraz 90) pochodzących z pojedynczego nagrania wideo.



a) **b)** **c)**

Rys. 5.20 Przykład zmian wysokości BB związany z chwilową okluzją dla analizy pojedynczego nagrania wideo: a) ujęcie nr. 83 (wzorcowe wysokości BB), b) ujęcie nr. 90 (korekta wysokości BB nieaktywna), c) ujęcie nr. 90 (korekta wysokości BB aktywna)

Na Rys. 5.20a przedstawiono fragment ujęcia nr. 83 analizowanego materiału wideo, na którym można zaobserwować wzorcowe wysokości obu wykrytych BB. Na Rys.5.20b przedstawiono fragment ujęcia nr. 90, na którym osoba w białej bluzie zostaje częściowo przysłonięta przez osobę w czarnej kurtce, co w rezultacie powoduje znaczną zmianę wysokości przypisanemu jej BB. W takim przypadku próba rzutowania punktu referencyjnego (środka dolnej krawędzi BB) skutkowałaby uzyskaniem błędnych wyników. W celu minimalizacji negatywnego wpływu okluzji na proces rzutowania 3D, w algorytmie śledzenia MOT zaimplementowano logikę porównującą wysokość nowego BB z poprzednio zapisaną podczas etapu aktualizacji. Jeśli wysokość nowej ramki jest mniejsza niż 90% wysokości BB z poprzedniego ujęcia oraz występuje brak detekcji punktorów szkieletu o numerach 11 i 14 (prawa i lewa kostka), wówczas nowy BB zostaje zapisany ze zmodyfikowaną wysokością równą wysokości uprzednio zapisanego BB. Zastosowanie tej logiki umożliwia utrzymanie ostatniej znanej prawidłowej wysokości BB przez cały czas trwania okluzji, do momentu ponownej detekcji punktorów szkieletu dolnej części ciała. Na Rys. 5.21 przedstawiono przykład ścieżki przejścia śledzonej osoby, w trakcie której następuje trzykrotna częściowa okluzja dolnej części ciała. Podczas okluzji tego typu, dolna krawędź BB przesuwa się ku górze obrazu, aby ramka mogła obejmować tylko widoczną (górną) część ciała. Powoduje to nagłą zmianę lokalizacji środka dolnej krawędzi w osi pionowej obrazu, co można zaobserwować na przebiegu czasowym oznaczonym niebieskim kolorem. Aktywowanie



Rys. 5.21 Przebieg czasowy lokalizacji śledzonej osoby podczas częściowej okluzji

opracowanej logiki korekty wysokości BB pozwala w znacznym stopniu utrzymać właściwą wysokość BB bez wpływu na jego lokalizację na obrazie, czego dowodem jest przebieg czasowy oznaczony kolorem zielonym. Jak można zaobserwować na Rys. 5.20c, w przypadku aktywowania opracowanej metody, wysokość BB osoby w białej bluzie utrzymuje ostatnią poprawnie zapamiętaną wysokość. Ponadto, korekta wysokości BB nie wpływa na jego lokalizację na obrazie (współrzędne X oraz Y pozostają niezmienione). Należy zwrócić uwagę, iż korekta BB działa tylko w przypadku spełnienia dwóch warunków: niezerowej liczby aktualizacji śledzonej osoby oraz ostatniego zapisanego BB wyznaczonego na podstawie modelu szkieletowego zawierającego niezerowe detekcje punktorów 11 i 14.

Podsumowując, zaproponowany algorytm kalibracji sceny umożliwia wygenerowanie macierzy 3D przeznaczonej do rzutowania współrzędnych każdego piksela obrazu 2D do współrzędnych 3D podłoża obserwowanej sceny. Zadanie to realizuje niezależnie od wysokości montażu (C_h), kąta nachylenia (δ) oraz kąta obrotu (ψ) kamery, co czyni go narzędziem uniwersalnym. Należy pamiętać, iż wygenerowana macierz rzutowania 3D jest przeznaczona wyłącznie do analizy sceny, dla której została opracowana. Wszelkie zmiany związane z polem widzenia kamery lub zastosowanie macierzy dla innej kamery wymagają ponownego procesu kalibracji sceny. Proces kalibracji natomiast nie wymaga użycia specjalistycznych urządzeń kalibrujących, a dzięki intuicyjnemu GUI może zostać przeprowadzony przez dowolnego użytkownika. W celu uniknięcia negatywnego wpływu okluzji dolnej części ciała, skutkujących nagłą zmianą referencyjnego punktu rzutowania, rozbudowano algorytm śledzenia MOT o dodatkową logikę. W przypadku zdiagnozowania okluzji, zaproponowana metoda zastępuje wysokość nowej detekcji BB ostatnią prawidłową wartością wysokości, zapisaną w historii śledzenia obiektu. Zastosowanie tej metody znacząco poprawia stabilność procesu rzutowania 3D,

co potwierdzają rezultaty przeprowadzonych badań empirycznych. Powyższe wnioski dowodzą, iż opracowany algorytm może stanowić cenne narzędzie w projektowanym systemie IVSS.

5.6 Podsumowanie

Proces kalibracji kamery przeprowadzono przy użyciu funkcji *cv::CalibrateCamera()* dostępnej w bibliotece OpenCV. Wyniki przeprowadzonej kalibracji wskazują na minimalne zniekształcenia styczne (wskaźniki p_1, p_2), natomiast wskaźniki zniekształceń radialnych (k_1, k_2, k_3) przyjmują wartości mogące wprowadzać dystorsje szczególnie zauważalne w obszarach krawędzi obrazu. W celu redukcji dystorsji zastosowano funkcję *cv::remap()* umożliwiającą realizację tego zadania w akceptowalnym czasie przetwarzania (ok. 4 ms dla pojedynczego obrazu FHD). Końcową korektę geometrii obrazu (po redukcji zniekształceń) zrealizowano poprzez zmianę argumentu *alpha* w funkcji *cv::getOptimalNewCameraMatrix()* do wartości 0.0.

W celu poprawy jakości obrazu pod kątem detekcji obiektów, autor opracował algorytm automatycznej korekty jasności. Algorytm korzysta z adaptacyjnego mechanizmu korekty współczynnika γ, co pozwala na automatyczną reguluję poziomu jasności, wprowadzając stosowną zmianę zarówno w przypadku obrazów zaciemnionych, jak i prześwietlonych. Zdolność adaptacji intensywności korekty do średniej ważonej jasności obrazu zapewnia uniwersalność zastosowania, a niski stopień skomplikowania umożliwia szybką realizację obliczeń w czasie ok. 4 ms dla pojedynczego obrazu FHD. Zastosowanie opracowanego algorytmu umożliwia zwiększenie liczby detekcji typu *TP* detektora OpenPose, co udowodniono na konkretnym przykładzie.

W projektowanym systemie IVSS użyto czterech detektorów obiektów: OpenPose, YOLO v4, YOLO v4 tiny oraz Tesseract OCR. Z uwagi na rozbudowaną architekturę, detektor OpenPose stanowi bazę programową projektowanego systemu. Spośród dostępnych modeli detektora OpenPose wybrano BODY_25, ponieważ (zgodnie z opinią twórców) umożliwia osiągnięcie najwyższej dokładności i wydajności w przypadku użycia GPU wspieranych technologią CUDA w porównaniu do pozostałych dostępnych modeli. Model BODY_25 umożliwia generowanie szkieletów kinematycznych sylwetek człowieka składających się z 25 punktorów, przy czym autor rozprawy postanowił korzystać tylko z 15 (od 0 do 14), reprezentujących najistotniejsze części ciała. Wybrany model został poddany badaniom empirycznym, mającym na celu określenie maksymalnego dystansu osoby od kamery, pozwalającego na uzyskanie zadowalającej pewności detekcji. Rezultaty przeprowadzonych badań dowodzą, iż efektywność detektora OpenPose w klasycznych warunkach sceny monitoringu jest zadowalająca. Należy jednak zwrócić uwagę na ograniczenia związane ze spadkiem pewności detekcji w przypadku analizy osób znajdujących się w odległości większej niż 18 m od kamery.

Oryginalny, wytrenowany model *yolov4* został użyty do celu detekcji następujących klas obiektów: „*człowiek*”, „*kot*”, „*pies*”, „*samochód osobowy*”, „*samochód ciężarowy*”, „*autobus*” oraz „*motocykl*”, przy czym finalnie wyniki detekcji czterech ostatnich klas są interpretowane jako pojedyncza klasa o nazwie „*pojazd*”.

W celu opracowania detektora obiektów klas „*tablice rejestracyjne*” oraz „*ogień*”, autor zrealizował proces uczenia maszynowego modelu *yolov4-tiny*. W uczeniu maszynowym użyto odpowiednio oznaczonych obrazów, które zostały pobrane z platformy internetowej Roboflow. Opracowana baza danych uczących stanowiła 45880 obrazy dla klasy „*tablice rejestracyjne*” i 47222 obrazy dla klasy „*ogień*”. Trening modelu został zrealizowany z użyciem dedykowanego oprogramowania, udostępnionego przez twórców YOLO v4. Przeprowadzono 33000 epok, o łącznym czasie trwania ok. 7.5 godziny. Optymalną wartość *AP* modelu osiągnięto dla epoki 16500, gdzie wartość ta jest równa 97% dla klasy „*tablice rejestracyjne*” i 46% dla klasy „*ogień*”. Rozbieżność tych wartości wynika z wysokiej złożoności kształtu, koloru i rozmiaru, jaki może przybierać ogień w warunkach rzeczywistych oraz możliwych okluzji spowodowanych towarzyszącym zadymieniem.

Implementacja detektora Tesseract OCR została zrealizowana w celu rozpoznawania numerów jednorzędowych tablic rejestracyjnych. Parametryzacja tego detektora sprowadza się do wyboru odpowiedniej metody segmentacji obrazu, którą w tym przypadku jest PSM 7.

Śledzenie wielu obiektów na obrazie zostało opracowane poprzez klasyczną syntezę filtra Kalmana i logiki przyporządkowania detekcji na podstawie minimalnej różnicy odległości pomiędzy nowymi i poprzednimi lokalizacjami wzorcowych punktów śledzenia. Algorytm MOT realizuje śledzenie tylko dla jednej klasy obiektów: „*człowiek*”. Z uwagi na fakt, iż OpenPose dostarcza bardziej szczegółowych informacji o sylwetce człowieka, autor postanowił, iż OpenPose pełni funkcję detektora nadrzędnego, natomiast YOLO v4 pomocniczego. Jako wzorcowy punkt śledzenia wybrano punkt środka górnej krawędzi BB, co pozwala uśrednić szerokość BB wskazując na domniemaną oś symetrii osoby oraz zupełnie wykluczyć wpływ zmian wysokości BB na kolejnych ujęciach mogących powstać w wyniku częściowej okluzji. Algorytm śledzenia realizuje cztery podstawowe funkcje: utworzenie nowego śledzonego obiektu, aktualizację położenia śledzonego obiektu nową detekcją, aktualizację położenia śledzonego obiektu przy użyciu predykatora oraz usunięcie obiektu z procedury śledzenia w przypadku utrzymującego się braku detekcji. Zastosowanie filtra Kalmana umożliwiło predykcję lokalizacji obiektu na obrazie w przypadku chwilowej, całkowitej okluzji, co pozytywnie wpływa na stabilność procesu śledzenia.

Autor rozprawy zaproponował algorytm kalibracji sceny, umożliwiający wygenerowanie macierzy rzutowania 3D celem estymacji każdego piksela obrazu 2D do współrzędnych 3D płaszczyzny obserwowanej sceny. Algorytm korzysta z podstawowych przekształceń

geometrycznych i trygonometrycznych, umożliwiając obliczenie macierzy rzutowania 3D, która zostanie użyta w dalszych etapach pracy do oceny dystansu oraz szybkości przemieszczenia osób poruszających się na obrazie. W celu weryfikacji pracy opracowanego algorytmu, przeprowadzono badanie symulacyjne oraz empiryczne. Analiza wyników badania symulacyjnego pozwala stwierdzić, iż algorytm wprowadza nieznaczne błędy obliczeniowe (maksymalnie 1.05%), których wartość jest w pełni akceptowana pod kątem możliwości implementacji algorytmu w projektowanym systemie IVSS. Analiza badania empirycznego pozwala stwierdzić, iż skuteczność proponowanego algorytmu jest zadawalająca przy dystansie osoby od kamery nie przekraczającym 18 m. Algorytm realizuje rzutowanie do przestrzeni 3D niezależnie od wysokości montażu (C_h), kąta nachylenia (δ) oraz kąta obrotu (ψ) kamery, co czyni go narzędziem uniwersalnym. Proces kalibracji sceny nie wymaga użycia specjalistycznych urządzeń kalibrujących, a dzięki intuicyjnemu GUI może zostać przeprowadzony przez dowolnego użytkownika. Dodatkowo, autor dokonał implementacji logiki realizującej korektę wysokości BB. Dzięki temu osiągnięto znaczącą redukcję wpływu okluzji dolnej części ciała na przebieg trajektorii śledzonej sylwetki. Ma to kluczowe znaczenie w kontekście rzutowania lokalizacji obiektu ze współrzędnych obrazu 2D do rzeczywistej przestrzeni 3D obserwowanej sceny.

6 Algorytmy detekcji zdarzeń alarmowych

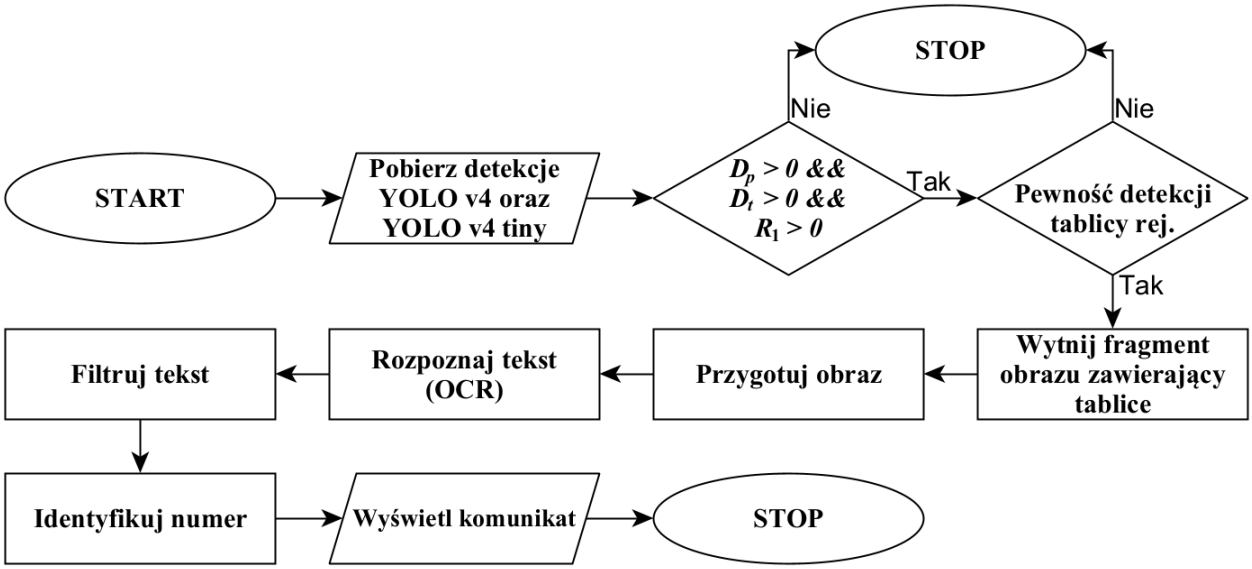
Rozdział ten zawiera opis algorytmów detekcji zdarzeń alarmowych, które zostały zaimplementowane w projektowanym systemie IVSS. Opisy poszczególnych algorytmów będą zawierać przyjęte cele i założenia, zastosowaną logikę operacyjną, testy oraz wyniki przeprowadzonych badań, jak również odniesienie do istniejących tego typu rozwiązań w literaturze. Poszczególne algorytmy detekcji zdarzeń alarmowych zostaną opisane w kolejności od najmniejszej częstości implementacji w pracach referencyjnych (zob. Rys. 3.5). W zakończeniu rozdziału zawarte zostaną informacje dotyczące syntezy wszystkich opracowanych algorytmów w formie wielozadaniowej aplikacji komputerowej.

6.1 Identyfikacja pojazdów

W niniejszej rozprawie zadanie identyfikacji pojazdów dotyczy automatycznego rozpoznawania numerów tablic rejestracyjnych samochodów (ANPR) na podstawie sygnału wizyjnego, pobranego z kamery monitoringowej. Obecnie większość tego typu rozwiązań (np. w systemach kontroli wjazdu [140]) wymaga użycia wypoziomowanej kamery, zamontowanej nisko nad podłożem oraz skierowania pola widzenia bezpośrednio na obszar najczęstszego występowania tablic rejestracyjnych. Takie rozwiązania, choć skuteczne w ustalonych warunkach, ograniczają uniwersalność zastosowania kamery tylko do jednego zadania. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje, iż tego typu rozwiązania nie zostały jak do tej pory zaimplementowane w wielozadaniowych systemach IVSS. W odniesieniu do luki badawczej, autor rozprawy opracował algorytm, umożliwiający realizację zadania ANPR z użyciem kamer CCTV, montowanych na wysokościach charakterystycznych dla klasycznego monitoringu wizyjnego. Schemat blokowy algorytmu przedstawiono na Rys. 6.1.

W pierwszej kolejności pobierane są wszystkie detekcje YOLO v4 dla obiektów klasy „*pojazd*” oraz wszystkie detekcje YOLO v4 tiny dla obiektów klasy „*tablica rejestracyjna*”. Dalej sprawdzane są warunki niezerowości wspomnianych klas obiektów, a także niezerowości liczby zaznaczonych ROI, które mają podlegać nadzorowi. Następnie, w sposób iteracyjny, sprawdzany jest warunek pewności detekcji tablic rejestracyjnych. Polega to na obliczeniu współczynników *IoU* w celu określenia, czy BB klasy „*pojazd”* zawiera się w obszarze ROI oraz czy BB klasy „*tablica rejestracyjna”* zawiera się w BB „*pojazd”*. Jeśli tak, algorytm rozpoczyna wstępne przetwarzanie obrazu, mające na celu przygotowanie do rozpoznawania tekstu przez detektor Tesseract OCR.

Proces przygotowania obrazu realizowany jest zgodnie z instrukcjami [138] i rozpoczyna się od wydobycia fragmentu obrazu oznaczonego jako „*tablica rejestracyjna*”. W dalszym kroku realizowane jest subtelne rozjaśnienie obrazu, polegające na zastosowaniu korekty γ o stałej



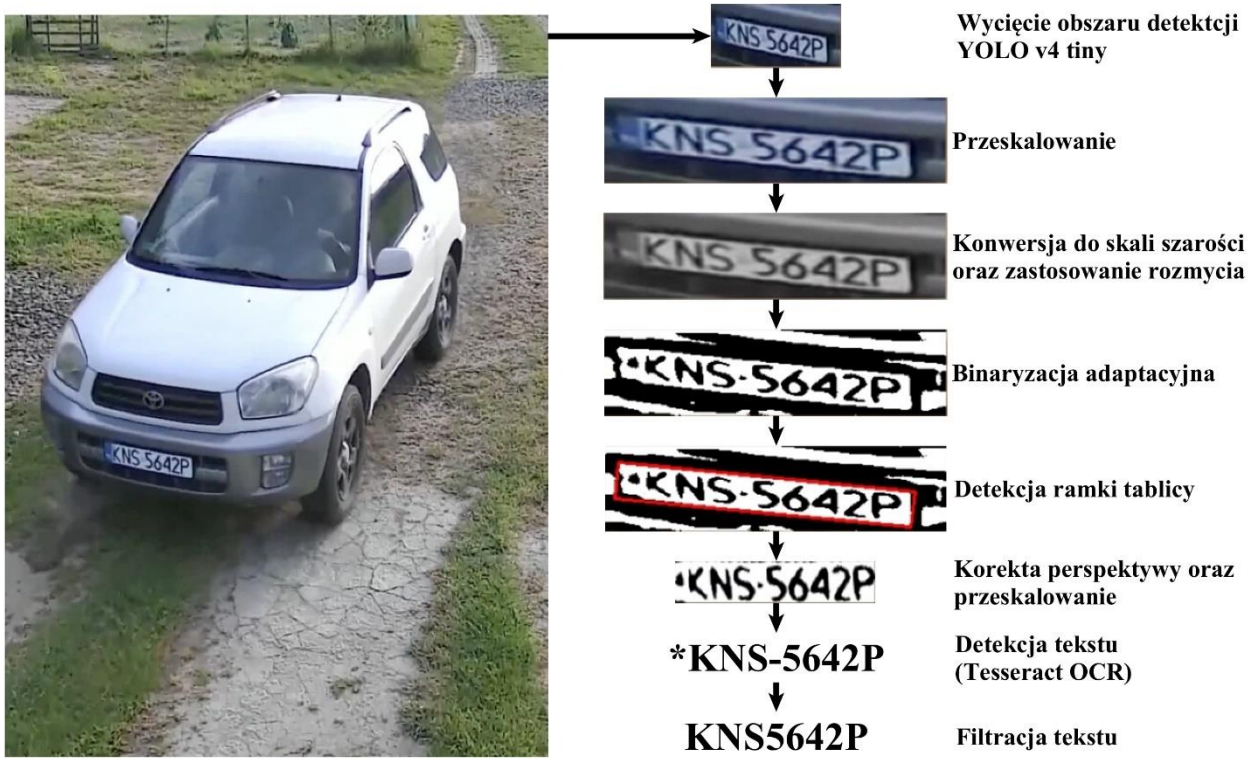
Rys. 6.1 Schemat blokowy algorytmu identyfikacji pojazdów, gdzie: D_p – liczba detekcji pojazdów, D_t – liczba detekcji tablic rejestracyjnych, R_1 – licznik ROI

wartości 0.9. Obraz w tej formie może charakteryzować się zbyt małą rozdzielczością dla zapewnienia poprawności wykonania dalszych kroków przetwarzania. Dlatego w następnym kroku następuje skalowanie obrazu do arbitralnej rozdzielczości 300 × 75 px. Kolejną operacją jest konwersja obrazu do skali szarości oraz zastosowanie rozmycia. Detektor Tesseract OCR przyjmuje obraz wejściowy w formie binarnej, zatem w kolejnym etapie zastosowano binaryzację adaptacyjną. Dowolny kąt obserwacji tablicy wprowadza konieczność zastosowania korekty perspektywy, mającej na celu automatyczne wypoziomowanie obszaru obrazu zawierającego tekst. Zadanie to zostało zrealizowane poprzez detekcję największego prostokątnego obszaru (domniemana ramka tablicy). Następnie wyznaczane są wierzchołki tego prostokąta, które dalej są używane w celu wycięcia najistotniejszej części obrazu (domniemanego tekstu) oraz transformacji perspektywy. W ostatnim kroku przygotowania, obraz zostaje przeskalowany do rozdzielczości 175 × 38 px. Wartość ta została dobrana empirycznie, pozwalając zachować właściwe proporcje tablicy rejestracyjnej opisane w rozporządzeniu ministerialnym [141] oraz zapewnić pożądaną dokładność detekcji tekstu przy relatywnie krótkim czasie przetwarzania.

Tak przygotowany obraz zostaje poddany analizie przez detektor Tesseract OCR. Tekst, będący wynikiem pracy detektora, zostaje poddany filtracji w celu usunięcia znaków nie znajdujących się w słowniku

$$\text{słownik} = \begin{bmatrix} 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, \mathrm{Q}, \mathrm{W}, \mathrm{E}, \mathrm{R}, \mathrm{T}, \mathrm{Y}, \mathrm{U}, \mathrm{I}, \\ \mathrm{O}, \mathrm{P}, \mathrm{A}, \mathrm{S}, \mathrm{D}, \mathrm{F}, \mathrm{G}, \mathrm{H}, \mathrm{J}, \mathrm{K}, \mathrm{L}, \mathrm{Z}, \mathrm{X}, \mathrm{C}, \mathrm{V}, \mathrm{B}, \mathrm{N}, \mathrm{M} \end{bmatrix}$$

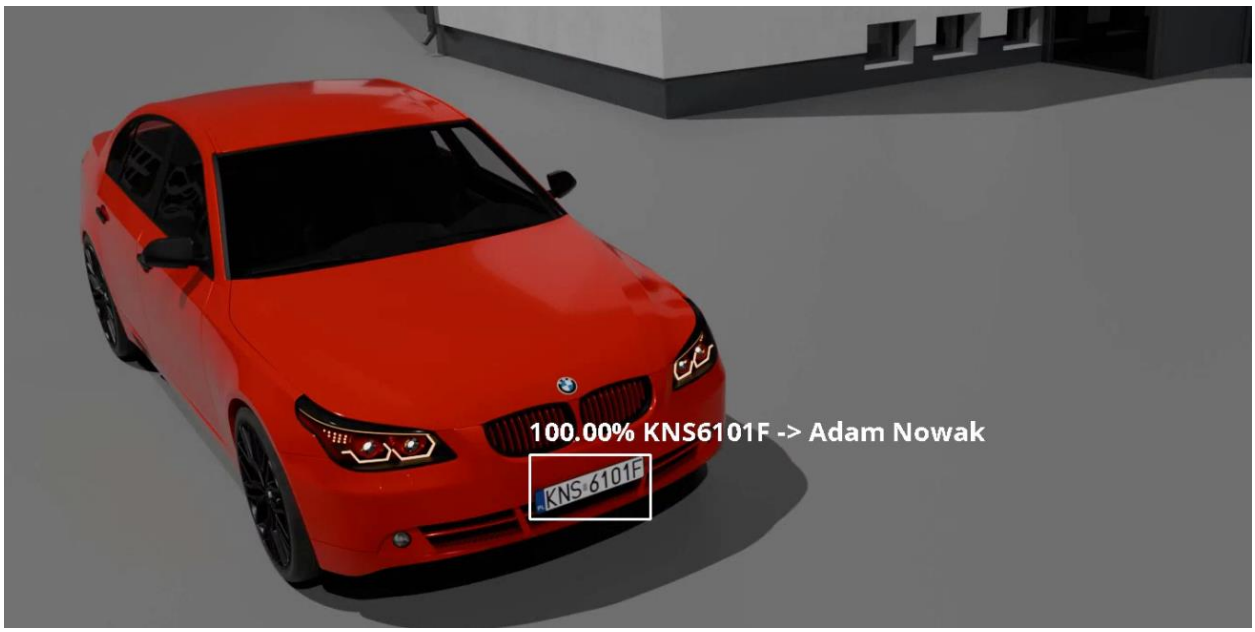
Słownik ten zawiera dopuszczalne znaki alfanumeryczne, z których budowane są numery tablic rejestracyjnych zgodnie z rozporządzeniem [141]. Na Rys. 6.2 przedstawiono wizualizacje opisanych etapów pracy algorytmu.



Rys. 6.2 Wizualizacja etapów pracy algorytmu identyfikacji pojazdów

Ostatni etap algorytmu realizuje proces identyfikacji tekstu. Polega to na porównaniu otrzymanego tekstu z tekstem zapisanym w bazie danych identyfikatorów pojazdów. Bazę tą stanowi plik CSV, w którym zapisano numery tablic i odpowiadające im personalia właścicieli pojazdów. Plik ten jest ładowany do pamięci podręcznej jednokrotnie podczas startu aplikacji. Procentowy wskaźnik prawdopodobieństwa (pewności) dopasowania identyfikacji jest obliczany na podstawie dwóch kryteriów: liczby dopasowanych znaków oraz kolejności ich występowania w wykrytym tekście. Rezultat pracy algorytmu zostaje wyświetlony na obrazie podglądu monitorowanej sceny w postaci komunikatu tekstowego.

Wstępne testy algorytmu zrealizowano z użyciem symulacji 3D opracowanej w środowisku Blender. Celem testów jest określenie maksymalnych wartości kątów widzenia oraz maksymalnej odległości pojazdu od kamery, dla których prawdopodobieństwo prawidłowej detekcji numerów wynosi minimum 85%. Symulowaną scenę opracowano przy użyciu komponentów dostępnych w środowisku Blender (budynki, pojazdy, itd.) oraz przy zachowaniu parametryzacji kamery wirtualnej zgodnej z parametrami fizycznej kamery używanej do testów. W celu osiągnięcia wysokiego stopnia odwzorowania istotnych detali, autor skorzystał z generatora tablic rejestracyjnych online PlatesMania [130]. Na Rys. 6.3 przedstawiono widok z kamery wirtualnej, obejmujący przykładową scenę dotyczącą wstępnych testów algorytmu. Na analizowanej scenie punkt centralny tablicy rejestracyjnej znajduje się w odległości 5.4 m od czujnika optycznego kamery (mierząc po najkrótszej linii). Kamera zamontowana jest na wysokości C_h = 3 m,



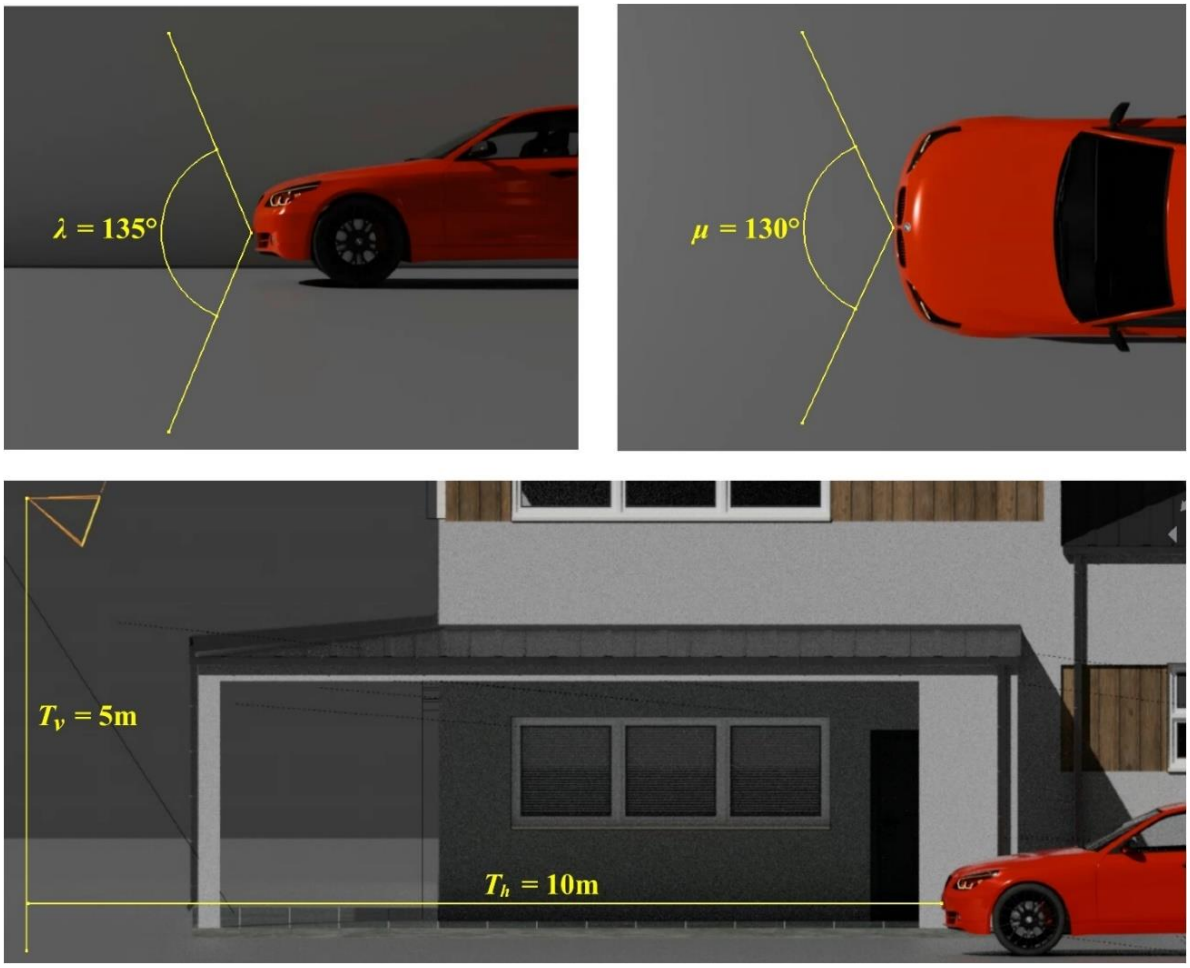
Rys. 6.3 Widok z kamery wirtualnej obejmujący przykładową scenę dotyczącą wstępnych testów algorytmu identyfikacji pojazdów

kąt δ = 70°, ψ = 0°. W tym przypadku tekst wygenerowany przy użyciu OCR Tesseract to „*(KNS*6101F*”. Po przeprowadzeniu filtracji tekstu otrzymano „*KNS6101F*”. Przefiltrowany tekst w stu procentach odpowiada numerowi tablicy zapisanemu w bazie danych, umożliwiając identyfikację właściciela pojazdu „*Adam Nowak*”, o czym świadczy komunikat tekstowy widoczny na ujęciu. Użyta baza danych identyfikatorów pojazdów (zob. Tab. 6.1) zawiera 10 numerów tablic wraz z odpowiadającymi im personaliami właścicieli pojazdów.

Tab. 6.1 Baza danych numerów tablic rejestracyjnych użyta w testach algorytmu identyfikacji pojazdów

L.p.	Personalia	Numery tablicy
1	Krzysztof Kowalski	K73398
2	Adam Nowak	KNS6101F
3	Anna Wilk	SBE12E57
4	Mateusz Makowski	WA25F
5	Joanna Kucia	KNS5642P
6	Ilona Ostrowska	KNS3021G
7	Bartosz Musiał	KNS6857K
8	Antoni Potoczek	KNSAF84
9	Roman Polański	KNS6101F
10	Justyna Święs	KNS2215G

Wizualizację wyników wstępnych testów przedstawiono na Rys. 6.4. Przeprowadzone testy umożliwiły określenie parametrów granicznych algorytmu, dla których prawdopodobieństwo prawidłowej detekcji numerów wynosi minimum 85%: maksymalny kąt widzenia tablicy w osi pionowej λ = 135°, maksymalny kąt widzenia tablicy w osi poziomej μ = 130°.



Rys. 6.4 Wizualizacja parametrów granicznych algorytmu identyfikacji pojazdów

Maksymalna odległość tablicy od kamery w poziomie T_h = 10 m, natomiast w pionie T_v = 5 m.

Analiza możliwości identyfikacji numerów tablic rejestracyjnych różnych typów wykazała pełną zdolność algorytmu do rozpoznawania zarówno zwykłych tablic z białym tłem, jak również tablic tymczasowych z czerwonymi znakami, zabytkowych z żółtym tłem oraz tablic dedykowanych dla pojazdów elektrycznych (z zielonym tłem). Na Rys. 6.5 przedstawiono rezultat pracy algorytmu dla wspomnianych typów tablic.

Rys. 6.5 Rezultat pracy algorytmu identyfikacji pojazdów dla różnych typów tablic rejestracyjnych

Kolejny etap badań dotyczył realizacji testów empirycznych. W tym celu użyto zestawu filmów testowych, zawierających łącznie 17413 ujęć (zob. rozdział 4.4.1). Do celu identyfikacji pojazdów użyto bazy danych identyfikatorów pojazdów przedstawionej w Tab. 6.1, natomiast w Tab. 6.2 zestawiono wyniki przeprowadzonych badań empirycznych.

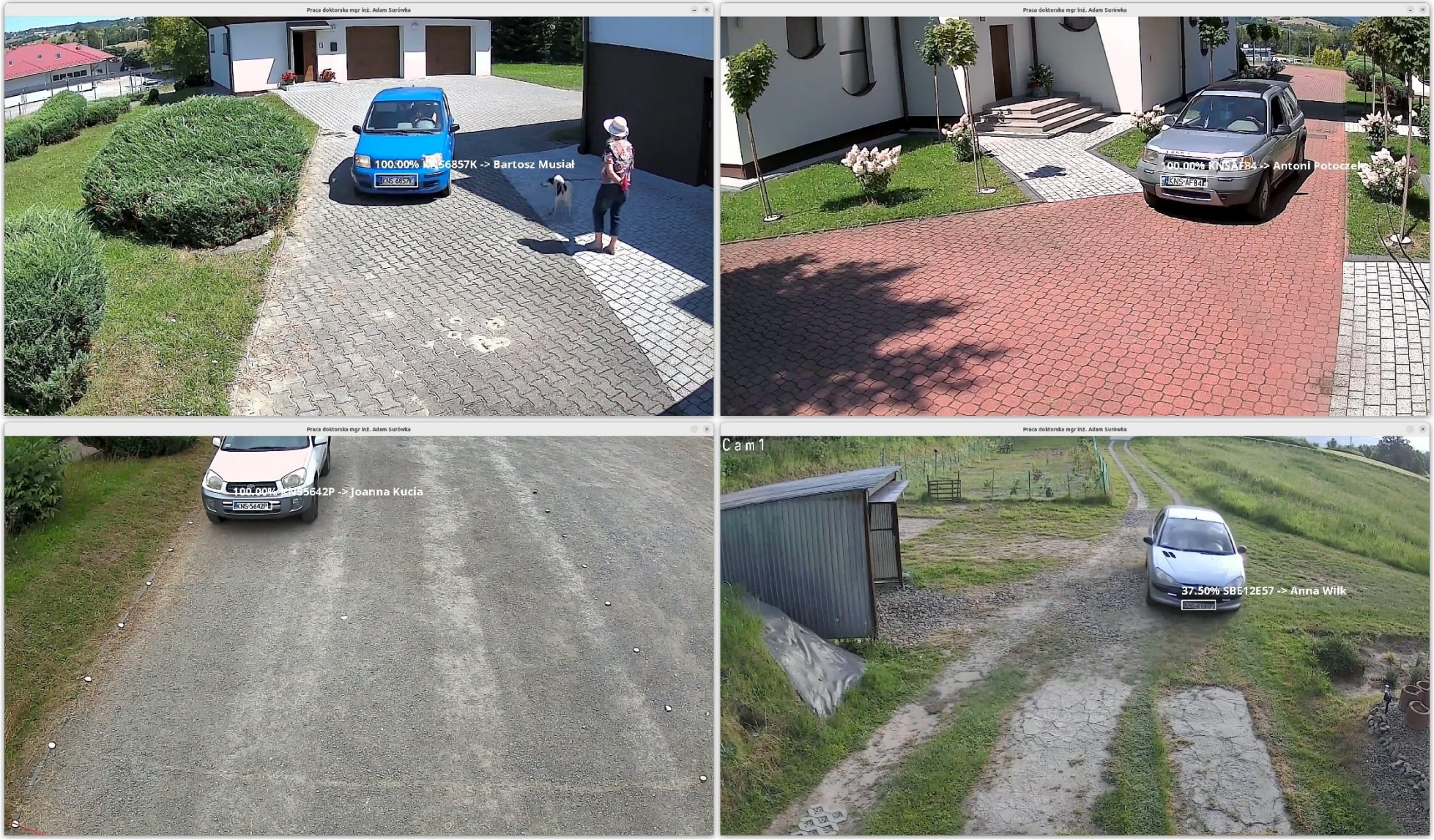
Tab. 6.2 Zestawienie wyników badań empirycznych algorytmu identyfikacji pojazdów

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
15108	2301	0	4	87%	87%	100%	93%

Wartość *TP* dotyczy wszystkich prawidłowych detekcji, których pewność została obliczona przez algorytm na minimum 85%. Wartość *FP* to suma wszystkich błędnych detekcji oraz detekcji prawidłowych o pewności poniżej 85%. Filmy testowe zostały opracowane w taki sposób, aby na każdym ujęciu znajdowała się przynajmniej jedna, w pełni widoczna tablica rejestracyjna. Z tego powodu wartość *TN*, a więc liczba poprawnie zidentyfikowanych braków tablicy, jest równa 0. Wartość *FN* dotyczy wszystkich przypadków braku detekcji tablicy rejestracyjnej, pomimo iż była widoczna na obrazie. Wartości dokładności, precyzji, czułości oraz F_1 zostały obliczone zgodnie z wzorami opisanymi w rozdziale 2.10. Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu numerów tablic rejestracyjnych, o czym świadczą wskaźniki dokładności i precyzji równe 87%, wskaźnik czułości równy 100% oraz F_1 równy 93%.

Na Rys. 6.6 przedstawiono przykładowe ujęcia poklatkowe, zapisane podczas testów empirycznych algorytmu. Informacja dotycząca identyfikacji pojazdu wyświetlana jest na podglądzie monitorowanej sceny w postaci komunikatu tekstowego oraz ramki w białym kolorze, co ułatwia lokalizację analizowanej tablicy rejestracyjnej na obrazie. Ujęcie znajdujące się w prawej dolnej części Rys. 6.6 wskazuje przykład błędnej detekcji. Algorytm z pewnością równą 37.5% identyfikuje numery rejestracyjne „*SBE12E57*”, choć w rzeczywistości są to „*KNS6101F*”. Pozostałe przypadki zostały rozpoznane prawidłowo z pewnością równą 100%. Rezultaty przeprowadzonych testów (zob. Rys. 6.6) udowadniają, iż niezależnie od kąta najazdu oraz odległości od kamery (w ustalonym zakresie) możliwe jest osiągnięcie pewności detekcji rzędu 100%.

Podsumowując, zaprojektowany algorytm identyfikacji pojazdów jest przeznaczony do rozpoznawania numerów tablic rejestracyjnych jednorzędowych, o standardowych rozmiarach i kolorach ujętych w rozporządzeniu ministerialnym [141]. Funkcjonalność ta nie jest dostępna w analizowanych systemach referencyjnych (zob. rozdział 3.4). Opracowany algorytm realizuje zadanie ANPR dla dowolnej liczby zaznaczonych stref ROI na obrazie, mogących przyjmować dowolne kształty, co umożliwia osiągnięcie wysokiego stopnia uniwersalności zastosowania.



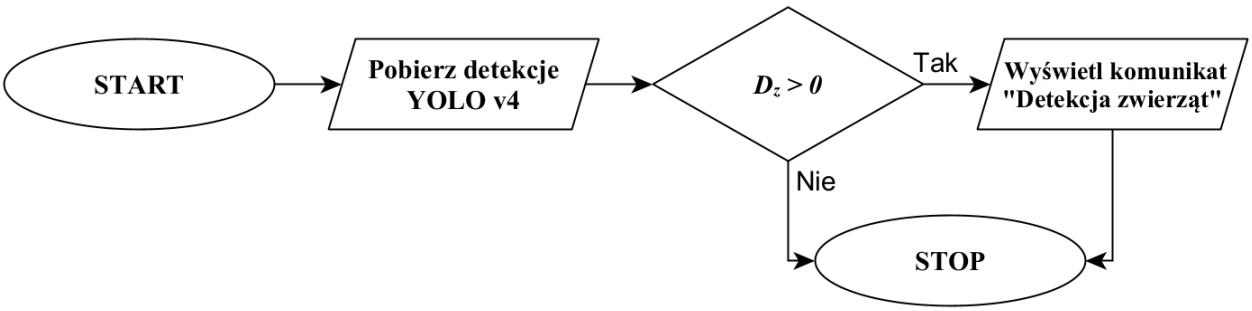
Rys. 6.6 Testy empiryczne algorytmu identyfikacji pojazdów

Algorytm korzysta z detektora YOLO v4 w celu detekcji pojazdów, detektora YOLO v4 tiny w celu detekcji tablic rejestracyjnych, detektora tekstu Tesseract OCR oraz dedykowanej logiki, umożliwiającej syntezę powyższych w celu identyfikacji właścicieli pojazdów zapisanych w bazie danych. Realizacja wstępnych testów umożliwiła określenie parametrów granicznych algorytmu, dla których prawdopodobieństwo prawidłowej detekcji numerów wynosi minimum 85%: maksymalny kąt widzenia tablicy w osi pionowej $\lambda = 135°$, maksymalny kąt widzenia tablicy w osi poziomej $\mu = 130°$, maksymalna odległość tablicy od kamery w poziomie $T_h = 10$ m, w pionie $T_v = 5$ m. Analiza otrzymanych wyników badań wskazuje na zadowalającą skuteczność algorytmu w rozpoznawaniu numerów tablic rejestracyjnych, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.2. Ponadto, analiza możliwości identyfikacji numerów tablic rejestracyjnych różnych typów wykazała zadawalającą zdolność algorytmu do rozpoznawania zarówno zwykłych tablic z białym tłem, jak również tablic tymczasowych z czerwonymi znakami, zabytkowych z żółtym tłem oraz tablic dedykowanych dla pojazdów elektrycznych (z zielonym tłem). Analiza jakościowa pracy algorytmu pozwala zauważyć, iż najlepsze rezultaty uzyskuje się w przypadkach detekcji tablic rejestracyjnych typu zwykłe oraz tymczasowe. Powodem tego może być bardziej efektywna binaryzacja obrazu, wynikająca z obecności białego koloru tła, niż w przypadku tablic o kolarze tła żółtym oraz zielonym. Pod względem orientacji kamery, najlepsze rezultaty zaobserwowano w przypadkach najazdu samochodu na wprost czujnika optycznego. Najczęstsze błędy detekcji zaobserwowano w przypadkach punktowego niedoświetlenia obrazu (obszaru tablicy) oraz przy skrajnych kątach widzenia, np. gdy tablica zostanie wykryta w dolnym prawym lub lewym rogu obrazu. Należy zwrócić uwagę, iż sprawność algorytmu w dużym stopniu zależy również od stanu jakości samej tablicy (uszkodzenia, zabrudzenia) oraz przejrzystości powietrza. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji tablic rejestracyjnych, szczególnie w punktowo zmiennych warunkach oświetleniowych oraz poprawy metody binaryzacji obrazu w przypadku analizy tablic z kolorowym tłem. Istotnym aspektem pracy algorytmu jest brak restrykcyjnych ograniczeń związanych z lokalizacją i orientacją kamery, dzięki czemu możliwe jest równoczesne użycie jej do innych zadań monitoringowych. Wyniki przeprowadzonych badań wskazują, iż efektywność proponowanego algorytmu jest zadowalająca, a jego implementacja znacząco poszerzy zakres zdolności operacyjnych projektowanego systemu IVSS.

6.2 Obecność zwierząt

W projektowanym systemie IVSS detekcja obecności zwierząt dotyczy identyfikacji psów i kotów poruszających się w strefie monitoringu wizyjnego. Psy i koty są najczęściej spotykanymi zwierzętami domowymi, których obecność może być niepożądana zwłaszcza w miejscach, takich jak magazyny żywności, obiekty przemysłowe czy obiekty infrastruktury medycznej.

Automatyczna detekcja obecności psów i kotów w takich miejscach pozwala na szybką interwencję. Oprócz minimalizacji potencjalnych szkód, zanieczyszczeń, a nawet zagrożeń związanych z ich obecnością możliwe jest wykrycie zagubionych zwierząt, co ułatwia ich odnalezienie i szybki powrót do właścicieli. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje, iż tego typu rozwiązania nie zostały jak do tej pory zaimplementowane w wielozadaniowych systemach IVSS. Wychodząc naprzeciw lukom badawczym, autor rozprawy opracował algorytm detekcji obecności zwierząt, którego schemat blokowy przedstawiono na Rys. 6.7.



Rys. 6.7 Schemat blokowy projektowanego algorytmu detekcji obecności zwierząt, gdzie: D_z – liczba wszystkich detekcji zwierząt

Praca algorytmu dotyczy analizy całego monitorowanego obszaru. W pierwszej kolejności pobierane są wszystkie detekcje YOLO v4 dla klas „*pies*” oraz „*kot*”. Jeśli wynik sumy detekcji obu klas jest niezerowy, wówczas na obrazie podglądu monitoringu wyświetlany zostaje komunikat informujący o wystąpieniu zdarzenia. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę obejmującą zlokalizowane zwierzę.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 16071 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.3.

Tab. 6.3 Zestawienie wyników badań empirycznych algorytmu identyfikacji obecności zwierząt

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
6204	99	2688	7080	55%	98%	47%	63%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w detekcji niepożądanej obecności zwierząt, o czym świadczą wskaźniki dokładności rzędu 55%, precyzji 98%, czułości 47% oraz F_1 63%.

Na Rys. 6.8 przedstawiono przykładowe ujęcia poklatkowe, zapisane podczas testów empirycznych algorytmu. Informacja dotycząca detekcji zwierzęcia wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym.

Rys. 6.8 Testy empiryczne algorytmu detekcji obecności zwierząt

Dodatkowo, zwierzę oznaczane jest czerwoną ramką ułatwiającą użytkownikowi jego lokalizację. Ujęcie znajdujące się w prawej dolnej części Rys. 6.8 stanowi przykład detekcji typu *FN*. Algorytm błędnie nie wykrywa obecności zwierzęcia na obrazie, w tym przypadku kota, pomimo jego obecności. W pozostałych przypadkach detekcji obecność zwierząt na obrazie realizowana jest w sposób prawidłowy.

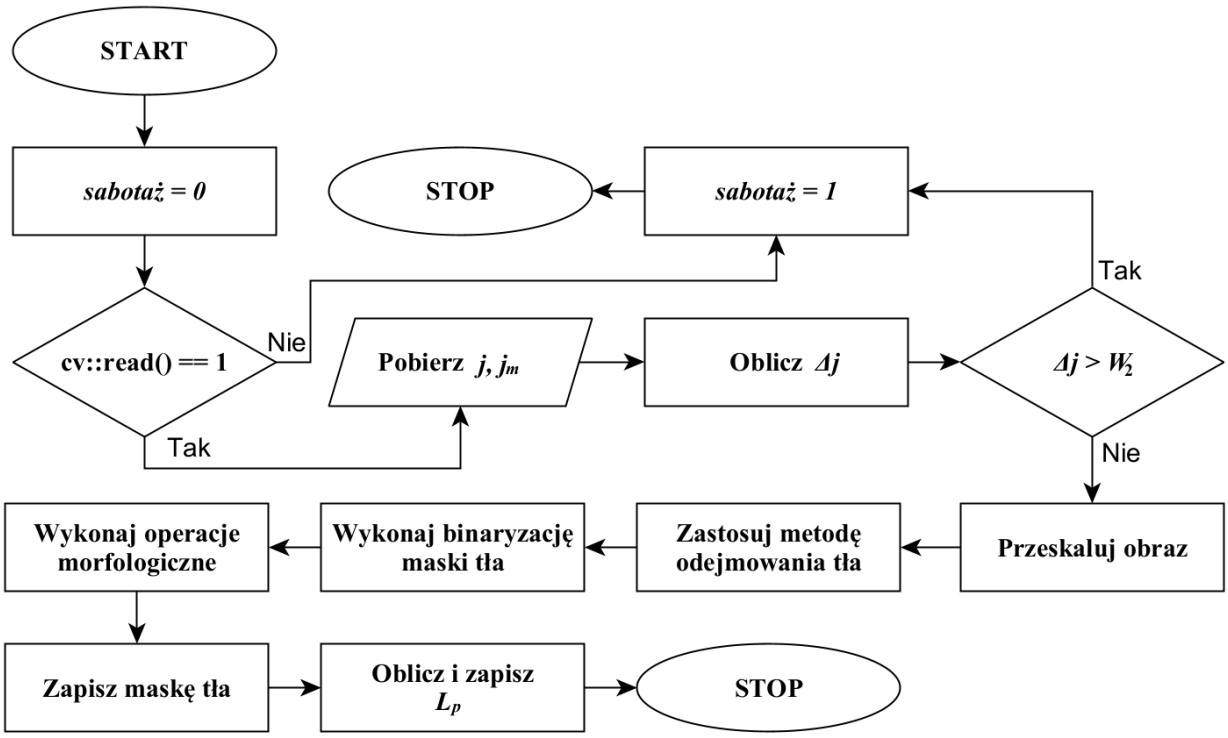
Podsumowując, opracowany algorytm detekcji zwierząt korzysta z detektora YOLO v4 w celu detekcji niepożądanej obecności psów oraz kotów w całym obszarze pola widzenia kamery. Funkcjonalność ta nie jest dostępna w analizowanych systemach referencyjnych (zob. rozdział 3.4). Realizacja badań opartych na testach empirycznych umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.3. Należy zwrócić uwagę, iż sprawność algorytmu całkowicie zależy od efektywności użytego detektora obiektów YOLO v4. Jednym z głównych powodów występowania błędnych detekcji typu *FN* może być charakterystyka zbioru danych MS-COCO [49], użytego w procesie uczenia bazowego modelu tego detektora. Większość zdjęć w zbiorze MS-COCO została wykonana z niskiej perspektywy widzenia, przypominającej zdjęcia profilowe. Obrazy rejestrowane przy użyciu kamery monitoringowej charakteryzują się natomiast relatywnie wysokim punktem obserwacji, gdzie pole widzenia zorientowane jest w dół pod zadanym kątem. Dodatkowo, obserwacja z zadanej wysokości wpływa na zmniejszenie rozmiaru zwierząt na obrazie, co ma negatywny wpływ na zdolność detekcji kotów oraz mniejszych ras psów. Pomimo relatywnie wysokiej liczby detekcji typu *FN*, algorytm jest w stanie prawidłowo zlokalizować znaczną część wystąpień zwierząt na obrazie przy zachowaniu niskiej liczby *FP*. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem redukcji detekcji typu *FN*. Proponuje się opracowanie dedykowanej bazy danych uczących, zawierającej zdjęcia kotów oraz psów zarejestrowane z klasycznego punktu widzenia kamery monitoringowej. W procesie tworzenia bazy należy uwzględnić różne rasy i rozmiary tych zwierząt, a także możliwe kombinacje ekspozycji przed kamerą, jak również przypadki częściowej okluzji. Tak opracowaną bazę danych można następnie użyć w procesie uczenia maszynowego modelu YOLO v4 lub YOLO V4 tiny. Rezultaty przeprowadzonych badań dowodzą, iż efektywność proponowanego algorytmu jest zadowalająca, co czyni go wartościowym i oryginalnym komponentem w projektowanym systemie IVSS.

6.3 Sabotaż kamery

Detekcja prób sabotażu kamery w projektowanym systemie IVSS dotyczy identyfikacji następujących zdarzeń: częściowego lub całkowitego zasłonięcia obiektywu kamery, obrotu obiektywu celem zmiany pola widzenia oraz próby oślepienia czujnika optycznego przy użyciu silnego źródła światła. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4)

wskazuje, iż tego typu rozwiązania nie zostały jak do tej pory zaimplementowane w wielozadaniowych systemach IVSS. Wychodząc naprzeciw lukom badawczym, autor rozprawy opracował algorytm detekcji sabotażu kamery, który przyjmuje formę dwuetapową. Pierwszy etap wykonywany jest w oddzielnym wątku, równolegle z detektorami YOLO v4, YOLO v4 tiny oraz OpenPose, co pozwala na redukcję czasu obliczeń. Drugi etap realizowany jest w pętli głównej programu i umożliwia użycie wyników wygenerowanych przez detektory, w celu ostatecznej weryfikacji wystąpienia zdarzenia sabotażu.

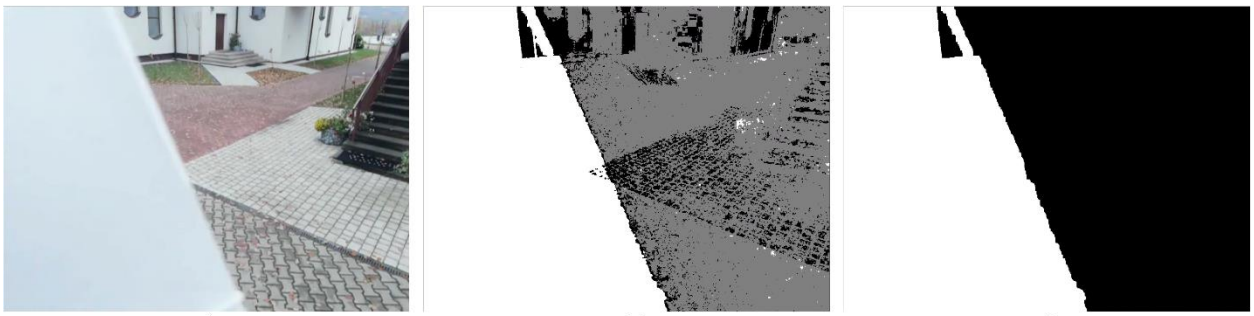
Na Rys. 6.9 przedstawiono schemat blokowy pierwszego etapu pracy zaprojektowanego algorytmu.



Rys. 6.9 Schemat blokowy pierwszego etapu pracy algorytmu detekcji sabotażu kamery, gdzie: *sabotaż* – flaga detekcji sabotażu, j – aktualna średnia ważona wartość jasności obrazu, j_m – wzorcowa średnia ważona wartość jasności obrazu, $\Delta j = |j - j_m|$, W_2 – wartość progowa zmiany jasności obrazu, L_p – liczba pikseli niezerowych maski tła

W pierwszym kroku sprawdzana jest poprawność pobrania nowego obrazu z kamery. Jeśli obraz nie został pobrany, funkcja *cv::read()* zwraca wartość negatywną, co skutkuje ustawieniem flagi *sabotaż* i przerwaniem pracy aplikacji z informacją dla użytkownika o możliwym wystąpieniu zdarzenia sabotażu, spowodowanym zakłóceniem transmisji sygnału wizyjnego (np. rozłączenie przewodów sygnałowych lub zasilających kamery). Jeśli obraz z kamery został poprawnie pobrany, realizowana jest analiza zmian średniej jasności obrazu $\Delta j = |j - j_m|$, gdzie j_m to średnia jasność wzorcowa (zapisana automatycznie przy starcie programu), natomiast j to aktualna średnia jasność obrazu, pobrana z algorytmu korekty jasności obrazu (zob. rozdział 5.2). W przypadku wykrycia nagłej zmiany jasności, przekraczającej wartość progową W_2, flaga *sabotaż* zostaje ustawiona, co skutkuje zgłoszeniem potencjalnego wystąpienia sabotażu,

mogącego wynikać z oślepienia obiektywu kamery. W przypadku braku detekcji nagłej zmiany jasności obraz zostaje przeskalowany w celu redukcji czasu obliczeń. Następnie wykonywana jest metoda odejmowania tła przy użyciu funkcji *cv::BackgroundSubtractorMOG2()*, która zwraca maskę tła obrazu. Dalej, wygenerowana maska podlega procesowi binaryzacji oraz operacjom morfologicznym typu otwieranie i zamykanie w celu redukcji szumów. W rezultacie czarnym kolorem oznaczone są obszary obrazu odpowiadające obrazowi tła wzorcowego, natomiast kolorem białym obszary znacząco różniące się od wzorca. Na Rys. 6.10 przedstawiono przykład analizy sabotowanej sceny z użyciem omówionej metody.

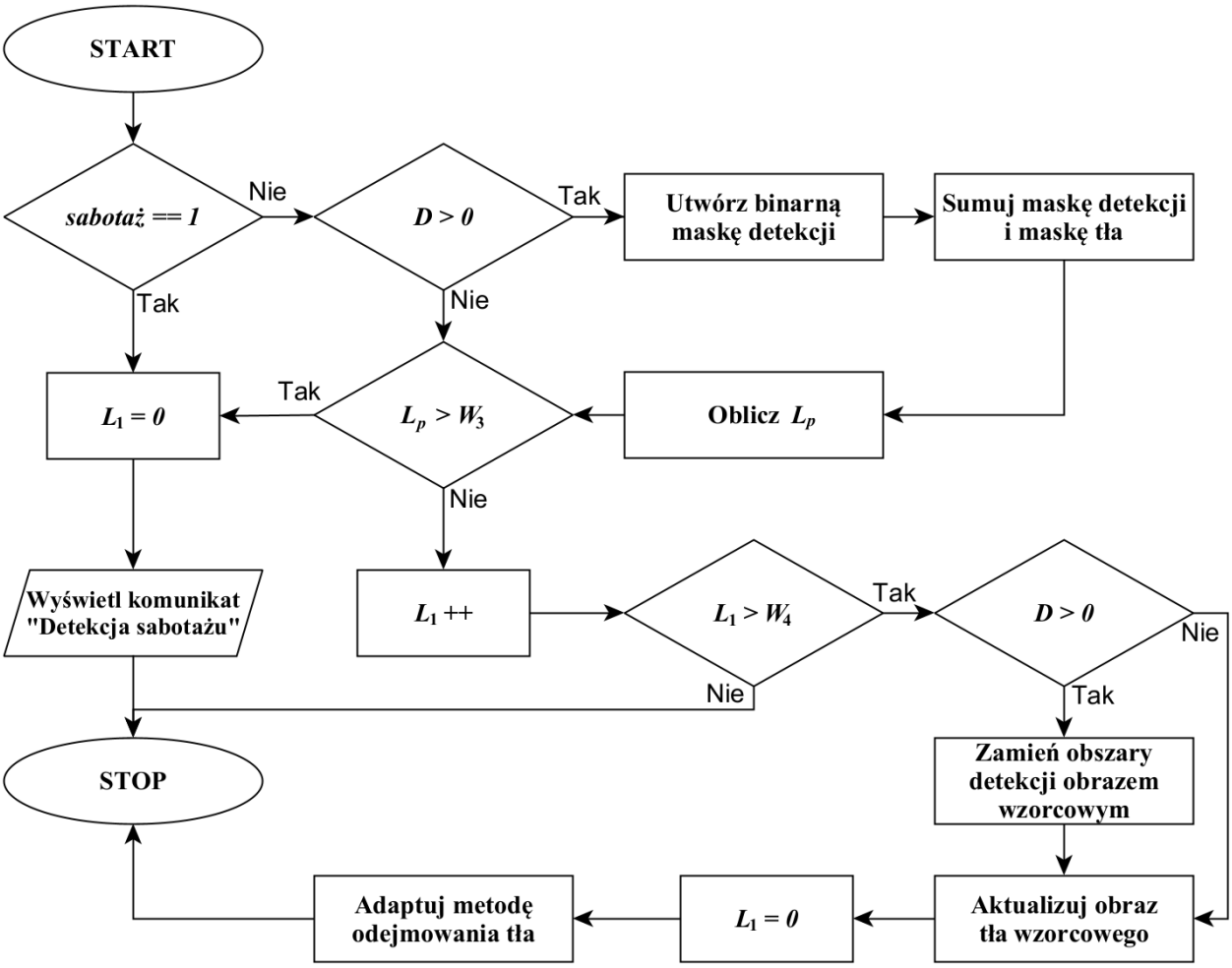


a) **b)** **c)**

Rys. 6.10 Przykład analizy sabotowanej sceny z zastosowaniem techniki odejmowania tła: a) oryginalny obraz z kamery, b) maska uzyskana w procesie odejmowania tła, c) maska tła po redukcji szumów

Rys. 6.10a ilustruje oryginalny obraz z kamery, której prawa część pola widzenia została częściowo przysłonięta. Na Rys. 6.10b zamieszczono widok maski uzyskanej w procesie odejmowania tła. Na Rys. 6.10c przedstawiono maskę tła po redukcji szumów. Wyznaczenie liczby pikseli niezerowych (białych) L_p tak przygotowanej maski tła umożliwia dokonanie oceny obrazu pod kątem możliwości wystąpienia istotnych zmian pierwszoplanowych.

Na Rys. 6.11 przedstawiono schemat blokowy drugiego etapu pracy zaprojektowanego algorytmu. Jeśli w poprzednim etapie flaga *sabotaż* została ustawiona, wówczas następuje zerowanie licznika braku detekcji sabotażu L_1, po czym na obrazie podglądu monitoringu zostaje wyświetlony komunikat informujący o wykryciu sabotażu kamery. W przeciwnym razie analizowana jest wartość L_p obliczona w poprzednim etapie pracy. Wartość L_p, przekraczająca wartość progową W_3, stanowi pierwszą przesłankę, dotyczącą możliwości wystąpienia celowego zasłonięcia lub obrotu obiektywu kamery. Przesłanka ta może zostać potwierdzona w dwóch przypadkach. Pierwszy przypadek dotyczy sytuacji, gdy liczba detekcji obiektów klasy „*pojazd*” oraz „*osoby*” D jest zerowa, co oznacza, że istotne zmiany pierwszoplanowe maski tła wynikają wyłącznie z prób sabotażu. Drugi przypadek dotyczy sytuacji, gdy wartość D jest niezerowa. Detekcje osób i pojazdów mogą wprowadzać istotne zmiany pierwszoplanowe w masce tła, dlatego w kolejnym kroku następuje ich wykluczenie poprzez zsumowanie maski detekcji i maski tła. Dalej następuje ponowne wyznaczenie L_p dla zaktualizowanej maski tła. Jeśli L_p nadal przewyższa wartość W_p, wówczas algorytm informuje o potencjalnym wystąpieniu zdarzenia



Rys. 6.11 Schemat blokowy drugiego etapu pracy algorytmu detekcji sabotażu kamery, gdzie: *sabotaż* – flaga detekcji sabotażu, D – liczba wszystkich śledzonych osób i pojazdów, L_1 – licznik kolejnych negatywnych detekcji sabotażu, L_p – liczba pikseli niezerowych maski tła, W_3 – wartość progowa liczby pikseli niezerowych, W_4 – wartość progowa liczby negatywnych detekcji sabotażu

sabotażu kamery. Takie rozwiązanie umożliwia wykluczenie błędnej detekcji sabotażu w przypadku pojawienia się dużej liczby osób lub pojazdów w monitorowanej przestrzeni.

Warunki środowiskowe obserwowanej sceny są zmienne w czasie. Z tego powodu zastosowana metoda odejmowania tła wymaga okresowej adaptacji. Adaptacja odbywa się tylko w przypadku braku detekcji sabotażu przez ustaloną liczbę następujących po sobie analizowanych ujęć $(L_1 > W_4)$. W celu uniknięcia adaptacji tła wzorcowego z występującymi na obrazie obiektami klasy „*pojazdy*” oraz „*osoby*”, algorytm usuwa tego typu obiekty zastępując je fragmentami dotychczasowego obrazu wzorcowego, zapisanego w pamięci. Ponadto, zapisywana jest również nowa wartość wzorcowa średniej jasności obrazu.

Badania empiryczne mające na celu określenie efektywności algorytmu, zostały zrealizowane przy użyciu zestawu nagrań wideo, który zawiera łącznie 16561 ujęć (zob. rozdział 4.4.1), gdzie 6233 ujęć dotyczy przypadku częściowego lub całkowitego zasłonięcia obiektywu kamery, 5216 ujęć dotyczy przypadku obrotu obiektywu oraz 5112 ujęć dotyczących przypadku oślepienia obiektywu kamery. Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.4.

Tab. 6.4 Zestawienie wyników badań empirycznych algorytmu detekcji sabotażu kamery

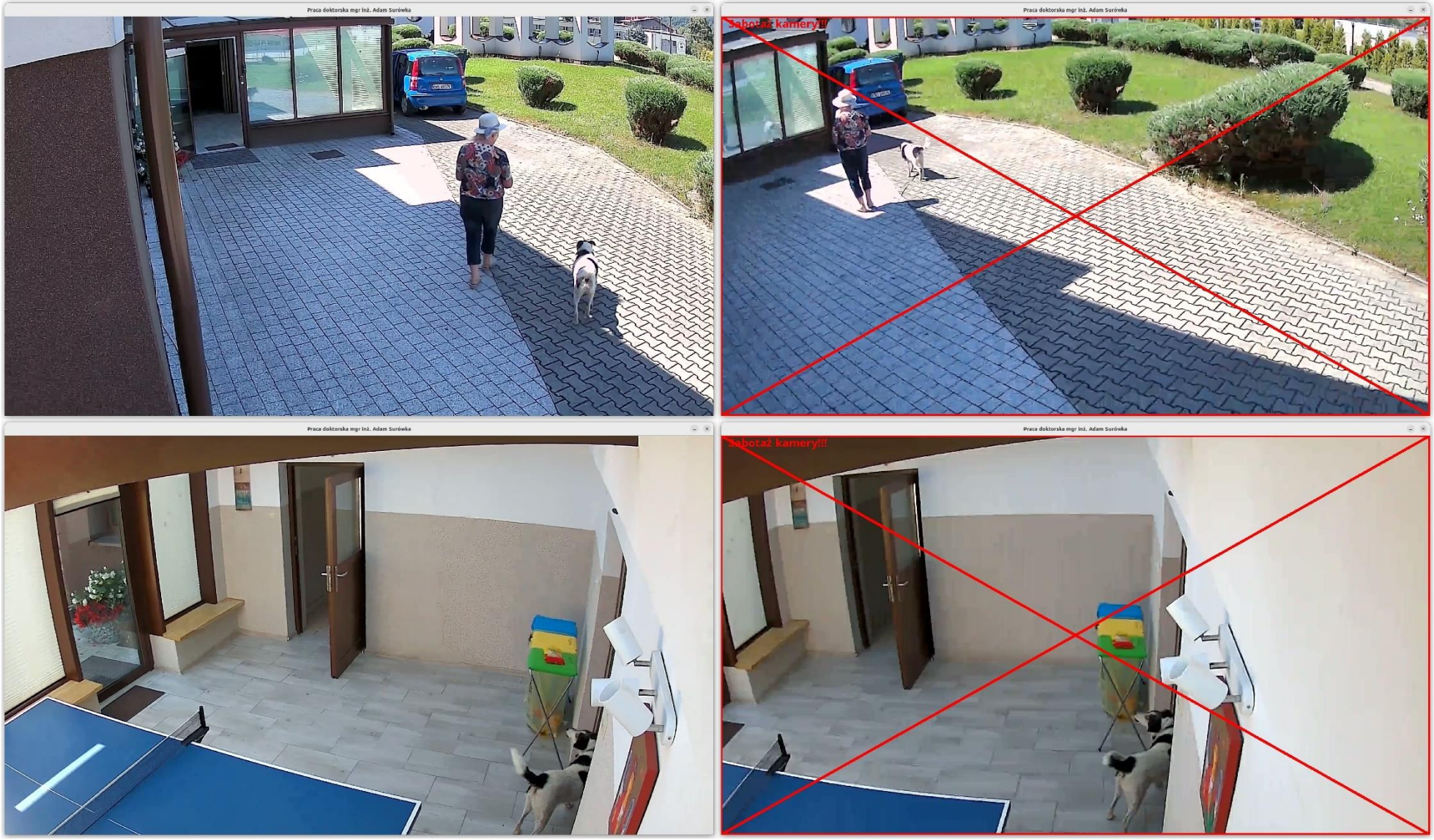
Typ sabotażu	*TP*	*FP*	*TN*	*FN*	Dokładność	Precyzja	Czułość	F_1
Zasłonięcie	3881	513	1672	167	89%	88%	96%	92%
Obrót	3259	250	1170	733	82%	93%	82%	87%
Oślepienie	3079	311	1472	54	93%	91%	98%	94%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w detekcji sabotażu, o czym świadczą wartości wyznaczników oceny jakości, a więc dokładność, precyzja, czułość oraz F_1. Dla detekcji zasłonięcia obiektywu wartości te są równe odpowiednio: 89%, 88%, 96% oraz 92%, dla obrotu obiektywu: 82%, 93%, 82% oraz 87%, dla oślepienia obiektywu: 93%, 91%, 98% oraz 94%.

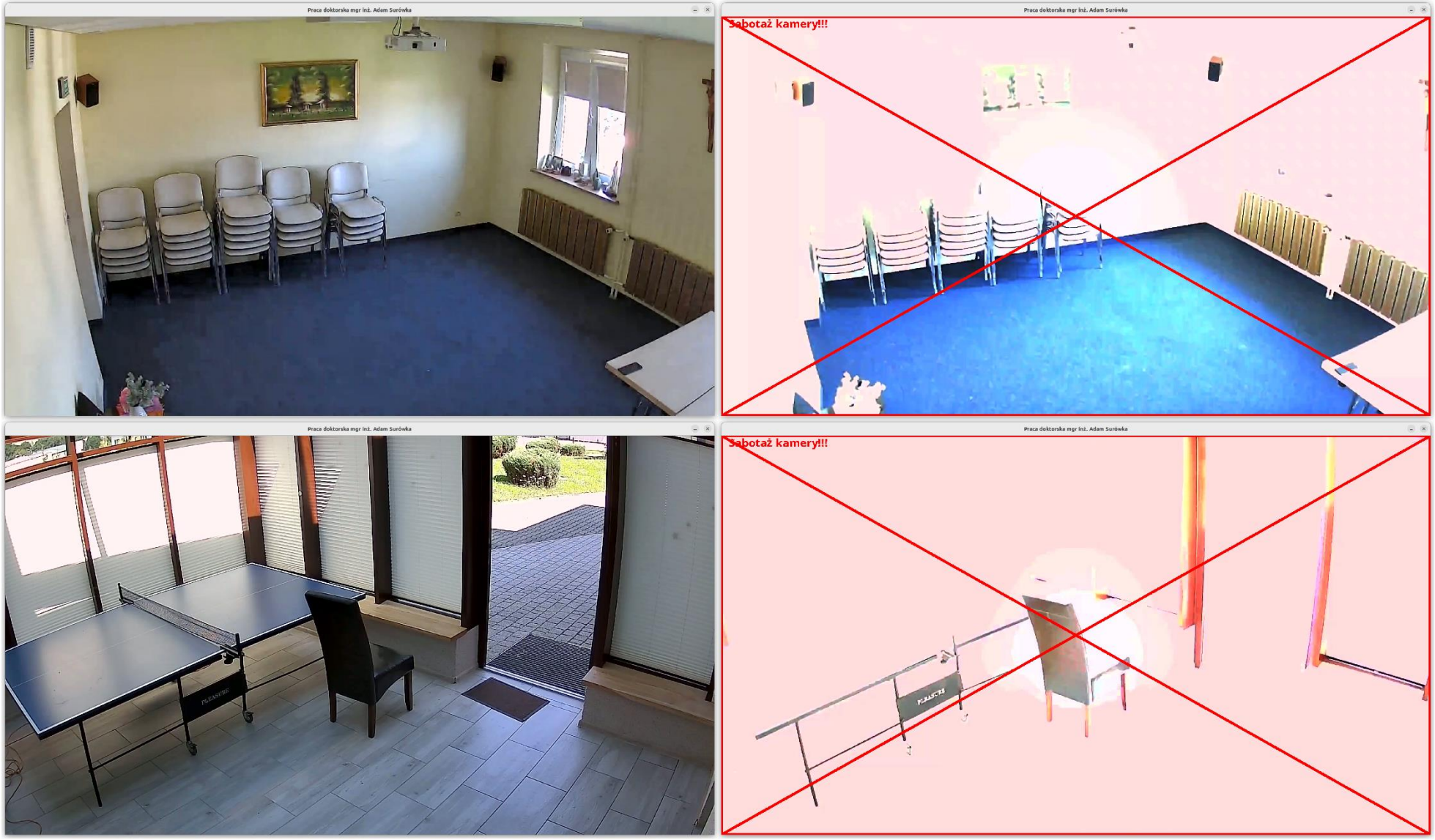
Podsumowując, zaprojektowany algorytm detekcji sabotażu kamery przeznaczony jest do identyfikacji następujących przypadków: częściowego lub całkowitego zasłonięcia obiektywu kamery, obrotu obiektywu oraz gwałtownej zmiany jasności obrazu (tzw. oślepienie obrazu). Funkcjonalność ta nie jest dostępna w analizowanych systemach referencyjnych (zob. rozdział 3.4). Opracowany algorytm przyjmuje formę dwuetapową. Pierwszy etap wykonywany jest w oddzielnym wątku, równolegle z detektorami YOLO v4, YOLO v4 tiny oraz OpenPose, co pozwala na redukcję czasu obliczeń. Drugi etap realizowany jest w pętli głównej programu i umożliwia użycie wyników wygenerowanych przez detektory obiektów, w celu ostatecznej weryfikacji wystąpienia zdarzenia sabotażu. W celu detekcji sabotażu kamery typu zasłonięcie oraz obrót obiektywu, zaprojektowany algorytm korzysta z funkcji odejmowania tła *cv::BackgroundSubtractorMOG2()* dostępnej w pakiecie biblioteki OpenCV. Dodatkowo, w celu usunięcia obiektów pierwszoplanowych z tła, używane są detekcje OpenPose oraz YOLO v4. W celu detekcji braku sygnału wizyjnego również użyto funkcjonalności dostępnych w pakiecie biblioteki OpenCV (funkcja *cv::read()*). W celu detekcji sabotażu kamery typu oślepienie, użyto wartości średniej jasności obrazu, pobranej z algorytmu automatycznej korekty jasności (zob. rozdział 5.2). Wyniki testów algorytmu wskazują na zadowalającą skuteczność w wykrywaniu sabotażu, z wartościami wyznaczników F_1 przekraczającymi 87% dla wszystkich trzech typów sabotażu. Przykłady detekcji sabotażu przez zasłonięcie, obrót oraz oślepienie obiektywu kamery przedstawiono odpowiednio na Rys. 6.12, 6.13 oraz 6.14.

Na Rys. 6.12 w lewym górnym rogu zaprezentowano wzorzec tła, natomiast pozostałe ujęcia przedstawiają rezultat prawidłowej detekcji sabotażu z użyciem różnego typu przysłon. Algorytm skutecznie wykrywał przypadki zasłaniania obiektywu kamery, niezależnie od rodzaju, koloru, struktury powierzchni oraz kształtu przysłony (np. dłoń, materiał tekstylny, tworzywo sztuczne).

Rys. 6.12 Testy empiryczne algorytmu detekcji sabotażu kamery, typ detekcji: zasłonięcie obiektywu



Rys. 6.13 Testy empiryczne algorytmu detekcji sabotażu kamery, typ detekcji: obrót obiektywu



Rys. 6.14 Testy empiryczne algorytmu detekcji sabotażu kamery, typ detekcji: oślepienie obiektywu

W trakcie badań zaobserwowano, iż jedną z częstszych przyczyn występowania detekcji typu *FP* może być pojawienie się na obrazie relatywnie dużych obiektów, niezidentyfikowanych przez detektory OpenPose oraz YOLO v4, które nie mogą przez to zostać wykluczone z algorytmu detekcji sabotażu (np. naczepa pojazdu dostawczego w strefie przeładunkowej, zasłaniająca większość pola widzenia kamery).

Przykłady detekcji sabotażu przez obrót obiektywu przedstawiono na Rys. 6.13. Po lewej stronie obrazu zamieszczono wzorce tła, natomiast po prawej odpowiednie rezultaty prawidłowej detekcji sabotażu. Detekcja obrotu obiektywu kamery była skuteczna w przypadkach, gdy kąt obrotu przekraczał 40°. W przypadku mniejszych kątów detekcja była mniej dokładna. Zaobserwowano także zależność dotyczącą jednorodności detali tła. Im obraz tła zawierał większą liczbę charakterystycznych detali, tym wyższa była sprawność detekcji zdarzenia sabotażu. Z kolei duża jednorodność (np. puste pomieszczenie o białych ścianach, podłodze i suficie) może powodować obniżenie skuteczności algorytmu.

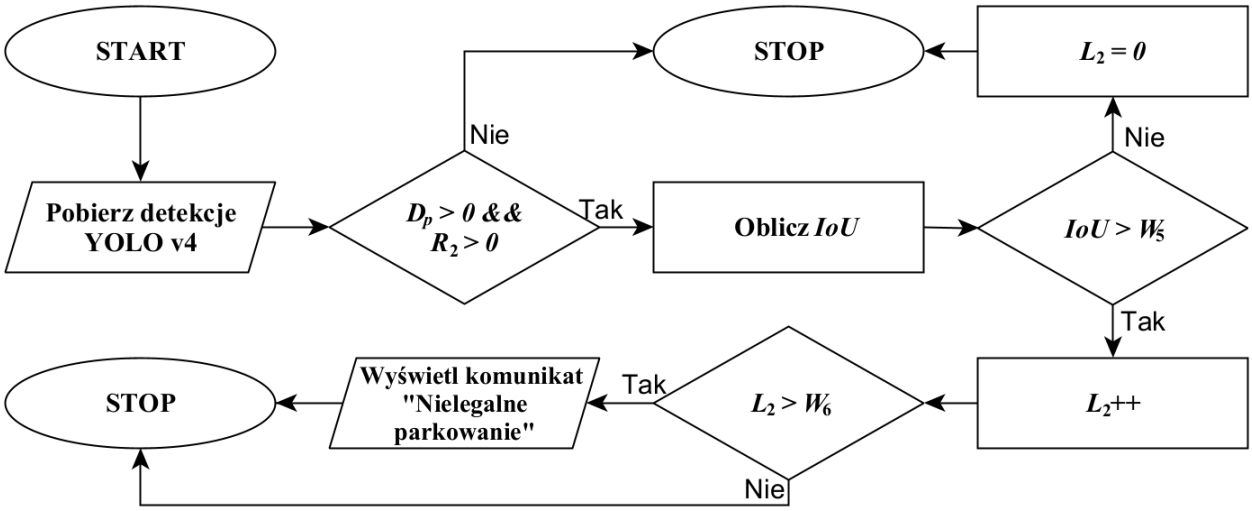
Detekcja oślepienia kamery silnym źródłem światła wykazała najwyższy stopień dokładności podczas obserwacji w klasycznych warunkach oświetlenia dziennego. Przykłady tego typu detekcji przedstawiono na Rys. 6.14. Po lewej stronie obrazu zamieszczono wzorce tła, natomiast po prawej odpowiednie rezultaty prawidłowej detekcji sabotażu. Detekcja tego typu sabotażu w warunkach skrajnie zmiennych w krótkim czasie może generować błędne alarmy (np. w przypadku nagłego włączenia oświetlenia w zaciemnionym pomieszczeniu lub odwrotnie). Wówczas algorytm nie będzie w stanie odpowiednio szybko zaadaptować nowego poziomu jasności tła.

Powyższe rezultaty badań wskazują, iż proponowany algorytm detekcji sabotażu kamery może stanowić cenne narzędzie w projektowanym systemie IVSS. Wczesne wykrywanie potencjalnych prób fizycznej manipulacji dokonywanych na kamerach monitoringowych może znacząco zwiększyć bezpieczeństwo nadzorowanych stref.

6.4 Nielegalne parkowanie

Funkcja wykrywania nielegalnego parkowania ma na celu automatyczną detekcję sytuacji, w której pojazd mechaniczny pozostaje w oznaczonej strefie przez czas dłuższy niż ustalona wartość progowa. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje tylko jedną pracę badawczą [101], w której funkcjonalność ta została zaimplementowana w wielozadaniowym systemie IVSS. Autorzy realizują detekcję pojazdów przy użyciu metod odejmowania tła, po czym analizują czas, w jakim ów obiekt pozostaje nieruchomy, co finalnie umożliwia podjęcie decyzji o wystąpieniu zdarzenia nielegalnego parkowania. Głównym problemem opisywanym przez autorów, jest użycie klasycznych metod odejmowania tła w celu detekcji i klasyfikacji obiektów. Wychodząc naprzeciw ograniczeniom ówczesnych technik

wizyjnych, opracowany przez autora rozprawy algorytm detekcji nielegalnego parkowania, korzysta z nowoczesnego detektora YOLO v4. Użyty detektor umożliwia precyzyjną klasyfikację różnych klas pojazdów, w tym „*samochód osobowy*”, „*samochód ciężarowy*”, „*autobus*”, oraz „*motocykl*”. Na Rys. 6.15 przedstawiono schemat blokowy zaprojektowanego algorytmu.



Rys. 6.15 Schemat blokowy algorytmu detekcji nielegalnego parkowania, gdzie: D_p – liczba wszystkich detekcji pojazdów, L_2 – licznik pozytywnych detekcji pojazdu w strefie ROI, R_2 – licznik stref ROI, W_5 – wartość progowa *IoU*, W_6 – wartość progowa liczby pozytywnych detekcji pojazdu w strefie ROI

W pierwszej kolejności pobierane są wszystkie detekcje YOLO v4 dotyczące wykrytych pojazdów, które wspólnie tworzą zbiór obiektów klasy „*pojazd*”. W przypadku niezerowej liczby detekcji obiektów klasy „*pojazd*” D_p oraz niezerowej liczby zaznaczonych stref wirtualnych R_2 dokonywana jest analiza wspólnej części obszaru zaznaczenia strefy oraz BB każdego z wykrytych pojazdów. Analiza opiera się na wyznaczeniu wartości wskaźnika

$$IoU = \left|\frac{A_r \cap B_p}{A_r \cup B_p}\right| \tag{6.1}$$

gdzie A_r oznacza obszar zaznaczenia strefy, natomiast B_p to obszar BB wykrytego pojazdu, przy czym obie te wartości dotyczą współrzędnych macierzy obrazu. Jeśli wartość wskaźnika *IoU* nie przekracza wartości progowej W_5, następuje zerowanie licznika detekcji nielegalnego parkowania L_2. W przeciwnym przypadku licznik L_2 zostaje zwiększony, a gdy osiągnie wartość większą niż wartość progowa W_6, następuje wyświetlenie komunikatu, dotyczącego detekcji nielegalnego parkowania pojazdu. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę, obejmującą nielegalnie parkujący pojazd.

W celu realizacji badań empirycznych użyto zestawu filmów o łącznej liczbie 18655 ujęć (zob. rozdział 4.4.1). Do celu identyfikacji zdarzenia nielegalnego parkowania stosowano oznaczenia ROI o różnych kształtach i rozmiarach, które nanoszono na obraz przy pomocy GUI podczas startu programu. Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.5.

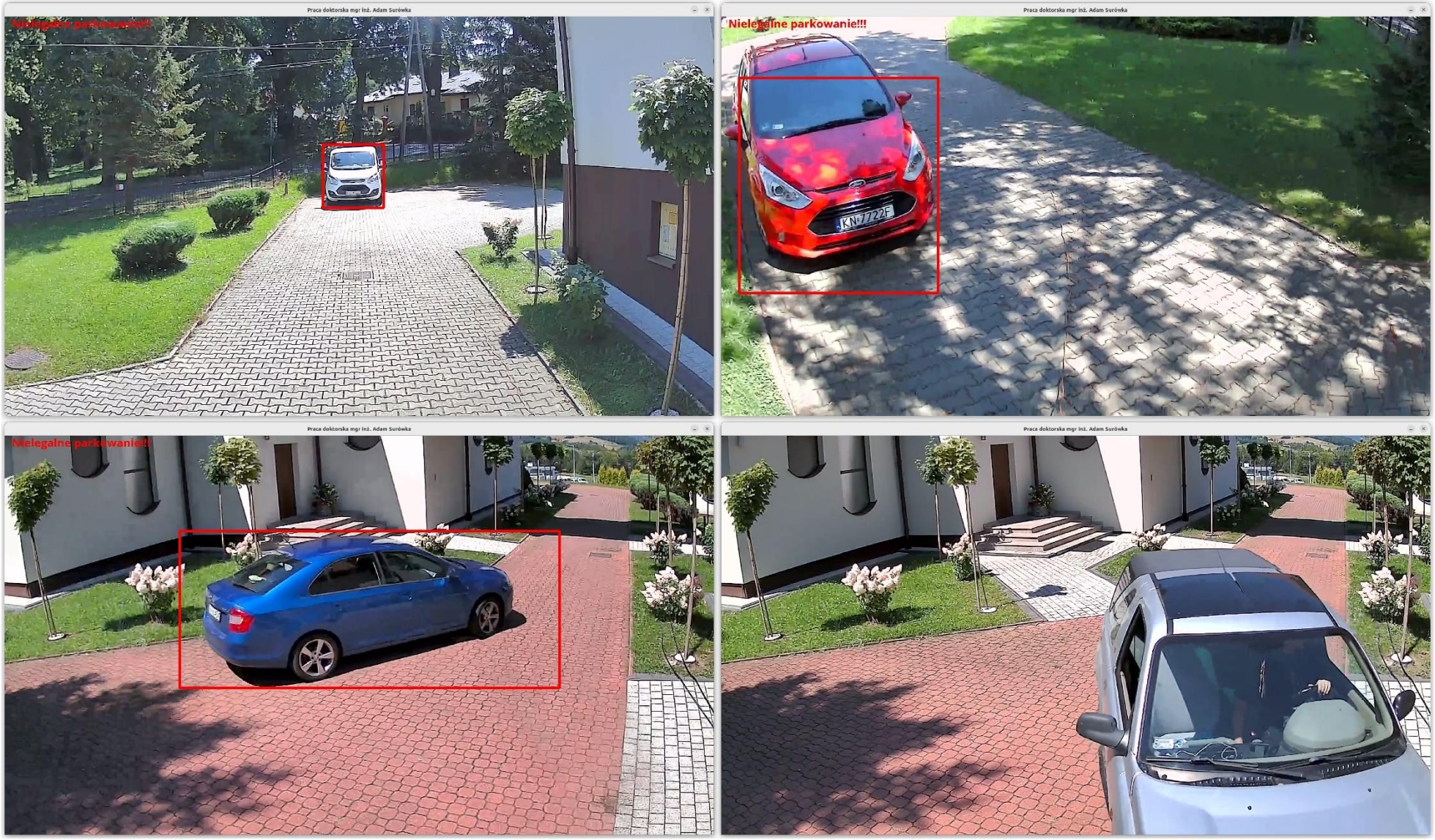
Tab. 6.5 Zestawienie wyników badań empirycznych algorytmu detekcji nielegalnego parkowania

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
13618	0	3673	1364	93%	100%	91%	95%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia nielegalnego parkowania, o czym świadczą wskaźniki dokładności rzędu 93%, precyzji 100%, czułości 91% oraz F_1 95%.

Na Rys. 6.16 przedstawiono przykładowe ujęcia poklatkowe, zapisane podczas testów empirycznych algorytmu. Informacja dotycząca detekcji zdarzenia wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo nielegalnie zaparkowany pojazd oznaczany jest czerwoną ramką, ułatwiającą użytkownikowi jego lokalizację. Ujęcie znajdujące się w prawej dolnej części Rys. 6.16 stanowi przykład detekcji typu *FN*. Algorytm błędnie nie wykrywa obecności pojazdu na obrazie, co może wynikać z częściowej okluzji. W pozostałych przypadkach detekcja nielegalnego parkowania realizowana jest w sposób prawidłowy.

Podsumowując, opracowany algorytm detekcji nielegalnego parkowania jest rozwinięciem algorytmu opisanego w pracy [101]. Oryginalność zaproponowanego rozwiązania polega na użyciu detektora YOLO v4 w celu precyzyjnej detekcji czterech popularnych typów pojazdów mechanicznych, do których należą: „*motocykl*”, „*samochód osobowy*”, „*samochód ciężarowy*” oraz „*autobus*”. Wyniki detekcji w połączeniu z dedykowaną logiką analizy czasu występowania pojazdu w wyznaczonej strefie ROI, umożliwiają identyfikację zdarzenia nielegalnego parkowania. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.5. Opracowany algorytm nie narzuca dodatkowych ograniczeń, związanych z lokalizacją i orientacją kamery, umożliwiając tym samym równoczesne jej użycie do innych zadań monitoringowych. Ponadto, algorytm umożliwia zaznaczenie dowolnej liczby stref wirtualnych, wyznaczających niedozwolone miejsca parkingowe. Strefy te mogą przyjmować dowolne kształty i rozmiary, co umożliwia osiągnięcie wysokiego stopnia uniwersalności zastosowania. Jednym z częstszych powodów występowania detekcji typu *FN*, są okluzje związane z chwilowym całkowitym lub częściowym zasłonięciem pojazdu. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji pojazdów. Proponuje się zastosowanie technik śledzenia pojazdów, celem wyeliminowania detekcji typu *FN* w przypadku wystąpienia całkowitej okluzji. Wyniki przeprowadzonych badań wskazują, iż efektywność proponowanego algorytmu jest zadowalająca, a jego implementacja znacząco poszerzy zakres zdolności operacyjnych projektowanego systemu IVSS.

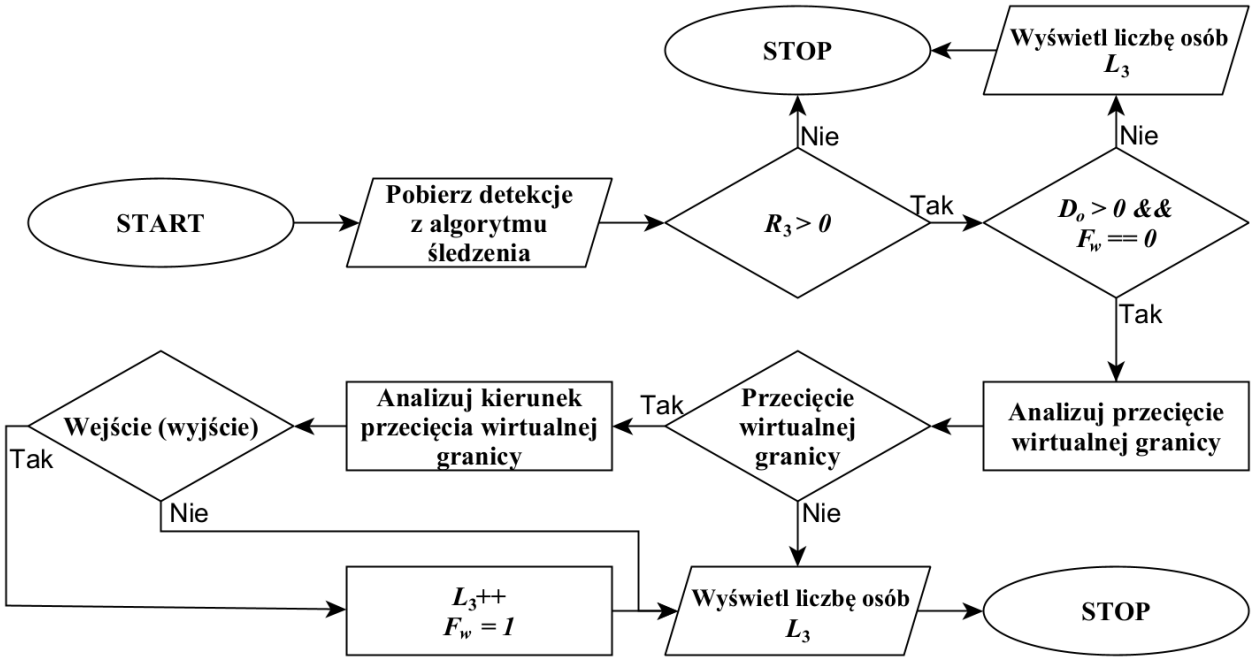


Rys. 6.16 Testy empiryczne algorytmu detekcji nielegalnego parkowania

6.5 Wejście (wyjście) ze strefy

Ilościowa analiza ruchu osób w obiektach użytku publicznego stanowi kluczowy aspekt strategii zarządzania oraz optymalizacji wielu operacji biznesowych. Informacja o liczbie osób wchodzących i wychodzących z obiektów handlowo-usługowych pozwala wspierać procesy zarządzania personelem, kreowania kampanii marketingowych czy planowania przestrzeni handlowej. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje tylko jedną pracę badawczą [109], w której autorzy implementują tą funkcję w wielozadaniowym systemie IVSS. Zaproponowane przez autorów rozwiązanie bazuje na zaznaczeniu wirtualnej linii przejścia, składającej się wyłącznie z dwóch punktów. Następnie analizowane są uśrednione wartości kątów natarcia ścieżek ruchu śledzonych osób względem zaznaczonej linii. Opisany „*model kątowy*" nie wyjaśnia jednak, w jaki sposób algorytm definiuje wewnętrzną i zewnętrzną stronę linii przejścia. Stanowi to podstawę do podważenia wiarygodności stwierdzenia, w którym autorzy deklarują zdolność algorytmu do automatycznego rozróżnienia zdarzenia wejścia i wyjścia niezależnie od układu obserwowanej sceny. Opracowany przez autora rozprawy algorytm, realizuje zadanie ilościowej analizy ruchu osób poprzez oznaczenie wirtualnej linii przejścia, składającej się z dwóch punktów. Ponadto, algorytm umożliwia użytkownikowi oznaczenie wewnętrznej strony linii przy pomocy dodatkowego punktu referencyjnego. Dzięki temu, system jest w stanie prawidłowo rozróżnić zdarzenie wejścia i wyjścia nie tylko dla dowolnego położenia i orientacji wirtualnej linii, ale również dla dowolnej orientacji i punktu montażu kamery (na zewnątrz lub wewnątrz monitorowanej przestrzeni). Opracowany algorytm korzysta z danych historycznych zgromadzonych przy użyciu algorytmu śledzenia MOT. Brak dodatkowych ograniczeń związanych z lokalizacją i orientacją kamery, umożliwia równoczesne jej użycie do innych zadań monitoringowych. Na Rys. 6.17 przedstawiono schemat blokowy zaprojektowanego algorytmu.

W pierwszej kolejności pobierane są detekcje z algorytmu śledzenia. Zapewnienie niezerowej liczby punktów tworzących linię przejścia R_3, jest warunkiem koniecznym do prawidłowego działania algorytmu. Wartość D_o określa liczbę aktualnie śledzonych osób na obrazie, natomiast F_w to flaga zapisywana w historii śledzonego obiektu określająca, czy dana osoba została już zakwalifikowana jako wchodząca (lub wychodząca) w poprzednich iteracjach algorytmu. Jeśli wartość D_o jest niezerowa oraz flaga F_w wskazuje na brak zakwalifikowania osoby, następuje tworzenie linii ścieżki ruchu śledzonej osoby. W tym celu, z algorytmu śledzenia pobierane są 2 ostatnie, niezerowe punkty środka dolnej krawędzi BB osoby. Kluczowy etap algorytmu polega na sprawdzaniu przecięć pomiędzy odcinkiem linii wirtualnej przejścia oraz odcinkiem linii ścieżki ruchu osoby, tworzonej na podstawie dwóch ostatnich punktów lokalizacji. Dla pary



Rys. 6.17 Schemat blokowy algorytmu analizy wejścia (wyjścia), gdzie: D_o – liczba wszystkich śledzonych osób, F_w – flaga wejścia (wyjścia) osoby, L_3 – licznik pozytywnych detekcji wejścia (wyjścia) osoby ze strefy, R_3 – licznik granic wirtualnych

odcinków obliczana jest orientacja punktów, umożliwiając określenie ich wzajemnego położenia. Na podstawie orientacji punktów algorytm ocenia, czy odcinki się przecinają. Uwzględniane są zarówno przypadki ogólne, jak i szczególne, takie jak kolinearność odcinków. Opisywana metoda detekcji przecięcia dwóch linii została zaadaptowana z [142].

Wyznaczenie kierunku przecięcia ścieżki jest możliwe dzięki znajomości punktu wzorcowego, określającego położenie wnętrza monitorowanej przestrzeni oraz chronologii punktów ścieżki ruchu osoby. W zależności od konfiguracji programowej, istnieje możliwość zliczania wejść bądź wyjść z przestrzeni. W przypadku identyfikacji wybranego zdarzenia, algorytm zwiększa globalny licznik wejść (lub wyjść) L_3 oraz ustawia flagę F_w na wartość 1, aby dana osoba nie była brana pod uwagę w dalszej sekwencji obrazów.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 15202 ujęć (zob. rozdział 4.4.1). W celu identyfikacji zdarzenia wejścia osoby do strefy zastosowano oznaczenie w postaci wirtualnej linii, naniesionej na obraz przy pomocy GUI podczas startu programu. Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.6.

Tab. 6.6 Zestawienie wyników badań empirycznych algorytmu analizy wejścia (wyjścia) ze strefy

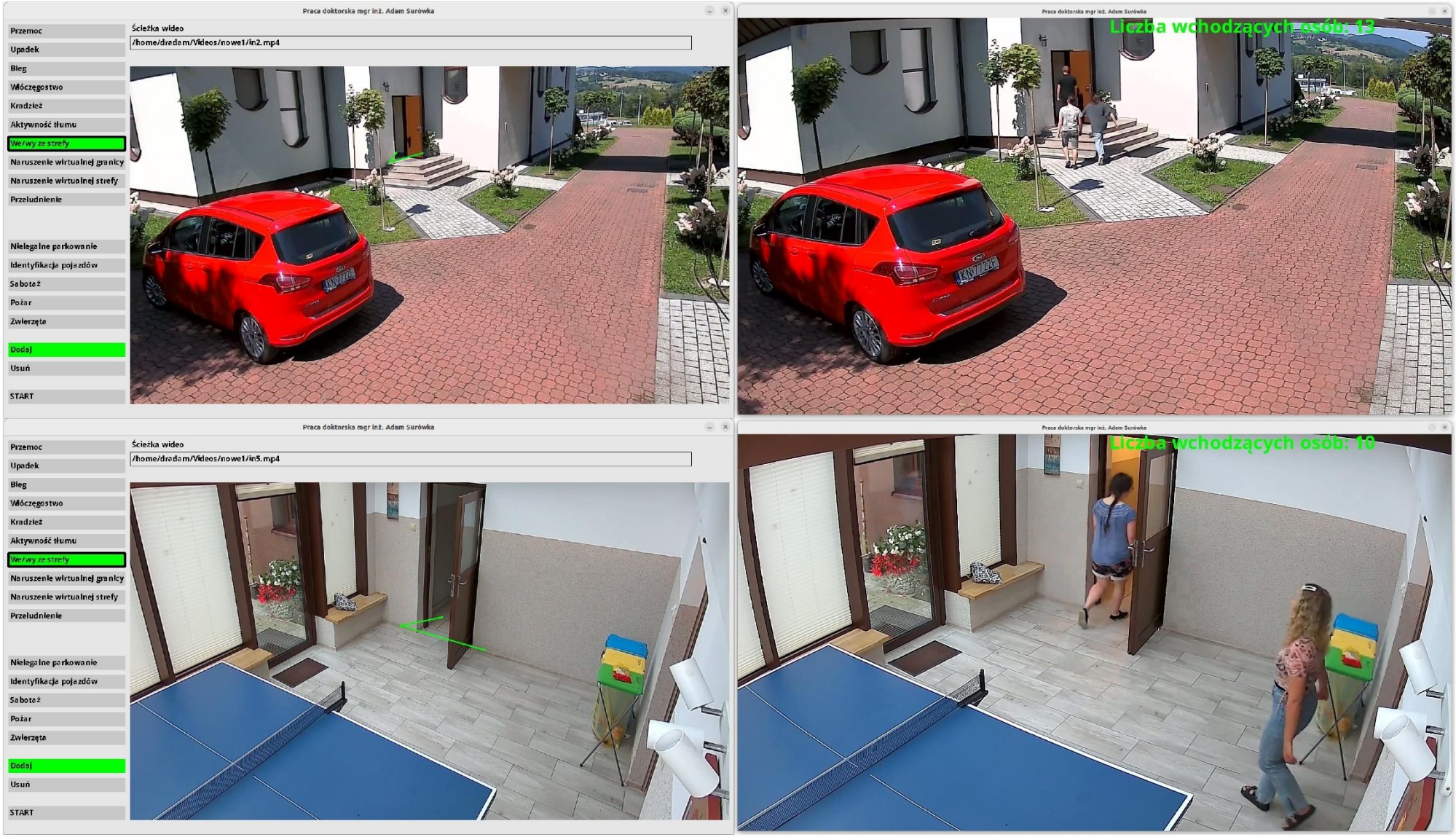
TP	*FP*	*TN*	*FN*	**Dokładność**	**Precyzja**	**Czułość**	F_1
430	51	14645	76	99%	89%	85%	87%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia wejścia (wyjścia) ze strefy, o czym świadczą wskaźniki dokładności rzędu 99%, precyzji 89%, czułości 85% oraz F_1 87%.

Rezultaty działania algorytmu przedstawiono na Rys. 6.18. Po lewej stronie obrazu znajdują się widoki startowego okna GUI systemu, na których można zaobserwować sposób nanoszenia wirtualnych linii, stanowiących wyznacznik detekcji wejścia. Każda linia składa się z trzech punktów, przy czym kolejność zaznaczania punktów jest istotna. Dwa pierwsze punkty tworzą wirtualną linię przejścia, natomiast trzeci określa wewnętrzną część monitorowanej strefy. Ujęcia po lewej stronie obrazu przedstawiają odpowiednio rezultaty detekcji zdarzenia wejścia w postaci komunikatu tekstowego, informującego o liczbie osób wchodzących.

Podsumowując, opracowany algorytm detekcji wejścia (wyjścia) ze strefy, realizuje zadanie ilościowej analizy ruchu osób przechodzących w ustalonym kierunku przez wirtualną linię przejścia, składającą się z dwóch punktów korzystając z logiki opisanej w [142]. Oryginalność rozwiązania wynika z możliwości manualnego oznaczenia strony wewnętrznej linii przy pomocy dodatkowego punktu referencyjnego na obrazie. Dzięki temu, w przeciwieństwie do pracy referencyjnej [109], opracowany algorytm jest w stanie prawidłowo rozróżnić zdarzenie wejścia i wyjścia nie tylko dla dowolnego położenia i orientacji wirtualnej linii, ale również dla dowolnej orientacji i punktu montażu kamery (na zewnątrz lub wewnątrz monitorowanej przestrzeni). Opracowany algorytm korzysta z danych historycznych zgromadzonych przy użyciu algorytmu śledzenia MOT. Brak dodatkowych ograniczeń związanych z lokalizacją i orientacją kamery, umożliwia równoczesne jej użycie do innych zadań monitoringowych. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.6. Zaobserwowano, iż jednym z częstszych powodów występowania detekcji typu *FN*, są okluzje związane z chwilowym całkowitym zasłonięciem osoby. Z kolei detekcje typu *FP* dotyczą najczęściej przypadku, w którym pojedyncza osoba przechodząca przez wirtualną linię zostaje zakwalifikowana podwójnie. Powodem tego może być zaszumienie ścieżki trajektorii ruchu osoby, wynikające z niedokładności detekcji osób lub/i braku dostatecznej filtracji ścieżki w algorytmie śledzenia. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji osób oraz filtracji ścieżek przejść w algorytmie śledzenia.

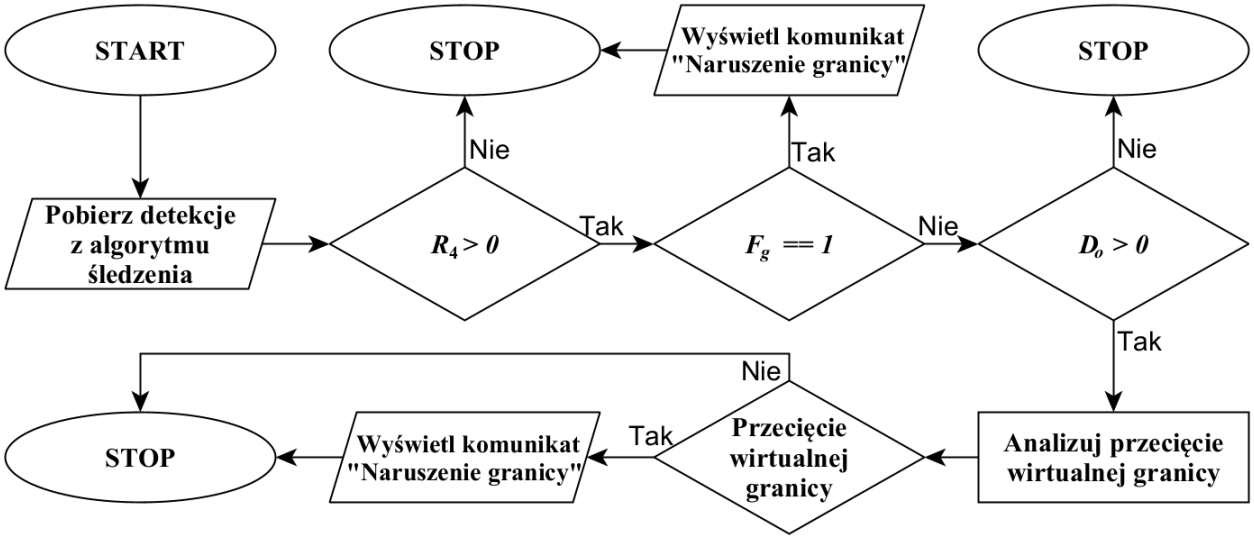
Rezultaty przeprowadzonych badań dowodzą, iż efektywność proponowanego algorytmu jest zadowalająca, co czyni go wartościowym komponentem w projektowanym systemie IVSS.



Rys. 6.18 Testy empiryczne algorytmu analizy wejścia (wyjścia) ze strefy

6.6 Naruszenie wirtualnej granicy

Detekcja naruszenia wirtualnej granicy dotyczy identyfikacji zdarzeń, w których ścieżka ruchu osoby przekracza linię wirtualną (granicę), zdefiniowaną programowo na obrazie monitoringu. Funkcja ta odgrywa kluczową rolę w zwiększeniu bezpieczeństwa obiektów użytku publicznego, takich jak szkoły czy obiekty handlowo-usługowe, jak również prywatnych posesji umożliwiając szybkie podjęcie działań prewencyjnych. Jest to szczególnie pożądana funkcja w przypadku, gdy w monitorowanej strefie dozwolony jest ruch osób po obu stronach wirtualnej granicy (np. przed i za ogrodzeniem posesji), natomiast uruchomienie procedury alarmowej ma nastąpić wyłącznie w momencie identyfikacji przejścia przez wyznaczoną linię (ogrodzenie). Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje tylko dwie prace badawcze [101,112], w których autorzy implementują tę funkcję w wielozadaniowych systemach IVSS. Podejście opisane w pracy [101], skoncentrowane jest na analizie cech wyglądu i ruchu w obszarze wokół wirtualnej granicy zamiast śledzenia ścieżek ruchu osób poruszających się w całym polu widzenia kamery. Praca [112] przedstawia natomiast metodę analizy modelu szkieletowego osoby wygenerowanego przy użyciu detektora OpenPose. W tym przypadku, detekcja zdarzenia realizowana jest na podstawie lokalizacji punktorów szkieletu osoby względem zaznaczonej wirtualnej granicy na aktualnym ujęciu, bez uwzględnienia historii ścieżki ruchu osoby. Zaprojektowany przez autora rozprawy algorytm detekcji naruszenia wirtualnej granicy, różni się od przedstawionych koncepcji. Główne różnice dotyczą użycia danych historycznych, zawierających ścieżki ruchu wszystkich śledzonych osób poruszających się na obrazie oraz opracowania logiki decyzyjnej w oparciu o zmodyfikowany algorytm analizy wejścia (wyjścia) ze strefy (zob. rozdział 6.5). Na Rys. 6.19 przedstawiono schemat blokowy zaprojektowanego algorytmu.



Rys. 6.19 Schemat blokowy algorytmu detekcji naruszenia wirtualnej granicy, gdzie: D_o – liczba wszystkich śledzonych osób, F_g – flaga przekroczenia wirtualnej linii przez osobę, R_4 – licznik granic wirtualnych

W pierwszej kolejności pobierane są detekcje z algorytmu śledzenia. Zapewnienie niezerowej liczby punktów tworzących wirtualną linię granicy R_4, jest warunkiem koniecznym do prawidłowego działania algorytmu. Następnym sprawdzanym warunkiem jest stan flagi F_g, zapisywanej w historii śledzonego obiektu, która określa, czy dana osoba przekroczyła już granicę w poprzednich iteracjach algorytmu. Jeśli dana osoba przekroczyła linię granicy wirtualnej w poprzednich iteracjach (flaga F_g została ustawiona na wartość 1), algorytm utrzymuje ten stan, informując o zdarzeniu naruszenia wirtualnej granicy na każdym kolejnym ujęciu. W przeciwnym razie, jeśli wartość D_o jest niezerowa, następuje tworzenie linii ścieżki ruchu śledzonej osoby. W tym celu, z algorytmu śledzenia pobierane są ostatnie, niezerowe punkty środka dolnej krawędzi BB osoby. Kluczowy etap algorytmu polegający na sprawdzaniu przecięć pomiędzy odcinkami linii wirtualnej granicy oraz linii ścieżki przejścia, realizowany jest na zasadzie opisanej w rozdziale 6.5. Jedyną różnicą jest konieczność sprawdzenia wszystkich par odcinków tworzących wirtualną linię granicy. W przypadku detekcji zdarzenia przecięcia linii wirtualnej granicy z linią ścieżki ruchu osoby, flaga F_g ustawiana jest na wartość 1. Oznacza to uruchomienie procedury alarmowej, polegającej na wyświetleniu stosownego komunikatu na obrazie podglądu monitorowanej sceny. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę, obejmującą sylwetkę osoby dokonującej naruszenia wirtualnej granicy.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 14270 ujęć (zob. rozdział 4.4.1). W celu identyfikacji zdarzenia naruszenia wirtualnej granicy stosowano oznaczenia w postaci wirtualnych linii, naniesionych na obraz przy pomocy GUI podczas startu programu. Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.7.

Tab. 6.7 Zestawienie wyników badań empirycznych algorytmu detekcji naruszenia wirtualnej granicy

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
120	0	14150	0	100%	100%	100%	100%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia naruszenia wirtualnej granicy, o czym świadczą wskaźniki dokładności, precyzji, czułości oraz F_1 równe 100%.

Rezultaty działania algorytmu przedstawiono na Rys. 6.20. Po lewej stronie obrazu znajdują się widoki startowego okna GUI systemu, na których można zaobserwować sposób nanoszenia wirtualnych linii, stanowiących wyznacznik detekcji naruszenia granicy.

Rys. 6.20 Testy empiryczne algorytmu detekcji naruszenia wirtualnej granicy

Każda linia może składać się z dowolnej liczby punktów, umożliwiając precyzyjne oznaczenie ogrodzenia o dowolnym kształcie i rozmiarze. Zaznaczone linie są w kolorze jasnozielonym. Ujęcia po prawej stronie obrazu przedstawiają odpowiednio rezultaty prawidłowej detekcji zdarzenia. Informacja dotycząca detekcji wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo, osoba przekraczająca granicę w miejscu niedozwolonym oznaczana jest czerwoną ramką, co ułatwia jej lokalizację.

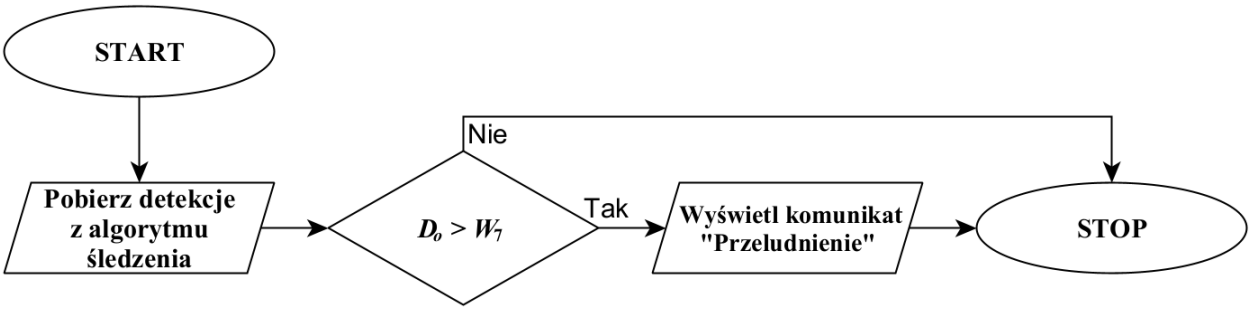
Podsumowując, opracowany algorytm detekcji naruszenia wirtualnej granicy, korzysta z detektorów OpenPose oraz YOLO v4 w celu detekcji osób na obrazie, algorytmu śledzenia osób oraz dedykowanej logiki, umożliwiającej detekcję przekroczenia wirtualnej linii (zob. rozdział 6.5) celem ostatecznej identyfikacji naruszenia wirtualnej granicy. Użytkownik ma możliwość zaznaczenia dowolnej liczby granic, składających się z dowolnej liczby punktów załamania, co wpływa na wysoki stopień uniwersalności zastosowania. Zaproponowane rozwiązanie stanowi rozwinięcie algorytmu opisanego w pracy [112]. Użycie danych historycznych dotyczących ścieżek ruchu osób, umożliwia nie tylko stwierdzenie, czy osoba znajduje się po niewłaściwej stronie linii wirtualnej, ale również pozwala określić, czy doszło do jej fizycznego przekroczenia. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.7. W trakcie realizacji badań nie zaobserwowano żadnych błędnych detekcji. Potencjalne detekcje typu *FN* mogą natomiast wystąpić w przypadku okluzji, związanych z chwilowym całkowitym zasłonięciem osoby lub brakiem detekcji osoby na obrazie. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji osób oraz filtracji ścieżek ruchu w algorytmie śledzenia. Wyniki przeprowadzonych badań wskazują, iż efektywność proponowanego algorytmu jest zadowalająca, a jego implementacja znacząco poszerzy zakres zdolności operacyjnych projektowanego systemu IVSS.

6.7 Przeludnienie

Jednym z głównych celów detekcji przeludnienia jest zapewnienie bezpieczeństwa w obiektach użytku publicznego w sytuacjach awaryjnych, takich jak pożary, trzęsienia ziemi czy ataki terrorystyczne, gdzie szybka i dokładna ocena liczby osób przebywających w danym obszarze, jest kluczowa dla skutecznej ewakuacji i minimalizacji ryzyka strat w ludziach. Ponadto, informacja o liczbie osób pomaga zapobiegać zatłoczeniu, umożliwia płynne zarządzanie ruchem oraz zwiększa efektywności działania systemów transportowych i innych usług publicznych. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje tylko dwie pozycje sprzed dwudziestu lat [97,99] opisujące implementacje tej funkcji w wielozadaniowym systemie IVSS. Ze względu na ówczesne ograniczenia w dziedzinie technik widzenia

komputerowego, implementacja detekcji przeludnienia została oparta na klasycznej metodzie odejmowania tła i analizie cechach wyglądu. Wychodząc naprzeciw lukom badawczym w tym zakresie, autor rozprawy zaprojektował algorytm, zapewniający możliwość precyzyjnej detekcji przeludnienia przy użyciu logiki, opartej na syntezie wyników detektorów YOLO v4, OpenPose oraz algorytmie śledzenia MOT.

Na Rys. 6.21 przedstawiono schemat blokowy zaprojektowanego algorytmu.



Rys. 6.21 Schemat blokowy zaprojektowanego algorytmu detekcji przeludnienia, gdzie: D_o – liczba wszystkich śledzonych osób, W_7 – wartość progowa maksymalnej liczby osób

Podczas startu programu przy pomocy GUI, użytkownik wprowadza do systemu wartość dopuszczalnej liczby osób, mogącej znajdować się jednocześnie na monitorowanym obszarze (wartość progowa W_7). W trakcie działania programu, algorytm na bieżąco porównuje liczbę aktualnie śledzonych osób na obrazie D_o z wartością W_7. Jeśli wartość D_o jest większa od W_7, wówczas na obrazie zostaje wyświetlony stosowny komunikat. W zależności od zapotrzebowania, algorytm może zostać również skonfigurowany do ciągłego wyświetlania liczby aktualnych detekcji D_o.

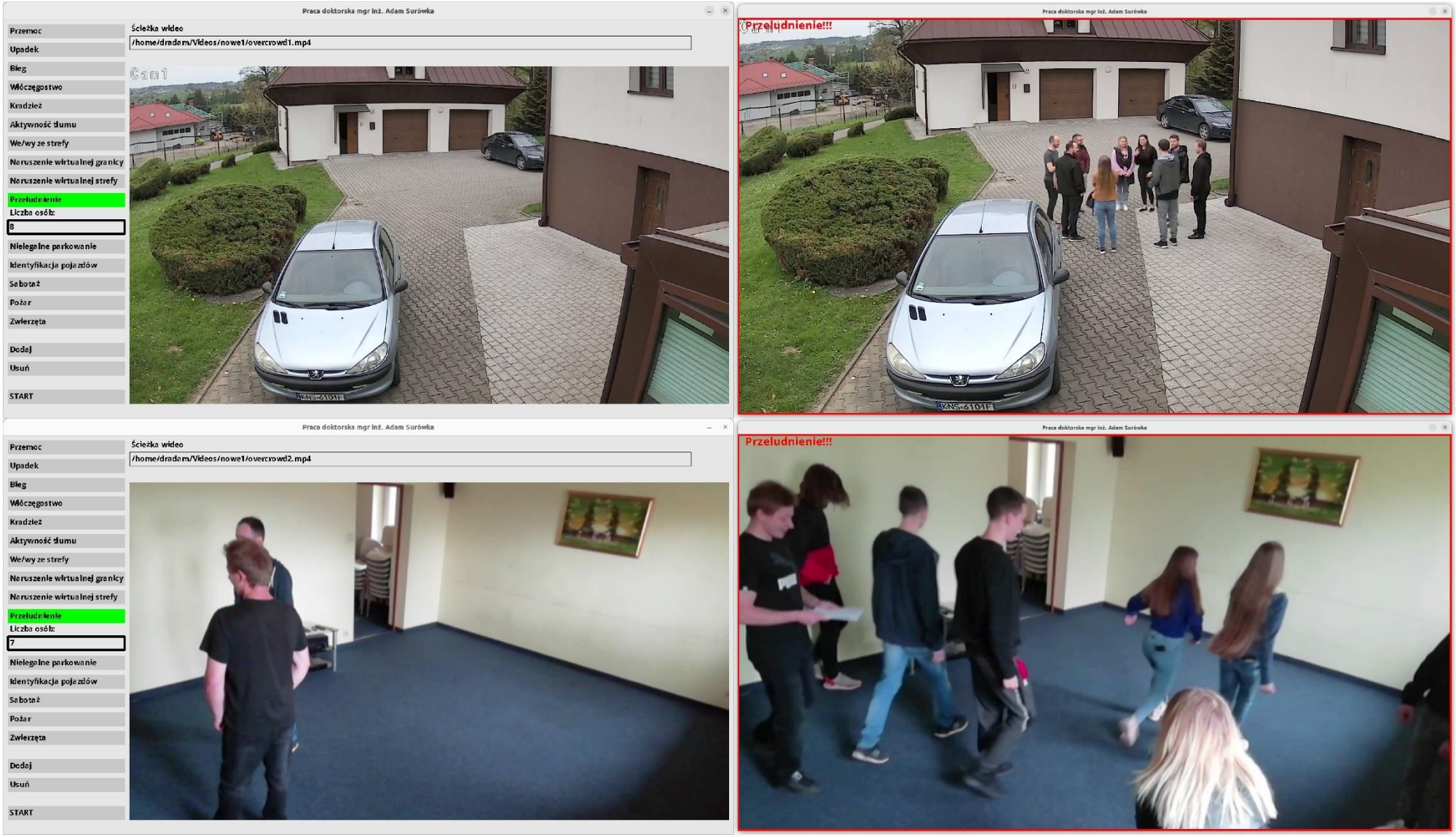
Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 16904 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.8.

Tab. 6.8 Zestawienie wyników badań empirycznych algorytmu detekcji przeludnienia

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
11367	49	3299	2189	87%	100%	84%	91%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia przeludnienia, o czym świadczą wskaźniki dokładności rzędu 87%, precyzji 100%, czułości 84% oraz F_1 91%.

Rezultaty działania algorytmu przedstawiono na Rys. 6.22. Po lewej stronie obrazu znajdują się widoki startowego okna GUI systemu, na których można zaobserwować sposób wprowadzania liczby osób, stanowiącej wartość graniczną detekcji przeludnienia.



Rys. 6.22 Testy empiryczne algorytmu detekcji przeludnienia

Ujęcia po prawej stronie obrazu przedstawiają odpowiednio rezultaty detekcji zdarzenia. Informacja dotycząca detekcji wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo, na obwodzie obrazu rysowana jest czerwona ramka, informująca o zdarzeniu dotyczącym całej obserwowanej sceny.

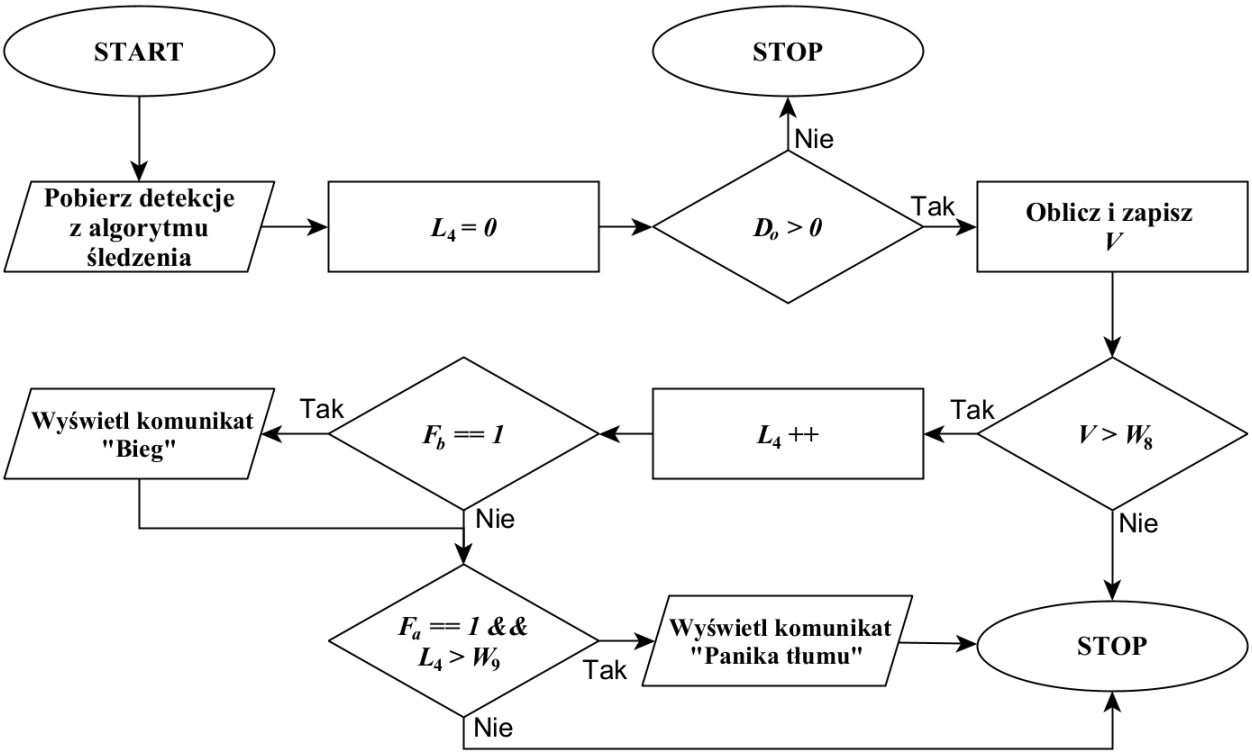
Podsumowując, opracowany algorytm detekcji przeludnienia stanowi rozwinięcie prac referencyjnych [97] oraz [99], umożliwiając korzystanie z detektorów OpenPose oraz YOLO v4 w celu identyfikacji zdarzenia, określanego jako przekroczenie progowej liczby osób wykrytych jednocześnie na obrazie. Dzięki funkcjonalnemu GUI, użytkownik ma możliwość określenia liczby osób, która stanowi wartość progową detekcji przeludnienia. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.8. W trakcie realizacji badań zaobserwowano, iż jedną z częstszych przyczyn występowania błędnych detekcji typu *FN*, są okluzje wynikające z chwilowego całkowitego zasłonięcia osoby lub brak detekcji osoby na obrazie. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji osób na obrazie. Wyniki przeprowadzonych badań wskazują, iż efektywność proponowanego algorytmu jest zadowalająca, a jego implementacja znacząco poszerzy zakres zdolności operacyjnych projektowanego systemu IVSS.

6.8 Bieg oraz analiza aktywności tłumu

W kontekście monitoringu publicznego, osoba biegnąca może wskazywać na wystąpienie potencjalnie niebezpiecznych sytuacji, takich jak ucieczka osoby zamieszanej w przestępstwo (zarówno ofiary jak i sprawcy), czy nagła ewakuacja spowodowana zagrożeniami typu pożar lub atak terrorystyczny. W systemach referencyjnych [104,113,115,116] można spotkać także pokrewną funkcjonalność, określaną jako detekcja paniki tłumu lub rozbieganie się tłumu. Funkcjonalność ta, nazywana odtąd analizą aktywności tłumu, dotyczy detekcji biegu kilku osób jednocześnie. Automatyczne wykrywanie i lokalizowanie biegnących osób może wpłynąć pozytywnie na wzrost skuteczności działań prewencyjnych oraz interwencyjnych poprzez szybką mobilizację odpowiednich służb porządkowych. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje sześć prac badawczych [104,110,113,115,116,120], w których autorzy implementują funkcję detekcji biegu i/lub analizę aktywności tłumu w wielozadaniowych systemach IVSS. Implementacja opisana w pracy [104] dotyczy zastosowania analizy histogramów, opracowanych na podstawie cech ruchu (przepływ optyczny, odejmowanie tła oraz kierunek ruchu). W pracy [110] użyto sieć IBSTM w celu analizy trajektorii ruchu opracowanej na podstawie rzeczywistych parametrów 3D obserwowanej sceny. Autorzy tejże pracy nie ujawniają metody transformacji układu współrzędnych 2D obrazu do rzeczywistych

współrzędnych sceny, lecz opisują użycie gotowych danych 3D dostarczonych z zewnętrznych źródeł (bazy danych uczących). W pracy [113] zastosowano detektor YOLO v1 i algorytm śledzenia TLD. W pracy [115] użyto analizę przepływu optycznego. W pracy [116] zastosowano ekstrakcję cech ruchu przy użyciu sieci I3D oraz klasyfikację zdarzeń przy użyciu sieci LSTM. Z kolei w pracy [120] użyto detektor OpenPose oraz model SVM do klasyfikacji zachowań. Wszystkie te techniki, pozwalają na osiągnięcie wysokiego stopnia skuteczności, jednak w większości nie odnoszą się do rzeczywistych parametrów 3D monitorowanej sceny. Korzystanie wyłącznie z danych dwuwymiarowych macierzy obrazu [104,113,115,116,120] lub uprzednio przygotowanych parametrów 3D konkretnej sceny [110] może ograniczać stopień skuteczności algorytmu w przypadkach monitorowania skomplikowanych scen przy dowolnej orientacji pola widzenia kamery.

Wychodząc naprzeciw tym ograniczeniom, autor rozprawy opracował algorytm detekcji biegu oraz analizy aktywności tłumu, którego schemat blokowy przedstawiono na Rys. 6.23.



Rys. 6.23 Schemat blokowy zaprojektowanego algorytmu detekcji biegu oraz analizy aktywności tłumu, gdzie: D_o – liczba śledzonych osób, F_a – flaga aktywności funkcji analizy tłumu, F_b – flaga aktywności funkcji detekcji biegu, L_4 – licznik biegnących osób, W_8 – wartość progowa maksymalnej szybkości przemieszczania się osoby, W_9 – wartość progowa maksymalnej dopuszczalnej liczby biegnących osób, V – szybkość przemieszczania się osoby

Oryginalność proponowanej metody polega na transformacji lokalizacji śledzonej osoby ze współrzędnych obrazu 2D do współrzędnych 3D monitorowanej sceny, umożliwiając określenie pokonanego dystansu oraz szybkości, z jaką osoba się porusza, niezależnie od orientacji pola widzenia kamery. Algorytm ten korzysta z danych historycznych zgromadzonych przy użyciu algorytmu śledzenia MOT oraz macierzy rzutowania 3D, uzyskanej w procesie kalibracji sceny (zob. rozdział 5.5).

W pierwszej kolejności pobierane są detekcje z algorytmu śledzenia. Następnie zostaje zerowana wartość licznika biegnących osób L_4. W przypadku niezerowej liczby śledzonych osób D_o obliczana zostaje średnia szybkość ruchu osoby V w metrach na sekundę. Proces ten realizowany jest w czterech krokach: wyznaczenie środkowego punktu dolnej krawędzi BB osoby na podstawie historii śledzenia ścieżki przejścia dla sekwencji maksymalnie 20 ostatnich ujęć, odczytanie współrzędnych 3D wyznaczonych punktów z macierzy rzutowania 3D, obliczenie sumy dystansów pomiędzy kolejnymi współrzędnymi 3D oraz obliczenie szybkości na podstawie sumy przebytego dystansu i czasu trwania pojedynczego ujęcia. Jeśli wartość V jest większa od wartości progowej W_8, wówczas zwiększany jest licznik L_4. W przypadku, gdy funkcja detekcji biegu jest aktywna (F_b = 1), wówczas na podglądzie obrazu monitoringu wyświetlony zostaje stosowny komunikat. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę, obejmującą sylwetkę biegnącej osoby. W dalszym kroku sprawdzane są dwa warunki: pierwszy, dotyczący włączenia funkcji analizy aktywności tłumu (F_a = 1) oraz drugi, dotyczący przekroczenia wartości progowej dopuszczalnej liczby biegnących osób W_9. Jeśli oba warunki zwracają prawdę, wówczas na podglądzie obrazu monitoringu wyświetlony zostaje stosowny komunikat, informujący użytkownika o potencjalnym wystąpieniu zdarzenia rozbiegania się/paniki tłumu. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę, wskazującą każdą biegnącą osobę.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu, użyto zestawu nagrań wideo o łącznej liczbie 17660 ujęć (zob. rozdział 4.4.1), gdzie 7608 ujęć dotyczy analizy aktywności tłumu, natomiast 10052 ujęć dotyczący przypadku detekcji biegu. Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.9.

Tab. 6.9 Zestawienie wyników badań empirycznych algorytmu detekcji biegu oraz analizy aktywności tłumu

Typ detekcji	***TP***	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
Bieg	2675	12	7012	353	96%	100%	88%	94%
Aktywność tłumu	1398	12	5849	349	95%	99%	80%	89%

Wskaźniki dokładności, precyzji, czułości oraz F_1 dla detekcji biegu przyjmują wartości odpowiednio: 96%, 100%, 88% oraz 94%, natomiast dla analizy aktywności tłumu odpowiednio: 95%, 99%, 80% oraz 89%. Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność opracowanego algorytmu.

Przykłady detekcji biegu przedstawiono na Rys. 6.24 natomiast analizy tłumu na Rys. 6.25.

Rys. 6.24 Testy empiryczne algorytmu detekcji biegu

Rys. 6.25 Testy empiryczne algorytmu analizy tłumu (panika/rozbieganie się tłumu)

Ujęcia znajdujące się w górnej części Rys. 6.24 przedstawiają prawidłowe detekcje napastnika uciekającego z miejsca zdarzenia. Ujęcia zamieszczone w dolnej części tego obrazu dotyczą tej samej sceny, gdzie biegnące osoby znajdują się w podobnej odległości od kamery, przy czym osoba w białej koszuli nie jest prawidłowo rozpoznawana jako biegnąca. Przykład ten pokazuje, że odległość od kamery oraz rozmiar śledzonej osoby mają znaczący wpływ na skuteczność detekcji.

Ujęcia znajdujące się w górnej części Rys. 6.25 przedstawiają prawidłowe detekcje wszystkich uciekających osób, o czym świadczy komunikat tekstowy oraz oznaczenie w postaci czerwonych ramek. Ujęcia zamieszczone w dolnej części tego obrazu również przedstawiają prawidłową detekcję zdarzenia, lecz w tych przypadkach nie wszystkie osoby uciekające zostały oznaczone na obrazie przy pomocy czerwonych ramek. Podobnie jak w przypadku detekcji biegu, jedną z częstszych obserwacji błędnych detekcji typu *FN* jest brak detekcji osoby lub chwilowe okluzje, uniemożliwiające przeprowadzenie wiarygodnej analizy szybkości ruchu.

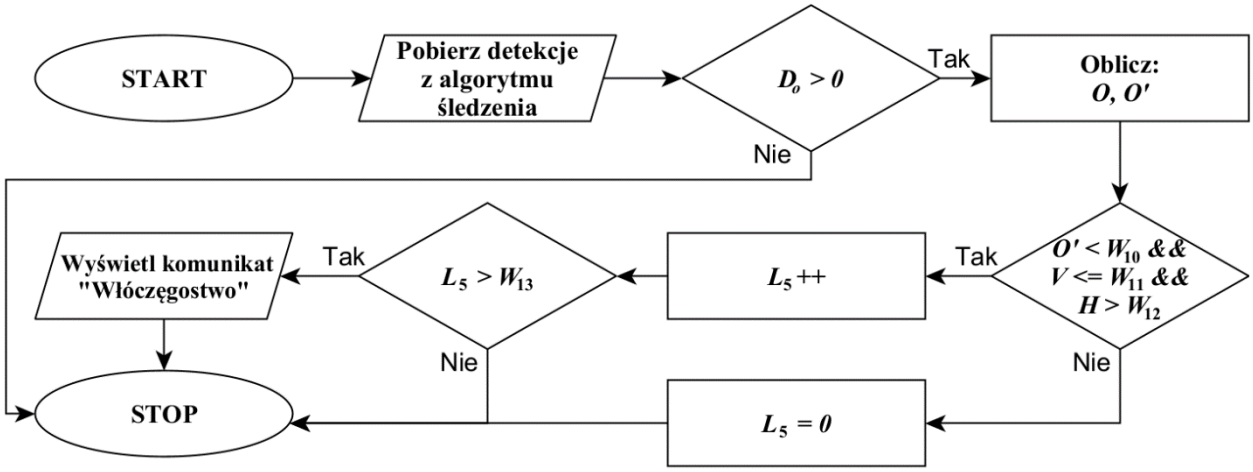
Podsumowując, zaprojektowany algorytm przeznaczony jest do identyfikacji biegu pojedynczych osób oraz detekcji paniki tłumu, rozumianej jako jednoczesne rozbieganie się grupy osób. W tym celu, algorytm korzysta z detektorów OpenPose i YOLO v4, danych historycznych zgromadzonych przy użyciu algorytmu śledzenia MOT oraz macierzy rzutowania 3D, uzyskanej w procesie kalibracji sceny (zob. rozdział 5.5). Oryginalność rozwiązania, polegająca na transformacji lokalizacji śledzonej osoby ze współrzędnych obrazu 2D do współrzędnych 3D monitorowanej sceny, umożliwia wiarygodne określenie pokonanego dystansu oraz szybkości, z jaką osoba się porusza niezależnie od orientacji pola widzenia kamery. Wyniki testów algorytmu wskazują na zadowalającą skuteczność w wykrywaniu opisanych zdarzeń alarmowych z wartościami wyznaczników F_1 przekraczającymi 89%. Rezultaty przeprowadzonych badań dowodzą, iż efektywność proponowanego algorytmu jest zadowalająca, co czyni go wartościowym komponentem w projektowanym systemie IVSS.

6.9 Włóczęgostwo

W kontekście niniejszej rozprawy detekcja włóczęgostwa dotyczy identyfikacji osób, pozostających w spoczynku lub poruszających się z relatywnie niską szybkością w tym samym obszarze monitoringu przez określony czas. Zachowanie to może wskazywać na osoby poszukujące okazji do popełnienia przestępstwa (kradzież, wandalizm, nielegalny handel), osoby z problemami psychicznymi, demencją, bezdomne lub dotknięte marginalizacją społeczną. Wykrywanie tego typu zachowań pozwala na podejmowanie odpowiednich działań zaradczych, takich jak zwiększenie obecności patroli służb porządkowych czy przeprowadzenie interwencji służb socjalnych. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje siedem prac badawczych [97,99,101,103,104,109,116], w których autorzy implementują funkcję

detekcji włóczęgostwa w wielozadaniowych systemach IVSS. Algorytmy opisane w pracach [97,101,103,104,109] charakteryzują się nieskomplikowaną logiką, opartą głównie na czasowo-przestrzennych cechach ruchu monitorowanych osób. W pracy [99] autorzy korzystają dodatkowo z uproszczonego algorytmu kalibracji sceny. Logika tego rozwiązania, oparta wyłącznie na założeniu, iż ludzie zazwyczaj poruszają się z podobną prędkością niezależnie od odległości od kamery (głębokości sceny), wprowadza istotne ograniczenia skuteczności detekcji. W pracy [116], w celu ekstrakcji cech ruchu oraz klasyfikacji zdarzeń alarmowych, autorzy używają modelu DL.

Autor rozprawy, inspirowany rozwiązaniem przedstawionym w pracy [99], opracował algorytm detekcji włóczęgostwa, działający w oparciu o cechy czasowo-przestrzenne, jak i cechy szybkości ruchu monitorowanych osób. Oryginalność implementacji dotyczy metod detekcji, klasyfikacji i śledzenia osób, a także metody szacowania szybkości ruchu, którą autor rozprawy realizuje przy użyciu macierzy projekcji 3D (zob. rozdział 5.5). Na Rys. 6.26 przedstawiono schemat blokowy zaprojektowanego algorytmu.



Rys. 6.26 Schemat blokowy zaprojektowanego algorytmu detekcji włóczęgostwa, gdzie: O – obszar przemieszczania się osoby na obrazie, O' – obszar przemieszczania się osoby w przestrzeni rzeczywistej, D_o – liczba wszystkich śledzonych osób, H – liczba elementów wektora historii śledzonej osoby, L_5 – licznik pozytywnych detekcji włóczęgostwa, W_{10} – wartość progowa obszaru przemieszczania się osoby, W_{11} – wartość progowa maksymalnej szybkości przemieszczania się osoby, W_{12} – wartość progowa liczby elementów wektora historii śledzonej osoby, W_{13} – wartość progowa pozytywnych detekcji włóczęgostwa, V – szybkość przemieszczania się osoby

W pierwszej kolejności pobierane są detekcje z algorytmu śledzenia MOT. W przypadku niezerowej liczby śledzonych osób D_o obliczany zostaje obszar przemieszczania się osoby O. Obszar O to pole powierzchni BB, utworzonego ze wszystkich dotychczas zapisanych lokalizacji śledzonej osoby (środkowych punktów dolnej krawędzi BB osoby).

Następnie, przy użyciu macierzy rzutowania, wartość O zostaje przeliczona na metry kwadratowe w celu oszacowania rzeczywistego pola powierzchni O' w układzie 3D obserwowanej sceny. System dokonuje pozytywnej identyfikacji wystąpienia włóczęgostwa w przypadku spełnienia trzech warunków: osoba porusza się po obszarze O' o polu powierzchni mniejszym niż wartość progowa W_{10}, szybkość ruchu osoby V jest mniejsza lub równa wartości

progowej W_{11} oraz liczba elementów wektora historii śledzonej osoby jest większa od wartości progowej W_{12}. Spełnienie tych warunków skutkuje zwiększeniem licznika l_5, stanowiącego bufor detekcji włóczęgostwa. Jeśli wartość l_5 jest większa niż wartość progowa W_{13}, wówczas zdarzenie włóczęgostwa zostaje zgłoszone w formie stosownego komunikatu tekstowego na podglądzie obrazu monitoringu. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę, obejmującą sylwetkę osoby przypisanej do detekcji włóczęgostwa.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 17201 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.10.

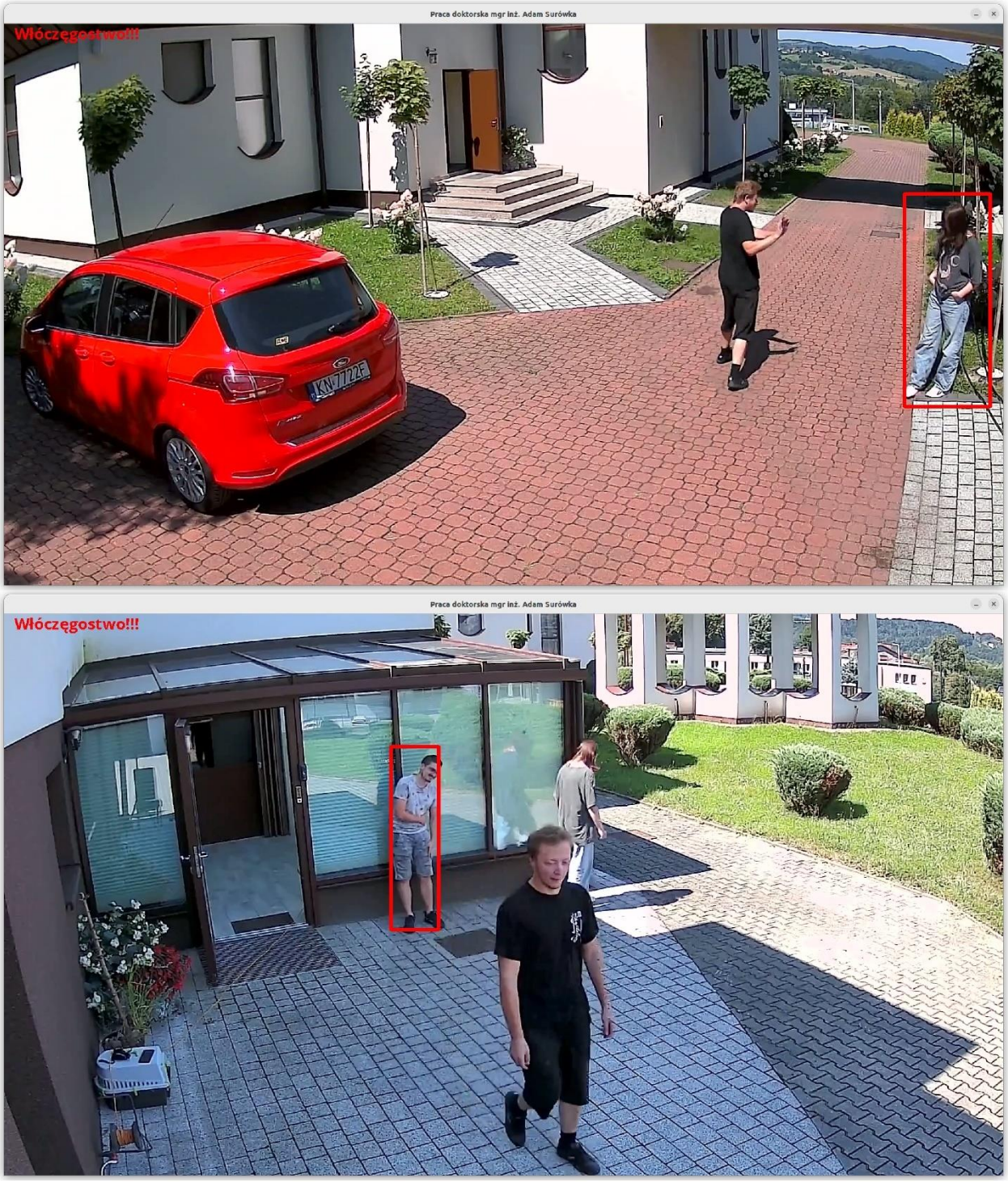
Tab. 6.10 Zestawienie wyników badań empirycznych algorytmu detekcji włóczęgostwa

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
9783	141	7053	224	98%	99%	98%	98%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia włóczęgostwa, o czym świadczą wskaźniki dokładności rzędu 98%, precyzji 99%, czułości 98% oraz F_1 98%.

Rezultaty działania algorytmu przedstawiono na Rys. 6.27. Oba ujęcia prezentują prawidłową detekcję zdarzenia, gdzie pojedyncza osoba, pozostająca w tym samym miejscu przez określony czas, zostaje skorelowana z możliwością wystąpienia włóczęgostwa. Informacja dotycząca detekcji wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo, osoba skorelowana ze zdarzeniem włóczęgostwa, oznaczana jest czerwoną ramką.

Podsumowując, opracowany algorytm detekcji włóczęgostwa wprowadza rozwiązania oryginalne, dotyczące użycia detektorów OpenPose oraz YOLO v4, a także algorytmu śledzenia oraz macierzy rzutowania 3D. Dzięki temu możliwe jest gromadzenie rzeczywistych parametrów trajektorii ścieżek przejść, takich jak obszar przemieszczania czy szybkość ruchu osoby. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości, zestawione w Tab. 6.10. W trakcie realizacji badań zaobserwowano, iż jedną z częstszych przyczyn występowania błędnych detekcji typu *FN*, są okluzje wynikające z chwilowego całkowitego zasłonięcia osoby lub brak detekcji osoby na obrazie. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji osób na obrazie. Powyższe rezultaty badań wskazują, że proponowany algorytm detekcji włóczęgostwa może stanowić cenne narzędzie w projektowanym systemie IVSS.



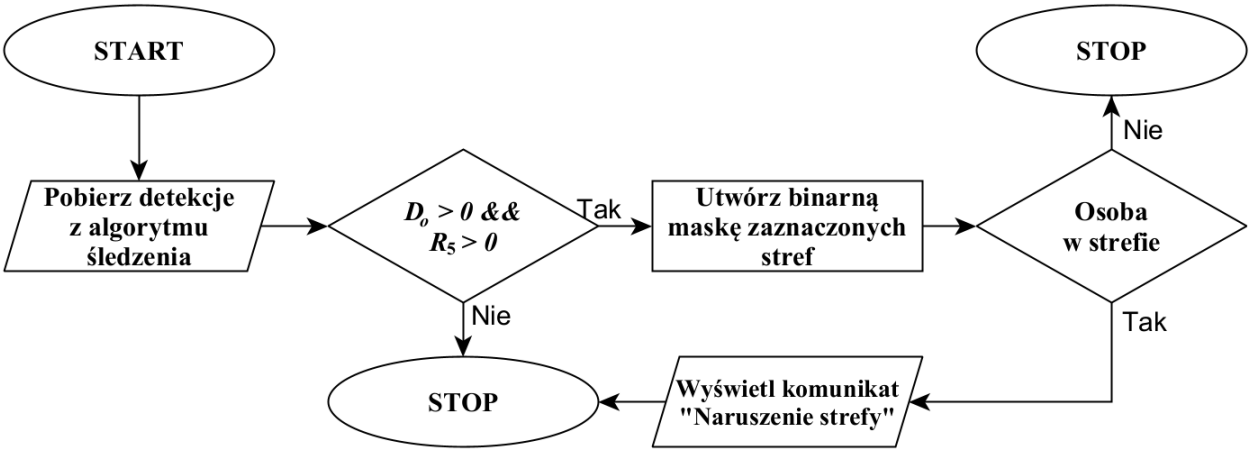
Rys. 6.27 Testy empiryczne algorytmu detekcji włóczęgostwa

6.10 Naruszenie wirtualnej strefy

W przeciwieństwie do detekcji naruszenia wirtualnej granicy (zob. rozdział 6.6), detekcja naruszenia strefy wirtualnej dotyczy precyzyjnie określonej przestrzeni, wydzielonej z monitorowanej sceny, w której każdorazowe wykrycie obecności osoby/osób skutkuje uruchomieniem procedury alarmowej. Skuteczna kontrola dostępu do stref zastrzeżonych jest kluczowa dla minimalizacji zagrożeń typu kradzieże, wandalizm, a nawet ataki terrorystyczne. Przez pojęcie strefy zastrzeżonej należy rozumieć wirtualne obszary zaznaczenia na obrazie,

obejmujące fizyczne strefy monitorowanej przestrzeni, do których dostęp jest ograniczony i kontrolowany ze względu na wymagania bezpieczeństwa. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje, iż funkcjonalność ta znajdowała się w centrum zainteresowań badaczy w latach 1997–2014. Wyniki badań opublikowane w tym okresie [97–101,104,106] pozwalają stwierdzić, iż jest to funkcja charakteryzująca się nieskomplikowaną logiką, której metody implementacji zostały wysoko rozwinięte. Ostatnia dekada wskazuje jednak na niemal całkowity brak realizacji badań w tym zakresie. Z tego powodu, autor rozprawy postanowił rozszerzyć zakres funkcjonalności projektowanego systemu IVSS o algorytm detekcji wejścia osób do stref zastrzeżonych.

Opracowany algorytm korzysta z danych historycznych, zgromadzonych przy użyciu algorytmu śledzenia MOT. Na Rys. 6.28 przedstawiono schemat blokowy zaprojektowanego algorytmu.



Rys. 6.28 Schemat blokowy zaprojektowanego algorytmu detekcji naruszenia wirtualnej strefy, gdzie: D_o – liczba wszystkich śledzonych osób, R_5 – licznik stref wirtualnych

W pierwszej kolejności pobierane są detekcje z algorytmu śledzenia. Jeśli liczba aktualnie śledzonych osób na obrazie D_o oraz licznik zaznaczonych stref wirtualnych R_5 przyjmują wartości niezerowe, wówczas tworzona jest binarna maska dla każdej zaznaczonej strefy. Maska ta tworzona jest z wektora punktów, tworzących daną strefę przy użyciu funkcji *cv::polygonMat()*. Następnie, przy użyciu funkcji *cv::pointPolygonTest()* sprawdzany jest warunek przynależności punktu ostatniej lokalizacji śledzonej osoby do obszaru strefy zaznaczonego na masce. Jeśli lokalizacja ta znajduje się w obszarze wirtualnej strefy, następuje zgłoszenie zdarzenia alarmowego w formie stosownego komunikatu, wyświetlanego w oknie podglądu monitorowanej sceny. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę, obejmującą sylwetkę osoby dokonującej naruszenia wirtualnej strefy.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 15339 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu

zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.11.

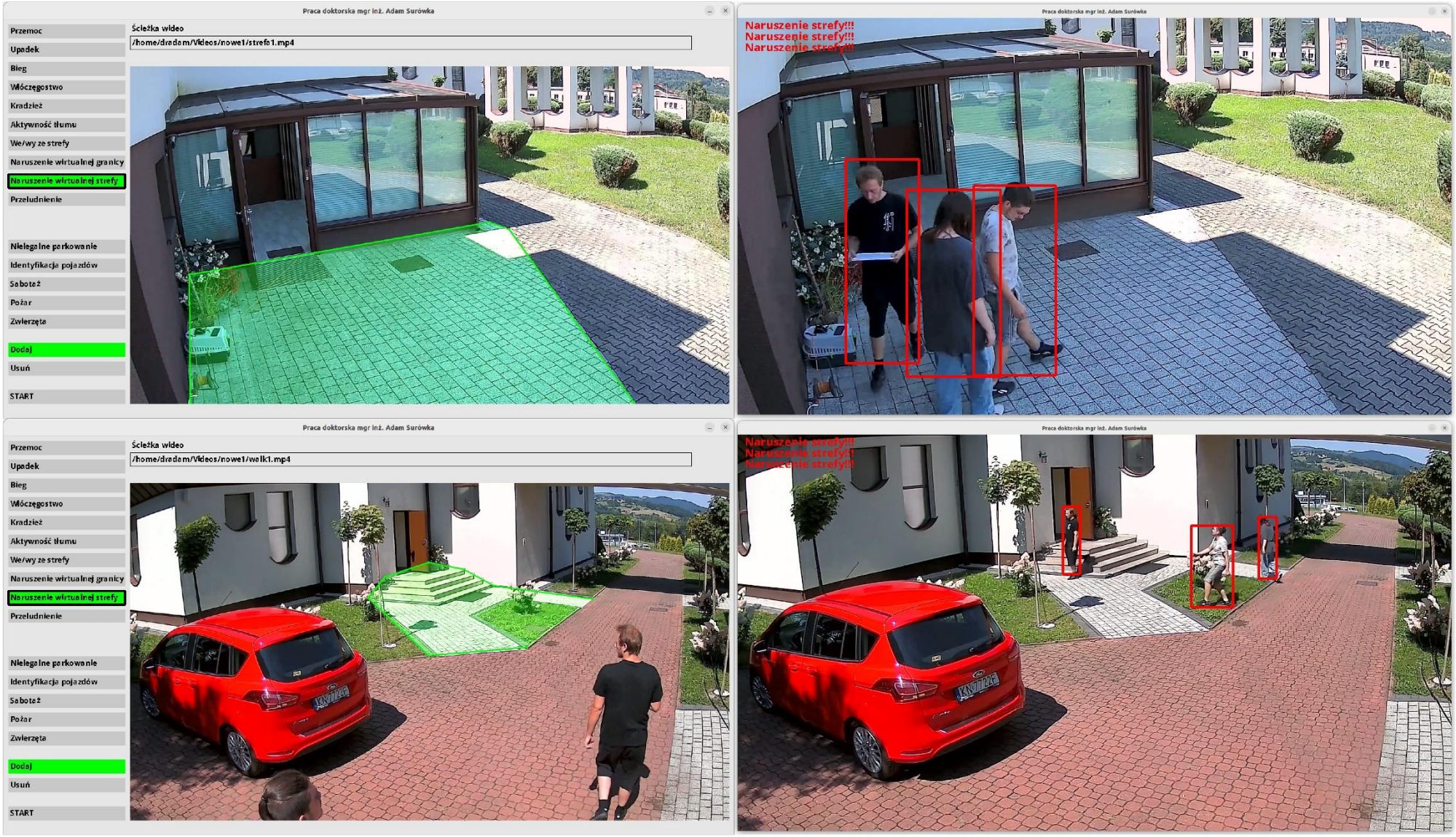
Tab. 6.11 Zestawienie wyników badań empirycznych algorytmu detekcji naruszenia wirtualnej strefy

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
12531	33	2493	282	98%	100%	98%	99%

Analiza otrzymanych wyników badań, wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu naruszenia wirtualnej strefy, o czym świadczą wskaźniki dokładności rzędu 98%, precyzji 100%, czułości 98% oraz F_1 99%.

Rezultaty działania algorytmu przedstawiono na Rys. 6.29. Po lewej stronie obrazu znajdują się widoki startowego okna GUI systemu, na których można zaobserwować sposób nanoszenia wirtualnych stref, stanowiących wyznacznik detekcji naruszenia wirtualnej strefy. Użytkownik ma możliwość oznaczenia dowolnej liczby stref zastrzeżonych. Każda strefa może składać się z dowolnej liczby punktów, tworzących jej obwód, umożliwiając precyzyjne oznaczenie obszaru o dowolnym kształcie i rozmiarze. Zaznaczone strefy na obrazie są jasnozielone. Ujęcia po prawej stronie obrazu przedstawiają odpowiednio rezultaty prawidłowej detekcji zdarzenia. Informacja dotycząca detekcji wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo osoba naruszająca obszar wirtualnej strefy oznaczana jest czerwoną ramką, co ułatwia jej lokalizację.

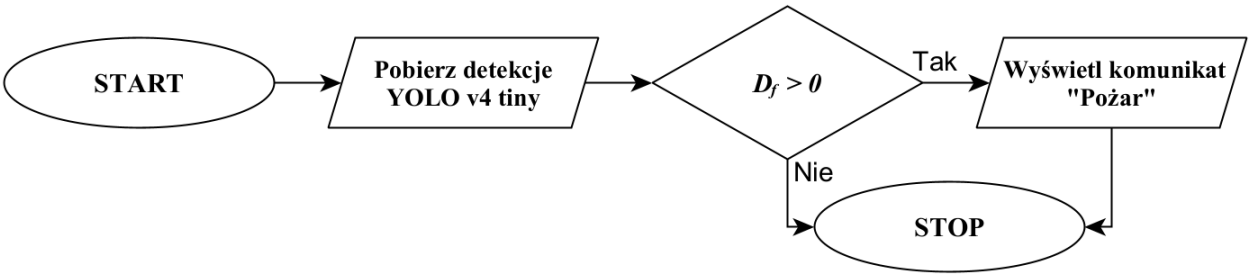
Podsumowując, opracowany algorytm detekcji naruszenia wirtualnej strefy korzysta z detektorów OpenPose, YOLO v4 oraz algorytmu śledzenia MOT, w celu detekcji osób znajdujących się w obszarze wirtualnej strefy, oznaczonej na obrazie przy pomocy GUI podczas startu systemu. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.11. W trakcie realizacji badań zaobserwowano, iż jedną z częstszych przyczyn występowania błędnych detekcji typu *FN*, są okluzje, wynikające z chwilowego całkowitego zasłonięcia osoby lub brak detekcji osoby na obrazie. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji osób na obrazie. Wyniki przeprowadzonych badań wskazują, iż efektywność proponowanego algorytmu jest zadowalająca, a jego implementacja znacząco poszerzy zakres zdolności operacyjnych projektowanego systemu IVSS.



Rys. 6.29 Testy empiryczne algorytmu detekcji naruszenia wirtualnej strefy

6.11 Pożar

Tradycyjne metody detekcji pożaru, takie jak czujniki dymu czy czujniki ciepła, mają swoje ograniczenia, wynikające głównie z opóźnień w wykrywaniu oraz ograniczonego zasięgu działania. Zastosowanie zaawansowanych technik widzenia komputerowego z obszaru ML, stanowi obiecującą alternatywę w zapewnianiu bezpieczeństwa p.poż. w miejscach użytku publicznego jak i w obiektach prywatnych. Świadczy o tym rosnące zainteresowanie badaczy tą tematyką, zaobserwowane na przestrzeni ostatnich czterech lat, którzy w celu detekcji omawianego zdarzenia używają modeli DL typu: I3D [116,121], C3D [117,122], CNN [119,123], MD-RAN [124] oraz AnomalyNet [125]. W przeciwieństwie do podobnych tego typu rozwiązań, opracowany przez autora rozprawy algorytm, realizuje zadanie detekcji pożaru na podstawie wyników detektora YOLO v4 tiny (zob. rozdział 5.3.2), który został wytrenowany m.in. w celu detekcji obiektów klasy „*ogień*” na obrazie. Na Rys. 6.30 przedstawiono schemat blokowy zaprojektowanego algorytmu.



Rys. 6.30 Schemat blokowy zaprojektowanego algorytmu detekcji pożaru, gdzie: D_f – liczba detekcji obiektów klasy „*ogień*”

Działanie algorytmu sprowadza się do ciągłej analizy ilościowej wyników detekcji, pochodzących z detektora YOLO v4 tiny. Jeśli liczba wykrytych obiektów klasy „*ogień*” D_f jest niezerowa, wówczas na obrazie podglądu monitorowanej sceny wyświetlany jest komunikat, informujący o potencjalnym wystąpieniu zdarzenia alarmowego. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę w miejscu detekcji obiektu klasy „*ogień*” na obrazie.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 16933 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.12.

Tab. 6.12 Zestawienie wyników badań empirycznych algorytmu detekcji pożaru

TP	*FP*	*TN*	*FN*	**Dokładność**	**Precyzja**	**Czułość**	F_1
14120	67	1584	1162	93%	100%	92%	96%

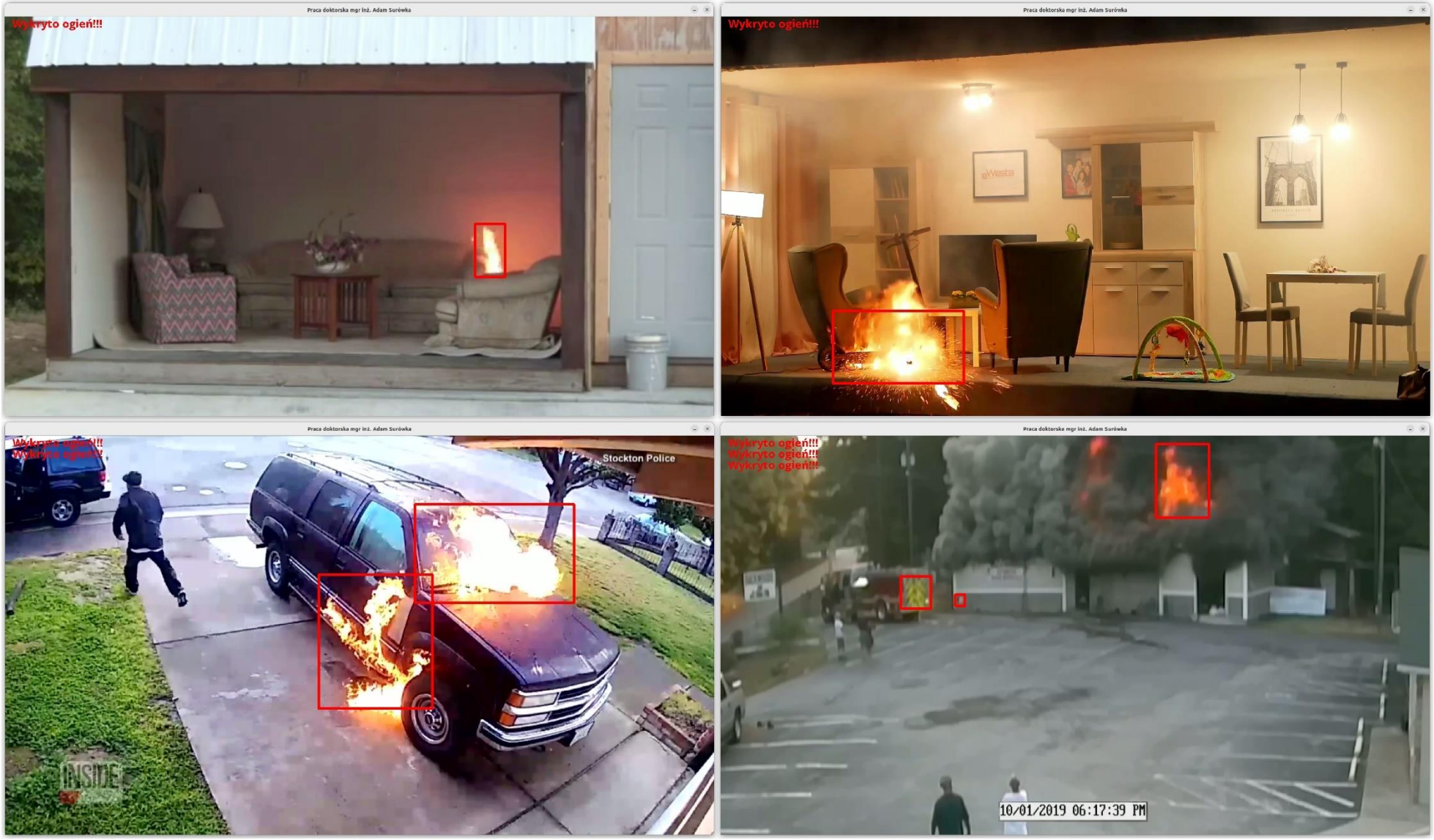
Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia pożaru, o czym świadczą wskaźniki dokładności rzędu 93%, precyzji 100%, czułości 92% oraz F_1 96%.

Na Rys. 6.31 przedstawiono przykładowe ujęcia poklatkowe, zapisane podczas testów empirycznych algorytmu. Informacja dotycząca detekcji wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo obszary występowania płomieni na obrazie oznaczane są czerwoną ramką, co ułatwia ich lokalizację. Ujęcie znajdujące się w prawej dolnej części Rys. 6.31, stanowi przykład detekcji typu *FP*. Oprócz prawidłowej detekcji płomieni na dachu budynku, algorytm błędnie wykrywa płomienie na pojeździe strażackim oraz na elewacji budynku. W pozostałych przypadkach detekcja płomieni na obrazie realizowana jest w sposób prawidłowy.

Podsumowując, opracowany algorytm detekcji pożaru oparty został w całości na wynikach specjalnie wytrenowanego modelu YOLO v4 tiny, umożliwiającego detekcję obiektów klasy „*ogień*”, celem identyfikacji wystąpienia pożaru na analizowanym obrazie. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.12. W trakcie realizacji badań zaobserwowano, iż przyczyną występowania błędnych detekcji może być niski stopień wytrenowania modelu YOLO v4 tiny dla klasy „*ogień*”, co zostało omówione w rozdziale 5.3.2. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji płomieni na obrazie. Rezultaty przeprowadzonych badań dowodzą, iż efektywność proponowanego algorytmu jest zadowalająca, co czyni go wartościowym komponentem w projektowanym systemie IVSS.

6.12 Upadek

Powszechnie wiadomo, iż przewrócenie się osoby może wynikać z przypadkowego potknięcia, jak i poważniejszych następstw chorobowych typu zawał serca czy udaru mózgu. Niezależnie od przyczyny, konsekwencje upadku mogą dotyczyć zarówno drobnych urazów mechanicznych ciała, poprzez utratę przytomności, aż do wypadków śmiertelnych. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje dziewięć prac badawczych [97,98,100,103,104,106,115,116,120], w których autorzy implementują funkcję detekcji upadku w wielozadaniowych systemach IVSS. Pomimo iż detekcja upadku stanowi tak ważny aspekt wielozadaniowych systemów IVSS, na przestrzeni ostatniej dekady można zaobserwować tendencję spadkową zainteresowania badaczy tym tematem. W pracy [97] detekcję upadku zrealizowano na podstawie analizy krawędzi poruszających się obiektów pierwszoplanowych, przy czym autorzy nie zdradzają szczegółów tej implementacji. W pracy [98] pozytywny wynik detekcji upadku jest możliwy przy spełnieniu jednego z trzech warunków.

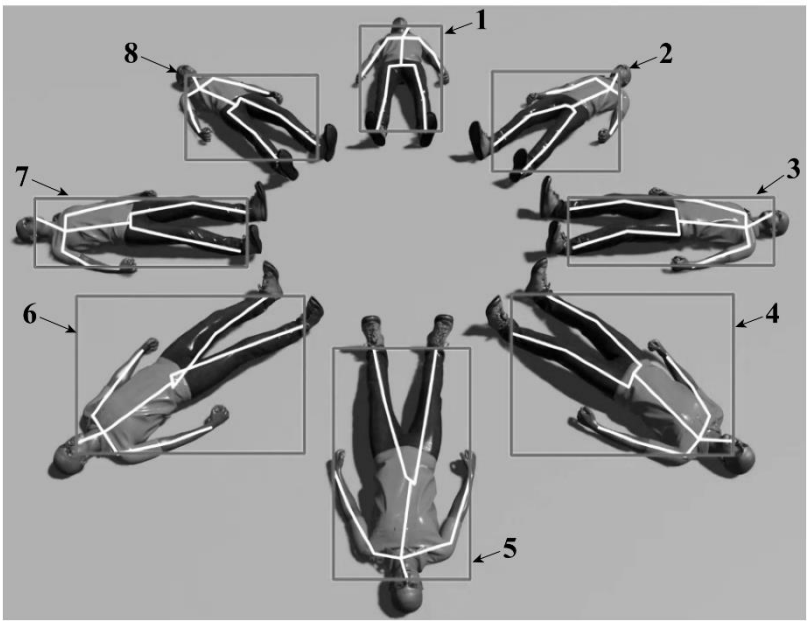


Rys. 6.31 Testy empiryczne algorytmu detekcji pożaru

Są to: szerokość obiektu pierwszoplanowego jest większa od jego wysokości, zerowe lub niewielkie przemieszczenie centroidy obiektu oraz lokalizacja centroidy w zastrzeżonej strefie. Przemieszczenie i lokalizacja osoby mogą stanowić jedynie przesłankę, co do wystąpienia zdarzenia upadku, natomiast użycie tych danych jako jednoznacznych wyznaczników może skutkować wzrostem liczby detekcji typu *FP*. W pracy [100] zastosowano techniki AI bez szczegółowego opisu implementacji, co uniemożliwia przeprowadzenie rzetelnego odniesienia. W pracy [103] pozytywny wynik detekcji upadku jest możliwy, gdy wysokość osoby zostanie oszacowana na wartość poniżej 1.5 m. Takie podejście może generować liczne detekcje typu *FP*, np. w przypadku analizy osób siedzących, poruszających się na wózku inwalidzkim czy analizy osób niskich oraz dzieci. W pracy [104] detekcja upadku realizowana jest poprzez analizę zmiany profilu kształtu ruchu z rozkładu wertykalnego na horyzontalny oraz zmiany szybkości ruchu (przyspieszenie) w momencie upadania. Takie podejście sprawdzi się głównie w przypadku potknięcia się osoby idącej spokojnym krokiem, ponieważ zmiany profilu kształtu ruchu oraz szybkości ruchu będą wówczas wyraźnie zauważalne. Problem detekcji może pojawić się, gdy przewróci się osoba biegnąca, której szybkość ruchu była wysoka przez cały czas trwania analizy. Innym problematycznym przykładem może być upadek, wynikający z zasłabnięcia lub ataku serca, gdzie proces upadania może być zbyt powolny, by został pozytywnie zidentyfikowany przez system. Zaproponowana logika detekcji upadku opisana w pracy [106], dotyczy tylko jednej sytuacji, w której dwie osoby zderzają się ze sobą, po czym jedna z nich pozostaje leżąca przez określony czas. Logika ta mocno ogranicza uniwersalność zastosowania ze względu na przeznaczenie do detekcji tylko jednej, złożonej sytuacji, wymagającej udziału dwóch osób. W pracy [115] autorzy korzystają z analizy cech ruchu, takich jak szybkość i kierunek przemieszczenia osoby, oszacowanych na podstawie wyników przepływu optycznego. Detekcja upadku jest specyficznym typem zachowania, którego identyfikacja wyłącznie na podstawie cech ruchu osoby (szybkości i kierunku przemieszczania) może skutkować licznymi błędnymi detekcjami. W pracy [116] detekcja upadku została zrealizowana poprzez zastosowanie złożonego modelu DL (architektury sieci I3D oraz LSTM), co pozwoliło na osiągnięcie zadowalających rezultatów. W pracy [120] autorzy korzystają z sekwencyjnej analizy cech morfologicznych i dynamicznych modeli szkieletowych, wygenerowanych przy użyciu zmodyfikowanego detektora OpenPose, natomiast logikę decyzyjną oparto na wynikach klasyfikatora SVM. Pomimo zastosowania precyzyjnej metody analizy modelu szkieletowego, skuteczność detekcji upadku została oszacowana na zaledwie 51.1%. Powodem tego może być zbyt ogólne podejście do ekstrakcji wektora cech ruchu, który w tej samej formie używany jest również w celu realizacji pozostałych funkcjonalności oferowanych przez system.

Wychodząc naprzeciw opisanym problemom badawczym, autor rozprawy opracował algorytmu detekcji upadku, motywowany pracą [120]. Proponowane rozwiązanie przeznaczone

jest do detekcji upadku osób, niezależnie od ich orientacji względem pola widzenia kamery, kierunku oraz szybkości przemieszczania się, interakcji z innymi ludźmi i obiektami, a nawet niezależnie od szybkości upadania. W tym celu użyto dane historyczne, zawierające ścieżki przejść osób, archiwizowane przy pomocy algorytmu śledzenia MOT oraz modele szkieletowe wszystkich śledzonych osób poruszających się na obrazie. Logika decyzyjna została natomiast zrealizowana w oparciu o proste funkcje warunkowe. W celu lepszego zrozumienia złożoności zadania detekcji upadku na podstawie danych wizyjnych 2D, na Rys. 6.32 przedstawiono osiem przykładów orientacji sylwetek ciała osoby przewróconej.

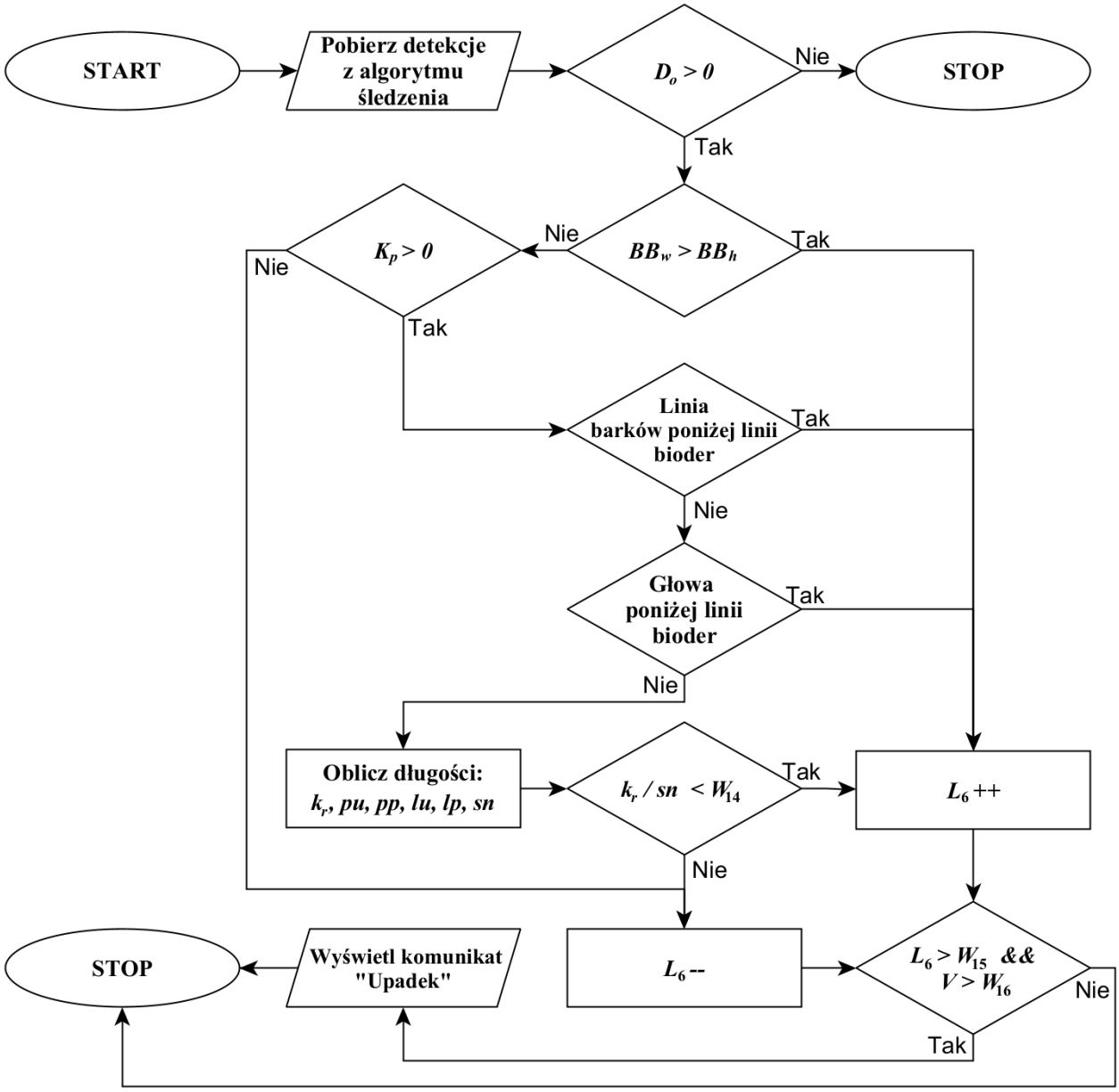


Rys. 6.32 Porównanie modeli szkieletowych sylwetek osób leżących w różnych orientacjach względem kamery

Przykłady te zostały opracowane w programie Blender, przy użyciu tego samego modelu wirtualnej sylwetki człowieka. Każda sylwetka została oznaczona przy pomocy graficznych markerów w postaci BB oraz modeli szkieletowych, opracowanych przy użyciu detektora OpenPose. Zaprezentowane przykłady umożliwiają zrozumienie, w jaki sposób projektowany system IVSS interpretuje sylwetki leżących osób w zależności od ich orientacji względem kamery. Ponadto Rys. 6.32 będzie stanowić odwołanie do specyficznych przypadków, opisywanych w dalszej części niniejszego rozdziału.

Na Rys. 6.33 przedstawiono schemat blokowy zaprojektowanego algorytmu. W pierwszej kolejności pobierane są detekcje z algorytmu śledzenia MOT. Jeśli liczba aktualnie śledzonych osób D_o przyjmuje wartość niezerową, wówczas sprawdzane zostają cztery kolejne instrukcje warunkowe, przy czym pozytywny wynik przynajmniej jednej z nich pozytywnie identyfikuje zdarzenie upadku.

Pierwsza instrukcja warunkowa dotyczy analizy profilu kształtu BB śledzonej osoby. Polega to na sprawdzeniu, czy szerokość BB (BB_w) jest większa od jego wysokości (BB_h). Pozytywny wynik tego warunku skutkuje zwiększeniem licznika detekcji upadku osoby L_6.



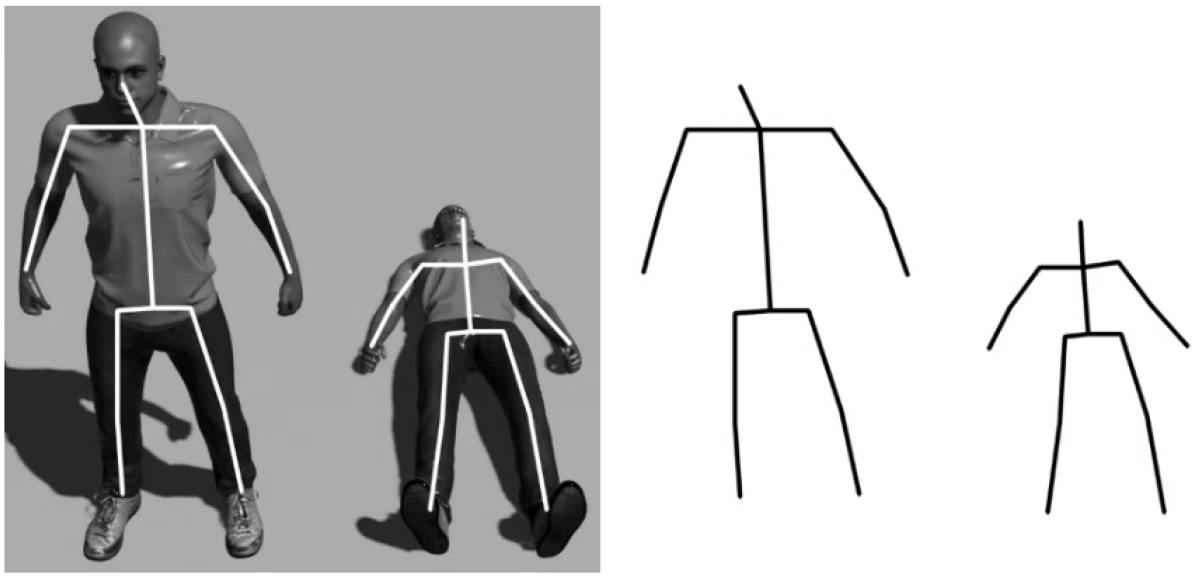
Rys. 6.33 Schemat blokowy zaprojektowanego algorytmu detekcji upadku, gdzie: BB_h – wysokość BB, BB_w – szerokość BB, D_o – liczba wszystkich śledzonych osób, K_p – liczba punktorów szkieletu śledzonej osoby, k_r – długość kręgosłupa, *pu* – długość prawego uda, *pp* – długość prawego podudzia, *lu* – długość lewego uda, *lp* – długość lewego podudzia, *sn* – średnia długość nóg, L_6 – licznik pozytywnych detekcji upadku, W_{14} – wartość progowa stosunku długości kręgosłupa do długości nóg, W_{15} – wartość progowa liczby pozytywnych detekcji upadku osoby, W_{16} – wartość progowa szybkości przemieszczania się osoby, *V* – szybkość przemieszczania się osoby

Analiza profilu kształtu BB jest jedną z najprostszych metod detekcji upadku, charakteryzującą się wysokim stopniem skuteczności w przypadkach, gdy upadająca osoba zorientowana jest bokiem do kamery i upada w przód lub w tył. Omawiane przypadki ilustruje Rys. 6.32 dla przykładów 2–4 oraz 6–8.

Im bardziej upadająca osoba zorientowana jest przodem, lub tyłem do kamery (zob. Rys. 6.32, przykłady 1 oraz 5), tym mniej skuteczna okazuje się analiza profilu kształtu BB, ponieważ relacja BB_w do BB_h pozostaje niemal identyczna jak w przypadku osoby stojącej. W celu detekcji zdarzenia, w którym osoba zorientowana przodem do kamery upada w przód lub zorientowana tyłem do kamery upada w tył, autor rozprawy opracował dodatkowe instrukcje warunkowe, oparte

na analizie modelu szkieletowego śledzonej osoby. Dotyczą one analizy położenia linii barków oraz położenia głowy względem linii bioder. Pozytywny wynik warunku, w którym linia barków lub głowa znajdują się poniżej linii bioder, skutkuje zwiększeniem licznika detekcji upadku osoby L_6. Warunki te określane są na podstawie lokalizacji odpowiednich punktorów szkieletu w odniesieniu do układu współrzędnych macierzy obrazu.

Ostatnia instrukcja warunkowa została opracowana w celu detekcji jednych z najtrudniejszych przypadków, w których osoba zorientowana przodem do kamery upada w tył lub zorientowana tyłem do kamery upada w przód. Na Rys. 6.34 przedstawiono wizualizację omawianego przypadku.



Rys. 6.34 Porównanie modelu szkieletowego osoby stojącej i leżącej zorientowanej przodem do kamery

W lewej części obrazu znajdują się dwa identyczne modele sylwetki człowieka (stojący i leżący) w odległości 0.5 m od siebie, natomiast po prawej stronie znajdują się wyłącznie modele szkieletowe owych sylwetek, wygenerowane przy użyciu detektora OpenPose. Jak można zaobserwować, oba modele szkieletowe charakteryzują się wysokim stopniem podobieństwa. Z uwagi na orientacje sylwetki na obrazie, próba oceny upadku przy pomocy analizy kształtu profilu BB nie wskaże różnicy w porównaniu z osobą stojącą. Ocena relacji linii barków oraz głowy względem linii bioder również nie przyniesie pożądanego rezultatu, ponieważ zarówno głowa, jak i linia barków pozostają zlokalizowane powyżej linii bioder na obrazie. W celu rozwiązania tego problemu, autor zaproponował instrukcję decyzyjną, opartą na analizie stosunku długości odcinka kręgosłupa do średniej długości nóg. Skorzystano tutaj z klasycznego zjawiska zniekształceń perspektywy widzenia kamery, w której to na podglądzie obrazu, obiekty znajdujące się bliżej kamery, wydają się być większe od tych znajdujących się w oddali. Obserwacja przykładu z Rys. 6.34 potwierdza tę tezę. Jedyną zauważalną różnicą w przypadku szkieletu osoby leżącej, jest nieproporcjonalnie większa długość nóg w porównaniu do długości kręgosłupa. Wynika to z następstw wspomnianych zniekształceń perspektywy. Relacja ta zatem pozwala na wiarygodne

rozróżnienie modelu szkieletowego osoby leżącej od stojącej lub chodzącej. Proces detekcji upadku dla omawianego przypadku rozpoczyna się od obliczenia długości następujących elementów szkieletu: kręgosłupa k_r, prawego uda *pu*, prawego podudzia *pp*, lewego uda *lu* oraz lewego podudzia *lp*. Możliwość wystąpienia różnego rodzaju okluzji może powodować brak detekcji niektórych punktorów szkieletu. Dlatego, w zależności od liczby dostępnych punktorów dolnej części ciała, a zatem niezerowych odcinków *pu*, *pp*, *lu* oraz *lp*, algorytm oblicza uśrednioną długość nóg *sn*. Następnie sprawdzany jest warunek, dotyczący relacji pomiędzy stosunkiem długości odcinka kręgosłupa do średniej długości nóg, a wartością progową W_{14}. W przeciwieństwie do określania stałej wartości mierzonej wysokości (w pracy [103]), wyznaczenie wartości progowej w formie stosunku k_r/sn, umożliwia osiągnięcie wysokiego stopnia uniwersalności zastosowania algorytmu niezależnie od wzrostu analizowanych osób.

Zastosowanie licznika L_6 oraz wartości progowej W_{15} umożliwia stworzenie bufora pozytywnych detekcji upadku osoby w celu wyeliminowania pojedynczych, błędnych detekcji typu *FP*. Jeśli $L_6 > W_{15}$ oraz szybkość przemieszczania się osoby *V* nie przekracza wartości progowej W_{16}, wówczas na obrazie podglądu monitorowanej sceny zostaje wyświetlony stosowny komunikat tekstowy, informujący o detekcji upadku. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę obejmującą sylwetkę leżącej osoby.

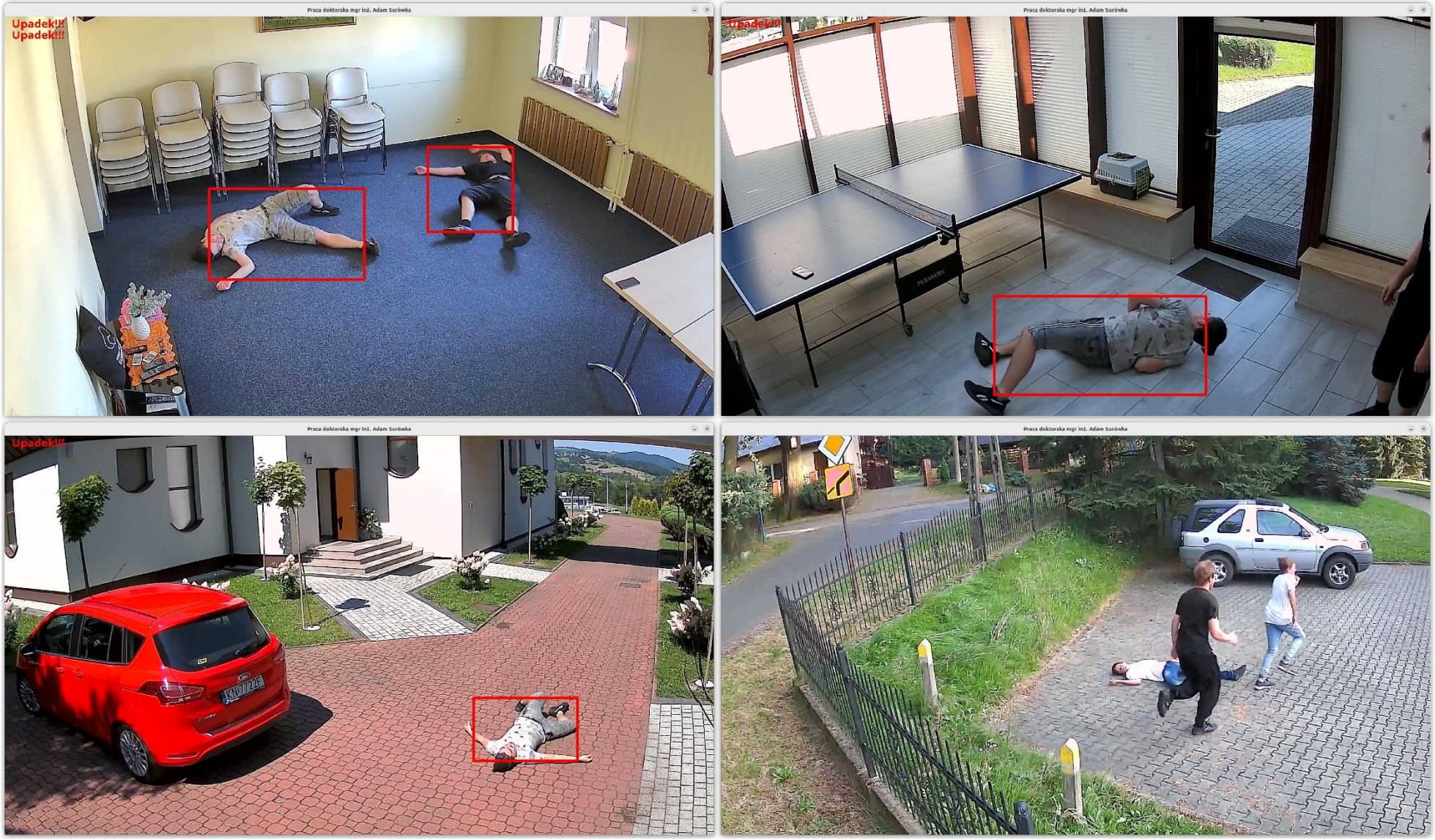
Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 16577 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów, opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.13.

Tab. 6.13 Zestawienie wyników badań empirycznych algorytmu detekcji upadku

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
11025	432	2960	2160	84%	96%	84%	89%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia upadku, o czym świadczą wskaźniki dokładności rzędu 84%, precyzji 96%, czułości 84% oraz F_1 89%.

Na Rys. 6.35 przedstawiono przykładowe ujęcia poklatkowe, zapisane podczas testów empirycznych algorytmu. Informacja dotycząca detekcji upadku, wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo osoba leżąca oznaczana jest czerwoną ramką, ułatwiającą użytkownikowi jej lokalizację. Ujęcie znajdujące się w prawej dolnej części Rys. 6.35 stanowi przykład detekcji typu *FN*, które wynika z chwilowej okluzji osoby leżącej. W pozostałych przypadkach detekcja upadku na obrazie realizowana jest w sposób prawidłowy.



Rys. 6.35 Testy empiryczne algorytmu detekcji upadku

Podsumowując, opracowany algorytm detekcji upadku korzysta z detektorów OpenPose i YOLO v4, algorytmu śledzenia MOT oraz dedykowanej logiki operacyjnej w celu identyfikacji wystąpienia zdarzenia upadku. Opracowane rozwiązanie stanowi rozwinięcie pracy [120], umożliwiające detekcję upadku osób, niezależnie od ich orientacji względem pola widzenia kamery, kierunku oraz szybkości przemieszczania się, interakcji z innymi ludźmi i obiektami, a nawet niezależnie od szybkości upadania. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości, zestawione w Tab. 6.13. W trakcie realizacji badań zaobserwowano, iż jedną z częstszych przyczyn występowania detekcji typu *FN*, są okluzje, wynikające z chwilowego całkowitego zasłonięcia osoby lub brak detekcji osoby na obrazie. Zaobserwowano również, iż detekcje typu *FP* mogą być powodowane błędną interpretacją sylwetki osoby siedzącej jako leżącej. Należy zwrócić uwagę, iż próba precyzyjnego określenia punktu granicznego pomiędzy sylwetką siedzącą a leżącą, stanowi poważne wyzwanie w związku z możliwością dowolnej orientacji osoby na scenie. Omawiany problem występuje przede wszystkim podczas obserwacji osoby zorientowanej bokiem do kamery z tułowiem odchylonym w tył. W celu redukcji detekcji typu *FN* oraz *FP* dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji osób na obrazie oraz implementacji logiki operacyjnej, umożliwiającej automatyczne rozróżnienie sylwetki siedzącej od leżącej w pozycjach przejściowych. Powyższe rezultaty badań wskazują, że proponowany algorytm detekcji upadku może stanowić cenne narzędzie w projektowanym systemie IVSS.

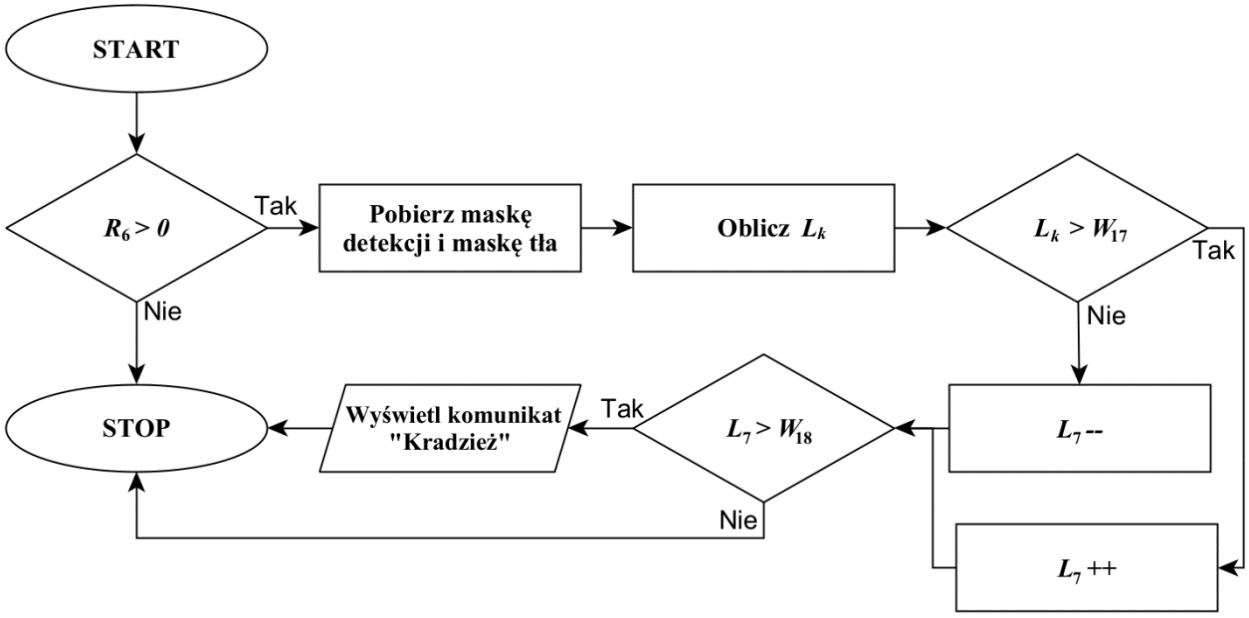
6.13 Kradzież

W projektowanym systemie IVSS detekcja kradzieży dotyczy identyfikacji braku przedmiotów statycznych w lokalizacjach, które zostały im uprzednio przypisane na obrazie przy pomocy GUI podczas startu programu. Przez określenie przedmioty statyczne, należy rozumieć dowolne obiekty, znajdujące się na monitorowanej scenie, których lokalizacja nie może ulec zmianie podczas trwania procedury monitorowania. Powszechnie wiadomo, iż akty kradzieży stanowią poważne wyzwanie dla służb odpowiedzialnych za utrzymanie bezpieczeństwa publicznego. Straty finansowe wynikające z tego typu incydentów, mogą być znaczne, a ich wpływ na poczucie bezpieczeństwa społeczności niebagatelny. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje, iż funkcja detekcji kradzieży została zaimplementowana w jedenastu pracach badawczych [103,109,113,116,117,119,121–125] z zakresu wielozadaniowych systemów IVSS. W pracy [103] detekcja kradzieży sprowadza się do identyfikacji zdarzenia, w którym osoba wchodząca do monitorowanej przestrzeni bez bagażu, przejmuje bagaż innej osoby i przemieszcza się z nim na scenie. W tym celu zastosowano metodę

odejmowania tła oraz analityczne metody oceny czasowo-przestrzennych cech ruchu. W pracy [109] zdarzenie kradzieży dotyczy sytuacji, w której wykryto obecność osoby przy kasie sklepowej, gdy w pobliżu nie ma kasjera. Podobnie jak w pracy [103], zastosowano tu metodę odejmowania tła. System jednak został rozbudowany o implementację klasyfikatora SVM, w celu analizy czasowo-przestrzennych cech ruchu. W pracy [113] detekcja kradzieży dotyczy tej samej sytuacji, co w [103]. W tym przypadku użyto jednak detektora YOLO v1 w połączeniu z wielowarstwowym modelem reprezentacji wiedzy EDCAR. W pozostałych pracach detekcja zdarzenia kradzieży została zrealizowana przy użyciu modeli DL typu: I3D [116,121], C3D [117,122], CNN [119,123], MD-RAN [124] oraz AnomalyNet [125]. Wyniki badań opublikowane w omówionych pracach, pozwalają stwierdzić, iż detekcja kradzieży to funkcja, której metody implementacji zostały wysoko rozwinięte. Niestety znaczna część z proponowanych metod uniemożliwia oznaczenie konkretnych obiektów zainteresowania, stanowiących podstawę obserwacji. Ponadto, autorzy zazwyczaj pomijają kwestie związane z adaptacją algorytmów do możliwości wystąpienia chwilowych okluzji monitorowanych obiektów, co może skutkować wzrostem liczby detekcji typu *FP*.

Wychodząc naprzeciw tym problemom, autor rozprawy opracował algorytm detekcji kradzieży obiektów statycznych z zastosowaniem metody odejmowania tła. Algorytm bazuje na wynikach algorytmu detekcji sabotażu kamery (zob. rozdział 6.3), co pozytywnie wpływa na wydajność obliczeniową systemu. Ponadto, użytkownik ma możliwość oznaczenia dowolnej liczby obiektów zainteresowania o dowolnym kształcie, co sprawia, że algorytm jest wysoce uniwersalny. Dodatkowo, algorytm został opracowany w sposób umożliwiający interpretację chwilowych okluzji obiektów zainteresowania, związanych z obecnością osób oraz pojazdów znajdujących się w monitorowanej przestrzeni, co prowadzi do redukcji liczby potencjalnych detekcji typu *FP*.

Na Rys. 6.36 przedstawiono schemat blokowy zaprojektowanego algorytmu. Do prawidłowego działania algorytmu wymagana jest niezerowa liczba zaznaczonych obiektów zainteresowania R_6. Dla każdej strefy zaznaczenia, obliczana jest liczba pikseli niezerowych L_k, reprezentująca stopień zmian pierwszoplanowych w tej strefie. Proces ten realizowany jest przy użyciu zsumowanej maski detekcji oraz tła, uzyskanej w drugim etapie pracy algorytmu detekcji sabotażu kamery (zob. Rys. 6.11). Maska ta została już zmodyfikowana, w celu usunięcia detekcji obiektów klasy „*osoba*” oraz wszystkich obiektów klasy „*pojazd*”. Dzięki temu, chwilowe okluzje obiektów zainteresowania, wywołane obecnością znanych klas obiektów, nie zostają błędnie sklasyfikowane jako zdarzenie kradzieży. Wartość progowa liczby pikseli niezerowych W_{17} określana jest iloczynem $p_a \cdot t_r$, gdzie p_a to liczba wszystkich pikseli składających się na zaznaczoną strefę, natomiast t_r to maksymalna wartość procentowa zmian w zaznaczonej strefie. Jeśli wartość L_k przekracza wartość progową W_{17} wówczas zostaje zwiększony licznik



Rys. 6.36 Schemat blokowy zaprojektowanego algorytmu detekcji kradzieży, gdzie: L_7 – licznik pozytywnych detekcji kradzieży, L_k – liczba pikseli niezerowych maski tła obiektu zainteresowania, R_6 – licznik stref wirtualnych zaznaczonych obiektów zainteresowania, W_{17} – wartość progowa liczby pikseli niezerowych, W_{18} – wartość progowa liczby pozytywnych detekcji kradzieży

pozytywnych detekcji kradzieży L_7. Zastosowanie licznika L_7 oraz wartości progowej W_{18} umożliwia stworzenie bufora pozytywnych detekcji kradzieży obiektu w celu wyeliminowania pojedynczych, błędnych detekcji typu *FP*. Jeśli $L_7 > W_{18}$, wówczas na obrazie podglądu monitorowanej sceny zostaje wyświetlony stosowny komunikat tekstowy, informujący o detekcji kradzieży. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę obejmującą strefę zaznaczenia skradzionego obiektu.

Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 18003 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.14.

Tab. 6.14 Zestawienie wyników badań empirycznych algorytmu detekcji kradzieży

TP	*FP*	*TN*	*FN*	**Dokładność**	**Precyzja**	**Czułość**	F_1
7021	0	9493	1489	92%	100%	83%	90%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia kradzieży obiektu statycznego, o czym świadczą wskaźniki dokładności rzędu 92%, precyzji 100%, czułości 83% oraz F_1 90%.

Rezultaty działania algorytmu przedstawiono na Rys. 6.37. Po lewej stronie obrazu znajdują się widoki startowego okna GUI systemu, na których można zaobserwować sposób nanoszenia wirtualnych stref, obejmujących nadzorowane przedmioty.

Rys. 6.37 Testy empiryczne algorytmu detekcji kradzieży

Każda strefa może składać się z dowolnej liczby punktów tworzących jej obwód, umożliwiając precyzyjne oznaczenie przedmiotu o dowolnym kształcie i rozmiarze. Zaznaczone strefy na obrazie są w jasnozielonym kolorze. Ujęcia po prawej stronie obrazu przedstawiają odpowiednio rezultaty prawidłowej detekcji zdarzenia. Informacja dotycząca detekcji, wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo obszar kradzieży na obrazie zostaje oznaczany czerwoną ramką, co ułatwia jego lokalizację.

Podsumowując, opracowany algorytm detekcji kradzieży korzysta z logiki algorytmu detekcji sabotażu kamery (zob. rozdział 6.3) oraz detektorów OpenPose i YOLO v4, w celu identyfikacji wystąpienia kradzieży obiektu statycznego w monitorowanym obszarze. Użytkownik ma możliwość oznaczenia dowolnej liczby obiektów zainteresowania o dowolnym kształcie, co sprawia, że algorytm jest wysoce uniwersalny. Dodatkowo, algorytm został opracowany w sposób umożliwiający interpretację chwilowych okluzji obiektów zainteresowania, związanych z obecnością osób oraz pojazdów, znajdujących się w monitorowanej przestrzeni, co prowadzi do redukcji liczby potencjalnych detekcji typu *FP*. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.14. W trakcie realizacji badań zaobserwowano, iż jedną z częstszych przyczyn występowania błędnych detekcji typu *FN*, są okluzje, wynikające z chwilowego całkowitego lub częściowego zasłonięcia nadzorowanego przedmiotu przez obiekty nieznanych klas. Wyniki przeprowadzonych badań wskazują, iż efektywność proponowanego algorytmu jest zadowalająca, a jego implementacja znacząco poszerzy zakres zdolności operacyjnych projektowanego systemu IVSS.

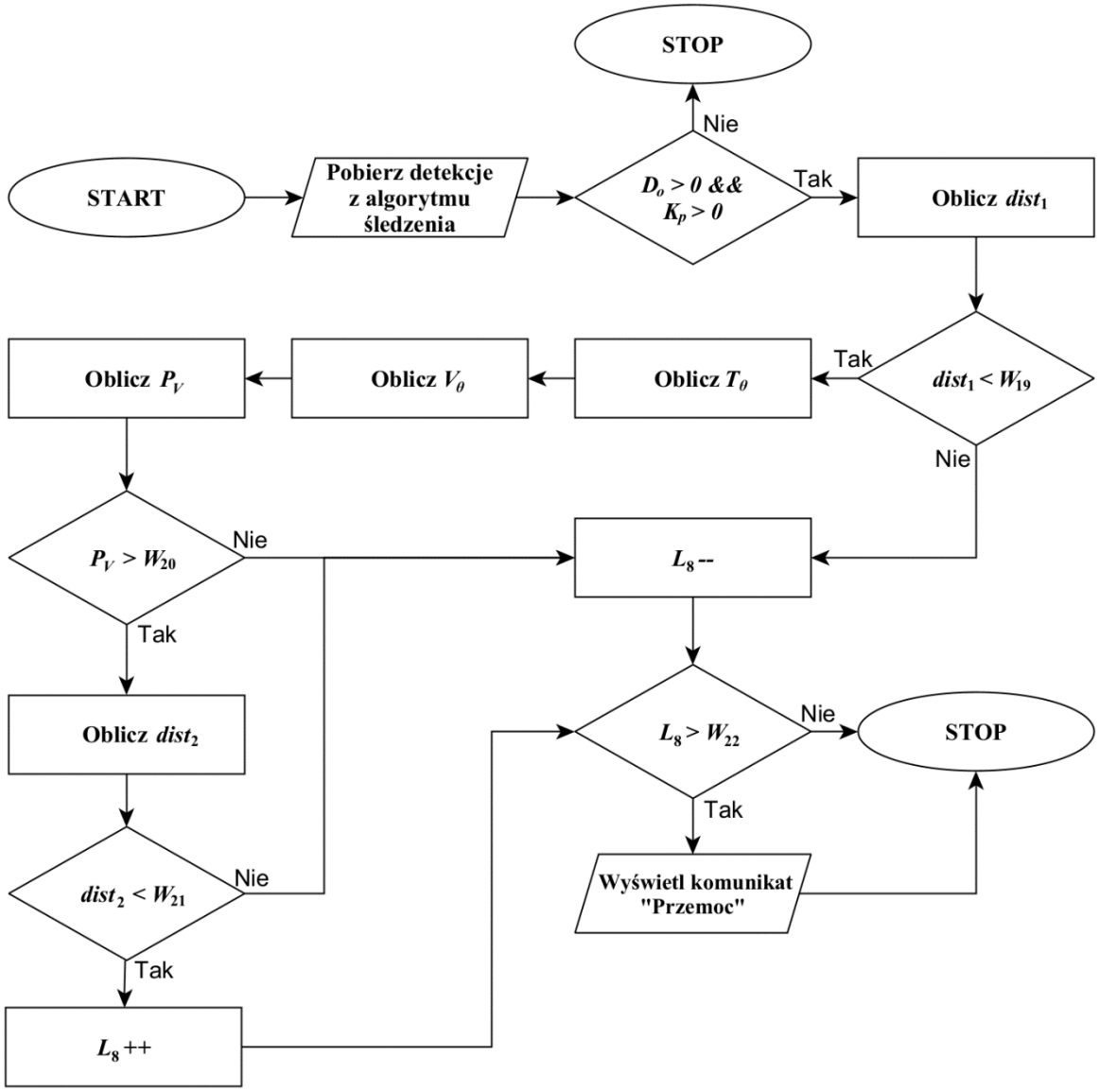
6.14 Przemoc

Detekcja przemocy w projektowanym systemie IVSS dotyczy identyfikacji interakcji co najmniej dwóch osób znajdujących się w bliskim zasięgu, z których co najmniej jedna wykazuje dynamiczne i agresywne ruchy ciała. Detekcja ta ma celu zgłoszenie wystąpienia zachowań typu uderzenia, kopnięcia, szarpanie, rzucanie oraz innych form bezpośredniego kontaktu fizycznego, które mogą prowadzić do obrażeń ciała, upadku, a nawet śmierci osób znajdujących się na obrazie. Funkcja detekcji przemocy na podstawie danych wizyjnych umożliwia natychmiastowe podjęcie działań prewencyjnych i interwencyjnych, w celu uniknięcia dalszej eskalacji tego typu zachowań. Przeprowadzona analiza systemów referencyjnych (zob. rozdział 3.4) wskazuje szesnaście prac badawczych [98,103,110,111,113,115–125] z zakresu wielozadaniowych systemów IVSS, w których dokonano implementacji metod, związanych z szeroko pojętą detekcją przemocy. W pracy [98] pozytywny rezultat detekcji przemocy jest możliwy, gdy zostaną spełnione trzy warunki: centroidy kształtów obiektów pierwszoplanowych poruszają się relatywnie blisko siebie, ruch ten

można sklasyfikować jako naprzemienne zbliżanie i oddalanie się oraz występują chaotyczne zmiany kształtów śledzonych obiektów. W pracy [103] autorzy rozbudowali logikę, zaczerpniętą z [98], dodając czwarty warunek, sprawdzający wartość progową minimalnej szybkości zbliżania i oddalania się kształtów obiektów pierwszoplanowych. W pracy [110] zastosowano rozwiązanie hybrydowe, polegające na połączeniu zestawu reguł semantycznych oraz modelu DL o nazwie IBSTM. W pracy [111] zastosowano detektor YOLO v2 w połączeniu z sieciami CNN oraz LSTM. W pracy [113] zastosowano detektor YOLO v1 w połączeniu z wielowarstwowym modelem reprezentacji wiedzy EDCAR. W pracy [115] zastosowano analizę przepływu optycznego, w celu wyznaczenia dwóch cech ruchu obiektów: szybkości oraz kierunku. W pracy [118] zastosowano ekstrakcję cech przy użyciu sieci CNN, a następnie klasyfikację zdarzeń przy użyciu sieci LSTM. W pracy [120] zastosowano zmodyfikowany model OpenPose (zastosowanie sieci ResNet-51 zamiast VGG-19 w etapie ekstrakcji cech), następnie analizę cech morfologicznych i dynamicznych punktorów szkieletów i klasyfikację zdarzeń przy użyciu SVM. W pozostałych pracach detekcja przemocy została zrealizowana przy użyciu modeli DL typu: I3D [116,121], C3D [117,122], CNN [119,123], MD-RAN [124] oraz AnomalyNet [125].

Logika opracowanego przez autora rozprawy algorytmu detekcji przemocy, opiera się na zmodyfikowanej metodzie, opisanej w [120]. Detekcja zdarzenia walki realizowana jest na podstawie ekstrakcji cech odległościowych, kątowych, dynamicznych i kształtowych modelu szkieletowego człowieka, wygenerowanego przy użyciu detektora OpenPose. Oryginalność proponowanego rozwiązania dotyczy analizy szybkości zmian 20 kątów modelu szkieletowego oraz użycia macierzy rzutowania 3D, w celu określenia rzeczywistego dystansu pomiędzy osobami na monitorowanej scenie. Na Rys. 6.38 przedstawiono schemat blokowy zaprojektowanego algorytmu.

W pierwszej kolejności pobierane są dane historyczne z algorytmu śledzenia MOT. Jeśli liczba śledzonych osób D_o jest niezerowa oraz liczba punktorów szkieletu śledzonej osoby K_p jest niezerowa, wówczas obliczana zostaje odległość analizowanej osoby od kamery $dist_1$. W tym celu korzysta się z macierzy rzutowania 3D. Wartość $dist_1$ pozwala określić, czy osoba znajduje się w obszarze, umożliwiającym detektorowi OpenPose wygenerowanie szkieletów z odpowiednim stopniem pewności (zob. Rys. 5.9). Szkielety osób znajdujących się zbyt daleko od kamery, nie będą dalej przetwarzane pod kątem detekcji przemocy, ponieważ ich niski stopień szczegółowości może przyczynić się do wzrostu liczby detekcji typu *FP*. Jeśli zatem wartość $dist_1$ jest mniejsza niż wartość progowa W_{19}, realizowana jest kolejna część algorytmu, związana z analizą modelu szkieletowego, składająca się z trzech etapów. Są to: obliczanie wektora miar kątów pomiędzy ustalonymi punktorami szkieletu T_θ, obliczanie wektora szybkości zmian tych kątów w czasie V_θ oraz obliczenie średniej szybkości zmian kątów dla pogrupowanych części szkieletu P_V.



Rys. 6.38 Schemat blokowy zaprojektowanego algorytmu detekcji przemocy, gdzie: D_o – liczba wszystkich śledzonych osób, $dist_1$ – odległość analizowanej osoby od kamery, $dist_2$ – odległość pomiędzy osobami, K_p – liczba punktorów szkieletu śledzonej osoby, L_8 – licznik pozytywnych detekcji przemocy, P_V – średnia szybkość zmian zestawów kątów, W_{19} – wartość progowa odległości osoby od kamery, W_{20} – wartość progowa szybkości zmiany kątów, W_{21} – wartość progowa dystansu pomiędzy osobami, W_{22} – wartość progowa liczby pozytywnych detekcji przemocy, T_θ – wektor miar kątów, V_θ – wektor szybkości zmian kątów

W pierwszym etapie, w celu zbudowania pełnego obrazu kinematyki ruchu badanej osoby, obliczane są kąty pomiędzy ustalonymi częściami szkieletu. Kąty te obliczane są zawsze dla trzech punktorów, np. dla lewego łokcia kąt obliczany jest na podstawie punktorów 5 (lewy bark), 6 (lewy łokieć) oraz 7 (lewy nadgarstek). Procedura ta jest powtarzana dla dwudziestu zestawów punktorów, których wykaz zamieszczono w Tab. 6.15.

Tab. 6.15 Wykaz zestawów punktorów szkieletu, analizowanych w celu detekcji przemocy

Identyfikator kąta	Nazwa kąta	Zestaw punktorów
θ_0	Prawy łokieć	[4, 3, 2]
θ_1	Prawe ramię–szyja	[3, 2, 1]
θ_2	Prawy nadgarstek–środek bioder	[4, 1, 8]

Identyfikator kąta	Nazwa kąta	Zestaw punktorów
θ_3	Prawy łokieć–środek bioder	[3, 1, 8]
θ_4	Lewy łokieć	[5, 6, 7]
θ_5	Lewe ramię–szyja	[1, 5, 6]
θ_6	Lewy nadgarstek–środek bioder	[7, 1, 8]
θ_7	Lewy łokieć–środek bioder	[6, 1, 8]
θ_8	Prawe kolano	[11, 10, 9]
θ_9	Prawe udo–środek bioder	[10, 9, 8]
θ_{10}	Prawe kolano–szyja	[10, 8, 1]
θ_{11}	Prawa kostka–szyja	[11, 8, 1]
θ_{12}	Lewe kolano	[12, 13, 14]
θ_{13}	Lewe udo–środek bioder	[13, 12, 8]
θ_{14}	Lewe kolano–szyja	[13, 8, 1]
θ_{15}	Lewa kostka–szyja	[14, 8, 1]
θ_{16}	Prawy nadgarstek–lewy nadgarstek	[4, 1, 7]
θ_{17}	Prawy łokieć–lewy łokieć	[3, 1, 6]
θ_{18}	Prawa kostka–lewa kostka	[11, 8, 14]
θ_{19}	Prawe kolano–lewe kolano	[10, 8, 13]

Na Rys. 6.39 przedstawiono graficzną reprezentację wybranych kątów opisanych w Tab. 6.15.

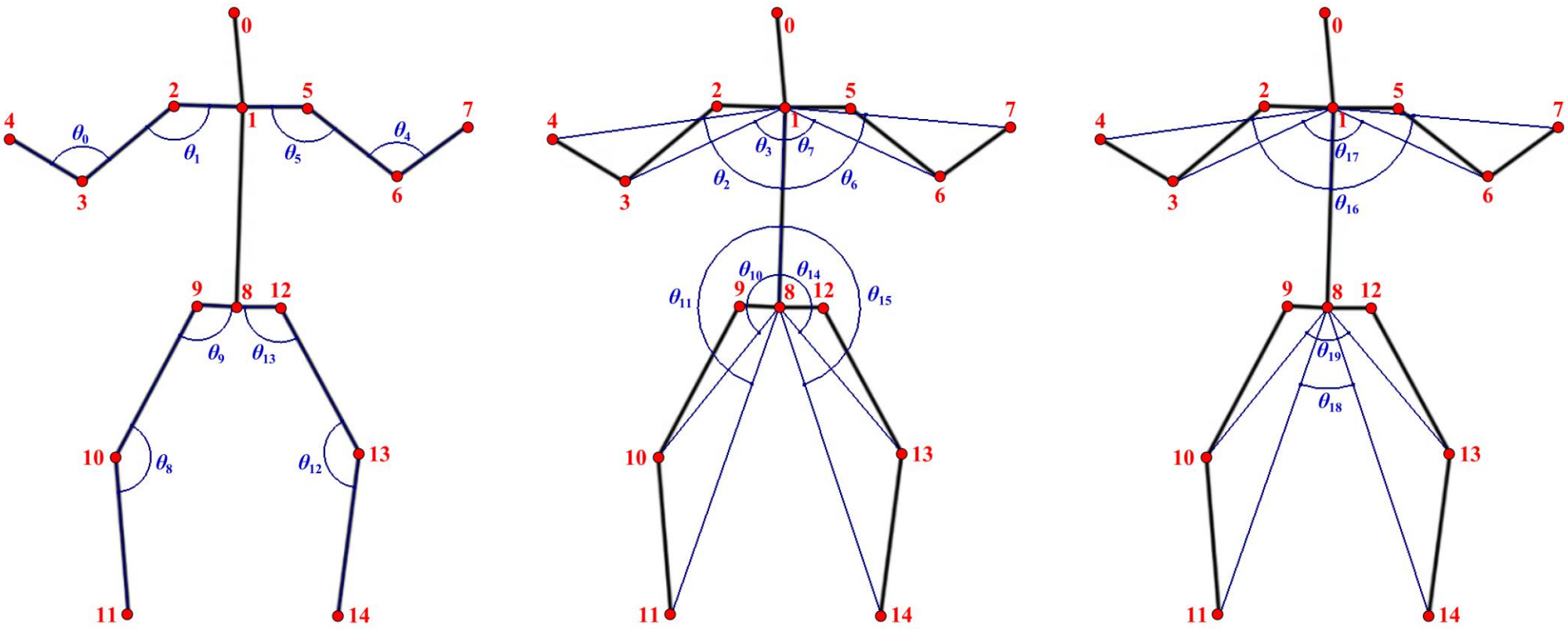
W drugim etapie, w celu zbudowania pełnego obrazu dynamiki ruchu badanych osób, obliczane są uśrednione szybkości zmian wybranych kątów. Zadanie to realizowane jest na podstawie zapisanych danych historycznych, dotyczących miar kątów w poprzednich ujęciach oraz znajomości częstotliwości rejestrowania wideo (*FPS*).

W trzecim etapie obliczane są średnie szybkości zmiany kątów dla czterech części ciała. Wybrane części ciała dotyczą czterech zestawów kątów, których wykaz zamieszczono w Tab. 6.16.

Tab. 6.16 Wykaz zestawów kątów części ciała opracowanych w celu detekcji przemocy

Nazwa części ciała	Zestaw kątów
Prawa ręka	[θ_0, θ_1, θ_2, θ_3, θ_{16}, θ_{17}]
Lewa ręka	[θ_8, θ_9, θ_{10}, θ_{11}, θ_{18}, θ_{19}]
Prawa noga	[θ_4, θ_5, θ_6, θ_7, θ_{16}, θ_{17}]
Lewa noga	[θ_{12}, θ_{13}, θ_{14}, θ_{15}, θ_{18}, θ_{19}]

Wybór tych części ciała umożliwia wiarygodną identyfikację dynamicznych ruchów osoby uczestniczącej w bójce.



Rys. 6.39 Graficzna reprezentacja wybranych kątów modelu szkieletowego w algorytmie detekcji przemocy

Jeśli średnia szybkość zmian kątów dowolnej z analizowanych części ciała przekracza dopuszczalną wartość graniczną W_{20}, stanowi to pierwszą przesłankę o możliwości wystąpienia zdarzenia przemocy. Przesłanka ta informuje o gwałtownych, dynamicznych ruchach danej części ciała. W celu potwierdzenia wystąpienia zdarzenia, obliczana jest dodatkowo odległość pomiędzy analizowaną osobą i wszystkimi pozostałymi śledzonymi osobami ($dist_2$). W tym celu, ponownie korzysta się z macierzy rzutowania 3D. Jeśli osoba charakteryzująca się dynamicznymi ruchami części ciała, znajduje się w zasięgu przynajmniej jednej innej śledzonej osoby ($dist_2 < W_{21}$), wówczas następuje zwiększenie licznika pozytywnych detekcji przemocy L_8. Zastosowanie licznika L_8 oraz wartości progowej W_{22}, umożliwia stworzenie bufora pozytywnych detekcji przemocy w celu wyeliminowania pojedynczych, błędnych detekcji typu *FP*. Jeśli $L_8 > W_{22}$, wówczas na obrazie podglądu monitorowanej sceny zostaje wyświetlony stosowny komunikat tekstowy, informujący o wystąpieniu zdarzenia alarmowego. Dodatkowo, w celu usprawnienia lokalizacji wystąpienia zdarzenia, algorytm rysuje czerwoną ramkę obejmującą osobę, charakteryzującą się gwałtownymi ruchami części ciała. Ponadto, wszystkie osoby potencjalnie zagrożone, a więc znajdujące się w zasięgu uderzenia bądź kopnięcia agresora, również oznaczane są na obrazie za pomocą czerwonej ramki.

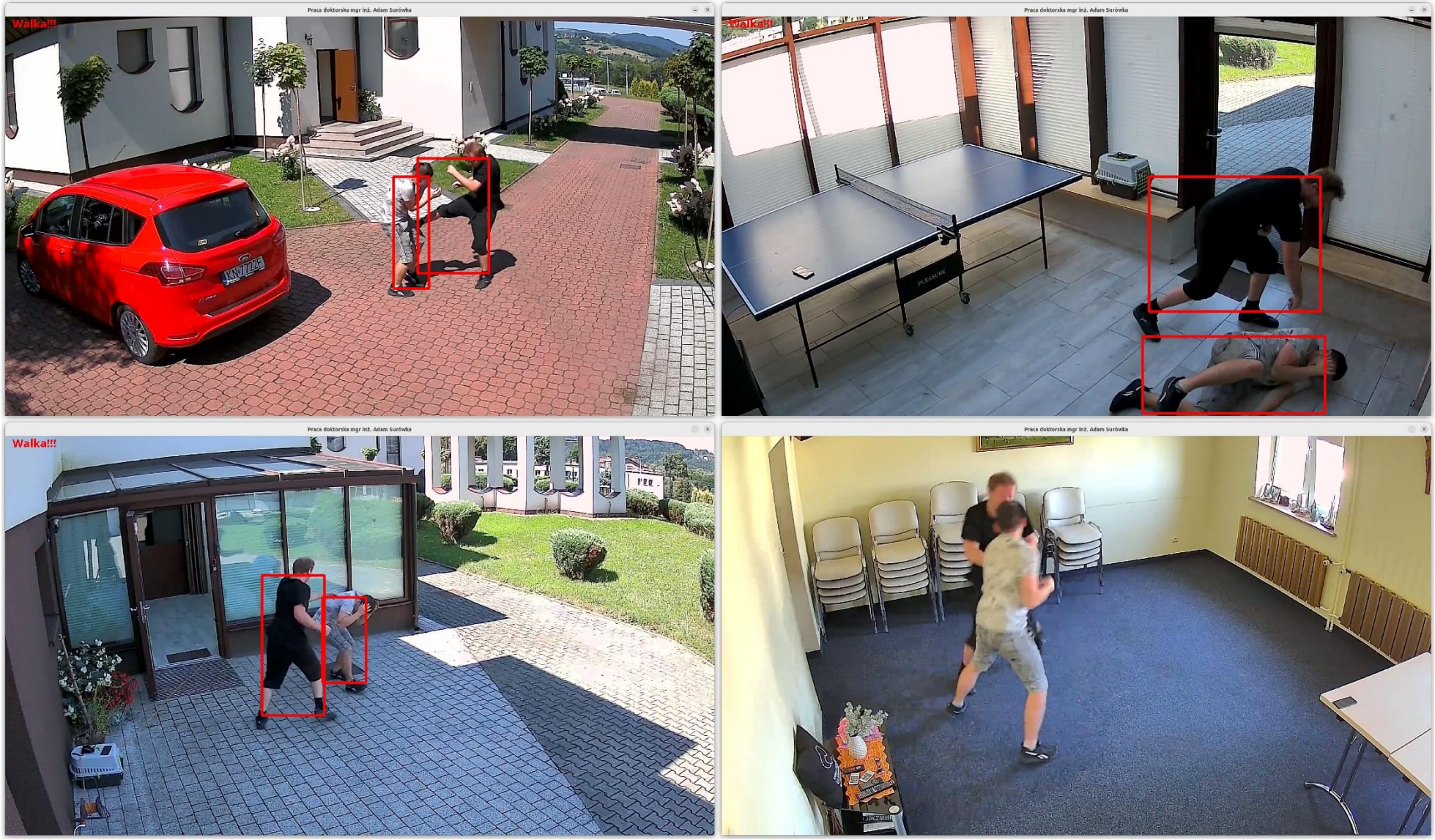
Projektowany algorytm został poddany badaniom empirycznym, mającym na celu określenie stopnia efektywności w środowisku rzeczywistym. W tym celu użyto zestawu nagrań wideo o łącznej liczbie 15900 ujęć (zob. rozdział 4.4.1). Wyznaczniki oceny jakości pracy algorytmu zostały obliczone na podstawie wzorów opisanych w rozdziale 2.10, natomiast ich wyniki zestawiono w Tab. 6.17.

Tab. 6.17 Zestawienie wyników badań empirycznych algorytmu detekcji przemocy

TP	***FP***	***TN***	***FN***	**Dokładność**	**Precyzja**	**Czułość**	F_1
4724	1040	6691	3445	72%	82%	58%	68%

Analiza otrzymanych wyników badań wskazuje na zadawalającą skuteczność algorytmu w rozpoznawaniu zdarzenia przemocy, o czym świadczą wskaźniki dokładności rzędu 72%, precyzji 82%, czułości 58% oraz F_1 68%.

Na Rys. 6.40 przedstawiono przykładowe ujęcia poklatkowe, zapisane podczas testów empirycznych algorytmu. Informacja dotycząca detekcji przemocy, wyświetlana jest w lewym górnym rogu obrazu, w postaci komunikatu tekstowego w kolorze czerwonym. Dodatkowo, wszystkie osoby znajdujące się w zasięgu potencjalnego agresora, zostają oznaczane czerwoną ramką, ułatwiającą użytkownikowi lokalizację zdarzenia. Ujęcie znajdujące się w prawej dolnej części Rys. 6.40, stanowi przykład detekcji typu *FN*, które wynika z częściowej chwilowej okluzji osoby w czarnym ubraniu oraz braku widoczności ataku, wyprowadzanego przez drugą osobę stojącą tyłem do kamery.



Rys. 6.40 Testy empiryczne algorytmu detekcji przemocy

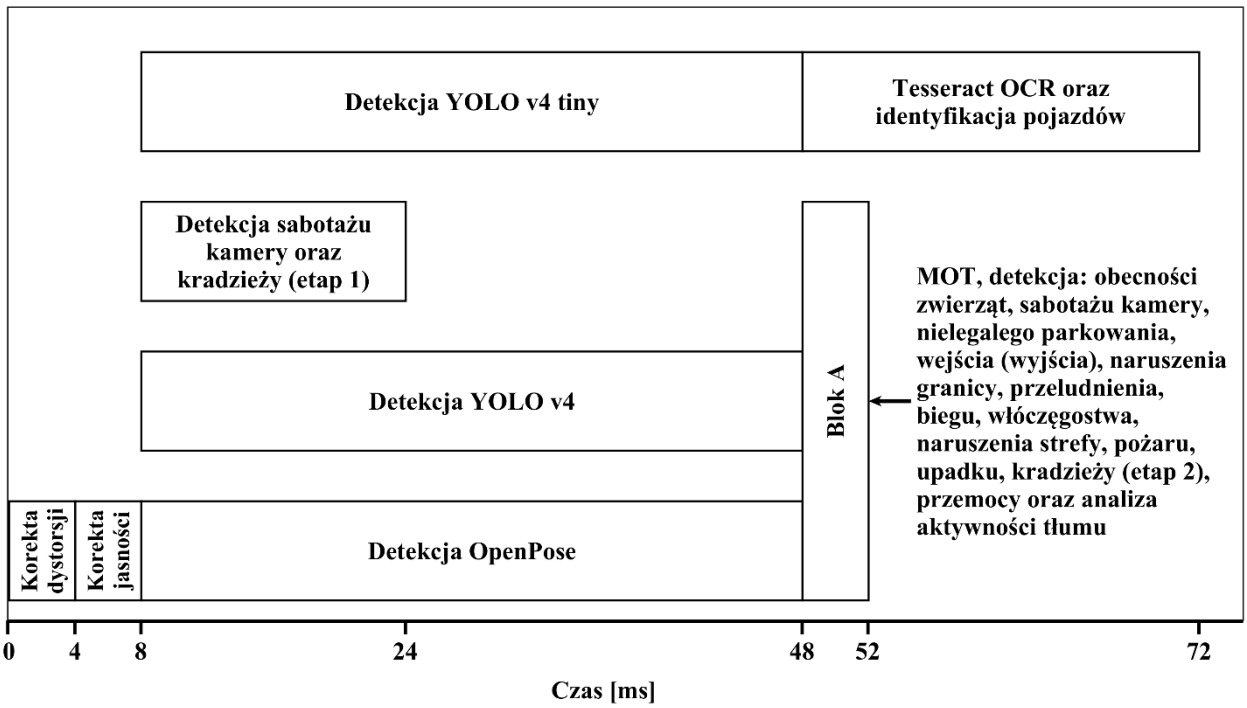
W pozostałych przypadkach detekcja przemocy na obrazie realizowana jest w sposób prawidłowy.

Podsumowując, opracowany algorytm detekcji przemocy korzysta z detektorów OpenPose oraz YOLO v4 w celu detekcji osób na obrazie, algorytmu śledzenia MOT w celu archiwizacji parametrów ruchu osób, macierzy rzutowania 3D w celu określenia rzeczywistego dystansu pomiędzy osobami oraz dedykowanej logiki operacyjnej. Proponowane rozwiązanie stanowi rozwinięcie metody, opisanej w pracy [120]. Detekcja zdarzenia walki realizowana jest na podstawie ekstrakcji cech odległościowych, kątowych, dynamicznych i kształtowych modelu szkieletowego człowieka, wygenerowanego przy użyciu detektora OpenPose. Oryginalność zaproponowanego rozwiązania dotyczy analizy szybkości zmian 20 kątów modelu szkieletowego oraz użycia macierzy rzutowania 3D, w celu określenia rzeczywistego dystansu pomiędzy osobami na monitorowanej scenie. Realizacja badań opartych na testach empirycznych, umożliwiła dokonanie oceny pracy algorytmu w warunkach rzeczywistych, wykazując zadowalającą skuteczność, o czym świadczą wartości wskaźników oceny jakości zestawione w Tab. 6.17. W trakcie realizacji badań zaobserwowano, iż jedną z częstszych przyczyn występowania błędnych detekcji typu *FN*, są okluzje, wynikające z chwilowego całkowitego zasłonięcia osoby lub brak detekcji osoby na obrazie. Dalsze badania powinny skupić się na optymalizacji algorytmu pod kątem poprawy stabilności detekcji osób na obrazie. Rezultaty przeprowadzonych badań dowodzą, iż efektywność proponowanego algorytmu jest zadowalająca, co czyni go wartościowym komponentem w projektowanym systemie IVSS.

6.15 Synteza zaawansowanych technik wizyjnych

Niniejszy rozdział zawiera odniesienie do istoty i problematyki syntezy zaawansowanych technik widzenia komputerowego oraz modeli uczenia maszynowego w ramach wielozadaniowego systemu IVSS. Synteza ta obejmuje integrację algorytmów wstępnego przetwarzania obrazów (korekta zniekształceń oraz jasności), detektorów obiektów (OpenPose, YOLO v4, YOLO v4 tiny oraz Tesseract OCR), algorytmu MOT oraz algorytmów detekcji piętnastu zdarzeń alarmowych. Przedstawione informacje dotyczą aspektów wielowątkowości opracowanej aplikacji, ukierunkowanej na możliwość osiągnięcia wysokiego stopnia optymalizacji szybkości działania. Ponadto, omówione zostaną szczegóły dotyczące badań, przeprowadzonych w celu określenia poprawności integracji wszystkich komponentów, ze wskazaniem oceny jakościowej oraz ilościowej pracy systemu w konfiguracji wielozadaniowej.

Na Rys. 6.41 przedstawiono chronologiczną sekwencję procesów, realizowanych przez system, w celu pełnej analizy pojedynczego obrazu, ze wskazaniem bloków wielowątkowych oraz czasów przetwarzania. Proces przetwarzania każdego ujęcia wideo rozpoczyna się od algorytmu korekty dystorsji oraz korekty jasności obrazu, wykonywanych sekwencyjnie w wątku głównym



Rys. 6.41 Chronologia wielowątkowości systemu z uwzględnieniem czasów przetwarzania

CPU. Celem jest stosowne przygotowanie obrazu dla późniejszej detekcji zdarzeń alarmowych. Każdy z tych algorytmów wykonywany jest w podobnym czasie około 4 ms.

Po wstępnej obróbce obrazu uruchamiany zostaje blok, składający się z czterech procesów pracujących równolegle. Blok ten tworzą detektory OpenPose, YOLO v4, YOLO v4 tiny oraz pierwszy etap algorytmów detekcji sabotażu kamery i detekcji kradzieży. Równoległa obsługa tych procesów umożliwia znaczne skrócenie czasu przetwarzania do około 40 ms łącznie. Algorytmy detekcji sabotażu kamery i detekcji kradzieży działają w oparciu o tę samą logikę operacyjną. Z uwagi na złożoność obliczeniową owych algorytmów, autor postanowił dokonać ich podziału na dwa etapy. W rezultacie, etap pierwszy (z czasem wykonania około 16 ms) realizowany jest równolegle z detektorami obiektów, natomiast etap drugi (z czasem wykonania około 0.5 ms) realizowany jest w kolejnym bloku czasowym.

W następnym kroku, system uruchamia kolejny blok przetwarzania równoległego, integrujący procesy związane z detektorem Tesseract OCR i algorytmem identyfikacji pojazdów (z czasem wykonania około 24 ms) oraz wszystkie pozostałe algorytmy detekcji zdarzeń alarmowych (w tym algorytm śledzenia MOT), z czasem przetwarzania około 4 ms.

Kolejność wykonywania poszczególnych procesów oraz możliwość uruchomienia ich w blokach przetwarzania równoległego została opracowana na podstawie występujących między nimi zależności. Procesy realizujące wstępne przygotowanie obrazu muszą zostać wykonane jako pierwsze w sposób sekwencyjny, z uwagi na konieczność modyfikacji tego samego obrazu przed dalszą obróbką. Dalej, wstępnie przetworzony obraz, może zostać przetworzony przy użyciu detektorów obiektów w sposób równoległy, gdyż nie wprowadzają one zmian w analizowanym obrazie. Pozytywny wynik detekcji obiektów jest wymagany w celu realizacji wszystkich

pozostałych algorytmów detekcji zdarzeń alarmowych oraz algorytmu śledzenia. Algorytmy te są uruchamiane sekwencyjnie, gdyż ich nieskomplikowana logika, oparta głównie na prostych funkcjach decyzyjnych, nie wpływa znacząco na szybkość pracy systemu.

Przedstawiona forma integracji procesów przetwarzania obrazu umożliwiła osiągnięcie średniego czasu analizy pojedynczego ujęcia wideo około 72 ms przy włączonych wszystkich funkcjach, co jest wynikiem satysfakcjonującym, biorąc pod uwagę złożoność operacji. Należy zwrócić uwagę, iż funkcja identyfikacji pojazdów stanowi najbardziej złożoną obliczeniowo logikę, której łączny czas wykonania definiuje jednocześnie maksymalny czas analizy pojedynczego obrazu. Wyłączenie detekcji ANPR skraca czas przetwarzania do około 52 ms.

W celu określenia stopnia skuteczności zaprojektowanego systemu IVSS do pracy w trybie wielozadaniowym, przeprowadzono szereg testów z użyciem dotychczas opracowanego zbioru nagrań wideo (zob. rozdział 4.4.1). Na Rys. 6.42 przedstawiono cztery przykładowe ujęcia, zapisane podczas pracy systemu uruchomionego w trybie wielozadaniowym. W lewym górnym rogu obrazu zamieszczono rezultat jednoczesnej detekcji osoby leżącej, jej obecności w strefie zastrzeżonej (cały obszar trawnika), detekcji zwierzęcia (psa) oraz detekcji nielegalnego parkowania (niebieski samochód). W prawym górnym rogu zamieszczono rezultat detekcji dwóch osób (w białych podkoszulkach), znajdujących się w strefie zastrzeżonej, detekcję jednej leżącej osoby, detekcję naruszenia wirtualnej granicy przez osobę w czarnym ubraniu oraz detekcję nielegalnego parkowania (srebrny samochód). W lewym dolnym rogu obrazu zamieszczono rezultat jednoczesnej detekcji zwierzęcia (kota), nielegalnego parkowania (czerwony samochód) oraz detekcję naruszenia strefy zastrzeżonej, obejmującej całe schody. W prawym dolnym rogu obrazu zamieszczono rezultat detekcji kradzieży przedmiotu statycznego, naruszenia strefy zastrzeżonej (obszar całego pomieszczenia) przez złodzieja oraz wynik liczenia wejść do pomieszczenia prawidłowo wskazujący wartość 1.

Podsumowując, analiza wyników przeprowadzonych badań wskazuje pełną kompatybilność wszystkich opracowanych algorytmów detekcji zdarzeń alarmowych. System IVSS jest w stanie realizować pracę wielozadaniową w sposób prawidłowy, niezależnie od liczby uruchomionych jednocześnie funkcji. Opisana strategia syntezy zaawansowanych technik widzenia komputerowego obejmuje integrację detektorów OpenPose, YOLO v4, YOLO v4 tiny oraz Tesseract OCR, a także dedykowanych algorytmów opracowanych przez autora. Zastosowanie wielowątkowego trybu pracy wybranych funkcji pozwoliło na osiągnięcie przetwarzania danych wizyjnych w czasie rzeczywistym. Na podstawie przeprowadzonych badań empirycznych stwierdzono, iż proponowany wielozadaniowy system IVSS stanowi w pełni funkcjonalne narzędzie, gotowe do implementacji w środowisku rzeczywistym.

Rys. 6.42 Testy empiryczne systemu w konfiguracji wielozadaniowej

6.16 Podsumowanie

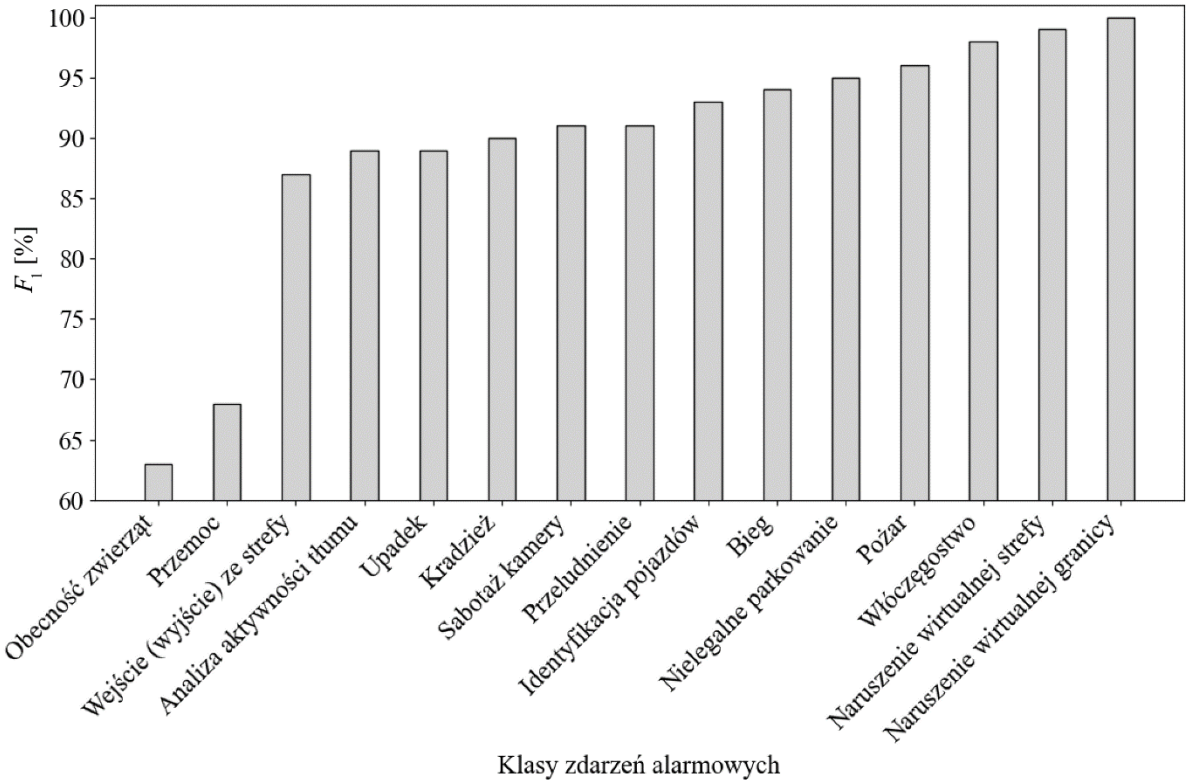
Autor opracował piętnaście algorytmów detekcji zdarzeń alarmowych. Są to: identyfikacja pojazdów (zob. rozdział 6.1), niepożądana obecność zwierząt (zob. rozdział 6.2), sabotaż kamery (zob. rozdział 6.3), nielegalne parkowanie (zob. rozdział 6.4), wejście (wyjście) ze strefy (zob. rozdział 6.5), naruszenie wirtualnej granicy (zob. rozdział 6.6), przeludnienie (zob. rozdział 6.7), bieg oraz analiza aktywności tłumu (zob. rozdział 6.8), włóczęgostwo (zob. rozdział 6.9), naruszenie wirtualnej strefy (zob. rozdział 6.10), pożar (zob. rozdział 6.11), upadek (zob. rozdział 6.12), kradzież obiektu statycznego (zob. rozdział 6.13) oraz przemoc (zob. rozdział 6.14).

W celu określenia stopnia skuteczności zaprojektowanego systemu IVSS, do pracy w trybie wielozadaniowym, przeprowadzono szereg testów z użyciem dotychczas opracowanego zbioru nagrań wideo (zob. rozdział 4.4.1). Zaproponowana forma integracji procesów przetwarzania obrazu (zob. rozdział 6.15), umożliwiła osiągnięcie średniego czasu analizy pojedynczego ujęcia wideo około 72 ms, przy aktywowanych wszystkich funkcjach, co jest wynikiem satysfakcjonującym, biorąc pod uwagę złożoność poszczególnych operacji. Największy udział w czasie analizy pojedynczego ujęcia ma funkcja identyfikacji pojazdów. Jej deaktywowanie skraca czas przetwarzania do około 52 ms.

Opracowane algorytmy zostały poddane badaniom empirycznym. Wyniki badań przedstawiono w Tab. 6.18. Wskaźnik F_1 uwzględnia zarówno liczbę fałszywych alarmów, jak i pominiętych trafień. Wobec tego można przyjąć, iż stanowi on bardziej miarodajną ocenę skuteczności działania algorytmu, szczególnie w przypadku niezrównoważonych danych, niż pojedyncze wyznaczniki typu dokładność, precyzja czy czułość. Mając na uwadze powyższe, dalsza analiza skuteczności pojedynczych algorytmów oraz całego systemu będzie bazować głównie na wskaźniku F_1. Na Rys. 6.43 przedstawiono wartości wskaźnika F_1, osiągnięte przez poszczególne algorytmy detekcji. Analiza wyników przedstawionych w Tab. 6.18 oraz na Rys. 6.43 pozwala stwierdzić, iż zaimplementowane następujące algorytmy: detekcja naruszenia wirtualnej granicy oraz strefy, detekcja włóczęgostwa, pożaru oraz nielegalnego parkowania cechuje najwyższa wartość wskaźnika F_1, dochodząca do 100%. Świadczy to o skutecznej klasyfikacji zdarzeń, gdzie informacje wizyjne są wyraźne i jednoznaczne. Algorytmy odpowiadające za detekcję zdarzeń dynamicznie zmiennych, takich jak bieganie czy przeludnienie, również uzyskały zadowalające wyniki, co świadczy o skutecznej klasyfikacji przypadków, związanych z nagłą zmianą zachowania w obserwowanej przestrzeni. Wartość wskaźnika F_1 = 93% świadczy o zadawalającej skuteczności działania algorytmu identyfikacji pojazdów, pomimo najwyższego stopnia złożoności implementacji na tle pozostałych algorytmów. Najniższą wartość wskaźnika F_1 odnotowano w przypadku detekcji przemocy oraz obecności zwierząt.

Tab. 6.18 Podsumowanie wyników badań wszystkich opracowanych algorytmów

Typ detekcji zdarzenia		***TP***	***FP***	***TN***	***FN***	**Dokładność [%]**	**Precyzja [%]**	**Czułość [%]**	**F_1 [%]**
Identyfikacja pojazdów		15108	2301	0	4	87	87	100	93
Obecność zwierząt		6204	99	2688	7080	55	98	47	63
Sabotaż kamery	Zasłonięcie	3881	513	1672	167	89	88	96	92
	Obrót	3259	250	1170	733	82	93	82	87
	Oślepienie	3079	311	1472	54	93	91	98	94
Nielegalne parkowanie		13618	0	3673	1364	93	100	91	95
Wejście (wyjście) ze strefy		430	51	14645	76	99	89	85	87
Naruszenie wirtualnej granicy		120	0	14150	0	100	100	100	100
Przeludnienie		11367	49	3299	2189	87	100	84	91
Bieg		2675	12	7012	353	96	100	88	94
Analiza aktywności tłumu		1398	12	5849	349	95	99	80	89
Włóczęgostwo		9783	141	7053	224	98	99	98	98
Naruszenie wirtualnej strefy		12531	33	2493	282	98	100	98	99
Pożar		14120	67	1584	1162	93	100	92	96
Upadek		11025	432	2960	2160	84	96	84	89
Kradzież		7021	0	9493	1489	92	100	83	90
Przemoc		4724	1040	6691	3445	72	82	58	68
Wartości uśrednione		**120343**	**5311**	**85904**	**21131**	**87**	**96**	**85**	**90**



Rys. 6.43 Wartości wskaźnika F_1 dla poszczególnych algorytmów detekcji

Trudności związane z detekcją przemocy mogą wynikać ze złożoności scenariuszy, jak również dynamiki zmienności oraz niejednoznaczności informacji zawartych w sygnale wizyjnym.

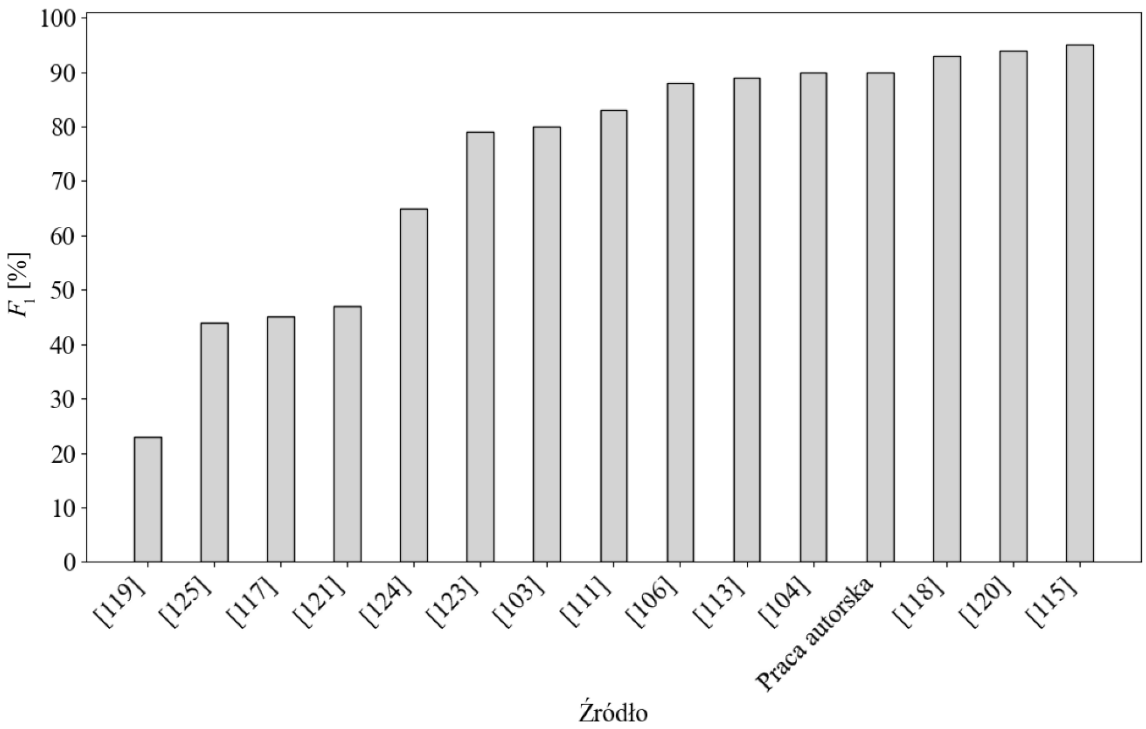
Najczęstszymi zaobserwowanymi przyczynami występowania detekcji typu *FN* oraz *FP* są okluzje, niestabilność detekcji oraz niestabilność procesu śledzenia obiektów na obrazie. Dalsze prace badawcze zostaną zatem zorientowane na zminimalizowanie wpływu tych czynników. W tym celu, planuje się opracowanie pojedynczego modelu DL, przeznaczonego zarówno do detekcji, klasyfikacji jak i również śledzenia wszystkich rozpatrywanych klas obiektów. Trening takiego modelu powinien zostać przeprowadzony z użyciem zbioru danych, uwzględniających przypadki częściowej okluzji oraz charakterystyczny dla monitoringu wizyjnego punkt widzenia kamery (z góry).

W celu oceny jakości opracowanego wielozadaniowego systemu detekcji zdarzeń, dokonano porównania otrzymanych wyników z wynikami osiągniętymi przez systemy referencyjne, opisane w literaturze naukowej (zob. rozdział 3.4). Porównanie to opiera się na najczęściej cytowanych wyznacznikach oceny jakości, takich jak dokładność, precyzja, czułość oraz wskaźnik F_1, których chronologiczne zestawienie opracowano w Tab. 6.19. W przypadku braku informacji dotyczących wartości wskaźnika F_1, wartość ta została obliczona przez autora rozprawy na podstawie zależności 2.9 oraz wskaźników precyzji i czułości (jeżeli zostały one opublikowane w danej pracy). W przypadku, gdy wskaźniki oceny jakości zostały opublikowane dla dwóch lub więcej baz danych, wówczas w Tab. 6.19 przedstawiono ich wartości uśrednione.

Tab. 6.19 Zestawienie wartości wyznaczników oceny jakości pracy referencyjnych systemów IVSS

Źródło	Data	Dokładność [%]	Precyzja [%]	Czułość [%]	F_1 [%]
[97]	1997	-	-	-	-
[98]	2004	-	-	-	-
[99]	2005	72	-	-	-
[100]	2005	-	-	-	-
[101]	2009	-	-	-	-
[103]	2012	-	92	70	80
[104]	2014	83	93	88	90
[106]	2014	-	100	78	88
[109]	2015	91	-	-	-
[110]	2017	90	-	-	-
[111]	2018	-	84	82	83
[112]	2019	-	-	-	-
[113]	2019	86	98	81	89
[115]	2020	94	96	95	95
[116]	2021	59	-	-	-
[117]	2021	45	45	45	45
[118]	2021	92	95	91	93
[119]	2022	23	23	24	23
[120]	2022	94	97	91	94
[121]	2023	47	47	47	47
[122]	2023	78	-	-	-
[123]	2024	-	79	79	79
[124]	2024	95	50	95	65
[125]	2024	51	47	42	44
Praca autorska	**2024**	**87**	**96**	**85**	**90**

Część analizowanych prac badawczych nie zawiera wyników badań w formie rozpatrywanych wyznaczników oceny jakości, co dodatkowo utrudnia przeprowadzenie rzetelnego odniesienia. Próba porównania skuteczności opracowanego systemu IVSS z systemami referencyjnymi, została zrealizowana głównie na podstawie wartości wskaźnika F_1 (zob. Rys. 6.44). Na podstawie analizy danych zawartych w Tab. 6.19 oraz na Rys. 6.44 można stwierdzić, iż opracowany system IVSS charakteryzuje się konkurencyjnymi wynikami w stosunku do systemów referencyjnych, szczególnie pod względem osiąganych wartości wskaźników precyzji (96%) oraz F_1 (90%). Wskaźnik dokładności obliczony dla całego systemu (87%), również jest na poziomie zadowalającym i porównywalnym z wieloma nowoczesnymi rozwiązaniami. Osiągi systemu dorównują lub nieznacznie ustępują systemom referencyjnym [104,115,118,120] pod kątem wybranych wskaźników oceny jakości. Należy jednak zauważyć, iż wyniki jakości pracy wspomnianych systemów referencyjnych zostały zbadane dla różnego typu konfiguracji sprzętowej i programowej. Systemy te różnią się liczbą i rodzajem detekcji zdarzeń alarmowych.



Rys. 6.44 Porównanie skuteczności opracowanego systemu IVSS z systemami referencyjnymi

Wobec tego, przeprowadzenie miarodajnego porównania jest co najmniej trudne. Na podstawie próby porównania opracowanego systemu IVSS z systemami referencyjnymi [104,115,118,120] wynika, że:

- Podstawową zaletą proponowanego rozwiązania jest jego wszechstronność. Działanie systemów referencyjnych jest ograniczone do kilku rodzajów zdarzeń spośród tych, które były przedmiotem badań. Autorski system IVSS przeznaczony jest do identyfikacji znacznie szerszego spektrum sytuacji alarmowych. Implementacja wszystkich piętnastu niezależnie parametryzowanych algorytmów, zapewnia znacznie szerszą funkcjonalność, spełniającą wymagania współczesnych systemów nadzoru publicznego.
- Autorski system IVSS działa w oparciu o syntezę zaawansowanych modeli widzenia komputerowego: YOLO v4 (detekcja obiektów), OpenPose (analiza ruchu i sylwetek osób) oraz Tesseract OCR (rozpoznawanie tekstu), analizę jasności obrazu, ekstrakcję tła, detekcję krawędzi oraz rzutowania 3D. Taka integracja umożliwia bardziej kompleksową analizę sceny – od rozpoznawania numerów tablic rejestracyjnych, po analizę aktywności ludzi. Wspomniane systemy referencyjne, choć mogą cechować się wyższą precyzją przy detekcji określonych zdarzeń, nie oferują możliwości syntezy różnych źródeł danych wizyjnych, co znacząco wpływa na ograniczenie ich modularności.
- Zaproponowany system IVSS umożliwia stosunkowo precyzyjną analizę przestrzenną sceny. Dzięki temu system może określać odległości między obiektami i śledzić ich ruchy

w odniesieniu do układu 3D przestrzeni rzeczywistej. Jest to szczególnie przydatne w analizie zdarzeń dynamicznych, gdzie analiza szybkości oraz dystansu między obiektami jest kluczowa. Żaden ze wspomnianych systemów referencyjnych nie oferuje tak zaawansowanej funkcjonalności.

- Choć inne systemy referencyjne mogą osiągać wyższe wartości wskaźników oceny jakości w odniesieniu do specyficznych zdarzeń, opracowany system IVSS pozwala analizować zarówno cechy dynamiczne (np. analiza trajektorii przemieszczania się osób), jak i statyczne (np. detekcja ognia czy sabotażu kamery). Połączenie tych funkcji oraz możliwość ich pełnej konfiguracji zapewnia systemowi wyższy poziom ogólnej użyteczności oraz skalowalności szczególnie w środowiskach, w których różnorodność zagrożeń wymaga bardziej kompleksowego podejścia.

Podsumowując, otrzymane rezultaty przeprowadzonych badań, potwierdzają zdolność systemu do realizacji zadań detekcji, identyfikacji i analizy rozpatrywanych zdarzeń w stopniu zadawalającym. System IVSS jest w stanie realizować pracę wielozadaniową w sposób prawidłowy, niezależnie od liczby uruchomionych jednocześnie funkcji, dzięki czemu może stanowić w pełni funkcjonalne oprogramowanie, współpracujące z już istniejącymi systemami monitoringu wizyjnego 2D. W porównaniu do systemów referencyjnych, skalowalność, integracja wielu zaawansowanych technologii oraz szeroki zakres funkcjonalny sprawiają, że proponowany system jest bardziej uniwersalnym, elastycznym i wszechstronnym narzędziem monitoringu wizyjnego w różnych kontekstach nadzoru przestrzeni publicznej.

7 Podsumowanie

Przedmiotem badań niniejszej rozprawy było opracowanie metody syntezy istniejących, zaawansowanych technik przetwarzania i analizy obrazu, w celu opracowania inteligentnej aplikacji komputerowej, realizującej automatyczne wykrywanie wybranych zdarzeń alarmowych. Na wstępie autor przeprowadził szczegółową analizę literatury w zakresie wielozadaniowych systemów IVSS. W jej wyniku zdefiniowano wyzwania i problemy badawcze. W celu realizacji badań, autor posłużył się takimi narzędziami jak: detektor obiektów YOLO v4, detektor modelu szkieletowego ciała człowieka OpenPose oraz detektor tekstu Tesseract OCR. W ramach przeprowadzonych prac badawczych opracowano zintegrowany system monitoringu wizyjnego, posiadający takie funkcjonalności jak: analiza przestrzenna sceny 3D, automatyczna korekta jasności obrazu, śledzenie obiektów MOT, identyfikacja pojazdów, detekcja niepożądanej obecności zwierząt, sabotażu kamery, nielegalnego parkowania, wejścia (wyjścia) ze strefy, naruszenia wirtualnej granicy, przeludnienia, biegu oraz analiza aktywności tłumu, detekcja włóczęgostwa, naruszenia wirtualnej strefy, pożaru, upadku, kradzieży obiektu statycznego oraz przemocy. Realizacja ww. funkcji została zintegrowana w formie programu komputerowego, przeznaczonego do pracy na popularnych platformach systemowych MS Windows, Linux czy Mac OS. Dzięki implementacji wielowątkowej z użyciem procesorów CPU oraz GPU osiągnięto przetwarzanie w czasie rzeczywistym. Wyniki przeprowadzonych przez autora badań pozwalają wnioskować, iż zaproponowane przezeń tezy przedstawione w rozdziale pierwszym zostały zrealizowane.

Zaproponowana metoda syntezy zaawansowanych technik widzenia komputerowego, opartych na modelach uczenia maszynowego, takich jak YOLO v4, OpenPose oraz Tesseract OCR, umożliwiła stworzenie rozbudowanego systemu IVSS, zdolnego do automatycznej detekcji i klasyfikacji wybranych zdarzeń alarmowych. Opracowany system stanowi funkcjonalne narzędzie, które można wdrożyć do pracy w środowiskach rzeczywistych, wykorzystując dane z już istniejących infrastruktur monitoringowych, opartych na konwencjonalnych kamerach 2D. Świadczą o tym rezultaty badań empirycznych, przeprowadzonych z użyciem nagrań testowych. Analiza wyników przeprowadzonych badań wskazuje, iż poszczególne algorytmy indywidualnie, jak również kompletny system IVSS, charakteryzuje się poprawnym działaniem i zadawalającą skutecznością wykrywania zdarzeń alarmowych, o czym świadczą osiągnięte uśrednione wskaźniki oceny jakości: dokładność 87%, precyzja 96%, czułość 85% oraz F_1 90%. Przedstawione wnioski dowodzą pierwszej tezy badawczej.

Opracowany zestaw algorytmów przetwarzania danych wizyjnych umożliwia ekstrakcję cech, identyfikację, klasyfikację oraz śledzenie wielu klas obiektów, w tym osób, pojazdów (samochodów osobowych, ciężarowych, autobusów, motocykli) oraz zwierząt (psów i kotów). W

połączeniu ze zdolnością systemu do zaawansowanej analizy przestrzennej sceny możliwa jest realizacja kompleksowej analizy zachowań, aktywności oraz interakcji wspomnianych klas obiektów, celem wnioskowania i raportowania wybranych zdarzeń alarmowych. Rozpatrywane typy zdarzeń alarmowych stanowią świadomy i przemyślany wybór autora rozprawy, uzasadniony istotnością z punktu widzenia nadzoru bezpieczeństwa publicznego. Powyższe wnioski poparte rezultatami badań empirycznych dowodzą drugiej tezy badawczej.

Według stanu wiedzy autora, do oryginalnych rozwiązań wprowadzonych w rozprawie należą:

- opracowanie złożonego systemu modułowego, oferującego automatyczne wykrywanie zdarzeń alarmowych na podstawie syntezy różnych typów informacji wizyjnej, takich jak modele szkieletowe, BB, tekst, jasność obrazu, ekstrakcja tła, detekcja krawędzi oraz rzutowanie 3D w oryginalnej konfiguracji,
- opracowanie bazy danych filmów testowych o łącznej liczbie 233786 ujęć, manualnie oznaczonych przez autora rozprawy w sposób umożliwiający precyzyjną identyfikację 15 klas zdarzeń alarmowych (zob. rozdział 4.4.1),
- opracowanie algorytmu automatycznej korekty jasności obrazu (zob. rozdział 5.2),
- opracowanie uniwersalnego algorytmu kalibracji sceny (zob. rozdział 5.5), niewymagającego użycia specjalistycznych przyrządów wzorcowo-pomiarowych,
- opracowanie algorytmów identyfikacji pojazdów ANPR (zob. rozdział 6.1), detekcji niepożądanej obecności zwierząt (pies oraz kot) (zob. rozdział 6.2), a także detekcji sabotażu kamery (zasłonięcie, oślepienie oraz obrót obiektywu) (zob. rozdział 6.3), które jak dotąd nie znalazły zastosowania w analizowanych systemach referencyjnych,
- rozwinięcie algorytmu detekcji nielegalnego parkowania [101], pozwalające na użycie danych z detektora obiektów YOLO v4 (zob. rozdział 6.4),
- rozwinięcie algorytmu detekcji wejścia (wyjścia) ze strefy [109], pozwalające na precyzyjną ocenę kierunku przekroczenia wirtualnej linii (zob. rozdział 6.5),
- rozwinięcie algorytmu detekcji naruszenia wirtualnej granicy [112], pozwalające na uwzględnienie danych historycznych dotyczących ścieżek ruchu analizowanych osób (zob. rozdział 6.6),
- rozwinięcie algorytmu detekcji przeludnienia [97,99], pozwalające na użycie syntezy danych z detektorów OpenPose oraz YOLO v4 (zob. rozdział 6.7),
- opracowanie algorytmów detekcji biegu i analizy aktywności tłumu (zob. rozdział 6.8) oraz włóczęgostwa (zob. rozdział 6.9) w oparciu o rzeczywiste parametry przestrzenne sceny 3D,

- opracowanie algorytmu detekcji pożaru z użyciem specjalnie wytrenowanego modelu YOLO v4 tiny, umożliwiającego detekcję obiektów klasy „*ogień*” (zob. rozdział 6.11),
- rozwinięcie algorytmu detekcji upadku osoby [120] pozwalające na identyfikację tego zdarzenia, niezależnie od orientacji osoby względem pola widzenia kamery, kierunku oraz szybkości przemieszczania się, interakcji z innymi ludźmi i obiektami, a nawet niezależnie od szybkości upadania (zob. rozdział 6.12),
- opracowanie algorytmu detekcji kradzieży statycznych przedmiotów, oznaczonych na obrazie, rozwiązującego problem okluzji, wynikającej z obecności osób oraz wybranych klas obiektów (zob. rozdział 6.13),
- rozwinięcie algorytmu detekcji przemocy [120], pozwalające na analizę szybkości zmian 20 kątów modelu szkieletowego oraz wykorzystanie rzeczywistych parametrów przestrzennych sceny 3D (zob. rozdział 6.14).

Przeprowadzone testy funkcjonalności zaprojektowanego systemu IVSS zostały ukierunkowane na analizę nagrań monitoringowych wykonanych w warunkach dziennego oświetlenia. Decyzja ta wynikała zarówno z ograniczeń czasowych towarzyszących realizacji badań, jak i z trudności w uzyskaniu dostępu do obiektów użytku publicznego w godzinach nocnych, a także z problemów ze znalezieniem ochotników do symulowania zdarzeń alarmowych w porze nocnej. Autor pragnie nadmienić, iż zastosowanie algorytmu automatycznej korekty jasności obrazu zwiększa potencjał adaptacyjny systemu podczas obserwacji scen ze zmiennym oświetleniem. Ponadto, współczesne kamery monitoringowe są często wyposażone w funkcję doświetlania obserwowanej sceny w warunkach nocnych, na przykład w oparciu o technologię podczerwieni (IR), co może dodatkowo wspierać skuteczność działania systemu w warunkach słabego oświetlenia.

Podsumowując, przedmiot rozprawy stanowi istotny wkład w rozwój dziedziny wielozadaniowych inteligentnych systemów nadzoru wideo. Opracowane rozwiązania innowacyjne pozwoliły na rozszerzenie istniejących funkcjonalności oraz na wprowadzenie nowych algorytmów, umożliwiając stworzenie wielozadaniowego systemu IVSS, stanowiącego skuteczne narzędzie monitoringu wizyjnego. Rezultaty przeprowadzonych badań potwierdziły, iż synteza różnych typów informacji wizyjnej, takich jak modele szkieletowe, BB, tekst, jasność obrazu, ekstrakcja tła, detekcja krawędzi oraz rzutowanie 3D, korzystnie wpływają na zdolność systemu do identyfikacji szerokiego spektrum zdarzeń alarmowych z zakresu bezpieczeństwa publicznego. Efekty tych badań wskazują także na znaczący potencjał aplikacyjny opracowanego systemu, który może być wykorzystany w wielu różnych sektorach, takich jak zarządzanie bezpieczeństwem publicznym, infrastruktura krytyczna czy komercyjne systemy nadzoru. Należy jednak pamiętać, że tego typu systemy, pomimo iż stanowią istotne wsparcie personelu obsługi

monitoringu, nie powinny zastępować człowieka w procesie podejmowania ostatecznej decyzji w sytuacjach kryzysowych.

Dalsze prace badawcze zostaną zorientowane na zminimalizowanie wpływu czynników, powodujących występowanie detekcji typu *FP* oraz *FN*, a więc okluzji, niestabilności detekcji oraz niestabilność procesu śledzenia obiektów na obrazie. W tym celu autor proponuje opracowanie pojedynczego modelu DL, przeznaczonego zarówno do detekcji, klasyfikacji jak i również śledzenia rozpatrywanych klas obiektów. Trening takiego modelu powinien zostać przeprowadzony z użyciem zbioru danych, uwzględniających przypadki częściowej okluzji oraz charakterystyczny dla monitoringu wizyjnego punkt widzenia kamery (z góry). Jako przykład tego typu implementacji można wskazać model CTracker [143]. Model ten analizuje pary następujących po sobie ujęć wideo, przewidując lokalizacje BB dla tych samych obiektów. Następnie, poprzez zastosowanie mechanizmu łączenia (*chaining*) oraz modułu uwagi (*attention*), model koncentruje analizę na istotne obszary obrazu, co zwiększa efektywność zarówno detekcji, jak i klasyfikacji, a także poprawia dokładność śledzenia obiektów w dynamicznych i złożonych scenach. Kierunki dalszych prac obejmują również rozszerzenie testów o dane pozyskane w zmiennych warunkach oświetleniowych, co pozwoli na pełniejszą ocenę uniwersalności i efektywności opracowanego systemu. Kolejnym istotnym aspektem rozwoju jest implementacja funkcji, umożliwiającej rejestrowanie podejrzanych aktywności w formie nagrań video oraz dziennika zdarzeń. Ma to na celu ułatwienie odszukania informacji dotyczących czasu, miejsca oraz okoliczności wystąpienia zdarzenia alarmowego w zapisie monitoringu, a także identyfikacji potencjalnych uczestników. Następnym krokiem będzie implementacja funkcji, umożliwiającej zdalne raportowanie użytkownika o potencjalnym wystąpieniu zdarzenia alarmowego. Zadanie to może zostać zrealizowane, np. poprzez automatyczne wysyłanie wiadomości e-mail lub SMS do użytkownika systemu w momencie detekcji określonej aktywności.

W ramach możliwości praktycznego zastosowania opracowanego systemu, rozpoczęto negocjacje, w celu ustalenia warunków jego wdrożenia do istniejącej infrastruktury monitoringu obiektów sakralnych w Parafii Najświętszej Maryi Panny Częstochowskiej w Nowym Sączu. Nadzór wizyjny obiektów sakralnych stanowi zatem pierwszą grupę docelową opracowanego systemu, natomiast nie ogranicza jego zastosowania do pracy wyłącznie w tym sektorze.

Dorobek naukowy

Wykaz publikacji doktoranta w tematyce rozprawy:

[1] A. Surówka (2020) Zastosowanie technik sztucznej inteligencji w systemach nadzoru wideo. *Konferencja Młodych Naukowców nt.: Nowe Trendy w Badaniach Naukowych – Wystąpienie Młodego Naukowca – edycja I.*

[2] A. Surówka (2020) Hybrid imaging-AI approach for handling critical situations in a fast-changing environment: preliminary study. *In MATEC Web of Conferences EDP Sciences*, 322, s. 1–7, DOI:10.1051/matecconf/202032201048, indeksowana w: Web of Science.

[3] A. Surówka (2020) Automatic video surveillance system: recognition of critical behaviors in the process of machine learning. *In 2020 21st International Conference on Research and Education in Mechatronics (REM), IEEE*, s. 1–4, DOI:10.1109/REM49740.2020.9313870, indeksowana w: Web of Science, Scopus.

[4] A. Surówka (2021) Preliminary Investigation into Machine-Learned 2D Automated Video Surveillance Systems. *In 2021 22nd International Carpathian Control Conference (ICCC), IEEE*, s. 1–5, DOI:10.1109/ICCC51557.2021.9454647, indeksowana w: Web of Science.

[5] A. Surówka (2021) Real-time Multi Pose Trajectory Tracking based on OpenPose Keypoints. *In 2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), IEEE*, 2, s. 813–817, DOI:10.1109/IDAACS53288.2021.9660867, indeksowana w: Web of Science, Scopus.

[6] A. Surówka, Multi-modular 2D Intelligent Video Surveillance System. *Digital Signal Processing* – w trakcie recenzji.

Udział w projektach naukowo-badawczych:

Narzędzie informatyczne wspierające proces decyzyjny wykonania projektu integracji OZE z podziemnym magazynem gazu w kawernach solnych, projekt nr POIR.04.01.01-00-0041/19, Narodowe Centrum Badań i Rozwoju, rozpoczęcie projektu: 01/06/2021 (24 m-ce). Projekt realizowany w konsorcjum naukowo-badawczym (Politechnika Krakowska, Instytut Gospodarki Surowcami Mineralnymi i Energią PAN, PGNiG, TwinIO Energy). Charakter udziału: **wykonawca** (zatrudnienie w projekcie: 9 m-cy). Zakres obowiązków: opracowanie bazy danych uczących oraz opracowanie i wytrenowanie modelu regresyjnego, opartego na sztucznej sieci neuronowej, realizującego zadanie prognozowania rynkowej ceny energii, jako element modelu optymalizacji sprzedaży energii z magazynów (kawern solnych).

Spis ilustracji

Rys. 2.1 Klasyczny model kamery otworkowej [5] 5
Rys. 2.2 Typy kamer stosowane w systemach IVSS: a) statyczna, b) PTZ (zewnętrzna), c) PTZ (wewnętrzna), d) tubowa, e) kopułowa, f) wieżyczkowa [16] 8
Rys. 2.3 Przykłady zniekształceń kamery: a) wzorzec idealny, b)–c) zniekształcenia radialne, d)–e) zniekształcenia styczne, f) zniekształcenia pryzmatyczne, g)–h) zniekształcenia perspektywiczne [5] 9
Rys. 2.4 Przykłady detekcji i klasyfikacji obiektów na obrazie [19] 10
Rys. 2.5 Rozwój technik detekcji i klasyfikacji obiektów w ujęciu historycznym [20] 11
Rys. 2.6 Rozwój detektorów YOLO w ujęciu historycznym [38] 14
Rys. 2.7 Wykres zależności dokładności i skali modelu dla technik opartych na uczeniu maszynowym i klasycznych technik analitycznych [3] 18
Rys. 2.8 Rezultat analizy obrazu techniką ANPR [56] 20
Rys. 2.9 Graficzna reprezentacja modeli sylwetek człowieka stosowanych w systemach CV: a) model kinematyczny, b) model planarny, c) model wolumetryczny [65] 23
Rys. 2.10 Klasyfikacja technik HPE typu ML stosowanych w obrazowaniu 2D [69] 24
Rys. 2.11 Graficzna reprezentacja dwugałęziowej architektury sieci konwolucyjnej modelu OpenPose [71] 25
Rys. 2.12 Przykłady adnotacji sylwetek osób stosowane w algorytmach MOT: a) centroida, b) wielopunktowy, c) ramka (BB), d) elipsa, e) przegubowy, f) szkieletowy, g) punkty kontrolne obrysu, h) pełny obrys, i) pełny obszar kształtu (segmentacja) [83] 29
Rys. 2.13 Ogólna klasyfikacja technik śledzenia obiektów [83] 30
Rys. 2.14 Ogólna klasyfikacja typów aktywności człowieka [87] 34
Rys. 2.15 Ogólna klasyfikacja technik HAR [88] 35
Rys. 2.16 Zależność elastyczności od wydajności obliczeniowej dla różnych jednostek obliczeniowych [3] 39
Rys. 2.17 Krzywa *ROC* oraz obszar *AUC* [92] 41
Rys. 3.1 Obszary najczęstszych zastosowań systemów IVSS [93] 45
Rys. 3.2 Przykład modelu przepływu danych w systemach IVSS [96] 47
Rys. 3.3 Wykres częstości stosowania poszczególnych wskaźników oceny jakości w celu realizacji testów referencyjnych systemów IVSS 57
Rys. 3.4 Wykres częstości użycia poszczególnych baz danych, stosowanych w celu realizacji uczenia maszynowego referencyjnych systemów IVSS 58
Rys. 3.5 Wykres częstości implementacji funkcji detekcji zdarzeń alarmowych w referencyjnych systemach IVSS 60

Rys. 4.1 Ogólny schemat budowy projektowanego systemu IVSS 65
Rys. 4.2 Schemat przepływu informacji w warstwie aplikacji 66
Rys. 4.3 GUI projektowanego systemu IVSS 67
Rys. 4.4 GUI – przykład realizacji oznaczeń: a) wirtualnych linii, b) wirtualnych obszarów . 68
Rys. 4.5 Biblioteka animacji Mixamo [129] 75
Rys. 4.6 Przykłady tablic rejestracyjnych wygenerowanych przy użyciu PlatesMania [130]: a) zwykłe, b) tymczasowe, c) zabytkowe, d) samochody elektryczne 75
Rys. 5.1 Przykład aranżacji procedury kalibracji kamery z użyciem biblioteki OpenCV i wzorca szachownicy [13] 77
Rys. 5.2 Tabela nominalnych rozmiarów czujników optycznych [139] 79
Rys. 5.3 Wizualizacja wpływu redukcji zniekształceń na obraz: a) oryginalny, b) po redukcji zniekształceń (*alpha* = 1.0), c) po redukcji zniekształceń (*alpha* = 0.0) 81
Rys. 5.4 Schemat blokowy algorytmu automatycznej korekty jasności obrazu, gdzie: j – średnia ważona jasność obrazu, γ – wartość korekty 82
Rys. 5.5 $\gamma = f(j)$ 83
Rys. 5.6 Wyniki detektora OpenPose uzyskane podczas analizy obrazu: a) oryginalnego o słabym oświetleniu $j = 73$, b) tego samego obrazu po zastosowaniu automatycznej korekty jasności $\gamma = 0.808$ 83
Rys. 5.7 Graficzna reprezentacja szkieletu kinematycznego wygenerowanego przy użyciu modelu BODY_25 85
Rys. 5.8 Widok sytuacyjny przebiegu testów detektora OpenPose 86
Rys. 5.9 Pewność detektora OpenPose w funkcji odległości osoby od kamery 87
Rys. 5.10 Wizualna ilustracja działania detektora OpenPose w funkcji odległości osoby od kamery 88
Rys. 5.11 Przebieg procesu uczenia modelu YOLO v4 tiny 90
Rys. 5.12 Charakterystyka wyników ewaluacji zestawów wag przeprowadzona na zbiorze walidacyjnym 91
Rys. 5.13 Schemat blokowy algorytmu MOT, gdzie: *EOF* – znacznik końca wektora, i – licznik pętli iteracyjnych, W_{BB} – wektor nowych detekcji BB, W_s – wektor śledzonych obiektów, W_1 – wartość progowa maksymalnej liczby okluzji obiektu 93
Rys. 5.14 Wizualizacja rezultatów pracy algorytmu śledzenia MOT 96
Rys. 5.15 Widok poglądowy symulowanej sceny, gdzie: C – punkt główny kamery, f – ogniskowa kamery, P_f – punkt przecięcia ogniskowej z płaszczyzną obrazowania, C_h – wysokość montażu kamery nad płaszczyzną podłoża, δ – kąt nutacji (nachylenia kamery do płaszczyzny podłoża), ψ – kąt obrotu własnego kamery (względem głównej osi optycznej), P_0 – punkt początkowy globalnego układu współrzędnych 97

Rys. 5.16 Wybór punktów referencyjnych P_1, P_2 oraz P_3 w widoku obrazu z kamery wirtualnej 99
Rys. 5.17 Symulacja realizacji pomiarów wartości d_1, d_2, d_3 oraz C_h na scenie 100
Rys. 5.18 Wykres zależności $\sigma_2 = f(C_h)$ dla zmiennych kątów [δ; ψ] 102
Rys. 5.19 Wyniki badań empirycznych algorytmu kalibracji sceny dla współrzędnych osi: a) X, b) Y 105
Rys. 5.20 Przykład zmian wysokości BB związany z chwilową okluzją dla analizy pojedynczego nagrania wideo: a) ujęcie nr. 83 (wzorcowe wysokości BB), b) ujęcie nr. 90 (korekta wysokości BB nieaktywna), c) ujęcie nr. 90 (korekta wysokości BB aktywna) 106
Rys. 5.21 Przebieg czasowy lokalizacji śledzonej osoby podczas częściowej okluzji 107
Rys. 6.1 Schemat blokowy algorytmu identyfikacji pojazdów, gdzie: D_p – liczba detekcji pojazdów, D_t – liczba detekcji tablic rejestracyjnych, R_1 – licznik ROI 112
Rys. 6.2 Wizualizacja etapów pracy algorytmu identyfikacji pojazdów 113
Rys. 6.3 Widok z kamery wirtualnej obejmujący przykładową scenę dotyczącą wstępnych testów algorytmu identyfikacji pojazdów 114
Rys. 6.4 Wizualizacja parametrów granicznych algorytmu identyfikacji pojazdów 115
Rys. 6.5 Rezultat pracy algorytmu identyfikacji pojazdów dla różnych typów tablic rejestracyjnych 115
Rys. 6.6 Testy empiryczne algorytmu identyfikacji pojazdów 117
Rys. 6.7 Schemat blokowy projektowanego algorytmu detekcji obecności zwierząt, gdzie: D_z – liczba wszystkich detekcji zwierząt 119
Rys. 6.8 Testy empiryczne algorytmu detekcji obecności zwierząt 120
Rys. 6.9 Schemat blokowy pierwszego etapu pracy algorytmu detekcji sabotażu kamery, gdzie: *sabotaż* – flaga detekcji sabotażu, j – aktualna średnia ważona wartość jasności obrazu, j_m – wzorcowa średnia ważona wartość jasności obrazu, $\Delta j = |j - j_m|$, W_2 – wartość progowa zmiany jasności obrazu, L_p – liczba pikseli niezerowych maski tła 122
Rys. 6.10 Przykład analizy sabotowanej sceny z zastosowaniem techniki odejmowania tła: a) oryginalny obraz z kamery, b) maska uzyskana w procesie odejmowania tła, c) maska tła po redukcji szumów 123
Rys. 6.11 Schemat blokowy drugiego etapu pracy algorytmu detekcji sabotażu kamery, gdzie: *sabotaż* – flaga detekcji sabotażu, D – liczba wszystkich śledzonych osób i pojazdów, L_1 – licznik kolejnych negatywnych detekcji sabotażu, L_p – liczba pikseli niezerowych maski tła, W_3 – wartość progowa liczby pikseli niezerowych, W_4 – wartość progowa liczby negatywnych detekcji sabotażu 124

Rys. 6.12 Testy empiryczne algorytmu detekcji sabotażu kamery, typ detekcji: zasłonięcie obiektywu 126
Rys. 6.13 Testy empiryczne algorytmu detekcji sabotażu kamery, typ detekcji: obrót obiektywu 127
Rys. 6.14 Testy empiryczne algorytmu detekcji sabotażu kamery, typ detekcji: oślepienie obiektywu 128
Rys. 6.15 Schemat blokowy algorytmu detekcji nielegalnego parkowania, gdzie: D_p – liczba wszystkich detekcji pojazdów, L_2 – licznik pozytywnych detekcji pojazdu w strefie ROI, R_2 – licznik stref ROI, W_5 – wartość progowa *IoU*, W_6 – wartość progowa liczby pozytywnych detekcji pojazdu w strefie ROI 130
Rys. 6.16 Testy empiryczne algorytmu detekcji nielegalnego parkowania 132
Rys. 6.17 Schemat blokowy algorytmu analizy wejścia (wyjścia), gdzie: D_o – liczba wszystkich śledzonych osób, F_w – flaga wejścia (wyjścia) osoby, L_3 – licznik pozytywnych detekcji wejścia (wyjścia) osoby ze strefy, R_3 – licznik granic wirtualnych 134
Rys. 6.18 Testy empiryczne algorytmu analizy wejścia (wyjścia) ze strefy 136
Rys. 6.19 Schemat blokowy algorytmu detekcji naruszenia wirtualnej granicy, gdzie: D_o – liczba wszystkich śledzonych osób, F_g – flaga przekroczenia wirtualnej linii przez osobę, R_4 – licznik granic wirtualnych 137
Rys. 6.20 Testy empiryczne algorytmu detekcji naruszenia wirtualnej granicy 139
Rys. 6.21 Schemat blokowy zaprojektowanego algorytmu detekcji przeludnienia, gdzie: D_o – liczba wszystkich śledzonych osób, W_7 – wartość progowa maksymalnej liczby osób 141
Rys. 6.22 Testy empiryczne algorytmu detekcji przeludnienia 142
Rys. 6.23 Schemat blokowy zaprojektowanego algorytmu detekcji biegu oraz analizy aktywności tłumu, gdzie: D_o – liczba wszystkich śledzonych osób, F_a – flaga aktywności funkcji analizy tłumu, F_b – flaga aktywności funkcji detekcji biegu, L_4 – licznik biegnących osób, W_8 – wartość progowa maksymalnej szybkości przemieszczania się osoby, W_9 – wartość progowa maksymalnej dopuszczalnej liczby biegnących osób, V – szybkość przemieszczania się osoby 144
Rys. 6.24 Testy empiryczne algorytmu detekcji biegu 146
Rys. 6.25 Testy empiryczne algorytmu analizy tłumu (panika/rozbieganie się tłumu) 147
Rys. 6.26 Schemat blokowy zaprojektowanego algorytmu detekcji włóczęgostwa, gdzie: O – obszar przemieszczania się osoby na obrazie, O' – obszar przemieszczania się osoby w przestrzeni rzeczywistej, D_o – liczba wszystkich śledzonych osób, H – liczba elementów wektora historii śledzonej osoby, L_5 – licznik pozytywnych detekcji włóczęgostwa, W_{10} – wartość progowa obszaru przemieszczania się osoby, W_{11} – wartość progowa

maksymalnej szybkości przemieszczania się osoby, W_{12} – wartość progowa liczby elementów wektora historii śledzonej osoby, W_{13} – wartość progowa pozytywnych detekcji włóczęgostwa, V – szybkość przemieszczania się osoby 149
Rys. 6.27 Testy empiryczne algorytmu detekcji włóczęgostwa 151
Rys. 6.28 Schemat blokowy zaprojektowanego algorytmu detekcji naruszenia wirtualnej strefy, gdzie: D_o – liczba wszystkich śledzonych osób, R_5 – licznik stref wirtualnych 152
Rys. 6.29 Testy empiryczne algorytmu detekcji naruszenia wirtualnej strefy 154
Rys. 6.30 Schemat blokowy zaprojektowanego algorytmu detekcji pożaru, gdzie: D_f – liczba detekcji obiektów klasy „*ogień*” 155
Rys. 6.31 Testy empiryczne algorytmu detekcji pożaru 157
Rys. 6.32 Porównanie modeli szkieletowych sylwetek osób leżących w różnych orientacjach względem kamery 159
Rys. 6.33 Schemat blokowy zaprojektowanego algorytmu detekcji upadku, gdzie: BB_h – wysokość BB, BB_w – szerokość BB, D_o – liczba wszystkich śledzonych osób, K_p – liczba punktorów szkieletu śledzonej osoby, k_r – długość kręgosłupa, pu – długość prawego uda, pp – długość prawego podudzia, lu – długość lewego uda, lp – długość lewego podudzia, sn – średnia długość nóg, L_6 – licznik pozytywnych detekcji upadku, W_{14} – wartość progowa stosunku długości kręgosłupa do długości nóg, W_{15} – wartość progowa liczby pozytywnych detekcji upadku osoby, W_{16} – wartość progowa szybkości przemieszczania się osoby, V – szybkość przemieszczania się osoby 160
Rys. 6.34 Porównanie modelu szkieletowego osoby stojącej i leżącej zorientowanej przodem do kamery 161
Rys. 6.35 Testy empiryczne algorytmu detekcji upadku 163
Rys. 6.36 Schemat blokowy zaprojektowanego algorytmu detekcji kradzieży, gdzie: L_7 – licznik pozytywnych detekcji kradzieży, L_k – liczba pikseli niezerowych maski tła obiektu zainteresowania, R_6 – licznik stref wirtualnych zaznaczonych obiektów zainteresowania, W_{17} – wartość progowa liczby pikseli niezerowych, W_{18} – wartość progowa liczby pozytywnych detekcji kradzieży 166
Rys. 6.37 Testy empiryczne algorytmu detekcji kradzieży 167
Rys. 6.38 Schemat blokowy zaprojektowanego algorytmu detekcji przemocy, gdzie: D_o – liczba wszystkich śledzonych osób, $dist_1$ – odległość analizowanej osoby od kamery, $dist_2$ – odległość pomiędzy osobami, K_p – liczba punktorów szkieletu śledzonej osoby, L_8 – licznik pozytywnych detekcji przemocy, P_V – średnia szybkość zmian zestawów kątów, W_{19} – wartość progowa odległości osoby od kamery, W_{20} – wartość progowa szybkości zmiany kątów, W_{21} – wartość progowa dystansu pomiędzy osobami,

W_{22} – wartość progowa liczby pozytywnych detekcji przemocy, T_θ – wektor miar kątów, V_θ – wektor szybkości zmian kątów 170
Rys. 6.39 Graficzna reprezentacja wybranych kątów modelu szkieletowego w algorytmie detekcji przemocy 172
Rys. 6.40 Testy empiryczne algorytmu detekcji przemocy 174
Rys. 6.41 Chronologia wielowątkowości systemu z uwzględnieniem czasów przetwarzania 176
Rys. 6.42 Testy empiryczne systemu w konfiguracji wielozadaniowej 178
Rys. 6.43 Wartości wskaźnika F_1 dla poszczególnych algorytmów detekcji 181
Rys. 6.44 Porównanie skuteczności opracowanego systemu IVSS z systemami referencyjnymi 183

Spis tabel

Tab. 2.1 Porównanie wybranych bibliotek wstępnego przetwarzania danych wizyjnych 6
Tab. 2.2 Porównanie wybranych detektorów jednostopniowych opartych na DL 19
Tab. 2.3 Porównanie wybranych bibliotek MPPE 28
Tab. 3.1 Chronologiczne zestawienie porównawcze referencyjnych systemów IVSS 49
Tab. 3.2 Wykaz detekcji zdarzeń alarmowych realizowanych przez analizowane systemy referencyjne IVSS 59
Tab. 4.1 Charakterystyka ilościowa opracowanej bazy nagrań wideo 71
Tab. 5.1 Wykaz klas obiektów rozpoznawanych przez wybrane detektory 84
Tab. 5.2 Charakterystyka ilościowa bazy danych uczących modelu *yolov4-tiny* 89
Tab. 5.3 Opis metod PSM detektora Tesseract OCR [138] 92
Tab. 5.4 Wyniki badań symulacyjnych opracowanego algorytmu kalibracji sceny 103
Tab. 6.1 Baza danych numerów tablic rejestracyjnych użyta w testach algorytmu identyfikacji pojazdów 114
Tab. 6.2 Zestawienie wyników badań empirycznych algorytmu identyfikacji pojazdów 116
Tab. 6.3 Zestawienie wyników badań empirycznych algorytmu identyfikacji obecności zwierząt 119
Tab. 6.4 Zestawienie wyników badań empirycznych algorytmu detekcji sabotażu kamery . 125
Tab. 6.5 Zestawienie wyników badań empirycznych algorytmu detekcji nielegalnego parkowania 131
Tab. 6.6 Zestawienie wyników badań empirycznych algorytmu analizy wejścia (wyjścia) ze strefy 134
Tab. 6.7 Zestawienie wyników badań empirycznych algorytmu detekcji naruszenia wirtualnej granicy 138
Tab. 6.8 Zestawienie wyników badań empirycznych algorytmu detekcji przeludnienia 141
Tab. 6.9 Zestawienie wyników badań empirycznych algorytmu detekcji biegu oraz analizy aktywności tłumu 145
Tab. 6.10 Zestawienie wyników badań empirycznych algorytmu detekcji włóczęgostwa 150
Tab. 6.11 Zestawienie wyników badań empirycznych algorytmu detekcji naruszenia wirtualnej strefy 153
Tab. 6.12 Zestawienie wyników badań empirycznych algorytmu detekcji pożaru 155
Tab. 6.13 Zestawienie wyników badań empirycznych algorytmu detekcji upadku 162
Tab. 6.14 Zestawienie wyników badań empirycznych algorytmu detekcji kradzieży 166
Tab. 6.15 Wykaz zestawów punktorów szkieletu, analizowanych w celu detekcji przemocy 170

Tab. 6.16 Wykaz zestawów kątów części ciała opracowanych w celu detekcji przemocy ... 171
Tab. 6.17 Zestawienie wyników badań empirycznych algorytmu detekcji przemocy 173
Tab. 6.18 Podsumowanie wyników badań wszystkich opracowanych algorytmów 180
Tab. 6.19 Zestawienie wartości wyznaczników oceny jakości pracy referencyjnych systemów IVSS 182

Literatura

[1] Rani, J.U. and Raviraj, P. (2023) Real-Time Human Detection for Intelligent Video Surveillance: An Empirical Research and In-depth Review of its Applications. *An Empirical Research and In-depth Review of its Applications. SN Computer Science*, 4, s. 1–11.

[2] Rajasekhar, P. (2022) AI in Video Analytics and Security. *BARC newsletter*, s. 1–4.

[3] Feng, X., Jiang, Y., Yang, X., Du, M. and Li, X. (2019) Computer vision algorithms and hardware implementations: A survey. *Integration*, s. 309–320.

[4] Sekar R (2023) THE HISTORY OF COMPUTER VISION. *IMAGE PROCESSING APPLICATIONS*, s. 4–6.

[5] Martins, P., Costelha, H., Bento, L. and Neves, C. (2020) Monocular camera calibration for autonomous driving—a comparative study. *In 2020 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), IEEE*, s. 306–311.

[6] Shilkrot, R. and Escrivá, D. (2018) Mastering OpenCV 4. A comprehensive guide to building computer vision and image processing applications with C++ – Third Edition. *Helion*, s. 48–50.

[7] Chaki, J. and Dey, N. (2018) A beginner's guide to image preprocessing techniques. *CRC Press*, s. 1.

[8] Sabanin, B. (2008) Intel® Integrated Performance Primitives. s. 1–4.

[9] VXL online library developers guide (2024). https://vxl.github.io/doc/release/books/core/book.html#SEC_Top (Data dostępu: 28.11.2024).

[10] Kraiss, K.-F., Alvarado, P. and Dörfler, P. (2004) LTI Image Processing Library Developer's Guide. *Lehrstuhl f˙ur Technische Informatik*, s. 1.

[11] Vedaldi, A. and Fulkerson, B. (2010) VLFeat: An open and portable library of computer vision algorithms. *In Proceedings of the 18th ACM international conference on Multimedia*, s. 1469–1472.

[12] Tschumperlé, D. (2012) The cimg library. *The cimg library. In IPOL 2012 Meeting on Image Processing Libraries*, s. 1–4.

[13] Bradski, G. and Kaehler, A. (2008) Learning OpenCV: Computer vision with the OpenCV library. s. 103–384.

[14] Singh, H. (2019) Basics of Python and Scikit Image. *In: Practical Machine Learning and Image Processing. Apress, Berkeley, CA*, s. 40–41.

[15] Parekh, R. (2021) Fundamentals of image, audio, and video processing using Matlab®: with applications to pattern recognition. *CRC Press*, s. 9–13.

[16] Elharrouss, O., Almaadeed, N. and Al-Maadeed, S. (2021) A review of video surveillance systems. *J Vis Commun Image Represent*, 77, s. 1–19.

[17] Official TP-Link Tapo C500 specifications (2024). https://www.tp-link.com/pl/home-networking/cloud-camera/tapo-c500/#specifications (Data dostępu: 28.11.2024).

[18] Specyfikacje techniczne statywu z amortyzacją powietrzną W809 (2024). https://fotoamigo.pl/Statyw-oswietleniowy-air-118-440cm-z-amortyzacja-powietrzna-W809 (Data dostępu: 28.11.2024).

[19] Kaur J and Singh W (2022) Tools, techniques, datasets and application areas for object detection in an image: a review. *Multimed Tools Appl*, 81, s. 38297–38351.

[20] Zou Z, Chen K, Shi Z, Guo Y and Ye J (2023) Object detection in 20 years: A survey. *Proceedings of the IEEE*, s. 1–22.

[21] Viola, P. and Jones M (2001) Rapid object detection using a boosted cascade of simple features. *In Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, s. 1–9.

[22] Viola, P. and Jones, M.J. (2004) Robust Real-Time Face Detection. *Int J Comput Vis*, 57, s. 137–154.

[23] Dalal, N. and Triggs, B. (2005) Histograms of oriented gradients for human detection. *In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, s. 886–893.

[24] Felzenszwalb, P. and McAllester, D. (2008) A discriminatively trained, multiscale, deformable part model. *In 2008 IEEE conference on computer vision and pattern recognition*, s. 1–8.

[25] Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 580–587.

[26] Girshick, R. (2015) Fast R-CNN. *In Proceedings of the IEEE international conference on computer vision*, s. 1440–1448.

[27] Ren, S., He, K., Girshick, R. and Sun, J. (2015) Faster R-CNN: Towards real-time object detection with region proposal networks. *Adv Neural Inf Process Syst*, 28, s. 1–9.

[28] He, K., Zhang, X., Ren, S. and Sun, J. (2015) Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans Pattern Anal Mach Intell*, 37, s. 1904–1916.

[29] Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017) Feature pyramid networks for object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 2117–2125.

[30] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016) You only look once: Unified, real-time object detection. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 779–788.

[31] Redmon, J. and Farhadi, A. (2017) YOLO9000: Better, faster, stronger. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 7263–7271.

[32] Redmon, J. and Farhadi, A. (2018) YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, s. 1–6.

[33] Bochkovskiy, A., Wang, C. and Liao, H. (2020) Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, s. 1–17.

[34] Jocher, G. (2024) YOLOv5 by Ultralytics. https://github.com/ultralytics/yolov5 (Data dostępu: 28.11.2024).

[35] Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Linyuan, Z., Xiaoming, X., Xiangxiang, C., Xiaoming, W. and Xiaolin, W. (2022) YOLOv6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, s. 1–17.

[36] Wang, C., Bochkovskiy, A. and Liao, H. (2023) YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, s. 7464–7475.

[37] YOLOv8 official documentation (2024). https://docs.ultralytics.com/ (Data dostępu: 28.11.2024).

[38] Terven, J. and Cordova-Esparza, D. (2023) A Comprehensive Review of YOLO: From YOLOv1 and Beyond. *arXiv preprint arXiv:2304.00501*, s. 1–34.

[39] Terven, J., Córdova-Esparza, D. and Romero-González, J. (2023) A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. *Mach Learn Knowl Extr*, 5, s. 1680–1716.

[40] Hussain, M. (2023) YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines*, 11, s. 1–25.

[41] Redmon, J. (2024) Darknet: Open source neural networks in C. https://pjreddie.com/darknet/ (Data dostępu: 28.11.2024).

[42] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C. (2016) SSD: Single shot multibox detector. *In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings*, 1, s. 21–37.

[43] Lin, T., Goyal, P., Girshick, R., He, K. and Dollár, P. (2017) Focal loss for dense object detection. *In Proceedings of the IEEE international conference on computer vision*, s. 2980–2988.

[44] He, K., Zhang, X., Ren, S. and Sun, J. (2016) Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 770–778.

[45] Law, H. and Deng, J. (2018) Cornernet: Detecting objects as paired keypoints. *In Proceedings of the European conference on computer vision (ECCV)*, s. 734–750.

[46] Zhou, X., Wang, D. and Krähenbühl, P. (2019) Objects as Points. *arXiv preprint arXiv:1904.07850*, s. 1–12.

[47] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. and Zagoruyko, S. (2020) End-to-end object detection with transformers. *In European conference on computer vision*, s. 213–229.

[48] Ridnik, T., Ben-Baruch, E., Noy, A. and Zelnik-Manor, L. (2021) Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*, s. 1–20.

[49] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L. (2014) Microsoft COCO: Common objects in context. *In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings*, 5, s. 740–755.

[50] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T. and Ferrari, V. (2020) The Open Images Dataset V4: Unified Image Classification, Object Detection, and Visual Relationship Detection at Scale. *Int J Comput Vis*, 128, s. 1956–1981.

[51] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B. and Vijayanarasimhan, S. (2016) YouTube-8M: A Large-Scale Video Classification Benchmark. *arXiv preprint*, s. 1–2.

[52] Dwyer, B., Nelson, J. and Hansen, T. (2024) Roboflow. https://roboflow.com (Data dostępu: 28.11.2024).

[53] Kaggle official (2024). https://www.kaggle.com/ (Data dostępu: 28.11.2024).

[54] Wang, X., He, Z., Wang, K., Wang, Y., Zou, L. and Wu, Z. (2023) A survey of text detection and recognition algorithms based on deep learning technology. *Neurocomputing*, 556, s. 1–23.

[55] Khan, M., Ilyas, M., Khan, I. and Alshomrani, S. (2023) A review of license plate recognition methods employing neural networks. *IEEE Access*, 11, s. 73613–73646.

[56] Shilkrot, R. and Escrivá, D. (2018) Mastering OpenCV 4. A comprehensive guide to building computer vision and image processing applications with C++ – Third Edition. *Helion*, s. 79–113.

[57] Poudel, U., Regmi, A.M., Stamenkovic, Z., Stamenkovic, Z. and Raja, S.P. (2023) Applicability of OCR Engines for Text Recognition in Vehicle Number Plates, Receipts and Handwriting. *Journal of Circuits, Systems and Computers*, 32, s. 1–51.

[58] Bobek, F. (2022) OCR engines comparison. *Masaryk University Faculty of Iinformatics Bachelor's thesis*, s. 18–20.

[59] Batra, P., Phalnikar, N., Kurmi, D., Tembhurne, J., Sahare, P. and Diwan, T. (2024) OCR-MRD: performance analysis of different optical character recognition engines for medical report digitization. *International Journal of Information Technology (Singapore)*, 16, s. 447–455.

[60] Charlton, A. (2023) OCR Report. *The University of Edinburgh*, s. 13.

[61] Tesseract OCR official documentation (2024). https://tesseract-ocr.github.io/tessdoc/ (Data dostępu: 28.11.2024).

[62] Easy OCR official documentation (2024). https://www.jaided.ai/easyocr/documentation/ (Data dostępu: 28.11.2024).

[63] Tesseract OCR: Improving the quality of the output (2024). https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html (Data dostępu: 28.11.2024).

[64] Chen, H., Feng, R., Wu, S., Xu, H., Zhou, F. and Liu, Z. (2023) 2D Human pose estimation: a survey. *Multimed Syst*, 29, s. 3115–3138.

[65] Dubey, S. and Dixit, M. (2023) A comprehensive survey on human pose estimation approaches. *Multimed Syst*, 29, s. 167–195.

[66] Onishi, K., Takiguchi, T. and Ariki, y (2008) 3D human posture estimation using the HOG features from monocular image. *In 2008 19th International Conference on Pattern Recognition, IEEE*, s. 1–4.

[67] Bo Wu and Ram Nevatia (2005) Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. *Tenth IEEE International Conference on Computer Vision, ICCV 2005*, s. 90–97.

[68] Andriluka, M., Roth, S. and Schiele, B. (2009) Pictorial structures revisited: People detection and articulated pose estimation. *In 2009 IEEE conference on computer vision and pattern recognition, IEEE.*, s. 1014–1021.

[69] Dang, Q., Yin, J., Wang, B. and Zheng, W. (2019) Deep Learning Based 2D Human Pose Estimation: A Survey. *Tsinghua Sci Technol*, 24, s. 663–676.

[70] Zheng, C., Wu, W., Chen, C., Yang, T., Zhu, S., Shen, J., Kehtarnavaz, N. and Shah, M. (2023) Deep Learning-based Human Pose Estimation: A Survey. *ACM Comput Surv*, 56, s. 1–37.

[71] Cao, Z., Simon, T., Wei, S.E. and Sheikh, Y. (2017) Realtime multi-person 2d pose estimation using part affinity fields. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 7291–7299.

[72] CMU-Perceptual-Computing-Lab (2024) OpenPose official repository. https://github.com/CMU-Perceptual-Computing-Lab/openpose (Data dostępu: 28.11.2024).

[73] Insafutdinov, E., Pishchulin, L., Andres, B., Andriluka, M. and Schiele, B. (2016) Deepercut: A deeper, stronger, and faster multi-person pose estimation model. *In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands*, 6, s. 34–50.

[74] Pishchulin, L., Insafutdinov, E., Tang, S., Andres, B., Andriluka, M., Gehler, P. and Schiele, B. (2016) Deepcut: Joint subset partition and labeling for multi person pose estimation. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 4929–4937.

[75] Insafutdinov, E. (2024) DeeperCut official repository. https://github.com/eldar/deepcut/tree/master (Data dostępu: 28.11.2024).

[76] ultralytics (2024) YOLO v8 official repository. https://github.com/ultralytics/ultralytics (Data dostępu: 28.11.2024).

[77] Fang, H.-S., Li, J., Tang, H., Xu, C., Zhu, H., Xiu, Y., Li, Y.-L. and Lu, C. (2022) Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, s. 1–17.

[78] Tan, M., Pang, R. and Le, Q. (2020) Efficientdet: Scalable and efficient object detection. *In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, s. 10781–10790.

[79] Machine Vision and Intelligence Group (2024) AlphaPose official repository. https://github.com/MVIG-SJTU/AlphaPose (Data dostępu: 28.11.2024).

[80] Andriluka, M., Pishchulin, L., Gehler, P. and Schiele, B. (2014) 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. *In Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, s. 3686–3693.

[81] OpenPose official documentation (2024). https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_00_index.html (Data dostępu: 28.11.2024).

[82] Soleimanitaleb, Z., Keyvanrad, M.A. and Jafari, A. (2019) Object tracking methods: a review. *In 2019 9th International Conference on Computer and Knowledge Engineering (ICCKE), IEEE*, s. 282–288.

[83] Almohaimeed, N. and Prince, M. (2019) A comparative study of different object tracking methods in a video. *International Journal of Computer Applications, 975, 8887.*, 181, s. 1–8.

[84] Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W. and Kim, T. (2022) Multiple object tracking: A literature review. *Artif Intell*, s. 1–49.

[85] Wang, C. and Yan, J. (2023) A Comprehensive Survey of RGB-Based and Skeleton-Based Human Action Recognition. *IEEE Access*, 11, s. 53880–53898.

[86] Arshad, M., Bilal, M. and Gani, A. (2022) Human activity recognition: Review, taxonomy and open challenges. *Sensors*, 22, s. 1–33.

[87] Saleem, G., Bajwa, U.I. and Raza, R.H. (2023) Toward human activity recognition: a survey. *Neural Comput Appl*, 35, s. 4145–4182.

[88] Beddiar, D.R., Nini, B., Sabokrou, M. and Hadid, A. (2020) Vision-based human activity recognition: a survey. *Multimed Tools Appl*, 79, s. 30509–30555.

[89] Nikolić, G.S., Dimitrijević, B.R., Nikolić, T.R. and Stojcev, M.K. (2022) A survey of three types of processing units: CPU, GPU and TPU. *In 2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), IEEE.*, s. 1–6.

[90] Hajirassouliha, A., Taberner, A.J., Nash, M.P. and Nielsen, P.M.F. (2018) Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms. *Signal Process Image Commun*, s. 101–119.

[91] Machupalli, R., Hossain, M. and Mandal, M. (2022) Review of ASIC accelerators for deep neural network. *Microprocessors and Microsystems*, 89, s. 1–2.

[92] Schlosser, T., Friedrich, M., Meyer, T. and Kowerko, D. (2023) A Consolidated Overview of Evaluation and Performance Metrics for Machine Learning and Computer Vision. s. 1–42.

[93] Myagmar-Ochir, Y. and Kim, W. (2023) A survey of video surveillance systems in smart city. *Electronics (Basel)*, 12, s. 1–34.

[94] Karbalaie, A., Abtahi, F. and Sjöström, M. (2022) Event detection in surveillance videos: a review. *Multimed Tools Appl*, 81, s. 35463–35501.

[95] Şengönül, E., Samet, R., Abu Al-Haija, Q., Alqahtani, A., Alturki, B. and Alsulami, A.A. (2023) An Analysis of Artificial Intelligence Techniques in Surveillance Video Anomaly Detection: A Comprehensive Survey. *Applied Sciences*, 13, s. 1–31.

[96] Bazhenov, N., Rybin, E. and Korzun, D. (2022) An Event-Driven Approach to the Recognition Problem in Video Surveillance System Development. *In 2022 32nd Conference of Open Innovations Association (FRUCT), IEEE*, s. 65–74.

[97] JL. Bruyelle, F. Cabestaing, D. Aubert, S. Bouchafa, S.A. Velastin, M.A. Vicencio-Silva and M. Wherett (1997) Project cromatica. *Image Analysis and Processing: 9th International Conference, ICIAP'97 Florence, Springer*, s. 757–764.

[98] L.M. Fuentes and S.A. Velastin (2004) Tracking-based event detection for CCTV systems. *Pattern analysis and applications, Springer*, 7, s. 356–364.

[99] S. A. Velastin, B. A. Boghossian, B. Ping Lai Lo, J. Sun and M. A. Vicencio-Silva (2005) PRISMATICA: toward ambient intelligence in public transport environments. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 35, s. 164–182.

[100] K. Schwerdt, D. Maman, P. Bernas and E. Paul (2005) Target segmentation and event detection at video-rate: the eagle project. *IEEE Conference on Advanced Video and Signal Based Surveillance, IEEE*, s. 183–188.

[101] R. Feris, A. Hampapur, Y. Zhai, R. Bobbitt, L. Brown, D. A. Vaquero, Y. Tian, H. Liu and M. Sun (2009) Case study: IBM smart surveillance system. *Intelligent Video Surveillance: Systems and Technology, by Taylor & Francis Group, LLC*, s. 47–76.

[102] A. Senior, A. Hampapur, Y. Tian, L. Brown, S. Pankanti and R. Bolle (2006) Appearance models for occlusion handling. *Image Vis Comput*, 24, s. 1233–1243.

[103] M. Elhamod and M. D. Levine (2012) Real-time semantics-based detection of suspicious activities in public spaces. *In 2012 Ninth Conference on Computer and Robot Vision, IEEE*, s. 268–275.

[104] M. K. Lim, S. Tang and C.S. Chan (2014) iSurveillance: Intelligent framework for multiple events detection in surveillance videos. *Expert Syst Appl*, 41, s. 4704–4715.

[105] Tan, C., Loy, C., Lai, W. and Lim, C. (2008) Robust modular ARTMAP for multi-class shape recognition. *In 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE*, s. 2405–2412.

[106] Lee, S. and Nevatia, R. (2014) Hierarchical abnormal event detection by real time and semi-real time multi-tasking video surveillance system. *Mach Vis Appl*, 25, s. 133–143.

[107] Huang, C. and Nevatia, R. (2010) High performance object detection by collaborative learning of joint ranking of granules features. *In 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE*, s. 41–48.

[108] Tu, P., Rittscher, J. and Kelliher, T. (2003) Site calibration for large indoor scenes. *In Proceedings of the IEEE Conference on Advanced Video and Signal Based Surveillance, IEEE*, s. 358–363.

[109] R. Arroyo, J. J. Yebes, L. M. Bergasa, I. G. Daza and J. Almazán (2015) Expert video-surveillance system for real-time detection of suspicious behaviors in shopping malls. *Expert Syst Appl*, 42, s. 7991–8005.

[110] Kardas, K. and Cicekli, N.K. (2017) SVAS: surveillance video analysis system. *Expert Syst Appl*, s. 343–361.

[111] K. Kwang-Eun and S. Kwee-Bo (2018) Deep convolutional framework for abnormal behavior detection in a smart surveillance system. *Eng Appl Artif Intell*, s. 226–234.

[112] Thyagarajmurthy, A., Ninad, M.G., Rakesh, B.G., Niranjan, S. and Manvi, B. (2019) Anomaly Detection in Surveillance Video Using Pose Estimation. *In Emerging Research in Electronics, Computer Science and Technology: Proceedings of International Conference, ICERECT 2018*, s. 753–766.

[113] L. Caruccio, G. Polese, G. Tortora and D. Iannone (2019) EDCAR: A knowledge representation framework to enhance automatic video surveillance. *Expert Syst Appl*, s. 190–207.

[114] Kalal, Z., Mikolajczyk, K. and Matas, J. (2011) Tracking-learning-detection. *IEEE Trans Pattern Anal Mach Intell*, 34, s. 1409–1422.

[115] Cheoi, K. (2020) Temporal saliency-based suspicious behavior pattern detection. *Applied Sciences*, 10, s. 1–18.

[116] Wan, B., Jiang, W., Fang, Y., Luo, Z. and Ding, G. (2021) Anomaly detection in video sequences: A benchmark and computational model. *IET Image Process*, 15, s. 3454–3465.

[117] Maqsood, R., Bajwa, U., Saleem, G., Raza, R. and Anwar, M. (2021) Anomaly recognition from surveillance videos using 3D convolution neural network. *Multimed Tools Appl*, 80, s. 18693–18716.

[118] Ullah, W., Ullah, A., Hussain, T., Khan, Z. and Baik, S. (2021) An efficient anomaly recognition framework using an attention residual LSTM in surveillance videos. *Sensors, 21(8), 2811.*, 21, s. 1–17.

[119] Vosta, S. and Yow, K. (2022) A cnn-rnn combined structure for real-world violence detection in surveillance cameras. *Applied Sciences*, 12, s. 1–15.

[120] Ye, L., Yan, S., Zhen, J., Han, T., Ferdinando, H., Seppänen, T. and Alasaarela, E. (2022) Physical Violence Detection Based on Distributed Surveillance Cameras. *Mobile Networks and Applications*, 27, s. 1688–1699.

[121] Mumtaz, A., Sargano, A.B. and Habib, Z. (2023) Robust learning for real-world anomalies in surveillance videos. *Multimed Tools Appl*, 82, s. 20303–20322.

[122] Ahmed, W. and Yousaf, M.H. (2023) A Deep Autoencoder-Based Approach for Suspicious Action Recognition in Surveillance Videos. *Arab J Sci Eng*, s. 1–16.

[123] Ali, M., Goyal, L., Sharma, C. and Kumar, S. (2024) Edge-Computing-Enabled Abnormal Activity Recognition for Visual Surveillance. *Electronics (Basel)*, 13, s. 1–17.

[124] Vidya, M.Q.M. and Selvakumar, S. (2024) An effective framework of human abnormal behaviour recognition and tracking using multiscale dilated assisted residual attention network. *Expert Syst Appl*, 247, s. 1–25.

[125] Mumtaz, A., Sargano, A.B. and Habib, Z. (2024) AnomalyNet: a spatiotemporal motion-aware CNN approach for detecting anomalies in real-world autonomous surveillance. *Vis Comput*, s. 1–22.

[126] Sultani, W., Chen, C. and Shah, M. (2018) Real-world anomaly detection in surveillance videos. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, s. 6479–6488.

[127] HP Z620 Workstation Specifications (2024). https://support.hp.com/us-en/document/c03270936 (Data dostępu: 28.11.2024).

[128] Official Blender website (2024). https://www.blender.org/ (Data dostępu: 28.11.2024).

[129] Official Mixamo website (2024). https://www.mixamo.com/ (Data dostępu: 28.11.2024).

[130] Nuber plates generator (2024). https://platesmania.com/pl/informer (Data dostępu: 28.11.2024).

[131] Official OpenCV calibration specifications (2024). https://docs.opencv.org/4.x/d4/d94/tutorial_camera_calibration.html (Data dostępu: 28.11.2024).

[132] Ścieżka dostępu programu kalibracji kamery w bibliotece OpenCV (2024). *opencv/samples/cpp/tutorial_code/calib3d/camera_calibration/camera_calibration.cpp*.

[133] Official OpenCV contrast and brightness specifications (2024). https://docs.opencv.org/4.x/d3/dc1/tutorial_basic_linear_transform.html (Data dostępu: 28.11.2024).

[134] Parametr “model_pose” (2024). https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/include/openpose/flags.hpp (Data dostępu: 28.11.2024).

[135] Zalecenia twórców OpenPose dotyczące przyspieszenia czasu detekcji (2024). https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_06_maximizing_openpose_speed.html (Data dostępu: 28.11.2024).

[136] Opis punktorów szkieletu modelu BODY_25 biblioteki OpenPose (2024). https://cmu-perceptual-computing-

lab.github.io/openpose/web/html/doc/md_doc_02_output.html#pose-output-format-body_25 (Data dostępu: 28.11.2024).

[137] Official YOLO v4 repository (2024). https://github.com/AlexeyAB/darknet (Data dostępu: 28.11.2024).

[138] Tesseract OCR quality improvement (2024). https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html (Data dostępu: 28.11.2024).

[139] Nominalne rozmiary czujników optycznych (2024). https://commonlands.com/blogs/technical/cmos-sensor-size (Data dostępu: 28.11.2024).

[140] Farag, M.S., El Din, M.M., and El Shenbary, H.A. (2019) Parking entrance control using license plate detection and recognition. *Indonesian Journal of Electrical Engineering and Computer Science*, 15, s. 476–483.

[141] Dz.U. 2017 poz. 2355 (specyfikacja tablic rejestracyjnych) (2024). https://isap.sejm.gov.pl/isap.nsf/download.xsp/WDU20170002355/O/D20172355.pdf (Data dostępu: 28.11.2024).

[142] How to check if two given line segments intersect? (2024). https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/ (Data dostępu: 28.11.2024).

[143] Peng, J., Wang, C., Wan, F., Wu, Y., Wang, Y., Tai, Y. and Fu, Y. (2020) Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. *In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings*, 4, s. 145–161.

Streszczenie

Przedmiotem badań niniejszej rozprawy jest opracowanie metody syntezy różnych typów informacji wizyjnej 2D w celu stworzenia całościowego i skutecznego systemu nadzoru przestrzeni publicznych, zdolnego do automatycznej detekcji i klasyfikacji wybranych zdarzeń alarmowych w środowisku rzeczywistym, z wykorzystaniem sygnału z konwencjonalnych kamer monitoringu wizyjnego. W tym kontekście, zaprojektowany system integruje następujące typy informacji wizyjnej: modele szkieletowe (analiza pozycji i ruchu osób), ramki „*bounding box*" (lokalizacja obiektów pierwszoplanowych), tekst (identyfikacja znaków alfanumerycznych), jasność obrazu (automatyczna korekta ekspozycji), ekstrakcję tła (detekcja zmian tła sceny), detekcję krawędzi (segmentacja detali obrazu) oraz rzutowanie 3D (analiza przestrzenna sceny). Zaprojektowany system IVSS umożliwia detekcję piętnastu typów zdarzeń alarmowych, w tym: identyfikacja pojazdów (ANPR), sabotaż kamery, niepożądana obecność zwierząt, nielegalne parkowanie, analiza wejścia i wyjścia ze strefy, naruszenie wirtualnej granicy, przeludnienie, bieg, analiza aktywności tłumu, włóczęgostwo, naruszenie wirtualnej strefy, pożar, upadek, kradzież oraz przemoc. W pracy wykorzystano takie narzędzia programistyczne z zakresu widzenia komputerowego jak: biblioteka OpenCV (odczyt oraz edycja obrazów), detektor YOLO v4 (detekcja wybranych klas obiektów), OpenPose (analiza sylwetki człowieka), oraz Tesseract OCR (interpretacja tekstu na obrazie). Zaproponowano również szereg autorskich rozwiązań, takich jak modułowa architektura, bazująca na różnych typach informacji wizyjnej 2D, oryginalna konfiguracja detekcji zdarzeń alarmowych czy autorskie algorytmy automatycznej korekty jasności obrazu, uniwersalnej kalibracji sceny, jak również identyfikacji pojazdów (ANPR), detekcji sabotażu kamery oraz detekcji niepożądanej obecności zwierząt, które nie są dostępne w analizowanych systemach referencyjnych. W pracy dokonano rozbudowy istniejących rozwiązań zaimplementowanych w systemach referencyjnych. Realizacja tego zadania dotyczyła głównie dostosowania i rozwoju istniejących algorytmów detekcji zdarzeń alarmowych, poprzez zastosowanie analizy przestrzennej sceny, jak również implementację detektorów obiektów YOLO v4 oraz OpenPose. Osiągnięte wartości wskaźników oceny jakości pracy systemu potwierdzają skuteczność opracowanego rozwiązania. Uśrednione wartości tychże wskaźników, takie jak dokładność (87%), precyzja (96%), czułość (85%) oraz F_1 (90%), wskazują na zadawalającą skuteczność działania systemu. System charakteryzuje się zdolnością do pracy w trybie wielozadaniowym w czasie rzeczywistym, modułowością oraz możliwością rozbudowy o nowe funkcje. Wyniki badań empirycznych, przeprowadzonych z użyciem dedykowanej bazy testowej, potwierdzają realizację założonych celów oraz weryfikują postawione tezy badawcze.

Summary

The main subject of this dissertation is the development of a method for synthesizing various types of 2D visual information to create a comprehensive and effective system for public space surveillance, capable of automatic detection and classification of selected alarm events in real-world environments using signals from conventional video surveillance cameras. In this context, the designed Intelligent Video Surveillance System (IVSS) integrates the following types of visual information: skeletal models (for analyzing human posture and movement), bounding boxes (for locating primary objects), text (for alphanumeric character recognition), image brightness (for automatic exposure correction), background extraction (for detecting scene changes), edge detection (for segmenting image details), and 3D projection (for spatial scene analysis). The proposed IVSS is capable of detecting 15 types of alarm events, including vehicle identification (ANPR), camera tampering, unwanted animal presence, illegal parking, zone entry and exit analysis, virtual boundary trespassing, overcrowding, running, crowd activity analysis, loitering, virtual zone violation, fire, fall, theft of static objects, and violence detection. The system leverages such computer vision tools as OpenCV (image reading and editing), YOLO v4 (detection of selected object classes), OpenPose (human skeleton analysis), and Tesseract OCR (text interpretation). Additionally, custom algorithms have been implemented to advance and optimize the system for alarm event detection tasks. The developed system introduces several original solutions, including a modular architecture based on various types of 2D visual information, an original configuration for alarm event detection functions, and proprietary algorithms for automatic image brightness correction, universal scene calibration, vehicle identification (ANPR), camera tampering detection, and unwanted animal presence detection –features not available in the analyzed reference systems. The dissertation also expands on existing solutions implemented in reference systems. This effort primarily involves adapting and enhancing existing algorithms for alarm event detection through spatial scene analysis and the integration of YOLO v4 and OpenPose detectors. The calculated performance metrics reveal the effectiveness of the proposed solution. Averaged values such as accuracy (87%), precision (96%), recall (85%), and F_1 score (90%) demonstrate high detection efficiency. The system operates in a multitasking real-time mode. Moreover, it is modular, customizable, and easy to extend with additional functionalities. Empirical studies, conducted using a dedicated test database, confirm the achievement of the stated objectives and validate the research hypotheses.