



WYDZIAŁ ELEKTRONIKI,  
TELEKOMUNIKACJI  
I INFORMATYKI

Imię i nazwisko studenta: Kacper Plesiak

Nr albumu: 180324

Poziom kształcenia: Studia pierwszego stopnia

Forma studiów: stacjonarne

Kierunek studiów: Automatyka, cybernetyka i robotyka

Profil: Systemy decyzyjne i robotyka

## PROJEKT DYPLOMOWY INŻYNIERSKI

Tytuł projektu w języku polskim: System wizyjny do analizy meczów piłki nożnej

Tytuł projektu w języku angielskim: Vision system for the analysis of football matches

Opiekun pracy: dr inż. Mariusz Domżalski

## OŚWIADCZENIE dotyczące pracy dyplomowej zatytułowanej: **System wizyjny do analizy meczów piłki nożnej**

Imię i nazwisko studenta: Kacper Plesiak  
Data i miejsce urodzenia: 13.05.2000, Gdańsk  
Nr albumu: 180324

Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki

Kierunek: automatyka, cybernetyka i robotyka

Poziom kształcenia: pierwszy

Forma studiów: stacjonarne

Typ pracy: projekt dyplomowy inżynierski

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2019 r. poz. 1231, z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t.j. Dz. U. z 2020 r. poz. 85, z późn. zm.),<sup>1</sup> a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

30.11.2022, Kacper Plesiak

Data i podpis lub uwierzytelnienie w portalu uczelnianym Moja PG

*\*) Dokument został sporządzony w systemie teleinformatycznym, na podstawie §15 ust. 3b Rozporządzenia MNiSW z dnia 12 maja 2020 r. zmieniającego rozporządzenie w sprawie studiów (Dz.U. z 2020 r. poz. 853). Nie wymaga podpisu ani stempla.*

<sup>1</sup> Ustawa z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce:

Art. 312. ust. 3. W przypadku podejrzenia popełnienia przez studenta czynu, o którym mowa w art. 287 ust. 2 pkt 1–5, rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.

Art. 312. ust. 4. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 5, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o podejrzeniu popełnienia przestępstwa.

## **STRESZCZENIE**

W niniejszej pracy zaproponowany został system wizyjny, zbierający informacje z kamery na temat pozycji piłkarzy i piłki na boisku, które w kolejnym kroku są prezentowane graficznie, rzutując je na dwuwymiarową płaszczyznę boiska z widokiem z góry. Założeniami projektowymi jest działanie w środowisku gry FIFA 21, z obrazem z kamery bez cięć. Jednym z, poruszanych w ramach pracy, zagadnień jest problematyka detekcji obiektów na boisku. Metodyka wykrywania obiektów na obrazie dzieli się na wykorzystanie algorytmów deterministycznych oraz sztucznych sieci neuronowych. Każda z nich ma swoje wady i zalety, przy czym metodą wybraną w niniejszej pracy jest ścieżka algorytmów deterministycznych. Kolejnym omówionym zagadnieniem, związanym z problematyką detekcji obiektów, jest metodyka śledzenia obiektów na obrazie. Problem polega na dopasowaniu piłkarzy wykrytych w klatce bieżącej do, istniejących w pamięci programu, piłkarzy wykrytych w klatce poprzedniej przetwarzanego filmu. Dopasowywanie odbywa się na podstawie analizy pozycji wszystkich wykrytych piłkarzy, a także kolorów ich koszulek. Ostatnim zagadnieniem jest problematyka estymacji pozycji kamery w przestrzeni oraz transformacja obrazu z kamery obserwującej środowisko w perspektywie trójwymiarowej, na dwuwymiarową projekcję płaszczyzny boiska z widokiem z góry. Problem ten został rozwiązany poprzez detekcję punktów przecięć pomiędzy wykrytymi liniami boiska i wykorzystanie zmieniania się ich pozycji do estymacji ruchu kamery. Całość została przetestowana na materiale testowym [31], na podstawie którego powstał materiał wyjściowy [32] oraz analiza błędów przetwarzania. Osiągnięta skuteczność detekcji piłkarzy wyniosła około 78%, skuteczność wykrywania piłki - około 55%, skuteczność śledzenia obiektów - około 12%. Ponadto przeanalizowana została dokładność projekcji na dwuwymiarową płaszczyznę boiska. Wyniki wskazują, że rozwiązanie można byłoby ulepszać w kolejnych iteracjach implementacyjnych, aby zwiększyć niezawodność systemu.

**Słowa kluczowe:** detekcja piłkarzy, detekcja boiska piłkarskiego, estymacja pozycji kamery

**Dziedzina nauki i techniki, zgodnie z wymogami OECD:** nauki inżynieryjne i techniczne, robotyka i automatyka

## **ABSTRACT**

This paper proposes vision system that collects data from broadcast camera about the players and ball positions on the pitch, which in the next step are presented graphically, projecting them onto two dimensional plane of the pitch with view from the top. Project assumptions are working in FIFA 21 game environment with video without any cuts. One of the covered topics, in this paper, is the problem of detecting the players on the pitch. Methodology of objects detection on the frame can be divided into the use of deterministic algorithms and artificial neural networks. Each of them has its own pros and cons, but in this paper, the chosen way is the path of deterministic algorithm. Another discussed issue, related with objects detection problem, is methodology of footballers tracking. The problem is to match the players detected in the current frame to, existing in the program memory, detected players in the previous frame of processed video. Matching is done, based on the position analysis of all the players on the pitch and their shirt colors. Last discussed topic is camera position estimation problem and transforming frame from camera, which is observing environment in three dimensional perspective, onto two dimensional plane of the pitch projection with view from the top. This problem has been resolved, by detecting intersect points between detected lines of the pitch and using change of their position for camera motion estimation. Whole system has been tested on testing video [31], on the basis of which the output video [32] and error analysis have been created. Achieved efficiency of the players detection was 78%, efficiency of the ball detection - around 55%, efficiency of objects tracking - around 12%. Moreover, an analysis of mapping the position of the objects onto the plane of the pitch has been carried out. Results show that solution might be improved in the next implementation iterations to gain better reliability of the system.

**Keywords:** footballers detection, football pitch detection, camera motion estimation

## **SPIS TREŚCI**

1. WSTĘP I CEL PRACY .....	6
2. PRZEGŁĄD LITERATURY .....	7
3. IMPLEMENTACJA SYSTEMU.....	9
3.1. Architektura programu .....	9
3.2. Przetwarzanie wstępne .....	11
3.3. Detekcja piłki .....	13
3.4. Detekcja piłkarzy .....	15
3.5. Metodyka śledzenia piłkarzy .....	16
3.6. Detekcja linii boiska i punktów charakterystycznych .....	17
3.7. Estymacja ruchu kamery .....	20
3.8. Projekcja obiektów na dwuwymiarową płaszczyznę boiska.....	20
4. TESTY .....	23
4.1. Skuteczność wykrywania piłkarzy na obrazie.....	23
4.2. Skuteczność wykrywania piłki na obrazie .....	25
4.3. Skuteczność śledzenia piłkarzy .....	26
4.4. Dokładność projekcji.....	26
4.5. Testy wydajnościowe .....	28
5. PODSUMOWANIE .....	30
Wykaz literatury .....	32
Wykaz rysunków .....	34
Dodatek A: Instrukcja dla użytkownika .....	35
Dodatek B: Instrukcja dla programisty .....	38
Dodatek C: Opis dyplomu .....	41

## **1. WSTĘP I CEL PRACY**

Dziedzina wizji komputerowej jest rozwijana od wielu lat. Pozwala ona na automatyzację procesów przemysłowych oraz ekstrakcję danych na temat obserwowanego środowiska. Systemy te wykorzystywane są w przemyśle, autonomicznych samochodach, medycynie, czy robotycę. W ostatnich latach są również wykorzystywane w sporcie i analizie danych. W praktyce wspomagają one decyzje podejmowane przez sztab szkoleniowe. Piłka nożna jest sportem, w którym taki system mogłyby zostać wykorzystane w celu poprawy jakości gry całej drużyny, co mogłoby przyczynić się do osiągania lepszych rezultatów w środowisku krajowym jak i międzynarodowym. Kluczem jest dostęp do ścieżek, którymi poruszali się piłkarze w trakcie trwania całego meczu, co ułatwiłoby sztabowi trenerowskiemu na zwrócenie uwagi na elementy gry do poprawy oraz słabe i mocne strony zespołu. Niniejsza praca koncentruje się na rozwinięciu tej gałęzi wizji komputerowej, proponując autorski projekt systemu wizyjnego, zbierającego informacje z kamery na temat pozycji piłkarzy i piłki na boisku, które w następnym kroku są prezentowane graficznie, rzutując je na dwuwymiarową płaszczyznę boiska z widokiem z góry. Taki system pozwalałby na automatyzację procesu pozycjonowania piłkarzy z przestrzeni na dwuwymiarowej płaszczyźnie boiska, przygotowując w ten sposób materiał dostosowany do analizy pozycji poszczególnych sportowców w trakcie trwania całego meczu.

Pierwszym zagadnieniem poruszonym w niniejszej pracy jest problematyka detekcji obiektów na obrazie. Jest to podstawa całego systemu, który kolekcjonuje dane z obrazu rzeczywistego i umożliwia projekcję obiektów w płaskiej przestrzeni dwuwymiarowej. Algorytmy detekcji obiektów dzielą się na dwa, główne nurty: algorytmy deterministyczne oraz sztuczne sieci neuronowe. Niniejsza praca skupia się na wykorzystaniu metod deterministycznych w celu osiągnięcia pożądanej skuteczności, która determinuje wydajność dalszych modułów systemu. Drugim poruszonym zagadnieniem jest problematyka śledzenia obiektów na obrazie. Jest to drugi najważniejszy element systemu, który wzbogaca analizę o możliwość sprawdzenia ścieżek wytyczonych przez konkretnych piłkarzy. Odpowiednio skuteczny algorytm śledzenia zawodników umożliwia również automatyzację analizy statystyk poszczególnych piłkarzy jak i całych zespołów, co w praktyce, mogłoby wspomóc ocenę sztabu szkoleniowego danej drużyny, co do jakości danego piłkarza jak i jego przydatności w zespole. Ostatnim wyróżnionym w pracy zagadnieniem jest problematyka śledzenia pozycji kamery w przestrzeni. Rozwiążanie tego problemu jest kluczowe z punktu widzenia osiągnięcia rzeczywistych wyników pozycjonowania piłkarzy oraz piłki na dwuwymiarowej płaszczyźnie boiska. Odpowiedni algorytm determinuje precyzyje finalnej projekcji obiektów, co znacznie ułatwia późniejszą analizę meczu przez sztab szkoleniowy.

Celem projektu jest stworzenie systemu pozycjonującego piłkę oraz piłkarzy wykrytych na boisku na dwuwymiarową płaszczyznę boiska z widokiem z góry, przy założeniach braku rotacji kamery oraz braku ruchu w osi Z (góra, dół), a także braku cięć, czyli natychmiastowych przejść z jednego widoku kamery do drugiego. Dodatkowym założeniem jest brak wszelkich nakładek graficznych na obrazie takich jak logo stacji telewizyjnej, czy panel wyniku meczu. Z tego względu testy systemu przeprowadzone zostały w środowisku gry FIFA, która umożliwia spełnienie większości założeń, przy czym pomijalnie niewielka rotacja oraz ruch kamery w osi Z nadal występują.

## 2. PRZEGŁĄD LITERATURY

W dziedzinie wizji komputerowej, na przestrzeni ostatnich lat, powstało wiele różnych rozwiązań dotyczących problemu wykrywania piłkarzy na obrazie oraz ich śledzenia na podstawie obrazu z kamery telewizyjnej. Dobrym przykładem jest *ASPOGAMO* (*Automated Sports Game Analysis Models*, z ang. zautomatyzowane modele analizy gier sportowych) [1], w którym zaproponowany został system analizy danych na temat pozycji piłkarzy oraz piłki, jak i interakcji pomiędzy piłkarzami, a piłką. System ten wykorzystuje te dane m.in. do określania wystąpienia sytuacji boiskowych takich jak możliwość zdobycia bramki, czy podania piłki. Innym przykładem innowacyjnego systemu, który pojawił się w ostatnich latach jest *Kinexon* [2], który za pomocą technologii *motion capture*, z ang. przechwytywanie ruchu, tworzy lokalną projekcję pozycji piłkarzy na dwuwymiarową płaszczyznę boiska.

Poza pełnymi systemami analizy danych powstają liczne prace badające różne algorytmy wykrywania obiektów na obrazie. Dowodem na to jest [3], w którym autorzy stawiają swoje rozwiązanie naprzeciw rozwiązaniu [4] w problemie wykrywania piłki. Pokazują w ten sposób, że klasyczne metody przetwarzania obrazu mogą zapewnić lepsze rezultaty niż rozwiązanie wykorzystujące dynamiczny filtr Kalmana. Innym przykładem jest [5], gdzie autorzy porównują różne algorytmy wykrywania pozy 3D piłkarzy na obrazie z kamery telewizyjnej, pod względem złożoności obliczeniowej, w czterech różnych scenariuszach opisujących widok kamery (widok boiska, widok pozaboiiskowy, zbliżenie na piłkarzy, zbliżenie na trybuny).

Wiele algorytmów wymaga dużej mocy obliczeniowej, co sprawia, że często nie działają one w czasie rzeczywistym. Autorzy [6] zaproponowali swoją implementację algorytmu wykrywania piłkarzy działającego na procesorze graficznym GPU. Umożliwiło to przetwarzanie obrazu w sposób równoległy, dzięki czemu zostało osiągnięte przyspieszenie 4-11 krotne względem równoważnego algorytmu działającego na procesorze CPU.

Należy nakreślić, że wiele wyżej wymienionych systemów jest bardzo kosztownych. Cały system analizy wizyjnej, algorytmy o wysokiej wydajności, czy wykorzystanie układu GPU w celu zyskania cechy działania systemu w czasie rzeczywistym może stanowić zbyt duży koszt dla regionalnych drużyn półprofesjonalnych i amatorskich. W [7] autorzy nakreślają, że na tego typu systemy pozwolić sobie mogą jedynie największe ligi europejskie. Jednocześnie skupiają się oni na optymalizacji kosztowej systemów do analizy meczów piłkarskich i proponują tanie rozwiązanie chmurowe, na które mogą pozwolić sobie zespoły z lig nieprofesjonalnych.

Inne interesujące rozwiązanie problematyki śledzenia piłkarzy zostało zaproponowane w [8]. Wykorzystuje ono obraz z jednej, poruszającej się kamery oraz filtr cząsteczkowy w celu śledzenia konkretnych piłkarzy na boisku. Poza systemami z jedną kamerą, w literaturze opisanych zostało wiele systemów wizyjnych opierających się na danych z kilku statycznych kamer. Przykładowe rozwiązanie można znaleźć w [9], gdzie obraz meczu zbierany jest z dwóch kamer w taki sposób, że jedną z nich obserwuje jedną połowę boiska, a drugą - drugą połowę boiska.

Powszechnym problemem śledzenia piłkarzy na materiale filmowym jest fakt występuowania kontaktów fizycznych między zawodnikami. Wiele algorytmów wykrywania piłkarzy tego nie uwzględnia i wykrywa takich zawodników jako jeden obiekt. Problem ten został zwizualizowany na Rys. 2.1. Z tym problemem autorzy poradzili sobie w [13] za pomocą technik opartych o metodologię grafów, tworząc algorytm opisujący interakcje piłkarzy między sobą.



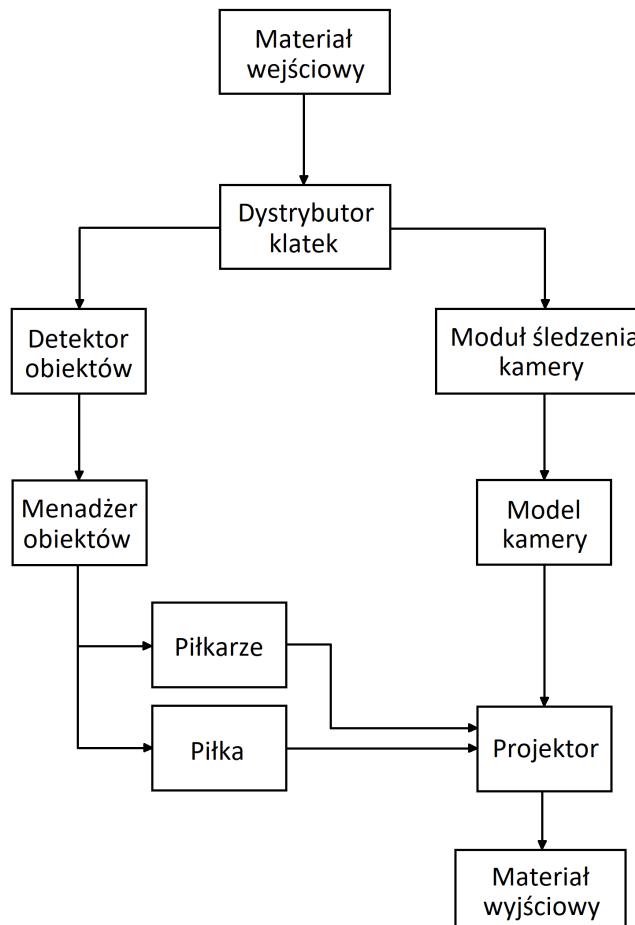
Rys. 2.1. Przykład wykrycia piłkarzy będących w bliskim kontakcie fizycznym

Wiele zagadnień z dziedziny analizy obrazu z meczów piłkarskich zostało rozwiązańnych na różne sposoby, co pozwala na stwierdzenie, że warto szukać innych, nowych metod, które w przeszłości mogą okazać się kluczowe w kontekście rozwoju wizji komputerowej. Literatura pokazuje, że przetwarzanie obrazu w celu detekcji piłkarzy i piłki można osiągnąć poprzez klasyczne metody przetwarzania obrazu [10], jak i z wykorzystaniem sztucznej inteligencji [11, 12]. Metody te mają swoje wady i zalety. Klasyczne metody przetwarzania obrazu mogą okazać się zawodne na poziomie skuteczności algorytmu. Z drugiej strony dają większą kontrolę nad rezultatem niż głębokie sieci neuronowe. Dodatkową ich zaletą jest to, że można je w łatwy sposób zrównoleglić, co umożliwia tworzenie algorytmów działających w czasie rzeczywistym. Natomiast sieci neuronowe np. te, które wykorzystywane są przez algorytm YOLO (*you only look once*, z ang. patrzysz tylko raz), mogą okazać się skuteczniejszą alternatywą. Ich wadą jest to, że wymagają złożonych modeli, które w wielu przypadkach uniemożliwiają pracę w czasie rzeczywistym.

### 3. IMPLEMENTACJA SYSTEMU

Problematyka wizji komputerowej sprowadza się do zastosowania metod przetwarzania obrazu w celu wykrycia na nim interesujących obiektów. W przypadku analizy meczu piłkarskiego z widoku kamery telewizyjnej takimi obiektemi są piłkarze, piłka oraz płyta boiska. W literaturze istnieje wiele metod podejścia do problemu detekcji piłkarzy i rzutowania ich na dwuwymiarową płaszczyznę boiska, przy czym można wyróżnić dwa główne nurty rozwiązań. Jednym z nich jest zastosowanie klasycznych metod przetwarzania obrazu opartych o manipulacje obrazem w celu ekstrakcji danych na temat pozycji piłkarzy, piłki oraz regionu boiska, na który skierowany jest widok z kamery. Drugim nurtem jest stosowanie metod sztucznej inteligencji, które najczęściej - w oparciu o sieci neuronowe, umożliwiają detekcję piłkarzy z zadowalającą skutecznością. W przypadku niniejszego rozwiązania wybrana została ścieżka klasycznych metod przetwarzania obrazu. Daje ona dużą przestrzeń do eksperymentowania, co może przyczynić się do rozwoju metod wizji komputerowej, nie tylko w specyficzny problemie analizy meczu piłkarskiego, ale również w przypadku innych problemów związanych z szeroko pojętą wizją komputerową.

#### 3.1. Architektura programu



Rys. 3.1. Schemat blokowy architektury programu

Program został napisany w języku Python z wykorzystaniem biblioteki OpenCV [14]. Na rys. 3.1 przedstawiona została architektura proponowanego rozwiązania. Materiałem wejściowym jest film przedstawiający mecz piłkarski z perspektywy kamery telewizyjnej. Z tego materiału odczytywane są kolejne klatki które trafiają do Dystrybutora Klatek. Dystrybutor Klatek jest elementem centralnym całej architektury. Jest on odpowiedzialny za przetwarzanie wstępne każdej klatki materiału, a następnie wysłanie odpowiednio spreparowanej klatki do Detektora Obiektów oraz Modułu Śledzenia Kamery. Dzięki zastosowaniu takiego centralnego elementu możliwe jest rozdzielenie następnych etapów algorytmu na dwa, niezależne procesy.

Pierwszym z nich jest proces detekcji obiektów oparty o Detektor Obiektów, który przetwarza, spreparowaną przez Dystrybutor Klatek, klatkę i wykrywa na niej obiekty będące kandydatami na piłkarzy lub piłkę. Kandydaci zostają skierowani do Menadżera Obiektów, w którym zostają im przypisane pewne cechy opisane w dalszej części pracy. Na ich podstawie określone jest, czy dany kandydat jest faktycznym obiektem, czy przykładem fałszywego wykrycia. Dodatkową rolą Menadżera Obiektów jest przechowywanie informacji na temat obiektów wykrytych w poprzednich klatkach. Dzięki temu może on lepiej określać, czy dany kandydat jest przykładem fałszywego wykrycia, czy jest faktycznym obiektem. Poza tym, te informacje są podstawą do dopasowywania obiektów wykrytych w bieżącej klatce, do obiektów zlokalizowanych w klatce poprzedniej, dzięki czemu możliwe jest śledzenie konkretnych obiektów. Obiekty potwierdzone jako rzeczywiste są następnie przesyłane do Projektora.

Drugim z procesów jest algorytm śledzenia ruchu kamery oparty o Moduł Śledzenia Kamery. Moduł ten wykonuje operacje przetwarzania obrazu na, spreparowanej przez Dystrybutor Klatek, klatce. Dzięki temu jest w stanie wykryć linie boiska, na podstawie których wyznaczane są punkty charakterystyczne, będące punktami intersekcji pomiędzy sasiadującymi liniami. Na ich podstawie estymowany jest ruch kamery, który przesyłany jest do Modelu Kamery. Model Kamery przekształca wykryty ruch na globalne koordynaty w przestrzeni. Koordynaty te są następnie przesyłane do Projektora.

Projektor jest elementem spinającym oba procesy. Moduł ten otrzymuje na wejście globalną pozycję kamery w przestrzeni oraz koordynaty piłkarzy i piłki na obrazie. Następnie wykonuje na nich szereg przekształceń, co pozwala na uzyskanie skutecznego rzutowania obiektów z obrazu kamery, obserwującej mecz w trójwymiarowej perspektywie, na dwuwymiarową płaszczyznę boiska.

Działanie całego programu opiera się o plik konfiguracyjny. Zmienne konfigurowane dzielą się na dwie kategorie: zmienne parametrów algorytmów przetwarzania obrazu oraz zmienne definiujące ostateczny wygląd pliku wyjściowego. Wśród zmiennych parametrów algorytmów znajdują się: przedziały wartości poszczególnych kolorów obrazu, które są wykorzystywane podczas tworzenia maski w konkretnych krokach algorytmu, opisanych w późniejszej części pracy, kolor wykrywanej piłki oraz długość przerwy pomiędzy końcem przetwarzania klatki bieżącej, a początkiem przetwarzania klatki następnej. Poza parametrami algorytmów, w pliku konfiguracyjnym można znaleźć zmienną określającą nazwę pliku wejściowego, ścieżkę do pliku wyjściowego, kolory obwiedni wokół wykrytych piłkarzy jakie mają się wyświetlać w materiale wyjściowym, zmienną określającą, czy te kolory obwiedni mają być wykorzystywane, czy mają być ustawione automatycznie oraz tło boiska wykorzystywane przy projekcji piłkarzy w Projektorze.

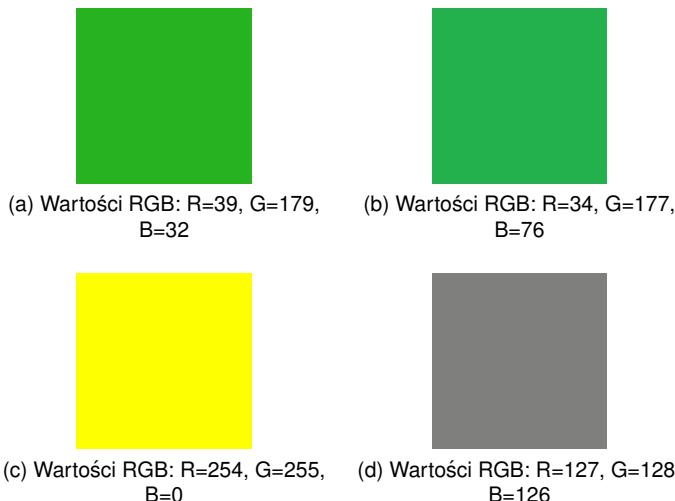
### 3.2. Przetwarzanie wstępne

Podstawą systemów do analizy meczów piłkarskich jest dobry algorytm wykrywania obiektów na obrazie, działający z wystarczającą skutecznością. Gruntem takiego algorytmu jest znalezienie ROI (*Region of interest* z ang. obszar zainteresowania). Jest to istotne, z punktu widzenia efektywności algorytmu, ze względu na kibiców znajdujących się na trybunach, którzy mogliby zostać wykryci i zinterpretowani jako piłkarze. W przypadku systemów opartych o kamery statyczne znalezienie ROI nie stanowi problemu, gdyż można je skonfigurować ręcznie. Sytuacja się komplikuje w przypadku systemów opartych o kamery poruszające się w przestrzeni. Dla nich istotnym jest, aby algorytm był w stanie znaleźć obszar zainteresowania niezależnie, w każdej klatce przetwarzanego filmu.

Narzucającym się rozwiązańiem jest wykrycie boiska na obrazie. Wiele źródeł w literaturze takich jak [15] twierdzi, że najlepszym do tego sposobem jest wykorzystanie algorytmów progowania (*thresholding* szeroko opisany w [16]) do stworzenia maski, powstałej na skutek stosowania zasady:

$$Maska(x, y) = \begin{cases} 0, & \text{dla } G(x, y) > R(x, y) > B(x, y) \\ 1, & \text{w przeciwnym wypadku} \end{cases} \quad (3.1)$$

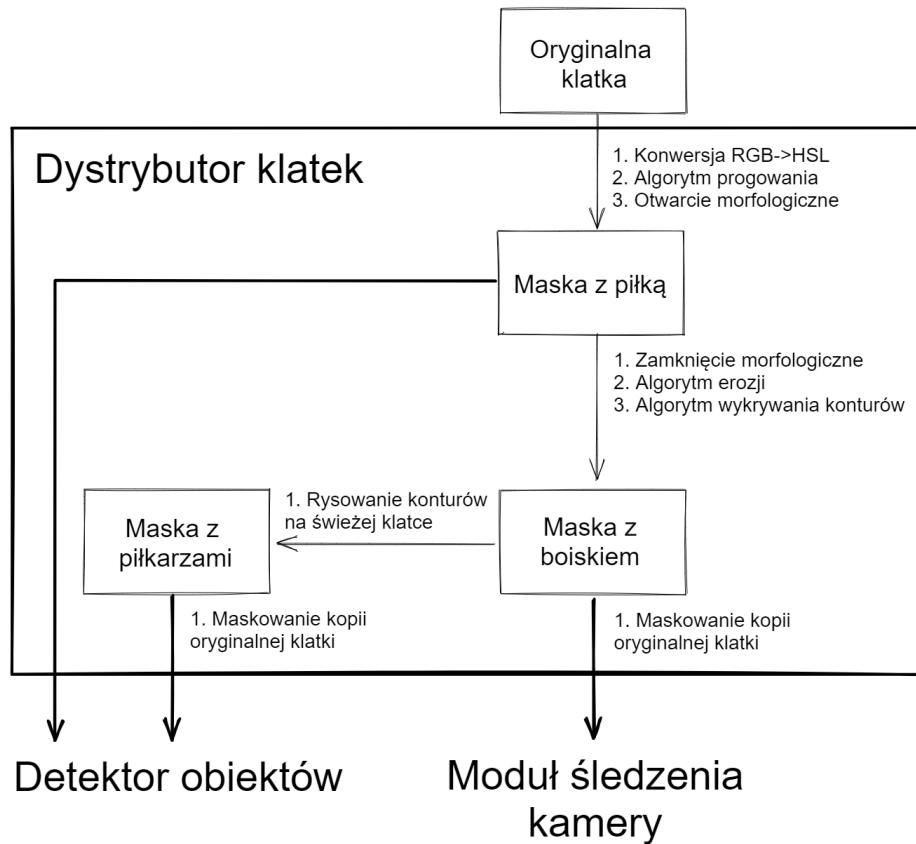
gdzie  $G(x, y)$  to wartość koloru zielonego w pikselu,  $R(x, y)$  to wartość koloru czerwonego w pikselu,  $B(x, y)$  to wartość koloru niebieskiego w pikselu. Takie progowanie pozwala na ekstrakcję boiska z wycięciem piłkarzy biegących po boisku, co jest dobrym punktem startowym do znalezienia ich pozycji na obrazie. Nie jest to jednak rozwiązanie perfekcyjne, gdyż nie odróżnia ono koloru zielonego, żółtego oraz odcienni szarości, co może przyczynić się do złego określenia regionu zainteresowania, a co za tym idzie, do zwiększonej aktywności algorytmu do fałszywych wykryć. Z drugiej strony dwa odcienie koloru zielonego mogą być od siebie odróżnialne przez co boisko nie będzie prawidłowo wycięte z obrazu i pojawią się defekty. Problem ilustruje przykład 3.2. Jest to spowodowane wykorzystywaniem modelu RGB (*Red, Green, Blue* z ang. Czerwony, Zielony, Niebieski), który dla każdego piksela opisuje intensywność kolorów: czerwonego, zielonego i niebieskiego. Model ten jest znacznym uproszczeniem rzeczywistości i może prowadzić do tego typu niedogodności.



Rys. 3.2. Kolory a,c,d nie są odróżniane przez zależność 3.1, kolory a i b są odróżniane przez zależność 3.1

Są to problemy, z którymi można sobie poradzić stosując dodatkowe przetwarzanie i operacje morfologiczne opisane w [18], jednakże może to znacznie obniżyć wydajność takiego rozwiązania. Z tego względu w algorytmie opisanym w niniejszej pracy, została wykorzystana inna, również oparta o algorytm progowania, metoda wykrywania ROI na obrazie. Metoda ta polega na przekształceniu obrazu z modelu RGB na model HSL (*Hue, Saturation, Lightness* z ang. Odcień, Nasycenie, Oświetlenie) [17], który opisuje kolory w sposób przypominający ludzką percepcję. Dzięki takiej konwersji kolory, z punktu widzenia człowieka, podobne do siebie, mają zbliżone wartości odcienia, co sprawia, że łatwiej jest kontrolować algorytm progowania. Wartości poziomów progowania w programie są konfigurowalne, co nadaje rozwiązań elastyczność i możliwość dostosowania w zależności od jakości obrazu z materiału wejściowego. Na tym etapie, w tle obrazu mogą pojawić się artefakty. Z tego powodu następnym krokiem jest wykonanie operacji otwarcia morfologicznego, które maskuje niedoskonałości algorytmu progowania i kończy krok poszukiwania wstępniego ROI.

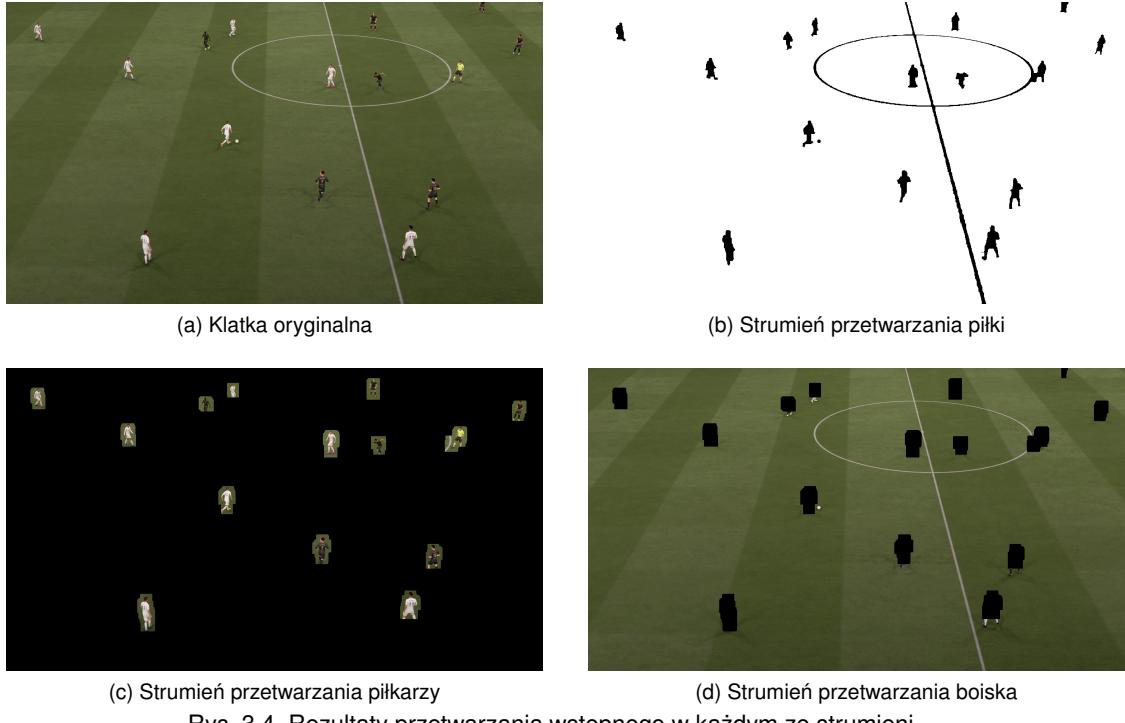
Przetwarzanie wstępne ma miejsce w Dystrybutorze Klatek i zostało ono zilustrowane w 3.3. Składa się z trzech strumieni przetwarzania: piłki, piłkarzy i boiska. Strumieniowość rozwiązania pozwala na uniezależnienie od siebie procesów przetwarzania, co umożliwia podejście do każdego z problemów detekcji: piłki, piłkarzy i boiska, indywidualnie. W praktyce strumienie posiadają wspólne kroki algorytmu, co zwiększa wydajność rozwiązania, ale może utrudnić proces wprowadzania zmian w przyszłości do konkretnych strumieni przetwarzania.



Rys. 3.3. Schemat blokowy przetwarzania wstępnego

Kolejnym krokiem wstępniego przetwarzania jest wykonanie operacji zamknięcia morfologicznego, umożliwiającej uwzględnienie w ROI linii boiska. Następnie wykonywany jest algorytm erozji, który uwydatnia rejony, w których znajdują się obiekty na obrazie. Kolejną czynnością jest wykorzystanie algorytmu wykrywania konturów opisanego w [19], który pozwala na sepeację re-

jonów z obiektami i uzyskanie ich pozycji na obrazie. Późniejszym krokiem przetwarzania wstępne jest narysowanie konturów na białej, świeżej klatce, co skutkuje maską, eliminującą cały obraz z wyjątkiem obszarów, w których mogą znajdować się piłkarze. Ostatnim elementem przetwarzania jest maskowanie oryginalnej klatki, otrzymując cechy umożliwiające dalsze przetwarzanie obrazu, indywidualne dla każdego z trzech strumieni przetwarzania. Rezultaty przetwarzania wstępne przedstawia 3.4



Rys. 3.4. Rezultaty przetwarzania wstępne w każdym ze strumieni

### 3.3. Detekcja piłki

Proces wykrywania piłki z poziomu kamery telewizyjnej może stanowić wyzwanie. Piłka jest małym elementem na obrazie i nieumiejętne jego przetwarzanie może skutkować utratą informacji o niej. Zbyt intensywne wykorzystywane operacje morfologiczne mogą sprawić, że piłka zostanie potraktowana jak szum i zostanie wyeliminowana z obrazu, tak jak dzieje się to podczas przetwarzania wstępne strumienia detekcji piłkarzy. Z tego powodu przetwarzanie wstępne piłki jest odseparowane od przetwarzania wstępne piłkarzy.

Pierwszym krokiem w procesie detekcji piłki jest wykonanie dodatkowej erozji maski z piłką, powstałej na skutek przetwarzania wstępne. Przeprowadzenie jej uwydatnia na obrazie piłkę, co zwiększa niezawodność algorytmu w jej wykrywaniu. Następnie wykrywane są wszystkie kontury, z których wyeliminowane zostają najmniejsze oraz największe, pozostawiając jedynie te o rozmiarze przypominającym piłkę. Przedział dopuszczalnych wartości rozmiaru konturów został wyznaczony eksperymentalnie. Rozmiar ten jest określany na podstawie pola powierzchni zakreślonego przez kontur, wyznaczanego poprzez algorytm korzystający z wzoru Greena [20] na całkę krzywoliniową:

$$\oint_C (L \, dx + M \, dy) = \iint_D \left( \frac{\partial M}{\partial x} - \frac{\partial L}{\partial y} \right) \, dx \, dy \quad (3.2)$$

gdzie  $C$  jest zamkniętą krzywą regularną, skierowaną dodatnio, będącą brzegiem obszaru regularnego  $D$ , a  $L$  i  $M$  to funkcje argumentów  $(x, y)$ , zdefiniowane na obszarze jednospójnym  $\mathbb{R}^2$ , zawierającym obszar  $D$ , które mają ciągłe pochodne cząstkowe. Algorytm wykorzystujący powyższe równanie został szerzej opisany w [21].

Pozostałe kontury to tzw. kandydaci na piłkę. Dla każdego kandydata przygotowywany jest *bounding box* (z ang. minimalna obwiednia, otaczająca dany kontur) oraz średni kolor pikseli we wnętrzu danej obwiedni na obrazie oryginalnym. Średni kolor określany jest na podstawie próbek z maksymalnie stu równomiernie rozmieszczenych punktów na siatce we wnętrzu całej obwiedni. Dany punkt jest brany pod uwagę do liczenia średniej, jeżeli spełnia zależność 3.1. Pozwala to na wyeliminowanie większości zielonych pikseli i na uniknięcie sytuacji, w której te piksele przeklamywałyby faktyczny, średni kolor piłki na obrazie. Poza średnim kolorem dla każdego kandydata obliczany jest wskaźnik mówiący o tym, jak bardzo dany kontur przypomina okrąg. Wskaźnik ten jest otrzymywany zgodnie ze wzorem:

$$circularity(c) = \frac{P_c(c)}{P_o(c)} \quad (3.3)$$

gdzie  $P_c(c)$  jest polem zajmowanym przez kontur  $c$ , a  $P_o(c)$  jest polem zajmowanym przez minimalny okrąg otaczający kontur  $c$ . W kolejnym kroku obliczany jest dystans koloru kandydata do oczekiwanej koloru piłki zdefiniowanego w pliku konfiguracyjnym. W najprostrzym przypadku można byłoby stwierdzić, że perfekcyjnym rozwiązaniem byłoby obliczenie odległości euklidesowej w przestrzeni  $\mathbb{R}^3$ , w której każda z osi jest osobnym kanałem RGB:

$$color\_distance(x_1, x_2) = \sqrt{\Delta R_{12}^2 + \Delta G_{12}^2 + \Delta B_{12}^2} \quad (3.4)$$

gdzie  $\Delta R_{12}$  to różnica pomiędzy wartościami kanału czerwonego pomiędzy kolorami  $x_2, x_1$ ,  $\Delta G_{12}$  to różnica pomiędzy wartościami kanału zielonego pomiędzy kolorami  $x_2, x_1$ ,  $\Delta B_{12}$  to różnica pomiędzy wartościami kanału niebieskiego pomiędzy kolorami  $x_2, x_1$ . W przypadku oceny podobieństwa dwóch kolorów jest to równanie niewystarczające. Model RGB został stworzony z myślą o maszynach, a nie ludziach. O wiele lepszym rozwiązaniem jest wykorzystanie równania "czerwonej średniej" 3.5, które zostało stworzone z myślą o ocenie podobieństwa kolorów zgodnie z ludzką percepcją:

$$\bar{r} = \frac{1}{2}(R(x_1) + R(x_2))$$

$$color\_distance(x_1, x_2) = \sqrt{(2 + \frac{\bar{r}}{256})\Delta R_{12} + 4\Delta G_{12}^2 + (2 + \frac{255 - \bar{r}}{256})\Delta B_{12}^2} \quad (3.5)$$

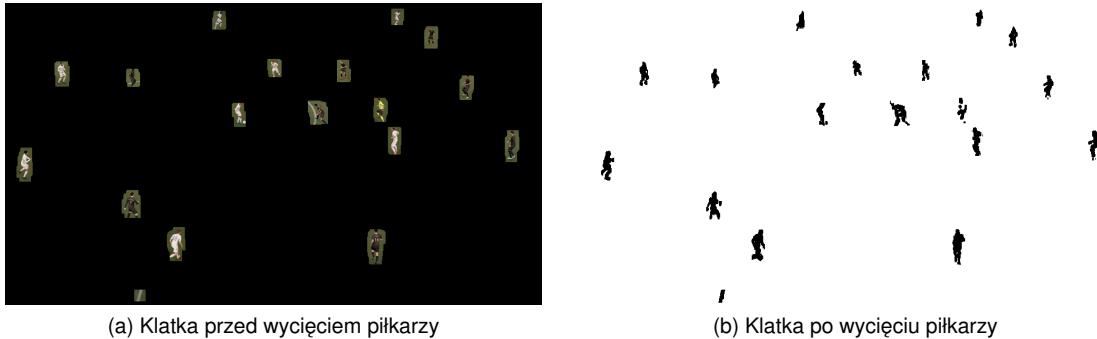
Taka metryka pozwala na obiektywną ocenę podobieństwa dwóch kolorów, dzięki czemu obliczony wskaźnik dystansu do oczekiwanej koloru piłki może być wykorzystany w ocenie, czy dany kandydat jest rzeczywistą piłką, czy przykładem fałszywego wykrycia. Ostatecznie obliczany jest wskaźnik końcowy, będący ważoną różnicą pomiędzy wskaźnikiem 3.3, a wskaźnikiem 3.5:

$$score(c) = 1000circularity(c) - color\_distance(color(c), color(ball)) \quad (3.6)$$

gdzie  $color(x)$  jest wartością koloru danym modelem RGB. Kontur z największą wartością  $score(c)$  jest interpretowany jako piłka i to jego pozycja jest później wyświetlana w Projektorze.

### 3.4. Detekcja piłkarzy

Wykrywanie zawodników to proces bardzo podobny do procesu detekcji piłki. W pierwszym kroku klatka powstała na skutek przetwarzania wstępniego strumienia wykrywania piłkarzy jest poddawana zależności 3.1, a następnie algorytmowi erozji, w celu wypełnienia niedokładności powstałych na skutek maskowania wzorem 3.1. Pozwala to na wycięcie z rejonów, w których znajdują się zawodnicy, ich sylwetki, co skutkuje, w późniejszych krokach algorytmu, znalezieniem lepszej minimalnej obwiedni otaczającej danego piłkarza. Rysunek 3.5 przedstawia rezultaty tego przetwarzania. Następnym krokiem, podobnie jak w przypadku procesu wykrywania piłki,



Rys. 3.5. Rezultaty wycięcia sylwetek piłkarzy z rejonów wyznaczonych w trakcie przetwarzania wstępnego

jest znalezienie konturów sylwetek na obrazie, wśród których eliminowane są te o najmniejszym i największym rozmiarze. Przedział dopuszczalnych wartości rozmiaru znalezionych sylwetek różni się od przedziału wyznaczonego na potrzeby detekcji piłki, ale również został on wyznaczony eksperymentalnie. Rozmiar konturów jest obliczany poprzez stosowanie algorytmu wykorzystującego zależność 3.2. Pozostałe kontury są tzw. kandydatami na piłkarzy, dla których wyznaczana jest minimalna obwiednia otaczająca danego kandydata oraz średni kolor wyznaczony podobnie jak w przypadku procesu wykrywania piłki. Główną różnicą w określaniu średniego koloru kandydata jest to, że punkty nie są równomiernie rozłożone wewnątrz całej obwiedni otaczającej piłkarza, a w obrębie wyznaczonego eksperymentalnie obszaru  $D_i$  w środku danej obwiedni, zdefiniowanego jako:

$$D_i = f(D_e) = \begin{cases} x_i = x_e + \frac{w_e}{3} \\ y_i = y_e + \frac{h_e}{5} \\ w_i = \frac{w_e}{3} \\ h_i = \frac{h_e}{10} \end{cases} \quad (3.7)$$

gdzie  $D_e$  to obszar zdefiniowany przez zmienne  $x_e$ ,  $y_e$  będącymi koordynatami lewego górnego rogu obwiedni otaczającej danego kandydata, a  $w_e$ ,  $h_e$  to odpowiednio szerokość i wysokość tej obwiedni. W praktyce obszar  $D_i$  w dużej mierze otacza koszulkę piłkarza, co umożliwia określenie jej koloru. Tak spreparowani kandydaci stają się piłkarzami, po czym trafiają do Menadżera Obiektów, który nimi zarządza, nadaje im odpowiednie identyfikatory oraz przypisuje ich do odpowiednich zespołów.



Rys. 3.6. Czarne i białe prostokąty to obwiednie  $D_e$  otaczające piłkarzy, a czerwone to obwiednie wewnętrzne  $D_i$  wyznaczone na podstawie zależności 3.7

### 3.5. Metodyka śledzenia piłkarzy

Śledzenie piłkarzy opiera się na systemie znajdowania kandydatów przez Detektor Obiektów, a następnie ich analizę i dopasowanie do istniejących już w pamięci programu zawodników. Dla każdego z kandydatów na piłkarza wykonywany jest szereg czynności porównujących go z zawodnikami z pamięci. Pierwszym krokiem jest wyznaczenie odległości kandydata do każdego z piłkarzy w systemie. Jest ona obliczana ze wzoru na odległość euklidesową:

$$\text{plain\_distance}(c, f) = \sqrt{\Delta x_{cf}^2 + \Delta y_{cf}^2} \quad (3.8)$$

gdzie  $x_{cf}$ ,  $y_{cf}$  to odpowiednio różnica pomiędzy współrzędnymi osi x i y kandydata  $c$  i piłkarza  $f$ . Dokładne wartości otrzymane z 3.8 nie są konieczne, więc w praktyce, we wzorze 3.8 można opuścić pierwiastek, który jest dla komputerów kosztowną operacją, co zwiększa wydajność rozwiązania:

$$\text{plain\_distance}(c, f) = |\Delta x_{cf}| + |\Delta y_{cf}| \quad (3.9)$$

Kolejnym krokiem jest wybranie piłkarza z pamięci, którego odległość do danego kandydata jest najmniejsza. Jeżeli odległość ta jest zbyt duża oraz w pamięci jest dostępne miejsce dla nowego zawodnika, taki kandydat jest traktowany jako nowy piłkarz i jest wpisywany do systemu. W przeciwnym wypadku, kandydat dopasowywany jest do wybranego wcześniej zawodnika, który przejmuje jego cechy: minimalną obwiednie otaczającą kandydata oraz średni kolor obwiedni wewnętrznej. W tym przypadku jest to jednoznaczne z procesem śledzenia piłkarza. Miejsc dla zawodników jest 25: po 11 dla obu drużyn oraz 3 dodatkowe miejsca dla sędziów, których trudno jest jednoznacznie odróżnić od piłkarzy.

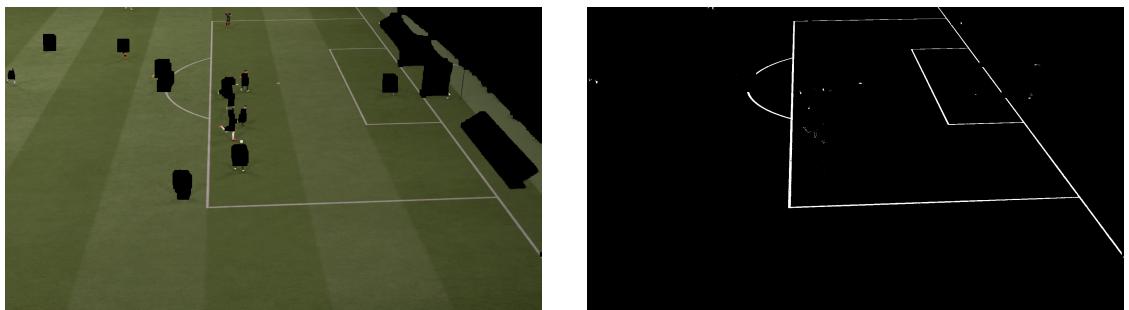
Dodanie nowego zawodnika do pamięci programu implikuje analizę tego, czy należy dodać nowy zespół do pamięci. Proces analizy poczyna się wyznaczeniem podobieństwa średniego koloru piłkarza z kolorami istniejących w pamięci drużyn z zależności 3.4. Następnie wybierany jest zespół o najbardziej podobnym kolorze do średniego koloru zawodnika, czyli o najmniejszej wartości odległości pomiędzy tymi kolorami. Jeżeli wartość ta jest zbyt duża i w pamięci programu jest dostępne miejsce na nową drużynę, to tworzony jest nowy zespół o kolorze założyciela, po czym jest on do niego przypisywany. W przeciwnym wypadku, nowy zawodnik przypisywany jest do wybranego wcześniej zespołu. Miejsc dla unikalnych drużyn są 2: dla drużyny gospodarzy

i gości, ignorując zespół sędziów. Jeżeli dany kandydat zostanie zinterpretowany jako istniejący w systemie zawodnik, nie następuje proces ewentualnego dodania nowego zespołu, jednakże potwierdzana jest przynależność do obecnej drużyny. Dzieje się to w taki sam sposób jak w przypadku dodawania nowego piłkarza, jednakże proces ten nie dopuszcza stworzenia nowego zespołu tzn. nie istnieje "zbyt duża" wartość odległości koloru zawodnika do koloru drużyny. Za każdym razem wybierany jest zespół o najmniejszej wartości odległości kolorów. Granica, dla której wartość odległości staje się "zbyt duża" została wyznaczona eksperymentalnie.

Dodatkowo, w każdej klatce filmu obliczany jest wektor prędkości poruszania się piłkarza po obrazie, na podstawie informacji o jego pozycji w klatce bieżącej oraz w klatce poprzedniej. Wektor ten jest wykorzystywany w momencie, gdy do danego piłkarza z pamięci nie zostanie przypisany żaden nowy kandydat. Pozycja tego piłkarza w takiej sytuacji jest korygowana o wartość jego wektora prędkości. Co więcej, jeżeli zawodnik nie zostaje wykryty przez 5 kolejnych klatek to następuje jego usunięcie z pamięci. Stosowanie takiego rozwiązania sprawia, że w sytuacji, w której algorytm detekcji nie wykryje poprawnie danego zawodnika w specyficznej klatce, system ten jest w stanie zachować ciągłość w jego śledzeniu.

### 3.6. Detekcja linii boiska i punktów charakterystycznych

W wizji komputerowej wykrywanie linii na obrazie, w większości przypadków, opiera się na wykorzystaniu transformaty Hougha, szerzej opisanej w [22]. Pozwala ona na względnie wydajną detekcję prostych linii w obrębie całego obrazu. Przed przepuszczeniem klatki przez ten algorytm należy ją odpowiednio przygotować. W niniejszym rozwiązaniu klatka, powstała na skutek wstępniego przetwarzania, przekształcana jest z modelu RGB na model HSL, po czym poddawana jest algorytmowi progowania, w celu stworzenia maski, która na oryginalnej klatce pozostawi jedynie linie boiska. Rezultat tego przetwarzania został przedstawiony na rys. 3.7. Tak spreparowana klat-



(a) Klatka przed podaniem algorytmowi progowania

(b) Klatka po podaniu algorytmowi progowania

Rys. 3.7. Rezultaty algorytmu progowania w detekcji linii boiska

ka jest w następnym kroku poddawana algorytmowi probabilistycznej transformaty Hougha, która została opisana w [23]. Algorytm pozwala na otrzymanie wszystkich linii na obrazie w postaci punktów  $x, y$  będących krańcami wykrytych odcinków.

W kolejnym kroku każdy wykryty odcinek jest przekształcany na kilka różnych postaci. Pierwszą z nich jest postać kierunkowa  $y = ax + b$ , która pozwala na wydajne obliczanie punktów należących do tej prostej. Znalezienie jej sprowadza się do rozwiązania układu dwóch równań liniowych:

$$\begin{cases} y_1 = ax_1 + b \\ y_2 = ax_2 + b \end{cases} \quad (3.10)$$

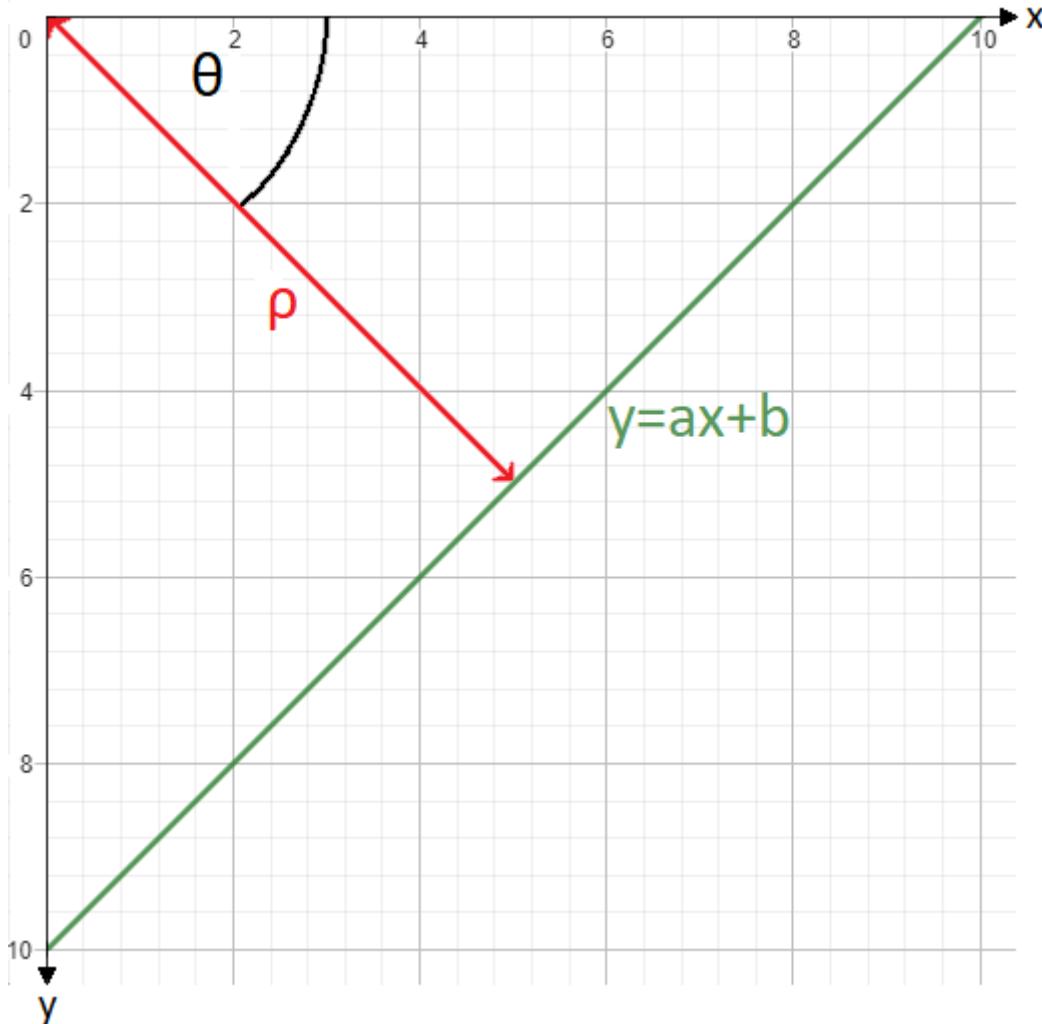
którego rozwiązaniem jest:

$$\begin{cases} a = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{dla } x_1 \neq x_2 \\ 10000 & \text{dla } x_1 = x_2 \end{cases} \\ b = y_1 - ax_1 \end{cases} \quad (3.11)$$

Warto zauważyć, że postać kierunkowa dla prostych idealnie pionowych przyjmuje wartość  $a = \infty$ . Z tego względu należy przybliżyć ten parametr, ustawiając go na odpowiednio dużą wartość. Zmiana orientacji prostej w przestrzeni względem zmian parametrów  $a, b$  odcinka opisanego prostą, daną postacią kierunkową  $y = ax + b$  o środku w punkcie  $S$  nie ma charakteru liniowego. Im większe  $|a|$ , tym zmiany kąta nachylenia tej prostej do osi x stają się coraz mniejsze. To sprawia, że postać ta nie jest dobrym wskaźnikiem do określania orientacji prostej. Orientacja prostej jest potrzebna w celach optymalizacyjnych algorytmu. Z tego powodu następnym krokiem jest przekształcenie wszystkich prostych z postaci kierunkowej na postać normalną:

$$x \cos(\theta) + y \sin(\theta) = \rho \quad (3.12)$$

gdzie  $\rho$  to odległość punktu  $(0, 0)$  od danej prostej  $y = ax + b$ , a  $\theta$  to kąt jaki zakreśla prosta prostopadła do  $y$  przechodząca przez ten punkt. Przekształcenie wizualizuje wykres 3.8. Postać ta



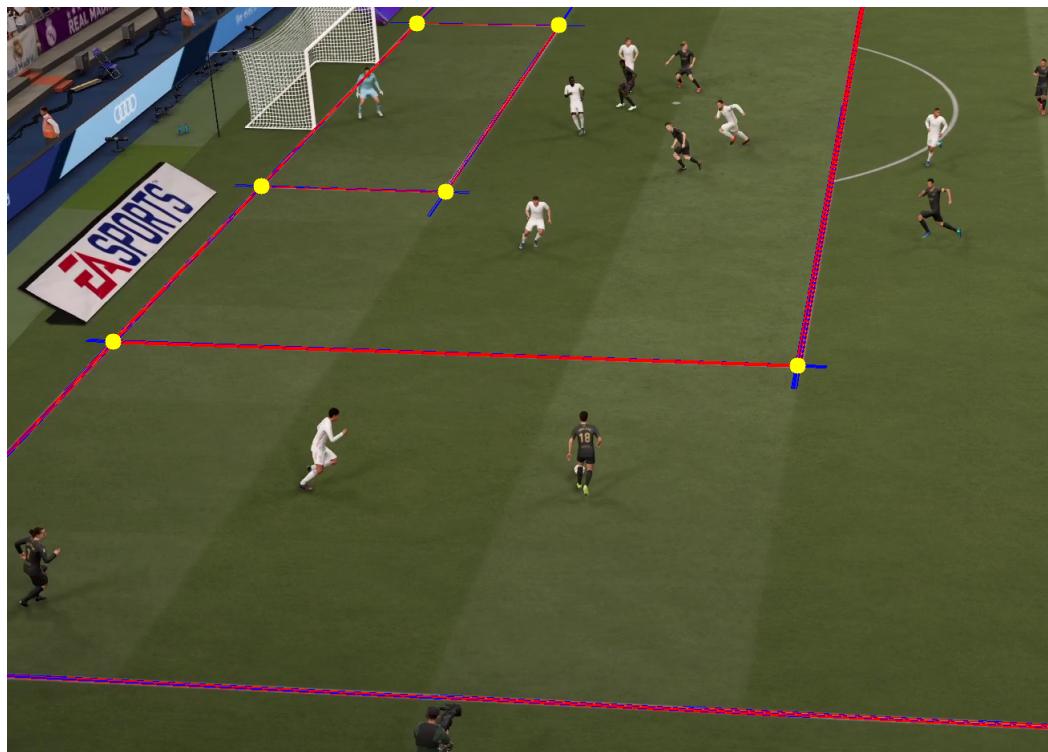
Rys. 3.8. Sposób reprezentacji prostej kierunkowej  $y = ax + b$  w postaci normalnej  $x \cos(\theta) + y \sin(\theta) = \rho$

pozwala na określenie czy prosta ma orientację pionową czy poziomą. Wszystkie proste spełniające zależność  $|\theta| > 0.2\text{rad}$  są traktowane jako proste o orientacji poziomej, a te które spełniają zależność  $0.7\text{rad} > |\theta| > 2.6\text{rad}$  są traktowane jako proste o orientacji pionowej. Pozostałe proste traktowane są jako proste o orientacji niezdefiniowanej i są odrzucone z dalszej części algorytmu. Umożliwia to skuteczniejsze wyznaczanie punktów charakterystycznych, które zostało opisane w dalszej części pracy. Mając zdefiniowaną orientację prostej, wyznaczane są odcinki wydłużone względem odcinków wykrytych przez probabilistyczną transformate Hougha. Dla prostych horyzontalnych wyznaczane są punkty oddalone od punktów pierwotnych o 30 pikseli w osi x, a dla prostych wertykalnych - o 30 pikseli w osi y. Wydłużona forma prostych jest formą ostateczną, umożliwiającą stabilniejsze wyznaczanie punktów charakterystycznych.

Punkty charakterystyczne to punkty przecięcia pomiędzy odcinkami pionowymi i poziomymi. Istnienie punktu przecięcia pomiędzy nimi jest sprawdzane na podstawie [24] oraz [25]. Jeżeli punkt istnieje to następuje jego obliczenie poprzez rozwiązywanie układu dwóch równań liniowych 3.10, którego rozwiązaniem jest:

$$\begin{cases} x = \frac{b_1 - b_2}{a_1 - a_2} \\ y = a_1 x + b_1 \end{cases} \quad (3.13)$$

Ostatnim krokiem jest agregacja punktów charakterystycznych o podobnych współrzędnych, w taki sposób, aby otrzymać zbiór kilku pojedynczych punktów rozmieszczonych na konkretnych pozycjach na boisku. Zagregowane punkty mają współrzędne będące średnią arytmetyczną współrzędnych wszystkich punktów biorących udział w agregacji do konkretnej pozycji. Ostateczny rezultat algorytmu wykrywania linii wraz z punktami charakterystycznymi został przedstawiony na rys. 3.9



Rys. 3.9. Czerwone odcinki to linie wykryte przez probabilistyczną transformate Hougha, niebieskie odcinki to wydłużona forma odcinków czerwonych, a żółte punkty to zagregowane punkty charakterystyczne

### 3.7. Estymacja ruchu kamery

Klasyczna odometria wykorzystuje w praktyce fizyczne czujniki określające zmianę pozycji w czasie. W przypadku niniejszego rozwiązania, dane z czujników kamery takich jak akcelerometr, czy żyroskop nie są dostępne. Z tego powodu do estymacji ruchu kamery wykorzystana została wizualna odometria [26], opierająca się na obrazie z niej. Polega ona na znajdywaniu punktów charakterystycznych w każdej klatce filmu i śledzeniu zmian ich pozycji. Literatura wskazuje, że dobrym rozwiązaniem jest użycie algorytmu *Optical Flow* (z ang. ruch optyczny) [27], który znajduje kluczowe punkty na obrazie i śledzi ich ruch w czasie. Następnie tworzy pole wektorowe, które wskazuje kierunek ruchu tych punktów. Jest to rozwiązanie kosztowne obliczeniowo, więc nie zostało ono zaimplementowane w niniejszym rozwiążaniu. Zamiast tego, do estymaty wykorzystywane są punkty charakterystyczne, będące punktami przecięcia pomiędzy liniami boiska.

W każdej klatce filmu, punkt wykryty w obecnej iteracji jest porównywany z punktami z klatki poprzedniej. Na podstawie zależności 3.9 wybierany jest najbliższy z nich. Jeżeli punkt nie znajduje się "zbyt daleko" od wybranego punktu to obliczany jest wektor przesunięcia  $\bar{m}_j$  tego punktu w czasie, z zależności 3.14:

$$\bar{m}_j = \begin{bmatrix} m_x \\ m_y \end{bmatrix} = \begin{bmatrix} P_{xi-1} - P_{xi} \\ P_{yi-1} - P_{yi} \end{bmatrix} \quad (3.14)$$

gdzie  $P_{xi}$ ,  $P_{yi}$  to wykryty punkt charakterystyczny w klatce bieżącej, a  $P_{xi-1}$ ,  $P_{yi-1}$  jest punktem dopasowanym do  $P_{xi}$ ,  $P_{yi}$ , wykrytym w klatce poprzedniej. Ostatecznie, estymowany wektorem ruchu kamery jest średnia arytmetyczna z wektorów przesunięcia wszystkich wykrytych punktów

$$\bar{w} = \begin{bmatrix} w_x \\ w_y \end{bmatrix} = \frac{\sum_{j=1}^N m_j}{N} \quad (3.15)$$

gdzie  $N$  to liczba dopasowanych do siebie par punktów  $P_{xi}$ ,  $P_{yi}$  i  $P_{xi-1}$ ,  $P_{yi-1}$ . Jeżeli dany punkt znajduje się zbyt daleko od najbliższego punktu, wybranego na podstawie 3.9, to traktowany jest on jako nowy punkt, a więc wektor  $\bar{m}_j$  nie jest dla niego obliczany. To sprawia, że takie punkty nie biorą udziału w liczeniu estymaty  $\bar{w}$ . Punkty znajdują się "zbyt daleko" od siebie, gdy wartość zależności 3.9 przekroczy pewną, wyznaczoną eksperymentalnie stałą.

### 3.8. Projekcja obiektów na dwuwymiarową płaszczyznę boiska

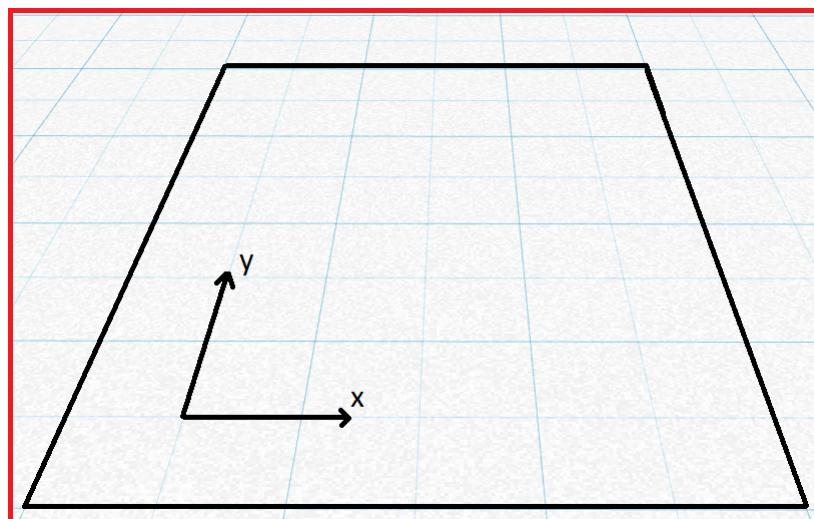
Projekcja wykrywanych obiektów na obrazie z kamery może sprowadzić się do transformacji przestrzeni trójwymiarowej, do przestrzeni dwuwymiarowej. Wykonanie takiego przekształcenia sprawia, że koordynaty obiektów wykrytych na tak spreparowanym obrazie są jednocześnie ich położeniem na płaszczyźnie 2D. Zaletą tego rozwiązania jest to, że późniejsze przekształcenia w celu wykonania projekcji z perspektywy lotu ptaka stają się liniowe. Z drugiej strony, wykonanie takiej transformacji jest kosztowne obliczeniowo i wymaga znajomości macierzy fundamentalnej [28], opisującej sposób obserwacji świata przez kamerę. Jest ona zależna od parametrów obiektywu, które w niniejszym rozwiążaniu nie są znane.

Kompromisem pomiędzy dokładnością, a złożonością obliczeniową jest przyjęcie pewnych założeń, a następnie oszacowanie stałej transformacji, prawdziwej gdy założenia są spełnione. Założeniem proponowanego rozwiązania jest stały kąt nachylenia kamery względem boiska i brak

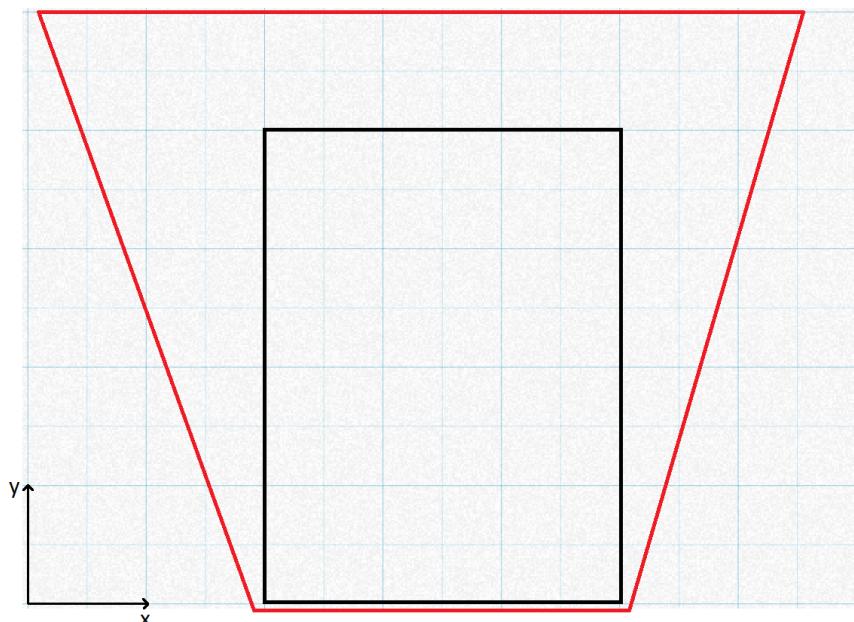
ruchu w osi Z (góra, dół). Przyjęcie takich założeń pozwala na zastosowanie stałej transformacji obrazu prostokątnego, widzianego na obrazie z kamery, na obraz trapezoidalny, imitujący obraz z lotu ptaka. Trapez jest ramą, wewnętrznej której wyświetlani będą piłkarze oraz piłka. Założenia sprawiają, że taki trapez ma stałe rozmiary i nachylenie względem boiska, co sprawia, że pozycja kamery na płaszczyźnie Projektora może zostać wyznaczona na podstawie prostej zależności:

$$\begin{bmatrix} c_{xi} \\ c_{yi} \end{bmatrix} = \begin{bmatrix} c_{xi-1} + w_x \\ c_{yi-1} + w_y \end{bmatrix} \quad (3.16)$$

gdzie  $c_{xi}$ ,  $c_{yi}$  to pozycja kamery w przestrzeni Projektora w klatce  $i$ , a  $w_x$ ,  $w_y$  to wartości wektora ruchu kamery, wyznaczonego ze wzoru 3.15.



(a) Perspektywa 3D kamery obserwującej boisko



(b) Perspektywa 2D z lotu ptaka

Rys. 3.10. Czerwone ramy na obrazach pokazują perspektywę 3D odpowiednio na obrazie z kamery i na rzutowaniu obrazu na płaszczyznę 2D. Czarne ramy pokazują perspektywę ukazującą perspektywę 2D odpowiednio na obrazie z kamery i na płaszczyźnie 2D. Proporcje zostały zachowane.

Finalna projekcja sprowadza się do liniowego przekształcenia współrzędnych 2D z obrazu kamery 3.10(czerwony prostokąt na rysunku A) na współrzędne wewnętrz, poruszającego się po płaszczyźnie Projektora, trapezu imitującego perspektywę, obserwującej boisko, kamery. Taka transformacja może zostać dokonana poprzez zamianę współrzędnych kartezjańskich na współrzędne jednorodne [29]. Wiąże się to z dodaniem do współrzędnych kartezjańskich  $x$ ,  $y$  dodatkowej wartości  $w$ , będącej czynnikiem skalującym. Współrzędne jednorodne umożliwiają przedstawienie dowolnej transformacji w przestrzeni za pomocą macierzy [30]. W tym przypadku, macierz ma rozmiar 3x3 i można ją zdefiniować jako:

$$\begin{bmatrix} u' \\ v' \\ w_1 \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \end{bmatrix} \begin{bmatrix} x' \\ y' \\ w_2 \end{bmatrix} \quad (3.17)$$

gdzie  $u'$ ,  $v'$ ,  $q$  to współrzędne prostokątne na obrazie z kamery,  $x'$ ,  $y'$ ,  $w$  to współrzędne rzutowane na trapez, a  $A$ ,  $B$ ,  $C$ ,  $D$ ,  $E$ ,  $F$ ,  $G$ ,  $H$ ,  $I$  to współczynniki macierzy przekształcenia. Znając współczynniki macierzy przekształcenia można skalować, obracać, przesuwać współrzędne w przestrzeni w dowolny sposób. W przypadku niniejszego rozwiązania równanie 3.17 wykorzystane będzie do transformacji współrzędnych z prostokątnego obrazu z kamery na trapezoidalne współrzędne na płaszczyźnie Projektora. Takie przekształcenie pozwala na przyjęcie  $I = 1$ , co sprawia, że macierz posiada 8 stopni swobody, a więc możliwym jest wyznaczenie jej na podstawie 8 punktów. Tymi punktami są 4 narożniki prostokąta obrazu, które są wyznaczone na podstawie rozdzielczości analizowanego filmu, oraz 4 narożniki trapezu wynikowego, które zostały dopasowane eksperymentalnie, w taki sposób, aby dawały najlepsze rezultaty. Całość sprowadza się do rozwiązania układu 8 równań liniowych:

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad (3.18)$$

Układ równań 3.18 rozwiązywany jest tylko raz, na początku działania programu. To pozwala zaoszczędzić moc obliczeniową potrzebną na pozostałe operacje algorytmu. Posiadając wypełnioną macierz przekształcenia, transformacja koordynatów obiektów, wykrytych na obrazie, na pozycje wewnętrz trapezu sprowadza się do wymnożenia ich przez macierz przekształcenia, a następnie zamiana ich ze współrzędnych jednorodnych na współrzędne kartezjańskie. Aby to zrobić należy każdą wartość współrzędnych jednorodnych podzielić przez czynnik skalujący  $w$ . Kolejnym krokiem jest korekcja otrzymanych koordynatów o wartość pozycji kamery w przestrzeni Projektora. Łącząc zależności 3.16 oraz 3.17 finalna transformacja z współrzędnych kartezjańskich wykrytych obiektów na obrazie na współrzędne kartezjańskie wyświetlane na płaszczyźnie Projektora jest obliczana ze wzoru:

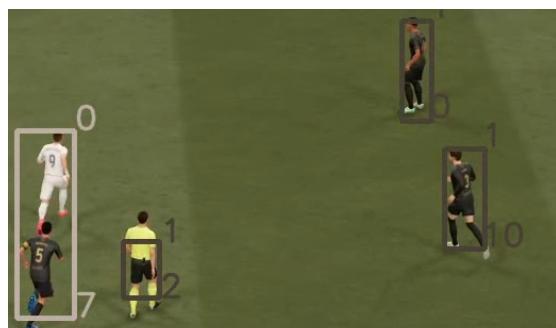
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \frac{u'}{w} \\ \frac{v'}{w} \end{bmatrix} + \begin{bmatrix} c_{xi} \\ c_{yi} \end{bmatrix} \quad (3.19)$$

## 4. TESTY

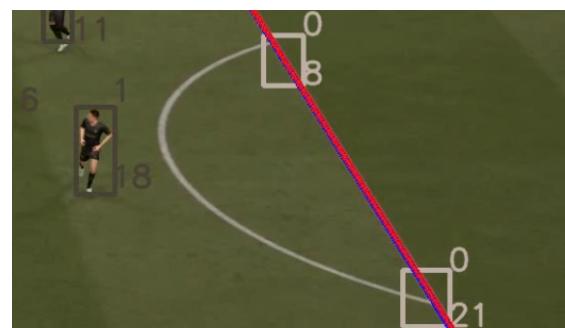
W ramach testów, wykorzystany został materiał filmowy przedstawiający rozgrywkę z gry FIFA 21. Pozwala ona na dostosowanie zachowania się kamery w taki sposób, żeby spełniała założenia projektowe dotyczące braku rotacji oraz ruchu w osi Z. Dodatkową zaletą jest możliwość dostosowania kolorów strojów, piłki, a także usunięcie wszelkich znaków wodnych i nakledek graficznych z obrazu, przy zachowaniu wysokiej jego rozdzielczości. Wszystkie testy zostały zrealizowane na jednym materiale filmowym [31].

### 4.1. Skuteczność wykrywania piłkarzy na obrazie

Efektywność detekcji obiektów można podzielić na efektywność detekcji piłkarzy oraz piłki. Skuteczność wykrywania piłkarzy jest najważniejszym wskaźnikiem jakości algorytmu, gdyż wpływa on na całą analizę meczu. Narzucającą się formą sprawdzenia efektywności tej części programu jest obliczenie stosunku wykrytych przez algorytm piłkarzy do oczekiwanej liczby piłkarzy jakie powinien wykryć algorytm, czyli wszystkich widzianych w danej klatce filmu. Tak skonstruowana metryka może przeklamywać rzeczywistość ze względu na możliwość wystąpienia błędów: fałszywych wykryć, braku detekcji poszczególnych piłkarzy, czy wykrycia dwóch lub więcej piłkarzy jako jednego tak jak na rys. 2.1. Z tego powodu skuteczność algorytmu wykrywania piłkarzy została zmierzona z wykorzystaniem stosunku wszystkich błędów detekcji do oczekiwanej liczby piłkarzy w danej klatce materiału filmowego. Błędy brane pod uwagę to: brak detekcji piłkarza, przypisanie



(a) Błąd interpretacji sędziego jako piłkarza oraz wykrycie dwóch piłkarzy jako jednego



(b) Błąd interpretacji pustej przestrzeni jako piłkarza



(c) Błąd braku detekcji piłkarza

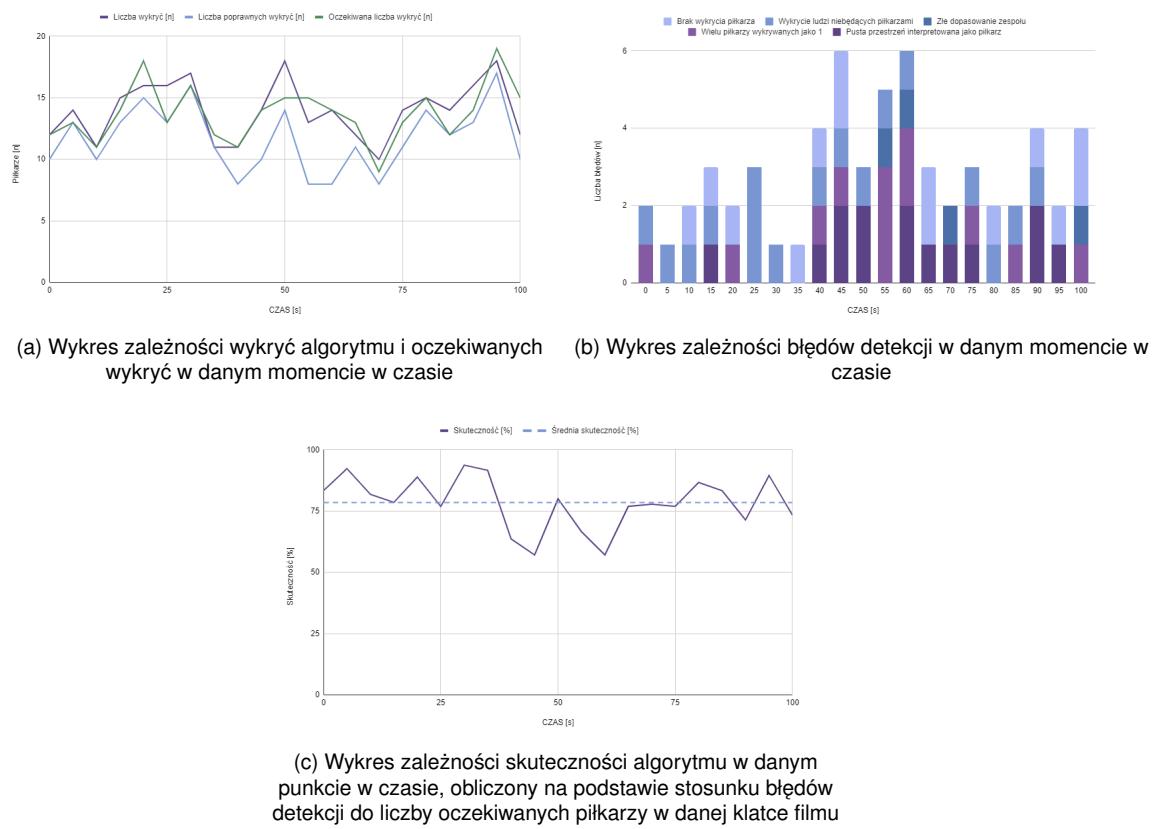


(d) Błąd przypisania piłkarza do zespołu, zielony bramkarz i czarny piłkarz są w jednej drużynie

Rys. 4.1. Przejawy błędów detekcji branych pod uwagę w 4.2

wykrytego piłkarza do złego zespołu, wykrycie sędziego i innych postaci niebędących piłkarzami,

wykrycie pustej przestrzeni jako piłkarza, wykrycie jednego piłkarza w miejscu, w którym występuje ich więcej. Poza tymi błędami, algorytm czasami wykazuje tendencje do zamieniania kolorów między drużynami. To znaczy, że zawodnicy z drużyny białej stają się zawodnikami z drużyną czarną i odwrotnie. Przejaw błędów na materiale wyjściowym został przedstawiony na rys. 4.1, a wyniki przeprowadzonych testów zostały przedstawione na rysunku 4.2. Metodyka testowania sprowadzała się do manualnego policzenia piłkarzy występujących na obrazie i zidentyfikowania poprawnych oraz błędnych wykryć. Z powodu nie zautomatyzowanego charakteru testu, zliczanie odbywało się na podstawie 20 próbek klatek z testowanego materiału filmowego. Okres próbowania wyniósł 5 sekund, co oznacza, że zliczanie odbywało się dla klatki występującej co 5 sekund, aż do setnej sekundy filmu. Średnia skuteczność tej części algorytmu dla tych danych testowych wyniosła około 78,5%.



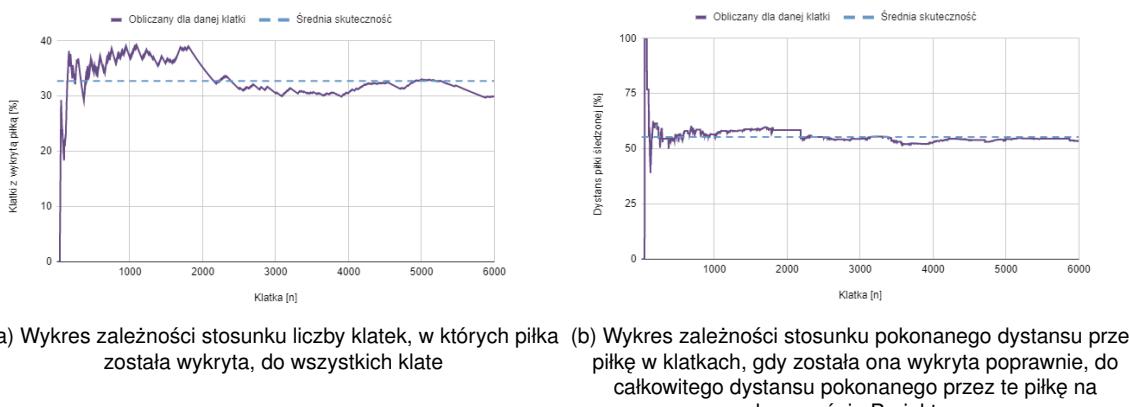
Rys. 4.2. Wykresy przedstawiające skuteczność algorytmu wykrywania piłkarzy w danym punkcie w czasie

Można zauważyć, że proponowane rozwiązanie cechuje się średniej klasy systemem wykrywania piłkarzy na obrazie. Algorytm nie radzi sobie z odróżnianiem piłkarzy od osób niebędących zawodnikami takich jak sędziowie, czy trenerzy. Błąd ten pojawia się zawsze, gdy taka osoba znajduje się obecnie na obrazie. Kolejnym problemem są kontakty fizyczne pomiędzy piłkarzami, które sprawiają, że są oni interpretowani jako jeden piłkarz. Źródłem kolejnych błędów są defekty pojawiające się podczas wstępnego przetwarzania zawodników. Prowadzi to do tego, że są one interpretowane jako piłkarz na obrazie. Ostatnim problemem jest to, że algorytm często nie wykrywa poprawnie jakiegoś zawodnika. Prowadzi to do przerwania ciągłości w jego śledzeniu oraz znacznie obniża skuteczność rozwiązania. Wad proponowanego algorytmu jest sporo, jednakże mimo tego, osiągnięta skuteczność jest zadowalająca i mogłaby być rozwijana w kolejnych iteracjach implementacyjnych.

## 4.2. Skuteczność wykrywania piłki na obrazie

Drugą, bardzo ważną metryką jest skuteczność wykrywania piłki na obrazie. Piłka na obrazie jest tylko jedna, a więc rodzaje błędów możliwych do popełnienia przez algorytm sprowadzają się do błędu braku wykrycia oraz błędu interpretacji pustej przestrzeni jako piłki (defekty). Z racji, że drugi z błędów nie występuje w materiale testowym, to metodyka sprawdzania efektywności tej części algorytmu polegała na zbadaniu stosunku liczby klatek, w których piłka została wykryta do liczby klatek, w których piłka nie została wykryta.

Dane wykorzystane do sporządzenia wykresu 4.3 (A) zostały przygotowane na podstawie 6 tysięcy klatek materiału testowego pobranych od 8 sekundy do 108 sekundy filmu. Metryka ta wskazuje na to, że algorytm słabo sobie radzi z wykrywaniem piłki. Jest to spowodowane tym, że piłka często jest przy nodze zawodnika, a to oznacza, że, w procesie przetwarzania, często tworzy z piłkarzami jeden, wielki kontur, który jest przez algorytm odrzucany ze względu na swój rozmiar. Jednakże, wykorzystując fakt, że kamera podąża za poruszającą się piłką i tym, że gdy algorytm nie wykrywa piłki to nie zmienia jej pozycji względem kamery, to niska skuteczność wykrywania piłki nie jest, aż tak drastycznie odczuwalna jak wskazywałaby na to zbadana metryka. Z tego powodu 4.3 (B) pokazuje dystans przebyty przez piłkę w trakcie jej śledzenia, w stosunku do całkowitego, pokonanego przez nią dystansu. Dystans pokonany przez piłkę jest wartością pobraną z Projektora, co znaczy, że zmieniał się nawet, gdy wykryta piłka nie zmieniała swojej pozycji względem kamery (sytuacja, w której piłka wykazuje ruch dokładnie taki jak ruch kamery). Średnia skuteczność obliczona na podstawie pierwszej metody dla tych danych testowych wynosi około 32.5%, a dla drugiej - około 55%.



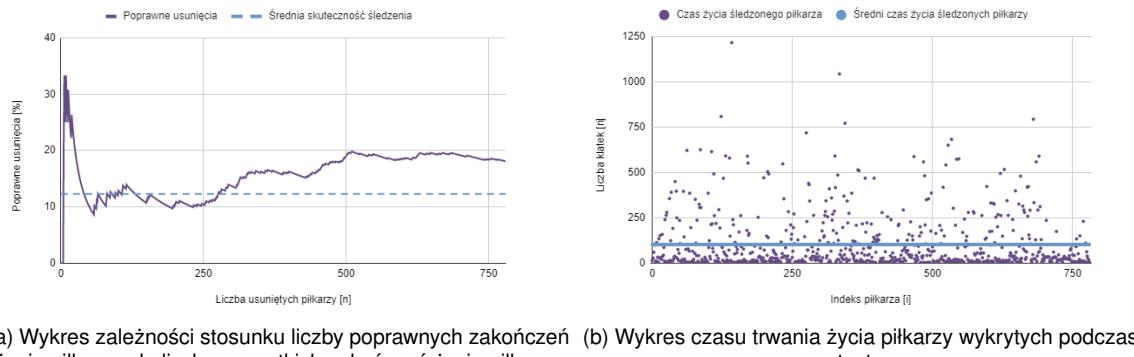
Rys. 4.3. Wykresy przedstawiające skuteczność algorytmu wykrywania piłki. Obliczone dla danych zbieranych do danego punktu w czasie

Proponowany algorytm wykrywania piłki nie jest wystarczająco efektywny. Piłka na obrazie jest często gubiona, co wpływa na wskaźniki jego jakości 4.3. Problemem jest to, że proces przetwarzania wstępnego piłki nie usuwa z maski z piłką ani linii boiska, ani piłkarzy, co implikuje sytuację, w której kontur piłki zlewa się z konturami linii boiska lub piłkarzy, co z kolei powoduje jej zgubienie. Nie jest to problem łatwy do rozwiązania ze względu na to, że piłka jest małym elementem na obrazie. Niewielkie stosowanie operacji morfologicznych mogłoby całkowicie zatracić informacje o niej w danej klatce. Należy zaznaczyć, że mimo słabych wskaźników jakościowych tej części algorytmu, ostateczny jego odbiór w materiale wyjściowym [32] nie jest, aż tak odczuwalny, co jest spowodowane tym, że kamera sama z siebie śledzi piłkę, a więc, gdy piłka jest

gubiona i później z powrotem wykrywana, zmiana pozycji nie jest, aż tak drastyczna. Może to napawać optymizmem w przypadku dalszych prac nad skutecznością rozwiązania.

### 4.3. Skuteczność śledzenia piłkarzy

Efektywność tej części algorytmu może zostać zmierzona na podstawie nieuzasadnionych zmian w identyfikatorach danych piłkarzy. Identyfikator co do zasady się nie zmienia, jednakże gdy algorytm zgubi danego zawodnika to usuwa go z pamięci, przy czym następnie jego wykrycie przypisuje mu inny identyfikator. Należy pamiętać, że zgubienie piłkarza ze względu na to, że nie znajduje się on już na obrazie z kamery, nie może być traktowane jako błąd. Błądem jest, gdy algorytm zgubi piłkarza, który nadal znajduje się na obrazie. Taką metrykę można przygotować śledząc cykl życia każdego z piłkarzy. Jeżeli dany zawodnik jest usuwany z pamięci, mimo że jego pozycja, w momencie jego usuwania, znajduje się w środku obrazu, a nie na jego obrzeżach, to oznacza, że piłkarz został zgubiony. Wykorzystując tę metrykę można również przygotować wyznacznik określający średni czas życia piłkarza w pamięci systemu.

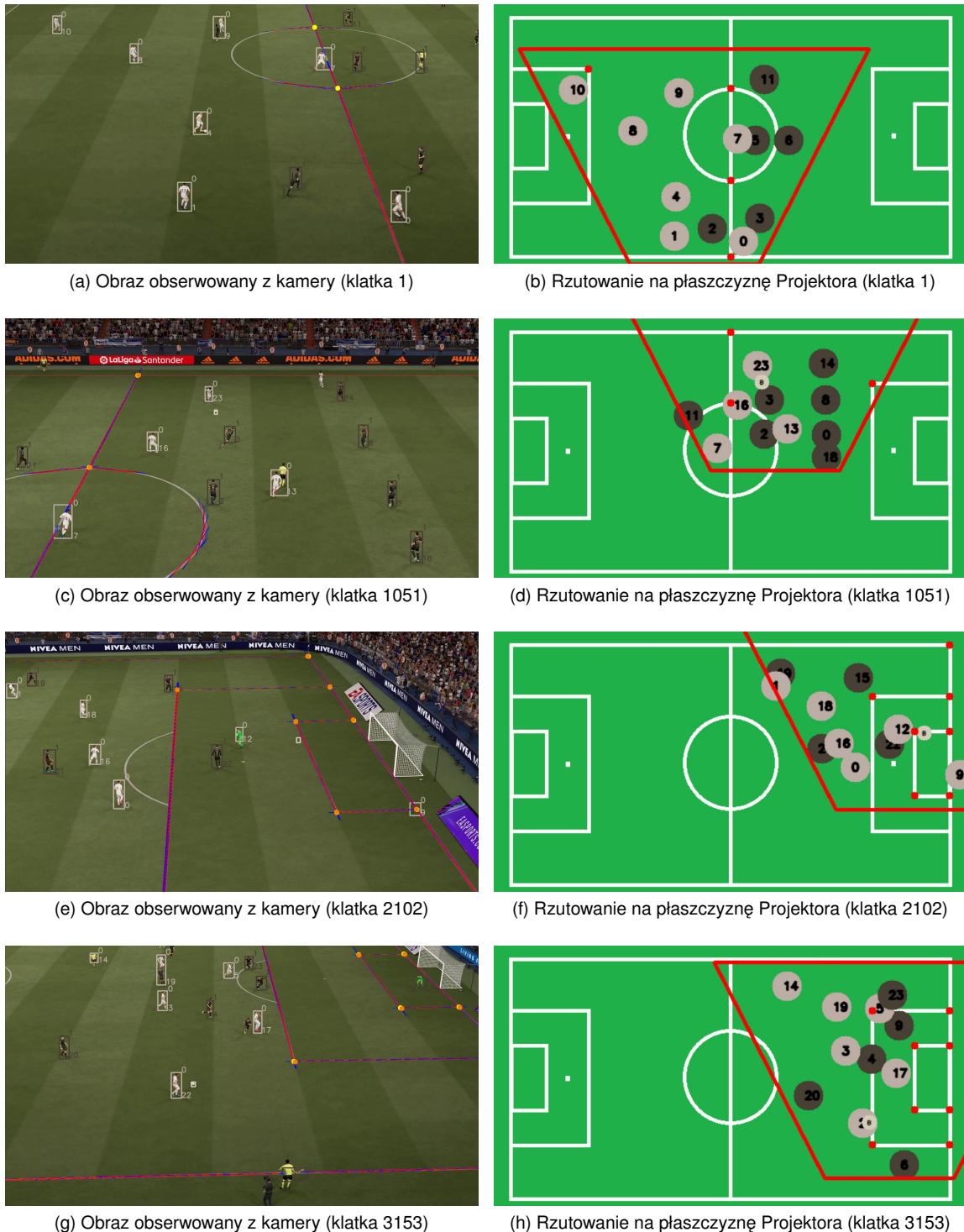


Rys. 4.4. Wykresy przedstawiające skuteczność algorytmu śledzenia piłkarzy

Można zauważyć, że skuteczność w śledzeniu piłkarzy jest bardzo niska, wynosi zaledwie 12%. W przeciągu 6 tysięcy klatek materiału, co przekłada się na 100 sekund filmu, wykrytych zostało 780 indywidualnych obiektów, z czego średni cykl życia każdego z nich wyniósł około 102 klatki, co przekłada się na 1,7 sekundy na filmie. Na tak niskie wyniki, składają się błędy w samym algorytmie śledzenia, ale również błędy algorytmu detekcji piłkarzy. Przykładem jest wpływ kontaktów fizycznych między zawodnikami, usuwający piłkarzy z pamięci i dodający nowego zawodnika z obwiednią obejmującą ich obu, który zostanie po chwili usunięty. Innym przykładem jest interpretacja przez algorytm pustych przestrzeni jako piłkarzy.

### 4.4. Dokładność projekcji

Na ostateczną dokładność projekcji obiektów wpływają wszystkie wcześniej opisane wyznaczniki jakości. Jednakże ignorując znikających, w trakcie trwania projekcji, piłkarzy oraz częste zmienianie się ich numerów identyfikacyjnych, a skupiając się jedynie na precyzyji odwzorowywania pozycji zawodników i piłki z przestrzeni trójwymiarowej na płaszczyznę 2D boiska, można odseparować dokładność projekcji od innych wyznaczników jakości. Jednakże jej zmierzenie w sposób automatyczny nie jest zadaniem trywialnym. Z tego powodu przeprowadzony został test manualny oparty o porównanie tego co jest obserwowane przez kamere z tym co jest wyświetlane na płaszczyźnie Projektora.

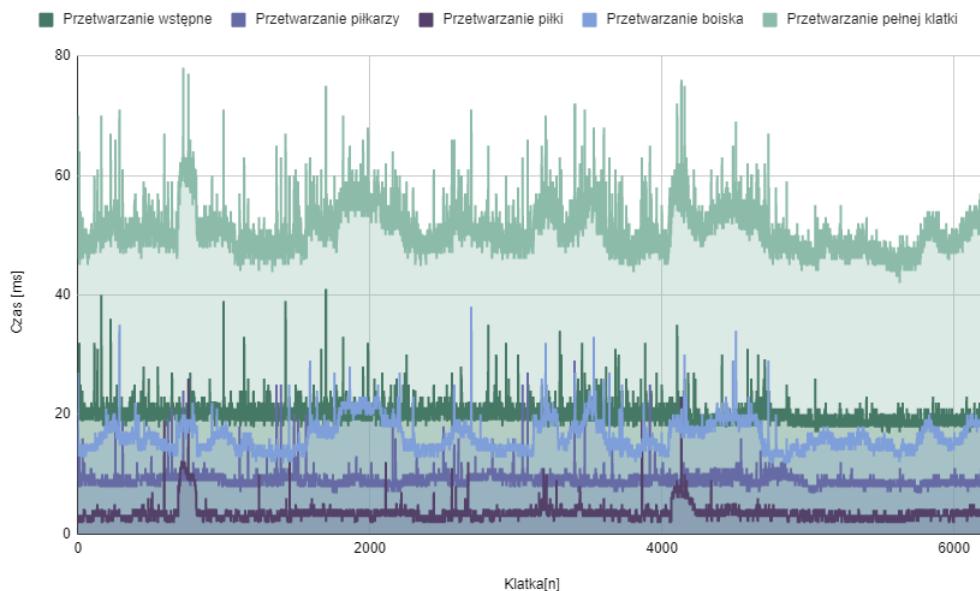


Rys. 4.5. Porównanie obrazów obserwowanych przez kamerę do ich projekcji na płaszczyźnie Projektor

Próbki z filmu pobierane były co 1050 klatek. Można zauważać, że pozycjonowanie mniejsze, więcej się zgadza. W 4.5 A oraz jej projekcji - B, widać że projekcja jest dobrze odwzorowana. Zawodnicy - zwłaszcza w środkowych częściach boiska są na podobnych pozycjach na obrazie i projekcji. Wyjątek stanowi zawodnik nr 10, który znajduje się zbyt daleko od gry. Na przykładzie 4.5 C i D można zwrócić uwagę na poprawną transformację perspektywy. Zawodnicy 14, 8, 0, 18 z drużyny o czarnych koszulkach znajdują się w jednej linii na płaszczyźnie Projekcji, mimo że ich współrzędne  $x$  na obrazie nie są jednakowe. W przypadku 4.5 E i F widać, że projekcja zaczyna się trochę rozjeżdzać. Ignorując błędy detekcji zawodników, piłkarze są wyświetlaniani zbyt wysoko na płaszczyźnie Projektora. Na rysunkach 4.5 G i H, piłkarze zaczynają się wyświetlać nie tylko za wysoko, ale również zbyt głęboko po prawej stronie. Na podstawie obserwacji można dojść do wniosku, że jakość projekcji może być zadowalająca, jednakże w przypadku dłuższych materiałów filmowych, może dojść do desynchronizacji pozycyjnej projekcji, względem tego co widoczne na obrazie z kamery. Dodatkowo, każde cięcie, czyli natychmiastowy przeskok kamery z jednego miejsca w drugie, kończy się desynchronizacją co jest widoczne na materiale wyjściowym [32] (czas 3:30-3:50).

#### 4.5. Testy wydajnościowe

Wyniki przedstawione w teście mogą się różnić od platformy sprzętowej. W przypadku metryk 4.6, testy zostały przeprowadzone na platformie z procesorem Intel Core I5 10400K z zegarem o częstotliwości 2,9GHz oraz kartą graficzną NVIDIA Geforce RTX 3060. Badają one czas trwania najważniejszych zespołów operacji przeprowadzonych w trakcie przetwarzania każdej klatki filmu. Weryfikowane zespoły operacji to: przetwarzanie wstępne, którego schemat przedstawiony jest w 3.4, algorytm wykrywania i śledzenia piłkarzy, algorytm wykrywania piłki oraz algorytm wykrywania punktów charakterystycznych do estymowania ruchu kamery. Dodatkowo przedstawiona została metryka ukazująca czas trwania przetwarzania jednej klatki filmu, w której skład wchodzą wszystkie operacje potrzebne do pełnego przetworzenia jednej klatki materiału testowego.



Rys. 4.6. Metryki przedstawiające czas potrzebny na wykonanie konkretnych zespołów operacji w czasie jednej klatki.

Można zauważyc, że najwięcej czasu zajmuje przetwarzanie wstępne. Wynika to z faktu, że przetwarzanie wstępne jest procesem wspólnym dla trzech strumieni przetwarzania: piłkarzy, piłki i boiska. W praktyce można byłoby podzielić ten czas na odpowiednie części i czas zajmowany przez nie dopisać do odpowiednich procesów. Przetwarzanie każdej klatki zajmuje średnio około 51ms, co sprawia, że w ciągu jednej sekundy możliwym jest przetworzenie 20 klatek co jest wynikiem zadowalającym. W tym czasie zużycie procesora sięga około 30%, a zużycie pamięci - około 170MB. Podgląd na metrykę 4.6 daje również obraz tego, nad czym należało popracować w celach optymalizacyjnych. Na pierwszy plan wyłania się proces przetwarzania wstępne oraz przetwarzania boiska w celu estymacji ruchu kamery.

## 5. PODSUMOWANIE

W niniejszej pracy przedstawiony został system wizyjny do analizy meczu piłkarskiego, wykrywający pozycje piłkarzy oraz piłki na boisku i rzutujący je na dwuwymiarową płaszczyznę boiska z widokiem z góry. W skutek projektu powstał algorytm odpowiedzialny za wykrywanie oraz śledzenie piłkarzy, algorytm odpowiedzialny za wykrywanie piłki, algorytm odpowiedzialny za estymację ruchu kamery, a także algorytm rzutujący wykryte pozycje obiektów na dwuwymiarową płaszczyznę boiska w regionie wyznaczonym przez globalną pozycję kamery w przestrzeni. Efektem systemu jest przetworzenie materiału wejściowego [31], którego wynikiem jest materiał wyjściowy [32].

Stworzenie takiego systemu od podstaw jest dużym wyzwaniem, gdyż do jego poprawnego funkcjonowania należy zadbać o najmniejsze szczegóły. Skuteczności algorytmów wykrywania determinują skuteczność algorytmów śledzenia, które z kolei wpływają na końcową precyzję projekcji. Wadą niniejszego rozwiązania są problemy z wykrywaniem obiektów na obrazie, w tym liczne błędy detekcji takie jak brak wykrycia danego piłkarza, czy przykłady fałszywych wykryć, do których można zaliczyć m.in. interpretowanie pustej przestrzeni jako piłkarza i rysowanie w tym miejscu obwiedni. Zazwyczaj miało to miejsce w okolicach punktów przecięcia się linii boiska, co może świadczyć o niedokładnym wykorzystaniu możliwości algorytmów progowania. Błędy te determinują średnią skuteczność wykrywania obiektów: 78% skuteczności detekcji zawodników i 55% skuteczności detekcji piłki.

Zaletą rozwiązania jest stosowanie systemów zapobiegawczych, które są w stanie zwiększyć skuteczność algorytmów wykrywania jak i algorytmów śledzenia. Jednym z takich systemów jest system usuwania piłkarzy z pamięci, dopiero gdy nie zostają oni wykryci przez 5 kolejnych klatek filmu. Pozwala to na uniknięcie sytuacji, w której drobny błąd przetwarzania w jednej, losowej klatce, skutkuje zgubieniem danego piłkarza i usunięciem go z pamięci. Innym przykładem środka zapobiegawczego jest sposób wykrywania piłki. Ze względu na to, że kamera porusza się za piłką to ruch ten niweluje relatywny ruch piłki na obrazie. Z tego powodu zastosowanym rozwiązaniem w przypadku zgubienia piłki z obrazu jest brak zmian w jej pozycji, relatywnie do kamery. To sprawia, że w praktyce, gdy piłka nie jest wykrywana to na projekcji porusza się ona zgodnie z ruchem kamery, a więc błąd wprowadzony przez problemy z wykryciem jest niwelowany do minimum, dzięki czemu ostateczne odczucia w stosunku do algorytmu wykrywania piłki, nie są aż tak złe jak wskazywałyby na to zmierzona 55% skuteczność algorytmu.

Wadą i zaletą w proponowanym rozwiązaniu jest zastosowanie aproksymacji widoku przestrzennego w trójwymiarze, za pomocą dwuwymiarowego trapezu. Wadą jest problem z dopasowaniem odpowiednich parametrów dla trapezu, żeby projekcja była maksymalnie dokładna oraz to, że po pewnym czasie projekcja rozjeżdża się względem obrazu z kamery. Natomiast zaletą jest dużo niższa złożoność obliczeniowa rozwiązania niż w przypadku dokładniejszych metod projekcji.

System ma potencjał, jednakże wiele elementów należałoby przeprojektować wykorzystując doświadczenie zebrane w niniejszym projekcie. Jednym z usprawnień mogłoby być wykorzystanie sieci neuronowych i algorytmów takich jak YOLO do wykrywania obiektów na obrazie. Mogłoby to zwiększyć skuteczność wykrywania, której mniejsza liczba błędów, miałaby zdecydowanie mniejszy wpływ na pozostałe moduły systemu. Wadą mogłoby być większe zużycie procesora oraz

pamięci, jednakże w takim przypadku można byłoby zastanowić się nad wykorzystaniem procesora graficznego GPU, w celu zwiększenia szybkości przetwarzania. Innym przykładem usprawnienia mogłoby być przepisanie programu na język C++, który jest znacznie szybszy od Pythona. Tak skonstruowany system, wraz z wykorzystaniem GPU, miałby potencjał na działanie w czasie rzeczywistym przy zachowaniu wysokiej skuteczności wykrywania. Wadą jest koszt czasowy, który trzeba ponieść na przygotowanie oraz wytrenowanie modelu sieci neuronowej, a także przepisanie rozwiązania na język niższego poziomu. Dodatkowym kosztem jest cena procesorów graficznych GPU, które mogłyby wykluczyć możliwość stosowania takiego systemu w ligach regionalnych i amatorskich. Odpowiednio wydajna podstawa systemu analizy umożliwiłaby dalszy rozwój rozwiązania w celu pełnej automatyzacji statystyk konkretnych sportowców, w tym mapy ciepła, czyli wyznaczone przestrzenie, w których piłkarze poruszali się najczęściej, czy statystyki podań, strzałów, dryblingów, przebiegniętych kilometrów i posiadania piłki.

## WYKAZ LITERATURY

- [1] Beetz M., von Hoyningen-Huene N., Kirchlechner B., Gedikli S., Siles F., Durus M., Lames M.: *ASPO-GAMO: Automated Sports Game Analysis Models*, Intelligent Autonomous Systems Group, Technische Universität München, Boltzmannstr. 3, 85748 Garching b. München, Niemcy
- [2] Blauberger P., Marzilger R., Lames M.: *Validation of Player and Ball Tracking with a Local Positioning System*, Sensors 21(4):1465, Luty 2021
- [3] Naushad Ali M. M., Abdullah-Al-Wadud M., Seok-Lyong L.: *An Efficient Algorithm for Detection of Soccer Ball and Players*, Konferencja: SIP (Signal Processing Image Processing and Pattern Recognition), wyspa Jeju, Korea Południowa 2012
- [4] Jong-Yun K., Tae-Yong K.: *Soccer Ball Tracking using Dynamic Kalman Filter with Velocity Control*, Sixth International Conference on Computer Graphics, Imaging and Visualization, 2009
- [5] Sun P., Zhao X., Zhao Y., Jia N., Cao D.: *Intelligent Optimization Algorithm of 3D Tracking Technology in Football Player Moving Image Analysis*, Wireless Communications and Mobile Computing, 13 stron, 2022
- [6] Laborda M. A. M., Moren E. F. T., Martinez-del-Rincon J., Jaraba E. H.: *Real-time GPU color-based segmentation of football players*, Journal of Real-Time Image Processing 7(4):1-13, grudzień 2011
- [7] Csanalosi G. , Dobreff G., Pasic A., Molnar M., Toka L.: *Low-Cost Optical Tracking of Soccer Players*, Machine Learning and Data Mining for Sports Analytics (pp.28-39), 2020
- [8] Dearden A., Demiris Y., Grau O.: *Tracking football player movement from a single moving camera using particle filters*, BBC Research, styczeń 2006
- [9] Nikos G., Dimitropoulos K., Karaman M.: *Football player tracking from multiple views: Using a novel background segmentation algorithm and multiple hypothesis tracking*, Konferencja: VISAPP 2007: Proceedings of the Second International Conference on Computer Vision Theory and Applications, numer rocznika 2, Barcelona, Hiszpania, Marzec 8-11, 2007
- [10] Huang P., Hilton A.: *Football Player Tracking for Video Annotation*, Konferencja: Visual Media Production, 2006. CVMP 2006. 3rd European Conference on, 2006
- [11] Mackowiak S., Konieczny J., Kurc M., Maćkowiak P.: *Football player detection in video broadcast*, Konferencja: Computer Vision and Graphics - International Conference, ICCVG 2010, Warszawa, Polska, Wrzesień 20-22, 2010, Obrady, Część II
- [12] Li C., Peng Q.: *Multitarget Tracking Algorithm in Intelligent Analysis of Football Movement Training Stance*, Security and Communication Networks 2022:1-8, sierpień 2022
- [13] Sullivan J., Nillius P., Carlsson S.: *Multi-target Tracking on a Large Scale: Experiences from Football Player Tracking*, Proceedings of the IEEE ICRA 2009 Workshop on People Detection and Tracking Kobe, Japonia, maj 2009
- [14] Dokumentacja biblioteki OpenCV do przetwarzania obrazu [online]. [dostęp 10.11.2022]. Dostępny w Internecie: <https://docs.opencv.org/4.x/index.html>
- [15] BetOnChart: *Effective way to collect football data* [online]. Medium. [dostęp 20.09.2022]. Dostępny w Internecie: <https://medium.com/betonchart/effective-way-to-collect-soccer-data-70ef69182eba>

- [16] Sahoo P., Soltani S., Wong A.: *A Survey of Thresholding Techniques*, Computer Vision Graphics and Image Processing 41(2):233-260, Luty 1988
- [17] Joblove G. H., Greenberg D.: *Color spaces for computer graphics*, ACM SIGGRAPH Computer Graphics numer rocznika 12(3), pp 20–25, sierpień 1978
- [18] Hendriks C. L. L., van Vliet L. J.: *Basic Morphological Operations, Band-Limited Images and Sampling*, artykuł konferencyjny, styczeń 2003
- [19] Suzuki S., Abe K.: *Topological structural analysis of digitized binary images by border following*, CVGIP 30 1, pp 32-46, 1985
- [20] Twierdzenie i wzór Greena [online]. Wikipedia: wolna encyklopedia. [dostęp 13.11.2022]. Dostępny w Internecie: [https://en.wikipedia.org/wiki/Green%27s\\_theorem](https://en.wikipedia.org/wiki/Green%27s_theorem)
- [21] Yang A.: *The Maths behind Contour Moments from OpenCV* [online]. Medium. [dostęp 15.11.2022]. Dostępny w Internecie: <https://medium.com/@aleozlx/the-maths-behind-contour-moments-from-opencv-491e5c348b91>
- [22] Hough P.V.C.: *Method and means for recognizing complex patterns*, Patent 3,069,654, Stany Zjednoczone, grudzień 1962
- [23] Kiryati N., Eldar Y., Bruckstein A. M.: *A probabilistic Hough transform*, Pattern Recognition, numer rocznika 24(4), strony 303-316, 1991
- [24] Iloczyn wektorowy i orientacja punktu względem wektora [online]. Polsko-Japońska Akademia Technik Komputerowych. [dostęp 15.11.2022]. Dostępny w Internecie: [https://edu.pjwstk.edu.pl/wyklady/asd/scb/asd13/main13\\_p2.html](https://edu.pjwstk.edu.pl/wyklady/asd/scb/asd13/main13_p2.html)
- [25] How to check if two given line segments intersect? [online]. Geeksforgeeks. [dostęp 16.11.2022]. Dostępny w Internecie: <https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/>
- [26] Odometria wizyjna [online]. Wikipedia: wolna encyklopedia. [dostęp 17.11.2022]. Dostępny w Internecie: [https://en.wikipedia.org/wiki/Visual\\_odometry](https://en.wikipedia.org/wiki/Visual_odometry)
- [27] Horn B. K. P., Schunck B. G.: *Determining Optical Flow*, Artificial Intelligence 17(1-3):185-203, sierpień 1981
- [28] Luong Q., Faugeras O. D.: *The Fundamental Matrix: Theory, Algorithms, and Stability Analysis*, International Journal of Computer Vision 17(1):43-75, styczeń 1996,
- [29] Współrzędne jednorodne [online]. Wikipedia: wolna encyklopedia. [dostęp 19.11.2022]. Dostępny w Internecie: [https://en.wikipedia.org/wiki/Homogeneous\\_coordinates](https://en.wikipedia.org/wiki/Homogeneous_coordinates)
- [30] Heckbert P.: *Projective Mappings for Image Warping*, 15-869, Image-Based Modeling and Rendering, wrzesień 1999
- [31] Plesiak K.: Materiał testowy [online]. Youtube. [dostęp 25.11.2022]. Dostępny w Internecie: <https://youtu.be/OJPbMIm43dY>
- [32] Plesiak K.: Materiał wyjściowy systemu [online]. Youtube. [dostęp 25.11.2022]. Dostępny w Internecie: <https://www.youtube.com/watch?v=y0FiBBBT6Cg&t=2s>
- [33] Plesiak K.: Kod źródłowy projektu [online]. Github. [dostęp 25.11.2022]. Dostępny w Internecie: <https://github.com/CamaroTheBOSS/Football-match-vision-system-engineering-project>
- [34] Plesiak K.: Materiał testowy do pobrania [online]. Google Drive. [dostęp 25.11.2022]. Dostępny w Internecie: <https://drive.google.com/drive/folders/1KylqkDRz4WniBonhIreFMLslozPVywCC?usp=sharing>

## WYKAZ RYSUNKÓW

2.1 Przykład wykrycia piłkarzy będących w bliskim kontakcie fizycznym . . . . .	8
3.1 Schemat blokowy architektury programu . . . . .	9
3.2 Kolory a,c,d nie są odróżniane przez zależność 3.1, kolory a i b są odróżniane przez zależność 3.1 . . . . .	11
3.3 Schemat blokowy przetwarzania wstępnego . . . . .	12
3.4 Rezultaty przetwarzania wstępnego w każdym ze strumieni . . . . .	13
3.5 Rezultaty wycięcia sylwetek piłkarzy z rejonów wyznaczonych w trakcie przetwarzania wstępnego . . . . .	15
3.6 Czarne i białe prostokąty to obwiednie $D_e$ otaczające piłkarzy, a czerwone to obwiednie wewnętrzne $D_i$ wyznaczone na podstawie zależności 3.7 . . . . .	16
3.7 Rezultaty algorytmu progowania w detekcji linii boiska . . . . .	17
3.8 Sposób reprezentacji prostej kierunkowej $y = ax + b$ w postaci normalnej $x \cos(\theta) + y \sin(\theta) = \rho$ . . . . .	18
3.9 Czerwone odcinki to linie wykryte przez probabilistyczną transformate Hougha, niebieskie odcinki to wydłużona forma odcinków czerwonych, a żółte punkty to zagregowane punkty charakterystyczne . . . . .	19
3.10 Czerwone ramy na obrazach pokazują perspektywę 3D odpowiednio na obrazie z kamery i na rzutowaniu obrazu na płaszczyznę 2D. Czarne ramy pokazują perspektywę ukazującą perspektywę 2D odpowiednio na obrazie z kamery i na płaszczyźnie 2D. Proporcje zostały zachowane. . . . .	21
4.1 Przejawy błędów detekcji branych pod uwagę w 4.2 . . . . .	23
4.2 Wykresy przedstawiające skuteczność algorytmu wykrywania piłkarzy w danym punkcie w czasie . . . . .	24
4.3 Wykresy przedstawiające skuteczność algorytmu wykrywania piłki. Obliczone dla danych zbieranych do danego punktu w czasie . . . . .	25
4.4 Wykresy przedstawiające skuteczność algorytmu śledzenia piłkarzy . . . . .	26
4.5 Porównanie obrazów obserwowanych przez kamerę do ich projekcji na płaszczyźnie Projektora . . . . .	27
4.6 Metryki przedstawiające czas potrzebny na wykonanie konkretnych zespołów operacji w czasie jednej klatki. . . . .	28

## DODATEK A: INSTRUKCJA DLA UŻYTKOWNIKA

Aby skorzystać i własnoręcznie przetestować system należy odpowiednio przygotować swoje środowisko komputerowe, aby spełnić zależności programu. Warunkiem wstępny jest posiadanie zainstalowanego języka Python w wersji co najmniej 3.10 oraz działający edytor tekowy.

Spełniając warunki wstępne można przystąpić do pobrania kodu źródłego programu. Posiadając zainstalowanego Gita (system kontroli wersji) na komputerze, można to zrobić za pomocą protokołu HTTPS wpisując odpowiednie komendy w terminalu:

```
cd ./sciezka/do/folderu/docegowego  
git clone https://github.com/CamaroTheBOSS/Football-match-vision-system-  
engineering-project.git
```

Listing 5.1: Komendy, które należy wpisać, aby pobrać projekt za pomocą Gita i protokołu HTTPS

W przypadku nieposiadania zainstalowanego systemu Git należy ręcznie pobrać projekt z repozytorium [33], klikając na *Code->Download ZIP*. Następnie należy go rozpakować w wybranym przez siebie folderze.

Po rozpakowaniu pobranego pliku, należy stworzyć wirtualne środowisko. Aby to zrobić trzeba otworzyć terminal w folderze z projektem i wykonać następujące komendy:

```
cd ./sciezka/do/folderu/z/projektem  
python -m venv venv  
venv\Scripts\activate.bat  
pip install opencv-contrib-python==4.5.5.64  
pip install keyboard
```

Listing 5.2: Komendy, które należy wpisać w konsoli CMD w systemie Windows, aby stworzyć wirtualne środowisko

Kolejnym krokiem jest stworzenie w ścieżce z projektem folderu o nazwie "video". Należy w nim umieścić film w formacie mp4 lub mkv, który ma zostać przetworzony. Może to być np. film wykorzystany do testów [34] lub inny. Następnie należy wejść w folder *src* i utworzyć plik *Config.py* w dowolnym edytorze tekowym. Trzeba w nim podmienić wartość *VIDEO* na, ujętą w cudzysłów, nazwę pliku, który ma być przetworzony wraz z jego rozszerzeniem np. *VIDEO="fifa.mkv"*.

Następnie należy przygotować odpowiednią konfigurację, dodając do pozycji *CONFIG* kolejną opcję uwzględniającą kilka wartości. Pierwszą z nich są granice wykorzystywane w algorytmach progowania - w postaci trzech wartości HSL dla dolnej granicy i trzech wartości HSL dla górnej granicy progowania. Algorytm progowania wykorzystywany jest w dwóch procesach: w procesie wycinania tła oraz procesie wycinania linii boiska, stąd granice należy skonfigurować dla obu tych procesów. Drugą wartością jest czas odstępu między generacją kolejnych klatek. Zwiększenie tej wartości jest wskazane w przypadku filmów z mniejszą rozdzielcością, gdyż czas przetwarzania takiego materiału znaczco spada. Ostatnią wartością, którą należy skonfigurować jest docelowy kolor wykrywanej piłki dany w postaci trzech wartości RGB. Odpowiednia wartość jest w stanie zapewnić lepszą skuteczność wykrywania piłki. Poza opcjami konfiguracji algorytmu, możliwe jest zdefiniowanie kolorów obwiedni jakie mają być wyświetlane podczas przetwarzania. Zmienna *use\_display\_colors* ustawiona na wartość False (z ang. fałsz) sprawi, że

dobór kolorów obwiedni odbywać się będzie automatycznie na podstawie kolorów koszulek piłkarzy. Ustawienie tej wartości na True (a ang. prawda) sprawi, że obwiednie będą generowane w kolorze zdefiniowanym przez zmienną *TEAM\_COLORS*. Aby zmienić te kolory należy odpowiednio zmodyfikować *TEAM\_COLORS*, aktualizując wartości RGB. Zwykle wystarczy zmodyfikować pierwsze dwa zespoły wartości na liście: pierwszy dla koloru obwiedni drużyny gospodarzy, a drugi dla drużyny gości. Ostatnią wagą uwagi zmienną jest *output\_file*, która definiuje nazwę oraz rozszerzenie pliku wyjściowego. Aby plik wyjściowy został przygotowany, należy ją zmodyfikować do postaci *output\_file="<nazwa pliku>.mp4"*.

Po zapisaniu zmian w pliku konfiguracyjnym, można przystąpić do uruchomienia programu poprzezłączenie terminalu w folderze *src*, znajdującego się w ścieżce z projektem i wykonanie polecenia:

```
cd ./sciezka/do/folderu/z/projektem/src  
python -m main
```

Listing 5.3: Komendy, które należy wpisać w konsoli CMD w systemie Windows, aby uruchomić program

Po uruchomieniu programu, na ekranie powinny pojawić się dwa okienka. Pierwsze - z obecnie przetwarzanym materiałem testowym, z naniesionymi na niego nakładkami graficznymi takimi jak obwiednie wokół piłkarzy, czy zaznaczone wykrywane linie. Drugim oknem jest okno z aktualną projekcją pozycji piłkarzy na dwuwymiarowej płaszczyźnie boiska. Czerwony trapez na projekcji przedstawia aktualnie obserwowany przez kamerę region boiska, a czerwone punkty to punkty charakterystyczne, które kamera aktualnie widzi. Materiał jest przetwarzany na bieżąco, co sprawia, że w przypadku wolniejszych jednostek centralnych CPU, wyświetlanie może nie być w pełni płynne.

Jeżeli program został skonfigurowany w taki sposób, aby przygotować plik wyjściowy to można go znaleźć w ścieżce z projektem w folderze *output*, powstały na skutek wykonania się całego przetwarzania. Można tam znaleźć trzy filmy: obraz z kamery z nakładkami graficznymi, obraz z projekcją oraz połączony obraz z kamery wraz z obrazem z projekcji na jednym filmie. Aby wyjść z programu w trakcie przetwarzania filmu należy wcisnąć przycisk Q na klawiaturze.

## DODATEK B: INSTRUKCJA DLA PROGRAMISTY

Aby pobrać kod źródłowy i uruchomić program należy podążać za instrukcjami opisanymi w Dodatku A. Następnie należy otworzyć projekt w skonfigurowanym przez siebie środowisku programistycznym takim jak Pycharm, czy Visual Studio Code.

W pliku *main.py* znajduje się główna klasa *Main* oraz pętla programu. W konstruktorze tej klasy zdefiniowane są wszystkie obiekty biorące udział w przetwarzaniu takie jak Dystrybutor Klatek, czy Detektor Obiektów. W metodzie *loop()* wykonywane są wszystkie instrukcje dla każdej klatki przetwarzanego filmu.

*FrameDistributor.py* to plik, który obejmuje cały Dystrybutor Klatek. Wykonywane w nim są instrukcje odpowiedzialne za wycinanie tła, wycinanie obiektów oraz rozsyłanie przetworzonych klatek do Detektora Obiektów i Modułu Śledzenia Kamery. W pliku tym można ulepszyć algorytm przetwarzania wstępnego rozwiązania.

*ObjectsDetector.py* to plik, obejmujący cały Detektor Obiektów. Wykonywane są w nim instrukcje odpowiedzialne za wykrywanie konturów obiektów, które są następnie interpretowane w Menadżerze Obiektów. Wśród metod tej klasy można znaleźć odseparowane od siebie metody odpowiedzialne za wykrywanie piłkarzy i piłki. Dodatkowo są tutaj przygotowywane odpowiednie wartości dla cech kandydatów: pozycja i rozmiar obwiedni oraz średni kolor wewnętrz obwiedni.

*FootballManager.py* to plik, który obejmuje cały Menadżer Obiektów. Znajdują się w nim komparatory pozycji, a także kolorów, które są wykorzystywane do dopasowywania piłkarzy wykrytych w klatce bieżącej do piłkarzy wykrytych w klatce poprzedniej. W konstruktorze tej klasy można również zmodyfikować maksymalną liczbę wykrywanych zawodników oraz drużyn. Możliwe jest to także podczas tworzenia obiektu Menadżera Obiektów w klasie głównej *Main*. Menadżer Obiektów posiada również metody odpowiedzialne za dodawanie i usuwanie piłkarzy z pamięci programu.

*Objects.py* to plik, zawierający w sobie definicje wszystkich wykrywanych obiektów. Klasy takie jak *Object* czy *MovingObject* to warstwy abstrakcji, które narzucają formę wszystkich pozostałych obiektów takich jak *Candidate*, *Footballer*, *Ball*. Metoda *track()* wykonywana jest, gdy dany obiekt, który został wykryty w bieżącej klatce, może być dopasowany do obiektu wykrytego w klatce poprzedniej. Metoda *update()* wykonywana jest po każdej klatce programu, a metoda *project()* wykorzystywana jest do projekcji na dwuwymiarową płaszczyznę boiska. Dodatkowo istnieje metoda *draw()* odpowiedzialna za rysowanie obwiedni obiektów na przetwarzanym materiale. Poza tym, znaleźć tu można klasę *Team*, która definiuje strukturę zespołu. Wśród najważniejszych informacji jakie należałyby wymienić jest to, że każdy obiekt klasy *Footballer* posiada atrybut determinujący obiekt klasy *Team*, do którego należy, a każdy obiekt klasy *Team* posiada listę z obiektami klasy *Footballer*, którzy są członkami danego zespołu.

*CameraTracker.py* to plik, obejmujący Moduł Śledzenia Kamery. Można w nim znaleźć algorytm odpowiedzialny za wykrywanie linii boiska oraz punktów charakterystycznych, na podstawie których estymowany jest ruch kamery w przestrzeni. Ruch ten przesyłany jest do Modelu Kamery, znajdującego się w pliku *Camera.py*, który zawiera w sobie Model Kamery. Model Kamery zamienia ruch o współrzędnych globalnych na ruch o współrzędnych na płaszczyźnie Projektora, a także znajduje się w nim algorytm transformacji współrzędnych obiektów na obrazie do współ-

rzędnych w Projektorze. W Module Śledzenia Kamery można ulepszać algorytm wykrywania linii boiska, a także algorytm estymacji pozycji kamery w przestrzeni.

*FootballProjector.py* to plik, zawierający w sobie cały Projektor. Metoda *project\_coordinates()* przekształca współrzędne z obrazu na współrzędne na płaszczyźnie Projektora, wykorzystując do tego globalną pozycję kamery. Metoda *show()* wyświetla projekcję użytkownikowi.

Poza plikami z najważniejszymi klasami, w projekcie znajdują się pliki pomocnicze: *Config.py*, w którym można dodawać dodatkowe opcje konfiguracyjne do programu, *Colors.py*, w którym zdefiniowane są kolory tekstu, wykorzystywane do czytelniejszego wyświetlenia danych programu, *Line.py* zawierający definicję klasy *Line*, która jest obiektem linii wykrywanej przez transformatora Hougha, a także *utils.py*, czyli plik z funkcjami pomocniczymi.

## DODATEK C: OPIS DYPLOMU

### C.1. Tytuł dyplomu

System wizyjny do analizy meczów piłki nożnej.

### C.2. Cel i przeznaczenie aplikacji

Celem projektu jest stworzenie algorytmu, wykrywającego pozycje piłkarzy oraz piłki, z możliwością identyfikacji konkretnych zawodników poprzez śledzenie ich przez cały czas trwania algorytmu, a następnie rzutowanie ich pozycji na dwuwymiarowej płaszczyźnie boiska z widokiem z góry. Założeniem jest działanie systemu w środowisku z kamerą poruszającą się jedynie w osiach X i Y, bez rotacji i ruchu w osi Z (góra, dół) oraz bez cięć, czyli natychmiastowych przeskóków kamery z jednego miejsca w przestrzeni na drugie, a także bez dodatkowych nakładek graficznych takich jak logo stacji telewizyjnej, czy panel z wynikiem meczu. Z tego powodu działanie rozwiązania jest ograniczone do środowiska gry FIFA, która umożliwia spełnienie tych założeń w stopniu prawie kompletnym. Drobna rotacja kamery oraz ruch w osi Z w tym środowisku nadal występuje, jednakże jest to akceptowalne. Poza środowiskiem gry FIFA, system może działać w środowiskach z kamerą obserwującą mecze lig regionalnych półprofesjonalnych lub amatorskich, w których kamera ustawniona jest na sztywnym, nieporuszającym się statywie. Przeznaczeniem projektu jest stworzenie podstawy systemu wizyjnego do analizy meczów piłkarskich, który można byłoby rozwijać w kolejnych iteracjach implementacyjnych, zwiększając skuteczność rozwiązania, a także dodając dodatkowe funkcje.

### C.3. Funkcjonalność

#### C.3.1. Opis realizowanych funkcji

Program przetwarza każdą klatkę filmu wejściowego i prezentuje jego wyniki w dwóch osobnych oknach. Pierwsze z nich pokazuje sposób i jakość wykrywania elementów na boisku, prezentując klatki z filmu wejściowego, rozszerzone o dodatkowe nakładki graficzne na:

- piłkę - w postaci kolorowej obwiedni wokół wykrytej piłki,
- piłkarzy - w postaci kolorowej obwiedni wokół danego piłkarza wraz z jego identyfikatorem i indeksem drużyny do której należy,
- linie boiska - pokolorowane, w celu odróżnienia linii, które program wykrywa od linii, których nie wykrywa,
- przecięcia linii boiska - w postaci punktów, z których żółte to punkty wykryte w klatce bieżącej, a pomarańczowe to punkty wykryte w klatce poprzedniej.

W drugim oknie wyświetla się projekcja pozycji piłkarzy, piłki oraz obszaru obserwowanego przez kamerę na dwuwymiarową płaszczyznę boiska z widokiem z góry. Piłkarze wyświetlani są w postaci dużych krążków w kolorach identyfikujących zespół, do którego należą. Dodatkowo w krążku znajduje się numer identyfikacyjny danego zawodnika. Piłka wyświetlana jest w postaci małego krążka, a obszar obserwowany przez kamerę - w postaci czerwonego trapeza. Dodatkowo, w programie istnieje możliwość konfiguracji algorytmu przetwarzania, zmieniając konkretne jego para-

metry, a także definicja pliku wyjściowego, która pozwala na zapisanie widoków z obu okien w postaci plików wideo z rozszerzeniem mp4 z wyłączeniem dodatkowego pliku wideo, który łączy w sobie oba widoki na jednym materiale.

#### *C.3.2. Lista przykładowych zastosowań*

Proponowany system może zostać wykorzystany do:

- analizy funkcjonowania konkretnej drużyny w ligach półprofesjonalnych i amatorskich
- dostarczenia dodatkowej rozrywki kibicom w amatorskich i półprofesjonalnych ligach szóstej piłkarskich, którzy mogliby, dzięki temu systemowi, obserwować mecz z innej perspektywy,
- stworzenia potężniejszego, bardziej rozbudowanego systemu, bazującego na proponowanym rozwiązaniu, który w analizie uwzględniałby statystyki konkretnych zawodników takie jak: skuteczność podań, czy skuteczność strzałów.

### **C.4. Ogólna architektura systemu**

System oparty jest na trzech modułach funkcjonalnych: module odpowiedzialnym za detekcję obiektów na obrazie, a także identyfikację konkretnych zawodników, module odpowiedzialnym za estymację ruchu kamery w przestrzeni oraz module odpowiedzialnym za wyświetlenie piłkarzy i piłki na dwuwymiarowej płaszczyźnie boiska, wykorzystując do tego globalną pozycję kamery. Program został napisany w języku Python z wykorzystaniem biblioteki OpenCV, umożliwiającej wysokopoziomowe korzystanie algorytmów przetwarzania obrazu.

### **C.5. Opis sposobu wytwarzania aplikacji**

Pierwszym krokiem było zbadanie istniejących rozwiązań w literaturze i Internecie, na podstawie których tworzone były pierwsze prototypy aplikacji, skupiające się na module odpowiedzialnym za wykrywanie ruchu kamery. Na tym etapie powstało założenie dotyczące kamery (brak rotacji i ruchu w osi Z). Następnym krokiem było przygotowanie algorytmu wykrywającego obiekty na obrazie. Przetestowane zostały możliwości, które dają algorytmy oparte o metody sztucznej inteligencji, takie jak YOLO, który został skonfrontowany z algorytmami deterministycznymi. Ze względu na zbadaną wydajność została podjęta ścieżka algorytmów deterministycznych. W kolejnym kroku stworzony został algorytm śledzenia zawodników umożliwiający identyfikację konkretnych piłkarzy. Ukończenie tej części systemu pozwoliło na połączenie całości rozwiązania i wykorzystanie wszystkich zdobytych przez system informacji w celu wyświetlenia pozycji zawodników, piłki oraz obszaru obserwowanego przez kamerę na dwuwymiarowej płaszczyźnie boiska z widokiem z góry. Ostatnim krokiem było przetestowanie rozwiązania i opis wyników projektu. Organizacyjnie, wyznaczone zostały dni pracy nad projektem ze sztywnym utrzymaniem narzuconego przez siebie harmonogramu na kolejne tygodnie pracy.

### **C.6. Ocena aplikacji**

System jest wydajny. Przy wykorzystaniu lepszych jednostek centralnych można osiągnąć działanie w czasie rzeczywistym, jednakże skuteczność jest niewystarczająca. Testy wskazują na 55% skuteczności algorytmu wykrywania piłki oraz 78% skuteczności algorytmu wykrywania piłkarzy, co przekłada się na słabą, 12% skuteczność śledzenia piłkarzy. Wynikają z tego licz-

ne błędy detekcji takie jak: błąd braku detekcji piłkarza, błąd wykrycia sędziego lub innych osób niebędących zawodnikami, czy błąd wykrycia pustej przestrzeni jako piłkarza oraz błędy śledzenia polegające na gubieniu konkretnych piłkarzy na obrazie, co skutkuje niskim, wynoszącym 1,7 sekundy, średnim czasem życia konkretnych obiektów w pamięci programu. Dodatkowo ruch kamery jest estymowany z narastającym błędem, co może skutkować rozjedzaniem się projekcji względem tego, co widoczne na obrazie. System nie jest skuteczniejszy od rozwiązań z literatury, wykorzystujących metody sztucznej inteligencji, co może nasunąć wniosek, że w następnych iteracjach implementacyjnych należałoby z nich skorzystać. Mimo tych problemów ostateczne wyniki, widoczne na przetworzonym materiale, napawają optymizmem. Rozwiązanie może znaleźć zastosowanie w meczach szóstek piłkarskich, w których kamera jest nieruchoma, a więc narastający błąd projekcji wynikający z błędu estymacji ruchu kamery niewystępowałby. To sprawia, że system mógłby być rozwijany w zaprojektowanej postaci lub przebudowywany na system wykorzystujący metody sztucznej inteligencji, które mogłyby okazać się skuteczniejsze w detekcji obiektów na obrazie. Rozwój systemu mógłby polegać na ulepszeniu skuteczności wykrywania, dodaniu dodatkowych metryk do analizy takich jak procent posiadania piłki, skuteczność podań i strzałów, a także na zdefiniowaniu sztywnych warunków wykorzystywania, w których system działa z wyższą skutecznością.