POLITECHNIKA WARSZAWSKA

DYSCYPLINA NAUKOWA – INFORMATYKA TECHNICZNA
I TELEKOMUNIKACJA /
DZIEDZINA NAUK – NAUKI INŻYNIERYJNO-TECHNICZNE

Rozprawa doktorska

mgr inż. Michał Kopania

Wspomaganie decyzji sędziów w zakresie miejsca upadku lotki względem pola gry podczas meczów badmintona

Promotor
prof. dr hab. inż. Artur Przelaskowski

Promotor pomocniczy
dr inż. Grzegorz Ostrek

WARSZAWA 2024

Streszczenie

Badminton uchodzi za najszybszy sport na świecie. Podczas meczu badmintona lotka w momencie uderzenia może poruszać się z prędkością przekraczającą 400 km/h [1, 2]. Aktualny rekord prędkości zarejestrowany przez firmę Yonex (producent sprzętu do badmintona) w kontrolowanym środowisku w dniu 14 kwietnia 2023 wynosi aż 565 km/h [3] i jest lepszy od poprzedniego z 2013 roku o 72 km/h [4]. Choć po samym uderzeniu lotka gwałtownie zwalnia, w momencie upadku na ziemię jej prędkość jest na tyle wysoka, że ludzkie oko nie jest w stanie dokładnie określić miejsca odbicia lotki od podłoża. Skutkuje to dość częstymi pomyłkami sędziowskimi – około 20%-24% decyzji sędziów jest błędna [5, 6], co prowadzi do wielu realnych problemów, m.in. niesprawiedliwych werdyktów sędziowskich wypaczających przebieg zawodów, przedłużających się kontrowersji, czy dyskusji uderzających w atrakcyjność rozgrywek. Główną motywacją realizowanych badań było opracowanie narzędzia wspomagającego, które usprawniłoby pracę sędziów liniowych – ograniczając liczbę ich błędnych decyzji, zwiększając klarowność analizy i przyspieszając konsensus w przypadkach wątpliwych przez obiektywizację dokonywanych ocen.

Zasadniczym osiągnięciem przedstawionym w niniejszej rozprawie jest oryginalne rozwiązanie w zakresie zastosowania wyników własnych badań naukowych w sferze gospodarczej, polegające na konstrukcji efektywnego modelu zachowań upadającej lotki względem linii referencyjnych, pozwalającego ustalić i zinterpretować miejsce upadku lotki przy możliwie dużej odporności na różnorodne, praktyczne zakłócenia i realistycznie zmieniające się uwarunkowania pomiarowe. Uzyskane wyniki własnych badań naukowych zostały zweryfikowane względem realnych zastosowań w sferze gospodarczej. Przeprowadzone w okresie kilku lat badania, oraz prace wdrożeniowe, które przedstawiono w niniejszej rozprawie, skutkowały opracowaniem i zaimplementowaniem praktycznego, komercyjnego systemu, który w warunkach realnych rozgrywek skutecznie wspomaga sędziów zawodów badmintonowych w ocenie miejsca upadku lotki i podjęciu kluczowej decyzji: „ w korcie” lub „ aut”. Unikalnym pomysłem odróżniającym badania autora od znanych z literatury [7,8,9,10] jest wyznaczenie momentu i miejsca upadku lotki w stosunku do linii kortu z wykorzystaniem układu pomiarowego, w którym każda z kamer obserwuje swoją przestrzeń (określoną linię kortu) a nie jak to jest w zazwyczaj spotykanych implementacjach, gdzie wiele kamer obserwuje tę samą przestrzeń (najczęściej cały kort).

Niniejsza rozprawa została przygotowana w ramach doktoratu wdrożeniowego edycji III. Celem doktoratu wdrożeniowego określonym przez Ministerstwo Nauki i Szkolnictwa

Wyższego jest „rozwiązanie oryginalnego problemu naukowego oraz zagadnienia praktycznego, w taki sposób, aby powstałe rozwiązanie można było wdrożyć". W związku z powyższym oprócz rozwiązań konkretnych problemów naukowych, w niniejszej rozprawie, przedstawiono również istotny aspekt praktyczny i biznesowy. Konkretne osiągnięcie naukowe zostało wdrożone w sferze gospodarczej – kompleksowy system wspomagania (wielokamerowy układ pomiarowy, strumieniowanie danych o dużej rozdzielczości czasowej, metody analizy i interpretacji wideo online, sugerowane decyzje wspierające pracę sędziów), opracowany wspólnie z drugim doktorantem, Jarosławem Nowiszem. System ten został zweryfikowany w praktyce podczas realnych zawodów sportowych, m.in. na Mistrzostwach Świata seniorów w Katowicach. Wyniki z systemu były prezentowane nie tylko sędziom, ale też wyświetlane na głównym ekranie w hali oraz podczas transmisji telewizyjnej. Zastosowane rozwiązanie całkowicie wyeliminowało wątpliwości zawodników dotyczące miejsca upadku lotki – w polu gry czy poza nim. Widać to doskonale na zapisie transmisji wideo z tych zawodów, który jest dostępny w serwisie youtube.com. Linki do kluczowych momentów transmisji prezentujących efekt działania systemu są podane w referencji [11] oraz [12].

W ramach prac badawczych autor podjął samodzielnie następujące zagadnienia:

- detekcja, segmentacja i śledzenie zachowań lotki (tj. niewielkich, szybko opadających obiektów o specyficznych cechach statycznych i dynamicznych) względem podłoża pokrytego liniami referencyjnymi na podstawie rejestrowanych online strumieni wideo,
- wyznaczenie ramki kluczowej, najbliższej chwili upadku (odbicia lotki od kortu) celem precyzyjnej lokalizacji miejsca odbicia,
- lokalizacja miejsca upadku (odbicia od podłoża) korka lotki względem referencyjnych linii kortu i określenie, czy w momencie odbicia lotka znajdowała się w polu gry czy poza nim,
- opracowanie intuicyjnego oprogramowania dla sędziów umożliwiającego rzetelną weryfikację i uwiarygodnienie automatycznej podpowiedzi systemu oraz jej ewentualną korektę przed udostępnieniem wizualizacji miejsca upadku lotki kibicom i zawodnikom.

Praca składa się z 5 rozdziałów.

Rozdział 1 wprowadza czytelnika w tematykę problemu i zasadnicze uwarunkowania zewnętrzne oraz istotne ograniczenia mające wpływ na prowadzone badania. Przedstawiono w nim wagę problemu błędnych decyzji sędziów oraz uzasadniono potrzebę opracowania

technologicznego narzędzia wspomagającego sędziów zawodów sportowych. Wskazano główne osiągnięcie i cel prowadzonych badań.

W rozdziale 2 dokonano przeglądu aktualnego stanu wiedzy i przedstawiono koncepcje i metody wspomagania decyzji sędziowskich.

Rozdział 3 prezentuje zasadnicze osiągnięcie i opisuje opracowaną metodę.

W rozdziale 4 opisano w jaki sposób zweryfikowano opracowaną metodę wraz z dyskusją uzyskanych wyników.

Podsumowanie i wnioski przedstawiono w rozdziale 5.

Słowa kluczowe: wizja komputerowa, sport, szybko poruszające się obiekty, detekcja obrazu, segmentacja małych obiektów na obrazie, wspomaganie decyzji sędziów liniowych, śledzenie i modelowanie ruchu obiektów, analiza i interpretacja obrazów

Abstract

Badminton is the fastest sport in the world. A badminton shuttlecock can travel at initial speeds in excess of 400 km/h [1, 2]. The current Guiness world record (registered on April 14th 2023) is 565 km/h [3] which is better by 72 km/h than a previous record from 2013 [4]. After the impact it slows down sharply, but when it hits the ground the speed is so high that the line judges often cannot tell if the shuttlecock was in or out. This leads to refereeing errors - about 20-24% of refereeing decisions are wrong [5, 6].

The research presented in this thesis was aimed at verifying whether it is possible to build a commercial system that in real conditions can help referees of badminton competitions in assessing the place of the shuttlecock fall – in or outside the court.

This thesis was prepared as part of the third edition of the implementation doctorate. The purpose of the implementation doctorate, as defined by the Ministry of Science and Higher Education, is "to solve an original scientific problem and a practical problem in such a way that the resulting solution can be implemented". Therefore, in addition to presenting scientific problems and their solutions, this thesis also addresses the practical and business aspects.

A specific scientific achievement was implemented in the economic sphere - the comprehensive support system developed together with the second doctoral student - Jarosław Nowisz (multi-camera measurement system, high-resolution time data streaming, online video analysis and interpretation methods, suggested decisions supporting the work of referees) was verified in practice during real sports competitions, in particular it was used in the Senior World Championships in Katowice. This can be seen perfectly in the video recording of the competition, which is available on youtube.com. Links to key moments of the broadcast presenting the effect of the system are provided in references [11] and [12].

As part of the research work, the following issues were undertaken:

- Detection and tracking fast-moving objects.
- Badminton shuttlecock segmentation.
- Finding the frame in the video stream where the shuttlecock hit the ground.
- Determining the position of the badminton shuttlecock cork in relation to the reference lines of the court and determining whether the shuttlecock was in or outside the playing field at the moment of the bounce.

- Developing easy-to-use control software for referees that allows for verification of the system's automatic decision.

The work consists of 5 chapters.

Chapter 1 introduces the reader to the subject of the problem and the main external conditions and significant limitations influencing the conducted research. It presents the importance of the problem of incorrect decisions of referees and justifies the need to develop a technological tool supporting referees of sports competitions. The main achievement and the aim of the conducted research are indicated.

Chapter 2 reviews the current state of knowledge and presents concepts and methods of supporting referee decisions.

Chapter 3 presents the main achievement and describes the developed method.

Chapter 4 describes how the developed method was verified along with a discussion of the obtained results.

A summary and conclusions are presented in Chapter 5.

Keywords: computer vision, sport analysis, fast-moving objects, image detection, object segmentation

Spis treści

1. Wprowadzenie – technologiczne wspomaganie decyzji sędziowskich w sporcie

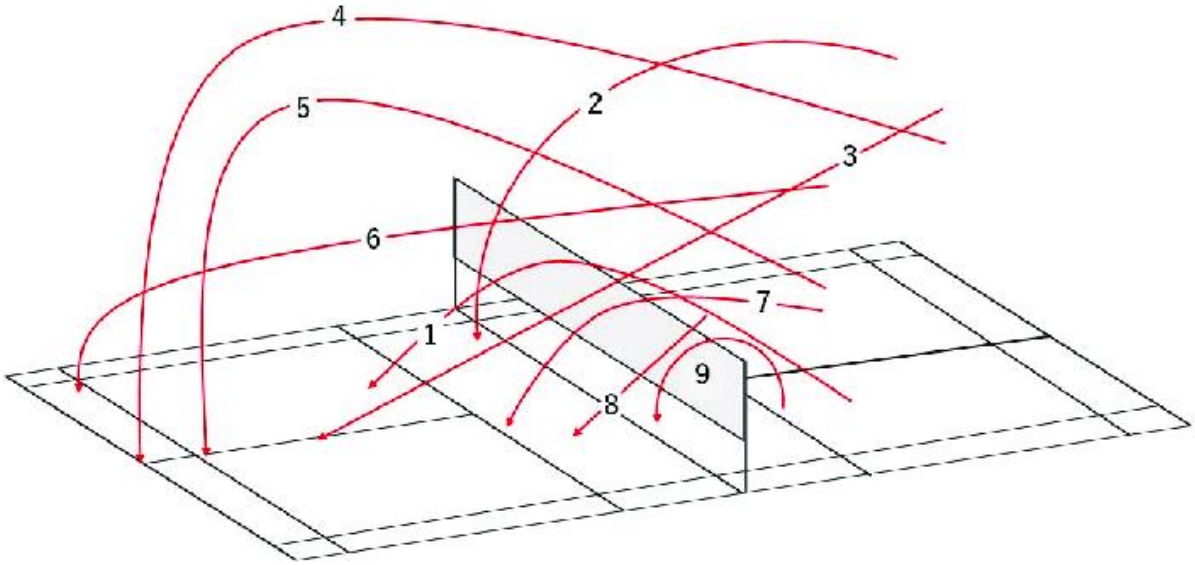
1.1. Motywacja

Zawodowy sport to ciężka praca, a sportowcy zarabiają głównie podczas zawodów. Autor nie zna dyscypliny sportowej, w której nie byłoby sędziów podejmujących kluczowe dla zawodnika decyzje. Jeden błędny werdykt sędziego może zdecydować o tym, czy zawodnik wygra mecz i zagra w finale, czy też odpadnie na etapie półfinału. Przykładowo finalista tenisowego turnieju US Open 2024 otrzyma nagrodę przynajmniej w wysokości 1,8 mln USD, a półfinalista – 1 mln. Można zaryzykować stwierdzenie, że pomyłka sędziego w kluczowym momencie meczu potencjalnie może „kosztować" 800 tys. dolarów. Choćby z tego powodu systemy wspomagania decyzji sędziowskich za pomocą obliczeniowych metod analizy i interpretacji bieżących zapisów wideo – od rozstrzygania klasycznego problemu „boisko czy aut", przez precyzyjną detekcję spalonego w skoku w dal, po stwierdzenie kto jest pierwszy w biegu na 100 metrów – są kluczowe dla zawodników.

Zgodnie z raportem IMARC Group [13] globalny rynek analizy sportów był wyceniony w 2022r. na 1.1 miliarda dolarów, z czego 35% dotyczyło analizy wideo. W publikacji *A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions* [14] autorzy dokonali przeglądu badań naukowych w dziedzinie analizy wideo z zawodów sportowych w wiodących czasopismach. W latach 2020-21 pojawiło się 77 publikacji (wzrost z 66 w latach 2018-19) poruszających ten temat. Najwięcej publikacji dotyczyło piłki nożnej – 26%, zaś badmintona – 4%.

Autor regularnie gra w badmintona i startuje w turniejach amatorskich i seniorskich. Biorąc udział w zawodach niejednokrotnie nie zgadzał się z decyzją sędziego, ale musiał ją uszanować, ponieważ nie było żadnego obiektywnego narzędzia pozwalającego na zweryfikowanie podjętej przez sędziego decyzji. Czasami podczas zawodów amatorskich, gdy nie ma sędziów, to inni zawodnicy sędziują nawzajem swoje mecze. Wbrew pozorom nie jest to proste i niejednokrotnie, gdy zadaniem autora było sędziowanie meczu, nie miał pewności, czy było boisko, czy aut, a nawet zdarzały się sytuacje, gdy dwie osoby obserwujące lotkę upadającą na kort nie mogły się zgodzić, czy lotka upadła w polu, czy nie. Dlatego też, wzorem innych dyscyplin, takich jak siatkówka czy tenis, autor postanowił rozpocząć badania nad systemem wspomagającym decyzje sędziowskie – by uniknąć subiektywnych ludzkich pomyłek, wynikających przede wszystkim z ograniczonych ludzkich zdolności percepcji ruchu

upadającej lotki. Mniejsza liczba publikacji naukowych dotyczących badmintona niż, na przykład, piłki nożnej czy tenisa spowodowana jest tym, że nagrody pieniężne w badmintonie są dużo niższe niż w wymienionych dyscyplinach. Największe turnieje badmintona mają pulę nagród do 2 mln dolarów, a tenisowe nawet 75 mln. Warto nadmienić, że w wielu sportach gra się piłką (będącą kulą), dzięki czemu ten sam model zachowania odbitej lub rzuconej piłki można zastosować w różnych konkurencjach. Niestety ze względu na kształt lotki teoretyczne modele mają ograniczoną praktyczną skuteczność, ponieważ nie uwzględniają wszystkich zmiennych. Na zachowanie lotki uderzanej z różną siłą za pomocą różnych technik ma wpływ nie tylko niepowtarzalna jej konstrukcja, ale też zmieniające się intensywnie w czasie jej właściwości (pióra lotki ulegają uszkodzeniu podczas gry) oraz uwarunkowania zewnętrzne (temperatura i wilgotność powietrza zależna od miejsca rozgrywania zawodów, wpływ klimatyzacji na tor lotu lotki etc.) W publikacjach [15, 16] autorzy stworzyli modele zachowań lotki, ale nie uwzględnili wspomnianych uwarunkowań, ponieważ testowali swoje modele w warunkach laboratoryjnych. Podjęta przez autora próba zastosowania tych modeli w warunkach rzeczywistych okazała się porażką. Zaobserwowane miejsce upadku lotki w stosunku do wyznaczonego przez modele ze wspomnianych publikacji zgadzało się tylko wtedy, gdy testowanym uderzeniem był długi serwis (lift) – czyli uderzenie do góry na koniec kortu. W badmintonie istnieje szereg różnych zagrań, po których lotka porusza się po różnych trajektoriach. Poniżej - [Rysunek 1] przedstawiono różne typy uderzeń lotki i ich typowe trajektorie. W rzeczywistości dwie trajektorie lotki uderzonej z taką samą prędkością początkową i pod tym samym kątem mogą się istotnie różnić. Duże znaczenie ma to, czy zawodnik podczas uderzania lotki ją podciął, a jeśli tak, to od której strony, a nawet czy jest leworęczny, czy praworęczny. Podcięcie ma istotny wpływ na trajektorię, ponieważ zwiększa lub zmniejsza ruch wirowy lotki (pióra w lotce nachodzą na siebie, więc swobodnie puszczona lotka wiruje zawsze w tę samą stronę). Wstępne próby poprawy tych modeli względem praktycznych kryteriów interpretacyjnych (ocena poprawności zagrań kończących) nie przyniosły spodziewanych efektów.



Rysunek 1 Różne typy uderzeń i ich trajektorie:. (1) serve, (2) drop, (3) smash, (4) clear, (5) lift, (6) drive, (7) block, (8) net kill, (9) net shot [1]

Z badań własnych wynikło, że efektywniejsze, szczególnie w kontekście wspomagania decyzji sędziowskich, są badania śledzące ostatnią fazę lotu lotki tuż nad kortem. Zdecydowano się więc na eksplorację problemów naukowych właściwie nieporuszanych nigdy wcześniej, tj. szybką analizę i interpretację wielu strumieni wideo z dobranej lokalnie przestrzeni obserwacji upadającej lotki. Wykorzystano przede wszystkim kryteria pragmatyczne (przepisy gry, formy oceny sędziowskiej, zróżnicowanie możliwych interpretacji, realistyczne uwarunkowania przestrzenno-czasowe zawodów, możliwe fluktuacje infrastruktury teleinformatycznej, dyskusje problemów sędziowskich, doświadczenia zawodników etc.). System pomiarowy i formy analizy doskonalono przez kilka lat z wykorzystaniem przede wszystkim kryteriów pragmatycznych, uwzględniających np. koszty sprzętu, istniejące ograniczenia w możliwościach rozmieszenia kamer, przepustowości kabli, możliwości zrównoleglenia obliczeń czy poprawy rozdzielczości dokonywanych online analiz, zakłócenia powodowane złym oświetleniem, przemieszczaniem się kibiców czy też kamerzystów. Te długotrwałe, prowadzone w różnych realiach badania pozwoliły uzyskać dużą niezawodność systemu i przede wszystkim jego użyteczność w bardzo zróżnicowanych warunkach realnych zawodów badmintonowych.

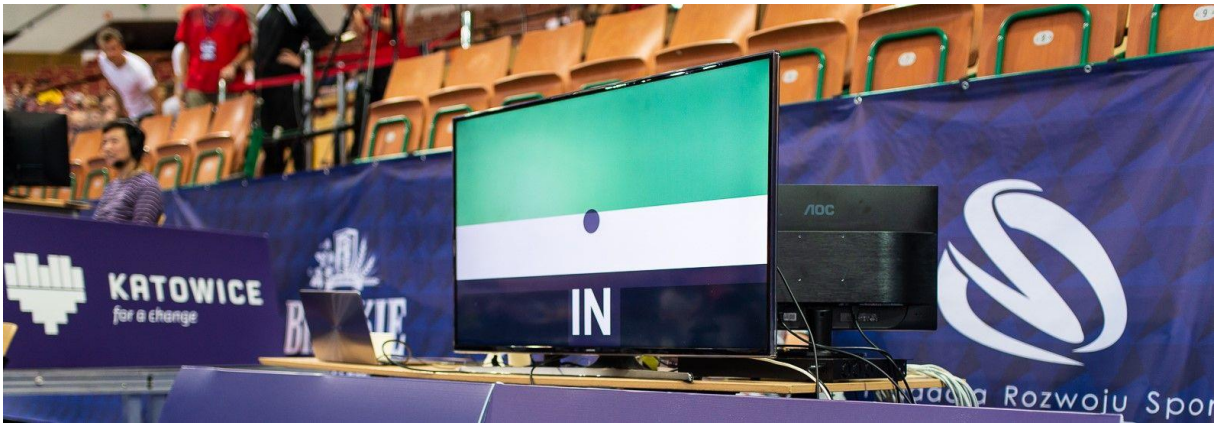
Aktualnie na rynku istnieje jeden system komercyjny (Hawk-Eye, Sony)[17], który wspomaga sędziów badmintonowych. Niestety jest on bardzo drogi, w związku z czym wykorzystuje się go tylko na największych turniejach badmintonowych (takich z pulą nagród powyżej 500 tys.

[1] Źródło rysunku: https://www.researchgate.net/figure/Illustration-of-different-stroke-types-The-trajectories-of-1-serve-2-drop-3_fig2_360647085. Autor: Ning Ding

USD), których jest kilkanaście w roku. Firma Sony nie udostępnia publicznie wyników swoich badań.

Sięgając po naukowe narzędzia i koncepcje, autor myślał przede wszystkim o praktycznych kryteriach ich weryfikacji. Wybierał takie rozwiązania, które nie będą wymagały bardzo drogiego sprzętu i dużych kosztów montażu czy też transportu.

1.2. Opis problemu



Rysunek 2 Wizualizacja miejsca upadku lotki wygenerowana przez opracowany system zaprezentowana podczas turnieju w Katowicach [2]

Podczas meczu badmintona sędziowie liniowi, siedzący na wprost linii ograniczających pole gry, obserwują linie i sygnalizują głównemu sędziemu, czy lotka upadła w polu gry, czy poza nim. W zależności od rodzaju uderzenia prędkość lotki tuż przed odbiciem o ziemię może wynieść:

- 29,8 m/s dla uderzeń smash – uderzonych z prędkością początkową 85 m/s po dystansie 5 m,
- 11,8m/s dla uderzeń smash – uderzonych z prędkością początkową 85 m/s po dystansie 9 m,
- 6,7 m/s dla uderzeń high clear. [18, 19] (dla tego rodzaju uderzeń lotka przy podłożu osiąga prędkość graniczną, bez względu na prędkość początkową uderzenia)

Wśród graczy elity najczęściej uderzeniem kończącym jest smash – 54% [18]. Przy takich uderzeniach, dla których lotka upada bardzo blisko linii, sędziowie mają duży problem z ocenieniem, czy był aut, czy nie. Jedna błędna decyzja sędziego może decydować o wyniku meczu. Dzięki wspomagającym systemom komputerowym rezultat meczu częściej może być sprawiedliwy. Warto nadmienić, że zmęczenie sędziego wpływa na jego koncentrację i również może powodować błędną ocenę i przyczyniać się do pomyłek. Należy też zaznaczyć, że pojedyncze mrugnięcie okiem trwa od 15 do 400 m/s i szacuje się, że każdy z nas mruga co

[2] Źródło własne

około 6-20 razy na minutę [20] , więc sędzia możne po prostu nie dostrzec, w którym miejscu odbiła się lotka.

Aby rozwiązać przedstawiony problem, należy rozwiązać szereg problemów badawczych oraz zagadnień praktycznych, które przedstawiono w niniejszej rozprawie.

1. Dobór i testy sprzętu, w tym wybór kamer, jednostek obliczeniowych, peryferiów.
2. Wyznaczenie możliwie optymalnego miejsca montażu kamer i pozostałego sprzętu na konkretnej hali.
3. Kalibracja kamer z wykorzystaniem przenośnego stanowiska kalibracyjnego.
4. Wyznaczenie maski pola gry dla obrazu z każdej kamery.
5. **Detekcja i śledzenie lotki w strumieniu wideo.**
6. **Wyznaczenie kluczowej klatki wideo – zarejestrowanej najbliżej momentu upadku lotki na kort.**
7. **Precyzyjne wyznaczenie pozycji korka lotki i miejsca odbicia lotki od podłoża w stosunku do referencyjnych linii kortu.**
8. Weryfikacja przez sędziego technicznego automatycznej podpowiedzi systemu.
9. Prezentacja kibicom i zawodnikom animacji wizualizującej miejsce upadku lotki i decyzję in/out.

Nad opracowaniem kompletnego systemu, którego efektem końcowym jest komputerowa animacja wizualizująca miejsce upadku lotki [Rysunek 2], [21] wraz z informacją, czy było pole, czy aut, autor pracował wraz z drugim doktorantem – Jarosławem Nowiszem. W niniejszej rozprawie opisano szczegółowo stworzone algorytmy i składowe systemu oraz badania, które autor prowadził samodzielnie – punkty 5,6,7,8 z powyższej listy. O pozostałych składowych, które były realizowane samodzielnie przez J. Nowisza lub były realizowane wspólnie, wspomniano jedynie ogólnie, w taki sposób, aby czytelnik miał obraz całości rozwiązania.

Aby powstałe rozwiązanie mogło zostać wdrożone komercyjnie, to musi ono spełniać następujące wymagania rynkowe:

- dokładność systemu powinna być tak wysoka, jak to tylko możliwe, przy czym satysfakcjonującym rynek wynikiem jest dokładność wyznaczenia miejsca upadku lotki w stosunku do linii ograniczających pole gry z dokładnością poniżej 13 mm (połowa średnicy korka lotki do badmintona określona przepisami [22]);
- system powinien działać w czasie rzeczywistym;

- ze względu na wymagania światowej federacji badmintona (BWF) system powinien zapewniać możliwość weryfikacji decyzji systemu przez sędziego technicznego na podstawie zapisu wideo. Procedura weryfikacyjna powinna być tak przygotowana, aby nie trwała dłużej niż 25 sekund;
- czas potrzebny na instalację sprzętu w obiekcie, w którym odbywają się mecze, nie powinien przekroczyć 4 godzin na jeden kort przy udziale dwóch osób;
- sprzęt wykorzystywany przez system powinien zajmować możliwie mało miejsca. Idealnie, jeśli będzie możliwe wysłanie sprzętu jedną przesyłką paletową (Paleta EURO o wymiarach 120 x 80cm);
- koszt urządzeń i peryferiów koniecznych do sprawnego działania systemu powinien być możliwie niski w stosunku do merytorycznych wymagań użytkowych.

W rozprawie wspomniano również o wyzwaniach operacyjnych i technicznych, a także ograniczeniach, które nie są bez znaczenia przy opracowaniu użytecznego i możliwego do wdrożenia na rynku rozwiązania. Są to w szczególności:

- ograniczenia czasowe związane z instalacją systemu w hali,
- ograniczenia transportowe związane z wielkością i wagą sprzętu,
- ograniczenia związane ze środowiskiem zewnętrznym – oświetlenie w hali, brak miejsca pomiędzy kortami,
- ograniczenia finansowe – wybór odpowiedniego sprzętu (kamery, okablowanie, komputery i peryferia),
- integracja z systemem live[3] scoring i live streaming[4].

1.3. Cel badań i główne osiągnięcie

Celem badań jest rozwiązanie realnego problemu wspomagania decyzji sędziów liniowych dotyczących oceny skuteczności poszczególnych zagrań kończących w czasie meczów badmintona przez weryfikację miejsca upadku lotki. Głównym osiągnięciem autora jest opracowanie skutecznej metody weryfikacji realnego miejsca upadku lotki względem linii referencyjnych, służącej wspomaganiu decyzji sędziowskiej(w boisku lub aut), która znalazła zastosowanie w kompleksowym systemie wspomagania kluczowych decyzji sędziowskich dotyczących zawodów sportowych gry w badmintona. Eksperymentalnie potwierdzono realne

[3] Live scoring system. Oprogramowanie prezentujące aktualny wynik meczu na elektronicznej tablicy wyników oraz podczas telewizyjnej transmisji wideo

[4] Live streaming system. Oprogramowanie służące do transmisji wideo w czasie rzeczywistym w Internecie. W szczególności w serwisach takich jak youtube czy facebook

zmniejszenie liczby pomyłek sędziowskich o 20%, co uznano za znaczącą korzyść użytkową o istotnych walorach komercyjnych.
Nowatorstwo i oryginalność zaproponowanego systemu wyrażają się przede wszystkim w kompleksowości realizacji – od projektu i realizacji systemu akwizycji, przez wyniki konkretnych analiz i interpretacji obliczeniowych, aż po konkretną formę wspomagania decyzji sędziowskich w realiach prowadzonych zawodów sportowych.

W ramach wykonanych prac badawczo-wdrożeniowych zrealizowano następujące główne cele naukowe i aplikacyjne:

- Opracowanie skutecznej metody lokalizacji miejsca upadku lotki względem pola gry na podstawie wielostrumieniowego zapisu wideo, pozyskanego za pomocą zoptymalizowanej metody pomiarowej (zapis w określonej geometrii przestrzennej dostosowanej adaptacyjnie do różnorodnych uwarunkowań realnych kortów i hal sportowych) będącej w zgodzie z odpowiednimi przepisami [22].
- Eksperymentalna weryfikacja skuteczności i przydatności opracowanego systemu wspomagania w zróżnicowanych warunkach rzeczywistych.
- Opracowanie stabilnego oprogramowania działającego w czasie rzeczywistym, które w wyniku prezentuje miejsce upadku lotki w stosunku do linii kortu.

Realizacja tych działań skutkowała opracowaniem wiarygodnej metody możliwie precyzyjnej detekcji miejsca upadku lotki na podstawie analizy obrazów z kamer ustawionych w określonej konfiguracji przy korcie do gry w badmintona. Dokładność detekcji jest wystarczająca do:

a) skutecznego wspomagania decyzji sędziowskich w przypadkach wątpliwych lub krytycznych,
b) wiarygodnego potwierdzania autu lub jego braku w przypadkach klarownych na zasadzie kontroli lub korekcji pomyłek ludzkich,
c) automatycznego obliczania statystyk z poprawności pierwotnych decyzji sędziów liniowych.

1.4. Istotne uwarunkowania prowadzonych badań

Poniżej zostały opisane zewnętrzne ograniczenia mające wpływ na badania i podjęte przez autora kluczowe decyzje. Opisano również, w sposób ogólny inne problemy, które musiały zostać rozwiązane, aby opracować kompleksowe rozwiązanie komercyjne, a które wykraczają poza badania autora opisane w niniejszej rozprawie.

Układ pomiarowy uwzględniający ograniczone możliwości obserwacyjne

Przed przystąpieniem do zbierania danych konieczne było wybranie odpowiedniego sprzętu. Mając na uwadze komercjalizacje tworzonego rozwiązania, autor przy wyborze sprzętu pomiarowego i obliczeniowego, kierował się również aspektem ekonomicznym. W związku z tym zaprojektowano taki układ pomiarowy, który zapewniał możliwie wysoką dokładność przy założonych ograniczeniach finansowych.

Dokładność systemu w dużym stopniu zależy od parametrów kamery rejestrującej obraz – głównie rozdzielczości przestrzennej oraz liczby klatek na sekundę (FPS ang. frames per second). Te dwa parametry determinują ilość danych, które kamera przesyła poprzez interfejs komunikacyjny. Wspomniane parametry kamery są też najważniejsze z punktu widzenia prowadzonych przez autora badań. Im wyższa liczba klatek na sekundę tym mniejsze rozmycie poruszającej się lotki oraz precyzyjniej można wyznaczyć moment odbicia lotki od podłoża. Z eksperymentów przeprowadzonych przez autora wynika, że aby osiągnąć zadowalające rezultaty należy rejestrować wideo z prędkością 150 klatek na sekundę lub wyższą. Natomiast bardziej szczegółowa treść obrazów pozwala na precyzyjniejsze wyznaczenie pozycji korka lotki w stosunku do referencyjnych linii kortu.

Dzisiejsze kamery oferują rozdzielczość 2832 px × 2840 px przy 190 FPS z interfejsem CoaXPress 2.0 (2× CXP-12). Producenci sensorów oferują sensory nawet o wyższych rozdzielczościach, ale ze względu na ograniczenia przepustowości interfejsu komunikacyjnego nie jest możliwe skonstruowanie kamery o maksymalnie wysokiej rozdzielczości oferowanej przez producentów sensorów i maksymalnie możliwej liczbie klatek na sekundę.

Istnieją cztery główne interfejsy [Tabela 1]: GigE, CoaXPress, USB i CameraLink do przesyłania danych z kamer. Jakakolwiek komunikacja z kamerami za pomocą WiFi lub innej technologii radiowej nie jest możliwa w zatłoczonych i podatnych na zakłócenia miejscach, takich jak hale sportowe, dlatego jedyną opcją są kable.

Standard interfejsu	Maksymalna długość kabla (m)	Przepustowość (Gbps)
CoaXPress 2.0	35-100	3.125-12.5
CameraLink	5 -10	2 (4 przy użyciu 2 kabli)
USB 3.0	3	5
GigE	100	1-10

Tabela 1 Główne rodzaje interfejsów kamer

Autor do rejestracji wideo użył szaroodcieniowych kamer o rozdzielczości 800 × 600 pikseli, pracujących z maksymalną prędkością 240 klatek na sekundę. Ze względu na spore odległości kamer od komputerów przetwarzających z nich obraz (w zależności od warunków panujących

w hali sięgających nawet kilkudziesięciu metrów) wybrano kamery z interfejsem GigE. Dodatkowo dzięki możliwości zasilania kamer poprzez kabel ethernetowy (Power Over Ethernet) ogranicza się do jednego przewodu liczbę przewodów podłączonych do jednej kamery.

Kiedy kilka lat temu autor zaczynał pracę nad systemem, rynek nie oferował w sprzętu o dużo lepszych parametrach. Dzisiaj alternatywą dla wybranych wtedy kamer byłyby kamery z interfejsem CoaXPress. Warto jednak nadmienić, że układ pomiarowy wykorzystujący kamery z interfejsem CoaXPress byłby kilkukrotnie droższy. Dużo droższe są kable oraz karty do akwizycji obrazu instalowane w komputerach.

Wspomniana rozdzielczość okazała się wystarczająca, aby osiągnąć zadowalającą dokładność lokalizacji spadającej lotki przy umiarkowanych wymaganiach dotyczących złożoności obliczeniowej.

Ilość danych przesyłanych z kamer i wydajność algorytmów wykrywających i śledzących lotkę określają niezbędną moc obliczeniową i liczbę komputerów. W przypadku wyższych rozdzielczości mamy więcej danych, co determinuje nam wyższą liczbę komputerów potrzebnych do ich przetworzenia (a co za tym idzie i podnosi koszty).

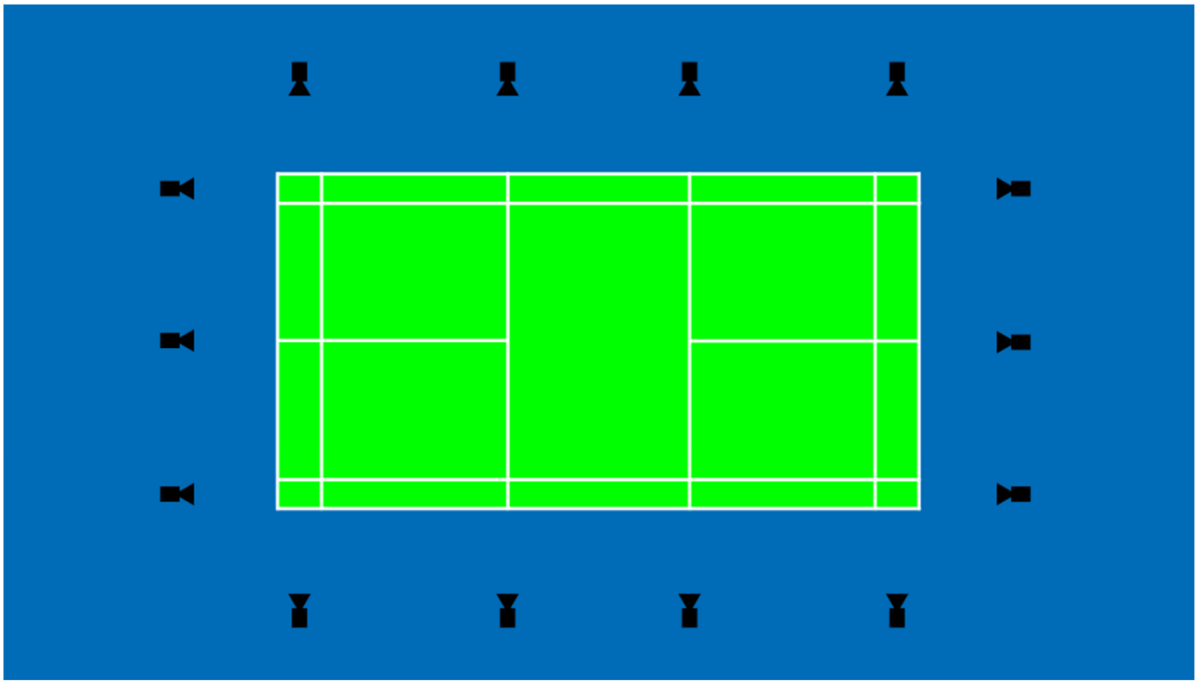
Jeśli chodzi o ustawienie kamer w stosunku do kortu, z jakimi można się spotkać, to stosuje się główne 3 strategie, które wraz z zaletami i wadami przedstawiono w tabeli [Tabela 2].

Sposób rozmieszczenia kamer	Zalety	Wady
Wiele kamer nad kortem. Każda kamera obejmuje w swoim polu widzenia cały kort.	• Możliwe śledzenie pełnej trajektorii lotki w 3D. • Wyznaczenie momentu odbicia lotki o podłoże wprost ze współrzędnych 3D lotki. • Możliwe zbieranie statystyk z przebiegu gry takich jak rodzaj uderzeń wygrywających, prędkość czy też kierunek uderzenia. • Wystarczy 6-9 kamer.	• Konieczna precyzyjna synchronizacja kamer, kalibracja i wyznaczenie wzajemnej pozycji kamer w ustalonym układzie odniesienia. • Kłopotliwy i długotrwały montaż, często wymagający podnośników. • Potrzeba setek metrów kabli. • Może być konieczny montaż w miejscach dostępnych dla publiczności (np. na trybunach) • W przypadku awarii np.: odłączenia się kabla od kamery, naprawa usterki może być bardzo problematyczna. • Mniej precyzyjne wyznaczenie korka lotki (lotka jest stosunkowo mała na obrazie).

Kamery ustawione wokół kortu. Każda z kamer obserwuje określone linie kortu	• Szybki i mało kłopotliwy montaż. Mniej kabli. • W przypadku awarii mało problematyczny serwis. • Nie jest konieczna synchronizacja kamer. • Można pominąć procedurę kalibracji kamer. • Duża lotka na obrazie pozwala na precyzyjniejsze wyznaczenie pozycji korka.	• Aby były widoczne wszystkie linie potrzeba minimum 10 kamer. Zalecane 14. • Nie jest możliwa rejestracja pełnej trajektorii lotki w związku z czym nie jest możliwe zebranie wielu interesujących statystyk dotyczących przebiegu gry. • Bardziej skomplikowana procedura wyznaczenia momentu odbicia lotki o podłoże.
Kamery nad kortem i wokół kortu	• Możliwe śledzenie pełnej trajektorii lotki w 3D. • Precyzyjne wyznaczenie momentu odbicia lotki o podłoże. • Możliwe zbieranie statystyk z przebiegu gry takich jak rodzaj uderzeń, prędkość czy też kierunek uderzenia. • Duża lotka na obrazie z kamer przy korcie pozwala na precyzyjniejsze wyznaczenie pozycji korka. • Uszkodzenie jednej kamery nie musi powodować, że system przestaje działać.	• Potrzeba największej liczby kamer. Przynajmniej 20. Oznacza też to najwyższy koszt takiego rozwiązania. • Konieczna precyzyjna synchronizacja kamer, kalibracja i wyznaczenie wzajemnej pozycji kamer w ustalonym układzie odniesienia. • Kłopotliwy i długotrwały montaż, często wymagający podnośników. • Potrzeba setek metrów kabli, a może nawet kilometrów. • Może być konieczny montaż w miejscach dostępnych dla publiczności (np. na trybunach). • W przypadku awarii np.: odłączenia się kabla od kamery znajdującej się nad kortem, naprawa usterki może być bardzo problematyczna.

Tabela 2 Możliwe strategie rozmieszczenia kamer podczas zawodów wraz głównymi z zaletami i wadami

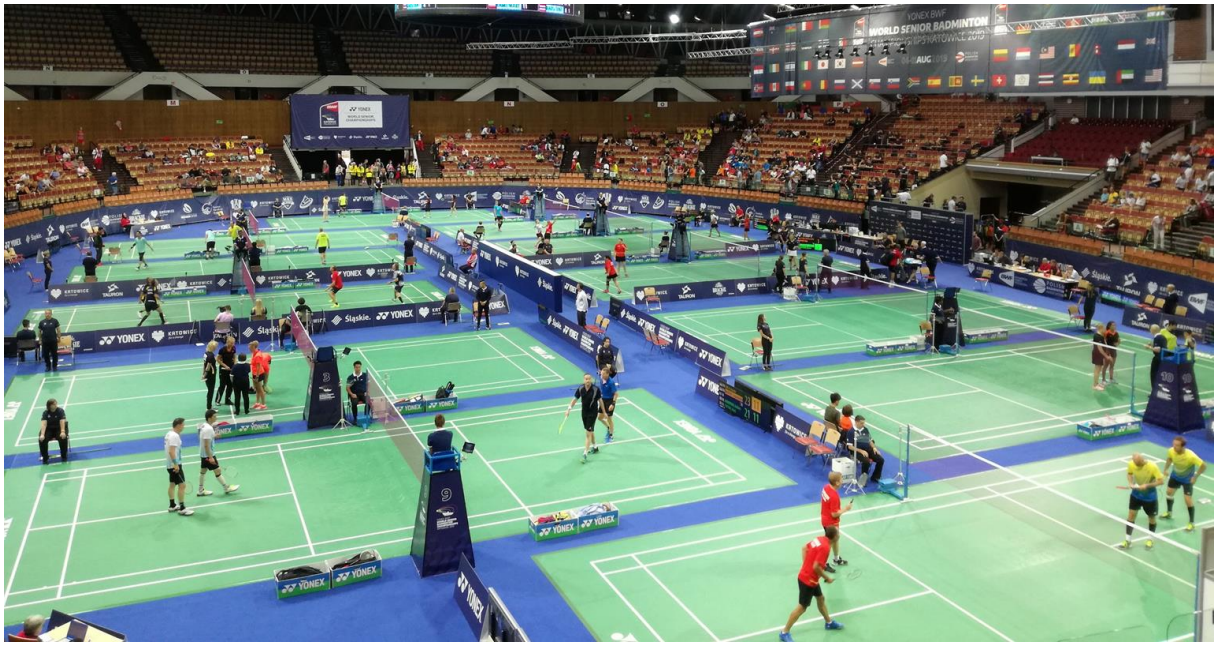
Autor zdecydował się na umieszczenie 14 kamer wokół kortu głównie z powodu możliwości szybkiego montażu. Sposób rozmieszczenia kamer zaprezentowano na rysunku [Rysunek 3].



Rysunek 3 Schemat i zdjęcie przedstawiające sposób rozmieszczenia 14 kamer wokół kortu

Bardzo często wokół kortów ustawione są banery reklamowe. Determinuje to lokalizację (za banerami) i wysokość na jakiej muszą być umieszczone kamery (nad banerem). Wysokość banerów jest określona przepisami [23] i nie może być żadnych elementów stojących w odległości mniejszej niż 1.5 od linii kortu – w praktyce nic nie można umieścić przed banerami reklamowymi. Z tego też powodu autor nie testował ustawienia kamer tuż nad podłożem.

Wspomniane zalety zastosowanego przez autora układu pomiarowego powodują, że bardzo szybko można zmienić ustawienie kamer. Podczas dużych turniejów, często na mecze finałowe, pozostawia się tylko jeden kort główny, wokół którego jest więcej pustej przestrzeni niż podczas wcześniejszej fazy turnieju, patrz [Rysunek 4, Rysunek 5]. Zmiana ustawień kortów często jest dokonywana w nocy pomiędzy ostatnimi dniami turniejowymi. Oznacza to, że po rozłożeniu mat, na których grają zawodnicy, ma się tylko kilka godzin na instalację i testy systemu w nowych warunkach przestrzennych.



Rysunek 4 Przykładowe rozmieszczenie kortów we wczesnej fazie turnieju. Mistrzostwa świata seniorów w Katowicach

Rysunek 5 Pozostawiony jeden kort podczas fazy finałowej (Igrzyska olimpijskie, Londyn 2012, źródło Wikipedia)

Nie bez znaczenia jest też ilość sprzętu, który musi zostać przetransportowany, a następnie umieszczony w hali. Im mniej potrzeba sprzętu (kable, komputery, peryferia) tym lepiej. Mniejsze gabaryty i waga sprzętu oznaczają mniejsze koszty transportu. W naszym systemie do przetwarzania danych używa się 7 komputerów klasy PC. Każdy komputer przetwarza dane z dwóch kamer.

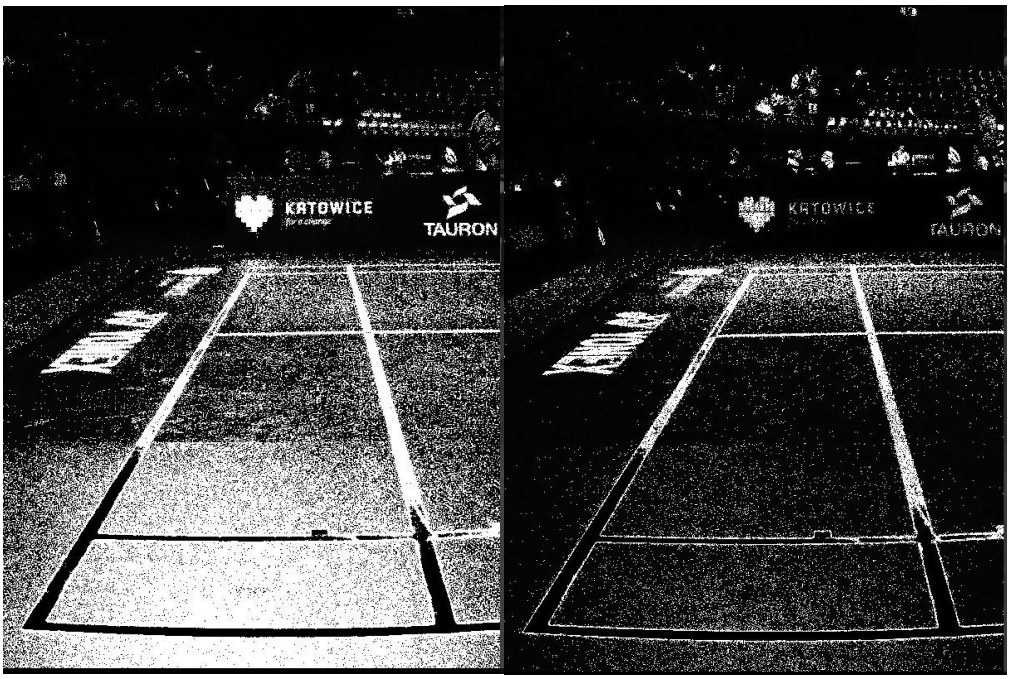
Dynamiczne warunki pomiarowe

Różne obiekty sportowe mają różne oświetlenie. Aby nagrywać wideo z prędkością ponad 150 FPS, oświetlenie musi być wystarczająco silne. Problem ten nie występuje w tenisie (z wyłączeniem meczów wieczornych), gdzie zwykle mamy światło dzienne.

Ważnym krokiem, który jest wykonywany podczas montażu kamer w hali jest dobranie możliwie najkrótszego, w zastanych warunkach oświetleniowych, czasu naświetlania sensora (w praktyce FPS kamery). Ocena, czy przy ustalonym czasie naświetlania obraz jest dobrze naświetlony dokonywana jest na podstawie analizy histogramu jasności pikseli zgodnie z metodą zaproponowaną przez Kuldeep Singh [24].

Nawet w hali o najgorszych warunkach oświetleniowych udało się zarejestrować odpowiednio naświetlone obrazy z prędkością 150 klatek na sekundę, ale tylko w jednym miejscu zainstalowano wystarczająco mocne światła, aby nagrać wideo z prędkością 200 FPS.

Opracowany przez autora algorytm detekcji lotki do badmintona wykorzystuje ramki różnicowe. Aby działał on poprawnie, to wymagane jest równomierne w czasie oświetlenie sceny. Niestety prąd zmienny, którym zasilane są lampy w halach powoduje ich migotanie (z częstotliwością 50 Hz). Na dziewięć odwiedzonych obiektów, w których organizowane są turnieje badmintona w Polsce, tylko jeden miał zainstalowane oświetlenie bez migotania. Na rysunku [Rysunek 6] przedstawiono przykładowe ramki różnicowe wygenerowane dla statycznej sceny przy migoczącym oświetleniu.



Rysunek 6 Przykładowe ramki różnicowe wygenerowane ze strumienia wideo zarejestrowanego z prędkością 190 klatek na sekundę w hali wyposażonej w migoczące oświetlenie

Konieczne było zatem opracowanie efektywnej metody kompensującej ten efekt. Autor skorzystał z wyjątkowo efektywnej metody kompensującej efekt nierównomiernego oświetlenia sceny spowodowany migotaniem oświetlenia opracowanej przez pana Nowisza. Została ona szczegółowo opisana w publikacji, której autor jest współautorem - „*A. Realtime flicker removal for fast video streaming and detection of moving objects*" [25]. Metoda polega na adaptacyjnym generowaniu masek kompensujących efekt migotania na podstawie analizy zmienności jasności pikseli w czasie. Dla stacjonarnych kamer obliczany jest poziom podobieństwa każdego piksela do tego samego piksela z poprzedniej klatki. Jeśli piksel zmienił się z powodu lokalnego ruchu w scenie, poziom podobieństwa będzie niski, w przeciwnym razie wysoki. Jeśli poziomy podobieństwa są wyższe od adaptacyjnie wyznaczonej wartości progowej, to różnicę w jasności pomiędzy takimi pikselami wykorzystuje się do utworzenia maski kompensującej efekt migotania. Opracowana metoda jest 250–300 razy szybsza niż znane na rynku komercyjne rozwiązania takie jak DeFlicker [26] i FlickerFree [27].

Wyznaczenie maski pola gry dla obrazu z każdej kamery

W swojej pracy badawczej autor nie zajmował się automatycznym wyznaczaniem pola gry. Dla każdej kamery maska pola, gry była wyznaczana ręcznie. Jednakże, automatyczne wyznaczenie pola gry jest korzystne z dwóch powodów:

- Jest szybsze niż wyznaczane przez operatora.

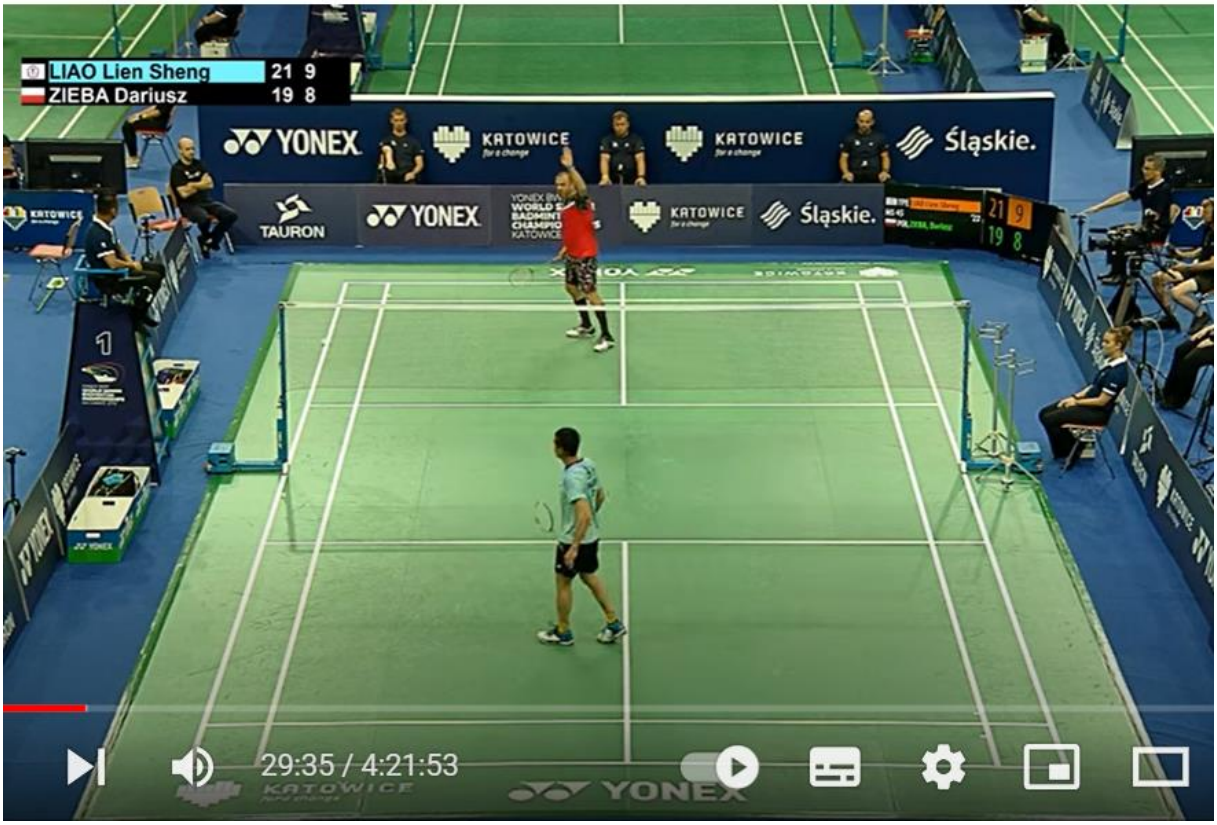
- W przypadku delikatnego poruszenia kamery (np. w wyniku drgań podłoża) maska pola gry może być automatycznie skorygowana.

Ten problem był badany dokładniej przez pana Nowisza, a rezultaty jego badań zostały opublikowane w naszym wspólnym artykule [6]. Metoda opracowana przez pana Nowisza polega na rozpoznaniu fragmentu kortu przez sieć neuronową (różne kamery widzą różne fragmenty kortu), a następnie na znalezieniu linii odpowiadających modelowi kortu wewnątrz maski wygenerowanej przez sieć neuronową. Ograniczenie obszaru, w którym szukane są linie tradycyjnymi metodami wizji komputerowej, znacząco zmniejsza liczbę analizowanych linii i pozwala automatycznie wybrać, te które są liniami kortu.

Prezentacja animacji wizualizującej miejsce upadku lotki i decyzję in/out

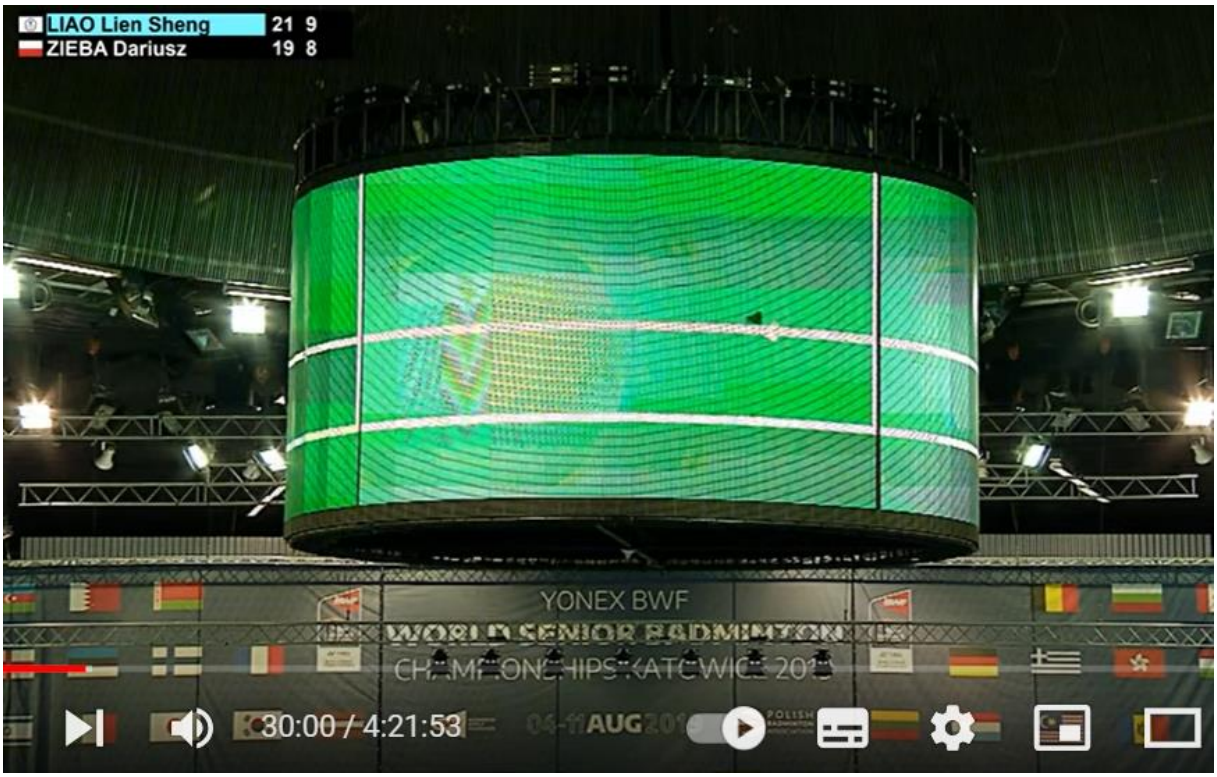
Wynik z systemu challenge[5] jest prezentowany jako animacja komputerowa pokazująca miejsce odbicia piłki lub lotki. Taki sposób prezentacji ma jedną podstawową zaletę. Zawodnicy i kibice nie mają wątpliwości czy było pole czy aut - nawet jeśli decyzja systemu jest błędna (nie ma rozwiązań w 100% skutecznych). Autor zachęca czytelnika to obejrzenia na youtube dwóch fragmentów meczów badmintona podczas których wynik z opracowanego systemu był prezentowany publiczności [11] oraz telewidzom [12]. Na zdjęciach poniżej, zaprezentowano kluczowe momenty z tych dwóch filmów.

[5] Definicja podana w Słowniczek pojęć punkt 1 na stronie 93

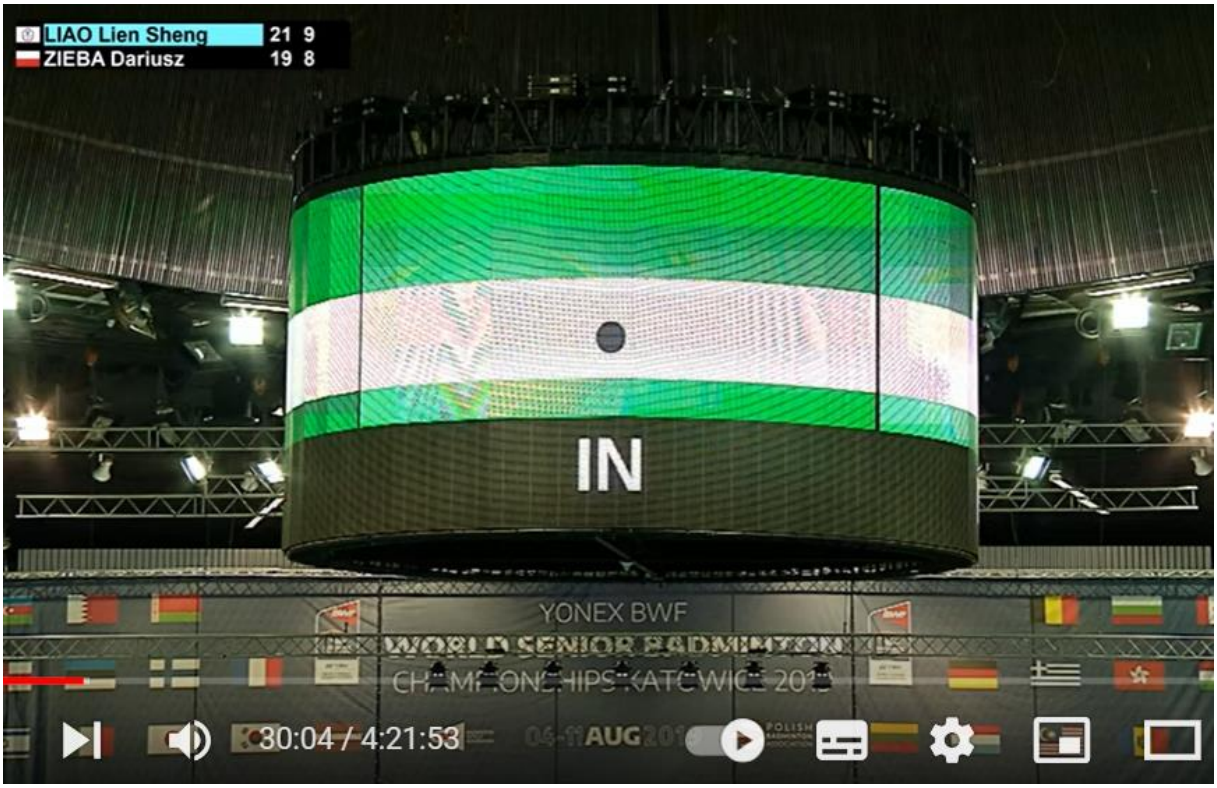


Rysunek 7 Zawodnik podnosi rękę prosząc o weryfikację decyzji sędziego przez system challenge. Widoczny za kortem po lewej sędzia sygnalizuje wyciągniętą ręką swoją decyzję - IN

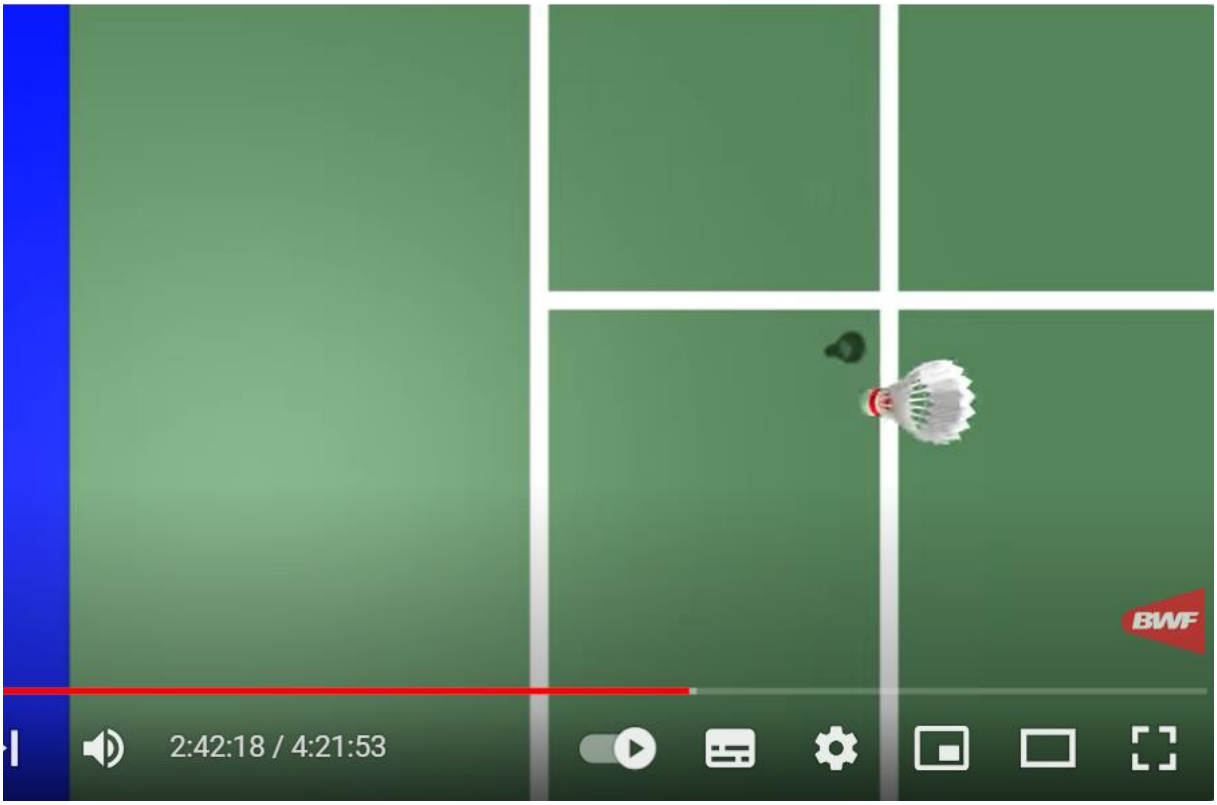
Rysunek 8 Oczekiwanie na wynik z systemu (w tym czasie sędzia techniczny dodatkowo weryfikuje poprawność decyzji systemu)



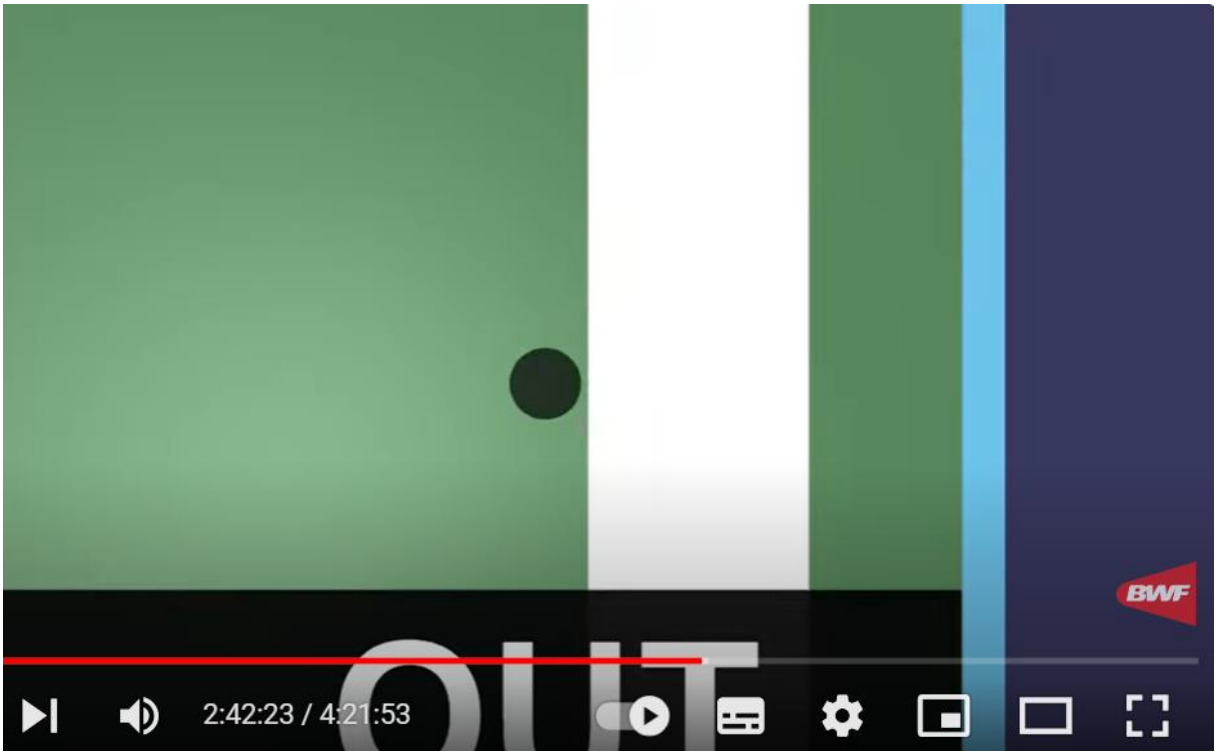
Rysunek 9 Animacja lecącej lotki wyświetlona na wielkim ekranie pod kopułą hali w Katowickim Spodku



Rysunek 10 Prezentacja lokalizacji miejsca upadku lotki w stosunku do linii kortu wraz z decyzją IN - w polu gry. W tym przypadku sędzia podjął prawidłową decyzję



Rysunek 11 Animacja lecącej lotki w kierunku linii końcowej prezentowana widzom podczas transmisji telewizyjnej (z innego meczu niż na wcześniejszych zdjęciach)



Rysunek 12 Prezentacja telewizyjna lokalizacji miejsca upadku lotki w stosunku do linii (z prawej strony widoczne wjeżdżające logo turnieju, które pojawia się pomiędzy animacją z systemu challenge, a transmisją wideo z kamer telewizyjnych.

2. Powiązane prace i stan wiedzy

2.1. Śledzenie i detekcja obiektów

Śledzenie obiektów w sekwencji wideo jest jednym z głównych tematów widzenia komputerowego. Określenie zachowań (kluczowych cech) i dynamiki interesującego nas obiektu jest niezbędne w przypadku każdego problemu śledzenia obiektów. Śledzenie małych, szybko poruszających się obiektów, takich jak lotka do badmintona (o zmieniającym się kształcie i nieprzewidywalnym, nie dającym się opisać formalnie torze ruchu), jest dość specyficznym zagadnieniem śledzenia obiektów. Nie wszystkie metody, które sprawdzają się podczas śledzenia większych obiektów takich jak ludzie, zwierzęta, pojazdy można z powodzeniem zastosować do śledzenia lotki lub piłki [28]. Nie oznacza to, że przedstawiając aktualny stan wiedzy nie warto wspomnieć o uznanych metodach śledzenia obiektów.

W zależności od badanego problemu, śledzenie obiektów można podzielić na dwie grupy: śledzenie pojedynczego obiektu (single object tracking) i śledzenie wielu obiektów (multi object tracking). W sporcie przykładem śledzenia pojedynczego obiektu może być śledzenie piłki, a wielu obiektów śledzenie piłkarzy podczas meczu piłki nożnej.

W ogólności algorytmy, śledzące obiekt/obiekty można podzielić też ze względu na metody w nich stosowane:

1. Zunifikowane metody śledzenia i wykrywania.

   Tego typu metody wykonują proces śledzenia i detekcji razem bez odrębnego detektora. Najczęściej te metody wymagają ręcznej inicjalizacji i mają stałą liczba obiektów w początkowej klatce. Następnie, obiekty te są lokalizowane i śledzone w kolejnych klatkach wideo.

   1.1. Metody bazujące na filtrach.

      Do tej kategorii można zakwalifikować przede wszystkim metody wykorzystujące filtr Kalmana [29] i filtry cząsteczkowe [30], czy też metody bazujące na śledzeniu jądra (Kernel Tracking) takie jak na przykład KFC [31]. Więcej informacji dotyczących tych metod znajduje się w tabeli [Tabela 3].

   1.2. Metody bazujące na wyszukiwaniu.

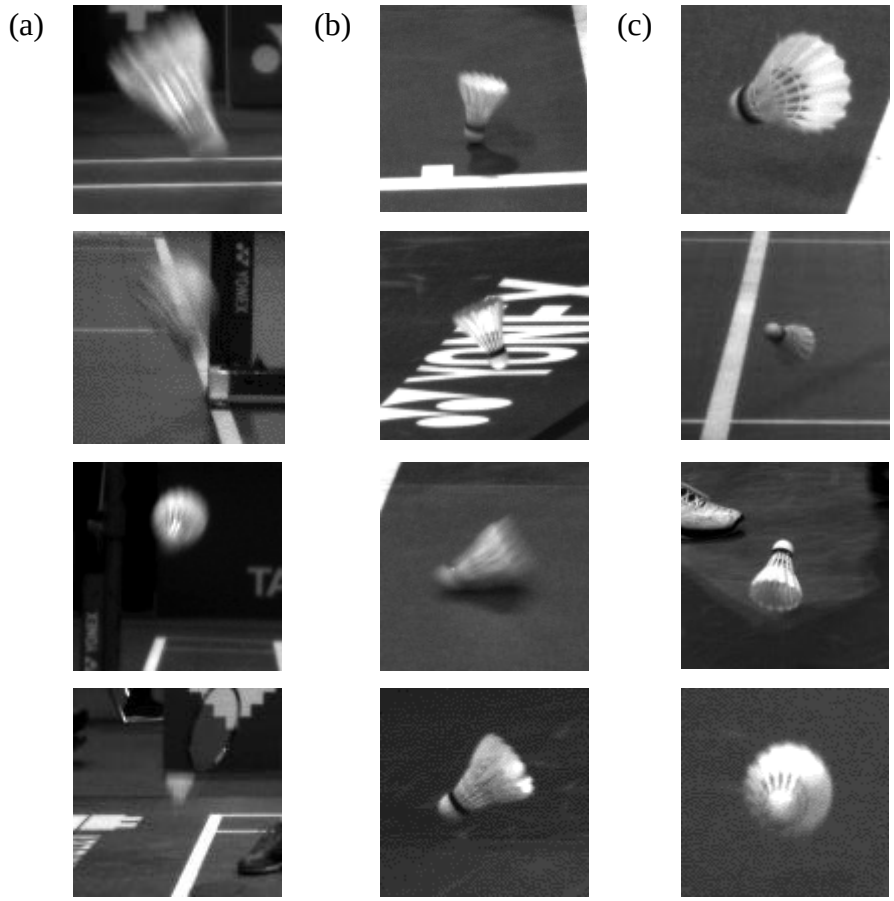
      Metody bazujące na wyszukiwaniu próbują odkryć najlepszy ślad poruszającego się obiektu poprzez rozległe przeszukiwanie, ze szczególnym naciskiem na śledzenie

małych obiektów. Wśród takich metod można wskazać metodę z wykorzystaniem wielokrotnych hipotez - Multiple Hypothesis Testing (MHT) zaproponowaną przez Blosteina i innych [32], która na podstawie określonej sekwencji pomiarów wyznacza prawdopodobieństwo, że to właśnie szukany obiekt ją wywołał. Ta metoda działa przy założeniu, że wartości intensywności tła i szumu są niższe niż średnia intensywność śledzonego obiektu. Metoda wiąże ze sobą ścieżki ruchu obiektów i dane obserwacyjne na przestrzeni kilku ramek tworząc wiele hipotez śledzenia. Hipotezy te są tworzone i utrzymywane w ramach dynamicznie rosnącego zbioru możliwych ścieżek obiektów, ale nie wszystkie z nich są równie prawdopodobne. MHT używa różnych kryteriów oceny (np. prawdopodobieństwa Bayesa lub oceny dopasowania na podstawie filtru Kalmana) do przypisania prawdopodobieństw poszczególnym hipotezom. Z czasem niektóre ścieżki mogą być odrzucane, ponieważ stają się mało prawdopodobne, a inne mogą zyskiwać na pewności. Niestety ta metoda napotyka wyzwania podczas śledzenia szybko poruszających się obiektów, ponieważ obszar wyszukiwania rośnie wykładniczo. Zaletą tej metody jest możliwość rozpoczynania śledzenia dla nowych obiektów pojawiających się w kadrze i kończenia dla tych, które z kadru zniknęły.

2. Metody śledzenia po detekcji (Track by detection)

   Pierwszą fazą tej metody jest lokalizacja śledzonych obiektów w pierwszej klatce. Kolejnym etapem jest modelowanie wyglądu śledzonego obiektu. Obiekt może ulec wizualnej transformacji z powodu zmiennych warunków oświetlenia, rozmycia ruchu, szumu obrazu, zmiany rozmiaru (gdy zbliża, lub oddala się od kamery), czy też z powodu rotacji. Ten etap ma na celu uchwycenie różnych cech i transformacji w celu poprawy niezawodności modelu. Obejmuje on konstruowanie opisów obiektów i modeli matematycznych w celu identyfikacji obiektów o zmiennym wyglądzie.

   Przykładowo, lotka do badmintona może bardzo różnie wyglądać [Rysunek 13].



Rysunek 13 Przykładowy widok lotki (a) w trakcie lotu, (b) w momencie odbicia i (c) po odbiciu

Ważnym etapem jest oszacowanie modelu ruchu. Taki model pozwala przewidzieć położenie obiektu na podstawie danych z poprzedniej/poprzednich klatek. Oszacowanie ruchu pozwala na określenie położenia obiektu w następnej, przyszłej klatce, a co za tym idzie ustaleniem współrzędnych obiektu i przewidywanego regionu obrazu, w którym z dużym prawdopodobieństwem może się znajdować śledzony obiekt.

Metody z tej grupy dobrze radzą sobie z sytuacjami, gdy pojawia się nowy obiekt w scenie. Śledzenie po detekcji jest najbardziej popularną metodą, a większość badań mieści się w tej kategorii. Należy jednak pamiętać, że skuteczność tych metod w dużej mierze zależy od dokładności użytego detektora. Wielu naukowców zajmujących się śledzeniem obiektów w zapisach wideo z zawodów sportowych używa jako detektora sieci YOLO [10, 48, 49, 50].

2.1. Metody bazujące na usuwaniu tła

Metody wykorzystujące ramki różnicowe i odejmowanie tła są popularne ze względu na swoją prostotę i wysoką wydajność oraz to, że dobrze sobie radzą w przypadku szybko poruszających się obiektów. Szczególnie często korzysta się z tych metod w

aplikacjach czasu rzeczywistego. Na przykład w publikacji [33] autorzy generują ramki różnicowe z dwóch kolejnych. Następnie wykrywają i śledzą piłkę tenisową oraz graczy na podstawie rozmiaru konturów na ramkach różnicowych.

Zhu i inni [34] w celu wykrywania ruchomych obiektów (dronów) generują cztery obrazy. Różnicowy obraz nr 1 - *O(1)* generowany na podstawie różnicy klatek *N* i *N*+1, obraz nr 2 - *O(2),* który jest obrazem różnicowym pomiędzy klatką *N+1* i *N+2*. Obraz *O(3)* generowany poprzez wykonanie logicznej operacji AND na obrazie *O(1)* i *O(2).* Czwarty obraz O(4) uzyskiwany jest poprzez operację XOR pomiędzy *O(2)* i *O(3)*. Ostatecznie obraz detekcji jest uzyskiwany poprzez wykonanie logicznej operacji AND na obrazach *O(1)* i *O(4)*. Pozostały na obrazie szum jest usuwany poprzez operację erozji[6] (ang. erode).

W swojej pracy Yin, Q i pozostali [35] do detekcji poruszających się obiektów na filmach satelitarnych używają akumulacyjnych ramek różnicowych, a następnie za pomocą filtru trajektorii ruchu usuwają fałszywych kandydatów - takich, których trajektorie nie są zgodne modelem ruchu śledzonych obiektów.

Natomiast Zhou, Y. i pozostali w swojej pracy [36] wskazują na wydajne i nienadzorowane podejście z wykorzystaniem konwolucyjnych sieci neuronowych. Stosują metodę usuwania tła w obrazowaniu ruchu szerokiego obszaru (WAMI). W pierwszym kroku dzięki usunięciu tła wyodrębnia się kandydatów o niskim kontraście. Kolejnym etapem jest usunięcie fałszywych kandydatów poprzez wytrenowaną konwolucyjną sieć neuronową uwzględniającą dane czasowe i przestrzenne.

Wykorzystując jako detektor sieć Faster R-CNN [37] Aguilar, C. i inni [38] zaproponowali metodę śledzenia wielu obiektów na nagraniach wideo zarejestrowanych przez satelity, która na wejściu otrzymuje ramkę różnicową wygenerowaną na podstawie 3 kolejnych ramek.

Autor w swoich badaniach skoncentrował się na eksplorowaniu metod właśnie z tej kategorii, ponieważ są wyjątkowo efektywne obliczeniowo, a układ pomiarowy (stacjonarne kamery) pozwala na ich zastosowanie w praktyce.

[6] Definicja wyjaśniona w Słowniczek pojęć punkt 2, strona 102

2.2. Metody klasyczne, z wykorzystaniem ekstrakcji cech – wykorzysujące różne metody przetwarzania obrazu w celu ekstrakcji kluczowych cech, a następnie klasyfikacji obiektów na podstawie tychże cech.

Metody te wykorzystują algorytmy i techniki bazujące na modelach matematycznych i heurystyce. Najczęściej etap detekcji wykonywany jest dwuetapowo. W pierwszej kolejności dokonywana jest ekstrakcja cech, a następnie za pomocą jakiegoś klasyfikatora dany obszar obrazu przypisywany jest do określonej klasy.

Do popularnych deskryptorów[7] cech należą:

- Histogram of Oriented Gradients (HOG) [39] – u podstaw tego deskryptora leży rozpoznawanie krawędzi na obrazie. Zakłada się, że lokalny wygląd i kształt obiektu na obrazie można opisać za pomocą rozkładu intensywności i kierunku gradientów. W celu wyznaczenia deskryptora obraz jest dzielony na małe sąsiadujące obszary zwane komórkami, a dla pikseli w każdej komórce obliczany jest histogram kierunków gradientu uwzględniający wartość gradientu.
- Haar-like features [40]. Deskryptor bazujący na kilku rodzajach elementów strukturalnych (prostokątnych obszarów zwanych kernelami), za pomocą których rozpoznawane są charakterystyczne obszary na obrazie – linie, krawędzie, szachownice (four-rectangle), itp.
- Scale-Invariant Features Transform (SIFT) [41]. Ideą tego deskryptora jest wyznaczenie kluczowych punktów dla szukanego obiektu oraz utworzenie wektora charakteryzującego każdy z nich.

Powiedzmy, że chcemy znaleźć na obrazie piłkę tenisową. Wiadomo, że cechą piłki tenisowej jest kształt okręgu i kolor jaskrawozielony. Możemy wyznaczyć prosty deskryptor zawierający kolor i kształt. Metoda polegająca na wydobyciu z obrazu tych obszarów, w których piksele mają kolor jaskrawozielony i sprawdzeniu czy wyekstrahowany obszar ma kształt okręgu jest wyjątkowo prostą metodą detekcji, która w pewnych określonych przypadkach może być wystarczająco dobra.

W zależności od problemu często konieczne jest opracowanie bardziej skomplikowanych modeli i metod. Na przykład, w swojej pracy [42] panowie Ahmadi, K.; Salari, E zaproponowali metodę składającą się z trzech etapów. W etapie pierwszym

[7] Patrz Słowniczek pojęć punkt 3, strona 102

dla każdej ramki generowane jest 6 podpasm wysokiej częstotliwości za pomocą transformacji falkowej Dual-Tree Complex Wavelet Transform (DT-CWT). Następnie potencjalni kandydaci wykrywani są przez detektor wykorzystującym adaptacyjny algorytm CFAR (Constant False Alarm Rate) służący do wykrywania szukanego obiektu na tle różnych zakłóceń. Ostatnim etapem jest klasyfikacja za pomocą SVM [43].

Filtr Kalmana [29] jest jednym z kluczowych elementów algorytmu śledzenia dronów opracowanego przez Son i in. [44]. Opracowane przez autorów metoda składa się z dwóch odrębnych trakerów (wykorzystujących przepływ ruchu, ang: motion flow, przepływ optyczny, ang: optical flow i histogram cech), predyktora (filtr Kalmana) i operacji udoskonalania. Pierwszy traker wykorzystuje model ruchu do identyfikacji poruszającego się obiektu, podczas gdy drugi wykorzystuje deskryptor na bazie histogramu cech kluczowych punktów. Następnie następuje porównanie histogramów obszarów zainteresowań z histogramem modelu i wybierany jest ten który jest najbardziej podobny. W kolejnym kroku położenie obiektu jest prognozowane za pomocą filtru Kalmana. Na koniec operacja udoskonalania jest wykorzystywana do ustalenia dokładnej lokalizacji obiektu.

2.3. Metody wykorzystujące sieci neuronowe

Głównym elementem metod z tej grupy jest sieć neuronowa, której zadaniem jest detekcja śledzonego obiektu.

Aktualnie, większość badaczy koncentruje się na udoskonalaniu metod detekcji obiektów wykorzystujących sieci neuronowe. Przełomowym momentem, od którego nastąpił rozkwit badań nad sieciami neuronowymi był rok 2012 kiedy model konwolucyjnej sieci neuronowej AlexNet [45] zwyciężył w konkursie ImageNet [46].

Cechą odróżniającą metody wykorzystujące sieci neuronowe od metod klasycznych jest to, że modele bazujące na sztucznych sieciach neuronowych nie wymagają uprzedniego przetwarzania obrazu i ekstrakcji cech, aczkolwiek mogą być zasilane przetworzonymi unormowanymi obrazami. Aby można było dokonać detekcji obiektu trenuje się model w procesie uczenia głębokiego (deep learning), z wykorzystaniem zbioru treningowego zawierającego różnorodne, dobre jakościowo przykłady. Przygotowanie dobrego zbioru nie jest zadaniem trywialnym i bardzo czasochłonnym. Dlatego bardzo często do trenowania i ewaluacji modeli wykorzystuje się zbiory danych dostępne w Internecie. Często również w

procesie uczenia sieci wykorzystuje się z już nauczony model i dostraja się go do nowych realiów (transfer learning). Jednym z najpopularniejszych zbiorów danych jest baza ImageNet [47] zawierająca 14,197,122 zaanotowane obrazy. Podczas procesu uczenia obliczane są wagi, których liczba w zależności od modelu może sięgać wielu milionów. Im większa liczba wag tym większa liczba wzorców może być zamodelowana przez sieć.

Niestety trening tak dużych modeli wymaga wysokowydajnego obliczeniowo i bardzo drogiego sprzętu. Dodatkowo pojawia się problem z wyjaśnialnością i interpretowalnością modelu. Przy tak skomplikowanych architekturach trudno jest precyzyjnie odpowiedzieć na pytanie dlaczego predykcja modelu była taka a nie inna.

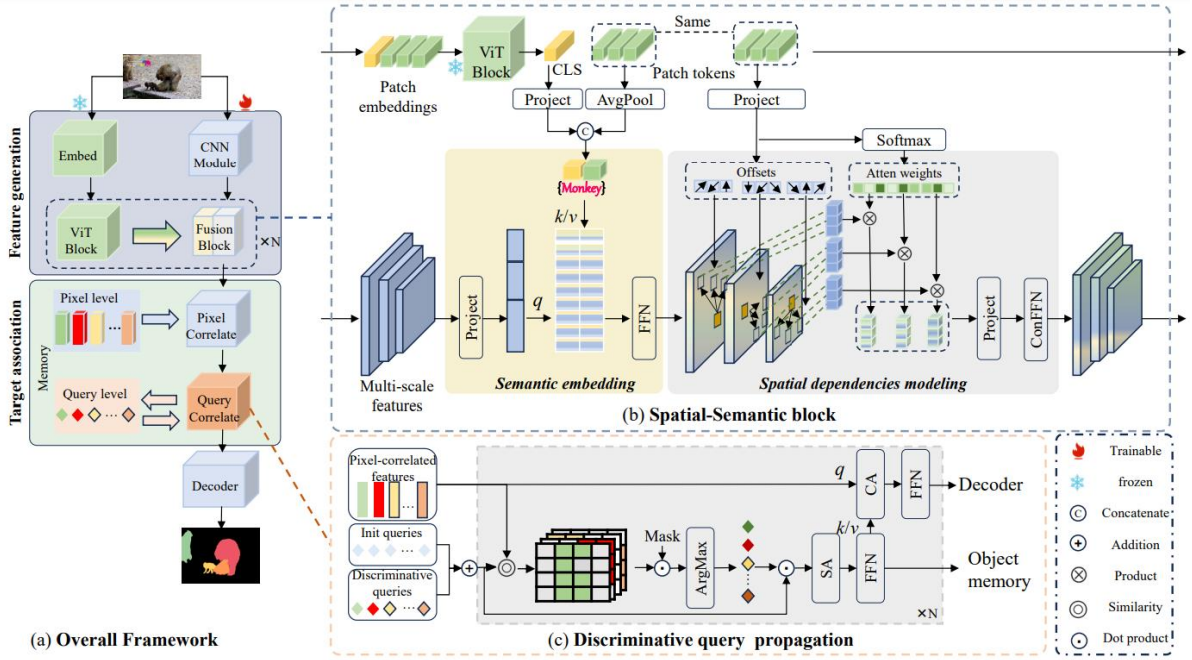
Ze względu na istotne ograniczenia czasowe – całkowity czas przetwarzania jednej klatki musi być krótszy niż 5 ms, oraz istotne ograniczenia finansowe, autor nie zdecydował się na dokładniejsze badanie metod śledzenia i detekcji korzystających z sieci neuronowych, ponieważ działały zbyt długo i opracował własny traker i detektor opisany w dalszej części rozprawy. W momencie, gdy autor rozpoczynał pracę nad systemem, przeanalizował benchmarki przedstawione przez innych badaczy [48, 49] z których wynikało, że czas inferencji dla badanych SOTA (State Of The Art) detektorów wynosił 51ms (Yolo v3), 100ms (Faster R-CNN [37]) z wykorzystaniem GPU NVIDIA GeForce GTX TITAN X.

Od tamtego czasu opracowano wiele nowych modeli sieci neuronowych oraz poprawiono moc obliczeniową kart graficznych. Aktualnie badacze podają możliwość uzyskania nawet 161 FPS dla modelu YOLOv7 [50] na karcie GPU NVIDIA Tesla V100. Jest to jednak ciągle nieznacznie dłużej niż 5ms.

Jednym z najbardziej znanych konkursów śledzenia obiektów w strumieniu wideo jest VOT challenge [51] organizowany od 2013 r. Wszystkie nadesłane w 2024r na konkurs modele wykorzystywały sieci neuronowe, a analizując rezultaty, można zauważyć, że w pierwszej piątce aż 4 modele wykorzystują, jako jeden z głównych elementów nadesłanego rozwiązania, Segment Anything Model (SAM) [52], lub jego modyfikację HQ-SAM [53]. Autor również weryfikował możliwość wykorzystania SAM przy rozwiązaniu badanego problemu (segmentacji lotki), a swoje obserwacje opisał w rozdziale 3.4, podrozdział *Porównanie zaproponowanej metody z Segment Anything Model* (strona 118).

Na uwagę zasługuje zwycięzca edycji 2024 - *S3_Track* [54], który uzyskał najwyższą wartość metryki według której były porównywane modele - Q (Tracking Quality) [55] co stanowi prawie 10% poprawę w porównaniu z drugim najlepszym modelem. *S3_Track* to jednoetapowy

traker, który śledzi wszystkie obiekty równocześnie. Składa się z modułu generowania cech uwzględniającego semantykę i modułu skojarzeń docelowych, który uwzględniające semantykę koreluje cechy z pikselami. Powstałe cechy korelacji są dekodowane do masek segmentacji. Ten model rozszerza model XMem [56]. Architektura modelu jest pokazana na rysunku [Rysunek 14]. Warto zauważyć, że w swojej architekturze zawiera moduł konwolucyjnej sieci neuronowej (architektura znana od 1989r.) i moduł wizualnego transformera [57] (Vision Transformer ViT), który jest stosunkowo nową architekturą (znaną od 2020r.). Model osiąga wydajność 8FPS z wykorzystaniem karty graficznej NVIDIA V100.



Rysunek 14 Architektura modelu S3_track[8]

W literaturze można się spotkać z wieloma metodami śledzenia obiektów. Poniżej przedstawiono wybrane metody, inne niż te wspomnianych wcześniej, wraz z ich wadami i zaletami.

Filtr Kalmana [29]

Filtr Kalmana, opracowany w 1960r., szacuje położenie obiektu i przewiduje jego ruch w kolejnej chwili czasowej na podstawie wewnętrznej reprezentacji stanu obiektu, w tym jego położenia, prędkości, a nawet przyspieszenia.

Aby przewidzieć przyszły stan, filtr wykorzystuje informacje z poprzedniego stanu obiektu i model matematyczny analizujący ruch obiektu. Model uwzględnia również wszelką

[8] Źródło: https://arxiv.org/abs/2407.07760

niepewność w ruchu obiektu (szum procesowy i szum pomiarowy). Oryginalnie, filtr Kalmana został opracowany dla układów liniowych i znalazł szerokie zastosowania w wielu gałęziach techniki. Modyfikacje filtru Kalmana pozwalają na pracę z układami niestacjonarnymi (zmienny w czasie model) i nieliniowymi (Rozszerzony Filtr Kalmana - użyty przy misji Apollo)

Zalety:

- Matematyczny model nie wymagający żadnego szkolenia.
- Wydajny obliczeniowo.

Wady:

- Mniejsze możliwości w porównaniu do nowoczesnych algorytmów wykorzystujących uczenie głębokie.
- Aby działał poprawnie muszą być spełnione pewne założenia, na przykład stałe przyspieszenie obiektu (dla wersji liniowej).
- Algorytm nie sprawdza się dobrze w scenariuszach z losowym ruchem obiektu.

KCF [58]

Kernel Correlation Filters to model matematyczny, który wyznacza cechy obiektu i uczy się odróżniać je od tła.

Po wyznaczeniu cech obiektu używa filtrów korelacji, aby skonstruować wielowymiarową relację między cechami, a prawdziwym obiektem. Skanując obraz wokół ostatniej znanej lokalizacji obiektu wyznacza się obszar o najwyższej korelacji. Przewiduje się, że obszar o najwyższej korelacji będzie zawierał obiekt.

Zalety:

- Szybkie obliczenia. Ponad 300 FPS.
- Niskie wymagania pamięciowe.
- Skuteczny w śledzeniu obiektów o złożonych teksturach, dzięki modelowanie nieliniowych zależności w danych.
- Nie wymaga dużych zbiorów danych do treningu. Wystarczy początkowy model obiektu, a algorytm może śledzić obiekt bazując na iteracyjnie aktualizowanym modelu.

Wady:

- Tradycyjny KCF napotyka wyzwania w takich warunkach, jak zmienna skala obiektu, kształt, złożona deformacja, lub gdy obiekt dotykają granic obrazu.

- W złożonych scenach gdzie wiele obiektów ma podobne cechy, może przeskakiwać z obiektu na obiekt, a gdy obiekt zniknie z pola widzenia kamery lub zostanie przesłonięty przez inny obiekt, śledzenie może zostać przerwane.

Median Tracker[59]

Median Tracker bazuje na metodzie Lucasa-Kanade [60] - pola przepływu optycznego, która zakłada, że jasność pikseli śledzonego obiektu jest niezmienna w czasie, a sąsiednie piksele mają ten sam wektor ruchu, a sam ruch jest mały w kolejnych klatkach. Median Tracker śledzi ruch obiektu do przodu i do tyłu w czasie i przewiduje dalszą pozycję obiektu w czasie rzeczywistym.

Zalety:

- Wystarczająco duża prędkość i dokładność śledzenia, jeśli obiekt nie jest przesłaniany przez inne obiekty, a pomiędzy klatkami nie zmienia istotnie swojego położenia.
- Nie trzeba znać modelu dynamiki poruszającego się obiektu.

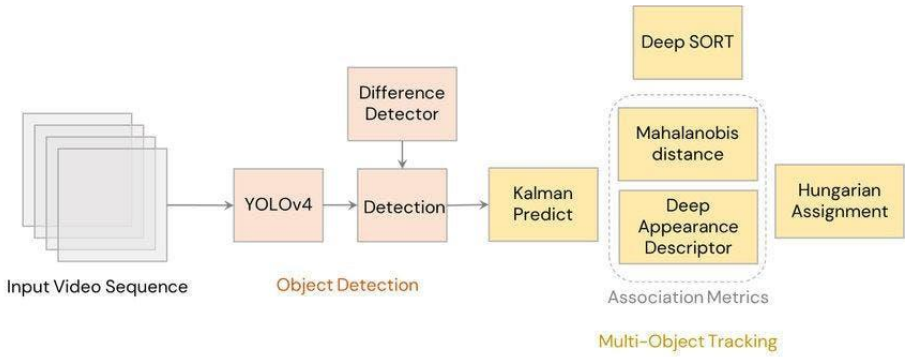
Wady:

- Duże prawdopodobieństwo utraty obiektu przy dużej prędkości jego ruchu.
- Po utraceniu obiektu nie można go ponownie odnaleźć

DeepSORT [61]

Algorytm Deep Simple Online Realtime Tracking (DeepSORT) pozwala na śledzenie równocześnie wielu obiektów i jest rozszerzeniem oryginalnego algorytmu SORT. Oryginalny algorytm SORT używa filtrów Kalmana do przewidywania ruchu obiektów i węgierskiego algorytmu (inaczej Kuhn Munkres algorytm [62]) do skojarzeń obiektów klatka po klatce.

DeepSORT wykorzystuje dodatkową neuronową sieć splotową (CNN) jako ekstraktor cech. Określają one wygląd obiektu i pozwalają algorytmowi odróżniać ruchome obiekty od statycznych. Architekturę algorytmu przedstawiono na rysunku poniżej



Rysunek 15 Architektura DeepSORT [9]

Zalety:

- Wydajna implementacja DeepSort zapewnia śledzenie w czasie zbliżonym do rzeczywistego.
- Może obsługiwać dowolną sieć detekcji wybraną przez użytkownika, taką jak YOLO lub RCNN.
- Odporny na okluzje i może rozróżniać różne obiekty w złożonych scenariuszach.

Wady:

- Trening sieci detekcji w celu uzyskania wysokiej dokładności wymaga rozległego zestawu danych.
- Duża zależność od jakości detektora.
- Wysokie wymagania obliczeniowe (głównie zależne od wykorzystanej jako detektor sieci neuronowej).

GOTURN [62] Generic Object Tracking

Jeden z pierwszych algorytmów śledzenia wykorzystujący głęboką sieć neuronową. Do sieci wprowadzane są dwa obrazy: „poprzedni" i „bieżący". Na „poprzednim" obrazie znana jest pozycja obiektu, natomiast na „bieżącym" obrazie pozycja obiektu musi zostać przewidziana. W ten sposób oba obrazy są przepuszczane przez splotową sieć neuronową, której wyjściem jest zestaw 4 punktów reprezentujących pola ograniczające przewidywaną pozycję obiektu.

Zalety:

- Działa z prędkością nawet 100 klatek na sekundę.
- Stosunkowo dobra odporność na szumy i zakłócenia.

[9] Źródło: https://www.researchgate.net/figure/Architecture-of-Deep-SORT-Simple-online-and-real-time-tracking-with-deep-association_fig2_353256407 Autor: Addie Ira Borja Parico

Wady:

- Dokładność śledzenia obiektów zależy od jakości danych, na których model został wytrenowany. Jeśli dane użyte do treningu nie były wystarczająco zróżnicowane, algorytm może gubić obiekty o nietypowym zachowaniu.
- Gubienie obiektów, które były zasłonięte przez dłuższy czas – brak mechanizmu odzyskiwania obiektu po zniknięciu.
- Przeskakuje ze śledzonego obiektu na inny, jeżeli prędkość pierwszego jest zbyt duża.

ByteTrack [64]

ByteTrack to algorytm śledzący wiele obiektów, którego skuteczność w dużym stopniu zależy od skuteczności detektora obiektów. Bazując na mechanizmie DeepSORT, ByteTrack wymaga, aby detektor wygenerował obszary detekcji bez względu na wysokość prawdopodobieństwa detekcji obiektu. W przypadku pól detekcji z wysokimi prawdopodobieństwami detekcji, ByteTrack wykonuje dopasowywanie cech i dopasowywanie IoU[10] (Intersection Over Union), podczas, gdy dla obszarów o niskimi prawdopodobieństwie tylko dopasowywanie IoU.

ByteTrack stara się zamodelować sposób, w jaki obiekty poruszają się i oddziałują na siebie w dynamicznym środowisku, dzięki czemu dobrze się sprawdza w śledzeniu obiektów w gęsto zaludnionych scenach, gdzie powszechne są przesłonięcia i szybkie ruchy, oraz utrzymywanie dokładnej identyfikacji nawet wtedy, gdy obiekty tymczasowo przestają być widoczne lub zostają zasłonięte.

Zalety:

- Możliwość integracji z dowolnymi detektorami, takimi jak YOLO8.
- Dość dobra wydajność (zależna głównie od detektora) około 30 FPS (YOLO8) na jednej karcie graficznej.
- Śledzenie wielu obiektów, niezależnie od złożoności środowiska.

Wady:

- Zależy od jakości detektora.
- Śledzenie małych obiektów może być problematyczne.
- Może mieć trudności w utrzymaniu śledzenia w dynamicznych scenach, gdzie obiekty szybko zmieniają położenie lub wygląd.

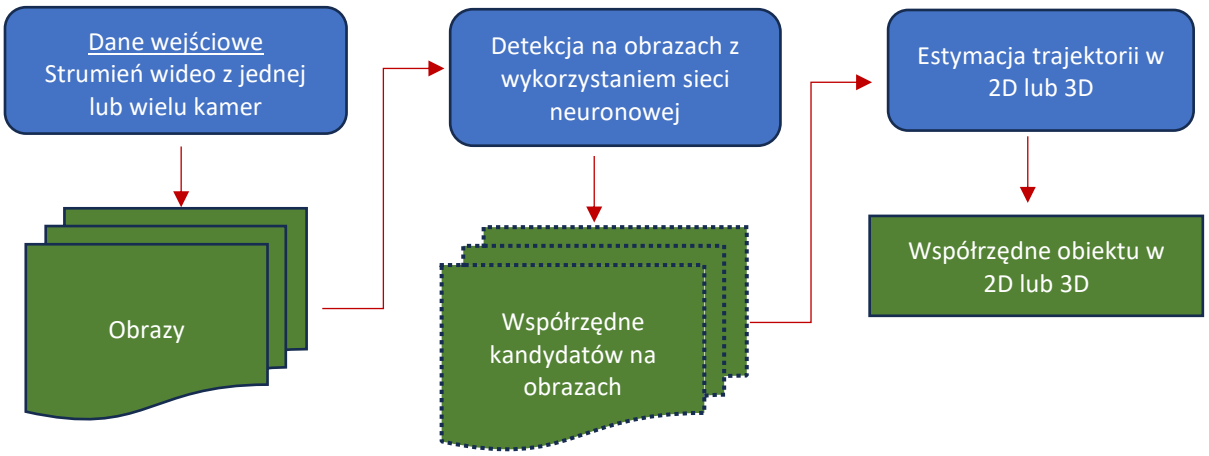
[10] Definicja podana w Słowniczek pojęć punkt 4 na stronie 93

Badacze do oceny skuteczności proponowanych rozwiązań często wykorzystują standardowe zbiory danych takie VOT [55], ALOV [65], OTB [66]. Ciekawych obserwacji dokonali autorzy publikacji „The World of Fast Moving Objects" [67]. Zauważyli oni, że najlepsze trakery, nie zawsze dobrze sobie radzą dla zbiorów z szybko poruszającymi się obiektami (ang: Fast Moving Objects – FMO). Do podobnych wniosków doszli Wei Chen i inni [7]. Wynika to z tego, że w powszechnie dostępnych zbiorach danych nagrania zawierające szybko poruszające się obiekty są nielicznie reprezentowane.

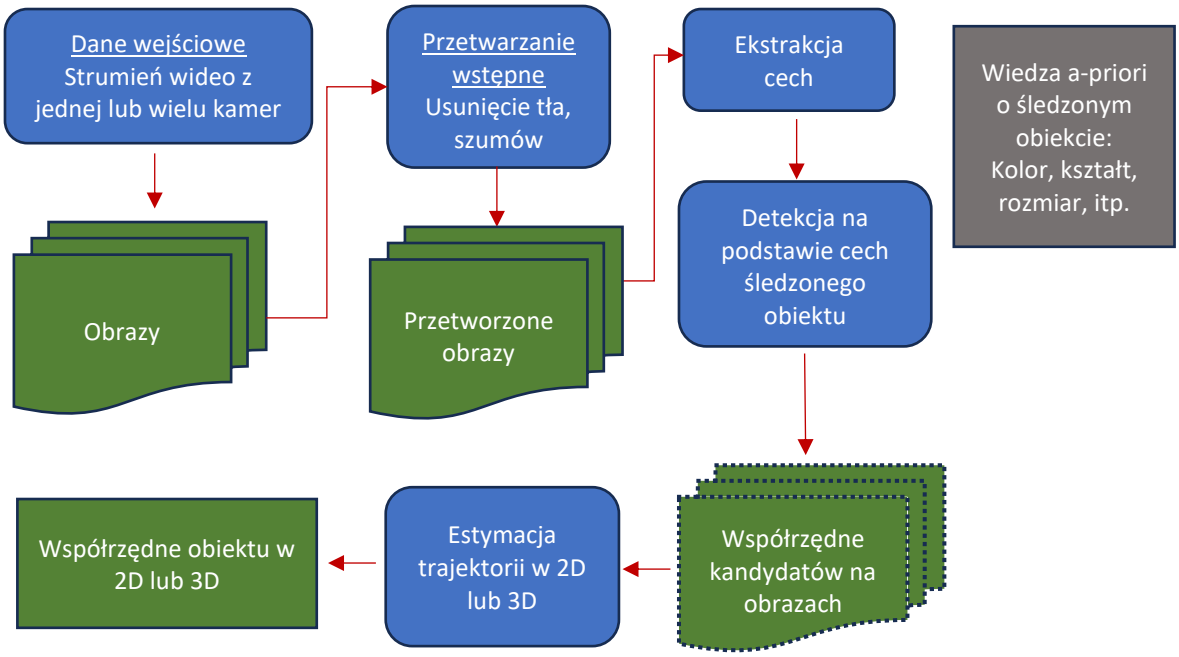
W wielu różnych sportach gra się piłką, lotką tylko w badmintonie. Z tego też powodu problem detekcji i śledzenia piłki [68, 69] jest częściej eksplorowany przez badaczy niż problem śledzenia lotki do badmintona. Czytając publikacje dotyczące problemu śledzenia piłki lub lotki można spotkać się z dwoma podejściami badaczy do rozwiązania tego zagadnienia.

- Z wykorzystaniem sieci neuronowych.
- Z wyznaczeniem kluczowych cech śledzonego obiektu i klasyfikacją obiektów na podstawie tychże cech.

Jako, że piłka i lotka są obiektami, które uderzone poruszają się po określonej trajektorii to w obu podejściach badacze wykorzystują modelowanie ruchu śledzonego obiektu w celu poprawy skuteczności śledzenia. Schematycznie przedstawiono te metody na rysunkach [Rysunek 16, Rysunek 17].



Rysunek 16 Schemat działania metod śledzenia piłki/lotki wykorzystujących sieci neuronowe (opracowanie własne)



Rysunek 17 Schemat działania metod śledzenia piłki/lotki na podstawie cech śledzonego obiektu (opracowanie własne)

Przykładowo Xinguo Yu i Hon Wai Leong zaproponowali dla piłki nożnej w [69] dwufazowy algorytm, który w pierwszej fazie generuje dla każdej klatki zestaw kandydatów na piłkę (na podstawie koloru, kształtu, rozmiaru), a następnie wykorzystuje je do obliczenia zestawu trajektorii, które to są wykorzystywane jako dodatkowa informacja dla trakera. Autorzy podają dokładność swojego algorytmu na poziomie 81%

Wiele prac badawczych dotyczy w śledzenia piłki tenisowej [33, 70, 71, 72, 73]. Zarówno piłka tenisowa jak i lotka badmintonowa są szybko poruszającymi się obiektami [67], jednakże zarówno kształt jak i dynamika poruszającej się lotki znacznie różni się od piłki tenisowej. Z tego powodu znajdujące się w literaturze metody detekcji i śledzenia piłki nie mogą zostać bezpośrednio zastosowane detekcji i śledzenia lotki. Zauważyli to też inni badacze zajmujący się problemem śledzenia lotki badmintonowej [74, 75].

W publikacji *Visual Tracking Method of a Quick and Anomalously Moving Badminton Shuttlecock* [74] autorzy śledzą lotkę na podstawie zapisu wideo z wielu kamer. Gdy lotka porusza się szybko i jest rozmyta (efekt motion blur), autorzy wykorzystują kształt rozmazanej lotki w celu jej lokalizacji. Dokładność lokalizacji lotki podana przez autorów wynosi 4cm.

W celu wyznaczenia pozycji lotki w 3D autorzy innej publikacji „Using FTOC to track shuttlecock for the badminton robot” [7] również używają systemu pomiarowego, gdzie wiele kamer obserwuje tę samą przestrzeń. Po usunięciu tła, następuje śledzenie lotki po detekcji, do której wykorzystywana jest metoda AdaBoost [76], a deskryptorem cech jest Haar-like features.

Wydajność zaproponowanej metody wynosi 10.65 FPS (obliczenia wykonywane na karcie graficznej NVIDIA M1200).

W [75] autorzy dla każdej ramki generują zbiór elementów na podstawie ramki różnicowej wygenerowanej z trzech kolejnych ramek. Wszystkie wygenerowane elementy są traktowane jako kandydaci na lotkę lub kandydaci na zawodnika. Następnie na podstawie oceny rozmiaru kandydata następuje przydział do jednej z grup – lotka, zawodnik. Ostateczna decyzja zostaje podjęta przy użyciu zestawu filtrów (rozmiar obiektu, kolor, trajektoria, prędkość poruszania się). Skuteczność zaproponowanego przez autorów trakera jest na poziomie 84%. Niewątpliwą zaletą tej metody jest jej szybkość działania wynosząca 350 ramek na sekundę. Podobne podejście, ale wykorzystujące technikę kształtu z sylwetki w celu modelowania w 3D zaproponowali Shishido i Hidehiko [74].

W 2021 roku Zhiguang Cao wraz z innymi [10] zaproponował dwie nowatorskie sieci o nazwach M-YOLOv2 i YOLOBR bazujące na Tiny YOLOv2. Autorzy zmodyfikowali architekturę Tiny YOLOv2, oraz funkcję straty, tak aby poprawić czas jaki potrzebuje sieć do wykrywania małych obiektów, takich jak lotki, oraz aby zachować więcej informacji semantycznych o małych obiektach. Oni podobnie jak Shishido i Hidehiko używają stereowizji i dwukamerowego zestawu pomiarowego ZED, a wydajność ich metody wynosi 22FPS (NVIDIA M1200).

Warto zwrócić uwagę, że żadna z zaproponowanych przez innych autorów metod detekcji i śledzenia lotki nie ma, ani wystarczającej dokładności, ani wydajności oczekiwanej przez rynek (patrz rozdział 1.2). Metody wykorzystujące sieci neuronowe są przede wszystkim zbyt wolne. Autorzy przywołanych publikacji podają wydajność od 10 do 22 FPS, co jest niewystarczające w celu precyzyjnego określenia lokalizacji szybko poruszającej się lotki. Rejestracja wideo z prędkości 20 FPS oznacza, że pomiędzy kolejnymi klatkami mija 50 ms. Biorąc pod uwagę uderzenie clear, po którym przy linii końcowej lotka porusza się z prędkością graniczną 6,7m/s [19] to i tak między kolejnymi klatkami pokona dystans 33cm. To nastręcza dodatkowe problemy – konieczna staje się ekstrapolacja położenia lotki. Proponowane przez badaczy, metody klasyczne są natomiast niewystarczająco dokładne – rynek oczekuje dokładności na poziomie 13mm, a proponowane metody mają dokładność około 40mm. Skuteczność trakerów na poziomie 81-84% też nie jest szczególnie wysoka.

W porównaniu z większymi lub bardziej jednorodnymi obiektami (np. piłka do siatkówki) śledzenie lotki jest zdecydowanie trudniejsze i wymaga istotnych zasobów sprzętowych [77].

Z tego też powodu, autor podjął decyzję o stworzeniu własnego trakera bazującego na metodzie usuwania tła. W szczególności, autor świadomie podjął decyzję o rezygnacji z trakera wykorzystującego sieci neuronowe – głównie z powodu długiego czasu inferencji. Autor w swoim rozwiązaniu, jako jeden z elementów, wykorzystuje filtr Kalmana, ponieważ jest to niewymagający treningu model matematyczny, który jest wyjątkowo efektywny obliczeniowo, a wymagania, które ma, aby działał poprawnie są spełnione w badanym scenariuszu - lotka przy podłożu porusza się ruchem o stałym przyspieszeniu.

2.2. Wyznaczenie momentu i miejsca odbicia lotki o podłoże

Rozwiązanie, opisanego wcześniej, problemu detekcji i śledzenia lotki jest jednym z elementów pracy badawczej autora koniecznym do precyzyjnego określenia miejsca upadku lotki w stosunku do referencyjnych linii kortu. Autor znalazł tylko jedną publikację „Computer vision approach to automatic linesman" autorstwa Leong, L.H.; Zulkifley, M.A.; Hussain, A.B. [78], która porusza badany przez autora temat - upadku lotki w korcie lub poza nim. W przeciwieństwie do autora, wspomnieni badacze nie testowali swojego rozwiązania w realnych warunkach – podczas turniejów badmintona. Dodatkowo, ich kamera obserwowała tylko tylną linię, a w 61 wybranych do testów sekwencjach wideo nie było zawodników - była widoczna tylko sama lotka. Ograniczyli również rodzaj zarejestrowanych uderzeń do dwóch - smecz i drop. Kolejnym uproszczeniem było to, że wybrali tylko takie sekwencje, w których ich detektor rozpoznał lotkę na wszystkich klatkach. Detekcję lotki przeprowadzili usuwając tło przez generowanie ramek różnicowych. Moment i miejsce odbicia w tej pracy został wybrany dla klatki, w której współrzędna y konturu lotki wyznaczonego z ramki różnicowej miała największą wartość i autorzy nie skupiali się na precyzyjnej jej segmentacji (w szczególności całkowicie pominęli wpływ cienia lotki na ramkę różnicową). W realnych warunkach jest to niezwykle istotne, w celu dokładnej lokalizacji miejsca odbicia lotki o podłoże. Ostatecznie dokładność zaproponowanego rozwiązania wyniosła 80% i jest istotnie niższa od osiągniętej przez autora – 94%.

Precyzyjna segmentacja jest istotna w celu wyznaczenia możliwie dokładnie miejsce odbicia lotki o podłoże. Tradycyjne podejście do problemu segmentacji bazuje na ekstrakcji niskopoziomowych cech opisujących właściwości krawędzi, kształtów, rozkład gradientów czy kolorów. Metody tradycyjne zazwyczaj mają niskie wymagania obliczeniowe, ale często też wymagają dopasowywania parametrów przez użytkownika. Aktualnie badacze skłaniają się ku metodom wykorzystującym sieci neuronowe, które ewoluowały w kierunku rozpoznawania tego co się dzieje na obrazie.

Omówienie metod potencjalnie przydatnych do segmentacji lotki

Analizując literaturę na temat segmentacji małych obiektów, a w szczególności z uwzględnieniem segmentacji piłki/lotki w sporcie, autor nie natrafił na wiele publikacji poruszających temat. Większość badaczy zajmujących się analizą zapisów wideo z zawodów sportowych, porusza w swoich publikacjach tylko sam problem detekcji, a nie detekcji i segmentacji. W tym rozdziale autor dokonuje przeglądu uznanych metod segmentacji (wykorzystywanych nie tylko w sporcie), które mogą być potencjalne przydatne do segmentacji lotki.

Na krótkie omówienie zasługuje kilka prac poruszających temat segmentacji lotki lub piłki.

T. Dierickx w swojej pracy [79] najpierw usuwa tło, a następnie na uprzednio przetworzonym obrazie dokonuje segmentacji lotki z wykorzystaniem metody progowania globalnego.

W swojej pracy [80] D'Orazio i inni przetwarzają wejściowy obraz wyznaczając krawędzie, a następnie wyszukują okręgi lub części okręgów w celu segmentacji piłki futbolowej wykorzystując metodę bazującą na Circle Hough Transform [81].

Metodę wykorzystującą splotową sieć neuronową zaproponowali Zandycke, Gabriel i inni [82], w której ich sieć (nazwana Ball R-CNN) została wytrenowana do predykcji mapy cieplnej piłki. Na etapie inferencji dodatkowe reguły wybierają spośród kandydatów na piłkę jednego najlepszego wykorzystując między innymi informacje o dynamice ruchu piłki. Wyniki zaproponowanego modelu porównują z wynikami uzyskanymi za pomocą sieci Mask-RCNN. Zaproponowany model działa z prędkością 38.39 FPS (obliczenia na karcie Nvidia GTX 1080 Ti dla obrazów o rozdzielczości 1024x512px) i jest istotnie szybszy od Mask-RCNN – 4.33 FPS.

Poniżej przedstawiono wady i zalety potencjalnie przydatnych do segmentacji lotki ogólnych metod segmentacji obiektów.

Progowanie

- globalne,
- lokalne,
- adaptacyjne,
- metoda Otsu [79]

Metody wykorzystujące progowanie polegają na wybraniu tych pikseli obrazu, których charakterystyka (intensywność, kolor) jest w określonym przedziale wartości.

Zaletą tej metody jest to, iż jest szybka i prosta, wadą jest trudność w optymalnym doborze wartości progowych.

Wyznaczenie krawędzi [84, 85]

Metoda ta zakłada, że pomiędzy różnymi obiektami następuje gwałtowna zmiana jasności pikseli. Po wykryciu krawędzi, obszary otoczone nimi mogą być zidentyfikowane jako obiekty.

Zalety:

- Wysoka skuteczność w znajdowaniu konturów i granic obiektów.
- Dobrze radzi sobie w obrazach o wyraźnych i ostrych granicach.

Wady:

- Może generować zbyt wiele krawędzi, komplikując segmentację.
- Wymaga dodatkowych kroków, aby połączyć krawędzie i utworzyć zamknięte kontury.

Segmentacja bazująca na regionach [86, 87]

- rozrost regionów,
- podział regionów,
- metoda działów wodnych

Metody te polegają na dzieleniu obrazu na obszary na podstawie jednorodności pewnych właściwości (koloru, tekstury).

Metoda rozrostu regionów startując z punktów startowych, łączy je w coraz większe obszary (dodając piksele) na podstawie spełnienia kryterium jednolitości. Wynik segmentacji często zależy od wyboru punktów startowych.

W metodzie podziału regionów następuje rekurencyjny podział obrazu na coraz to mniejsze obszary i poszukiwanie obszarów jednolitych.

Zalety:

- Dobrze segmentuje obszary o jednolitych cechach, nawet przy niskim kontraście (o ile regiony są jednorodne).

Wady:

- Wrażliwa na wybór punktów startowych i parametrów progów.
- Może wygenerować zbyt dużą liczbę małych regionów.

Metoda działów wodnych

Metoda działów wodnych rozpoczyna tworzenie regionów (rozlewisk) począwszy od lokalnych minimów intensywności jasności pikseli, i wyznacza granice w miejscach spotykania się rozlewisk lub największych gradientów.

Zalety:

- Działa dobrze w przypadku obrazów z wyraźnymi lokalnymi minimami.
- Skuteczna w segmentacji elementów o wyraźnych granicach.

Wady:

- Częstym problemem algorytmu jest nadmierna segmentacja (oversegmentation) z powodu dużej liczby minimów lokalnych.
- W celu poprawy rezultatów często wymaga dodatkowego przetwarzania, np. zastosowania markerów

Metody grafowe [88, 89]

Metody grafowe stosują algorytmy z teorii grafów do celów segmentacji.

Najpopularniejszy jest algorytm Graph-cut [90], który traktuje każdy piksel obrazu jako węzeł grafu. Wprowadza węzły oznaczające obszary tła i źródła i łączy każdy piksel z węzłami obszarów krawędziami, których waga oznacza prawdopodobieństwo przynależności do tła lub źródła. Następnie wyszukuje rozcięcia (cuts) rozdzielające obiekty tła i źródła, którego miarą jest zbiór krawędzi, a wśród tych przecięć wyszukuje minimalne (takie którego rozmiar waga lub energia jest najmniejsza)

Zalety:

- Pozwala na uzyskanie precyzyjnych rezultatów.
- Skuteczna dla obrazów o wyraźnej różnicy pomiędzy tłem i źródłem.

Wady:

- Spora złożoność obliczeniowa, szczególnie dla dużych obrazów.
- Aby uzyskać zadowalające rezultaty należy prawidłowo dobrać różne parametry algorytmu (zależne od przetwarzanego obrazu).

Active Contour Models (ACMs) Atywne kontury - Snake

Aktywny kontur - to model pewnej deformowalnej krzywej, która wykorzystywana jest do obrysu kształtów. Wymaga zainicjowania konturu niedaleko obiektu, który później zostaje przesuwany w stronę obiektu. Jest metoda zaproponowana w 1988r przez Kass'a [91], która dopasowuje krzywą do granic obiektu minimalizując pewną funkcję energii. Od tamtej pory wielokrotnie udoskonalana, a ostatnie prace połączyły ACM z sieciami głębokimi [92].

Zalety:

- Dobrze radzi sobie z dokładnym dopasowaniem kształtu obiektu, nawet w przypadku złożonych konturów.
- Model aktywnego konturu może dynamicznie dostosowywać swój kształt do złożonych obiektów, nawet jeśli ich granice są nieciągłe lub zakryte.
- Może radzić sobie z obiektami o słabo widocznych granicach.
- Oprócz krawędzi, model może być rozszerzony o inne cechy obrazu, takie jak intensywność, kolor czy tekstura.

Wady:

- Wynik segmentacji może być wrażliwy na początkowe ustawienie krzywej. Zła inicjalizacja może prowadzić do niepoprawnej segmentacji.
- ACM minimalizuje pewną funkcję energii, co może prowadzić do utknięcia w lokalnym minimum i braku poprawnego dopasowania do obiektu, zwłaszcza w przypadku szumów czy niewyraźnych granic.
- Model preferuje segmentowanie obiektów o stosunkowo gładkich konturach. Segmentacja obiektów o ostrych kątach lub nieregularnych krawędziach może być mniej skuteczna.

- Obliczenia związane z minimalizacją energii i deformacją krzywej mogą być kosztowne obliczeniowo, co sprawia, że metoda może być wolna, zwłaszcza w przypadku dużych obrazów.
- Krawędzie obiektów mogą być zakłócone przez szum w obrazie, co utrudnia precyzyjne dopasowanie konturu. Może być wymagane dodatkowe przetwarzanie obrazu w celu usunięcia szumu.

Metody wykorzystujące sieci neuronowe

Mask R-CNN [93]

Mask Region-based convolutional neural network rozszerza architekturę sieci faster R-CNN poprzez dodanie kolejnej gałęzi, która przewiduje maski segmentacji we wskazanych regionach zainteresowań.

W pierwszym etapie następuje ekstrakcja cech z obrazu za pomocą bazowej sieci splotowej (CNN). Te cechy są używane do dalszego przewidywania regionów, detekcji obiektów i segmentacji.

Drugim etapem jest propozycja regionów (Region Poposal Network - RPN), w których prawdopodobnie znajdują się obiekty.

W ostatnim etapie następuje predykcja klas obiektów i wybór regionu o najwyższej wartości IoU oraz wygenerowanie dokładnej maski segmentacyjnej.

Zalety:

- Potrafi rozróżniać i segmentować poszczególne obiekty (instancje) tej samej klasy (np. kilka balonów na jednym obrazie). Każdy obiekt ma własną maskę segmentacyjną.
- Wysoka precyzja w wyznaczaniu granic obiektów.
- Elastyczna i modularna architektura. Można łatwo modyfikować poszczególne komponenty np. zmienić sieć bazową, w zależności od potrzeb i zasobów sprzętowych.
- Detekcja i segmentacja w jednym modelu.
- Skalowalność do obrazów o różnych rozdzielczościach.

Wady:

- Wymaga dużych zasobów GPU i pamięci, co może ograniczać jego zastosowanie do urządzeń o wystarczających parametrach sprzętowych.

- Działa wolniej w porównaniu do prostszych modeli detekcji, zwłaszcza przy obrazach o dużej rozdzielczości lub zawierających wiele obiektów.
- Wymaga dużych, dobrych jakościowo zestawów danych do treningu.
- Problemy z małymi obiektami. Dla małych obiektów dostępne są ograniczone informacje o cechach, ponieważ są one generowane z wykorzystaniem mniejszej liczby pikseli. Utrudnia to modelowi zidentyfikowanie precyzyjnych granic małych obiektów. Dodatkowo RPN może mieć trudności z wygenerowaniem odpowiednich regionów zainteresowania, które precyzyjnie otaczają niewielkie obiekty.
- Brak segmentacji semantycznej, gdzie każdemu pikselowi na obrazie przypisywana jest etykieta klasy - segmentuje obiekty na poziomie instancji.

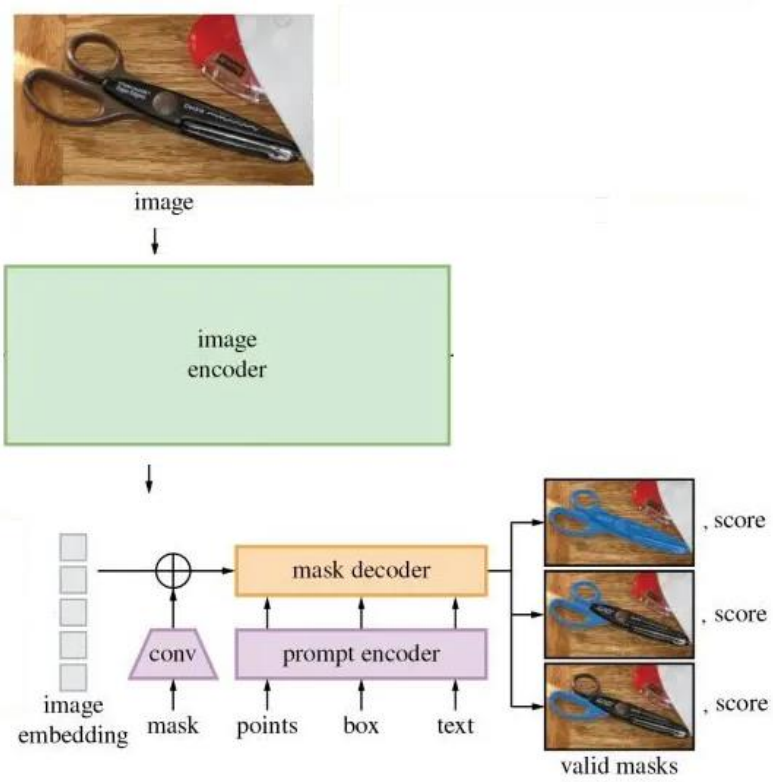
SAM [94]

Segment Anything Model to model, który pozwala na segmentację dowolnego obiektu (tzw. zero-shot segmentation) w przeciwieństwie do innych modeli (np. Mask R-CNN, które to potrafią znajdować tylko te klasy obiektów, dla których zostały wytrenowane). W celu segmentacji konieczne jest wskazanie obszaru, w którym znajduje się segmentowany obiekt, lub kilku punktów leżących w obrębie obiektu (monit użytkownika). Model wykorzystuje architekturę kodera-dekodera, gdzie bazujący na wizualnym transformerze (Vision Transformer [57]), główny koder (image encoder):

- dzieli obraz wejściowy na małe fragmenty (patches), które są przetwarzane równocześnie,
- transformer przetwarza te małe fragmenty, ucząc się wzorców i relacji między różnymi częściami obrazu,
- jako wynik zwraca zwartą reprezentację obrazu (image embedding), która zawiera istotne cechy obrazu.

Drugim koderem jest koder monitów, który przyjmuje monity użytkownika jako dane wejściowe i przekształca je na ujednoliconą reprezentację wektorową, którą model potrafi interpretować.

Dekoder maski natomiast przekształca te wskazówki w dokładne maski segmentacyjne. Schemat działania przedstawiono na [Rysunek 18]



Rysunek 18 Architektura Segment Anything Model[11]

Zalety:

- Wysoka precyzja i elastyczność.
- Skuteczność w segmentacji różnorodnych obrazów.
- Interaktywność – dekoder szybko reaguje na wskazówki użytkownika.

Wady:

- Wymaga dużych zbiorów danych do treningu.
- Wymaga wskazówek użytkownika.
- Może mieć problemy z segmentacją dla obrazów, gdzie granice pomiędzy obiektami są niejednoznaczne lub rozmazane, lub gdy obiekt jest częścią większej struktury.
- Może wystąpić problem nadmiernej segmentacji (oversegmentation) lub niewystarczającej segmentacji (undersegmentation).

[11] Źródło https://ai.meta.com/research/publications/segment-anything/

- Może mieć problemy z segmentacją obiektów (szczególnie o nietypowych kształtach), które nie pojawiły się w zbiorze treningowym.
- Znaczne zapotrzebowanie na zasoby obliczeniowe (GPU).

2.3. Wiodące komercyjne rozwiązania wspomagające decyzje sędziowskie w sporcie

Na rynku istnieją firmy, które oferują gotowe rozwiązania z wizji komputerowej dla sportu, jednakże są one bardzo drogie, także ze względu na monopolizację rynku. Najbardziej znanymi na rynku firmami oferującymi komercyjne rozwiązania do detekcji i śledzenia obiektów w sporcie są: Hawk Eye Innovations (Sony) [17], Stats Perform [95], TRACAB [96]. Niestety żadna z tych firm nie przedstawia szczegółów swoich badań, w związku z czym nie można tych badań, ani powtórzyć, ani potwierdzić wyników, którymi chwalą się wspomniane firmy. Z biznesowego punktu widzenia jest to zrozumiałe, z naukowego kłopotliwe. Światowy lider - firma Hawk Eye Innovations wskazuje średni błąd dokładności lokalizacji miejsca odbicia piłki tenisowej na poziomie 3.6 mm [97] nie podając jednakże żadnych szczegółów w jaki sposób dokonano tych szacunków.

Z informacji udostępnianych przez wspomniane firmy można uzyskać tylko bardzo ogólną wiedzę co do użytych technologii. Dla przykładu rozwiązanie firmy TRACAB, które jest głównie używane w piłce nożnej do śledzenia zawodników, wykorzystuje algorytmy uczenia maszynowego i konwolucyjne sieci neuronowe [98]. Warto tutaj nadmienić, że wizyjne metody śledzenia obiektów są narażone na błędy spowodowane okluzjami. Dlatego też często wymagana jest korekcja tych błędów przez ludzkiego operatora [99].

Ciągle jeszcze oferowane rozwiązanie nie są w 100% skuteczne. Podczas turnieju tenisowego na kortach Wimbledonu w 2022r. podczas ćwierćfinałowego meczu zawodnicy Joe Salisbury i Rajeev Ram odmówili dalszej gry po kontrowersyjnej decyzji systemu Hawk Eye [100]. Po błędnej decyzji (doskonale widocznej na filmie [101]) podczas badmintonowego turnieju DAIHATSU Indonesia Masters 2021 światowa organizacja badmintona i firma Hawk Eye wydali oświadczenie, w którym przepraszają za ten błąd [102]. Z inny przykładem błędnej decyzji systemu Hawk-Eye w rugby można zapoznać się oglądając [103]. Pomimo wspomnianych błędnych decyzji, systemy komputerowe w ogólności są bardziej dokładne niż ludzie [104]. Dlatego też od 2001 roku aplikacje wizji komputerowej są używane w wielu dyscyplinach sportowych [8, 9], a w badmintonie system Hawk Eye pojawił się w 2014r.

3. Charakterystyka zasadniczego osiągnięcia i opis opracowanej metody

3.1. Gromadzenie danych do eksperymentów

Istotną wartością pracy badawczej autora jest to, że oprócz danych zarejestrowanych podczas eksperymentów w klubie Kahuna w Warszawie, dane był też zabierane podczas realnych zawodów - w różnych obiektach sportowych posiadających też różne oświetlenie. Były to między innymi hale w Katowicach (Mistrzostwa świata seniorów), Wesołej (mistrzostwa juniorów Mazowsza), Głubczycach (mistrzostwa Polski U17), Częstochowie (Yonex Polish Open), Warszawie (mistrzostwa seniorów Mazowsza). Co więcej, zostały zarejestrowane tylko te sytuacje, gdy zawodnik nie zgadzał się z decyzją sędziego liniowego. Zebrane materiały wideo zostały wykorzystane od opracowania i testów algorytmów. Finalna ocena skuteczności opracowanego systemu odbyła się zgodnie z poniższą procedurą.

- Zdecydowano, że dane do oceny skuteczności algorytmów (zbiór testowy) zostaną zarejestrowane w hali, w której nigdy wcześniej nie dokonywano pomiarów. Dane zostały zebrane podczas największego turnieju badmintonowego jaki odbył się w Polsce - mistrzostw świata seniorów Katowice 2019 oraz mistrzostw Polski U17 w Głubczycach.
- Podczas turniejów zbierano dane w następujący sposób:
    - Za każdym razem, gdy zawodnik nie zgodził się z decyzją sędziego podnosił rękę, prosząc o wideo weryfikację. Po tym sygnale operator systemu uruchamiał procedurę zapisu ostatnich 24 sekund danych z kamery, w polu widzenia której była lotka.
- Zapisane pliki zostały wczytane na wejściu do stworzonego na potrzeby projektu oprogramowania do anotacji danych.
- Ponieważ w zapisanym klipie przez większość czasu nic się nie dzieje – są widoczne linie kortu, osoba anotująca dane z 24 sekundowego klipu starała się wybierać te fragmenty, na których widać było poruszające się obiekty (zawodników, lotkę, publiczność itp.) oraz moment odbicia lotki o podłoże. Nie oznacza to, że wśród danych do testów nie ma takich obrazów, na których widać tylko linie kortu i nic więcej. Jednakże jest ich zdecydowanie mniej, niż gdyby zachowano do anotacji całe 24 sekundowe sekwencje. Dla każdej ramki w

wybranych sekwencjach oznaczono pozycję lotki oraz moment odbicia korka lotki o podłoże. Dla ramki, w której nastąpiło odbicie o ziemie operator zaznaczał obszar, w którym lotka dotykała podłoża i decydował czy lotka jest w polu gry, czy poza nim. Drugi operator dokonywał weryfikacji anotacji i korygował błędy. Ostateczna weryfikacja czy lotka upadła w polu gry, czy poza nim była dokonana przez licencjonowanego sędziego badmintona panią Joannę Mądry[12]

Tak utworzono zbiór danych liczący 13442 obrazy. Na 1793 obrazach operator zaznaczył pozycję lotki (ground truth), na 11649 ramkach nie było lotki.

- Zbiór danych został podzielony na zbiór treningowy, walidacyjny oraz testowy. Zbiór testowy zawierał nagrania z Katowic i Głubczyc i został wykorzystany do ostatecznej weryfikacji całości rozwiązania. Dane z pozostałych hal zostały podzielone na zbiór treningowy i walidacyjny. 75% sekwencji przydzielono losowo do zbioru treningowego a 25% do zbioru walidacyjnego.

- Zebrane dane zostały wykorzystane do opracowania odpowiednich modeli i algorytmów oraz do oceny ich skuteczności. Zostały wyznaczone odpowiednie miary, a te modele, które były najlepsze zostały ostatecznie wybrane.

3.2. Ogólny opis zaproponowanej metody

Jak już wspomniano na początku, ze względu na wymagania implementacyjne (ograniczone gabaryty, mobilność, łatwość instalacji i re-konfiguracji, koszt etc.) autor poświęcił dużą uwagę na stworzeniu takich rozwiązań algorytmicznych, które nie będą wymagały dużej mocy obliczeniowej i nie będą miały wysokich wymagań pamięciowych, równocześnie zapewniając dostarczenie wyników w czasie rzeczywistym. Z tego powodu też w pierwszej kolejności autor tworzył modele o ograniczonej złożoności, które następnie były udoskonalane, w celu zapewnienia wystarczająco wysokiej skuteczności rozpoznania miejsca upadku lotki. Autor wykorzystując realia gry w badmintona i wiedzę domenową wspartą własnym doświadczeniem, zdobytym przez kilkanaście lat grania w turniejach badmintona, rozszerzył znane z literatury koncepcje o własne rozwiązania ekstrakcji i selekcji kluczowych cech, wyznaczanych na podstawie zarejestrowanych sekwencji wideo zawierających lecącą lotkę i moment jej upadku na kort. Opracowane rozwiązania pozwoliły efektywnie rozwiązać przedstawione problemy badawcze.

[12] Nr licencji 383 https://pzbad.pl/wp-content/uploads/2024/07/sedziowie_z_kwalifikacjami_01072024.pdf

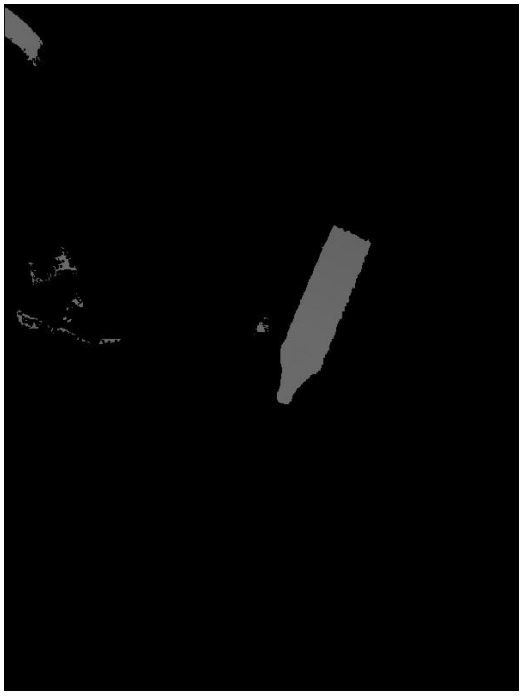
Podstawowym problemem jest ustalenie położenia upadającej lotki względem linii kortu na podstawie jednej kluczowej ramki (najbliższej momentowi upadku), którą udało się zarejestrować z zadaną rozdzielczością czasową (kompromis względem ograniczeń sprzętowych i czasowych oraz uwarunkowań zewnętrznych - głównie oświetleniowych, panujących w hali sportowej).

Aby rozwiązać podstawowy problem badawczy należy też rozwiązać też szereg innych problemów o które wymieniono w rozdziale 1. Poniżej, w sposób ogólny, przedstawiono metodę rozwiązania trzech kluczowych problemów badanych przez autora, które zostały szczegółowo opisane w dalszej części rozprawy.

Detekcja i śledzenie lotki

Aktualnie większość badaczy problem detekcji obiektów na obrazie rozwiązuje z użyciem złożonych modeli sieci neuronowych. Autor również korzysta z sieci neuronowej (YOLOv3), ale tylko wtedy gdy podstawowy moduł śledzący zgubił lotkę. To wyjątkowe, hybrydowe podejście pozwala na działanie opracowanego rozwiązania w czasie rzeczywistym.

W podstawowym module śledzącym, do detekcji poruszającej się lotki (taka nas tylko interesuje), autor wykorzystał odpowiednio przetworzone dane ze strumienia wideo, tworząc ramki różnicowe oraz akumulacyjne ramki różnicowe. Dzięki odpowiedniemu doborowi parametrów (kamery jak i parametrów modułu tworzącego akumulacyjne ramki różnicowe) poruszająca się lotka na akumulacyjnej ramce różnicowej ma kształt podobny do kredki [Rysunek 19]. Ten charakterystyczny kształt jest znajdowany i śledzony w każdej klatce strumienia wideo. Warto zauważyć, że kształt jaki ma lotka na akumulacyjnej ramce różnicowej wyznacza równocześnie kierunek, w którym się porusza. W opracowanej metodzie detekcji i śledzenia lotki, ze względu na możliwe okluzje, autor wykorzystuje też szaroodcieniowe obrazy zarejestrowane przez kamerę, które są wejściem do sieci neuronowej. Schemat algorytmu został szczegółowo opisany w dalszej części rozprawy.

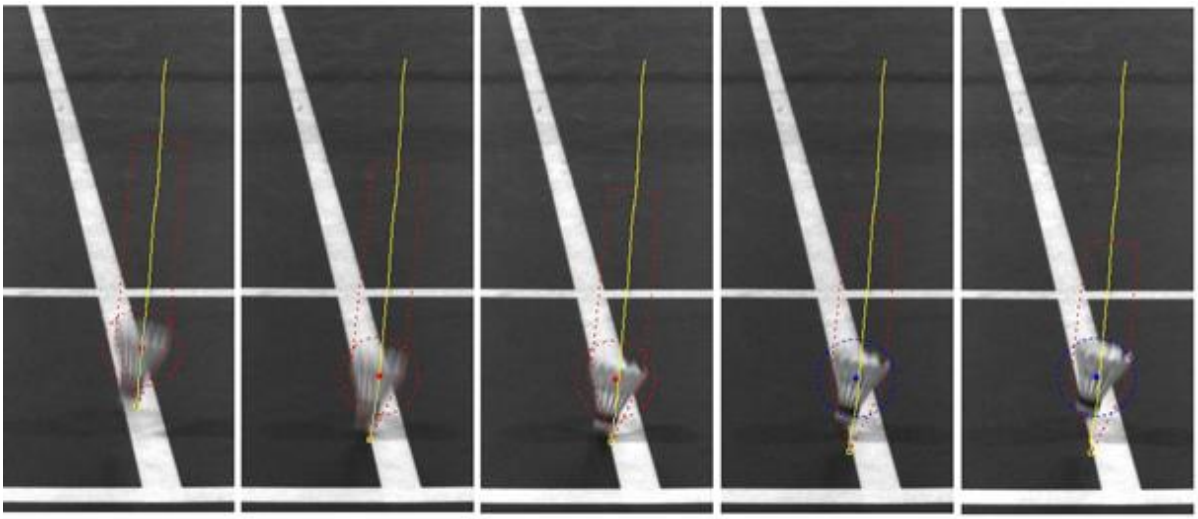


Rysunek 19 Akumulacyjna ramka różnicowa, na której lotka ma charakterystyczny kształt

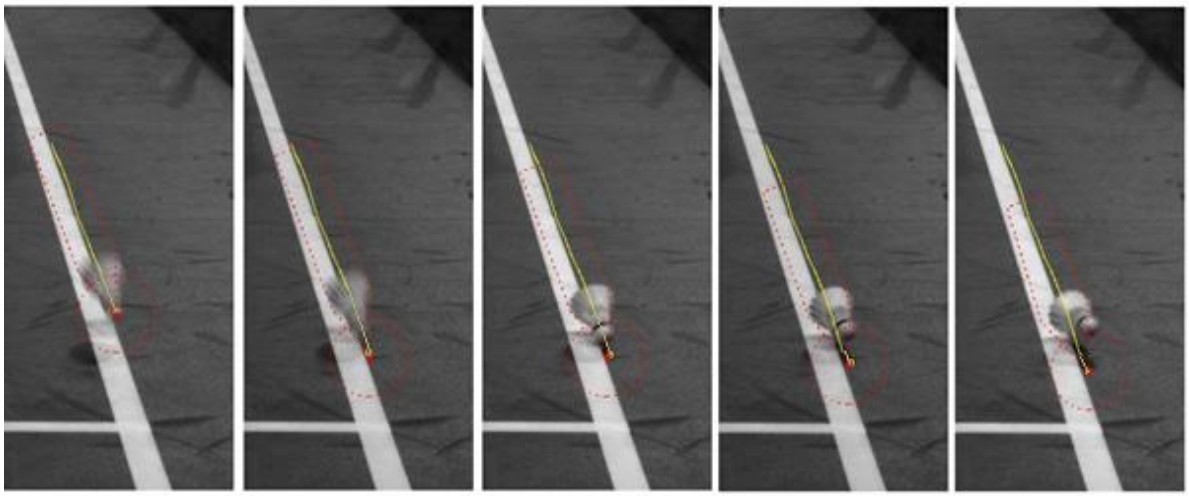
Wyznaczenie kluczowej ramki, w której nastąpiło odbicie o podłoże

W związku z tym zastosowanym układem kamer [Rysunek 3] nie jest możliwe wyznaczenie pozycji lotki w przestrzeni trójwymiarowej. W układzie kamer nad kortem, gdy mamy trajektorię lotki w 3D można wyznaczyć moment odbicia analizując zmienność trajektorii lotki. W przypadku, gdy mamy tylko dane w 2D wyznaczenie momentu odbicia na podstawie zmienności trajektorii jest trudniejsze - szczególnie, gdy lotka leci płasko w stronę kamery, a po odbiciu prawie nie zmienia swojej trajektorii w stosunku do kamery. Takie sytuacje zdarzają się najczęściej po uderzeniu smash, które zawodnicy głównie kierują w kierunku linii bocznej. Gdy lotka upada w okolicy linii końcowej (dzieje się tak po uderzeniu clear), to można wyznaczyć moment odbicia na podstawie śledzenia zmienności trajektorii. Jednakże ze statystyk zebranych przez autora [Tabela 21, strona 116] wynika, że gdy lotka upada w okolicy linii końcowej, zawodnicy proszą o weryfikację z systemu challenge tylko w 1/3 przypadków, więc metoda analizy zmienności trajektorii nie zawsze będzie skuteczna. Biorąc pod uwagę poczynione obserwacje, autor opracował metodę, która w celu wyznaczenia kluczowej ramki, w której nastąpiło odbicie o podłoże, analizuje również inne dane.

Na rysunku [Rysunek 20] przedstawiono przykładowe zachowanie lotki upadającej na linie końcową po uderzeniu clear, a na rysunku [Rysunek 21] na linie boczną po uderzeniu smash. Żółtym kolorem zaznaczono trajektorię lotki wygenerowaną przez traker, a żółta kropka oznacza pozycję korka zwróconą przez traker.



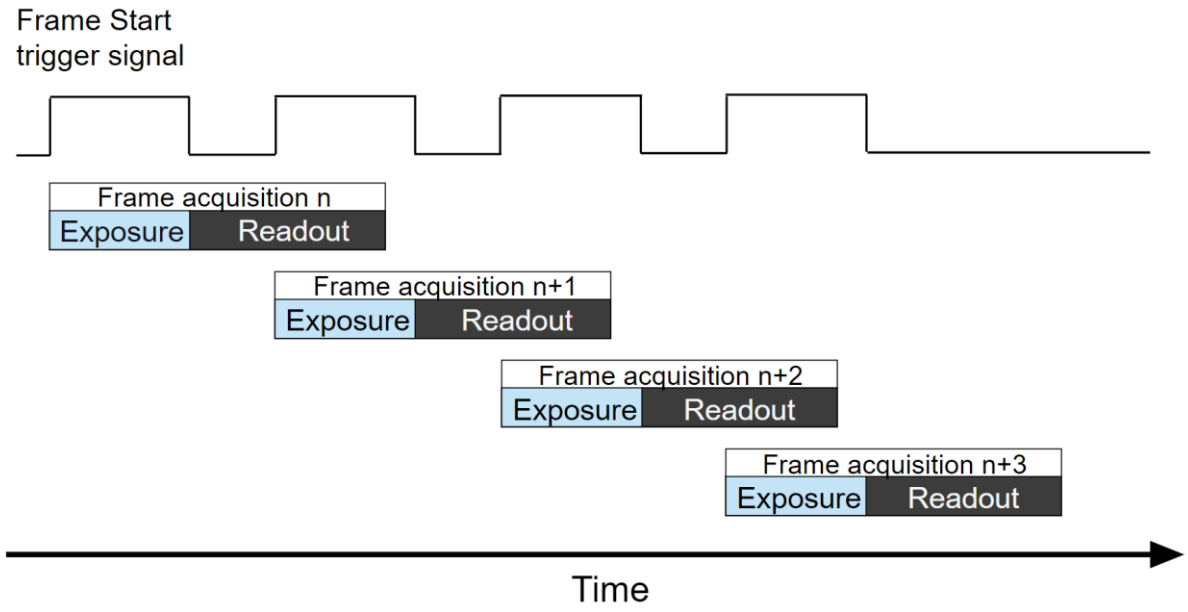
Rysunek 20 Kolejne klatki z lotką odbijająca się od podłoża w okolicy linii końcowej po uderzeniu clear



Rysunek 21 Lotka odbijająca się od podłoża w okolicy linii bocznej po uderzeniu smash

U podstaw algorytmu wyznaczającego kluczową ramkę jest fakt, że w momencie odbicia lotka istotnie zmniejsza swoją prędkość (korek przestaje się poruszać). Autor wyznacza szereg cech uzyskanych na etapie przetwarzania danych, które następnie są wejściem do modelu uczenia maszynowego. Wytrenowany model odpowiada na pytanie czy w danej ramce było odbicie o podłoże czy też nie.

Powstaje pytanie czy kamera zawsze zarejestruje moment odbicia o podłoże, czy też chwilę przed lub po. Aby odpowiedzieć na to pytanie, trzeba wiedzieć jaka jest przerwa czasowa pomiędzy rejestracją kolejnych klatek przez kamerę. Ta przerwa czasowa zależy od modelu kamery i ustawień jej parametrów. Do rejestracji wideo użyto kamer Basler ace UacA800-200gm [105] działających w trybie free run [106] (z maksymalną możliwą szybkością). Dla tego trybu sposób akwizycji danych przez kamerę przedstawiono na rysunku [Rysunek 22].



Rysunek 22 Sposób w jaki kamera naświetla sensor i odczytuje z niego dane dla kolejnych ramek[13]

Przerwa czasowa pomiędzy naświetlaniem sensora kolejnych klatek jest sumą czasu potrzebnego na odczyt danych z sensora kamery (Readout) i opóźnieniem rozpoczęcia ekspozycji [107]. Użyte przez autora kamery pozwalają na rozpoczęcie naświetlanie nowej ramki, w czasie gdy dane z sensora kamery dla poprzedniej ramki są jeszcze odczytywane. Dla tej konkretnej kamery są to (podane przez producenta) wartości: Sensor Readout Time –10 µs oraz Exposure Start Delay - 3µs. Czyli maksymalna przerwa pomiędzy naświetlaniem kolejnych ramek może wynieść 13 mikrosekund. Jak już wspomniano wcześniej, mimo że lotka może mieć prędkość początkową powyżej 400 km/h, to ze względu na swój kształt, po uderzeniu gwałtownie zmniejsza swoją prędkość. Nie zarejestrowano przy podłożu prędkości wyższych niż 9.72 m/s. Oznacza to, że w czasie 13 mikrosekund lotka może pokonać dystans 0,126 mm. Jest to maksymalny błąd wynikający z tego, że na kluczowej ramce możemy nie mieć dokładnego momentu odbicia. Ta wartość jest tak niska, że z punktu widzenia badanego problemu pomijalna i autor przyjął, iż zawsze w strumieniu wideo jest ramka z zarejestrowanym momentem odbicia lotki o podłoże, a nie chwilę przed lub po.

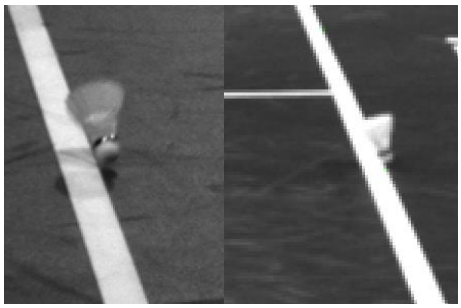
Wyznaczenie miejsca odbicia lotki o podłoże

W celu wyznaczenia miejsca odbicia lotki o podłoże konieczna jest jej precyzyjna segmentacja na kluczowej ramce (tej która została wybrana jako ta, w której było odbicie o podłoże).

Układ nakładających się na siebie piór lotki powoduje, że lotka oprócz tego, iż porusza się ruchem postępowym, to jeszcze wiruje, w związku z czym nawet w momencie odbicia się lotki

[13] Źródło: https://docs.baslerweb.com/overlapping-image-acquisition#non-overlapping-image-acquisition

o podłoże pióra lotki są mocno rozmyte [Rysunek 23]. To rozmycie lotki powoduje, że krawędzie lotki nie odcinają się ostro od tła, co jest szczególnie problematyczne, gdy w tle białej lotki znajduje się biała linia. Co gorsze, najczęściej zawodnik prosi o weryfikację decyzji sędziego, właśnie wtedy, gdy lotka w momencie odbicia znajduje się w pobliżu linii. Z tego powodu znane metody segmentacji, których większość bazuje na gradientach, kiepsko radzą sobie z precyzyjną segmentacją lotki. Autor pragnie zauważyć, że w celu określenia miejsca odbicia lotki o podłoże w stosunku do referencyjnych linii kortu, kluczowa jest segmentacja korka lotki (ze względu na swój kształt, lotka zawsze odbija się o podłoże korkiem). Aby wyznaczyć na obrazie korek lotki, autor opracował nowatorską metodę polegającą na obliczeniu okręgu wyznaczającego korek lotki na podstawie wyznaczonych uprzednio cech. W szczególności autor nie szuka na obrazie konturu lotki, tylko szuka ciemnego paska, który oddziela korek lotki od piór. Ciemny pasek zawsze okala korek i mocno odcina się od białego korka i piór – co ułatwia jego segmentację [Rysunek 20, Rysunek 21, Rysunek 23]. Ponieważ wszystkie lotki muszą spełniać normy dotyczące rozmiaru, to po wyznaczeniu tego paska możliwe jest, dokonując obliczeń matematycznych, wyznaczenie okręgu odpowiadającego korkowi lotki. Samo wyznaczenie ciemnego paska, a precyzyjniej jednego charakterystycznego punktu znajdującego się na prostej przechodzącej przez środek lotki, to zadanie wytrenowanego modelu uczenia maszynowego szczegółowo opisanego w dalszej części.



Rysunek 23 Lotka odbijająca się po podłoże. Widoczne rozmycie piór oraz czarny pasek oddzielający korek od piór.

3.3. Szczegółowy opis algorytmów, eksperymenty i wyniki badań eksperymentalnych

Detekcja i śledzenie lotki

Aby móc śledzić dowolny obiekt, najpierw należy go znaleźć na obrazie. Do znalezienia konkretnego obiektu można zastosować wiele metod, od najprostszych, takich detekcja na podstawie cech obiektu takich jak kolor i kształt, po metody wykorzystujące sieci neuronowe.

W rozdziale 2 przedstawiono wybrane metody śledzenia obiektów, które z są stosowane przy detekcji różnorodnych obiektów. Stosowanie ogólnego, uniwersalnego podejścia do wykrywania lotki jest problematyczne z kilku powodów.

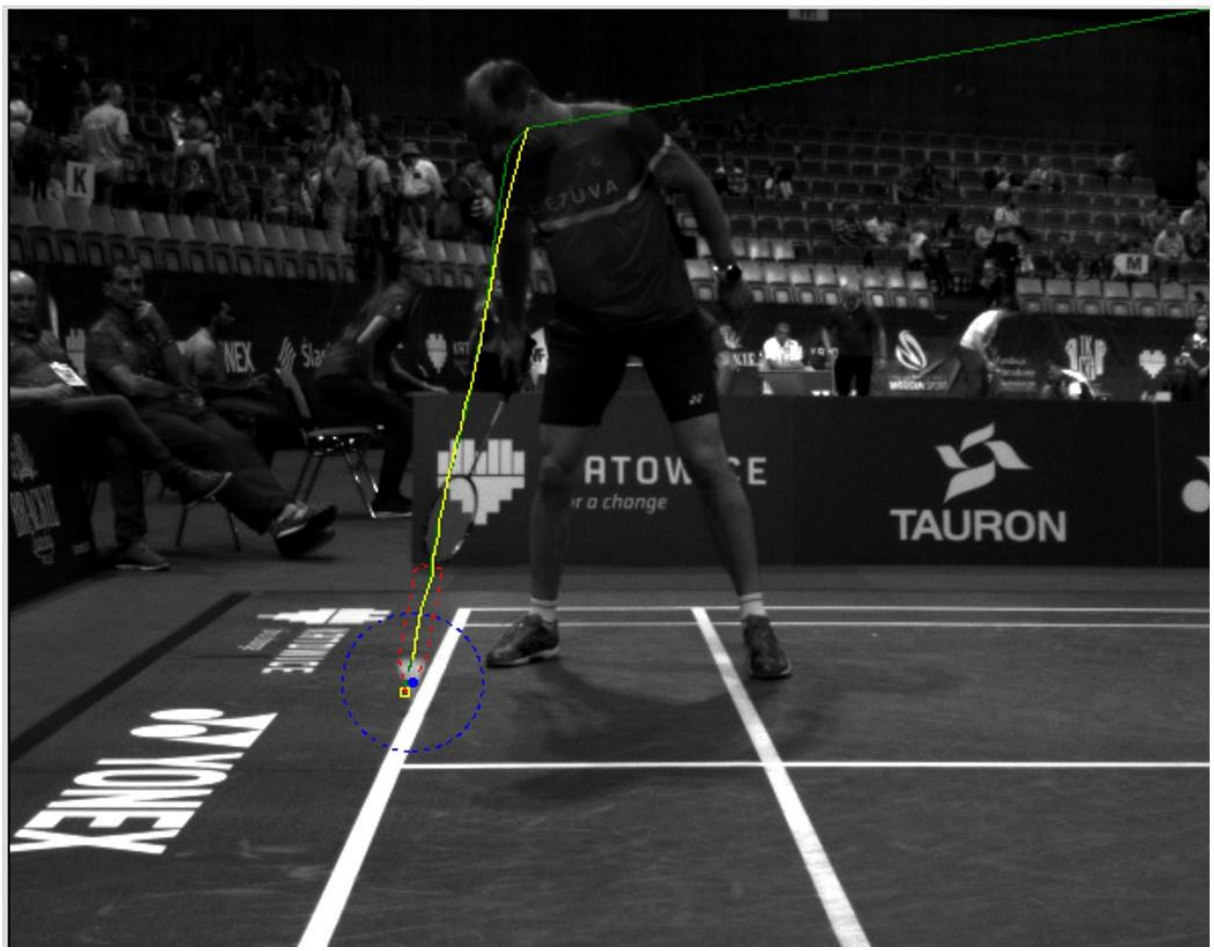
1. W zależności od kąta pod jakim kamera obserwuje lotkę, kształt lotki do badmintona może być postrzegany na obrazie jako koło lub trójkąt [Rysunek 13], dlatego odnalezienie jej na obrazie jest trudniejsze niż znalezienie piłki, która jest zawsze widziana jako kształt kolisty.
2. W przypadku ustawienia kamer przy liniach kortu, rozmiar lotki może znacząco zmienić się w miarę zbliżania się do kamery. Lotka może być zaobserwowana w odległości od 1,5m do 8,2m. Daje to przeszło pięciokrotną różnicę w wielkości lotki.
3. Prędkości z jakimi może poruszać się lotka mają dużą rozpiętość. W momencie odbicia lotki przez zawodnika może to być nawet powyżej 400 km/h (uderzenie smecz) ale już w momencie odbicia o ziemię może to być prędkość terminalna – 6.7 m/s [108]. Dodatkowo, prędkość lotki zaraz po uderzeniu jej przez zawodnika gwałtownie się zmniejsza. Lotka, w przeciwieństwie do piłki tenisowej, nie odbija się tak sprężyście od podłoża. W związku z czym po odbiciu porusza się dużo wolniej niż przed odbiciem.
4. Biały kolor lotki może być bardzo podobny do sceny (skarpetki zawodnika, białe linie kortu, litery na tablicach ogłoszeniowych).
5. Lotka często porusza się w złożonych sytuacjach: poruszający się gracze, szybko poruszające się rakiety zawodników i zmieniające się tło.
6. Ze względu na stożkowy kształt i środek ciężkości umieszczony w okolicy korka (a nie w środku lotki), trudno jest przewidzieć trajektorię lotu zgodnie z prawami fizyki opisanymi prostymi wzorami, zwłaszcza po uderzeniu, lub odbiciu o podłoże, gdy lotka obraca się i ma niestabilną trajektorię [109].

Dzięki ustawieniu kamer blisko kortu [Rysunek 3], zawęża się pole widzenia każdej z kamer do najważniejszego z punktu widzenia badanego problem obszaru, czyli okolic linii pola gry. W ten sposób też ogranicza się widoczność innych poruszających się obiektów (np. graczy) - lotka jest często jedynym poruszającym się obiektem.

Upraszcza to również problem niestabilnej trajektorii w momencie odbicia lotki przez zawodnika. Trajektoria lotki się stabilizuje dopiero po pewnym czasie od odbicia, a przy ziemi jest już stabilna i łatwa do wyznaczenia na podstawie kilku kolejnych ramek. Ponadto prędkość

lotki na ostatnim metrze przed uderzeniem o podłoże jest prawie stała. Trajektoria lotki jest wyznaczana na podstawie kolejnych pozycji korka, wyznaczonego z akumulowanych ramek różnicowych. Jest ona dopasowywana do linii prostej - [Rysunek 24].

Kolejną zaletą ustawienia kamery w pobliżu kortu jest to, że rozmiar lotki w pikselach jest większy, niż w przypadku, gdy kamera obejmuje swoim polem widzenia cały kort i jest umieszczona w dużej odległości od pola gry. Ustawienie kamery w niewielkiej odległości od kortu pozwala na osiągnięcie większej dokładności lokalizacji miejsca upadku lotki.



Rysunek 24 Trajektoria lotki przy linii oraz przewidywana pozycja lotki wyznaczona za pomocą filtru Kalmana (kolor zielony), Prawdziwa pozycja loki w kolejnych ramkach wyznaczona przez operatora (kolor żółty)

Opis metody zastosowanej do detekcji lotki

Kolejne kroki jakie wykonywane są przez algorytm detekcji lotki w celu znalezienia lotki w strumieniu wideo są następujące:

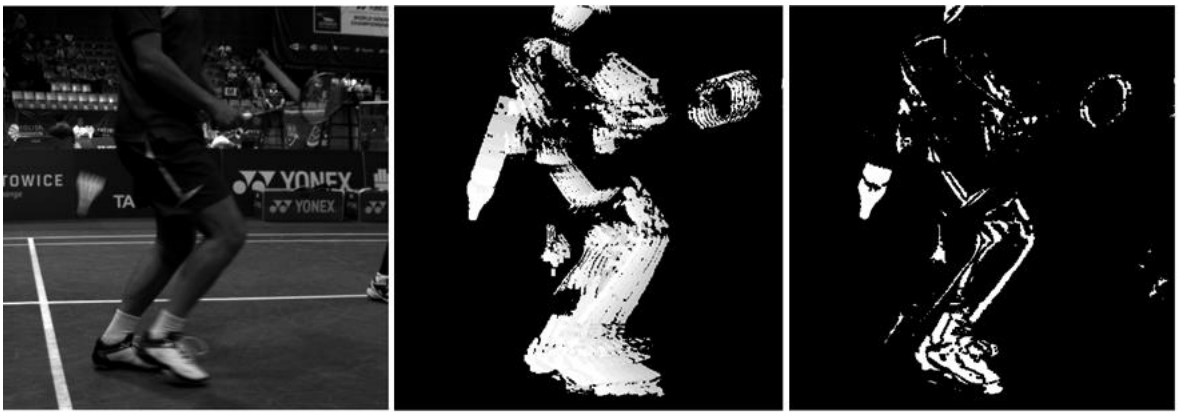
1. wygenerowanie ramki różnicowej z dwóch kolejnych ramek,
2. usunięcie dużych obiektów z ramki różnicowej,

3. wygenerowanie akumulacyjnej ramki z 7 kolejnych ramek różnicowych z pkt 2,
4. wygenerowanie zestawu kandydatów na lotkę z akumulacyjnej ramki różnicowej z pkt 3,
5. zastosowanie zestawu filtrów odrzucających ze zbioru kandydatów tych, którzy nie pasują do modelu lotki,
6. wyznaczenie trajektorii dla każdego z kandydatów na lotkę pozostałych w zbiorze kandydatów,
7. usunięcie ze zbioru tych kandydatów, dla których trajektoria nie pasuje do modelu trajektorii poruszającej się przy linii lotki.

Powyższe kroki algorytmu są opisane bardziej szczegółowo w dalszej części rozprawy.

Lotka jest szybko poruszającym się obiektem na zwykle nieruchomym tle. Dlatego też zastosowanie metody polegającej na generacji obrazów różnicowych z kolejnych klatek, pozwala na szybkie odróżnienie poruszających się obiektów od statycznego tła.

Z drugiej strony, gdy lotka uderza w podłoże to może na chwilę przestać się poruszać i „zniknąć" z obrazu różnicowego. Dlatego w przedstawionym rozwiązaniu oprócz ramki różnicowej z dwóch kolejnych zaproponowana metoda generuje akumulacyjną ramkę różnicową z 7 kolejnych obrazów różnicowych. Dzięki temu nie zgubimy lotki w momencie uderzenia o ziemię. Dla zadanej prędkości rejestracji wideo (150-200FPS) eksperymentowano z różną liczbą ramek, z której generowana jest akumulacyjna ramka różnicowa i dla wartości 7 osiągnięto najlepsze wyniki. Zbyt mała liczba ramek różnicowych do akumulacji powoduje, że trudno odróżnić lotkę od innych poruszających się obiektów. Zbyt duża, natomiast powoduje, że w momencie, gdy lotka porusza się w pobliżu innego poruszającego się obiektu (np. zawodnika), to te dwa obiekty mogą się połączyć i lotka przestanie mieć charakterystyczny kształt i nie zostanie znaleziona. Przykład takiej sytuacji znajduje się na rysunku [Rysunek 25]



Rysunek 25 Lotka, która poruszała się na tle przesuwającego się zawodnika. Z lewej - oryginalny obraz zarejestrowany przez kamerę, środkowy - akumulacyjna ramka różnicowa, prawy - ramka różnicowa.

To nowatorskie podejście wykorzystuje, zazwyczaj niepożądany, efekt rozmycia znajdowanego obiektu. Parametry progowania podczas tworzenia ramki różnicowej zostały tak dobrane, aby po utworzeniu ramek różnicowych ich połączenie w ramkę akumulacyjną utworzyło dla lotki obiekt o kształcie zbliżonym do kredki [Rysunek 19, Rysunek 25 (środkowy), Rysunek 26, Rysunek 29a].

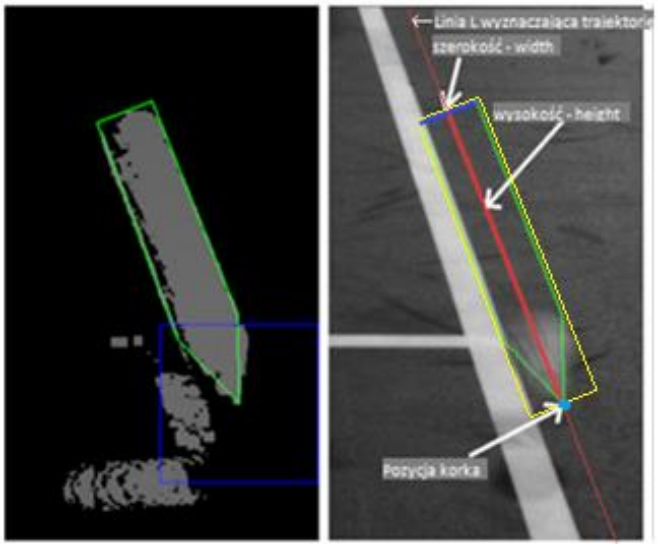
W zaproponowanej metodzie intensywność pikseli w ramce akumulacyjnej odpowiada kolejności obrazów użytych do jej wygenerowania. Piksele z późniejszych obrazów różnicowych uzyskują niższe wartości niż piksele z wcześniejszych obrazów (od jasnego do ciemnego odcienia szarości). W ten sposób uzyskuje się dodatkową informację o kierunku ruchu obiektu. Na takim obrazie znajdowani są kandydaci na lotkę. Każdy wydzielony obszar (blob) trafia do zbioru kandydatów i dla każdego kandydata obliczany jest zestaw cech. Cechy są obliczane dla każdego bloba wyznaczonego na akumulacyjnej ramce różnicowej. Lista wyznaczonych cech jest przedstawiona w tabeli [Tabela 3]. Na podstawie tych cech i zdefiniowanych uprzednio filtrów ze zbioru odrzucani są ci kandydaci, którzy nie pasują do modelu poruszającej się lotki. Jeśli w zbiorze pozostanie więcej niż jeden kandydat, to dla każdego kandydata z obrazu zarejestrowanego przez kamerę wycinamy część obrazka o rozmiarze 104x104 px, którego środek jest wyznaczony przez położenie kandydata i przekazujemy go do sieci neuronowej wytrenowanej do rozpoznawania lotki do badmintona. Kandydat, dla którego prawdopodobieństwa bycia lotką jest najwyższe pozostaje w zbiorze. Jeśli po zastosowaniu filtrów w zbiorze kandydatów nie pozostanie żaden kandydat, a w poprzedniej chwili czasowej (*t-1*) algorytm znalazł lotkę, to z wykorzystaniem filtru Kalmana przewidywana jest, w chwili *t*, pozycja lotki: punkt - *P(x,y)*, a wycinek obrazu o środku w *P* przekazywany jest do sieci neuronowej. Jeśli w przekazanym wycinku obrazu sieć znalazła lotkę, to pozycja *P(x,y)* jest zapamiętywana jako pozycja lotki w chwili *t* oraz przepisywane są

pozostałe cechy lotki z chwili *t-1*. Jeśli sieć nie znalazła lotki, to najprawdopodobniej oznacza to, że lotka została całkowicie zasłonięta przez zawodnika. Z punktu widzenia skuteczności systemu taka sytuacja jest niepożądana, szczególnie gdy zawodnik przesłonił lotkę jak uderzała o podłoże. W praktyce sytuacje, gdy zawodnik całkowicie zasłania lotkę zdarzają się, tylko gdy sporna sytuacja dotyczy linii końcowej. Na szczęście linie końcową obserwują dwie kamery, więc lotka zawsze będzie widoczna na obrazie z przynajmniej jednej kamery.

Cecha	**Opis**
kontur (*C*)	Lista punktów, które definiują wydzielony na obrazie obszar (blob) będący kandydatem na lotkę.
prostokąt (*RECT*)	Lista 4 punktów definiujących rotowany prostokąt, w który wpisany jest kontur C. Wizualnie cechę tę zaprezentowano kolorem żółtym na rysunku [Rysunek 26 (prawy)]
szerokość (width) (*W*)	Krótszy z boków prostokąta, w który wpisany jest kontur (szerokość lotki wyznaczona przez końcówki piór). Wizualnie cechę tę zaprezentowano na rysunku [Rysunek 26] – kolor niebieski.
długość (height) (*H*)	Dłuższy z boków prostokąta, w który wpisany jest kontur. Wizualnie cechę tę zaprezentowano na rysunku [Rysunek 26] – kolor czerwony.
stosunek szerokości do długości (*AR*)	Wartość szerokość/długość $$AR = \frac{W}{H}$$
linia dopasowania (*FL*)	Linia, która jest wyznaczona w taki sposób, że suma odległości punktów konturu od tej linii jest minimalna.
kąt (*A*)	Kąt jaki tworzy linia dopasowania z osią oX obrazu w zakresie od 0 do 180 stopni.
kontur lotki (*SC*)	Lista 5 punktów wyznaczonych na podstawie cechy RECT, które określają dla danego prostokąta kontur lotki. Wizualnie cechę tę zaprezentowano kolorem zielonym na rysunku [Rysunek 26] Z modelu lotki, wiemy, że pióra są wbite w korek pod kątem 21.28 stopnia w stosunku do osi lotki. Zatem wyznaczenie konturu lotki polega na „ścięciu" dwóch boków prostokąta RECT pod tym właśnie kątem.

odległość konturu lotki (*SC*) od konturu (C) -(*DIST2SC*)	Miara ta jest wyznaczana w następujący sposób: 1. Wyznaczany jest nowy kontur (lista punktów) $CHC = convexHull(C)$ zgodnie z algorytmem [110] 2. Dla *SC* oraz *CHC* obliczany jest obszar jaki zajmują te kontury zgodnie z formułą Green'a $A_{SC} = contourArea(SC)$ $A_{CHC} = contourArea(CHC)$ 3. Wyznaczana jest wartość *DIST2SC* jako: $DIST2SC = \left\lvert 1 - \frac{A_{CHC}}{A_{SC}} \right\rvert$

Tabela 3 Lista cech obliczanych dla każdego kandydata na lotkę



Rysunek 26 Wizualizacja cech lotki (po lewej akumulacyjna ramka różnicowa)

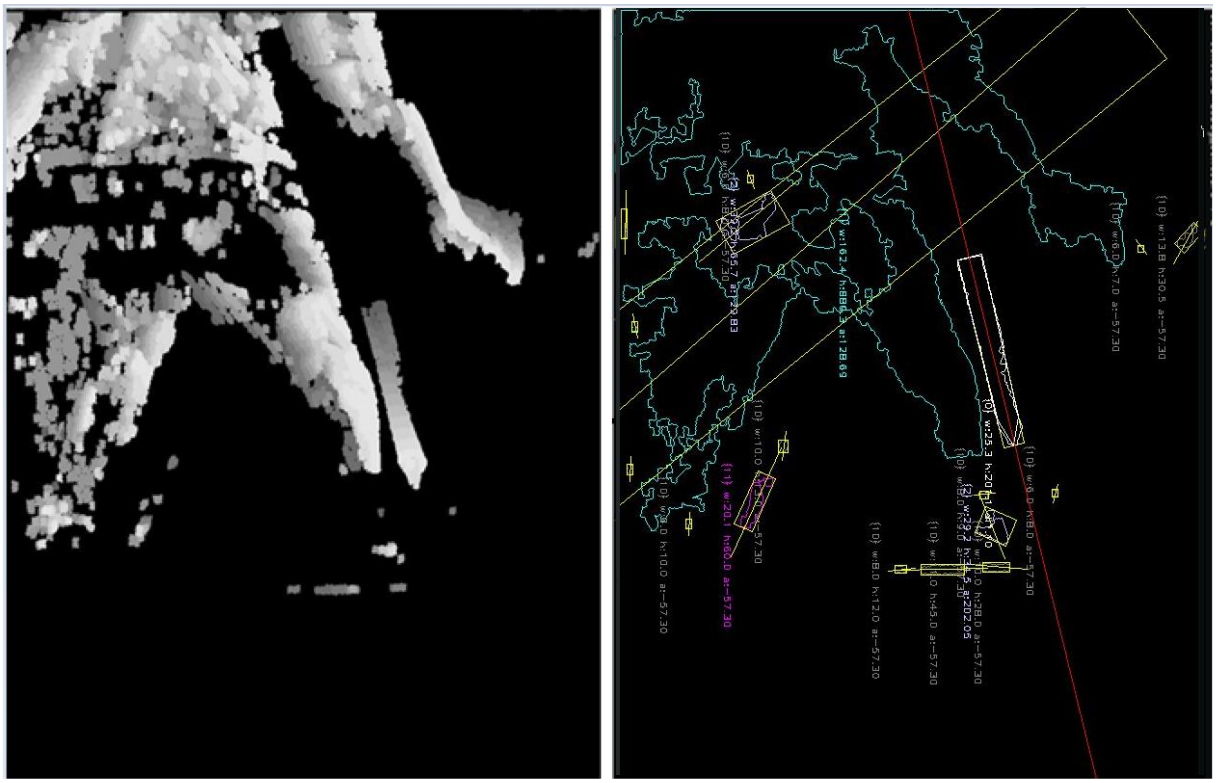
Na rysunku [Rysunek 27] przedstawiono akumulowaną ramkę różnicową oraz kandydatów na lotkę. Kolor oznacza rodzaj filtra, który spowodował odrzucenie kandydata. Każdy z kandydatów jest obwiedziony prostokątem, który wyznacza wysokość i szerokość kandydata. Poprowadzono też linie dopasowania (*FL*), które wyznaczają kierunek poruszania się kandydata na lotkę. Białym kolorem zaznaczono kandydata, który został ostatecznie wybrany jako lotka. Dla tego kandydata wyznaczono też pozycję korka.

Algorytm śledzący wyznacza korek lotki jako punkt w następujący sposób.

- Dla kandydata, który został ostatecznie wybrany jako lotka wyznaczamy okalający go zrotowany prostokąt (*RECT*).

- Znając kierunek poruszania się lotki, wyznaczamy krótszy z boków tego prostokąta, a w jego środku punkt reprezentujący korek. Zwizualizowano to na rysunku [Rysunek 26] – błękitna kropka

Wyznaczona w ten sposób pozycja korka jest wystarczająca dla algorytmu śledzącego, ale niewystarczająca do wyznaczenia miejsca odbicia lotki o podłoże i oceny czy było pole czy aut. Dokładne wyznaczenie miejsca odbicia zostało opisane w rozdziale *Algorytm wyznaczania korka lotki i miejsca odbicia o podłoże*.



Rysunek 27 Akumulacyjna ramka różnicowa (po lewej) oraz kandydaci na lotkę (po prawej)

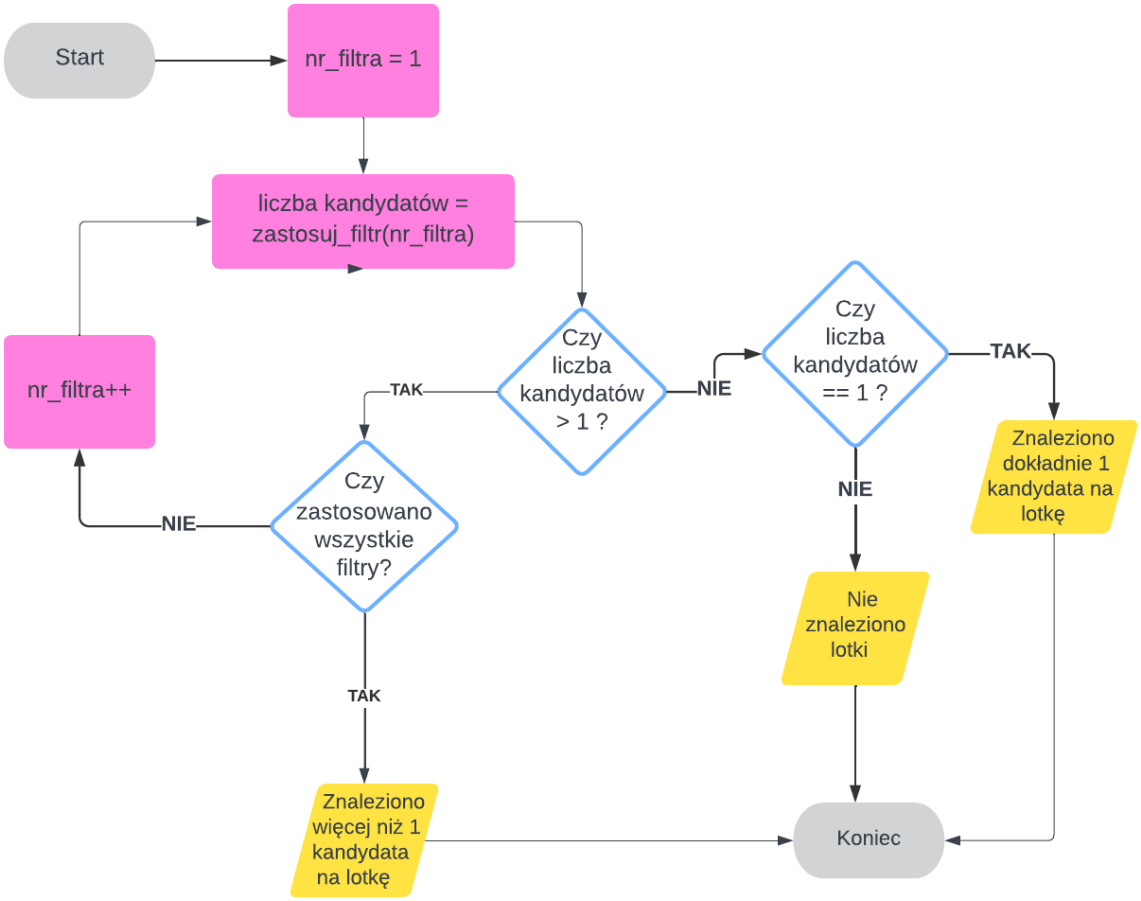
Kluczowym elementem rozwiązania jest zestaw filtrów. Filtry są uruchamiane sekwencyjnie tak długo jak pozostaje więcej niż jeden kandydat na lotkę. Na rysunku [Rysunek 28] przedstawiono schemat algorytmu filtrowania kandydatów. Kolejność filtrów i sposób ich działania przedstawia się następująco.

1. Usuń kandydatów o za dużym rozmiarze. Są to tacy, dla których wysokość w pikselach jest większa od wartości progowej *MaxHeigth* (sposób wyliczenia tej oraz innych wartości progowych opisano *Sposób wyznaczenie wartości progowych wykorzystywanych przez algorytm detekcji lotki*) lub szerokość jest większa od wartości *MaxWidth* (na rysunku [Rysunek 27] oznaczone kolorem błękitnym). Wartość *MaxHeigth* zależy od tego ile klatek na sekundę rejestrują kamera (FPS), a dokładnie

od tego jak długo jest naświetlany sensor kamery (im dłużej tym wartość *MaxHeight* większa – bo rozmycie lotki jest większe), odległości kamery od boiska oraz od tego z jaką rozdzielczością obrazu pracuje kamera (im wyższa rozdzielczość tym wyższa wartość *MaxHeight*). Natomiast wartość *MaxWidth* zależy od rozdzielczości obrazu i odległości kamery od boiska.

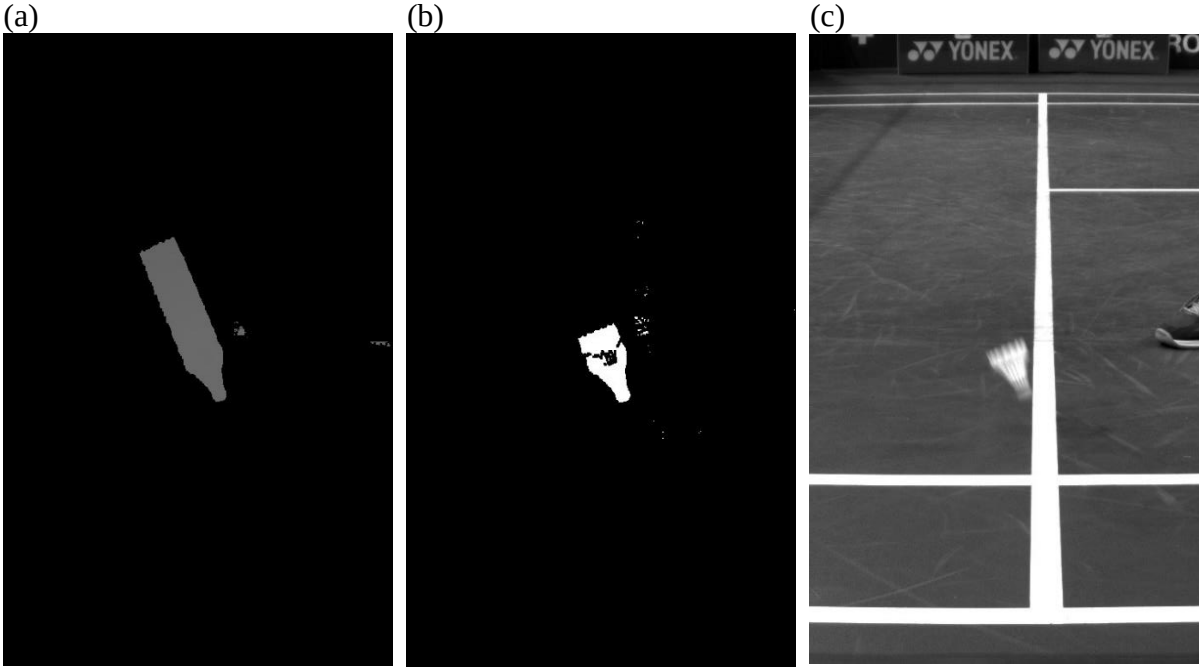
2. Usuń kandydatów o zbyt małych wymiarach. Takich, dla których wysokość jest mniejsza od wartości *MinHeigth* lub szerokość jest mniejsza od wartości *MinWidth*. Wartości te oraz *MaxHeight* i *MaxWidth* są wyliczane każdorazowo po ustawieniu kamer przy korcie i ustaleniu z jaką ilością klatek na sekundę będzie rejestrowany obraz.
3. Usuń tych kandydatów, którzy nie poruszają się w kierunku podłoża. Dla każdego kandydata mamy obliczony kąt, który z osią X tworzy wektor poruszania się obiektu. Jeśli ten kąt nie mieści się w uprzednio wcześniej wyznaczonym zakresie, to taki kandydat jest odrzucany. Sposób w jaki wyznaczono zakres tych kątów przedstawiono poniżej, w rozdziale *Wyznaczenie* zakresu kątów jaki może utworzyć trajektoria poruszającej się lotki w stosunku do podłoża.
4. Usuń kandydatów, dla których współczynnik *AR* – stosunek szerokości (*W*) do długości (*H*) nie mieści się w ustalonych granicach (*MinAspectRatio, MaxAspectRatio*).
5. Usuń kandydatów, których kształt nie przypomina zatemperowanego ołówka. W związku z tym, że kamery są ustawione na wysokości około 80 cm i patrzą w dół na linie, to lotka, która porusza się w kierunku podłoża na akumulowanej ramce różnicowej będzie miała kształt przypominający zatemperowany ołówek (patrz [Rysunek 29 (a)]). Ten filtr weryfikuje jak bardzo kształt kandydata odbiega od modelowego kształtu. Jeśli wyliczona odległość jest zbyt duża, to dany kandydat jest odrzucany. Szczegółowy opis wyliczenia tej miary znajduje się w tabeli [Tabela 3] – *DIST2SC*. Jeśli wartość tej miary jest większa niż 0,7 to kandydat jest odrzucany. Wartość progowa 0,7 została dobrana eksperymentalnie.
6. Jeśli w zbiorze kandydatów pozostał więcej niż jeden kandydat, to porównaj pozycję każdego z pozostałych kandydatów z pozycją lotki z poprzedniej ramki. Wybierz tych kandydatów, którzy są w odległości mniejszej niż 15cm od pozycji lotki z poprzedniej ramki - progowa wartość 15 cm wynika z maksymalnej drogi jaką może przebyć lotka w końcowej fazie lotu w czasie od 5ms do 6,6ms. Jeśli po tej operacji nie pozostał w zbiorze żaden kandydat, to oznacza, że najprawdopodobniej zgubiono lotkę (została przesłonięta) lub śledzono inny obiekt niż lotka (np. skarpetkę zawodnika). W takim

przypadku zwróć wszystkich kandydatów i przekaż ich do sieci neuronowej w celu dokonania wyboru jednego z nich.



Rysunek 28 Algorytm wyboru lotki spośród kandydatów na lotkę poprzez zastosowanie kolejnych filtrów

(a) (b) (c)



Rysunek 29 Przykładowe obrazy analizowane przez przedstawiony algorytm. (a) akumulacyjny ramka różnicowa, (b) ramka różnicowa, (c) oryginalna ramka zarejestrowana przez kamerę

Sposób wyznaczenie wartości progowych wykorzystywanych przez algorytm detekcji lotki

Poniżej opisano sposób wyznaczenia wartości progowych używanych w algorytmie detekcji lotki.

Wartość progowa ***MaxWidth*** w pikselach jest wyznaczana zgodnie ze wzorem:

$$\boldsymbol{MaxWidth} = REAL\_WIDTH * max\_image\_model\_distance\_ratio + MARGIN\_IN\_PIXELS$$

gdzie:
REAL_WIDTH - szerokość lotki w milimetrach określoną przepisami wynosząca 65mm,
max_image_model_distance_ratio - stosunek szerokości najbliższej kamerze linii w pikselach do jej rzeczywistej szerokości w milimetrach (40mm)
MARGIN_IN_PIXELS – stała wartość zapewniająca margines błędu - wyznaczona eksperymentalnie na 6 pikseli

Wartość progowa ***MinWidth*** w pikselach jest wyznaczana zgodnie ze wzorem:

$$\boldsymbol{MinWidth} = REAL\_WIDTH * min\_image\_model\_distance\_ratio$$

gdzie:
min_image_model_distance_ratio - stosunek szerokości najdalszej od kamery linii będącej po tej samej stronie siatki co kamera w pikselach do jej rzeczywistej szerokości w milimetrach (40mm)

Wartość progowa ***MinHeigth*** w pikselach jest wyznaczana zgodnie ze wzorem:

$$\boldsymbol{MinHeigth} = REAL\_HEIGHT * min\_image\_model\_distance\_ratio$$

gdzie:
REAL_HEIGHT - wysokość lotki w milimetrach określoną przepisami - 85mm

Wartość progowa ***MaxHeigth*** w pikselach jest wyznaczana zgodnie ze wzorem:

$$\boldsymbol{MaxHeigth} = (blur\_size + REAL\_HEIGHT) * max\_image\_model\_distance\_ratio * number\_of\_frames\_in\_accumulated\_diff$$

gdzie:
$blur\_size$ – wysokość rozmazania krawędzi piór lotki wynikająca z ruchu postępowego spowodowana czasem ekspozycji sensora kamery. Przykładowo, jeśli lotka porusza się ruchem jednostajnym z prędkością 200km/h, a czas ekspozycji wynosi 1/150 sek., to końce piór pokonają drogę 0.37m. blur_size wyznaczono zgodnie ze wzorem:

$$\boldsymbol{blur\_size} = \frac{max\_velocity}{FPS}$$

$max\_velocity$ – maksymalna prędkość jaką może mieć lotka przy linii w mm/sek. wyznaczona na podstawie obserwacji autora - 15m/sek. Nie zaobserwowano prędkości wyższych niż 9.72 m/s, ale wyznaczając tę wielkość przyjęto znaczny margines.
FPS – liczba klatek na sekundę rejestrowana przez kamerę
$number\_of\_frames\_in\_accumulated\_diff$ - stała wartość wyznaczona eksperymentalnie na 7

Wartość progowa ***MaxAspectRatio*** jest wyznaczana zgodnie ze wzorem:

$$\boldsymbol{MaxAspectRatio} = \frac{MaxHeight}{MaxWidth}$$

Wartość progowa ***MinAspectRatio*** jest wyznaczana zgodnie ze wzorem:

$$\boldsymbol{MinAspectRatio} = \frac{MinHeight}{MinWidth}$$

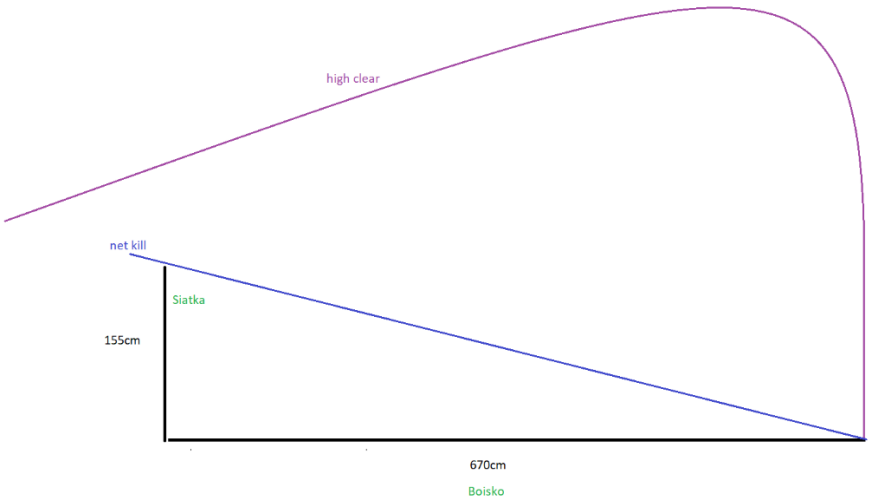
Wyznaczenie zakresu kątów jaki może utworzyć trajektoria poruszającej się lotki w stosunku do podłoża.

W zależności od tego, którą linię obserwuje kamera oraz biorąc pod uwagę ograniczenia związane ze specyfiką gry w badmintona, można określić zakres możliwych kątów jakie poruszająca się w kierunku podłoża lotka może utworzyć z osią oX (podłożem). [Rysunek 30]

pokazuje w przybliżony sposób trajektorie dwóch skrajnych uderzeń: high clear – uderzenie obronne, net kill – bardzo mocne uderzenie atakujące z bliskiej odległości od siatki, po których lotka upada na końcową linię (670 cm od siatki). Jak łatwo obliczyć, w skrajnym przypadku, aby lotka upadła w boisku, to trajektoria lotki, która przelatuje nad siatką (o wysokości 155 cm) może utworzyć kąt z podłożem od 13 (*arctan(155/670) = 13,02*) do 90 stopni. Dla linii serwisowej, która jest w odległości 198 cm od siatki, rozumując tak samo jak w przypadku linii końcowej, będą to wartości od 38 do 90 stopni. Uwzględniając jednakże specyfikę serwisu, czyli to że:

- zawodnik serwujący nie może serwować „z góry", tylko musi uderzyć lotkę na wysokości poniżej 115 cm,
- musi serwować ze swojego pola serwisowego – czyli w odległości co najmniej 198 cm od siatki,
- stara się zaserwować tuż nad siatką,
- zawodnik serwujący uderza lotkę dużo lżej niż podczas zbicia z siatki lub uderzenia smash, co powoduje, że trajektoria lotki jest paraboliczna, a nie zbliżona do linii prostej jak w przypadku uderzenia smash

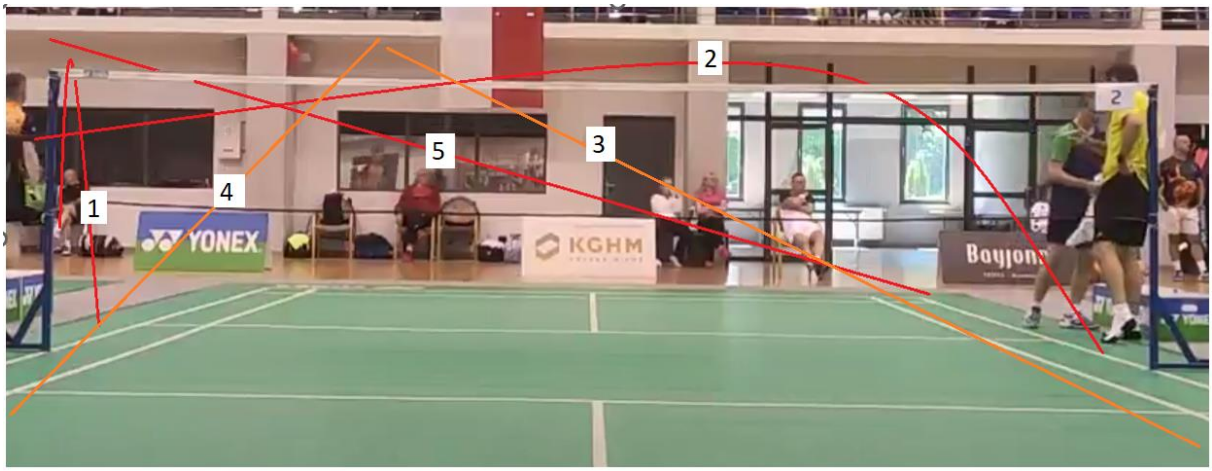
oraz obserwacje i doświadczenia autora, pozwalają założyć, że możliwy kąt jaki może utworzyć z podłożem lotka upadająca w pobliżu linii serwisowej będzie wynosił od 60 do 75 stopni. Na zdjęciu [Rysunek 29c] widać lotkę upadającą w okolicy linii serwisowej po serwisie zawodnika.



Rysunek 30 Trajektorie uderzeń high clear i net kill lotki upadającej na końcową linię

Podobnie rozumując, można wyznaczyć możliwy zakres kątów dla linii bocznej. Maksymalny kąt to 90 stopni dla uderzenia wzdłuż linii bocznej. Minimalny kąt zaobserwujemy dla zbicia z

siatki po przekątnej i wyliczany jest podobnie - uwzględniając wysokość siatki i szerokość boiska (610 cm) – *arctan(155/610) = 14.3* stopnia. Przykładowe trajektorie lotki zostały zaznaczone na zdjęciu [Rysunek 31].



Rysunek 31 Przykładowe uderzenia i trajektorie lotki dla uderzeń, które są przez zawodników najczęściej kierowane w okolicy linii bocznej
1 – Uderzenie short net
2 – Uderzenie net cross
3 – Uderzenie cross smash
4 – Uderzenie straight smash
5 – Uderzenie net kill

Wyznaczając ostateczne zakresy kątów, autor uwzględnił specyfikę uderzeń oraz wpływ klimatyzacji (z obserwacji poczynionych przez autora wynika, że w dużych obiektach sportowych poruszające się powietrze może wpłynąć na opadającą lotkę i spowodować jej przesunięcie nawet o 30-40cm).

W szczególności uderzenie net kill (zbicie), dla którego przyjęto trajektorię jako linię prostą, bardzo rzadko kończy się upadkiem lotki na linię końcową. Najczęściej lotka po takim uderzeniu upada w połowie kortu. Jeśli jednak upadnie na linię końcową, to trajektoria nie będzie już linią prostą. Podobnie ze jest ze zbiciem po przekątnej. Jeśli takie zbicie jest wzdłuż całej długości siatki to trajektoria nie będzie już linią prostą – po prostu nie da się uderzyć tak mocno lotki, aby można było przyjąć linię prostą jako trajektorię. W związku z powyższym autor skorygował wyliczone wcześniej zakresy kątów i ostatecznie w filtrze uwzględniającym możliwe zakresy kątów przyjęto następujące wartości:

Linia	Kąt minimalny	Kąt maksymalny
Końcowa	25	95
Serwisowa	60	80
Boczna	30	95

Tabela 4 Zakres kątów w stopniach jakie trajektoria lotki może utworzyć z podłożem

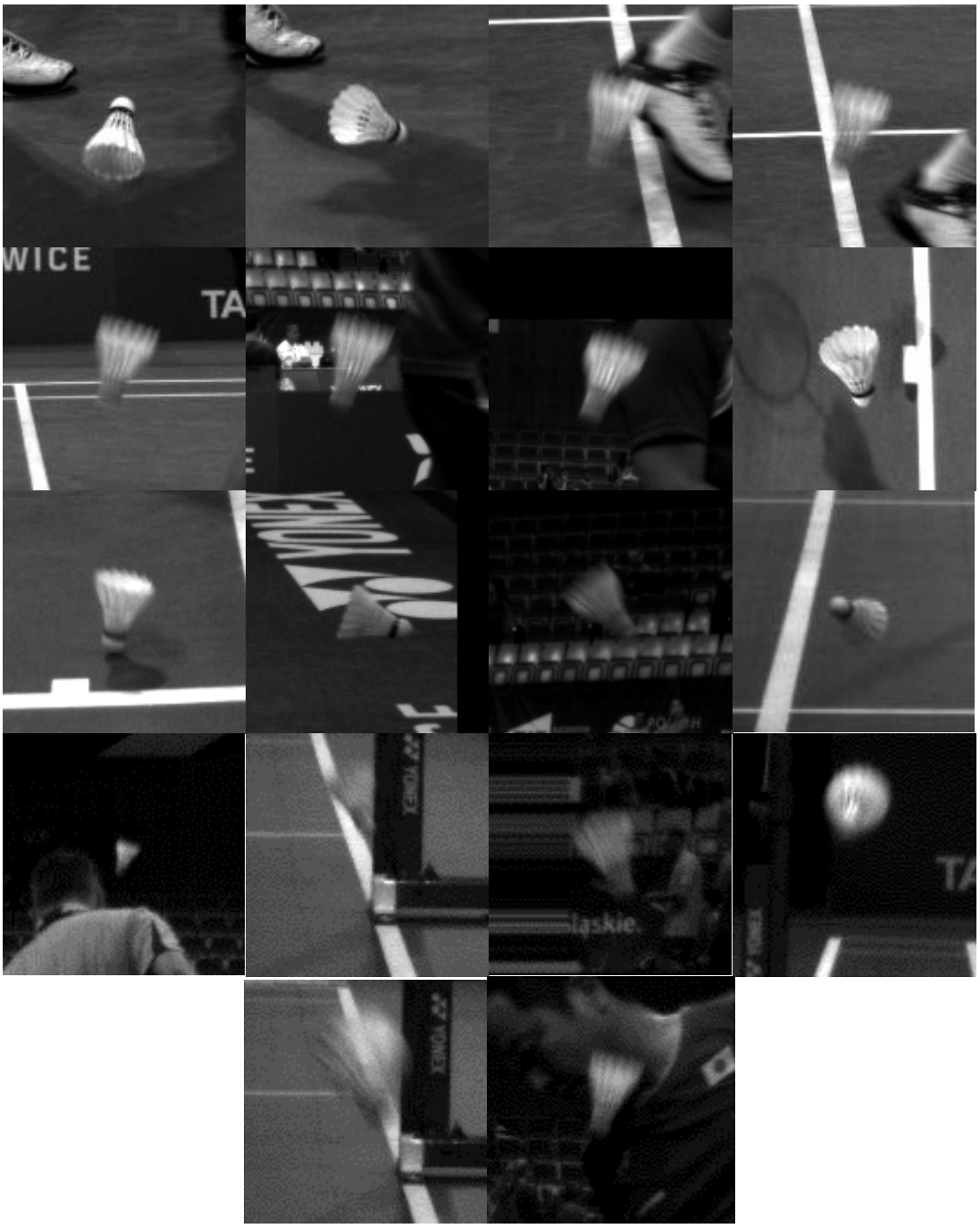
Wartości te zostały potwierdzone eksperymentalnie.

Przedstawiona metoda pozwala na szybkie znalezienie lotki na obrazie. Cały proces trwa poniżej 2 ms na komputerze wyposażonym w procesor Intel Core i7 9th Gen. W sytuacji, gdy nie znaleziono lotki lub pozostało więcej niż 2 kandydatów to uruchamiany jest dodatkowy proces wyszukiwania lotki przez sieć neuronową. Proces ten trwa 12ms na komputerze wyposażonym w CPU: 9th Gen. Intel® Core™ i7, GPU: NVIDIA® GTX 1080.

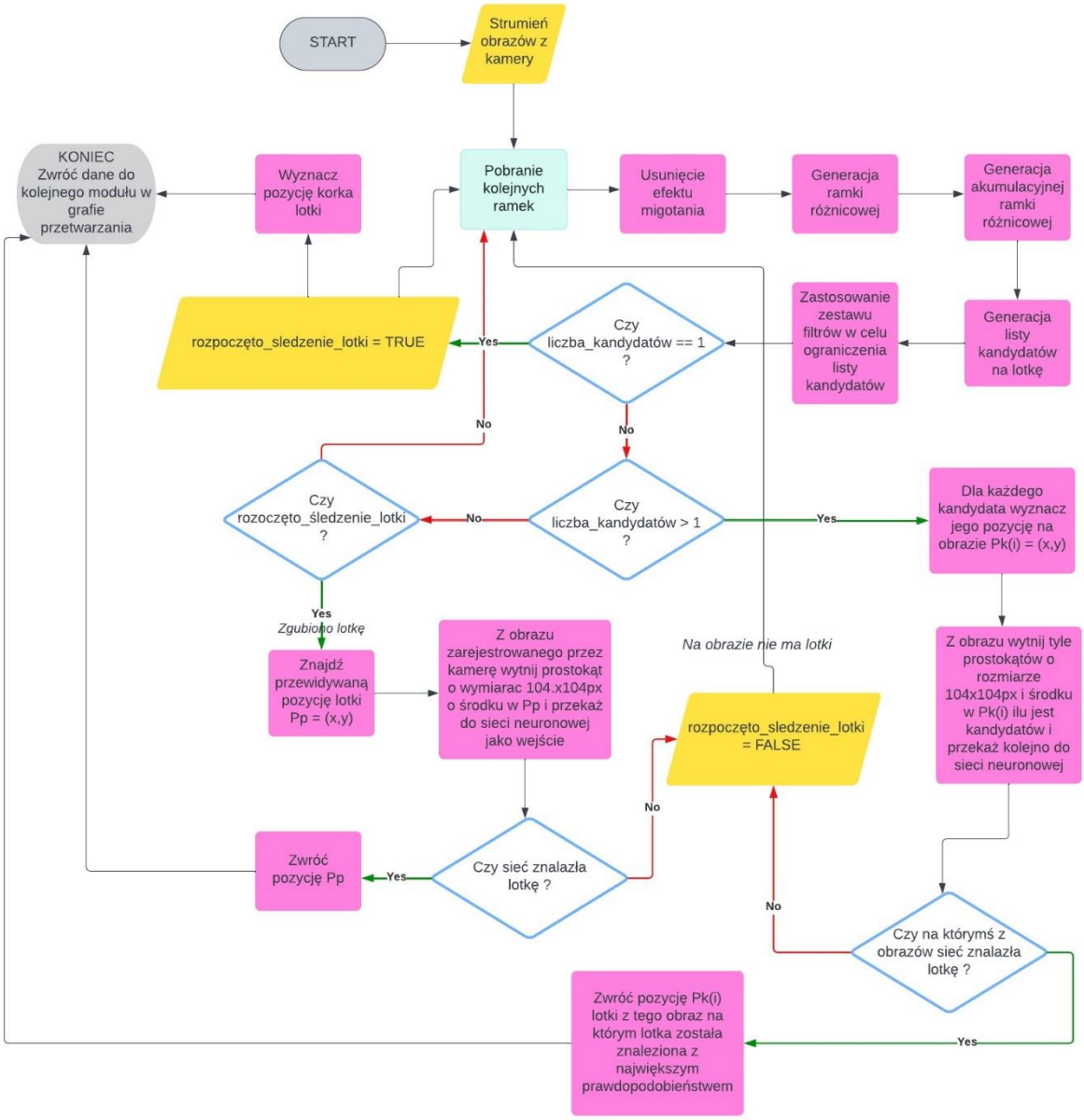
Wytrenowana sieć neuronowa to zmodyfikowany model Tiny Yolo3 [111]. Sieć została wytrenowana zestawem danych zebranych podczas turniejów badmintona. Oryginalna sieć Tiny Yolo3 oczekuje na wejściu trzykanałowych zdjęć. W związku z tym, że użyto kamer szaroodcieniowych, to architektura została zmieniona tak, że sieć akceptuje obrazy jednokanałowe. Na rysunku [Rysunek 32] przedstawiono przykładowe obrazy użyte do treningu sieci. Dla tej sieci osiągnięto dla zbioru walidacyjnego miarę mAP@0.50 - 94%, recall - 77% oraz F1-score[14] na poziomie 86%.

Kluczowe dla całości rozwiązania jest to, aby działało w czasie rzeczywistym dla kamer rejestrujących obraz z prędkością do 200 klatek na sekundę. To założenie wymusza średni czas przetwarzania poniżej 5ms. Dzięki temu, że sieć neuronowa otrzymuje tylko wycinek całego obrazu z kamery oraz dzięki temu, że jest uruchamiana tylko w momencie, gdy pierwszy proces nie dał jednoznacznej odpowiedzi to średni czas przetwarzania jednego obrazu jest poniżej 3ms. Na rysunku [Rysunek 33] zaprezentowano strukturę algorytmu znajdowania i śledzenia lotki. Dwa przykładowe wyniki działania tego algorytmu w postaci filmów w zwolnionym tempie można zobaczyć pod adresami: https://drive.google.com/file/d/1rsvcY-5Q25iN5Hmcv4nW_DXIWq9ZAR7J/view?usp=sharing oraz https://drive.google.com/file/d/1-Btu3YHeyYYm29MncjXMvjHcUePi2EI_/view?usp=sharing.

[14] Sposób wyznaczenia miary mAP, recall, precision, F1-score podano w *Słowniczek pojęć* na stronach 94-95



Rysunek 32 Przykładowe obrazy, na których trenowano sieć neuronową



Rysunek 33 Schemat algorytmu znajdowania i śledzenia lotki

Wyniki i ich omówienie

Zgromadzone dane ground-truth połączono z danymi z trakera i dokonano weryfikacji czy pozycja lotki zwrócona przez traker zgadza się z pozycją oznaczoną przez użytkownika. Przyjęto, że traker odnalazł lotkę we właściwym miejscu na obrazie, jeśli odległość pozycji korka lotki zaznaczonej przez operatora od pozycji wyznaczonej przez traker wyniosła poniżej 16 pikseli. W przeciwnym razie uznajemy, że traker zgubił lotkę.

Dla 1524 ramek na których operator zaznaczył lotkę, algorytm wykorzystujący akumulacyjne ramki różnicowe (MR) poprawnie znalazł lotkę w 1250 przypadkach. W 137 przypadkach lotka została zgubiona przez algorytm MR, a wycinek obrazu z przewidywaną pozycją lotki został

przekazany do modułu z siecią neuronową (MNN). Sieć odnalazła lotkę w 100 z 137 przypadkach. Szczegółowe wyniki dla modułu MR i połączonych moduł MR + MNN przedstawiono w tabeli [Tabela 5].

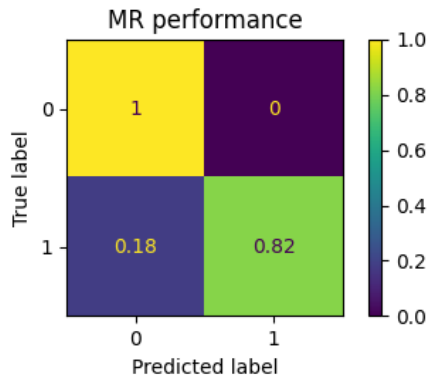
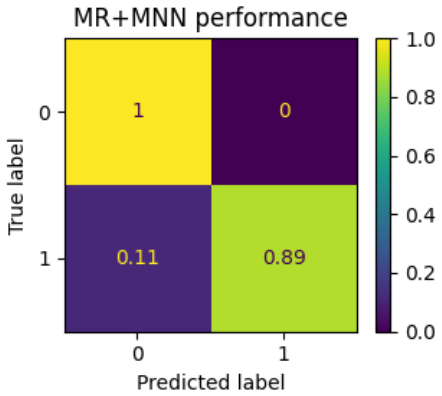
Model	Macierz pomyłek[15]	Accuracy	Recall	Preci -sion	F1	Balanced accuracy[16]
MR		0.98	0.82	1.00	0.90	0.91
MR + MNN		0.98	0.89	1.00	0.94	0.94

Tabela 5 Wyniki skuteczności opracowanej metody znajdowania lotki

Ja widać opracowana metoda w 11% zgubiła śledzoną lotkę. Działo się to w dwóch przypadkach:

1. lotka została zgubiona po odbiciu o podłoże,
2. lotka została zgubiona, ponieważ poruszała się na tle zawodnika.

Przypadek pierwszy występował zdecydowanie częściej. Na rysunkach [Rysunek 20, Rysunek 21] zaprezentowano dwa przypadki błędnie wyznaczonej pozycji korka przez traker (żółta kropka na końcu żółtej linii dla przykładów z prawej strony). Z punktu widzenia celu badań, śledzenie lotki po odbiciu o podłoże nie jest konieczne, gdyż opracowana metoda lokalizacji

[15] Definicja podana w *Słowniczek pojęć* punkt 6, strona 93.
[16] Sposób wyliczenia miar Accuracy, Recall, Precision, F1, Balanced accuracy podano w *Słowniczek pojęć* na stronie 94

miejsca upadku lotki w stosunku do linii kortu nie wymaga śledzenia lotki po jej odbiciu. W zasadzie można by usunąć z danych takie przypadki i tym samym uzyskać lepsze wyniki (w zaimplementowanym ostatecznie rozwiązaniu traker przestaje śledzić lotkę po jej odbiciu). Przypadek drugi występował tylko dla nagrań z kamer obserwujących linie tylną (a tych jest około 1/3). W przypadku linii bocznych zawodnicy zawsze byli na tyle daleko od lotki, że nigdy jej nie przesłaniali.

Całkowity brak fałszywie pozytywnych alarmów (lotki nie było na nagraniu, a traker ją znalazł) wynika ze sposobu gromadzenia danych. Nagrania były rejestrowane tylko podczas realnych zawodów i tylko wtedy, gdy zawodnik prosił o weryfikację decyzji sędziego – a wtedy lotka jest dość długo w polu widzenia kamery. Sytuacji, gdy na nagraniu nie ma lotki, a jest inny poruszający się obiekt (np. zawodnik) było stosunkowo niewiele, a opracowany traker doskonale sobie z takimi sytuacjami poradził.

Wyznaczenie w strumieniu wideo kluczowej ramki, w której nastąpiło odbicie lotki o podłoże

Aby była możliwa ocena, czy lotka upadała w polu gry, czy poza nim, konieczne jest, znalezienie w strumieniu wideo ramki, w której faktycznie nastąpiło odbicie lotki o podłoże.

Autor przeprowadził szereg eksperymentów w celu opracowania skutecznej metody wyznaczenia ramki, w której nastąpiło odbicie o lotki o podłoże. Eksperymenty rozpoczęto od stworzenia prostego estymatora (*Model Podstawowy MP*), który na podstawie tego jak zmienia się wielkość konturu reprezentującego lotkę na ramce różnicowej decydował czy lotka odbiła się od podłoża czy nie. Następnie wyliczono szereg innych cech, które były wejściem do różnych algorytmów uczenia maszynowego. Jak już wspomniano wcześniej, istotnym ograniczeniem jest czas w związku z tym algorytmy, w których przeprowadzane są czasochłonne obliczenia nie mogą zostać wykorzystane w docelowej komercyjnej implementacji.

Zbiór danych

Do eksperymentów autor przygotował treningowy oraz walidacyjny zbiór danych składający się z nagrań zarejestrowanych podczas turniejów badmintona w różnych halach. Kamery szaroodcieniowe rejestrowały obraz z prędkością od 150 do 200 klatek na sekundę (FPS) w rozdzielczości 800x600 pikseli. W rozdziale *3.1 Gromadzenie danych do eksperymentów* opisano sposób w jaki przygotowano reprezentatywny zbioru danych. Po wydzieleniu z całego zbioru danych zbioru testowego (nagrania z Katowic i Głubczyc) pozostało 11158 próbek z

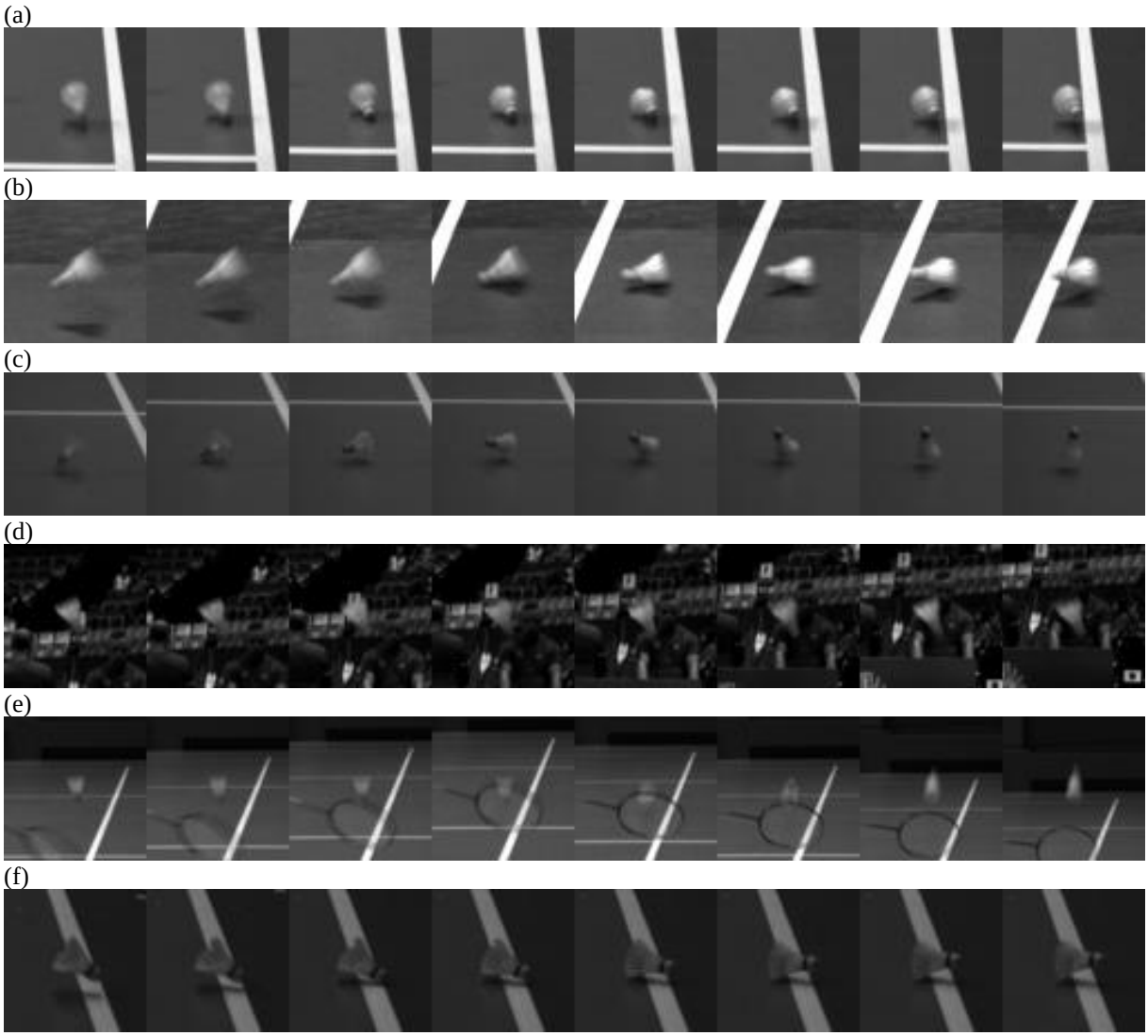
czego liczność klasy 0 (brak odbicia o podłoże) wynosi 10960, a klasy 1 (obicie o podłoże) – 198. Zbiór ten podzielono na treningowy i walidacyjny w stosunku 75:25. Taki podział zapewnia, że zarówno w zbiorze treningowym jak i walidacyjnym są nagrania z kamer obserwujących każdą linie – końcowe, boczne, serwisowe.

Jak widać dane są wysoce niezbalansowane stosunek klasy 0 do 1 wynosi 55:1. Dlatego też do oceny skuteczności badanych metod zastosowano miarę balanced accuracy.

Dane zostały zaanotowane i oznaczono ramkę, w której nastąpiło odbicie o ziemię, oraz pozycję korka. Czasami osoba anotująca dane miała problem z oceną w której ramce nastąpiło odbicie o kort. Przyczyny były następujące: lotka była w dużej odległości od kamery, lub lotka dotykała podłoża przez dwie ramki ślizgając się po korcie [Rysunek 34 a,b]. Wzięto tę niepewność pod uwagę podczas oceny skuteczności algorytmu i wyznaczono dokładność stosując dwa kryteria:

1) algorytm wyznaczył moment odbicia w dokładnie tej samej ramce co ground-truth,
2) algorytm wyznaczył moment odbicia z tolerancją jednej ramki w stosunku do ground-truth.

Na rysunku [Rysunek 34] zaprezentowano przykładowe zestawy ramek, które były w zbiorze danych wykorzystanym w eksperymentach.

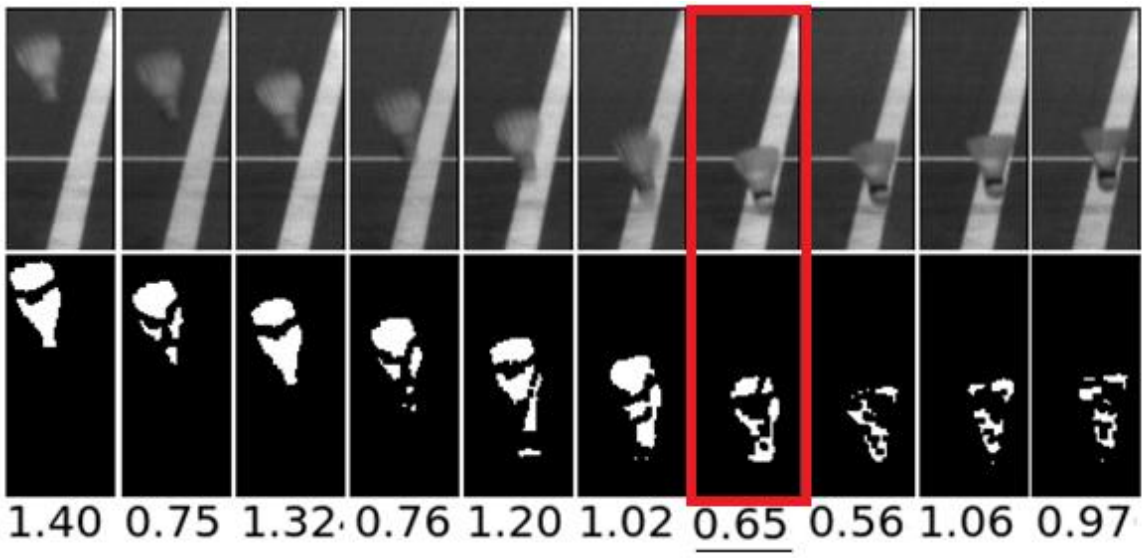


Rysunek 34 Przykładowe zestawy kolejnych ramek ze strumienia wideo, które znalazły się w zbiorze danych wykorzystanym w eksperymentach

W momencie, gdy lotka odbija się od podłoża jej prędkość poruszania się gwałtownie maleje. W przypadku, gdy lotka ślizga się po korcie jej prędkość również maleje, ale nie tak gwałtownie jak w przypadku odbicia. To czy lotka odbije się od podłoża, czy też przez chwilę prześlizgnie po nim, zależy głównie od uderzenia zawodnika (tego pod jakim kątem lotka upada na podłoże) oraz od współczynnika tarcia maty na której grają zawodnicy. Certyfikowane maty sportowe powinny spełniać normę EN 14904, która dopuszcza pewien zakres współczynnika tarcia jaki powinna spełniać taka mata (może się różnić nawet o ponad 30%). Nie bez znaczenia na ten współczynnik ma również zużycie maty. Ze względu na specyfikę gry sytuacje, w których lotka upada na kort pod mały kątem i się ślizga, są stosunkowo rzadkie. W większości przypadków lotka odbija się sprężyście od podłoża.

Model podstawowy

Model podstawowy - **MP** zbudowano na podstawie obserwacji, że w momencie odbicia lotka gwałtownie zmniejsza swoją prędkość. Obserwując ramki różnicowe, można zauważyć, że liczba pikseli reprezentujących różnice pomiędzy kolejnymi ramkami o numerach *t-1* i *t* będzie najmniejsza dla ramki w momencie odbicia. Aby uniezależnić się od odległości lotki od kamery wyznaczono współczynnik *diff_size_change_ratio* $D = \frac{DS(t)}{DS(t-1)}$, który określa jak gwałtownie zmieniła się liczba pikseli na ramce różnicowej pomiędzy ramkami *t-1* i *t*. *DS(t)* – oznacza liczbę pikseli na ramce różnicowej o numerze *t*. Im wartość bliższa zeru tym gwałtowniejsza zmiana prędkości poruszania się lotki. Na rysunku [Rysunek 35] przedstawiono przykładową sekwencję obrazów zawierającą odbicie lotki o podłoże oraz wartości współczynnika *D*.



Rysunek 35 Ramki różnicowe przed odbiciem, w momencie odbicia i po odbiciu oraz wartość współczynnika D. Czerwoną obwiednią zaznaczono ramkę w której nastąpiło odbicie o podłoże.

Na podstawie treningowego zbioru danych wyznaczono wartość progową współczynnika *D* dla których miara balanced accuracy miała największą wartość. Osiągnięto wartość: – $D_{min} = 0.88$. Jeśli wartość *D* spełnia równanie: $D < D_{min}$, to model zwraca informacje o tym, że w danej ramce nastąpiło odbicie o podłoże, w przeciwnym wypadku – brak odbicia. Skuteczność takiego modelu przedstawiono w tabeli [Tabela 6]

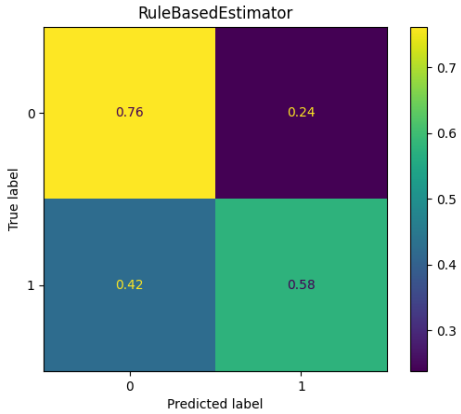
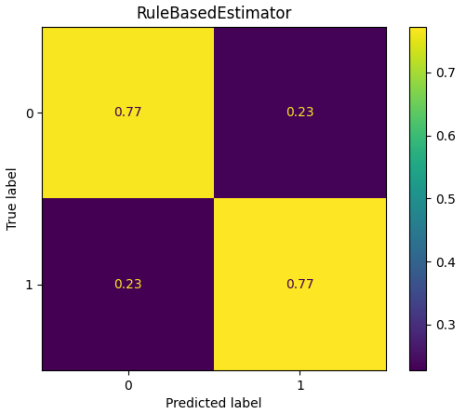
Model	Macierz pomyłek	Accuracy	Recall	Precision	F1	Balanced accuracy
MP		0.761	0.582	0.053	0.093	0.675
MP z tolerancją jednej ramki		0.772	0.770	0.062	0.123	0.776

Tabela 6 Skuteczność modelu MP

Jak widać skuteczność modelu *MP* nie jest wyjątkowo wysoka (szczególnie niska jest precyzja), za to obliczeniowo model ten jest wyjątkowo mało wymagający. W celu poprawy wyników, autor podjął dalsze prace badawcze i wyliczył więcej cech i zbadał, które z nich są najbardziej użyteczne oraz, które ze znanych modeli uczenia maszynowego dają najlepsze wyniki.

Badanie modeli uczenia maszynowego

Z punktu widzenia modeli uczenia maszynowych, kluczowe jest wyznaczenie takich cech, aby model mógł na ich podstawie w optymalny sposób dokonać klasyfikacji. Duża liczba modeli opisywanych w literaturze wymaga, aby dane były zbalansowane. Jeśli dane są niezbalansowane, to przed treningiem modelu należy je odpowiednio zbalansować. Metody balansowania danych można podzielić na dwie grupy: powiększania klasy mniejszościowej (oversampling) i zmniejszania klasy większościowej (undersampling). Do najpopularniejszych metod balansowania danych należą random unersampling, random oversampling, smote, tomek links, near miss. Autor w ramach badań zweryfikował skuteczność wielu modeli wraz różnymi metodami balansowania danych. Wszystkie implementacje opisywanych modeli oraz metod

balansowania danych zostały pobrane z biblioteki scikit-learn [112]. Wyniki przedstawiono w tabeli [Tabela 9]. Kolejnym etapem badań było sprawdzenie, czy dla modeli, które dały najlepsze wyniki ich wersje dla danych niezbalansowanych dadzą lepsze wyniki. Wyniki tych eksperymentów przedstawiono w tabeli [Tabela 10].

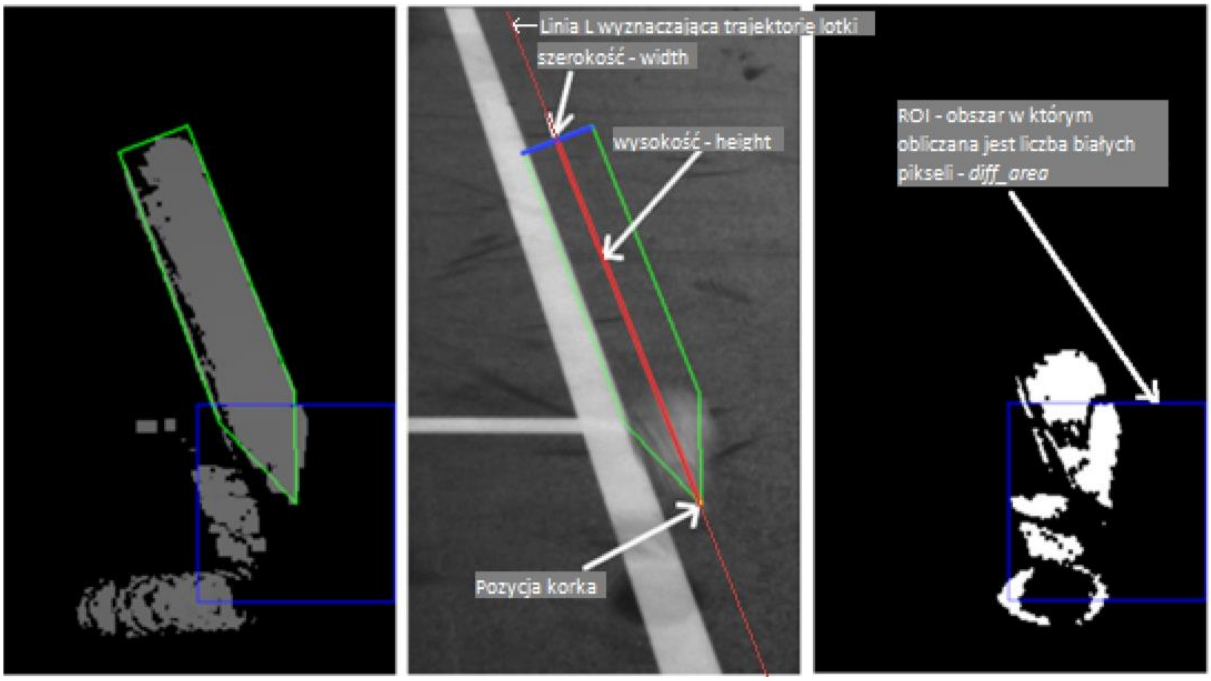
Wyznaczenie zbioru cech

Cechy zostały wyliczone na podstawie przetworzonych danych ze strumienia wideo takich jak wspomniane wcześniej ramki różnicowe i akumulacyjne ramki różnicowe. W tabeli [Tabela 7] przedstawiono wyznaczone cechy, a na rysunku [Rysunek 36] wizualizację wybranych cech.

Cecha	Opis
diff_size_change_ratio (D)	Jak bardzo zmieniła się liczba pikseli reprezentujących różnicę pomiędzy kolejnymi ramkami ze strumienia wideo. Szczegółowo opisane w rozdziale *Model podstawowy.*
angle_change_ratio	Jak zmienił się kąt który tworzy trajektoria poruszającej się lotki z podłożem. Obliczony jako: $\frac{kąt\ w\ ramce\ bieżącej}{kąt\ w\ ramce\ poprzedniej}$
acc_diff_to_diff_ratio	Obliczane podobnie jak *diif_size_change_ratio* z tą różnicą, że źródłem danych jest akumulowana ramka różnicowa, a nie ramka różnicowa
dist_travelled	W pikselach euklidesowa odległość jaką pokonał korek lotki w czasie $\frac{1}{FPS}$. Czyli odległość pomiędzy pozycją korka w ramce *n* a ramce poprzedniej – *n-1*.
prev_3_frames_dist_travelled	W pikselach euklidesowa odległość pomiędzy pozycją korka w ramce bieżącej: *n,* a ramce o 3 wcześniejszej: *n-3*.
local_min_from_3_points	Ta wartość mówi jak bardzo zmienia się wartość *diff_size_change_ratio* na przestrzeni 3 ramek. Obliczona jako: *(prev_size_change_ratio - diff_size_change_ratio) + (next_diff_size_change_ratio - diff_size_change_ratio),* gdzie: *prev_size_change_ratio* oznacza wyliczone *diff_size_change_ratio* w ramce *n-1*, a *next_diff_size_change_ratio* oznacza wyliczone *diff_size_change_ratio* w ramce *n+1*
local_min	Wartość będąca sumą **diff_size_change_ratio** dla kolejnych ramek, dla których **diff_size_change_ratio** maleje. Można to traktować jako swego rodzaju wartość gradientu spadku **diff_size_change_ratio.** Algorytm obliczenia tej cechy reprezentuje poniższy pseudo kod: *local_min = 0, i=current_frame_index -1* *prev_size_change_ratio = diff_size_change_ratio* for frame *i* **while** *prev_size_change_ratio > diff_size_change_ratio:* *local_min += (prev_size_change_ratio - diff_size_change_ratio)* *i = i-1* *prev_size_change_ratio = diff_size_change_ratio* for frame *i* *i=current_frame_index +1* *next_diff_size_change_ratio = diff_size_change_ratio* for frame *i* **while** *next_diff_size_change_ratio > diff_size_change_ratio:* *local_min += (next_diff_size_change_ratio - diff_size_change_ratio)* *i = i+1* *next_diff_size_change_ratio = diff_size_change_ratio* for frame *i*
aspect_ratio	Obliczone jako: $\frac{height}{width}$
height_change_ratio	Obliczone jako: $\frac{height}{height\ dla\ ramki\ poprzedniej}$
width_change_ratio	Obliczone jako: $\frac{width}{width\ dla\ ramki\ poprzedniej}$

dist_travelled_wrt_width	Obliczone jako: $\frac{dist\_travelled}{width}$
prev_3_frames_dist_travelled_wrt_width	Obliczone jako: $\frac{prev\_3\_frames\_dist\_travelled}{width}$

Tabela 7 Cechy oraz sposób ich wyznaczenia. Pogrubioną czcionką zaznaczono te, które ostatecznie zostały wybrane.



Rysunek 36 Sposób wyznaczenia cech width, height, pozycji korka i obszaru zainteresowania. Od lewej – akumulacyjna ramka różnicowa, oryginalna ramka ze strumienia wideo, ramka różnicowa. ROI jest to obszar zdefiniowany jako prostokąt o boku 128 pikseli o środku w punkcie (c_x, c_y) gdzie (c_x, c_y) jest pozycją korka zwróconą przez traker.

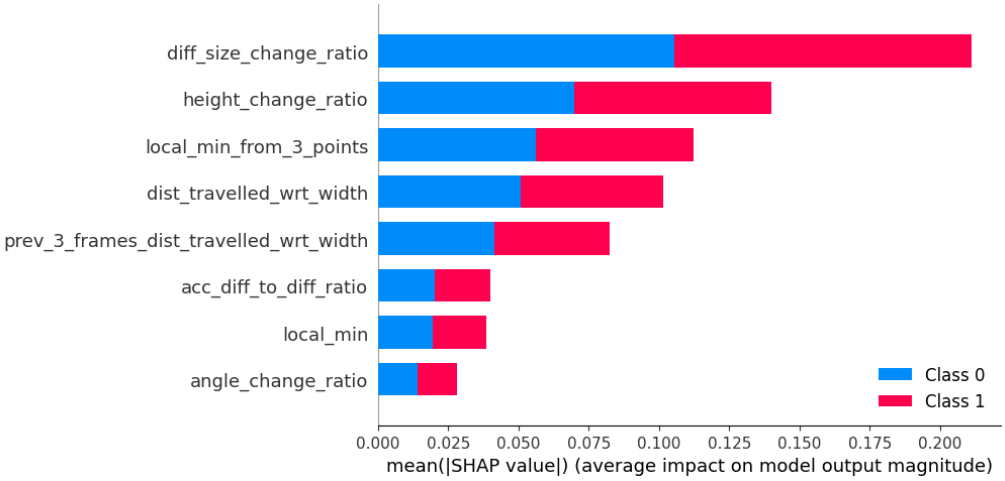
Z wykorzystaniem nadzorowanej metody rekursywnego usuwania cech dokonano wyboru cech. Przeprowadzono również 10 eksperymentów z różnymi zestawami cech. Wyniki tych eksperymentów dla klasyfikatora BalancedRandomForrest zaprezentowano w tabeli [Tabela 8]. Dla pozostałych testowanych klasyfikatorów wyniki były podobne. Zestaw cech nr 8 okazał się najlepszy. Dla wybranego zestawu cech policzono ich korelacje [Rysunek 37] oraz wpływ jaki mają na model [Rysunek 38].

Nr	Zestaw cech	Balanced accuracy
0	diff_size_change_ratio, angle_change_ratio,acc_diff_to_diff_ratio, dist_travelled, prev_3_frames_dist_travelled, dist_trajectory_line_from_prev_3_frames, local_min,local_min_from_3_points,aspect_ratio,height_change_ratio,width_change_ratio,dist_travelled_wrt_width, prev_3_frames_dist_travelled_wrt_width	0.843
1	diff_size_change_ratio, acc_diff_to_diff_ratio, dist_travelled,prev_3_frames_dist_travelled, local_min, local_min_from_3_points,aspect_ratio, height_change_ratio, width_change_ratio	0.838
2	diff_size_change_ratio, acc_diff_to_diff_ratio, dist_travelled,prev_3_frames_dist_travelled, local_min,aspect_ratio, height_change_ratio, width_change_ratio	0.769

3	diff_size_change_ratio, acc_diff_to_diff_ratio, dist_travelled_wrt_width,prev_3_frames_dist_travelled, local_min, local_min_from_3_points,aspect_ratio, height_change_ratio, width_change_ratio,prev_3_frames_dist_travelled_wrt_width	0.846
4	diff_size_change_ratio, angle_change_ratio,acc_diff_to_diff_ratio, dist_travelled, prev_3_frames_dist_travelled, dist_trajectory_line_from_prev_3_frames,local_min,local_min_from_3_points	0.840
5	diff_size_change_ratio, angle_change_ratio,acc_diff_to_diff_ratio, dist_travelled, prev_3_frames_dist_travelled, local_min,local_min_from_3_points	0.863
6	diff_size_change_ratio, angle_change_ratio,acc_diff_to_diff_ratio, dist_travelled, prev_3_frames_dist_travelled, local_min,local_min_from_3_points,aspect_ratio	0.850
7	diff_size_change_ratio, angle_change_ratio,acc_diff_to_diff_ratio, dist_travelled, prev_3_frames_dist_travelled, local_min,local_min_from_3_points,height_change_ratio	0.842
8	**diff_size_change_ratio, angle_change_ratio,acc_diff_to_diff_ratio, dist_travelled_wrt_width, prev_3_frames_dist_travelled_wrt_width, local_min,local_min_from_3_points,height_change_ratio**	**0.851**
9	diff_size_change_ratio, acc_diff_to_diff_ratio, dist_travelled_wrt_width, prev_3_frames_dist_travelled_wrt_width, local_min,local_min_from_3_points,height_change_ratio	0.849

Tabela 8 Porównanie miar balanced accuracy dla różnych zestawów cech modelu BalancedRandomForestClassifier(n_estimators= 226, criterion= "entropy", max_depth= 11, max_features= 0.3, min_samples_leaf=1, sampling_strategy= "not minority", bootstrap= True, oob_score= True, replacement= False, random_state=0, n_jobs=-1)

Rysunek 37 Współczynnik korelacji Pearson-a dla wybranych cech



Rysunek 38 Wpływ poszczególnych cech na wynik klasyfikacji dla modelu BalancedRandomForest

Wyniki

W pierwszej kolejności, autor zbadał jakie wyniki dla wszystkich cech dają popularne modele klasyfikacji. Ze względu na niezbalansowany zbiór danych miary F1, Precision, Recall są obliczone jako średnia ważona uwzględniająca liczność klasy 0 i 1. Szczegółowo wyniki przedstawiono w tabeli [Tabela 9]. Do dalszych eksperymentów zostały wybrane modele, które dały najlepsze rezultaty. Kolejne testy zostały przeprowadzone dla tych modeli, które są przystosowane do niezbalansowanych danych - BalancedRandomForest oraz BalancedBaggingClassifier – [Tabela 10]. Ostatnim krokiem była optymalizacja hyperparametrów zarówno dla wersji z dokładnym rozpoznaniem przez klasyfikator ramki, w której nastąpiło odbicie jak i z rozpoznaniem momentu odbicia z tolerancją jednej ramki. Wyznaczone optymalne wartości hyperparametrów przedstawiono w tabelach [Tabela 11] (dokładny moment odbicia) oraz [Tabela 14] (moment odbicia z tolerancją jednej ramki). Ostateczne wyniki przedstawiono w tabeli [Tabela 12] (dokładny moment odbicia) oraz [Tabela 15] (moment odbicia z tolerancją jednej ramki). Wykonane eksperymenty polegające na wybraniu najważniejszych cech oraz optymalizacji parametrów klasyfikatora wyznaczającego dokładny momentu odbicia lotki o podłoże pozwoliły poprawić wynik z 0.843 do 0.873 dla miary ballanced accuracy i modelu BalancedRandomForest. Podobnie dla modelu BalancedBaggingClassifier – optymalizacja hyperparmetrów pozwoliła na poprawę miary ballanced accuracy z 0.849 do 0.878. Podsumowując wynik przeprowadzonych eksperymentów, zamiast balansować dane przed treningiem modelu, lepiej jest zastosować model, który sam podczas treningu na podstawie liczności klas dokona odpowiedniej strategii samplowania danych oraz przydzielania odpowiednich wag. Przykładowo, dla modelu RandomForrest po zbalansowaniu danych uzyskano ballanced accuracy na poziomie 0.781, a

dla BallancedRandomForest [113], gdzie każde drzewo ma odpowiednio zbalansowaną próbkę danych- 0.843. Czyli o 6% lepszy wynik.

Warto nadmienić, że wybranie najlepszych cech oraz optymalizacja hyperparametrów modeli pozwoliły poprawić wynik o 3%.

Klasyfikator	Metoda balansowania	F1	Precision	Recall	Accuracy	Bal acc
GaussianNB	RandomOverSampler	0,033	0,017	0,907	0,144	0,519
	ADASYN	0,034	0,017	0,907	0,162	0,529
	NearMiss	0,031	0,016	0,481	0,507	0,495
	SMOTE	0,034	0,017	0,907	0,168	0,532
	AllKNN	0,036	0,018	0,944	0,178	0,555
	RandomUnderSampler	0,036	0,018	0,944	0,172	0,552
	TomekLinks	0,034	0,017	0,907	0,168	0,531
XGBClassifier	RandomOverSampler	0,217	0,263	0,185	0,978	0,588
	ADASYN	0,172	0,116	0,333	0,948	0,646
	NearMiss	0,041	0,021	0,870	0,350	0,606
	SMOTE	0,197	0,137	0,352	0,954	0,658
	AllKNN	0,274	0,317	0,241	0,979	0,616
	RandomUnderSampler	0,100	0,054	0,796	0,770	**0,783**
	TomekLinks	0,141	0,294	0,093	0,982	0,544
DecisionTree	RandomOverSampler	0,202	0,185	0,222	0,972	0,603
	ADASYN	0,177	0,110	0,463	0,931	0,701
	NearMiss	0,038	0,019	0,815	0,326	0,567
	SMOTE	0,174	0,107	0,463	0,929	0,700
	AllKNN	0,123	0,092	0,185	0,957	0,578
	RandomUnderSampler	0,076	0,040	0,704	0,723	0,713
	TomekLinks	0,137	0,117	0,167	0,966	0,573
RandomForest	RandomOverSampler	0,092	0,273	0,056	0,982	0,527
	ADASYN	0,274	0,217	0,370	0,968	0,674
	NearMiss	0,043	0,022	0,944	0,314	0,624
	SMOTE	0,257	0,209	0,333	0,969	0,656
	AllKNN	0,034	0,250	0,019	0,983	0,509
	RandomUnderSampler	0,099	0,053	0,796	0,766	**0,781**
	TomekLinks	0,000	0,000	0,000	0,983	0,499
HistGradientBoost	RandomOverSampler	0,270	0,236	0,315	0,973	0,649
	ADASYN	0,196	0,123	0,481	0,936	0,713
	NearMiss	0,041	0,021	0,889	0,322	0,601
	SMOTE	0,224	0,143	0,519	0,942	0,734
	AllKNN	0,141	0,194	0,111	0,978	0,552
	RandomUnderSampler	0,105	0,056	0,815	0,776	**0,795**
	TomekLinks	0,061	0,167	0,037	0,981	0,517
BaggingClassifier	RandomOverSampler	0,205	0,265	0,167	0,979	0,580
	ADASYN	0,209	0,140	0,407	0,950	0,683
	NearMiss	0,044	0,023	0,870	0,396	0,629

	SMOTE	0,169	0,116	0,315	0,950	0,638
	AllKNN	0,086	0,188	0,056	0,981	0,526
	RandomUnderSampler	0,097	0,052	0,759	0,773	**0,766**
	TomekLinks	0,065	0,250	0,037	0,983	0,518
LinearSVC	RandomOverSampler	0,036	0,018	0,926	0,201	0,557
	ADASYN	0,036	0,018	0,926	0,205	0,560
	NearMiss	0,027	0,014	0,333	0,616	0,477
	SMOTE	0,036	0,019	0,926	0,208	0,561
	AllKNN	0,000	0,000	0,000	0,984	0,500
	RandomUnderSampler	0,040	0,021	0,352	0,730	0,544
	TomekLinks	0,000	0,000	0,000	0,984	0,500
NuSVC	RandomOverSampler	0,033	0,017	0,796	0,235	0,511
	ADASYN	0,032	0,016	0,796	0,221	0,504
	NearMiss	0,034	0,018	0,500	0,539	0,520
	SMOTE	0,032	0,016	0,796	0,222	0,505
	RandomUnderSampler	0,033	0,017	0,963	0,103	0,526

Tabela 9 Porównanie wyników różnych modeli uczenia maszynowego wraz z uwzględnieniem metod balansowania danych

Klasyfikator	**F1**	**Precision**	**Recall**	**Accuracy**	**Balanced accuracy - wybrane cechy**
BalancedBaggingClassifier	**0.332**	**0.206**	**0.840**	**0.857**	**0.849**
BalancedRandomForestClassifier	0.257	0.149	0.920	0.776	0.843
BalancedBaggingClassifier estimator=HistGradientBoostingClassifier	0.325	0.204	0,80	0.860	0.831
Z tolerancją jednej ramki					
BalancedBaggingClassifier	**0.415**	**0.265**	**0.960**	**0.886**	**0.921**
BalancedRandomForestClassifier	0.296	0.175	0.960	0.807	0.880
BalancedBaggingClassifier estimator=HistGradientBoostingClassifier	0.412	0.265	0.920	0.889	0.904

Tabela 10 Porównanie wyników 3 klasyfikatorów przystosowanych do niezbalansowanych danych dla wybranego zestawu cech

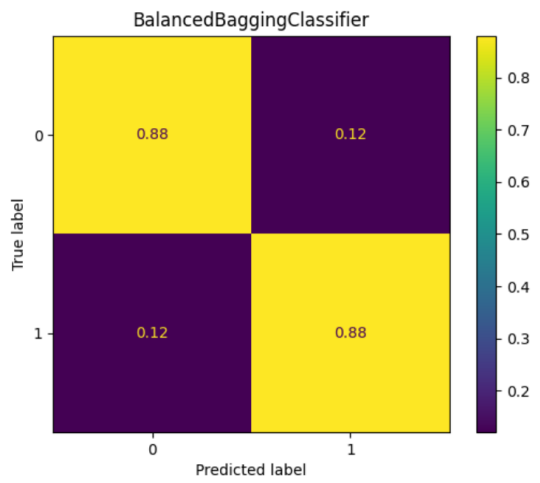
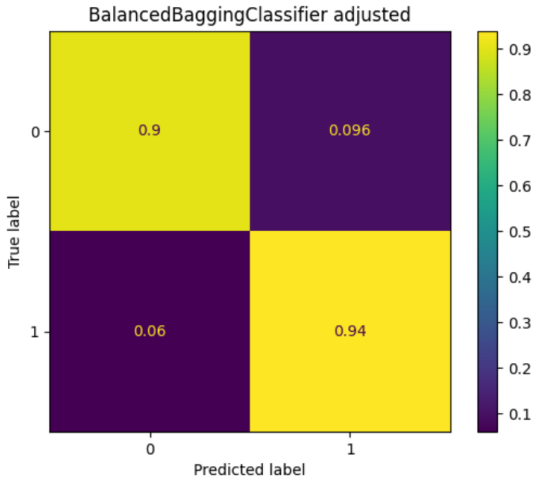
Wyznaczone hyperparametry dla modeli
BalancedBaggingClassifier(n_estimators=47, max_features=0.7547689530533495, bootstrap=True, replacement=True, sampling_strategy='not minority', random_state=0, n_jobs=-1)
BalancedRandomForestClassifier(n_estimators= 124, criterion= 'entropy', max_depth= 14, max_features= 'log2', min_samples_leaf= 1, sampling_strategy= 'all', bootstrap= False, oob_score= False, replacement= False, random_state=0, n_jobs=-1)

BalancedBaggingClassifier(estimator=HistGradientBoostingClassifier(random_state=0, class_weight='balanced', learning_rate=0.05894441800895756, max_iter=355, min_samples_leaf=10), n_estimators=249, sampling_strategy="all", random_state=0, n_jobs=-1)

Tabela 11 Wartości hyperparametrów dla dokładnej klasyfikacji momentu odbicia lotki o podłoże

Klasyfikator	F1	Precision	Recall	Accuracy	Balanced accuracy – wybrane cechy
BalancedBaggingClassifier	**0.377**	**0.240**	**0.88**	**0.877**	**0.878**
BalancedRandomForestClassifier	0.296	0.176	0.94	0.811	0.873
BalancedBaggingClassifier estimator=HistGradientBoostingClassifier	0.365	0.233	0.84	0.876	0.859
Tolerancja jednej ramki					
BalancedBaggingClassifier	0.456	0.301	0.94	0.905	0.922
BalancedRandomForestClassifier	0.336	0.204	0.96	0.840	0.897
BalancedBaggingClassifier estimator=HistGradientBoostingClassifier	**0.468**	**0.309**	**0.96**	**0.908**	**0.932**

Tabela 12 Porównanie wyników klasyfikatorów po optymalizacji parametrów dla dokładnej klasyfikacji momentu odbicia o podłoże

Macierze pomyłek – wartości hyperparametrów zoptymalizowane dla klasyfikatorów wyznaczających dokładny moment odbicia o podłoże	
Dokładny moment odbicia	Moment odbicia z dokładnością do jednej ramki
BalancedBaggingClassifier	
	

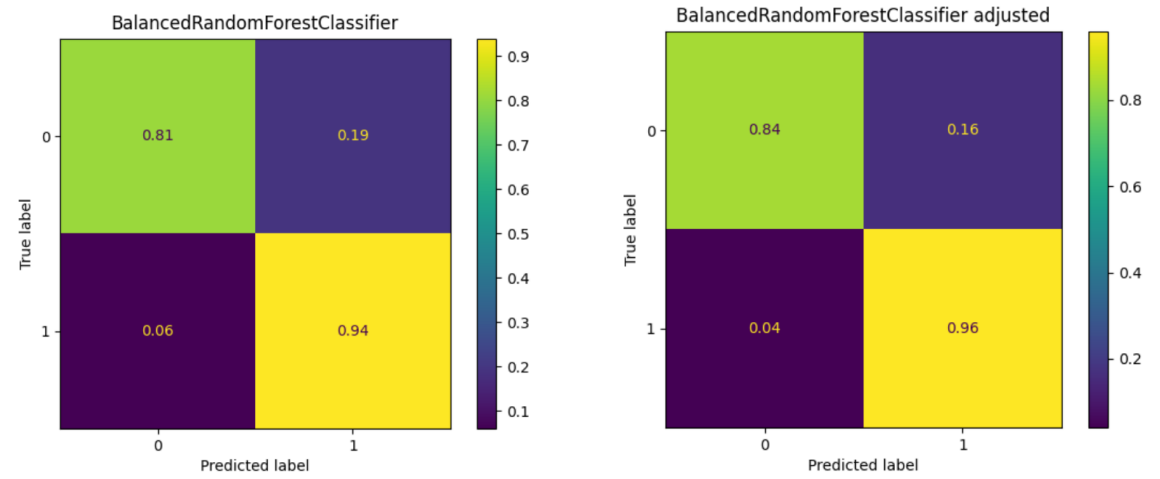
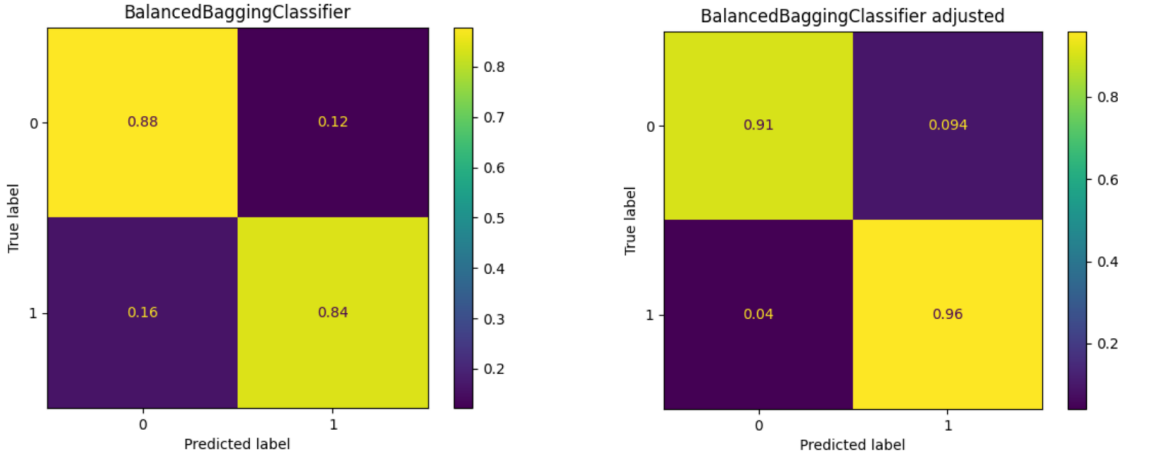
BalancedRandomForestClassifier

BalancedBaggingClassifier estimator=HistGradientBoostingClassifier


Tabela 13 Macierze pomyłek dla testowanych klasyfikatorów po optymalizacji hyperparametrów przedstawionej w tabeli []

Wyznaczone hyperparametry dla modeli
BalancedBaggingClassifier(n_estimators=156, max_features=0.8532097379055883, bootstrap=True, replacement=True, sampling_strategy='all', random_state=0, n_jobs=-1),
BalancedRandomForestClassifier(n_estimators= 19, criterion= 'entropy', max_depth= 12, max_features= None, min_samples_leaf= 1, sampling_strategy= "all", bootstrap= False, oob_score= False, replacement= True, random_state=0, n_jobs=-1)

BalancedBaggingClassifier(estimator=HistGradientBoostingClassifier(random_state=0,
class_weight='balanced',
learning_rate=0.10406405953447745,
max_iter=438,
min_samples_leaf=10),
n_estimators=198,
sampling_strategy="all",
random_state=0, n_jobs=-1)

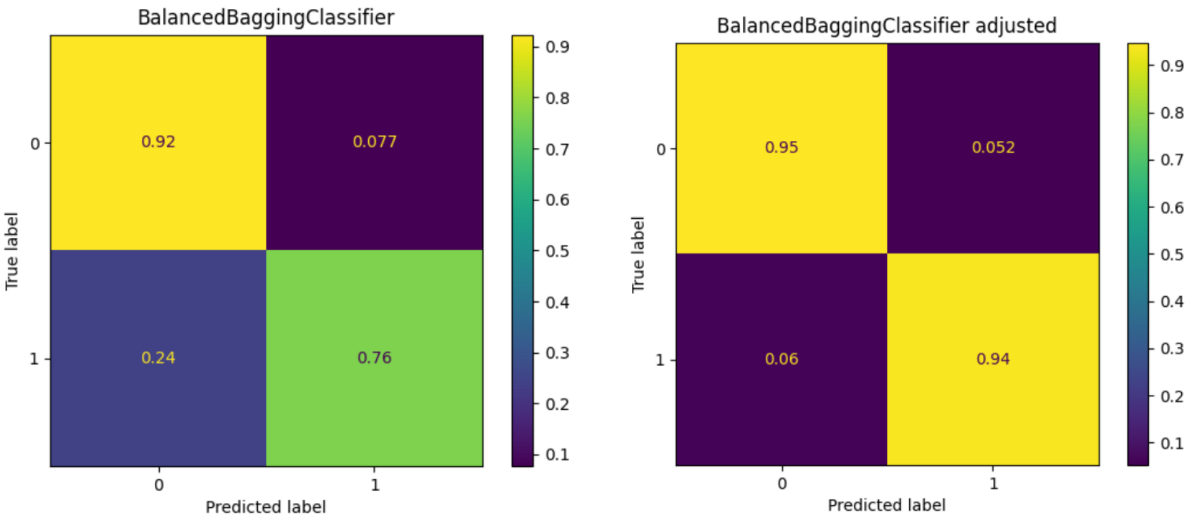
Tabela 14 Wyznaczone hyperparametry dla klasyfikatorów z tolerancją jednej ramki

Classifier	F1	Precision	Recall	Accuracy	Balanced accuracy
BalancedBaggingClassifier	0.434	0.304	0.76	0.916	0.841
BalancedRandomForestClassifier	**0.369**	**0.234**	**0.86**	**0.876**	**0.868**
BalancedBaggingClassifier estimator=HistGradientBoostingClassifier	0.362	0.232	0.82	0.878	0.850
Tolerancja jednej ramki					
BalancedRandomForestClassifier	**0.602**	**0.443**	**0.94**	**0.947**	**0.944**
BalancedBaggingClassifier	0.459	0.301	0.96	0.904	0.931
BalancedBaggingClassifier estimator=HistGradientBoostingClassifier	0.475	0.315	0.96	0.910	0.934

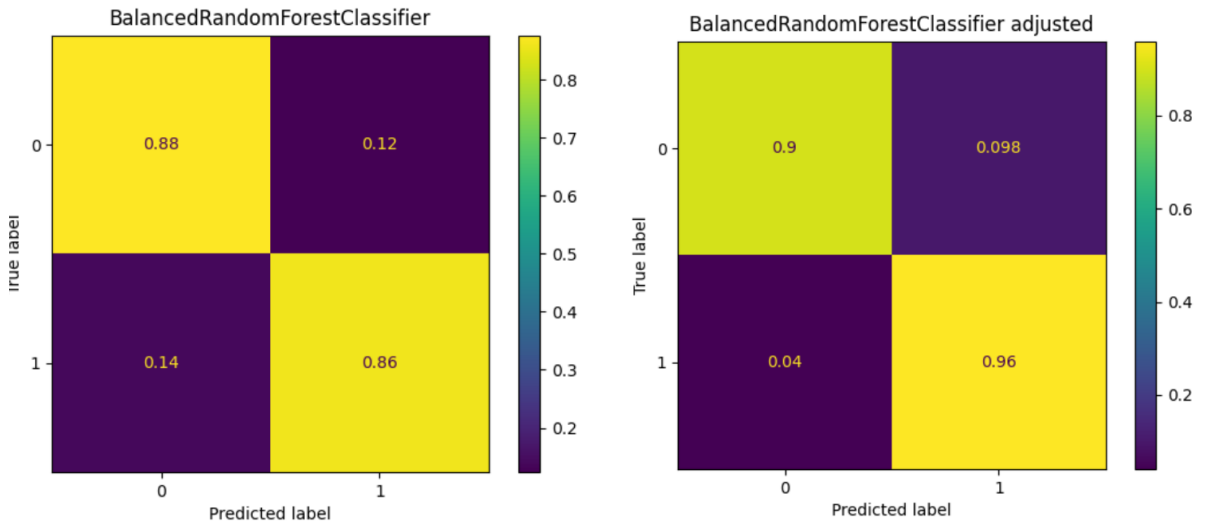
Tabela 15 Porównanie wyników klasyfikatorów po optymalizacji parametrów dla klasyfikacji momentu odbicia o podłoże z tolerancją jednej ramki

Macierze pomyłek – wartości hyperparametrów zoptymalizowane dla klasyfikatorów wyznaczających moment odbicia o podłoże z tolerancją jednej ramki

Dokładny moment odbicia	Moment odbicia z dokładnością do jednej ramki
BalancedBaggingClassifier	



BalancedRandomForestClassifier



BalancedBaggingClassifier estimator=HistGradientBoostingClassifier

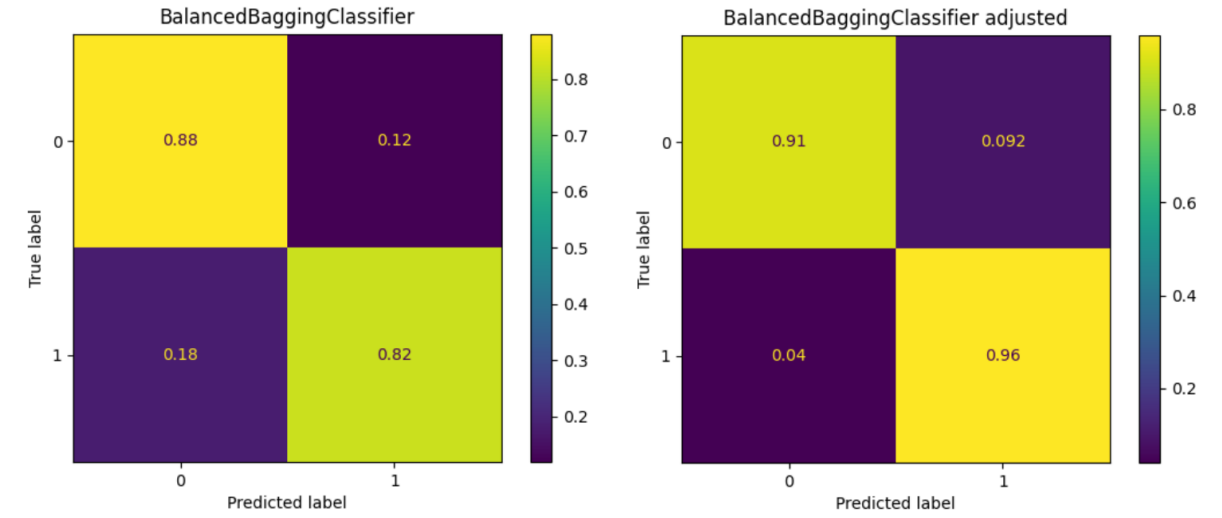


Tabela 16 Macierze pomyłek klasyfikatorów po optymalizacji hyperparametrów dla klasyfikacji momentu odbicia o podłoże z tolerancją jednej ramki

Model BalancedRandomForest uzyskał lepszy wynik od BalancedBaggingClassifier, dla klasyfikacji z tolerancją jednej ramki i ten model wykorzystano w końcowym rozwiązaniu. Jeśli chodzi o wydajność obliczeniową, to z wykorzystaniem CPU Intel i9 model BalancedRandomForest średnio na predykcję potrzebował 0.024 ms, a model BlancedBaggingClassifier 0.394 ms

Wyniki - dyskusja

Dane wykorzystane do trenowania modeli pochodzą z modułu śledzenia i rozpoznawania lotki. Moduł ten zwraca pozycję korka do badmintona z pewną dokładnością, Z tego względu porównanie pozycji korka pomiędzy dwiema sąsiadującymi ramkami wideo nie jest wystarczającym kryterium do oceny czy nastąpiło odbicie o podłoże, czy też nie. Opisany na

początku tego rozdziału *Model Podstawowy* ma niezadowalającą skuteczność z tego powodu, że na liczbę pikseli w ramce różnicowej oprócz prędkości poruszania się lotki ma wpływ również ruch wirowy lotki oraz cień, który rzuca na podłoże. Aby osiągnąć wyższą skuteczność, autor wyznaczył wiele cech, które pozwoliły na poprawienie skuteczności modelu z 67% do 88% (ballanced accuracy). Ze względu na to, że podczas anotacji momentu odbicia osoba anotująca nie zawsze była pewna czy odbicie nastąpiło w zaznaczonej przez nią ramce czy poprzedniej/następnej to zweryfikowano też skuteczność modelu z tolerancją jednej ramki. Z punktu widzenia całości rozwiązania pomyłka modelu w wyznaczeniu momentu odbicia lotki o podłoże o jedną ramkę jest istotna tylko wtedy, gdy lotka poruszała się przed odbiciem bardzo płasko i szybko oraz odbiła się bardzo blisko krawędzi linii. W pozostałych przypadkach dokładne wyznaczenie pozycji korka (opisane w dalszej części rozprawy) w stosunku do linii kortu nie będzie obarczone na tyle istotnym błędem, aby odpowiedź systemu czy lotka upadła w korcie czy poza nim była niezgodna z prawdą. Zezwalając na tolerancję jednej ramki skuteczność zaproponowanego rozwiązania wzrasta do 94% dla miary ballanced accuracy.

Autor przeanalizował przypadki, dla których model pomylił się w wyznaczeniu kluczowej ramki o więcej niż 2 ramki. Zasadniczo błędne przypadki można podzielić na dwie grupy:

1) Pomyłka bardzo duża - kilkanaście, kilkadziesiąt ramek.
2) Pomyłka niewielka – kilka ramek.

Przypadków z grupy pierwszej jest kilka. Są to sytuacje, w których model jako kluczową ramkę wyznaczył tę w której nastąpiło drugie odbicie lotki o podłoże. Jeśli chodzi o drugą grupę, to pomyłki są spowodowane cieniem rzucanym przez lotkę, który może być widoczny na ramce różnicowej [Rysunek 36] co ma wpływ na wartość cechy *diff_size_change_ratio*. W tej grupie znajdują się też specyficzne sekwencje wideo, na których zarejestrowano uderzenie lotki o parkiet po płaskim uderzeniu smecz do bocznej linii kortu. W takich przypadkach cechy *angle_change_ratio, height_change_ratio, dist_travelled_wrt_width* mają podobne wartości, jak wtedy gdy lotka leci nad kortem. Autor nie eksplorował innych rozwiązań, które pozwoliłyby na poprawę skuteczności opracowanego modelu w tym przypadku z następujących powodów:

a) problem występuje tylko w opisanym specyficznym przypadku,
b) pomyłka o kilka ramek z punktu widzenia decyzji, czy było pole, czy aut ma znaczenie tylko, gdy lotka po uderzeniu leci płasko i po przekątnej,

c) ze statystyk zebranych przez autora wynika, że opisanych powyżej sytuacji jest poniżej 6%.

Autor ma świadomość, że jest tu miejsce na poprawę i być może zastosowanie sieci neuronowych pozwoliłoby na osiągnięcie lepszych rezultatów.

Jeśli chodzi o poprawę skuteczności w przypadku, gdy model zwrócił ramkę w której nastąpiło drugie odbicie to autor proponuje następującą modyfikację mającą na celu obsługę wspomnianego scenariusza.

Jeśli w danej sekwencji model zwrócił prawdopodobieństwo odbicia o podłoże większe niż 50% dla kilku ramek, to nie jest zwracana ramka z najwyższym prawdopodobieństwem, tylko dokonywana jest korekta na tej zasadzie, że promowane są odbicia wcześniejsze i te, które były bliżej linii kortu. Niestety ze względu na brak czasu autor nie mógł dokładnie zbadać przedstawionej koncepcji.

Autor podjął natomiast próbę usuwania cienia rzucanego przez lotkę. Metoda polegała na zbadaniu jak zmniejsza się jasność pikseli w określonym obszarze zainteresowań. Niestety rezultaty jakie zostały uzyskane nie były zadowalające, a poprawa ich wymagałaby ilości pracy niewspółmiernej do przewidywanych korzyści.

3.4. Wyznaczenie miejsca odbicia się lotki o ziemię w stosunku do referencyjnych linii kortu

Aby ocenić, czy lotka upadła w polu gry, czy poza nim konieczne jest wyznaczenie na obrazie obszaru pola gry oraz miejsca upadku lotki. Autor nie zajmował się automatycznym wyznaczeniem obszaru pola gry. Referencyjne linie kortu zostały wyznaczone manualnie.

W celu wyznaczenia miejsca odbicia lotki od podłoża koniecznie jest precyzyjne wyznaczenie korka lotki. Ze względu swój kształt lotka zawsze odbija się od podłoża korkiem, więc do podjęcia decyzji pole/aut nie jest konieczna segmentacja całej lotki. Segmentacja całej lotki jest problematyczna z kilku powodów:

- Nawet w momencie odbicia lotki od podłoża lotka jest delikatnie rozmyta, ponieważ układ piór lotki powoduje, że lotka w trakcie lotu wiruje. Z tego powodu lotka nie ma ostrych krawędzi i większość popularnych metod ma problemy z precyzyjną segmentacją.
- Biały korek i białe pióra mogą zlewać się z białą linią kortu – patrz [Rysunek 39]. Warto zauważyć, że najczęściej zawodnik żąda weryfikacji decyzji sędziego liniowego,

właśnie wtedy gdy lotka upada w pobliżu linii (czyli jest widoczna na jej tle). Warto zaznaczyć, że tego problemu nie ma w tenisie gdzie linie są białe, a piłka fluorescencyjnie żółta.

Autor sprawdził jak z segmentacją lotki radzą sobie klasyczne algorytmy takie algorytmy jak grabCut [90] oraz algorytmy wykorzystujące sieci neuronowe takie jak Segment Anything Model [50].

Na rysunku [Rysunek 39] przedstawiono wynik segmentacji lotki z wykorzystaniem algorytmu grabCut.

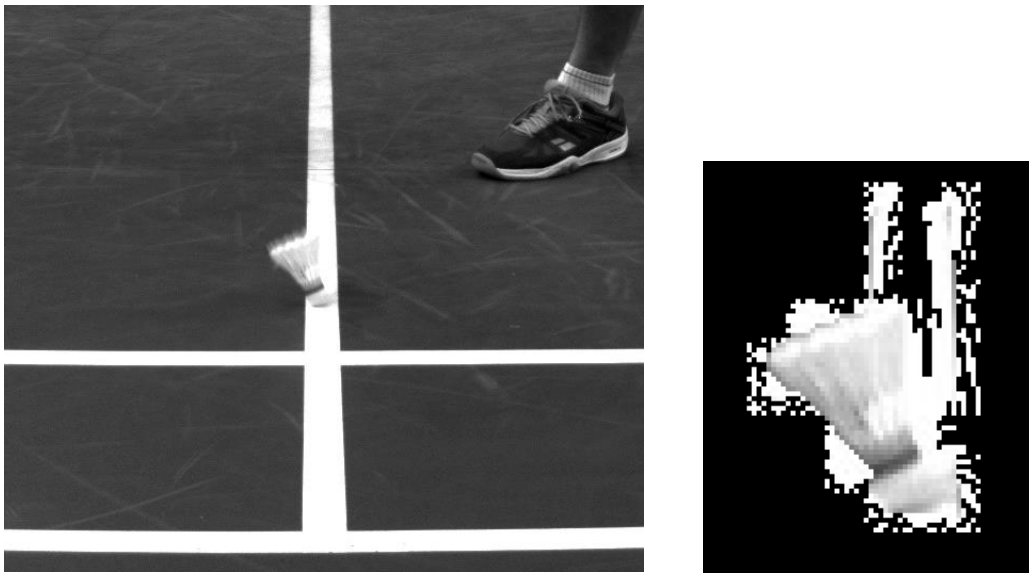
Na rysunkach [Rysunek 40, Rysunek 43] przedstawiono wynik działania Segment Anything Model – SAM, uznawanego za state-of-the-art.

Badając wyniki zwracane przez SAM dokonano dwóch rodzajów eksperymentów:

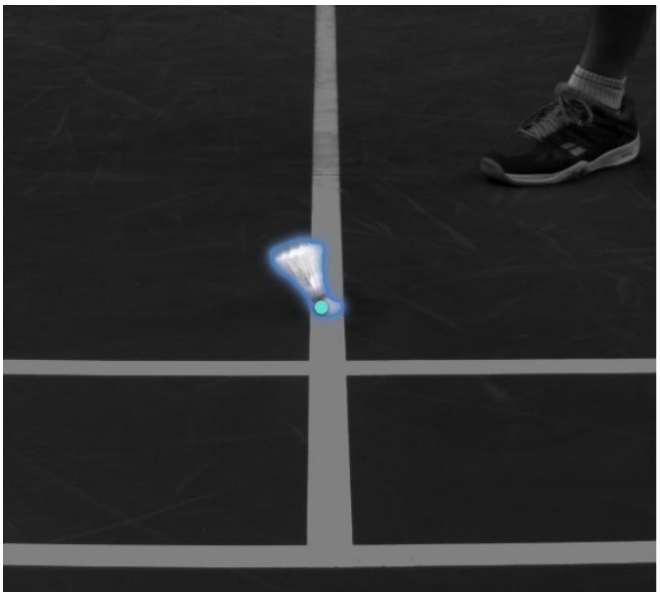
1. Algorytmowi SAM dokładnie wskazano korek od lotki (błękitna kropka) i poproszono o jej segmentacje – [Rysunek 40]. W wyniku otrzymano wysegmentowaną lotkę - [Rysunek 41].
2. Wskazując obszar (biały prostokąt), w którym należy dokonać segmentacji (lotka jest w centrum obszaru zainteresowań) – [Rysunek 42]) jako wynik segmentacji otrzymano to co jest widoczne na rysunku [Rysunek 43].

Algorytm śledzący lotkę wyznacza obszar zainteresowań, w którym jest lotka, więc podanie dokładnej pozycji korka lotki algorytmowi Segment Anything Model jest niemożliwe i w praktyce nie da się uzyskać wyniku przedstawionego na rysunku [Rysunek 41], możliwe jest uzyskanie wyniku zaprezentowanego na [Rysunek 43]. Ze względu na niezadowalające wyniki uzyskane autor zdecydował się na opracowanie własnego innowacyjnego algorytmu wyznaczania korka lotki do badmintona a szczegółowe porównanie proponowanej przez autora metody z SAM przedstawiono w rozdziale *Porównanie zaproponowanej metody z Segment Anything Model* na stronie 118.

W przypadkach, gdy lotka jest w dużej odległości od linii algorytm SAM radzi sobie dość dobrze [Rysunek 44, Rysunek 45], jednakże takie sytuacje, gdy zawodnik prosi o weryfikację zdarzają się niezmiernie rzadko (Wtedy jest to taktyczna zagrywka zawodnika, aby zyskać czas i wybić przeciwnika z rytmu meczowego).

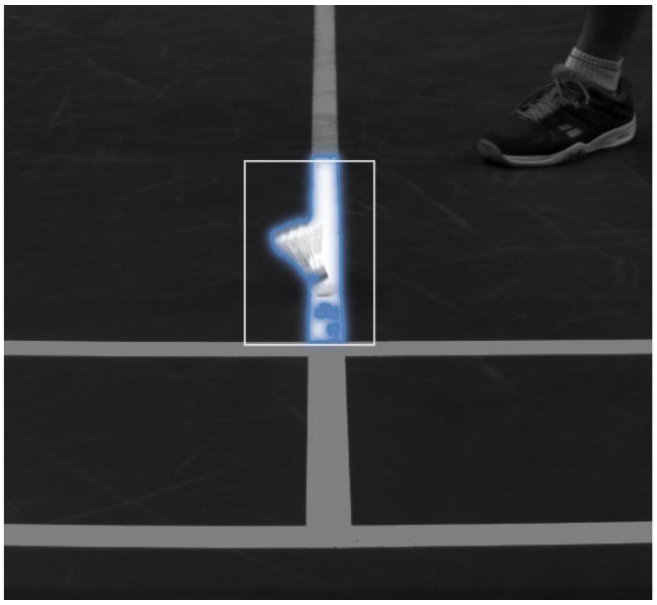


Rysunek 39 Lotka na tle białej linii i wynik segmentacji algorytmem GrabCut



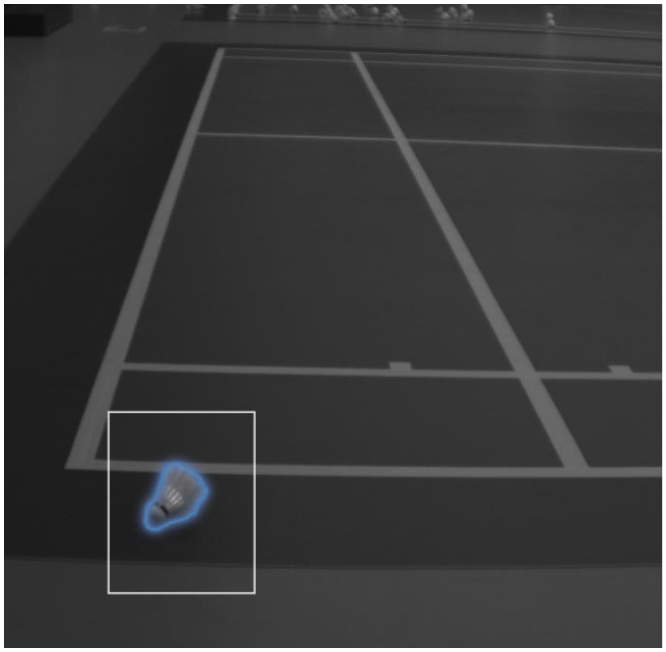
Rysunek 40 Segmentacja lotki przez algorytm Segment Anything Model poprzez precyzyjne wskazanie (błękitny punkt) segmentowanego obiektu

Rysunek 41 Wynik segmentacji lotki przez algorytm SAM



Rysunek 42 Segmentacja lotki przez Segment Anything Model poprzez wskazanie obszaru w którym segmentować obiekt

Rysunek 43 Wynik segmentacji lotki przez algorytm SAM



Rysunek 44 Segmentacja lotki będącej daleko poza kortem przez algorytm SAM poprzez wskazanie obszaru w którym segmentować obiekt

Rysunek 45 Wynik segmentacji lotki przez SAM

Algorytm wyznaczania korka lotki i miejsca odbicia o podłoże

Opracowany przez autora algorytm zakłada, że znana jest trajektoria lotki i przybliżona pozycja korka lotki oraz szerokość lotki w pikselach. Te dane dostarczane są przez algorytm śledzenia lotki opisany w rozdziale *Detekcja i śledzenie lotki*.

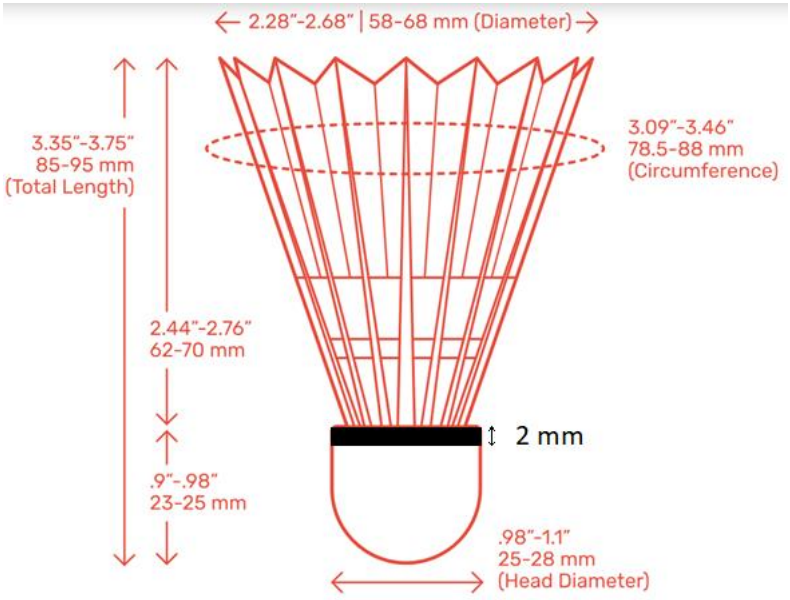
Jak już przedstawiono wcześniej, segmentacja rozmazanej białej lotki, która znajduje się na tle białej linii jest trudna, dlatego też autor zdecydował się na znalezienie ciemnego paska, który odcina biały korek od białych piór, a następnie wyliczenie okręgu odpowiadającemu korkowi. Przepisy nie wymagają, aby lotka posiadała taki pasek, ale na rynku nie ma lotek, które nie miałyby tego paska. Kolor tego paska jest zawsze ciemny – najczęściej czarny. Ten czarny pasek jest cienką taśmą, która jest konieczna ze względów produkcyjnych (poprawia trwałość lotki), pasek ten też ułatwia sędziom serwisowym ocenę czy podczas serwisu zawodnik uderzył lotkę w korek, czy w pióra (uderzenie w pióra podczas serwisu jest błędem [115]).

Ponieważ ten pasek okala korek dookoła, to ruch wirowy lotki nie ma wpływu na rozmazanie jego dolnej i górnej krawędzi. Ruch postępujący lotki powoduje jego rozmazanie, ale w momencie odbicia lotka praktycznie zatrzymuje się i rozmycie nie jest duże. Dlatego też im więcej klatek na sekundę rejestruje kamera tym lepiej. Dane, dla których przeprowadzano eksperymenty zostały pozyskane z kamer pracujących z prędkościami od 150 do 200 klatek na sekundę.

Na rysunku [Rysunek 46] przedstawiono wymiary lotki określone przepisami gry [115]. Przepisy dopuszczają pewną dowolność, ale z pomiarów przeprowadzone przez autora wynika, że wszystkie lotki turniejowe (Yonex, Victor) mają średnicę korka 26mm.

W celu wyliczenia miejsca styku lotki z podłożem przejęto że:

- kamera jest ustawiona na wprost linii prostopadle do podłoża,
- średnica korka wynosi 26mm,
- średnica lotki w najszerszym miejscu – końcówki piór wynosi 65 mm,
- wysokość korka – 25 mm,
- wysokość paska okalającego korek – 2 mm,
- średnica obszaru styku korka lotki z podłożem – ½ średnicy korka – 13 mm.



Rysunek 46 Wymiary lotki określone przez przepisy[17]

Wyznaczenie miejsca styku lotki z podłożem

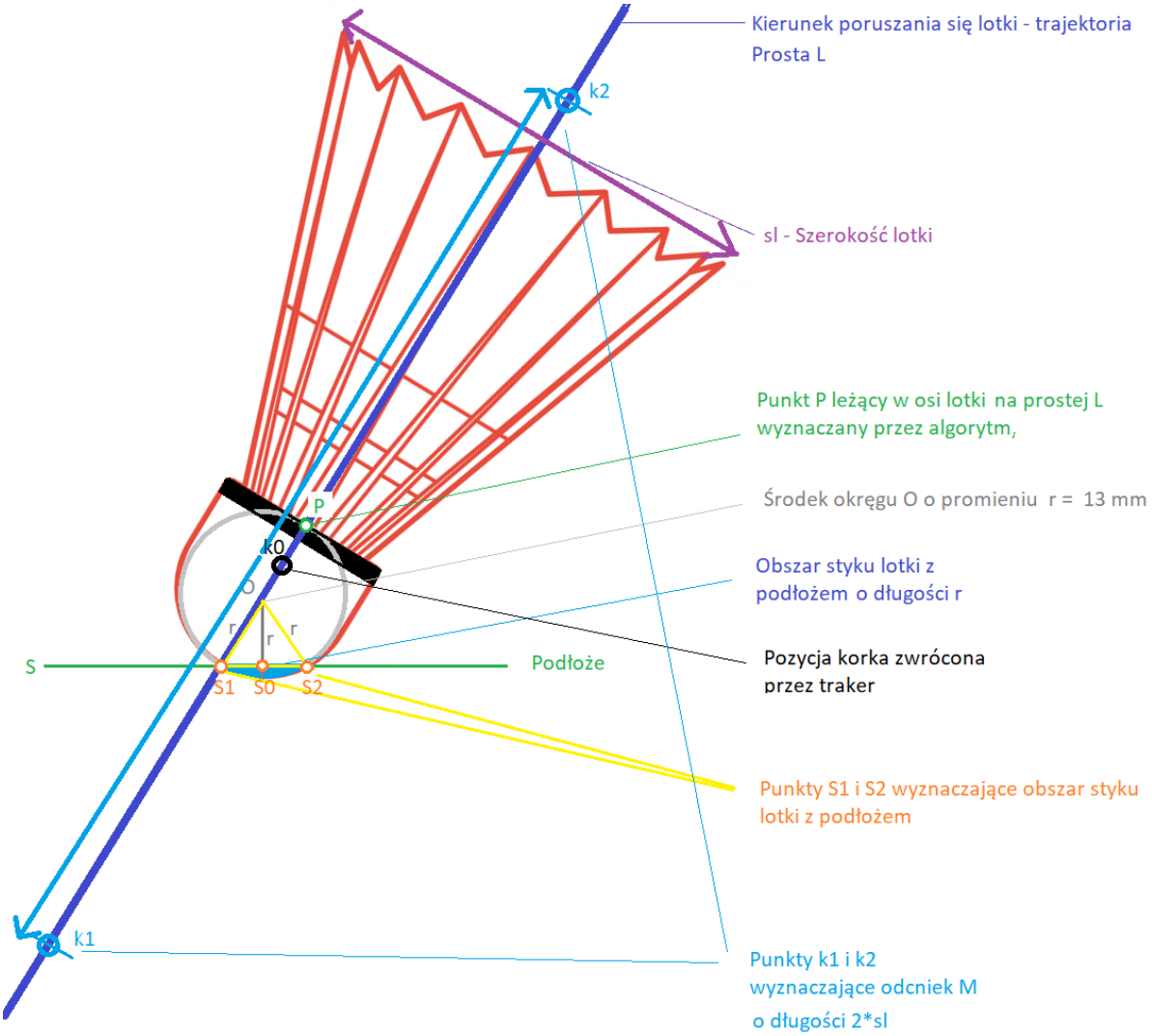
Miejsce styku lotki z podłożem zwizualizowano na rysunku [Rysunek 47]. Algorytm wyznaczający obszar styku lotki z podłożem opiera się prostych operacjach matematycznych i został opisany w punktach, poniżej.

1. Algorytm na wejściu otrzymuje równanie prostej *L*, która jest wyznaczona przez traker na postawie trajektorii lotki oraz szerokość lotki w pikselach (*sl*).
2. Niech punkt *P* oznacza punkt na prostej L, leżący na dalszej od korka krawędzi czarnego paska [Rysunek 46, Rysunek 47]. Algorytm wyznaczenia punktu *P* jest szczegółowo opisany w dalszej części - *Wyznaczenie punktu P*.
3. Niech okrąg *o* oznacza okrąg wpisany w korek, a punkt *O* środek okręgu. Punkt *O* leży na prostej *L* w odległości *r* = 13mm (połowa średnicy korka) od punktu *P*.
4. Niech linia *S* oznacza linię wyznaczającą podłoże kortu. Linia *S* jest prostopadła do prawej krawędzi obrazu.
5. Wyznaczany jest trójkąt równoboczny o wierzchołkach (*O, S1, S2*), którego podstawa jest na linii *S*.

[17] Źródło: https://www.dimensions.com/element/badminton-shuttlecock

6. Na linii prostopadłej do linii *S* i przechodzącej przez punkt *O* wyznaczany jest punkt *S0*, który leży na wysokości trójkąta równobocznego na środku jego podstawy w odległości $\frac{\sqrt{3}}{2}\ r$ od punktu *O*
7. Mając punkty *O, S0* wyznaczane są pozostałe współrzędne wierzchołków trójkąta równobocznego – *S1 i S2*.

Punkty *S1* i *S2* wyznaczają odcinek, który reprezentuje obszar styku lotki z podłożem. W zależności od obserwowanej przez kamerę linii, położenie punktu *S1* lub *S2* determinuje czy lotka upadła w polu czy poza nim. Jeżeli kamera patrzy na linię tylną, będzie to punkt *S2*, jeśli na linię boczną prawą, będzie to punkt *S1*.

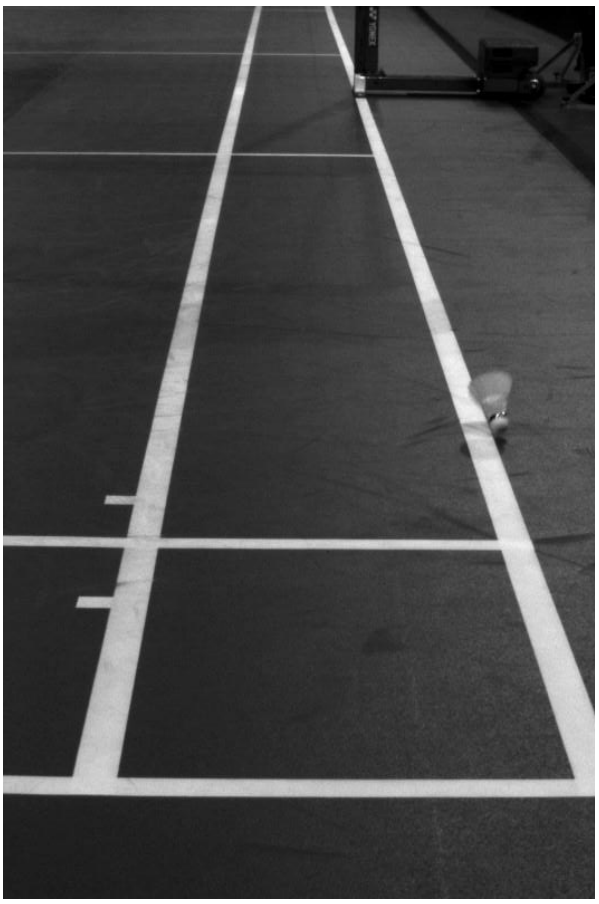


Rysunek 47 Wizualizacja metody wyznaczania obszaru styku lotki z podłożem

W zależności od obserwowanej przez kamerę linii można wyznaczyć dwie reprezentatywne grupy upadku lotki:

- Grupa 1 – lotka upada w pobliżu linii bocznej - [Rysunek 48]
- Grupa 2 – lotka upada w pobliżu linii tylnej lub serwisowej – [Rysunek 49].

Jak widać, w zależności od grupy, rzut lotki na płaszczyznę obrazu nieznacznie się różni. Najbardziej jest to widoczne w przypadku uderzenia net kill [Rysunek 30].

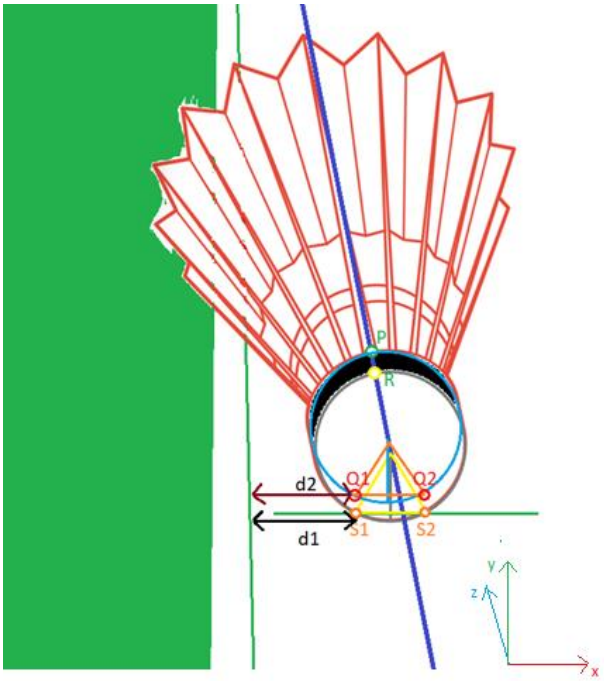


Rysunek 48 Lotka upadająca w pobliżu linii bocznej

Rysunek 49 Lotka upadająca w pobliżu linii końcowej

Na rysunku [Rysunek 47] widoczna jest sytuacja, gdy kamera patrzy na linię końcową. Wtedy lotka jest prostopadła do kamery i nie ma skrótu perspektywicznego. W przypadku kamery patrzącej na linie boczne, sytuacja jest inna. Skrót perspektywiczny powoduje, że po wyznaczeniu punktu *P* powinniśmy wyznaczyć elipsę, a nie okrąg *o*, aby wyznaczyć punkty *S1* i *S2*. Autor celowo nie wyznacza elipsy tylko okrąg, gdyż ze względu na niewielki rozmiar korka od lotki oraz dużą odległość kamery od lotki w stosunku do jej rozmiaru, wyznaczona elipsa i tak byłaby zbliżona do okręgu.

Wyznaczając okrąg, a nie elipsę punkty wyznaczające obszar styku lotki z podłożem będą przesunięte w kierunku piór. [Rysunek 50] obrazuje sytuacje w której lotka po uderzeniu smecz po prostej upada w pobliżu linii. Prawidłowe punkty wyznaczające obszar styku to *S1* i *S2*. Opisana metoda wyznaczy punkty *Q1* i *Q2*. Należy jednak zwrócić uwagę, że dla oceny czy było pole czy aut istotna jest odległość punktu *Q1* od linii - na rysunku oznaczona jako *d2* (dla prawidłowo wyznaczonego punktu *S1* odległość od linii to *d1)*. W związku z tym, że kamera jest ustawiona tak, iż patrzy na boczną linie na wprost, to odległości *d1* i *d2* są praktycznie takie same. Autor postanowił zaakceptować ten błąd, ponieważ w porównaniu z wielkością błędu jaki może powodować niedokładne wyznaczenie punktu *P* błąd spowodowany opisanym uproszczeniem jest na tyle mały, iż obiektywnie akceptowalny. Najważniejsze jest jak najdokładniejsze wyznaczenie punktu *P*.



Rysunek 50 Wyznaczenie obszaru styku lotki z podłożem dla kamery patrzącej na linie boczną

Wyznaczenie punktu P

W metodzie zaproponowanej przez autora, do wyznaczenia punktu *P*, wymagana jest znajomości trajektorii lotki oraz szerokość lotki w pikselach (jest ona różna w zależności od odległości lotki od kamery). Przyjęto, że trajektoria lotki przy podłożu jest linią prostą. Linia, która wyznacza trajektorię oraz szerokość lotki w pikselach (cecha *width* [Tabela 3]) wyliczana jest przez moduł śledzący korek lotki opisany w rozdziale *Opis metody zastosowanej do detekcji lotki*. Trajektoria lotki może zmienić się w momencie odbicia, dlatego też do wyznaczenia linii *L* przyjmowana jest trajektoria, którą lotka miała, przed odbiciem. Do

wyznaczenia linii *L* wykorzystywane są wszystkie punkty trajektorii przed odbiciem zwracane przez traker. Aby dopasować linię do punktów trajektorii autor zastosował metodę RANSAC[18] [114].

Błędne wyznaczenie momentu odbicia ma wpływ na wyznaczenie prostej *L*, a co za tym idzie i wyznaczenie punktu *P*. Zastosowany RANSAC w pewien sposób koryguje wpływ błędnego wyznaczenia momentu odbicia na dopasowanie punktów trajektorii do prostej, ale w momencie, gdy nie mamy wielu punktów trajektorii i oprócz punktów przed odbiciem weźmiemy też punkty po odbiciu to nasza linia *L* będzie wyznaczona niedokładnie. Największy problem występuje wtedy, gdy algorytm znajdujący klatkę, w której nastąpiło odbicie o podłoże pomyli się i znajdzie nie pierwsze, a drugie odbicie.

Punkt *P* wyznaczany jest jako współrzędne piksela obrazu odpowiadającemu miejscu, gdzie rozpoczyna się czarny pasek oddzielający pióra od korka lotki. Na linii *L* wyznaczany jest odcinek *M*, a następnie w celu wyznaczenia punktu *P* sprawdzany jest każdy piksel leżący na odcinku *M*. Odcinek *M* ograniczony jest przez punkty *k1* i *k2*, które znajdują się w odległości *sl* każdy od punktu *k0* (wyznaczona przez traker pozycja korka), a *sl* jest szerokością lotki - [Rysunek 47].

Dla każdego piksela leżącego na odcinku *M* wyznaczany jest wektor cech opisany w tabeli [Tabela 17], który przekazywany jest jako wejście do wytrenowanego modelu uczenia maszynowego. Model zwraca prawdopodobieństwo, z jakim wektor wejściowy opisuje piksel będący miejscem rozpoczęcia czarnego paska.

Wyznaczenie wektora cech

W poniższej tabeli przedstawiono sposób wyznaczenia cech dla pojedynczego piksela leżącego na odcinku M. Przeliczenie pikseli na milimetry jest dokonywane zgodnie ze wzorem:

$$D_{mm} = \frac{D_{px} * 65.0}{shuttle\_width} \qquad (1)$$

gdzie:

D_{px} – Przeliczana wartość w pikselach (np. dystans do linii)

65.0 – Stała będąca szerokością lotki w milimetrach określona przepisami

[18] Skrócony opis algorytmu RANSAC znajduje się w *Słowniczek pojęć*, punkt 13 na stronie 96

$shuttle\_width$ – Szerokość lotki w pikselach wyznaczona przez traker dla analizowanej ramki (cecha *width* [Tabela 3]).

Nazwa	**Opis**
gradient	Wielkość gradientu policzona jako $$G(n) = IL\big(M_{xy}(n+1)\big) - IL(M_{xy}(n-1))$$ gdzie: M_{xy} – uporządkowany zbiór współrzędnych obrazu *(x,y)* wyznaczonych na odcinku M ułożony w kolejności zgodnie z kierunkiem poruszania się lotki *n* – indeks punktu na w zbiorze M_{xy} *IL* – intensywność jasności piksela na analizowany obrazie w punkcie *(x,y)* w zakresie [0,255]
gradient_0	Wielkość gradientu policzona jako $$G_0(n) = IL\big(M_{xy}(n)\big) - IL(M_{xy}(n-1))$$ gdzie: M_{xy} – uporządkowany zbiór współrzędnych obrazu *(x,y)* wyznaczonych na odcinku M ułożony w kolejności zgodnie z kierunkiem poruszania się lotki *n* – indeks punktu na w zbiorze M_{xy} *IL* – intensywność jasności piksela na analizowany obrazie w punkcie *(x,y)* w zakresie [0,255]
gradient_norm	Wielkość gradientu G_{norm} wyznaczona jak w przypadku *gradient* (*G*) z tą różnicą, że dla każdego analizowanego obrazu jasność piksela *IL* została znormalizowana do [0,1] - IL_{norm}. Normalizacja min-max przeprowadzona dla pikseli ze zbioru M_{xy} $$G_{norm}(n) = IL_{norm}\big(M_{xy}(n+1)\big) - IL_{norm}(M_{xy}(n-1))$$
gradient_0_norm	Wielkość gradientu $G_{0\ norm}$wyznaczona jak w przypadku *gradient_0* (G_0) z tą różnicą, że jasność piksela *IL* została znormalizowana do [0,1] $$G_{0\ norm}(n) = IL_{norm}\big(M_{xy}(n)\big) - IL_{norm}(M_{xy}(n-1))$$

next_local_max_value	Wartość w lokalnym maksimum znajdującym się w zbiorze M_{xy} za analizowanym punktem obliczona jako: $$LMax(n) = abs(G(m+1)) + abs(G(m-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} m – indeks punktu, w którym jest lokalne maksimum m > n Lokalne minima i maksima wyznaczane są na podstawie gradientu *G* – znajdują się w tych punktach, gdzie gradient zmienia znak.
next_local_max_norm_value	Wartość w lokalnym maksimum znajdującym się w zbiorze M_{xy} za analizowanym punktem obliczona jako: $$LMax(n) = abs(G_{norm}(m+1)) + abs(G_{norm}(m-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} m – indeks punktu, w którym jest lokalne maksimum m > n
prev_local_min_value	Wartość w lokalnym minimum znajdującym się w zbiorze M_{xy} przed analizowanym punktem obliczona jako: $$LMin(n) = abs(G(m+1)) + abs(G(m-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} m – indeks punktu, w którym jest lokalne minimum m < n
prev_local_min_norm_value	Wartość w lokalnym minimum znajdującym się w zbiorze M_{xy} przed analizowanym punktem obliczona jako: $$LMin(n) = abs(G_{norm}(m+1)) + abs(G_{norm}(m-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} m – indeks punktu, w którym jest lokalne minimum m < n
next_local_max_luminosity	Wartość znormalizowanej intensywności jasności piksela IL_{norm} w pierwszym lokalnym maksimum znajdującym się w zbiorze M_{xy} za analizowanym punktem.

prev_local_min_luminosity	Wartość znormalizowanej intensywności jasności piksela w pierwszym lokalnym minimum znajdującym się w zbiorze M_{xy} przed analizowanym punktem.
dist_to_next_max	Dystans w pikselach pomiędzy analizowanym punktem, a punktem reprezentującym lokalne maksimum znajdującym się w zbiorze M_{xy} za analizowanym punktem
dist_to_next_max_mm	Dystans w milimetrach pomiędzy analizowanym punktem, a punktem reprezentującym lokalne maksimum znajdującym się w zbiorze M_{xy} za analizowanym punktem
dist_to_prev_min	Dystans w pikselach pomiędzy analizowanym punktem, a punktem reprezentującym lokalne minimum znajdującym się w zbiorze M_{xy} przed analizowanym punktem
dist_to_prev_min_mm	Dystans w milimetrach pomiędzy analizowanym punktem, a punktem reprezentującym lokalne minimum znajdującym się w zbiorze M_{xy} przed analizowanym punktem
local_min	Jeśli analizowany punkt znajduje się w lokalnym minimum to wartość ta jest obliczona jako: $$LMin(n) = abs(G(n+1)) + abs(G(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero
local_min_0	Jeśli analizowany punkt znajduje się w lokalnym minimum to wartość ta jest obliczona jako: $$LMin(n) = abs(G_0(n+1)) + abs(G_0(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero

local_min_norm	Jeśli analizowany punkt znajduje się w lokalnym minimum to wartość ta jest obliczona jako: $$LMin(n) = abs(G_{norm}(n+1)) + abs(G_{norm}(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero
local_min_0_norm	Jeśli analizowany punkt znajduje się w lokalnym minimum to wartość ta jest obliczona jako: $$LMin(n) = abs(G_{0\ norm}(n+1)) + abs(G_{0\ norm}(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero
local_max	Jeśli analizowany punkt znajduje się w lokalnym maksimum to wartość ta jest obliczona jako: $$LMax(n) = abs(G(n+1)) + abs(G(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero
local_max_0	Jeśli analizowany punkt znajduje się w lokalnym maksimum to wartość ta jest obliczona jako: $$LMax(n) = abs(G_0(n+1)) + abs(G_0(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero

local_max_norm	Jeśli analizowany punkt znajduje się w lokalnym maksimum to wartość ta jest obliczona jako: $$LMax(n) = abs(G_{norm}(n+1)) + abs(G_{norm}(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero
local_max_0_norm	Jeśli analizowany punkt znajduje się w lokalnym maksimum to wartość ta jest obliczona jako: $$LMax(n) = abs(G_{0\ norm}(n+1)) + abs(G_{0\ norm}(n-1))$$ gdzie: n – indeks analizowanego punktu w zbiorze M_{xy} W przeciwnym razie zero
p_avg	Średnia jasność pikseli w zbiorze M_{xy}
p_stdev	Odchylenie standardowe jasności pikseli w zbiorze M_{xy}
p_max	Maksymalna jasność pikseli w zbiorze M_{xy}
p_min	Minimalna jasność pikseli w zbiorze M_{xy}
luminosity	Jasność piksela w analizowanym punkcie $$I(n) = IL\big(M_{xy}(n)\big)$$ gdzie: M_{xy} – zbiór współrzędnych obrazu *(x,y)* wyznaczonych na odcinku M ułożony w kolejności zgodnie z kierunkiem poruszania się lotki *n* – indeks punktu na w zbiorze M_{xy} *IL* – intensywność jasności piksela na analizowany obrazie w punkcie *(x,y)* w zakresie [0,255]
luminosity_norm	Znormalizowana jasność piksela w analizowanym punkcie $$I_{norm}(n) = IL_{norm}\big(M_{xy}(n)\big)$$ gdzie: M_{xy} – zbiór współrzędnych obrazu *(x,y)* wyznaczonych na odcinku M ułożony w kolejności zgodnie z kierunkiem poruszania się lotki *n* – indeks punktu na w zbiorze M_{xy} IL_{norm} – intensywność jasności piksela na analizowany obrazie w punkcie *(x,y)* w zakresie [0,1]

Aby obliczyć tę cechę w pierwszej kolejności wyznaczana jest średnia wartość jasności z 10% najjaśniejszych (*bright_avg)* i 5% najciemniejszych (*dark_avg)* pikseli ze zbioru M_{xy}.

Następnie wyznaczany jest model jasności pikseli w okolicy punktu *P*. Model przedstawiono na rysunku [Rysunek 51].

Jest to kernel w kształcie krzyża o środku w punkcie *S*. Piksele znajdujące się na poziomym ramieniu mają jasność *dark_avg* a na pionowym – *bright_avg*.

Następnie dla każdego punktu ze zbioru M_{xy} obliczamy błąd dopasowania do modelu. Jest to zsumowana absolutna różnica jasności pomiędzy pikselami kernela, a pikselami obrazu podzielona przez liczbę punktów kernela. Opisuje to wzór:

$$Diff = \frac{\sum_{k=0}^{l}\left(abs\left(K(k) - IL\left(C_{xy}(k)\right)\right)\right)}{l}$$

gdzie:

K – Kernel

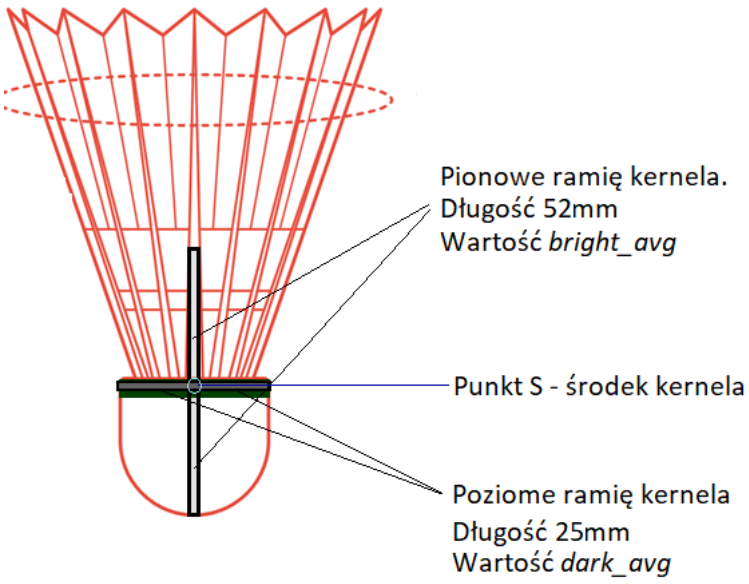
abs_diff_sum_norm

k – k-ty piksel kernela

l – Liczba pikseli w kernelu

C_{xy} – Współrzędne punktu analizowanego obrazu dla k-tego punktu kernela

IL – Jasność piksela w punkcie C_{xy}



Rysunek 51 Sposób wyznaczenia kernela

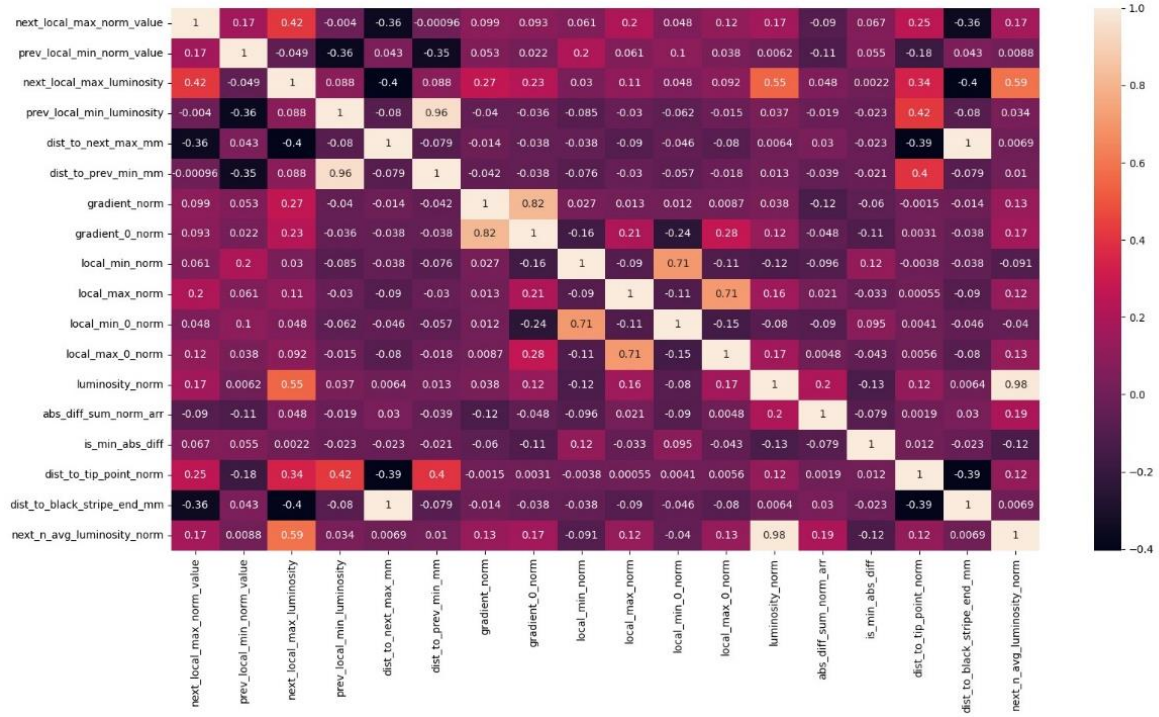
is_min_abs_diff	Wartość 1 dla tych elementów ze zbioru M_{xy}, dla których cecha abs_diff_norm ma najniższą wartość.
abs_diff_sum_norm_arr_min	Minimalna wartość cechy abs_diff_norm dla analizowanej ramki .
abs_diff_sum_norm_arr_max	Maksymalna wartość cechy abs_diff_norm dla analizowanej ramki.
abs_diff_sum_norm_arr_mean	Średnia wartość cechy abs_diff_norm dla analizowanej ramki.
abs_diff_sum_norm_arr_std	Odchylenie standardowe cechy abs_diff_norm dla analizowanej ramki.
dist_to_tip_point_norm	Dystans w milimetrach od analizowanego punktu ze zbioru M_{xy} do korka wyznaczonego przez traker.
shuttle_width	Szerokość lotki w pikselach wyznaczona przez traker (cecha *width* [Tabela 3]).
next_n_avg_luminosity	Średnia jasność następnych *k* pikseli. Gdzie *k* oznacza liczbę pikseli zawartych w szerokości czarnego paska (obliczona jako 1/13 szerokości lotki w pikselach zwróconej przez traker) $$I_{avg}(n) = \frac{\sum_{i=0}^{i=k} I(i)}{k}$$ gdzie: *n* – indeks analizowanego punktu w zbiorze M_{xy} *k – k = (int)shuttle_width/13* *k > n*
dist_to_black_stripe_end	Jak odległe w pikselach jest następne lokalne maksimum od przewidywanego końca czarnego paska (wyliczonego na podstawie szerokości paska). Wartość zero oznacza, że następne lokalne maksimum powinno być właśnie w odległości szerokości paska.
dist_to_black_stripe_end_mm	Jak wyżej, tylko w milimetrach

Tabela 17 Sposób wyznaczenia wektora cech

Spośród wyznaczonych cech odrzucono te które są silnie ze sobą skorelowane i nie poprawiają predykcji modelu. Ostatecznie wybrano 18 cech przedstawionych poniżej:

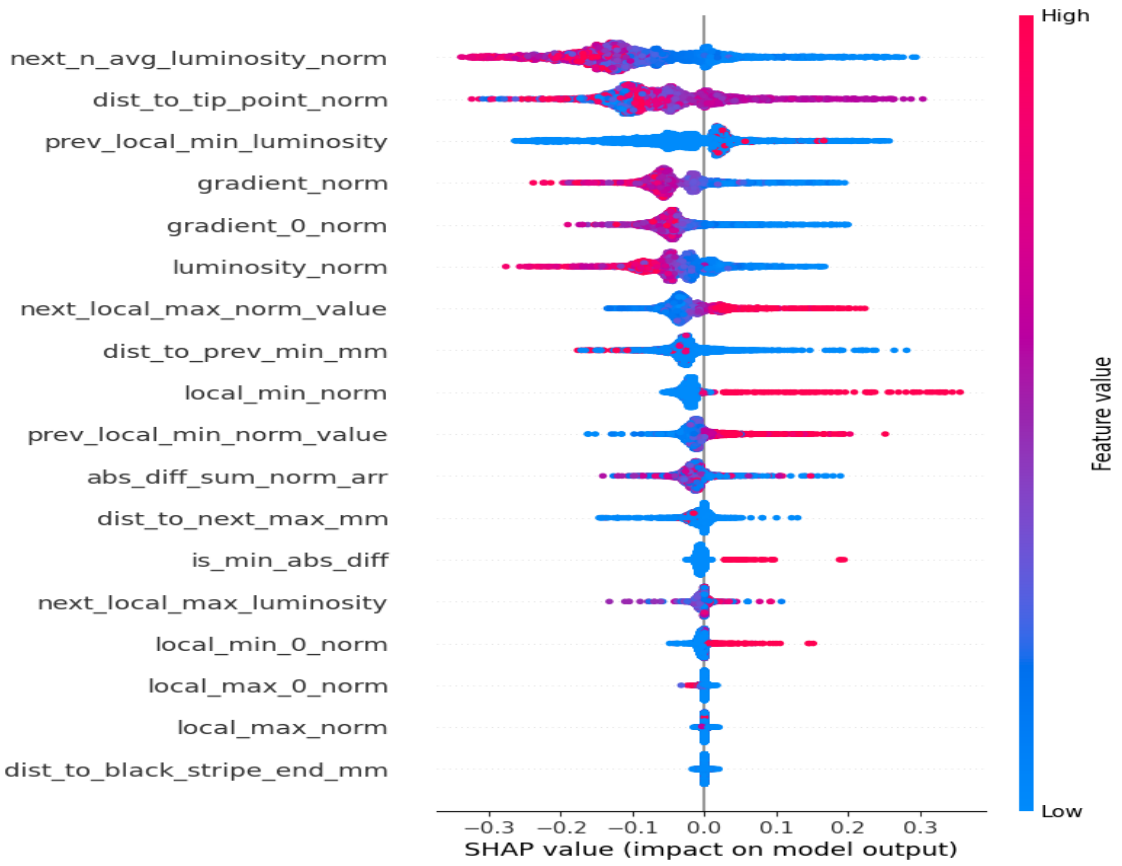
1. next_local_max_norm_value
2. prev_local_min_norm_value
3. next_local_max_luminosity
4. prev_local_min_luminosity
5. dist_to_next_max_mm
6. dist_to_prev_min_mm
7. gradient_norm
8. gradient_norm_0
9. local_min_norm
10. local_max_norm
11. local_min_0_norm
12. local_max_0_norm
13. luminosity_norm
14. abs_diff_sum_norm
15. is_min_abs_diff
16. dist_to_tip_point_norm
17. dist_to_black_stripe_end_mm
18. next_n_avg_luminosity_norm

Współczynniki korelacji liniowej Pearsona dla wybranych cech zaprezentowano na rysunku [Rysunek 52]



Rysunek 52 Współczynniki korelacji liniowej Pearsona dla wybranych cech

Autor zbadał również, które cechy są mają największy wpływ na predykcje modelu. Cechy wraz z wartościami SHAP[19] (SHapley Additive exPlanations) [115] zaprezentowano na rysunku [Rysunek 53]. Cechy są ułożone w kolejności od najistotniejszej do tej mającej najmniejszy wpływ na predykcję.

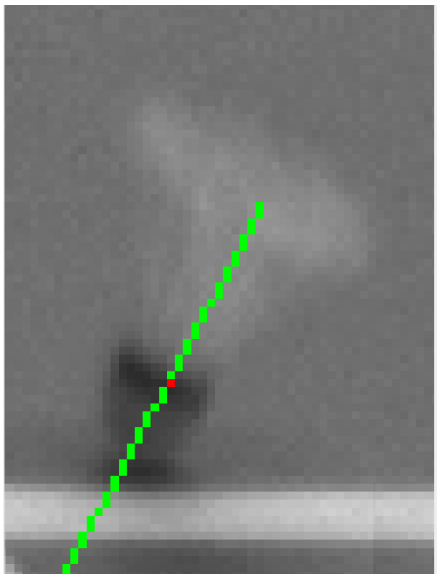


Rysunek 53 Wartości SHAP values dla cech modelu

Trening modelu, wyniki i dyskusja

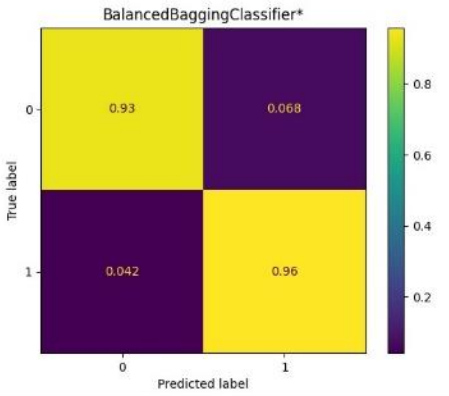
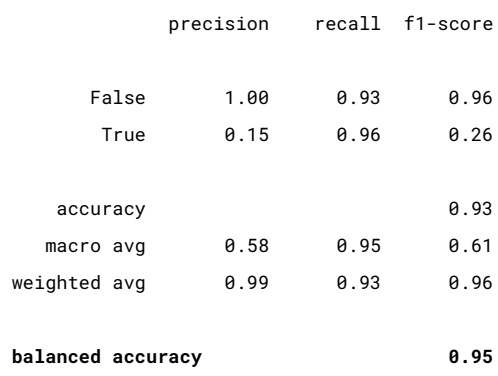
Zbiór filmów, na których zarejestrowano odbicie o ziemię został podzielony na zbiór treningowy, walidacyjny i testowy zgodnie z procedurą opisaną w rozdziale *3.1 Gromadzenie danych do eksperymentów*. Z każdego filmu wybrano ramkę, w której nastąpiło odbicie o ziemię. Dla tej ramki, dla każdego piksela znajdującego się na odcinku *M* wyznaczono wektor cech opisany powyżej. Autor na każdym takim obrazie zaznaczył na odcinku *M* piksel *P*, który jest w miejscu rozpoczęcia się czarnego paska – [Rysunek 54].

[19] Skrócony opis techniki SHAP znajduje się w *Słowniczek pojęć*, punkt 14 na stronie 96



Rysunek 54 Anotacja. Zaznaczenie na linii obrazującej trajektorię lotki – L (kolor zielony), piksela gdzie zaczyna się czarny pasek (kolor czerwony)

Podobnie jak w przypadku trenowania modelu mówiącego o tym, czy w danej ramce nastąpiło odbicie o ziemię, autor przetestował kilka modeli i wybrał ten dający najlepsze rezultaty. Przetestowane modele i wyniki przedstawiono w tabeli [Tabela 18]. Zbiór danych jest silnie niezbalansowany – negatywnych (inny piksel niż *P*) obserwacji jest 98.76% a pozytywnych (piksel *P*) 1.24%, dlatego do oceny, który model jest najlepszy autor wybrał metrykę ballanced accuracy.

Model	Macierz pomyłek	Wyniki
Balanced Baggigng Classifier		

	precision	recall	f1-score
False	1.00	0.93	0.96
True	0.15	0.96	0.26
accuracy			0.93
macro avg	0.58	0.95	0.61
weighted avg	0.99	0.93	0.96
balanced accuracy			**0.95**

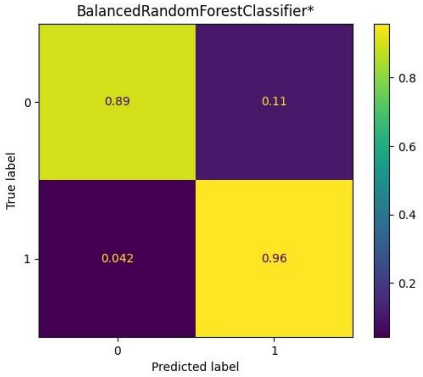
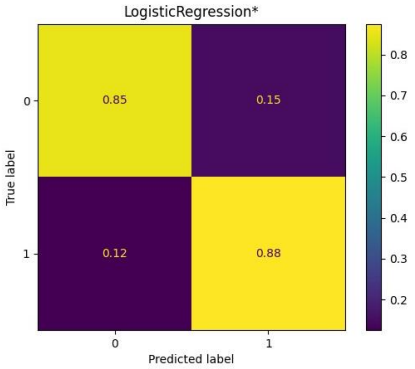
Balanced Random Forest Classifier		precision recall f1-score False 1.00 0.89 0.94 True 0.10 0.96 0.19 accuracy 0.89 macro avg 0.55 0.93 0.56 weighted avg 0.99 0.89 0.93 **balanced accuracy 0.93**
Logistic regression		precision recall f1-score False 1.00 0.85 0.92 True 0.07 0.88 0.13 accuracy 0.85 macro avg 0.53 0.86 0.52 weighted avg 0.99 0.85 0.91 **balanced accuracy 0.86**

Tabela 18 Przetestowane modele i ich skuteczność
macro avg – średnia arytmetyczna podanych miar dla obu klas
weighted avg – średnia ważona miar

Autor zoptymalizował również hyperparametry dla przedstawionych modeli (implementacja z biblioteki scikit-learn). W tabeli [Tabela 19] przedstawiono wartości tychże parametrów dla każdego z modeli:

Model	**Hyperparametry**
BalancedBaggingClassifier	estimator=HistGradientBoostingClassifier(class_weight='balanced', learning_rate=0.08158911576437505, max_iter=381, min_samples_leaf=13), n_estimators=194, sampling_strategy="not minority"
BalancedRandomForestClassifier	n_estimators = 114, criterion = 'gini', max_depth = 14, max_features = 'log2', min_samples_leaf = 2, sampling_strategy = 'majority', bootstrap = False, oob_score = False, replacement = False

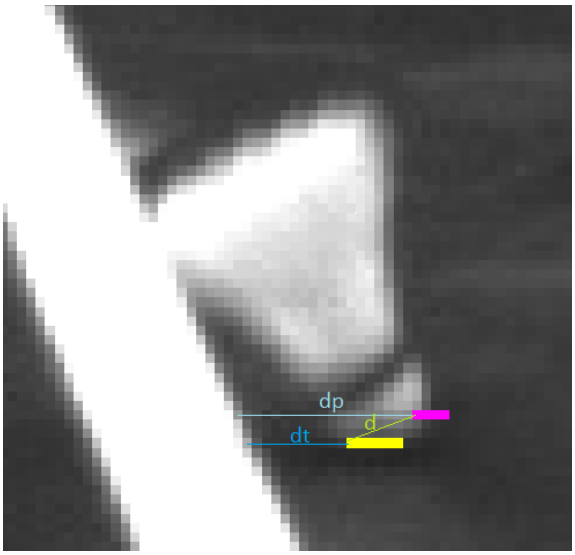
LogisticRegression	class_weight='balanced', solver='saga', penalty = 'elasticnet', tol = 5.343043561755692e-05, C = 0.813014219912614, fit_intercept = True, intercept_scaling = 0.5433414328659414, max_iter = 1160, warm_start = False, l1_ratio = 0.9040450235295895

Tabela 19 Hyperparametry testowanch modeli

Z punktu widzenia sędziego najistotniejsza jest odpowiedź na pytanie czy opracowane rozwiązanie poprawnie oceniło to czy lotka odbiła się w korcie, czy poza nim. Mniej istotne jest to, czy system myli się o 5 czy 10 milimetrów. Aby odpowiedzieć na to pytanie autor wykonał dalsze eksperymenty.

Ze zbioru walidacyjnego, dla każdego nagrania z osobna wybrano ramkę, w której nastąpiło odbicie lotki o podłoże. Dla tej ramki autor użył wytrenowanego modelu i dla każdego piksela na linii *M* wyliczył prawdopodobieństwo, że analizowany piksel jest szukanym pikselem *P* (wyznaczającym rozpoczęcie się czarnego paska okalającego korek lotki). Dla piksela o najwyższym prawdopodobieństwie wyznaczono miejsce odbicia lotki o podłoże zgodnie z metodą opisaną w rozdziale *Algorytm wyznaczania korka lotki i miejsca odbicia o podłoże*, a następnie obliczono odległość przewidywanego miejsca upadku od zaznaczonego przez operatora prawdziwego miejsca upadku - *d*, a także odległość prawdziwego miejsca upadku od linii – *dt* i odległość przewidywanego miejsca upadku lotki od linii – *dp*. Odległości te zobrazowano na rysunku [Rysunek 55] – w przypadku idealnej predykcji wartości *d* oraz błąd *err=abs(dp-dt)* wynoszą zero. Odległość *dp* jest liczona jako dystans od linii do jednego z punktów S1 lub S2, w zależności od tego, który jest bliższy linii kortu [Rysunek 47, Rysunek 50]. Zweryfikowano też, czy przewidywane przez model miejsce upadku jest po tej samej stronie linii co prawdziwe miejsce upadku (czyli czy decyzja systemu IN/OUT zgadza się z prawdą). Błąd w pikselach wyznaczenia odległości przewidywanego miejsca upadku lotki w stosunku do prawdziwego przeliczono na milimetry zgodnie ze wzorem (1) ze strony 102.

Miejsce upadku lotki (ground-truth) zostało zaznaczone przez autora na każdej wspomnianej ramce, a następnie zweryfikowane przez kwalifikowanego sędziego.



Rysunek 55 Miejsce upadku lotki wyznaczone przez algorytm (kolor purpurowy) i ground-truth (kolor żółty)

W tabeli [Tabela 20] przedstawiono wyniki dla 3 testowanych modeli. Różnica w błędzie pomiędzy przewidywanym miejscem upadku, a błędem w wyznaczeniu odległości od linii do miejsca upadku lotki wynika z tego, że gdy model wyznaczył miejsce upadku dokładnie przesunięte wzdłuż linii, to błąd odległości od linii jest niezmienny. Takie sytuacje zdarzają się dla kamer obserwujących linie boczne.

Model	**Średni błąd wyznaczenia odległości miejsca upadku od linii – *avg(abs(dp – dt))***	**Średnia euklidesowa odległość przewidywanego od prawdziwego miejsca upadku lotki - *d***	**Zła decyzja IN/OUT**
BalancedBaggingClassifier	6.91 mm	9.41 mm	2.08%
BalancedRandomForestClassifier	7.54 mm	10.98 mm	6.25%
LogisticRegression	9.20 mm	13.56 mm	6.25%

Tabela 20 Skuteczność decyzji IN/OUT dla analizowanych modeli

Jak widać proponowane rozwiązanie jest skuteczne w 98%. Podczas mistrzostw świata seniorów – Katowice 2019, autor zweryfikował skuteczność decyzji sędziów liniowych. Na 211 przypadków, w których zawodnik nie zgadzał się z decyzją sędziego liniowego i poprosił o wideo weryfikacje, w 51 przypadkach sędzia się mylił [Tabela 21]. Wynika z tego, że sędziowie liniowi mylą się w 24.2% przypadków. Primo L. i inni [115] zebrali podobne statystyki podczas olimpiady w Rio de Janeiro, 2016 oraz mistrzostw świata w Glasgow, 2017 - sędziowie pomylili się w 11 z 56 przypadków (19.6%). Różnica pomiędzy statystykami zebranymi przez autora, a tymi podanymi w [115] może wynikać z tego, że podczas zawodów w Rio de Janeiro i Glasgow sędziowie byli bardziej doświadczeni oraz tego, że ci badacze zebrali mniej danych od autora.

Decyzja	Liczba próśb o weryfikację
Sędzia podjął poprawną decyzję	160
Błędna decyzja sędziego	51
Razem	211
Która linia była sprawdzana	**Jak często**
Linia boczna	109
Linia tylna	75
Tylna linia serwisowa w deblu	20
Linia serwisowa środkowa	7

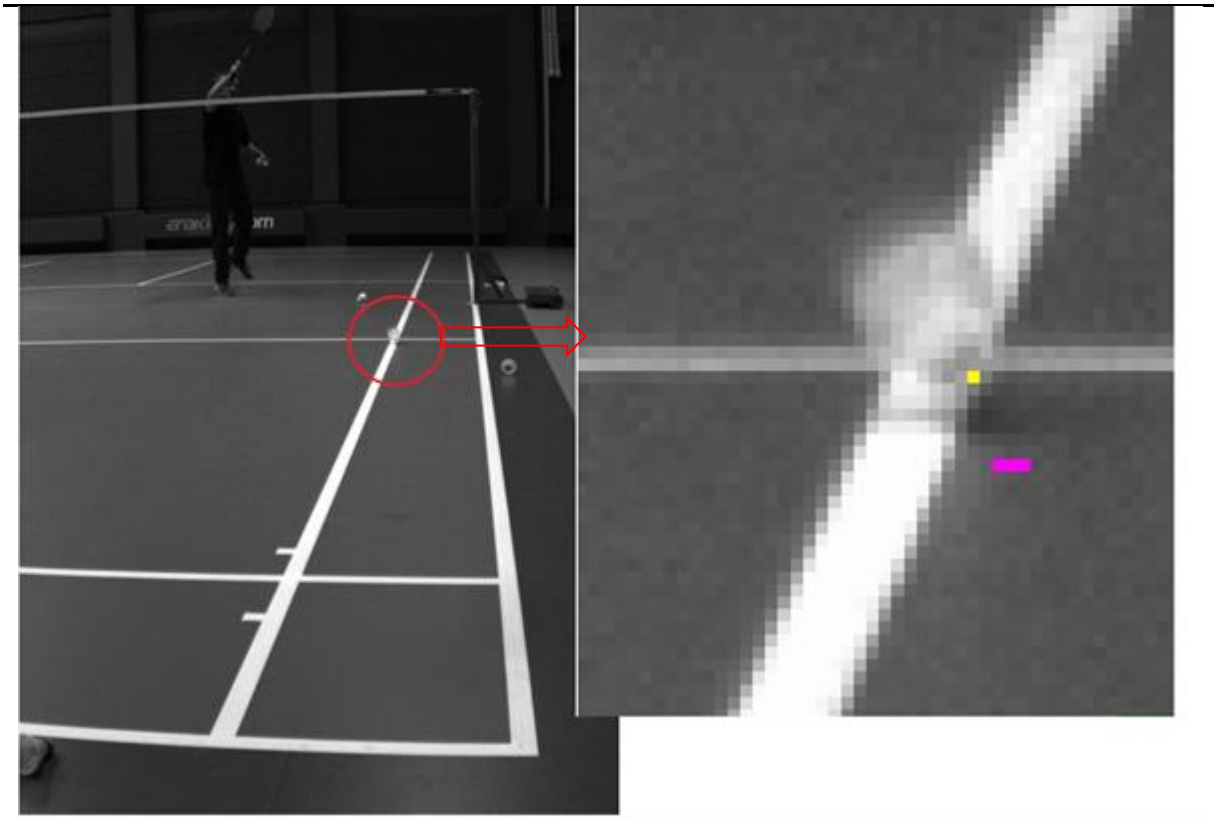
Tabela 21 Statystyka weryfikacji decyzji sędziów liniowych

Autor dodatkowo sprawdził jeszcze to, jak działa jego rozwiązanie w szczególnie trudnych dla sędziego liniowego przypadkach. Jeżeli lotka upada 20 cm od linii, a zawodnik prosi o wideo weryfikacje, tylko w celach taktycznych, to pomyłka sędziego jest mało prawdopodobna, a takie sytuacje podczas zawodów się zdarzają. Dlatego też autor ze zbioru danych wybrał te nagrania, w których lotka upada w odległości od linii mniejszej niż jej szerokość - 4 cm i dla nich zweryfikował skuteczność proponowanego rozwiązania – [Tabela 22].

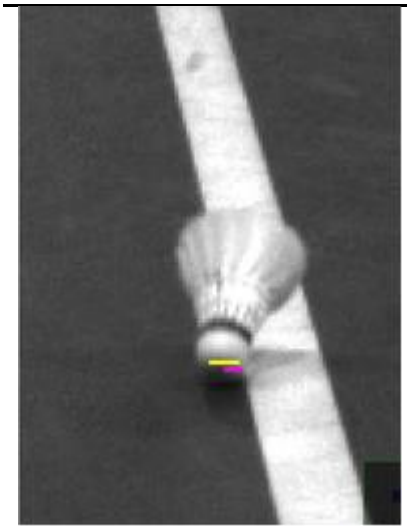
Model	Średni błąd wyznaczenia odległości miejsca upadku od linii	Średnia odległość od przewidywanego do zaznaczonego miejsca upadku	Zła decyzja IN/OUT
BalancedBaggingClassifier	6.97 mm	10.55 mm	4.84 %

Tabela 22 Skuteczność decyzji IN/OUT dla sytuacji, gdy lotka upada w pobliżu linii

Autor przeanalizował przypadki błędnej decyzji systemu i w tabeli [Tabela 23] zaprezentował wybrane sytuacje oraz wskazał przyczyny pomyłek. Kolorem żółtym zaznaczone zostało miejsce upadku wskazane przez sędziego, kolorem purpurowym miejsce upadku wyznaczone przez algorytm.



Lotka znajduje się daleko od kamery oraz jest mocno rozmazana, przez co czarny pasek jest słabo widoczny. Algorytm źle wyznaczył punkt *P* – cień rzucony przez lotkę został pomylony z czarnym paskiem. Z tego powodu miejsce upadku jest przesunięte i znajduje się na zewnątrz pola gry.



Niewielkie przesunięcie wynikające ze złego wyznaczenia linii *L* przez traker.

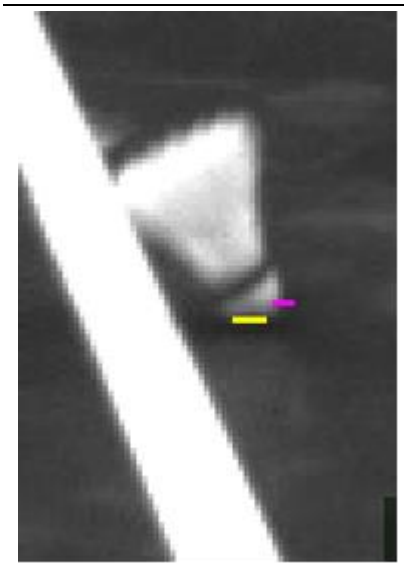
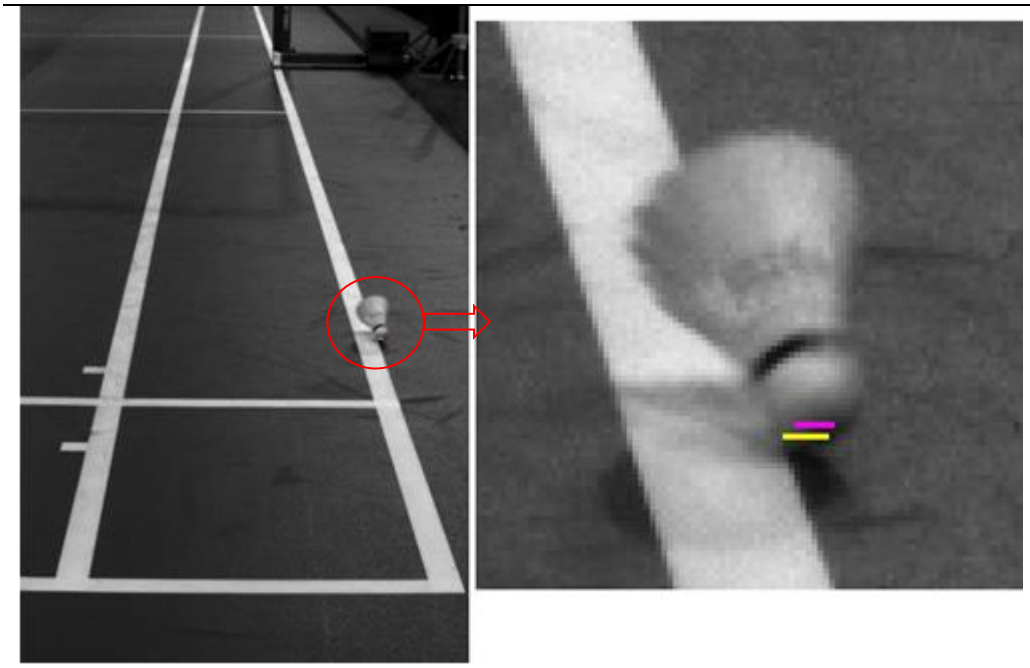
	Niewielkie przesunięcie wynikające ze złego wyznaczenia linii *L* przez traker, które nie skutkuje błędną decyzją IN/OUT.
	W tym przypadku wydaje się, że sędzia błędnie zaznaczył miejsce upadku. Zdaniem autora lotka upadła poza polem gry i decyzja systemu - OUT jest poprawna

Tabela 23 Przykłady błędnych decyzji IN/OUT systemu. Kolorem żółtym zaznaczono miejsce upadku wyznaczone przez sędziego, a kolorem purpurowym wyznaczone przez system

Porównanie zaproponowanej metody z Segment Anything Model

W celu porównania wyników przedstawionych w tabeli [Tabela 22] z wynikami uzyskanymi za pomocą uznawanej za state of the art metody SAM, autor wykonał następujący eksperyment.

Dla każdego obrazka w zbiorze danych zawierającego te nagrania, w których lotka upada w odległości mniejszej niż 4 cm od linii, wyznaczono obszar zainteresowań, który został przekazany do SAM w celu wyznaczenia masek segmentacji.

Obszar zainteresowań został wyznaczony jako kwadrat o środku w punkcie będącym pozycją korka lotki wskazanej przez traker i o boku równym n-krotności szerokości lotki zwróconej przez traker. Za n przyjmowano różne liczby i weryfikowano otrzymane rezultaty.

W wyznaczony obszarze dokonano segmentacji z wykorzystaniem algorytmu SAM. Algorytm SAM umożliwia generowanie trzech masek na podstawie monitu użytkownika (np. w danym obszarze zainteresowań lub we wskazanym punkcie). Na zdjęciach [Rysunek 56] przedstawiono trzy przykładowe maski (oznaczone kolorem niebieskim) wygenerowane przez SAM dla tego samego obrazu wejściowego. Wybór najlepszej maski został dokonany zgodnie z procedurą opisaną poniżej.

W pierwszym kroku usunięto tzw. wyspy – czyli te obszary danej maski, które nie są połączone z głównym (o największej powierzchni) obszarem oraz dziury. Maska przedstawiona na zdjęciu [Rysunek 56 a)] składa się z dwóch obszarów, a po usunięciu wysp i dziur jest przedstawiona na rysunku [Rysunek 57].

Dla każdej z masek obliczono współczynnik Jaccarda (również nazywany w literaturze anglojęzycznej jako Intersection over union - IoU zgodnie ze wzorem podanym w rozdziale 6 punkt 4). Jako predykcję przyjęto wygenerowaną maskę, a jako ground-truth obraz różnicowy. Dla obrazowanego przykładu zaprezentowany na rysunku [Rysunek 58 b)]. Ta optymalizacja ma na celu odrzucenie tych masek, które oprócz lotki obejmują również linie kortu.

Wśród wszystkich masek zwróconych przez SAM wybrano tę o najwyższej wartości IoU. *Wartości IoU dla analizowanego przykładu wynoszą odpowiednio:*

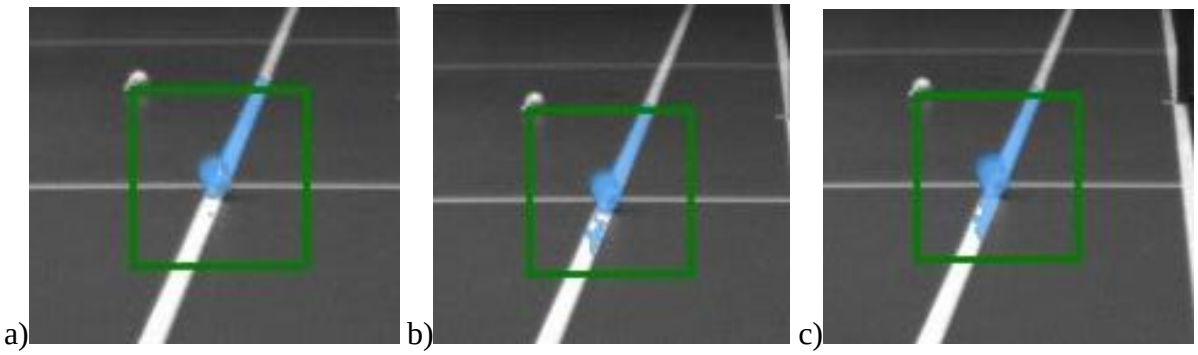
maska a) 0,111; b) 0,106; c) 0,108.

Maska a) ma najwyższą wartość IoU i dlatego została wybrana jako ta najlepsza.

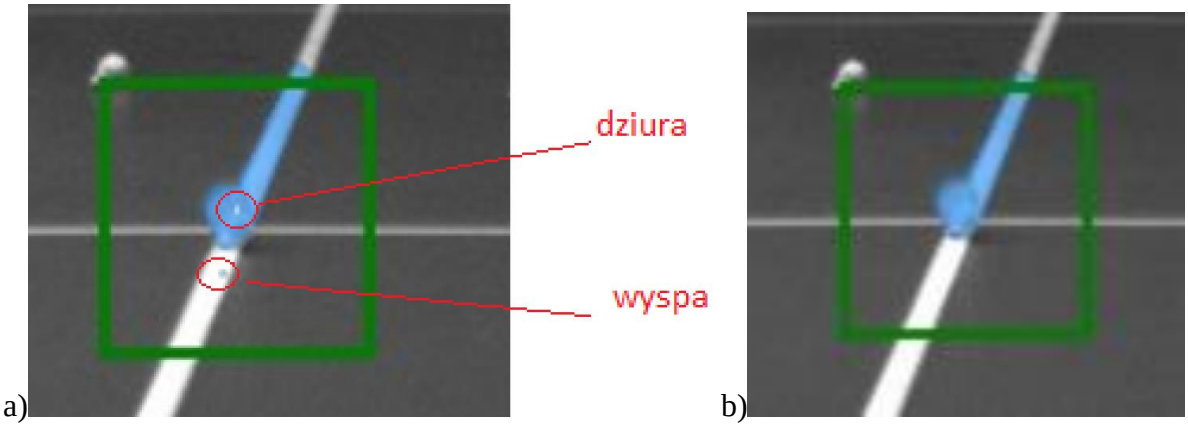
Opisana procedura ma na celu zoptymalizowanie wyników jakie można uzyskać wykorzystując SAM.

Kolejnym krokiem było wyznaczenie obszaru styku korka lotki z podłożem. Ze względu na ustawienie kamer w stosunku do podłoża i linii, można przyjąć, że piksel maski o największej wartości współrzędnej y (zgodnie z układem współrzędnych zorientowanym tak jak na [Rysunek 59]) odpowiada punktowi *S0* z rysunku [Rysunek 47]. Na marginesie, warto zauważyć, że gdyby wybrano maskę c) to punkt *S0* byłby znacznie przesunięty w dół. Następnie, w sposób opisany w rozdziale *Wyznaczenie miejsca styku lotki z podłożem* na stronie

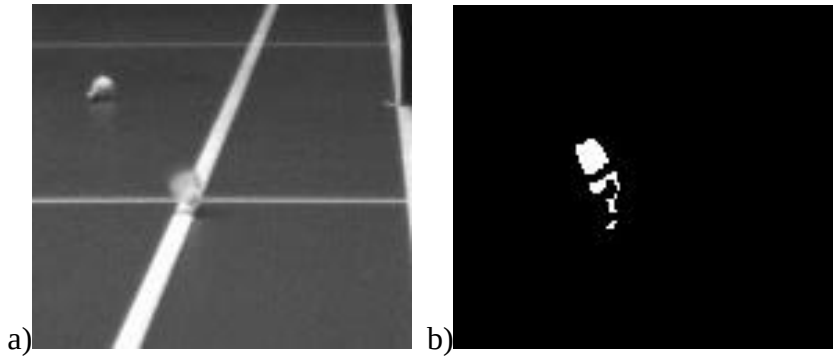
98, wyznaczono punkty *S1* i *S2*, które wyznaczają linie styku lotki z podłożem – na zdjęciu [Rysunek 58] oznaczoną kolorem purpurowym.



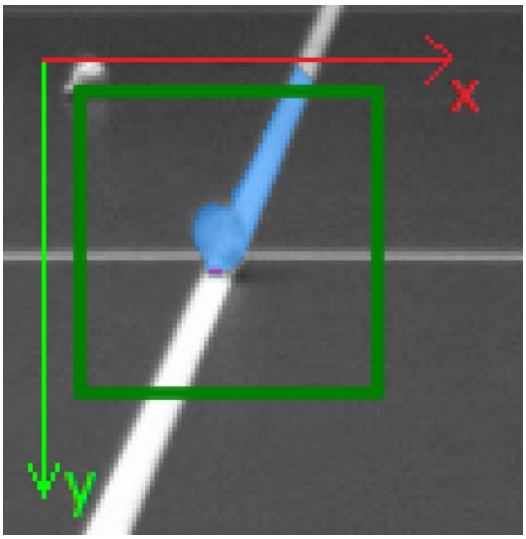
Rysunek 56 Przykładowe 3 maski (niebieski kolor) wygenerowane przez SAM dla wskazanego przez użytkownika obszaru zainteresowań (zielony kwadrat)



Rysunek 57 Maska segmentacji przed a) i po b) usunięciu wysp i dziur dla maski z rysunku Rysunek 56 a)

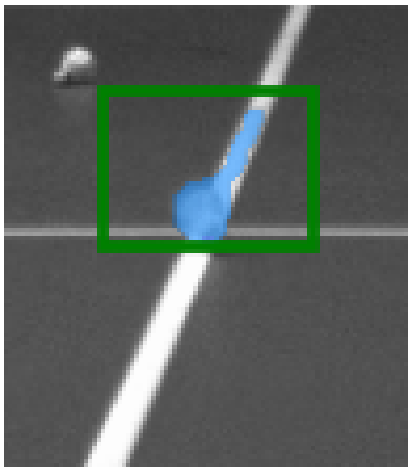


Rysunek 58 Zbliżenie na upadającą lotkę a) oraz obraz różnicowy b) utworzony z dwóch sąsiadujących klatek zapisu wideo



Rysunek 59 Wyznaczone miejsce odbicia lotki o podłoże (kolor purpurowy)

Na koniec wykonano jeszcze jeden eksperyment i wyznaczono prostokątny obszar zainteresowań w taki sposób, że dalszy (wg współrzędnej y) bok prostokąta jest oddalony o 0.25 szerokości lotki od współrzędnej y korka lotki zwróconej przez traker. Celem takiego ustawienia obszaru zainteresowań jest objęcie jak najmniejszego obszaru leżącego poza obszarem wyznaczonym przez lotkę. Ma to jednak taką wadę, że obszar zainteresowań może nie objąć korka lotki - [Rysunek 60]. Wyniki przedstawionego eksperymentu z uwzględnieniem wpływu wielkości obszaru zainteresowań na segmentację oraz porównanie z proponowaną metodą przedstawiono w tabeli [Tabela 24].



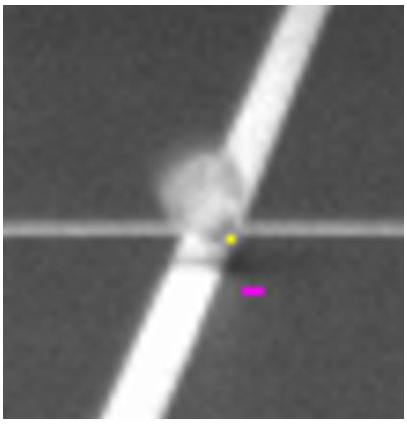
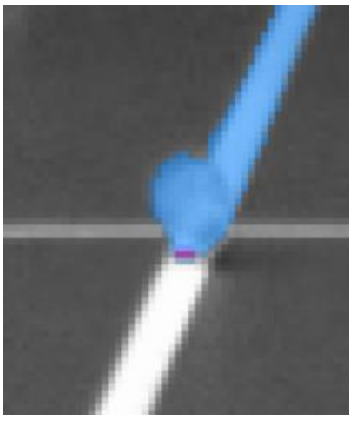
Rysunek 60 Zmniejszenie obszaru zainteresowań do prostokąta

Model	Średni błąd wyznaczenia odległości miejsca upadku od linii	Średnia odległość od przewidywanego do zaznaczonego miejsca upadku	Zła decyzja IN/OUT
Proponowana metoda	**6.97 mm**	**10.55 mm**	**4.84 %**
SAM n=2.0	9.49 mm	18.26 mm	16.23 %
SAM n=2.5	9.91 mm	19.27 mm	12.90 %
SAM n=3.0	9.23 mm	17.01 mm	12.90 %
SAM n=3.5	10.15 mm	18.89 mm	12.90 %
SAM n=4.0	14.21 mm	25.58 mm	19.36 %
SAM (obszar prostokątny)	**8.32 mm**	**13.42 mm**	**9.68 %**

Tabela 24 Porównanie skuteczność proponowanej metody i SAM, dla sytuacji, gdy lotka upada w pobliżu linii kortu

Jak widać proponowana przez autora metoda jest istotnie lepsza. Jeśli chodzi o błąd, to SAM dla kwadratowego obszaru zainteresowań ma o 2.26mm większy błąd od proponowanej metody, czyli przeszło o 32%. Co ciekawe jeśli chodzi o błędną decyzję IN/OUT to mamy tu prawie 3 krotnie gorszy wynik. Najlepszy wynik daje metoda SAM dla obszaru prostokątnego, ale i tak ma błąd większy o przeszło 19% i daje dwukrotnie gorszy wynik jeśli chodzi o błędną decyzję oceny upadku lotki w korcie czy poza nim.

Jeśli chodzi o czas to działania omawianych metod, to SAM z wykorzystaniem GPU NVidia RTX 3070Ti działa średnio 1064,80 ms, a metoda proponowana przez autora z wykorzystaniem CPU Intel i9 2.97ms

Proponowana metoda	SAM	Uwagi
		Pomimo tego że SAM połączył część linii kortu z lotką, to tak szczęśliwie się złożyło, że korek lotki został dość poprawnie wysegmentowany i wynik z SAM jest lepszy od proponowanej metody. Lotka jest daleko od kamery - obrazy po lewej są istotnie powiększone (całe zdjęcie widać w [Tabela 23] – pierwszy wiersz)

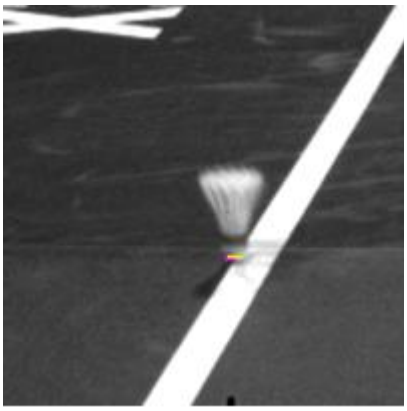
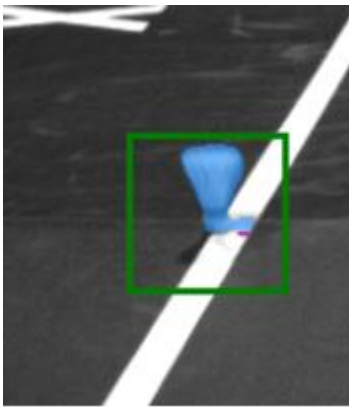
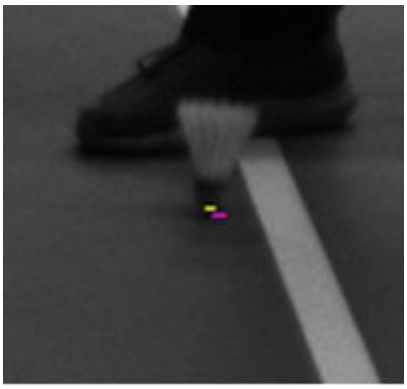
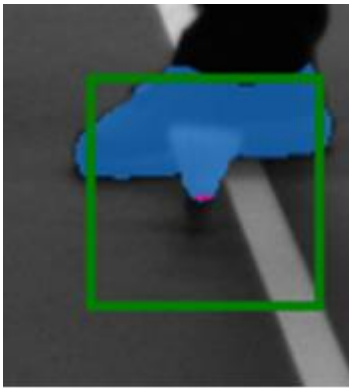
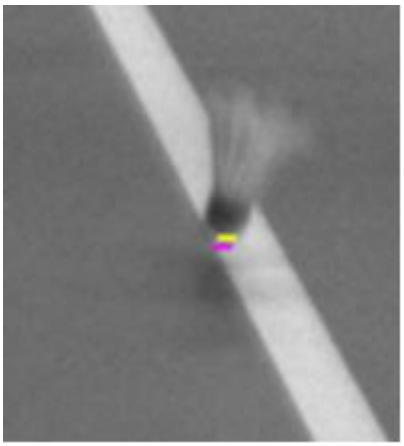
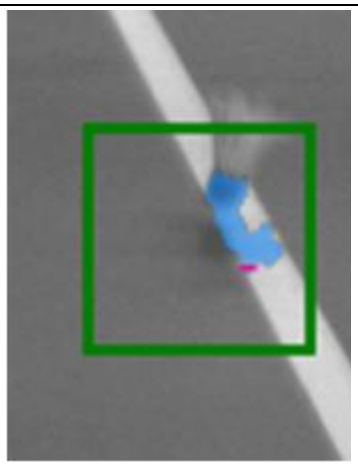
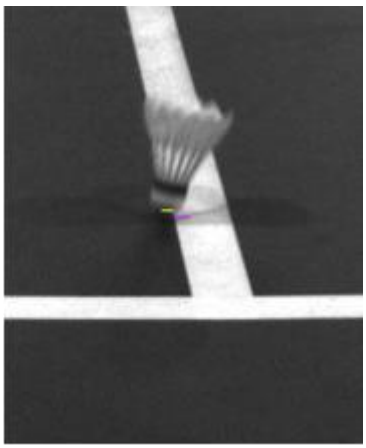
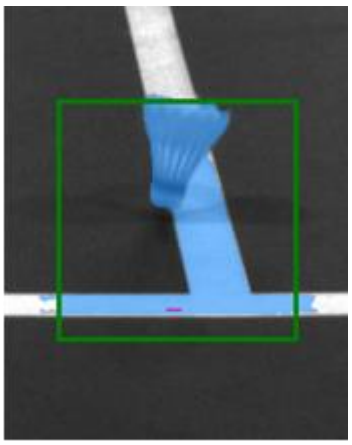
		SAM połączył część linii z korkiem lotki. Lotka upada w miejscu w którym jest łączenie dwóch części maty do badmintona.
		Pomimo nadmiernej segmentacji – maska lotki obejmuje but o białej podeszwie, to i tak miejsce upadku zostało wyznaczone prawidłowo
		Obszar zainteresowań nie objął całej lotki. SAM połączył cień rzucany przez lotkę na linię z korkiem lotki.
		Lotka upadła blisko kamery i jest dobrze oświetlona, a mimo to SAM połączył lotkę z linią boczną i końcową

Tabela 25 Przykłady segmentacji lotki przez SAM wraz z omówieniem i porównaniem z proponowaną metodą (kolorem purpurowym zaznaczono przewidywane miejsce upadku lotki, a kolorem żółtym ground-truth

Największą wadą SAM z punktu widzenia badanego problemu jest to, że często łączy lotkę i linie kortu w jeden segment. SAM zadziała dobrze jeśli ma dokładnie wskazany obszar zainteresowań. Niestety automatycznie wyznaczony, w procesie detekcji, obszar zainteresowań zawsze będzie obarczony pewnym błędem.

4. Skuteczność zaproponowanego rozwiązania

Zaproponowane rozwiązanie składa się z 3 głównych elementów, które szczegółowo zostały opisane wcześniej.

1. Detekcji i śledzenia lotki.
2. Znajdowania w strumieniu wideo ramki, w której nastąpiło odbicie o podłoże.
3. Wyznaczenia miejsca upadku lotki i oceny czy było ono w polu gry czy poza nim.

Ostatecznym celem zaproponowanego rozwiązania jest decyzja - czy lotka upadła w polu gry czy poza nim. W tym celu autor przeprowadził ostateczną weryfikację.

Po wytrenowaniu i walidacji modeli z użyciem danych zebranych w różnych halach i w różnorodnych warunkach oświetleniowych, autor przeprowadził ostateczne testy z użyciem danych zebranych podczas dwóch turniejów badmintonowych - Mistrzostw Świata seniorów w Katowicach oraz mistrzostw Polski U17 w Głubczycach. Podczas tych turniejów były rejestrowane tylko te sytuacje, kiedy zawodnik nie zgadzał się z decyzją sędziego. Z takiego zbioru danych autor wybrał najtrudniejsze do oceny przez sędziów sytuacje – te, gdy lotka upadała na kort w odległości mniejszej niż 4 cm od linii. W ten sposób do testów wybrano 62 sekwencje wideo. Dla każdej sekwencji autor zaznaczył miejsce upadku lotki i ocenił czy było pole czy aut. Anotacje miejsca upadku lotki zostały zweryfikowane przez kwalifikowanego sędziego. Przygotowany w ten sposób zbiór testowy został użyty do ostatecznej weryfikacji skuteczności proponowanego rozwiązania.

We wszystkich przypadkach testowych algorytm śledzący lotkę (opisany w rozdziale *Detekcja i śledzenie lotki*) poprawnie znajdował lotkę w momencie odbicia o podłoże i przed odbiciem.

Dla każdej sekwencji wytrenowany model (opisany w rozdziale *Wyznaczenie w strumieniu wideo kluczowej ramki, w której nastąpiło odbicie lotki o podłoże*) wyznaczył ramkę, w której nastąpiło odbicie o podłoże - czyli tę, dla której prawdopodobieństwo zwrócone przez model było najwyższe.

Dla tej ramki za pomocą modelu opisanego w rozdziale *Wyznaczenie miejsca odbicia się lotki o ziemię w stosunku do referencyjnych linii kortu* dla każdej sekwencji wyznaczono miejsce odbicia o podłoże i na tej podstawie podjęto decyzję czy lotka upadła w polu gry czy poza nim.

Ostateczne wyniki przedstawiono tabeli [Tabela 26]

		Prawdziwe wartości	
		IN	OUT
Przewidywane wartości	IN	**33**	3
	OUT	1	**25**

	precision	recall	f1-score	liczność
IN	0.97	0.92	0.94	36
OUT	0.89	0.96	0.93	26
accuracy			0.94	62
macro avg	0.93	0.94	0.93	62
weighted avg	0.94	0.94	0.94	62
Balanced accuracy	0.94			

Tabela 26 Ostateczne wyniki skuteczności zaproponowanej metody

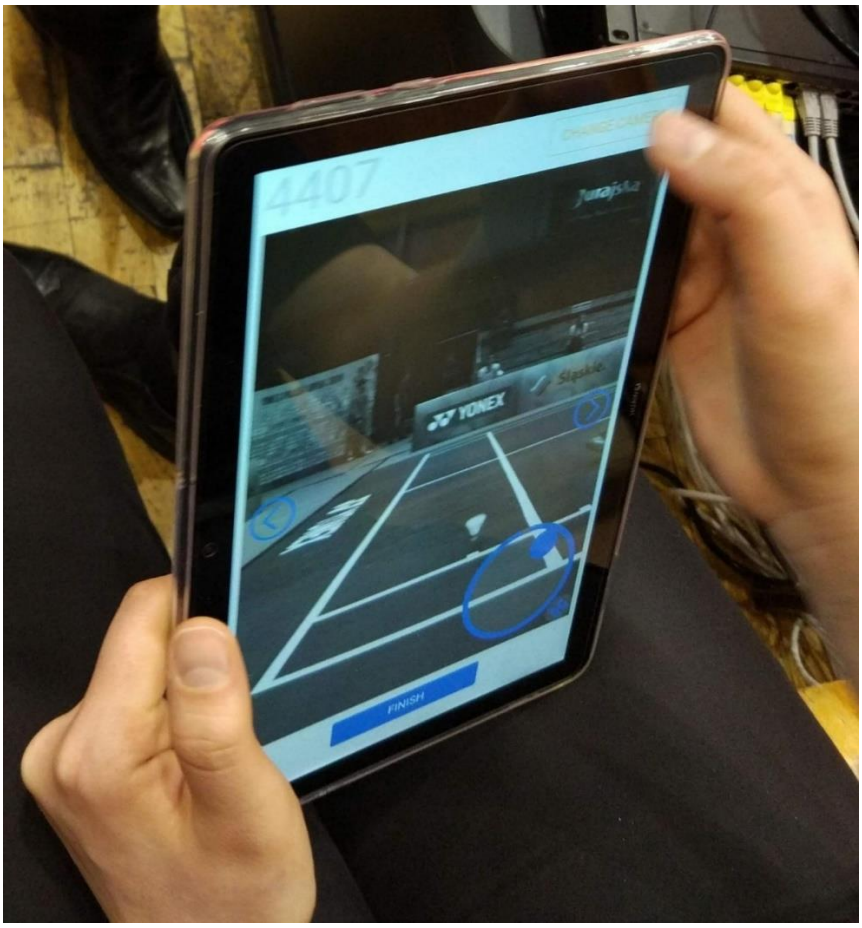
Jak widać zaproponowane rozwiązanie myli się w 6%. W porównaniu z sędziami liniowymi, którzy podejmują błędne decyzje w 20-24% przypadków (w zależności od doświadczenia) proponowane rozwiązanie jest istotnie skuteczniejsze niż człowiek.

Jeśli chodzi o średni błąd wyznaczenia odległości miejsca upadku od linii, to wyniósł on 19.48mm. Jest to wynik znacznie gorszy niż ten który został osiągnięty dla przypadków, gdy analizowano dokładnie tę ramkę, w której nastąpiło odbicie o podłoże (patrz [Tabela 20]), a nie tę ramkę którą wskazał model przewidujący moment odbicia. Zazwyczaj model przewidujący moment odbicia robi to dokładnie lub myli się o 1, 2 ramki. Niestety dla zestawu danych użytych do testów jest kilka przypadków, gdzie pomyłka modelu wyznaczającego moment odbicia o podłoże była większa niż 2 ramki. W szczególności w dwóch przypadkach jest bardzo duża i wyniosła odpowiednio 91 i 108 ramek różnicy. Autor przeanalizował te dwa przypadki. W jednym przypadku model wyznaczył ramkę wskazującą odbicie o podłoże nie dla pierwszego, a dla drugiego odbicia. W drugim przypadku pomyłka była spowodowana tym, że algorytm zarejestrował odbicie, lotki, które nastąpiło na korcie obok. Sytuacja ta została zarejestrowana dla kamery obserwującej linie tylną, podczas turnieju w Głubczycach, gdzie odległości pomiędzy kortami były mniejsze niż te wymagane przepisami. Jeśli odrzuci się te dwa przypadki, to wspomniany błąd zmniejsza się do 12.97mm. Jest to mniej niż połowa szerokości korka.

4.1. Weryfikacja wyniku z systemu przez sędziego technicznego

Skuteczność na poziomie 94% w porównaniu ze skutecznością sędziów wydaje się wystarczająco wysoka, aby zaproponowane rozwiązanie można było wdrożyć komercyjnie. Jednakże zawodnicy, jak i organizacje takie jak BWF (Badminton World Federation), BE (Badminton Europe) czy chociażby PzBad (Polski związek Badmintona) oczekują 100% skuteczności. Dlatego też autor opracował dodatkowy haptyczny interfejs do systemu umożliwiający weryfikację decyzji systemu przez sędziego [Rysunek 61, Rysunek 62]. W przypadku, gdy sędzia analizując zapis wideo w zwolnionym tempie (ramka po ramce) zauważy pomyłkę systemu, to ma możliwość korekty wyznaczonego miejsca upadku. Dzięki wyjątkowo wygodnemu interfejsowi [Rysunek 62] weryfikacja zajmuje maksymalnie kilkanaście sekund.

Rysunek 61 Sędziowie testujący system podczas jego weryfikacji w trakcie turnieju w Częstochowie



Rysunek 62 Dotykowy interfejs wraz z unikalnym sposobem przewijania zapisu wideo

5. Dyskusja i podsumowanie rozprawy

Lotka do badmintona może osiągać prędkość przekraczającą 400 km/h, co czyni tę grę najszybszym ze wszystkich sportów pod względem dynamiki. Niewielki rozmiar korka oraz zmienna prędkości lotki powodują, że sędziom niezwykle trudno jest precyzyjnie ocenić miejsce upadku lotki na kort.

W ramach doktoratu wdrożeniowego autor podjął się opracowania systemu wspomagającego decyzje sędziowskie w badmintonie. Opracowanie użytecznego rozwiązania nie jest zadaniem banalnym i należy rozwiązać wiele problemów naukowych i technicznych, aby takie rozwiązanie można było wdrożyć. W niniejszej rozprawie autor dokładnie omówił trzy z nich:

- detekcja i śledzenie lotki,
- wyznaczenie kluczowej ramki, w której nastąpiło odbicie o podłoże,
- wyznaczenie miejsca odbicia lotki o podłoże.

W porównaniu z innymi sportami rozwiązanie przedstawionych problemów badawczych jest szczególne trudne w badmintonie ze względu na:

- Zmienną prędkość lotki. Lotka może mieć przy podłożu prędkość 10-krotnie niższą niż w momencie uderzenia przez zawodnika. Dla porównania: piłka tenisowa traci około połowę swojej prędkości.
- Niestabilną i zależną od wielu czynników trajektorię lotki, podatną na zmianę spowodowaną ruchami powietrza.
- Niewielki rozmiar korka lotki, przeszło dwa razy mniejszy od średnicy piłki tenisowej [Rysunek 64].
- Biały kolor lotki, który zlewa się z białymi liniami.
- Rozmycie lotki na rejestrowanym obrazie spowodowane jej ruchem postępowym i wirowym.
- Sztuczne oświetlenie, które nie jest tak silne jak naturalne oświetlenie dzienne i w zasadzie wykluczające rejestrację wideo z prędkością powyżej 200 klatek na sekundę.
- Różnorodne warunki oświetleniowe w halach (często wyposażonych w migoczące z częstotliwością 50Hz lampy).
- Różnorodne warunki przestrzenne, często zmieniające się nawet w czasie tego samego turnieju.

- Różnorodne dodatkowe obiekty, pojawiające się w niewielkiej odległości od linii kortu, takie jak banery reklamowe, kamerzyści, czy też zawodnicy grający na korcie obok.

O tym, że badany problem jest wyjątkowo trudny, świadczy również znikoma liczba publikacji naukowych omawiających badane przez autora zagadnienia. W szczególności temat upadku lotki w korcie lub poza nim podjęto tylko w jednej znanej autorowi publikacji [78]. Co ciekawe: zagadnienie lokalizacji piłki względem referencyjnych linii pola gry dla innych dyscyplin sportowych, takich tenis, piłka nożna, czy też siatkówka, również nie jest szeroko badane przez innych naukowców. W liczącej 68 stron publikacji *Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions* [14] autorzy umieścili 292 referencje do publikacji dotyczących wizji komputerowej w sporcie, ale nie wskazali ani jednej referencji do publikacji poruszających temat lokalizacji piłki/lotki w stosunku do referencyjnych linii pola gry. Stosunkowo dużo jest publikacji dotyczących metod detekcji i śledzenia piłki lub graczy (głównie wykorzystujących głębokie sieci neuronowe). Już to, że tak mało jest badań innych naukowców, czyni badania autora wyjątkowymi w skali światowej.

Autor podejrzewa, iż oprócz obiektywnych trudności związanych z gromadzeniem danych podczas zawodów sportowych (organizator zawodów musi wyrazić zgodę na umieszczenie sprzętu rejestrującego w hali i zapewnić badaczom choćby dostęp do zasilania oraz dodatkową przestrzeń w obiekcie), brak zainteresowania badaczy tym tematem wynika również z tego, że istniejący od 20 lat system Hawk-Eye zmonopolizował rynek oprogramowania wspomagającego decyzje sędziowskie. Szalenie trudno jest stworzyć konkurencyjne rozwiązanie do takiego, które jest już rozwijane komercyjnie od 20 lat.

Mimo to autor podjął to wyzwanie, licząc na to, że dzięki swojemu zaangażowaniu i wykorzystaniu unikalnej wiedzy domenowej, zdoła opracować rozwiązanie, które będzie użyteczne rynkowo.

Autor, będąc zawodnikiem, dobrze zna środowisko badmintonowe, dzięki czemu organizatorzy turniejów i Polski Związek Badmintona przychylnie patrzyli na jego prace i pozwalali na umieszczanie podczas zawodów sprzętu rejestrującego. Wspomniany turniej w Katowicach trwał 9 dni i przez te 9 dni, codziennie przez 10 godzin, autor znajdował się w hali i rejestrował dane oraz testował opracowane rozwiązanie.

Dzięki temu autor miał możliwość zgromadzenia zróżnicowanych zapisów wideo i przetestowania różnorodnych ustawień sprzętu rejestrującego oraz zauważenia istotnych ograniczeń, mających wpływ na jego badania. Inni badacze często nie mają takich możliwości

i zmuszeni są analizować zapisy wideo z transmisji telewizyjnych lub rejestrują dane w jednorodnych warunkach laboratoryjnych.

Autorowi udało się opracować satysfakcjonujące rynek rozwiązanie, co nie oznacza, że nie jest ono pozbawione wad i nie ma miejsca na poprawę. Poniżej wypunktowano wady i zalety opracowanej metody w podziale na cztery kategorie.

1. **Ogólna ocena rozwiązania**
   - Zalety:
     - Rozwiązanie efektywne kosztowo.
     - Działające w czasie rzeczywistym.
     - Szybki montaż i bezproblemowy transport - nie są wymagane kamery nad kortem.
     - Przetestowane w realnych warunkach turniejowych.
     - Zintegrowane z systemami live scoring i live streaming.
   - Wady:
     - Dokładność lokalizacji miejsca upadku lotki niewiele lepsza niż akceptowalna przez rynek - 13mm.
     - Nie zawsze wynik jest zgody z prawdą i wymaga korekcji przez sędziego technicznego, co wydłuża do kilkunastu sekund czas, po którym zawodnicy widzą oficjalną decyzję.
     - Brak rejestracji całej trajektorii lotki – nie można zebrać interesujących statystyk, takich jak na przykład średnia prędkość początkowa uderzenia kończącego.
2. **Detekcja i śledzenie lotki**
   - Zalety:
     - Metoda wyjątkowo efektywna czasowo.
     - Metoda wyjaśnialna – jasne kryteria odrzucania kandydatów na lotkę.
   - Wady:
     - Konieczne wstępne przetwarzanie obrazu i usunięcia efektu migotania.
     - Jeśli traker zacznie śledzić obiekt, to dopóki go nie zgubi, nie zacznie śledzić innego obiektu. Nie został rozwiązany problem, który powstaje, gdy najpierw pojawia się w polu widzenia kamery lotka z kortu obok i, mimo że w polu widzenia może się pojawić po chwili druga lotka (i to z kortu, który nas interesuje), to i tak traker nie rozpocznie jej śledzenia, dopóki nie przestanie śledzić pierwszej lotki.
     - Przybliżona pozycja korka na etapie detekcji. Nie jest ona wystarczająco dokładna do wiarygodniej oceny miejsca upadku lotki.

- Rzucany przez będącą przy podłożu lotkę cień powoduje, że ma on istotny wpływ na wyznaczenie pozycji korka przez traker.
- Często lotka jest gubiona zaraz po odbiciu. W opracowanej metodzie nie ma to większego znaczenia, jednakże ogranicza możliwości poprawy dokładności metody.

3. **Wyznaczenie kluczowej ramki**
   - Zalety:
     - Szybkość działania.
   - Wady:
     - Silne założenia wejściowe – działa w połączeniu z trakerem i korzysta z danych dostarczonych przez moduł śledzący lotkę. Błędne dane z trakera powodują błędy w wyznaczeniu kluczowej ramki.
     - Możliwe bardzo duże pomyłki. W szczególności, gdy kluczowa ramka zostanie wyznaczona w momencie drugiego odbicia się lotki o podłoże.
4. **Precyzyjna segmentacja korka lotki i lokalizacja miejsca odbicia w stosunku do linii kortu**
   - Zalety:
     - Szybkość działania.
     - Działa dokładniej niż uznany za state of the art Segment Anything Model.
   - Wady:
     - Zależy od danych dostarczonych przez moduł detekcji i śledzenia lotki – trajektorii i szerokości lotki. Błędne dane z trakera mogą powodować błędną segmentację korka lotki.
     - Cień lotki może powodować błędne wyznaczenie korka od lotki
     - Metoda zakłada, że lotka musi mieć ciemny pasek oddzielający korek od piór. Tego nie wymagają przepisy, ale autor do tej pory nie widział lotki, która nie miałaby tego paska. Niestety, jeśli lotka nie będzie posiadać czarnego paska, metoda całkowicie przestanie działać.

W ocenie autora oryginalność opracowanej metody polega na osiągnięciu zadowalających rezultatów (w czasie rzeczywistym) w różnorodnych środowiskach przy wykorzystaniu ograniczonych zasobów sprzętowych i finansowych. Nie byłoby to możliwe, gdyby nie posiadana przez autora wiedza domenowa.

W związku z coraz większą świadomością widzów i zawodników, jakie są możliwości wizyjnych systemów komputerowych, oczekuje się, że systemy wspomagające decyzje sędziowskie będą coraz powszechniej stosowane podczas każdego turnieju, a nie tylko w czasie największych imprez. Aby było to osiągalne, muszą one być możliwie przystępne. Oznacza to umiarkowany koszt oraz łatwość montażu i transportu, co zapewnia opracowane przez autora rozwiązanie.

Udoskonalenie metody wymaga dalszych badań. W szczególności warto zbadać najnowsze modele sieci neuronowych. Coraz tańszy i wydajniejszy sprzęt oraz coraz większe możliwości sieci neuronowych powodują, że w niedługiej przyszłości metody bazujące na sieciach neuronowych mogą działać dokładniej i równie efektywnie czasowo, jak metoda opracowana przez autora.

Na zakończenie autor chciałby poruszyć temat certyfikacji systemów wspomagających decyzje sędziowskie przez związki sportowe. Rozwiązanie opracowane przez autora wspólnie z panem Nowiszem zostało certyfikowane przez Polski Związek Badmintona oraz zaakceptowane przez światową organizację badmintona (BWF), ale w ocenie autora procedura, którą kierował się PzBad i BWF pozostawia wiele do życzenia.

Procedura weryfikacji systemu przez przedstawicieli BWF miała miejsce podczas turnieju Yonex Polish Open w Częstochowie. Polegała ona na tym, że wyznaczeni sędziowie korzystali systemu podczas zawodów „w cieniu". Zawodnicy nie mieli możliwości poproszenia o wideo weryfikację, ale sędziowie pracujący „w cieniu" weryfikowali decyzję sędziów liniowych, porównując ją z decyzją systemu i analizując zapis losowo wybranych nagrań wideo, oceniali skuteczność weryfikowanego rozwiązania. Autor musi uczciwie przyznać, że taka procedura nie jest obiektywna, a decyzja o dopuszczeniu do użytku podczas zawodów systemu wspomagającego sędziów (takiego jak opisany w tej rozprawie czy też podobnego), zależy tylko od pisemnej rekomendacji sędziów, a nie od mierzalnych wskaźników.

Brak oficjalnej procedury weryfikacyjnej powoduje, że dostawcy takich systemów nie mogą rzetelnie porównać swoich rozwiązań. Skutkuje to również tym, że trudno jest podważyć podejmowane przez oficjeli związków sportowych decyzje o dopuszczeniu rozwiązania do użytku podczas zawodów. Mając na uwadze brak oficjalnych wytycznych, autor, przeprowadzając swoje badania, korzystał z własnych zbiorów danych, a eksperymenty przeprowadził zgodnie z ogólnie przyjętymi standardami i swoją najlepszą wiedzą.

Warto nadmienić, że Światowa Federacja Tenisa (ITF) udostępnia na swojej stronie internetowej niezmienną od 2020 roku oficjalną procedurę ewaluacyjną takich systemów [117], ale procedura ta nie mówi, jak mierzyć dokładność systemu. W celu ewaluacji procedura zakłada użycie kamer szybkoklatowych - *"high-speed video cameras are used as the sole and definitive method by which accuracy is established. The "mark" left on the court surface following impact is used only to aid in the selection of impacts to analyze"*.

Zgodnie z tymi wytycznymi, autor również użył zapisu wideo z kamer szybkoklatkowych do weryfikacji skuteczności systemu, a nie badał śladu odbicia na podłożu, na którym taki ślad byłby widoczny - na przykład na rozsypanym proszku.

Z powodu braku oficjalnych zbiorów danych (zatwierdzonych i udostępnionych przez organizacje sportowe) trudno jest ocenić, czy wskazana przez autora dokładność jest lepsza bądź też gorsza niż dokładność rozwiązań istniejących już na rynku. Co więcej - firmy, które mają w swojej ofercie takie systemy, nie udostępniają żadnych testowych zbiorów danych, ani też sposobu obliczenia mierzalnych wskaźników. Autorowi udało się dotrzeć jedynie do marketingowej informacji z firmy Hawk-Eye, że ich system lokalizacji miejsca odbicia piłki w tenisie ma średni błąd na poziomie 3.6mm. Wspomniana firma nie udostępnia takiej informacji w przypadku badmintona. Według autora ze względu na to, iż korek lotki jest znacznie mniejszy niż piłka do tenisa [Rysunek 64] oraz to, że lotka jest biała i zlewa się z białymi liniami, a piłka do tenisa jest zielona i wyraźnie odcina się od linii, problem oceny, czy lotka upadła w polu gry, czy poza nim, jest dużo trudniejszy niż ocena, czy piłka tenisowa odbiła się w korcie czy poza nim. Nie bez znaczenia jest też to, że tenisiści w momencie odbicia piłki o kort, są w znacznie większej odległości od niej niż badmintoniści od lotki. Brak dodatkowych obiektów na korcie tenisowym również znacznie ułatwia segmentację piłki tenisowej.

Najdłużej dostępnym na rynku systemem elektronicznej oceny upadku piłki w polu gry bądź poza nim jest system firmy Hawk-Eye, który od 2024 roku zastąpił sędziów liniowych na największych turniejach tenisowych rozgrywanych na nawierzchni twardej. Interesujące jest to, że wspomniany system nie jest używany na mączce. Na tej nawierzchni w przypadku wątpliwości sędzia po prostu sprawdza ślad odbicia piłki.

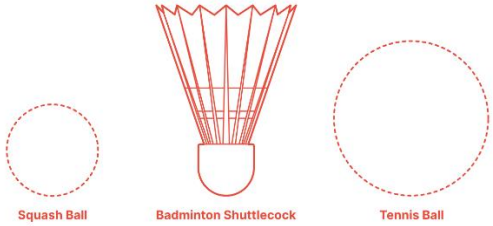
Według autora obecne rozwiązania, jakkolwiek są skuteczniejsze w podejmowaniu decyzji IN/OUT niż ludzie, to nie są skuteczne w 100% [100, 102]. Dodatkowo autor pragnie zwrócić uwagę, że decyzja systemu jest prezentowana publiczności w postaci komputerowej animacji, a nie jako prawdziwy zapis wideo. Ma to podstawową zaletę - nawet jeśli decyzja systemu jest

błędna, to zawodnicy z nią nie dyskutują, gdyż nie mają referencyjnego zapisu wideo ani śladu na korcie.

W przypadku tenisa wspomniany system Hawk-Eye prezentuje decyzję IN/OUT od razu, a w przypadku badmintona - decyzja jest prezentowana po kilkunastu sekundach. Autor uważa, że w badmintonie decyzja systemu Hawk-Eye jest również weryfikowana przez oddelegowanego sędziego technicznego. Przykładowo: podczas meczu Caroline Marin w 1/8 finału igrzysk w Paryżu 2024 na decyzję systemu widzowie musieli czekać aż 45 sekund. Autor na podstawie zapisów wideo 16 meczów badmintona, rozegranych podczas Mistrzostw Świata z 2021 roku, obliczył średni czas, jaki potrzebny był na prezentację IN/OUT systemu Hawk-Eye (18,00 s) i porównał to ze średnim czasem, jaki był potrzebny na prezentację wyniku z systemu opracowanego poprzez autora (18,43 s).

Opracowane rozwiązanie, ze względu na brak kamer nad kortem, potrzebuje mniej sprzętu i okablowania niż wspomniany Hawk-Eye [118], a wszystko, co jest potrzebne do instalacji systemu na korcie, mieści się w bagażniku większego samochodu osobowego [Rysunek 63].

Rysunek 63 Cały sprzęt potrzebny do instalacji systemu na jednym korcie



Rysunek 64 Porównanie wielkości lotki z piłką do tenisa i squash[20]

[20] Źródło: https://www.dimensions.com/element/badminton-shuttlecock

Opracowana metoda od samego początku była tworzona z myślą o badmintonie. Z tego też powodu nie może zostać bezpośrednio wykorzystana w innych dyscyplinach sportu. Uogólnienie metody na inne dyscypliny sportu wymaga zmian w algorytmach (dopasowanie wartości progowych, zmiana zestawu wyliczanych cech dla modeli) oraz zebraniu nowego zbioru danych do testów. Jednakże pewna część bibliotek, które powstały podczas tworzenia oprogramowania, może zostać wykorzystana, np. interfejs do kamer rejestrujących dane, biblioteka usuwająca efekt migotania oświetlenia, równoległe przetwarzanie strumienia wideo z wielu kamer.

Kolejne kroki, które wykonuje oprogramowanie, wspomagające sędziów liniowych w sportach, gdzie mamy piłkę i ograniczone pole gry, są w ogólności takie same, aczkolwiek ze względu na różne specyfiki poszczególnych gier, w detalach będą się różnić. Przykładowo: piłka do siatkówki, w przeciwieństwie do lotki, ulega mocnym odkształceniom w momencie odbicia o parkiet i ta jej właściwość musi zostać uwzględniona podczas opracowywania algorytmów oceniających, czy piłka była w polu gry, czy poza nim.

Podsumowując: autor ma nadzieję, że przeprowadzone przez niego badania przyczynią się do popularyzacji systemów wspomagania decyzji sędziowskich w jego ulubionej dyscyplinie sportowej, a związki sportowe opracują obiektywne procedury certyfikacji takich systemów.

6. Słowniczek pojęć

1. **System challenge** w sporcie to mechanizm, który pozwala zawodnikom lub drużynom na zgłoszenie wątpliwości co do decyzji sędziowskiej i poproszenie o jej ponowne rozpatrzenie z wykorzystaniem technologii wspomagającej. Zwykle system ten jest wspomagany przez powtórki wideo, lub zaawansowane systemy komputerowe analizujące a decyzja sędziego może być zmieniona, jeśli analiza technologiczna wykaże, że doszło do błędu.
2. **Erozja** to jedna z podstawowych operacji morfologicznych, stosowana głównie do przetwarzania obrazów binarnych. Erozja powoduje "kurczenie" się obiektów w obrazie, co oznacza zmniejszanie rozmiaru białych obszarów na obrazie binarnym. Oblicza lokalne minimum na obszarze danego jądra. Erozja podczas przesuwaniu jądra (elementu strukturalnego o zadanej wielkości) po obrazie oblicza minimalną wartość z pikseli nakładających się na jądro i zastępuje analizowany piksel obrazu tą wartością.
3. **Deskryptor** - Zestaw cech, które reprezentują obiekt lub dane. Deskryptory są używane do opisywania i porównywania obiektów, umożliwiając ich identyfikację lub klasyfikację.
4. **IoU - Intersection over Union** - miara używana do oceny jakości modeli segmentacyjnych i detekcyjnych w zadaniach związanych z wizją komputerową. IoU mierzy, jak dobrze wykryty obiekt (tzw. predykcja) pokrywa się z rzeczywistym obiektem (tzw. ground truth).
5. IoU jest obliczane jako stosunek pola wspólnego części wykrytej przez model (Intersection) do pola sumy obu obszarów (Union) zgodnie ze wzorem:

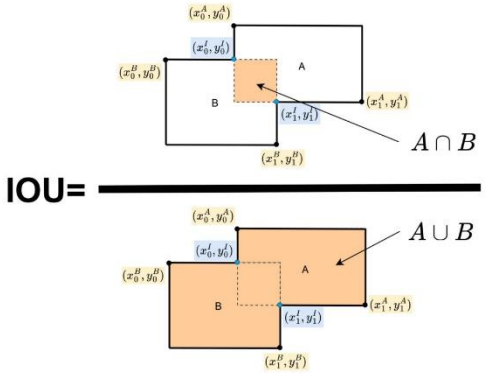
$$IoU(A, B) = \frac{(A \cap B)}{(A \cup B)}$$

Gdzie A – Obszar zajmowany przez predykcję,

B – Obszar zajmowany przez ground truth

Wynik IoU mieści się w przedziale od 0 do 1, gdzie:
IoU = 0: Brak pokrycia predykcji z rzeczywistym obiektem.
IoU = 1: Idealne pokrycie predykcji z rzeczywistym obiektem.

Można to przedstawić graficznie:



Rysunek 65 Graficzna reprezentacja miary IOU[21]

6. **Confusion matrix (macierz pomyłek)** - narzędzie używane do oceny wydajności modeli klasyfikacyjnych. Jest to tabela, która wizualizuje, jak dobrze model klasyfikuje próbki do odpowiednich klas, porównując przewidywane klasy z rzeczywistymi klasami.

   Macierz pomyłek składa się z czterech podstawowych elementów:

   1. **True Positives (TP)** – Liczba próbek poprawnie zaklasyfikowanych do danej klasy.
   2. **True Negatives (TN)** – Liczba próbek, które model poprawnie zaklasyfikował jako nie należące do danej klasy.
   3. **False Positives (FP)** – Liczba próbek błędnie zaklasyfikowanych jako należące do danej klasy (tzw. fałszywe alarmy).
   4. **False Negatives (FN)** – Liczba próbek, które model błędnie sklasyfikował jako nie należące do danej klasy, mimo że faktycznie do niej należały.

   Przykład macierzy pomyłek dla klasyfikacji binarnej:

	Predykcja: Pozytywna	**Predykcja: Negatywna**
Rzeczywista: Pozytywna	**True Positive (TP)**	False Negative (FN)
Rzeczywista: Negatywna	False Positive (FP)	**True Negative (TN)**

[21] Źródło: https://machinelearningspace.com/intersection-over-union-iou-a-comprehensive-guide/

7. **Precision (precyzja)** - miara używana w klasyfikacji binarnej i wieloklasowej określająca, jaki procent z wykrytych przez model pozytywnych przykładów jest faktycznie poprawny. Precision skupia się na tym, jak trafne są pozytywne predykcje modelu, czyli ile z nich rzeczywiście należy do klasy pozytywnej. Wyraża się wzorem:

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)}$$

Gdzie:

True Positives (TP) – liczba prawidłowo wykrytych pozytywnych przykładów.

False Positives (FP) – liczba przykładów błędnie zaklasyfikowanych jako pozytywne.

Wysoka precyzja oznacza, że model generuje niewiele fałszywych alarmów (False Positives), czyli większość predykcji pozytywnych jest trafna. Niska precyzja wskazuje, że wiele przykładów zaklasyfikowanych jako pozytywne w rzeczywistości do tej klasy nie należy.

Precision jest szczególnie ważne w sytuacjach, gdy fałszywe alarmy mają duże konsekwencje, np. w diagnostyce medycznej.

8. **Recall (czułość)** - miara używana w klasyfikacji, która określa, jaki procent rzeczywistych pozytywnych przykładów został poprawnie wykryty przez model. Wyraża się wzorem:

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)}$$

Recall mierzy, jak dobrze model potrafi wykryć wszystkie przypadki należące do klasy pozytywnej.

9. **F1 (F1-score)** - miara używana do oceny jakości modeli klasyfikacyjnych, która pokazuje, jak dokładnie model oddaje wyniki, mające znaczenie. Należy zauważyć, że im bardziej niezrównoważony jest zestaw danych, tym niższy może być wynik F1, nawet przy tej samej dokładności ogólnej. Miara ta jest szczególnie przydatna, gdy mamy do czynienia z niezrównoważonymi danymi, gdzie jedna klasa występuje znacznie częściej niż inne. F1-score balansuje między precyzją a czułością, dając ogólną miarę wydajności modelu i wyraża się wzorem:

$$F1 = 2\frac{Precision * Recall}{Precision + Recall}$$

10. **Accuracy (dokładność)** - miara wydajności modeli klasyfikacyjnych, określająca procent poprawnych przewidywań względem wszystkich przewidywań modelu. Mówi ona, jak często model klasyfikuje poprawnie zarówno pozytywne, jak i negatywne przykłady i wyraża się wzorem:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy może być mylące w przypadku niezrównoważonych zbiorów danych, gdzie jedna klasa jest znacznie liczniejsza niż inne. W takich sytuacjach model może osiągać wysoką dokładność, ale niekoniecznie dobrze radzić sobie z klasą mniejszościową. W takich przypadkach lepiej jest użyć innych miar: F1, ballanced accuracy

11. **Ballanced accuracy** - miara wydajności modeli klasyfikacyjnych, szczególnie przydatna w przypadku niezrównoważonych zbiorów danych, gdzie jedna klasa jest znacznie liczniejsza od innych. Miara accuracy (dokładność) może być myląca, gdy jedna klasa dominuje, dlatego balanced accuracy koryguje tę nierówność, biorąc pod uwagę dokładność dla każdej klasy oddzielnie. Dla klasyfikacji binarnej wyraża się wzorem:

$$Balanced\ accuracy = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right)$$

12. **Mean average precision (mAP)** - miara wydajności modeli wykrywania obiektów i segmentacji obrazów, często używana w zadaniach związanych z wizją komputerową. Ocena ta jest kombinacją dwóch aspektów: **precyzji** (ang. precision) oraz **czułości** (ang. recall), gdzie precyzja mierzy, jaki odsetek wykryć jest prawidłowy, a czułość określa, jaki odsetek rzeczywistych obiektów został wykryty. mAP oblicza się poprzez znalezienie średniej precyzji (average precision - AP) dla każdej klasy i następnie uśrednienie jej dla zadanej liczby klas zgodnie z wzorem:

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

Gdzie:

n – liczba klas,

AP_k – Średnia precyzja dla klasy k

Average Precision (AP) to miara, która ocenia jakość modelu wykrywania obiektów na podstawie krzywej precyzji i czułości. Jest to zintegrowana wartość z tej krzywej, która mierzy, jak dobrze model wykrywa obiekty, biorąc pod uwagę różne progi decyzyjne (próg decyzyjny oznacza wartość predykcji powyżej, której model przydzielił próbkę do danej klasy – zazwyczaj 0.5). *AP* ocenia zarówno liczbę poprawnych wykryć (precyzja), jak i zdolność modelu do wykrywania wszystkich obiektów (czułość). Wartość AP można obliczyć jako pole pod krzywą precyzji i czułości (PR-AUC). Dla zestawu progów decyzyjnych czułości *R* i odpowiadających im precyzji *P*, *AP* jest definiowane jako:

$$AP = \int_0^1 P(R)\, dR$$

W praktyce, zamiast ciągłej krzywej, oblicza się sumę wartości precyzji dla różnych wartości czułości. *AP* jest uśrednianą wartością precyzji dla różnych poziomów czułości, zwykle mierzonych w punktach z zestawu progów decyzyjnych.

Jakość modeli segmentacji obiektów podaje w połączeniu mAP i IoU - mAP@IoU, podając średnią precyzję modelu obliczoną przy zastosowaniu progu Intersection over Union (IoU). Przykładowo **mAP@0.50 = 0.77** oznacza, że model uzyskał mAP równe 0.77 przy założeniu IoU o wartości 0.50 i wyższej. Próg 0.50 oznacza, że wykrycie obiektu przez model zostanie uznane za poprawne, jeśli obszar predykcji pokrywa co najmniej 50% rzeczywistego obszaru obiektu.

13. **RANSAC (Random Sample Consensus)** - algorytm, służący do dopasowania modelu do danych z dużą ilością szumów lub wartości odstających (outliers). Jest powszechnie używany w takich zadaniach jak estymacja parametrów modelu, dopasowanie linii, płaszczyzn, czy homografii w obrazach. RANSAC działa iteracyjnie i polega na próbkowaniu danych w celu znalezienia najlepszego modelu. Kolejne kroki algorytmu są następujące:
    1. Losowy wybór próbek.

       Losowo wybierana jest minimalna liczba punktów danych potrzebnych do estymacji parametrów modelu. Dla dopasowania linii wystarczą dwa punkty, dla dopasowania płaszczyzny trzy punkty itp.

    2. Dopasowanie modelu.

       Na podstawie wybranych punktów estymowane są parametry modelu. Model ten jest dopasowywany wyłącznie na podstawie losowo wybranych próbek.

3. Ocena zgodności modelu (Consensus).
   Sprawdzane jest, ile punktów z całego zbioru danych pasuje do tego modelu, czyli znajduje się "blisko" modelu (zgodnie z ustalonym progiem tolerancji na odległość). Punkty te są nazywane inlierami, a punkty, które nie pasują, są outlierami.

4. Iteracyjne powtarzanie kroków 1-3.
   Kroki 1-3 są powtarzane wielokrotnie (przez ustaloną liczbę iteracji). Za każdym razem algorytm losuje nowy zestaw próbek i dopasowuje model.

5. Wybór najlepszego modelu.
   Spośród wszystkich iteracji wybierany jest model, który ma największą liczbę inlierów (punktów zgodnych z modelem). Ten model uznaje się za ostateczne rozwiązanie.

6. Estymacja końcowa.
   Na podstawie wszystkich inlierów wybranego modelu, przeprowadza się ostateczne dopasowanie modelu.

14. **SHAP** - technika wyjaśnialności modeli uczenia maszynowego, która opiera się na wartościach Shapleya. Wartości te pochodzą z teorii gier kooperacyjnych i służą do wyjaśnienia wpływu poszczególnych zmiennych na wynik modelu predykcyjnego. Główna idea polega na obliczeniu, jak duży wpływ na ostateczną predykcję miałaby każda z nich, gdybyśmy ją stopniowo dodawali do modelu.

    SHAP tworzy wszystkie możliwe kombinacje cech i oblicza zmiany w przewidywanej wartości, gdy dana cecha jest dodawana do podzbioru innych cech. Na podstawie tych kombinacji oblicza, jak bardzo dana cecha przyczynia się do zmiany wyniku predykcji.

    Wartości Shapleya są idealne do wyjaśniania modeli uczenia maszynowego, ponieważ odpowiadają na pytanie: Jaki jest wpływ każdej cechy na wynik predykcji? SHAP łączy te wartości z metodami interpretacji modeli i pozwala na wyjaśnianie złożonych modeli takich jak XGBoost, SVM i sieci neuronowe.

    Główną wadą SHAP jest wysoki koszt obliczeniowy zwłaszcza dla dużych zbiorów danych i modeli o wielu cechach. Jednak dostępne są przybliżone metody, takie jak Kernel SHAP i Tree SHAP, które znacząco obniżają koszt obliczeniowy.

7. Bibliografia

1. Lung-Ming Chen, Yi-Hsiang Pan, Yung-Jen Chen. *A Study of Shuttlecock's Trajectory in Badminton*. Journal of Sports Science and Medicine 2009, (08), 657 – 662.
2. https://www.youtube.com/watch?v=BRoGLwpfwns [dostęp 30.09.2024].
3. https://youtu.be/pCGgT_PJkcs [dostęp 30.09.2024].
4. https://www.youtube.com/watch?v=7C56kivaOV4 [dostęp 30.09.2024]
5. Primo, L.; Gutiérrez-Suárez, A.; Gómez, M.Á. *Analysis of challenge request success according to contextual variables in elite badminton*. Ger. J. Exerc. Spor.t Res. 2019, 49, 259–265.
6. Kopania, M.; Nowisz, J.; Przelaskowski, A. *Automatic Shuttlecock Fall Detection System in or out of a Court in Badminton Games—Challenges, Problems, and Solutions from a Practical Point of View*. Sensors 2022, 22, 8098. https://doi.org/10.3390/s22218098
7. Chen, Wei & Liao, Tingbo & Li, Zhihang & Lin, Haozhi & Xue, Hong & Zhang, Le & Guo, Jing & Cao, Zhiguang. (2019). *Using FTOC to Track Shuttlecock for the Badminton Robot*. Neurocomputing. 334. 182-196. 10.1016/j.neucom.2019.01.023.
8. Thomas, G.; Gade, R.; Moeslund, T.B.; Carr, P.; Hilton, A. *Computer Vision for Sports: Current Applications and Research Topics*. Comput. Vis. Image Underst. 2017, 159, 3–18.
9. Moeslund, T.B.; Thomas, G.; Hilton, A. *Computer Vision in Sports*; (Electronic) Advances in Computer Vision and Pattern Recognition; Springer: Berlin/Heidelberg, Germany, 2014.
10. Zhiguang Cao, Tingbo Liao, Wen Song, Zhenghua Chen, Chongshou. *Detecting the shuttlecock for a badminton robot: A YOLO based approach*. Expert Systems With Applications 2021; 164 113833
11. https://www.youtube.com/watch?v=C1nebaWyemk&t=1768s [dostęp 30.09.2024]
12. https://youtu.be/C1nebaWyemk?t=9700 [dostęp 30.09.2024]
13. https://www.imarcgroup.com/sports-analytics-market [dostęp 30.09.2024]
14. Naik, B.T.; Hashmi, M.F.; Bokde, N.D. *A Comprehensive Review of Computer Vision in Sports: Open Issues, Future Trends and Research Directions*. *Appl. Sci.* 2022, *12*, 4429. https://doi.org/10.3390/app12094429

15. Baptiste Darbois Texier, Caroline Cohen, David Quéré, Christophe Claneta, *Shuttlecock dynamics*, Procedia Engineering, Volume 34, 2012; Pages 176-181, ISSN 1877-7058, https://doi.org/10.1016/j.proeng.2012.04.031.
16. Chen, L., Pan, Y., & Chen, Y. *A Study of Shuttlecock's Trajectory in Badminton*. Journal of sports science & medicine 2009; 8 4, 657-62.
17. https://www.hawkeyeinnovations.com [dostęp 30.09.2024]
18. Caroline Cohen1, Baptiste Darbois Texier1, David Quéré2 and Christophe Clanet1. *The physics of badminton.* Published 1 June 2015. IOP Publishing Ltd and Deutsche Physikalische Gesellschaft New Journal of Physics, Volume 17, June 2015
19. Yusup, U.; Rusdiana, A. *The Analysis of Shuttlecock Velocity Based on the Field Test Method of The Home Industry Products in Indonesia*. J. Pendidik. Jasm. Dan Olahraga 2020, 5, 10–16.
20. Nyström, Marcus ; Andersson, Richard ; Niehorster, Diederick C. ; Hessels, Roy S. ; Hooge, Ignace T. C. *What is a blink? Classifying and characterizing blinks in eye openness signals*. Behavior research methods, 2024, Vol.56 (4), p.3280-3299
21. https://www.youtube.com/watch?v=C1nebaWyemk&t=8458s [dostęp 30.09.2024]
22. https://extranet.bwf.sport/docs/document-system/81/1466/1470/Section%204.1%20-%20Laws%20of%20Badminton%20-%2029%20May%202023%20V2.1.pdf, https://corporate.bwfbadminton.com/statutes/ [dostęp 30.09.2024]
23. https://extranet.bwf.sport/docs/document-system/81/1466/1471/Section%205.3.4%20-%20Specs%20for%20Int_l%20Standard%20Facilities%20-%2011%20November%202023%20V2.1.pdf [dostęp 30.09.2024]
24. Kuldeep Singh, Rajiv Kapoor, *Image enhancement using Exposure based Sub Image Histogram Equalization, Pattern Recognition Letters*, Volume 36, 2014, Pages 10-14, ISSN 0167-8655, https://doi.org/10.1016/j.patrec.2013.08.024.
25. Nowisz, J.; Kopania, M.; Przelaskowski, A. *Realtime flicker removal for fast video streaming and detection of moving objects*. Multimed. Tools Appl. 2021, 80, 14941–14960.
26. https://revisionfx.com/products/deflicker/ [dostęp 30.09.2024]
27. https://digitalanarchy.com/Flicker/main.html [dostęp 30.09.2024]
28. Kang Tong, Yiquan Wu, Fei Zhou, *Recent advances in small object detection based on deep learning: A review*, Image and Vision Computing, Volume 97, 2020, 103910, ISSN 0262-8856, https://doi.org/10.1016/j.imavis.2020.103910.

29. Kalman, R.E. *A New Approach to Linear Filtering and Prediction Problems*. Trans. ASME—J. Basic Eng. 1960, 82, 35–45.
30. Isard, M., Blake, A.: *Condensation - Conditional Density Propagation for Visual Tracking*, Int. J. of Computer Vision, vol. 29, no. 1, pp. 5–28, 1998
31. J. Henriques, R. Caseiro, P. Martins and J. Batista, *High-Speed Tracking with Kernelized Correlation Filters* in IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 37, no. 03, pp. 583-596, 2015.; doi: 10.1109/TPAMI.2014.2345390. https://arxiv.org/pdf/1404.7584
32. Blostein, S.D.; Huang, T.S. *Detecting Small, Moving Objects in Image Sequences Using Sequential Hypothesis Testing*. IEEE Trans. Signal Process. 1991, 39, 1611–1629.
33. Archana, M.; Geetha, M.K. *Object Detection and Tracking Based on Trajectory in Broadcast Tennis Video*. Procedia Comput. Sci. 2015, 58, 225–232.
34. Zhu, M.; Wang, H. *Fast Detection of Moving Object Based on Improved Frame-Difference Method*. In Proceedings of the 2017 6th International Conference on Computer Science and Network Technology, ICCSNT, Dalian, China, 21–22 October 2017; Volume 2018, pp. 299–303.
35. Yin, Q.; Hu, Q.; Liu, H.; Zhang, F.; Wang, Y.; Lin, Z.; An, W.; Guo, Y. *Detecting and Tracking Small and Dense Moving Objects in Satellite Videos: A Benchmark*. IEEE Trans. Geosci. Remote Sens. 2022, 60, 1–18.
36. Zhou, Y.; Maskell, S. *Detecting and Tracking Small Moving Objects in Wide Area Motion Imagery (WAMI) Using Convolutional Neural Networks (CNNs)*. In Proceedings of the FUSION 2019 22nd International Conference on Information Fusion, Ottawa, ON, Canada, 2–5 July 2019.
37. Ren, Shaoqing; He, Kaiming; Girshick, Ross; Sun, Jian (2017-06-01). *"Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks"*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 39 (6): 1137–1149. arXiv:1506.01497. doi:10.1109/TPAMI.2016.2577031. ISSN 0162-8828.
38. Aguilar, C.; Ortner, M.; Zerubia, J. *Small Object Detection and Tracking in Satellite Videos With Motion Informed-CNN and GM-PHD Filter*. Front. Signal Process. 2022, 2, 827160.
39. Dalal, N. and Triggs, B., *"Histograms of Oriented Gradients for Human Detection,"* IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, San Diego, CA, USA.

40. Paul Viola and Michael J. Jones. *Rapid object detection using a boosted cascade of simple features*. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I–511. IEEE, 2001.
41. D. G. Lowe, *"Object recognition from local scale-invariant features,"* Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1150-1157 vol.2, doi: 10.1109/ICCV.1999.790410.
42. Ahmadi, K.; Salari, E. *Small Dim Object Tracking Using Frequency and Spatial Domain Information*. Pattern Recognit. 2016, 58, 227–234.
43. Cortes, C., Vapnik, V. *Support-vector networks*. Mach Learn 20, 273–297 (1995). https://doi.org/10.1007/BF00994018
44. Son, S.; Kwon, J.; Kim, H.Y.; Choi, H. *Tiny Drone Tracking Framework Using Multiple Trackers and Kalman-Based Predictor*. J. Web Eng. 2021, 20, 2391–2412.
45. Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). *"ImageNet classification with deep convolutional neural networks"* (PDF). Communications of the ACM. 60 (6): 84–90.
46. https://image-net.org/challenges/LSVRC/2012/results.html [dostęp 30.09.2024]
47. https://image-net.org/ [dostęp 30.09.2024]
48. Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. https://doi.org/10.48550/arXiv.2004.10934
49. Huang, Jonathan & Rathod, Vivek & Sun, Chen & Zhu, Menglong & Korattikara, Anoop & Fathi, Alireza & Fischer, Ian & Wojna, Zbigniew & Song, Yang & Guadarrama, Sergio & Murphy, Kevin. (2016). *Speed/accuracy trade-offs for modern convolutional object detectors*. CVPR 2017, https://doi.org/10.48550/arXiv.1611.10012
50. C. -Y. Wang, A. Bochkovskiy and H. -Y. M. Liao, *"YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors"*. 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 2023, pp. 7464-7475, doi: 10.1109/CVPR52729.2023.00721
51. https://www.votchallenge.net/ [dostęp 30.09.2024]
52. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W., Dollár, P., & Girshick, R. (2023). *Segment Anything*. ArXiv. /abs/2304.02643 https://arxiv.org/abs/2304.02643

53. Ke, Lei and Ye, Mingqiao and Danelljan, Martin and Liu, Yifan and Tai, Yu-Wing and Tang, Chi-Keung and Yu, Fisher *Segment Anything in High Quality*, NeurIPS, 2023. https://github.com/SysCV/sam-hq
54. Li, X., Miao, D., He, Z., Wang, Y., Lu, H., Yang, M.H. *Learning spatial-semantic features for robust video object segmentation*. arXiv preprint arXiv:2407.07760 (2024)
55. https://data.votchallenge.net/vots2023/measures.pdf [dostęp 30.09.2024]
56. H. K. Cheng and A. G. Schwing. Xmem. *Long-term video object segmentation with an atkinson-shiffrin memory model*. In ECCV, 2022
57. Dosovitskiy, Alexey; Beyer, Lucas; Kolesnikov, Alexander; Weissenborn, Dirk; Zhai, Xiaohua; Unterthiner, Thomas; Dehghani, Mostafa; Minderer, Matthias; Heigold, Georg; Gelly, Sylvain; Uszkoreit, Jakob (2021-06-03). *"An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale"*. arXiv: 2010.11929.
58. Henriques, Joao & Caseiro, Rui & Martins, Pedro & Batista, Jorge. (2014). *High-Speed Tracking with Kernelized Correlation Filters*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 37. 10.1109/TPAMI.2014.2345390. https://arxiv.org/pdf/1404.7584
59. Z. Kalal, K. Mikolajczyk and J. Matas, *"Forward-Backward Error: Automatic Detection of Tracking Failures"*, 2010 20th International Conference on Pattern Recognition, Istanbul, Turkey, 2010, pp. 2756-2759, doi: 10.1109/ICPR.2010.675. https://kahlan.eps.surrey.ac.uk/featurespace/tld/Publications/2010_icpr.pdf
60. B. D. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision*. Proceedings of Imaging Understanding Workshop, pages 121--130
61. N. Wojke, A. Bewley and D. Paulus, *"Simple online and realtime tracking with a deep association metric"*, 2017 IEEE International Conference on Image Processing (ICIP), Beijing, China, 2017, pp. 3645-3649, doi: 10.1109/ICIP.2017.8296962. https://arxiv.org/pdf/1703.07402.pdf
62. Harold W. Kuhn, *"The Hungarian Method for the assignment problem"*, Naval Research Logistics Quarterly, 2: 83–97, 1955.
63. Held, David & Thrun, Sebastian & Savarese, Silvio. (2016). *Learning to Track at 100 FPS with Deep Regression Networks*. 10.48550/arXiv.1604.01802. https://arxiv.org/abs/1604.01802

64. Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, Xinggang Wang. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box* arXiv:2110.06864. https://doi.org/10.48550/arXiv.2110.06864
65. A.W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. *Visual tracking: An experimental survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(7):1442–1468, July 2014
66. Y. Wu, J. Lim, and M.-H. Yang. *Online object tracking: A benchmark*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.
67. Rozumnyi, Denys & Kotera, Jan & Sroubek, Filip & Novotny, Lukas & Matas, Jiri. (2017). *The World of Fast Moving Objects*. Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition.
68. Kamble, P.R., Keskar, A.G. & Bhurchandi, K.M. *Ball tracking in sports: a survey*. Artif Intell Rev 52, 1655–1705 (2019). https://doi.org/10.1007/s10462-017-9582-2
69. Xinguo Yu, Hon Wai Leong, Qi Tian *Trajectory-Based Ball Detection and Tracking in Broadcast Soccer Video* IEEE TRANSACTIONS ON MULTIMEDIA, 2006; VOL. 8, NO. 6, DECEMBER 2006
70. Pingali GS, Jean Y, Carlbom I. *Real time tracking for enhanced tennis broadcasts*. In: CVPR, 1998; pp 260–265
71. Zhou, Xiangzeng, Lei Xie, Qiang Huang, Stephen J. Cox, and Yanning Zhang. *"Tennis ball tracking using a two-layered data association approach"*. IEEE Transactions on Multimedia 17, no. 2 (2014): 145-156.
72. Reno, Vito, Nicola Mosca, Roberto Marani, Massimiliano Nitti, Tiziana D'Orazio, and Ettore Stella. *"Convolutional neural networks based ball detection in tennis games"*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1758-1764. 2018.
73. Polk, Tom, DominikJäckle, Johannes Häußler, and Jing Yang. *"CourtTime: Generating actionable insights into tennis matches using visual analytics"*. IEEE Transactions on Visualization and Computer Graphics 26, no. 1 (2019): 397-406.
74. Shishido, Hidehiko *"Visual Tracking Method of a Quick and Anomalously Moving Badminton Shuttlecock"*. ITE Trans. on MTA 2017; Vol. 5, No. 3, pp. 110-120
75. Zhishen Huang and Wai-kuen Cham, *A Novel Algorithm For Shuttlecock Tracking*, Proceedings of APSIPA Annual Summit and Conference, 2015

76. Y. Freund, R. Schapire, *"A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting"*, 1995.
77. Hiemann, A.; Kautz, T.; Zottmann, T.; Hlawitschka, M. *Enhancement of Speed and Accuracy Trade-Off for Sports Ball Detection in Videos—Finding Fast Moving, Small Objects in Real Time*. Sensors 2021, 21, 3214.
78. Leong, L.H.; Zulkifley, M.A.; Hussain, A.B. *Computer vision approach to automatic linesman*. In Proceedings of the IEEE 10th International Colloquium on Signal Processing and Its Applications, Kuala Lumpur, Malaysia, 7–9 March 2014; pp. 212–215.
79. T. Dierickx, *"Badminton game analysis from video sequences"*, Master's thesis, Universiteit Gent. Faculteit Ingenieurswetenschappen en Architectuur., 2014
80. D'Orazio, T., Guaragnella, C., Leo, M., & Distante, A. (2004). *A new algorithm for ball recognition using circle Hough transform and neural classifier*. Pattern Recognition, 37(3), 393–408. doi:10.1016/s0031-3203(03)00228-0
81. Duda, R. O., and Hart, P. E. *"Use of the Hough Transformation to Detect Lines and Curves in Pictures"*. Communications of the ACM, Vol. 15, No. 1, 1972, pp. 11–15.
82. Zandycke, Gabriel & Vleeschouwer, Christophe. (2019). *Real-time CNN-based Segmentation Architecture for Ball Detection in a Single View Setup*. 51-58. 10.1145/3347318.3355517.
83. Otsu, N. (1979). *A threshold selection method from gray-level histograms*
84. Canny, J. (1986). *"A computational approach to edge detection"*. IEEE Transactions on pattern analysis and machine intelligence. (6): 679– 698
85. Rong, W., Z. Li, W. Zhang, and L. Sun. (2014). *"An improved CANNY edge detection algorithm"*. In: 2014 IEEE international conference on mechatronics and automation. IEEE. 577–582.
86. Adams, R. and L. Bischof. (1994). *"Seeded region growing"*. IEEE Transactions on pattern analysis and machine intelligence. 16(6): 641–647.
87. Ramli, M. F. and K. N. Tahar. (2020). *"Homogeneous tree height derivation from tree crown delineation using Seeded Region Growing (SRG) segmentation"*. Geo-spatial Information Science. 23(3): 195– 208.
88. Moore, A. P., S. J. Prince, J. Warrell, U. Mohammed, and G. Jones. (2008). *"Superpixel lattices"*. In: 2008 IEEE conference on computer vision and pattern recognition. IEEE. 1–8.

89. Liu, M.-Y., O. Tuzel, S. Ramalingam, and R. Chellappa. (2011b). *"Entropy rate superpixel segmentation"*. In: CVPR 2011. IEEE. 2097– 2104
90. Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. (2004). "GrabCut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. 23, 3 (August 2004), 309–314. https://doi.org/10.1145/1015706.1015720
91. Kass, M., A. Witkin, and D. Terzopoulos. (1988). *"Snakes: Active contour models"*. International journal of computer vision. 1(4): 321–331.
92. Chen, X., B. M. Williams, S. R. Vallabhaneni, G. Czanner, R. Williams, and Y. Zheng. (2019c). *"Learning active contour models for medical image segmentation"*. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 11632–11640.
93. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. *Mask R-CNN*. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2961–2969.
94. Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alex Berg, Wan-Yen Lo, Piotr Dollar, Ross Girshick, *Segment Anything*, arXiv:2304.02643. https://github.com/facebookresearch/segment-anything.
95. https://www.statsperform.com [dostęp 30.09.2024]
96. https://tracab.com [dostęp 30.09.2024]
97. https://www.researchgate.net/publication/339921713_Analysing_time_pressure_in_professional_tennis/fulltext/5e6c2fdc458515e555795066/Analysing-time-pressure-in-professional-tennis.pdf [dostęp 30.09.2024]
98. https://tracab.com/products/tracab-technologies/tracab-optical [dostęp 30.09.2024]
99. Linke D, Link D, Lames M (2020). *Football-specific validity of TRACAB's optical video tracking systems*. PLoS ONE 15(3): e0230179. https://doi.org/10.1371/journal.pone.0230179
100. https://eu.usatoday.com/story/sports/tennis/2022/07/06/hawkeye-controversy-wimbledon-doubles-team-halts-play-after-line-call/7820717001/ [dostęp 30.09.2024]
101. https://www.youtube.com/watch?v=cMq5hgqDnEQ [dostęp 30.09.2024]
102. https://bwfworldtour.bwfbadminton.com/news-single/2021/11/21/bwf-infront-pan-asia-and-hawk-eye-statement [dostęp 30.09.2024]
103. https://www.youtube.com/watch?v=1c7dR76n3vg [dostęp 30.09.2024]

104. Kovalchik, S.; Jeff, S.; Reid, M. *Player, official or machine?: Uses of the challenge system in professional tennis*. Int. J. Perform. Anal. Sport. 2017, 17, 1–9.
105. https://www.baslerweb.com/en/shop/aca800-200gm/ [dostęp 30.09.2024]
106. https://docs.baslerweb.com/free-run-image-acquisition [dostęp 30.09.2024]
107. https://docs.baslerweb.com/acquisition-timing-information [dostęp 30.09.2024]
108. M. Hubbard, A. Cooke, *Spin dynamics of the badminton shuttlecock*, 6th International Symposium on Computer Simulation in Biomechanics (1997), pp. 42-43
109. https://www.youtube.com/watch?v=xTYsQTHLHk8 [dostęp 30.09.2024]
110. Jack Sklansky. *Finding the convex hull of a simple polygon*. Pattern Recognition Letters, 1(2):79–83, 1982.
111. Joseph Redmon, Ali Farhadi, *"YOLOv3: An Incremental Improvement"*, arXiv:1804.02767, https://doi.org/10.48550/arXiv.1804.02767
112. https://scikit-learn.org/ [dostęp 30.09.2024]
113. Chao Chen, Andy Liaw, Leo Breiman, and others. *Using random forest to learn imbalanced data*. University of California, Berkeley, 110(1-12):24, 2004.
114. M. A. Fischler and R. C. Bolles (1981). *"Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"*. Comm. of the ACM. 24: 381--395. doi:10.1145/358669.358692
115. Lundberg, Scott M and Lee, Su-In, *A Unified Approach to Interpreting Model Prediction*, Advances in Neural Information Processing Systems 30 w I. Guyon and U. V. Luxburg and S. Bengio and H. Wallach and R. Fergus and S. Vishwanathan and R. Garnett, rok 2017, strony 4765--4774 https://papers.nips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf
116. Primo, L.; Gutiérrez-Suárez, A.; Gómez, M.Á. *Analysis of challenge request success according to contextual variables in elite badminton*. Ger. J. Exerc. Spor.t Res. 2019, 49, 259–265.
117. https://www.itftennis.com/media/7365/elc-evaluation-paper-revision-26.pdf [dostęp 30.09.2024]
118. https://www.youtube.com/watch?v=XhQyVnwBXBs [dostęp 30.09.2024]