Wydział Podstawowych Problemów Techniki
Politechnika Wrocławska

System wspomagający grę w darta z wykorzystaniem analizy obrazu w czasie rzeczywistym

Piotr Andrzejewski
Nr indeksu: 244931

Praca inżynierska napisana
pod kierunkiem
dr. Marcina Zawady

Politechnika
Wrocławska
Wrocław 2020

Spis treści

Wstęp

—tu będzie wstęp—
Wymagania funkcjonalne:

- wykrywanie momentu wbicia rzutki
- obliczanie pozycji wbitej rzutki
- analiza kilku rzutów bez konieczności wyciągania lotek z tarczy
- wyświetlanie informacji dotyczących rzutu w aplikacji mobilnej
- zaznaczanie pozycji rzutki na diagramie tarczy

Wymagania niefunkcjonalne:

- działanie w czasie rzeczywistym, pozwalające na płynną grę
- niski koszt
- duże możliwości konfiguracji

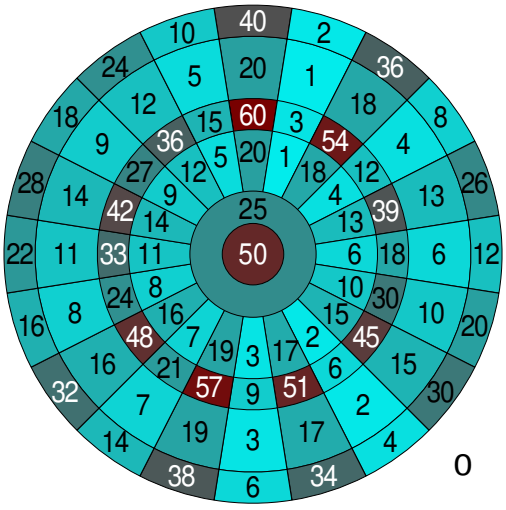
Analiza problemu

W tym rozdziale przeanalizowano problemy zawarte w tematyce niniejszej pracy oraz przedstawiono wszystkie okoliczności, w jakich system będzie funkcjonował. Omówiono podstawy gry w darta, potrzebne do zrozumienia trudności, którym trzeba sprostać przy projektowaniu i implementacji systemu. Następnie głębiej zbadano wymagania funkcjonalne, wymienione we wstępie. Zostały wprowadzone pewne matematyczne modele, które pomogą w reprezentacji rozwiązania. Podano cel stworzenia aplikacji mobilnej i jej role. Dokładnej analizie zostały poddane ograniczenia i wymagania niefunkcjonalne systemu.

2.1 Gra w darta

Dart, znany w Polsce również pod nazwą „rzutki" lub „lotki", jest grą, w której zawodnicy rzucają lotkami do oddalonej od nich tarczy, w celu zdobycia punktów.

Tarcza, zazwyczaj wykonana z sizalu, znajduje się na wysokości 173 cm, w odległości 237 cm od gracza. Ma kształt koła o średnicy 45 cm i składa się z pól, którym przyporządkowane są wartości punktowe. Dzieli się ona na 20 wycinków kołowych, punktowanych od 1 do 20 (pola standardowe). Są one jednak przecięte dwoma pierścieniami: *double* (zewnętrznym) i *treble* (wewnętrznym). Trafienie w zewnętrzny pierścień oznacza podwojenie liczby punktów przypisanych do danego pola standardowego, a wewnętrzny – potrojenie. Na tarczy znajdują się także segmenty *inner bull*, czyli koło znajdujące się w środku, dające 50 punktów oraz *outer bull* – pierścień okalający środkowe koło, za którego trafienie przyznaje się 25 punktów. Rzut poza wymienione pola jest liczony za zero punktów. Tarcza wraz z liczbą punktów przypisaną do każdego z pól jest pokazana na rysunku 2.1.



Rysunek 2.1: Tarcza do darta z przypisaniem punktów
Źródło: Cmglee, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=49462410

Najpopularniejszym wariantem darta jest *501*, w którym gracze rozpoczynają rozgrywkę, mając na koncie

501 punktów. Każdy z nich wykonuje w jednej turze 3 rzuty, a suma zdobytych punktów jest odejmowana od stanu ich konta. Wygrywa ten, kto jako pierwszy osiągnie dokładnie 0 punktów. Innym znanym wariantem jest *Round The Clock* – gracze mają za zadanie trafić w każde pole na tarczy z zakresu od 1 do 20 w kolejności zgodnej z ruchem wskazówek zegara, a na końcu w pole *inner bull* (tj. 20, 1, 18, ..., 12, 5, 50).

2.2 Wykrycie rzutu i przypisanie do segmentu

Głównym problemem, jakim zajmuje się niniejsza praca, jest wykrycie rzutu przez system, a następnie uzyskanie segmentu, w jaki trafiła lotka. Po włączeniu systemu powinien on nieustannie analizować dane wejściowe i wysyłać sygnał, gdy rozpozna, że rzutka wbiła się w tarczę. Sygnał ten rozpocznie procesowanie danych wejściowych pod kątem obliczenia konkretnej pozycji rzutki. Najczęstszy scenariusz gry w darta wygląda tak, że każdy z graczy rzuca kolejno 3 razy, wyciąga swoje rzutki i tura przechodzi na kolejnego gracza. Z tego powodu do wygody użytkowania potrzebna jest obsługa kilku rzutów (przynajmniej trzech) bez wyciągania rzutki po każdym rzucie. Idealną sytuacją byłaby taka, gdyby bez interwencji użytkownika system był w stanie reagować na poprawne rzuty, ale ignorować wyciąganie lotek z tarczy pomiędzy turami.

Docelową daną wyjściową, którą ma dostarczać system, jest numer jednego z segmentów tarczy. W tym wypadku, zbiorem możliwych rozwiązań byłby

$$W = \{0, 25, 50\} \cup A \cup B \cup C, \text{ gdzie } A = \{1, \ldots, 20\}, B = \{2 \cdot a : a \in A\}, C = \{3 \cdot a : a \in A\}.$$

Trzeba jednak zauważyć, że taki sposób reprezentacji powodowałby stratę pewnych informacji. Mianowicie, dla każdego $w \in (A \cap B) \cup (A \cap C) \cup (B \cap C)$ nie można by było jednoznacznie stwierdzić, jaki segment na tarczy reprezentuje. Przykładowo, pole numer 14 z podwójnego pierścienia (podwojona siódemka), byłoby tak samo przedstawiane jak standardowe pole 14. Należy zatem doprecyzować sposób reprezentacji rzutu: każdy rzut jest przedstawiany jako para (w, t), gdzie $w \in W$ to liczba punktów za rzut, a $t \in \{S, D, T\}$. S oznacza pole standardowe ($\in A \cup \{0, 25, 50\}$), D – pole podwójne (*double*, $\in B$), T – pole potrójne (*triple*, $\in C$). Dzięki temu unika się wyżej przedstawionej niejednoznaczności: jeden segment oznacza się jako $(14, D)$, a drugi jako $(14, S)$.

Dodatkowo, przydatne może okazać się zapisanie również pośredniej reprezentacji pozycji rzutki, nie jako konkretnego pola na tarczy, ale dokładnego punktu wbicia. Nie każde rozwiązanie wymaga posiadania takich informacji, ale większość prawdopodobnie będzie je musiała wyliczyć. W takiej sytuacji punkt można przedstawiać za pomocą:

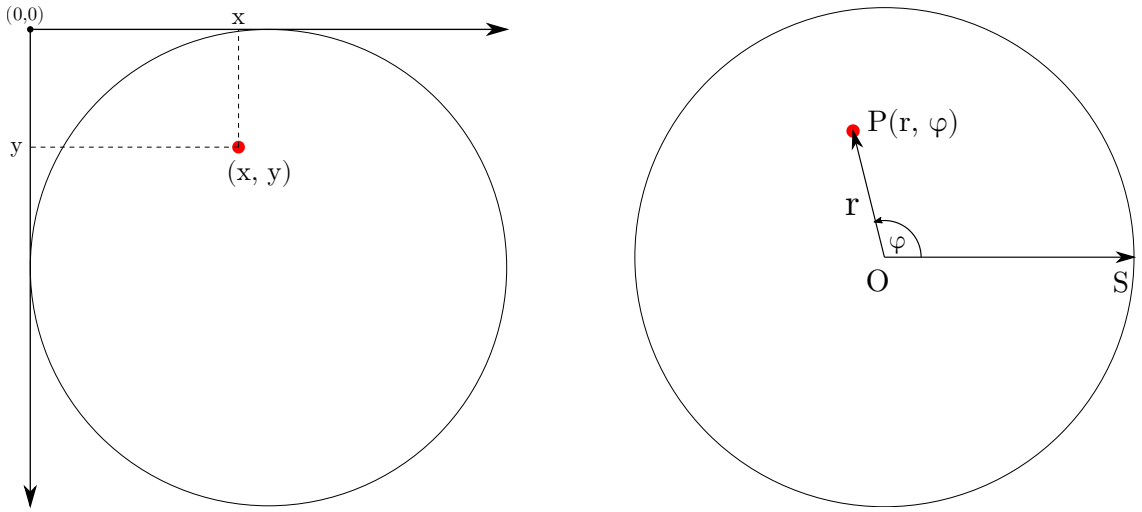
(a) współrzędnych kartezjańskich, tj. (x, y), gdzie punkt $(0, 0)$ to np. lewy górny róg kwadratu opisanego na okręgu tarczy

(b) współrzędnych biegunowych, tj. (r, φ), gdzie biegunem O jest środek tarczy, a oś biegunowa OS przechodzi przez punkt S – najbardziej wysunięty na prawo punkt tarczy

Oba podejścia są pokazane na rysunku 2.2.

2.3 Aplikacja mobilna

Dodatkową częścią systemu jest aplikacja mobilna, która ma na celu atrakcyjną prezentację danych użytkownikowi, urozmaicenie rozgrywki oraz poprawę użyteczności (*user experience*). W wersji podstawowej, będzie ona przede wszystkim stanowić interfejs graficzny dla systemu wbudowanego. Dzięki aplikacji mobilnej, będzie można oglądać wyniki poszczególnych rzutów oraz oglądać miejsca wbicia rzutki na wirtualnej tarczy. W rozbudowanej wersji, można do aplikacji dodać wiele innych funkcjonalności, np. wpływanie na stan gry lub wyświetlanie statystyk.



(a) współrzędne kartezjańskie (b) współrzędne biegunowe

Rysunek 2.2: Rodzaje reprezentacji pozycji rzutki.
Źródło: Opracowanie własne.

2.4 Ograniczenia

Podczas tworzenia systemu należy pamiętać o ograniczeniach, które należy spełnić, by osiągnąć wszystkie wymagania, również te niefunkcjonalne. Oczywistym ograniczeniem, a jednocześnie wyznacznikiem jakości jest dokładność analizy rzutu. Może być wyrażona w różny sposób, ale ma na celu ukazanie, jak duże i/lub częste są błędy w określaniu pozycji rzutki. Bez względu na rozwiązanie, dokładność może wahać się i zależeć od wielu czynników, jednakże uśrednione dane dają natychmiastową informację o tym, na ile użyteczny jest to moduł. Głównym celem niniejszej pracy nie jest maksymalizowanie dokładności, lecz powinna być ona na poziomie pozwalającym na podstawową rozgrywkę.

Innym, istotnym wskaźnikiem działania systemu jest szybkość jego działania. Jednym z celów jest możność opisania go jako *system czasu rzeczywistego*. Oznacza to, że jego poprawność nie zależy jedynie od dokładności zwracanych wyników, lecz również od czasu, jaki upłynął od nadejścia danych wejściowych do otrzymania wyniku. Sytuacja z analizą rzutów odzwierciedla taką sytuację – gracz nie chce rzucać wielu rzutek, a następnie, po czasie, zobaczyć statystyki rzutów. Aplikacja powinna działać na zasadzie akcja-reakcja: po każdym z rzutów, w krótkim czasie, powinna się pojawić informacja o wyniku. Efektem pożądanym jest sytuacja, gdy gracz nie będzie musiał czekać z następnym rzutem na zakończenie analizy poprzedniego. W przypadku, gdy oczekiwanie na wynik będzie zbyt długie, system stanie się bezużyteczny, nawet jeśli rezultaty będą bardzo dokładne. Należy zachować balans pomiędzy dokładnością a szybkością działania programu.

Rozwiązanie powinno być łatwo i szeroko konfigurowalne. Z pewnością pojawią się w nim parametry specyficzne dla np. konkretnej tarczy lub rzutek. Wszystkie tego typu części należy dobrze udokumentować oraz zapewnić prostotę modyfikacji, tak by jeden kod źródłowy mógł być z łatwością używany przez wiele osób, nawet jeśli ich lokalne parametry nie pokrywają się z tymi, które występowały u autora rozwiązania.

Ostatnią, bardziej pragmatyczną restrykcją jest kwestia ceny całego przedsięwzięcia. Tym, co ma wyróżniać niniejsze rozwiązanie od innych jest fakt, że będzie wykonane przy użyciu łatwo dostępnych i możliwie tanich komponentów.

Teoretyczne rozwiązanie problemu

Ten rozdział ma za zadanie przedstawić teoretyczną (matematyczną) część rozwiązania głównego problemu – uzyskania pozycji rzutki. Na początku zaprezentowany i podzielony na części został ogólny pomysł rozwiązania. Pokazane zostały również alternatywne propozycje podejścia do problemu, wraz z uzasadnieniem, dlaczego nie zostały one wybrane. Następnie omówiono podstawowe terminy i zagadnienia, których znajomość potrzebna jest do pełnego zrozumienia dalszych tematów. Główną częścią rozdziału jest prezentacja wszystkich kluczowych obliczeń i operacji, które zachodzą podczas analizy rzutu, krok po kroku. Detale dotyczące niektórych algorytmów zostaną przedstawione w późniejszych rozdziałach, w tym zaś uznaje się je za tzw. czarne skrzynki (ang. *black box*), to znaczy elementy programu, które przyjmują pewne dane wejściowe i zwracają wyniki, lecz ich implementacja nie jest rozważana.

3.1 Idea rozwiązania

Pierwszą decyzją, jaką należało podjąć podczas tworzenia rozwiązania, było ustalenie, na podstawie jakich danych wejściowych system powinien rozpoznawać rzut. Postawiono na ten sam typ danych, którym człowiek posługuje się przy ocenie rzutu, czyli obraz. W systemie rolę wzroku pełnią kamery, które co określony czas dostarczają zdjęcia tarczy. Dzięki ciągłemu porównywaniu następujących po sobie klatek, system wykrywa rzut i rozpoczyna przetwarzanie obrazu z wbitą lotką. Do określenia dokładnej pozycji rzutki wykorzystano metodę triangulacji. Pozwala ona na obliczenie dystansu do obiektu za pomocą zdjęć z dwóch kamer. Tarcza i kamery są zamocowane w stelażu o znanych wymiarach. Kamery w ustalonej odległości od siebie, a ich soczewki są ustawione prostopadle do powierzchni tarczy. Następnie, po zmianie układu współrzędnych, dzięki znanym dokładnym wymiarom tarczy, pozycja rzutki jest przyporządkowywana do konkretnego segmentu punktowego.

Pomysłem również opartym na przetwarzaniu obrazu, jest próba uzyskania miejsca wbicia rzutki z użyciem tylko jednej kamery, która jest umiejscowiona przed tarczą, naprzeciw jej. Obniżyłoby to koszty i ułatwiło obliczenia, jednak byłoby niepraktyczne, ponieważ to gracz musi stać na wprost tarczy, a kamera znajdowałaby się pomiędzy nim a tarczą, w środku strefy lotu. Cechowałoby się to także małą przenoszalnością, trudno byłoby zawsze ustawić kamerę idealnie na wysokości środka tarczy. Gdyby jednak kamera była przesunięta względem półprostej prostopadłej do płaszczyzny tarczy, przechodzącej przez jej środek, to przetwarzanie obrazu i obliczenia mocno by się utrudniły, gdyż kamera patrzyłaby pod pewnym kątem na tarczę. Trudniejsze byłoby również mocowanie kamery.

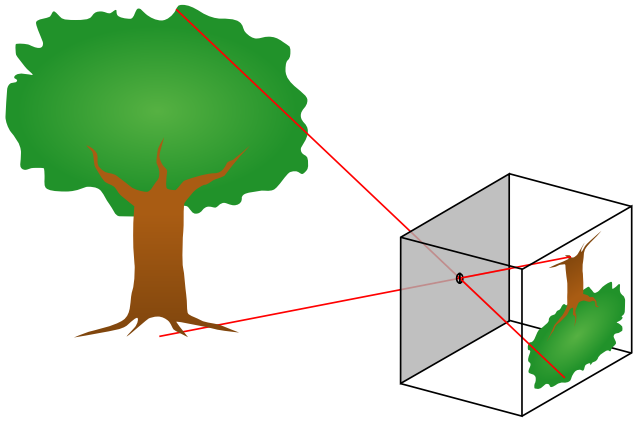
Kolejnym wariantem, branym pod uwagę przy nakreślaniu rozwiązania, było zastosowanie techniki multilateracji, która polega na określaniu położenia obiektu za pomocą mierzenia różnic w czasie dotarcia sygnału do kilku odbiorników. W omawianym przypadku można by użyć mikrofonów na obrzeżach tarczy, które oczekiwałyby na dźwięk wydany przez wbicie się rzutki. Następnie porównując różnice w czasie wykrycia dźwięku, system mógłby wyliczyć pozycję rzutki. To rozwiązanie również obniżyłoby koszty systemu, lecz dawałoby gorsze rezultaty. Tego typu techniki są przede wszystkim stosowane w geodezji, przy dużych odległościach między odbiornikami a emiterem. Tutaj różnice pomiędzy przybyciem dźwięku do poszczególnych mikrofonów byłyby minimalne, przez co trudne byłoby zachowanie wystarczającej precyzji. Wąskim gardłem byłby również przetwornik analogowo-cyfrowy, ponieważ różnice te mogłyby być mniejsze od różnicy w czasie pomiędzy dwiema kolejnymi próbkami, powstałymi podczas procesu próbkowania w przetworniku.

Pod uwagę brano również analizę obrazu z użyciem sztucznej inteligencji. Potencjalnie można by wytrenować sieć neuronową na podstawie wielu zdjęć tarczy z wbitą rzutką. Najważniejszymi wadami tego rozwiązania w stosunku do metody triangulacji są ograniczone możliwości poprawy w przypadku niezado-

walających wyników oraz trudność debugowania. Zaimplementowany ostatecznie sposób można było dużo łatwiej podzielić na części, sprawdzić małe elementy systemu, dzięki czemu błędy w działaniu programu są szybciej lokalizowane i naprawiane. W każdym momencie autor kodu źródłowego jest świadomy, jakie procesy zachodzą w systemie i może na nie wpływać, ograniczona jest rola tzw. czarnych skrzynek. Prawdopodobnie, gdyby użyto sztucznej inteligencji, niemożliwe byłoby uzyskanie np. pozycji rzutki we współrzędnych kartezjańskich, a jedyną daną wyjściową byłoby pole, w jaki trafiła lotka.

3.2 Podstawy fotografii

Urządzeniem, które jest w stanie dostarczać optyczne odzwierciedlenia ograniczonego wycinka przestrzeni trójwymiarowej za pomocą dwuwymiarowych zdjęć, jest aparat fotograficzny. Z biegiem czasu stawał się on coraz bardziej zaawansowany technologicznie, jednak do potrzeb niniejszej pracy wystarczającym modelem jest pierwszy model aparatu, czyli aparat otworkowy (*camera obscura*). Jego budowa jest bardzo prosta: w zaciemnionym pudełku, na jednej z jego ścian tworzy się niewielki otwór, za pomocą którego wpada światło. Na przeciwległej ścianie tworzy się obrócony obraz tego, co znajduje się na wprost pudełka. Schemat działania aparatu otworkowego pokazany jest na rysunku 3.1.

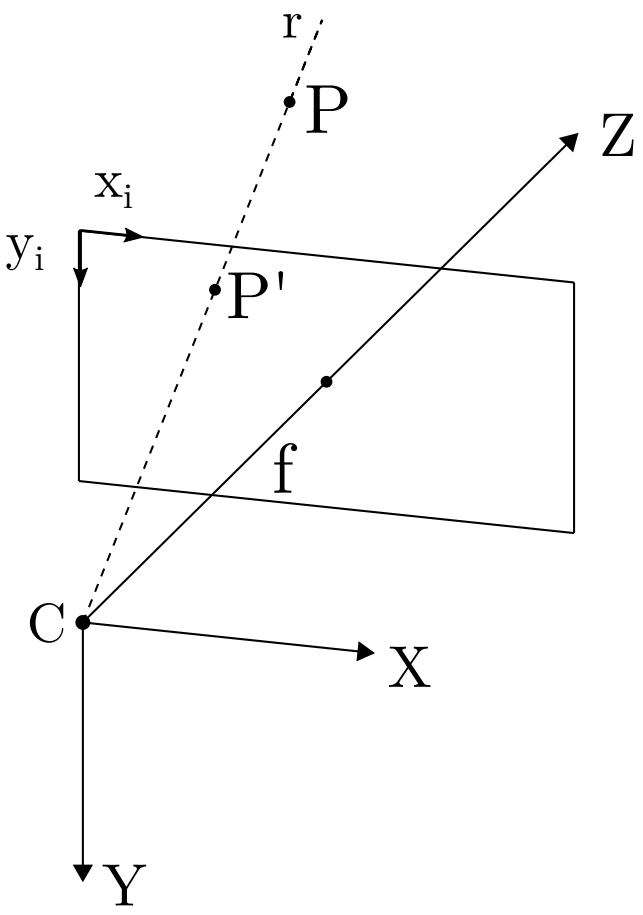


Rysunek 3.1: Zasada działania aparatu otworkowego
Źródło: http://commons.wikimedia.org/wiki/Image:Pinhole-camera.png

W schematach (zwłaszcza związanych z grafiką komputerową) często przedstawia się powyższy model w uproszczonej formie, gdzie obraz tworzy się pomiędzy środkiem aparatu (otworem) a fotografowanym obiektem [5]. Przykład takiego schematu jest widoczny na rysunku 3.2. Osie XYZ reprezentują trójwymiarowy układ współrzędnych, którego początkiem jest środek aparatu, oznaczony jako C. W tym samym układzie jest fotografowany obiekt (punkt), zaznaczony literą P. W odległości f od środka kamery, zwanej jako ogniskowa, znajduje się fragment płaszczyzny, nazywanej płaszczyzną obrazu (eng. *image plane*), na której powstaje obraz. Oś Z jest do niej prostopadła i przechodzi przez jej środek. Z płaszczyzną związany jest dwuwymiarowy układ współrzędnych. Odzwierciedleniem punktu P na obrazie jest punkt P'. Powstaje on na skutek przecięcia się półprostej (eng. *ray*, na rysunku jako r) przechodzącej przez środek kamery i punkt P z płaszczyzną obrazu. Jest to analogia do pojedynczego promienia światła, które wpada przez otwór w aparacie otworkowym. Ponieważ trójwymiarowe punkty są przedstawiane w dwuwymiarowym układzie, tracone są informacje o głębokości, a punkt na obrazie reprezentuje nieskończoną liczbę punktów znajdujących się na prostej r.

W systemie zostały użyte kamery, dzięki którym można zaobserwować ruch. Ich fizyczna budowa nie różni się od aparatu, są one jedynie przystosowane do tego, by robić wiele zdjęć w krótkim czasie, dzięki czemu, po wyświetleniu zdjęć jeden po drugim, mózg interpretuje je jako ciągły ruch.

Przed przedłożeniem całego procesu przetwarzania i analizy obrazu, należy zrozumieć, czym jest obraz z informatycznego punktu widzenia. Zdjęcie cyfrowe (w grafice rastrowej) interpretowane jest jako macierz $A_{m \times n} = [a_{ij}]$, gdzie każdy element a_{ij} odpowiada jednemu pikselowi, czyli kwadratowi wypełnionemu jed-



Rysunek 3.2: Diagram optyczny aparatu otworkowego
Źródło: Opracowanie własne

nolitym kolorem. Piksele powstają z podzielenia płaszczyzny obrazu na gęstą siatkę niewielkich kwadratów. Kolor piksela najczęściej reprezentowany jest jako trójka uporządkowana (r, g, b), gdzie r oznacza intensywność koloru czerwonego, g – zielonego, a b – niebieskiego. Standardowo, każda z trzech składowych koloru jest zapisywana na 8 bitach i przyjmuje wartości od 0 do 255. Złożenie na siebie tych trzech barw podstawowych w różnym stopniu nasycenia pozwala na przedstawianie dowolnej barwy.

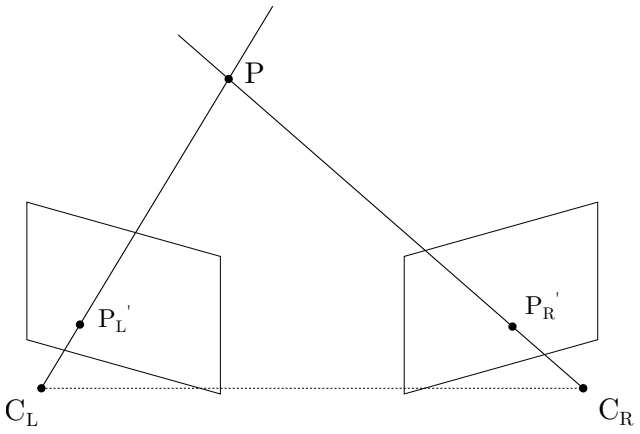
3.3 Triangulacja

Pojęcie triangulacji jest różnie rozumiane w zależności od okoliczności, w jakich jest używane. W geometrii (podobnie jak w grafice komputerowej) oznacza ono proces podziału figury na trójkąty, w socjologii metodę prowadzenia badań społecznych, zaś w teorii grafów triangulacją grafu G można nazywać maksymalny planarny nadgraf grafu G. Należy zatem jasno podkreślić, w jaki sposób rozumiany jest ten termin w niniejszej pracy. Wiąże się on z dwoma dziedzinami: geodezją i rozpoznawaniem obrazów (lub widzeniem komputerowym, od ang. *computer vision*). W obu przypadkach oznacza on wyznaczenie pozycji lub odległości do obiektu za pomocą danych z dwóch miejsc, pomiędzy którymi dystans jest znany.

Triangulacja w geodezji jest używana od wielu lat, dzięki niej uzyskano przybliżone wymiary Ziemi oraz skonstruowano wiele map. Znając położenie punktów A i B, możliwe jest uzyskanie pozycji punktu C, widocznego z A i B. W tym celu, oblicza się kąty $\measuredangle CAB$ z punktu A i $\measuredangle CBA$ z punktu B. Następnie, konstruuje się trójkąt ABC. Znając długość odcinka $|AB|$ i oba kąty z nim sąsiadujące, możliwe jest jednoznaczne

wyznaczenie wszystkich boków trójkąta, dzięki czemu uzyskuje się odległość do mierzonego obiektu. Dzięki powtarzaniu tej metody, gdzie nowo zmierzony punkt staje się daną wejściową do kolejnego pomiaru, możliwe jest tworzenie całych sieci triangulacyjnych, np. pomiędzy miastami.

Rozpoznawanie obrazów interpretuje triangulację bezpośrednio w odniesieniu do zdjęć. Do wytłumaczenia pojęcia w tym kontekście często używa się geometrii epipolarnej (wielobiegunowej). Polega ona na umieszczeniu dwóch modeli z rysunku 3.2 w jednym układzie. Typowy, prosty model tej geometrii, przedstawiono na rysunku 3.3. Widać na nim dwa aparaty (model otworkowy), ustawione pod kątem 45 stopni do czytelnika, gdzie kąt pomiędzy jednym a drugim aparatem wynosi 90 stopni. Na podstawie danych z jednej kamery można uzyskać informację o półprostej przechodzącej przez szukany punkt. Dwie kamery dają dwie takie półproste, których miejsce przecięcia określi jednoznacznie położenie obiektu. Niestety, w rzeczywistości wiele etapów powstawania i przetwarzania obrazu jest narażone na błędy, przez co półproste w praktyce nigdy nie będą idealnie odzwierciedlać rzeczywistości. Wpływają na to zniekształcenia wprowadzane przez soczewkę optyczną, błędy pomiarów odległości między kamerami czy zaawansowanie technologiczne kamer (nie są one zgodne z modelem otworkowym, a także zawierają np. oprogramowanie sprzętowe zmieniające zdjęcie w celu jego poprawy). Powoduje to jednak nie tylko zmniejszenie dokładności, ale sytuację, gdzie przeprowadzenie triangulacji nie będzie w ogóle możliwe, gdyż półproste mogą się nie przeciąć, przez co nie uda się uzyskać pożądanego punktu. Wprowadza to konieczność przetwarzania półprostych w takim celu, by zminimalizować błąd i znaleźć najlepsze przybliżenie szukanego punktu. Stosuje się w tym celu różne metody, których opis przekracza zakres tej pracy.



Rysunek 3.3: Geometria epipolarna
Źródło: Opracowanie własne

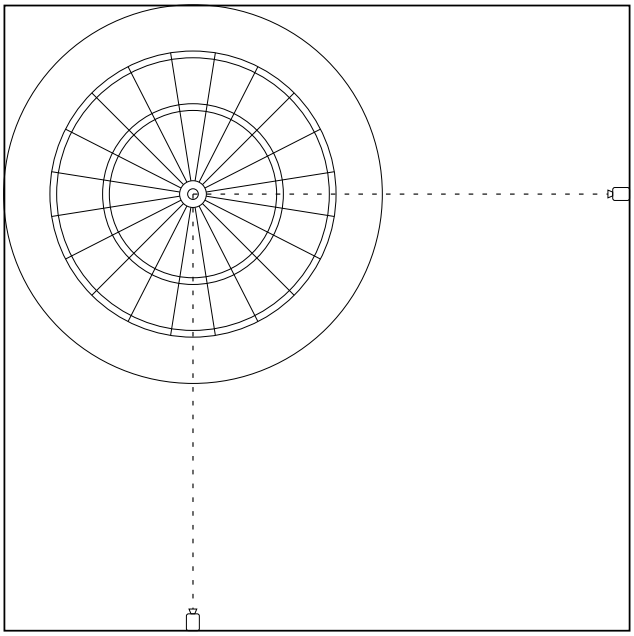
Metoda triangulacji zaimplementowana w obecnej wersji systemu jest połączeniem obu metod. Działa ona w oparciu o dwie kamery skierowane do siebie pod kątem prostym i analizuje dwuwymiarowe obrazy przez nie dostarczone, przetwarza współrzędne pikseli. Różni się ona jednak od klasycznego podejścia do triangulacji, używanego w widzeniu komputerowym: zamiast wyznaczania półprostych ze środka kamery, przechodzących przez punkt na obrazie, a następnie prób wyznaczenia przecięcia tychże półprostych, wyznaczane i poddane obliczeniom są przybliżone kąty dla każdej z kamer, analogicznie jak w wyżej przedstawionym, geodezyjnym podejściu do problemu. Początkowo planowano zaimplementowanie tej wersji jedynie jako pierwsze przybliżenie pozycji rzutki z powodu mniejszej liczby dodatkowych problemów do rozwiązania, ale okazała się ona na tyle dokładna, że pozostawiono ją. Jest to jednak pole do dalszych badań, mogących podnieść skuteczność systemu.

3.4 Stelaż

W systemie, w którym dokładność ma tak duże znaczenie, podczas każdego etapu projektowania i implementacji, należy pamiętać o zminimalizowaniu ewentualnych błędów. Stąd wynika konieczność stabilizacji

układu tarcza-kamera-kamera, tak by wszystkie wymiary konieczne do przeprowadzenia triangulacji były ustalone i nie zmieniały się pomiędzy poszczególnymi testami.

Częścią, która spełnia te potrzeby, jest stelaż wykonany z drewna. Ma on kształt niskiego, otwartego prostopadłościanu (bez jednej ściany). W jego lewym górnym rogu umiejscowiona jest tarcza. Jedna kamera znajduje się na prawym (patrząc z góry) boku prostopadłościanu, na linii poziomej, przechodzącej przez środek tarczy, druga zaś na dolnym boku, na linii pionowej. Schematyczny rzut z góry na stelaż pokazany jest na rysunku 3.4a, a zdjęcie wykonanego stelaża na rysunku 3.4b.

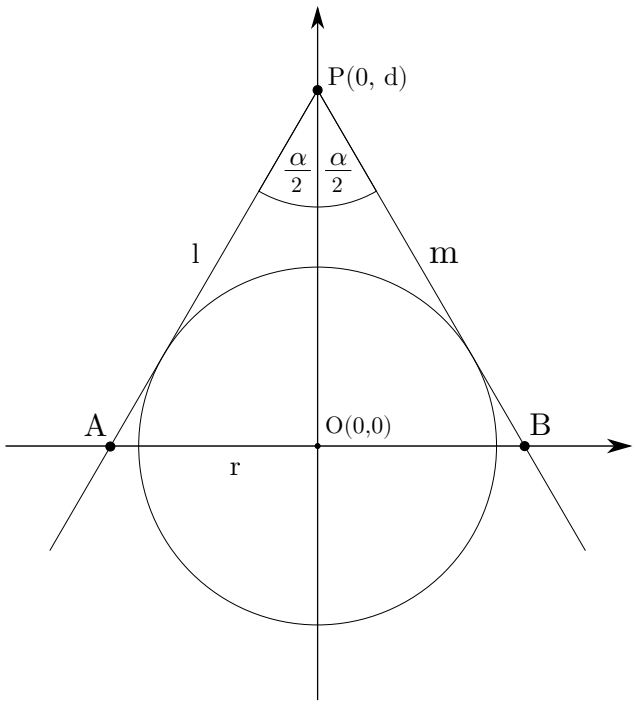


(a) Szkic stelażu (rzut z góry)
Źródło: Opracowanie własne

(b) Ukończony stelaż
Źródło: Opracowanie własne

Przed wykonaniem stelaża należało określić, jakie powinien mieć wymiary. Podstawowym warunkiem było to, że kamera musi obejmować cały obszar tarczy, a jednocześnie cały stelaż powinien być jak najmniejszy. Dlatego pierwszym krokiem było odpowiedzenie na pytanie:

Pytanie 3.1 *W jakiej minimalnej odległości od środka tarczy w kształcie koła o promieniu r powinna znaleźć się kamera o kącie widzenia α, by na zdjęciu cała tarcza była widoczna?*



Rysunek 3.5: Rysunek pomocniczy
Źródło: Opracowanie własne

Do przedstawienia rozwiązania pomocny jest rysunek 3.5. P oznacza środek kamery, O to środek tarczy (początek układu współrzędnych), d to odległość pomiędzy P a O. Przyjmuje się, że kamera jest zwrócona dokładnie na wprost tarczy, dlatego odcinek OP jest dwusieczną kąta APB. Pole wyznaczone przez półproste l i m to pole widzenia kamery. By cała tarcza była widoczna, koło musi być zawarte w tym polu. Ponieważ zadaniem jest znalezienie minimalnej odległości, wystarczy by proste l i m były styczne do okręgu.

Kąty:

$$\measuredangle PAO = \measuredangle PBO = \pi - \frac{\pi}{2} - \frac{\alpha}{2} = \frac{\pi - \alpha}{2}$$

Równania prostych i okręgu:

$$l : y = a_1 x + b_1$$
$$m : y = a_2 x + b_2$$
$$x^2 + y^2 = r^2$$

Ponieważ współczynnik kierunkowy prostej jest równy tangensowi kąta nachylenia do osi OX, otrzymuje się:

$$a_1 = \tan(\frac{\pi - \alpha}{2})$$
$$a_2 = \tan(\pi - \frac{\pi - \alpha}{2}) = \tan(\frac{\pi + \alpha}{2})$$

Półproste l i m przecinają się w punkcie $P(0, d)$:

$$a_1 \cdot 0 + b_1 = a_2 \cdot 0 + b_2$$
$$b_1 = b_2 = d$$

Proste przecinają się z osią OY w tym sam punkcie, więc mają ten sam wyraz wolny, równy d.

$$l : y = \tan(\frac{\pi - \alpha}{2})x + d$$
$$m : y = \tan(\frac{\pi + \alpha}{2})x + d$$

Teraz wystarczy, by następujący układ równań miał jedno rozwiązanie (można by również użyć równania drugiej prostej, nie ma to znaczenia):

$$x^2 + y^2 = r^2$$
$$y = \tan(\frac{\pi + \alpha}{2})x + d$$

Podstawienie:

$$x^2 + (\tan(\frac{\pi + \alpha}{2})x + d)^2 - r^2 = 0$$
$$x^2 + \tan^2(\frac{\pi + \alpha}{2})x^2 + 2\tan(\frac{\pi + \alpha}{2})dx + d^2 - r^2 = 0$$
$$[\tan^2(\frac{\pi + \alpha}{2}) + 1]x^2 + 2\tan(\frac{\pi + \alpha}{2})dx + d^2 - r^2 = 0$$

To równanie kwadratowe musi mieć dokładnie jedno rozwiązanie, dlatego $\Delta = 0$:

$$\begin{aligned}\Delta &= [2\tan(\frac{\pi + \alpha}{2})d]^2 - 4 \cdot [\tan^2(\frac{\pi + \alpha}{2}) + 1] \cdot (d^2 - r^2) = \\ &= 4\tan^2(\frac{\pi + \alpha}{2})d^2 - 4[\tan^2(\frac{\pi + \alpha}{2})d^2 - \tan^2(\frac{\pi + \alpha}{2})r^2 + d^2 - r^2] = \\ &= 4[\tan^2(\frac{\pi + \alpha}{2})d^2 - \tan^2(\frac{\pi + \alpha}{2})d^2 + \tan^2(\frac{\pi + \alpha}{2})r^2 - d^2 + r^2] = \\ &= 4[(\tan^2(\frac{\pi + \alpha}{2}) + 1)r^2 - d^2]\end{aligned}$$

$$\Delta = 0$$
$$4[(\tan^2(\frac{\pi + \alpha}{2}) + 1)r^2 - d^2] = 0$$
$$(\tan^2(\frac{\pi + \alpha}{2}) + 1)r^2 - d^2 = 0$$
$$d^2 = (\tan^2(\frac{\pi + \alpha}{2}) + 1)r^2$$
$$d_1 = r\sqrt{\tan^2(\frac{\pi + \alpha}{2}) + 1} \vee d_2 = -r\sqrt{\tan^2(\frac{\pi + \alpha}{2}) + 1}$$

Ponieważ $d > 0$ (z rysunku), to można odrzucić rozwiązanie d_2 – wyrażenie pod pierwiastkiem jest dodatnie (kwadrat zwiększony o jeden), promień również jest dodatni, więc całe $d_2 < 0$. Ostatecznie:

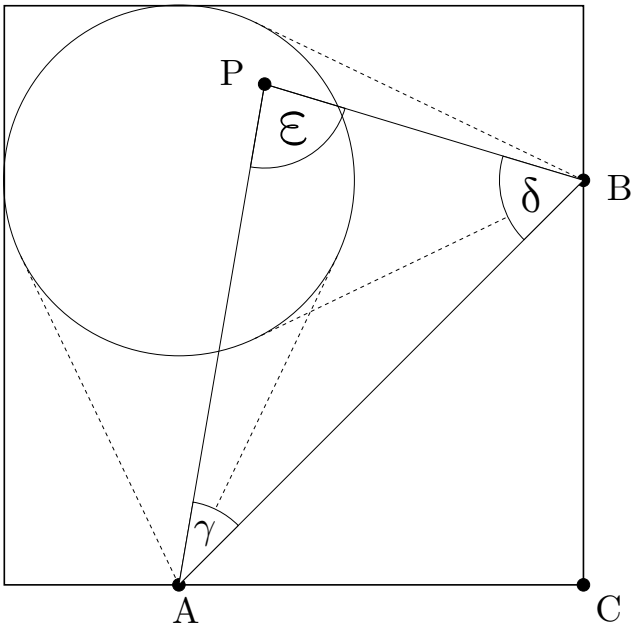
$$d = r\sqrt{\tan^2(\frac{\pi + \alpha}{2}) + 1}$$

Dla tarczy i kamery użytych w systemie, gdzie $r = 22{,}5$ cm, a $\alpha = 53{,}37°$, $d = 50{,}1$ cm.

Na podstawie tej wartości ustalono wymiary całego stelaża. Najmniejsze możliwe wymiary, to $(d + r) \times (d + r)$, czyli 72,6 cm × 72,6 cm. Wykonany stelaż jest nieco większy (74,5 cm × 74,5 cm), m.in. z powodu uwzględnienia niezerowej grubości kamery oraz pozostawienia zapasu na ewentualne modyfikacje i pomyłki.

3.5 Obliczenia

Zgodnie z wcześniej przedstawioną ideą, rozwiązanie zostało oparte na stworzeniu trójkąta z dwoma znanymi kątami i jednym znanym bokiem, co pozwala później na obliczenie wszystkich boków trójkąta. Zobrazowane jest to na rysunku 3.6, do którego kolejne części pracy często będą nawiązywać. Wspomnianym trójkątem jest $\triangle ABP$. A i B to położenia kamer, a P to miejsce wbicia rzutki, które jest celem obliczeń. Kąty γ i δ obliczane są na podstawie pozycji pikseli, a odległość pomiędzy kamerami jest znana lub łatwa do obliczenia z twierdzenia Pitagorasa w $\triangle ABC$.



Rysunek 3.6: Szukany trójkąt w układzie stelaża
Źródło: Opracowanie własne

Oto etapy obliczeń, składających się na rozwiązanie głównego problemu:

- uzyskanie pozycji piksela, który reprezentuje miejsce wbicia rzutki (dla każdej z kamer)
- obliczenie (ze współrzędnych piksela) kąta pomiędzy kamerą a rzutką (dla każdej z kamer)
- umiejscowienie kątów w układzie związanym z kamerami, sformułowanie trójkąta kamera-kamera-rzutka
- obliczenie nieznanych kątów i boków trójkąta
- obliczenie pozycji (x, y) rzutki, a następnie zamiana układu odniesienia na taki, gdzie $(0, 0)$ to środek tarczy
- zamiana współrzędnych kartezjańskich na biegunowe
- przyporządkowanie współrzędnych biegunowych (r, φ) do pola punktowego (segmentu) na tarczy

Pierwszą fazą analizy rzutu jest przetwarzanie obrazu, którego wynikiem są dane o dwóch pikselach, uznanych za miejsce wbicia lotki, po jednym przez każdą z kamer. Jest to trudne zadanie, które zostanie szerzej omówione w następnym rozdziale.

Obliczenie kąta pomiędzy kamerą a rzutką jest oparte na prostym założeniu, iż kamera liniowo dystrybuuje miejsce na zdjęciu w stosunku do kąta, pod jakim znajduje się punkt. Oznacza to, że piksele wysunięte

najbardziej na lewo odpowiadają kątowi zerowemu, najbardziej na prawo – maksymalnemu kątowi widzenia kamery (α), a wszystkie pomiędzy nimi są proporcjonalnie mniejsze lub większe (liniowa interpolacja). Założenie jest dość silne, ponieważ przy niektórych obiektywach, np. typu rybie oko, fragmenty położone przy brzegach obrazu są w innym stopniu powiększone, niż te, które są bliskie środka (dystorsja). W takich przypadkach przyjęcie takiego założenia skutkowałoby dużymi błędami.

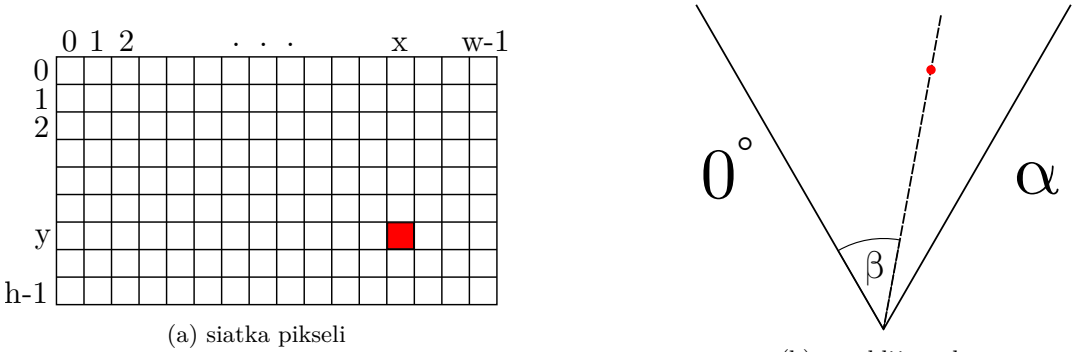
Dla zdjęcia o wymiarach $w \times h$ pikseli, zrobionego przez kamerę o kącie widzenia α, na którym rozważany jest piksel o pozycji (x, y), zachodzi proporcja:

$$\frac{x}{w} = \frac{\beta}{\alpha}$$

gdzie β oznacza kąt od kamery do rzutki. Stąd:

$$\beta = \frac{x \cdot \alpha}{w}$$

Warto zauważyć, że wartość y nie ma w tej sytuacji znaczenia, wpływ na wynik ma tylko pozycja na osi poziomej. Na rysunku 3.7a pokazana jest siatka pikseli z oznaczeniami rzędów i kolumn, wraz z zaznaczonym pojedynczym pikselem. Rysunek 3.7b przedstawia, jaką informację udaje się otrzymać na podstawie pozycji owego piksela – kąt β.



(a) siatka pikseli

(b) przybliżony kąt

Rysunek 3.7: Użycie współrzędnych piksela do wyznaczenia kąta
Źródło: Opracowanie własne.

Jednak otrzymany kąt β nie odpowiada na razie kątom zaznaczonym na rysunku 3.6. Należy go poddać jeszcze niewielkim transformacjom.

$$\gamma = 45° - (\beta_A - \frac{\alpha}{2})$$
$$\delta = 45° + \beta_B - \frac{\alpha}{2},$$

gdzie β_A, β_B to kąty β odpowiednio dla kamery A i B.

Aby zrozumieć powyższe przekształcenia, dobrze jest posłużyć się przykładem, gdy punkt P znajduje się dokładnie na środku tarczy. Wtedy trójkąt ABP jest równoramiennym trójkątem prostokątnym, a więc $\gamma = \delta = 45°$, lecz $\beta_A = \beta_B = \frac{\alpha}{2}$. Należy przyjąć tę sytuację jako sytuację wyjściową do kolejnych obliczeń. Dlatego pierwszym krokiem jest zmiana zbioru wartości, które przyjmuje β: zakres, zamiast od 0 do α, powinien wynosić od $-\frac{\alpha}{2}$ do $\frac{\alpha}{2}$. Ta zmiana dokonuje się przez odjęcie od wyliczonego kąta wartości $\frac{\alpha}{2}$. Jest to teraz kąt nie od lewego ramienia do prostej przecinającej punkt P, lecz od prostej biegnącej przez środek obrazu do prostej przecinającej punkt P. Dzięki temu, dla punktu z przykładu, kąt wynosi 0°, ponieważ znajduje się dokładnie na wprost kamery. Kolejną zależnością, którą należy uwzględnić, jest fakt, iż im większy jest kąt β_A, tym mniejszy jest kąt γ. Stąd wynika konieczność zmiany znaku kąta dla kamery A, przez co pierwszy (licząc od lewej) piksel będzie oznaczał kąt $\frac{\alpha}{2}$, a ostatni $-\frac{\alpha}{2}$ (a nie odwrotnie). Dla kamery B nie ma potrzeby takiej zamiany: im większy kąt β_B, tym większy kąt δ. Po przedstawionych transformacjach należy dodać jeszcze 45°, tak by punkt znajdujący się na wprost kamery był ostatecznie interpretowany jako 45°.

Dzięki powyższym obliczeniom, dostępne są wszystkie dane potrzebne do jednoznacznego wyznaczenia trójkąta ABP. Jego trzeci kąt można obliczyć, korzystając z własności, że suma kątów w trójkącie wynosi 180°:

$$\measuredangle APB = \varepsilon = 180° - \gamma - \delta$$

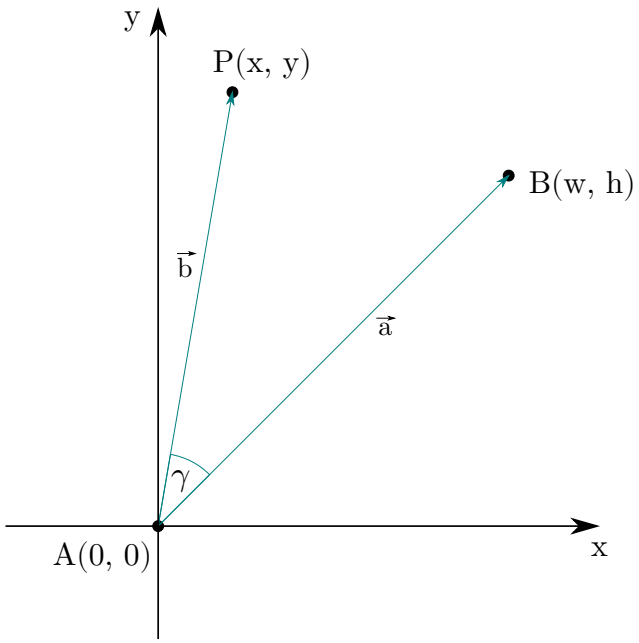
Długości boków uzyskuje się za pomocą twierdzenia sinusów:

$$\frac{|AB|}{\sin\varepsilon} = \frac{|BP|}{\sin\gamma} = \frac{|AP|}{\sin\delta}$$

$$|BP| = \frac{|AB| \cdot \sin\gamma}{\sin\varepsilon}$$

$$|AP| = \frac{|AB| \cdot \sin\delta}{\sin\varepsilon}$$

Na podstawie danych o trójkącie ABP, należy obliczyć pozycję punktu P, umieszczając trójkąt w układzie współrzędnych, w którym punkt A jest początkiem. Szkic pomocniczy znajduje się na rysunku 3.8. Pozycja punktu B jest określona przy tworzeniu stelaża – najczęściej będzie to (d, d), jednak w poniższych obliczeniach zakłada się ogólniejszy przypadek, gdzie współrzędne punktu na obu osiach nie muszą się sobie równać. Przyjęto oznaczenie $B(w, h)$. Jedynymi szukanymi w tym podproblemie są x oraz y, czyli współrzędne punktu P. Obliczenia opierają się na działaniach wektorowych: wyznacza się wektor $\overrightarrow{AP} = \overrightarrow{b}$ oraz $\overrightarrow{AB} = \overrightarrow{a}$.



Rysunek 3.8: Układ współrzędnych o środku w A
Źródło: Opracowanie własne

Główne równanie wynika z wzoru na kąt między wektorami:

$$\cos\gamma = \frac{\vec{a} \circ \vec{b}}{|\vec{a}| \cdot |\vec{b}|} = \frac{[w, h] \circ [x, y]}{|\vec{a}| \cdot |\vec{b}|} = \frac{wx + hy}{|\vec{a}| \cdot |\vec{b}|}$$

$$|\vec{a}| \cdot |\vec{b}| \cos\gamma = wx + hy$$

$$x = \frac{|\vec{a}| \cdot |\vec{b}| \cos\gamma - hy}{w}$$

Drugą zależnością, z której należy skorzystać, jest wzór na długość wektora:

$$|\vec{b}| = \sqrt{x^2 + y^2}$$
$$|\vec{b}|^2 = x^2 + y^2$$

Podstawienie:

$$|\vec{b}|^2 = \left(\frac{|\vec{a}| \cdot |\vec{b}| \cos\gamma - hy}{w}\right)^2 + y^2$$
$$|\vec{b}|^2 = \frac{(|\vec{a}| \cdot |\vec{b}| \cos\gamma - hy)^2}{w^2} + y^2$$
$$|\vec{b}|^2 = \frac{1}{w^2}\left(|\vec{a}|^2 \cdot |\vec{b}|^2 \cos^2\gamma - 2|\vec{a}| \cdot |\vec{b}| h \cos\gamma\ y + h^2 y^2\right) + y^2$$
$$|\vec{b}|^2 = \frac{1}{w^2}|\vec{a}|^2 \cdot |\vec{b}|^2 \cos^2\gamma - \frac{2}{w^2}|\vec{a}| \cdot |\vec{b}| h \cos\gamma\ y + \left(1 + \frac{h^2}{w^2}\right) y^2$$
$$\left(1 + \frac{h^2}{w^2}\right)y^2 - \frac{2}{w^2}|\vec{a}| \cdot |\vec{b}| h \cos\gamma\ y + |\vec{b}|^2 \left(\frac{|\vec{a}|^2 \cos^2\gamma}{w^2} - 1\right) = 0$$

Otrzymano równanie kwadratowe postaci $a_2 y^2 + a_1 y + a_0$, gdzie:

$$a_2 = 1 + \frac{h^2}{w^2}$$
$$a_1 = -\frac{2}{w^2}|\vec{a}| \cdot |\vec{b}| h \cos\gamma$$
$$a_0 = |\vec{b}|^2 \left(\frac{|\vec{a}|^2 \cos^2\gamma}{w^2} - 1\right)$$

Wzory na pierwiastki tego równania nie są podawane, ponieważ również w implementacji korzysta się z zewnętrznej biblioteki do wyliczania pierwiastków, odpornej na błędy numeryczne. W przypadku, gdy równanie ma dwa rozwiązania, wybierane jest $y > 0$.

Pierwsza współrzędna obliczana z wcześniej już przedstawionego wzoru:

$$x = \frac{|\vec{a}| \cdot |\vec{b}| \cos\gamma - hy}{w}$$

Następnie należy przenieść układ współrzędnych w taki sposób, by jego początkiem był środek tarczy. Przekształceniem, które to zapewnia, jest funkcja f:

$$f(x, y) = (x, y + h)$$

W celu łatwiejszego przyporządkowania pozycji rzutki do pola na tarczy, warto przedstawić ją za pomocą współrzędnych biegunowych, tzn. promienia r_p od środka tarczy (O) do punktu P oraz kąta φ pomiędzy półprostą OB a wektorem $\overrightarrow{OP}$.
Promień r_p wyliczany jest ze wzoru na długość wektora (z tw. Pitagorasa):

$$r_p = \sqrt{x^2 + y^2}$$

Wzór na wartość kąta φ jest zależny od ćwiartki układu, w jakiej znajduje się punkt:

$$\varphi = \begin{cases} arctg(\frac{y}{x}) & \text{gdy } x > 0 \wedge y \geqslant 0 \\ arctg(\frac{y}{x}) + 2\pi & \text{gdy } x > 0 \wedge y < 0 \\ arctg(\frac{y}{x}) + \pi & \text{gdy } x < 0 \\ \frac{\pi}{2} & \text{gdy } x = 0 \wedge y > 0 \\ \frac{3\pi}{2} & \text{gdy } x = 0 \wedge y < 0 \end{cases}$$

Ostatnim etapem jest przypisanie współrzędnym biegunowym konkretnego pola na tarczy. Patrząc na kołową budowę tarczy łatwo zauważyć, że kąt φ wskazuje, w który wycinek kołowy trafiła rzutka, a promień r_p pozwala określić, czy jest to pole standardowe, podwójne, potrójne itd. Do reprezentacji segmentu będzie używana notacja z sekcji 2.2.

Formalnie, należy sformułować funkcję g, która każdej parze (r_p, φ) przyporządkuje parę (w, t):

$$g : \mathbb{R}_+^0 \times [0, 2\pi] \to W \times T$$

W tym celu wprowadza się pomocnicze funkcje: g_1 oraz g_2. Pierwsza z nich na podstawie długości promienia określa, z którego pierścienia pochodzi miejsce wbicia rzutki. Na rysunku 3.9 wprowadzono oznaczenia r_1, r_2, r_3, r_4, r_5, r_6, które oznaczają długości promieni kolejnych okręgów na tarczy. Są to wartości charakterystyczne dla danej tarczy, które należy uzyskać z największą dokładnością, tak by precyzyjnie przeprowadzić ten etap obliczeń.

$$g_1 : \mathbb{R}_+^0 \to T \cup \{\textit{inner bull}, \textit{outer bull}, \textit{null}\}$$

$$g_1(r_p) = \begin{cases} \textit{inner bull} & \text{gdy } r_p \leqslant r_1 \\ \textit{outer bull} & \text{gdy } r_1 < r_p \leqslant r_2 \\ S & \text{gdy } r_2 < r_p \leqslant r_3 \vee r_4 < r_p \leqslant r_5 \\ T & \text{gdy } r_3 < r_p \leqslant r_4 \\ D & \text{gdy } r_5 < r_p \leqslant r_6 \\ \textit{null} & \text{gdy } r_p > r_6 \end{cases}$$

Funkcja g_2 określa, na którym spośród dwudziestu wycinków kołowych znajduje się lotka. Ciąg (a_n), indeksowany od zera, zawiera wartości od 1 do 20 w kolejności ich występowania na tarczy zgodnie z ruchem przeciwnym do ruchu wskazówek zegara, zaczynając od pola nr 6. Na podstawie kąta wyliczany jest indeks elementu ciągu, który reprezentuje szukany wycinek kołowy. Dzieje się to za pomocą prostej interpolacji: kąt pełny to 2π, więc pojedynczy wycinek ma kąt $\frac{2\pi}{20}$. Kąt φ (pomniejszony o połowę kąta wycinka numer 6) jest dzielony przez wielkość kąta pojedynczego wycinka, co jednoznacznie przyporządkowuje kąt do wycinka.

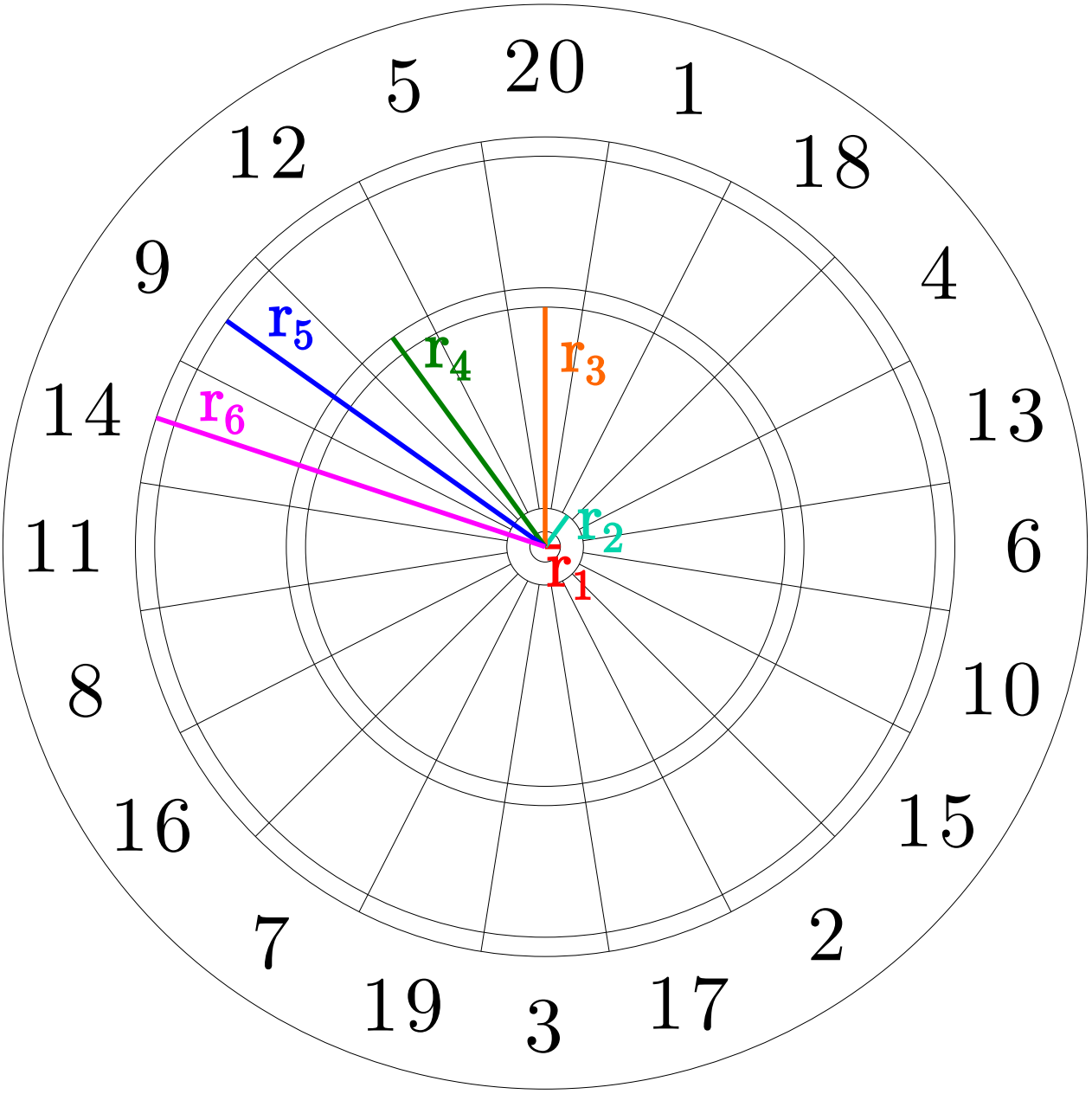
$$g_2 : [0, 2\pi] \to A$$

$$(a_n) = (6, 13, 4, \ldots, 2, 15, 10)$$

$$g_2(\varphi) = a \left\lfloor \frac{(\varphi + \frac{1}{2} \cdot \frac{2\pi}{20}) \mod 2\pi}{\frac{2\pi}{20}} \right\rfloor$$

Ostatecznie, korzystając z pomocniczych funkcji, g_1 określa pole punktowe w następujący sposób:

$$g(r_p, \varphi) = \begin{cases} (0, S) & \text{gdy } g_1(r_p) = \textit{null} \\ (25, S) & \text{gdy } g_1(r_p) = \textit{inner bull} \\ (50, S) & \text{gdy } g_1(r_p) = \textit{outer bull} \\ (g_2(\varphi), S) & \text{gdy } g_1(r_p) = S \\ (2 \cdot g_2(\varphi), D) & \text{gdy } g_1(r_p) = D \\ (3 \cdot g_2(\varphi), T) & \text{gdy } g_1(r_p) = T \end{cases}$$



Rysunek 3.9: Tarcza z zaznaczonymi promieniami
Źródło: Opracowanie własne

Projekt systemu

Niniejszy rozdział traktuje o procesach systemu, jego składowych i powiązaniach między nimi. Do ich przedstawienia używane są opisy słowne oraz diagramy UML. Istotnym elementem jest prezentacja sposobu, w jaki przetwarzane są obrazy w systemie oraz algorytmów z tym związanych.

4.1 Przypadki użycia

Powiązany z systemem jest obecnie jeden, główny przypadek użycia. Aktorem jest gracz, który po uruchomieniu aplikacji mobilnej łączy się z systemem wbudowanym podłączonym do tarczy. Po informacji o ustanowieniu połączenia, może on zacząć wykonywać rzuty. Po każdym rzucie, w krótkim czasie, otrzymuje on informację o polu, w jakie trafił oraz wizualizację miejsca wbicia lotki na graficznym schemacie tarczy. Aplikacja, w razie potrzeby, sygnalizuje problemy z połączeniem na linii klient-serwer.

4.2 Urządzenia i komponenty

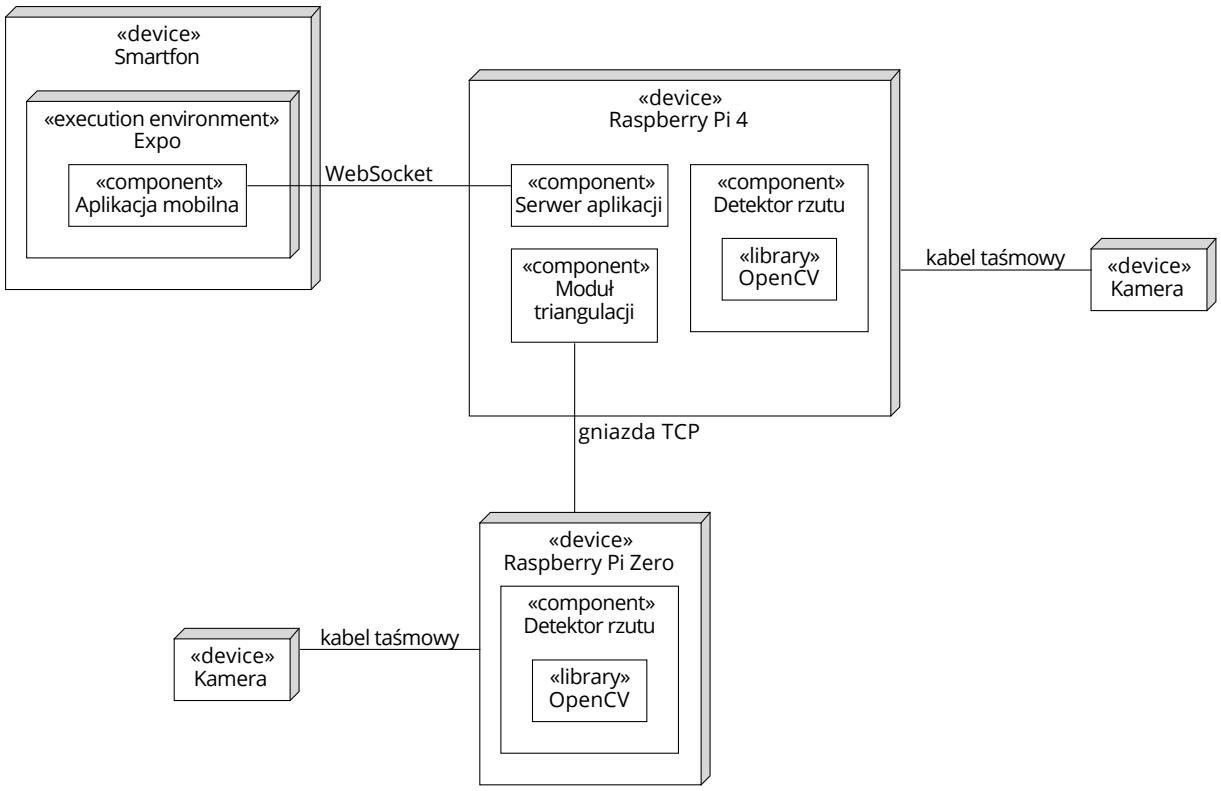
W systemie występują następujące urządzenia:

- minikomputer `Raspberry Pi 4`
- minikomputer `Raspberry Pi Zero W`
- dwie kamery `Camera Module v1 (OmniVision OV5647)`
- smartfon

Na diagramie wdrożenia (4.1) pokazano te urządzenia wraz z połączeniami między nimi oraz komponentami, jakie wchodzą w ich skład. `Raspberry Pi 4` jest najważniejszym urządzeniem, które synchronizuje pracę innych oraz dokonuje obliczeń. Łączy się z kamerą za pomocą kabla taśmowego, a używa jej do wykrywania rzutu za pomocą biblioteki `OpenCV`. Analogicznie wygląda to w przypadku `Pi Zero`, na którym znajduje się druga instancja detektora rzutu. Po wykryciu rzutu informacje wysyłane są do `Pi 4` z użyciem gniazd TCP, a następnie poddane triangulacji. Na `Pi 4` uruchomiony jest również serwer dla aplikacji mobilnej, która uruchamiana jest na smartfonie w środowisku Expo [2]. Połączenie następuje zgodnie ze standardem WebSocket.

4.3 Diagramy aktywności

Na diagramie aktywności (4.2) przedstawiono logikę procesu zachodzącego w systemie z zachowaniem wysokiego poziomu ogólności. System rozpoczyna działanie od rozwidlenia na przetwarzanie obrazu z użyciem kamery dolnej oraz prawej. W obu przypadkach wygląda to identycznie - w pętli wykonywane są zdjęcia i poddawane analizie. W przypadku wykrycia rzutu pętla zostaje przerwana, a program synchronizuje się z drugą kamerą i przeprowadza obliczenia prowadzące do pozycji rzutki, która zostaje wyświetlona w aplikacji mobilnej. Następnie proces rozpoczyna się od początku.



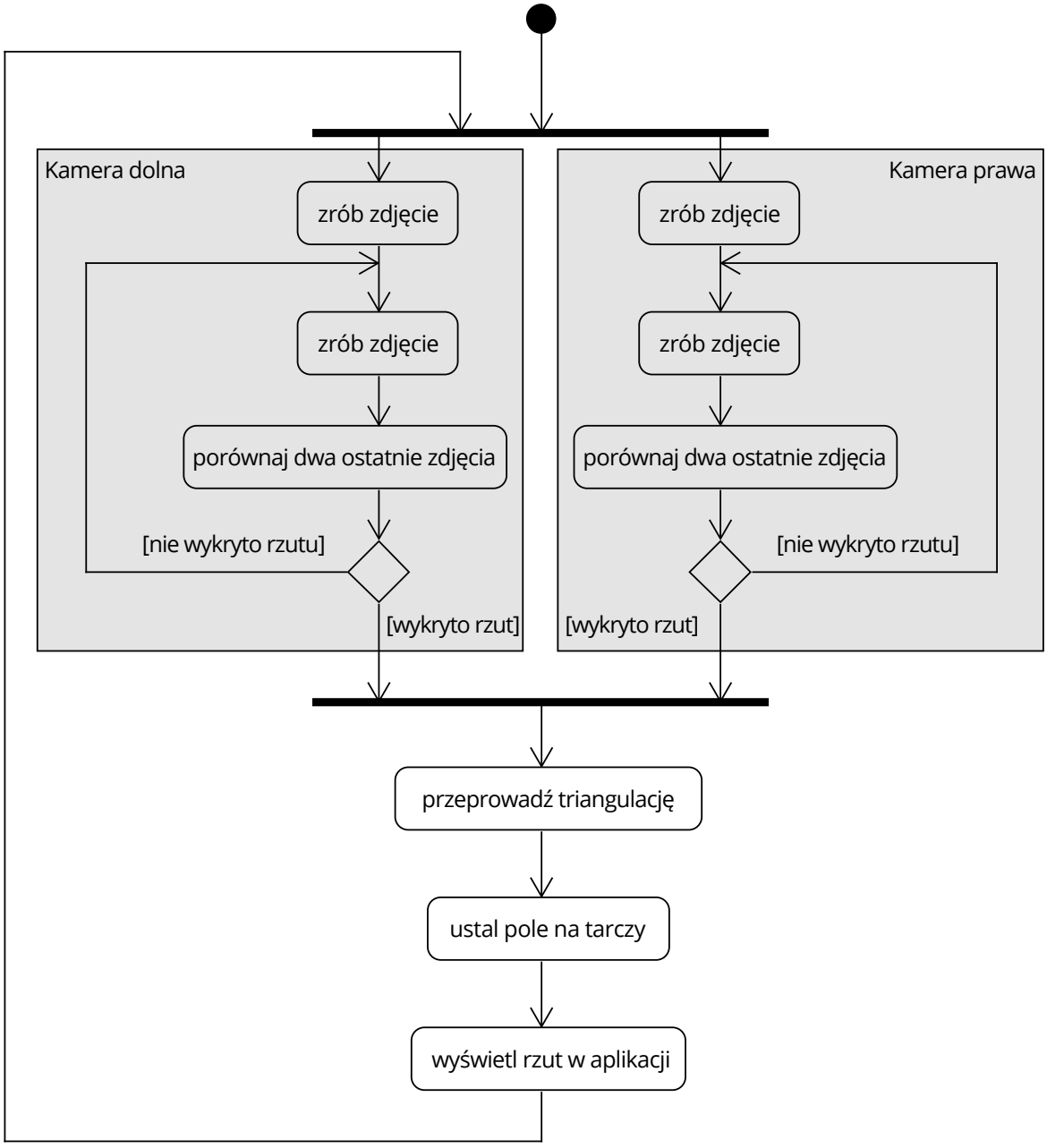
Rysunek 4.1: Diagram wdrożeniowy
Źródło: Opracowanie własne

4.4 Diagramy sekwencji

Diagram sekwencji (4.3) pokazuje, w jaki sposób odbywa się komunikacja pomiędzy poszczególnymi komponentami. Jako pierwszy startuje program na `Raspberry Pi 4`. Najpierw czeka on na połączenie z drugim minikomputerem, `Pi Zero` oraz w osobnym wątku tworzy serwer dla aplikacji mobilnej. Gracz może wtedy włączyć aplikację, która natychmiastowo połączy się z tymże serwerem. W tym momencie gracz może zacząć wykonywanie rzutów. `Pi 4` oraz `Pi Zero` równolegle wykrywają rzut, lecz po wykryciu rzutu przez pierwszego z nich, czeka on na dane o położeniu piksela, nadchodzące z drugiego. Później, po przeprowadzeniu obliczeń, informacje o rzucie są przesyłane od serwera do aplikacji mobilnej. Możliwe jest podłączenie wielu klientów – serwer posiada listę podłączonych użytkowników i wysyła informacje do każdego z nich.

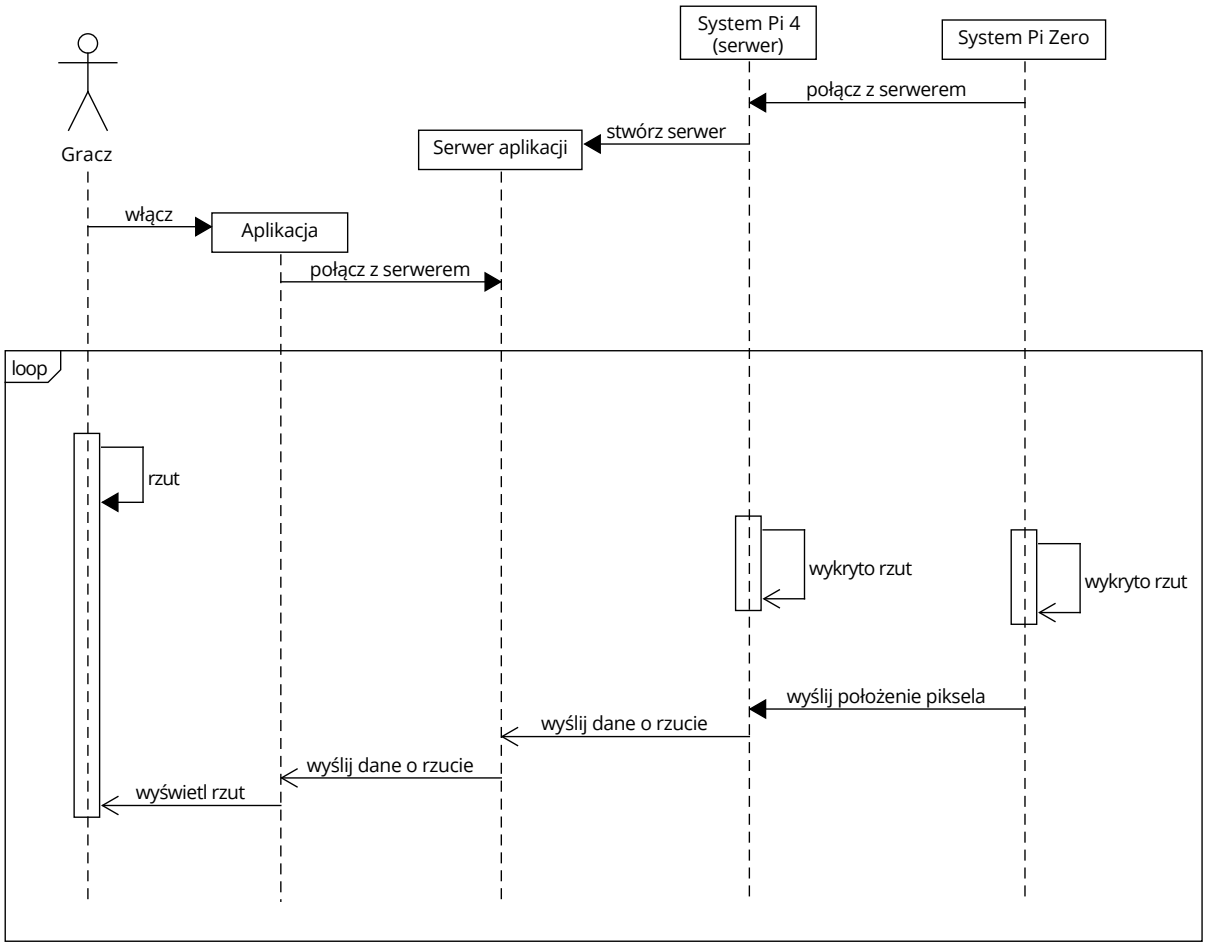
4.5 Przetwarzanie obrazu

Podstawowym algorytmem, który nie został opisany w poprzednim rozdziale, jest proces przetwarzania obrazu. Zadaniem, które należy wykonać za pomocą metod związanych z tą dziedziną, jest odjęcie tła (ang. *background substraction*). Polega ono na wyodrębnieniu elementów pojawiających się na pierwszym planie, na podstawie następujących po sobie zdjęć. W prezentowanym systemie użyte zostało w celu wykrycia rzutu, a następnie ustalenia samej pozycji rzutki na zdjęciu. W ogólności jest to zadanie trudne, ze względu na możliwe zmiany tła, spowodowane m.in. minimalnym ruchem lub zmianą warunków oświetlenia. Podejść do jego rozwiązania jest wiele [6], jednak w niniejszej pracy, z powodu niewielkiej liczby tego typu czynników, zastosowano jedno z najprostszych - odejmowanie od siebie dwóch kolejnych klatek.



Rysunek 4.2: Diagram aktywności
Źródło: Opracowanie własne

Każda z kamer, co pewien krótki czas, wykonuje zdjęcie. Następnie, dwa ostatnie zdjęcia z danej kamery, są od siebie odejmowane. Dzięki reprezentacji macierzowej, działanie to odbywa się zgodnie z zasadami bezwzględnego odejmowania macierzy. Jego wynikiem jest nowe zdjęcie, w której piksele niezmienione w czasie pomiędzy wykonaniem klatek powinny być bliskie wartości 0 (piksel czarny), zaś te reprezentujące pojawienie się nowego obiektu – wyraźnie większe od zera. Aby zmniejszyć występujące szumy oraz przekształcić obraz



Rysunek 4.3: Diagram sekwencji
Źródło: Opracowanie własne

na binarny (czarno-biały), stosuje się następnie funkcję progową (ang. *threshold*), która wszystkie piksele o wartościach niższych od ustalonego progu zamienia na czarne, a wyższe – na piksele białe.

Tak przygotowane zdjęcie należy poddać wykrywaniu konturów. Konturem nazywa się zbiór punktów reprezentujących krzywą [6]. W przypadku problemu omawianego w pracy, celem jest wykrycie konturu wokół rzutki, która pojawiła się na ostatnio wykonanym zdjęciu. Algorytm wykrywania konturów jest dość skomplikowany, jego szczegóły można znaleźć w [7]. Po uzyskaniu listy konturów wybierany jest największy z nich (pod względem powierzchni, którą wyznacza), a następny poddany analizie, czy jest to kontur rzutki. W tym miejscu należy wyznaczyć charakterystyczne cechy lotki na zdjęciu, opierając się na empirycznych danych pochodzących z testów. W obecnej implementacji kontur jest uznawany za kontur rzutki wtedy, gdy powierzchnia przez niego wyznaczona mieści się w określonych doświadczalnie granicach. Innymi sposobami jest na przykład sprawdzanie proporcji długości do szerokości w prostokącie okalającym kontur. Jeśli nie wykryto żadnych konturów lub nie spełniają one warunków, uznaje się, że rzut nie wystąpił pomiędzy badanymi ujęciami.

Ostatnią fazą jest wybranie pojedynczego piksela, którego pozycja będzie reprezentować miejsce wbicia rzutki: jest to najniżej położony piksel konturu (znajdujący się najbliżej tarczy).

4.6 Opis protokołów

W systemie do komunikacji używa się dwóch protokołów: TCP [3] i WebSocket [4]. Pierwszy z nich używany jest do wysłania informacji z `Pi Zero` do `Pi 4` w sposób binarny: 3 liczby całkowite (pozycja x, pozycja y oraz liczba porządkowa rzutki), po 4 bajty na liczbę, w formacie *big endian*. Drugi protokół stosuje się przy przekazaniu komunikatu o wyliczonej pozycji rzutki. Jego treść jest w formacie `JSON` i została pokazana na listingu 1.

```
{
    "x": 23.123516,
    "y" : 5.7825,
    "segment" : 20
}
```

Listing 1: Przykładowy komunikat o wykrytej rzutce

Implementacja systemu

–tu będzie rozdział 4–

Instalacja i wdrożenie

–tu będzie rozdział 5–

Podsumowanie

– tu będzie podsumowanie –

Bibliografia

[1] British Darts Organisation Playing Rules and Tournament Rules. https://www.bdodarts.com/images/bdo-content/doc-lib/B/bdo-playing-rules.pdf.

[2] Strona projektu Expo. https://expo.io/.

[3] Transmission Control Protocol. RFC 793, Wrze. 1981.

[4] The WebSocket Protocol. RFC 6455, Gru. 2011.

[5] M. Forsman. Point cloud densification. Praca magisterska, Umeå University, 01 2011.

[6] A. Kaehler, G. Bradski. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., wydanie 1st, 2016.

[7] S. Suzuki, K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.