Techniki multimedialne – sprawozdanie

Temat własny - Automatyczny system zliczania punktów w grze w darta z wykorzystaniem wizji komputerowej.

Rafał Pyłko 325506

Opis projektu

Projekt polega na stworzeniu prototypowego oprogramowania w języku Python, które będzie w stanie automatycznie rozpoznawać i zliczać punkty zdobyte podczas gry w darta. Aplikacja będzie analizować nagranie wideo pochodzące ze statycznie ustawionej kamery - telefonu na statywie. Kluczowym elementem systemu jest przetwarzanie obrazu w celu korekcji perspektywy, kalibracji pól punktowych tarczy oraz detekcji rzutek, a następnie przeliczanie ich pozycji na konkretny wynik punktowy.

Wykorzystane biblioteki

Biblioteki python wykorzystane do realizacji projektu i niezbędne do uruchomienia programu to:

- **OpenCV-Python (cv2):** podstawowa biblioteka do wszystkich operacji związanych z wizją komputerową: wczytywanie wideo i obrazów, konwersja przestrzeni barw, transformacje geometryczne (perspektywa), operacje morfologiczne, rysowanie na obrazie oraz tworzenie prostego GUI.
- **Numpy (np):** biblioteka do operacji numerycznych, w szczególności do pracy z macierzami (obrazy, macierze transformacji) i wektorami (współrzędne punktów).
- **math:** biblioteka wykorzystywana do obliczeń trygonometrycznych niezbędnych przy konwersji współrzędnych kartezjańskich na biegunowe (siatka punktów).
- **os:** biblioteka wykorzystana do organizowania ścieżek do pliku .mp4 i innych zależności.
- **json:** przydatna do zapisania ustawień kalibracji tarczy, aby nie wykonywać kalibracji każdorazowo

Instrukcja uruchomienia

1. Do uruchomienia programu niezbędne są wcześniej wspomniane biblioteki i interpreter python.
2. W katalogu roboczym powinien znajdować się:

- Skrypt: *dart_system.py*
- Nagranie rozgrywki dart'a z statycznej kamery: *dart_game_video.mp4*

3. Program należy uruchomić z wiersza poleceń lub przez IDE.
4. Po uruchomieniu w konsoli pojawi się proste menu z instrukcją obsługi zawierającą 3 możliwe akcje:
    - Wpisanie **s** i potwierdzenie **enter**: rozpoczęcie rozgrywki – program analizuje nagranie, jeżeli istnieje już kalibracja tarczy.
    - Wpisanie **c** i potwierdzenie **enter**: uruchomienie kalibrowania tarczy.
    - Wpisanie **q** i potwierdzenie **enter**: zakończenie działania programu.
5. Po wybraniu kalibracji w konsoli pojawiają się jasne instrukcje co należy wykonać.

Działanie systemu i implementacja

Działanie systemu można podzielić na dwa etapy:

- Kalibracja tarczy
- Automatyczne zliczanie punktów na podstawie obrazu z kamery

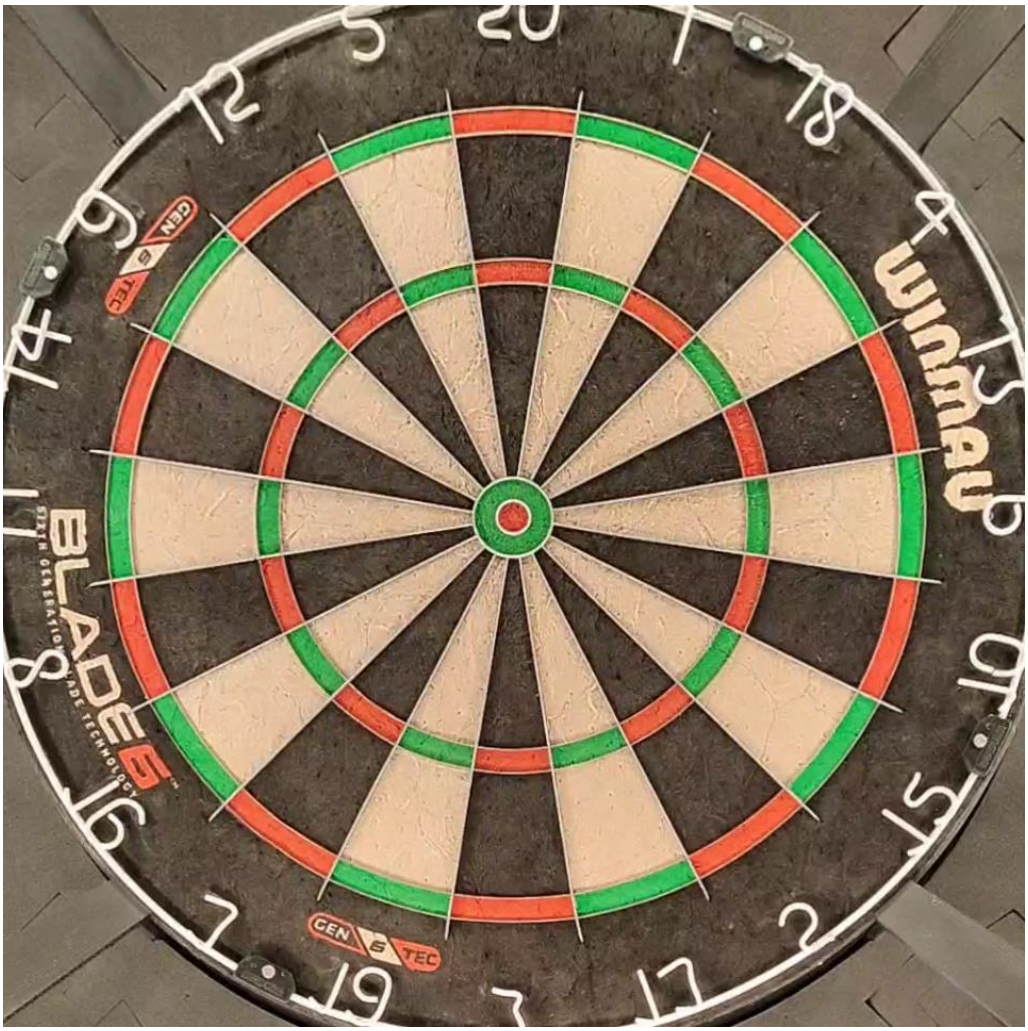
Kalibracja tarczy

Proces kalibracji jest kluczowym etapem inicjalizującym działanie systemu. Jego celem jest odwzorowanie geometrii tarczy widzianej pod kątem przez kamerę na płaski, znormalizowany układ współrzędnych, co umożliwia precyzyjne określanie trafionych pól. Cały proces został podzielony na następujące kroki:

1. **Inicjalizacja i wstępne przetwarzanie obrazu** W pierwszej fazie program otwiera źródło wideo i pobiera pierwszą dostępną klatkę. Ponieważ kamera obejmuje szeroki kadr, obraz zostaje poddany operacji przycięcia. Program wycina fragment klatki, odrzucając zbędne tło, co przyspiesza dalsze obliczenia i skupia uwagę algorytmu na samej tarczy.

Rysunek 1 Pierwsza klatka nagrania przycięta do tarczy

2. **Wyznaczenie macierzy transformacji perspektywicznej** Aby skorygować zniekształcenia wynikające z ustawienia kamery pod kątem (efekt elipsy zamiast koła), użytkownik proszony jest o wskazanie 4 punktów na zewnętrznym obwodzie tarczy. Algorytm sortuje pobrane współrzędne w ustalonej kolejności (lewy-góra, prawy-góra, prawy-dół, lewy-dół), a następnie, wykorzystując funkcję *cv2.getPerspectiveTransform*, oblicza macierz homografii. Pozwala ona na przekształcenie widoku perspektywicznego na widok ortogonalny ("z góry"). Ważne jest aby zaznaczone punkty tworzyły kwadrat, kiedy kąt obserwacji jest równy 0˚.
3. **Prostowanie obrazu** Na podstawie obliczonej macierzy następuje "wyprostowanie" obrazu przy użyciu funkcji *cv2.warpPerspective*. Wynikiem tej operacji jest kwadratowy obraz, na którym tarcza odzyskuje swój okrągły kształt. Jest to niezbędne do zastosowania matematycznego modelu pól punktowych opartego na współrzędnych biegunowych.



Rysunek 2 Widok tarczy po tranformacji perspektywicznej

4. **Mapowanie pól punktowych** Na wyprostowanym obrazie następuje etap wiązania wymiarów fizycznych tarczy z pikselami obrazu. Użytkownik wskazuje:
    - Idealny środek tarczy (Bullseye).
    - Dwa punkty na zewnętrznej krawędzi pola podwójnego. System oblicza odległość euklidesową (*math.dist*) między środkiem a punktami brzegowymi, wyznaczając średni promień tarczy w pikselach. Na tej podstawie, korzystając z ustandaryzowanych proporcji tarczy do darta, program generuje wirtualną siatkę sektorów i pierścieni (Double, Triple, Bull).

Rysunek 3 Widok tarczy z wirtualnym polem punktowym

5. **Zapis kalibracji** Ostatnim krokiem jest utrwalenie wyników kalibracji. Macierz transformacji perspektywicznej, współrzędne środka tarczy oraz jej promień zostają zapisane do pliku *calibration_data.json*. Dzięki temu przy kolejnym uruchomieniu gry system automatycznie wczytuje te parametry, eliminując konieczność ponownej kalibracji dla tego samego ustawienia kamery.

Automatyczne zliczanie punktów

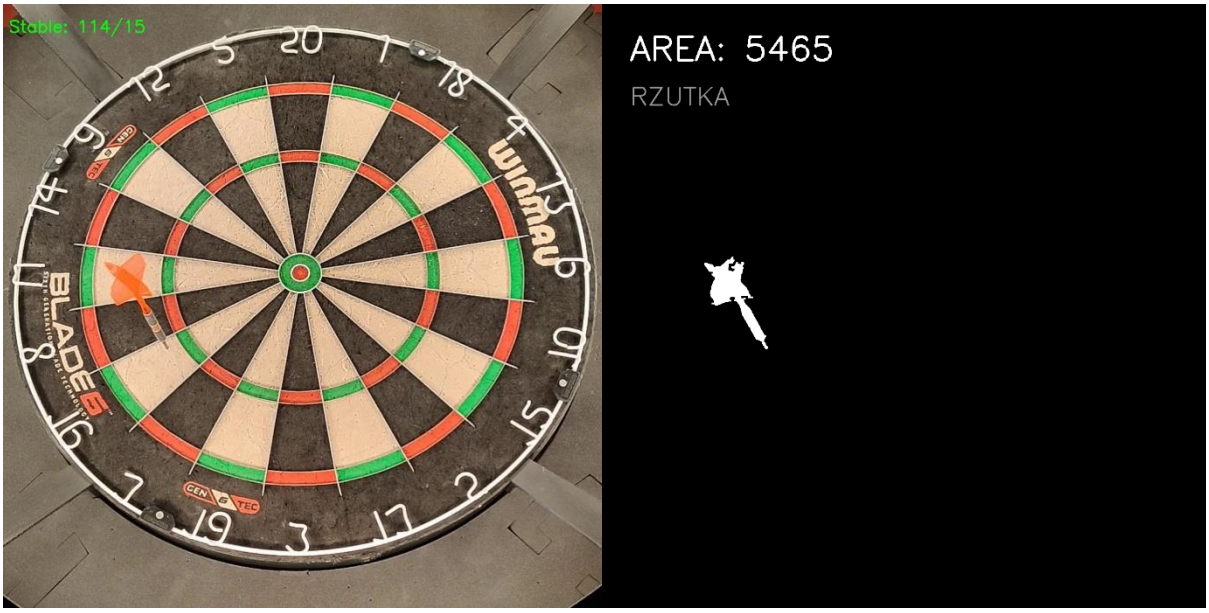
Po pomyślnej kalibracji system przechodzi w tryb ciągłej analizy wideo (pętla główna). Algorytm działa w czasie rzeczywistym, przetwarzając każdą klatkę obrazu w celu wykrycia momentu wbicia rzutki, zlokalizowania jej grota i przypisania odpowiedniej wartości punktowej. Proces ten realizowany jest w następujących krokach:

1. **Wstępne przetwarzanie i stabilizacja obrazu** Każda klatka pobrana z kamery jest konwertowana do skali szarości (*cv2.cvtColor*), a następnie poddawana rozmyciu Gaussa (*cv2.GaussianBlur*). Zabieg ten ma na celu redukcję szumów matrycy kamery, które mogłyby generować fałszywe detekcje. System posiada zaimplementowany mechanizm stabilizacji – analiza rzutu następuje dopiero wtedy, gdy obraz jest nieruchomy przez określoną liczbę klatek (zdefiniowaną

parametrem MIN_FRAMES_STABLE). Zapobiega to błędnym obliczeniom w momencie, gdy rzutka jest jeszcze w locie lub drga po uderzeniu w tarczę.

2. **Detekcja ruchu i segmentacja (Odejmowanie tła)** Kluczową techniką wykorzystaną do znalezienia rzutki jest odejmowanie tła (ang. *Background Subtraction*). Program przechowuje w pamięci obraz "czystej" tarczy (tło). W momencie ustabilizowania obrazu, algorytm oblicza różnicę absolutną (*cv2.absdiff*) między aktualną klatką a zapamiętanym tłem. Wynikowa różnica jest poddawana binaryzacji (progowaniu), co tworzy czarno-białą maskę, na której białe piksele reprezentują nowe obiekty (rzutkę).
3. **Filtracja morfologiczna i analiza konturów** Surowa maska często zawiera "szum" (pojedyncze piksele) oraz "dziury" wewnątrz wykrytego obiektu. Aby uzyskać spójny kształt rzutki, zastosowano operacje morfologiczne:
    - **Zamknięcie (Morphological Close):** "Skleja" rozerwane fragmenty rzutki (np. oddzielne wykrycie lotki i trzonu).
    - **Otwarcie (Morphological Open):** Usuwa drobne artefakty i szum z otoczenia.

    Następnie algorytm wyszukuje kontury na masce. Zastosowano filtr wielkościowy (MIN_NOISE_AREA, MIN_AREA_DART oraz MAX_AREA_DART), który odrzuca zmiany zbyt małe (np. drgania) lub zbyt duże (np. wyciągnięcie wbitych lotek).



Rysunek 4 Obraz tarczy z wykrytą rzutką i jej maska

4. **Lokalizacja punktu wbicia (Tip Detection)** Dla wyodrębnionego konturu rzutki konieczne jest znalezienie punktu wbicia. W tym celu algorytm analizuje współrzędne wszystkich punktów tworzących kontur i wyszukuje ten o największej wartości na osi Y (najniższy punkt na obrazie). Jest to skuteczne założenie dla

standardowego rzutu, gdzie rzutka opada grawitacyjnie, a fakt obserwowania tarczy przez kamerę od spodu utrudnia wystąpienie innej sytuacji.

Rysunek 5 Rzutka z zaznaczonym bounding boxem i punktem wbicia

5. **Transformacja i obliczenie wyniku** Współrzędne wykrytego grota (w układzie kamery) są przekształcane przy użyciu macierzy perspektywicznej (uzyskanej podczas kalibracji) na współrzędne wirtualnej, płaskiej tarczy. Ostatnim etapem jest matematyczna konwersja układu kartezjańskiego (X, Y) na biegunowy (kąt i promień względem środka tarczy). Na podstawie odległości od środka (promień) określa się pierścień (Single, Double, Triple, Bull), a na podstawie kąta – sektor punktowy.
6. **Logika gry i obsługa gracza** System automatycznie aktualizuje wynik i sumuje punkty. Dodatkowo zaimplementowano logikę wykrywania końca rundy. Gdy pole powierzchni wykrytych zmian przekroczy zdefiniowany próg (MAX_PLAYER_AREA), system interpretuje to jako podejście gracza do tarczy w celu wyjęcia lotek, co skutkuje automatycznym resetem rundy i aktualizacją wzorca tła.

Podsumowanie

Początkowe plany projekty zakładały użycie metod uczenia maszynowego, np. sieci YOLO do wykrywania punktu wbicia lotki, ale zaprezentowane podejście okazało się wystarczająco skuteczne i dużo prostsze.

System został przystosowany do jednego nagrania, ale duża liczba edytowalnych parametrów powinna zapewnić możliwość dostosowania do innych stałych perspektyw nagrywania tarczy, innych warunków oświetleniowych oraz wyglądu samych lotek. Ponadto system z założenia powinien obsługiwać obraz w czasie rzeczywistym np. z kamerki internetowej, co czyni go przydatnym w grze w darta na żywo.

Największym ograniczeniem jest jedna perspektywa na tarczę, co sprawia problem z wzajemnym zasłanianiem się lotek.