

AKADEMIA GÓRNICZO-HUTNICZA

im. St. Staszica w Krakowie

WEAIiE, Katedra Automatyki i Robotyki

HCI2023\_4\_ozn

Przedmiot: Interfejsy człowiek-komputer

Temat projektu: Zawartość i forma raportu ze zrealizowanego projektu.

Spis treści:

Spis treści

1. [Abstrakt 2](#_Toc134145036)
2. [Wstęp i badania literaturowe 2](#_Toc134145048)
3. [Koncepcja proponowanego rozwiązania 3](#_Toc134145061)
4. [Realizacja zadań projektowych 4](#_Toc134145069)
5. [Podsumowanie i wnioski 5](#_Toc134145074)
6. [Literatura 6](#_Toc134145078)

Wykonali: Mikołaj Osiecki, Aleksander Brawański

EAIiIB rok *4 AiR*

konsultant: *Dr inż. Jaromir Przybyło*

Wersja 1.0

Kraków, maj 2023.

# Abstrakt

Celem projektu jest stworzenie aplikacji, która pozwoli na detekcję przeszkód w czasie rzeczywistym podczas wirtualnej gry ping-pong. Aby osiągnąć ten cel, zdecydowano się na wykorzystanie zewnętrznej kamery. Wykrywanie obiektów odbywa się dzięki zastosowaniu metody Feature Matching, której implementacja jest dostępna w środowisku programistycznym Python za pomocą biblioteki OpenCV.

Głównym zadaniem aplikacji jest przekazywanie w czasie rzeczywistym danych o położeniu oraz rodzaju wykrytej przeszkody na komputer sterujący grą korzystając z zaawansowanych technik przetwarzania obrazów. Dzięki temu, gracze będą mieli możliwość zmierzenia się z nowymi wyzwaniami urozmaicającymi rozgrywkę.

# Wstęp i badania literaturowe

Celem projektu jest stworzenie sterowania do gry „Pong” pozwalającego na detekcję przedmiotów lub kształtów umieszczonych, tak aby możliwe było przesłanie informacji na temat ich kształtu i położenia do głównej aplikacji gry. Wykryty obiekt jest następnie odwzorowywany na planszy gry i stanowi przeszkodę w trakcie rozgrywki. Ponieważ urządzeniem rejestrującym dane będzie kamera, należy przed rozpoczęciem detekcji, przeprowadzić jej kalibrację.

1. Możliwe sposoby detekcji oraz środowiska.

Rozpoznawanie kształtów lub przedmiotów może zostać zrealizowane na wiele sposobów, są to między innymi:

* + Detekcja krawędzi
  + Dopasowanie podobieństw (Feature Matching)
  + Wykrywanie obiektów za pomocą sieci neuronowej

Każda z tych metod może zostać zaimplementowana zarówno w środowisku MATLAB jak i Python.

Proponowane rozwiązanie:

Wybranym algorytmem detekcji zostało dopasowanie podobieństw. W porównaniu do algorytmu detekcji krawędzi jest on bardziej odporny na zniekształcenia obrazu takie jak cienie czy też niepożądane obiekty w polu widzenia kamery. Dodatkowo w przypadku błędnej detekcji lub jej całkowitego braku, algorytm dopasowania podobieństw nie będzie wysyłał złych danych do głównej aplikacji.

W porównaniu do wykrywania obiektów za pomocą sieci neuronowej, brak jest konieczności tworzenia zbiorów treningowych oraz czasochłonnego treningu samej sieci.

Środowiskiem wybranym do implementacji jest Python ze względu na bibliotekę OpenCV implementującą wiele funkcji potrzebnych do realizacji postawionego celu.

1. Podział zadań w grupie:
   * Mikołaj Osiecki – realizacja algorytmu kalibracji kamery oraz algorytmu detekcji obiektów
   * Aleksander Brawański – komunikacja z aplikacją główną gry

# Koncepcja proponowanego rozwiązania

Przed rozpoczęciem detekcji, należy dokonać kalibracji kamery. Dzięki temu będzie możliwe skorygowanie obrazu w celu usunięcia jak największej ilości zniekształceń w obrazie. W tym celu należy zaimplementować algorytm kalibracji kamery uruchamiany na początku programu.

Po poprawnej kalibracji kamery należy rozpocząć periodyczne zapisywanie obrazu z kamery. Okres między wykonaniem poszczególnych zdjęć powinien być na tyle duży by pozwolić na analizę obrazu oraz na grę z wykrytą przeszkodą – należy uniknąć sytuacji, w której gracze nie są w stanie wejść w żadną interakcję z przeszkodą ze względu na jej zbyt częstą zmianę.

Po wykryciu przeszkody należy rozpocząć analizę obrazu w celu detekcji określonych wcześniej przedmiotów oznaczających przeszkodę. W przypadku wykrycia kilku przeszkód – przykładowo z powodu błędnej detekcji – należy wybrać jedną np. poprzez porównanie ilości znalezionych podobieństw. Konieczne jest również ustawienie minimalnego progu detekcji, tak aby zminimalizować ryzyko wyników fałszywie pozytywnych.

Po zakończeniu procesu detekcji, jej wynik zostanie przesłany do aplikacji głównej gry. W skład przesłanych informacji powinno wchodzić:

* + Pozycja przeszkody
  + Rodzaj przeszkody

# Realizacja zadań projektowych

Algorytm kalibracji zawarty jest w klasie Calibration. Jej najważniejszym parametrem inicjalizacyjnym jest wartość boolowska określająca czy mają zostać wykonane nowe zdjęcia kalibracyjne. Jeśli podana zostanie wartość “True”, program wykona zadaną przez użytkownika liczbę zdjęć, w przeciwnym razie do procedury kalibracji wykorzystane zostaną zdjęcia aktualnie znajdujące się w wybranym folderze. Wśród tychże zdjęć algorytm poszukiwał będzie takich, na których widoczna jest szachownica o wymiarze 4 na 6 pól. Ze zdjęć w których udało się znaleźć szachownicę, zostanie następnie wylosowane jedno wobec którego zostanie przeprowadzona kalibracja. Algorytm zwraca takie współczynniki jak współczynnik zniekształcenia czy optymalna macierz kamery, które to będą w późniejszej części programu używane do poprawy jakości zdjęcia, na którym ma być wykryty żądany obiekt.

W klasie Matcher znajduje się implementacja algorytmu wyszukiwania podobieństw. Porównuje on podane zdjęcie z szablonami. Jeśli algorytm znajdzie liczbę podobnych cech w obrazach przekraczającą ustalony wcześniej próg, zaliczy je jako obrazy podobne oraz zinkrementuje licznik trafień. Po porównaniu obrazu wejściowego ze wszystkimi szablonami, zwrócona zostanie wartość licznika trafień oraz licznika podobieństw zliczającego liczbę wszystkich znalezionych trafień między obrazem a szablonami. Jest on przeskalowany do ilości szablonów, aby zapobiec oszukiwaniu algorytmu poprzez zwiększanie ilości szablonów

Cały proces wykonywania zdjęć oraz analizy został zamknięty w osobne wątki - wątek wykonywania zdjęć, oraz osobny wątek dla każdego rodzaju wykrywanej przeszkody. Dzięki temu proces analizy może zostać wykonany równolegle, znacząco przyspieszając pracę programu. Wątki analizy oczekują na sygnał od wątku wykonywania zdjęć, aby zapobiec wtórnej analizie tego samego obrazu oraz dla zaoszczędzenia mocy obliczeniowej podczas czekania na kolejne zdjęcie.

Wybór wybranej przeszkody następuje tuż przed wykonaniem nowego zdjęcia. Takie podejście sprawia, że sygnał o przeszkodzie jest wysyłany zawsze po tym samym okresie czasu, niezależnie od czasu analizy zdjęcia. W najgorszym przypadku, czyli takim w którym analiza obrazu nie zakończyła się przed wykonaniem kolejnego zdjęcia, wybrana zostanie przeszkoda na podstawie danych z poprzedniej analizy.

Komunikacja z grą odbywa się poprzez protokół REST API. Jest to najlepszy sposób pozwalający na przesyłanie danych w czasie rzeczywistym między klientem i serwerem, niezależnie od platform i języka programowania.

# Podsumowanie i wnioski

Założenia projektowe zostały zrealizowane, dzięki aplikacji można sprawnie rozpoznawać przeszkody oraz przesyłać uzyskane informacje do komputera sterującego grą.

Wybrana metoda detekcji obrazu to metoda wykrywania podobieństw (Feature Matching) ze względu na oszczędzanie zasobów, większą dokładność w porównaniu do innych metod oraz prostotę implementacji. W celu przyspieszenia procesu zdecydowano się na wielowątkowość aplikacji. Wprowadzono również zabezpieczenie przed wysyłaniem błędnych danych (lub ich braku) do aplikacji sterującej.

Środowisko Python w połączeniu z odpowiednimi bibliotekami pozwala na sprawne tworzenie i testowanie kodu. Dzięki algorytmom biblioteki OpenCV można w efektywny sposób stworzyć aplikację pozwalającą na wykrywanie i analizowanie obiektów na obrazach pozyskiwanych z kamery.

# Literatura

[1] L. Feng, X. Qu, X. Ye, K. Wang and X. Li, "Fast Feature Matching in Visual-Inertial SLAM," 2022 17th International Conference on Control, Automation, Robotics and Vision (ICARCV), Singapore, Singapore, 2022, pp. 500-504, doi: 10.1109/ICARCV57592.2022.10004220.