Sterowanie za pomocą gestów

Damian Kogut, Szymon Stelmach

Politechnika Wrocławska, Wrocław, Polska

Abstrakt. Sterowanie za pomocą gestów oferuje interesujące możliwości w zakresie interakcji pomiędzy człowiekiem a komputerem (HCI). Rozpoznawanie gestów mimo, iż często stosowane nie należy do zadań trywialnych, gdzie chęć wykonywania tego w czasie rzeczywistym wymaga od nas zastosowania ulepszonych wersji wcześniej ukształtowanych metod. W tym artykule zaprezentowane zostaną techniki oraz metody pozwalające nam na rozpoznawanie gestów w czasie rzeczywistym. Dodatkowo połączymy owe techniki wraz z serwisem YouTube realizując w ten sposób sterowanie przy pomocy rozpoznanych gestów. Pojedyncza dłoń jest obserwowana przez pojedynczą kamerę, gdzie sylwetka dłoni jest ekstrahowana przy pomocy metod segmentacji. Na sylwetkę dłoni nałożona zostaje otoczka wypukła pozwalając na śledzenie ułożenia palców dłoni. Na podstawie ilości podniesionych palców determinujemy dany gest. Informacja o rozpoznanym geście jest przekazywana do wykorzystywanego modelu umożliwiając sterowanie.

Słowa kluczowe: Rozpoznawanie gestów, Interakcja pomiędzy człowiekiem a komputerem, Otoczka Wypukła.

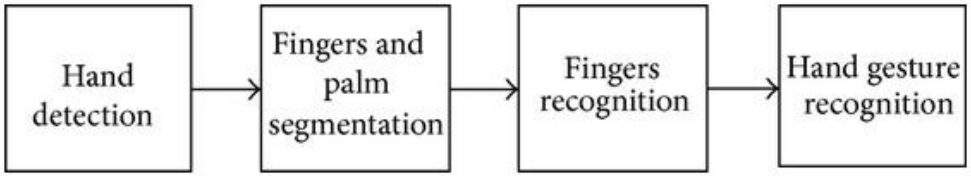
1 Wprowadzenie

Rozpoznawanie gestów ewoluowało na przestrzeni ostatnich lat poczynając od specjalistycznych rękawic zakładanych na dłoń do czysto wizyjnych systemów umożliwiających rozpoznawanie. Wizualna interpretacja gestów dłoni umożliwia nam przeprowadzenie bezkontaktowych interakcji pomiędzy człowiekiem a komputerem. Wiele rozbudowanych modeli zawiera dużo informacji wykraczających poza obszar zainteresowania z poziomu aplikacji. Prócz dłoni, środowisko zawiera czynniki komplikujące proces detekcji gestów, takie jak zatłoczone tło czy też brak odpowiedniego oświetlania. Dodatkowym aspektem, na który należy zwrócić uwagę jest odcień skóry, który nie może się zlewać z aktualnym tłem. W podejściu wizyjnym skupiamy się na wydobyciu informacji z przekazu wideo pozwalającej nam na analizę dłoni w celach detekcyjnych.

W tym artykule chcemy zaprezentować metody pozwalające na rozpoznawanie gestów w czasie rzeczywistym oparte na badaniu położenia palców. Następnie zdobyte informacje będą łączone z serwisem YouTube tworząc bezdotykowy system interakcji pomiędzy człowiekiem a komputerem. Reszta pracy została zorganizowana w następny sposób: ogólny przegląd systemu zostanie przedstawiony w sekcji 2, sekcja 3 skupia się na wyjaśnieniu technik detekcyjnych, a sekcja 4 dokładnie opisuje proces segmentacji dłoni. Sekcja 5 przedstawia metody rozpoznawania palców, a sekcja 6 przedstawia rozpoznawanie gestów. Przeprowadzone eksperymenty zostaną przedstawione w sekcji 7, natomiast drobne podsumowanie pracy zostanie opisane w sekcji 8.

2 Przegląd systemu

Na samym początku wykonujemy detekcję dłoni przy wykorzystaniu kamery internetowej. Na tym etapie wykonywana jest czynność separacji pierwszego planu od planu drugiego. Potem następuje segmentacja palców oraz dłoni. W tym etapie otaczamy dłoń otoczką wypukłą. Następnie korzystając z metody defektów wypukłości bądź też punktów ekstremalnych określamy położenie palców. Ostatnim elementem jest rozpoznanie gestu badając ilość podniesionych palców a następnie przekazanie informacji do serwisu YouTube.



Rysunek 1: *Przegląd systemu detekcji*

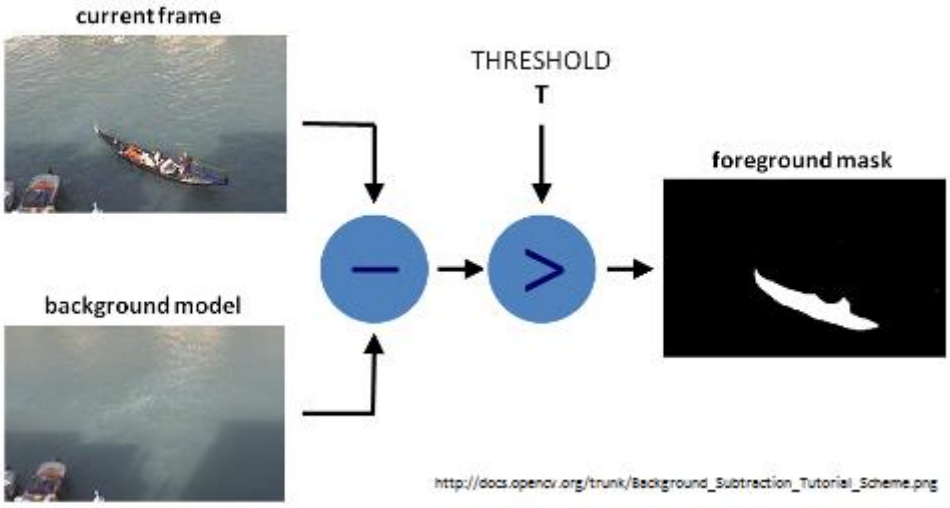
3 Detekcja dłoni

Detekcja dłoni jest niezbędnym czynnikiem w całej pracy, ponieważ reszta procesu polega na informacjach uzyskanych w pierwszym etapie. Metoda, którą wykorzystaliśmy w pracy jest połączeniem ekstrakcji tła wraz z przekształceniem obrazu do odcieni szarości oraz zastosowaniem szumu gaussowskiego w celach eliminacji niechcianych artefaktów na obrazie.

Do przeprowadzenia operacji ekstrakcji tła wykorzystaliśmy metodę binaryzacji Otsu. Polega ona na konwersji obrazu w odcieniach szarości do obrazu binarnego. Jest to metoda progowania globalnego, oparta na histogramie. Polega na minimalizacji sumy ważonej wariancji dwóch klas – tła oraz obiektów znajdujących się na pierwszym planie, co przekłada się na maksymalizację wariancji międzyklasowej. Obraz traktujemy jako zbiór pikseli, gdzie każdy z nich przyjmuje wartość z pewnego zbioru $[1,2,\dots,L]$. Następnie sporządzamy histogram obrazu, a liczebności poszczególnych klas oznaczamy jako $[n_1, n_2, \dots, n_L]$. Łączną liczbę pikseli oznaczamy jako N. Naszym zadaniem jest znalezienie odpowiedniej wartości progu k. Wszystkie piksele o wartości większej od progu będziemy przypisywać do jednej klasy, natomiast pozostałe do drugiej. Optymalnym progiem k jest taki, dla którego suma ważona wariancji wewnątrzklasowej jest najmniejsza. Aby znaleźć wartość optymalną sprawdzamy wszystkie możliwe wartości progów. Całość formalnie została spisana przy pomocy następującego wzoru:

$$\sigma_B^2(k^*) = \max_{0 \le k < L} \sigma_B^2(k), gdzie\ \sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2$$

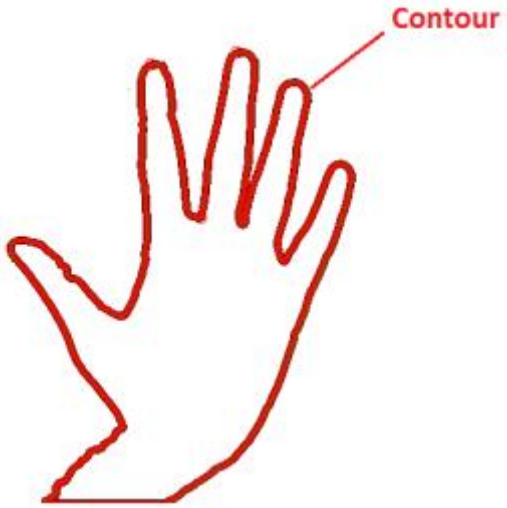
Efekt tej metody można zaobserwować na poniższym obrazku:



Rysunek 2: *Ekstrakcja pierwszego planu od tła.*

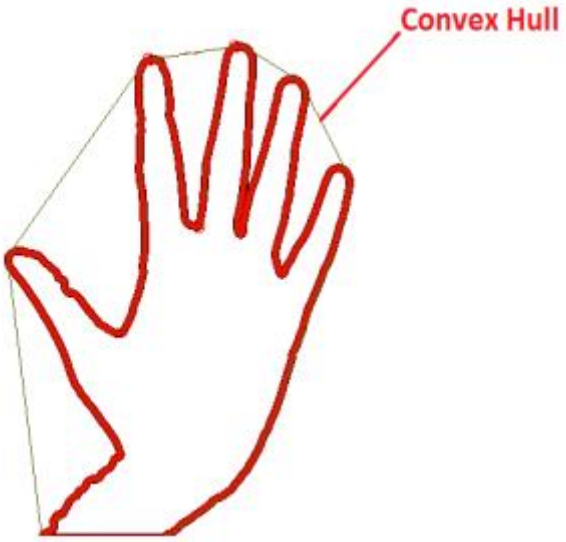
4 Segmentacja dłoni

Do segmentacji dłoni wykorzystaliśmy konturowanie oraz metodę otoczki wypukłej. Kontur obrysowywał biały element otrzymywany w wyniku ekstrakcji tła z obrazu metodą binaryzacji Otsu. Dzięki zastosowaniu szumu gaussowskiego pozbyliśmy się niechcianych artefaktów na obrazie przez co drobne zakłócenia w wersji finalnej nie były obrysowane co utrudniałoby dalszy przebieg procesu. Konturowanie kształtu dłoni wygląda następująco:



Rysunek 3: *Kontur dłoni*

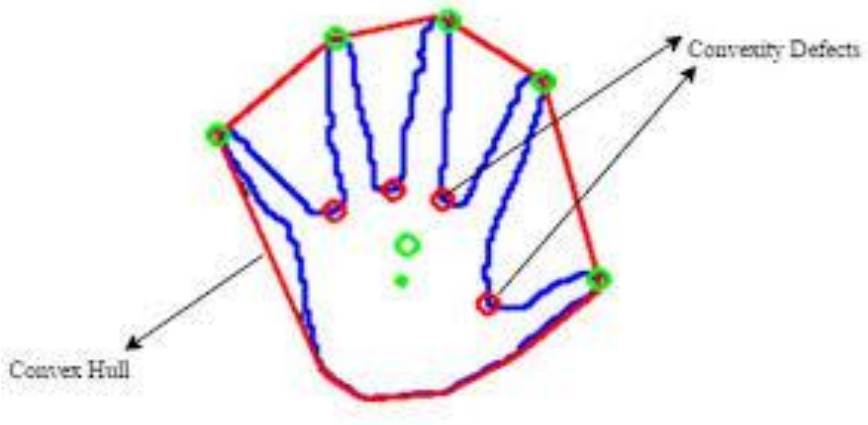
Powyższy kontur wykorzystujemy do metody otoczki wypukłej. Wypukła otoczka jest to najmniejszy wielokąt wypukły taki, że każdy punkt z danego zbioru będzie leżał albo na brzegu wielokąta albo w jego wnętrzu. Otoczenie konturu dłoni otoczką wypukłą wygląda następująco:



Rysunek 4: *Otoczenie konturu otoczką wypukłą*

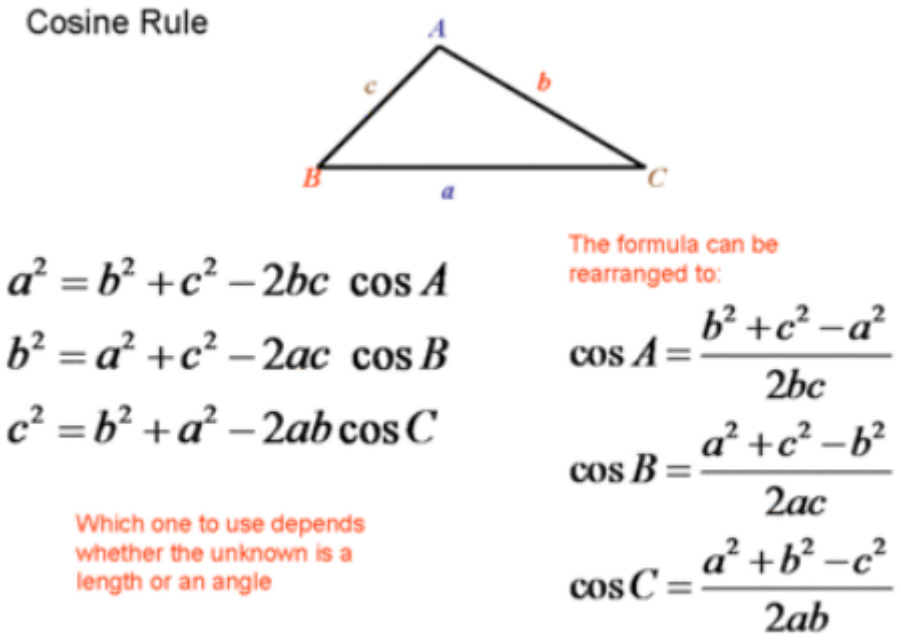
5 Rozpoznawanie palców

Rozpoznawanie palców w naszej pracy zrealizowaliśmy na dwa sposoby. Sposób pierwszy polega na wykorzystaniu defektów wypukłości. Defekt jest wykrywany, jeżeli kontur okrywanego obiektu jest oddalony od otoczki wypukłej. W skrócie można powiedzieć, iż są to najgłębsze punkty odchyleń naszej otoczki. Defekty te zostały zaznaczone na poniższym obrazku przy pomocy czerwonych kółek.



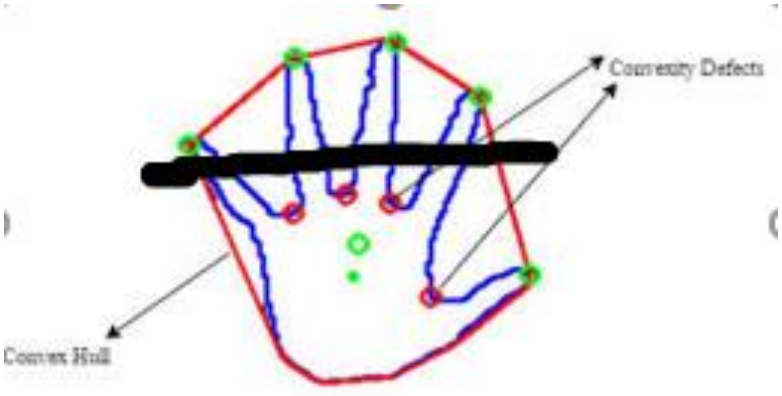
Rysunek 5: *Defekty wypukłości*

W poszukiwaniu defektów posłużyliśmy się metodą kosinusów pozwalającą nam na obliczenie kąta pomiędzy palcami. Jeżeli znaleziony kąt był mniejszy niż 90 stopni był on uznawany za defekt. Dzięki temu na podstawie ilości znalezionych defektów określamy aktualnie pokazywany gest



Rysunek 6: *Reguła kosinusów.*

Metoda druga polega na wykorzystaniu punktów ekstremalnych naszej otoczki. Dzięki tej metodzie wyznaczamy środek dłoni poprzez dodanie ekstremum z lewej oraz prawej, a następnie podzielenie przez dwa. Następnie tworzymy linię, powyżej naszego środka dłoni i badamy, ile razy owa linia została przecięta. W ten sposób mamy dokładne informacje na temat ilości podniesionych palców co później przekładamy na odpowiedni gest.



Rysunek 7: *Badanie położenia palców z wykorzystaniem punktów ekstremalnych.*

6 Rozpoznawanie gestów

W naszej pracy wykorzystując wcześniej przedstawione metody rozpoznajemy 6 gestów. Całość odbywa się na podstawie ilości podniesionych palców. Ilość palców jest liczona zarówno na podstawie defektów wypukłości jak i punktów ekstremalnych badających ilość przecięć. Wybrane gesty oraz odpowiadająca im ilość policzonych palców prezentują się następująco:

- 0 palców – skała
- 1 palec – wskazywanie
- 2 palce – znak pokoju
- 3 palce – znak głównej bohaterki z filmu Igrzyska Śmierci
- 4 palce – znak drużyny baseballowej z Nowego Yorku
- 5 palców – cześć, znak powitania

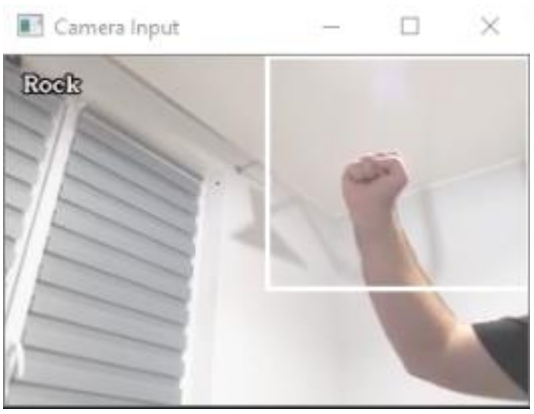
Następnie wybraliśmy 3 gesty, którym została przypisana funkcjonalność serwisu YouTube. Do generowania wydarzeń wykorzystaliśmy bibliotekę pyautogui, która umożliwia symulację klawiatury. Dzięki temu po wykryciu odpowiedniego gestu do systemu była przekazywana informacja o naciśnięciu przycisku.

7 Eksperymenty

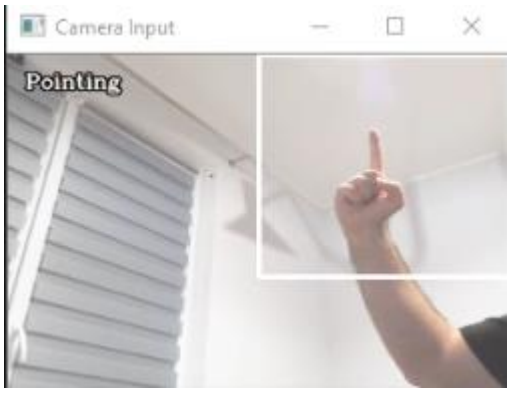
System został zaimplementowany na komputerze stacjonarnym o następującej specyfikacji:

- Intel Core I7-7700K
- Nvidia GeForce GTX 1070
- 16 GB pamięci RAM
- Windows 10

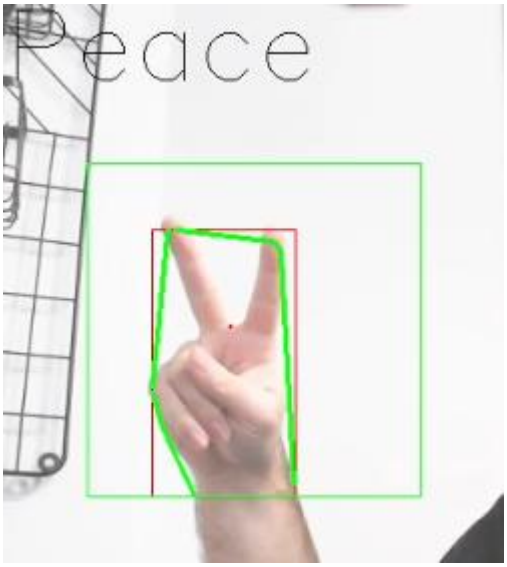
Za sprzęt do przechwytywania obrazu wideo wykorzystaliśmy kamerę internetową firmy Logitech. Testy wykrywania poszczególnych gestów przeprowadziliśmy zgodnie z kolejnością wymienioną w 6.



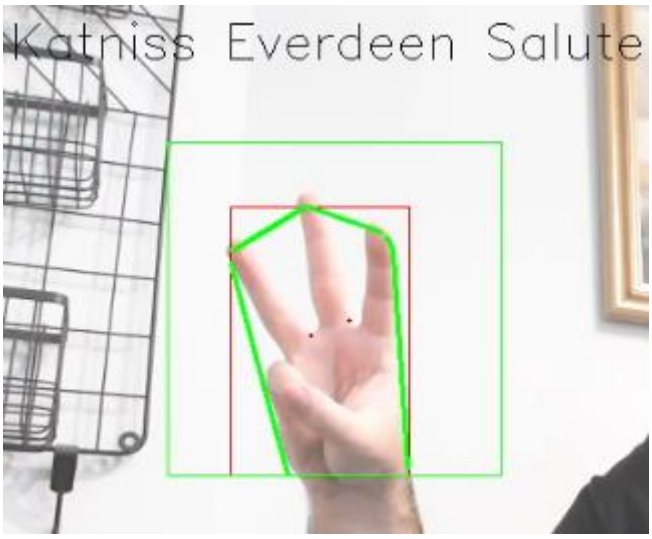
Rysunek 8: *Gest – skała.*



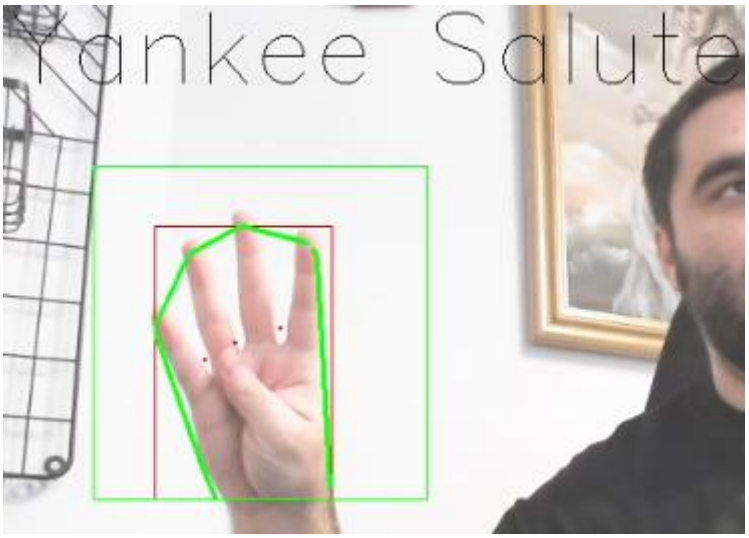
Rysunek 9: *Gest wskazywania.*



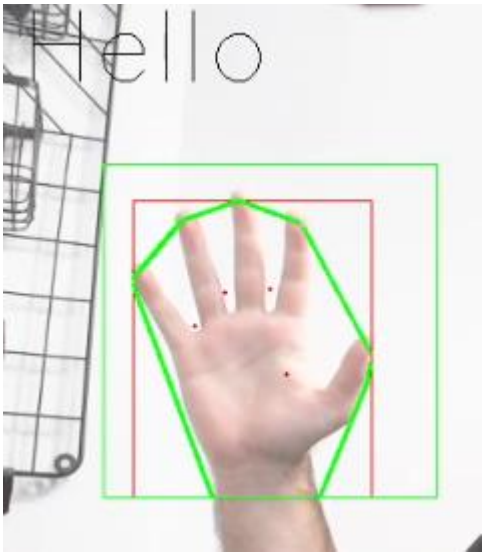
Rysunek 10: *Znak pokoju.*



Rysunek 11: *Gest Katniss Everdeen.*



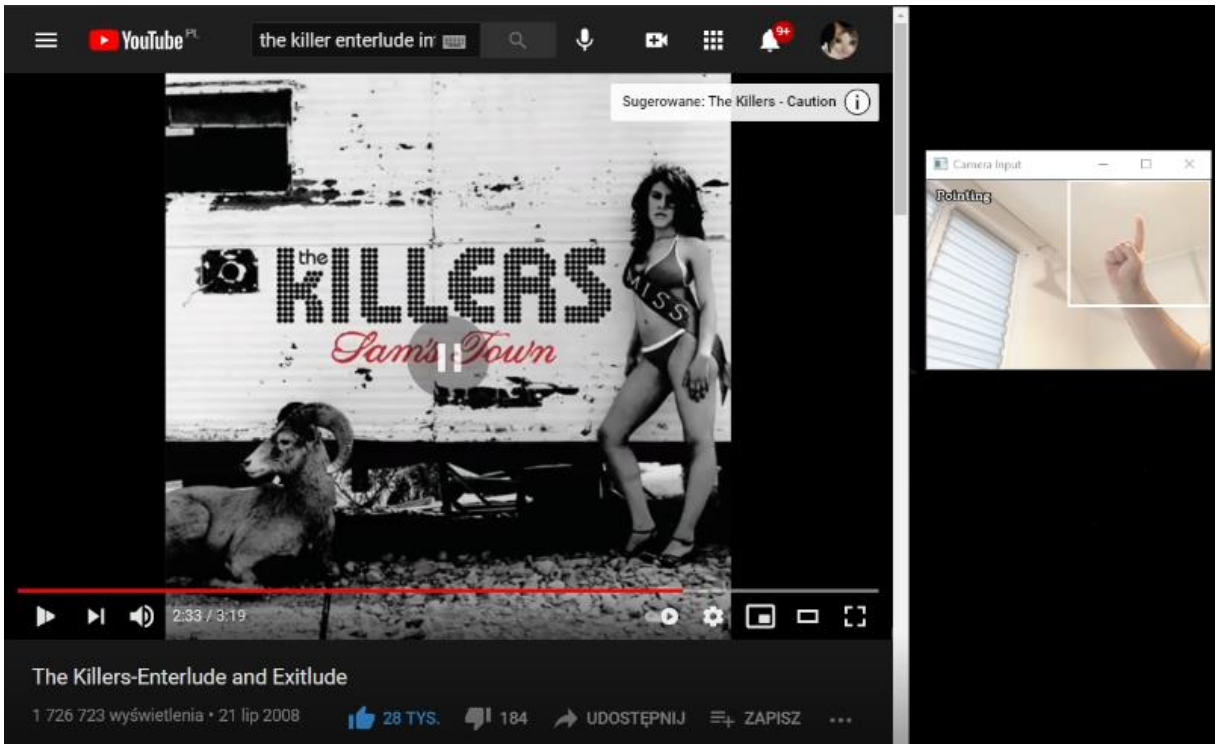
Rysunek 12: *Znak New York Yankees.*



Rysunek 13: *Gest powitalny.*

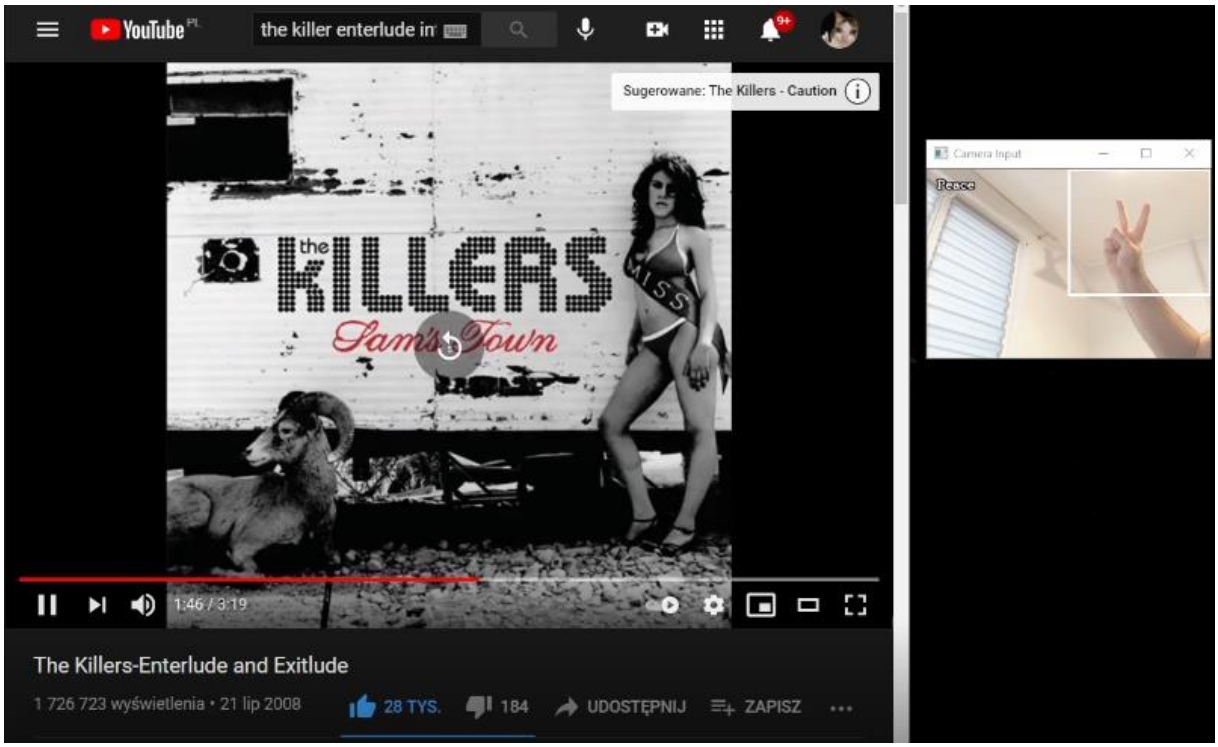
Ostatnim etapem eksperymentów było przetestowania sterowania przy pomocy gestów na serwisie YouTube. Ze względu na charakter działania naszego systemu postawiliśmy na prostą funkcjonalność pozwalającą na zatrzymywanie/odtwarzanie filmu oraz przesuwanie do przodu i cofanie. Gesty, które do tego wykorzystaliśmy prezentują się następująco:

- Gest wskazywania -> odtwarzanie/zatrzymywanie (przycisk na klawiaturze ‘k’)
- Znak pokoju -> cofanie filmu o 5 sekund (strzałka w lewo)
- Gest Katniss Everdeen -> przewijanie filmu do przodu o 5 sekund (strzałka w prawo)



Rysunek 14: *Odtwarzanie/zatrzymywanie.*

Rysunek 15: *Przewijanie do przodu.*



***Rysunek 16**: Cofanie filmu.*

8 Podsumowanie

Jak większość systemów zajmujących się rozpoznawaniem gestów, nasze podejście również nie było pozbawione wad dotyczących zarówno użytkownika jak i otoczenia. Kamera musi być statyczna oraz na przechwytywanym obrazie może znajdować się tylko jedna dłoń. Dodatkowo dłoń użytkownika nie powinna obracać się zbyt mocno co może powodować komplikacje związane z procesem detekcji. Dłoń wykonująca gest musi pozostać statyczna przez krótki okres, aby dany gest został rozpoznany. Najlepsze efekty uzyskuje się poprzez eliminację zatłoczenia tła (przykładowo: szafki, książki, obrazy) oraz przy wykorzystaniu jednokolorowego, stonowanego tła. Sterowanie modelem, również nie było pozbawione wad. Z racji tego, iż system działał w czasie rzeczywistym rozpoznany gest generował więcej niż jedno wydarzenie przez co podczas testów można było zauważyć częste kilkukrotne wykonywanie tej samej czynności. Dodatkowo wykorzystywana biblioteka wprowadzała pewne opóźnienia, co skutkowało dłuższą symulacją naciśnięcia przycisku co wpływało negatywnie na jakość sterowania.

9 Bibliografia

1. Gogul Ilango: Hand Gesture Recognition using Python and OpenCV (Kwiecień 2017)

2. Madhav Mishra: Hand Detection and Finger Counting Using OpenCV-Python (Sierpień 2020)

3. Karan Gupta: Python OpenCV: Capture Video from Camera (Styczeń 2020)

4. Aashni Haria, Archanasri Subramanian: Hand Gesture Recognition for Human Computer Interaction (Sierpień 2017)

5. Usama Sayed, Mahmoud A. Mofaddel, Samy Bakheet i Zenab El-Zohry: Human Hand Gesture Recognition (Wrzesień 2018)