Podstawy uczenia maszynowego

Sterowowanie prezentacją slajdów w programie Google Slides za pomocą gestów dłoni

Konrad Sochacki 227958

Wstęp

Celem projektu było wykonanie programu, zdolnego przetwarzać obraz z kamerki tak, aby za pomocą gestów dłoni można było sterować prezentacją w programie Google Slides.

Wykorzystane technologie i bilbioteki:

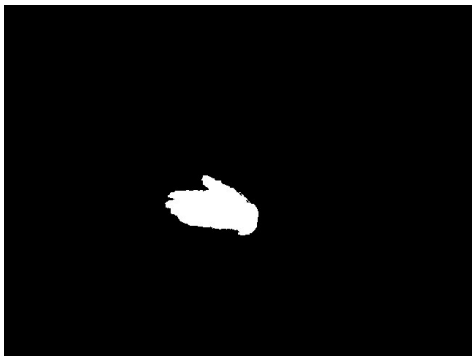
- DroidCam – aplikacja która pozwala na podłączenie kamerki z Androida do komputera i wygodne przychwytywanie obrazu.
- Python i jego biblioteki:
  - cv2 – przetwarzanie obrazu
  - numpy – zaawansowane obliczenia
  - pynput – sterowanie klawiaturą i myszką z poziomu pythona
  - sklearn – implementacje metod uczenia maszynowego
- Google Slides – internetowy program do prezentacji sladjów, wybrany ze względu na równą dostępność dla każdego systemu operacyjnego za pośrednictwem przeglądarki internetowej.

Funkcjonalność

Sterowanie programem jest możliwe za pomocą dłoni w zielonej rękawiczce. Umożliwia ona wygodne wycięcie maski interesującego nas kształtu z wybranego koloru. Ważne jest również oświetlenie i kalibracja zakresu jasności naszej rękawiczki.

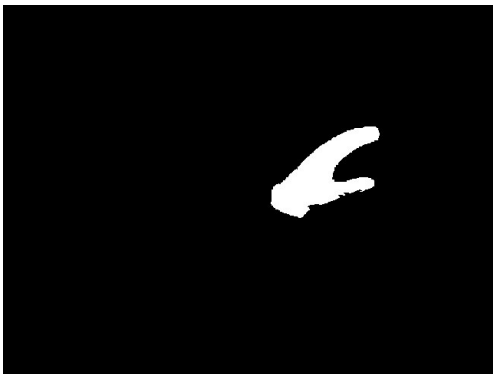
Dostępne jest 7 gestów sterowania:

- „close” - zamknięta pięść oznacza pozycję neutralną, nie nadaje żadnego sygnału. Analogia do „luzu” w skrzyni biegów.

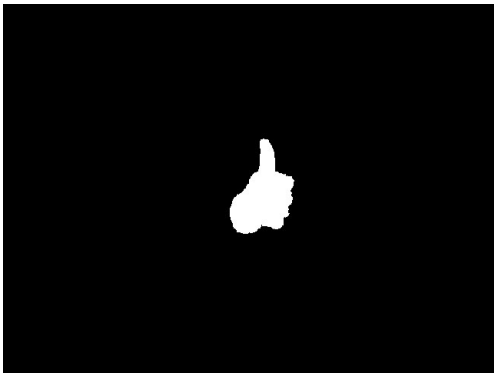


- „left” - dłoń obrócona w lewo z perspektywy kamery (w prawo z perspektywy użytkownika) – przejście o jeden slajd do przodu

- „right” – dłoń obrócona w prawo z perspektywy kamery (w lewo z perspektywy użytkownika) – przejście o jeden slajd do tyłu



- „play” - odtwórz media / zatrzymaj odtwarzanie. Włączenie filmiku czy muzyki zamieszczonej na prezentacji. Powtórne pokazanie gestu zatrzymuje odtwarzanie.



- „volume up” - zgłaśnianie dźwięku

- „volume down” - ściszanie dźwięku

- „pointer” - laserowy wskaźnik, najciekawsza funkcja. Po wykonaniu takiego gestu na ekanie pojawia się czerwony laserowy wskaźnik, a następnie kiedy użytkownik porusza ręką to wskaźnik odwzorowuje jego ruchy na ekranie. Punkt wskaźnika jest wyznaczany jako wierzchołek palca wskazującego.

Z perspektywy użytkownika oprócz samych gestów ważne jest również to jak „dokładnie” znak musi być pokazany. Zakładamy, iż użytkownik niekoniecznie będzie chciał trzymać dłoń w pozycji neutralnej zawsze wtedy kiedy nie chce przekazać żadnego sygnału. Wiele osób lubi gestykulować podczas prezentacji. Możemy zatem sterować tym jak bardzo dokładnie dany znak musi zostać pokazany aby był uznany oraz ile takich samych znaków pod rząd musi wystąpić aby zostały zakwalifikowane, za pomocą parametrów.

Implementacja

Projekt ma strukturę osobnych skryptów w Pythonie realizujących poszczególne funkcjonalności niezbędne do uzyskania finalnego modelu.

1. photograph.py – prosty skrypt umożliwiający szybkie pozyskanie zdjęć. Wybieramy folder zapisu i za pomocą spacji robimy kolejne zdjęcia. Dla każdego gestu wykonano 100 zdjęć.
2. image_utils – zbiór funkcji do przetwarzania obrazów. Przede wszystkim zbiera wszystkie potrzebne obrazy, podzielone na kategorie zdefiniowane w pliku constants.py, wyciąga z nich białą maskę na czarnym tle interesującego nas kształtu dłoni oraz z tak przetworzonego obrazu wyciąga cechy niezbędne dla uczenia maszynowego.

   Przetwarzanie zwykłego zdjęcia na czarno białą maskę to:

   - zamiana przestrzeni kolorów z BGR do HSV
   - określenie zakresu kolorów, które nas interesują (w przestrzeni HSV)
   - progowanie obrazu za pomocą zdefiniowanych zakresów
   - znalezienie wszystkich białych konturów na obrazie
   - wyciągnięcie najdłuższego z nich i narysowanie go na czarnym tle z wypełnionym środkiem

   Ekstraktowane cechy z kształtu to:

   - powierzchnia
   - obwód
   - „Central Moments”
   - „Moments”
   - „Hu Moments”
3. Main2.py – skrypt w którym odbywa się zasadnicza część uczenia maszynowego. Ładuje odpowiednio przetworzone obrazy, cechy i ich etykiety. Normalizuje dane – od każdej próbki odejmuje średnią wszystkich i dzieli przez ich odchylenie standardowe. Dzięki temu zakres wielkości żadnej z nich nie dominuje nad inną. Następnie wykonuje redukcję cech za pomocą metody PCA. Jest to istotne ponieważ jeśli wykonalibyśmy uczenie na początkowych 35 cechach to prawdopodobnie układ „przeuczyłby” się (ang. overfitting) tzn. dopasował jedynie do podanych mu przykładów i miał problem z obrazami spoza zbioru treningowego. PCA redukuje 35 początkowych cech do 9. Ostatni i najważniejszy etap to uczenie właściwe. Odbywa się ono za pomocą metody SVC pakietu sklearn, który jest w praktyce szybszą i lepiej skalowalną implementacją klasycznego SVM. Zwraca on konkretny model zdolny określać etykiety nowych obrazów na podstawie przetworzonych cech. W praktyce będziemy chcieli taki model wczytać w dowolnym momencie z dysku i podać mu dowolny obraz z dłonią w zielonej rękawiczce, dlatego oprócz samego modelu potrzebujemy również informacji do normalizacji i redukcji cech. W tym celu stworzono klasę agregującą te składniki, która przechowuje nauczony model SVC, model pca, średnią z próbek przed normalizacją i ich odchylenie standardowe.
4. camera_controller_py – finalny program obsługujący sterowanie programu. Wczytuje z dysku zadany mu wyuczony model klasyfikatora i uruchamia kamerkę. Zczytuje obrazy, zamienia je na czarno białe maski i daje klasyfikatorowi do określenia etykiety. Klasyfikator oprócz określenia samej najbardziej prawdopodobnej etykiety zwraca listę prawdopodobieństwa przynależności do wszystkich etykiet. Za pomocą odpowiedniego parametru możemy sterować progiem kiedy gest uznajemy za poprawny. Następnie program

zlicza wystąpienie takich gestów i jeżeli wystąpi określona ich ilość to uruchamia akcję dla danego gestu.

Wyjątkiem jest pointer, ponieważ aby uzyskać płynne poruszanie kursora potrzebuje on zdecydowanie mniejszego progu ilości wystąpień. Jego implementacja zawiera również nieco więcej przetwarzania obrazu niż dla innych gestów. Po wykryciu pointera uruchamiana jest funkcja która określa najbardziej wysunięty punkt czyli koniec palca wskazującego. Następnie określa w jakim położeniu jest w stosunku do całego pola poruszania dłonią i mapuje to na obraz monitora. W ten sposób w wygodnym zakresie poruszania dłonią możemy wskazać na dowolne miejsce na ekranie.

Wyniki

Sam wynik nauki klasyfikatora był bardzo dobry – 100% lub bardzo bliski, rzędu ~99%. W praktyce jednakże sprawdzało się to różnie. Program działa znakomicie w wersji bez wskaźnika laserowego. W wersji ze wskaźnikiem ma drobne problemy z wykryciem gestu „right”. Wynika to zapewne z tego, że są nieco podobne ze znakiem wskaźnika. Możemy obniżyć próg pewności przydziału do etykiety, czym ułatwimy modelowi wykrycie gestu „right”, ale tym samym ryzykujemy zbyt „pochopne” wykrywanie innych gestów.

Ważną kwestią jest również poza samymi znakami jest również oświetlenie i wycinanie odpowiedniej maski. Dla ułatwienia prowadziłem projekt w pokoju z zasłoniętymi żaluzjami z takim samym oświetleniem. Przy zmianie pojawiały się problemy z detekcją gestów.

Z ciekawości sprawdziłem jak radzą sobie prostsze metody – redukcja cech za pomocą metody SelectKBest pakietu sklearn oraz uczenie metodą KNN. Liczyły się zauważalnie dłużej i chociaż radziły sobie z rozróżnianiem dwóch gestów (98%), tak dalej dokładność drastycznie spadała ~85%, ~70%, ~60% itd.

Przyszłość projektu

Projekt ma spore pole do rozwoju. Przede wszystkim warto „zdjąć rękawiczkę” oraz przystosować go do różnych rodzajów oświetlenia, tła oraz pozycji użytkownika. Należałoby poprawić precyzję klasyfikacji i zastanowić się nad dostosowaniem progów detekcji dla standardowego użytkownika. Dalej, zweryfikować czy baza danych została wykonana poprawnie. Można również dodać obsługę innych programów do prezentacji np. PowerPointa, jest to jednakże bardziej inżynierska, a mniej naukowa część pracy. Funkcjonalna część projektu będzie kontynuowana w ramach przedmiotu Głębokie Sieci Neuronowe.