

HandTone

Magdalena Hermann

May 2024

1 Cel projektu

Mój projekt polega na stworzeniu modelu, który rozpoznaje konkretne gesty mojej reki i przypisuje im odpowiednie nuty muzyczne, które są odtwarzane.

2 Zawartość repozytorium

2.1 Pliki dotyczące dźwięku

2.1.1 Tone.py

W tym pliku skupiamy się na tworzeniu dźwięku za pomocą bibliotek pygame, numpy i math. Używamy wzorów matematycznych, aby generować różne kształty fal dźwiękowych, takie jak sinusoida. Następnie tworzymy klasę Tone, która w swojej funkcji play wykorzystuje wcześniej określone wzory do generowania dźwięków o określonej częstotliwości.

2.1.2 note_map.json oraz utils.py

W pliku note_map.json znajduje się lista z określonymi częstotliwościami dla poszczególnych dźwięków. Dzięki temu, odwołania do dźwięków będą bardziej czytelne, a my będziemy mogli łatwo uzyskać częstotliwość odpowiedniego dźwięku, korzystając z jego nazwy. Plik utils.py używa modułu json do odczytania danych z pliku note_map.json, które są przechowywane w zmiennej NOTE_MAP w postaci słownika.

2.1.3 Note.py

Plik Note.py zawiera definicje klasy Note, która umożliwia granie nut poprzez wykorzystanie klasy Tone oraz utils.py. Pozwala to precyjnie określić długość nuty, jej częstotliwość oraz rodzaj fali akustycznej.

2.2 Pliki dotyczące gestu

2.2.1 collect_img.py

Ten kod służy do zbierania danych obrazowych do klasyfikatora. Importuje moduł os, który umożliwia interakcje z systemem plików. Tworzy katalog DATA_DIR ('data'), jeśli nie istnieje. Następnie inicjuje kamery za pomocą OpenCV. W pętli for tworzony jest folder dla każdej klasy (zakładamy, że jest 7 klas), jeśli jeszcze nie istnieje. W każdej iteracji użytkownik musi nacisnąć klawisz 'q', aby rozpocząć zbieranie danych dla danej klasy. Po naciśnięciu klawisza 'q' rozpoczyna się zbieranie danych. Obrazy są przechwytywane z kamery i zapisywane w folderze odpowiadającym danej klasie, z odpowiednimi nazwami plików. Proces zbierania danych kończy się po zebraniu określonej liczby obrazów dla każdej klasy (dataset_size).

2.2.2 create_dataset.py

Ten kod importuje moduł os do interakcji z systemem plików oraz moduł pickle do serializacji danych. Następnie importuje niezbędne biblioteki takie jak mediapipe, cv2 (OpenCV) oraz matplotlib.pyplot. Tworzona jest instancja Hands z mediapipe, która będzie wykorzystywana do wykrywania punktów kluczowych na dloniach. W pętli for przeglądane są wszystkie podkatalogi w katalogu DATA_DIR. Dla każdego podkatalogu przetwarzane są obrazy znajdujące się wewnątrz, a dane o punktach kluczowych na dloniach są zbierane do listy data. Na koniec dane są serializowane do pliku za pomocą pickle w formacie data.pickle

2.2.3 train_classifier.py

Ten kod importuje moduł pickle, który umożliwia serializację i deserializację obiektów. Następnie importowane są klasy RandomForestClassifier, train_test_split i accuracy_score z modułu sklearn.ensemble oraz sklearn.model_selection. Dane są wczytywane z pliku za pomocą pickle i przypisywane do zmiennej data_dict. Dane i etykiety są przekształcane do postaci tablic numpy. Następnie dane są podzielone na zbiór treningowy i testowy za pomocą train_test_split, a model RandomForestClassifier jest trenowany na zbiorze treningowym. Po przeprowadzeniu predykcji na zbiorze testowym, obliczana jest dokładność klasyfikacji za pomocą funkcji accuracy_score. Na koniec model jest zapisywany do pliku 'model.p' za pomocą pickle.

2.2.4 inference_classifier.py

Ten program wczytuje zserializowany model z pliku model.p i przypisuje go do zmiennej model. Inicjuje połączenie z kamerą za pomocą cv2.VideoCapture(0). Następnie inicjuje obiekt Hands z biblioteki MediaPipe, który wykrywa położenie dloni na obrazie. Definiuje parametry dotyczące generacji dźwięku, takie jak typ fali i oktawie. W nieskończonej pętli program odczytuje ramki z kamery, przetwarza je za pomocą MediaPipe w celu wykrycia dloni i rysuje na nich

punkty reprezentujace położenie dloni. Nastepnie oblicza różnicę w położeniach punktów dloni względem ich minimalnych wartości (min-max scaling) i tworzy wektor cech data_aux. Sprawdza długość wektora cech i w razie potrzeby uzupełnia go zerami lub skraca do odpowiedniej długości. Dokonuje predykcji nuty za pomocą wcześniejszej wczytanego modelu. Na podstawie predykcji określa nazwe nuty i odtwarza ją za pomocą klasy Note. Na końcu rysuje prostokat wokół dloni, wyświetla nazwe nuty oraz aktualnie wybrana oktawę i typ fali na obrazie. Program reaguje na wciskane klawisze, zmieniając typ fali i oktawę w zależności od klawisza. Po wcisnięciu klawisza 'q' zamyka połaczenie z kamerą i zamyka wszystkie okna.