RaspberryPi Jam – Smart mirror

Inteligentne lustro składa się z trzech głównych części – serwer, skrypt do rozpoznawania twarzy oraz stronę, która nam te informacje wyświetli.

Skrypt rozpoznający twarze wysyła informację o nazwie wykrytej twarzy na serwer, tam dane są zapisywane a następnie strona za pomocą javascriptu wysyła zapytanie na serwer, żeby te informacje pozyskać. Wszystko wydaje się łatwe i proste, jednak wymaga to chwili uwagi i odrobiny pracy.

Dostarczone skrypty

Do wykorzystania dostarczone są skrypty, które mają na celu sprawdzenie działania kamery (validate_face_detection.py), zebranie danych (gather_data.py) oraz wytrenowanie modelu (trainer.py).

Elementy do zaimplementowania

server.py

Do implementacji serwera http zostanie użyty moduł Flask (lekki pythonowy framework). Na tym etapie należy zaimplementować dwa endpointy. Serwer posłuży nam do routingu informacji o wykrytej twarzy (nazwa użytkownika) oraz do przechowania tej informacji (nie ma sensu bawić się w bazy danych, żeby przechować jeden string, więc zrobimy to zmienną globalną 😊 – brzydkie rozwiązanie ale najszybsze). Dokumentacja Flaska jest bardzo pomocna.

1. Pierwszy endpoint będzie nam służył do odebrania informacji ze skryptu face_detection i zapisania tej informacji do zmiennej. Metodą POST string ten zostanie wysłany z skryptu i odebrany na odpowiednim porcie serwera (domyślny dla Flaska to 5000).
2. Drugi endpoint będzie służył nam do odczytania wykrytej nazwy z zmiennej i przekazania jej do frontendu. Zapytania GET o nazwę użytkownika będą co jakiś czas wysyłane do serwera i zwracane oraz wyświetlane na stronie.

face_detection.py

Do detekcji twarzy zostanie użyty moduł opencv. Rozpoznanie twarzy odbędzie się na podstawie klasyfikatora. Skrypty do trenowania twarzy i zbierania danych zostały przygotowane wcześniej i są gotowe. Na tym etapie wymagana jest tylko implementacja detekcji twarzy.

1. Na początku należy przechwycić wideo z kamery (cv2.VideoCapture). Następnie w pętli while odczytywać kolejne ramki metodą read(). Opcjonalnie użyć metody set, aby zmienić rozdzielczość.
2. Następnie musimy przekonwertować obraz do skali szarości. Możemy użyć do tego funkcji cvtColor z modułu opencv (cv2).
3. Kolejnym krokiem będzie detekcja twarzy na obrazie. Należy użyć do tego funkcji detectMultiScale, którą możemy wywołać na predefiniowanym klasyfikatorze (cv2.CascadeClassifier) i wykorzystując odpowiedni plik xml (haarcascade_frontalface_default.xml), który zawiera parametry detekcji twarzy. Ta metoda zwróci nam tablicę z rozpoznanymi twarzami. Jako parametr należy podać też minSize czyli minimalne wartości dla których twarz zostanie rozpoznana
4. Jeżeli została rozpoznana jakaś twarz należy spróbować dopasować ją do twarzy, na których został wyuczony nasz recognizer. Twarz reprezentowana jest tutaj jako krotka współrzędnych oraz przesunięć (współrzędne x, y oraz przesunięcia w, h). Należy teraz użyć funkcji predict na wyuczonym wcześniej recognizerze (cv2.face.LBPHFaceRecognizer_create()). Zwróci nam on identyfikator twarzy, z którą został skojarzony podczas nauki oraz confidence.
5. Podczas rozpoznania twarzy należy wysłać nazwę osoby, wykorzystując identyfikator do serwera http (post request). Na przykład wykorzystując moduł requests. W przypadku braku rozpoznania jakiejkolwiek twarzy, wysłać pusty string (na adres localhost i port 5000).

Frontend (index.html)

Na stronie będziemy wyświetlać pobrane informacje o wykrytym użytkowniku. Znajdą się tam też inne elementy jak zegar, data itd.

1. Główną funkcjonalnością do zaimplementowania jest pobieranie danych z serwera. Można do tego wykorzystać moduł XMLHttpRequest. Gdy pobrana zostanie nazwa użytkownika należy go powitać, wpisując tą nazwę do pola detectedName oraz ustawić timeout na kolejne zlecenie pobrania danych (np. funkcją setTimeout).
2. Jeżeli pobrany string z nazwą jest pusty, oznacza to, że nie wykryliśmy żadnej twarzy, więc po prostu po jakimś czasie pobierzmy dane na nowo. Należy tutaj zwrócić uwagę, żeby timeouty przy wykrytej i niewykrytej twarzy były różne żeby uniknąć efektu togglowania, gdy na chwilę nasza twarz nie zostanie wykryta, ale pozostanie cały czas w zasięgu wzroku kamery.
3. Drugą funkcjonalnością jest implementacja zegara i wyświetlania daty. Należy je wpisać odpowiednio do pól clock i date. Można użyć funkcji JS setInterval, żeby wywołać pobranie godziny i daty co określony interwał czasowy.

Ulepszenia

Jeżeli dotarłeś tutaj – gratulacje! Wykonałeś podstawowe zadania, które zostały przewidziane dla uczestników. To jednak nie koniec. Chcesz aby twoje lustro zapierało dech w piersiach? Dla ambitnych przygotowaliśmy dodatkowe zadania/ulepszenia. Nie są one ze sobą powiązane, więc możesz wybrać dowolne z poniższej listy i spróbować zaaplikować do twojego lustra.

- dodanie informacji o warunkach pogodowych (w sieci dostępne są strony z publicznym API, które pozwalają na pobranie informacji pogodowych w danym regionie, spróbuj pobrać takie dane i umieścić je na lustrze)
- dodanie informacji o komunikacji miejskiej z pobliskiego przystanku
- synchronizacja z informacjami z twojego kalendarza
- zmiana koloru i kroju czcionek oraz lokacji obiektów na stronie
- metoda predict zwraca jeszcze jedną wartość, jaką jest confidence, można wykorzystać ją do uwzględnienia pewności wykrycia danej twarzy
- zastanowić się i zaimplementować obsługę wykrywania wielu twarzy jednocześnie i wyświetlania spersonalizowanych informacji
- … sky is the limit 😉