Dokumentacja

Projekt - Podstawy teleinformatyki

Malowanie obrazów biorąc pod uwagę ruch ciała

Konrad Marciniak
Mateusz Durlak

Spis treści:

1. Opis tematu

Tematem projektu jest stworzenie aplikacji, która umożliwia malowanie obrazów za pomocą kamerki internetowej oraz ruchu ciała. Zespół zdecydował się na ten temat z uwagi na poznaną na wcześniejszych zajęciach technologię OpenCV, która miałaby być główną pomocą w rozwiązaniu problemu.

2. Podział prac

Konrad Marciniak	Mateusz Durlak
Decyzja o zastosowanym języku programowania	Wyszukanie dostępnych technologii
Opracowanie sposobu działania aplikacji	
Detekcja kolorów na kamerze	Wyszukanie odpowiednich filtrów do detekcji
Dopracowanie kodu oraz dodatkowych funkcjonalności (np. zapis obrazu)	Zaimplementowanie malowania
Nagranie filmu z zaprezentowanym działaniem aplikacji	Instrukcja
Opracowanie dokumentacji	

3. Funkcjonalności aplikacji

3.1. Rysowanie

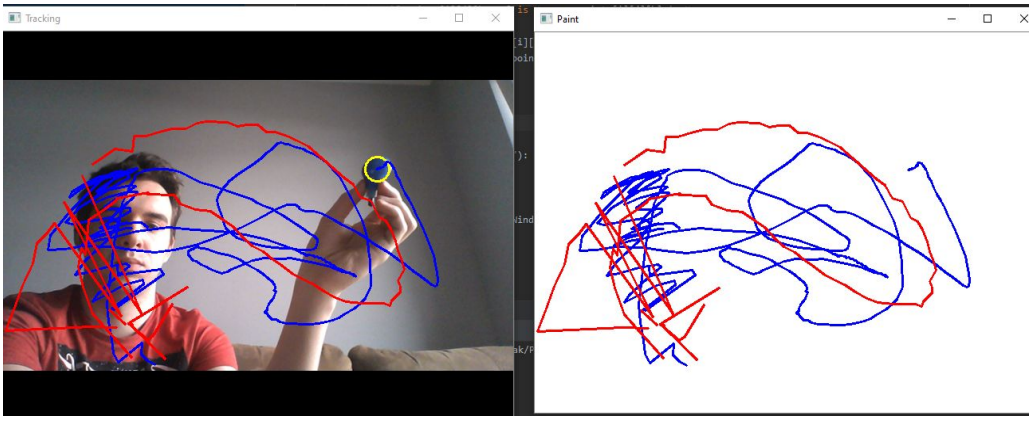
Malowanie odbywa się za pomocą przedmiotów o odpowiednim kolorze (zielonym, czerwonym oraz niebieskim). Za pomocą ruchu przedmiotami program generuje linie na oknie kamery oraz w oknie malowanego obrazu.

3.2. Zapis obrazu

Po namalowaniu obrazu za pomocą przycisku s (save) program zapisuje namalowany obraz do katalogu aplikacji.

3.3. Wyczyszczenie obrazu

Za pomocą przycisku ‘c’ (clear) możliwe jest wyczyszczenie namalowanego obrazu. Zdecydowano się na dodanie takiej funkcjonalności z uwagi na to, że użytkownik może niepostrzeżenie namalować niechciane kontury.



Rys. 1 Działanie aplikacji

4. Wybrane technologie

4.1. OpenCV

OpenCV (Open Source Computer Vision library) jest bezpłatną,open-source'ową biblioteką do użytku komercyjnego i edukacyjnego, bez konieczności udostępniania swoich projektów opartych o jej implementację. Została napisana w języku C i C++ i rozwijana jest za pomocą wrapperów w wielu językach programowania takich jak C#, Python, Ruby, Matlab czy Java. Posiada wielu użytkowników na całym świecie –doskonałym przykładem jest grupa tematyczna o OpenCV na portalu Yahoo, która aktualnie posiada ponad 48.000 użytkowników.Biblioteka ta została stworzona na potrzeby aplikacji czasu rzeczywistego gdzie wydajność obliczeniowa programów jest bardzo istotna. Napisana jest w zoptymalizowanym języku C/C++ i wykorzystuje możliwości jakie dają popularne od kilku lat procesory wielordzeniowe.Jednym z głównych celów biblioteki OpenCV jest dostarczenie narzędzia, które pozwoli tworzyć zarówno proste programy jak również zaawansowane projekty ludziom z różnym poziomem wiedzy na temat programowania i przetwarzania obrazów.Amatorzy za pomocą OpenCV i kamery internetowej mogą poznawać elementy przetwarzania obrazów w zakresie: wykrywania krawędzi, segmentacji czy filtracji obrazów bez większego wysiłku. Zaś Sami specjaliści nie muszą poświadczana pisanie bazowych funkcji w zaawansowanych 10 projektach –nigdy więcej wymyślania koła! Możliwe jest to dzięki wbudowanym w bibliotekę ponad 500 funkcjom, które obejmują wiele obszarów komputerowej wizji takich jak robotyka, stereowizja, bezpieczeństwo, analiza obrazów medycznych czy kalibracja kamery. Wiele projektów wykorzystujących zdjęcia z lotu ptaka czy mapowanie ulic takie jak –„Google Maps", czy „Google StreetView",wykorzystuje kalibrację kamery oraz metody „zszywania" obrazów, które są częścią OpenCV. Bezzałogowe samoloty, urządzenia do detekcji obiektów czy układy monitoringu również oparte są o wizje komputerową. Inspiracją do zgłębienia tematyki OpenCV był artykuł„Innovations from a robot rally" w SCIENTIFIC AMERICAN o wyścigu „Grand Challenge" przeznaczony dla autonomicznych pojazdów.Wyścig odbył się w październiku roku 2005,a organizatorem była amerykańska agencja DARPA,zajmująca się rozwojem

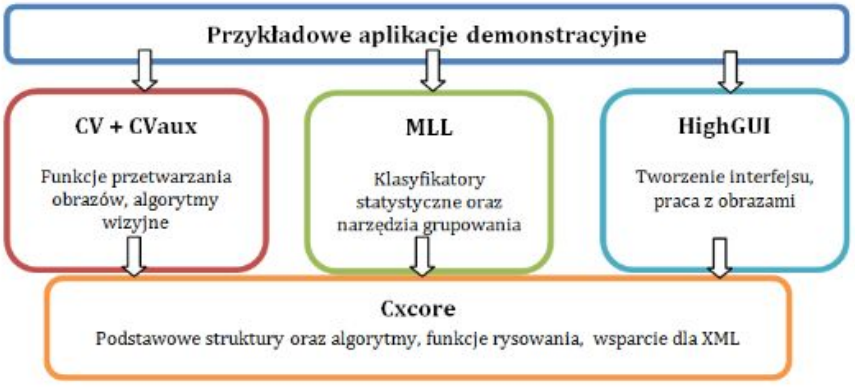
technologii wojskowej.Samochód „Stanley” –Volkswagen Touareg przejechał samodzielnie trasę 212 km po pustyni Mojave oraz południu Las Vegas w czasie 6 godzin i 54 minut. Jedną z głównych ról przy projektowaniu systemu wizyjnego odegrała właśnie biblioteka OpenCV.

Początki OpenCV sięgają końca lat 90’. W tamtych latach korporacja Intel rozpoczęła kilka projektów opartych o pracę w czasie rzeczywistym –takich jak śledzenie promieni oraz wyświetlanie obiektów 3D na ścianie. Jeden z autorów tych projektów wizytując w tym czasie amerykańskie uniwersytety zauważył, że kilka grup uniwersyteckich takich jak„MIT Media Lab” z MIT w Massachusetts Posiada wiele rozbudowanych narzędzi do przetwarzania obrazów, które były propagowane i jednocześnie udoskonalane przez studentów. Właśnie przy współpracy tych grup oraz pomocy Intel’s Performance Library Team zostało stworzone jądro z zaimplementowanym kodem oraz specyfikacją algorytmów. Dane te zostały przesłane do Intel’s Russian Library Team w Niższym Nowogrodzie w Rosji –gdzie właśnie powstała biblioteka OpenCV. Głównymi celami OpenCV jakie zostały założone w trakcie jej budowy było stworzenie darmowego, otwartego i zoptymalizowanego kodu dla podstawowych funkcji przetwarzania obrazów. Kod ten musiał być czytelny i łatwy do przenoszenia, gdyż był podstawą dla deweloperów, którzy mieli go rozwijać. Możnaby Się zastanawiać dlaczego Intel stworzył bardzo pomocą bibliotekę dla deweloperów całkowicie za darmo. Idea była bardzo prosta –wykorzystanie tej biblioteki pomagało w projektowaniu aplikacji wymagających dużej mocy obliczeniowej, a tym samym szybkich procesorów. Sprzedaż komponentów firmy Intel była bardziej opłacalna niż wypuszczenie na rynek dodatkowego oprogramowania. Pomysł mógł być zrealizowany tylko przy dużej popularności tej biblioteki. Intel nie pomylił się z prognozami –do tej pory ze strony projektu[W2]biblioteka OpenCV została ściągnięta 2.858.950 razy, co stanowi 40.5 TB danych.

Pierwsza wersja„alpha” biblioteki została udostępniona w roku 2000. Następne, udoskonalone wersje to: „Beta 1” –2001 (wsparcie dla Linuxa), „Beta 2” –2002, „Beta 3” –2003, „Beta 4” –2004, „Beta 5” –2005. Oficjalna wersja OpenCV o nazwie „1.0” powstała dopiero w październiku 2006 roku. Spora aktywność deweloperów w kierunku rozwijania projektu spowodowała powstawanie kolejnych

wersji: „1.1pre1” –2008, „2.0” –2009. Od kwietnia 2010 roku dostępna jest wersja„2.1”. Z informacji zawartych na stronie projektu wynika, że wersja 2.2 ma pojawić się w październiku 2010 roku.Biblioteka OpenCV przeznaczona jest do pracy z systemami Linux, Mac OSX oraz Windows pod kompilatorami Visual Studio 2005, 2008 (wraz z wersjami Express), MinGW –Windows oraz GCC 4.x –Linux i MacOS.

OpenCV składa się z pięciu podstawowych komponentów: CV oraz Vaux czyli komponentów zawierających funkcję transformacji, filtracji oraz konwersji przestrzeni obrazów, funkcję analizy obrazów takie jak selekcja, operacje morfologiczne, detekcję krawędzi oraz obsługę histogramów, detekcję obiektów, kalibrację kamery, rekonstrukcję sceny 3D i inne,ML –Machine Learning Library, jak sama nazwa wskazuje zawiera funkcje tworzenia klasyfikatorów bazujących na statystyce odgrywających znaczącą rolę w uczeniu maszyn sposobu detekcji,HighGUI –zawiera metody akwizycji i zwalniania obrazów, sekwencji wideo, narzędzia tworzenia interfejsu okienkowego, suwaków, obsługi myszy etc.xCore –Podstawowy komponent biblioteki zawiera operacje na tablicach, algebrę macierzy, funkcje matematyczne, transformacje Fouriera, wsparcie dla plików XML, narzędzia rysowania obiektów 2D i inne.



Rys. 2 Architektura OpenCV

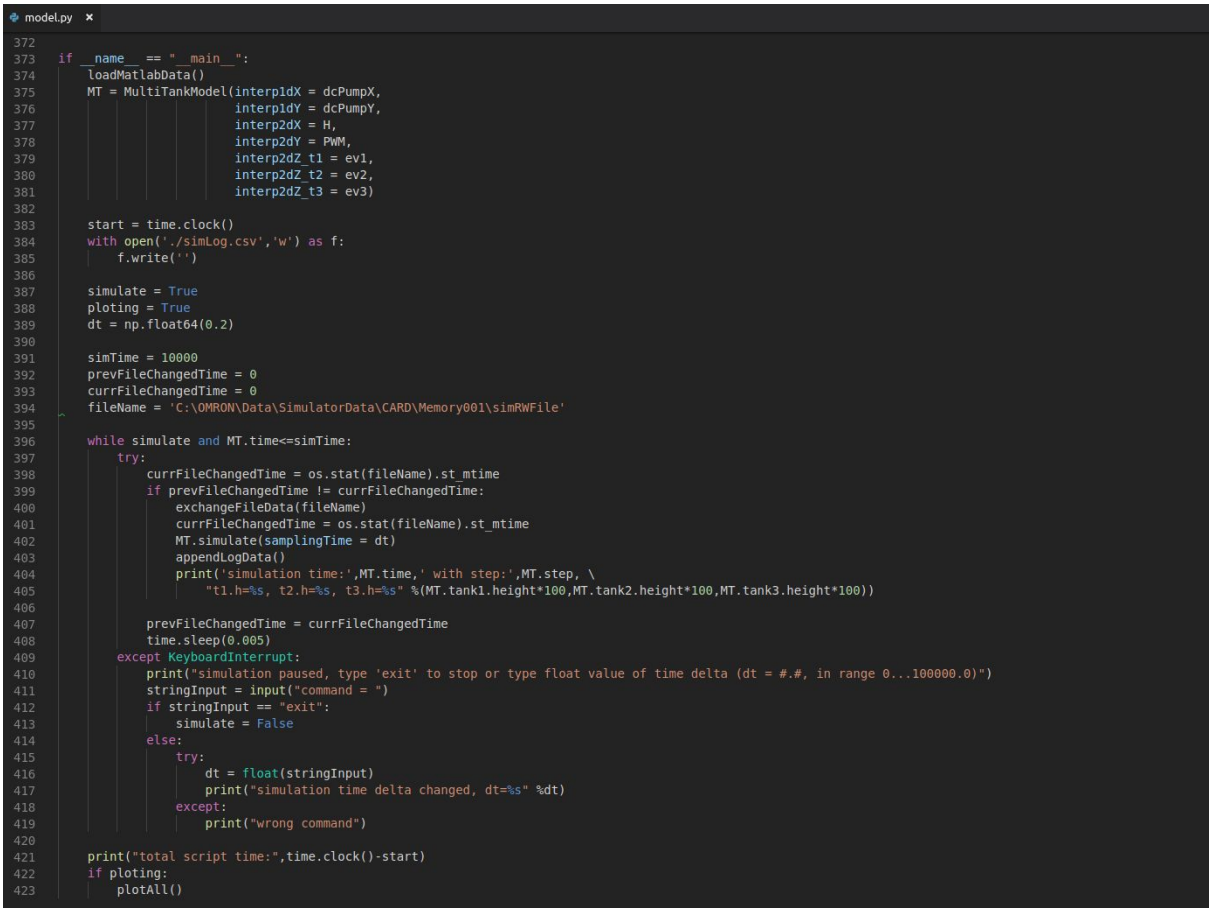
4.2. Python

Python to obiektowy język programistyczny, który wyróżnia się dynamiką, łatwością w nauce i wielością możliwości wykorzystania. Ideą przewodnią, która przyświecała jego twórcą, była chęć zapewnienia czytelności kodu źródłowego. Właśnie dlatego składnia języka programowania Python wyróżnia się klarownością i zwięzłością. Jego interpretery są dostępne na wiele systemów operacyjnych. Python rozwijany jest na otwartej licencji przez Python Software Foundation, która jest organizacją non-profit. Standardowa implementacja języka to Python (napisany w języku programowania C).

Mimo obco brzmiącej nazwy, Python jest powszechnie wykorzystywany w serwisach, aplikacjach czy programach, z których korzystamy na co dzień. Dla przykładu popularny YouTube został w większości napisany w Pythonie. Wybór języka został dokonany ze względu na potrzebę zapewnienia dużej wydajności, a także łatwej implementacji najnowszych funkcjonalności. Z Pythona korzysta również w swoich aplikacjach Google. Mowa tu o takich aplikacjach jak chociażby Google App Engine czy Google Wave. Warto również wspomnieć o wykorzystaniu Pythona przez NASA. Aplikacje napisane w tym języku stosowane są chociażby do zarządzania kontrolą startową wahadłowców. Python jest też wykorzystywany przez Nokię, a także w połączeniu z innymi językami i technologiami.
Python posiada bardzo szerokie możliwości wykorzystania. Z jego pomocą można tworzyć zarówno serwisy internetowe, jak i aplikacje komputerowe czy sieciowe, a nawet gry. Python pozwala na tworzenie dynamicznych stron internetowych w prosty sposób. Język jest bardzo efektywny. Duża w tym zasługa frameworków takich jak jak Django czy Pylons. Właśnie dlatego korzysta się z niego głównie przy tworzeniu stron, które są bogate w różnego rodzaju funkcje.

Za pomocą Pythona można chociażby wykorzystać API (interfejs programowania aplikacji/usługi) serwisu takiego jak Facebook. Dla przykładu wykorzystanie biblioteki Facebook pozwala na nieskomplikowane tworzenie aplikacji dla tego portalu. Skrypty i aplikacje napisane w Pythonie znajdują również

zastosowanie w nauce czy finansach, gdy potrzebne jest generowanie wykresów lub przetwarzanie danych. Język sprawdzi się również świetnie w przypadku gier i aplikacji wykorzystujących technologię 3D. Często udostępnia się w Pythonie Interfejs Programowania Aplikacji silnika gry napisanego w C lub C++. Jako przykład udanego efektu takich prób można podać chociażby Świątynię Pierwotnego Zła. Wyżej została już wspomniana platforma Google App Engine, którą stworzono dla rozproszonego hostingu aplikacji. Ją również oparto na Pythonie. Dzięki temu Google App Engine może zaoferować serwisom www taką skalowalność, jaką posiadają wszystkie inne usługi i aplikacje tej firmy.



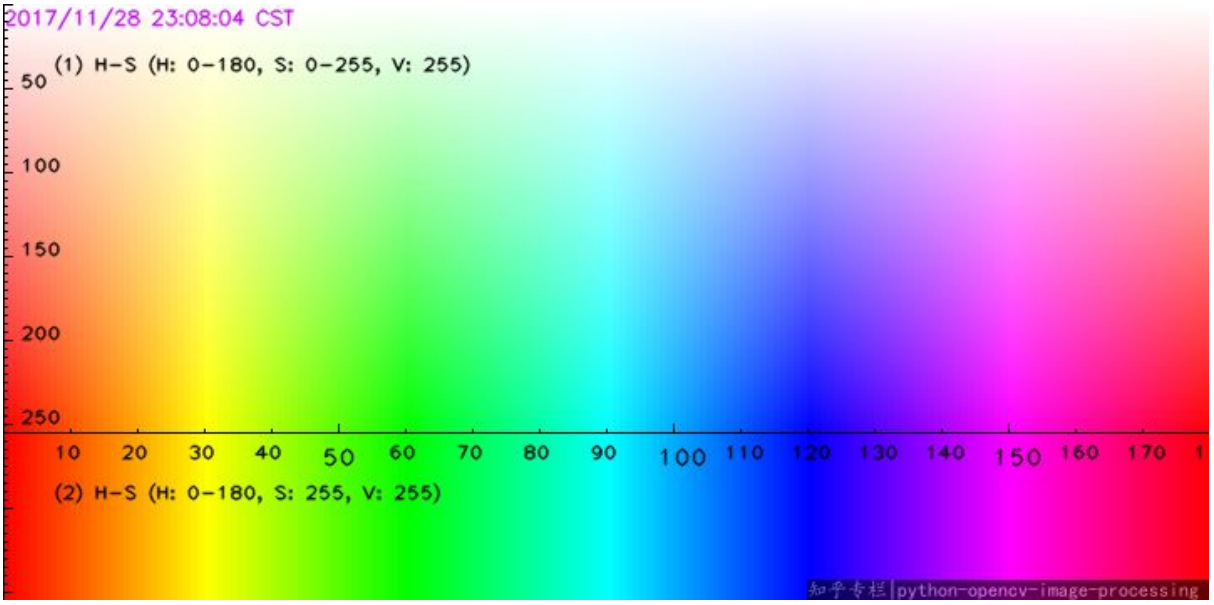
Rys. 3 Przykładowy kod w języku *python*

5. Architektura rozwiązania

Rozwiązanie udało się uzyskać przy pomocy jednego skryptu korzystającego z odpowiednich biblitoek zawartych w pliki *resources*. Aplikacja otwiera dwa okna:

- Tracking - obraz z kamery oraz widoczne rysowane linie,
- Paint - malowany obraz.

Za pomocą filtrów HSV oraz warunków brzegowych dla odpowiedniego koloru (przykładowo dla niebieskiego są to wartości [100,60,60] oraz [140,255,255] program próbuje dokonać detekcji danego koloru na obrazie z kamery. Po wyznaczeniu kolory program rysuje okrąg wokół konturu po czym od jego środka malowana jest linia w danym kolorze.



Rys. 4 Skala HSV

Najważniejsze fragmenty kodu:

- wyszukanie konturów - filtracja przez warunki brzegowe HSV, erozja (zwężenie), zastosowanie gradientu morfologicznego, dylatacja,

```
blueMask = cv2.inRange(hsv, blueLower, blueUpper)
blueMask = cv2.erode(blueMask, kernel, iterations=2)
blueMask = cv2.morphologyEx(blueMask, cv2.MORPH_OPEN, kernel)
blueMask = cv2.dilate(blueMask, kernel, iterations=1)

(cnts, _) = cv2.findContours(blueMask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
```

- rysowanie koła wokół konturu,

```
if len(cnts) > 0:
    cnt = sorted(cnts, key=cv2.contourArea, reverse=True)[0]
    ((x, y), radius) = cv2.minEnclosingCircle(cnt)
    # koło w okół kontury
    cv2.circle(frame, (int(x), int(y)), int(radius), (0, 255, 255), 2)
    M = cv2.moments(cnt)
    center = (int(M['m10'] / M['m00']), int(M['m01'] / M['m00']))
```

- zmiana koloru zależnie od indeksu.

```
if colorIndex == 0:
    bpoints[bindex].appendleft(center)
elif colorIndex == 1:
    gpoints[gindex].appendleft(center)
elif colorIndex == 2:
    rpoints[rindex].appendleft(center)
```

6. Problemy i rozwiązania

6.1. Szukanie konturów

Podstawowym i właściwie jedynym problemem było wyszukanie odpowiednich konturów dla wybranych kolorów. Za pomocą odpowiedniego *researchu* udało się jednak za pomocą odpowiednich funkcji biblioteki OpenCV znaleźć interesujące kontury. Przy pomocy indeksowania kolorów HSV należało ustawić odpowiednie warunki brzegowe dla wybranych kolorów, a następnie przeprowadzić kilka operacji na obrazie, a mianowicie:

- erozja - operacja ta przykłada obrócony element strukturalny do każdego piksela na obrazie. Jeżeli choć jeden piksel z sąsiedztwa ma wartość równą zero, punkt centralny również otrzymuje wartość zero. W przeciwnym wypadku jego wartość nie ulega zmianie,
- gradient morfologiczny - różnica pomiędzy dylatacją, a erozją danego obrazu,
- dylatacja - operacja ta przykłada obrócony element strukturalny do każdego piksela na obrazie. Jeżeli choć jeden piksel z sąsiedztwa ma wartość równą jeden, punkt centralny również otrzymuje wartość jeden. W przeciwnym wypadku przypisywane jest mu zero.

7. Instrukcja

Program ukazuje dwa okna (Drawing oraz Tracking). Okno Drawing ukazuje narysowany obraz natomiast Tracking ukazuje detekcję elementów w kamerze. Za pomocą klawiatury można dokonać następujących czynności:

- zamknąć program (q),
- wyczyścić obraz (c),
- zapisać obraz (s).

Program rozpoznaje trzy podstawowe kolory (niebieski, zielony oraz czerwony). Za pomocą kredek/pisaków oraz ruchów dłonią przy włączonej kamerze można stworzyć rysunek, który widoczny jest w oknie Drawing.

8. Literatura

https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html
https://pl.wikipedia.org/wiki/Cyfrowe_przetwarzanie_obraz%C3%B3w_binarnych#Znajdowanie_konturu
http://aragorn.pb.bialystok.pl/~boldak/DIP/CPO-W05-v01-50pr.pdf
http://analizaobrazu.x25.pl/articles/19
http://pforczmanski.zut.edu.pl/homepage/wp-content/uploads/w07-cechy_ksztaltu.pdf