POLITECHNIKA WROCŁAWSKA
WYDZIAŁ INFORMATYKI I ZARZĄDZANIA

Wspomagana rzeczywistość (ang. augmented reality) wyświetlanie obiektu trójwymiarowego w czasie rzeczywistym w miejscu występowania określonego znacznika.

Analiza obrazów i wideo

Maciej Urbaniak (200842)

WROCŁAW 2019

Spis treści

1. Opis problemu

Celem projektu było wyświetlenie trójwymiarowego obiektu w miejscu występowania określonego znacznika, co zostało potem rozszerzone o dowolny znacznik zdefiniowany przez użytkownika.

Rzeczywistość rozszerzona jest to połączenie dwóch światów jakimi są ten rzeczywisty oraz wygenerowany komputerowo. Obecnie wykorzystuje się dodatkowe możliwości jakie dostarczają urządzenia mobilne(ang. smartphone) i poza oczywistym wykorzystaniem kamery stosuje się również położenie(GPS) oraz kierunek w jakim jest ułożone urządzenie(kompas) razem z możliwością wykrycia trajektorii ruchu urządzenia(akcelerometr i żyroskop). Przykładem wykorzystania takich rozwiązań jest bardzo popularna gra Pokemon Go(1). Jeśli natomiast miało to by być coś bliższego realizowanemu w tym semestrze projektowi, wymieniłbym rozwiązanie firmy Popcode(2) oraz Nyatla(4), do których porównam się w rozdziale 4.3 poświęconym wynikom działania programu.

1.1. Przyjęte założenia i ograniczenia

Zakładam że wybrany znacznik posiadać będzie wystarczająco wiele cech oraz nie będzie przechylony pod zbyt wysokim kątem.
Przez cechy należy rozumieć ułożenie się pikseli w określone kształty: linie i ich zakończenia, krawędzie, rogi, siatki, koła lub okręgi.
Dodatkowe ograniczenia narzucone zostały przez wybrane rozwiązania jak język programowania, system operacyjny, możliwości obliczeniowe procesora oraz zastosowanej kamery(szczegóły w rozdziale 3. Dane).

2. Opis przyjętej metody

Przyjęta metoda polega na algorytmach które znajdują punkty szczególne na obrazie z kamery oraz wybranego znacznika a następnie dopasowują je parami, aby na ich podstawie wyliczyć wektor przemieszczenia. Tak więc jakość i liczba odnalezionych punktów będzie istotnie zależała od wybranego znacznika oraz użytego sprzętu w postaci kamery.

Użyte algorytmy:

— ORB (Oriented FAST and Rotated BRIEF)(7)
— SIFT (Scale-Invariant Feature Transform)(5)

SIFT jest algorytmem opatentowanym `https://patents.google.com/patent/US6711293`, ale w Europie podobnie jak w Polsce nie dopuszcza się patentowania algorytmów, więc wynikło wiele dyskusji nad komercyjnym wykorzystaniem tego rozwiązania (więcej na `https://patents.stackexchange.com/questions/3388/is-sift-algorithm-patent-valid-in-europe`). Wyszukuje on cechy stabilne, czyli niewrażliwe na zmianę skali i obroty oraz przypisuje im orientację. Więcej informacji w publikacji ”Object recognition from local scale-invariant features.”(5).

Następnym krokiem było dopasowanie znalezionych punktów z obrazu znacznika do strumienia obrazów z kamery. Pojedyncze łączenie punktów generuje wiele nieprawidłowych odwzorowań, dlatego wykorzystano odległość euklidesową między grupą trzech znalezionych. Wartość progowa zaproponowana w pracy ”Augumenting reality, naturally”(6) wynosiła 0.8 ale zmieniona została w projekcie na 0.7 po empirycznym sprawdzeniu działania na własnych przykładach.

Na wybranych punktach szukana jest homografia pomiędzy dwoma obrazami. Polega ona na znalezieniu odpowiedniej transformacji pomiędzy dwoma zadanymi płaszczyznami z uwzględnieniem skali. Do jej wyszukania użyto algorytmów:

— brutalnego (każdy punkt z każdym)
— LMeDS
— RANSAC

Wyznaczona homografia i parametry kamery posłużyły do umieszczenia obiektu 3D na scenie poprzez wyliczenie macierzy projekcji(projection matrix).

2.1. Wykorzystane komponenty obce i ich użyta funkcjonalność

Python 3 - język programowania w wersji (Python 3.6.6 | packaged by conda-forge | (default, Jul 26 2018, 09:53:17) [GCC 4.8.2 20140120 (Red Hat 4.8.2-15)] on linux)

OpenCV - biblioteka open source `https://opencv.org` (wersja '3.4.2') skorzystano z wielu wbudowanych algorytmów takich jak RANSAC czy ORB dodatkowo używana jest do wyświetlania procesu działania programu.

Blender - program do grafiki 3D `https://www.blender.org` (wersja 2.79b) użyto do stworzenia wyświetlanego obiektu(kubek).

Math - biblioteka używana do operacji matematycznych `https://docs.python.org/3/library/math.html`

NumPy - biblioteka do m.in. obliczeń na macierzach wielowymiarowych oraz algebry liniowej `http://www.numpy.org`

2.2. Architektura zaimplementowanego programu

Program zaimplementowano w języku Python w wersji 3.6.6

2.2.1. load_obj.py

Zawiera klasę dzięki której wczytywany jest obiekt 3D z pliku o rozszerzeniu .obj a dokładniej jego wierzchołki i ściany.

2.2.2. frame_obj.py

Odnosi się do wyznaczonego przez użytkownika znacznika. Zawiera rozmazanie Gaussa w celu redukcji szumów oraz wyznaczenie punktów za pomocą SIFT i ORB.

2.2.3. main.py

Służy do obsługi głównego okna dodatkowo znajdują się w nim funkcje potrzebne do wyrysowania obiektu.

2.2.4. kubek.obj

Zawiera kubek stworzony w programie blender o rozszerzeniu .obj.

2.3. Opis ”wkładu własnego”

— Stworzenie kubka w programie Blender
— Porównanie i wybór algorytmów
— Wygładzanie ruchu obiektu
— Eksperymenty przeprowadzone z użyciem różnych algorytmów
— Przedstawienie łączenia punktów kluczowych
— Redukcja nagłych ”skoków” obiektu
— Stworzenie ostatecznego programu
— Eksperymenty i wyniki ostatecznego działania programu

3. Dane

Dane w postaci video zostały sporządzone własnoręcznie za pomocą dostępnej kamery firmy ”Quanta Computer, Inc.” w maksymalnie dostępnej rozdzielczości 640x480 pikseli.
System operacyjny: Linux Mint 19
Procesor: Intel® Core™ i3-7100U

Wybrane zapisy video znajdują się na dołączonej płycie w folderze videos.

Plik .blend oraz wyeksportowany plik .obj stworzonego kubka znajduje się w folderze packages.

Ze względu na charakter projektu czyli wyświetlanie znacznika w czasie rzeczywistym zebrane dane są bardzo tendencyjne. Powtórzenie eksperymentów będzie z pewnością generować różne wyniki w zależności od zastosowanej kamery, warunków oświetleniowych, wybranego znacznika czy choćby losowych szumów i wpływu samego eksperymentatora, a przez to wykonywania innych ruchów w przestrzeni. Dodatkowo przy dużym obciążeniu systemu zauważono, że nie zawsze zostaje obliczona pozycja obiektu. Objawia się to szybkim pojawianiem się i znikaniem wyświetlanego kubka, dlatego też wszelkie badania przeprowadzone zostały przy ograniczeniu procesów działających w tle.

4. Wyniki eksperymentalne

4.1. Cel i zakres badań

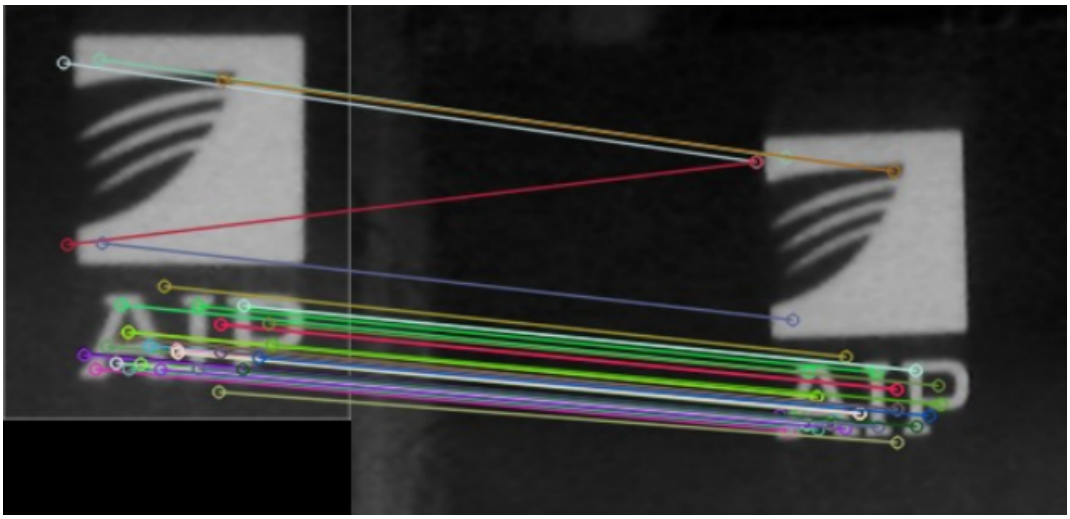
Celem było porównanie działania programu dla dostępnych na rynku rozwiązań a w szczególności do rozwiązania firmy Popcode(2) oraz Nyatla(4).

4.2. Miary

Miary którymi można zmierzyć działanie programu można podzielić na dwie kategorie: użyteczność dla użytkownika(jakość użytkowa) oraz poprawność wyników(liczba znalezionych punktów, dokładność obliczeń). Dla przeprowadzonych eksperymentów wybrano subiektywne porównanie z Popcode(2) oraz Nyatla(4) jako iż będzie miało ono znacznie większe znaczenie dla rozrywkowego zastosowania programu jakim jest wyświetlanie obiektu 3D. W przemyśle również stosuje się AR jak w Augmented Reality Software System for Quality Inspection - CAQ AG `https://www.youtube.com/watch?v=tQepBwJppYQ`, ale wyświetlanie obiektów wciąż jest przybliżone.

4.3. Wyniki

Nagrane eksperymenty znajdują się w folderze videos. Znacznik wykorzystany do eksperymentów znajduje się na rysunku poniżej. Jest to wizytówka z logiem ”Akademickiego Inkubatora Przedsiębiorczości” o wymiarach 9cm na 5cm (wymiary loga 2cm na 3cm). Wybrano algorytmy dające najlepsze wyniki: SIFT oraz RANSAC. Badania przeprowadzono w tym samym pomieszczeniu i o stałych warunkach oświetleniowych.



Rysunek 4.1: Testowany znacznik(wizytówka)

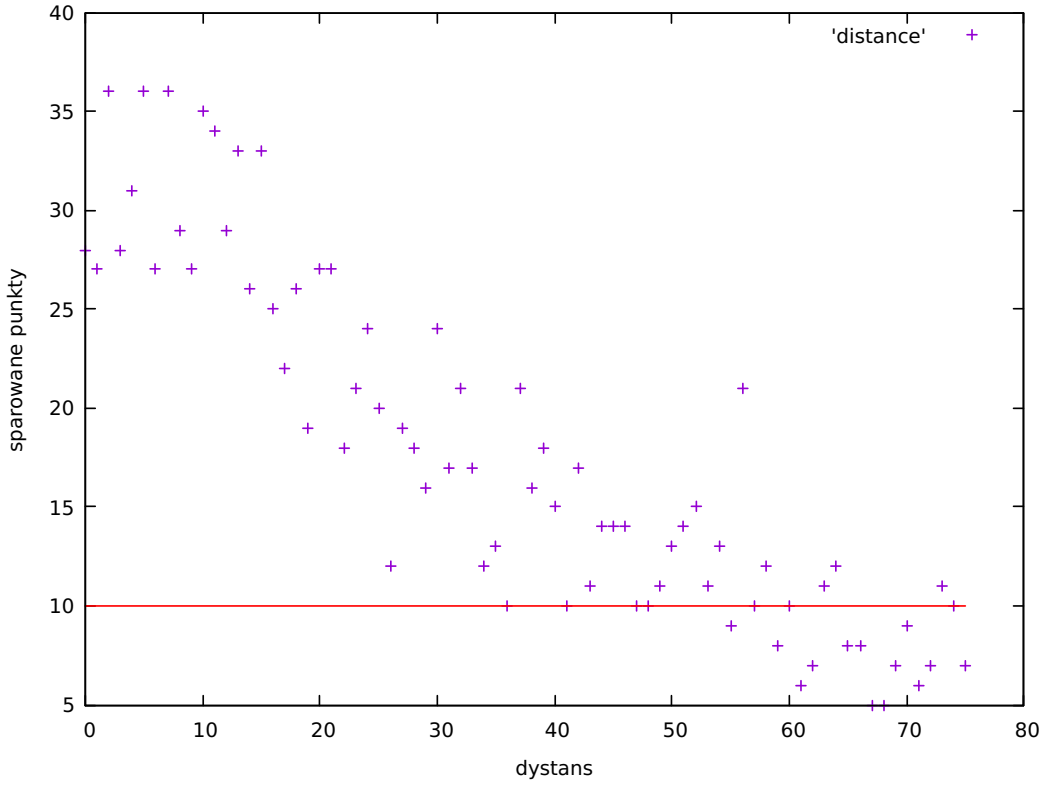
4.3.1. video_1

Przy wolnym przesuwaniu obiekt jest wykrywany cały czas Przy szybkim następuje zgubienie obiektu po czym przeniesienie go na właściwe pole. Dodatkowo w porównaniu z Popcode(2) i Nyatla(4) obiekt zamiast wykazywać się gwałtownymi przesunięciami jest bardziej płynny (wynik uśredniania pozycji obiektu biorąc pod uwagę wcześniejsze klatki)

4.3.2. video_2

Przy oddalaniu obiektu bardzo szybko tracone są wykrywane punkty przez co pozycja w przestrzeni jest nieprawidłowo obliczana. Spowodowane jest to szumem, niską rozdzielczością kamery oraz małymi rozmiarami wybranego znacznika. W rozwiązaniach wcześniej wspomnianych firm nie przedstawiono wyświetlania dla oddalonych znaczników.

Wykres poniżej przedstawia eksperyment w którym znacznik został przystawiony w odległości około 15 centymetrów a następnie oddalony na odległość około 1 metra.



Rysunek 4.2: Wykres dystansu od znajdywanych par punktów. Czerwona linia wyznacza minimalną liczbę poniżej której znacznik nie będzie wykrywany.

4.3.3. picture_1

Przeprowadzono test polegający na wybraniu większego znacznika dzięki czemu zwiększono dystans wyświetlania obiektu z około 40cm od kamery do około 120cm. Warto również zwrócić uwagę na szumy, które są bardzo widoczne na wspomnianym obrazie i utrudniają, w połączeniu z niską rozdzielczością, odnalezienie mniejszych znaczników.

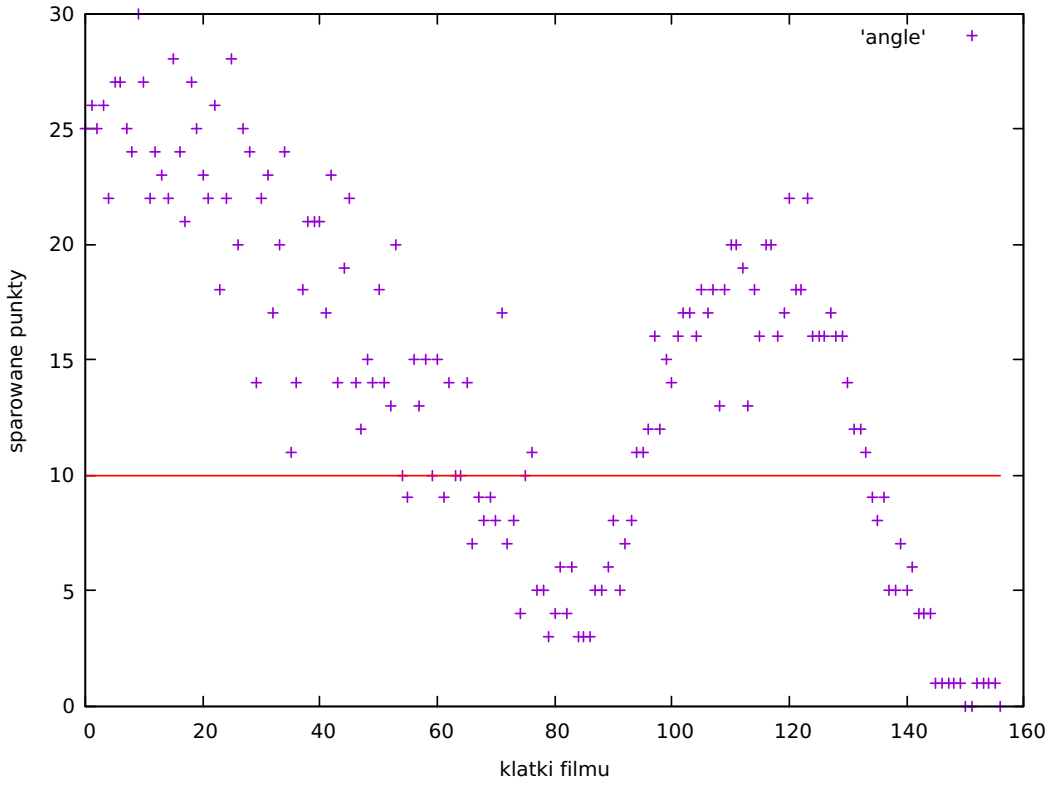
4.3.4. video_3

W porównaniu do ustalonych znaczników Aruco, częściowe przysłonięcie znacznika do poziomu około 30% nie zaburza wyświetlania obiektu. Natomiast przy 50% pozycja obiektu jest nieprawidłowa a zasłonięcie dalszej części powoduje niedostateczną liczbę punktów sparowanych aby określić, że znacznik występuje na obrazie. Za minimalną liczbę punktów odpowiedzialny jest parametr MIN_MATCH_COUNT który można zmienić, ale wymagane są co najmniej 4 prawidłowo wyznaczone punkty w celu wyliczenia homografii. Dodatkowo zmniejszenie tego parametru może powodować wykrywanie nieprawidłowych znaczników w scenach o dużej ilości szczegółów poprzez przypadkowe podobieństwa.

4.3.5. video_4

Przy obrocie około 45 stopni następują problemy z prawidłowym wyznaczeniem pozycji obiektu 3D, a przy większej z niemożnością odnalezienia znacznika. Spowodowane jest to zmniejszeniem rozdzielczości oraz rozmiaru wykrywanej powierzchni i przez to większym wpływem szumu. Na filmach Popcode(2) i Nyatla(4) znacznik cały czas jest dobrze widoczny oraz dużo większy.

Wykres prezentuje eksperyment w którym znacznik początkowo jest w pozycji równoległej do kamery. Następnie obracany jest do około 45 stopni a potem znowu wraca do pozycji początkowej. Spadek przy końcu wykresu spowodowany jest zdjęciem znacznika z pola kamery przed zakończeniem pomiarów.



Rysunek 4.3: Test zmiany nachylenia znacznika do kamery. Czerwona linia wyznacza minimalną liczbę sparowanych punktów poniżej której znacznik nie będzie wykrywany.

Jak widać na wykresie powyżej znacznik jest częściowo rozpoznawany przy dużym nachyleniu, ale dalsze wyświetlanie obiektu 3D przy spadku liczby parowanych punktów poniżej 10 powoduje że jest on bardzo niestabilny.

4.3.6. video_5

Badanie wpływu zmiany oświetlenia wykazało że przy gwałtownej zmianie oświetlenia kamera potrzebuje 2-4 klatek na dostrojenie. Wyniki tego badania będą zależeć w głównej mierze od użytej kamery.

4.4. Podsumowanie

Mimo zastosowania większych znaczników w rozwiązaniach firm Popcode(2) i Nyatla(4) wciąż występują okresowe zaniki oraz znaczne drgania obiektu 3D (przykładowe filmy prezentujące działanie znajdują się w folderze videos). Uważam przez to że obecne rozwiązanie po okresie dodatkowego dopracowania może próbować z nimi konkurować, jednak będzie ono wciąż dalekie do osiągnięć większych korporacji jak np. ARcore firmy Google `https://www.youtube.com/watch?v=6nxmHtdHrxw`

Główną zaletą jak i wadą opracowanego rozwiązania jest możliwość wybrania własnego znacznika. Sprawia to że użytkownik może wybrać dowolny płaski obiekt i nie jest zmuszony do druku różnych znaków oraz przyklejania ich na wybrane powierzchnie. Ale oznacza to też iż może on wybrać trudno rozpoznawalny wzorzec przez co znacznie obniża się skuteczność prawidłowego wyświetlenia obiektu 3D.

Bibliografia

[1] Gra Pokemon Go https://play.google.com/store/apps/details?id=com.nianticlabs.pokemongo&hl=pl

[2] Strona główna firmy Popcode http://popcode.info

[3] Prezentacja działania NyARToolkit5 https://www.youtube.com/watch?time_continue=35&v=h_p89B1i6u0

[4] Strona internetowa firmy Nyatla http://nyatla.jp/nyartoolkit/wp/

[5] Lowe, David G. "Object recognition from local scale-invariant features." Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, 1999.

[6] Gordon, Iryna. Augmenting reality, naturally. Diss. University of British Columbia, 2004.

[7] Rublee, Ethan, et al. "ORB: An efficient alternative to SIFT or SURF." Computer Vision (ICCV), 2011 IEEE international conference on. IEEE, 2011.

[8] Dokumentacja biblioteki OpenCV https://docs.opencv.org