Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Informatyki, Elektroniki i Telekomunikacji

KATEDRA INFORMATYKI

PRACA MAGISTERSKA

KSAWERY GŁAZ

ANALIZA WPŁYWU SKŁADOWYCH SYSTEMU ROZSZERZONEJ RZECZYWISTOŚCI NA JAKOŚĆ UZYSKIWANEGO OBRAZU.

PROMOTOR:
prof. dr hab. inż. Krzysztof Boryczko

Kraków 2016

OŚWIADCZENIE AUTORA PRACY

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

. . . . . .

PODPIS

Spis treści

1. Wprowadzenie

Modelowanie w ostatnich latach stało się istotnym zagadnieniem w wielu dziedzinach naukowych. Podstawowym celem modelowania stało się uproszczenie rzeczywistości umożliwiające poddanie jej procesowi badawczemu. Takie podejście umożliwia między innymi analizę zjawiska będącego przedmiotem badań w zmienionej skali czasowo-przestrzennej, adekwatną zmianę skali obiektu badań, analizę procesów zachodzących w skalach czasowych rzędu nano- lub mikrosekund czyli trudnych do uchwycenia w warunkach laboratoryjnych. Modelowanie umożliwia również badanie wybranego, jednego aspektu zagadnienia będącego przedmiotem analizy.

Odrębnym zagadnieniem w technikach modelowania jest modelowanie obiektów i scen trójwymiarowych. Dotyczy ono tworzenia obiektów i scen wykorzystujących głównie obiekty dostępne w bibliotekach różnych programów i zasadniczo ogranicza się do scen statycznych. Znacznie bardziej ciekawym rozwiązaniem jest propozycja połączenia scen pochodzących ze świata rzeczywistego z obiektami lub scenami generowanymi komputerowo. Klasyczne w tym zakresie rozwiązania bazują na połączeniu obrazu świata rzeczywistego dostarczanego przez kamerę z generowaną w czasie rzeczywistym grafiką 3D. Należy zwrócić uwagę, iż obraz świata rzeczywistego może być dostarczany przez sygnały o różnych częstotliwościach podstawowych. Spotyka się również rozwiązania, w których wykorzystywana jest fuzja sygnałów z kamer pracujących w zakresie fal widzialnych dla człowieka oraz fal radiowych. Powstały w ten sposób obraz określa się rozszerzoną rzeczywistością (ang. Augmented Reality).

W literaturze można spotkać wiele definicji rozszerzonej rzeczywistości. Najistotniejsze wydają się być jednak wymagania stawiane tego typu systemom. Wśród najważniejszych należy wymienić przede wszystkim łączenie świata rzeczywistego z wirtualnym, konieczność pracy w czasie rzeczywistym oraz umożliwianie ruchów każdego elementu w trzech wymiarach.

Pozornie prosty postulat dotyczący łączenia scen świata rzeczywistego z generowanymi scenami wirtualnymi kryje w sobie konieczność rozwiązania szeregu problemów naukowych oraz technologicznych. Efektywność zaproponowanych w tym zakresie metod przekłada się bezpośrednio na jakość uzyskanej sceny (modelowania). Dodatkowe ograniczenie wymuszające konieczność generowania wynikowej sceny w czasie rzeczywistym wymusza użycie właściwych dla danego zastosowania algorytmów dedykowanych dla danej architektury sprzętowej oraz ich specyficzną implementację.

1.1. Cele pracy

Celem głównym niniejszej pracy jest stworzenie systemu do tworzenia aplikacji z zakresu rozszerzonej rzeczywistości. Szczególny nacisk zostanie położony na jakość uzyskiwanych scen, co jest zdeterminowane głównie precyzją łączenia obrazu rzeczywistego i scen wirtualnych. Dla tak zaproponowanego celu ogólnego zrealizowano kilka celów szczegółowych. Należą do nich:

- Analiza dostępnych algorytmów łączenia obrazu rzeczywistego i scen generowanych.
- Analiza metod oceny jakości sceny rozszerzonej rzeczywistości oraz propozycja własnych kryteriów w tym zakresie.
- Propozycja własnych algorytmów dla metod rozszerzonej rzeczywistości uwzględniających określone kryteria jakości.
- Ocenę jakości zaproponowanych algorytmów.
- Sformułowanie ogólnych zasad tworzenia aplikacji dla przedmiotowego zakresu.

1.2. Zawartość pracy

Niniejsza praca zawiera opis realizacji merytorycznie spójnych etapów koniecznych dla stawianego celu ogólnego. W rozdziale pierwszym ....

2. Przegląd istniejących rozwiązań w zakresie rozszerzonej rzeczywistości

W tym rozdziale zostało omówione kilka bibliotek implementujących narzędzia wykorzystywane w rozszerzonej rzeczywistości. W internecie można znaleźć dużo w tym także darmowych i open source bibliotek, które możemy wykorzystać do różnych programów na przykład gier, które wykorzystują rozszerzoną rzeczywistość.

2.1. Ogólne założenia systemów rozszerzonej rzeczywistości

Na początku warto wspomnieć o kilku ogólnych założeniach, które charakteryzują systemy rozszerzonej rzeczywistości. Poniżej w kilku punktach zostało opisane kilka cech takich systemów:

– Obraz wirtualny bądź jego elementy nakładane są na obraz rzeczywisty uzyskany z kamery. Najczęściej punktem odniesienia są różnego rodzaju markery, jednakże mogą to być także różne charakterystyczne przedmioty otoczenia z pozyskanego obrazu rzeczywistego.

– Jednym z najtrudniejszych elementów w rozwijaniu aplikacji wykorzystujących rozszerzoną rzeczywistość jest dokładne obliczanie punktu widzenia użytkownika w czasie rzeczywistym, dzięki czemu wirtualne obrazy są dokładnie dostosowane do rzeczywistych obiektów.

– Obraz rzeczywisty ma zazwyczaj bardziej skomplikowane wymagania kalibracji kamery i rejestracji obrazu.

– Każdy z takich systemów wymaga kalibracji kamery, aby przy rozpoznawaniu obrazów można łatwo było określić położenie i orientację markerów, oraz odpowiednio wyświetlać przypisane im obiekty wirtualne. Niektóre z tych systemów posiadają własną implementację umożliwiającą taką kalibrację.

– Biblioteki opierające się na markerach, rozpoznają je najczęściej poprzez przetworzenie obrazu rzeczywistego na obraz binarny (czarno-biały) i znajdują w ten sposób czarną ramkę ograniczającą dany marker, a następnie zajmują się ich identyfikacją.

2.2. ARToolkit

Jest to jedna z najbardziej popularnych bibliotek open source wykorzystywanych w rozszerzonej rzeczywistości. Posiada także wiele rozszerzeń takich jak FLARToolkit i wiele podobnych. Kod źródłowy dla tego projektu jest zamieszczony na GitHub [1], skompilowane SDK dla wszystkich platform (Mac OS X, PC, Linux, Android, iOS) wraz z wtyczką dla ARToolKit Unity3D, dostępne są na stronie głównej projektu [1]. Zaletami tej biblioteki są:

– Solidne śledzenie markerów.

– Dobre wsparcie kalibracji kamery.

– Wieloplatformowe oraz posiada wpsarcie dla wielu języków programowania.

– Zoptymailizowane dla urządzeń mobilnych.

– Wsparcie dla unity3D oraz OpenSceneGraph.

ARToolKit wykorzystuje techniki wizualizacji komputerowej by obliczyć pozycję kamery i orientację względem kwadratowych kształtów lub płaskich powierzchni teksturowanych, pozwalając programiście nakładanie wirtualnych przedmiotów. ARToolKit obsługuje obecnie klasyczny kwadratowy znacznik, kod kreskowy 2D, wiele innych rodzajów markerów. Ponadto ARToolKit obsługuje dowolną kombinację powyższych razem. Szybkie, precyzyjne śledzenie dostarczone przez ARToolKit umożliwiło szybki rozwój tysiącu nowych i ciekawych aplikacji rozszerzonej rzeczywistości. ARToolkit obsługuje zarówno pliki wideo jaki obraz rzeczywisty bezpośrednio z kamerki.

Biblioteka opiera się na kwadratowych markerach składających się z białego tła oraz posiadających szeroką ciemną (zazwyczaj czarną) otoczkę wytyczającą granice markera. Konstrukcja ta czyni marker wyjątkowym, łatwym do rozpoznania. Są one rozpoznawane, wykrywane, a następnie używane do obliczania położenia w przestrzeni 3D. ARToolkit działa następująco:

– Aparat rejestruje wideo w polu widzenia kamery i wysyła je do komputera.

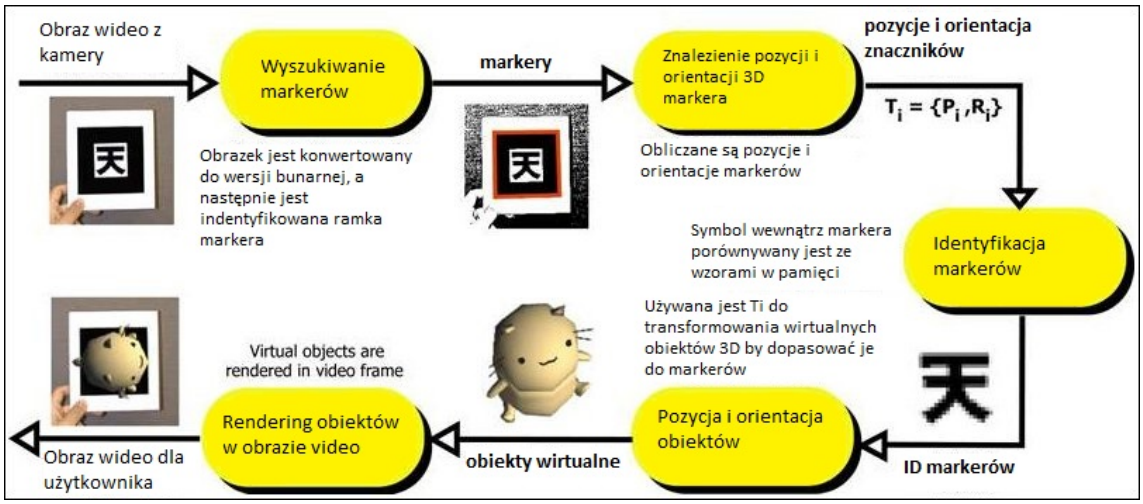
– Oprogramowanie na komputerze przeszukuje każdą klatkę wideo dla dowolnych kształtów kwadratowych (markery kwadratowe).

– Jeśli kwadratowy marker zostanie znaleziony i treść obrazu osadzona przez wzór, jest dopasowana i zidentyfikowana to oprogramowanie wykorzystuje matematykę do obliczenia, w stosunku do aparatu, zarówno pozycję czarnego kwadratu jak i orientację wzoru.

– Gdy pozycja i orientacja kamery są znane, model grafiki komputerowej jest rysowany za pomocą położenia obliczonej pozycji i orientacji dopasowania.

– Model ten jest rysowany w planie przechwyconego wideo i są śledzone ruchy wideo w tle, a następnie model jest dołączany do tła.

– Wynikiem końcowym jest pokazany na wyświetlaczu obiekt w tle obrazu z kamery, więc widz widzi renderowany graficzny model nad strumieniem wideo świata rzeczywistego.



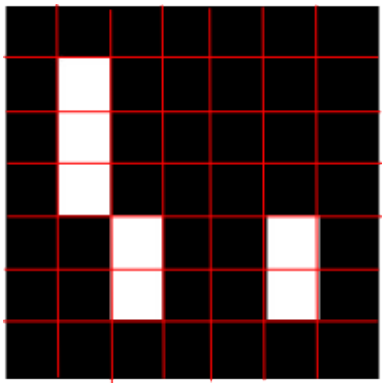
Rysunek 2.1: Działanie ARToolkit krok po kroku

Rysunek 2.1 podsumowuje powyższe kroki.

ARToolkit posiada jednak pewne ograniczenia. Obiekt wyświetlany jest tylko wtedy gdy marker jest w polu widzenia kamery. Jeżeli choć część markera zostanie czymś przysłonięta to wirtualny obiekt znika. Natomiast jeżeli granice markera będą poza polem widzenia kamery to obiekt zostanie obcięty. Istnieje również problem w zakresie śledzenia optycznego, ponieważ kiedy markery są przeniesione dalej od kamery to znaczniki zajmują mniej pikseli widzenia kamery i wynikiem jest posiadanie niewystarczającej ilości szczegółów by wyśledzić i zidentyfikować marker. Im większy fizycznie osadzony wzór markera tym dalej wzór markera można wykryć, a więc większa umiejętność wykrywania markera. Wreszcie, wyniki śledzenia są również zależne od warunków oświetleniowych. Zbyt mocne oświetlenie może tworzyć odbicia i odblaski (plamy) na paierowych markerach i tak sprawiają, że trudniej znaleźć kwadratowy marker. Cienie mogą być rzucone w poprzek papieru, rozbijając białe obszary na obrazie z kamery.

2.3. ArUco

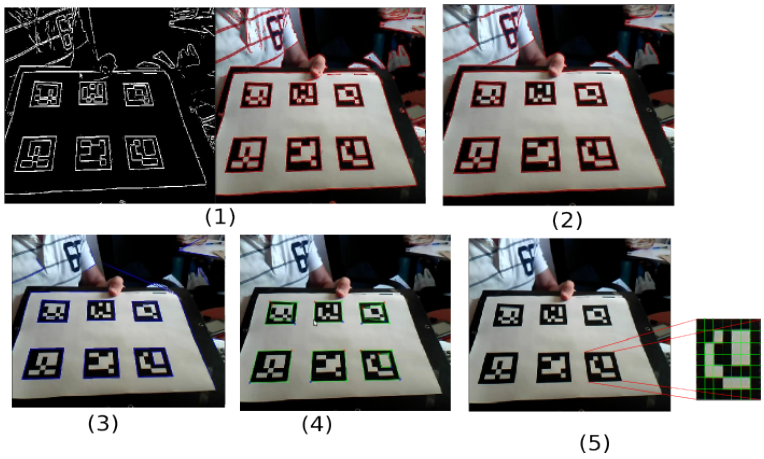
Podobnie jak ARToolkit biblioteka ArUco opiera sie na kwadratowych znaczniakach ograniczonych czarną ramką. Jest to prosta biblioteka napisana w C++ umożliwiająca detekcję markerów za pomocą niewielkiej ilości kodu. Biblioteka także nie jest bardzo wymagająca, gdyż potrzeba doinstalować tylko bibliotekę OpenCV, na której się ona opiera. Powstały także implementacje w językach Java (dokumentacja tylko w języku hiszpańskim na stronie twórców projektu) i Python, tak więc korzystanie z biblioteki jest możliwe na każdej platformie, na której da się zainstalować i skonfigurować bibliotekę OpenCV. Niestety nie ma możliwości aby wykorzystać ArUco do programowania aplikacji mobilnych. Zdecydowaną zaletą tego projektu jest uproszczona budowa markerów, które kodowane są binarnie za pomocą białych i czarnych pól kwadratowych wewnątrz ramki. Dodatkowo prosta instalacja oraz przykładowe programy podane na stronie producentów biblioteki [2] powodują, że zapoznanie się oraz jej użytkowanie jest dużo łatwiejsze i przyjemniejsze.



Rysunek 2.2: Przykładowy marker rozpoznawany przez ArUco.

Jak widać na rysunku 2.2 kwadratowy marker możemy podzielić siatką na mniejsze równych rozmiarów kwadraty (dokładnie 7x7) jednakże te brzegowe są poświęcone na ramkę markera, tak więc mamy 5x5 pól do binarnego kodowania znacznika (gdzie pole czarne oznacza 0, a pole białe 1) co powinno ułatwić, a nawet przyspieszyć jego detekcję. Każdy marker zawiera kod, który składa się z 5 słów, po pięć bitów każde. Pojedyńcze słowo ma przeznaczone 2 bity na informację oraz 3 pozostałe do detekcji błędów. Dzięki temu możemy uzyskać 2^{10} różnych id.

Detekcja markerów w ArUco przebiega następująco:



Rysunek 2.3: Detekcja markerów w ArUco.

– Zamieniamy uzyskany z kamery obraz na obraz binarny.

– Dla uzyskanego czarno-białego obrazu stosujemy Adaptive Thresholding by wykryć wszystkie krawędzie (Zdjęcie (1) na obrazku 2.3).

– Wykrywamy kontury. Po tym kroku mamy wykryte nie tylko kontury markerów, ale także różne inne nie potrzebne nam kontury, dlatego musimy je przefiltrować:

  – Usuwamy krawędzie z małą liczbą punktów (Zdjęcie (2) na obrazku 2.3).

  - Aproksymujemy łamane konturów i zachowujemy tylko te, które posiadają dokładnie 4 wypukłe rogi (Zdjęcie (3) na obrazku 2.3).

- Sortujemy uzyskane rogi w kierunku przeciwnym do wskazówek zegara.
- Usuwamy z prostokąty znajdujące się zbyt blisko siebie ponieważ Adaptive Thresholding wykrywa też prostokąty wewnętrzne w markerach (Zdjęcie (4) na obrazku 2.3).
- Identyfikujemy markery:
  - Usuwamy perspektywę projekcji (kamera może nie być ustawiona prostopadle do markera) aby uzyskać widok prostopadły markera (Zdjęcie (5) na obrazku 2.3).
  - Dzielimy marker za pomocą algorytmu Otsu na macierz czarnych i białych kwadratowych pól.
  - Identyfikacja kodu wewnątrz markera z pominięciem ramki.
- Dla prawidłowych markerów analizujemy rogi używając interpolacji subpikseli.
- Na końcu obliczamy współrzędne markerów w obrazie kamery poprzez jej parametry uzyskane w kalibracji.

Kolejną zaletą ArUco w przeciwieństwie do ARToolkit jest fakt, że jeżeli w trakcie nagrywania obrazu zostanie przysłonięta część markera to obiekt nie znika całkowicie tylko jest przysłaniany, co obrazuje chociażby przykładowa aplikacja do gry w szachy wykorzystująca ArUco [3] jak i kilka innych przykładów podanych na stronie twórców [2]. Jedną z drobnych niedogodności mogą być warunki oświetleniowe. Jeżeli marker zostanie oświetlony mocnym stumieniem światła to na obrazie pojawi się wielka biała plama w tym miejscu.

2.4. Vuforia

3. Propozycja autorskiego rozwiązania

Zaproponowane autorskie rozwiązanie.

4. Ocena jakości

Ocena jakości.

5. Podsumowanie i wnioski

Podsumowanie i wnioski.

Bibliografia

[1] Ben Vaughan, Philip Lamb, Wally Young. *ARToolkit*. http://www.artoolworks.com/ https://github.com/artoolkit

[2] Rafael Munoz-Salinas. *ArUco: a minimal library for Augmented Reality applications based on OpenCV*. http://www.uco.es/investiga/grupos/ava/node/26

[3] Augmented Reality Chess https://www.youtube.com/watch?v=sqAIvO6wCQI