POLITECHNIKA WARSZAWSKA

WYDZIAŁ MATEMATYKI  
 I NAUK INFORMACYJNYCH

Detekcja typu terenu na podstawie zdjęć satelitarnych i danych z OpenStreetMap

Projekt na potrzeby przedmiotu „Aplikacje i usługi GIS”

Tomasz Luśtyk

Paweł Paradowski

Spis treści

[1. Wstęp teoretyczny 3](#_Toc386547143)

[2 Opis zadania 3](#_Toc386547144)

[3 OpenCV - krótki opis biblioteki 4](#_Toc386547145)

[3.1 OpenCV – wykorzystane funkcjonalności do projektu 4](#_Toc386547146)

[3.2 Co udało się zrobić 5](#_Toc386547147)

[3.3 Detekcja konturu 5](#_Toc386547148)

[4 Podejście awaryjne – Matlab 5](#_Toc386547149)

[5 Podejście algorytmiczne 5](#_Toc386547150)

[6 Literatura 6](#_Toc386547151)

# Wstęp teoretyczny

Pokrycie terenu jest ściśle związane z jego użytkowaniem przez człowieka i jest bardzo istotnym elementem zmieniającego się środowiska na Ziemi. Odzwierciedla ono biofizyczne cechy środowiska, natomiast użytkowanie terenu mówi nam o jego funkcji w przestrzeni. W badaniach teledetekcyjnych, obserwując pokrycie terenu, możemy wnioskować o jego użytkowaniu.

Szczególnie istotnym źródłem informacji o środowisku geograficznym są dane satelitarne. Przy robieniu badań dotyczących zmian pokrycia terenu bardzo ważnymi cechami zdjęć satelitarnych są:

* synoptyczność,
* duża powtarzalność obrazowania takich samych terenów,
* automatyzacja procesu ich interpretacji,
* dostępność zdjęć.

[](http://wiki.openstreetmap.org/w/images/4/4f/MarekDuzyObszarNaRastrze4.jpg)

Obraz nr 1. Konstruowanie linii prostopadłych do osi konstrukcjyjnej

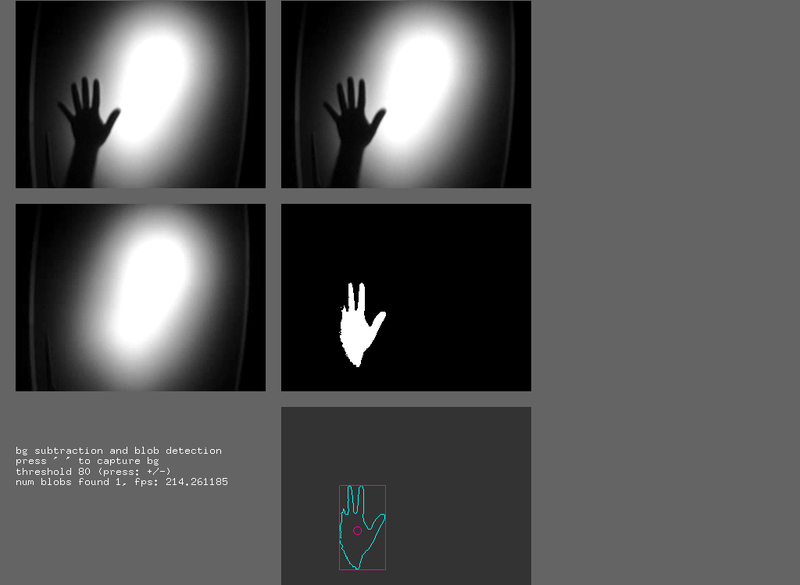
# Opis zadania

Celem naszego projektu jest rozpoznawanie różnych obszarów terenu na podstawie zdjęć satelitarnych i danych udostępnianych przez takie serwisy jak m.in. OpenStreetMap czy GoogleMaps. Chociaż w oryginalnym temacie projektu było wspomniane o celu minimalnym (rozróżnianie budynków od reszty terenu), będziemy starali się rozpoznawać w miarę możliwości inne rodzaje terenu.

Z dostępnych w literaturze oraz sieci materiałów wybraliśmy dwa podejścia, które chcielibyśmy wykorzystać przy analizie zdjęć satelitarnych do rozpoznawania określonego przez nas wzorca. Skupiliśmy się na pierwszym z nich (opisanym poniżej) mając przy tym nadzieję, że to właśnie on pomoże nam w rozwiązaniu naszego problemu projektowego.

# OpenCV - krótki opis biblioteki

Naszym pierwszym podejściem, które chcemy wykorzystać jest uznana biblioteka typu opensource – OpenCV. Jest to biblioteka funkcji wykorzystywanych podczas obróbki obrazu, oparta na otwartym kodzie i zapoczątkowana przez firmę Intel. Jest ona wieloplatformowa (m.in. możemy z niej korzystać pod Linuxem, Windowsem oraz w Mac OS X), stworzona w języku C. Istnieją także nakładki umożliwiające korzystanie z niej również w językach C++, C# oraz Python. Głównym zamysłem autorów było oparcie się na przetwarzaniu obrazu w czasie rzeczywistym.

[](http://upload.wikimedia.org/wikipedia/commons/8/87/OfxOpenCV.png)

Obraz nr 2. Działanie OpenCV w oprogramowaniu openFrameworks

## OpenCV – wykorzystane funkcjonalności do projektu

Główną funkcjonalnością, którą będziemy wykorzystywać z OpenCV będzie rozpoznawanie wzorca w obrazie. W samej bibliotece możemy korzystać z tzw. klasyfikatora kaskadowego Haara, który zawiera w sobie sieć neuronową. Można w niej m.in. nauczyć klasyfikator na podstawie kolekcji przykładów (pozytywnych i negatywnych) rozpoznawać element (z obrazka pozytywnego) na dowolnym obrazku. Dwa główne kroki rozpoznawania wzorca to:

* Trenowanie
* Detekcja na obrazie

Dokumentacja, która dotyczy tej biblioteki opisuje m.in. jak wytrenować klasyfikator (jak go przygotować, aby dobrze rozpoznawał zadane wzorce). Do realizacji tych kroków, OpenCV udostępnia dwie aplikacje – *opencv\_haarttraining* oraz *opencv\_traincascade*. Skorzystamy z tej drugiej, ponieważ jest to nowsza wersja poprzedniej, rozszerzona o obsługę wielu procesorów i pewne nowe możliwości.

Dodatkowowo, są też udostępnione pomocne narzędzia:

* *opencv\_createsamples* – stosowana do przygotowania zbioru danych pozytywnych i negatywnych przykładów. Zbiór danych jest w formacie wspieranym przez obie (wymienione wyżej) aplikacje. Na wyjściu dostajemy plik z rozszerzeniem \*.vec, co jest binarnym odpowiednikiem zawierającym obrazek.
* *opencv\_performance* – może być użyte do oszacowania jakości klasyfikatora, ale tylko dla *opencv\_haartraining*.

## Co udało się zrobić

Na ten moment, udało nam się zgromadzić skrypty pozwalające na sprawne przygotowanie próbek pozytywnych z przygotowanych przez nas obrazów (skrypt wywołujący w odpowiedni sposób *opencv\_createsamples* dla listy podanych przez nas obrazków np. budynków). Zbadaliśmy też działanie funkcji *opencv\_traincascade* i po początkowych nieudanych próbach, dobraliśmy odpowiednie parametry pozwalające na przygotowanie klasyfikatora. Jesteśmy więc w stanie uzyskać plik .xml z klasyfikatorem w oparciu o przykłady pozytywne i negatywne. Typowy trening klasyfikatora trwa zazwyczaj około kilku godzin do 1 doby (zależnie od podanych parametrów, co ma też wpływ na docelową jakość klasyfikatora). Obecnie posiadamy 1 klasyfikator, wyuczony na obrazach budynków o czerwonych dachach ze zdjęć satelitarnych Google Earth.

Niestety, dotychczas nie udało nam się wykorzystać klasyfikatora do detekcji budynków. Jest to spowodowane problemami z wczytaniem klasyfikatora z pliku xml. Planujemy skorzystać z wrappera EmguCV, który pozwoli nam na kodowanie w C#. Uprzednio próbowaliśmy naszych sił z OpenCVSharp i samego OpenCV (C++), jednak bez skutku. W razie kolejnych niepowodzeń możemy spróbować uruchomić detekcję w środowisku node.js bądź python.

Wiadomości podawane na wyjściu narzędzia trenującego klasyfikator wskazują, że nasz pierwszy klasyfikator jest sprawny, choć o jego skuteczności w praktyce musimy się jeszcze przekonać.

## Detekcja konturu

Klasyfikator zapewni nam wykrycie prostokątnego obszaru zawierającego budynek, jednak nie pozwoli na wyznaczenie obrysu budynku. W tym celu możemy zastosować algorytm detekcji krawędzi w obrębie prostokąta wskazanego przez klasyfikator.

# Podejście awaryjne – Matlab

W razie niepowodzenia z OpenCV, chcielibyśmy wykorzystać środowisko Matlab, w którym również istnieje możliwość uczenia klasyfikatora kaskadowego. Co więcej, wydaje nam się, nauczenie i przetestowanie klasyfikatora w Matlabie jest prostsze, niż w OpenCV, jednak możliwości parametryzacji zdają się być mniejsze, natomiast bardziej zaawansowane wykorzystanie klasyfikatora w aplikacji mniej wygodne. Ponieważ wykorzystanie OpenCV jest dla nas bardziej priorytetowe, niż wykorzystanie Matlaba, nie będziemy szerzej opisywać mechanizmu uczenia i rozpoznawania wzorca w środowisku Matlab [1] .

# Podejście algorytmiczne

Dotarliśmy do pracy [2], w której opisywany jest algorytm detekcji budynków w obrazach satelitarnych za pomocą algorytmu o charakterze analitycznym, bez uczenia maszynowego. Jednakże algorytm ten jest złożony, a biorąc pod uwagę brak dostępu do kodu aplikacji, implementacja tego algorytmu jest poza naszym zasięgiem.

# Literatura

1. <http://www.mathworks.com/help/vision/ref/traincascadeobjectdetector.html>
2. <http://iris.usc.edu/outlines/papers/1998/lin-cviu98.pdf>