Politechnika Warszawska
Wydział Elektroniki i Technik Informacyjnych

Przetwarzanie cyfrowe obrazów

Rozpoznawanie loga sieci Tesco - sprawozdanie

Michał Dziekoński

30 stycznia 2017

1 Temat projektu

Dla indywidualnie wybranej klasy obrazów dobrać, zaimplementować i przetestować odpowiednie procedury wstępnego przetworzenia, segmentacji, wyznaczania cech oraz identyfikacji obrazów cyfrowych. Powstały w wyniku projektu program powinien poprawnie rozpoznawać wybrane obiekty dla reprezentatywnego zestawu obrazów wejściowych.

Wybrana klasa obrazów to zdjęcia zawierające logo sieci hipermarketów Tesco - czerwony tekst składający się z kolejno ułożonych liter T, E, S, C, O. Zestaw obrazów składa się z pięciu plików, z czego na każdym z nich znajdują się przynajmniej dwa fragmenty do rozpoznania.

2 Kompilacja i instrukcja uruchomienia

2.1 Kompilacja

Kompilacja programu jest możliwa za pomocą dwóch programów:

- CMake - komendy **cmake** i **make** (kompatybilne z CLion)
- Scons - komenda **scons** (możliwe do uruchomienia z terminala, buduje program do pliku **build/run**)

Do kompilacji wymagany jest kompilator wspierający opcję **C++1z** (wymagane wsparcie dla Ńested namespace definition"). Kompilatory wspierające tę opcję to:

- GCC w wersji 6.0.0

- Clang w wersji 3.6
- MSVC w wersji 14.3

Do kompilacji wymagana jest również biblioteka OpenCV w wersji co najmniej 3.2.0, zainstalowana na stałe w systemie operacyjnym.

2.2 Instrukcja obsługi

Program uruchamiany z okna terminala przyjmuje następujące opcje:

- **–file=FILEPATH** (wymagany) - ścieżka do pliku obrazu z rozpoznawanym obrazem
- **–binary** (opcjonalny) - flaga przełączająca tryb wyświetlania wyniku z obrazka oryginalnego na obrazek binarny (po operacji binaryzacji)

3 Opis działania projektu

Program stworzony na potrzeby projektu ma za zadanie wykonać serię operacji na wcześniej załadowanym obrazie w celu wykrycia fragmentów zawierających logo Tesco. Ogólną sekwencję działania procesu rozpoznawania można opisać następująco:

1. Wstępna obróbka obrazu - wstępne polepszenie jakości obrazu w kontekście rozpoznawania, ma na celu uwydatnić cechy przydatne w dalszej obróbce lub usunąć zakłócenia negatywnie wpływające na ten proces.
2. Binaryzacja obrazu - transformacja obrazu kolorowego do postaci binarnej, gdzie elementy znaczące w punktu widzenia dalszego przetwarzania są koloru białego, pozostałe są czarne.
3. Obróbka obrazu binarnego - obraz w postaci binarnej pozwala na wykonanie pewnych dodatkowych transformacji, które dodatkowo mogą usprawnić proces rozpoznawania. W tym kroku możliwe są np. operacje poprawiania konturów elementów obrazu.
4. Segmentacja - jest to wstępne grupowanie pikseli obrazu w segmenty składające się na większe, jednolite obiekty sceny.
5. Wstępna filtracja - ponieważ pewne grupy pikseli nie stanowią żadnych znaczących z punktu widzenia rozpoznawania obrazów elementów (np. obiekty tła, które przeszły przez proces binaryzacji razem z właściwymi obiektami do rozpoznania), przed uruchomieniem właściwego rozpoznawania warto z góry usunąć fragmenty, które na pewno nie będą składać się na rozpoznawany fragment.
6. Detekcja poszukiwanych fragmentów - ostatni krok polegający na dopasowaniu znalezionych obiektów do siebie w taki sposób, by sprawdzić czy tworzą razem spójną całość, i tym samym są (lub nie) obiektem który mamy rozpoznać w scenie.

3.1 Wstępna obróbka obrazu

W pierwszym kroku sprawdzono, czy obrazy wejściowe wymagają jakiejkolwiek obróbki wstępnej. W ramach wyostrzania obrazów zastosowano filtr Unsharp Mask. Okazało się jednak, że żadne ze zdjęć nie wymaga stosowania tego filtru (dla żadnego zdjęcia nie osiągnięto ani poprawy, ani pogorszenia jakości w dalszym rozpoznawaniu obrazu). Obrazy wykorzystane w testach były już ostre, więc nie wymagały dodatkowego ulepszania.

3.2 Binaryzacja obrazu

Kolejnym krokiem była binaryzacja obrazu. Ponieważ zadanie polegało na rozpoznawaniu czerwonego teksu, naturalnym pierwszym wyborem było zastosowanie prostego progowania obrazu za pomocą doświadczalnie otrzymanych wartości dla każdej ze składowych. Okazało się jednak, że nie wszystkie zdjęcia mają tą samą barwność tekstu zawartego w logo, przez co nie udało się uzyskać ogólnego progu dla wszystkich próbek.

Znacznie lepsze rezultaty dało wykorzystanie miksera kolorów do kanału monochromatycznego - dla każdego z kanału kolorów udało się dobrać takie współczynniki, które pozwoliły wyodrębnić interesujące informacje, porzucając jednocześnie szum tła. Po mieszaniu kolorów zastosowano proste progowanie w skali szarości.

3.3 Obróbka obrazu binarnego

Następny krok polegał na dodatkowym wyostrzeniu konturów obiektów sceny. W ramach eksperymentów zastosowano dwa przekształcenia - erozję oraz dylację, które w teorii powinny poprawić kontury obiektów oraz wypełnić wszelkie ewentualne ubytki w obiektach.

Okazało się jednak, że nie wszystkie obrazy dobrze żeagująńa takie przekształcenia. Ze względu na to, że niektóre fragmenty obrazów były stosunkowo małe, transformacje doprowadzały do odwrotnego efektu niż zamierzony - zamiast poprawiać kontury, usuwały piksele niszcząc kształty obiektów. Ponieważ pierwotne kontury były na tyle dobre by móc kontynuować proces rozpoznawania, transformacje dylacji i erozji zostały wyłączone.

3.4 Segmentacja

Przygotowany obraz binarny został następnie poddany procesowi segmentacji, czyli grupowania pikseli w spójne obszary. Wykorzystano do tego algorytm Floodfill, który dla każdego białego piksela starał się odpowiednio żalać"danym identyfikatorem jego sąsiadów, tworząc przez to zgrupowanie pikseli. Wykorzystana implementacja algorytmu Floodfill korzysta z ręcznie przygotowanego stosu na "przeszukiwanych sąsiadów"głównie ze względu na to, że standardowa wersja rekurencyjna znacznie spowalniała działanie programu, a dla większych obrazów mogła wywoływać błędy braku pamięci operacyjnej. Każda grupa pikseli została na końcu zapakowana do kontenera przechowującego zarówno same piksele, jak i tzw. Bounding Box całego obszaru.

Rysunek 3.1: Przykład segmentacji obrazu.

3.5 Wstępna filtracja

Kolejnym etapem było filtrowanie niechcianych obiektów ze sceny. W pierwszym kroku nastąpiła eliminacja obiektów które są zbyt małe lub zbyt duże jak na litery loga. W drugiej części sprawdzane były niezmienniki momentowe segmentów (od M1 do M7), które ze względu na swoje właściwości pozwalają wykrywać podobne elementów, niezależnie od ich skalowania czy obrotu. W ramach eksperymentów wyznaczono przedziały wartości tych niezmienników, w których można było stwierdzić czy dany segment należy do np. klasy liter T lub S. Wszystkie inne obiekty, których niezmienniki nie znajdowały się w odpowiednich przedziałach, były odrzucane.

3.6 Detekcja poszukiwanych fragmentów

Ostatnim krokiem było rozpoznanie i stwierdzenie, czy wykryte segmenty zawierające litery są poprawnie ułożone i stanowią w swoim obrębie logo Tesco. Jako "kontur"podstawowy zostały wykorzystane litery T i O, ponieważ leżą na skraju loga. Program sprawdza wszystkie możliwe kombinacje wykrytych liter T i O, próbuje dopasować je do jednego fragmentu (sprawdzając, czy odległość między ich środkami pozwala na umieszczenie wewnątrz mniej więcej trzech dodatkowych liter E, S, C). Jeśli to się powiedzie, program stara się znaleźć pasujące w przewidywane miejsca kolejne, wewnętrzne litery wyrazu. W przypadku dopasowania wszystkich pięciu liter tworzony jest nowy segment, stanowiący Bounding Box dla całego loga Tesco.

Rysunek 3.2: Przykład ostatecznego wyniku rozpoznawania loga Tesco. Obraz zawiera 11 rozpoznanych fragmentów, wyświetlany jest w formacie binarnym.

4 Wnioski

Zgodnie z założeniami, projekt rozpoznaje wszystkie poprawne wystąpienia loga sieci Tesco na wszystkich obrazach z zestawu testowego. Na niektórych zdjęciach znajdują się również fragmentaryczne loga (np. naderwana część pudełka, zasłaniająca jedną z liter), które choć w fazie segmentacji i filtracji zostały poprawnie rozpoznane jako fragmenty całości loga, tak w ostatecznej fazie nie zostały uznane za pełne logo (co jest zgodne z oczekiwaniami - tylko logo zawierające wszystkie litery w poprawnej kolejności zostanie oznaczone jako pełny segment loga).

Binaryzacja obrazu z wykorzystaniem miksera kolorów okazała się znacznie lepszą metodą niż progowanie kolorów. Pozwoliła na znacznie lepsze wyodrębnienie właściwych fragmentów loga przy jednoczesnym wyeliminowaniu mniejszych lub większych anomalii będących składnikami tła. Podczas eksperymentów, przejście na binaryzację mikserem pozwoliło znacznie skrócić czas wykonywania programu w dalszych fazach. Dodatkowym atutem było również uniezależnienie się od operatorów poprawy jakości, czy to w fazie wstępnej (kolorowy obrazek) czy w fazie binarnej - mikser, w przeciwieństwie do progowania kolorów, nie wprowadzał znacznych anomalii na krawędziach czy wewnątrz obiektów, dzięki czemu obyło się bez użycia operacji morfologicznych do poprawy kształtów obiektów.

Potencjalnym polem do rozwoju projektów (również interdyscyplinarnie) jest klasyfikacja liter. W tym projekcie rozpoznawanie przynależności do danej klasy (np. litery

T) jest oparte na prostym systemie zakresowym, gdzie każda litera ma określony eksperymentalnie zakres wartości współczynników momentowych. Tak wyznaczone zakresy naturalnie nie będą pasowały do większych próbek testowych, gdzie może okazać się że klasy i ich ograniczenia wartościowe są nadmiernie dopasowane do początkowego zestawu testowego. Przy większym, oznakowanym zbiorze testowym można by zastosować np. naiwny klasyfikator Bayesowski do automatycznego wyznaczenia poprawnych przedziałów, dla zbiorów nieoznakowanych można by podjąć próbę wykorzystania algorytmów klastrujących jako formę uczenia bez nadzoru.