Wydział Elektroniki
i Technik Informacyjnych
POLITECHNIKA WARSZAWSKA

Przetwarzanie Cyfrowe Obrazów

Wykrywanie logo sieci sklepów Lidl

Kaczmarski Robert
293377

Warszawa 2022

1. Polecenie

   Dla obrazów sieci sklepów Lidl należy 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. W trakcie projektu należy przetestować wybrane algorytmy i ocenić ich praktyczną przydatność.

2. Logo sieci sklepów Lidl

Rys. 1. Logo sieci sklepów Lidl [1].

   Na Rys. 1. zostało przedstawione oficjalne logo sieci sklepów Lidl. Logo Lidl ma kształt kwadratu i można je podzielić na części w zależności od koloru:

   - niebieskie: tło, dwie litery ‘L’ oraz litera ‘D’;

   - czerwone: okrąg, kropka nad ‘I’ oraz reszta litery ‘I’;

   - żółte: koło z wyciętymi literami, środek litery ‘D’.

   Miejsca w jakich można zobaczyć logo Lidl to między innymi: nad wejściem do sklepu, na torbach na zakupy lub innych gadżetach bądź ubraniach tej marki np. klapki, skarpetki.

3. Akwizycja obrazu

   Z uwagi na brak sklepu lidl blisko miejsca zamieszkania, zdjęcia zostały pozyskane dzięki wyszukiwarce Google oraz oficjalnemu fanpage sklepu Lidl na Twitterze. Zdjęcia przedstawiają logo nad wejściem do sklepu w dobrym oświetleniu, a także z gorszym podświetleniem. Zebrane zdjęcia zapisane zostały w formacie jpeg.

4. Ogólny algorytm przetwarzania obrazu

   Algorytm do wykrywania logo Lidl można podzielić na poszczególne fazy:
   - Wstępne przetwarzanie – zmiana rozmiaru, poprawa jakości obrazu.
   - Zmiana przestrzeni barw – oddzielenie informacji o kolorze i jasności.
   - Segmentacja- wydzielenie obszarów.
   - Wyznaczanie cech – dla każdego wykrytego segmentu

- Identyfikacja – na podstawie opisu kształtu.

5. Użyte narzędzia

Do wykonania zadania został użyty język Python w wersji 3.9.9. Wybór języka padł na Pythona z uwagi na to, iż z Pythona korzystam na co dzień, a C++ nie używałem od kilku lat. Pewnym jest uzyskanie gorszych wyników wydajnościowych niż gdyby został użyty język C++. W programie wykorzystywane są tylko dozwolone funkcje pakietu OpenCV odpowiedzialne za czytanie obrazu z pliku, zapis obrazu do pliku oraz wyświetlenie obrazu. Wszystkie inne funkcjonalności zostały napisane samodzielnie.
Instrukcja użycia programu znajduje się w pliku README.md.

6. Wstępne przetwarzanie

W pierwszym kroku algorytmu wykrywania logotypu Lidl jest zmniejszenie rozmiaru obrazu, poprawa jakości oraz usunięcie zakłóceń.

6.1. Zmiana rozmiaru

Do zmiany rozmiaru obrazu często wykorzystuje się metody interpolacji. W programie zostały zaimplementowane dwie metody interpolacji. Pierwsza z nich to metoda interpolacji najbliższym sąsiadem (resize. NearestNeighbourInterpolationResizer()), która polega na kopiowaniu wartości sąsiedniego piksela w miejsce uzupełnianego piksela. Druga zaimplementowana metoda to metoda interpolacji bikubicznej (resize. BicubicInterpolationResizer()). Metoda ta oblicza wartość piksela jako średnią arytmetyczną poziomów jasności piksela interpolowanego i wszystkich pikseli bezpośrednio sąsiadujących z tym pikselem. W interpolacji bikubicznej bierze się pod uwagę szesnaście najbliższych pikseli uzupełnianego piksela.

Rys. 2. Zmiana rozmiaru metodą interpolacji najbliższym sąsiadem dla obrazu foto_1.jpeg na rozmiar (400, 500)

Rys. 3. Zmiana rozmiaru metodą interpolacji bikubicznej dla obrazu foto_1.jpeg na rozmiar (400, 500).

Dla zaimplementowanej metody bikubicznej na Rys. 3. widać różne małe defekty powstałe w wyniku interpolacji. W szczególności defekty pojawiły się przy cienkim drucie znajdującym się na obwodzie dachu sklepu. Być może jest to

spowodowane bugiem w kodzie, dlatego w dalszej części projektu została wykorzystana szybsza metoda interpolacji najbliższym sąsiadem.

6.2. Filtracja

Do poprawy jakości zostały zaimplementowane filtry następujące filtry:

- Filtr gaussa (filters.filter_gaussian()
- Filtr medianowy (filters.filter_median())
- Filtr rankingowy (filters.filter_ranking())
- Erozja (filters.erosion())
- Dylacja (filters.dilation())

6.3. Wyrównanie histogramu

Jako możliwość poprawienia kontrastu obrazu dodana została metoda wyrównania histogramu (histogram. histogram_equalization()).

7. Zmiana przestrzeni barw

Dla przetworzonego obrazu została zmieniona przestrzeń barw z BGR na HSV (color_conversions.BGR2HSV()). Zmiana ta ułatwia wykrywanie na zdjęciu obiektów o danej barwie, ale różnym oświetleniu. Do konwersji pikseli została użyta formuła znaleziona w internecie [2].
Dodatkowo dodana konwersja odwrotna, czyli z HSV do BGR (color_conversions.HSV2BGR())oraz konwersja z BGR do RGB (color_conversions.BGR2RGB()), która była potrzebna, aby wykorzystać pakiet matplotlib do rysowania ramki dla każdego znalezionego logo bądź segmentu. Przykładowy wynik zmiany przestrzeni barw został zmieszczony na Rys. 4.

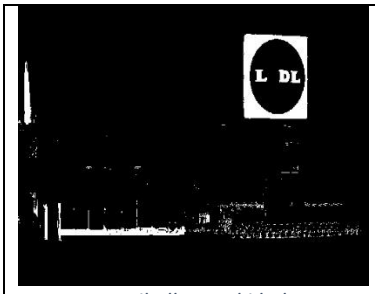
Rys. 4. Obraz foto_1.jpeg po zmianie barw z BGR na HSV.

8. Segmentacja

Najważniejszym etapem wykrywania logo jest segmentacja obrazu. W programie została użyta segmentacja przez progowanie (segmentation.ColorMask.create_mask()) dla brzegowych wartości H, S, V zawartych w tabeli 1. W wyniku segmentacji przez progowanie powstały 3 maski zawierające jedynie piksele zawierające się w danym zakresie.

Tabela 1. Wartości brzegowe h, s, v segmentacji przez progowanie.

	Hmin	Smin	Vmin	Hmax	Smax	Vmax
niebieski	200	70	0	300	255	255
czerwony	300	50	50	360	255	255
żółty	20	70	100	60	255	255

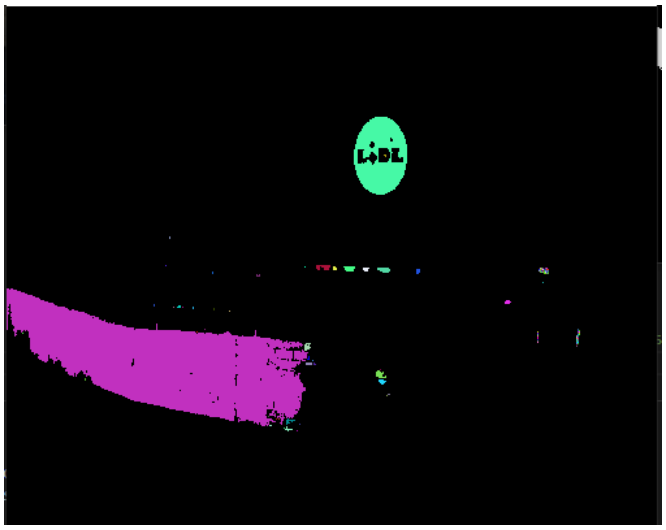


Rys. 5. Wynik dla maski koloru niebieskiego.

Rys. 6. Wynik dla maski koloru żółtego.

Rys. 7. Wynik dla maski koloru czerwonego.

Następnym krokiem w programie było podzielenie otrzymanych masek kolorów na segmenty połączonych ze sobą pikseli. Do oznaczenia segmentów wykorzystano zmodyfikowany algorytm floodfill (segmentation.get_segments()), który wykorzystuje się do wypełniania zamkniętych obszarów [3]. Po wyznaczeniu binarnej maski danego koloru, dla każdego piksela wywoływany jest algorytm floodfill, który sprawdza, czy piksel jest biały, jeśli tak to przeszukiwaniem wszerz (BFS) odwiedza sąsiednie piksele i zmienia je na wcześniej ustalony, losowy kolor. Przykład działania algorytmu prezentuje Rys. 4.



Rys. 8. Wynik działania algorytmu floodfill dla obrazu foto_5.jpeg i maski koloru żółtego.

Znalezione w ten sposób fragmenty tworzą obiekt klasy Segment. W wyniku segmentacji może powstać bardzo dużo małych fragmentów, dlatego znalezione fragmenty dla każdego koloru przesiewane są przez sito akceptujące tylko fragmenty o polu większym niż zadana wartość (metoda segmentation.filter_small_segments()).

9. Wyznaczenie cech

Klasyczne cechy takie jak: pole powierzchni, obwód, średnica, środek ciężkości, nie są wystarczające do akceptowalnego rozpoznawania segmentu. Wykrywane logo, może mieć różne rozmiary, być pochylone bądź obrócone. Alternatywnymi cechami, które można wyznaczyć są momenty geometryczne. Momenty geometryczne dzielimy na centralne i zwykłe. Cechą momentów centralnych jest niezmienność niezależnie od dokonanej translacji. Dzięki momentom centralnym można policzyć niezmiennicze momenty, które jak sama nazwa wskazuje mogą być niezmienne w zależności od rozmiaru, pochylenia lub obrotu segmentu.
Wykorzystane wzory do wyznaczenia cech zostały zaczerpnięte z instrukcji do laboratorium nr 3 z przedmiotu POBR [4].

10. Identyfikacja

Końcowym krokiem programu jest identyfikacja, czyli niezmiennicze momenty wszystkich segmentów każdego z trzech badanych kolorów, porównywane były z wartościami modelowymi. Modelowe wartości to graniczne wartości niezmienniczych momentów, które zostały wyznaczone w sposób eksperymentalny - metodą prób i błędów.
Logo sieci sklepów Lidl składa się z ośmiu elementów, co zostało wylistowane w drugim rozdziale tego sprawozdania. Żółte wypełnienie litery D jest zbyt małe, dlatego nie będzie brane pod uwagę podczas identyfikacji.
Podejść do identyfikacji logo Lidla może być wiele, zależnie od tego na jakich częściach logo się skupimy. Przykładowo można zaniechać identyfikacji niebieskiego tła i skupić się tylko na żółtym kole i niebieskich literkach i wtedy dla segmentów,

które zostały po porównaniu z modelowymi wartościami sprawdzone zostaną stosunki rozmiarów segmentów. Jeśli odległości między odpowiednimi literami i kołem będą spełniały pewien z góry ustalony stosunek, to złożenie tych segmentów będzie uznane za wykryte logo.
W projekcie jednak analizowane były segmenty niebieskiego tła i żółtego koła. Przefiltrowane segmenty były następnie porównywane z modelowymi wartościami niezmienniczych momentów, następnie do każdego znalezionego niebieskiego tła przypisywane było żółte koło (identification. identify_logo_from_segments()), które mieści się w środku prostokąta opisanego na niebieskim tle. Każda znaleziona para (niebieskie tło, żółte koło) uznawane były za znalezione logo, po czym wszystkie znalezione logo były wyświetlane na obrazie w formacie RGB (draw.draw_bounding_boxes()). Otrzymane w tym podejściu wyniki były zadowalające, chociaż niewykluczone są przyszłe usprawnienia działania programu.

11. Wyniki

Otrzymane wyniki zostały zaprezentowane na poniższych rysunkach. Wszystkie wyniki otrzymane zostały z wykorzystaniem interpolacji najbliższym sąsiadem.

Rys. 8. Wykryte logo dla obrazu foto_3.jpg.

Rys. 9. Wykryte loga dla obrazu 2_loga.jpg.

Rys. 10. Wykryte logo dla obrazu foto_5.jpeg.

Rys. 11. Wykryte logo dla obrazu foto_1.jpg

Rys. 12. Wykryte logo dla obrazu foto_2.jpg.

Rys. 13. Wykryte logo dla obrazu foto_4.jpeg.

Ciekawa sytuacja wystąpiła dla zdjęcia foto_4.jpeg. Wykryte zostało tylko jedno logo. Na Rys. 14. widać, że prawe logo nie stanowi jednego dużego segmentu, ale dużo mniejszych. Co więcej aby temu zaradzić wykorzystywane były różne filtry: gaussa, rankignowy, medianowy, erozja, dylacja, operacja zamknięcia. Nie przynosiło to jednak pozytywnych rezultatów. Być może ze zdjęciem jest coś nie tak, ponieważ jego akwizycja odbyła się z fanpage na Twitterze.

Rys. 14. Podział na segmenty algorytmem floodfill dla obrazu foto_4.jpeg.

12. Możliwe usprawnienia

Jednym z możliwych usprawnień może być sprawdzanie porównanie stosunków wysokości do szerokości znalezionych fragmentów. Kolejne możliwe usprawnienie to dodatkowe zastosowanie zaimplementowanych filtrów tak, aby niebieskie tło nie łapało podobnych pikseli obok, przez co zwiększało się jego pole powierzchni.
Inne usprawnienie mogłoby polegać na dodatkowym dodaniu przy analizowaniu całości logo np. czerwonej litery ‘I’ lub liter ‘L’, ‘D’. Utwierdziłoby to w przekonaniu, że wykryte logo to rzeczywiście logo Lidl, a nie np. logo IKEA.

13. Bibliografia

[1] https://pl.m.wikipedia.org/wiki/Plik:Lidl-Logo.svg
[2] https://www.rapidtables.com/convert/color/rgb-to-hsv.html
[3] https://pl.wikipedia.org/wiki/Flood_fill
[4] pobr_cpom_lab3.pdf