[POBR] Rozpoznawanie logo pepsi na obrazie

Adamski Maciej

Styczeń 2023

1 Wprowadzenie

Celem projektu było stworzenie oprogramowania umożliwiającego identyfikację logo Pepsi na zdjęciach (przykładowe logo pokazane jest na rysunku 1). Rozwiązanie podzielone jest na trzy główne części - poprawę jakości obrazu, segmentację i analizę cech - których wynikiem są plamy pixeli uznawane za poszukiwany obiekt. Wykorzystywane w projekcie algorytmy nie pochodzą z żadnej zewnętrznej biblioteki - są napisane ręcznie. Dodatkowo wraz z projektem dostarczone zostały testy jednostkowe pozwalające testować poszczególne metody.

Jedynymi wykorzystywanymi bibliotekami w projekcie są:

- *OpenCV* - do wczytywania i przechowywania w macierzach obrazów
- *Catch2* - do tworzenia testów jednostkowych

Rysunek 1: Przykład logo Pepsi

2 Zmiana przestrzeni barw *BGR* na *HSV*

OpenCV wczytując zdjęcia do macierzy *cv::Mat* przechowuje je domyślnie w przestrzeni barw *BGR* (Blue, Green, Red). Ze względu, że każda składowa tej przestrzeni przenosi jednocześnie informację o chrominancji i luminancji oraz dodatkowo są ze sobą wzajemnie skorelowane - są kiepskim wyborem w analizie i segmentacji. Z tego powodu zostaje zmieniona na barwy *HSV* (Hue, Saturation, Value). W jej przypadku tylko jedna składowa niesie informację o luminacji. Dodatkowo są w dużo bardziej luźnej relacji niż w przypadku BGR lub RGB.

2.1 Obliczanie poszczególnych składowych HSV

Ze względu, że *cv::Mat* dla obrazów przechowuje wartości pikseli w typie *uchar* (8 bitów: 0-255), składowe HSV muszą być przechowywane w inny sposób niż w literaturowym opracowaniu tej przestrzeni barw. Składowa H zostaje spłaszczona dwukrotnie do zakresu $0-180$ (np. literaturowe $\angle 360°$ odpowiada liczbie 180). Dla wartości S oraz V zakres $0-1$ zostaje rozciągnięty do $0-255$.

Dla algorytmów obliczania każdej z trzech składowych HSV wspólne jest wykorzystanie wartości maksymalnych i minimalnych danego piksela w BGR.

```
auto max = std::max({b, g, r});
auto min = std::max({b, g, r});
```

Hue Jeśli $max == min$ to wartość H równa jest 0. W innym przypadku wynik zależy, która ze składowych BGR ma największą wartość.

```
auto delta = max - min;
auto hue { 0.0 };

if (max == r)
{
hue = (g - b) / delta;
}
else if (max == g)
{
    hue = (b - r) / delta;
    hue += 2;
}
else if (max == b)
{
    hue = (r - g) / delta;
    hue += 4;
}

hue *= 30;

if (hue < 0)
    { hue += 180 }
```

Saturation Jeśli maksymalna wartość w przestrzeni BGR danego piksela równa jest 0 to wartość nasycenia również będzie 0. W pozostałych przypadkach obliczana jest w następujący sposób:

```
auto saturation = (max - min) / max * 255.0;
```

Value Wyliczenie składowej V jest najprostsze, ponieważ wynikową wartością jest maksymalna składowa BGR.

```
auto value = max;
```

3 Poprawa jakości zdjęcia

Zanim na obrazie zaczną segmentowane, a następnie analizowane obszary zdjęcie zostaje poddane poprawie jakości - w projekcie odbywa się to przez zastosowanie filtru medianowego oraz wyostrzającego.

3.1 Filtr medianowy

Na początku obraz zostaje poddany filtracji z wykorzystaniem filtru medianowego o oknie analizy 3×3. Jest to nieliniowy filtr pozwalający na usuwanie zakłóceń lub szumu ze zdjęć - w szczególności zniekształcenia typu *z ang. Salt and Pepper*. Polega on na wybraniu środkowej wartości (mediany) wszystkich pikseli w danym otoczeniu.

Ze względu na trójkanałową naturę obrazu filtracja tego typu przeprowadzana jest już w przestrzeni barw HSV. W kolorach RGB istnieją pewne relacje między poszczególnymi składowymi, które powinny zostać zachowane - natomiast filtr medianowy pomija te zależności. W przestrzeni HSV składowe nie posiadają tak mocnej relacji, przez co taki obraz może być w ten sposób przetwarzany.

```
// Values to wektor zawierajacy wartosci pikseli w danym oknie analizy
    std::sort(values.begin(), values.end());
    return values[values.size() / 2];
```

3.2 Filtr wyostrzający

Do wyostrzenia obrazu został wykorzystany filtr o masce pokazanej poniżej. Dla takiego jądra wykonuje się operację splotu z obrazem.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Źródło: https://en.wikipedia.org/wiki/Kernel_(image_processing).
Pozwoliło to uwypuklić krawędzie na obrazie w szczególności dla niebieskiego elementu logo.

4 Segmenatacja

Etap ten polega na wydzieleniu pewnych jednorodnych obszarów, które mogą być poszukiwanymi elementami obrazu. W projekcie składa się on z wyznaczenia plamek spełniających pewne kryteria koloru w przestrzeni HSV, a następnie poddania ich operacjom morfologicznym. Wyjściem segmentacji jest obraz binarny oraz zbiór plamek zapisanych w wektorze.

4.1 Progowanie koloru

Najpierw segmentacja obrazu odbywa się z użyciem metody progowania. Podawany jest pewien zakres wartości składowych *HSV*, które musi spełniać dany piksel, żeby przyjął wartość 255 w wyjściowym obrazie binarnym.

Progowane jest zarówno czerwona, jak i niebieska część logo Pepsi - zakres wartości poszczególnych elementów podane są w tabelach 1 i 2.

	Minimum	**Maksium**
Hue	165	10
Saturation	70	255
Value	70	255

Tabela 1: Wartości progowanych składowych HSV dla czerwonego elementu

Dla koloru czerwonego zakres wartości *Hue* może być niezrozumiały ze względu, że wartość minimalna jest większa niż maksymalna. Wynika to ze stożkowego charakteru przestrzeni barw *HSV*. W jej reprezentacji barwa czerwona przechodzi przez kąt $\angle 360° \ / \ \angle 0°$. Z tego względu podany w tabeli 1 zakres można zapisać jako sumę zbiorów $< \angle 330°, \angle 360° > \cup < \angle 0°, \angle 20° >$, czyli w przypadku wartości w projekcie - $< 165, 180 > \cup < 0, 10 >$.

	Minimum	**Maksium**
Hue	95	130
Saturation	50	255
Value	50	255

Tabela 2: Wartości progowanych składowych HSV dla niebieskiego elementu

W przypadku czerwonego koloru progowanie daje bardzo dobre rezultaty, w których pozostała część obrazów zostaje odrzucona. Gorzej sprawdza się to dla niebieskiej części logo - w jej przypadku część etykiety na butelce czy puszki ma podobne barwcy co logo. Z tego względu niebieskie progowanie przynosi gorsze rezultaty. Największy problem w tym wypadku jest, jeśli zanika biały okrąg wokół logo, ponieważ uzyskane w dalszej części potoku plamki często tworzone są z połączenia etykiety i poszukiwanego elementu zdjęcia.

```
bool checkIfColorInRange(uchar color, const ColorRange& colorRange)
{
    if (color < colorRange.min || color > colorRange.max)
```

```
        {
            return false;
        }
        return true;
    }
```

4.2 Operacja morfologiczna otwarcia

Morfologiczne otwarcie polega na wykonaniu najpierw operacji erozji, a następnie dylacji. Erozja polega na usunięciu z obrazu pikseli, które mają w sąsiedztwie co najmniej jeden piksel nieaktywny - jest to również filtr minimalny wybierający wartość minimalną z obszaru. Dylacja natomiast aktywuje piksel, jeśli przynajmniej jeden z jego otoczenia jest aktywny - odpowiednik filtru maksymalnego.

(a) Obraz wejściowy (b) Erozja (c) Dylacja (d) Otwarcie

Rysunek 2: Przykład działania operacji morfologicznych o wielkości 5×5

W projekcie służy do kolejnego odfiltrowania szumu, w szczególności takiego, który powstałego w procesie progowania. Dla obszarów czerwonego i niebieskiego użyto operacji otwarcia o następujących oknach analizy:

	Erozja	**Dylacja**
Czerwone	3×3	3×3
Niebieskie	5×5	5×5

Tabela 3: Rozmiar okna erozji i dylacji dla obu elementów logo

4.3 Tworzenie obszarów punktów

Po wykonaniu operacji progowania (a także poprawy przez otwarcie) uzyskuje się obraz binarny. Dla niego wykonywane jest przeszukiwanie przestrzeni w celu utworzenia wektorów pixeli, które ze sobą sąsiadują tworząc ciągły obszar. Działa to na zasadzie przeszukiwania wszerz, gdzie dla aktywnego pixela dodaje się wszystkich sąsiadów do kolejki analizy.

5 Analiza cech

Następnym etapem rozpoznawania logo Pepsi jest analizy cech i na ich podstawie filtracji wykrytych obszarów. Oceniane są trzy następujące cechy:

- liczba pikseli należących do obszaru
- stosunek boków prostokąta opisanego na obszarze
- niezmienniki momentowe

W poniższych rozdziałach (liczba odrzuconych obszarów) skuteczność poszczególnych filtracji badana jest dla połączonego obrazu ze wszystkich 8 podstawowych zdjęć testowych.

5.1 Wielkość obszaru - liczba pikseli

Pierwszym kryterium filtracji wykrytych plamek jest odrzucenie ich ze względu na zbyt małą lub zbyt dużą liczbę pikseli, które zawierają. Binarny obraz będący wynikiem może zawierać dużo pojedynczych pikseli, które spełniają kryterium barwy - swego rodzaju szum. Filtrowanie na podstawie rozmiaru pozwoli je w bardzo łatwy sposób odrzucić. Dodatkowo w projekcie detekcji logo Pepsi występują na przykład etykiety na butelce lub puszki w kolorze bardzo zbliżonym do niebieskiej części wykrywanego elementu - takie obszary będą zawierały dużo większą liczbę pikseli.

```
auto pointsCount = points.size();
return pointsCount >= min && pointsCount <= max;
```

	Minimalny rozmiar	Maksymalny rozmiar
Czerwony	1 000	200 000
Niebieskie	300	75 000

Tabela 4: Rozmiar okna erozji i dylacji dla obu elementów logo

Dla 8 testowanych podstawowych zdjęć pozwoliło to bardzo drastycznie zmniejszyć liczbę plamek pozostałych do analizy.

- dla niebieskiego koloru: z 3139 obszarów usunięto 2675
- dla czerwonego kolor: z 3877 obszarów usunięto 3631

5.2 Współczynnik kształtu - stosunek boków

Kolejnym kryterium filtrowania obszarów jest stosunek boków prostokąta opisanego na danym obszarze. Taka analiza cech jest możliwa ze względu, że elementy logo mają specyficzny kształt.

```
auto lenghtA = bottomRightCorner.x - topLeftCorner.x;
auto lenghtB = bottomRightCorner.y - topLeftCorner.y;

auto ratio = static_cast<double>(lenghtA) / static_cast<double>(lenghtB);
if (ratio >= 1)
{
    ratio = 1.0 / ratio;
}
return ratio >= min && ratio <= max;
```

	Wartość minimalna	Wartość maksymalna
Czerwone	0.65	1.00
Niebieskie	0.40	0.95

Tabela 5: Rozmiar okna erozji i dylacji dla obu elementów logo

Dla 8 testowanych podstawowych zdjęć pozwoliło to odrzucić niektóre plamki:

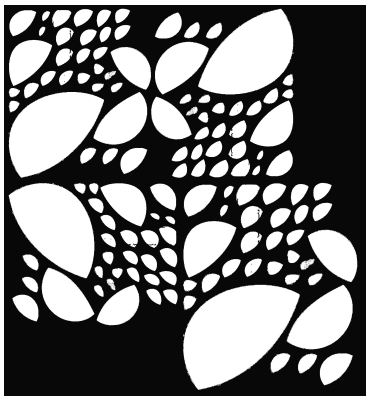
- dla niebieskiego koloru: z 464 obszarów usunięto 127
- dla czerwonego kolor: z 246 obszarów usunięto 134

5.3 Momenty geometryczne

Ostatnią analizowaną cechą są niezmienniki momentowe. Charakteryzują one figury płaskie i rozłożenie ich punktów w układzie współrzędnych. Niektóre z nich potrafią dla danej figury przyjmować praktycznie stałe (stąd nazwa niezmienniki) wartości niezależnie od translacji, rotacji czy skalowania. Znalezienie takich wartości może pozwolić identyfikować obiekty ze analizę położenia należących do niego pixeli.

Ze względu na wieloetapowość algorytmu obliczającego niezmienniki momentowe *Hu*, nie zostaje tu on opisany.

W projekcie filtrowane są wartości wszystkich momentów, a nie tylko wybranych przyjmujących stałą wartość. Wynika to z faktu, że nie daje to wielkiego narzutu obliczeniowego, a pozwala ewentualnie odfiltrować dodatkowe obszary. Zakresy wartości momentów były wstępnie wyliczane na spreparowanym obrazie składającym się z różnych wystąpień danego obiektu poddanemu różnym przekształceniom (przykład rysunek 3). Następnie doświadczalnie dostosowywane do rzeczywistych obrazów.



Rysunek 3: Przykład spreparowanego obrazu, na którym obliczane były zakresy momentów geometrycznych

Ostateczne zakresy niezmienników dla obszarów niebieskich i czerwonych zostały przedstawione w tabelach 6 i 7.

Moment	Wartość minimalna	Wartość maksymalna
Hu 0	0.16	0.43
Hu 1	0.00013	0.097
Hu 2	0.00008	0.02
Hu 3	0.00000082	0.009
Hu 4	-0.000000015	0.00015
Hu 5	-0.0000018	0.0031
Hu 6	0.0064	0.021
Hu 7	-0.0012	0.00001
Hu 8	-0.00012	0.00001
Hu 9	-0.000053	0.0000039

Tabela 6: Zakres niezmienników geometrycznych dla czerwonej części logo

Moment	Wartość minimalna	Wartość maksymalna
Hu 0	0.2	0.43
Hu 1	0.014	0.135
Hu 2	0.0015	0.26
Hu 3	0.00015	0.0061
Hu 4	-0.000015	0.00005
Hu 5	-0.04	0.0006
Hu 6	0.007	0.015
Hu 7	-0.0027	0.00001
Hu 8	-0.000086	0.00001
Hu 9	-0.0003	0.00033

Tabela 7: Zakres niezmienników geometrycznych dla niebieskiej części logo

Filtrowanie po niezmiennikach geometrycznych pozwoliło odrzucić:

- dla niebieskiego obszaru: z 337 obszarów usunięto 253
- dla czerwonego obszaru: z 112 obszarów usunięto 45

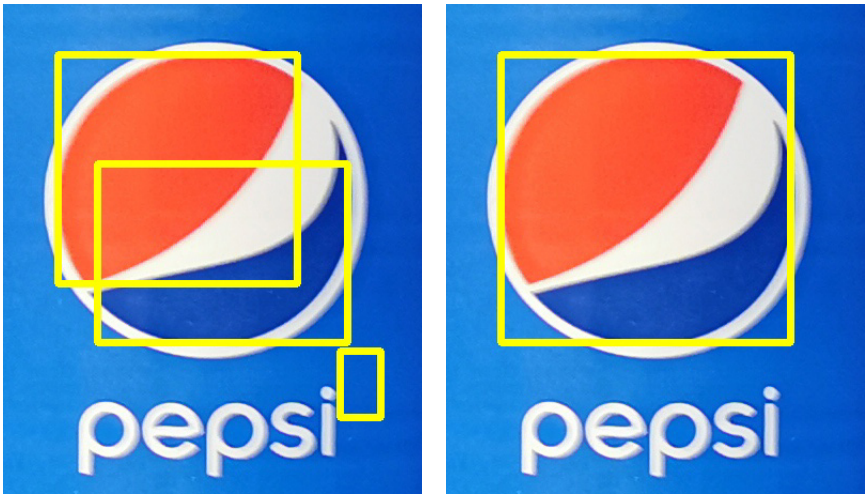
6 Łączenie niebieskich i czerwonych obszarów

Ostatnim elementem w potoku przetwarzania obrazu jest połączenie ze sobą sąsiadujących obszarów niebieskich i czerwonych. Etap ten spełnia również funkcję dodatkowej filtracji, ponieważ jeśli dany obszar nie stworzy z innym pary to zostaje usunięty i nie wyświetlany na finalnym obrazie.

Celem dobrania dwóch elementów do siebie wykorzystywany jest algorytm sprawdzający zawieranie się (ang. *overlapping*) prostokątów opisujących dany obszary. Wprowadzony został również parametr pozwalający na pewien błąd i margines między dwiema ramkami - pozwoliło to polepszyć ostateczny wynik detekcji. W finalnej wersji projektu maksymalna odległość między prostokątami wynosiła 10 pikseli.

Testowane było również podejście opierające się na maksymalnej odległości między środkami dwóch obszarów. Jednak rezultaty uzyskiwane w tym podejściu okazały się gorsze niż przez sprawdzanie nakładania się prostokątów.

Na rysunku 4 przedstawiony jest wynik łączenia z wykorzystaniem nakładania się, a także przykład jak pozwala to dodatkowo odrzucić błędne obszary.



(a) Przed łączeniem (b) Po łączeniu

Rysunek 4: Wynik działania operacji łączenia obszarów

```
// overlapError -> parametr wprowadzajacy margines bledu,
// ktory sprawia, ze laczone sa rowniez prostokaty
// nie nakladajace sie, ale bedace bliskim sasiedztwie

// rB - redBlob, obszar czerwonego koloru
// bB - blueBlob, obszar niebieskiego koloru
// RC - prawy dolny rog prostokata
// LC - lewy gorny rog prosotkata

if (!(      ((rB_RC.y + overlapError) < (bB_LC.y - overlapError))
         || ((bB_RC.y + overlapError) < (rB_LC.y - overlapError))
         || ((rB_RC.x + overlapError) < (bB_LC.x - overlapError))
         || ((bB_RC.x + overlapError) < (rB_LC.x - overlapError))
    ))
{
```

```
        // Prostokaty nakladaja sie
    }
```

7 Możliwość dalszych prac i poprawy jakości projektu

W projekcie logo Pepsi było rozpoznawane na podstawie dwóch elementów składowych - niebieskiej i czerwonej części. Możliwe byłoby wykrywanie dodatkowo białej przestrzeń między nimi lub biały okrąg całego logo. Sprawiłoby to, że finalne łączenie plamek wymagałoby trzech bliskich obszarów.

Poprawę jakości mogłoby również przynieść lepsze dobranie parametrów progowania czy niezmienników geometrycznych. Do tego celu można by zastosować wybrany algorytm przeszukiwania przestrzeni, genetyczny czy optymalizacyjny. Zmiana hiperparametrów mogłaby przynieść lepsze filtrowanie na etapie analizy cech, ale również bardziej dokładną segmentację po kolorze obszaru.

Warto by również zaimplementować i przetestować inne algorytmy polepszające jakość obrazu. Przykładowo, zastąpienie maski *sharpening* metodą *USM (UnSharp Mask)* mogłoby lepiej wyostrzyć obraz, co miałoby pozytywny wpływ na segmentację obrazu.

8 Testy jednostkowe

W projekcie zostały przygotowane również testy jednostkowe sprawdzające działanie metod:

- Konwersji kolorów BGR -> HSV
- Filtrowania, operacji konwolucyjnych
- Operacji morfologicznych erozji i dylacji
- Progowania kolorów
- Detekcji plam/obszarów
- Obliczania momentów i niezmienników