Przetwarzanie i Rozpoznawanie Obrazów
Aleksander Grzyb 100998

Wykrywanie powtórzeń danego obiektu

Wprowadzenie

Celem tej pracy jest stworzenie algorytmu, który przyjmowałby na wejście dwa obrazy. Pierwszy z nich przedstawia obiekt, który chcemy wykrywać na drugim obrazie. Algorytm powinien wykryć wszystkie powtórzenia danego obiektu, niezależnie od jego rotacji czy rozmiaru.

W tej pracy zostanie przedstawiona użyta metoda, następnie zostaną zaprezentowane osiągnięte wyniki. Na koniec wymienione będą dalsze kroki jakie mogą być podjęte by ulepszyć zastosowaną metodę.

Metoda

Żeby wykryć jakikolwiek obiekt trzeba mieć wcześniej jasno określone jego cechy. Cechy takiego obiektu powinny być unikalne dla niego by bezsprzecznie określały ten obiekt. Istnieje dużo różnych metod, które definiują w jaki sposób znaleźć unikalne cechy na obrazie, a później te cechy reprezentować w programie.

W tej pracy do wykrywania i reprezentowania cech zostały użyte dwa algorytmy, które są w tej pracy również porównywane:

- **SURF (Speeded Up Roboust Features)**
- **SIFT (Scale Invariant Feature Transform)**

Każdy z tych algorytmów jest w stanie wykryć szukany obiekt niezależnie od jego rotacji czy skali. Podczas wykrywania obiektu używając **SURF**, czy **SIFT** wymagane są następujące kroki:

1. Wykrywane są cechy w obrazie z obiektem oraz w obrazie, gdzie szukamy ten obiekt.
2. Obliczane są reprezentacje tych cech.
3. Reprezentacje tych cech są porównywane by znaleźć te, które do siebie najbardziej pasują pomiędzy dwoma obrazami. W tej pracy wykorzystywana jest technika **Nearest Neighbors**, która zwraca nam najbardziej pasujące do siebie cechy pomiędzy obrazem z szukanym obiektem, a obrazem, w którym szukamy tego obiektu.
4. Mając już najbardziej podobne cechy obliczamy macierz transformacji pomiędzy punktami gdzie znajdują się te najbardziej podobne cechy pomiędzy dwoma obrazami.

5. Mając macierz transformacji oraz rozmiar obrazu z obiektem możemy jego ramkę przekształcić tą macierzą i umieścić na obrazie, w którym szukamy obiektu. Jeżeli wszystko zostało wykonane poprawie ta ramka będzie wskazywać położenie szukanego obiektu.

Największym wyzwaniem oraz problemem by zrealizować cel tej pracy jest wykrywanie więcej niż jednej instancji danego obiektu. Problem polega na tym, że wykrywając cechy na obrazie, który zawiera wiele szukanych obiektów cechy z różnych obiektów mogą być łączone w jeden obiekt oraz błędnie interpretowane. Rozważmy przykład wykrywania białej kartki papieru na obrazie gdzie znajdują się cztery takie kartki. Najpierw wykrywamy cechy na obrazie, na którym znajduje się tylko jedna kartka. Najbardziej unikalne cechy białej kartki to jej rogi. Następnie wykrywamy cechy na obrazie gdzie znajdują się cztery kartki. Zostaje wykryte 16 różnych rogów. Algorytm nie mając pojęcia w jaki sposób kartki są umieszczone na obrazie oraz nie mając pojęcia jakiego kartka jest rozmiaru może uznać, że np. lewy-górny róg pierwszej kartki i prawy-górny róg drugiej kartki to jedna kartka. Taki problem jest ciężki do wyeliminowania jeżeli nie posiadamy żadnej dziedzinowej wiedzy na temat obiektu, którego szukamy.

Rozwiązaniem użytym w tej pracy jest stworzenie tzw. **Sliding Window**. Jest to technika, która polega na tym, że poruszamy się oknem o określonej wielkości po obrazie, w którym mamy wykryć obiekt. Dzięki tej technice algorytm analizuje tylko określony fragment obrazu. Przez co zwiększamy szanse na to, że w wyciętym fragmencie będzie tylko jeden obiekt. Miejsce wykrytego obiektu jest przechowywane we wewnętrznej strukturze danych obiektu i później te współrzędne obiektu w fragmencie są zamieniane na współrzędne tego obiektu w całym obrazie. Poniżej znajduje się obraz, w którym przedstawiona została technika **Sliding Window**. Jak widzimy w **Sliding Window** aktualnie znajduje się tylko jedna gazeta przez co staje się o wiele prostsza do wykrycia. **Sliding Window** porusza się od lewej do prawej strony, a następnie schodzi jeden wiersz niżej.

Podczas tworzenia algorytmu powstał jeszcze jeden problem. Algorytm czasem zwraca wykryty obiekt w miejscu gdzie go nie ma. W przypadku takiej sytuacji wykryty obiekt ma mocno zdeformowane kształty, ponieważ jest nieprawdziwy. Dzięki temu ten problem można rozwiązać poprzez obliczenie stosunku szerokości i długości wykrytego obiektu i porównanie jej ze stosunkiem szerokości i długości obiektu oryginalnego. Jeżeli różnica pomiędzy tymi stosunkami nie jest większa niż określony akceptowalny błąd, obiekt jest uznawany jako rozpoznany.

Poniżej jest przedstawiony zastosowany algorytm w uproszczonej formie:

```
loadQueryImage()
loadSceneImage()
detectKeypointsInQueryImage() // Wykrywanie cech w obrazie z obiektem
calculateDescriptorsForKeypointsInQueryImage() // Obliczanie
reprezentacji dla cech w obrazie z obiektem
for (x = 0; x < sceneImageWidth - windowWidth; x = x + step)
      for (y = 0; y < sceneImageHeight - windowHeight; y = y + step)
            createWindowImage(x, y, windowHeight, windowWidth) //
            Poruszanie Sliding Window po obrazie
            detectKeypointsInWindowImage() // Wykrywanie cech w Sliding
            Window
            calculateDescriptorsForKeypointsInWindowImage() // Obliczenie
            reprezentacji dla cech w Sliding Window
            findMatchesBetweenQueryImageAndWindowImage() // Wykorzystując
            technikę Nearest Neighbors odnajdujemy najbardziej podobne do
            siebie cechy pomiędzy dwoma obrazami
            findTransformMatrixBetweenMatches() // Mając najlepsze cechy
            liczymy macierz transformacji pomiędzy punktami, w których są
            najbardziej podobne do siebie cechy
            if objectWasCorrectlyMatched() // Porównujemy stosunki
            długości i szerokości pomiędzy obiektem, a obiektem, który
            znaleźliśmy
                  transformQueryImageFrame() // Jeżeli obiekt został
                  poprawnie wykryty transformujemy ramkę obiektu za pomocą
                  macierzy transformacji obliczonej wcześniej
                  drawFrameInSceneImage() // Rysujemy tę ramkę na obrazie,
                  w którym szukamy obiektów i ramka wskazuje nam miejsce,
                  w którym znaleźliśmy obiekt
            end
      end
end
```

Wyniki

Algorytm był testowany na dwóch obiektach, gazecie oraz kawie.

Przez to, że te obiekty są kolorowe oraz mają dużo w sobie charakterystycznych kształtów idealnie nadają się do testowania algorytmu.

KAWA

Algorytm **SURF**

Algorytm **SIFT**

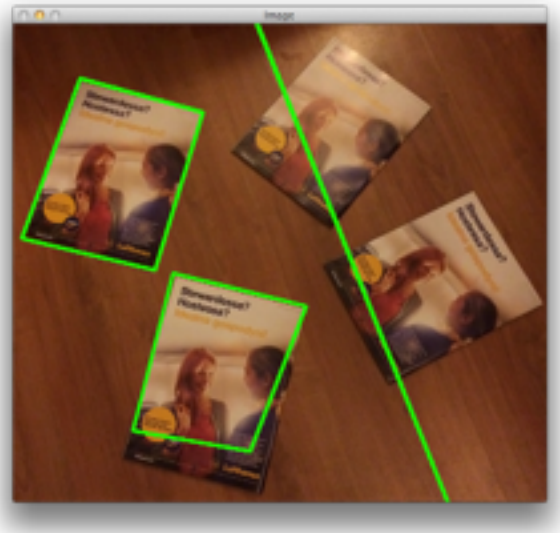
Czasy przetwarzania:

- **SURF** - 0,970695 s.
- **SIFT** - 2,168330 s.

Algorytm **SIFT** o wiele lepiej poradził sobie z tym zadaniem wykrywając więcej obiektów. Natomiast algorytm **SURF** jest o wiele szybszy, co może się przydać jeżeli w czasie rzeczywistym chcielibyśmy śledzić te obiekty.

GAZETA

Algorytm **SURF**



Algorytm **SIFT**

Czasy przetwarzania:

- **SURF** - 1,127139 s.
- **SIFT** - 2,781576 s.

Jak widzimy algorytm **SURF** był w tym przypadku o wiele bardziej skuteczny oraz znowu się potwierdziło, że **SURF** jest o wiele szybszy od algorytmu **SIFT**.

Podsumowanie oraz dalsze kroki

Podsumowując algorytm **SURF** jest o znacząco szybszy od algorytmu **SIFT** przez co jest idealnym kandydatem jeżeli chcielibyśmy przeprowadzać śledzenie obiektu w czasie rzeczywistym. Jeżeli chodzi o skuteczność to tak jak pokazały wyniki w tej pracy zależy dużo od sceny oraz obiektu. Algorytm można by było poprawić jeszcze poprzez:

- Zastosowanie zmiennej wielkości **Sliding Window**. W moim programie używana jest stała wielkość tego okna. Jest to trochę naiwne, ponieważ nie wiemy jakiej wielkości są obiekty w obrazie, w którym je szukamy. Mogą być o wiele mniejsze niż **Sliding Window** lub w ogóle się w nim nie mieścić.
- Zastosowanie algorytmu do usuwania tła z obrazu. Dużo cech było wykrywanych w miejscach gdzie jest tło, przez co generowało się dosyć sporo błędów.
- Zastosowanie uczenia maszynowego. Dla tej opcji musielibyśmy posiadać duży zbiór zdjęć obiektów. Uczenie maszynowe moglibyśmy zastosować na podstawie cech zwracanych przez algorytmy **SURF** i **SIFT**.