Techniki multimedialne
Ćwiczenie 3
Detekcja krawędzi

Opracował Dawid Warchoł
Politechnika Rzeszowska
Katedra Informatyki i Automatyki

Cel ćwiczenia

Ćwiczenie to ma na celu zapoznanie studenta z:

- metodami detekcji krawędzi na obrazach,
- scalaniem krawędzi o różnych orientacjach,
- binaryzacją obrazów monochromatycznych.

Należy napisać program, który wykrywa krawędzie poziome oraz pionowe na obrazie oraz scala je, a następnie dokonuje binaryzacji z eksperymentalnie dobranym progiem. Program należy napisać w języku C++ z wykorzystaniem biblioteki OpenCV oraz wstępnie przygotowanych fragmentów kodu zapisanych w pliku *edge_detection_todo.cpp*. Fragmenty te powinny być uzupełnione zgodnie z komentarzami umieszczonymi w odpowiednich miejscach oraz instrukcją wykonania ćwiczenia przedstawioną poniżej. Plik z kodem programu oraz dodatkowe pliki (np. obrazów) zapewnia prowadzący zajęcia.

Uwaga: Nie należy wpisywać wymiarów poszczególnych obrazów "na sztywno" w kodzie. Warto skorzystać z pól `rows` i `cols` obiektów przechowujących obraz.

Uwaga 2: Wykrywanie krawędzi, ich scalenie i skalowanie należy wykonać bez użycia gotowych funkcji, takich jak *sobel*, *filter2D,* c*onvertScaleAbs*, *normalize* z biblioteki OpenCV.

Zadanie 1. Detekcja krawędzi poziomych i pionowych

a) Wczytać do pamięci obraz JPEG (domyślnie plik *hand.jpg*). Obraz powinien być przechowywany jako 1-kanałowa (szara) mapa bitowa.

b) Wykonać detekcję krawędzi poziomych, która polega na zastąpieniu wartości każdego piksela obrazu sumą jego sąsiedztwa przemnożoną przez odpowiedni element maski. Należy wykorzystać maskę podaną przez prowadzącego. Przykładowe maski dla detekcji krawędzi poziomych przedstawiono poniżej.

Maska Prewitt dla krawędzi poziomych: $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Maska Sobela dla krawędzi poziomych: $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Otrzymana tablica może zawierać elementy ujemne lub przekraczające wartość 255. Następnym krokiem jest więc przeskalowanie jej wartości bezwzględnej do przedziału [0 - 255]:

$$kr'_{poz} = scale_{0,255}(|kr_{poz}|).$$

Krok ten jest istotny, ponieważ zmienne typu *unsigned char* w języku C++ nadają się do przechowywania wartości ze wspomnianego zakresu (obrazy w bibliotece OpenCV są przechowywane jako *unsigned char*). Należy postąpić analogicznie, jeśli zamierzamy wyświetlić jedynie krawędzie pionowe.

Wskazówka: Skalowanie wartości *x,* gdzie $x \in X$, do przedziału $[Y_{min} - Y_{max}]$ można wykonać w następujący sposób:

$$x' = \frac{(x - X_{min}) \cdot (Y_{max} - Y_{min})}{(X_{max} - X_{min})},$$

gdzie:

- X - zbiór skalowanych wartości,
- x' - przeskalowana wartość,
- X_{min}, X_{max} - minimalna i maksymalna wartość występująca w zbiorze X,
- Y_{min}, Y_{max} - minimalna i maksymalna wartość przedziału docelowego.

Należy zwrócić uwagę, że wartości X_{min} i x_{max} obliczamy po wyznaczeniu wartości bezwzględnych obrazu krawędzi.

c) Wykonać detekcję krawędzi pionowych w sposób analogiczny do wykrywania krawędzi poziomych, przy użyciu innej maski. Należy wykorzystać maskę podaną przez prowadzącego. Przykładowe maski dla detekcji krawędzi pionowych przedstawiono poniżej.

Maska Prewitt dla krawędzi pionowych: $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$

Maska Sobela dla krawędzi pionowych: $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$

Zadanie 2. Scalenie wykrytych krawędzi poziomych i pionowych

Należy scalić wykryte krawędzie obu orientacji poprzez dodanie do siebie wartości bezwzględnych odpowiadających sobie elementów tablicy krawędzi poziomych i pionowych. Scalanie krawędzi wykonujemy w następujący sposób:

$$kr = scale_{0,255}(|kr_{poz}| + |kr_{pion}|),$$

gdzie:

- kr - tablica krawędzi scalonych,
- kr_{poz} - tablica krawędzi poziomych,
- kr_{pion} - tablica krawędzi pionowych,
- $scale_{x,y}$ - skalowanie do przedziału [x-y].

Wartości X_{min} i X_{max} w przypadku krawędzi scalonych należy wyznaczać na podstawie sumy wartości bezwzględnych tablic kr_{poz} i kr_{pion}. Należy zwrócić uwagę, że suma ta nie może być zapisana w tablicy typu *unsigned char* przed skalowaniem, ponieważ nastąpiłoby obcięcie informacji. Można więc wykorzystać pomocniczą tablicę typu *int* lub porównywać jedynie tymczasowe sumy pojedynczych pikseli obu tablic.

Zadanie 3. Binaryzacja z progiem ustalonym eksperymentalnie

Wykonać binaryzację obrazu scalonych krawędzi, czyli przypisać wartość 0 pikselom o jasności poniżej ustalonego progu oraz wartość 255 pozostałym pikselom.

Fragmenty jaśniejsze na obrazie z wyznaczonymi krawędziami oznaczają większe prawdopodobieństwo występowania w danym miejscu krawędzi. Tak więc po wykonaniu binaryzacji piksele białe będą oznaczały, że w danym miejscu występuje krawędź. Należy zdecydować, jaki próg binaryzacji jest najbardziej odpowiedni w przypadku naszego obrazu oraz algorytmu. Jeśli próg będzie zbyt mały, wynikowy obraz będzie zawierał fragmenty błędnie sklasyfikowane jako krawędzie; jeśli natomiast próg będzie zbyt duży, niektóre krawędzie w wynikowym obrazie nie zostaną zaznaczone.

Należy zastosować suwak (slider) biblioteki OpenCV (funkcja *createTrackbar*), który pozwala dynamicznie modyfikować próg binaryzacji. Ten element GUI wysyła sygnał, który może uruchomić funkcję ponownej binaryzacji obrazu za każdym razem, gdy użytkownik przesunie suwak.

Źródła

[1] M. Wysocki, T. Kapuściński: *Systemy Wizyjne*, Wydawnictwo Uniwersytetu Rzeszowskiego, 2013.

[2] OpenCV 3.4.5. dokumentacja: https://docs.opencv.org/3.4.5/.