Warszawa, 15.03.2017

Wykrywanie krawędzi – porównanie algorytmów

Dokumentacja wstępna

Łukasz Napora
Kamil Żak

1. Opis problemu

Istotną rolę w dziedzinie przetwarzania obrazów stanowią algorytmy wykrywania krawędzi. Idea oraz algorytmy istnieją od bardzo dawna, jednak w dobie szybkiego rozwoju sztucznej inteligencji oraz IoT (ang. *Internet of Things*), ważne jest aby takie algorytmy działały wydajnie i możliwie bezbłędnie. Może to mieć duże znaczenie np. w kontekście autonomicznych pojazdów, które już teraz zaczynają jeździć po naszych ulicach.

W niniejszym projekcie postanowiliśmy zbadać wydajność (czas działania) oraz błędogenność następujących algorytmów wykrywania krawędzi:

- Krzyż Robertsa
- Sobel
- Canny
- Filtr Laplace'a

Działanie powyższych algorytmów przetestujemy na pewnych szczególnych typach zdjęć:

- zdjęcie krajobrazu robione nocą
- zdjęcie miasta (dużo obiektów) robione za dnia

Powyższe sytuacje są wobec siebie w dużym stopniu komplementarne i powinny wyłonić najlepszy algorytm wykrywania krawędzi w ogólnym zastosowaniu.

Porównanie czasowe będzie polegało na sprawdzeniu jaka liczba milisekund minęła pomiędzy uruchomieniem, a zakończeniem danego algorytmu. Niestety sytuacja benchmarku nie jest oczywista w kontekście sprawdzenia błędogenności algorytmu – w tym przypadku będziemy skuteczność oceniać „na oko".

2. Opis algorytmu

- Krzyż Robertsa

Jeden z najwcześniejszych algorytmów wykrywania krawędzi (1963r.).

Zasada działania:

1. Obliczenie różnic luminacji pikseli położonych koło siebie po przekątnych. Działanie to można zapisać jako splot macierzy obrazu z każdą z przedstawionych poniżej macierzy rozmiaru 2x2:

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

2. Dodanie wartości bezwzględnych wartości z punktu 1

Wynikiem tych operacji są dwie macierze przedstawiające pochodne kierunkowe obliczone dla kierunków 135° oraz 45°. Wynikowy obraz krawędzi powstaje po obliczeniu różnic modułów z odpowiadających sobie elementów macierzy.

Krzyż Robertsa jest cały czas w użyciu ze względu na szybkość obliczeń. Przy porównaniu z późniejszymi algorytmami do wykrywania krawędzi (m.in. Sobel, Canny) wykazuje mniejszą odporność na szumy, daje jednak węższe krawędzie.

- ## Sobel

Operator dyskretnego różniczkowania, umożliwiający aproksymację pochodnych kierunkowych intensywności obrazu w ośmiu kierunkach co 45°.

Wyznaczenie pochodnej kierunkowej odbywa się za pomocą operacji dwuwymiarowego dyskretnego splotu macierzy obrazu z macierzą 3x3 charakterystyczną dla danego kierunku zwaną jądrem (kernel) przekształcenia. Macierze te są antysymetryczne w stosunku do kierunku wykrywanej krawędzi. Zbiór 8 macierzy pozwala na określenie kierunku od 0° do 315° z krokiem 45°. Dla kierunku 0° wykrywane są krawędzie pionowe, a dla 90° – krawędzie poziome. Operacja splotu wyznacza w pierwszym przypadku estymatę pochodnej cząstkowej względem osi X, a drugim względem osi Y.

Operator Sobela dokonuje operacji uśredniania pochodnej (z wagami 1, 2, 1) z trzech linii równoległych do kierunku różniczkowania. Dzięki temu ma mniejszą wrażliwość na zakłócenia w obrazie (szum) niż Krzyż Robertsa.

- ## Canny

Metoda została opracowana przez Johna F. Canny w 1986 roku. Wykorzystuje wielostopniowy algorytm w celu detekcji wielu różnych krawędzi w obrazie.

Algorytm składa się z następujących etapów:

1. Redukcja szumu

Ponieważ algorytm jest czuły na obecność szumu w surowym nieobrobionym obrazie, należy dokonać splotu z filtrem Gaussa. Efektem tego zabiegu będzie lekko rozmazany obraz, który nie będzie zawierał pojedynczych zakłóceń.

2. Szukanie natężenia gradientu obrazu

Po redukcji szumu przystępujemy do wyznaczenia nachylenia krawędzi oraz jej kierunku. Wymienione wartości mogą być wyznaczone z następujących wzorów:

$$G = \sqrt{{G_x}^2 + {G_y}^2} \qquad \theta = \arctan\left(\frac{G_y}{G_x}\right)$$

gdzie G_x jest wartością pochodnej w kierunku poziomym natomiast G_y jest wartością pochodnej w kierunku pionowym.

Kąt detekcji krawędzi zaokrąglony jest do czterech przypadków reprezentujących pion, poziom oraz dwóch przekątnych (np. 0, 45, 90 i 135 stopni).

3. Usuwanie niemaksymalnych pikseli

Kolejny etap obejmuje "pocienianie" krawędzi w sposób zapewniający ich ciągłość. Efektem jest ciągła linia złożona z pojedynczych pikseli.

4. Progowanie z histerezą

Progowanie ma na celu usunięcie nieistotnych krawędzi, które mają nachylenie (stromość) poniżej ustawionego progu. Progowanie z histerezą powoduje, że do już wykrytych krawędzi są dołączane następne piksele mimo spadku nachylenia, aż do osiągnięcia dolnego progu wykrywania. Takie postępowanie zapobiega dzieleniu krawędzi w miejscach słabszego kontrastu.

- Filtr Laplace’a

Zastosowanie filtrów w przetwarzaniu obrazów oznacza, że do obliczenia nowej wartości punktu brane są pod uwagę wartości punktów z jego otoczenia. Każdy piksel z otoczenia wnosi swój wkład - wagę podczas przeprowadzania obliczeń.

Filtry Laplace’a:

1. LAPL1

0	-1	0
-1	4	-1
0	-1	0

2. LAPL2

-1	-1	-1
-1	8	-1
-1	-1	-1

3. LAPL3

1	-2	1
-2	4	-2
1	-2	1

4. Laplace’a ukośny

-1	0	-1
0	4	0
-1	0	-1

5. Laplace’a poziomy – ograniczony do wykrywania krawędzi poziomych

0	-1	0
0	2	0
0	-1	0

6. Laplace’a pionowy – ograniczony do wykrywania krawędzi pionowych

0	0	0
-1	2	-1
0	0	0

3. Technologia

Implementacja zostanie wykonana w programie Visual Studio 2015 z wykorzystaniem środowiska .NET oraz języka C++/C#. Dokładny wybór języka podejmiemy podczas początków implementacji kodu.

Stworzony program będzie aplikacją konsolową, która jako wejście będzie przyjmowała ścieżkę do folderu ze zdjęciami do przetworzenia, a wynikiem będzie:

- Zbiór obrazków przetworzonych każdym z ww. algorytmów (4x liczba obrazków na wejściu)
- Dokument tekstowy zawierający zapis przetwarzania wykonania programu – w tym także czasy wykonania poszczególnych algorytmów dla każdego z obrazków wejściowych