POZNAN UNIVERSITY OF TECHNOLOGY ACADEMIC JOURNALS
No 96 Electrical Engineering 2018

DOI 10.21008/j.1897-0737.2018.96.0021

Paweł KOWALSKI*, Maciej CZYŻAK*

ALGORYTMY WYKRYWANIA KRAWĘDZI W OBRAZIE

Wykrywanie krawędzi jest pierwszym etapem w cyfrowym przetwarzaniu obrazów. Operacja ta polega na usunięciu informacji takich jak kolor czy też jasność, a pozostawieniu jedynie krawędzi. Efektem tej operacji jest znaczna redukcja ilości danych do dalszej analizy. Pozwala to na zastosowanie w następnych etapach przetwarzania bardziej złożonych algorytmów rozpoznawania obiektów na podstawie kształtu. W artykule zaprezentowano zastosowanie algorytmów Robertsa, Sobela, Previtt, Kirscha i Scharra. Zaproponowano też nowy, efektywny obliczeniowo algorytm dedykowany do wykrywania krawędzi poziomych. Algorytmy zostały porównane w zastosowaniu do detekcji przewodów. Kryteriami porównania były skuteczność wykrywania krawędzi przewodów wysokiego napięcia oraz szybkość działania. Algorytmy zostały zaimplementowane z wykorzystaniem biblioteki OpenCV oraz przetestowane na zestawie zdjęć przedstawiających przewody wysokiego napięcia.

SŁOWA KLUCZOWE: wykrywanie krawędzi, operatory detekcji krawędzi, przetwarzanie obrazu, wykrywanie linii wysokiego napięcia.

1. WSTĘP

Obraz cyfrowy jest źródłem wielkiej ilości informacji. Wraz ze wzrostem częstotliwości odświeżania obrazu oraz rozdzielczości, obróbka obrazu staje się coraz trudniejsza. W aplikacjach działających w czasie rzeczywistym podczas analizy obrazu konieczna jest szybka redukcja rozmiaru obrabianych danych na etapie wstępnej analizy. Generalnie proces analizy obrazu można podzielić na trzy główne etapy:

1 – wstępna obróbka obrazu (redukcja szumów, eliminacja zakłóceń, konwersja do odcieni szarości oraz detekcja krawędzi),
2 – lokalizacja obiektów w obrazie oraz rozpoznawanie kształtów,
3 – analiza sceny złożonej z obiektów.

Metody wykrywania przewodów wysokiego napięcia były rozpatrywane m.in. w [1–5]. Działają one zwykle na drugim lub trzecim etapie i opierają swe działanie na wykrytych wcześniej krawędziach. W pracy przedstawiony zostanie główny element pierwszego etapu obróbki obrazu (etap 1) obejmujący wykrycie krawędzi w obrazie. Przed wykrywaniem krawędzi usuwane są kolory z obrazu

* Politechnika Gdańska

poprzez proste przeliczenie z formatu RGB na jedną wartość liczbową reprezentującą poziom jasności. Poszukuje się algorytmów wykrywania krawędzi wymagających możliwie najmniejszego nakładu obliczeniowego, co umożliwia ich realizację w czasie rzeczywistym. W pracy przedstawiono przegląd istniejących algorytmów wykrywania krawędzi oraz zaproponowano nowy algorytm. Istotnym parametrem z obliczeniowego punktu widzenia jest rozmiar okna, czyli obszaru uwzględnianego podczas pojedynczego kroku obliczeniowego. Zaproponowany algorytm stosuje do wykrywania krawędzi mniejsze okno niż inne algorytmy, co znacznie zmniejsza nakład obliczeniowy przy zachowaniu podobnej jakości. Jest on pewną specjalizowaną wersją ukierunkowaną na wykrywanie w obrazie przewodów wysokiego napięcia. W rozdziale 2 przedstawiono przegląd algorytmów detekcji krawędzi, w rozdziale 3 nowy algorytm wykrywania krawędzi, a w rozdziale 4 porównanie algorytmów.

2. PRZEGLĄD ALGORYTMÓW DETEKCJI KRAWĘDZI

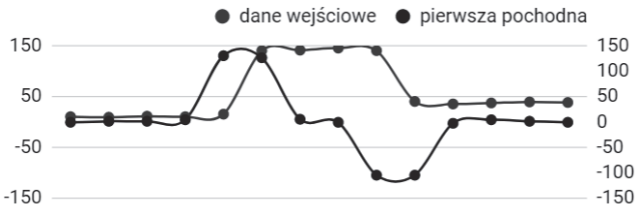
Detekcja krawędzi jest jednym z elementów pierwszego etapu analizy obrazu. Po operacji tej wynikowy obraz poddawany jest głębszej analizie na wyższym poziomie. Pierwotny obraz kolorowy, w którym każdy piksel jest reprezentowany w RGB, wymagałby działania na trzech liczbach dla każdego piksela. W celu zmniejszenia nakładu obliczeniowego, do opisu piksela stosowana jest tylko jedna liczba reprezentująca intensywność szarości, co daje w wyniku obraz monochromatyczny. Pojedynczy piksel przechowuje informację 8-bitową. Wynikiem detekcji krawędzi jest obraz, w którym piksele reprezentujące krawędzie mają stosunkowo duże wartości, a pozostałe wartości bliskie 0. Końcowym wynikiem powinien być obraz binarny reprezentowany przez tablicę bitów, w której każdy bit odpowiada jednemu pikselowi, gdzie 1 oznacza krawędź, a 0 brak krawędzi.

Część przykładowego wiersza z danymi przedstawiono poniżej w tabeli 1. Przyjmuje się, że krawędź znajduje się w miejscu gwałtownej zmiany wartości. W przykładzie wyróżniono takie miejsca.

Tabela 1. Przykładowy wiersz z danymi.

10	9	11	10	15	140	141	145	140	40	35	37	39	38

Przy takiej reprezentacji, wartości pikseli można traktować jako kolejne wartości funkcji. Jeżeli zostanie dokonana interpolacja przebiegu, następnie obliczona zostanie pierwsza pochodna funkcji interpolującej, można będzie zauważyć, że lokalne ekstrema funkcji wskazują miejsca występowania krawędzi (rys. 1).



Rys. 1. Przebiegi funkcji interpolującej utworzonej na podstawie pikseli obrazu oraz jej pierwszej pochodnej

Poniżej zostaną przedstawione wybrane algorytmy wykrywania krawędzi. Swoje działanie opierają one na splocie okna z odpowiednią maską, co oznacza dwuwymiarową cyfrową filtrację obszaru okna. Algorytmy zostały zaimplementowane oraz przetestowane z wykorzystaniem biblioteki OpenCV [6] oraz przykładowego obrazu przekonwertowanego do skali odcieni szarości.

2.1. Operator krzyżowy Robertsa

Jako pierwszy z algorytmów wykrywania krawędzi zostanie rozpatrzony operator krzyżowy Robertsa [7]. Algorytm wykorzystuje okno o rozmiarze 2x2 piksele. Na podstawie wartości pikseli w oknie, wyliczana jest nowa wartość piksela zapisywana w tablicy reprezentującej obraz wynikowy. W [7] wzór stosowany do obliczania nowej wartości piksela ma postać:

$$y_{i,j} = \sqrt{x_{i,j}} \tag{1a}$$

$$z_{i,j} = \sqrt{(y_{i,j} - y_{i+1,j+1})^2 + (y_{i+1,j} - y_{i,j+1})^2} \tag{1b}$$

gdzie $x_{i,j}$ jest wartością piksela obrazu wejściowego, a $z_{i,j}$ jest wartością piksela dla obrazu wynikowego. Oryginalny algorytm zawiera czasochłonne operacje pierwiastkowania oraz potęgowania. Podobną skuteczność oraz znaczne uproszczenie obliczeń można osiągnąć poprzez modyfikację algorytmu z wykorzystaniem wartości bezwzględnej zamiast pierwiastkowania, co daje następującą zależność:

$$z_{i,j} = \left|W_{i,j} * R_1\right| + \left|W_{i,j} * R_2\right| \tag{2}$$

gdzie $z_{i,j}$ jest pikselem obrazu wynikowego, $W_{i,j}$ macierzą reprezentującą okno pobrane z obrazu wejściowego:

$$W_{i,j} = \begin{bmatrix} x_{i,j} & x_{i+1,j} \\ x_{i,j+1} & x_{i+1,j+1} \end{bmatrix} \tag{3}$$

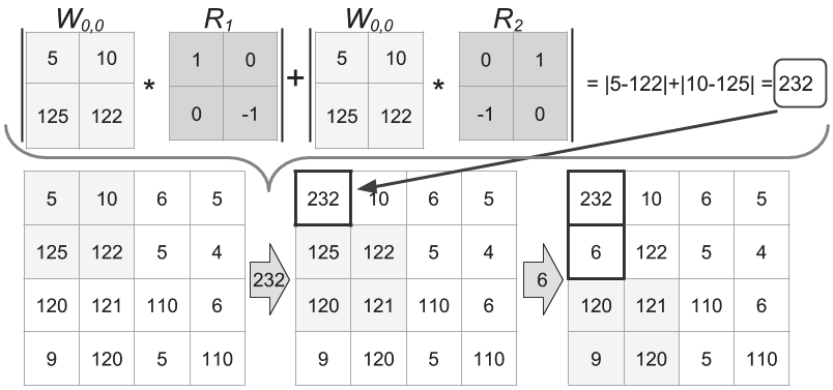
a R_1 oraz R_2 są maskami o następującej postaci:

$$R_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad R_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{4}$$

Finalnie otrzymujemy:

$$z_{i,j} = \left| x_{i,j} - x_{i+1,j+1} \right| + \left| x_{i+1,j} - x_{i,j+1} \right| \tag{5}$$

Na rysunku 2 przedstawiono przykład działania algorytmu Robertsa, z użyciem zależności (2) i (5).



Rys. 2. Graficzna reprezentacja wykrywania krawędzi przy użyciu operatora krzyżowego Robertsa

Przedstawiony algorytm dokonuje zmian istniejącego obrazu bez tworzenia nowego, co pozwala na ograniczenie rozmiaru niezbędnej pamięci. W pierwszym kroku przeprowadzane są obliczania dla fragmentu obrazu reprezentowanego przez okno $W_{0,0}$ o rozmiarze 2 x 2 piksele. Następnie obliczany jest splot okna $W_{0,0}$ z maskami R_1 i R_2. Wynikiem obliczeń dla okna $W_{0,0}$ jest wartość 232. Liczba ta zastępuje poprzednią wartość piksela z lewego górnego rogu ($x_{0,0}$). Wysoka wartość oznacza wykrycie krawędzi. W kolejnym kroku okno jest przesuwane o jeden piksel w dół i ponownie obliczany jest splot z maskami R_1 i R_2. W tym przypadku wynikiem jest liczba 6 (rys. 2). Mała wartość oznacza brak krawędzi. Operacje te wykonywane są kolejno dla wszystkich pikseli obrazu wejściowego. Poniżej przedstawiono przykładową implementację algorytmu Robertsa w postaci funkcji *Roberts_apply*.

```
void Roberts_apply(Mat img){
 int R1, R2;
 for (int x=0; x<img.cols-1; x++)
  for (int y=0; y<img.rows-1; y++){
   R1 = img.at<unsigned char>(y, x) - img.at<unsigned char>(y+1, x+1);
   R2 = img.at<unsigned char>(y, x+1) - img.at<unsigned char>(y+1, x);
   img.at<unsigned char>(y, x) = abs(R1)+abs(R2); } }
```

Funkcja jako parametr formalny przyjmuje dwuwymiarową macierz pikseli *img*. *Mat* jest wbudowanym typem biblioteki OpenCV, umożliwiającym reprezentację obrazu w postaci macierzy. Działanie funkcji polega na wykonaniu zależności (5) dla każdego piksela z wyjątkiem ostatniego wiersza i ostatniej kolumny. W wyniku działania funkcji do macierzy *img* zostaną zapisane wartości wykrytych krawędzi.

2.2. Operator Sobela

Operator Sobela został przedstawiony w [8]. Wykorzystuje on okno o rozmiarze 3x3 piksele oraz dwie maski G_1 oraz G_2, o następującej postaci:

$$G_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{6}$$

Wykrywanie krawędzi przebiega podobnie jak w algorytmie Robertsa. Różnica polega na wielkości okna. W przypadku zastosowania masek (6) obliczenia wykonywane są na podstawie okna o rozmiarze 3x3. Algorytm ten do obliczenia jednego piksela wynikowego $z_{i,j}$ stosuje następującą zależność:

$$\begin{aligned} z_{i,j} = &\left| -x_{i,j} + x_{i+2,j} - 2(x_{i,j+1} - x_{i+2,j+1}) - x_{i,j+2} + x_{i+2,j+2} \right| + \\ &\left| x_{i,j} + x_{i+2,j} + 2(x_{i+1,j} - x_{i+1,j+2}) - x_{i,j+2} - x_{i+2,j+2} \right| \end{aligned} \tag{7}$$

gdzie $x_{i,j}$ jest wartością piksela obrazu wejściowego. W celu zachowania skali szarości identycznej z obrazem wejściowym, wyliczoną wartość $z_{i,j}$ należy przed zapisem do pamięci podzielić przez 4 [8]. Operację tę można szybko wykonać stosując przesunięcie w prawo o dwa bity.

2.3. Operator Prewitt

Operator Prewitt [9] wykorzystuje okno o rozmiarze 3x3 piksele. Jego działanie różni się od operatora Sobela jedynie zastosowanymi maskami P_1 i P_2:

$$P_1 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad P_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{8}$$

W przypadku operatora Prewitt, aby zachować skalę szarości identyczną z obrazem wejściowym, wynik splotu należy podzielić przez 3.

2.4. Operator kompasowy

Operator ten również opiera swe działania na splocie okna z odpowiednimi maskami. Jednak w tym przypadku zestaw masek jest generowany na podstawie maski podstawowej poprzez obrót o wybrany kąt. Jednym z operatorów kompasowych jest operator Kirscha [10]. Podobnie jak operatory Sobela i Previtt, stosuje on maski 3x3 generowane na podstawie maski głównej K_1 poprzez jej cykliczny obrót o 45°. W ten sposób powstaje zestaw 8 masek kierunkowych przedstawionych poniżej:

$$K_1=\begin{bmatrix}5&5&5\\-3&0&-3\\-3&-3&-3\end{bmatrix},\ K_2=\begin{bmatrix}-3&5&5\\-3&0&5\\-3&-3&-3\end{bmatrix},\ K_3=\begin{bmatrix}-3&-3&5\\-3&0&5\\-3&-3&5\end{bmatrix}$$
$$K_4=\begin{bmatrix}-3&-3&-3\\-3&0&5\\-3&5&5\end{bmatrix},\ K_5=\begin{bmatrix}-3&-3&-3\\-3&0&-3\\5&5&5\end{bmatrix},\ K_6=\begin{bmatrix}-3&-3&-3\\5&0&-3\\5&5&-3\end{bmatrix} \tag{9}$$
$$K_7=\begin{bmatrix}5&-3&-3\\5&0&-3\\5&-3&-3\end{bmatrix},\ K_8=\begin{bmatrix}5&5&-3\\5&0&-3\\-3&-3&-3\end{bmatrix}$$

Wykrywanie krawędzi polega na obliczeniu splotu z kolejnymi maskami kierunkowymi K_n:

$$z_{i,j} = max(W_{i,j}*K_n),\ \text{dla } n=1, 2\ldots8 \tag{10}$$

Wynikiem obliczeń dla poszczególnych pikseli jest maksymalna wartość splotu (10). Innym przykładem operatora kompasowego jest operator Scharra [11].

Operator działa analogicznie do operatora Kirscha, zmianie ulegają jedynie maski. Wykorzystuje on 8 następujących masek powstałych w wyniku obracania maski S_1 o 45°:

$$S_1=\begin{bmatrix}-3&0&3\\-10&0&10\\-3&0&3\end{bmatrix},\ S_2=\begin{bmatrix}-10&-3&0\\-3&0&3\\0&3&10\end{bmatrix},\ S_3=\begin{bmatrix}-3&-10&-3\\0&0&0\\3&10&3\end{bmatrix}$$
$$S_4=\begin{bmatrix}0&-3&-10\\3&0&-3\\10&3&0\end{bmatrix},\ S_5=\begin{bmatrix}3&0&-3\\10&0&-10\\3&0&-3\end{bmatrix},\ S_6=\begin{bmatrix}10&3&0\\3&0&-3\\0&-3&-10\end{bmatrix} \tag{11}$$

$$S_7 = \begin{bmatrix} 3 & 10 & 3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix},\ S_8 = \begin{bmatrix} 0 & 3 & 10 \\ -3 & 0 & 3 \\ -10 & -3 & 0 \end{bmatrix}$$

3. NOWY ALGORYTM WYKRYWANIA KRAWĘDZI

Głównym celem pracy było znalezienie algorytmu szybszego od algorytmów przedstawionych w p.2 oraz lepiej przystosowanego do wykrywania krzywych reprezentujących przewody w obrazie. Spośród algorytmów rozważanych w rozdziale 2, najszybszy jest algorytm Robertsa. Wymaga on najmniejszej ilości obliczeń na jeden piksel obrazu wynikowego, tj. czterech pobrań wartości z pamięci, trzech operacji addytywnych oraz dwóch obliczeń wartości bezwzględnej.

Proponowany nowy algorytm do obliczenia nowej wartości piksela wykorzystuje 2 piksele i wymaga jednego odejmowania oraz obliczenia wartość bezwzględnej. Maska proponowanego algorytmu ma postać:

$$P_1 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{12}$$

Maska w (12) jest podobna do maski zastosowanej w [12] do wykrywania prostych. W obliczeniach z użyciem maski (12) wykorzystywane są jedynie dwa skrajne piksele okna, co daje następującą formułę obliczeniową:

$$z_{x,y} = \left| v_{x,y} - v_{x,y+2} \right| \tag{13}$$

Wynikiem splotu obrazu wejściowego z maską (12) jest różnica wartości dwóch pikseli $v_{x,y}$ i $v_{x,y+2}$ (13), gdzie x jest numerem kolumny, a y numerem wiersza. W macierzy reprezentującej obraz przechowywane są liczby całkowite nieujemne, więc niezbędne jest również usunięcie znaku otrzymanej liczby poprzez użycie wartości bezwzględnej. Pewną wadą algorytmu jest to, że ze względu na swoje przeznaczenie do wykrywania krawędzi poziomych, nie jest efektywny dla krawędzi pionowych. Jednak przy wykrywaniu krawędzi poziomych jest równie skuteczny jak pozostałe algorytmy. W tabeli 2 zawierającej porównanie algorytmów, nowy algorytm oznaczono symbolem „1x3”.

Opracowano również drugą wersję algorytmu wykrywającego krawędzie z wykorzystaniem okna 1x2 i następującej formule obliczeniowej:

$$z_{i,j} = \left| v_{x,y} - v_{x,y+1} \right|. \tag{14}$$

Zależność (14) jest faktycznie splotem z maską M_1 o postaci:

$$M_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \tag{15}$$

W tabeli 2, algorytm ten oznaczono jako „1x2". Czasy wykonania algorytmów „1x2" i „1x3" są te same. Podobny algorytm użyto w [12] do wykrywania prostych.

Efektem działania każdego z algorytmów przedstawionych w p.2 i p.3 jest macierz wyjściowa wypełniona wartościami naturalnymi. Aby jednoznacznie określić miejsce wykrycia krawędzi, macierz ta zamieniana jest na binarną. W tym celu przeprowadzane jest progowanie o następującej formie:

$$v^{+}_{x,y} = \begin{cases} 1 & if \quad v_{x,y} > T \\ 0 & if \quad v_{x,y} \le T \end{cases} \tag{16}$$

gdzie $v_{x,y}$ oznacza piksel macierzy przed progowaniem, a $v^{+}_{x,y}$ piksel po progowaniu, T reprezentuje wartość progową. Piksele o wartościach większych niż wartość progowa T są uznawane za krawędź i przypisuje się im wartość 1, zaś pozostałym wartość 0, co oznacza brak krawędzi.

4. PORÓWNANIE ALGORYTMÓW WYKRYWANIA KRAWĘDZI

Porównanie skuteczności algorytmów zostało wykonane z wykorzystaniem zdjęć przewodów elektrycznych. Algorytmy zostały zaimplementowane programowo. Dla realizacji porównania została opracowana zależność określająca skuteczność wykrywania krawędzi o następującej postaci:

$$eff = \sum_{i=1}^{n} \sum_{x=0}^{img\_width-1} f_i(x) \tag{17}$$

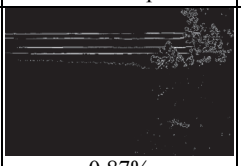
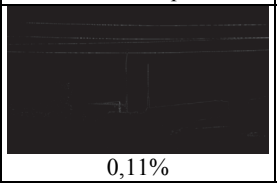
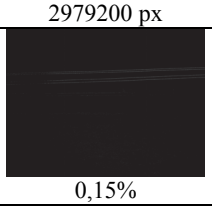
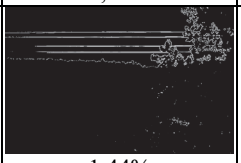
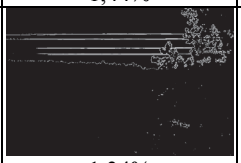
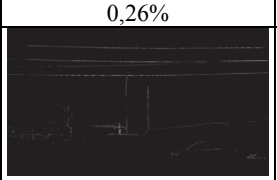
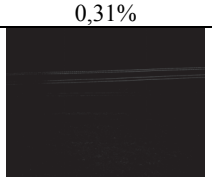
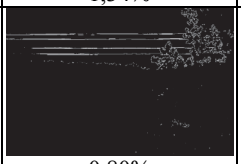
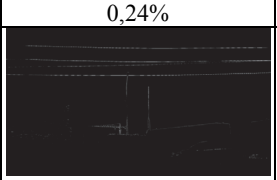
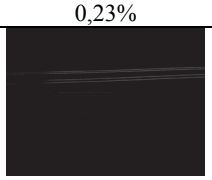
Zależność (17) jest sumarycznym wskaźnikiem efektywności metody *eff*, gdzie n jest liczbą widocznych przewodów, a *img_width* szerokością obrazu w pikselach. Ponadto $f_i(x)$ z (17) ma postać:

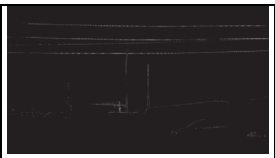
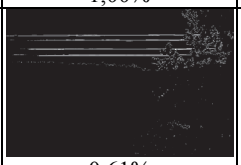
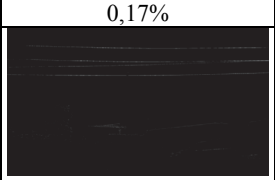
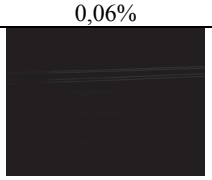
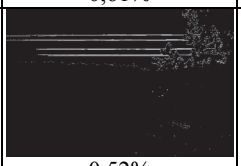
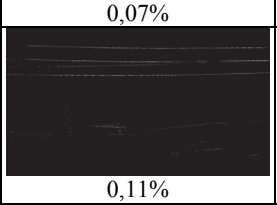
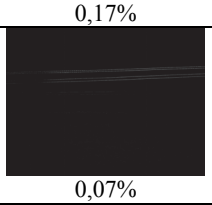
$$f_i(x) = \begin{cases} 1 & if \quad \exists v_{x,y} \in W_{i,x} \\ 0 & if \quad \forall v_{x,y} \notin W_{i,x} \end{cases} \tag{18}$$

Funkcja (18) jest funkcją sprawdzającą skuteczność wykrycia i-tego przewodu w kolumnie x, gdzie $W_{i,x}$ oznacza zbiór pikseli reprezentujących przewód i-ty w kolumnie pikseli x, a $v_{x,y}$ oznacza piksel binarnego obrazu wejściowego o współrzędnych (x, y). Miarą skuteczności wykrywania i-tego przewodu jest liczba kolumn w których został on wykryty (wewnętrzna suma w (17)). Ogólna miara skuteczności *eff* jest wyznaczana jako suma miar dla wszystkich przewodów. Bardzo istotnym czynnikiem wpływającym na efektywność algorytmów jest wartość progu, która powinna być dobrana w sposób optymalny. Dla po-

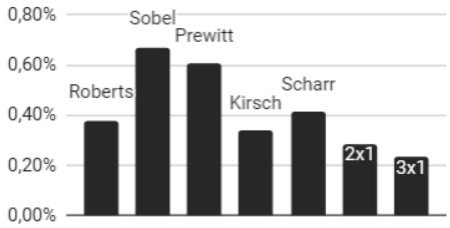
równania poszczególnych algorytmów kluczowym jest optymalny dobór progu *T*. W tym celu dla każdego algorytmu przeprowadzono serię prób progowania z różną wartością *T*. Próg *T* został dobrany indywidualnie w taki sposób, aby był możliwie największy, a jednocześnie kryterium *eff* dawało 90% maksymalnej wartości. Skuteczność algorytmów została porównana poprzez zliczenie ilości pikseli wykrytych poza krawędziami przewodów (niepoprawnie wykryte krawędzie). Tabela 2 oraz rys. 3 przedstawiają wyniki porównania wielkości generowanego szumu przez poszczególne algorytmy. Szum jest stosunkiem liczby pikseli z niepoprawnie wykrytymi krawędziami do liczby wszystkich pikseli obrazu.

Tabela 2. Porównanie wielkości generowanego szumu przez algorytmy wykrywania krawędzi

Obraz w skali szarości, rozmiar	 650000 px	 9734400 px	 2979200 px	Średni szum
Roberts	 0,87%	 0,11%	 0,15%	0,38%
Sobel	 1,44%	 0,26%	 0,31%	0,67%
Previtt	 1,34%	 0,24%	 0,23%	0,60%
Kirch	 0,80%	 0,16%	 0,05%	0,34%

Scharr	 1,00%	 0,17%	 0,06%	0,41%
1x2	 0,61%	 0,07%	 0,17%	0,29%
1x3	 0,52%	 0,11%	 0,07%	0,23%

W tabeli 2 przedstawiono wyniki przeprowadzonego eksperymentu porównawczego z wykorzystaniem trzech zdjęć. Pierwszy wiersz zawiera obrazy wejściowe w skali szarości. Obrazy te posłużyły do porównania algorytmów wykrywania krawędzi. Każdy z następnych wierszy zawiera nazwę algorytmu oraz wynik działania algorytmu po progowaniu wraz z wartością wygenerowanego szumu. Ostatnia kolumna zawiera średnią wartość generowanego szumu. Dwa ostatnie algorytmy generują mniejszy szum, gdy obraz wejściowy zawiera krawędzie pionowe. Na rys. 3 zaprezentowano średnie wartości szumu generowane przez poszczególne algorytmy.



Rys. 3. Średnia wielkość szumu generowanego przez poszczególne algorytmy wykrywania krawędzi

Jak można zauważyć, przy optymalnym progowaniu najmniejszy szum generowany jest przez algorytm 3x1, potwierdza to skuteczność zaproponowanego algorytmu do wykrywania przewodów elektrycznych.

5. PODSUMOWANIE

W pracy zaproponowano nowy algorytm wykrywania krawędzi w obrazie wykorzystywany do wykrywania przewodów elektrycznych. Algorytm ten jest dostosowany do wykrywania krawędzi poziomych. Zapewnia on podobną skuteczność wykrywania krawędzi, jak znane algorytmy: Robertsa, Sobela, Previtt, Kirscha i Scharra. Potrzeba opracowania takiego algorytmu wynikła z konieczności redukcji nakładu obliczeniowego ze względu na planowane jego wykorzystanie do pracy w czasie rzeczywistym na platformie mobilnej. Przedstawiono także wyniki działania poszczególnych algorytmów z zastosowaniem indywidualnego optymalnego progowania dla każdego z nich.

LITERATURA

[1] Yetgin O. E., Gerek O. N., PLD: Power line detection system for aircrafts, International Artificial Intelligence and Data Processing Symposium (IDAP) Artificial Intelligence and Data Processing Symposium (IDAP), Sep 2017.
[2] Guang Z., Jinwei Y., I-Ling Y., Farokh B., Robust Real-Time UAV Based Power Line Detection and Tracking, IEEE International Conference on Image Processing (ICIP), Sep, 2016, pp. 744-748.
[3] Candamo J., Kasturi R., Goldgof D., Sarkar S., Detection of Thin Lines using Low-Quality Video from Low-Altitude Aircraft in Urban Settings, IEEE Transactions on Aerospace and Electronic Systems, Volume 45, Number 3, July 2009, pp. 937-949.
[4] Karakose, E., Performance evaluation of electrical transmission line detection and tracking algorithms based on image processing using UAV, International Artificial Intelligence and Data Processing Symposium (IDAP), 1-5 Sep, 2017.
[5] Weiran C, Linlin Z., Jianda H., Tianran W., Yingkui D., High voltage transmission line detection for uav based routing inspection, IEEE/ASME International Conference on Advanced Intelligent Mechatronics Advanced Intelligent Mechatronics (AIM), July 2013, pp. 554-558.
[6] Open Source Computer Vision Library, Reference Manual, 2014.
[7] Roberts L. G.: Machine perception of three-dimensional solids, PhD thesis, MIT, Lincoln Laboratory, May 22, 1963, pp. 82.
[8] Sobel I., Feldman G., An isotropic 3x3 image gradient operator, presented at Stanford Artificial Intelligence Project (SAIL), 1968.
[9] Prewitt J. M. S.,Object Enhancement and Extraction, Picture processing and Psychopictorics, Academic Press,1970.
[10] Kirsch R. A., Computer Determination of the Constituent Structure of Biological Images, Computers and Biomedical Research, Volume 4, Number 3, June 1971, pp. 315-328.
[11] Scharr H., Optimal Operators in Digital Image Processing, PhD thesis, Heidelberg, May 10, 2000, pp. 178.

[12] Burns J. B., Hanson A. R., Riseman E. M., Extracting Straight Lines, IEEE Transactions on Pattern Analysis & Machine Intelligence, Volume 8, Number 4, 1986, pp. 425-455.

EDGE DETECTION ALGORITHMS IN PICTURES

Edge detection is the first step in digital image processing. This operation involves removing information such as colour or brightness and leaving edges. This data size reduction makes that the data amount for the further analysis is significantly smaller. This allows to use more complex algorithms for recognizing objects based on shape in the next processing stages. The work presents the application of the known edge detection algorithms as these of Roberts, Sobel, Prewitt, Kirch and Scharr. Moreover, a new, computationally effective, edge detection algorithm dedicated for horizontal edges is proposed. The algorithms have been compared for detection of electric wires. The criterions of comparison were the effectiveness of edge detection applied to high voltage wires and the speed of operation. The algorithms have been implemented using the OpenCV library and tested on a set of images of high voltage wires.

(Received: 06.02.2018, revised: 22.03.2018)