Janusz Cieślar[*]

IMPLEMENTACJA I TESTOWANIE PROCEDURY AUTOMATYCZNEGO ŚLEDZENIA KRAWĘDZI NA OBRAZACH CYFROWYCH[**]

1. Wstęp

W 1999 roku w zakładzie Fotogrametrii i Informatyki Teledetekcyjnej AGH powstał pomysł realizacji projektu automatyzacji wyznaczania parametrów lin odciągowych z wykorzystaniem zdjęć cyfrowych. Obecnie zadania to jest realizowane w ramach projektu KBN. Warunkami początkowymi były założenia maksymalnego zautomatyzowania procesu pomiarowo–obliczeniowego, oraz (w celu uzyskania wysokiej dokładności) wykorzystania dużej liczby punktów (ok. kilku tysięcy). Aby spełnić oba powyższe założenia należało opracować i przetestować algorytmy automatycznego śledzenia linii na obrazach cyfrowych pod kątem znajdowania przebiegu obrazów lin.

Celem artykułu jest pokazanie sposobu działania, testowanie (dla różnych danych wejściowych) oraz ocena algorytmu śledzenia krawędzi metodą heurystyczną, zaimplementowaną na grafie skierowanym mającym postać drzewa. Algorytm został zaadaptowany na podstawie [1], został on tylko nieznacznie zmodyfikowany w celu lepszego działania i stworzenia możliwości kontroli. Test algorytmu został przeprowadzony pod kątem jego przydatności dla potrzeb systemu automatyzacji wyznaczania parametrów lin odciągowych wież wiertniczych wykonywanego w ramach projektu KBN. Niniejszy artykuł jest pewną kontynuacją badań prowadzonych w tym zakresie. Wyniki dotychczasowych badań (opis działania i testowanie innych procedur śledzenia linii) zostały zamieszczone w [2].

2. Założenia algorytmu

Opisywany algorytm wymaga określenia na obrazie dwóch punktów: początkowego x_A i końcowego x_B krawędzi, która ma zostać zlokalizowana. Zlokalizowanie krawędzi polega

[*] Akademia Górniczo-Hutnicza, Wydział Geodezji Górniczej i Inżynierii Środowiska

[**] Artykuł opracowano w ramach programu badań statutowych (nr 11.11.150.459) Zakładu Fotogrametrii i Informatyki Teledetekcyjnej AGH w 2001 roku

na określeniu współrzędnych (na obrazie) punktów, pomiędzy którymi jest ona zawarta. Przebieg krawędzi jest określany przez analizę a następnie wybór kolejnych pikseli, począwszy od piksela początkowego x_A, a skończywszy na pikselu końcowym x_B.
W celu zakwalifikowania danego piksela, jako kandydata do piksela krawędzi, musi on spełniać następujące warunki:

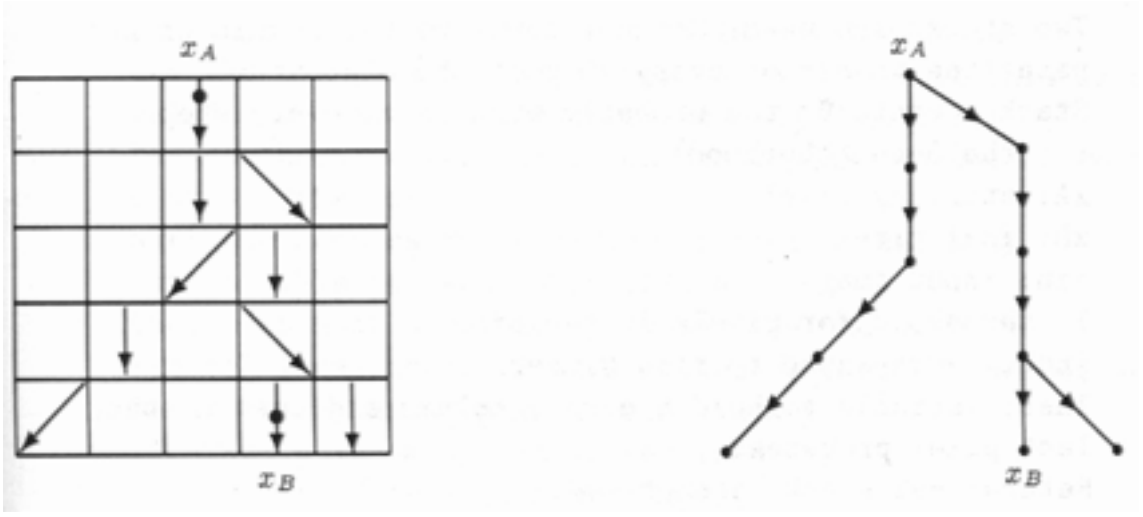
$$\left|e(x_i)\right| \geq TD \qquad \left|e(x_j)\right| \geq TD \tag{1}$$

$$\left|e(x_i)\right| \leq TG \qquad \left|e(x_j)\right| \leq TG \tag{2}$$

gdzie:
x_i, x_j są odpowiednio pozycjami piksela bieżącego i następnego, natomiast $e(x_i)$ i $e(x_j)$ są wartościami jasności piksela na tych pozycjach, TD jest pewnym zadanym progiem dolnym, a TH progiem górnym dla jasności piksela.

$$\left|\phi(x_i) - \phi(x_j)\right| \bmod 2\pi \leq T_2 \tag{3}$$

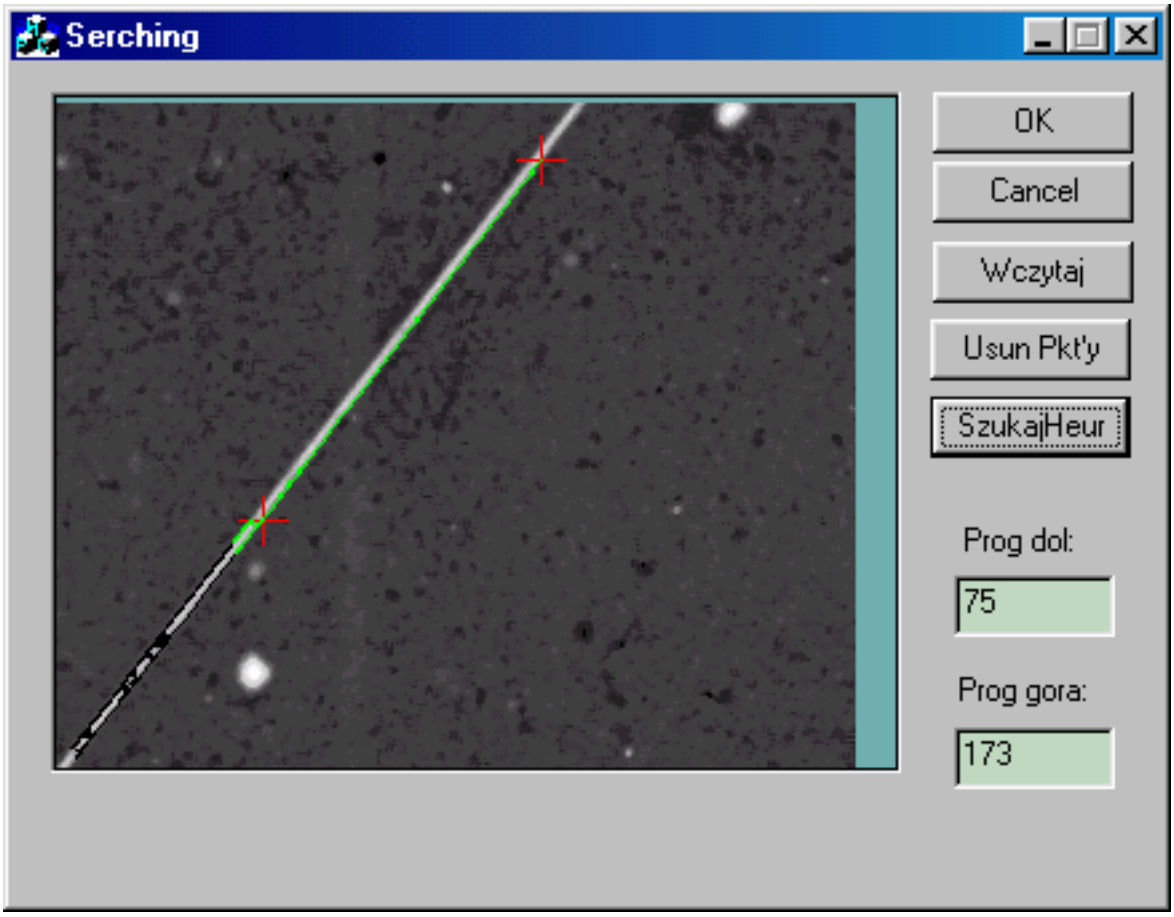
Warunki (1) i (2) powodują odrzucenie pikseli, których wartości jasności leżą poza zadanym zakresem. Warunek (3) bazuje na założeniu, że krzywizna lokalizowanej krawędzi jest mniejsza od zadanej. W zastosowanym algorytmie warunek (3) nie musi być spełniony w sposób dosłowny; oznacza to, że spośród pikseli kandydujących – spełniających warunek (1) (2) – w pierwszej kolejności jest wybierany piksel spełniający warunek (3), a jeżeli nie ma takiego piksela, to brane są pozostałe piksele wg. kolejności wyznaczonej przez atrybut kosztu. Każdy piksel kandydujący do krawędzi ma przydzielony atrybut kosztu. Atrybut kosztu przyjmuje bezwzględną wartość jasności danego piksela.



Rys. 1.
1a) przykładowe drogi, po jakich może poruszać się algorytm w trakcie działania,
1b) drzewo obrazujące te drogi przejścia algorytmu.

3. Działanie

Algorytm został opracowany z myślą o wykrywaniu krawędzi a nie linii, jednakże krawędź w rozumieniu niniejszego artykułu musi być rozumiana nieco inaczej niż w artykule [3], gdzie za krawędź uważa się granicę, prostopadle do której **gwałtownie** ulega zmianie pewna własność obrazu, jak jasność, kolor czy tekstura. Ponadto zakłada się, że po obu stronach krawędzi, sąsiadujące obszary są jednorodne z punktu widzenia analizowanej cechy. Działanie opisywanego algorytmu oparte jest na założeniu, że występuje **łagodna** zmiana własności obrazu. W związku z tym w przytoczonej definicji należało by słowo **gwałtownie** zamienić na **łagodnie**. Jednakże taka zmiana sprawia, że definicja przestaje być ścisła, w związku z czym należało by określić (w pikselach) na jakiej długości (w kierunku prostopadłej do krawędzi) ma miejsce to łagodne przejście. Warto również zaznaczyć, że krawędź liny odfotografowanej na obrazie cyfrowym ma zawsze łagodne przejście – zatem w rzeczywistości nasz warunek jest zawsze spełniony. Z praktycznych obserwacji zachowania się algorytmu dla różnych krawędzi ustalono, że *długość przejścia* powinna wynosić co najmniej 3 piksele, aby algorytm był w stanie działać poprawnie i nie powinna być zbyt duża np. mniejsza niż 6 pikseli, ponieważ zwiększanie *długości przejścia* zmniejsza dokładność określenia krawędzi przez algorytm. W przypadku zmniejszania *długości przejścia* zwiększa się wrażliwość algorytmu na dokładność usytuowania punktów początkowego x_A i końcowego x_B.

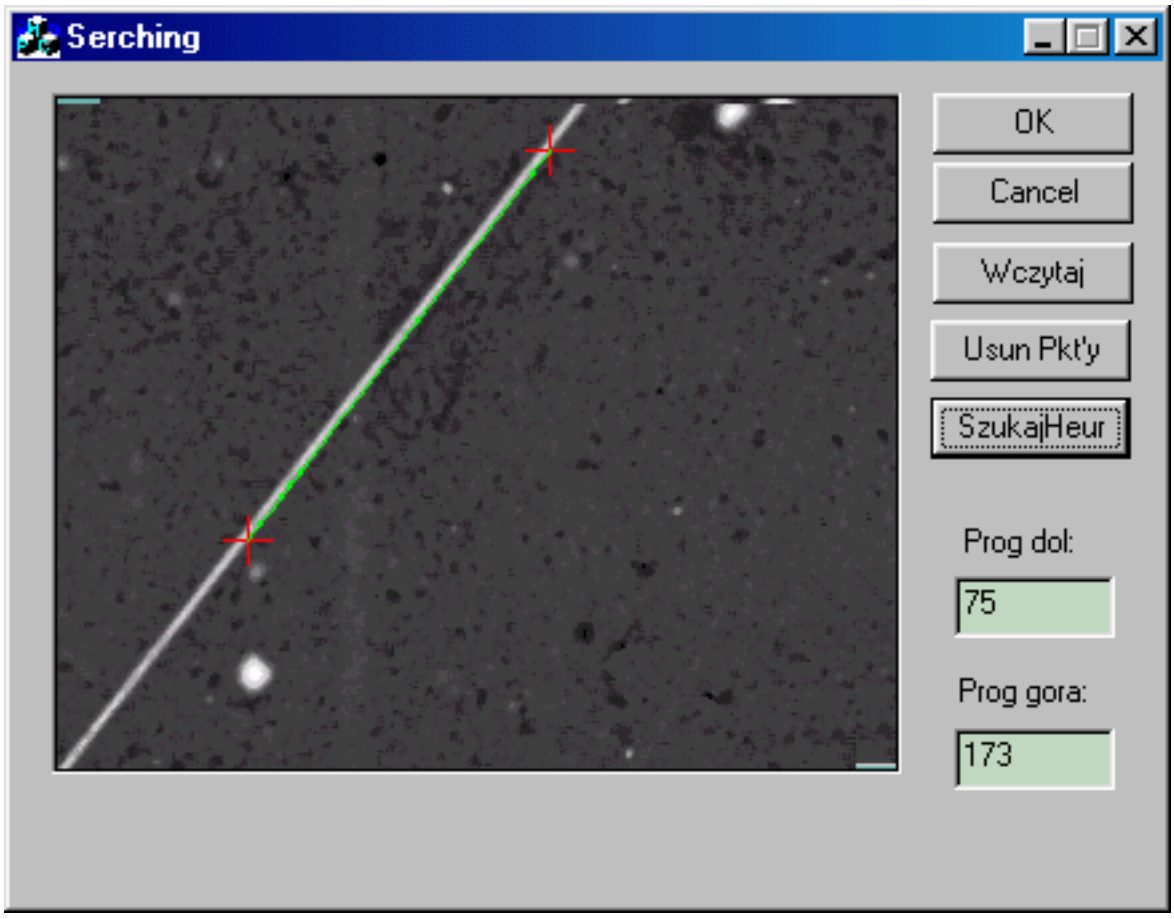


Rys. 2. Wpływ złego usytuowania punktów (początkowego i końcowego) na działanie algorytmu.

Przeszukiwanie odbywa się począwszy od piksela początkowego x_A i jest zakończone sukcesem (jeśli zostanie osiągnięty piksel końcowy x_B), lub porażką (jeśli nie osiągnięto piksela końcowego x_B). Algorytm na każdym etapie przeszukiwania dołącza do bieżącego łańcucha pikseli następny, wybierając najlepszy spośród pikseli kandydujących (zob. rozdział 2). Postępując w ten sposób albo zostanie osiągnięty piksel końcowy (zakończenie sukcesem) lub nastąpi wejście w „ślepą uliczkę" – nie będzie żadnych następnych pikseli spełniających wymagane kryteria (droga okazała się niewłaściwa). W przypadku „ślepej uliczki" algorytm wraca do najbliższego węzła, gdzie rozpoczyna przejście nową drogą zaczynając od tej, dla której koszt jest najmniejszy. Algorytm kończy działanie albo w przypadku gdy znalazł drogę do piksela końcowego, lub wówczas, gdy przeszukawszy wszystkie możliwe drogi, nie znalazł właściwej. Na rysunku 1a przedstawiono przykładowe drogi, po jakich może poruszać się algorytm w trakcie działania, natomiast na rysunku 1b przedstawiono drzewo obrazujące te drogi.

4. Testy

W celu zobrazowania działania algorytmu przeprowadzono szereg testów. Poniżej zamieszczono rezultaty działania programu dla różnych obrazów i różnych ustawień parametrów początkowych.



Rys. 3. Prawidłowo dobrane położenia punktów (początkowego x_A i końcowego x_B) oraz parametrów wejściowych (próg dolny, próg górny).

Zamieszczone rysunki testów, oprócz wyników działania zawierają cały interfejs programu testującego. Po prawej stronie okienka dialogowego interfejsu programu testującego znajdują się kolejno następujące przyciski:

„OK.” oraz „Cancel” - zakończenie działania programu,

„Wczytaj” – wczytanie testowanego obrazu,

„Usun Pkt’y” – usuwa zaznaczone punkty (początkowy i końcowy),

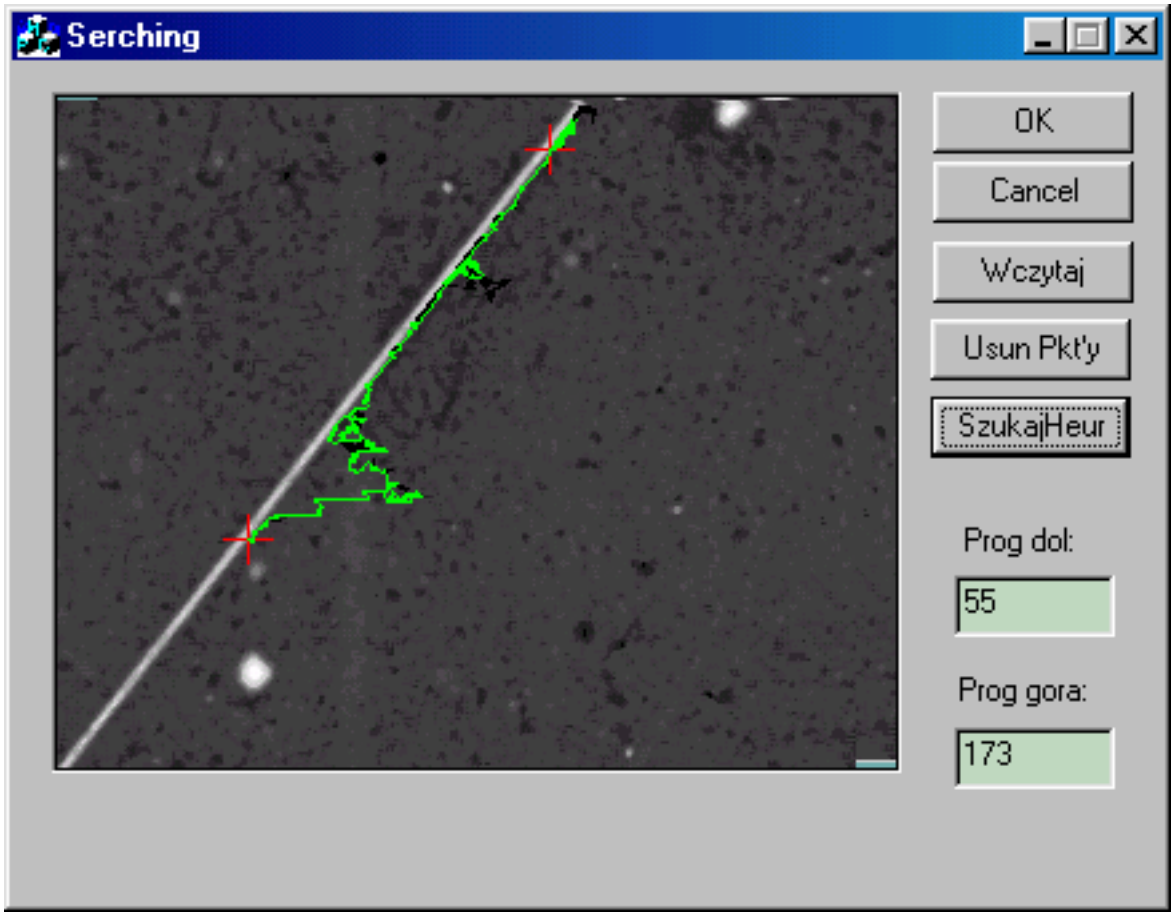
„SzukajHeur” – uruchamia procedurę automatycznego śledzenia.

W dolnej części widoczne są dwa okna edycji służące do ustawiania parametrów działania algorytmu:

„Prog dol” – ustawienie wartości progu dolnego,

„Prog gora” – ustawienie wartości progu górnego.

Zaznaczenie punktów (początkowego i końcowego) odbywa się przez ustawienie kursora myszki nad odpowiednim punktem obrazu i kliknięcie lewego klawisza. Miejsce wstawienia punktu oznaczane jest przy pomocy czerwonego krzyża.



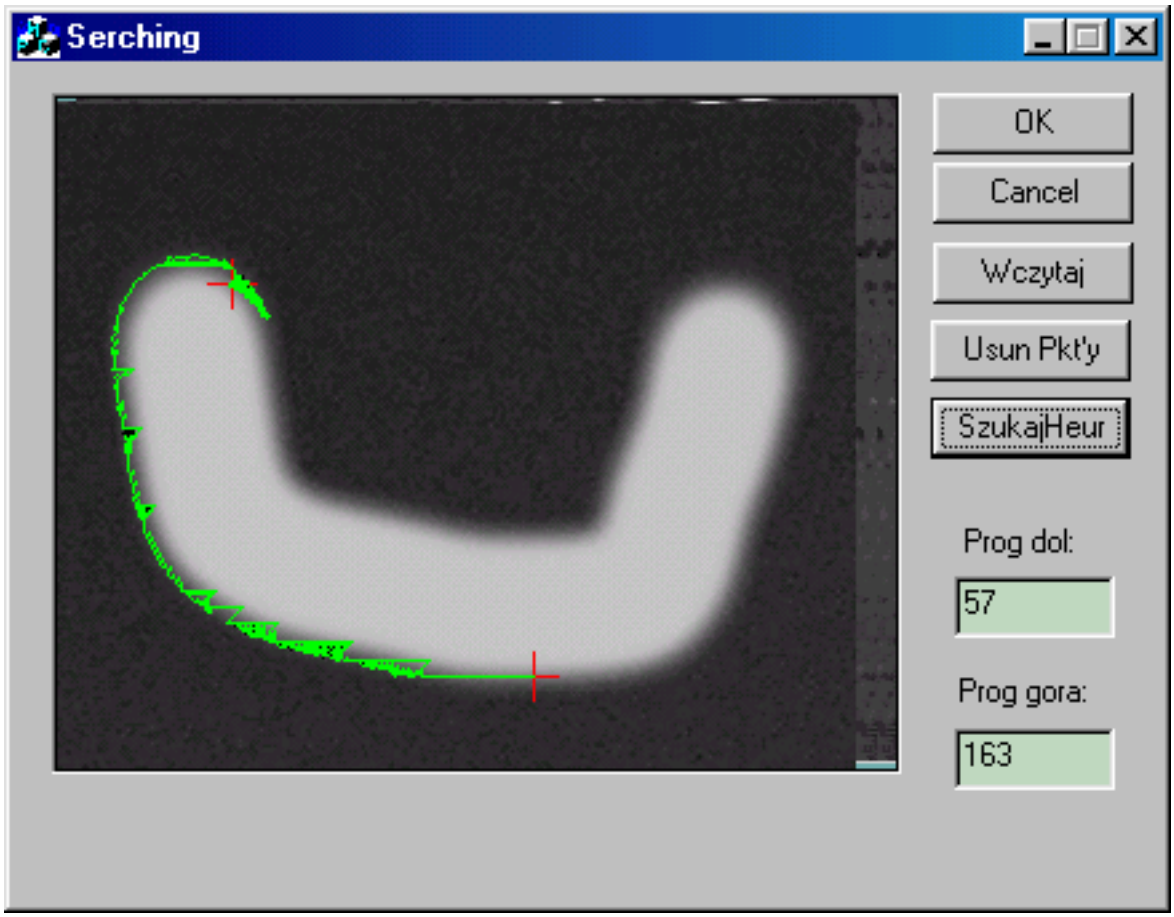
Rys. 4. Ustawienie zbyt niskiej wartość parametru *próg dolny*.

Rysunek 2 stanowi ilustrację wpływu złego usytuowania punktów (początkowego i końcowego) na działanie algorytmu. Dokładność usytuowania punktów ma szczególne znaczenie w przypadku małych długości przejścia (rozdział 4) i małych różnic jasności między pikselami tła i obszaru, dla którego jest znajdywana krawędź. W implementacji algorytmu jest wykonywane zerowanie wartości jasności tych pikseli, które nie prowadziły

do uzyskania właściwej drogi. Jest to widoczne w dolnej części rysunku 2 (lewy dolny narożnik) w postaci ciemnej otoczki wokół obrazu linii. Algorytm nie zapewnia od razu przejścia w kierunku punktu końcowego x_B i czasem może nastąpić przejście w kierunku przeciwnym, a później powrót. Rysunek 2 pokazuje jeszcze jeden mankament – przeskok z krawędzi jednej na drugą co jest możliwe w przypadku bardzo wąskich obiektów liniowych.

Rysunek 3 pokazuje wynik działania dla prawidłowo dobranego położenia punktów początkowego x_A i końcowego x_B oraz parametrów wejściowych (próg dolny, próg górny).

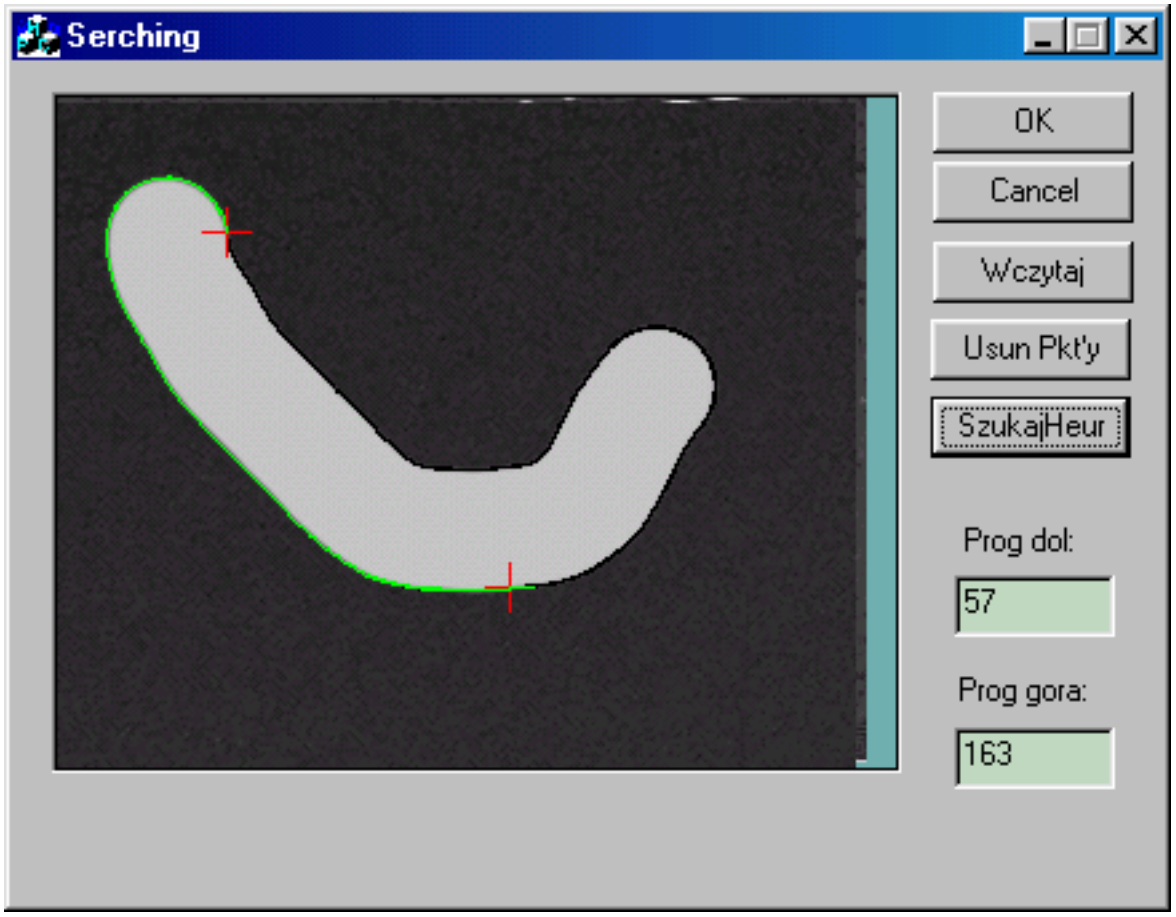
Rysunek 4 przedstawia przykład ustawienia zbyt niskiej wartość parametru *próg dolny*, co uwidacznia się poprowadzeniem drogi przez obszar tła.



Rys. 5. Zbyt duża *długości przejścia* (18 pikseli).

Na rysunku 5 *długość przejścia* wynosi ok. 18 pikseli, na obrazie dobrze uwidacznia się pewne „niezdecydowanie” algorytmu w postaci mocno poszarpanej krawędzi. W przypadku zwiększania *długości przejścia*, algorytm wykazuje większą tolerancje na ustawienie wartości parametrów wejściowych (*próg dolny*, *próg górny*). Jednak nie należy tego faktu traktować jako zaletę, ponieważ bardzo mocno zmniejsza się wtedy dokładność określenia krawędzi.

Na rysunku 6 *długość przejścia* wynosi 4 piksele; rysunek ten jest przykładem dobrze dobranych parametrów wejściowych (*próg dolny*, *próg górny*), oraz dobrego rozmieszczenia punktów x_A i x_B.



Rys. 6. Przykład prawidłowej *długości przejścia* (4 piksele) oraz dobrze dobranych parametrów wejściowych.

5. Wnioski

Przeprowadzone próby i analizy pozwalają sformułować poniższe wnioski.

1. Dla polepszenia działania algorytmu należałoby zmodyfikować obliczanie atrybutu *koszt*, tak aby lepiej oceniał właściwą drogę. Należałoby wprowadzić inne kryteria wyboru pikseli kandydujących do krawędzi np.:

$$\left|e(x_i) - e(x_j)\right| \leq T_1 \tag{4}$$

   różnica jasności piksela bieżącego i następnego mniejsza od zadanego progu [1].
2. Algorytm jest bardzo wrażliwy na zmiany położenia punktów x_A i x_B oraz wejściowych parametrów (*próg dolny*, *próg górny*) szczególnie dla małych *długości przejścia*.
3. Ma miejsce duże zużycie pamięci spowodowane dynamicznym przechowywaniem informacji o węzłach oraz przebiegu bieżącej drogi. W przypadku dużych obrazów, może to spowodować nadmierną ilość zaalokowanej pamięci. Dodatkowo w sytuacji, gdy ustawiono zbyt małą wartość parametru *próg dolny*, pamięć przydzielona dla krawędzi może rozrosnąć się do nieprzewidywalnych rozmiarów, a nawet doprowadzić do zablokowania działania programu.

4. Możliwość przeskoku z jednej krawędzi na drugą w przypadku wąskich linii praktycznie dyskryminuje stosowanie tego algorytmu dla wyznaczania przebiegu lin odciągowych.
5. Krawędź wyznaczona przez algorytm może zawierać zapętlenia, tzn. mogą wystąpić wielokrotne piksele w danej kolumnie lub wierszu obrazu, co nie jest dopuszczalne dla jednoznacznej identyfikacji krawędzi. Dlatego należałoby dodatkowo zastosować jakiś algorytm filtrujący.
6. Stosunkowo trudno w sposób intuicyjny ustalić prawidłowe wartości parametrów wejściowych (*próg dolny*, *próg górny*) oraz określić położenie punktów x_A i x_B. Wymagane jest pewne doświadczenie, a najlepiej rozumienie istoty działania algorytmu.
7. W wyniku przeprowadzonych prób stwierdzono, że badany algorytm w obecnej postaci nie nadaje się zbyt dobrze do wykorzystania dla potrzeb znajdowania przebiegu lin na obrazach cyfrowych. Dalsze prace nad lepszą adaptacją algorytmu wydają się bezcelowe, gdyż wymagało by to zbyt dużych nakładów w stosunku do osiągniętych korzyści. Najbardziej opłacalne z przyczyn ekonomicznych (czas) jak i z punktu widzenia możliwości osiągnięcia zadowalających efektów, wydaje się pójście w kierunku rozwoju algorytmów mniej skomplikowanych np. tych opisanych w [2].

Bibliografia

[1] I. Pitas, Digital image processing algorithms and applications.

[2] Archiwum Fotogrametrii i Teledetekcji vol. 10; Kraków 2000, J. Cieślar, Automatyzacja śledzenia linii na obrazach cyfrowych.

[3] Archiwum Fotogrametrii i Teledetekcji vol. 10; Kraków 2000 W. Mierzwa, S. Mikrut, Automatyczna identyfikacja elementów liniowych na obrazach cyfrowych.

[4] L. R. Foulds, Graph Theory Applications.

[5] J. D. Foley, A. van Dam, S. K. Feiner, J. F. Hughes, R. L. Phillips, Wprowadzenie do grafiki komputerowej.

[6] Oficyna Kallimach Kraków 1996; J. Grębosz, Symfonia C++.

[7] Helion Gliwice 1999; D. Chapman, Visual C++ dla każdego.

[8] Helion Gliwice 1999; R. S. Wright, M. Sweet, OpenGL Księga Eksperta.

[9] Journal of Photogrammetry and Remote Sensing, 50(4) 1995: 29-37; G. Vosselman, Application of tree search methods in digital Photogrammetry.