AGH Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Raport na temat:

Segmentacja i wyznaczanie położenia igły cytorobota

Wykonał zespół badawczy: **dr hab. inż. Paweł Rotter**
inż. Wiktor Muroń
Piotr Lizończyk
prof. dr hab. inż. Witold Byrski

Kraków, 2015

Narodowe Centrum Badań i Rozwoju

Praca wykonana w ramach projektu CYTOROBOT finansowanego przez Narodowe Centrum Badań i Rozwoju w latach 2012-2015.
Umowa nr PBS1/A9/1/2012 w ramach Programu Badań Stosowanych w ścieżce A

PBS Program Badań Stosowanych

Spis treści

Celem zadania 6 jest utworzenie aplikacji i algorytmów zdolnych określić aktualne położenie igły oraz stwierdzić czy nastąpiło poprawne przebicie korka butelki za pomocą igły strzykawki.

W ramach realizacji zadania zaproponowaliśmy kilka alternatywnych metod. Do każdej z nich potrzebne było wyłonienie w pierwszej kolejności pierwszego planu za pomocą segmentacji tła MOG2, wykorzystanej już wcześniej w badaniach drobinek i dużych obiektów. Z tego wynika, że analizę igły możemy przeprowadzać jedynie w trakcie jej ruchu. Ograniczenie to ma sens, gdyż w założeniu projektowym ważne jest, aby aplikacja była odporna na zmianę igły (w przeciwnym przypadku **musielibyśmy** posiadać bazę wszystkich dostępnych igieł, a znalezienie igły byłoby dodatkowo utrudnione, gdy część igły byłaby wewnątrz pojemnika).

Również ważne jest, aby uodpornić algorytmy na zróżnicowane tło, w którym działyby się wszystkie prace. Ze względu na fakt, że w danym momencie rodzaj tła nie jest ustalony, zdecydowano że najlepszym bo najbardziej uniwersalnym podejściem będzie zastosowanie metody segmentacji pierwszego planu (ruchomego) i w ten sposób pozbycie się wpływu tła.

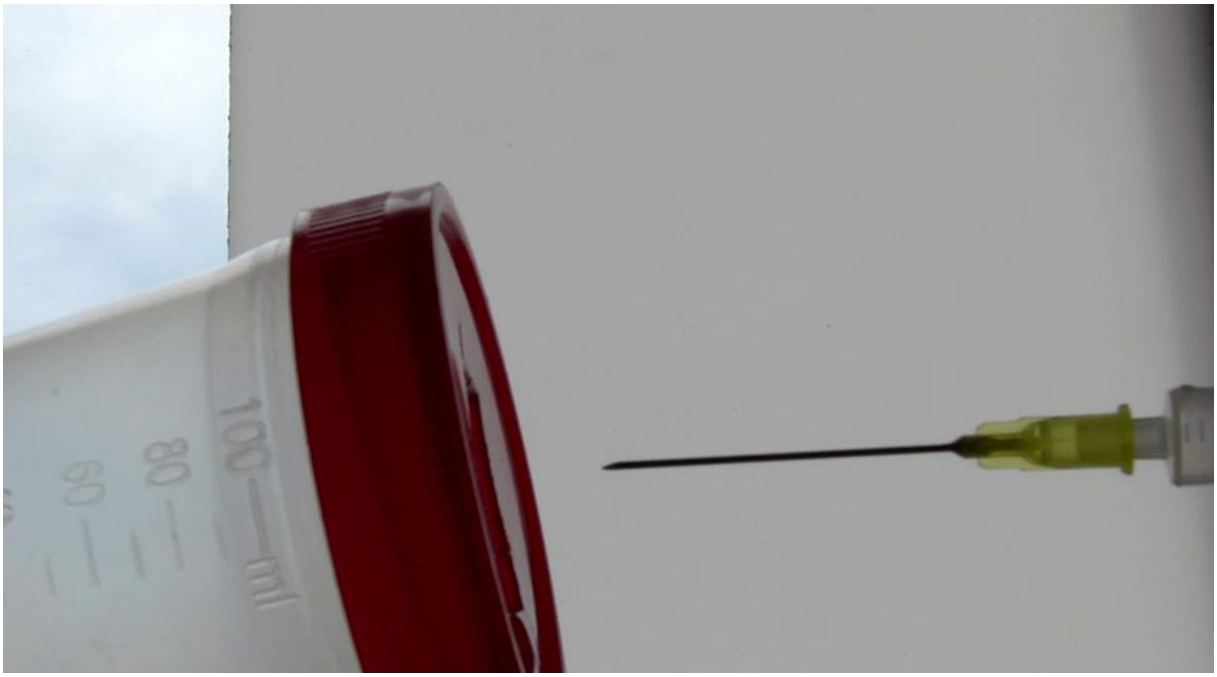
Trzy zaproponowane metody to:

- Metoda rzutowania na oś OX. Korzystanie z faktu, że igła jest równoległa lub niewiele odchylona od osi OX. Wyłonienie pierwszego planu i znalezienie fragmentu o stałej grubości w pewnym jak najszerszym zakresie, który jest uznawany za igłę.
- Transformata Hougha wykorzystana do wykrycia linii na pierwszym planie wyłonionym. Daje dość dokładną oś igły, na której leży w danym momencie krawędź igły. Sposób wyboru i aktualizowania tej osi w trakcie działania programu jest ściśle określony.
- Dyskretna transformata Fouriera – daje przybliżony kąt nachylenia igły do osi OX, co pozwala na potwierdzenie wyników z poprzednich dwóch sposobów.

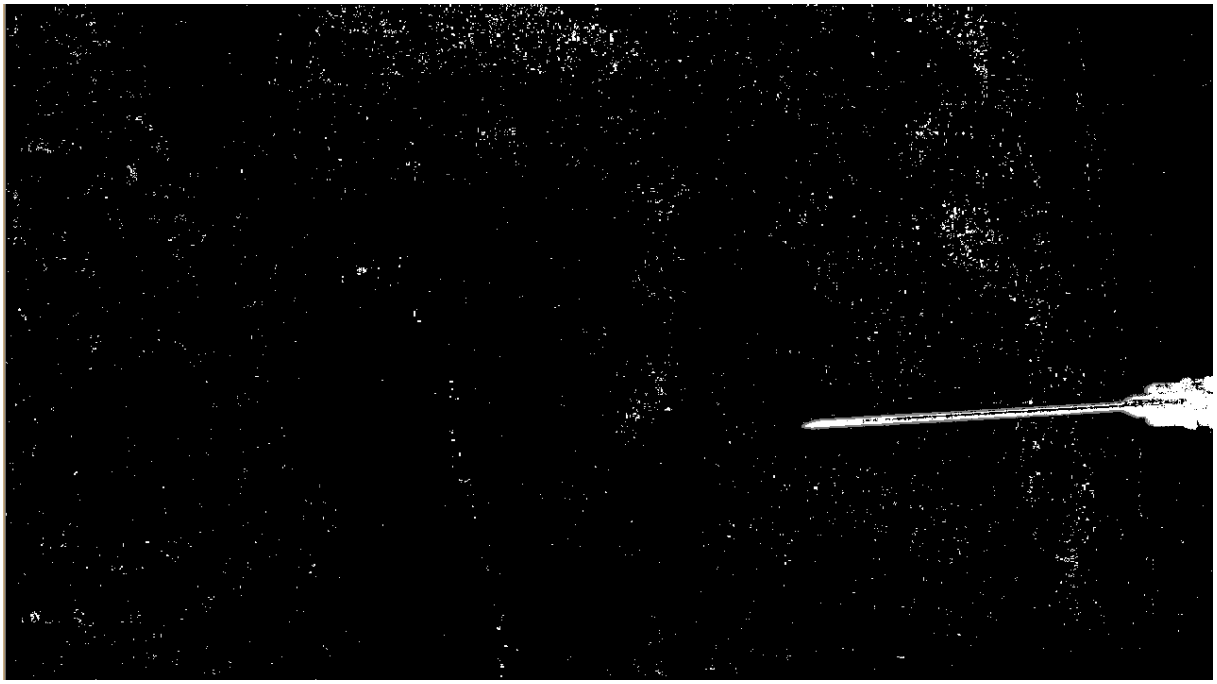
1 Metoda rzutowania na oś OX

1.1 Wydobycie informacji o obrazie

Na Rys. 1 przedstawiono obraz otrzymywany z kamery do dalszej obróbki. Na jego podstawie za pomocą algorytmu segmentacji cv::MOG2 otrzymano maskę pierwszego planu (Rys. 2)



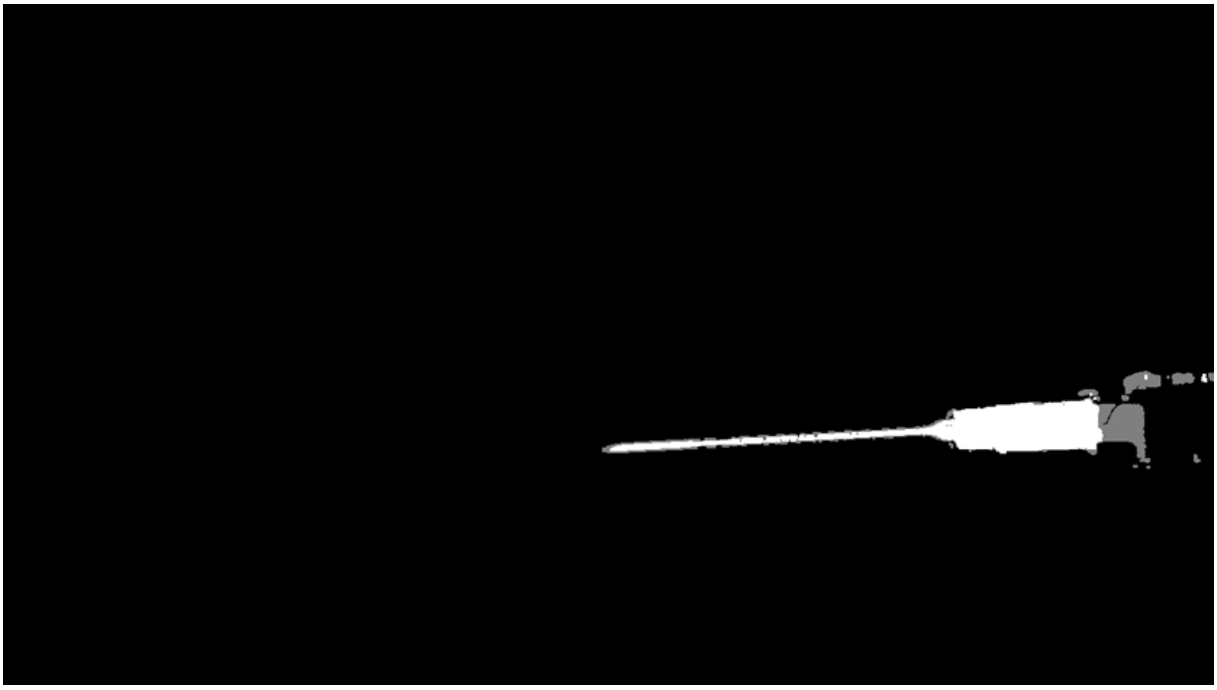
Rys. 1 Kadr z nagrania źródłowego, obrazującego wejście igły do pojemniczka i wyjście



Rys. 2 Obraz pierwszego planu otrzymanego za pomocą segmentacji MOG

Występuje problem z dość dużym zaszumieniem obrazu przy odpowiednio wysokiej jakości nagrywania. W tym przypadku obraz był nagrywany w jakości 1080p i to sprawiło, że segmentacja MOG dała dość dużo pikseli typowo odpowiadających losowym szumom, stąd też konieczność

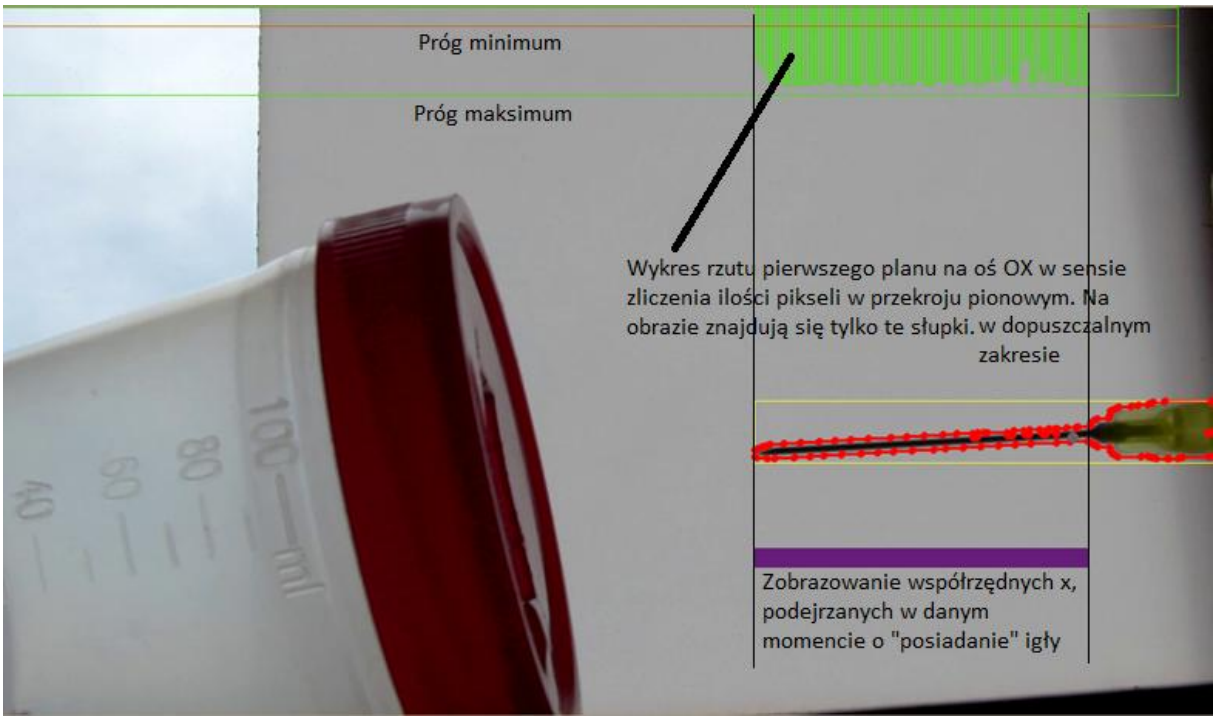
silnego odszumiania obrazu. Odszumianie nie mogło się odbywać bez ograniczeń, gdyż sam obiekt zainteresowania, czyli igła jest cienka, więc w skutek stosowania zbyt intensywnego odszumiania obrazu mogłaby zniknąć lub stracić swoje własności. Najskuteczniejsze okazało się odszumienie maski pierwszego planu za pomocą operacji dylatacji i erozji (Rys. 3).



Rys. 3 Maska pierwszego planu po odszumieniu dylatacją i erozją

W dalszym etapie następuje analiza otrzymanej maski. Wyszukiwane są one w sposób badający jedynie „grubość" warstwy pikseli na przekroju wzdłuż OY.

Jedynymi parametrami ustalanymi dla danej instancji programu są Próg maksimum i minimum. Dla każdej współrzędnej x obrazu, zliczane są wszystkie piksele wzdłuż osi OY, należące do wysegmentowanego pierwszego planu i należące do tej kolumny. Jeżeli liczba tych pikseli znajduje się w ustalonym przez próg MIN i MAX zakresie, to ta współrzędna x zostaje włączona do zbioru „podejrzanych" współrzędnych. Jedynie dla tych współrzędnych rysowany jest wykres ilości pikseli w pierwszym planie, w kolorze zielonym na górze obrazu. Reszta współrzędnych nie jest już dalej uwzględniana (Rys. 4)

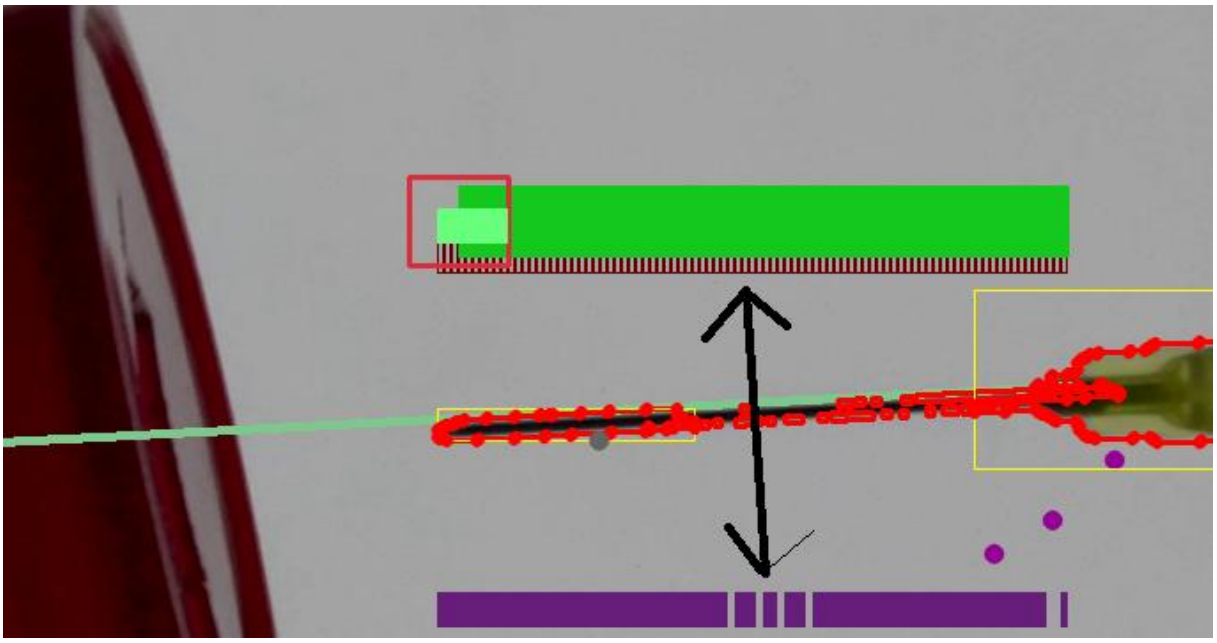


Rys. 4 Powstawanie zielonego i fioletowego czujnika obecności igły

Na Rys. 4 fioletowy zakres pokazuje współrzędne x podejrzane. Ze względu na zaszumienie i czasami niemożliwość idealnego dobrania progu MIN i MAX, powstają luki, wewnątrz dłuższego bloku, który powinien być w całości zapełniony. Analogicznie ze względu na te same czynniki czasem powstają osamotnione wykrycia w okolicach, w których nie powinny się pojawić.

1.2 Algorytm naprawiający wykrycie

W celu pozbycia się tego problemu napisano algorytm naprawiający, realizowany przez funkcję RebuildNeedles. Algorytm ten bazuje na następujących zasadach. Efekt naprawienia tym algorytmem to blok brązowych słupków (Rys. 5), powiązany strzałką z pierwotnym fioletowym. Jak widać wyeliminowane są w nim przypadkowe luki i izolowane rozpoznania, co umożliwia dalszą poprawną pracę na otrzymanych danych.

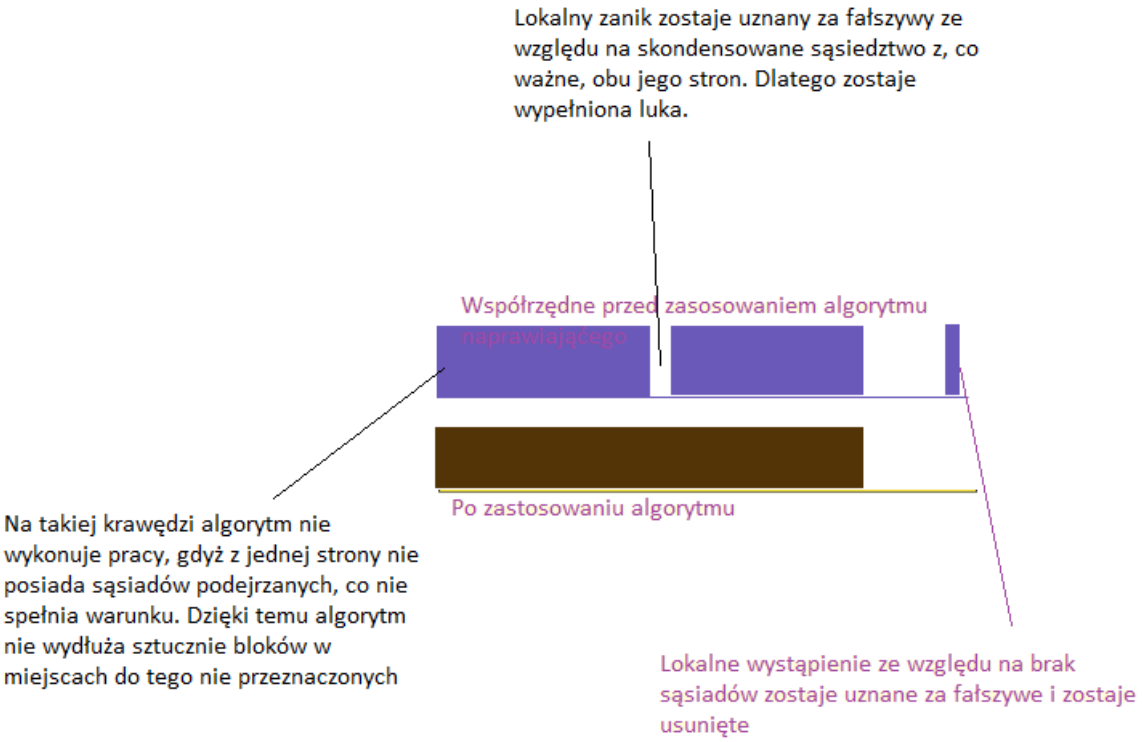


Rys. 5 Efekt działania funkcji RebuildNeedles

Przebieg algorytmu:

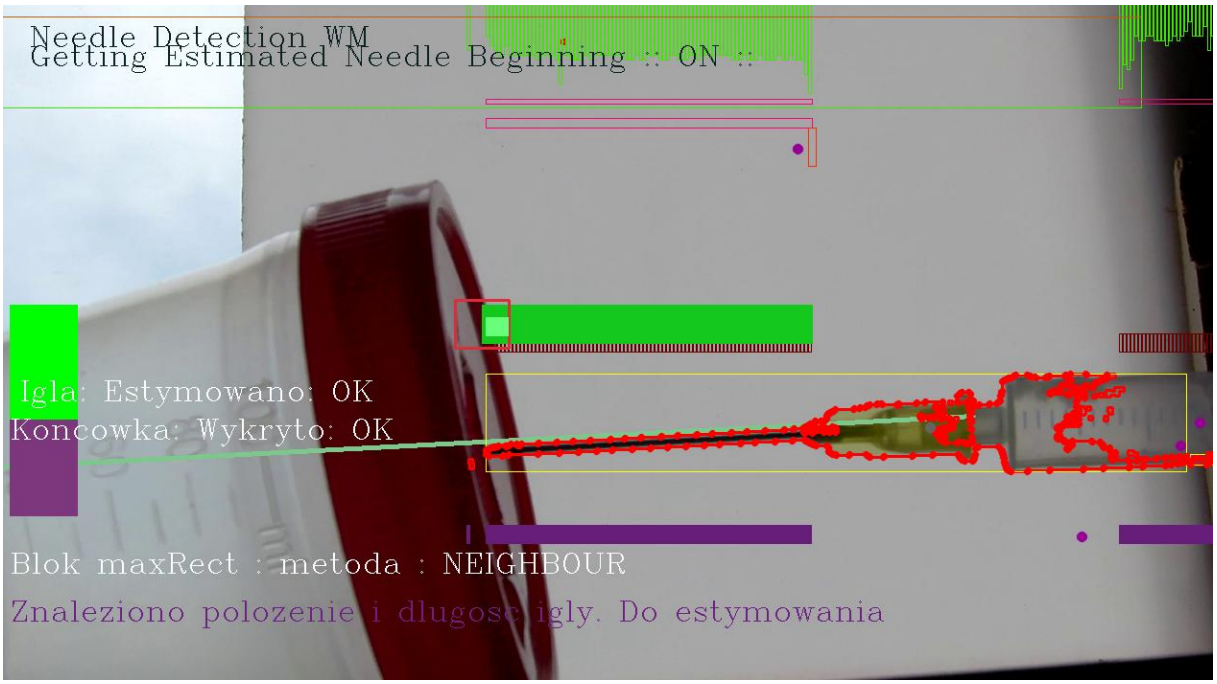
Dla każdej współrzędnej x:

- Zlicz współrzędne podejrzane w zakresie od 4 na lewo do 4 na prawo od aktualnej współrzędnej, oddzielnie zliczając piksele na lewo i na prawo.
- Jeżeli dana współrzędna jest podejrzana
    - Jeżeli podejrzanych na lewo jest >2, lub na prawo jest >2, to aktualna współrzędna pozostaje podejrzana, jeżeli warunek nie jest spełniony to uznajemy aktualną współrzędną za niepodejrzaną (nie idzie do brązowego bloku)
- Jeżeli dana współrzędna nie jest podejrzana
    - Jeżeli podejrzanych na lewo jest >3 i podejrzanych na prawo jest >0, lub na lewo >0 i na prawo >3, to aktualna współrzędna zostaje uznana za podejrzaną, jeżeli warunek nie jest spełniony to pozostawiamy aktualną współrzędną uznawaną za niepodejrzaną (nie idzie do brązowego bloku)



Rys. 6 Zobrazowanie metody RebuildNeedles

Algorytm ten pozostawia do dalszej pracy bloki, które uznać można za nielosowe, tzn. nie ma w nich małych odcinków, które w przypadkowy sposób powstały w wyniku małych wahań progów (Rys. 7).



Rys. 7 Przykład działania z końcowego programu

1.3 Analiza otrzymanych informacji

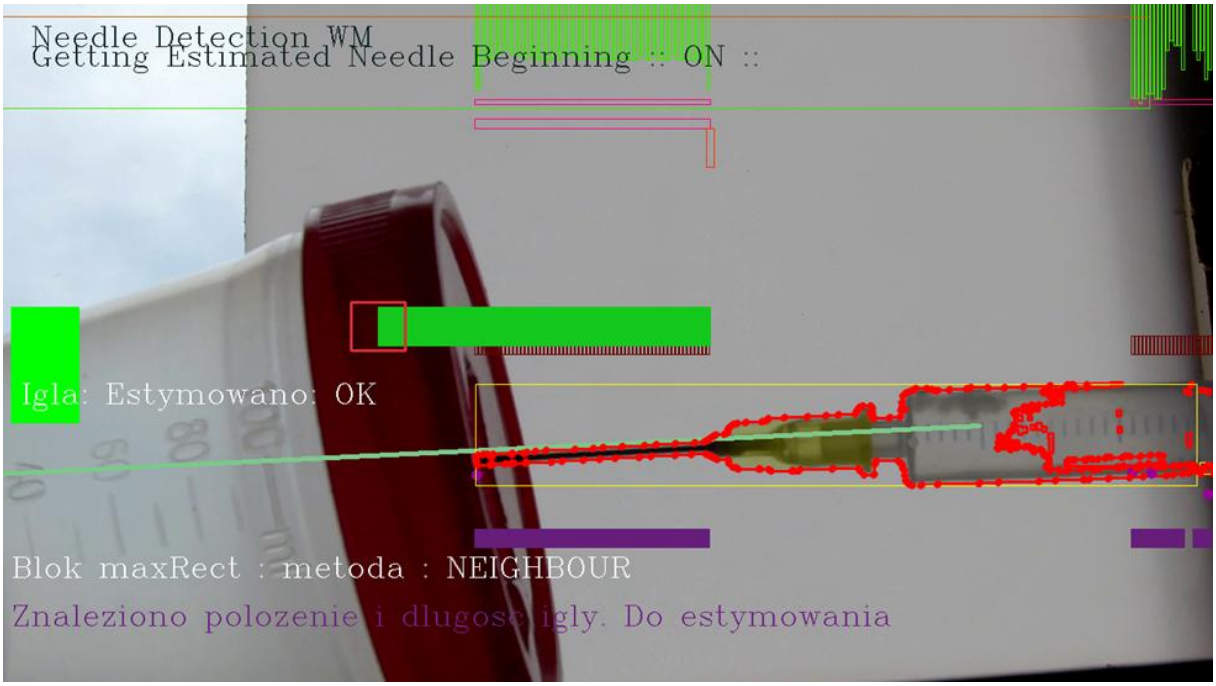
Kolejnym krokiem jest wydobycie informacji, gdzie dokładnie jest igła w danym momencie. Program działa w dwóch trybach:

a) tryb *przed rozpoznaniem* – kiedy nie wykryto jeszcze igły na żadnej klatce
b) tryb *po rozpoznaniu* – kiedy już znamy położenie igły na poprzedniej klatce działania programu

W trybie a) spośród naprawionych brązowych bloków szukamy początku i końca bloku o największej rozpiętości. Zapamiętujemy jego położenie i długość dla następnych iteracji. Jeżeli przez kolejne 10 klatek ten najdłuższy wykryty blok nie zmienia znacząco długości, to uznajemy, że całość igły jest już w kadrze i została poprawnie wykryta. Jej początek pobieramy z aktualnego wykrytego max. długości bloku, a długość obieramy jako średnią max. długości bloków z tych właśnie 10 poprzednich iteracji. Tym samym przechodzimy do trybu „po rozpoznaniu" – b)

W trybie b) znamy długość igły i jej początek (położenie) z poprzedniej klatki. W aktualnym momencie szukamy brązowego bloku, którego początek znajduje się najbliżej początku znalezionej igły z poprzedniej iteracji. Stąd na obrazie metoda: NEIGHBOUR. Ważne jest aby, odległość ta nie przekraczała pewnej stałej, określonej jako maksymalna zmiana położenia początku igły na klatkę. Kiedy znaleziono blok, którego początek uznajemy za początek igły, to możemy zająć się obliczaniem położenia końca igły. Znając długość igły z etapu a) po prostu dodajemy do współrzędnych tą długość i otrzymujemy współrzędną x, w której okolicy powinna znajdować się końcówka igły w danym momencie. To oszacowanie jest konieczne, ponieważ w pewnym momencie igła wejdzie do pojemnika, a więc końcówka przestanie być fizycznie wykrywalna przez moment, kiedy wieczko będzie ją zasłaniało.

Wyznaczana rozpiętość (w sensie szerokości) w danym momencie igły to jasnozielony prostokąt. Jego początek jest w danym momencie wyszukiwany z brązowych bloków, a koniec jest obliczany przez znaną długość igły (Rys. 8).

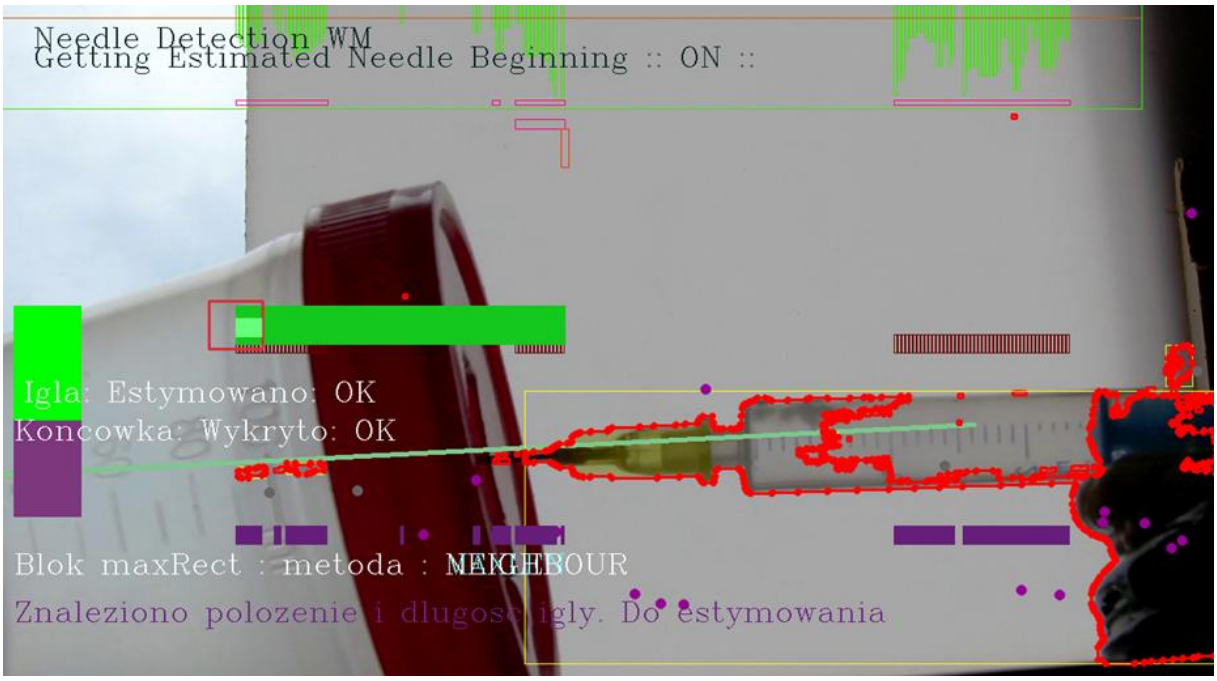


Rys. 8 Estymacja igły pomimo braku jej całości w kadrze za pomocą zielonego prostokąta

Kiedy w kolejnym kroku wyznaczony koniec igły znajduje się za wieczkiem, z oczywistych względów w miejscu końcówki nie może być wykryty brązowy blok, gdyż w tym miejscu igła jest

zasłonięta. Program nie uznaje tego za błąd, ale czeka aż pojawi się część brązowego bloku po drugiej stronie wieczka.

Następnie następuje moment, gdy igła przebiła wieczko w poprawny sposób. Jeżeli wykryto obecność brązowego bloku w miejscu, gdzie zgodnie z obliczeniami (jasnozielony blok estymacji) powinna być końcówka igły, to tym samym można uznać przebicie za poprawne (Rys. 9).



Rys. 9 Poprawne przebicie wieczka igły.

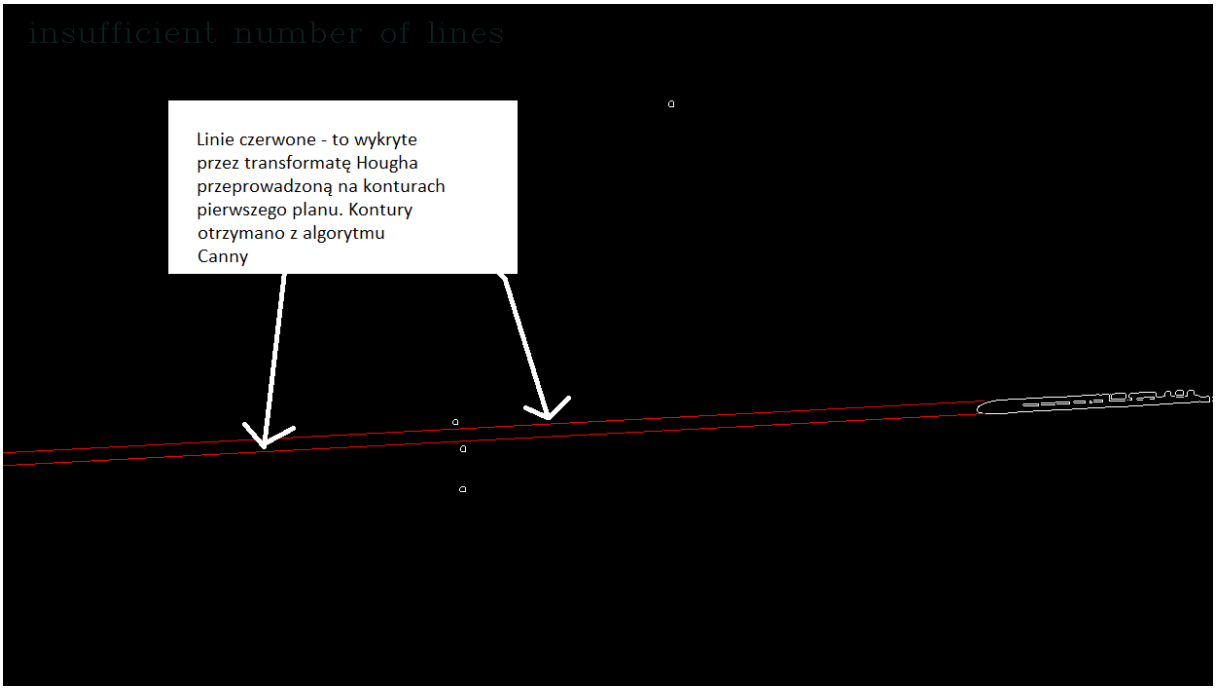
2 Metoda używająca transformaty Hougha

Przedstawiona poniżej metoda daje przybliżone położenie osi igły. Powstaje ono, przy założeniu, że na początku nie ma igły w kadrze. Następnie w wyniku analizy ruchu otrzymujemy pierwszy plan za pomocą segmentacji MOG.

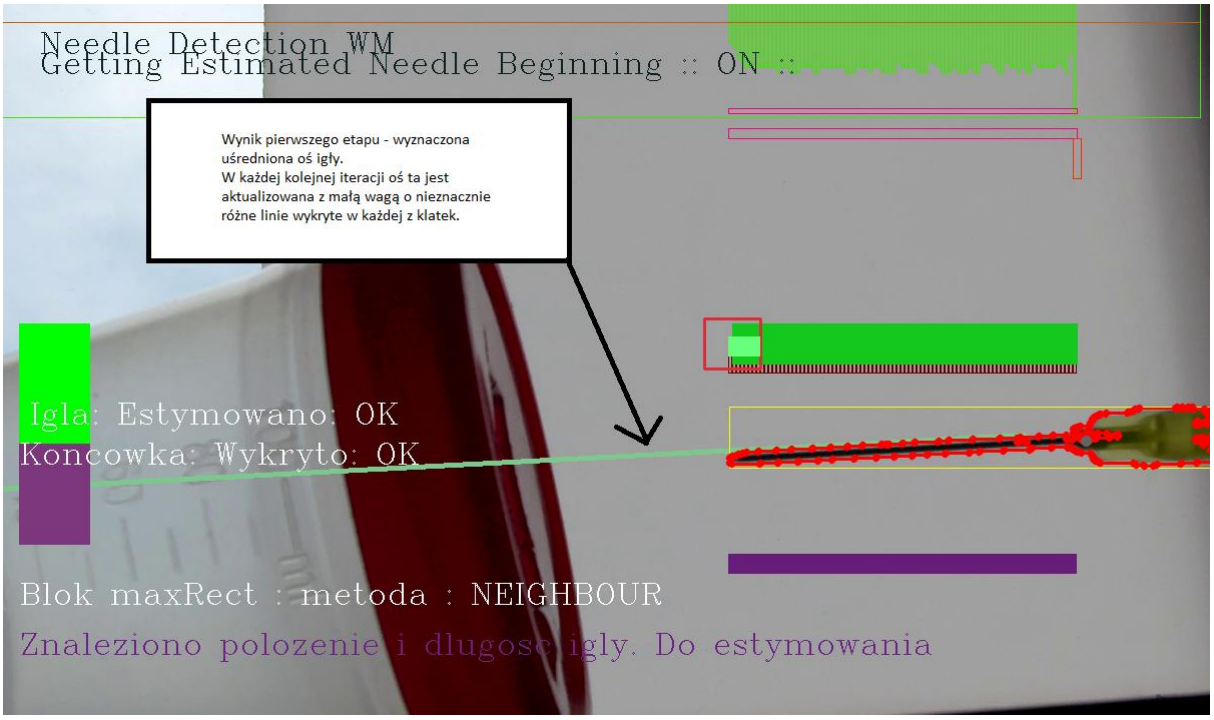
Na bazie wyniku tej segmentacji uruchamiamy algorytm transformaty Hougha o odpowiednich parametrach, zabezpieczających przed wykryciem nieodpowiednich, np. zbyt krótkich linii (wykrywanie linii – cv:: HoughLines). Daje nam to pierwsze linie, które są liniami odpowiadającymi igle wchodzącej w kadr. Kiedy otrzymujemy odpowiednio dużą (przekraczającą ustalony parametr - NUMBER_NEXT_MEDIUM_FRAMES = 15) liczbę linii wykrytych (np. dolnych i górnych krawędzi igły) to uśredniamy wszystkie te linie do jednej linii zwanej tymczasową osią igły. Wszystkie te pierwsze linie wstawiamy do listy cyklicznej o długości NUMBER_NEXT_MEDIUM_FRAMES =15.

Do następnego kroku przechodzimy, gdy mamy już zapełnioną listę cykliczną z pierwszego kroku i mamy tymczasową oś igły. W każdym kolejnym kroku wyszukujemy nadal linii i spośród wszystkich odnalezionych szukamy tych, które nie są znacząco (ustalone w programie empirycznie parametry) odległe od tymczasowej osi igły. Każdą linię spełniającą kryterium niezbyt dużej odmienności od osi dodajemy do listy, usuwając najstarszą z byłych linii w zamian. Następnie na podstawie zaktualizowanej listy obliczamy nową, uśrednioną linię osi. Daje nam to oś, o dość dobrej inercji i

odporności na zakłócenia powstałe przez przypadkowe wykrycia linii w losowych miejscach. Efekty widać na Rys. 1, Rys. 2.



Rys. 1 Proces powstawania osi igły. Każda z nowych czerwonych linii, która nie różni się znacząco (położeniem i kątem nachylenia) od aktualnej osi, zostaje dodana do listy tworzącej nową oś igły.

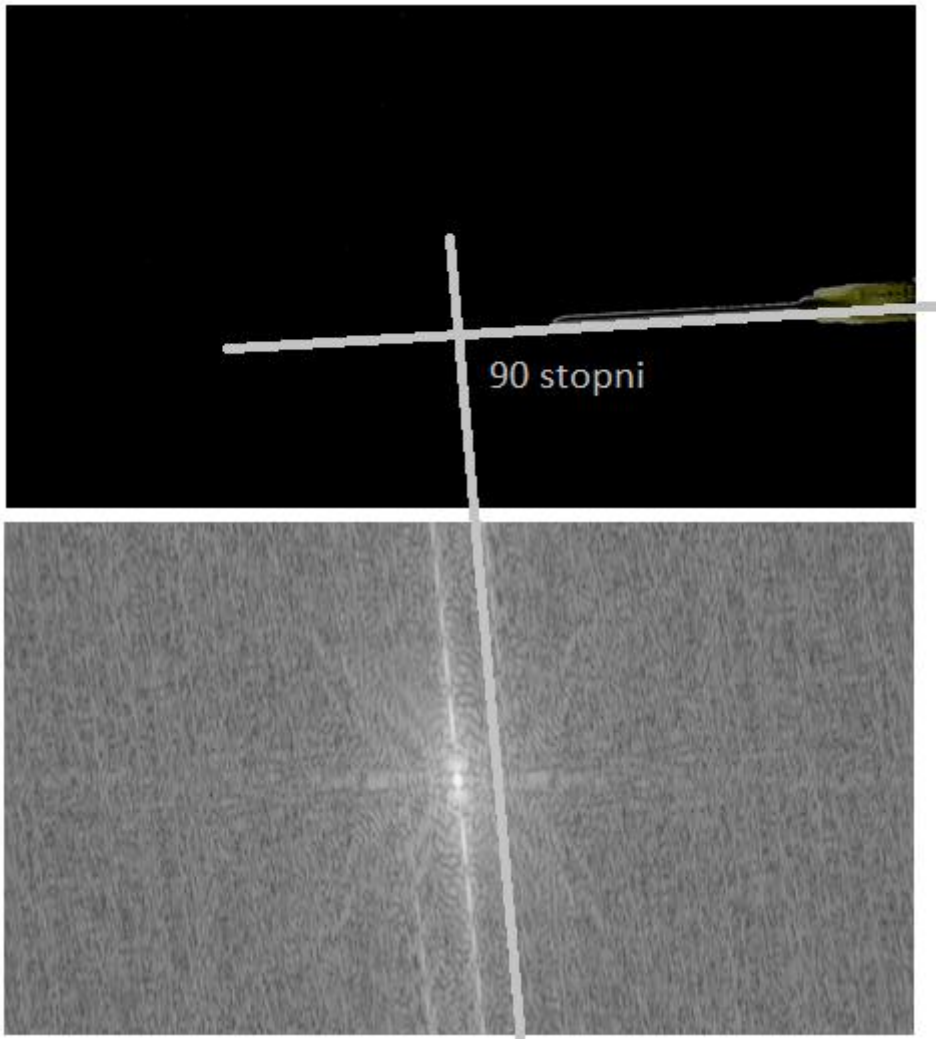


Rys. 2 Naniesienie efektu działania algorytmu na główne okno programu. Jak widać, oś powstała w wyniku działania algorytmu skutecznie odzwierciedla aktualne położenie igły.

3 Metoda wykorzystująca DFT – dyskretną transformatę Fouriera

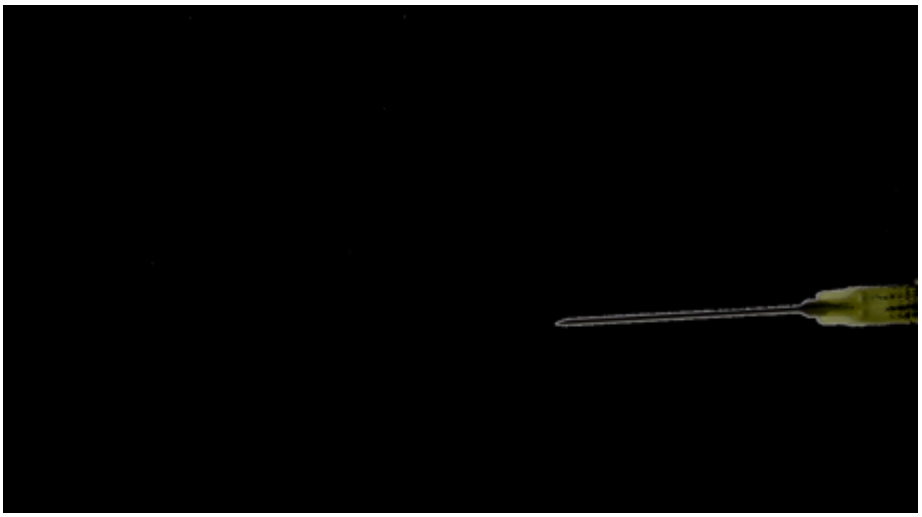
Metoda ta wykorzystuje pewne własności optyczne wykresu dyskretnej transformaty Fouriera, a mianowicie fakt, że przy małej ilości danych w obrazie wzorcowym transformata daje obraz, w którym dość wyraźnie jest zaznaczona oś prostopadła do dominującej osi całego obrazu (np. strzykawki).

Dla obrazu wzorcowego, będącego ekstrakcją pierwszego planu otrzymuje się obraz transformaty (Rys. 1), które należy następnie analizować.

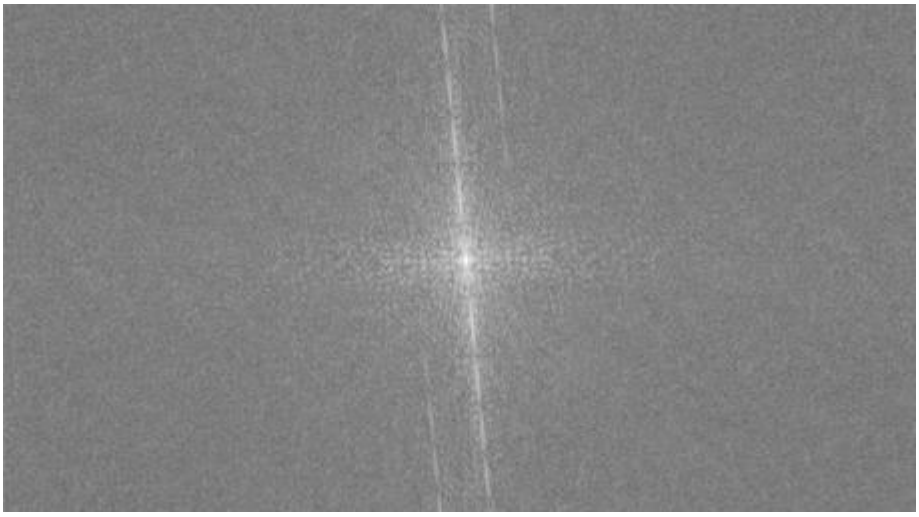


Rys. 1 Zobrazowanie zależności wzorca i transformaty DFT. Próbka i rezultat 1

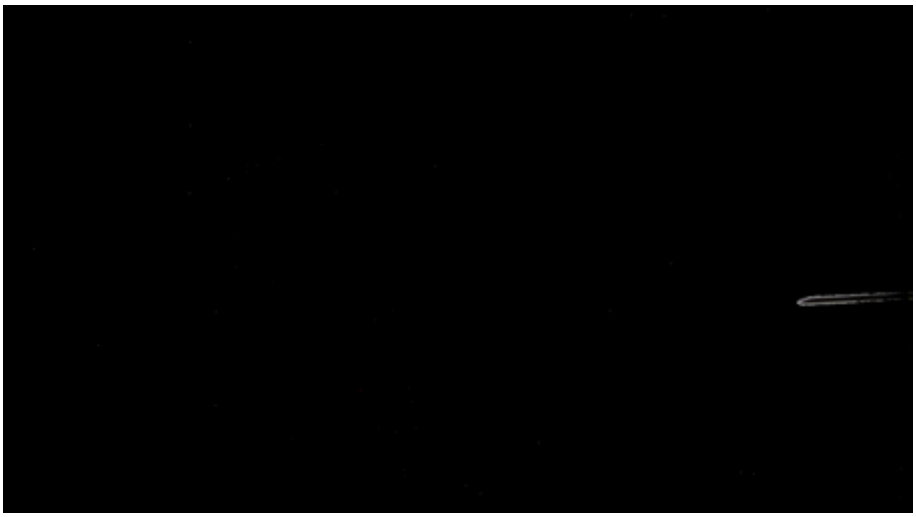
Dzięki takiemu dość szybkiemu i pobieżnemu działaniu (bez dokładnej analizy np. ilości wykrywanych linii) możemy np. mieć dodatkową informację na temat nachylenia osi igły do OX, co jest pomocne do sprawdzenia poprawności informacji co do nachylenia wykrytego w metodzie używającej transformaty Hougha.



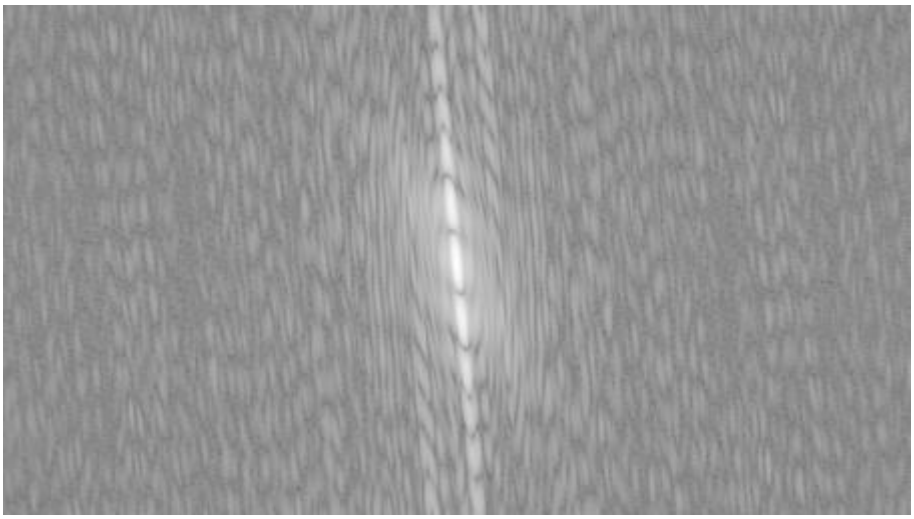
Rys. 2 Próbka 2



Rys. 3 Rezultat 2



Rys. 4 Próbka 3 – mała część igły w planie



Rys. 5 Rezultat 3

Jak zaobserwowano w różnych sytuacjach (Rys. 2, Rys. 3 oraz Rys. 4, Rys. 5) efekt otrzymany przez ten algorytm może przynieść, jeżeli zajdzie taka potrzeba w projekcie, dość obiecujące wyniki.