



A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Projekt dyplomowy inżynierski

Sprzętowo-programowy system wizyjny wspomagający autonomiczne lądowanie drona.

Hardware-software vision system supporting the autonomous landing of a drone.

Autor: Jakub Kłosiński
Kierunek studiów: Automatyka i Robotyka
Opiekun pracy: dr inż. Tomasz Kryjak

Kraków, 2019

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpozna bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, videogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . .

Spis treści

1. Wprowadzenie	7
1.1. Cele pracy	7
1.2. Zawartość pracy.....	7
2. Metody przetwarzania obrazu i sposoby sterowania w procesie autonomicznego lądowania drona	9
3. Opis implementacji	11
3.1. Zastosowana platforma sprzętowa.....	11
3.2. Implementacja modelu programowego	11
3.3. Implementacja toru wizyjnego w układzie	12
3.3.1. Konwersja z przestrzeni barw RGB do YCbCr	12
3.3.2. Binaryzacja.....	12
3.3.3. Mediana.....	12
3.3.4. Erozja i dylatacja.....	13
3.3.5. Środek ciężkości i prostokąt otaczający.....	13
3.3.6. Indeksacja.....	14
3.4. Komunikacja układu ZYBO Z7-20 ze sterownikiem Pixhawk	17
4. Testy i ocena wyników	19
4.1. Test wykrywania koloru znacznika w przestrzeni barw YCbCr	19
4.2. Wyznaczenie kąta widzenia kamery przy różnych ustawieniach rozdzielczości	20
4.3. Test detekcji znacznika na obrazie	20
4.4. Test regulacji położenia drona	21
4.5. Lądowanie na nieruchomym lądowisku	22
5. Podsumowanie i kierunki dalszych prac.....	29

1. Wprowadzenie

1.1. Cele pracy

1.2. Zawartość pracy

2. Metody przetwarzania obrazu i sposoby sterowania w procesie autonomicznego lądowania drona

W przypadku lądowania na stacjonarnym celu głównym problemem jest skuteczna detekcja znacznika znajdującego się na lądowisku. Do realizacji tego zadania niezbędne jest wykonanie szeregu etapów przetwarzania wizyjnego. W pracy [1] przedstawione zostały następujące kroki przetwarzania obrazów:

- zamiana obrazu kolorowego na obraz w skali szarości,
- binaryzacja ze stałym progiem,
- indeksacja,
- odrzucenie obiektów o liczbie pikseli mniejszej niż zadany próg,
- binaryzacja ze stałym progiem,
- identyfikacja znacznika.

Progi binaryzacji oraz odrzucenia małych obiektów zostały dobrane eksperymentalnie.

Aby dodatkowo wyeliminować biały szum oraz zakłócenia typu „sól i pieprz” w [2] wprowadzono operację mediany z oknem 3x3 na obrazie w skali szarości. Aby usunąć zbyt duże obiekty, zdecydowano się również wprowadzić górną granicę liczby pikseli należących do obiektu. W algorytmie przedstawionym w pracy [3] binaryzacji dokonano w przestrzeni HSV, na podstawie składowych S i V. Pozwoliło to na przeprowadzenie segmentacji niezależnej od koloru. Przed rozpoznawaniem kształtów, na obrazie binarnym dokonano kolejno erozji, detekcji krawędzi oraz dylatacji. Wykonanie erozji pozwoliło na eliminację z obrazu małych grup pikseli.

Istotnym elementem planowania autonomicznego lądowania drona jest wybór charakterystyki znacznika. W [4] przedstawiono marker złożony z trzech figur: kwadratu, koła i krzyża. Zaimplementowany został algorytm detekcji lądowiska polegający na kolejnym wykrywaniu figur i uzyskiwaniu coraz lepszej znajomości pozycji celu.

Prostsze rozwiązanie zostało pokazane w [5], gdzie znacznik ma kształt litery H. Odległość drona od lądowiska obliczana jest na podstawie liczby pikseli dzielących środek ciężkości znacznika od środka obrazu. Opisano tam również metodę wyznaczania orientacji drona względem takiego markera. Polega ona na znalezieniu piksela najbardziej odległego od środka ciężkości.

Inny znacznik został użyty w [1]. Składa się on z czterech pierścieni na czarnym tle. Każdy pierścień posiada unikalny stosunek promienia wewnętrznego do zewnętrznego i stanowi oddzielny obiekt dla systemu wizyjnego. Obiekty, które nie mają dokładnie jednej dziury wewnętrznej, są pomijane. Każdy z pozostałych jest ostatecznie identyfikowany za pomocą współczynnika kształtu. Zaletą takiego znacznika jest możliwość dołożenia kolejnych, większych pierścieni, jeśli lądowisko ma być widoczne z większej wysokości.

Implementacja toru wizyjnego w systemie potokowym na platformie sprzętowej rodzi dodatkowe trudności. Bez dołączenia dodatkowej pamięci RAM niemożliwe są działania na całej ramce obrazu. Jest to uciążliwe przy wykonywaniu operacji kontekstowych, wymagających znajomości otoczenia piksela. Problem rozwiązano w [3], gdzie przedstawiono koncepcję realizacji takich operacji przy użyciu tablicy 3×3 i bufora.

W procesie autonomicznego lądowania drona, na podstawie wyznaczonej odległości od celu, do UAV wysyłane są sygnały sterujące. W [6] wykorzystano trzy regulatory PID, zadające prędkość drona. Równocześnie minimalizowana jest każda ze składowych wektora położenia. [1] zwraca uwagę na błędy spowodowane pochyleniem i przechyleniem drona. Przed wysłaniem sygnału do regulatora wykonywana jest korekta uchybu na podstawie pomiaru kąta pochylenia i przechylenia.

Metody lądowania na statycznym lądowisku nie zawsze znajdują zastosowanie do śledzenia poruszającej się platformy. W przypadku, gdy lądowisko przestaje być widoczne, możliwe jest przewidywanie jego ruchu. W [4] użyto algorytmu pozwalającego na predykcję zachowania platformy. Wykorzystano model dynamiczny celu oraz rozszerzony filtr Kalmana. Spośród możliwych trajektorii dotarcia do celu wybierana jest najlepsza pod względem energetycznym.

Podsumowując, wykonanie lądowania na statycznym lądowisku wymaga skutecznej detekcji znacznika znajdującego się na platformie. Znalezienie względnej pozycji markera umożliwia przekazanie informacji o uchybie do regulatora sterującego dronem. Lądowanie na poruszającej się platformie może zostać wykonane w podobny sposób, jednak wprowadzenie przewidywania ruchu lądowiska daje odporność na zaniki widoczności. Wymaga to jednak użycia znacznie bardziej zaawansowanych narzędzi.

3. Opis implementacji

3.1. Zastosowana platforma sprzętowa

Zdecydowano się na płytę rozwojową Digilent Zybo Z7-20 oraz kamerę Digilent PCAM 5C. Przeprowadzenie syntezy i implementacji możliwe było przy użyciu darmowego oprogramowania Vivado SDK WebPack. Podczas prac używano wersji 2018.2. Płyta wyposażona była w układ FPGA XC7Z020-1CLG400C stanowiący część rekonfigurowalną (PL – ang. *programmable logic*) oraz dwurdzeniowy procesor Cortex-A9 (PS – ang. *processing system*), taktowany z częstotliwością 667 MHz. Oba komponenty zostały wykorzystane w projekcie. Do dyspozycji projektanta pozostało 53 200 tablic LUT (ang. *Look-up Table*), 106 400 przerzutników *flip-flop* oraz 630 KB pamięci blokowej RAM.

Producent na swojej stronie internetowej zapewnia projekt demonstracyjny połączenia kamery i płytki [7]. Przez port szeregowy możliwa jest zmiana rozdzielczości, szybkości akwizycji ramek, współczynnika korekcji gamma, ustawień balansu bieli. Możliwe są następujące opcje dotyczące dwóch pierwszych parametrów:

- 1280 x 720, 60 fps,
- 1920 x 1080, 15 fps,
- 1920 x 1080, 30 fps.

Testy pokazały również, że dla rozdzielczości 1280 x 720 większy jest kąt widzenia kamery. Kwestię tę opisano w sekcji 4.2. Ze względu na powyższy fakt oraz przyspieszenie obliczeń, zdecydowano się na najmniejszą dostępną rozdzielczość.

Pobrany projekt stanowił bazę do dalszych prac.

3.2. Implementacja modelu programowego

Model programowy został napisany w pakiecie Matlab przy wykorzystaniu funkcji dostępnych w Image Processing Toolbox. Pozwoliło to na sprawne wykonywanie operacji na obrazach. Aby możliwe było wprowadzanie ramek obrazu do modelu, należało umożliwić akwizycję obrazów na kartę SD. Uaktywniono interfejs procesora SD0, do projektu w SDK dodano bibliotekę „xilffs” i, korzystając z funkcji

systemu plików opisanych w [8], napisano zapis ramek do pliku. Ze względu na prostotę pliku, składającego się jedynie z nagłówka oraz wartości kolejnych składowych RGB, zdecydowano się na format ppm. W późniejszym stadium projektu, z powodu różnic na obrazie zapisywany na ramkę i wyświetlany na ekranie, zdecydowano się na zmianę tej koncepcji. Zaimplementowano cały tor wizyjny w części PL układu Zybo Z7-20 i przechwytywano przetworzone obrazy wyświetlane na monitorze. Dzięki takiemu podejściu podczas testowania widziano rzeczywiste wyniki przetwarzania.

3.3. Implementacja toru wizyjnego w układzie

3.3.1. Konwersja z przestrzeni barw RGB do YCbCr

Przestrzeń barw YCbCr wykorzystuje 3 zmienne: składową luminancji Y, składową różnicową chrominacji Cb, która wyraża różnicę między luminancją, a niebieskim, oraz składową różnicową chrominacji Cr, oznaczającą różnicę między luminancją, a czerwonym. Zaletą stosowania tej przestrzeni barw jest oddzielenie sygnału luminancji od sygnałów chrominancji, pozwalające na dokonywanie dalszej obróbki sygnału przy mniejszej zależności od oświetlenia. Konwersję z przestrzeni barw RGB do YCbCr wykonano zgodnie ze wzorem 3.1. Przy implementacji wykorzystano sprzętowe mnożarki oraz sumatory.

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ -0,168736 & -0,331264 & 0,5 \\ 0,5 & -0,418688 & 0,081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad (3.1)$$

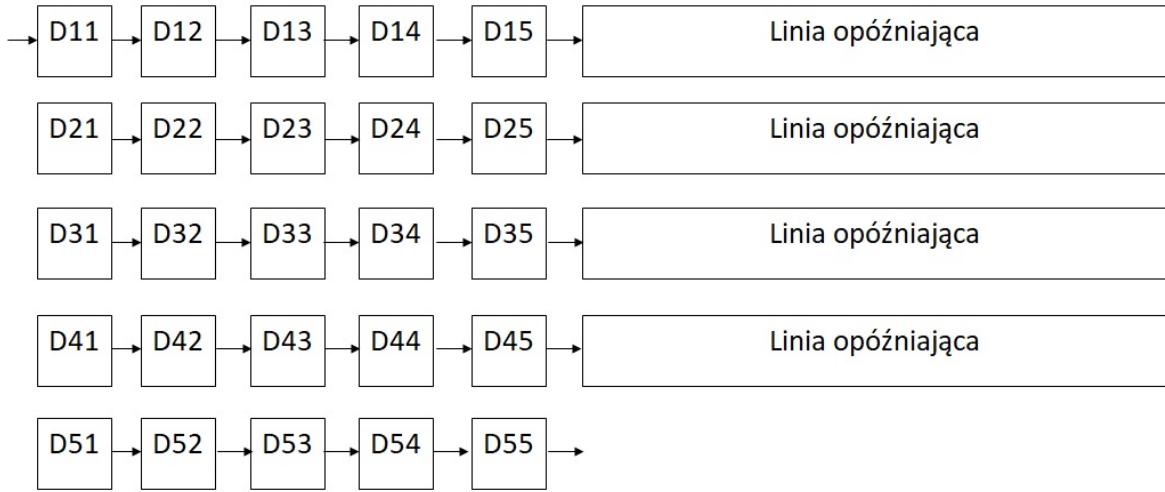
3.3.2. Binaryzacja

Piksel wyjściowy otrzymywał wartość maksymalną (kolor biały), jeśli wartości Cb i Cr mieściły się pomiędzy wyznaczonymi eksperymentalnie progami. W innym przypadku pikselowi przypisywana była wartość 0 (kolor czarny). Moduł binaryzacji pracował z zerową latencją.

3.3.3. Mediana

Mediana to operacja kontekstowa, w której pikselowi wyjściowemu przypisuje się wartość średniorównoważoną uporządkowanego zbioru wartości pikseli z otoczenia piksela wejściowego. W przypadku działania na obrazie binarnym operacja taka może być przeprowadzona przez obliczenie sumy wartości pikseli kontekstu, a następnie porównanie jej z połową maksymalnej wartości tej sumy. Ze względu na łatwość implementacji oraz wystarczające działanie, zdecydowano się na rozpatrywanie kontekstu w kształcie kwadratu o boku 5 pikseli. Spowodowało to konieczność zapamiętywania kontekstu piksela w 25 rejestrach oraz 4 linii obrazu w długich liniach opóźniających zbudowanych w oparciu o pamięć BRAM. Schemat wyznaczania kontekstu przedstawiono na rysunku 3.1. Sygnały synchronizacji zostały doklejone do wartości piksela i w przedstawionej strukturze przesuwają się razem z nim. Sumę wyliczano

w 2 etapach, dodając najpierw elementy w wierszach, potem sumując wyniki. Latencja modułu wynosiła 2, należało zatem opóźnić sygnały synchronizacji o tyle taktów zegara.



Rys. 3.1. Schemat wyznaczania kontekstu dla mediany, erozji i dylatacji.

3.3.4. Erozja i dylatacja

Erozja i dylatacja to operacje morfologiczne, w których pikselowi wyjściowemu przypisuje się wartość odpowiednio najmniejszego i największego piksela w sąsiedztwie piksela wejściowego. Sąsiedztwo piksela określa kształt i rozmiar elementu strukturalnego. Podobnie jak w przypadku mediany, zdecydowano się na kwadrat o boku 5 pikseli. Moduł erozji ustawia wartość piksela wyjściowego na maksymalną wartość, gdy w sąsiedztwie znajdują się same białe piksele. W module dylatacji zwracana jest wartość maksymalna, gdy przynajmniej jeden piksel w sąsiedztwie ma kolor biały. Latencja modułów wynosi 1.

3.3.5. Środek ciężkości i prostokąt otaczający

Wyznaczając środek ciężkości pikseli należących do obiektu, wykorzystano wzory 3.2, 3.3 i 3.4.

$$m_{00} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{ij} \quad (3.2)$$

$$m_{10} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} i * x_{ij} \quad (3.3)$$

$$m_{01} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} j * x_{ij} \quad (3.4)$$

gdzie:

N – szerokość obrazu w pikselach,

M – wysokość obrazu w pikselach,

x_{ij} – wartość piksela o współrzędnych i, j obrazu zbinaryzowanego.

Na ich podstawie obliczono środek ciężkości przy zastosowaniu wzorów 3.5 i 3.6.

$$X_{sc} = \frac{m_{10}}{m_{00}} \quad (3.5)$$

$$Y_{sc} = \frac{m_{01}}{m_{00}} \quad (3.6)$$

X_{sc} – współrzędna pozioma środka ciężkości,

Y_{sc} – współrzędna pionowa środka ciężkości.

W module na podstawie sygnałów synchronizacji oraz wymiarów obrazka wyznaczono współrzędne aktualnie przetwarzanego piksela. Jeśli jest to piksel należący do obiektu, następuje zwiększenie wartości odpowiednich rejestrów zgodnie ze wzorami 3.2, 3.3 i 3.4. Po przejściu przez całą ramkę obrazu wykonywane jest dzielenie na podstawie wzorów 3.5 i 3.6.

W module zintegrowano wyznaczanie środka ciężkości oraz prostokąta otaczającego. Znalezienie prostokąta sprowadza się do wyznaczenia skrajnych jego punktów na górze, dole, po lewej oraz prawej stronie. Do odpowiednich rejestrów trafiają wartości współrzędnych piksela, jeśli wykraczają poza aktualną zawartość rejestrów.

Moduł zwraca wartości współrzędnych środka ciężkości oraz skrajnych punktów prostokąta otaczającego.

3.3.6. Indeksacja

Indeksacja to operacja pozwalająca na wydobycie z obrazu poszczególnych obiektów. Obiekty rozumiane są jako grupy połączonych ze sobą pikseli. Zazwyczaj na wejście modułu indeksacji podawany jest obraz zbinaryzowany, natomiast na wyjściu pojawia się ramka z pikselami, których wartość odpowiada przypisanej do danego obiektu etykiecie. Rozważane jest otoczenie każdego piksela, składające się z trzech pikseli nad nim oraz jednego po lewej stronie, tak jak zostało to przedstawione na rysunku 3.2. Indeksacji dokonuje się bezpośrednio na obrazie wejściowym. Podczas iteracji po wszystkich pikselach, w przypadku znalezienia piksela należącego do któregoś z obiektów, może zajść jeden z trzech przypadków:

- (a) w otoczeniu piksela znajdują się same piksele nienależące do żadnego obiektu,
- (b) otoczenie zawiera jeden lub więcej pikseli, którym została wcześniej przypisana taka sama etykieta L,
- (c) w otoczeniu znajdują się piksele posiadające różne etykiety.

W pierwszym przypadku pikselowi zostaje przypisana nowa etykieta. Gdy spełniony jest warunek b, punkt otrzymuje etykietę L, natomiast jeśli zachodzi przypadek c, przypisywana jest mniejsza z etykiet.



Rys. 3.2. Sąsiedztwo piksela brane pod uwagę przy indeksacji

W ten sposób otrzymuje się obraz wstępnie poetykietowany. Najczęściej posiada on więcej przypisanych etykiet, niż obiektów. Dlatego istnieje konieczność złączenia ze sobą pewnych etykiet przy użyciu tablicy klejów. Tablica ta zawiera informację, które etykiety powinny zostać złączone. W przypadku a do tablicy klejów na pozycji odpowiadającej etykiecie zapisywana jest etykieta, natomiast gdy zachodzi opcja c etykietę mniejszą zapisuje się pod indeksem większym. Do sklejenia etykiet potrzebna jest druga iteracja, tym razem po obrazie wstępnie poetykietowanym.

Powyższy fakt jest główną przeszkodą w łatwym wykonaniu takiego algorytmu w systemie potokowym. Bez zapamiętywania całej ramki, w przypadku sklejania etykiet, niemożliwy jest powrót do wcześniej przetwarzanych pikseli. Pomimo tego, możliwe jest obliczenie pewnych cech obiektów, takich jak: pole, współrzędne środka ciężkości, prostokąt otaczający. W pracy [9] podano sposób, w jaki można tego dokonać. Opiera się on na scaleniu nie samych wartości pikseli, ale obliczanych na bieżąco parametrów obiektu.

Implementacja w języku Verilog rodzi dodatkowo trudności związane z określeniem przypadku istnienia tej samej lub różnych etykiet w otoczeniu piksela. Utrudnienie stanowi brak możliwości wykorzystania funkcji eliminującej zera z wektora, czy znajdującej minimum i maksimum. O ile zachodzenie przypadku a jest łatwe do wykrycia, to przypadki b i c wymagały rozważenia kilku możliwości. Zdecydowano się na wykrywanie ich za pomocą flag bitowych. Na ich podstawie wnioskowano o zachodzącym aktualnie przypadku oraz wskazywano, od którego piksela z otoczenia powinna zostać przepisana etykieta. Oprócz tego, na bieżąco obliczano prostokąt otaczający oraz liczbę pikseli należących do każdego ze znalezionych obiektów.

Po poetykietowaniu całej ramki obrazu wykorzystano tablicę klejów do złączenia obliczanych na bieżąco cech. Ten etap algorytmu zaimplementowano jako maszynę stanów:

- Stan 0 – Oczekiwanie na sygnał końca ramki wyznaczany na podstawie synchronizacji pionowej. W momencie wykrycia sygnału następuje rejestrowanie tablicy sklejeń i obliczonych parametrów, zerowanie tablic wypełnianych w kolejnym etapie oraz przejście do stanu 1.
- Stan 1 – Iteracja po tablicy sklejeń i uzupełnianie rzeczywistych wartości cech obiektów.
- Stan 2 – Uporządkowanie tablic z wyznaczonymi parametrami.
- Stan 3 – Obliczenie pola prostokąta otaczającego dla każdego znalezioneego obiektu. Wykorzystywana jest mnożarka o latencji 3.
- Stan 4 – Szukanie obiektu spełniającego warunki minimalnej wielkości pola powierzchni oraz stosunku pola prostokąta otaczającego do pola obiektu. W przypadku znalezienia takiego kształtu, na wyjście trafiają współrzędne jego prostokąta otaczającego. Jeśli żądany kształt nie zostanie znaleziony, informacja o tym również pojawi się na wyjściu.

Schemat przetwarzania danych po każdej ramce pokazano na rysunku 3.3

Tablica sklejeń:

1	1
2	1
3	1
4	4
5	5
6	5
7	4

Znalezione położenia lewego boku prostokąta:

1	15
2	10
3	20
4	25
5	22
6	17
7	8

Najmniejsze wartości dla każdego obiektu:

1	10
2	0
3	0
4	8
5	17
6	0
7	0

Wartości przygotowane do wyboru znalezionego obiektu:

1	10
2	8
3	17
4	0
5	0
6	0
7	0

Rys. 3.3. Schemat procesu przetwarzania informacji po każdej ramce obrazu z przykładowymi danymi.

Główną trudnością w implementacji opisanego algorytmu w układzie FPGA jest stosunkowo duże zapotrzebowanie na zasoby sprzętowe. Należy zarezerwować miejsce na cechy każdego potencjalnego obiektu. Ogranicza to liczbę możliwych etykiet. Zarezerwowanie miejsca dla 30 obiektów nie przekroczyło możliwości układu Zybo. Badania pokazały, że jest to wystarczająca liczba etykiet do sprawnego działania toru wizyjnego.

3.4. Komunikacja układu ZYBO Z7-20 ze sterownikiem Pixhawk

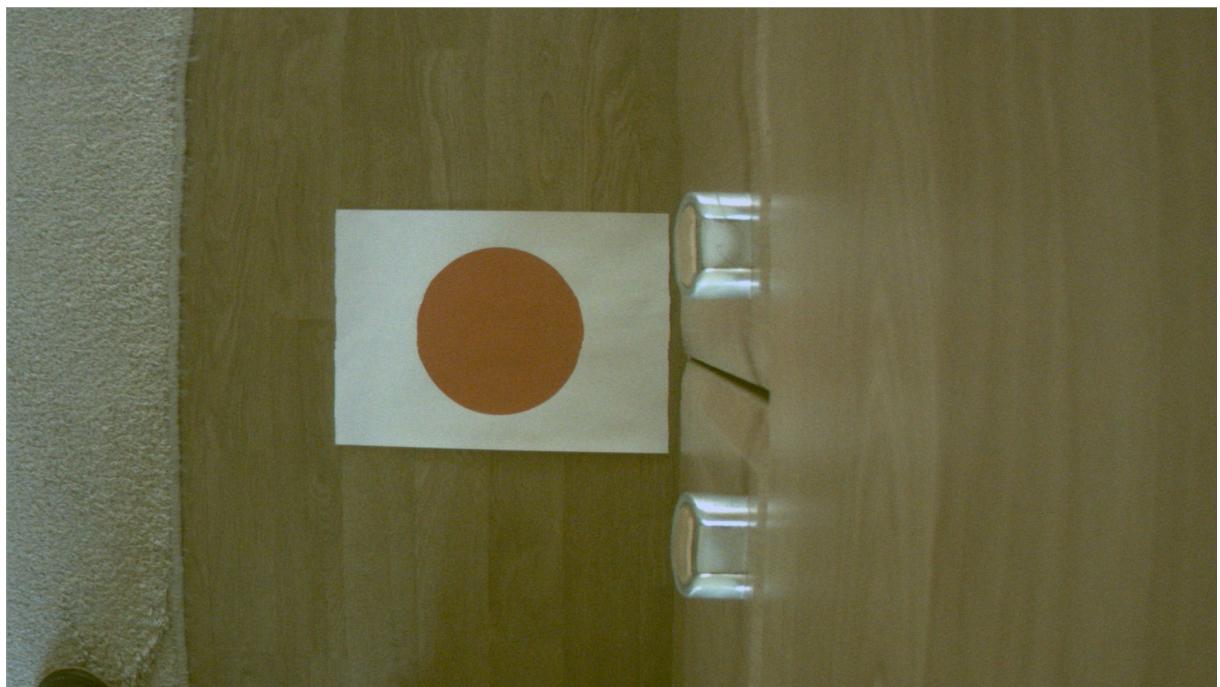
— Tutaj chciałbym opisać protokół Mavlink używany do komunikacji ze sterownikiem. Obecnie została nawiązana komunikacja i udało się wysłać komendę uzbrojenia drona w trybie STABILIZE. W najbliższym czasie planuję przeprowadzić test wysyłania komend ruchu w trybie GUIDED, który wymaga połączenia GPS.

4. Testy i ocena wyników

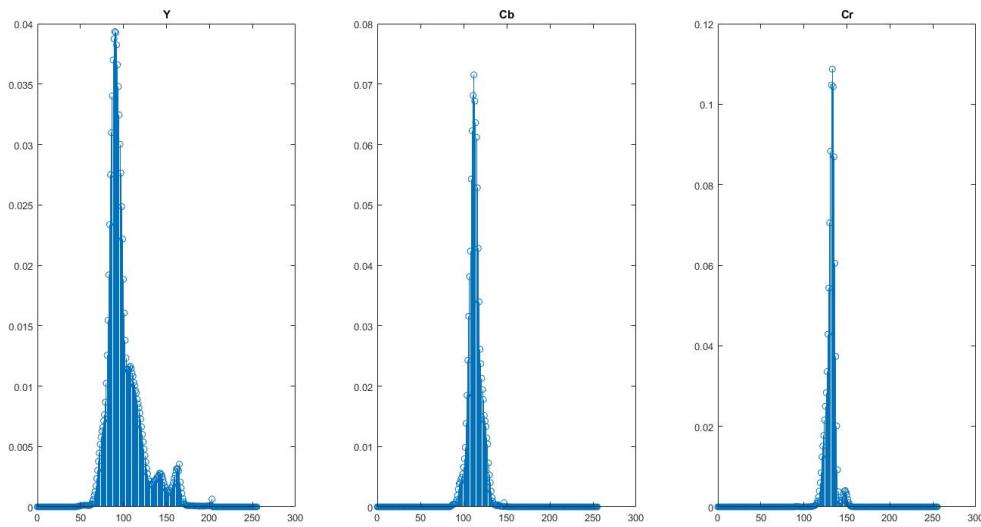
4.1. Test wykrywania koloru znacznika w przestrzeni barw YCbCr

W celu zbadania możliwości wykrycia koloru znacznika przygotowano 3 fotografie wykonane kamerą PCAM 5C. Każde ze zdjęć przedstawia czerwone koło (ksztalt znacznika był przy tym teście mniej istotny) i zostało zrobione przy innym oświetleniu.

— Tutaj planuję opis poniższych zdjęć (szczególnie histogramów) wraz z wnioskami, że możliwa jest skuteczna detekcja czerwonego znacznika na podstawie składowej Cr.



Rys. 4.1. Obraz wykonany przy najmniejszym z badanych oświetleń



Rys. 4.2. Histogram obrazu wykonanego przy najmniejszym z badanych oświetleń

4.2. Wyznaczenie kąta widzenia kamery przy różnych ustawieniach rozdzielczości

Do zbadania kąta widzenia kamery wykorzystano moduł wyznaczający prostopadłe linie przechodzące przez środek obrazu. Następnie umieszczono kamerę na wysokości 49 cm i skierowano ją w dół. Odczytanie odległości od środka obrazu do jego krawędzi w poziomie i pionie pozwoliło na wyznaczenie kątów widzenia kamery. Skorzystano ze wzoru 4.1.

$$\alpha = \arctan \frac{l}{h} \quad (4.1)$$

gdzie:

α – kąt widzenia kamery,

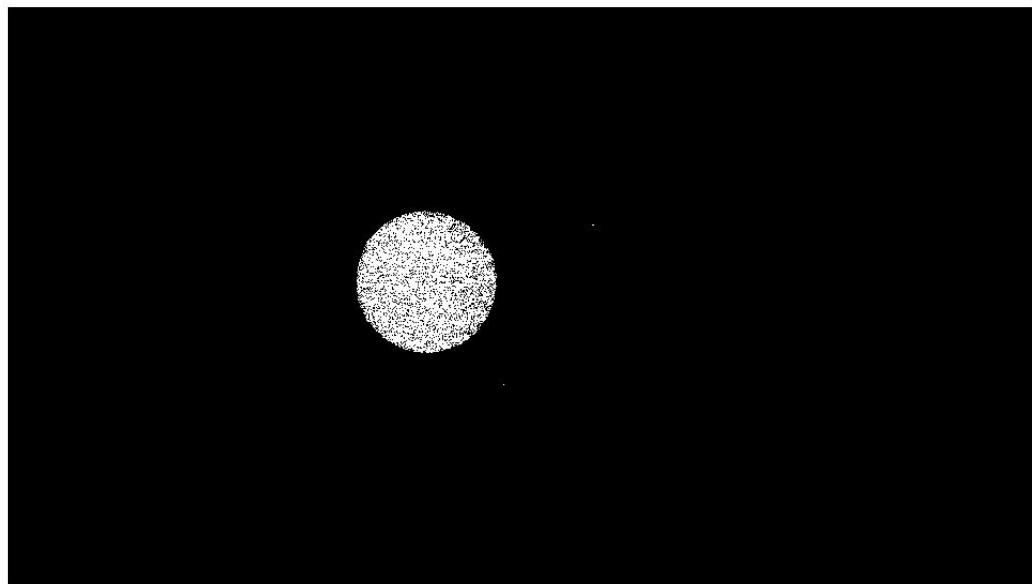
h – wysokość, na jakiej umieszczona jest kamera,

l – odległość środka obrazu od jego krawędzi.

Na podstawie obrazu przedstawionego na rysunku 4.10 obliczono kąty widzenia kamery w pionie i poziomie dla rozdzielczości 1920 x 1080. Dla rozdzielczości 1280 x 720 skorzystano z obrazu pokazanego na rysunku 4.11. Korzystając ze wzoru 4.1 otrzymano przybliżone wyniki, przedstawione w tabeli 4.1.

4.3. Test detekcji znacznika na obrazie

Na rysunku 4.12 przedstawiono zdjęcie znacznika wykonane z wysokości jednego metra. Zastosowanie binaryzacji (rysunek 4.13) pozwoliło na oddzielenie znacznika od tła. Na obrazie znajdują się



Rys. 4.3. Binaryzacja obrazu wykonanego przy najmniejszym z badanych oświetleń

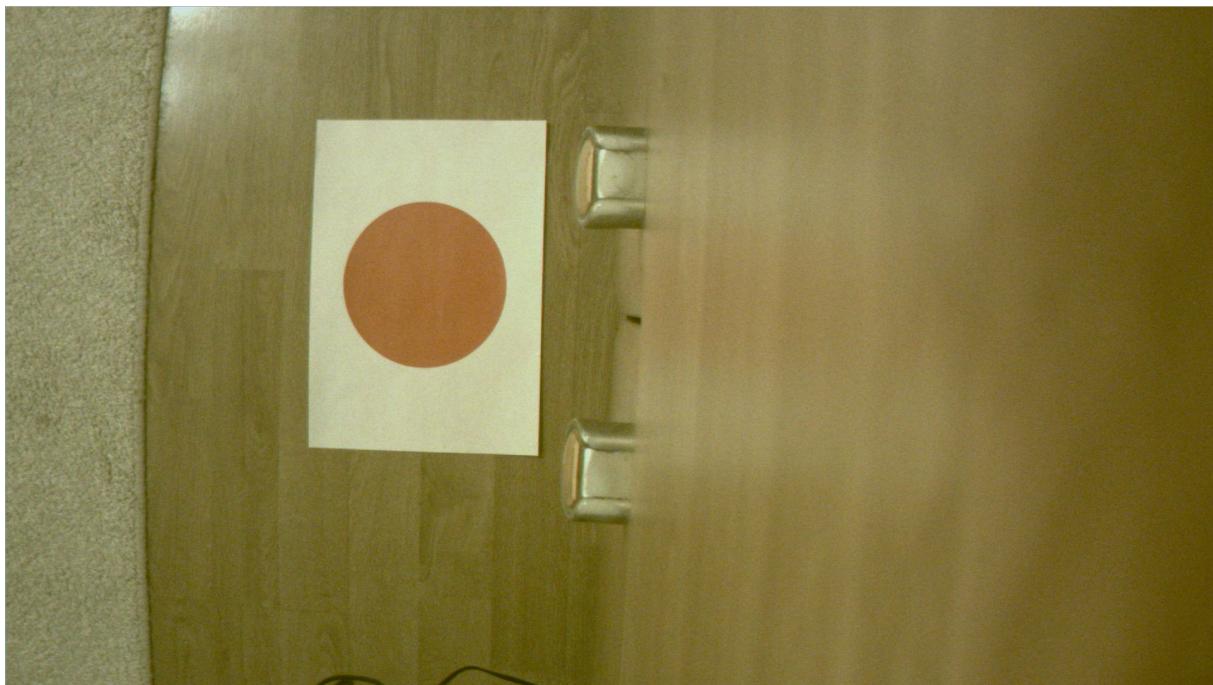
Tabela 4.1. Wartości kąta widzenia kamery w zależności od rozdzielczości

Rozdzielczość	Kąt widzenia kamery w pionie w stopniach	Kąt widzenia kamery w poziomie w stopniach
1920 x 1080	14	27
1280 x 720	19	38

jednak inne niewielkie obszary, które również uzyskały kolor biały. Wykorzystanie mediany spowodowało znaczne zmniejszenie tych obszarów, jednakże na rysunku 4.14 widoczne są nadal małe grupy białych pikseli nienależących do markera. Zastosowanie morfologicznego otwarcia, składającego się z erozji i dylatacji, pozwoliło na całkowite wyeliminowanie błędnych białych pikseli. Na koniec wykorzystano również operację zamknięcia, polegającą na wykonaniu najpierw dylatacji, a potem erozji. Na rysunkach 4.15 i 4.16, przedstawiających rezultaty kolejno otwarcia i zamknięcia, nie widać większych różnic. W pewnych sytuacjach jednak otwarcie prowadzi do podziału obiektu na kilka części. W takich przypadkach wykonanie zamknięcia może ponownie połączyć obiekty. Może to być istotne w dalszych pracach nad projektem przy wykorzystaniu indeksacji.

4.4. Test regulacji położenia drona

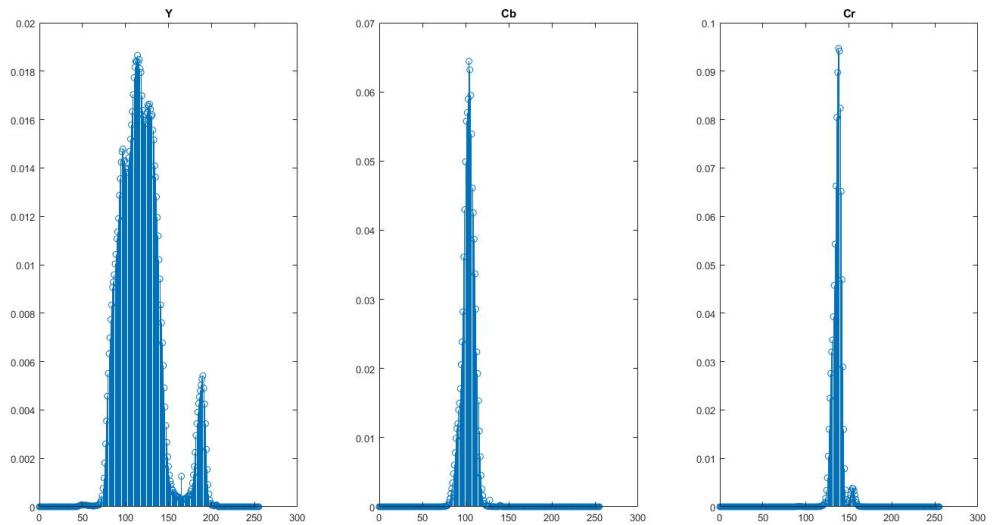
— Test będzie polegał na zadawaniu prędkości w x i y proporcjonalnej do liczby pikseli dzielających środek obrazu od aktualnego położenia znacznika. Celem będzie doprowadzenie do sytuacji stabilnego unoszenia drona nad znacznikiem. Rejestrowanie uchybu w x i y i ich zapis na kartę SD pozwoli na sporządzenie wykresów tych wartości w czasie.



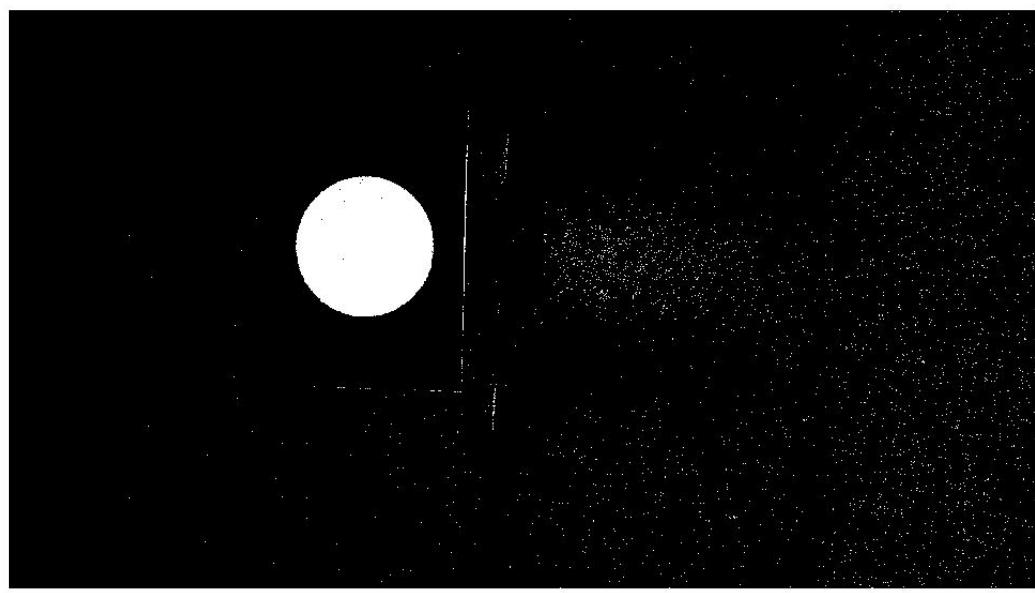
Rys. 4.4. Obraz wykonany przy średnim oświetleniu

4.5. Lądownie na nierzuchomym lądowisku

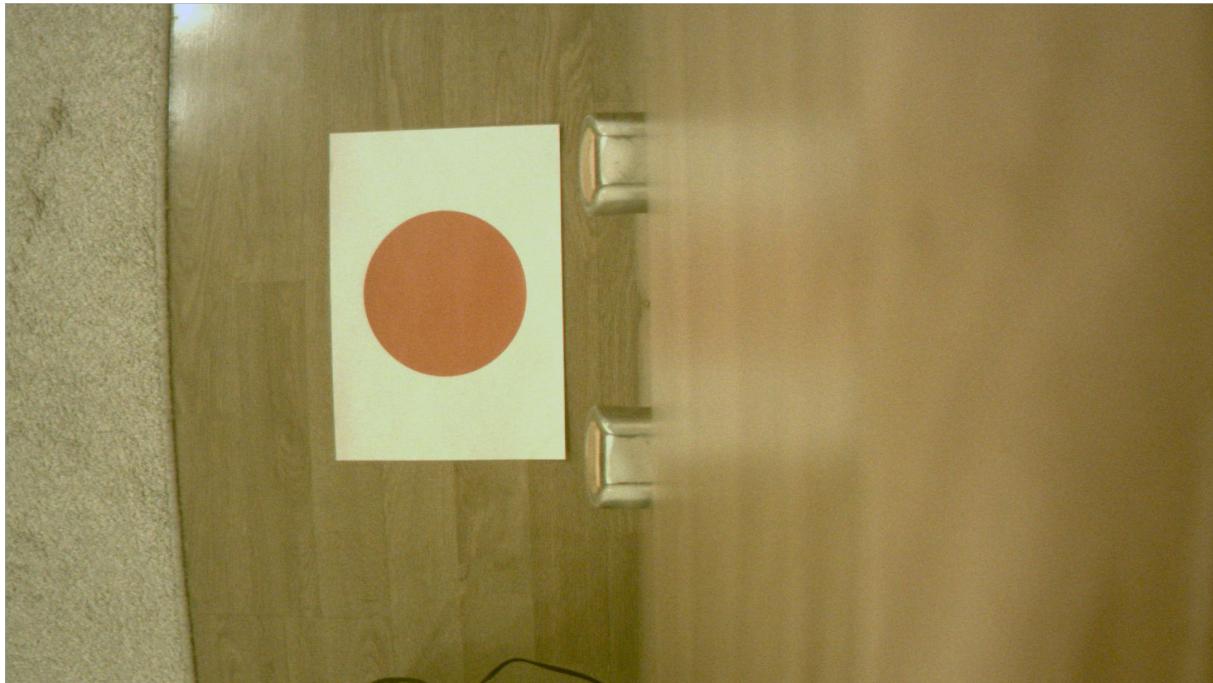
— Do poprzedniego testu zostanie dodana funkcjonalność lądownia.



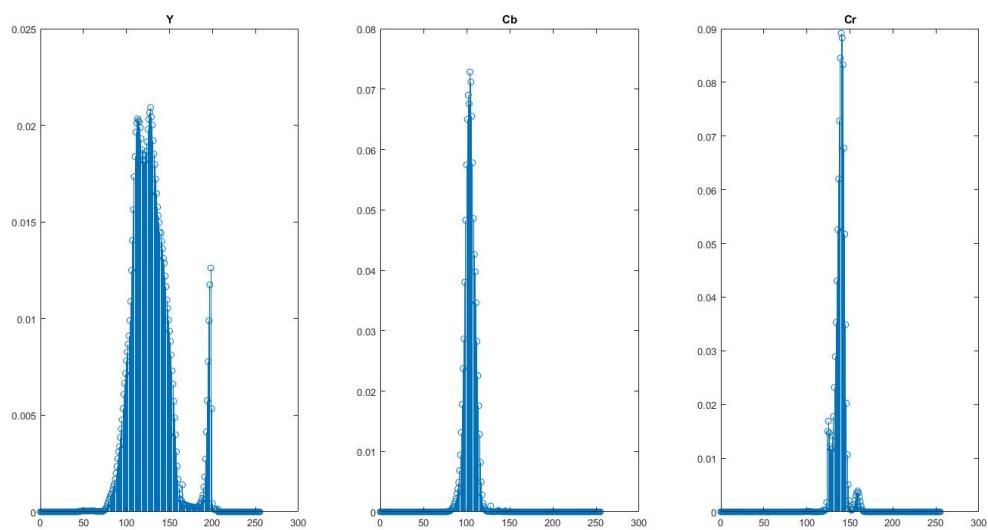
Rys. 4.5. Histogram obrazu wykonanego przy średnim oświetleniu



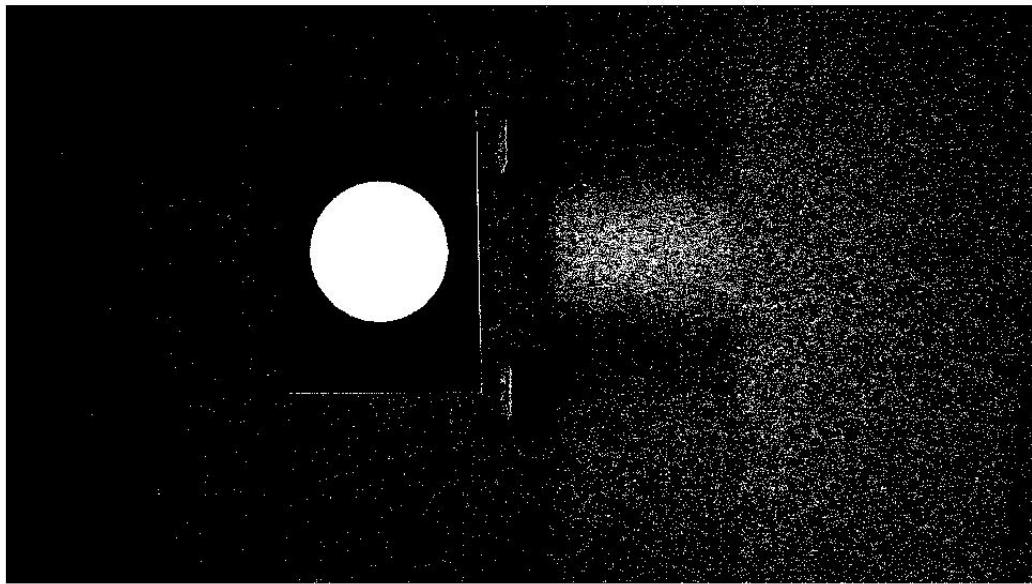
Rys. 4.6. Binaryzacja obrazu wykonanego przy średnim oświetleniu



Rys. 4.7. Obraz wykonany przy największym z badanych oświetleń



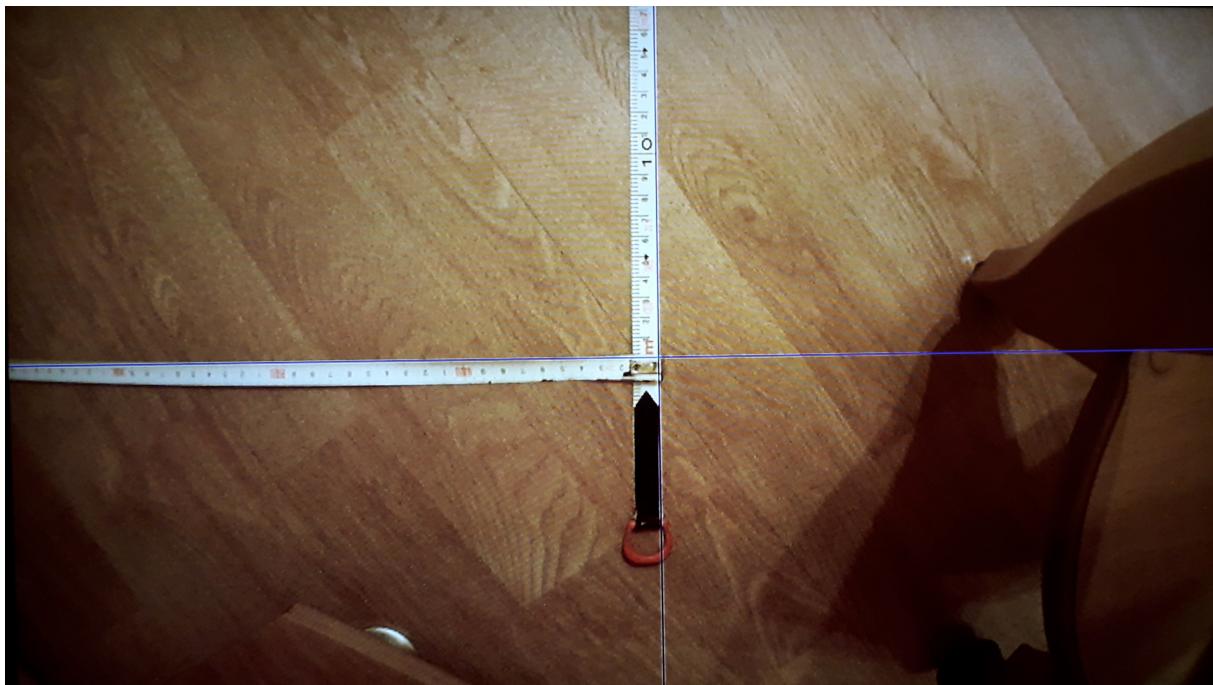
Rys. 4.8. Histogram obrazu wykonanego przy największym z badanych oświetleń



Rys. 4.9. Binaryzacja obrazu wykonanego przy największym z badanych oświetleń



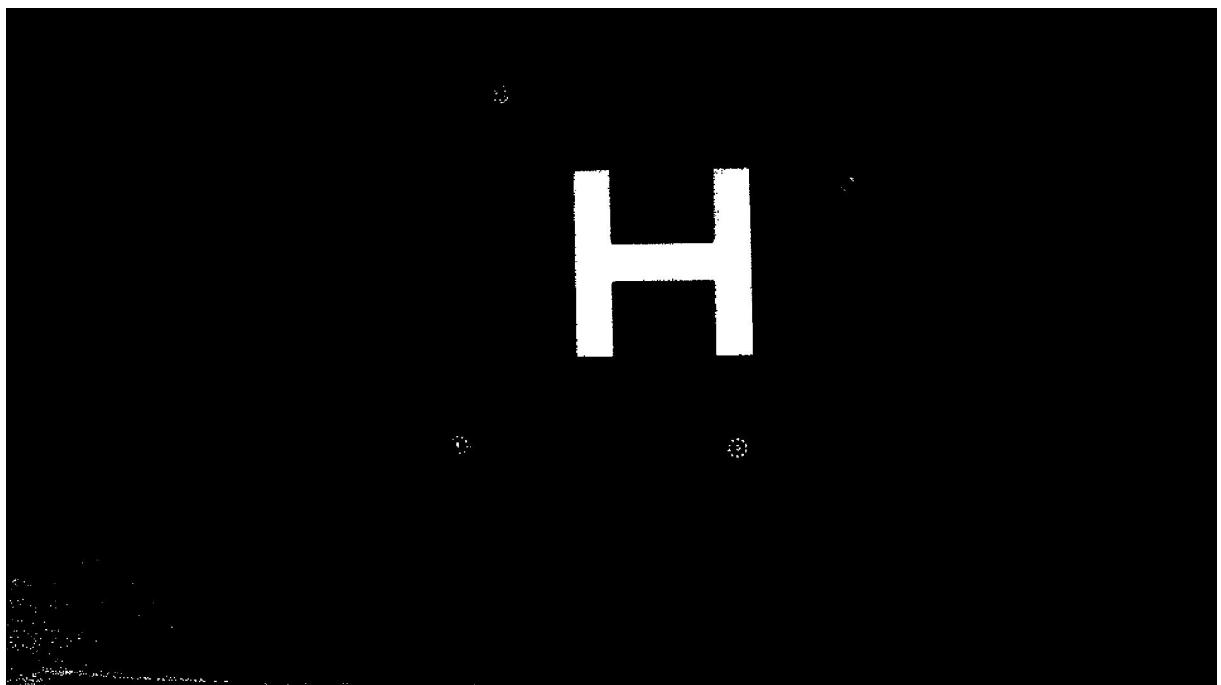
Rys. 4.10. Obraz służący do wyznaczenia kąta widzenia kamery dla rozdzielczości 1920 x 1080.



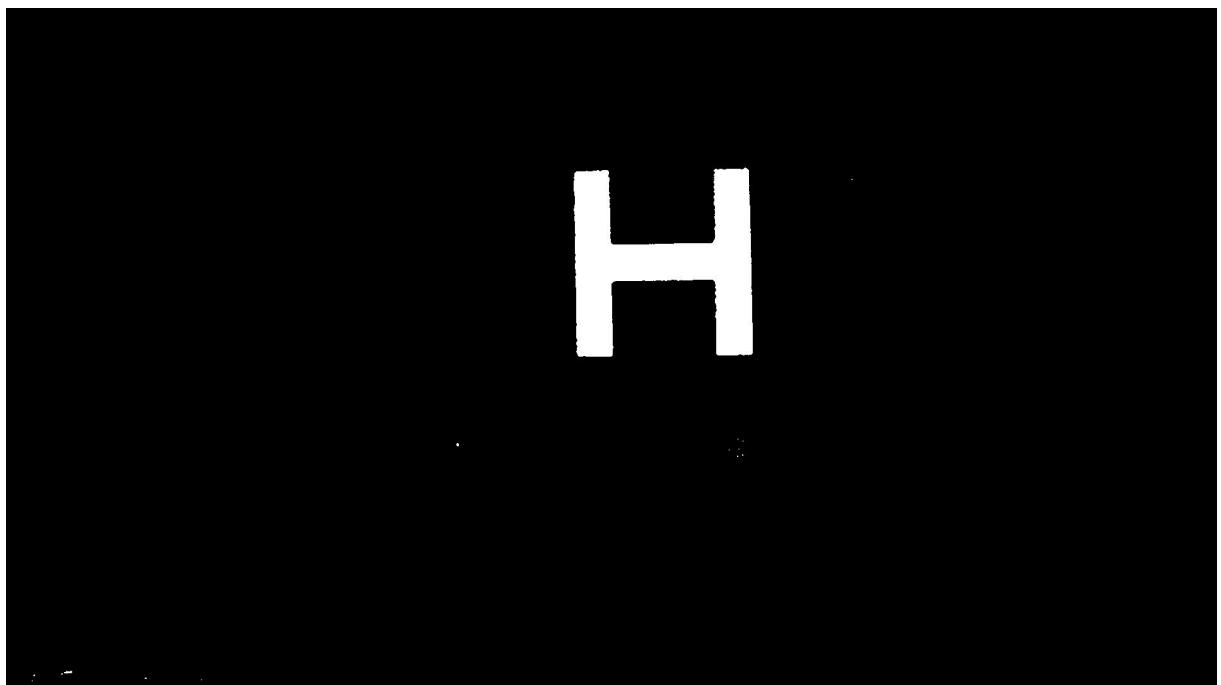
Rys. 4.11. Obraz służący do wyznaczenia kąta widzenia kamery dla rozdzielczości 1280 x 720.



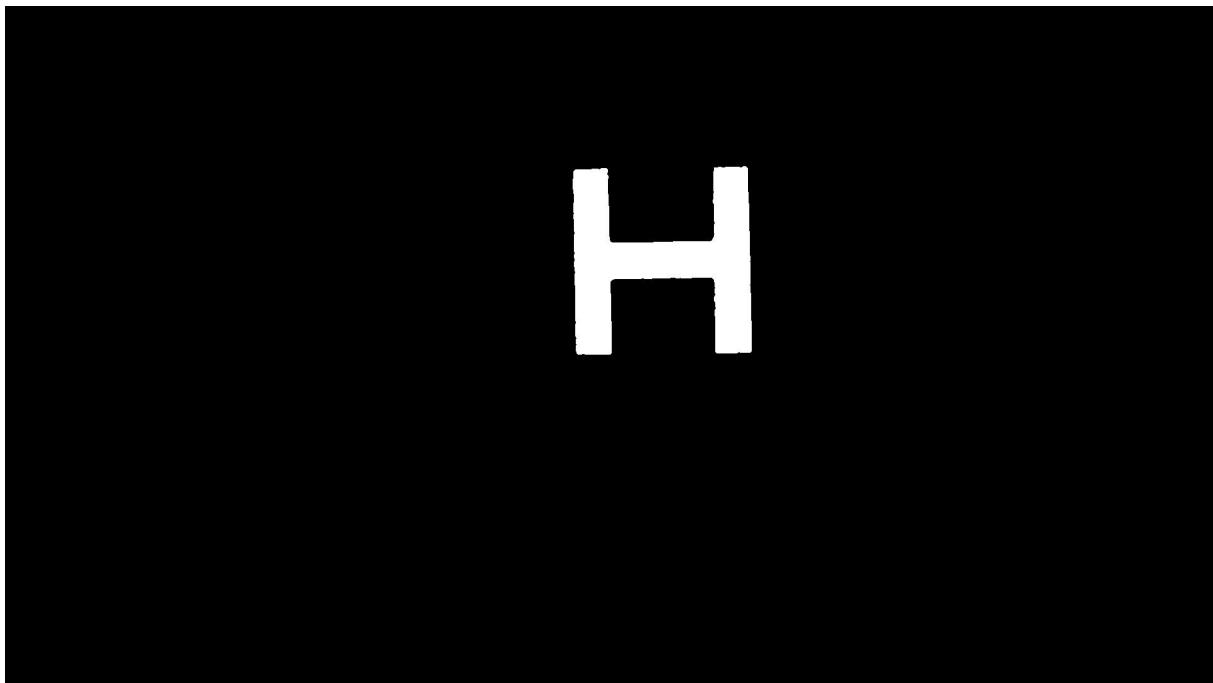
Rys. 4.12. Obraz oryginalny



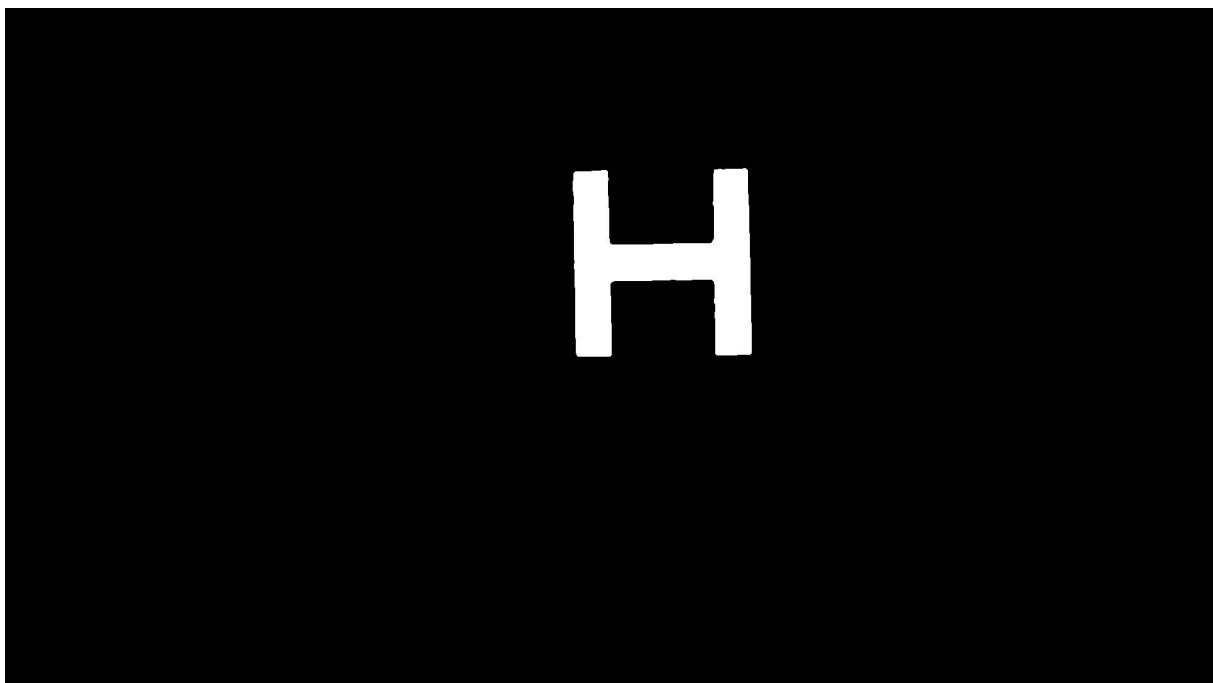
Rys. 4.13. Obraz po binaryzacji



Rys. 4.14. Obraz po medianie



Rys. 4.15. Obraz po otwarciu



Rys. 4.16. Obraz po zamknięciu

5. Podsumowanie i kierunki dalszych prac

Bibliografia

- [1] Sven Lange i inni. *A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments*. Munich, Germany: IEEE, 2009.
- [2] Yang Fan, Shi Haiqing i Wang Hong. „A Vision-Based Algorithm for Landing Unmanned Aerial Vehicles”. W: *2008 International Conference on Computer Science and Software Engineering*. INSPEC Accession Number: 10426910. Hubei, China: IEEE, 2008, s. 994.
- [3] A. Price i inni. *Real time object detection for an unmanned aerial vehicle using an FPGA based vision system*. Hubei, China: IEEE, 2008.
- [4] Davide Falanga i inni. *Vision-based Autonomous Quadrotor Landing on a Moving Platform*. Shanghai, China: IEEE, 2017.
- [5] Yang Fan, Shi Haiqing i Wang Hong. „A Vision-Based Algorithm for Landing Unmanned Aerial Vehicles”. W: *2008 International Conference on Computer Science and Software Engineering*. INSPEC Accession Number: 10426910. Hubei, China: IEEE, 2008, s. 995.
- [6] Vidya Sudavan i inni. *Vision based autonomous landing of an Unmanned Aerial Vehicle on a stationary target*. Jeju, South Korea: IEEE, 2017.
- [7] <https://reference.digilentinc.com/learn/programmable-logic/tutorials/zybo-z7-pcam-5c-demo/start>. Dostęp: 2019-11-03.
- [8] http://elm-chan.org/fsw/ff/00index_e.html. Dostęp: 2019-11-03.
- [9] Abdul Malik i in. „Real-time Component Labelling with Centre of Gravity Calculation on FPGA”. W: (sty. 2011).