



A G H

AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

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

Projekt dyplomowy

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, 2020

*Serdecznie dziękuję mojemu promotorowi
Panu Dr. inż. Tomaszowi Kryjakowi oraz innym
pracownikom Katedry Automatyki i Robotyki
za wsparcie merytoryczne i okazaną pomoc.*

Streszczenie

W projekcie zaprezentowano sprzętowo-programowy system wizyjny wspomagający autonomiczne lądowanie drona. System składa się z kamery, laserowego czujnika wysokości oraz heterogenicznego układu obliczeniowego Zynq SoC. Na podstawie wyznaczonego położenia lądowiska oraz aktualnej wysokości generowane jest sterowanie, które następnie przekazywane jest do kontrolera drona. Obliczenia podzielone zostały pomiędzy logikę reprogramowalną i system procesorowy układu Zynq. Wykonane testy potwierdziły możliwość przeprowadzenia lądowania na statycznym lądowisku.

Słowa kluczowe: autonomiczne lądowanie drona, system wizyjny, Zynq SoC, lidar, Pixhawk

Abstract

The project presents a hardware-software vision system supporting the autonomous landing of the drone. It consists of a camera, laser based altitude sensor and a heterogeneous Zynq SoC computing platform. Based on the determined landing position and current altitude, control signals are generated, which are then transmitted to the drone controller. The calculations were divided between the reprogrammable logic and the processor system of the Zynq device. The conducted tests confirmed the possibility of landing on a static landing pad.

Keywords: autonomous drone landing, vision system, Zynq SoC, lidar, Pixhawk

Spis treści

1. Wprowadzenie	9
1.1. Cele pracy	10
1.2. Zawartość pracy.....	10
2. Metody przetwarzania obrazu i sposoby sterowania w procesie autonomicznego lądowania drona	11
3. Sprzęt wykorzystany w projekcie	15
3.1. Platforma obliczeniowa	15
3.2. Moduł rejestrujący obraz	17
3.3. Platforma statku powietrznego	17
3.4. Sterownik drona.....	18
3.5. Laserowy czujnik wysokości	19
4. Implementacja i ewaluacja modelu programowego	21
4.1. Opis zastosowanych operacji przetwarzania wizyjnego.....	21
4.2. Wybór znacznika	22
4.2.1. Kształt	23
4.2.2. Kolor	23
4.3. Analiza kąta widzenia kamery przy różnych ustawieniach rozdzielczości	27
4.4. Podsumowanie	29
5. Implementacja i ewaluacja systemu sprzętowo-programowego.....	31
5.1. Akwizycja obrazu	32
5.1.1. Zapis ramek na kartę SD	32
5.2. Podłączenie laserowego czujnika wysokości	33
5.3. Konwersja z przestrzeni barw RGB do YCbCr	33
5.4. Binaryzacja	33
5.5. Mediana	33
5.6. Otwarcie	35
5.7. Indeksacja	35

5.8. Wyznaczanie środka ciężkości	38
5.8.1. Wyznaczanie prostokąta otaczającego i pola powierzchni	39
5.9. Algorytm sterowania	39
5.10. Integracja płytki z autopilotem	40
5.11. Ewaluacja.....	40
5.11.1. Test sterowania.....	40
5.11.2. Test reakcji autopilota na zadawane komendy	40
5.12. Podsumowanie.....	41
6. Podsumowanie	43
A. Spis zawartości płyty CD.....	47

1. Wprowadzenie

W ostatnich latach zaprezentowano szereg przykładów, które pokazują wszechstronność i przydatność dronów w wielu dziedzinach przemysłu. Według raportu Skyward z 2018 roku, 1 na 10 przebadanych firm w Stanach Zjednoczonych używała bezzałogowych statków powietrznych. Aż 88 procent z nich odczuła pozytywne strony rozpoczęcia korzystania z nich w przeciągu roku lub krócej. Do najczęściej wymienianych zalet dronów należy zdobywanie większej ilości informacji, bardziej efektywna praca oraz oszczędność czasu. Udział dronów w rynku będzie się stale powiększał, gdyż 3 na 4 przedsiębiorców planuje zwiększać wydatki przeznaczane na operacje wykonywane przez drony [1]. O wzroście zainteresowania dronami może również pośrednio świadczyć fakt wprowadzania nowych regulacji prawnych – konieczności wyposażania dronów w nadajnik numeru identyfikacyjnego [2]. Zwiększa to bezpieczeństwo lotów i może otworzyć drogę do masowego wykorzystania tej technologii w branżach takich jak:

- film i fotografia lotnicza – możliwość wykonywania ujęć z powietrza,
- wirtualna rzeczywistość – tworzenie skanów 3D przestrzeni,
- geodezja – rejestrowanie danych dla tworzenia modeli terenu,
- rolnictwo – stosowanie środków ochrony roślin, monitoring upraw,
- ochrona osób i mienia – prowadzenie nadzoru z powietrza,
- ochrona środowiska – monitorowanie zanieczyszczeń.

Ważnym kierunkiem rozwoju bezzałogowych statków powietrznych jest ich autonomizacja, czyli przystosowanie do lotów bez nadzoru człowieka. Wymaga to wyposażenia dronów w systemy monitorowania otoczenia oraz odpowiednie algorytmy sterowania. Kluczową fazą autonomicznego lotu drona jest lądowanie. Bliskość ziemi wymaga dokładnego i szybkiego sterowania ruchem statku powietrznego. W wielu przypadkach podstawowym czujnikiem jest system wizyjny. Generowany przez niego strumień obrazu powinien być przetwarzany szybko i efektywnie, co stanowi duże wyzwanie. Jedną z możliwych do zastosowania platform obliczeniowych, która spełnia wspomniane wymagania, są reprogramowalne układy heterogeniczne np. Zynq SoC (ang. *System on Chip*).

1.1. Cele pracy

Celem pracy było stworzenie sprzętowo-programowego systemu wizyjnego, będącego komponentem systemu autonomicznego lądowania drona. Pierwszym krokiem prac było przeprowadzenie analizy literatury naukowej – głównie dotyczącej detekcji i śledzenia oznaczonego lądowiska. W drugim etapie należało wykonać model programowy algorytmu, który pozwala na wykrycie i podążanie w kierunku lądowiska. Wybór oznaczenia miejsca do lądowania był również częścią prac. Założeniem było wyposażenie drona w kamerę o osi optycznej skierowanej prostopadle do podłoża i wykonanie lądowania na statycznej platformie. Należało również ocenić możliwość wykonania lądowania na ruchomym celu. Ponadto należało przygotować komponent umożliwiający pomiar wysokości – laserowy czujnik wysokości Garmin Lidar v3. W trzecim etapie należało wspomniany system podzielić na część sprzętową i programową oraz zaimplementować w układzie Zynq SoC na karcie ZYBO Z7-20. Wejściem powinien być obraz z kamery PCAM oraz informacja z wysokościomierza, a wyjściem sterowanie dla drona – regulacja położenia względem lądowiska oraz wysokości. Ostatnim etapem prac była próba integracji wykonanego podsystemu ze sterownikiem drona i wykonania lądowania w sposób autonomiczny. Testy należało przeprowadzić przy zachowaniu zasad bezpieczeństwa – kluczowe było upewnienie się co do niezawodności działania systemu.

1.2. Zawartość pracy

W rozdziale 2 zamieszczono przegląd prac naukowych dotyczących problematyki autonomicznego lądowania drona. Rozdział 3 opisuje sprzęt wykorzystany podczas realizacji projektu. W rozdziale 4 przedstawiono opis modelu programowego wraz z uzasadnieniem wyboru poszczególnych modułów, dyskusję na temat wyboru wyglądu znacznika, a także analizę kąta widzenia kamery. Rozdział 5 poświęcony implementacji sprzętowo-programowej całego systemu, począwszy od odbioru sygnału wizyjnego, a skończywszy na wysyłaniu sygnałów sterujących dronem. Część ta obejmuje również ewaluację komunikacji z autopilotem oraz test „ręcznego” sterowania na podstawie informacji ze stworzonego systemu. W ostatnim rozdziale zamieszczono podsumowanie prac oraz przedstawiono możliwości dalszego rozwoju projektu.

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

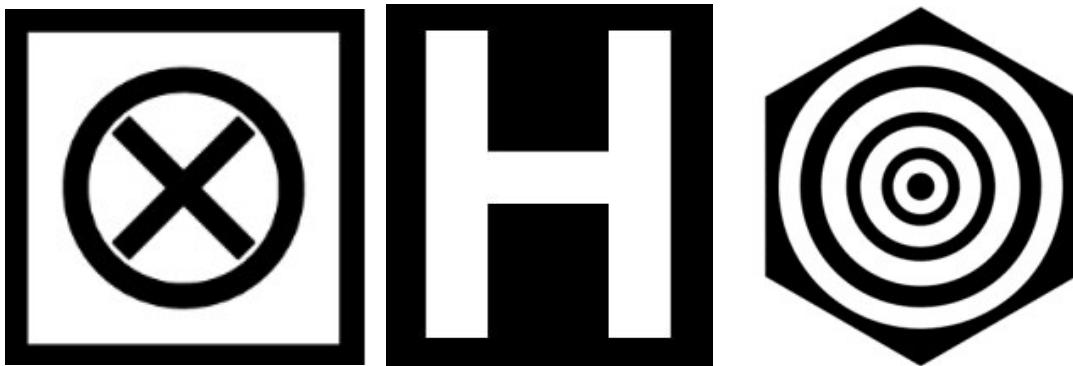
Wykonywanie różnorodnych scenariuszy misji przez drona wymaga osiągnięcia określonego poziomu autonomii. Bezzałogowy statek powietrzny musi być w stanie wystartować, nawigować oraz wyładować bez bezpośredniego udziału człowieka. Przemieszczanie drona pomiędzy różnymi punktami nawigacyjnymi przebiega względnie bezproblemowo, kiedy dostępny jest sygnał GPS. Kłopotu nie sprawia również start wykonywany bez pomocy operatora (z ustalonego lądowiska, przy braku przeskódekerenowych). Potencjalnie największe wyzwanie stanowi autonomiczne lądowanie drona. Platforma obliczeniowa wraz z zespołem sensorów zainstalowana na statku powietrznym musi być w stanie wykryć lądowisko oraz wysłać odpowiednie sygnały sterujące. Efektem działania algorytmu sterującego musi być zajęcie dogodnej pozycji do obniżenia lotu i ostatecznie przyziemienie w określonym punkcie. Zadania autonomicznego lądowania drona można podzielić na dwie kategorie:

- lądowanie na platformie pozostającej w bezruchu względem ziemi,
- lądowanie na poruszającej się platformie.

W przypadku lądowania na oznaczonym, stacjonarnym lądowisku, głównym problemem jest skuteczna detekcja znacznika oraz jego śledzenie. Do realizacji tego zadania niezbędne jest wykonanie sześciu kroków. Przykładowe rozwiązanie z pracy [3] obejmuje:

- zamianę obrazu kolorowego (RGB) na obraz w skali szarości,
- binaryzację ze stałym progiem,
- indeksację,
- odrzucenie obiektów o liczbie pikseli mniejszej niż zadany próg,
- identyfikację znacznika.

Wartości progów binaryzacji oraz odrzucenia małych obiektów zostały dobrane eksperymentalnie. Aby dodatkowo wyeliminować biały szum oraz zakłócenia typu „sól i pieprz”, w pracy [4] wprowadzono operację mediany z oknem 3x3 na obrazie w skali szarości. Dodatkowo zastosowano ograniczenie na maksymalny rozmiar obiektu. W algorytmie przedstawionym w pracy [5] binaryzacji dokonano



(a) Znacznik użyty w pracy [6] (b) Znacznik użyty w pracy [7] (c) Znacznik użyty w pracy [3]

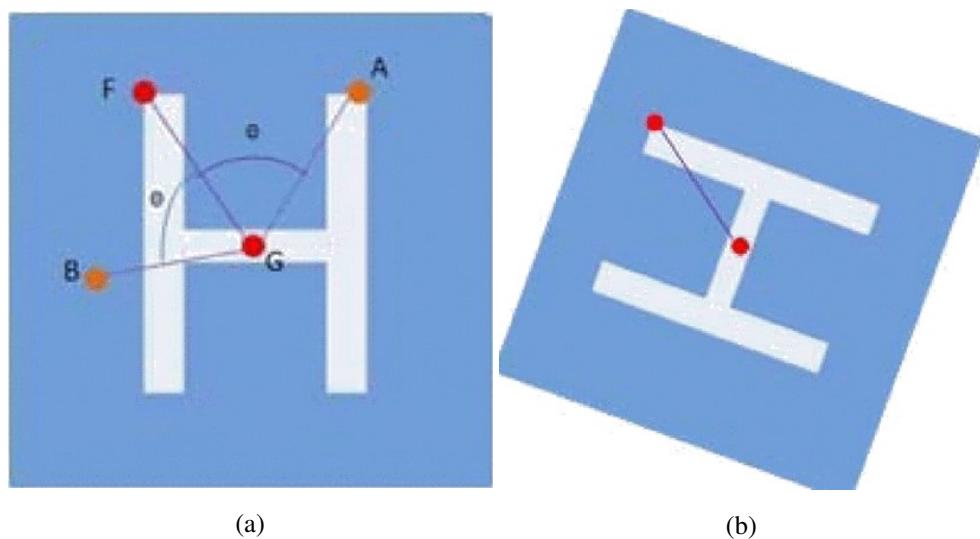
Rys. 2.1. Porównanie znaczników użytych w pracach [3],[6] i [7]

w przestrzeni barw HSV (ang. *Hue Saturation Value*), na podstawie składowych S i V. Pozwoliło to na przeprowadzenie segmentacji niezależnej od koloru (barwy). Przed rozpoznawaniem kształtów, na obrazie binarnym dokonano kolejno erozji oraz dylatacji, co pozwoliło na eliminację z obrazu małych grup pikseli.

Istotnym elementem planowania autonomicznego lądowania drona jest wybór znacznika lądowiska. W pracy [6] przedstawiono marker złożony z trzech figur: kwadratu, koła i krzyża. Znacznik pokazano na rys. 2.1a. Zaimplementowany został algorytm detekcji lądowiska, który bazował na kolejnym wykrywaniu wymienionych figur, co pozwoliło na coraz lepsze określenie położenia celu.

Prostsze rozwiązanie zostało pokazane w pracy[7], gdzie znacznik ma kształt litery **H** (Rys. 2.1b) Odległość drona od lądowiska obliczana była 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, a następnie wykreśnięciu linii przechodzącej przez te dwa punkty (punkty F i G na rys. 2.2a). Otrzymana linia nie określa jednak pozycji znacznika w sposób jednoznaczny. Do określenia, czy zachodzi przypadek przedstawiony na rys. 2.2a, czy też 2.2b, obliczono kąt θ pomiędzy dwoma narożnymi punktami (F i A) markera. Kąt obliczany jest na podstawie zapisanego obrazu znacznika. Następnie wykorzystując znalezioną wcześniej linię, obliczane są współrzędne tych dwóch punktów (punkty A i B na rys. 2.2a). Kolejną czynnością jest sprawdzenie, który z punktów leży bliżej obszaru markera, czyli litery **H**. Jeśli jest to punkt A – zachodzi przypadek z rys. 2.2a i dron ustawiony jest w osi znacznika, jeśli B –rys. 2.2b.

Inny znacznik został użyty w pracy [3]. Składa się on z czterech pierścieni na czarnym tle (rys. 2.1c) Każdy pierścień posiada unikalny stosunek promienia wewnętrznego do zewnętrznego i stanowi oddzielny obiekt dla systemu wizyjnego. Obiekty, które nie mają dokładnie jednej „dziury” wewnętrz, 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.

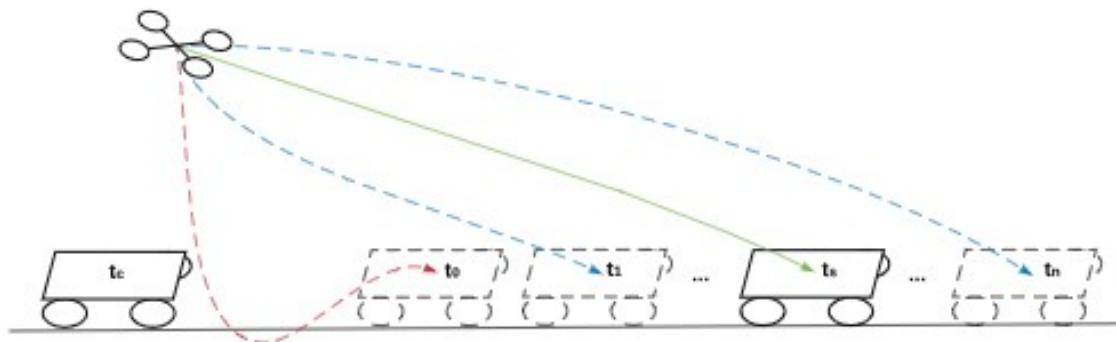


Rys. 2.2. Ilustracja określania pozycji znacznika w pracy [7]

W procesie autonomicznego lądowania drona, na podstawie wyznaczonej odległości od celu, do bezzałogowego statku powietrznego wysyłane są sygnały sterujące. W pracy [8], do kontroli prędkości drona, wykorzystano trzy regulatory PID. Dzięki nim równocześnie minimalizowana jest każda z trzech składowych wektora położenia. W pracy [3] zwrócono uwagę na błędy spowodowane pochyleniem i przekrętem drona. Przed wysłaniem sygnału do regulatora wykonywana jest korekta uchybu na podstawie pomiaru kąta pochylenia i przekrętu. Krok ten był potrzebny z uwagi na sztywne umieszczenie kamery na ramie drona (brak tzw. gimbalu).

W przypadku wykonywania lądowania na poruszającej się platformie, do opisanego wcześniej problemu detekcji znacznika dochodzi również konieczność implementacji bardziej skomplikowanego algorytmu sterowania oraz bardziej niezawodnego śledzenia położenia lądowiska. Wysyłane sygnały sterujące muszą uwzględniać ruch lądowiska. Przemieszczanie drona powinno nadążać za zmianą położenia platformy. W takim przypadku układ regulacji jest układem śledzącym.

W [6] użyto algorytmu pozwalającego na predykcję zachowania celu. Wykorzystano dynamiczny model celu oraz rozszerzony filtr Kalmana. Na rys. 2.3 przedstawiono schemat wyboru trajektorii. System planuje kilka możliwych dróg dotarcia do lądowiska. Każda z nich ma początek w aktualnej pozycji drona, a kończy się w przewidzianym położeniu platformy. Przyszły stan poruszającego się celu jest określany na podstawie jego modelu dynamicznego, począwszy od ostatniej dostępnej estymaty z filtra Kalmana. Trajektorie wymagające sterowania spoza dostępnego zakresu lub kolidujące z przeszkodami, są odrzucone (trajektoria oznaczona czerwoną przerywaną linią). Spośród wszystkich możliwych dróg (niebieskie przerywane linie) wybierana jest ta wymagająca użycia najmniejszej ilości energii. Zadanie optymalizacji jest wykonywane przy wykorzystaniu szybkiej wielomianowej generacji trajektorii minimalizującej trzecią pochodną położenia.



Rys. 2.3. Przykład wybierania trajektorii z pracy [6].

Podsumowując, wykonanie lądowania na statycznym lądowisku wymaga skutecznej detekcji znacznika znajdującego się na platformie, który określa jego lokalizację. Znalezienie względnej pozycji markera umożliwia przekazanie informacji o uchybie do regulatora sterującego dronem. Do urządzenia wykonawczego, czyli silników, dociera ostatecznie informacja o pożądanej wartości ciągu, która spowoduje zbliżenie do celu. Po ustabilizowaniu unoszenia drona nad lądowiskiem następuje polecenie stopniowego obniżania lotu, aż do zetknięcia z ziemią. W przypadku lądowania na ruchomym celu nie ma możliwości doprowadzenia do zawisu nad platformą. Należy zatem przeprowadzić predykcję zachowania lądowiska, wygenerować możliwe trajektorie i użyć jednej z metod optymalizacji do wyboru najlepszej z nich.

3. Sprzęt wykorzystany w projekcie

3.1. Platforma obliczeniowa

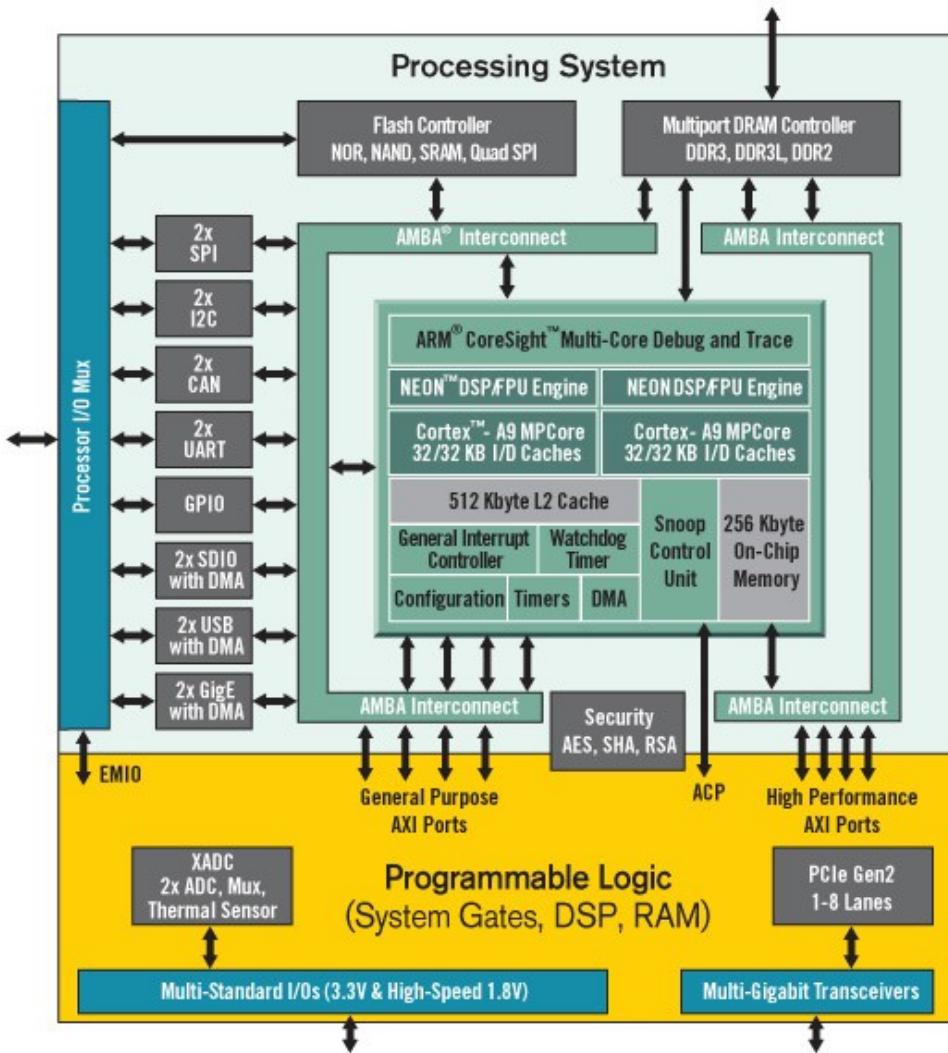
W pracy wykorzystano platformę Digilent ZYBO Z7-20 z układem Zynq SoC (ang. *System on Chip*) XC7Z020-1CLG400C (rys. 3.1). Układ dostępny na karcie określa się jako heterogeniczny tj. stanowi połączenie części rekonfigurowalnej (PL – ang. *programmable logic*) oraz systemu procesorowego z dwurdzeniowym procesorem ARM Cortex-A9 taktowanym z częstotliwością 667 MHz (PS – ang. *processing system*). Na rysunku 3.2 przedstawiono architekturę układu Zynq SoC. System procesorowy zaznaczony



Rys. 3.1. Karta ZYBO Z7-20 wraz z kamerą PCAM 5C.

został na zielono, natomiast część rekonfigurowalna na żółto. Część PS, oprócz dwurdzeniowego procesora ARM, zawiera wiele komponentów, między innymi:

- wydajne kontrolery 1Gb Ethernet, USB 2.0, SDIO (ang. *Secure Digital Input Output*),
- kontrolery SPI, UART, CAN, I2C,
- kontroler pamięci DDR3,
- infrastrukturę magistrali *Advanced Microcontroller Bus Architecture Interconnect* (AMBA),
- inne kontrolery peryferiów z wejściami/wyjściami multipleksowanymi do 54 dedykowanych pinów MIO (ang. *Multiplexed Input Output*),



Rys. 3.2. Architektura układu Zynq SoC [9].

- piny EMIO (ang. Extended MIO) pozwalające na podłączenie komponentów poprzez część PL.

Kontrolery peryferiów są podłączone do części PS poprzez magistralę AMBA w trybie *slave*. W ten sposób uzyskują dostęp do rejestrów odczytu/zapisu, adresowalnych w pamięci procesora. Również część rekonfigurowalna jest połączona z magistralą AMBA poprzez porty AXI jako *slave* lub *master*. Daje to możliwość szybkiej komunikacji między układem FPGA, a procesorem. Ponadto, moduły zaimplementowane w części PL mogą wywoływać przerwania w procesorze i otrzymywać dostęp DMA.

Poniżej przedstawiono komponenty części rekonfigurowalnej:

- 53 200 tablic LUT (ang. *Look-up Table*),
- 106 400 przerzutników *flip-flop*,
- 630 KB pamięci blokowej RAM,
- 6 obszarów zarządzania zegarami CMT (ang. *Clock Management Tiles*),

- konwerter analogowo-cyfrowy.

Do pozostałych części karty ZYBO Z7-20 należą między innymi:

- łącznik kamery PCAM ze wsparciem MIPI CSI-2,
- wejściowy oraz wyjściowy port HDMI,
- slot na kartę SD,
- 4 przełączniki,
- 5 diod LED,
- 6 portów Pmod (ang. *Peripheral modules*).

Porty Pmod umożliwiają podłączanie do karty dodatkowych peryferiów.

3.2. Moduł rejestrujący obraz

W projekcie wykorzystano kolorową kamerę Digilent PCAM 5C. Jest to moduł przeznaczony do użycia z dedykowanymi płytami rozwojowymi firmy Digilent. Jednym z kompatybilnych układów jest ZYBO Z7-20. Na rys. 3.1 przedstawiono kamerę podłączoną do układu. Podstawowe parametry modułu zestawiono poniżej:

- Rozdzielcość: 5 megapikseli,
- Matryca: OV5640,
- Interfejs danych: MIPI CSI-2,
- Złącze: 15-pinowe FFC.

Sposób podłączenia kamery do karty ZYBO Z7-20 przedstawione zostały w podrozdziale 5.1.

3.3. Platforma statku powietrznego

W projekcie wykorzystano dron sześciowirnikowy, znajdujący się na wyposażeniu Studenckiego Koła Naukowego AVADER. Jego budowa była częścią innego projektu [10]. Komponenty używanego bezzałogowego statku powietrznego to:

- rama DJI F550,
- śmigła o średnicy równej 22,86 cm oraz skoku 12,7 cm,
- silniki DJI 2312/960KV z kontrolerami 420 LITE,

- czterokomorowa bateria LiPo o nominalnym napięciu 14,8 V i pojemności 6450mAh,
- nadajnik radiowy FrSky Taranis X9D Plus, odbiornik FrSky X8D,
- sterownik 3DR Pixhawk.

3.4. Sterownik drona

Urządzeniem bezpośrednio komunikującym się z kontrolerami silników jest sterownik. W pracy wykorzystano urządzenie Pixhawk – popularny kontroler lotu ogólnego przeznaczenia, którego oprogramowanie jest dostępne na otwartej licencji. Zdjęcie urządzenia przedstawiono na rys. 3.3. Jego główne



Rys. 3.3. Widok na górną część sterownika Pixhawk.

parametry to:

- procesor Cortex-M4F z zegarem 168 MHz,
- czujniki: akcelerometr, żyroskop, kompas magnetyczny, barometr i zewnętrzny moduł GPS,
- interfejsy: UART, CAN, I2C, SPI,
- wejście karty SD,
- zewnętrzny przełącznik bezpieczeństwa uzbrajania,
- wielokolorowa dioda pokazująca stan pracy.

Konfiguracja sterownika jest możliwa przy użyciu programu *Mission Planner*. Jest to aplikacja przeznaczona dla stacji naziemnej umożliwiająca:

- strojenie czujników autopilota,

- monitorowanie stanu sterownika (uzbrojenie silników, orientacja statku powietznego),
- planowanie, zapisywanie i wgrywanie planów autonomicznych misji statku powietznego,
- pobieranie i analizowanie dzienników misji.

Po instalacji oprogramowania ArduPilot sterownik umożliwia pracę w różnych trybach. Dostępne są 23 tryby wbudowane, z czego 5 jest rekomendowanych [11]. Istnieją tryby wspierające różne poziomy i typy stabilizacji, a zadania wykonywane przez autopilota w każdym z nich nieco się różnią. Poniżej przedstawiono podsumowanie najważniejszych cech rekomendowanych trybów:

- *Stabilize* – pozwala operatorowi na manualne sterowanie dronem, stabilizując jego pochylenie i przechylenie,
- *Alt Hold* – kontroluje ustawienie mocy silników w celu utrzymania wysokości,
- *Loiter* – utrzymuje aktualną pozycję drona, jeśli operator nie wydaje żadnych poleceń,
- *Return-To-Launch* – dron unosi się na domyślną wysokość 15 metrów, a następnie powraca do miejsca, w którym został uzbrojony. Tryb ten wymaga sygnału GPS,
- *Auto* – dron realizuje zaprogramowaną wcześniej misję, poruszając się po zdefiniowanych punktach. Wymagany jest sygnał GPS.

Z punktu widzenia autonomicznego lądowania, odpowiednim trybem do sterowania dronem jest tryb *Guided*. Umożliwia on bowiem wysyłanie dronowi komend dotyczących zmiany szybkości i kierunku lotu. Zazwyczaj wykorzystywana jest komunikacja ze stacją naziemną drogą radiową (przy użyciu programu Mission Planner), lecz wysyłanie komend przez urządzenie umieszczone na platformie drona również jest możliwe. Komunikacja taka przebiega z wykorzystaniem protokołu MAVLink (ang. *Micro Air Vehicle Link*). Umożliwia on przesyłanie danych i komend z wykorzystaniem prawie każdego rodzaju transmisji szeregowej [12]. W tabeli 3.1 przedstawiono opis ramki protokołu. Kwestię połączenia autopilota i układu ZYBO Z7-20 opisano w podrozdziale 5.10.

3.5. Laserowy czujnik wysokości

W projekcie do pomiaru wysokości wykorzystano urządzenie Garmin Lidar Lite v3. Jego główne parametry to:

- Napięcie zasilania – 5 V,
- Zasięg pomiaru – 40 m,
- Dokładność – 2,5 cm (pomiar do 5 m),
- Wykorzystywana długość fali – 905 nm,

Tabela 3.1. Opis ramki protokołu MAVLink

Indeks bajtu	Zawartość	Wartość	Uwagi
0	Znak początku pakietu	0xFE	Wskazuje na początek nowego pakietu.
1	Długość danych	n (0-255)	Informuje o liczbie przesyłanych danych.
2	Sekwencja pakietu	0-255	Wartość zwiększana przy każdej transmisji. Umożliwia detekcję utraty pakietów.
3	ID systemu	1-255	ID systemu wysyłającego. Umożliwia odróżnienie różnych sieci.
4	ID komponentu	0-255	ID komponentu wysyłającego. Umożliwia odróżnienie różnych komponentów w jednej sieci.
5	ID wiadomości	0-255	Oznacza typ wiadomości i umożliwia jej zdekodowanie.
6-n+6	Dane	(0-255) bajtów	Zawartość wiadomości. Zależy od jej ID.
n+7-n+8	Suma kontrolna (młodszy i starszy bajt)		Umożliwia wykrycie błędów transmisji

**Rys. 3.4.** Laserowy czujnik odległości Garmin Lidar Lite v3 [13]

- Interfejsy – I2C lub PWM z zewnętrznym wyzwalaniem pomiaru.

Na rys. 3.4 przedstawiono moduł wraz z dołączonymi przewodami.

4. Implementacja i ewaluacja modelu programowego

Ważną rolę w tworzeniu kolejnych modułów odegrał model programowy, czyli program napisany w dowolnym języku programowania i wykonywany na komputerze PC, którego działanie oddaje funkcjonalność modułu tworzonego w sprzęcie. Porównanie wyników działania modelu programowego i symulacji modułu daje informację o poprawności implementacji sprzętowej. Model programowy, wraz z uzasadnieniem użycia poszczególnych operacji, przedstawiono na początku rozdziału. Następnie skupiono się na wyborze kształtu i koloru znacznika umieszczonego na lądowisku. Na koniec przedstawiono opis badań związanych z kątem widzenia kamery przy różnych ustawieniach rozdzielczości.

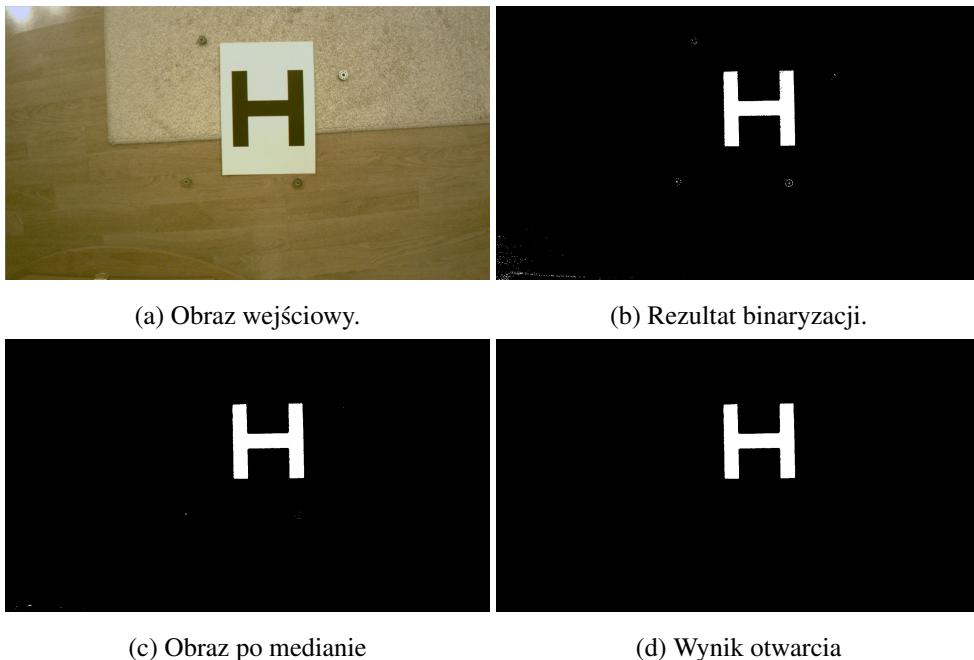
4.1. Opis zastosowanych operacji przetwarzania wizyjnego

Model programowy został napisany w pakiecie Matlab przy wykorzystaniu funkcji dostępnych w bibliotece *Image Processing Toolbox*. Pozwoliło to na szybkie prototypowanie systemu wizyjnego, składającego się z:

- konwersji z przestrzeni RGB do YCbCr,
- binaryzacji ze stałymi progami,
- mediany,
- otwarcia,
- indeksacji jednoprzebiegowej wyznaczającej pole powierzchni i prostokąt otaczający obiektów.

Piksel w przestrzeni barw YCbCr opisują trzy składowe: Y (luminancja), Cb (chrominancja, która wyraża różnicę między luminancją, a kolorem niebieskim) oraz Cr (chrominancja, która wyraża różnicę między luminancją, a kolorem czerwonym). Zaletą stosowania tej przestrzeni barw jest oddzielenie sygnału luminancji od sygnałów chrominancji, co pozwala na realizację przetwarzania sygnału przy mniejszej zależności od oświetlenia. Zastosowanie binaryzacji pozwala na oddzielenie znacznika od tła. Na wyjściu modułu znajdują się tylko dwa rodzaje pikseli: należące i nienależące do obiektu (rys. 4.1b).

Na obrazie często znajdują się jednak inne niewielkie obszary (zakłócenia), które zostały błędnie wykryte. Pozostawienie ich na wejściu modułu indeksacji zwiększyłoby niepotrzebnie liczbę nadawanych etykiet, która w rozwiązaniach sprzętowych jest zwykle ograniczona (podrozdział 5.7). Częściowym



Rys. 4.1. Przykładowe rezultaty binaryzacji, mediany i otwarcia obrazu.

rozwiązaniem problemu jest zastosowanie mediany. Jest to operacja kontekstowa, w której pikselowi wyjściowemu przypisuje się wartość średnioróżnicy uporządkowanego zbioru wartości pikseli z otoczenia piksela wejściowego. Wykorzystanie mediany powoduje znaczne zmniejszenie błędnych obszarów, jednakże niekiedy na obrazie obecne są nadal małe grupy białych pikseli nienależących do obiektu (rys. 4.1c).

Z tego powodu zdecydowano się na wykorzystanie operacji morfologicznych. Erozja i dylatacja to operacje, w których pikselowi wyjściowemu przypisuje się wartość odpowiednio najmniejszego i największego piksela w sąsiedztwie piksela wejściowego. Sąsiedztwo piksela określa kształt i rozmiar elementu strukturalnego. Zastosowanie morfologicznego otwarcia, składającego się z erozji i dylatacji, pozwala niekiedy na całkowite wyeliminowanie błędnych białych pikseli (rys. 4.1d). W projekcie założono jednak, że błędne piksele mogą nie zostać całkowicie usunięte i powiększyć próbę poradzenia sobie z tym problemem. Indeksacja jednoprzepływowa to operacja pozwalająca na wyodrębnienie z obrazu cech poszczególnych obiektów. Obiekty rozumiane są jako grupy połączonych ze sobą pikseli. Ze względu na wykorzystywany w dalszej analizie współczynnik kształtu, obliczano prostokąt otaczający i pole obiektów.

4.2. Wybór znacznika

Autonomiczne lądowanie drona w oparciu o system wizyjny wymaga wyposażenia lądowiska w marker. Jego wygląd musi umożliwiać łatwą detekcję miejsca lądowania. Z powodu wykonywania operacji

na zewnątrz, znacznik powinien również ułatwiać znalezienie go w zmiennych warunkach (zachmurzenie, cieśń).

4.2.1. Kształt

Detekcja kształtów może być realizowana na różne sposoby. Najprostszym jest progowanie współczynników kształtu, do bardziej zaawansowanych należy wyspecjalizowana deskrypcja cech (SIFT (ang. *Scale-Invariant Feature Transform*), SURF (ang. *Speeded-Up Robust Features*), HOG (ang. *Histogram of Oriented Gradients*)) i użycie klasyfikatorów (kNN (ang. *k-Nearest Neighbours*), SVM (ang. *Support Vector Machine*), sieci neuronowe). W implementowanej pierwszej wersji systemu zdecydowano się na klasyfikację przy użyciu współczynnika kształtu.

Cechami obiektu łatwymi do wyliczenia w systemie potokowym są pole figury i najmniejszy prostokąt, w którym figura się mieści (prostokąt otaczający). Postanowiono zatem wykorzystać współczynnik kształtu przedstawiony we wzorze 4.1.

$$W = \frac{P_p}{P_o} \quad (4.1)$$

Gdzie:

W – współczynnik kształtu,

P_p – pole prostokąta otaczającego,

P_o – pole obiektu.

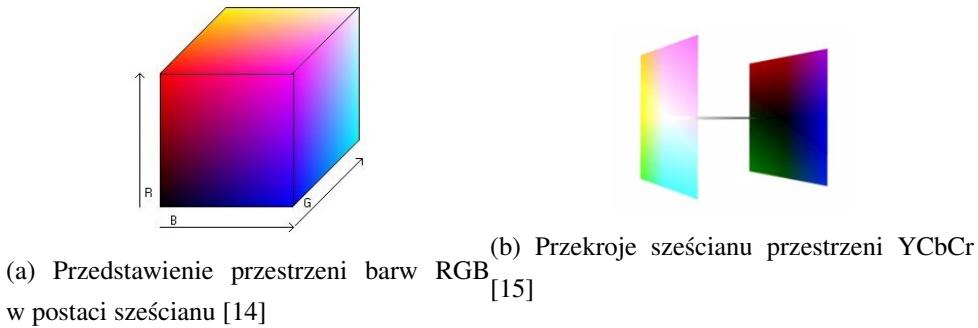
Aby taki współczynnik umożliwiał detekcję należało dobrać odpowiedni kształt znacznika. Zdecydowano się na krzyż, gdyż przy każdej jego orientacji pole prostokąta jest znacznie większe od pola obiektu. Dodatkowo, zastosowanie krzyża wydłużonego może dostarczyć informacji o orientacji znacznika.

4.2.2. Kolor

Kolor znacznika powinien umożliwiać jego łatwą segmentację. Z tego powodu pożądane jest silne skontrastowanie figury i tła. Najbardziej naturalnym rozwiązaniem jest rozważenie kontrastu w przestrzeni barw RGB, gdyż taki sygnał jest dostarczany przez kamerę.

W przestrzeni RGB każdy piksel opisywany jest przez trzy składowe: czerwoną, zieloną i niebieską. Możliwe jest przedstawienie tego systemu w formie sześcianu (rys. 4.2a). Można wyznaczyć przekątną łączącą punkty, dla których wszystkie współrzędne są identyczne. Rozciąga się ona od koloru czarnego w początku układu współrzędnych, do białego dla maksymalnych wartości składowych, przechodząc przez różne stopnie szarości. Wykorzystanie dużej odległości między kolorami i użycie czarno-białego znacznika wydaje się zatem najprostszym pomysłem.

W pierwszym kroku do testów przygotowano czarny znacznik na białym tle. Kamerą PCAM 5C wykonano trzy zdjęcia przy różnym poziomie oświetlenia (rys. 4.3a, 4.3c, 4.3e). Wykonanie takich zdjęć



Rys. 4.2. Modele przestrzeni RGB i YCbCr

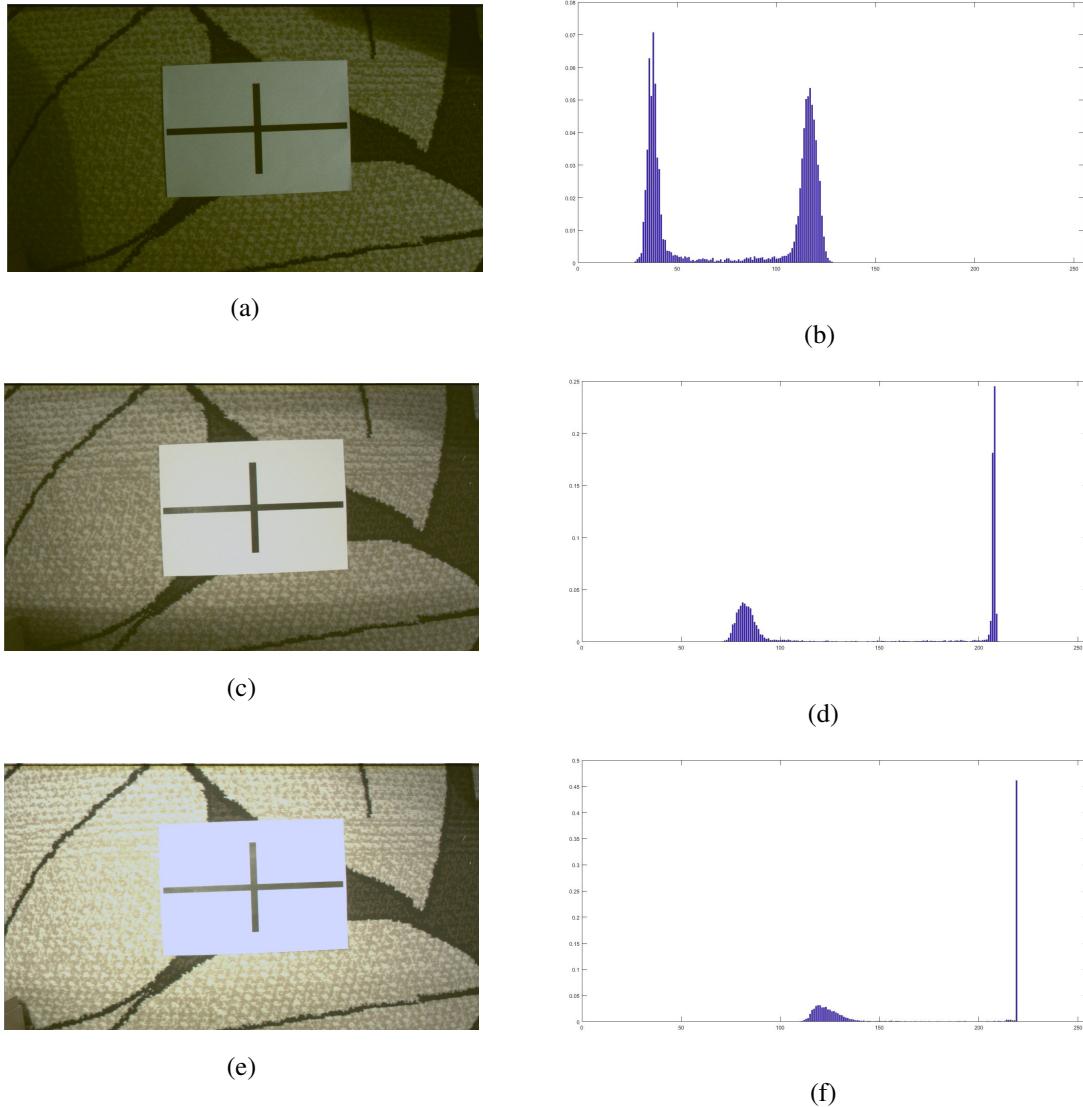
było możliwe po implementacji akwizycji ramek na kartę SD (podrozdział 5.1.1). Po przejściu na obraz w skali szarości, posługując się narzędziem *roipoly* w programie Matlab, obliczono histogramy obszaru znacznika (rys. 4.3b, 4.3d, 4.3f).

Analiza histogramów wskazuje na silne uzależnienie położenia maksimów odpowiadających tłu i znacznikowi od oświetlenia. Dla kolejnych obrazów, wartości progów binaryzacji mogłyby zawierać się w zakresach: 50-100, 100-200, 150-210. Poziom białego tła dla obrazka najsłabiej oświetlonego jest mniejszy niż wartość czarnego koloru znacznika najlepiej oświetlonego. W takiej sytuacji niemożliwy jest dobór stałego progu umożliwiającego skuteczną binaryzację.

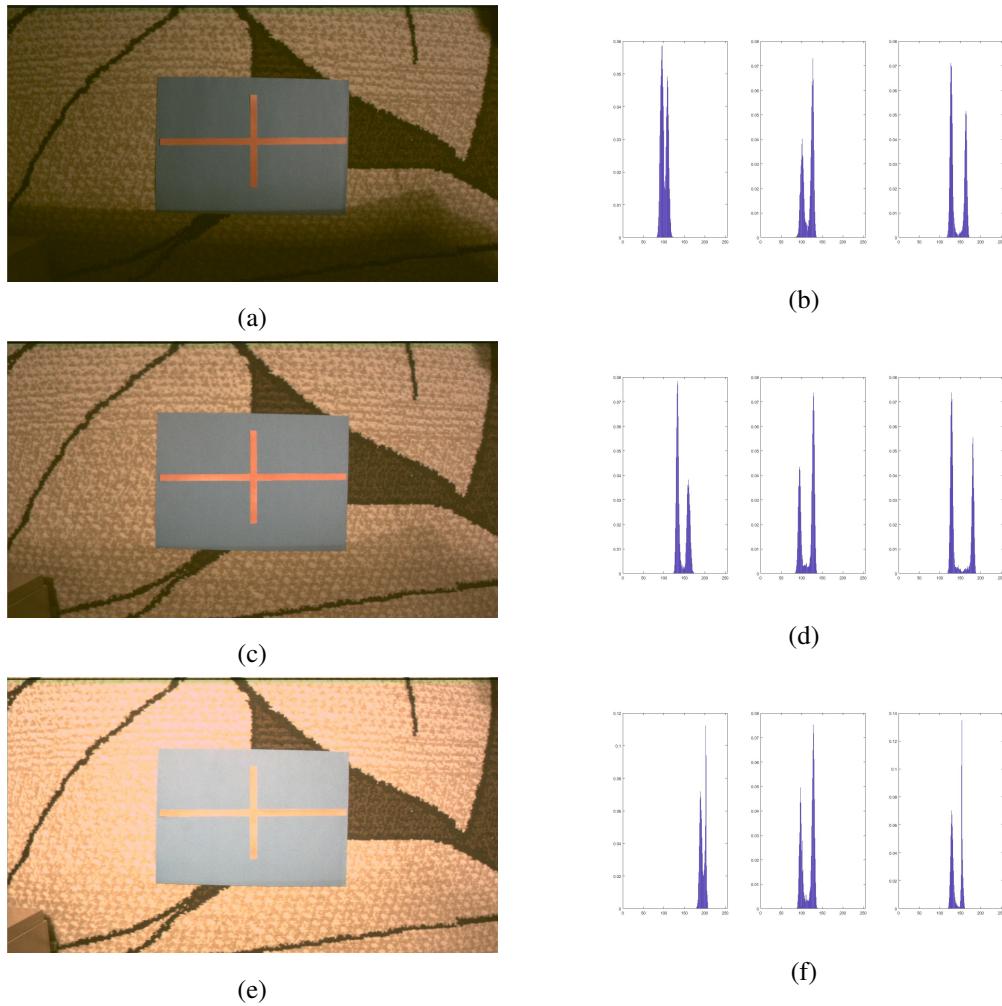
Z tego powodu zdecydowano się na powtórzenie eksperymentu w przestrzeni YCbCr. Tak jak opisano w rozdziale 4.1, piksel w tej przestrzeni opisuje współrzędną luminancji i dwie składowe chrominacji. Podobnie jak w przypadku RGB, przestrzeń YCbCr również da się przedstawić w postaci sześcianu. Tym razem skala szarości przebiega przez środek płaszczyzny Cb-Cr i za zmianę poziomu szarości odpowiada współrzędna luminancji (rys. 4.2b). Dla każdej wartości piksela informacja o jasności oddzielona jest od barwy.

Kolor znacznika określono jako czerwony, natomiast tło niebieskie, gdyż te barwy znajdują się na przeciwnych stronach płaszczyzny Cr-Cb. Zdjęcia wykonano przy takich samych poziomach oświetlenia jak poprzednio. Histogramy kolejnych obrazów przedstawiono na rysunku 4.4b, 4.4d, 4.4f.

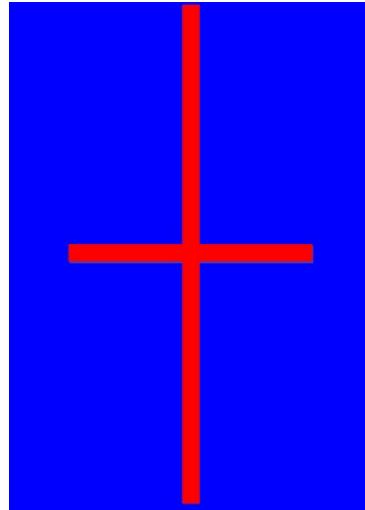
Zgodnie z oczekiwaniami, zmiana poziomu oświetlenia przełożyła się na zmianę wartości luminancji, natomiast składowe chrominancji nie zmieniały się znacząco. Każdy z obrazów może być skutecznie zbinaryzowany przy użyciu stałych progów: górnego 120 dla składowej Cb i dolnego 150 dla składowej Cr. Różnice pomiędzy wartościami składowych dla znacznika i tła nie są jednak duże – wynoszą około 30 poziomów. Ostatecznie zdecydowano się na znacznik przedstawiony na rys. 4.5.



Rys. 4.3. Zdjęcia pierwszej wersji znacznika wraz z histogramami obszaru znacznika



Rys. 4.4. Obrazy i histogramy obszarów znacznika w przestrzeni YCbCr.



Rys. 4.5. Wybrana postać znacznika

Tabela 4.1. Wartości kąta widzenia kamery w zależności od rozdzielczości

Rozdzielczość	Kąt widzenia kamery w pionie	Kąt widzenia kamery w poziomie
1920 x 1080	28°	54°
1280 x 720	38°	76°

4.3. Analiza kąta widzenia kamery przy różnych ustawieniach rozdzielczości

W celu zbadania kąta widzenia kamery stworzono moduł nakładający na obraz prostopadłe linie przecinające się w jego środku. 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 (4.2).

$$\alpha = 2 \arctan \frac{l}{h} \quad (4.2)$$

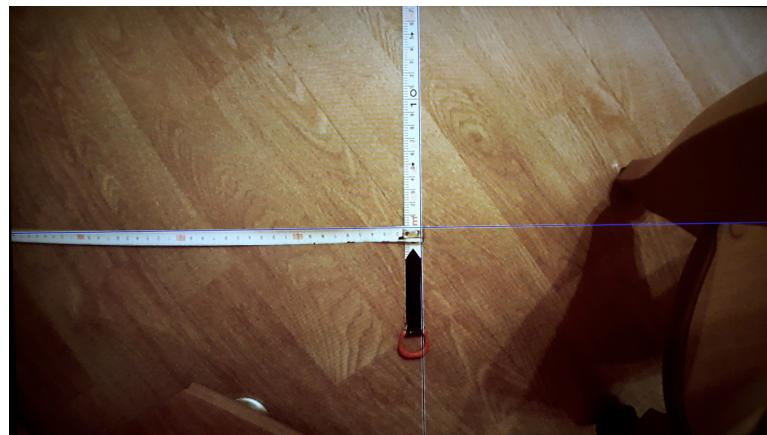
gdzie:

α – kąt widzenia kamery,

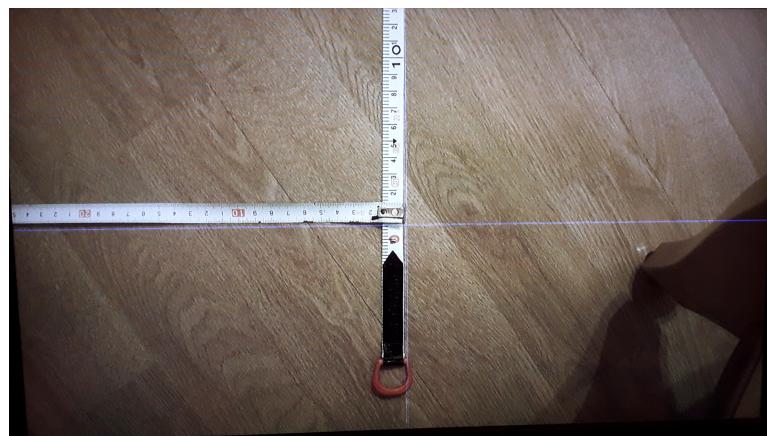
h – wysokość, na jakiej umieszczona jest kamera,

l – odległość środka obrazu od jego krawędzi.

Na podstawie obrazu przedstawionego na rysunku 4.6a obliczono kąty widzenia kamery w pionie i poziomie dla rozdzielczości 1920 x 1080. Dla rozdzielczości 1280 x 720 skorzystano z obrazu pokazanego na rysunku 4.6b. Korzystając ze wzoru (4.2) otrzymano przybliżone wyniki, przedstawione w tabeli 4.1.



(a) Rozdzielcość 1280 x 720.



(b) Rozdzielcość 1920 x 1080.

Rys. 4.6. Obrazy służące do wyznaczenia kąta widzenia kamery.

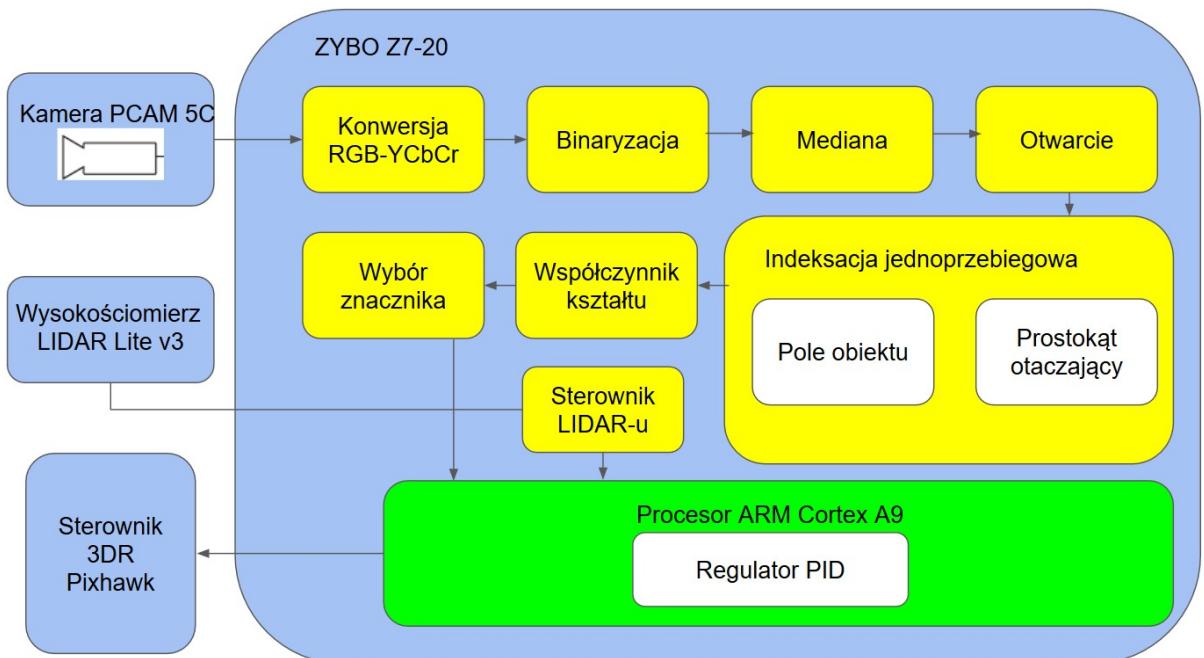
Tak jak napisano w podrozdziale 5.1, dla rozdzielczości 1280 x 720 kąt widzenia kamery jest większy, co w połączeniu z mniejszą liczbą pikseli do przetworzenia zdecydowało o wyborze tej rozdzielczości. Uzyskane wyniki oznaczają również, że z wysokości 1 m dron byłby w stanie wykryć obiekty w odległości 34 cm z przodu i z tyłu, oraz 78 cm po bokach.

4.4. Podsumowanie

Implementacja modelu programowego umożliwiła przetestowanie działania poszczególnych modułów przetwarzania wizyjnego. Narzędzia programu Matlab okazały się również bardzo pomocne w analizie koloru znacznika. Sprawdzenie kąta widzenia kamery pozwoliło zorientować się z jakiej odległości lądowisko będzie dla drona widoczne.

5. Implementacja i ewaluacja systemu sprzętowo-programowego

Na rysunku 5.1 przedstawiono ogólny schemat zaimplementowanego systemu. Na niebiesko zaznaczono zostały wykorzystywane urządzenia, na żółto oznaczono moduły zaimplementowane w części rekonfigurowalnej, natomiast na zielono zaznaczono system procesorowy. Sygnał wizyjny z kamery trafia do części reprogramowalnej i jest następnie przetwarzany przez poszczególne moduły. Po wyborze znacznika informacja o jego pozycji i uchybie regulacji trafia do procesora. System procesorowy wysyła komendy sterujące do kontrolera drona. Zadaniem układu regulacji jest przesunięcie drona nad lądowiskiem, umożliwiając wykonanie lądowania. W dalszej części rozdziału omówiono poszczególne komponenty systemu.



Rys. 5.1. Schemat przedstawiający zaimplementowany system.

5.1. Akwizycja obrazu

Firma Digilent, na swojej stronie internetowej, udostępnia projekt demonstracyjny połączenia kamery i płytka [16]. Pozwala on pobrać obraz z kamery oraz wyświetlić go na monitorze. Ponadto, przez port szeregowy możliwa jest zmiana rozdzielczości, szybkości akwizycji ramek, współczynnika korekcji gamma oraz 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.

Przeprowadzone eksperymenty pokazały, że przy rozdzielczości 1280 x 720 kąt widzenia kamery jest większy. Ze tego względu zdecydowano się użyć tej rozdzielczości w docelowym systemie. Przeprowadzanie syntezy i implementacji projektu możliwe było przy użyciu darmowego oprogramowania Vivado oraz SDK w wersji WebPack (w wersji 2018.2). Opisany projekt stanowił bazę do dalszych prac.

5.1.1. Zapis ramek na kartę SD

Aby możliwe było użycie ramek z kamery w modelu programowym, należało zaimplementować akwizycję obrazów z modułu PCAM na kartę SD. W użytym projekcie bazowym zastosowano połączenie toru wizyjnego AXI z zewnętrzną pamięcią RAM dostępną na karcie Zybo poprzez moduł *AXI Video Direct Memory Access*. Szesnastkowy adres miejsca w pamięci, gdzie znajduje się wartość składowej R pierwszego piksela ramki, ustawiany jest przy konfiguracji modułu VDMA i przesyłany jest do użytkownika przy inicjalizacji połączenia. Możliwe jest jego odczytanie i tym samym uzyskanie dostępu do zawartości pamięci przy wykorzystaniu wskaźnika w języku C++.

W celu umożliwienia komunikacji z kartą SD uaktywniono interfejs procesora SD0. Następnie do projektu w SDK dodano bibliotekę „xilffs” i korzystając z funkcji systemu plików opisanych w [17], zaimplementowano zapis ramek obrazu do pliku. Ze względu na prostotę pliku zdecydowano się na format *ppm*. W formacie tym nie występuje kompresja i przez to jest on nieefektywny pod względem zapotrzebowania na pamięć. Niemniej jednak, dysponując odpowiednio dużą ilością miejsca na karcie i chcąc w łatwy sposób zapisać dane, warto zastosować właśnie ten format. Zapis danych w formacie *ppm* jest prosty, gdyż plik składa się jedynie z nagłówka (typ pliku, rozmiary obrazka, maksymalna wartość składowych) oraz kolejnych wartości w formie pojedynczych bajtów (jeśli maksymalna wartość jest mniejsza niż 256).

5.2. Podłączenie laserowego czujnika wysokości

Czujnik wysokości podłączono z zastosowaniem wyjścia PWM (ang. *Pulse-Width Modulation*). W tym trybie czujnik ustawia stan wysoki na wyjściu przez czas proporcjonalny do zmierzonej odległości. Stan wysoki trwa 10 µs na każdy zmierzony centymetr. Pin wyzwalania zwarte na stałe do masy, powodując ciągłe wykonywanie pomiaru. Wyjście PWM podłączono do trzeciego pinu portu Pmod JB układu ZYBO. Okazało się, że możliwe jest działanie modułu przy zasilaniu bezpośrednio z płytki z napięciem 3,3 V.

Moduł odczytujący wysokość zaimplementowany został jako maszyna stanów. Stan pierwszy to oczekiwanie na pojawienie się stanu wysokiego. Po wykryciu zbocza narastającego następuje zerowanie rejestrów oraz przejście do stanu drugiego. W tym stanie liczony jest czas trwania stanu wysokiego. Wywoływanie procesu z częstotliwością 1 MHz, czyli co 1 µs pozwala na łatwą konwersję zmierzonego czasu na odległość. Odczytana wysokość jest następnie przesyłana do procesora z użyciem rejestrów AXI.

5.3. Konwersja z przestrzeni barw RGB do YCbCr

Konwersję z przestrzeni barw RGB do YCbCr wykonano zgodnie ze wzorem 5.1. Przy implementacji wykorzystano sprzętowe mnożarki oraz sumatory. Na rysunku 5.2 przedstawiono schemat operacji arytmetycznych dla jednej ze składowych. Aby możliwe było prowadzenie obliczeń, wszystkie stałe należało przedstawić w postaci liczb stałoprzecinkowych. Do reprezentacji stałoprzecinkowej użyto rejestrów o szerokości 18 bitów.

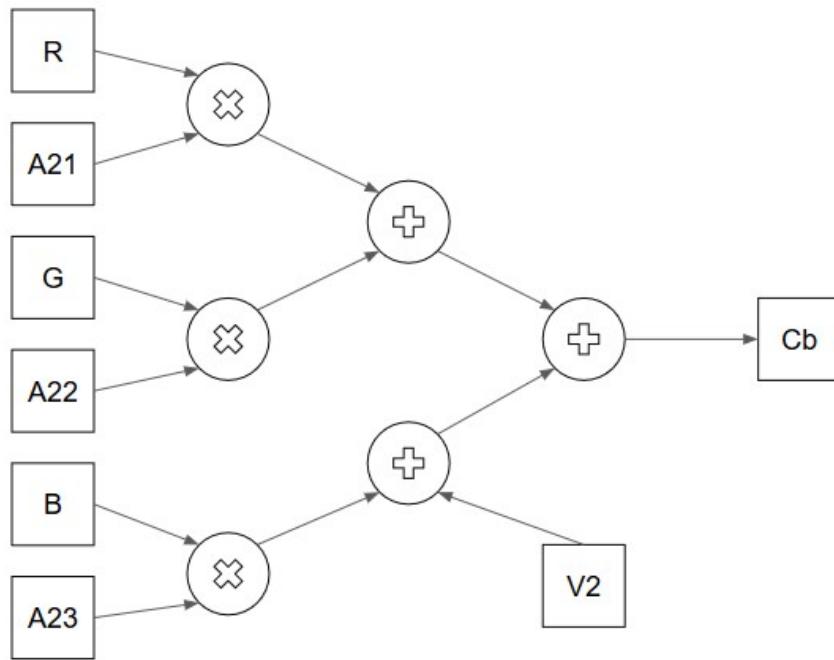
$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0,299 & 0,587 & 0,114 \\ -0,168736 & -0,331264 & 0,5 \\ 0,5 & -0,418688 & 0,081312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \quad (5.1)$$

5.4. 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). Do szybkiej zmiany progów wykorzystano komunikację PS-PL z użyciem rejestrów AXI – progi mogły być zmieniane z poziomu terminala i testy nie wymagały ponownych implementacji projektu.

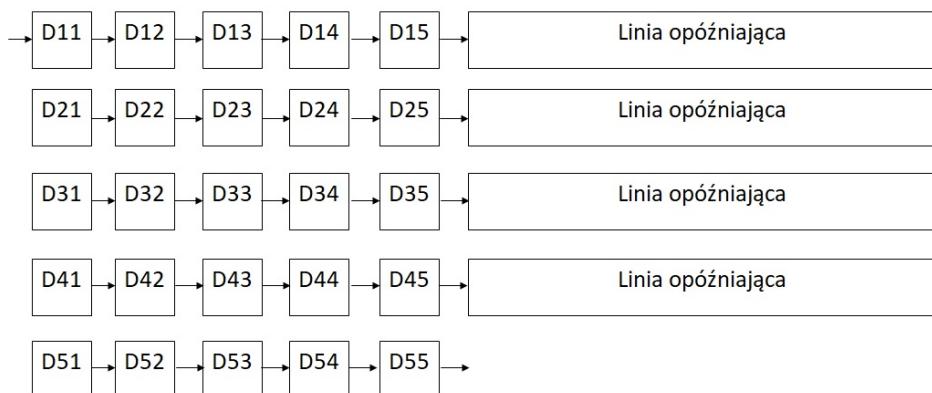
5.5. Mediana

W przypadku działania na obrazie binarnym operacja mediany może być przeprowadzona przez obliczenie sumy wartości pikseli wewnątrz kontekstu, a następnie porównanie jej z połową maksymalnej

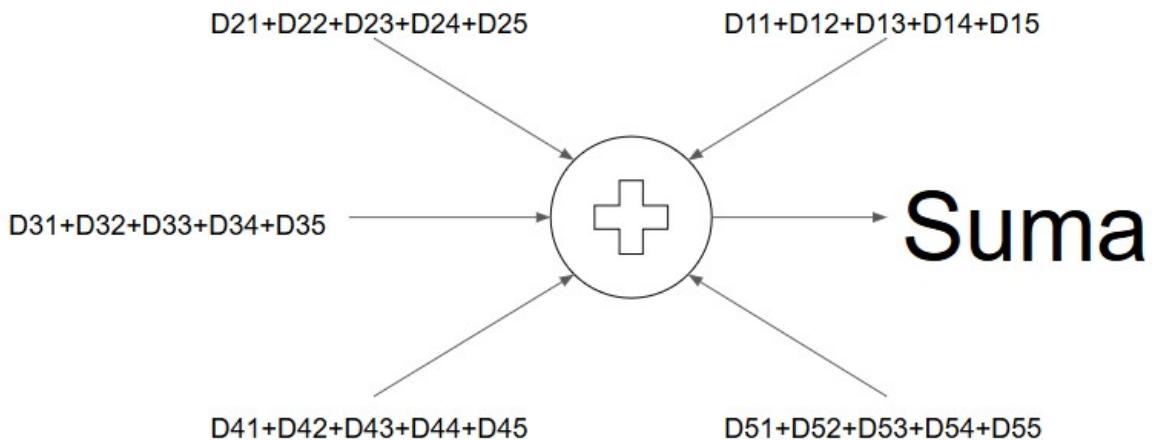


Rys. 5.2. Schemat wyznaczania składowej C_b . $A_{21} = -0,168736, A_{22} = -0,331264, A_{23} = 0,5, V_2 = 128$.

wartości tej sumy. Ze względu na łatwość implementacji oraz zadowalające wyniki filtracji, 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 5.3. 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 (Rys. 5.4).



Rys. 5.3. Schemat wyznaczania kontekstu dla mediany, erozji i dylatacji.



Rys. 5.4. Schemat wyznaczania sumy otoczenia piksela.

5.6. Otwarcie

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 wartość maksymalną. Podobnie jak w przypadku mediany, jako element strukturalny zdecydowano się na kwadrat o boku 5 pikseli i zastosowano rozwiązańe podobne jak przy medianie (rys. 5.3).

5.7. Indeksacja

Zazwyczaj na wejście modułu indeksacji podawany jest obraz zbinaryzowany, natomiast na wyjściu pojawia się obraz, na którym wartość pikseli 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 5.5. 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:

- w otoczeniu piksela znajdują się tylko piksele należące do tła,
- otoczenie zawiera jeden lub więcej pikseli, którym została wcześniej przypisana taka sama etykieta L ,
- w otoczeniu znajdują się piksele posiadające różne etykiety (np. $L1$ i $L2$).

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.

D5	D4	D3
D2	D1	

Rys. 5.5. 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 sklejeń. Tablica ta zawiera informację, które etykiety powinny zostać złączone. W przypadku *a* do tablicy sklejeń na pozycji odpowiadającej etykiecie zapisywana jest etykieta, natomiast gdy zachodzi opcja *c* etykietę mniejszą zapisuje się pod indeksem większej. Należy zaznaczyć, że opisana wersja obsługi sklejeń działa poprawnie tylko w przypadku braku tzw. „łańcuchów sklejeń” tj. dla uproszczonych kształtów. 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 [18] podano sposób, w jaki można tego dokonać. Opiera się on na scaleniu nie samych wartości pikseli, ale obliczanych na bieżąco parametrów obiektu. Implementacja indeksacji w języku Verilog rodzi trudności związane z określeniem przypadku istnienia tej samej lub różnych etykiet w otoczeniu piksela.

O ile wykrycie przypadku *a* jest łatwe, 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. Zgodnie z opisanymi wcześniej zasadami uzupełniano również tablicę sklejeń. Oprócz tego, na bieżąco obliczano prostokąt otaczający oraz liczbę pikseli należących do każdego ze znalezionych obiektów.

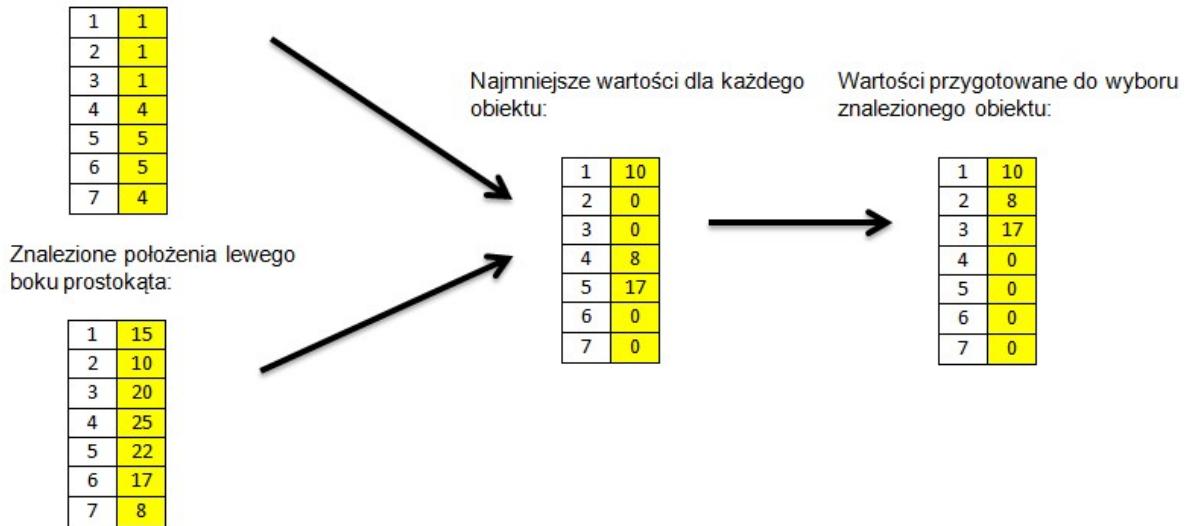
Po poetykietowaniu całej ramki obrazu wykorzystano tablicę sklejeń do złączenia obliczanych na bieżąco cech. Ten etap algorytmu zaimplementowano jako maszynę stanów:

- Stan 0 – Oczekiwanie na sygnał końca ramki wyznaczany na podstawie synchronizacji pionowej. W momencie wykrycia sygnału następuje rejestrowanie tablicy sklejeń i obliczonych parametrów (w następnym taktie zostaną one zresetowane), 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 znalezionego 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.

Schemat przetwarzania danych po każdej ramce pokazano na rysunku 5.6.

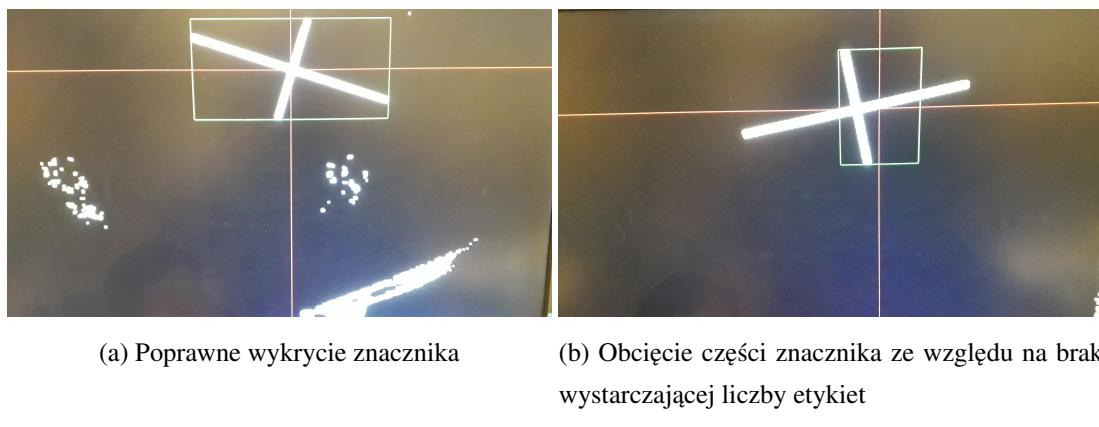
Tablica sklejeń:



Rys. 5.6. Schemat procesu przetwarzania informacji po każdej ramce obrazu przy zastosowaniu indeksacji z przykładowymi danymi. Tablica sklejeń daje informację o konieczności sklejenia etykiet 1, 2, 3. Ponieważ przykład dotyczy lewego boku prostokąta, spośród liczb 15, 10, 20 (odpowiadającym etykietom 1, 2, 3) wybierana jest najmniejsza. Ostatni krok przetwarzania to usunięcie środkowych zer z wektora.

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. Zasoby części rekonfigurowalnej układu dostępnego na karcie ZYBO Z7-20 pozwoliły na zarezerwowanie miejsca dla 30 etykiet. Przy pewnej określonej orientacji znacznika nie jest to niestety wystarczająca liczba etykiet (rys. 5.7).

Aby możliwa była sprawna detekcja miejsca lądowania niezależnie od orientacji markera, należało zmienić nieco koncepcję systemu. Innym podejściem było obliczanie środka ciężkości wszystkich białych pikseli. Eliminuje to możliwość klasyfikacji obiektów ze względu na kształt (pozostaje klasyfikacja po kolorze) i tym samym czyni system podatnym na zakłócenia. Pojawienie się w kadrze innych grup pikseli o tym samym kolorze spowoduje przesunięcie środka ciężkości. Jednak w warunkach testowych możliwe jest zastosowanie tła o kolorze silnie kontrastującym z barwą znacznika. Z tego powodu moduł wyznaczający środek ciężkości (podrozdział 5.8) był wykorzystywany podczas testowania rozwiązania. Wyznaczanie pola figury i prostokąta otaczającego (podrozdział 5.8.1) umożliwiło odrzucanie niepoprawnych detekcji i wysyłanie sygnału o braku znalezienia znacznika.



Rys. 5.7. Rezultaty indeksacji w zależności od orientacji znacznika

5.8. Wyznaczanie środka ciężkości

Do wyznaczania środka ciężkości pikseli należących do obiektu wykorzystano wzory (5.2), (5.3) i (5.4).

$$m_{00} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} x_{ij} \quad (5.2)$$

$$m_{10} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} i * x_{ij} \quad (5.3)$$

$$m_{01} = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} j * x_{ij} \quad (5.4)$$

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 (5.5) i (5.6).

$$X_{sc} = \frac{m_{10}}{m_{00}} \quad (5.5)$$

$$Y_{sc} = \frac{m_{01}}{m_{00}} \quad (5.6)$$

gdzie:

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 5.2, 5.3 i 5.4. Po przejściu przez całą ramkę obrazu wykonywane jest dzielenie na podstawie wzorów 5.5 i 5.6.

5.8.1. Wyznaczanie prostokąta otaczającego i pola powierzchni

Znalezienie prostokąta otaczającego sprowadza się do wyznaczenia skrajnych jego punktów na górze, dole, po lewej oraz prawej stronie. Na podstawie sygnałów synchronizacji oraz wymiarów obrazka wyznaczano współrzędne aktualnie przetwarzanego piksela. Jeśli jest to piksel należący do obiektu, następuje inkrementacja jego pola powierzchni. Do odpowiednich rejestrów trafiają wówczas również wartości współrzędnych piksela, jeśli wykraczają poza aktualną zawartość rejestrów:

- do rejestru zawierającego górnego bok prostokąta trafi współrzędna wierszowa, jeśli będzie ona mniejsza od aktualnej,
- do rejestru zawierającego dolny bok prostokąta trafi współrzędna wierszowa, jeśli będzie ona większa od aktualnej,
- do rejestru zawierającego lewy bok prostokąta trafi współrzędna kolumnowa, jeśli będzie ona mniejsza od aktualnej,
- do rejestru zawierającego prawy bok prostokąta trafi współrzędna kolumnowa, jeśli będzie ona większa od aktualnej,

5.9. Algorytm sterowania

Informacjami o detekcji lądowiska, koniecznymi do realizacji algorytmu sterowania, są:

- uchyb regulacji w obu osiach, rozumiany jako odległość środka obrazu od środka wykrytego lądowiska (centrum prostokąta otaczającego lub środka ciężkości),
- wiadomość o znalezieniu znacznika.

Informacje te przesyłano z części rekonfigurowalnej do systemu procesorowego przy wykorzystaniu rejestrów AXI.

Dodatkowo, w celu analiza działania systemu, istnieje możliwość podglądu przetworzonego obrazu na monitorze, podłączonym poprzez port HDMI. Możliwy jest podgląd kolejnych etapów przetwarzania – wybór etapu następuje przez zmianę ustawień przełączników na płytce.

Algorytm sterowania bazuje na dyskretnym regulatorze PID, zrealizowanym w systemie procesorowym układu. Regulacji podlegają składowe x i y wektora położenia drona względem znacznika określającego lądowisko. Jeśli znacznik zostanie utracony z pola widzenia kamery na czas 1 sekundy, dron przechodzi w tryb unoszenia się nad ziemią.

5.10. Integracja płytki z autopilotem

W układzie Zynq z karty ZYBO Z7-20 wykorzystano interfejs UART0. Wyprowadzono piny TX i RX z systemu procesorowego i podłączono je do portu Pmod JB. Po stronie sterownika Pixhawk użyto portu TELEM 2. Szybkość transmisji to 115200 bodów.

Wysyłanie komend opiera się na wykorzystaniu gotowych funkcji dostępnych w sieci [19]. Przygotowują one daną wiadomość zgodnie z wymaganiami protokołu MAVLink. Następnie wywoływane są funkcje wysyłające przygotowaną tablicę bajtów przez odpowiedni UART. W projekcie wykorzystano stworzone przy wykonywaniu pracy [10] funkcje opakowujące kolejne etapy przetwarzania komend.

5.11. Ewaluacja

Zaimplementowane rozwiązania należało przetestować. Eksperimentem pozwalającym sprawdzić działanie laserowego czujnika odległości, przetwarzania wizyjnego i algorytmu sterującego był test opisany w podrozdziale 5.11.1. Test ten wymagał również integracji urządzeń na platformie. Podrozdział 5.11.2 opisuje natomiast test komunikacji karty i sterownika drona.

5.11.1. Test sterowania

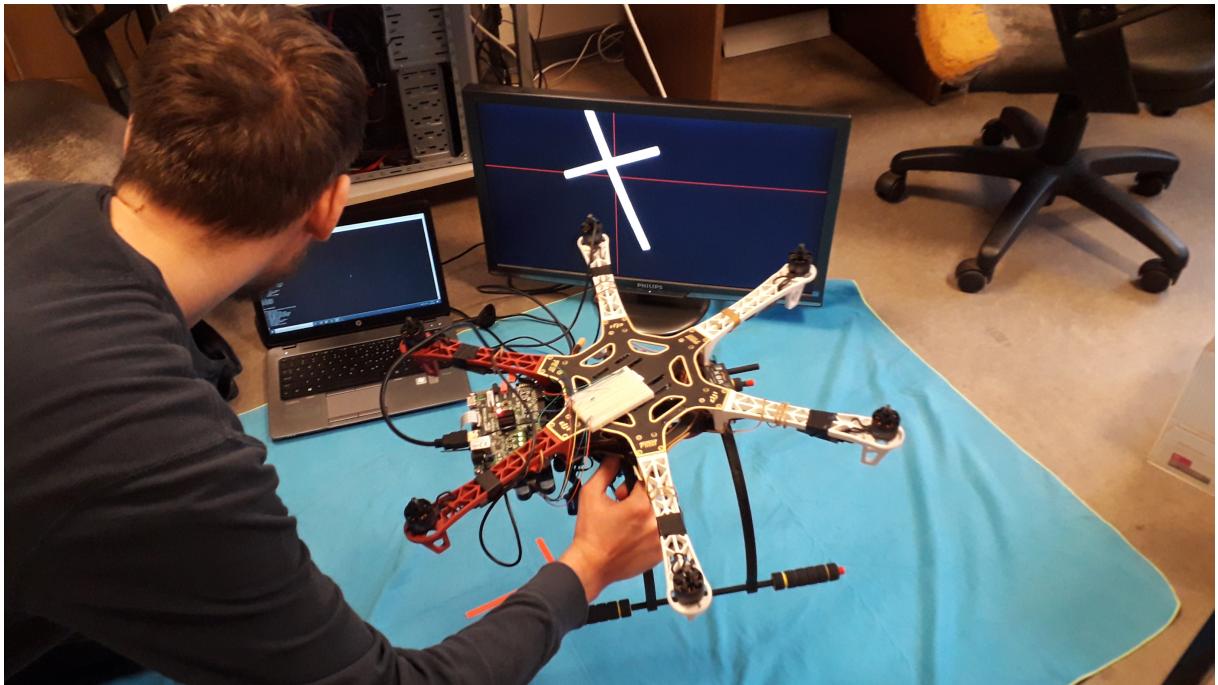
Test ten polegał na „ręcznym” wykonywaniu komend pochodzących z systemu procesorowego – dron był trzymany w ręce. Zaplanowano 3 fazy lotu: wznoszenie na określoną wysokość, kierowanie drona w stronę znacznika oraz lądowanie. Przy takim rozwiążaniu niemożliwe było zadawanie prędkości wynikającej z regulacji PID. Ograniczono się do wysyłania komend: „w górę”, „w lewo”, „do przodu” itp. Postarano się, aby podłożе miało kolor niebieski, kontrastujący z czerwoną barwą znacznika. Zdjęcie z eksperimentu przedstawiono na rys. 5.8.

Test wykazał prawidłowe działanie laserowego czujnika odległości. Poprawnie podawane były również komendy dotyczące zmiany pozycji drona.

5.11.2. Test reakcji autopilota na zadawane komendy

Test polegał na wysyłaniu do autopilota komend z karty ZYBO Z7-20. Podczas eksperimentu śmiała drona były zjęte. Wewnątrz budynku, możliwe było wysłanie komendy uzbrojenia silników w trybie *Stabilize*. Jak opisano w rozdziale 3.4, nie jest to jednak tryb pozwalający na wysyłanie komend ruchu. Aby możliwe było kierowanie dronem, należało przeprowadzić eksperiment na zewnątrz, w celu utrzymania dostatecznie silnego sygnału GPS. Po uaktywnieniu trybu *Guided* wysłano kolejno komendy uzbrojenia silników, startu i lotu z określoną prędkością. Silniki reagowały na wysyłane rozkazy.

Po eksperymencie można było wysnuć wniosek o poprawnej konfiguracji komunikacji pomiędzy kartą ZYBO, a sterownikiem Pixhawk.



Rys. 5.8. Zdjęcie z eksperymentu. Kartę ZYBO Z7-20 zamontowano z przodu platformy. Poniżej umieszczono kamerę i lidar. Na górze znajduje się płytka prototypowa do łatwego łączenia komponentów.

5.12. Podsumowanie

W ramach implementacji sprzętowo-programowej wykonano szereg modułów, które przetestowano symulacyjne oraz w sprzęcie na karcie Zybo. Ponadto przeprowadzono ewaluację całego systemu. Test sterowania pokazał możliwość wykrycia lądowiska i obliczania takiego sterowania dronem, aby lądowanie odbyło się w wyznaczonym miejscu. Test komunikacji ze sterownikiem pozwolił natomiast potwierdzić poprawność połączenia pomiędzy platformą obliczeniową, a autopilotem.

6. Podsumowanie

W pracy przedstawiono sprzętowo-programową realizację systemu wspomagającego autonomiczne lądowanie drona. System zrealizowano w układzie Zynq SoC na karcie ZYBO Z7-20. Część rekonfigurowalna pozwoliła na szybkie przetwarzanie obrazu, natomiast system procesorowy umożliwił sprawną realizację algorytmu sterowania i wysyłanie komend do sterownika drona.

Zaimplementowano dwie wersje systemu wizyjnego: z indeksacją jednoprzebiegową i identyfikacją znacznika na podstawie koloru i współczynnika kształtu oraz z wyznaczaniem środka ciężkości i identyfikacją markera przy użyciu jego barwy. Poprawną segmentację na podstawie koloru przy niewielkiej zależności od oświetlenia umożliwiła binaryzacja w przestrzeni barw YCbCr, w oparciu o składowe Cb i Cr. Zaprojektowano również taki znacznik, aby ułatwić jego wykrycie. Wykorzystanie modułów mediany i otwarcia pozwoliło na filtrację maski binarnej.

Wprowadzenie do układu sygnału z czujnika lidar umożliwiło dokładną kontrolę wysokości. Zaimplementowano także regulację PID, której celem jest ustawnie drona nad znacznikiem. Eksperymenty pokazały możliwość wykonania symulowanego lotu testowego składającego się ze startu, regulacji położenia i lądowania w wyznaczonym miejscu. Przeprowadzone testy dowiodły również możliwości wysyłania do sterownika komend, między innymi dotyczących zmiany kierunku i szybkości lotu.

Kolejnym celem powinno być wykonanie lotu testowego (start – regulacja położenia – lądowanie) przy sterowaniu na podstawie wizyjnego sprzężenia zwrotnego. Etapem pośrednim powinno być przeprowadzenie testu pozwalającego ustalić, jaką najmniejszą prędkość akceptuje autopilot, jaka największa prędkość powoduje ruch drona bez wyraźnego przechylenia i pochylenia oraz z jaką częstotliwością mogą być wysyłane komendy. Te informacje pozwoląby na dobór nastaw regulatora. W przypadku niemożliwości realizacji komend z odpowiednią częstotliwością rozważyć można inną koncepcję: na podstawie wysokości i aktualnej pozycji znacznika na obrazie można byłoby wyznaczać od razu przesunięcie (zamiast prędkości), które doprowadziłyby dron ponad znacznik. Zadawanie przesunięcia również jest wspierane przez oprogramowanie ArduPilot. Takie podejście wymagałoby przeprowadzenia odpowiedniej kalibracji systemu – znalezienia zależności pomiędzy odlegością w pikselach, a rzeczywistą dla danej wysokości.

Powysze działania możliwe są przy zastosowaniu komponentów użytych w projekcie. Użycie kamery o większym kącie widzenia pozwoliłoby jednak na detekcję lądowiska z większej odległości. Innym kierunkiem dalszych prac może być próba wykonania lądowania na poruszającej się platformie. Analiza literatury naukowej dostarczyła informacji co do niezbędnych działań, jakie musi wykonywać

taki system. Są to: śledzenie lądowiska, predykcja ruchu lądowiska, generacja trajektorii pozwalających dotrzeć do celu, wybór jednej z dopuszczalnych oraz realizacja odpowiedniego sterowania.

Bibliografia

- [1] <http://go.skyward.io/rs/902-SIU-382/images/2018StateofDrones.pdf>. Dostęp: 2020-01-18.
- [2] <https://spectrum.ieee.org/aerospace/aviation/us-commercial-drone-deliveries-will-finally-be-a-thing-in-2020>. Dostęp: 2020-01-18.
- [3] S. Lange, N. Sunderhauf i P. Protzel. „A vision based onboard approach for landing and position control of an autonomous multirotor UAV in GPS-denied environments”. W: *2009 International Conference on Advanced Robotics*. 2009, s. 1–6.
- [4] Y. Fan, S. Haiqing i W. Hong. „A Vision-Based Algorithm for Landing Unmanned Aerial Vehicles”. W: *2008 International Conference on Computer Science and Software Engineering*. T. 1. 2008, s. 993–996. DOI: [10.1109/CSSE.2008.309](https://doi.org/10.1109/CSSE.2008.309).
- [5] A. Price i in. „Real time object detection for an unmanned aerial vehicle using an FPGA based vision system”. W: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006*. 2006, s. 2854–2859. DOI: [10.1109/ROBOT.2006.1642134](https://doi.org/10.1109/ROBOT.2006.1642134).
- [6] D. Falanga i in. „Vision-based autonomous quadrotor landing on a moving platform”. W: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, s. 200–207. DOI: [10.1109/SSRR.2017.8088164](https://doi.org/10.1109/SSRR.2017.8088164).
- [7] Y. Fan, S. Haiqing i W. Hong. „A Vision-Based Algorithm for Landing Unmanned Aerial Vehicles”. W: *2008 International Conference on Computer Science and Software Engineering*. T. 1. 2008, s. 993–996. DOI: [10.1109/CSSE.2008.309](https://doi.org/10.1109/CSSE.2008.309).
- [8] V. Sudevan, A. Shukla i H. Karki. „Vision based autonomous landing of an Unmanned Aerial Vehicle on a stationary target”. W: *2017 17th International Conference on Control, Automation and Systems (ICCAS)*. 2017, s. 362–367. DOI: [10.23919/ICCAS.2017.8204466](https://doi.org/10.23919/ICCAS.2017.8204466).
- [9] <https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/reference-manual>. Dostęp: 2020-01-18.
- [10] M. Mach. „Wbudowany system wizyjny do śledzenia obiektów dla potrzeb nawigacji bezzałogowego statku powietrznego (UAV)”. Praca magisterska. AGH, 2017.
- [11] <https://ardupilot.org/copter/docs/flight-modes.html>. Dostęp: 2020-01-18.
- [12] <https://ardupilot.org/dev/docs/mavlink-basics.html>. Dostęp: 2020-01-21.

- [13] <https://www.sparkfun.com>. Dostęp: 2020-01-31.
- [14] <http://www.algorytm.org/modele-barw/model-rgb.html>. Dostęp: 2020-01-20.
- [15] <https://www.youtube.com/watch?v=3dET-EoIMM8>. Dostęp: 2020-01-20.
- [16] <https://reference.digilentinc.com/learn/programmable-logic/tutorials/zynq-z7-pcam-5c-demo/start>. Dostęp: 2019-11-03.
- [17] http://elm-chan.org/fsw/ff/00index_e.html. Dostęp: 2019-11-03.
- [18] Abdul Malik i in. „Real-time Component Labelling with Centre of Gravity Calculation on FPGA”. W: (sty. 2011).
- [19] https://github.com/mavlink/c_library_v1/tree/master/common. Dostęp: 2020-01-22.

A. Spis zawartości płyty CD

Na dołączonej do pracy płycie CD znajdują się następujące foldery:

- Matlab – zawiera pliki modeli programowych, testów, obróbki obrazów oraz zdjęcia niezbędne do ich uruchomienia,
- Vivado – zawiera projekt główny oraz testowy, wykorzystywany do komunikacji z autopilotem,
- Tex – zawiera pliki tekstowe \LaTeX wraz z rysunkami pojawiającymi się w tekście.