

# QUADROCOPTER

(sterowanie, stabilizacja, autopilot)

*Projekt przejściowy 2014/15  
specjalności Robotyka  
na Wydziale Elektroniki*

Politechnika Wrocławska 2014

Autorzy

*Marcin Ciopcia*

*Michał Drwiega*

*Daniel Gut*

*Alicja Jurasik*

*Agata Leś*

*Kacper Nowosad*

*Piotr Semberecki*

*Hanna Sienkiewicz*

*Mateusz Stachowski*

*Paweł Urbaniak*

*Krzysztof Zawada*

Skład raportu wykonano w systemie L<sup>A</sup>T<sub>E</sub>X



Praca udostępniana na licencji Creative Commons: *Uznanie autorstwa-Użycie niekomercyjne-Na tych samych warunkach 3.0*, Wrocław 2014. Pewne prawa zastrzeżone na rzecz Autorów. Zezwala się na niekomercyjne wykorzystanie treści pod warunkiem wskazania Autorów jako właścicieli praw do tekstu oraz zachowania niniejszej informacji licencyjnej tak długo, jak tylko na utwory zależne będzie udzielana taka sama licencja. Tekst licencji dostępny na stronie: <http://creativecommons.org/licenses/by-nc-sa/3.0/pl/>

# Spis treści

## I. Wstęp

<b>1. Wstęp</b>	7
1.1. Quadcopter	7
1.1.1. Własności obiektu	7
1.2. Dostępne rozwiązania	7
1.3. Cel pracy	7

## II. Zadania

<b>2. Stabilizacja w punkcie</b>	12
2.1. Czujniki odległości	12
2.1.1. Dobór czujników odległości	12
2.1.2. Rozmieszczenie czujników odległości	12
2.2. Interfejs do istniejącego oprogramowania i sprzętu	12
2.2.1. Flying Machine Arena	12
2.2.2. Rozpoznanie protokołu quadcoptera	12
2.2.3. Specyfikacja interfejsu	12
2.2.4. Rozpoznanie możliwości platformy	12
2.2.5. Elektronika	12
2.2.6. Oprogramowanie interfejsu	12
2.2.7. Testy	12
2.3. Zdalne debugowanie	12
2.3.1. Dostępne rozwiązania	12
2.3.2. Link radiowy	12
2.3.3. Przegląd rozwiązań FPV	12
2.4. Algorytm stabilizacji w punkcie	12
2.4.1. Algorytmy rozpoznawania przesunięcia	12
2.4.2. Implementacja algorytmu	13
<b>3. Stabilizacja w pomieszczeniu</b>	15
<b>4. Unikanie kolizji</b>	17
<b>5. Autopilot</b>	19

## III. Testy

<b>6. Testy stabilizacji</b>	23
<b>7. Testy unikania kolizji</b>	25
<b>8. Testy autopilota</b>	27

## IV. Zakończenie

<b>9. Podsumowanie</b>	31
<b>10. Dodatki</b>	33
<b>11. Bibliografia</b>	35



Część I

**Wstęp**



# 1. Wstęp

## 1.1. Quadrocopter

### 1.1.1. Własności obiektu

## 1.2. Dostępne rozwiązania

## 1.3. Cel pracy





Część II

**Zadania**





## 2. Stabilizacja w punkcie

### 2.1. Czujniki odległości

#### 2.1.1. Dobór czujników odległości

Specyfikacja potrzeb

Dostępne rozwiązania

Dobór czujników

#### 2.1.2. Rozmieszczenie czujników odległości

Przegląd rozwiązań

Propozycja rozmieszczenia

### 2.2. Interfejs do istniejącego oprogramowania i sprzętu

#### 2.2.1. Flying Machine Arena

#### 2.2.2. Rozpoznanie protokołu quadrocoptera

#### 2.2.3. Specyfikacja interfejsu

Przygotowanie specyfikacji interfejsu

Wybór interfejsu

#### 2.2.4. Rozpoznanie możliwości platformy

#### 2.2.5. Elektronika

Projekt

Uruchomienie

#### 2.2.6. Oprogramowanie interfejsu

#### 2.2.7. Testy

### 2.3. Zdalne debugowanie

#### 2.3.1. Dostępne rozwiązania

Ground Station

#### 2.3.2. Link radiowy

#### 2.3.3. Przegląd rozwiązań FPV

## Wstęp

VO (ang. Visual Odometry) jest to proces estymacji ruchu pojazdu poprzez badanie zmian ruchu dzięki zdjęciom otrzymanym z pokładowej kamery. Warunki dobrej estymacji to

- odpowiednie oświetlenie sceny,
- dostosowanie szybkości zmienności otoczenia do użytego algorytmu, preferuje się raczej scenę statyczną,
- wybór odpowiednich zdjęć do przetwarzania.

## Zalety

Ze względu na formę zawodów, quadcopter będzie poruszał się w zamkniętym pomieszczeniu, gdzie jak wiadomo sygnał GPS zawodzi. Dlatego zdecydowaliśmy się na użycie kamery pokładowej. Wizualna odometria może współpracować z innymi rozwiązaniami, dlatego finalnie do stabilizacji w punkcie system wizyjny zostanie połączony z czujnikami laserowymi oraz ultradźwiękowymi. Zminimalizuje to błąd estymacji ruchu, a co za tym idzie, będziemy mogli przeciwdziałać dryfowi.

## Ogólny schemat algorytmu SVO

Program SVO (Semi-direct Monocular Visual Odometry) jest zaimplementowany w Rosie. Wykorzystuje on obrazy dostarczone z pokładowej kamery. Quadcopter zbiera informacje ze środowiska poprzez zdjęcia w dyskretnych chwilach czasu

$$I_1, I_2, \dots, I_n.$$

Odpowiednie macierze transformacji opisują relację pomiędzy dwiema pozycjami kamery

$$A_k = \begin{bmatrix} R_{k,k-1} & T_{k,k-1} \\ 0 & 1 \end{bmatrix}.$$

Pozycję kamery można obliczyć następująco

$$C_k = C_{k-1} \cdot A_k,$$

począwszy od znanej pozycji  $C_0$ . Głównym zadaniem algorytmu jest obliczenie transformacji  $A_k$ , po to żeby obliczyć pozycję kamery  $C_k$ , a co za tym idzie jej trajektorię, wykorzystując do tego obrazy  $I_k$ .

## Kalibracja kamery

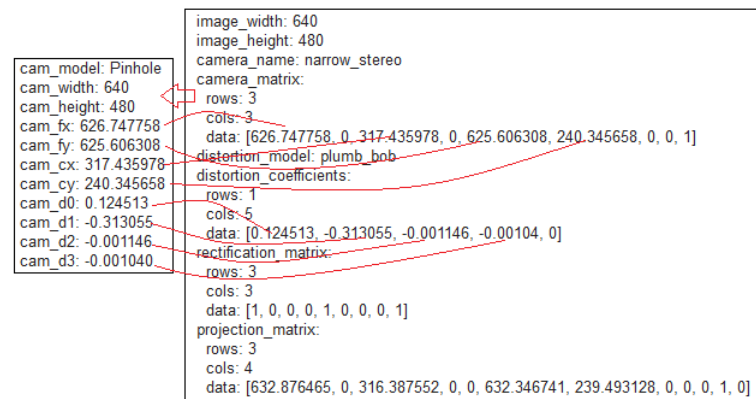
Do uruchomienia algorytmu potrzebny jest plik kalibracyjny. Kalibrację kamery wykonano standardowym narzędziem zaimplementowanym w Rosie w paczce `camera_calibration`. Wyniem tej operacji jest plik `*.yaml`. Na potrzeby algorytmu przekonwertowano plik `*.yaml` na format `*.yaml`.

Rysunek 2.1 przedstawia ręczne przekonwertowanie plików kalibracyjnych.

## Rezultat algorytmu

Do dalszego przetwarzania informacji dostarczonych z działającej paczki SVO można wykorzystać wiadomość publikowaną poprzez topic `/svo/points`.

### 2.4.2. Implementacja algorytmu



Rysunek 2.1. Konwersja z \*.yaml na \*.yaml

```

header:
  seq: 5892
  stamp:
    secs: 1418673322
    nsecs: 170325262
  frame_id: /world
ns: trajectory
id: 8195
type: 1
action: 0
pose:
  position:
    x: -0.00851521046703
    y: -0.00349843886924
    z: 0.0106530645546
  orientation:
    x: 0.0
    y: 0.0
    z: 0.0
    w: 0.0
scale:
  x: 0.006
  y: 0.006
  z: 0.006

std_msgs/Header header
uint32 seq
time stamp
string frame_id
string ns
int32 id
int32 type
int32 action
geometry_msgs/Pose pose
geometry_msgs/Point position
float64 x
float64 y
float64 z
geometry_msgs/Quaternion orientation
float64 x
float64 y
float64 z
float64 w
geometry_msgs/Vector3 scale
float64 x
float64 y
float64 z

```

Rysunek 2.2. Wycinek informacji zawartych w topicu /svo/points

### **3. Stabilizacja w pomieszczeniu**





## 4. Unikanie kolizji



## 5. Autopilot



Część III

Testy



## 6. Testy stabilizacji





## 7. Testy unikania kolizji



## 8. Testy autopilota



Część IV

**Zakończenie**



## 9. Podsumowanie





## 10. Dodatki



## 11. Bibliografia