CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Bachelor's Thesis

Pavel Zedník

Relative visual localization in swarms of unmanned aerial vehicles

Department of Cybernetics

Thesis supervisor: **Dr. Martin Saska**

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Pavel Z e d n í k
Study programme:	Cybernetics a Robotics
Specialisation:	Robotics
Title of Bachelor Project:	Relative Visual Localization in Swarms of Unmanned Aerial Vehicles

Guidelines:

The aim of this thesis is to design, implement and verify a system providing relative localization of particles of swarms of UAVs based on visual information given by onboard sensors.
Work plan:

1. To implement and verify methods for filtering of sensor data providing positions of neighbors in the swarm. [1-3]
2. To design and implement methods for fusion of multiple-data from swarm particles. [1-3]
3. To integrate all system components including sensor data reading and communication between the particles.
4. To adapt the developed system for on-line utilization with onboard Gumstix Overo computer. [4]
5. To verify the system with a set of camera modules developed for swarm of unmanned aerial vehicles.

Bibliography/Sources:

[1] Jetto, L.: Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted Kalman filter. (1999). Control Engineering Practice, vol. 7, no.6, pp.763-771.
[2] Kong, F.; Chen, Y.; Xie, J.; Zhou, Z.: Mobile Robot Localization Based on Extended Kalman Filter. 2006 6th World Congress on Intelligent Control and Automation, pp. 9242-9246.
[3] Kim, J.; Kim, Y.; Kim, S.: An Accurate Localization for Mobile Robot Using Extended Kalman Filter and Sensor Fusion. (2008) Neural Networks, pp.2928-2933.
[4] Gumstix software development. [online]. [cit. 2011-12-12]. http://gumstix.org/software-development.html

Bachelor Project Supervisor: Ing. Martin Saska, Dr. rer. nat.

Valid until: the end of the winter semester of academic year 2012/2013

[illegible]
prof. Ing. Vladimír Mařík, DrSc.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
-6-
L.S.

[illegible]
prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 9, 2011

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Pavel Zedník
Studijní program:	Kybernetika a robotika (bakalářský)
Obor:	Robotika
Název tématu:	Vzájemná vizuální lokalizace členů roje bezpilotních letounů

Pokyny pro vypracování:

Cílem práce je navrhnout a implementovat systém umožňující relativní lokalizaci helikoptér roje na základě vizuální informace poskytované palubními senzory.
Plán prací:
1. Implementovat a ověřit metody pro filtraci senzorických dat poskytující informaci o poloze sousedů v roji. [1-3]
2. Navrhnout a implementovat metody pro fúzi dat z několika entit roje. [1-3]
3. Integrovat všechny komponenty systému, včetně vyčítání senzorických dat a komunikace mezi členy roje.
4. Přizpůsobit systém pro použití na palubním počítači Gumstix Overo. [4]
5. Verifikovat výsledný systém s použitím sady kamerových modulů vyvinutých pro roj bezpilotních letounů.

Seznam odborné literatury:

[1] Jetto, L.: Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted Kalman filter. (1999). Control Engineering Practice, vol. 7, no.6, pp.763-771.
[2] Kong, F.; Chen, Y.; Xie, J.; Zhou, Z.: Mobile Robot Localization Based on Extended Kalman Filter. 2006 6th World Congress on Intelligent Control and Automation, pp. 9242-9246.
[3] Kim, J.; Kim, Y.; Kim, S.: An Accurate Localization for Mobile Robot Using Extended Kalman Filter and Sensor Fusion. (2008) Neural Networks, pp.2928-2933.
[4] Gumstix software development. [online]. [cit. 2011-12-12]. http://gumstix.org/software-development.html

Vedoucí bakalářské práce: Ing. Martin Saska, Dr. rer. nat.

Platnost zadání: do konce zimního semestru 2012/2013

Mařík

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 12. 2011

Declaration

I hereby declare that I have completed this thesis independently and that I have used only the sources (literature, software, etc.) listed in the enclosed bibliography.

In Prague on 28.5.2012 [illegible]

Acknowledgements

I would like to thank my supervisor, Dr. Martin Saska for his guidance and help throughout this project.

I would also like to thank my parents for their support during my studies.

Abstract

The aim of this thesis is to develop and implement a system providing relative localization in swarms of unmanned aerial vehicles based on visual information provided by an on-board camera system. The basic methods for localization are discussed and usage of the Kalman filter in data fusion is shown. A thesis is developed in collaboration with the COLOS project, which is focused on development of a complex system for controlling the swarm of UAVs. A probability model of the patter recognition system in the camera module is also prepared. Developed algorithm for the data fusion in swarms of helicopters is implemented and tested on the camera module.

Abstrakt

Cílem této práce bylo navrhnout a implementovat systém pro relativní lokalizaci bezpilotních helikoptér roje na základě vizuální informace poskytnuté palubním kamerovým systémem. Jsou probrány základní metody vzájemné lokalizace a použití Kalmanova filtru pro fůzi dat. Práce je součástí projektu COLOS, jehož cílem je navrhnout komplexní systém pro řízení robotického roje. V rámci práce byl dále zpracován pravděpodobnostní model kvality detekce blobu v pracovním prostoru kamerového systému. Vyvinutý algoritmus pro fůzi dat z více helikoptér byl implementován a poté i otestován pro běh ve spolupráci s kamerovým modulem.

Contents

List of Figures

List of Tables

1 Introduction

The localization is one of the crucial elements of autonomous robotic systems. To be able to localize itself, each robot requires a localization measurement, which provides the feedback for the robot actions and response to the situation in the surrounding environment. But all this gathered information include uncertainty, which complicates determining the position.

Using the larger number of robots that are able to cooperate (i.e., the swarm of robots), many measurements with different uncertainties can be done in one time. The goal is to optimally combine the measurements made by surrounding robots in swarm. This allows to reduce resulting uncertainty and obtain better estimate of measured data. One of the possible and frequently used techniques, which combines the information from different sources is called Kalman filter.

This thesis is part of the COLOS project, which aims at develop the control system for swarms of autonomous helicopters, which will allow to localize and control members in the swarm. The thesis is focused specifically on the topic of cooperative localization based on the fusion of position data acquired from the vision system developed for swarm members.

In this thesis, we will first discuss a basic localization methods, derive the equations for the Kalman filter and show the algorithm performance. We also describe the hardware platform on which results of this thesis were implemented and tested.

The core of thesis is the development of the data fusion algorithm. The main principle of the algorithm is that each swarm member uses the vision system to relatively localize surrounding members. The information on their position is then transmitted to the other members. So, each member receiving localization estimates of his own. These estimates are then fused and resulting position is used in the internal estimator. For more detailed description and algorithm derivation, see Section 6.

In addition, the function describing the probability of localization in a certain place of camera workspace is also implemented. Using this function during the controlling of swarm movements allows to prevent the swarm members from leaving the area in which they can be detected by other swarm members.

2 Localization methods

In this section we will first describe some basic localization techniques in robotics. The second part is devoted to the distribute localization methods.

2.1 Basic localization

The basic methods for localization can be divided into two groups, the method based on relative and absolute position measurements.

2.1.1 Relative localization

The relative position localization, sometimes called *a dead reckoning*, is based on knowledge of previous position which, is updated according to the speed and direction of movement and time passed from the last known position. This method has been used for a long time, originally for estimating the position on boat or plane [5, 1]. Because the estimates are based on the previous positions the error in the estimate increases in time.

Odometry Odometry is one of the most often used methods for the positioning. It is based on the integration of information provided by a moving sensors. The common example is the usage of wheel encoders to count the rotation of the robot wheels [2]. By integration of this information, it is possible to estimate position of moving robot. However, the error quickly increases with the travelled distance and measurement becomes very inaccurate. With all these drawbacks, odometry is still the easiest method for the position estimation.

Inertial navigation The inertial navigation uses motion sensors (accelerometers) and the rotation sensors (gyroscopes), to compute position, orientation and velocity of moving object [17]. The principle is similar to the odometry, the information from sensors are integrated to obtain position estimate. This also leads to increasing error with the travelled distance.

2.1.2 Absolute localization

Unlike the relative localization the absolute localization method is independent on the previous estimate, it is derived only from the current measurement. This indicates the advantage of not increasing error over the motion. There are two main methods for absolute localization, one using the landmarks and the other using maps.

Landmark This method is based on the usage of landmarks which are detected by a robot. There are two types of landmarks, the active and passive ones.
Active landmarks actively send position information, from which robot determine its position. To determine the position from the information provided by landmarks, the triangulation or the trilateration is often used [23]. The triangulation is based on the measurement of the angles to the landmarks and the trilateration on measurement of distances to the landmarks. An example of active landmark system is the GPS which uses the trilateration to determine position on the Earth.
The passive landmarks do not transmit any signal, the robot have to actively search and detect them. The passive landmarks can be artificial or natural. Artificial landmarks are the landmarks designed to be detected by robots. For example, they may be various coloured geometric figures [6]. The natural landmarks are the landmarks which haven't been designed to the usage with robots, they have already been part of robots' environment. The examples of natural landmarks can be roads, trees or windows [3].

Map A map based positioning uses the model matching between information from a sensor and a priori known geometric or topological map. This approach uses the geometric features to compute localization of the robot [23].

2.2 Sensor fusion

The Sensor fusion is based on the algorithms solving the position problem by combining the estimates from the position measurements based on the relatives and absolute localization methods. While combing the measurements from a more sensors, the problem of estimation of the resulting measurement occurs. The importance of the sensor fusion is the ability to obtain a more accurate estimate and reduce the uncertainty. If the information acquired from each sensors differs, the fusion can provide a complex information about a situation.

A tool for the sensor fusion presented in this thesis is called Kalman filter. Introduction to the filter and derivation of the equations and the algorithm is shown in the Section 3. The basic usage for the position of the tracked object is proposed in [27]. Also the framework for the robot localization using the Bayesian estimation and the Kalman filtering is presented in [20] and the approach for the collective localization in [21] and [25].

The usage of the Kalman filter as the tool for the fusion of the measurements from a more sensors is often presented. See [12] where the Extended Kalman filter is used to estimate the position of the robot by fusing a data from the ultrasonic satellite and the inertial navigation system. The similar approach is shown in [10] where the odometric and sonar sensor measurements are fused by the Extended Kalman filter. For the other example of the data fusion from the sensors providing the absolute and relative measurements, see [14].

Considering the swarm of cooperating robots, the data fusion algorithms are gaining the importance. A distributed localization in a group of robots is experimentally studied by [15]. The comparison of the Kalman estimation and the Triangulation techniques for the target localization is evaluated in the [28].

A several papers present the applied algorithms for the cooperative localization using the data fusion. At first, see [22] where the vision based system for the cooperative object detection, localization and tracking using the Wireless Sensor Networks is presented.

In [11] the multi robot visual localization is presented. The paper introduces the parallel fusion of the measurements from more robots. This principle is used as a part of the developed algorithm in this thesis.

The last presented example of the cooperative localization is the [19].The localization methods in a group of robots equipped with the range finders and targets are considered.

3 Kalman Filter

The Kalman filter, presented by Rudolf E. Kálmán in 1960, is considered to be one of the greatest achievements in the estimation theory in the twentieth century. With wide range of usage, from the trajectory estimator for the Apolo program, automated missile guidance system to the receiver of the GNSS [8], the Kalman filter is now the basic tool for a system estimation.

The Kalman filter is a recursive filter estimating the state of a dynamics system from a series of measurements. Under certain assumptions it is also optimal with the respect to virtually any criterion that makes sense [16].

In this section we will derive the equations and describe the algorithm for the Kalman Filter.

3.1 Equations

The Kalman filter addresses the problem of estimation the state $x \in \mathbb{R}^n$ of the discrete-time system [26, 7]. The system model describes how the state of the system evolves over time. The state equation is

$$x_k = Ax_{k-1} + w_{k-1}, \tag{1}$$

where A is an $n \times n$ matrix and w_{k-1} is the process noise reflecting interfering influences in the system. The measurement equation

$$z_k = Hx_k + v_k \tag{2}$$

describes how the measurements $z \in \mathbb{R}^m$ are related to the states. The symbol H represents $m \times n$ matrix and $z \in \mathbb{R}^n$ is the noise in the measurement.

The process and measurement noises are assumed to be zero mean Gaussian white noises independent upon each other.

$$p(w) \sim N(0, Q) \tag{3}$$

$$p(v) \sim N(0, R) \tag{4}$$

We will define $\hat{x}_k^- \in \mathbb{R}^n$ to be *a priory* state estimate in the step k known from the step prior to the step k and $\hat{x}_k \in \mathbb{R}^n$ to be *a posteriori* state estimate in the step k known from the measurement z_k [18]. *A priory* and *a posteriori* estimate errors are

$$e_k^- = x_k - \hat{x}_k^-, and \tag{5}$$

$$e_k = x_k - \hat{x}_k. \tag{6}$$

The covariance of the *a priori* estimate error is then

$$P_k^- = E[e_k^- e_k^{-T}], \tag{7}$$

and the covariance of the *a posteriori* estimate error is

$$P_k = E[e_k e_k^T]. \tag{8}$$

In deriving the equations for the Kalman filter we begin with finding the equation that computes the *a posteriori* estimate of $\hat{x}_k$ as linear combination of *a priori* estimate $\hat{x}_k^-$ and a weighted difference between the measurement z_k and prediction of the measurement $H\hat{x}_k^-$ [26] as

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-). \tag{9}$$

The $n \times m$ matrix K is called Kalman gain and minimises the *a posteriori* error covariance equation 6. The Kalman gain is then

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}. \tag{10}$$

Let us consider what happens, if the noise in the measurement is close to zero.

$$\lim_{R_k \to 0} K_k = H^{-1} \tag{11}$$

The covariance of the measured noise is close to zero and the Kalman filter weights the residual more heavily which means that actual measurement z_k is trusted more and the predicted measurement $H\hat{x}_k^-$ is trusted less.

Another situation occurs if the *a priori* estimate error covariance P_k^- is close to zero.

$$\lim_{P_k^- \to 0} K_k = 0 \tag{12}$$

The Kalman filter weights the residual less heavily which means that z_k is trusted less and the predicted measurement $H\hat{x}_k^-$ is trusted more.

3.2 Algorithm

We will shortly describe the algorithm of Kalman filter. Equations for the Kalman filter can be divided into two groups, *prediction* equations and *correction* equations. Prediction equations project the current state and error covariance estimate to obtain the *a priori* estimate for the next time step. The correction equations incorporates a new measurement to the *a priori* estimate to obtain a new *a posteriori* estimate [26].

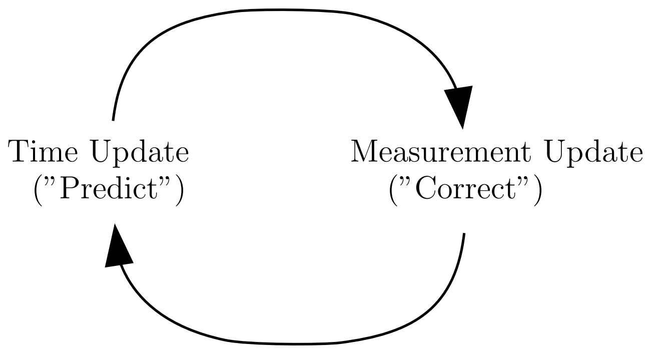


Figure 1: Kalman filtre cycle (source [26]).

During the prediction (also called the time update), the Kalman filter compute *a priori* state estimate $\hat{x}_k^-$ based on the *a posteriori* estimate $\hat{x}_{k-1}$ and also update uncertainty P_k^-.

$$\hat{x}_k^- = A\hat{x}_{k-1} \tag{13}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{14}$$

The correction step (also called measurement update) is used only when there is a measurement. The equations correct the most recent beliefs by incorporating the information from the direct measurements.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{15}$$

$$P_k = (I - K_k H)P_k^-, \tag{16}$$

where

$$K_k = P_k^- H^T(HP_k^- H^T + R)^{-1}. \tag{17}$$

The whole algorithm is repeated after each prediction and correction step with the previous *a posteriori* estimates used to predict the new *a priori* estimates.

4 Hardware platform

In this section we will introduce the platform on which the algorithms and software in this thesis are developed and tested.

For the actual swarm of the helicopters the MK L4-ME quadricopters are planned to be used. This helicopters will be fitted with the developed camera module running all software necessary for the localization and control among the swarm members.

4.1 Camera module

The camera module (see Figure 2), developed for the COLOS project in the Gerstner Laboratory for Intelligent Decision Making and Control at CTU, is the main tool used for developing and testing software in this thesis. The module is designated to be fitted on the helicopters in the swarm and provides capability for all necessary computations related to the swarm movement.

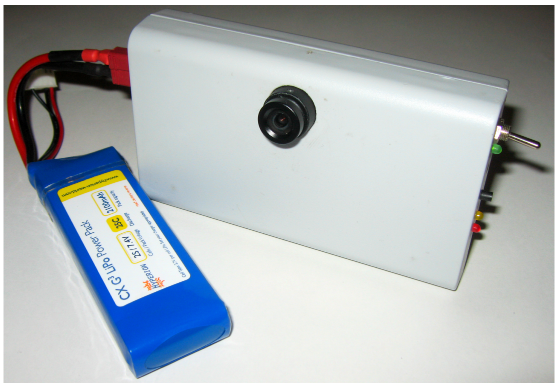


Figure 2: Camera module with battery.

4.1.1 Hardware

The hardware of the module consists of four electronic boards (see Figure 3). The main board is the Gumstix Overo board equipped with the OMAP 3503 processor and 802.11b/g wireless communication. The other boards are Caspa camera board fitted with the Aptina MT9V032 CMOS sensor, voltage regulator board and minicom board which provides power and connectivity to the main board [9].

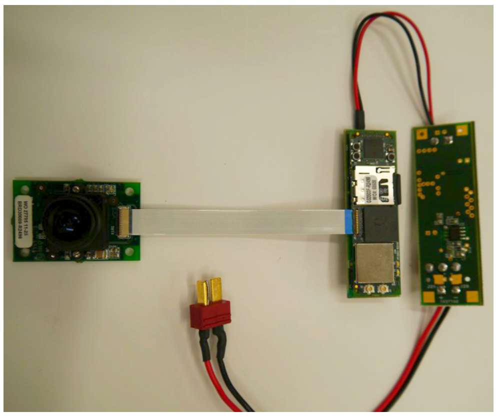


Figure 3: Connected boards (source [6]).

4.1.2 Software

The operating system used in the camera module is a Linux distribution tweaked for use on the Gumstix Overo board.

The administration of the module can be done by using a standard serial connection or over the WiFi connection. The module is set to be automatically connected to the open (without encryption) adhoc network with SSID COLOS and static IP addressing. IP address of the module is in range 10.10.40.17x, where x varies for each module. The connection to the module and remote controlling is possible over ssh with user *root* and no password [6].

4.1.3 Tracker Server

The main application running in the camera module is the tracker server. This application acts as a daemon providing information about the tracked objects. The tracker server captures image which is processed by blob finder using the camera. Determined coordinates of the tracked object are transmitted to the connected clients over UDP protocol. The tracker server is also able to use a wide range of options (e.q. camera resolution, blob size, frame-rate) specified in the configuration file.

4.1.4 Client applications

The camera module software also offers applications for the remote access to the camera.

The first application is the tracker client. It's a simple client side application for receiving the position of the tracked objects which are sent by the tracker server over UDP protocol.

The second application is the *tcam*, which is a more sophisticated application offering the interactive mode, where a window with the captured image is shown. The application itself also contains two blob finders (colour based and ring based). The main purpose of the application is direct testing of the blob finder algorithm on the images captured in the camera. Other useful features are the adjustment of the camera control setting and a creation of the setting file.

4.1.5 Patter detection

One of the key features of the camera module is the pattern detection. The blob finder algorithm is implemented both in the tracker server and *tcam* application (ring based finder is used by default).

The detection of the ring pattern (see Figure 4) is based on the image segmentation and finding two discs forming a black and white ring. The computed size of the blob is then compared to the expected size of used pattern (size of the ring).

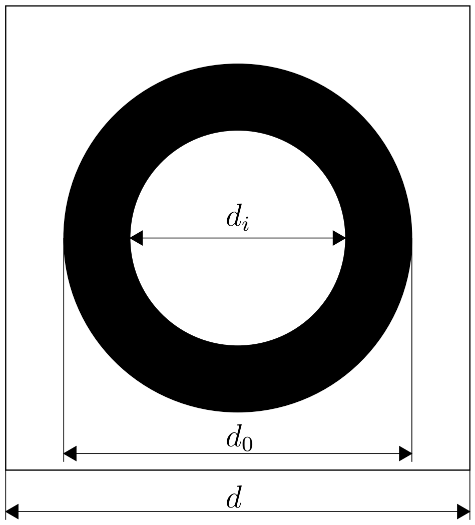


Figure 4: Pattern for the circle detector (source [6]).

5 Model of reliability

In this section a model of reliability of the relative localization using the image pattern (blob) recognition in the camera module is described. The main part of the model development is to create a function written in Matlab or C which can predict probability of the pattern recognition on specific coordinates related to the camera module. This will allow to control the swarm movement with the aim to keep swarm members in the sight of the camera. The development of the reliability model consists of three main parts.

- Calibration of the camera system to compensate properties of the optical system (see Section 5.1).
- Measuring the recognition rate in whole space around the camera (see Section 5.2).
- Evaluation of this measured data and creating function providing probability distribution similar to the behaviour established by measuring (see Section 5.3).

5.1 Camera calibration

The camera calibration have been done using the Camera calibration toolbox for matlab [4]. This software allows easy calibration from several images containing the calibration rig - the checker-board pattern (see Figure 5).

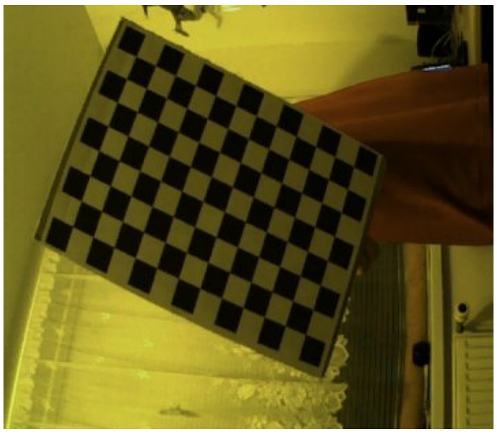
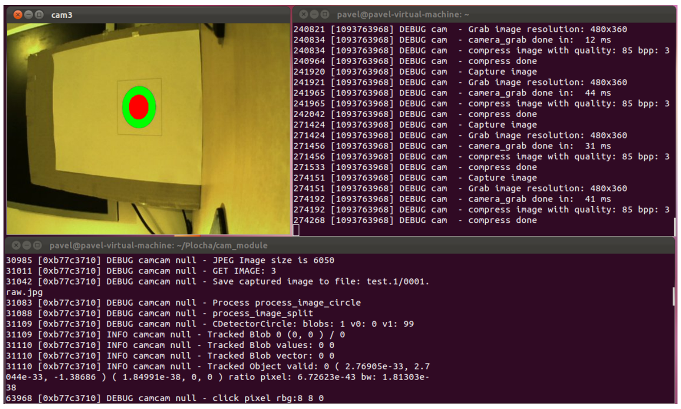


Figure 5: Image containing the calibration rig.

The camera module includes a client application *tcam*, which is suitable for a manual access to the camera server, adjusting the camera setting (contrast and brightness) and obtaining single images and processing them. See application interface in Figure 6. This software was used to obtain images required for the camera calibration.


cam3
pavel@pavel-virtual-machine: ~
pavel@pavel-virtual-machine: ~/Plocha/cam_module

```
240821 [1093763968] DEBUG cam  - Grab image resolution: 480x360
240834 [1093763968] DEBUG cam  - camera_grab done in:  12 ms
240834 [1093763968] DEBUG cam  - compress image with quality: 85 bpp: 3
240964 [1093763968] DEBUG cam  - compress done
241920 [1093763968] DEBUG cam  - Capture image
241921 [1093763968] DEBUG cam  - Grab image resolution: 480x360
241965 [1093763968] DEBUG cam  - camera_grab done in:  44 ms
241965 [1093763968] DEBUG cam  - compress image with quality: 85 bpp: 3
242042 [1093763968] DEBUG cam  - compress done
271424 [1093763968] DEBUG cam  - Capture image
271424 [1093763968] DEBUG cam  - Grab image resolution: 480x360
271456 [1093763968] DEBUG cam  - camera_grab done in:  31 ms
271456 [1093763968] DEBUG cam  - compress image with quality: 85 bpp: 3
271533 [1093763968] DEBUG cam  - compress done
274151 [1093763968] DEBUG cam  - Capture image
274151 [1093763968] DEBUG cam  - Grab image resolution: 480x360
274192 [1093763968] DEBUG cam  - camera_grab done in:  41 ms
274192 [1093763968] DEBUG cam  - compress image with quality: 85 bpp: 3
274268 [1093763968] DEBUG cam  - compress done
```

```
30985 [0xb77c3710] DEBUG camcam null - JPEG Image size is 6050
31011 [0xb77c3710] DEBUG camcam null - GET IMAGE: 3
31042 [0xb77c3710] DEBUG camcam null - Save captured image to file: test.1/0001.
raw.jpg
31083 [0xb77c3710] DEBUG camcam null - Process process_image_circle
31088 [0xb77c3710] DEBUG camcam null - process_image_split
31109 [0xb77c3710] DEBUG camcam null - CDetectorCircle: blobs: 1 v0: 0 v1: 99
31109 [0xb77c3710] INFO camcam null - Tracked Blob 0 (0, 0 ) / 0
31110 [0xb77c3710] INFO camcam null - Tracked Blob values: 0 0
31110 [0xb77c3710] INFO camcam null - Tracked Blob vector: 0 0
31110 [0xb77c3710] INFO camcam null - Tracked Object valid: 0 ( 2.76905e-33, 2.7
044e-33, -1.38686 ) ( 1.84991e-38, 0, 0 ) ratio pixel: 6.72623e-43 bw: 1.81303e-
38
63968 [0xb77c3710] DEBUG camcam null - click pixel rbg:8 8 0
```

Figure 6: tcam application interface with detected blob.

For the calibration, using the *tcam* application, contrast and brightness have been adjusted on the values appropriate to the conditions during the capturing. Then several images with variously tilted calibration rig have been captured. These images have been loaded into the calibration toolbox. The image corners were automatically extracted from the known number and size of printed squares (see Figure 7).

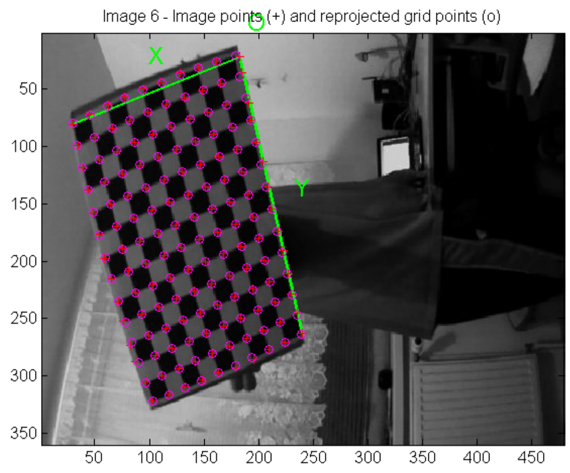


Figure 7: Calibration rig with extracted image corners.

After proceeding all images calibration parameters have been computed and stored in the file *Calib_Results.m*, which can be directly used by the tracker server in the camera module.

5.2 Measuring the camera workspace

The next step is the measurement of the camera workspace. The measurement has been done for circle patterns of various diameters (14, 7, 6 and 3.5 cm) with camera resolution 480x360 pixels which provides a good compromise between the achieved frame-rate and maximal measured distance.

The camera workspace has been measured using the tracker client running on PC. The information about the tracked object has been sent by tracker server running in the camera module.

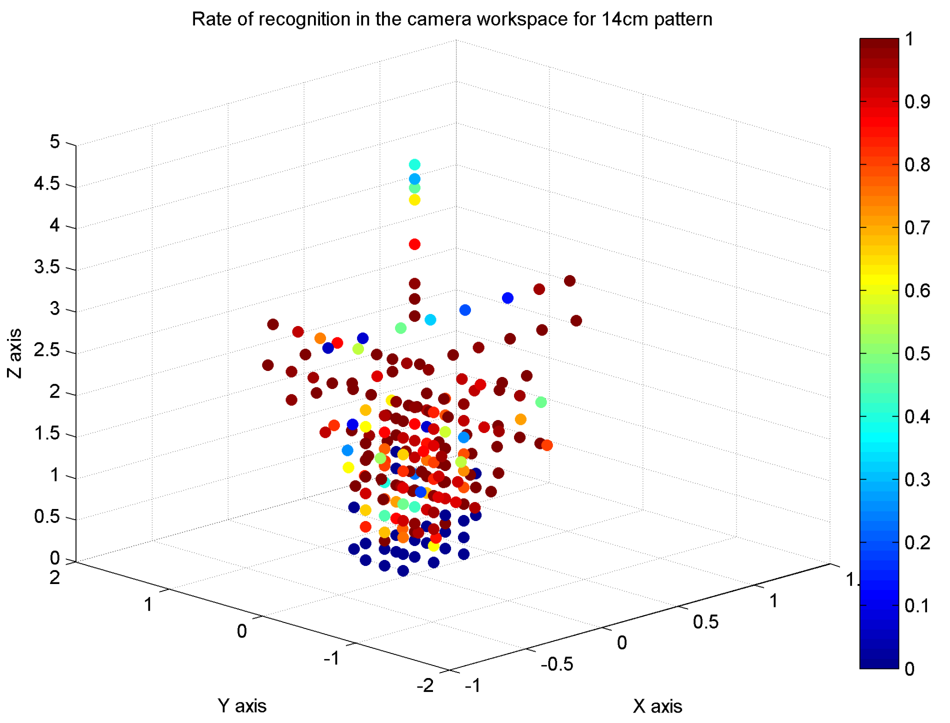


Figure 8: Rate of recognition in the camera workspace for 14 cm pattern.

Each pattern was placed in various places to fully determine the camera workspace and its properties. Resulting measured data were manually divided according to the exact measurement location. We created number of points in space by that to which a measured data are assigned. For each point, there is then a computed probability of pattern recognition in that location. Orientation of the coordinate system for these points depends on the camera placement, in this thesis we consider x axis to be horizontal direction, y axis vertical direction and z axis a distance from the camera.

These points create a map of probability of the pattern recognition in the camera workspace. See Figures 8 and 9 where the rate of the successful pattern recognition in each point is displayed by colour.

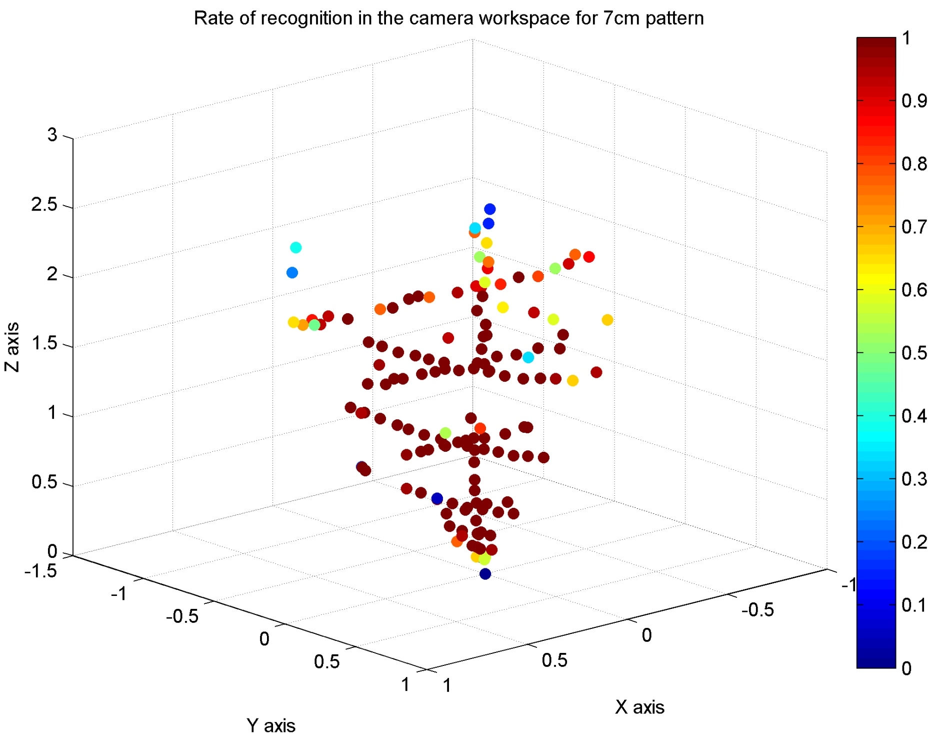


Figure 9: Rate of recognition in the camera workspace for 7 cm pattern.

To directly compare a reliability of the blob recognition according to the pattern size and distance, see Figure 10 where a reliability of recognition for each pattern placed in various distances is shown.

We can see that distance on which the pattern is correctly recognised increases with the pattern size. With a smaller distances (ca. 0.5 m - 1 m), every pattern is recognised with the reliability about 95%. With the increasing distance, a recognition rate for the smaller patterns is decreasing.

Taking the size of the pattern and maximal recognised distance into account, we can assume that best compromise are the patterns with 6 cm and 7 cm diameters which provide a good reliability of the recognition on a greater distances, while keeping the pattern relatively small.

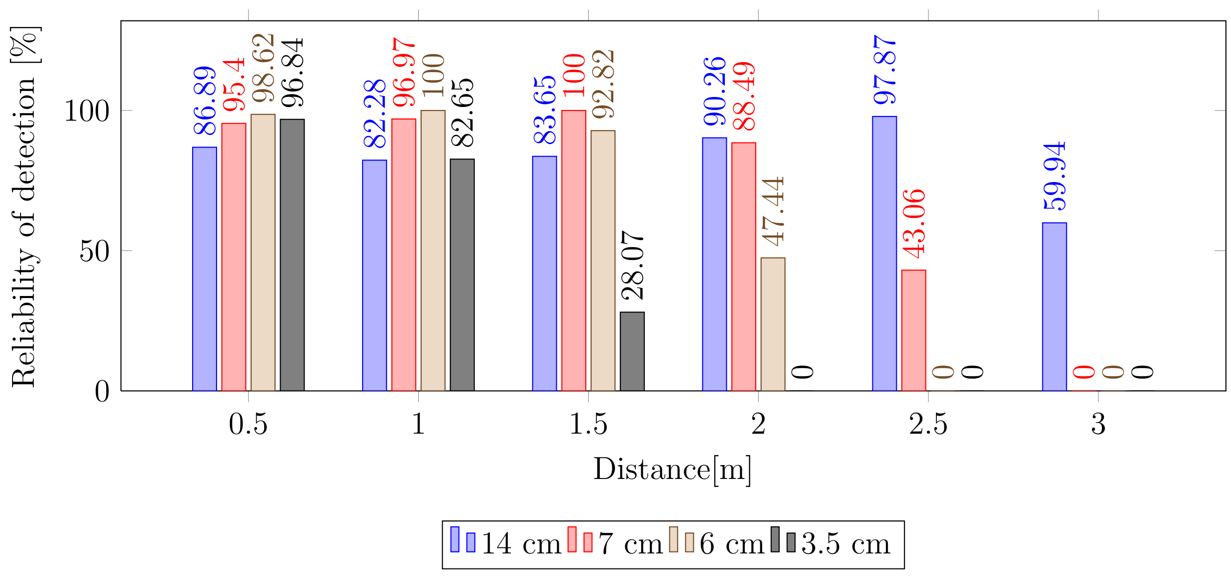


Figure 10: Rate of recognition for different blob sizes by distance on Z-axis.

5.3 Development of the probability model

The last step is a design of the probability model of the camera vision system.

The blob position is determined by a three basic parameters. A two angles (horizontal and vertical position of the blob in the camera image), and distance of the blob from camera (determined by the expected blob size). Due to its physical nature the correct recognition of the image pattern is limited only by these three parameters. For example, the reliability of the recognition drops for the very long or short distances (this is caused mainly by the blob size), or if the blob is on the edge of the camera viewing angle. The principle of the algorithm is based on this assumption.

At first, from the given Cartesian coordinates of the tracked pattern, the horizontal and vertical angles are computed (ϕ and α respectively). According to the blob size, the parameters of the probability distributions for both angles and distance of the blob from the camera are computed independently. By analysis of the measured camera workspace, the probability distributions have been chosen as follows. The middle part is constant with the measured average reliability (see Figure 12d), the marginal part with a zero probability of the recognition and the transition between these two parts estimated by the Hann window (see Figure 11). By multiplication of these parameters, the resulting estimate of the probability of recognition is then obtained.

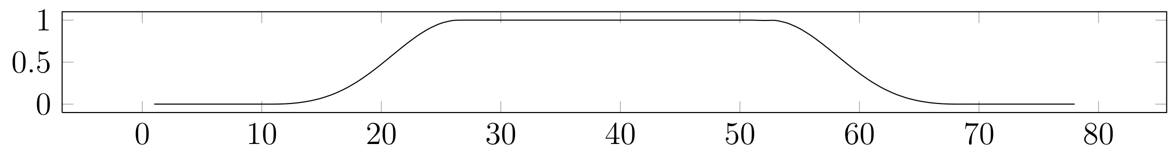
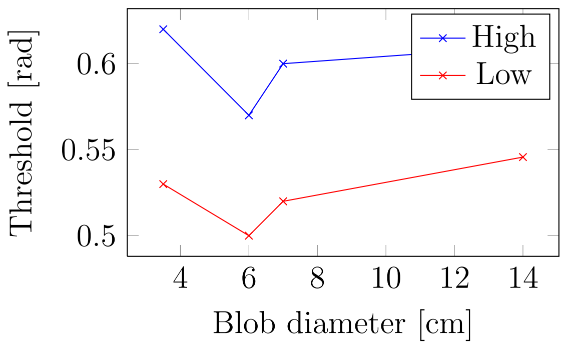


Figure 11: Probability distribution.

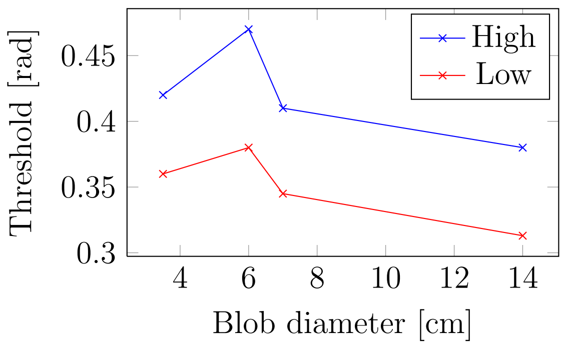
In essence, for the chosen probability distribution a five parameters have to be found. At first the average reliability of the detection in the central part of the camera workspace (basically the recognition rate in the most suitable part of the camera workspace). For the result of the experimental measurement, see Figure 12d. In the algorithm, the observed values were linearly approximated. The maximal detection rate corresponds with the pattern with a 6 cm diameter. For the smaller or bigger patterns, the recognition rate decreases.

The next four parameters represent the threshold values which divide the probability function into five parts (a maximal recognition rate in the middle, zero recognition rate and the Hann window on each side). Due to symmetry, for the ϕ and α angles, only two parameters are considered. These parameters in fact mean a position of the Hann window in the probability function. No dependence of these parameters on the blob size was found (see Figures 12a and 12b). So, for the purpose of the algorithm, the ϕ and α angles are considered to be constant.

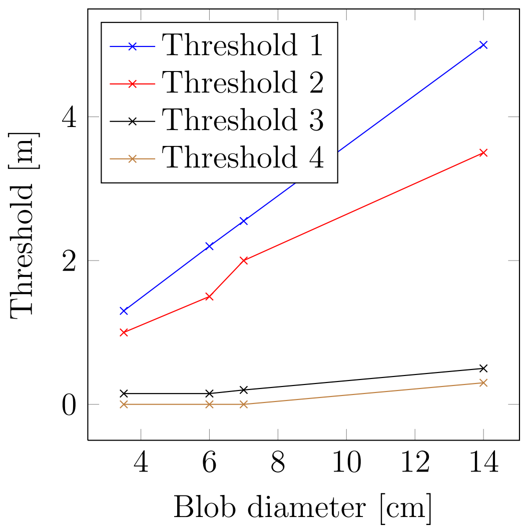
The probability function for the distance on the Z-axis is described by all four parameter. These parameters divide the probability function on part with the zero probability of recognition (i.e. to close or to far from the camera), a part with maximal probability of recognition (middle distances) and the transition between them. Parameters are proportional to the blob size (see Figure 12c) and therefore approximated by a linear function.



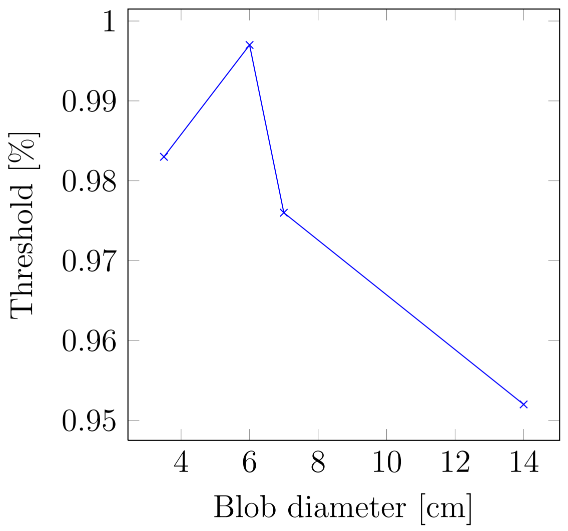
(a) A threshold values for the α angle according to the size of the blob.



(b) A threshold values for the ϕ angle according to the size of the blob.



(c) A threshold values for the distance on Z-axis according to the size of the blob.



(d) Average pattern detection reliability according to the size of the blob.

Figure 12: Measurements for determination of probability distribution parameters.

All presented parameters have been experimentally found for the the camera resolution 480x360 pixels and approximated for a various diameters of patterns. The created matlab function also allows the usage of more cooperating cameras (the function returns probability from a camera with the best detection rate).

For the results of the developed algorithm , see Figure 13. This figure shows the predicted rate of the recognition in the Y-Z plane (X-axis is considered to be zero) and the 7 cm pattern. Figure 14 shows the recognition rate in a whole camera workspace (points with a zero recognition rate are omitted).

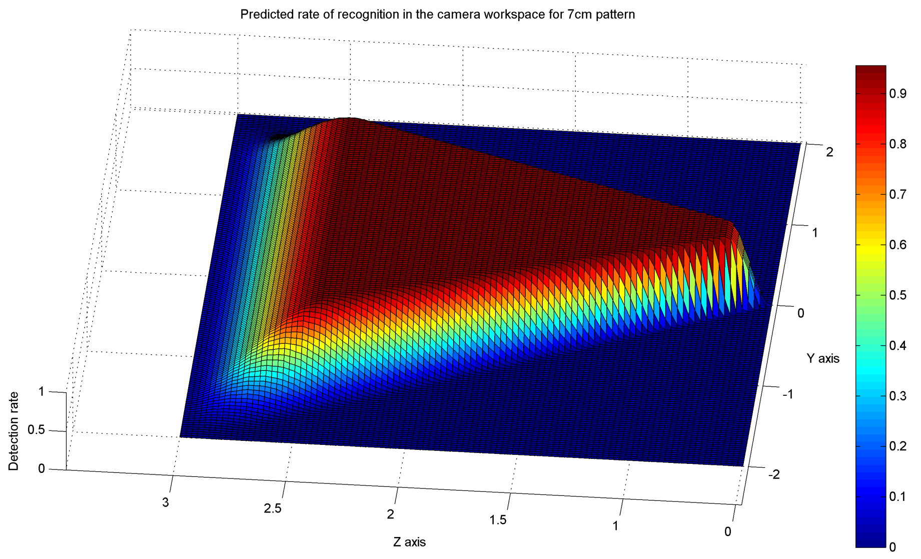


Figure 13: Predicted rate of recognition in the camera workspace for 7 cm pattern.

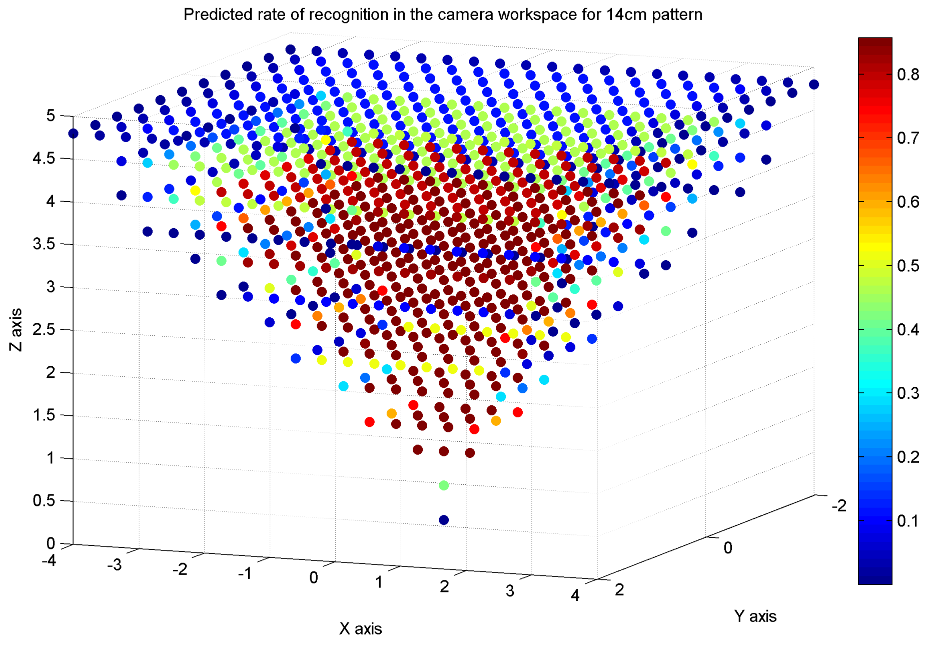


Figure 14: Predicted rate of recognition in the camera workspace for 14 cm pattern.

6 Relative localization

The second goal of this thesis was to develop and test the algorithm for the fusion of visual sensor data to obtain better position estimate of swarm members.

The goal was to compute a position of a helicopter from measurements made by surrounding helicopters fitted with video cameras. Each observing helicopter uses image data and its own known global position to compute the position of the observed helicopter in global coordinates. These computed positions are then combined to obtain more accurate position estimate.

The main idea of the algorithm is to provide a distributed method for the localization among a swarm members. Each helicopter observes surrounding members and computes their position using the vision system. This position estimates are then broadcast in the swarm. Each helicopter receives the position estimates made by the surrounding members. These position estimates are then fused using the algorithm described in [11]. The resulting position estimate is then used to update the internal Kalman filter [27], which provides the position and speed estimates for the helicopter.

In this section we will derive the algorithm used for the data fusion then we will describe the implementation and evaluate algorithm on generated datasets and experimentally measured data.

6.1 Algorithm

At first we will derive a theoretical solution of our data fusion algorithm. There are three main parts of the algorithm. The first is the position computing where the global position of the observed helicopter is computed from known the position of the observing helicopter and measured data from the vision system. The second part is the parallel fusion where measurements from more helicopters are merged. The last part is the serial fusion where we use Kalman filter to track the observed helicopter.

6.1.1 Position computing

The first step is computing the global position of the observed helicopter. Let P be a known position of the observing helicopter and M the position of the observed helicopter.

In this case we assume that Roll and Pitch angles are zero, so we consider only the Yaw angle denoted θ . From the image processing we obtain the distance between the observed and the observing helicopter d and two angles φ and α (see Figure 15).

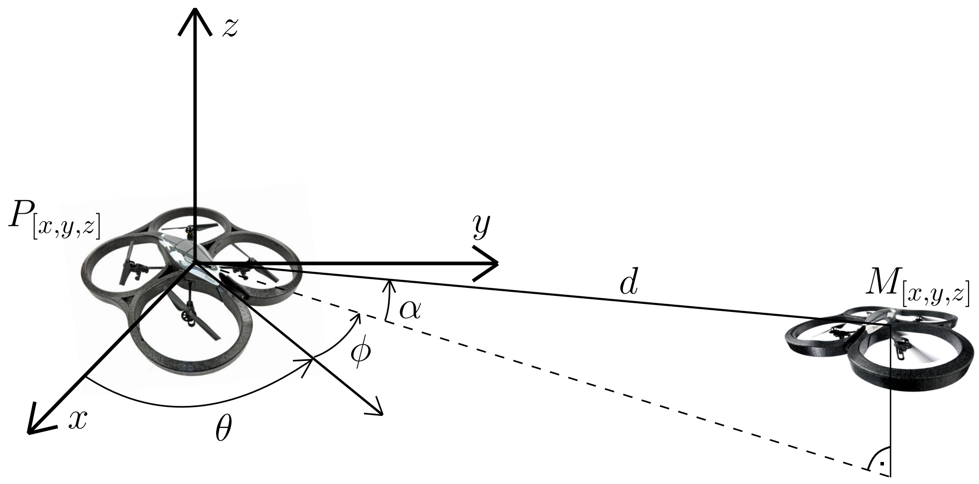


Figure 15: Coordinate system of observing helicopter P and camera measurements.

For the global position M of observed helicopter we can write:

$$x_M = x_P + d \cdot \cos\alpha \cdot \cos(\varphi + \theta) \tag{18}$$

$$y_M = y_P + d \cdot \cos\alpha \cdot \sin(\varphi + \theta) \tag{19}$$

$$z_M = z_P + d \cdot \sin\alpha. \tag{20}$$

The next step is to derive the covariance matrix for our transformed measurement.

We assume that known coordinates of the observing helicopter and measurements from the camera are independent so the measurement covariance matrix is

$$C = \begin{bmatrix} \sigma_{x_P}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{y_P}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{z_P}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\theta^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_d^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\varphi^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\alpha^2 \end{bmatrix}. \tag{21}$$

According to [24], we can obtain the covariance matrix for transformed coordinates as

$$C_t = J \cdot C \cdot J^T, \tag{22}$$

where J is the Jacobian matrix of M, which can be expressed as

$$J = \begin{bmatrix} 1 & 0 & 0 & -d \cdot \cos\alpha \cdot \sin(\varphi+\theta) & \cos\alpha \cdot \cos(\varphi+\theta) & -d \cdot \cos\alpha \cdot \sin(\varphi+\theta) & -d \cdot \sin\alpha \cdot \cos(\varphi+\theta) \\ 0 & 1 & 0 & d \cdot \cos\alpha \cdot \cos(\varphi+\theta) & \cos\alpha \cdot \sin(\varphi+\theta) & d \cdot \cos\alpha \cdot \cos(\varphi+\theta) & -d \cdot \sin\alpha \cdot \sin(\varphi+\theta) \\ 0 & 0 & 1 & 0 & \sin\alpha & 0 & -d \cdot \cos\alpha \end{bmatrix}. \tag{23}$$

6.1.2 Parallel fusion

The next step in our algorithm is parallel fusion for optimal combinations of measurements from more observing helicopters. Let assume that one helicopter with the unknown position is observed by two helicopters. Each of the observing helicopters compute position of the unknown one. Let us denote these computed positions as M_1 respectively M_2 with the covariance matrices C_{t1} and C_{t2}.

For the fusion, as described in [11] we use weighted least square criterion to get the optimal position M,

$$(M_1 - M)^T C_{t1}^{-1} (M_1 - M) + (M_2 - M)^T C_{t2}^{-1} (M_2 - M) \,. \tag{24}$$

After derivation of the equation 24 with respect to M we get the necessary condition of a local optimal solution

$$(M_1 - M)^T C_{t1}^{-1} + (M_2 - M)^T C_{t2}^{-1} = 0. \tag{25}$$

Then we can get the local optimal solution M,

$$M{=}M_1 + K\left(M_2 - M_1\right), \tag{26}$$

where K can be obtained as

$$K = \left(C_{t1}^{-1} + C_{t2}^{-1}\right)^{-1} C_{t2}^{-1} = C_{t1}\left(C_{t1} + C_{t2}\right)^{-1}. \tag{27}$$

Similar approach is then used to obtain updated covariance matrix, so we get

$$C = C_{t1} - K \cdot C_{t1}\,. \tag{28}$$

By this approach we have combined more measurements to get the optimal one.

6.1.3 Serial fusion

The next step is to use Kalman filter to track position of the observed helicopter. In our case we don't use any input so the matrix B is zero.

Basic equations of the Kalman filter are then:

Prediction:

$$x_k^- = Ax_{k-1}^+ \tag{29}$$

$$P_k^- = AP_{k-1}^+A^T + Q_{k-1} \tag{30}$$

Correction:

$$x_k^+ = x_k^- + K_k\left(z_k - Hx_k^-\right) \tag{31}$$

$$P_k^+ = \left(I - K_kH\right)P_k^- \tag{32}$$

$$K_k = P_k^- H^T \left(H P_k^- H^T + R_k \right)^{-1} \tag{33}$$

Our state vector consists of position on each axis of the coordinate system (x, y and z), and velocity (v_x , v_y and v_z). First three vectors are directly measured, the speeds are hidden states. The state vector x_k is

$$x_k = \begin{bmatrix} x_M \\ v_x \\ y_M \\ v_y \\ z_M \\ v_z \end{bmatrix}. \tag{34}$$

We also assume that the speeds along each axes are constant. We can deduce the rest of the system matrices from this

$$A = \begin{bmatrix} 1 & dt & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{35}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{36}$$

Matrix R_k is previously derived matrix C and matrix Q represents the noise in the system which is identified experimentally. The variance of the random noise added to the generated data is used for the generated dataset. In case of the measured dataset desired variances were estimated from the measured data and then manually adjusted to achieve the best performance.

6.2 Implementation

The algorithm described above, was implemented by using Matlab. This allowed quick testing, debugging and evaluating of the algorithm on testing datasets.

For the potential use on the camera module, which can be fitted on the helicopter, the algorithm was rewritten in C using Small Matrix Toolbox for C programmers [13]. This toolbox allows to create matrix object in C programs and use basic matrix operation above these objects (matrix multiply, determinant, computing of inverse matrix, etc.).

Evaluation of the algorithm on the camera modules have been done using the pair of applications developed and supplied with the camera module. For the communication the server-client architecture was used. The tracked object coordinates were obtained from the camera image by blob finder and then sent over UDP by the tracker server running in the camera module. Tracker clients running on the independent PC received object coordinates which have been then passed to the algorithm where the position fusion has been done.

6.3 Experimental results

In this part we present some experimental results of the implemented algorithm. The first tests used for the algorithm testing were made on the generated datasets. Other tests were made on real measured data. This have been done using two different pattern detectors at first the less accurate colour blob detector and then the improved circle blob detector implemented in the camera module.

6.3.1 Generated dataset

For the testing we first generated required data vectors for position of the observing helicopters and measurements to which then a pseudo-random noise with normal distribution was added. To achieve more realistic conditions the noise added to the second helicopter was higher then to the first one. In experiment the stationary helicopters observe a helicopter moving along the x axis (see Figure 16).

Because of the different noise in measurements of each helicopter we assume that the variances of the measurements from the second helicopter are twice as high as from the first one.

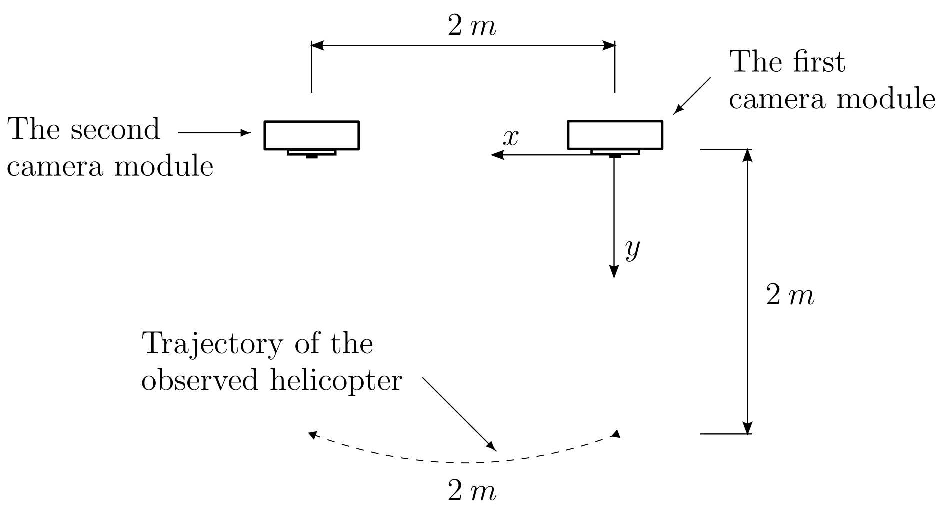


Figure 16: Arrangement of the experiment.

The results of the experiment (see Figures 17 and 18) shows a different noise in measurements from each helicopter. The first figure shows the position of the observed helicopter on x axis - i.e., moving two meters along the x axis. The second figure then shows the position on the y axis. From the resulting computed position (green colour) it is evident that the fusion algorithm was able to reduce the noise in measurements.

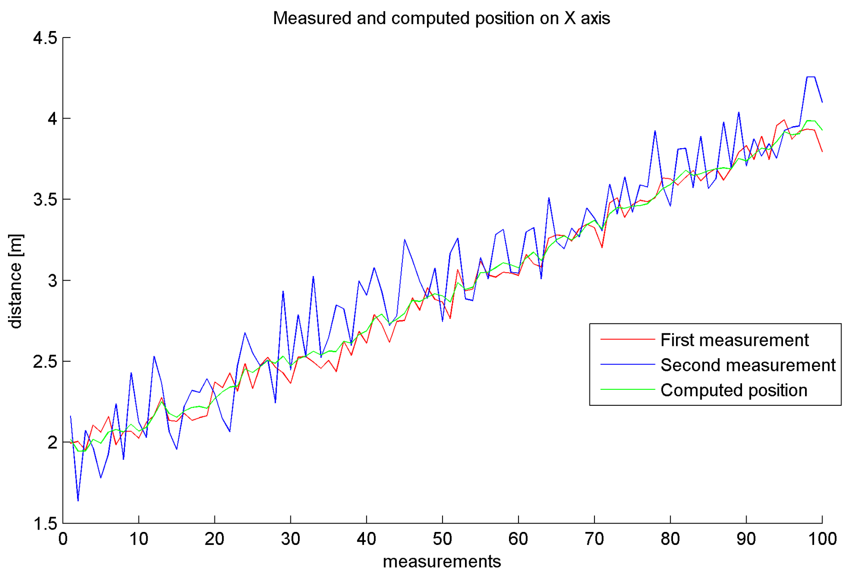


Figure 17: Measured and computed position on X axis.

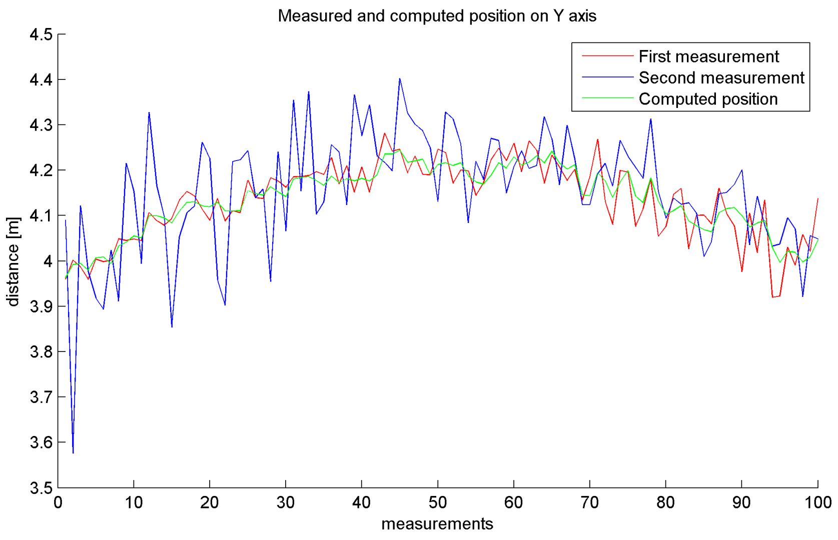


Figure 18: Measured and computed position on Y axis.

6.3.2 Colour detector

In this part we evaluate the datasets experimentally measured in the laboratory and then processed them by using the colour detector.

The measured object was performing a similar movement like in the generated dataset but this time moving along the y axis (see Figure 19).

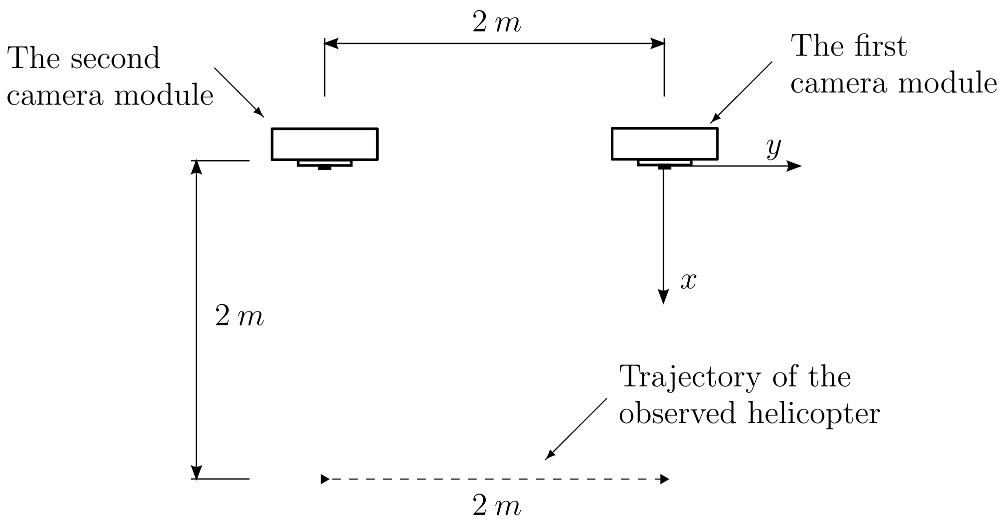


Figure 19: Arrangement of the experiment.

The Figures 20 and 21 show the results of data fusion. This two figures can be divided into three parts. In the first part, from the beginning to the measurement number 120 the object is observed only by the second helicopter because for the first helicopter the object is out of the camera. So even though the measurement from the second helicopter is accurate the result of the fusion is strongly disturbed by the noise from the vision system of the first helicopter. The second part, measurements from 120 up to the 180, are properly measured by both helicopters. For the rest of the figure the situation is similar to the first part but this time the measurement from the second helicopter is missing. As we can see this missing measurements strongly corrupt the resulting fusion.

We can get much better results by removing these incorrect values. Using prior knowledge about the location of these values (in the first part there is corrupted signal from the first helicopter and in the third part it is a signal from the second helicopter) it is possible to improve the results of the fusion algorithm.

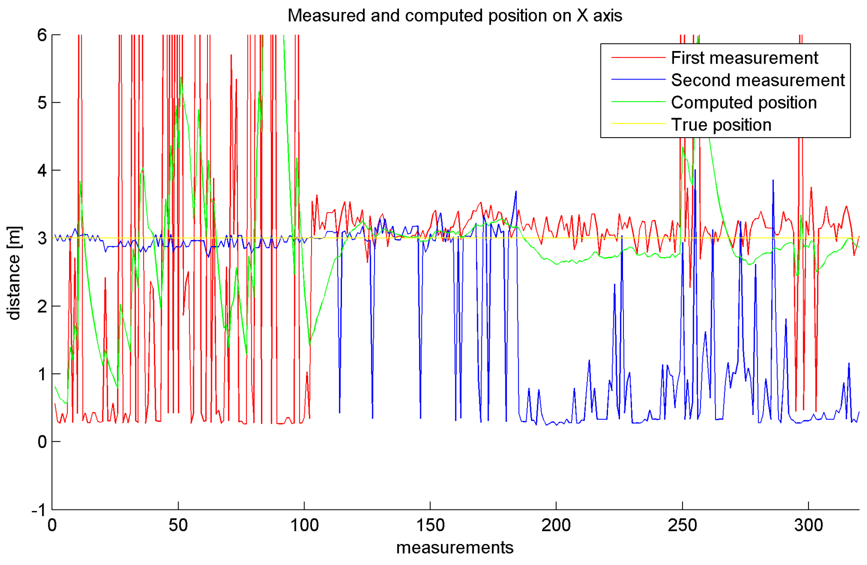


Figure 20: Measured and computed position on X axis.

The described approach is shown in Figures 22 and 23 which correspond to the Figures 20 and 21 mentioned earlier. In this case the first part (from the beginning to measurement number 120) is estimated by using data only from the second helicopter and the third part (measurements from 180 to 320) only by data from the first helicopter. As we already stated this improves performance if a measurement from one helicopter is missing.

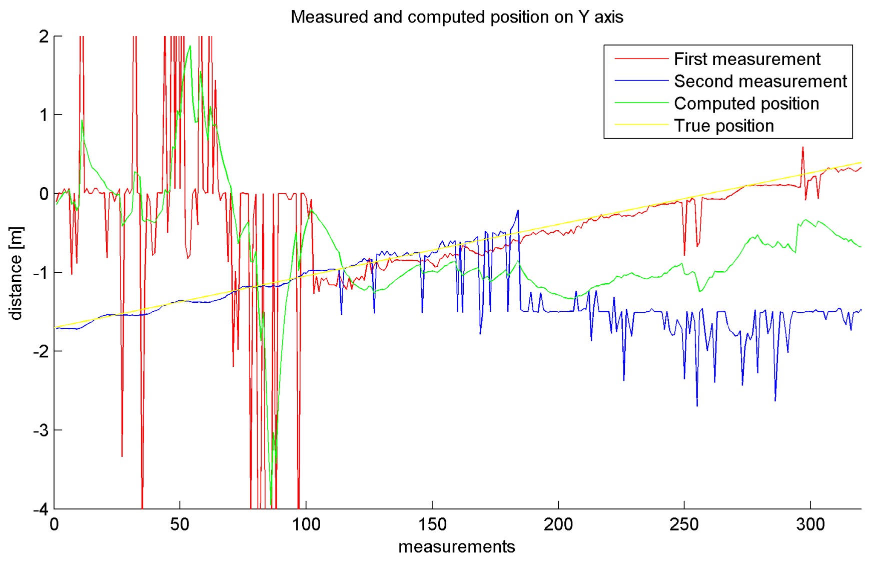


Figure 21: Measured and computed position on Y axis.

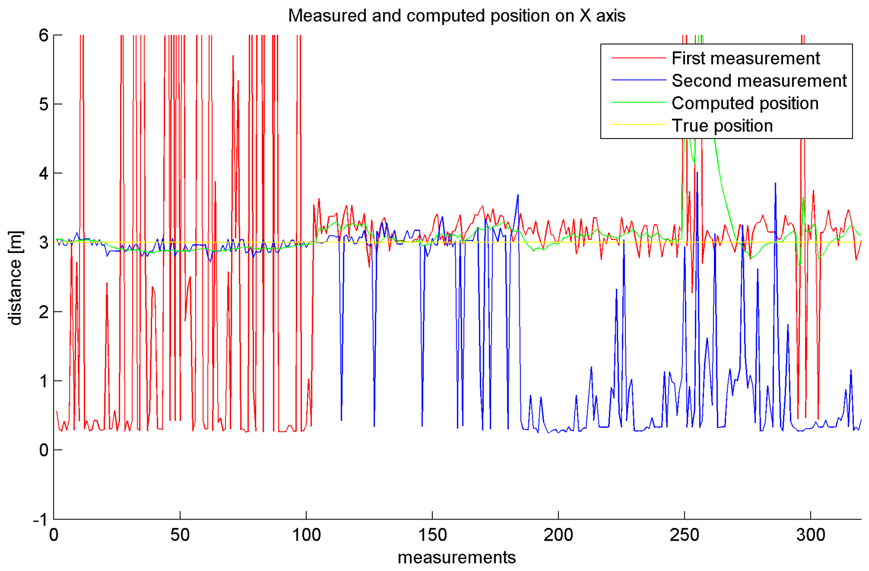


Figure 22: Measured and computed position on X axis with prior knowledge about the measurement.

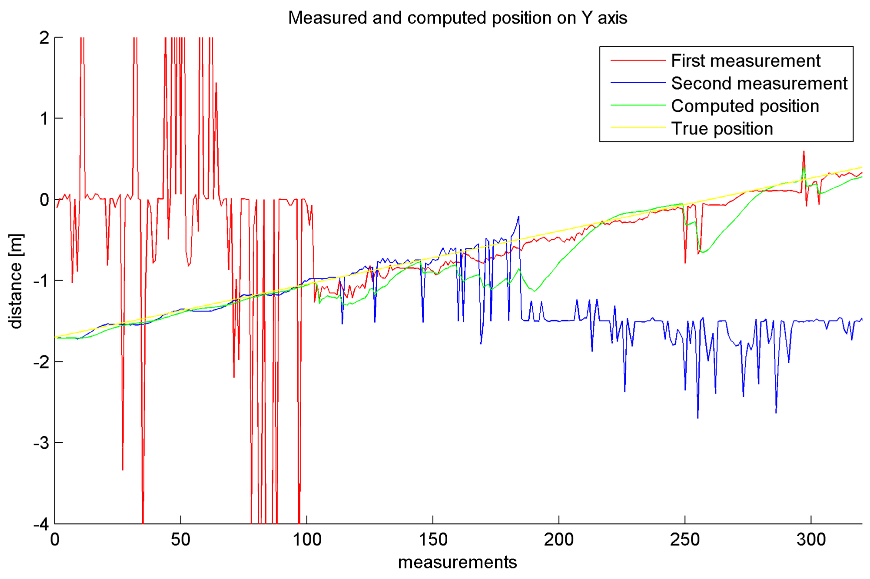


Figure 23: Measured and computed position on Y axis with prior knowledge about the measurement.

We also compared positions estimated from both or only one measurement (see Figures 24 and 25). Figure 25 shows that if one measurement (in this case the second one) is much noisier than another one, the result of data fusion is worse that one of the initial measurements. So sometimes it is better to use only the less noisy measurement.

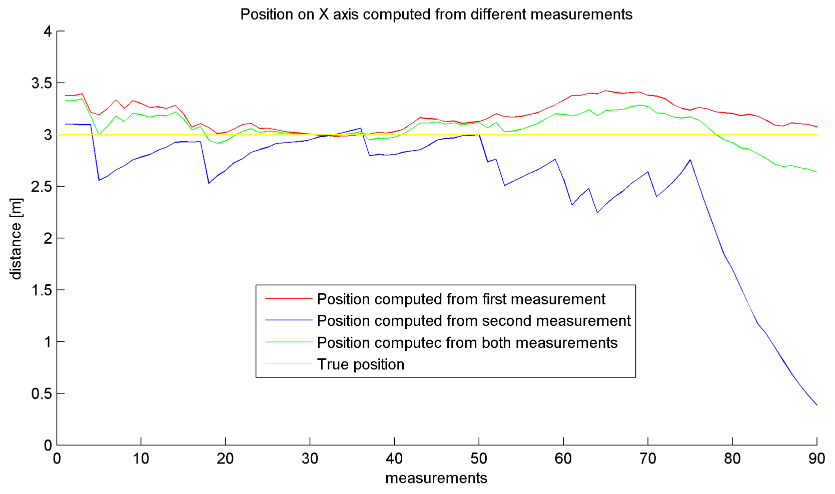


Figure 24: Position on X axis computed from different measurements.

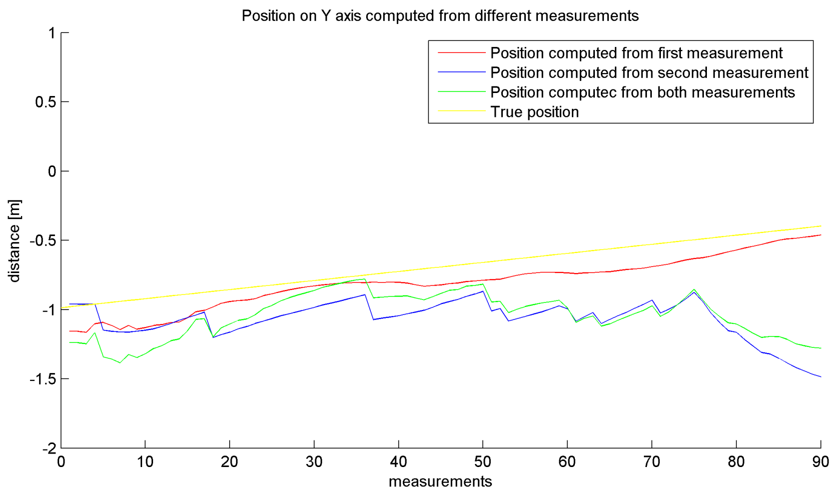


Figure 25: Position on Y axis computed from different measurements.

Testing on real data shows problems related to this vision system. Apparently if the vision system doesn't recognize the position of the pattern it still sends measurements which are obviously incorrect and should be therefore distinguished and removed from measurements.

6.3.3 Circle detector

In this part we tested our algorithm using experimentally measured data obtained from the vision system using the circle detector (see Figure 26). The main advantage of this detector compared with the colour detector (in addition to better recognition rate), is an ability to recognise if there is a detected image pattern in the picture. This feature solves problems related to the vision system based on the colour detector so we are able to decide which measurements are correctly detected and separate incorrect ones from the data fusion.

In the first experiment two camera modules were used to track image pattern moving up and down along the Z-axis. For the experiment arrangement, see Figure 27, where positions of the both camera modules, a measured tracked object positions and resulting computed positions are shown.

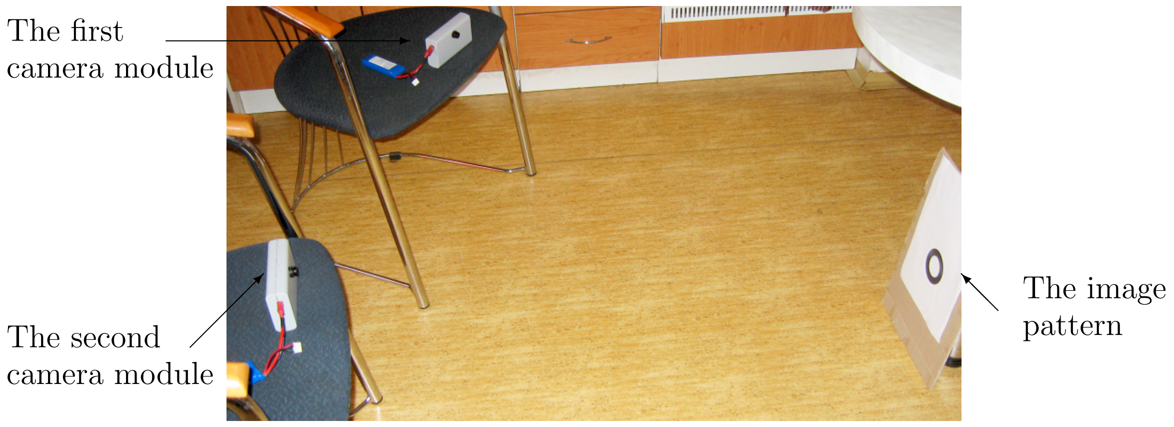


Figure 26: Camera modules tracking the image pattern during the experiment.

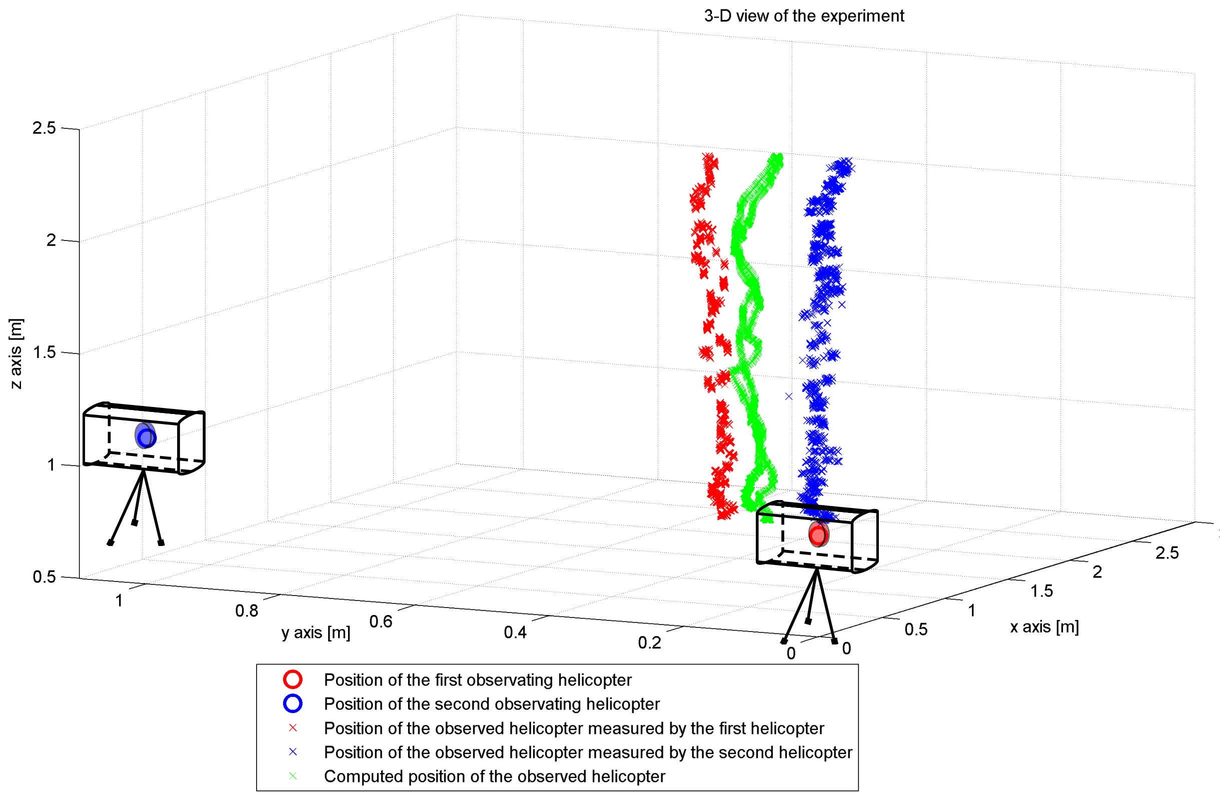


Figure 27: Arrangement of the experiment.

The 3-D figure of the experiment arrangement clearly shows that the tracked object is recognized by both camera modules moreover with the fairly good precision. Although, it is also obvious that the recognised position is subject of constant error so each camera module determined the position of image pattern on a slightly different place. This constant error is probably caused by uncertainty in the position of the camera module.

Next Figure 28 shows Z-axis of the tracked object during the experiment. In the figure the constant position error can be also seen. This example shows the advantage of the data fusion which is able to reduce these constant errors.

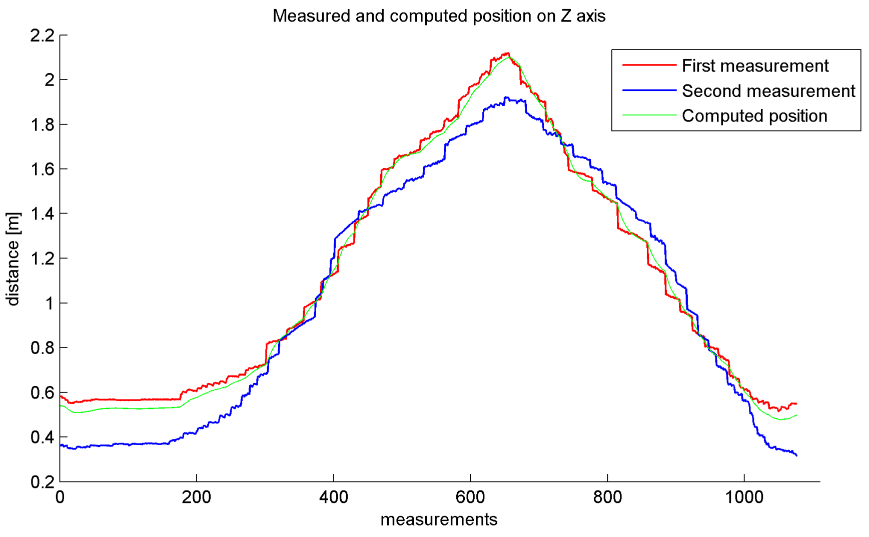


Figure 28: Measured and computed position on Z axis.

During the testing of the fusion algorithm on camera modules with the circle detector there appeared a problem with the recognition frame-rate which is varying. This fact complicated the data fusion from more camera modules. Due this reason the fusion algorithm has been slightly modified to overcome this difficulty. The algorithm is now able to process the measurements with variable time of receiving.

The modification is quite simple the algorithm is checking for the measurement in each time step but it is processed only if the new measurement arrives. Single arrived measurement is directly used in the estimator, on the other hand, if more measurements arrive in the same time the parallel fusion is applied.

The results of this modification can be seen in the next experiment.

In this experiment two camera module were used to track the image pattern moving in the circle in front of these modules (see Figure 29). Because of the circle size, the tracked blob got out of the scope from the individual camera module several times during the experiment. This resulted in the situation when both camera modules have been loosing the blob position repeatedly during the experiment.

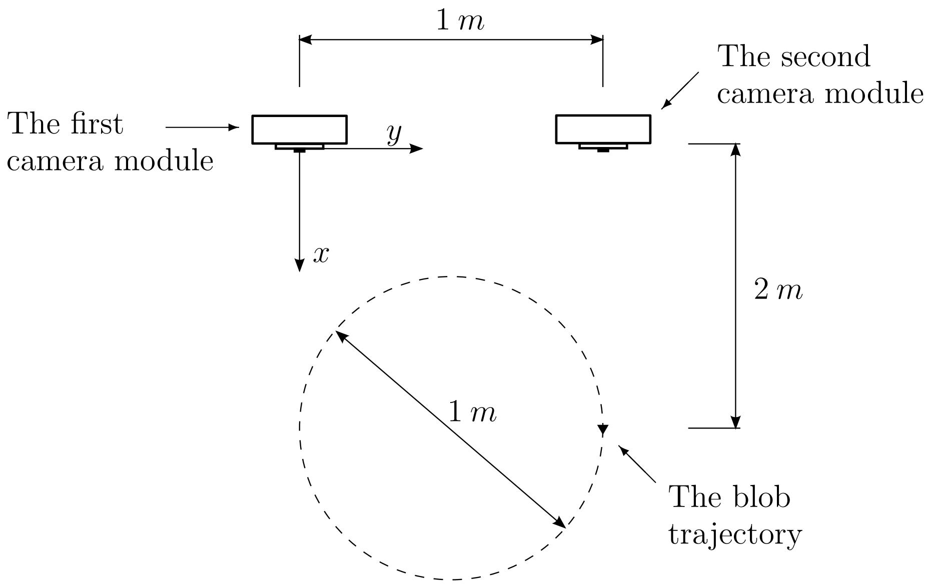


Figure 29: Arrangement of the experiment.

For the experiment see Figures 30 and 31. We can notice a various discontinuities in the measurements made by the camera modules which are caused by limited frame-rate in the camera modules. Another important knowledge about the figures are constant lines in the camera modules measurements. These lines indicates that the current camera module lost the image pattern out of sight. As we can see from the figures, although the discontinuity of measurements and frequent losses of the tracked image pattern, the fusion algorithm is able to estimate successfully the approximate position of tracked blob using the information from both complementary measurements.

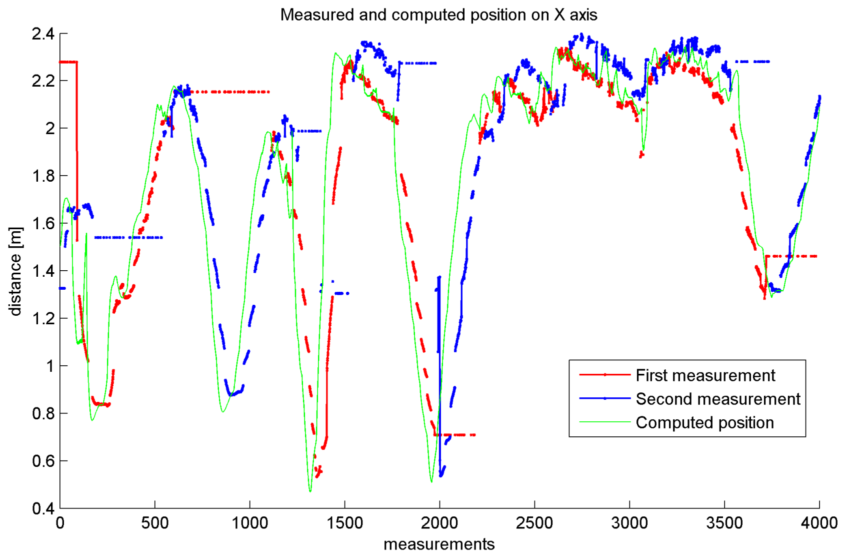


Figure 30: Measured and computed position on X axis.

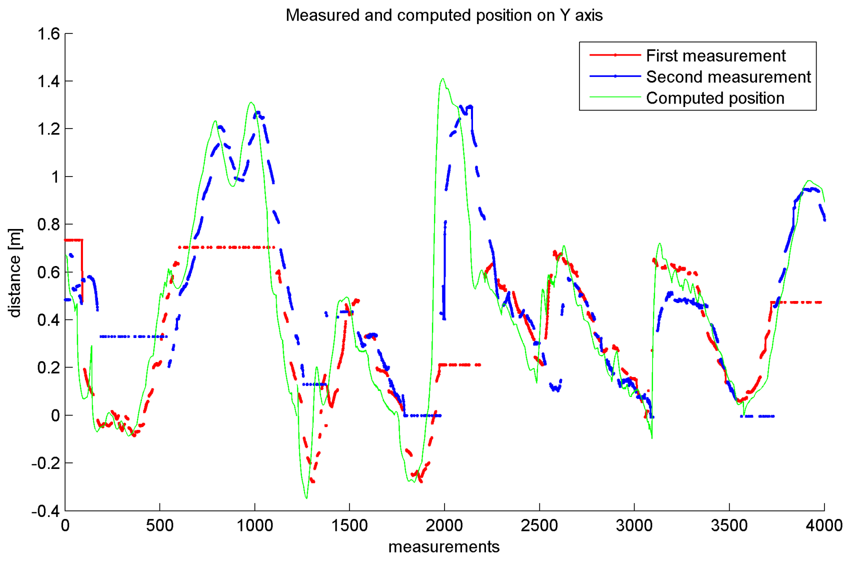


Figure 31: Measured and computed position on Y axis.

The next experiment was similar to the previous one. Two camera modules tracked the image pattern moving in the shape of the parallelogram. See the top view on the measured experiment in the Figure 32.

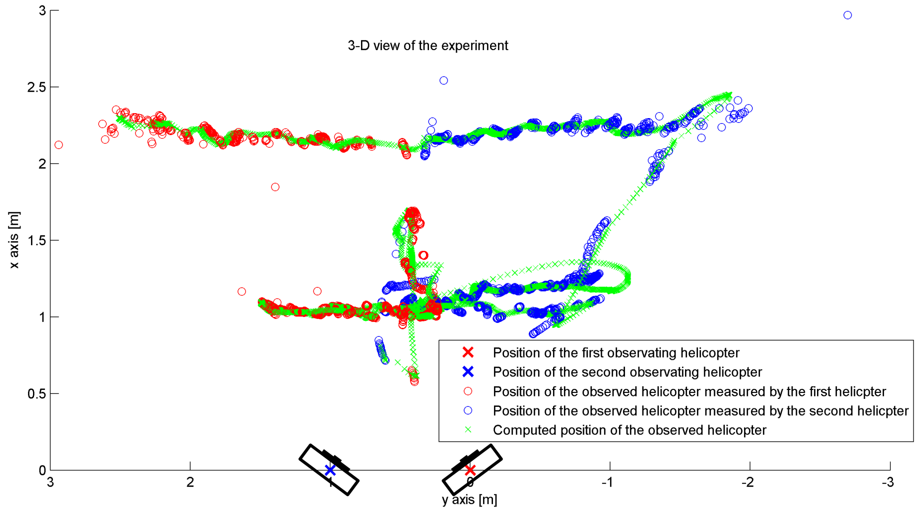


Figure 32: Arrangement of the experiment.

Once again we see two camera modules this time placed at 45 degree angle. In fact each camera module faces with different direction and covers different part of the surrounding area. The modules share just a very small overlapping part of camera workspace.

Still, as seen in Figure 33 and 34, the algorithm is able to estimate the resulting shape of trajectory.

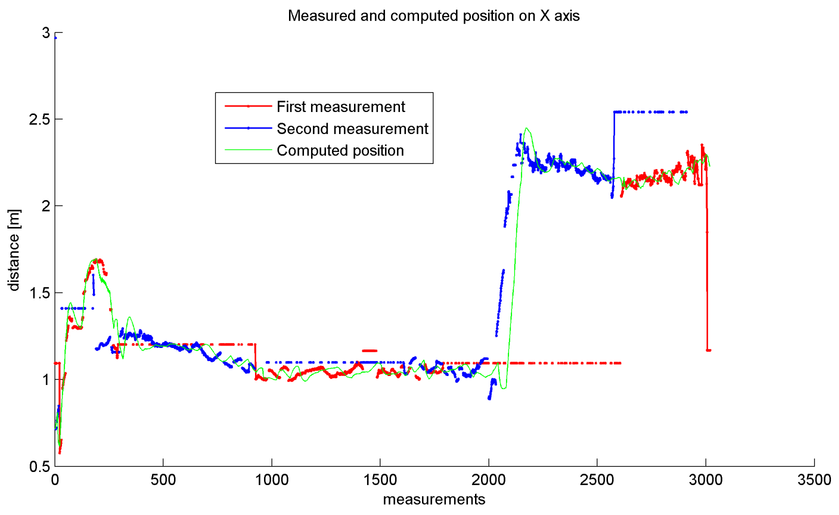


Figure 33: Measured and computed position on X axis.

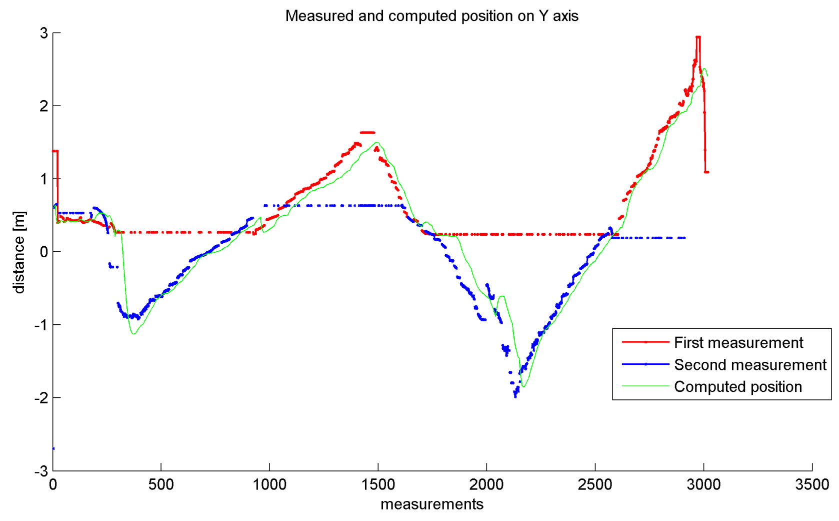


Figure 34: Measured and computed position on Y axis.

So far the experiments only with two camera modules have been shown. Due to the lack of the camera modules the range of the possible experiments have been limited. In fact, only a two camera modules were available for the experimenting. To overcome this difficulty and test the fusion algorithm with a larger amount of the camera modules, different approach have been used.

Instead of a several camera modules simultaneously tracking the image pattern, only one module have been used. The whole experiment (i.e. movement of the image pattern) have been repeated several times, with the camera module placed in the different locations. Evaluating this measured data the possibility of fusion from the several camera modules have been tested.

This last experiment shows the fusion from a five camera modules. For the arrangement of the experiment and the 3-D view of the measured and computed position estimates, see Figure 35.

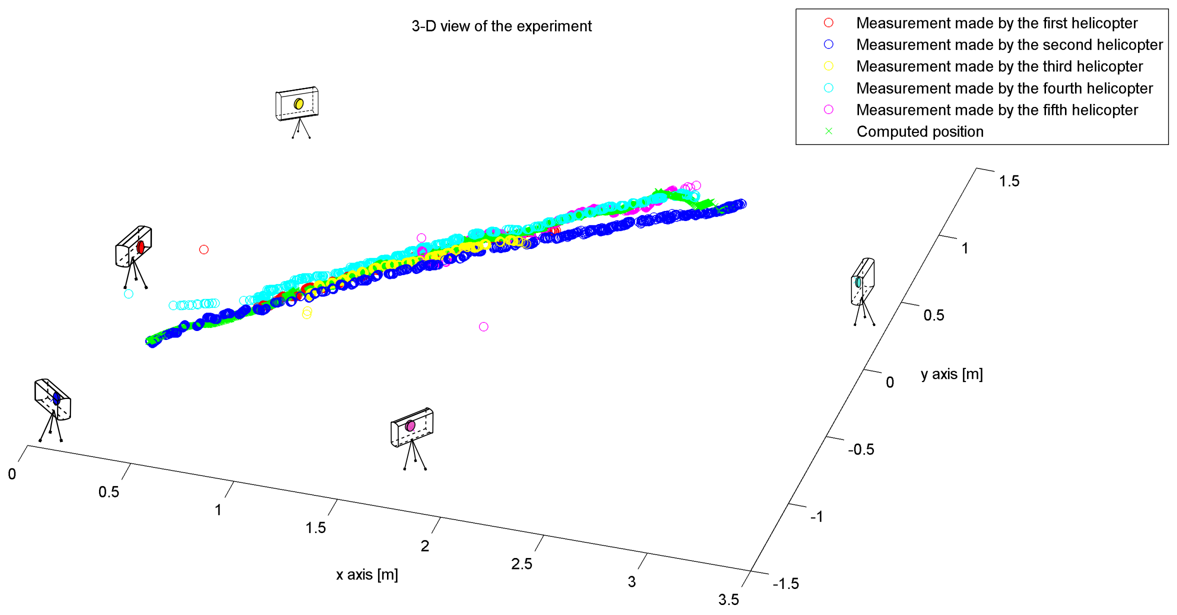


Figure 35: 3-D view of the experiment.

A five camera modules placed in the various locations tracked the image pattern moving in a straight line across the experiment workspace. As well as in the previous experiments with the camera module, the problem with varying camera recognition frame-rate occurred. This caused a discontinuities in the camera measurement.

Also due to the experiment arrangement each camera module is able to track the image pattern only in a limited part of its motion. This missing measurements can be seen as a straight (horizontal) line in the Figures 36 and 37. The algorithm estimated the resulting position for the entire range of the image pattern motion. The inaccuracy of the estimated position is caused probably by the permanent error in the camera module position. Other problems are the varying frame-rate and the missing precise synchronisation of the measurements. This causes the lack of the measurements for the fusion (i.e. there is often only a single measurement in a one time-step), so only a state of the internal estimator is updated, but no data fusion (which is able to reduce the permanent errors in the measurement) is applied.

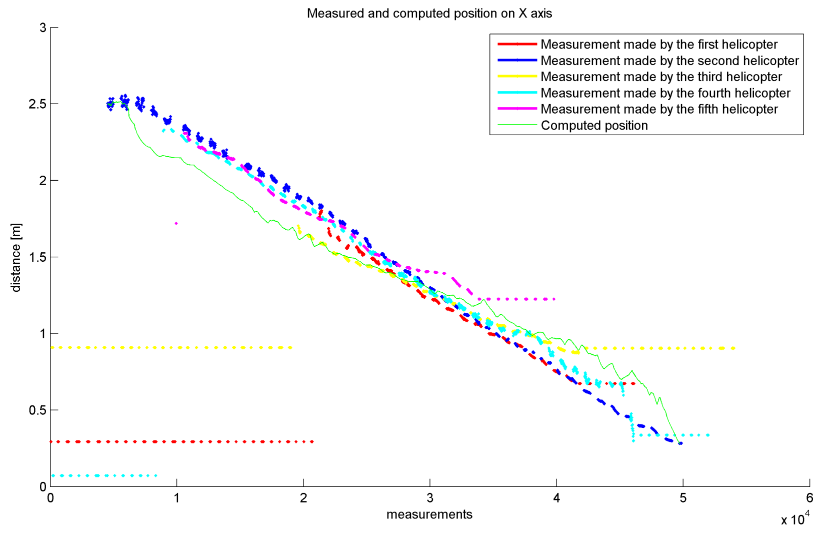


Figure 36: Measured and computed position on X axis.

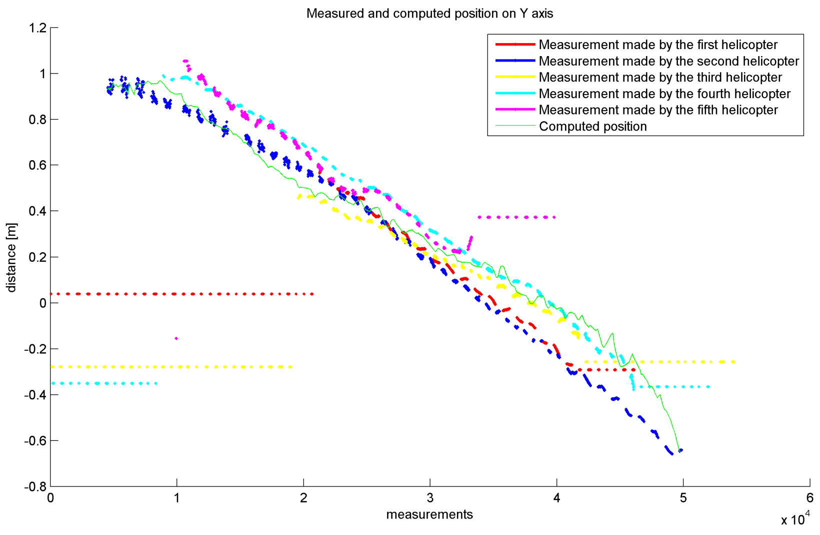


Figure 37: Measured and computed position on Y axis.

7 Conclusion

In this thesis we discussed the problem of the cooperative localization and Kalman filter algorithm and its usage for the estimation and data fusion. We also described the hardware on which our results were evaluated.

Work in this thesis is a part of the COLOS project aimed at developing a complex control system for swarm of unmanned aerial vehicles. For purposes of this project two algorithms were developed and implemented.

At first we developed the algorithm for cooperative localization in swarm of unmanned aerial vehicles which uses the data fusion of location information provided by the vision system implemented in the camera modules. The algorithm was tested and verified on the generated datasets and also evaluated in the actual experiments using the vision system implemented in the camera modules. For these experiments the server-client architecture of the swarm was used. The camera modules represented servers which broadcast information about the position of tracked objects and a user computer acted as the client receiving the position information and processing the data fusion algorithm.

As you can see in Section 6.3 the algorithm proved to be able to reduce uncertainty in the tracked object position by fusing more position estimates. Another benefit of the algorithm is the ability to deal with vision system outages (mostly caused by moving the object out of the camera workspace) in an individual camera module and sufficiently estimate the entire patch of the tracked object motion.

However, the evaluation of the algorithm on real experiments was problematic due to the lack of the camera modules. By that time only two of them were available. This limited the range of possible experiments. Still by using only one camera module to repeatedly capture a scene with the experiment from different positions and angles, we were able to verify the algorithm on the fusion of several position estimates provided by five camera measurements.

We also developed a probability model of vision system in the camera module which can predict the probability of recognition according to the position of the tracked object in the camera workspace.

The results of this work can be subsequently used in the COLOS project. For the full implementation of the fusion algorithm in the camera modules would be desirable to rewrite the tracker server in the camera module and implement fully distributed communication among the swarm members.

References

[1] C. Adams. Is "dead reckoning" short for "deduced reckoning"? `http://www.straightdope.com/columns/read/2053/is-dead-reckoning-short-for-deduced-reckoning/`, November 2002.

[2] D.P. Anderson. Imu odometry. `http://www.geology.smu.edu/~dpa-www/robo/Encoder/imu_odo/`, October 2006.

[3] J. Borenstein, H.R. Everett, and L. Feng. *Where Am I? Systems and Methods for Mobile Robot Positioning.* 1996.

[4] J.Y. Bouguet. Camera calibration toolbox for matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/`, May 2012.

[5] N. Bowditch. *The American practical navigator : an epitome of navigation.* The Agency For sale by the Supt. of Docs., U.S. G.P.O, Bethesda, Md. Washington, DC, 2002.

[6] J. Faigl, T. Krajník, J. Chudoba, M. Saska, and L. Přeučil. *Project COLOS - Technical Report 2.1, Camera Module for Onboard Relative Localization in SWARM of robots*, 0.7 edition.

[7] B.P. Gibbs. *Advanced kalman filtering, least-squares and modeling a practical handbook.* Wiley, Hoboken, N.J, 2011.

[8] M. Grewal. *Kalman filtering : theory and practice using MATLAB.* Wiley, Hoboken, N.J, 2008.

[9] Gumstix. Gumstix small open source hardware. `http://www.gumstix.com/`, May 2012.

[10] L. Jetto, S. Longhi, D. Vitali, et al. Localization of a wheeled mobile robot by sensor data fusion based on a fuzzy logic adapted kalman filter. *Control Engineering Practice*, 7:763–771, 1999.

[11] S. Kai-Tai, T. Chi-Yi, and C.H. Cheng-Hsien. Multi-robot cooperative sensing and localization. In *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*, pages 431 –436, sept. 2008.

[12] J. Kim, Y. Kim, and S. Kim. An accurate localization for mobile robot using extended kalman filter and sensor fusion. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 2928–2933. IEEE, 2008.

[13] P. Ko. Small matrix toolbox for c programmers. `http://www.programmersheaven.com/download/15668/download.aspx/`, May 2012.

[14] F. Kong, Y. Chen, J. Xie, G. Zhang, and Z. Zhou. Mobile robot localization based on extended kalman filter. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 9242–9246. IEEE, 2006.

[15] R. Kurazume and S. Hirose. An experimental study of a cooperative positioning system. *Autonomous Robots*, 8(1):43–52, 2000.

[16] P. Maybeck. *Stochastic models, estimation and control.* Academic Press, New York, 1979.

[17] Inertial Navigation. Inertial navigation – forty years of evolution. *Evolution*, 13(3):140–149, 1998.

[18] R. Negenborn. Robot localization and kalman filters. Master's thesis, Utrecht University, Utrecht, Netherlands, 2003.

[19] I.M. Rekleitis, G. Dudek, and E.E. Milios. Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2690–2695. IEEE, 2002.

[20] S.I. Roumeliotis and G.A. Bekey. Bayesian estimation and kalman filtering: a unified framework for mobile robot localization. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 3, pages 2985 –2992 vol.3, 2000.

[21] S.I. Roumeliotis and G.A. Bekey. Collective localization: A distributed kalman filter approach to localization of groups of mobile robots. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2958–2965. IEEE, 2000.

[22] J.M. Sanchez-Matamoros, J.R.M. Dios, and A. Ollero. Cooperative localization and tracking with a camera-based wsn. In *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, pages 1–6. IEEE, 2009.

[23] A. Singhal. Issues in autonomous mobile robot navigation. Technical report, 1997.

[24] T. Soler and M. Chin. On transformation of covariance matrices between local cartesian coordinate systems and commutative diagrams. Technical report, National Ocean Service, NOAA.

[25] O. Sushkov and W. Uther. Robot localisation using a distributed multi-modal kalman filter. Technical report, and Friends. Technical report, University of New South Wales, 2006.

[26] G. Welch and G. Bishop. An introduction to the kalman filter.

[27] Z. Yifeng. A kalman filter based registration approach for asynchronous sensors in multiple sensor fusion applications. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 2, pages ii – 293–6 vol.2, May 2004.

[28] G. York and D. Pack. Comparative study on time-varying target localization methods using multiple unmanned aerial vehicles: Kalman estimation and triangulation techniques. In *Networking, Sensing and Control, 2005. Proceedings. 2005 IEEE*, pages 305–310. IEEE, 2005.

Appendix

CD Content

In table 1 the names of all root directories on CD are listed

Directory name	**Description**
bp	bachelor thesis in pdf format.
sources	source codes
datasets	measured datasets

Table 1: CD Content