CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR’S THESIS

Vít Cibulka

Cooperative Tracking of a Moving Object by a Formation of Unmanned Helicopters

Department of Cybernetics
Thesis supervisor: Dr. Martin Saska

May 2016
Author statement for undergraduate thesis:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, date.......................... .......................................

signature
BACHELOR PROJECT ASSIGNMENT

Student: Vít Cibulka

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Cooperative Tracking of a Moving Object by a Formation of Unmanned Helicopters

Guidelines:
The aim of the thesis is to adapt the method of cooperative object tracking by formations of Micro Aerial Vehicles (MAVs) [1] for using with platforms of Multi-Robot Systems group at CTU that rely on onboard relative localization [2-3]. Work plan:

- In cooperation with authors of [1], to understand their approach and to re-implement it in C.
- To verify SW components in robotic simulator V-REP and to adapt the method for using with the relative localization approach [2-3].
- To analyze performance of the system with disturbances of sensors and actuators that occur in real deployment.
- To realize a multi-robot experiment with at least two MAVs following the target. The information about relative positions between the MAVs and relative position of the target will be obtained using the onboard system [2-3] or a virtual target will be used (thesis advisor will decide whether the experiment with real or virtual target will be conducted based on availability of the vision system).

Bibliography/Sources:

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

Valid until: the end of the summer semester of academic year 2016/2017

L.S.

prof. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 14, 2016
Acknowledgements

Firstly, I would like to thank my mother for her encouragement and support during my studies. I am also grateful to Dr. Martin Saska who dedicated a lot of time to help with my thesis.
Anotace

Cílem této práce je implementovat algoritmus pro sledování pohybujícího se objektu pomocí formace autonomních bezpilotních helikoptér. Tento algoritmus je v této práci upraven pro použití s reálnými helikoptérami. Algoritmus uvažuje více helikoptér kvůli větší robustnosti vůči nepřesnosti senzorů a vůči vnějším rušivým vlivům. Navržený algoritmus je ověřen numerickými simulacemi a částečně i reálnými experimenty. Při kterých se odhalily nedostatky použitého systému.

Annotation

The goal of this thesis is to implement an algorithm for tracking a moving object by a formation of autonomous unmanned aerial vehicles. This algorithm is adapted for using with real robots. The algorithm considers a formation of robots in order to improve the system’s stability and to deal with noisy detections and external influences. Proposed algorithm is tested in numerical simulations a partially in experiments with real robots.
Chapter 1

Introduction

Unmanned aerial vehicles (UAVs), namely quadrotors are accessible, relatively cheap and small flying robots with great maneuverability. These attributes make them very popular as they have many fields of use, such as rescue missions ([14]), package transport ([17]) and surveillance ([15] and [16]).

This thesis focuses on filming a moving object with a formation of UAVs, which is widely applicable task. Designed, implemented and verified techniques of autonomous cooperative aerial filming may be used for filming athletes during race or 3D reconstruction. This thesis extends the article [1] for using with real UAVs at Multi-Robot Systems group from CTU in Prague ([3]). This system has been designed for basic research and testing of formation flying (see [7], [8], [9], [10], [11]) and bio-inspired UAV swarming ([12] and [13]). While the swarming techniques have been applied in tasks of cooperative surveillance ([15] and [16]), the basic formation flying approaches are being applied in tasks of cooperative RFID localization and the cooperative filming, which is the main objective of this thesis. The goal is to use the UAV system ([4],[3]) for verification of the numerical results from [1] and to discover the limitations of the developed subsystems such as relative localization described in [3], which has not been so far tested in demanding tasks such as localizing multiple moving objects by a flying UAV.
Chapter 2

Problem formulation

The goal of the method described in this thesis is to design an algorithm, which would allow formation of $N$ UAVs to track a given target despite noisy detections, possible malfunctions of some UAVs and losing the target from the fields of view of some UAVs.

The formation is assumed to have a circular shape with the target inside the circle. Graph of the formation is a cycle containing $N$ vertices, each vertex represents one UAV. A neighbour of $i^{th}$ UAV is any UAV whose vertex is adjacent to the vertex of $i^{th}$ UAV. All UAVs can communicate with their neighbours directly via onboard communication system such as Xbee, which was used during experiments.

Each UAV can measure relative positions of other UAVs, the relative localization system used during experiments is described in [5].

The target can be any object or person. The UAV must be capable of locating the target on the camera image with a sufficient framerate (at least 30Hz).

UAVs can communicate only with other UAVs, which means that calculations must be computed onboard. The only exception to this is an observer gathering real-time telemetrics from the UAVs.

The area surrounding the formation is assumed to be without obstacles.
Description of the algorithm

The algorithm described in [1] consists of three main parts: vision-based thrust control, distributed consensus and the intersection rule from [2]. These parts will be described one by one at first, to show their individual contribution to the algorithm.

3.1 Vision-based thrust control

The goal of this particular part is to control the UAV movement using the information from the camera image only. The camera image has a coordinate system, the origin is in the center of the image, as seen in fig. 3.1.

![Coordinate system on the camera image](image)

The control law is designed to keep the target in the origin of the coordinate system. Assuming the person is the target, the UAV can bring the person to the origin by turning or strafing left. The method described in [1] proposes one possible approach to this problem.

3.1.1 Thrust control method description

Thrust gain is obtained as:

\[ m_i(k) = 1 - \exp\left(-\frac{1}{2} \bar{x}_{t,i}(k)^T \Sigma^{-1}_m \bar{x}_{t,i}(k) \right), \]  

(3.1)
where \( m_i(k) \) is the thrust gain for \( i^{th} \) UAV in \( k^{th} \) time step, \( \bar{x}_{t,i}(k) = \begin{bmatrix} x_{t,i}(k) \\ y_{t,i}(k) \end{bmatrix} \) is the target’s position on the camera image of \( i^{th} \) UAV, and \( \Sigma_m \in \mathbb{R}^{2 \times 2} \) is a covariance matrix.

If we consider \( \Sigma_m \) as a diagonal matrix, then the formula in eq. (3.1) can be re-written using

\[
\Sigma_m = \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix}
\]  

such as

\[
m_i(k) = 1 - \exp \left( -\frac{1}{2} \begin{bmatrix} x_{t,i}(k) & y_{t,i}(k) \end{bmatrix} \begin{bmatrix} \frac{1}{d_1} & 0 \\ 0 & \frac{1}{d_2} \end{bmatrix} \begin{bmatrix} x_{t,i}(k) \\ y_{t,i}(k) \end{bmatrix} \right),
\]  

which can be simplified as

\[
m_i(k) = 1 - \exp \left( -\frac{x_{t,i}(k)^2}{2d_1} + \frac{y_{t,i}(k)^2}{2d_2} \right).
\]

\( \Sigma_m \) is used to normalize the coordinates from the camera image, since the horizontal and vertical fields of view are typically not the same. Examples of different \( \Sigma_m \) matrices can be found in fig. 3.2.

\begin{figure}[h]
\centering
\begin{subfigure}{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{a.png}
\caption{\( \Sigma_m = \begin{bmatrix} 50 & 0 \\ 0 & 50 \end{bmatrix} \)}
\end{subfigure}
\begin{subfigure}{0.45\textwidth}
\centering
\includegraphics[width=\textwidth]{b.png}
\caption{\( \Sigma_m = \begin{bmatrix} 20 & 0 \\ 0 & 50 \end{bmatrix} \)}
\end{subfigure}
\caption{Gain function with different parameters. The colors indicate different values of the gain \( m_i(k) \)}
\end{figure}
Thrust gain $m_i(k)$ specifies how far the $i^{th}$ UAV has to move, thrust direction specifies where the UAV has to move. To be able to calculate the thrust direction, the vector $\vec{x}_{t,i}(k) = \begin{bmatrix} x_{t,i}(k) \\ y_{t,i}(k) \end{bmatrix}$ must be transformed from the camera image coordinate system defined in fig. 3.1 to the UAV coordinate system defined in fig. 3.3.

![Coordinate system of $i^{th}$ UAV.](image)

Figure 3.3: Coordinate system of $i^{th}$ UAV.

$$\vec{x}_{t,i}(k) = T \vec{x}_{t,i}(k),$$

(3.5)

where $T$ is a transformation matrix and $\vec{x}_{t,i}(k) = \begin{bmatrix} \tilde{x}_{t,i}(k) \\ \tilde{y}_{t,i}(k) \end{bmatrix}$ is a vector with coordinates transformed into the UAV coordinate system. Now we can calculate the thrust along each axis as

$$a_x(k) = \text{sign}(\tilde{x}_{t,i})m_i(k),$$

(3.6)

$$a_y(k) = \text{sign}(\tilde{y}_{t,i})m_i(k).$$

(3.7)

This restricts the UAV to move only in 8 directions, which can be seen in fig. 3.4. Each arrow corresponds with the thrust. Its length indicates the magnitude from eq. (3.3) and its direction is obtained by eq. (3.6) and eq. (3.7). The thrust gain is applied only if the target distance from the center of the image is increasing. For this a derivative of $a_x$ and $a_y$ is used as
\[ \dot{a}_x(k) = |a_x(k)| - |a_x(k - \tau_a)|. \]  

Combining eq. (3.6) with eq. (3.8) results in

\[ \dot{a}_x(k) = |m_i(k)| - |m_i(k - \tau_a)| \]  

and

\[ \dot{a}_y(k) = |m_i(k)| - |m_i(k - \tau_a)|, \]  

where \( \tau_a \in \{1, 2, 3, \ldots\} \).

Small \( \tau_a \) makes the UAV react faster, but then the UAV is more influenced by noise, whereas large \( \tau_a \) provides more robust solution with slower reaction time.

The velocity needed to center the target in the image is obtained by

\[ \bar{v}_i(k) = \begin{cases} 
\alpha a_{x,i}(k) + v_{x,i}(k - 1), & \text{if } \dot{a}_x(k) > 0 \\
v_{x,i}(k - 1), & \text{otherwise}
\end{cases} \]  

\[ \bar{v}_i(k) = \begin{cases} 
\alpha a_{y,i}(k) + v_{y,i}(k - 1), & \text{if } \dot{a}_y(k) > 0 \\
v_{y,i}(k - 1), & \text{otherwise}
\end{cases} \]  

\[ \vec{v}_i(k) = \begin{bmatrix} 
\bar{v}_{x,i}(k) \\
\bar{v}_{y,i}(k)
\end{bmatrix} \]  
is the output of the thrust control method calculated for time step \( k \) and \( \vec{v}_i(k - 1) = \begin{bmatrix} 
v_{x,i}(k - 1) \\
v_{y,i}(k - 1)
\end{bmatrix} \) is the velocity obtained by eq. (3.13) and eq. (3.14) in the previous time step.

### 3.2 Distributed consensus

The obtained velocity \( \vec{v}_i(k) \) is shared with neighbouring UAVs to achieve the consensus, an agreement on the velocity of the formation. This is important ability of the system to solve a lost sight of the target by a UAV, which is
still able to fly in the formation using the information received from others. Another example could be a UAV perceiving the target incorrectly, perhaps due to seeing false positives or wrongly calibrated camera. Such UAV is still able to maintain its position in the formation using the distributed consensus approach.

The algorithm is based on using the average value of the velocity of the UAV and the velocities received from its neighbours.

\[
v_{x,i}(k) = \frac{1}{\text{card}(D_i)} \sum_{j \in D_i} \bar{v}_{x,j}(k), \quad (3.13)
\]

\[
v_{y,i}(k) = \frac{1}{\text{card}(D_i)} \sum_{j \in D_i} \bar{v}_{y,j}(k), \quad (3.14)
\]

where \(D_i\) represents a set containing \(i^{th}\) UAV and its neighbours. \(\text{card}(D_i)\) is the cardinality of the set, which is a number of elements in \(D_i\). Note that the average is done only with adjacent neighbours (defined in Problem formulation), not with the entire formation, which makes the formation flexible.
3.3 Intersection rule

Intersection rule is a simple method used for maintaining the formation. It is presumed that each UAV knows the relative position of its neighbours.

Let us assume a formation arrangement from fig. 3.5 and an ideal distance between two UAVs denoted as $d_{\text{init}}$, which is shown as a circle around each UAV. The goal of this method is to have each UAV in the intersection of the two neighbouring circles and so to maintain a constant distance $d_{\text{init}}$ from each other. For example, UAV E keeps the distance $d_{\text{init}}$ from its neighbours A and F. On the other hand, UAV B is outside of the formation. Depending on the constraints described below, UAV B either sets its setpoint to the intersection $P_2$ and returns to the formation, or continues in its flight using the velocities calculated in eq. (3.13) and eq. (3.14). Note that empty intersection is not considered in this thesis. It is described in [2].
The relative position of the setpoint, which can be calculated from eq. (3.13) and eq. (3.14) is obtained as

\[
x^{v}_{s,i}(k) = x_{s,i}(k-1) + T_s \cdot v_{x,i}(k), \tag{3.15}
\]

\[
y^{v}_{s,i}(k) = y_{s,i}(k-1) + T_s \cdot v_{y,i}(k), \tag{3.16}
\]

\[
\vec{x}^{v}_{s,i}(k) = \begin{bmatrix} x^{v}_{s,i}(k) \\ y^{v}_{s,i}(k) \end{bmatrix}, \tag{3.17}
\]

where \(x^{v}_{s,i}(k)\) and \(y^{v}_{s,i}(k)\) are relative coordinates of the \(i^{th}\) UAV setpoint in time step \(k\), and \(T_s\) is the sample time.

Nearest intersection can be defined by

\[
\vec{x}^{b}_{s,i} = \arg \min_{x} \| \vec{P}(k) - \vec{x}^{v}_{s,i}(k) \|, \tag{3.18}
\]

where \(\vec{P}(k)\) is used to represent both intersections,

\[
\vec{P}(k) = P_u(k), u \in \{1, 2\}. \tag{3.19}
\]

The final step is to decide whether to use \(\vec{x}^{v}_{s,i}(k)\) or \(\vec{x}^{b}_{s,i}(k)\) as the setpoint position \(\vec{x}_{s,i}(k)\).

\[
\vec{x}_{s,i}(k) = \begin{cases} 
\vec{x}^{b}_{s,i}(k), & \text{if } \epsilon_{i,j}(k) > \epsilon \text{ or } \epsilon_{i,j}(k) > \epsilon \\
\vec{x}^{v}_{s,i}(k), & \text{otherwise} 
\end{cases} \tag{3.20}
\]

where

\[
\epsilon_{i,j}(k) = \| x^{v}_{s,i}(k) - x^{v}_{s,j}(k) \| - d_{init}. \tag{3.21}
\]

\(\epsilon_{i,j}(k)\) is a difference between the current distance of setpoints and \(d_{init}\).

\(\epsilon\) is a constant value, that determines the maximum relative distance two setpoints can have before being set to the appropriate intersection.

Graphical interpretation of these values can be seen in fig. 3.6
Figure 3.6: Graphical explanation of values used in the intersection rule. The blue circle indicates the ideal distance from UAV A. The red circle indicates the maximum distance from UAV A.
Chapter 4

Improvements and adaptation of the algorithm for using with real UAVs

The method described in [1] was tested only in numerical simulations. Implementation for using the method with the system described in [3] required some changes. All of them will be described in this chapter.

4.1 Thrust Control function

According to eq. (3.11) and eq. (3.12), the velocity changes if \( \dot{a}_x(k) > 0 \) or \( \dot{a}_y(k) > 0 \), which means that the velocity changes if the target moves away from the UAV. This behaviour is desired, but when the UAV approaches the target, making \( \dot{a}_x(k) < 0 \) or \( \dot{a}_y(k) < 0 \), the velocity does not change and the UAV flies over the target and the whole process repeats, which results in oscillations around the target, shown in fig. 4.1. This problem can be solved by using a different regulator, such as PI shown in fig. 4.2. Both experiments were done with UAV flying at altitude 2.4m above ground.

![Figure 4.1: UAV following target moving at 0.5m/s, with \( \alpha = 0.007, d_1 = d_2 = \frac{1}{14} \) and \( \tau_a = 5 \). Yellow line represents the target’s velocity, red line the UAV’s velocity. The plot was made in V-REP simulator.](image-url)
4.2 Relative altitude between the UAV and the target

Both the PI controller and the original controller from eq. (3.1) fail when the UAV and/or the target change their relative altitude. An example can be seen in fig. 4.3. Note that the smaller UAV is the target in this case. It is clearly visible, that although the movement of the smaller UAV is the same in both cases, it will be perceived differently by the bigger UAV because altitudes $h_{1a}$ and $h_{1b}$ are different. In case b), the small UAV is almost out of field of view, whereas in case a) it is not even touching the edge of it. This means that knowing heights of both the filming UAV and the target is crucial for the tracking. The article [1] assumes that the UAV altitude is known and it does not change.

Figure 4.3: Showcase of different perception of the same movement depending on the altitude. $h_2 = h_{2a} = h_{2b}$, $h_{1a} < h_{1b}$. The UAV model is from [18].
The height/altitude of the target or the relative height are not discussed there, although this aspect is very important if using non-ideal real UAVs.

### 4.2.1 Calculating relative distance

In the following example, a UAV with its own coordinate system shown in fig. 4.4 is considered. The orientation of the UAV is defined by 3 angles: Roll, Pitch and Yaw show in fig. 4.5

![Coordinate system](image1)

**Figure 4.4:** Coordinate system

![Roll, Pitch and Yaw angles](image2)

**Figure 4.5:** Roll, Pitch and Yaw angles

The process of calculating the relative distance \( \begin{bmatrix} x_t \\ y_t \end{bmatrix} \) will be shown only for \( y \) axis. The calculation for \( x \) axis is similar.

#### 4.2.1.1 Relative distance of the target in \( y \)-coordinate

Consider the situation in fig. 4.6. Assuming that values \( \alpha, h_2, h_1, q_y \) and \( \beta \) are known, it is possible to calculate the relative distance \( y_t \) of target \( T \) using trigonometric functions.
Figure 4.6: Side-view on a UAV tracking a target T
Note that the influence of the Pitch angle is neglected. Roll angle is denoted as $\alpha$.

$$y_t = k_1 + k_2,$$

where

$$k_2 = h_2 \tan(\alpha).$$

The value of $k_1$ can be calculated using a triangle $G_TMT$ as

$$q_y d_y - k_1 = h_1 \tan(\gamma + \alpha),$$

where $d_y$ is the length of the image plane along $y$ axis and $q_y \in (0; 1)$ is a value obtainable from the image, its meaning is shown in fig. 4.6. For example, if the target was in the center then $q_y = \frac{1}{2}$.

The angle $\gamma$ can be calculated using $q_y$ as

$$\gamma = \arctan \left( \frac{q_y d_y + k_2}{h_2} \right) - \alpha$$

and $k_1$ as

$$k_1 = q_y d_y - h_1 \tan(\gamma + \alpha).$$

The only unknown variable is $d_y$, which can be obtained as

$$\tan(\alpha + \beta) = \frac{d_y + k_2}{h_2},$$

$$d_y = h_2 \tan(\alpha + \beta) - k_2.$$  

Distance $y_t$ is obtained using equations eq. (4.1), eq. (4.2), eq. (4.5), eq. (4.4), and eq. (4.7) as

$$y_t = h_2 \left[ q_y (\tan(\alpha + \beta) - \tan(\alpha)) + \tan(\alpha) \right] - h_1 \left[ q_y (\tan(\alpha + \beta) - \tan(\alpha)) + \tan(\alpha) \right].$$

The equation can be further simplified:

$$y_t = (h_2 - h_1) \left[ q_y \tan(\alpha + \beta) + (1 - q_y) \tan(\alpha) \right].$$

Now it is important to calculate the distance the UAV has to move in order to get the target in the center of the image. This distance will be referred to as $y_s$ and is shown in fig. 4.7 as the target horizontal distance from the center line $p$, which is also shown in fig. 4.7.

$$y_s = y_t - m,$$

where

$$m = (h_2 - h_1) \tan(\alpha + \beta/2).$$

$y_s$ can be used instead of $y_{s,i}^{v}$ from eq. (3.16) as the $y$-coordinate of the UAV setpoint.
4.3 Intersection rule for small formations

The Intersection rule from section 3.3 works well with large formations, but small formations can break while still satisfying the constraints of the rule. Consider the formation of 4 UAVs shown in fig. 4.8. UAV A is neighbouring with B and C, and the Intersection rule eq. (3.20) will be applied to them. fig. 4.9 shows the drawback of the method. UAVs A and D can freely drift in the white area, possibly crashing into each other. This can be solved by changing eq. (3.20) and eq. (3.21) as

\[
\vec{x}_{s,i} = \vec{x}_{v,i} + (\vec{x}_{p,i} - \vec{x}_{v,i}) \cdot u
\]

(4.12)

where

\[
u = \begin{cases} 
\frac{\|\vec{x}_{p,i} - \vec{x}_{v,i}\|}{\epsilon} & \text{if } \|\vec{x}_{p,i} - \vec{x}_{v,i}\| < \epsilon \\
1 & \text{otherwise} 
\end{cases}
\]

(4.13)

The constant \(\epsilon\) now defines a circle around the closest intersection. Whenever \(\vec{x}_{v,i}\) is outside the circle, \(u\) is set to 1 and eq. (4.12) sets the setpoint to \(\vec{x}_{p,i}\). If \(\vec{x}_{v,i}\) was inside the circle, the term \((\vec{x}_{p,i} - \vec{x}_{v,i}) \cdot u\) would act like a force that pushes \(\vec{x}_{v,i}\) to the intersection. The greater the distance \(\|\vec{x}_{p,i} - \vec{x}_{v,i}\|\), the greater the force.
This approach solves the problem with free drifting and makes the movement smoother.

Figure 4.8: Formation of 4 UAVs

Figure 4.9: White area indicates where can UAVs A and D be stabilized according to section 3.3 without being pushed to the closest intersection.
Chapter 5

Simulation results

5.1 Comparison of thrust control methods

This section will compare all three thrust control algorithms from section 3.1 (approach from [1]), section 4.1 (PID control) and section 4.2 (Relative localization). From now on, the algorithms will be referred to as A, B and C respectively. The comparison will be performed on a track in fig. 5.1. All simulations in this section were done in V-REP PRO EDU v3.3.0 using a Quadricopter model with a camera.

Figure 5.1: Target trajectory for comparing thrust control methods. The red ball indicates the start. Each square has size of length 2.5m.
5.1.1 Algorithm A

This section presents simulation results of the original approach from [1] using eq. (3.20) for determination of the position of the setpoint.

Used constants: $\alpha = 0.001, \tau_a = 2, d_1 = d_2 = \frac{2}{15}$.

5.1.1.1 Target velocity 0.1m/s

The heatmap in fig. 5.3b shows that the target was not centered. The same is shown in fig. 5.3a. Notice how $x$ and $y$ synchronize with opposite phase after some time. This is due to the 8-directional movement shown in fig. 3.4, the same behaviour can be observed on the heatmap.

It can also be noticed that the error in fig. 5.3a is increasing, which resulted in losing the target ($x$ and $y$ values are set to 50\% upon losing the target).

Graphs in fig. 5.4 show how the setpoint of the UAV oscillated around the target, which caused oscillations in the camera image. Graphs in fig. 5.5 show how much the actual UAV and setpoint coordinates differed from the ideal trajectory. Again, in fig. 5.5b, it can be seen that the difference was increasing, which resulted in losing the target.

This behaviour is caused by the fact that the setpoint does not change its velocity unless the target is getting away from the center of the image as was explained in section 4.1.
5.1. COMPARISON OF THRUST CONTROL METHODS

CHAPTER 5. SIMULATION RESULTS

(a) Difference between the target position on the image and the image center. The values are relative to the image dimensions.

(b) Heatmap showing where the target appeared on the image of the camera. Warmer color means that the target spent more time on that particular area of the image.

Figure 5.3: Data from the camera image.

(a) x position.

(b) y position.

Figure 5.4: Absolute coordinates of the target and the setpoint.

(a) Deviation from the ideal position on x axis.

(b) Deviation from the ideal position on y axis.

Figure 5.5: Deviation from the ideal position for observing the target. The ideal position was calculated using eq. (4.11).
This method lost the target after few seconds in simulations with faster target velocity.

5.1.2 Algorithm B

This section presents simulation results of the PI controller from section 4.1. Used PID constants: $P = 1, I = 0.028, D = 0$.

Note that the simulation is always paused at the start in order to wait for MATLAB to establish connection with V-REP. The controller was functional during the pause and integrated the target position. Because of this, depending on the length of the pause, the setpoint drifted towards the target, causing oscillations at the start of recording.

5.1.2.1 Target velocity 0.1m/s

![Graph](image)

(a) Difference between the target position on the image and the image center. The values are relative to the image dimensions.

![Heatmap](image)

(b) Heatmap showing where the target appeared on the image of the camera. Warmer color means that the target spent more time on that particular area of the image.

**Figure 5.6:** Data from the camera image.

In this case, the results are nearly perfect. The PID controller was able to stabilize the UAV and keep the target in the center of the image. Graphs in fig. 5.7 imply smooth movement of the UAV.
5.1. COMPARISON OF THRUST CONTROL METHODS

CHAPTER 5. SIMULATION RESULTS

Figure 5.7: Absolute coordinates of the target and the setpoint.

Figure 5.8: Deviation from the ideal position for observing the target. The ideal position was calculated using eq. (4.11).
5.1.2.2 Target velocity 0.3m/s

(a) Difference between the target position on the image and the image center. The values are relative to the image dimensions.

(b) Heatmap showing where the target appeared on the image of the camera. Warmer color means that the target spent more time on that particular area of the image.

Figure 5.9: Data from the camera image.

Graph in fig. 5.9a shows large oscillations on the y axis. The oscillations around 10s, 15s and 25s were caused by the change of motion direction of the target.

(a) x position.

(b) y position.

Figure 5.10: Absolute coordinates of the target and the setpoint.

The overall movement of the setpoint was still smooth with minor oscillations during the start. Largest difference of the setpoint and target position can be seen before 15s, where the controller did not react fast enough. Impact of this behaviour is visible in fig. 5.9a as well.

Largest differences between the real and ideal position were again near 10s and 15s with the largest errors on the y axis.
5.1. COMPARISON OF THRUST CONTROL METHODS

CHAPTER 5. SIMULATION RESULTS

Figure 5.11: Deviation from the ideal position for observing the target. The ideal position was calculated using eq. (4.11).

5.1.3 Algorithm C

This section presents simulation results of the method described in section 4.2. This method determines the position of the setpoint using directly the information from the camera image. Smooth movement is ensured by the onboard controller of the UAV (Model Predictive Control method in case of experiments [4] or the controller in V-REP used in the simulations in this section). Note that algorithm C differs from the previous ones in the fact, that it completely relies on the onboard controller and is suited for the multi-robot experiments with platform of CTU in Prague [3]. Previous algorithms were specifically tuned in order to work with the controller provided in V-REP. This controller is however rather simple and does not have any form of saturation of control action. Therefore, the relative distance of the target calculated by algorithm C will be altered as

\[ \vec{x}_{s,new} = 0.5 \vec{x}_{s}, \]  

(5.1)

where \( \vec{x}_{s} \) is a vector containing the setpoint coordinates calculated by the method in section 4.2. \( \vec{x}_{s,new} \) is a new vector of setpoint coordinates, which will be used in the simulations in this section. This approach reduces the control action generated by the V-REP controller, which would otherwise cause oscillations and instability. The results of this algorithm are worse than in section 5.1.2, because the calculated setpoint position had to be altered by eq. (5.1).

5.1.3.1 Target velocity 0.1 m/s

The graphs in fig. 5.13b are not as smooth as in fig. 5.7b, this was probably caused by halving the target’s calculated distance in eq. (5.1).
CHAPTER 5. SIMULATION RESULTS

1. COMPARISON OF THRUST CONTROL METHODS

(a) Difference between the target position on the image and the image center. The values are relative to the image dimensions.

(b) Heatmap showing where the target appeared on the image of the camera. Warmer color means that the target spent more time on that particular area of the image.

Figure 5.12: Data from the camera image.

(a) x position.

(b) y position.

Figure 5.13: Absolute coordinates of the target and the setpoint.

(a) Deviation from the ideal position on x axis.

(b) Deviation from the ideal position on y axis.

Figure 5.14: Deviation from the ideal position for observing the target. The ideal position was calculated using eq. (4.11).
5.1.3.2 Target velocity 0.3m/s

(a) Difference between the target position on the image and the image center. The values are relative to the image dimensions.

(b) Heatmap showing where the target appeared on the image of the camera. Warmer color means that the target spent more time on that particular area of the image.

**Figure 5.15:** Data from the camera image.

The results with higher velocity are worse than in the case of algorithm B. The graph in fig. 5.15a shows that it took approximately 2x more time to recover from the change of direction.

(a) x position.

(b) y position.

**Figure 5.16:** Absolute coordinates of the target and the setpoint.

In fig. 5.16b, there is a visible delay between the target distance and the setpoint. This is caused by the fact that the target calculated distance was halved by eq. (5.1), therefore the setpoint can theoretically never reach the desired position and will always be behind the target.
5.2 Simulations with a formation of 3 UAVs

This section will compare intersection rules from section 3.3 and section 4.3. They will be referred to as D and E, respectively. Both simulations will use the PID thrust control method from section 4.1. All simulations were done in V-REP PRO EDU v3.3.0 using three Quadricopter models. Each Quadricopter has a camera for filming the target and proximity sensors for detecting relative positions of other UAVs.

For purposes of testing the ability of the UAVs to keep the formation in a more realistic way, the calculated setpoint positions were altered by error values $E_a$ and $E_b$. 
5.2. SIMULATIONS WITH A FORMATION OF 3 UAVS

Figure 5.19: Quadricopters with vision sensors (blue) and proximity sensors (purple).

\[ \vec{x}_{s,i,new}(k) = \vec{x}_{s,i}(k) + \left( \frac{1}{\text{card}(D_i)} \sum_{j \in D_i} E_{a,j}(k) \right) + E_{b,i}(k), \]  

(5.2)

where \( \vec{x}_{s,i,new}(k) \) is the new setpoint position altered by errors, this setpoint position will be used in simulations in this section. \( \vec{x}_{s,i}(k) \) is the setpoint position calculated by the PID controller from section 4.1.

\( E_{a,j}(k) \) represents measurement errors of \( j^{th} \) UAV in time step \( k \). This error affects onboard calculations and is shared via distributed consensus from section 3.2. \( D_i \) represents a set containing \( i^{th} \) UAV and its neighbours. \( E_{b,i}(k) \) represents navigation errors. Error values were randomly generated in following intervals:

\[ E_{a,j}(k) \in [-0.1m, 0.1m], \]  

(5.3)

\[ E_{b,i}(k) \in [-0.05m, 0.05m]. \]  

(5.4)

Graphs in fig. 5.20 show that the algorithm E keeps the distances very close to 1.8m unlike the algorithm D. Both were able to keep the relative distances in pre-defined margins. Graphs in fig. 5.21 show the average target distance from the image center. The average was made in each time step from the data from all the cameras that had vision of the target.

It can be seen that recovering from sudden change was longer in case of the algorithm E. The algorithm D has better results because individual UAVs are not constrained in movement as long as their relative distances are satisfying the pre-defined constraints from eq. (3.20).

Graphs in fig. 5.22 show how many UAVs were seeing the target during the simulation. In the case (b), the target was repeatedly lost during a short period of time. This was caused by the errors defined in eq. (5.2) which were causing
5.2. SIMULATIONS WITH A FORMATION OF 3 UAVS

Figure 5.20: Relative distances between UAVs.

Figure 5.21: Average distance of the target from the center of the image.

Figure 5.22: Number of UAVs seeing the target during the simulation.
oscillations of the camera when the target was near the edge of the camera image.
Chapter 6
Experimental results

This section will discuss results from real experiments. MikroKopters L4-ME (in fig. 6.1) from the Multi-Robot Systems group were used during the experiments (see [3] and [4]). Each MikroKopter was equipped with a vision-based blob detection system for measuring relative positions of other MikroKopters (see [5] and [6]). Blobs are black circles on a white background. The blob detection algorithm knows the circle’s proportions and is able to estimate its distance based on the image from the camera. Photo of blobs can be seen in fig. 6.2. Each MikroKopter and the target were equipped with a blob.

In order to distinguish among individual blobs, the MikroKopters were set to detect blobs only in predefined non-overlapping areas, where other MikroKopters and the target would be placed. All of this was done while the MikroKopters were on the ground. After positioning everything in the correct location, the MikroKopters took off the ground in order to start the experiment.

\[
\vec{x}_j^u(k) = \begin{cases} 
\vec{x}_u^u(k), & \text{if } |\vec{x}_j^u(k - 1) - \vec{x}_u^u(k)| < D \\
\vec{x}_j^u(k - 1), & \text{otherwise}
\end{cases} \tag{6.1}
\]

where \(\vec{x}_j^u(\cdot)\) is a vector containing coordinates of the \(j^{th}\) MikroKopter (or the
target) and \( \vec{x}_u(\cdot) \) is a vector with coordinates of \( u^{th} \) blob from the newly obtained set \( U \), for \( j \in J \) and \( u \in U \). \( D \) is a pre-defined constant. Its value was 0.8m during the experiments, it was chosen with regards to expected distances between blobs and the capability of the blob detector. The value specifies the maximum expected distance blob could travel between two frames. This approach depends on framerate and precision of the blob detector. Another factor worth considering is the detection distance, because UAV airflow was significantly influencing other UAVs that were too close (in indoor environment close to walls the distance was around 4m) as seen in fig. 6.3a.

Measured parameters of the blob detector are in table 6.1. Resolution 640x480 did not allow for larger relative distances between UAVs, causing oscillations during which the blob detector was not able to reliably detect the blobs as seen in fig. 6.3b. Higher resolution of the camera allowed for greater relative distances at the cost of low framerate, which also resulted in losing track of the blobs. Another problem was false-positive detection, meaning the detector detected blobs that did not exist.

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Framerate</th>
<th>Max. distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>640x480</td>
<td>30Hz</td>
<td>3.7m</td>
</tr>
<tr>
<td>1280x720</td>
<td>5-10Hz</td>
<td>7m</td>
</tr>
</tbody>
</table>

Table 6.1: Measured parameters of the blob detector

The conducted experiments with 1, 2, and finally 3 UAVs and tests of subsystems of the entire framework identified limitations of the relative localization system ([5]) used for stabilization of multiple UAVs ([3]) in the particular application of the moving target tracking. Although the system is usable for testing of basic methods of formation flying as shown in [7],[8],[9],[10] and [11], in our case the system needs to be improved. Possible solution would be improv-
(a) Altitude of a UAV. Another UAV was launched around 45s, causing large oscillations.

(b) Number of blobs detected by a UAV. There were only 3 blobs around the UAV, meaning there were up to 3 false-positive detections.

Figure 6.3: Flight data from the experiment

ing the blob detection algorithm to work at higher frequencies with resolution 1280x720. Also if each blob had a unique marker, the UAV would never lose track of which blob belongs to which UAV/target. This would probably made the experiment possible even at lower frequencies.

The photos from the experiments in fig. 6.4 show multiple examples of the instability of the UAVs caused by short relative distance. Notice how one UAV is always much higher than the other one.

The photo in fig. 6.5 shows testing of the intersection rule method described in section 3.3 adapted for using with small formations.
CHAPTER 6. EXPERIMENTAL RESULTS

Figure 6.4: Photos showing instability of two relatively close UAVs.

Figure 6.5: Testing the intersection rule method.
Chapter 7

Conclusion

The goal of this thesis was to implement the algorithm described in [1] and to adapt it for using with platforms of Multi-Robot Systems group at CTU ([3]). The algorithm and all its adaptations were implemented in C and tested in the V-REP simulator. The adaptations made to the algorithm are described in chapter 4. Their comparison with the original approach from [1] can be seen in chapter 5.

The hardware experiment verified the intersection rule described in section 3.3 and also verified the ability to track the target with 1 UAV. Experimenting with 2 UAVs revealed problems with mutual influence of the UAVs and some additional problems with the relative localization system ([5]). These problems were identified and motivated improvement of these subsystems to be usable in applications such as moving target tracking.
Bibliography


[18] SimplePlanes QuadCopter. Retrieved from https://jundroo.blob.core.windows.net/simpleplanes/GameData/aircraft/2/REJ8os-SideView.png

Appendix A

CD Content

In table table A.1 are listed names of directories on the CD.

<table>
<thead>
<tr>
<th>Directory name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BP</td>
<td>Bachelor’s thesis in pdf</td>
</tr>
<tr>
<td>BP_source</td>
<td>latex source codes</td>
</tr>
<tr>
<td>vrep_C_client</td>
<td>source codes of all tested algorithms</td>
</tr>
<tr>
<td>vrep_matlab_client</td>
<td>MATLAB scripts for plotting data from V-REP</td>
</tr>
<tr>
<td>vrep_scenes</td>
<td>V-REP scenes</td>
</tr>
</tbody>
</table>

**Table A.1:** Contents of the CD