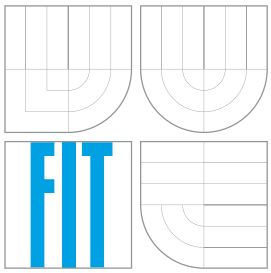
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OPTICKÁ LOKALIZACE VELMI VZDÁLENÝCH CÍLŮ VE VÍCEKAMEROVÉM SYSTÉMU

OPTICAL LOCALIZATION OF VERY DISTANT TARGETS IN MULTICAMERA SYSTEMS

SEMESTRÁLNÍ PROJEKT
TERM PROJECT

AUTOR PRÁCE
AUTHOR

Bc. JAN BEDNAŘÍK

VEDOUCÍ PRÁCE
SUPERVISOR

doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2015

Abstrakt

Výtah (abstrakt) práce v českém jazyce.

Abstract

Výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Klíčová slova v českém jazyce.

Keywords

Klíčová slova v anglickém jazyce.

Citace

Jan Bednařík: Optical Localization of Very Distant Targets in Multicamera Systems, semestrální projekt, Brno, FIT VUT v Brně, 2015

Optical Localization of Very Distant Targets in Multicamera Systems

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. Ing. Adama Herouta, Phd. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Bednařík
January 11, 2016

Contents

Chapter 1

Introduction

- coincidence rangefindrer: https://en.wikipedia.org/wiki/Coincidence_rangefinder
- proc pouzit opticky system 0 hloubkove senzory maji maly dosah a nefunguji ve venkovnim prostredi - Radar (radio detection and ranging) is a device with the active emission of the radio signal. To us it for localization of the object either single radar measuring both the angle and the distance to a target or multiple radars measuring only the distance to a target and triangulating its position can be used [17][2]. +sonar, laser

- vyhody/nevyhody systemu / princip rage finderu vyuzival uz thales https://en.wikipedia.org/wiki/Th

OLS = Optical Localization System - zminit problematiku zastaniceni a rektifikace a odkaz na prislusnou kapitolu

1.1 Background

1.2 Problem formulation

1.3 Related work

The principle of the optical localization of objects is well known for quiet a long time (the first optical range finders emerged in 18th century[15]) and nowadays is still mainly used for military purposes and in robotics. However the professional systems aiming on automatic localization of aerial (as well as terrestrial and underwater) targets mostly rely on active devices, such as radars or sonars and there is very little use of pure passive optical devices (such as RGB cameras, thermal imaging cameras, etc.). The reason might be the complexity of the whole solution or the operation constrained by the weather conditions (see Section 1.1).

On the other hand since the application of multiple view geometry has been one of hot topics in the computer vision during the past decade, many R&D groups (specializing mostly on robotics) attempt to base their solutions on multi camera optical localization.

One of a widely used approaches is to set up so called *intelligent space*[9], the bounded area under surveillance of multiple cameras reporting to the the central system. In [11] multiple terrestrial robots are detected, tracked and localized by the multi camera system. For all cameras the intrinsics and extrinsics are known beforehand and do not change over time. A similar approach is used also in [13] but this system uses the robot's odometers as well in order to improve the resulting position estimated by the optical system. The system presented in [10] then relies on four ordinary RGB cameras with known parameters

(intrisics, extrinsics) to estimate the position of the easy to detect and static object using the Perpendicular Foot Method algorithm.

Even though all of the aforementioned approaches utilize the ordinary RGB cameras for detection and tracking and in general estimate the location of the object using the triangulation algorithm, they all assume the object moves strictly within the specified bounded region and they rely on fixed positions of the cameras. Thus they do not have to deal with the imprecisions arose from the uncertainty of the current camera pose estimation and with the problems of transferring the target.

As for the detection and tracking of the distant aerial targets, one of the most recent approaches was presented by Rozantsev et al.[16]. The algorithm combine both the appearance and motion clue in order to distinguish the object from the complex background.

1.4 Thesis organization

- napad: rozdelit uvod do pokapitol - background - related work - problem formulation - system overview - thesis organization

- Slouží k zasazení řešené problematiky do širšího kontextu - o optickych dalkomerech - vyhody systemu (pasivni radar) oproit aktivnim systemum - vyuziti

- v podobě stručného obsahu jednotlivých kapitol definuje strukturu písemné práce.

– dalsi kapitoly –

formulaci cíle práce, charakteristiku současného stavu řešené problematiky a teoretická a odborná východiska řešených problémů.

Chapter 2

System overview

This section presents the main hardware building blocks of the OLS, the camera stations, and describes the basic pipeline which the system must perform in order to localize a given object. From the implementation point of view, the OLS is built on the ROS framework (see Section 9) which present certain conventions, most importantly the orientation of the coordinate frame which will be used throughout the document (see Figure 2.1).

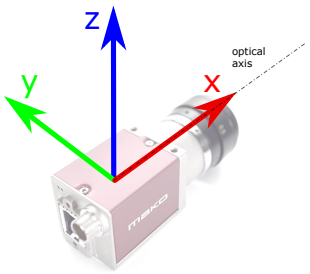


Figure 2.1: A frame orientation convention used throughout the work, adopted from ROS. A right handed frame with X axis aiming forward, Y axis left and Z axis up.

2.1 Camera stations

The main building block of the OLS system is a `camera station` (CS). CS consists of the `camera unit` (CU), which is a collection of hardware necessary for capturing the images, manipulating the position of the camera and estimating the geographical coordinates of the CS (for detailed description see Section 3), and the computation unit processing the data (a PC or other device). In general the OLS is designed to work with arbitrary number of CSs (of course there must be at least two units) and there are two types of CSs:

overview station Only one of the CSs is selected to become the overview station. Its main objective is to scan the surrounding environment, discover the aerial objects, distribute the information to the tracking stations and prospectively take part in the tracking phase. The station dispose of the camera equipped with the zooming lens allowing for scanning greater distances and the PC serving the purpose of the main entry point for the human operator (see Section 9.2.1).

tracking station All other stations become tracking stations which scan the surrounding environment, detect the aerial objects and perform tracking. All cameras ar equipped with the fixed focal length lenses.

As for the network topology, the OLS is base don a star pattern where each tracking station communicates only with the overview station (see Section 9.2.1). The positional organization of the CSs depends on the number of tracking stations, which should always be placed so that their positions projected to the horizontal plane would form a regular polygon with the overview unit in the middle (see Figure 2.2), however in more complex environments this condition does not have to strictly hold (see Figure 2.3).

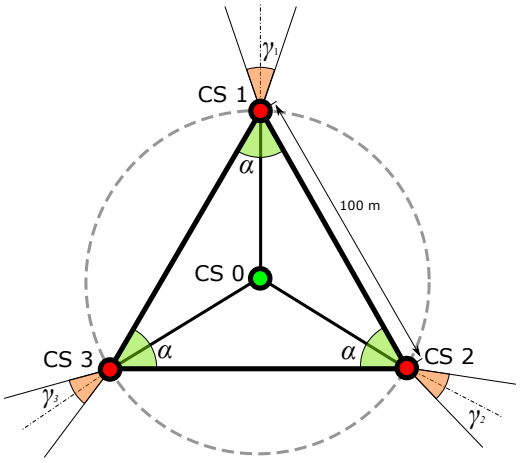


Figure 2.2: The schematic view of the ideal-case organiztaion of all camera units where all three tracking units make up an equilateral triangle with the overviw unit in the middle. The inital orientation of the tracking units is shown as well as their FOVs $\gamma_{1-3} = 38°$ (assuming camera Prosilica GT 1290C and a lense with equivalent 50mm focal length).

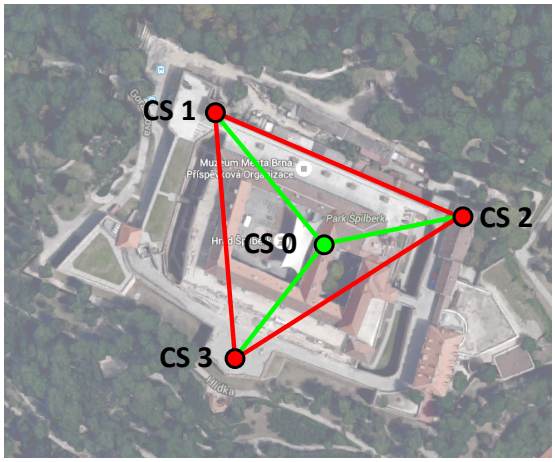


Figure 2.3: An use case scenario showing the organization of the camera units within the system set to protect a real world area (the castle Spilberk in Brno, Czech Republic). As can be seen in the Figure, given the possibilities of the protected area (some CUs mounted on the rooftops) the CUs do not form an equilateral triangle, neither is the overview unit positioned in the middle.

2.2 System pipeline

Before the 3D location of the given target is estimated the system must perform several operations. After powering on the system all camera stations start scanning the surrounding environment. The scanning process is autonomous, however should the operator decide to interfere the overview unit (through the GUI and peripheral devices) is used in order to control the camera.

Once some of the stations detects an aerial object, it saves the visual information distinguishing the object and notifies the overview station. The overview station selects the the best candidate(s) to track the newly discovered object and notifies the chosen unit with the essential data about the object (direction in which it was spotted, visual clues, global id, etc.).

The other unit(s) must first detect the newly spotted object. If it succeeds it initiates the tracking. The overview unit then performs the triangulation and estimates the 3D location of the target.

Chapter 3

Camera unit

A camera unit is a component of a camera station consisting of a hardware necessary to estimate the absolute geographical position of a station, absolute orientation, relative position and orientation with regards to the rest of the stations and a hardware for positioning a camera and capturing an image stream (more in Section 3.1). A 3D location of a camera as well as the direction of the optical axis must be known for each captured frame, thus a model corresponding to the real hardware must be designed (see Section 3.2).

3.1 Devices

A camera unit (see Figure 3.1) consists of a surveying tripod providing a solid base on which a manipulator (P&T unit[1]) together with the LED target for stationing (see Section 5) is mounted as well as all of the sensors (a GPS sensor, an inclinometer, a camera).

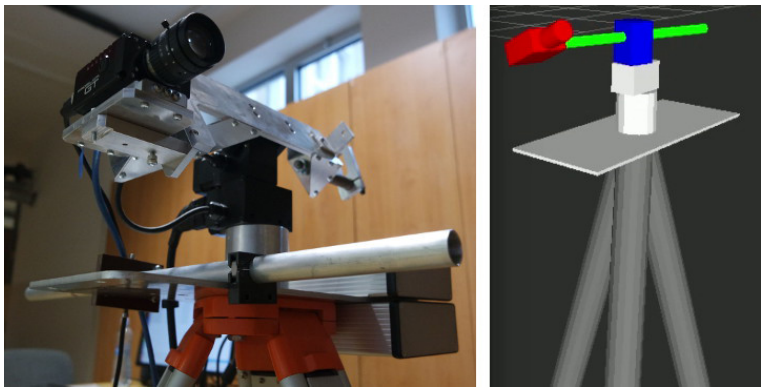


Figure 3.1: A photograph of the upper part of the camera unit consisting of a manipulator Flir PTU-D46-70 with the aluminum mount carrying a camera Prosilica GT 1290C (left) and a corresponding 3D model created for Gazebo simulator (right, see Section 9.3).

manipulator Flir PTU-D46-70 A professional manipulator PTU-D46-70[2] produced by a well established manufacturer Flir is used (see Figure 3.2). As compared to the other professional manipulators this is an entry level device consisting of two stepper motors (pan

[1] From english Pan and Tilt.
[2] A website of product Flir PTU-D46-70: http://www.flir.com/mcs/view/?id=53712

and tilt axes). the device is capable of maximum angular speed of $60°/s$ with the resolution of $0.003°$ while the payload must not exceed 4.08 kg [3]. The operational range is limited to $[-180°, 180°]$ in azimuth and $[-47°, 80°]$ in elevation. The manipulator incorporates no position feedback, the position is inferred from the number of steps and the current resolution, thus it is necessary not to overload the manipulator, otherwise it could loose synchrony and report wrong position.

camera Prosilica GT 1290C A camera Prosilica GT 1290C[3] is an industrial camera manufactured by the company Allied Vision (see Figure 3.2). It is a color camera equipped with CCD sensor (type 1/3") with the resolution of 1280×960 px and support for 33.3 FP and it communicates through gigabit Ethernet [18]. What is important the camera natively supports the PTP protocol for precise time synchronization which is a crucial feature in each application relying on stereo vision as it is capable of time synchronization devices within the range of nanoseconds [7].

Figure 3.2: The photographs of the manipulator Flir PTU D46-70 (left) and the camera Prosilica GT 1290C (right).

3.2 Model

A camera unit model is based on the kinematic chain consisting of six joints and five links corresponding to the distance between the separate part of the tripod and the manipulator (see Figure 3.3). The starting joint `ground` itself is dependent on the reference location (let's call it `world`) which represents the origin of the global coordinate frame. The transformation between the `world` and `ground` reflects the positioning and heading of the given manipulator within the environment (which is estimated during the stationing process, see Section 5).

The kinematic chain is designed as the composition of transformation matrices where a single joint can be located by applying the Euclidean transformation on the position of the joint it is dependent on. For instance the transformation matrix M_{cam} of the joint `camera` can be derived as:

$$M_{cam} = M_{ele} T_{cam} R_{Z_{cam}} R_{X_{cam}} R_{Y_{cam}}, \tag{3.1}$$

where M_{ele} is the transformation matrix of the joint `ele` which the joint `camera` is dependent on.

[3] A website of product Prosilica GT 1290C: https://www.alliedvision.com/en/products/cameras/detail/1290-1.html

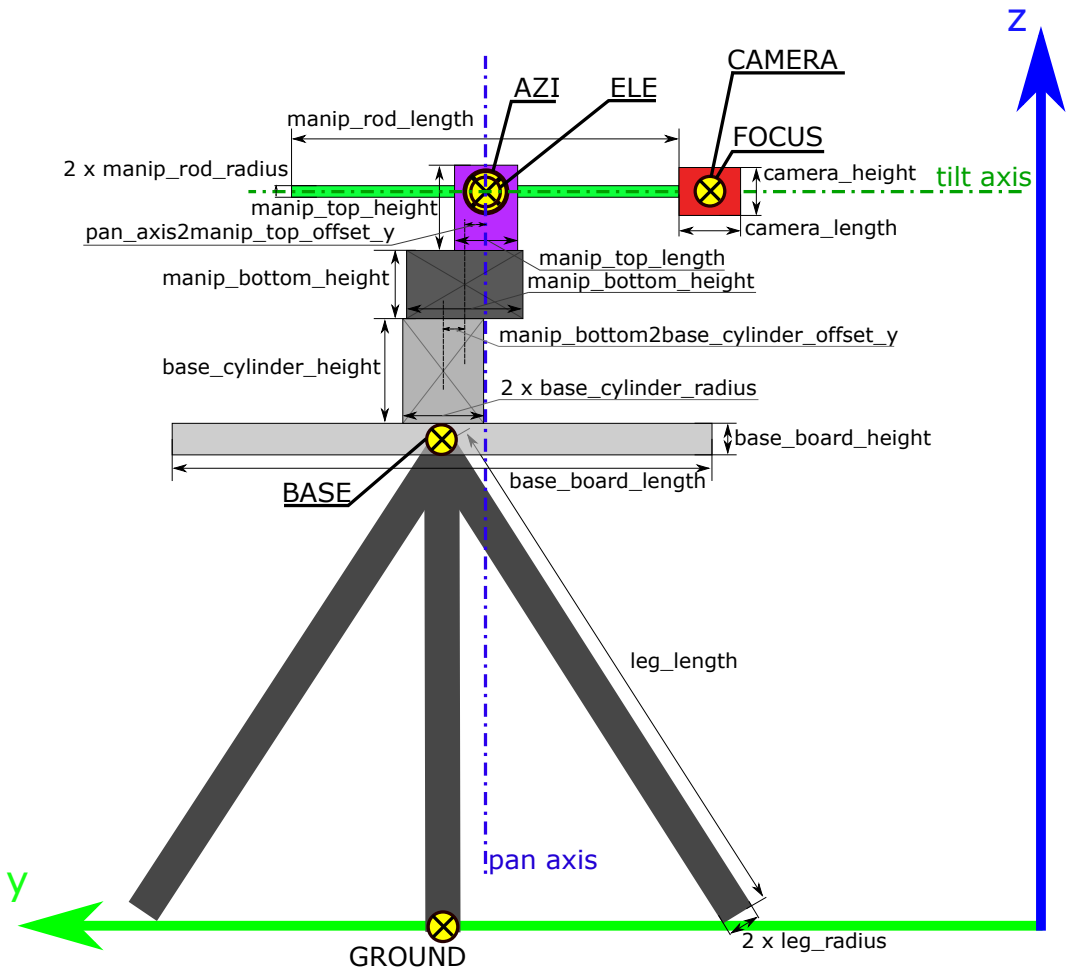


Figure 3.3: A schematic view of a kinematic chain of a camera unit with the joints depicted by the yellow circles with black crosses. The sizes of all components necessary to specify the translation matrices between consecutive joints are shown as well. Note that this is a rear view, i.e. the optical axis of the camera is seen from behind. Thus the joints `camera` and `focus` overlap as they both lie on the optical axis.

Chapter 4

Sensitivity analysis

The precision of the system can be defined in the means of the frame-by-frame Euclidean distance between the estimated location and the real (ground truth) location of the given target. The precision is impacted by multiple independent factors, thus it is essential to perform the sensitivity analysis in order to discover and prospectively alleviate the most prominent contributors of the overall error.

- vysvetleni typu chyb: - system error (systemova chyba) - nesoulad modelu CU s realnou konstrukci - nespravne mereni heading - detekce a tracking ? - uncertainty of the input of the system - GPS mereni - data inklinometru

- vysvetlit, ze se budeme snazit potlacit jen nektere chyby (neresime treba nepresnost mezi modelem a realnou CU z hledsiak translaci mezi klouby)

- rozdeleni na chybu rotace a translace - nepresnost v rotaci je daleko zavaznejsi, nez nepresnost v translaci - priklad a obrazek vlivu nepresnosti o x mrad na lokalizaci cile ve vzdalenosti y m/km

- zdroj chyb: - detekce - tracking - GPS pozice - natoceni vuci severu - rozliseni PTU - model PTU - translace mezi klouby - model PTU - rotace mezi klouby - uchyceni kamery - rotace podle opticke osy - rotace podle osy azimutu - rotace podle osy elevace

Chapter 5

Stationing and rectification

As explained in section 4 the precision of the whole system is dependent on on the uncertainty of the system input as well as on the imprecision of the camera unit construction. The process of stationing aims to alleviate the uncertainty of the system input while the main purpose of the rectification is to reduce the difference between the real camera unit and its model.

5.1 Stationing

Since the stationing is considered to be already working subsystem of the whole project (and thus is not dealt with within the scope of this work) only the main principle will be briefly described. The stationing is composed of two parts: finding the geographical north and finding the relative azimuthal and elevation angles between each pair of camera units.

5.1.1 Geographical north

Though it is common practice to estimate the heading[1] using the magnetometer, this device is unsuitable for this project since the accuracy of the concurrent professional class magnetometers is insufficient (see Section 4). For instance the accuracy of the magnetometers meant for compassing applications produced by Honeywell, the multinational company focusing on aerospace systems, range from hundreds to thousands of milliradians [6].

In order to find the orientation of each camera unit placed in the outdoor environment, distinctive landmarks (created either by human or nature) with known geographical positions are used. For each such a landmark the manipulator is rotated so that the optical axis of the camera would intersect that landmark and both the azimuth and elevation value is registered. Using triangulation the geographical position of the camera unit is derived.

Different possible approach takes advantage of the celestial objects, such as the moon, sun or stars for which the current geographical position is known as well. Nevertheless this approach can only be used between the sunset and the dawn.

5.1.2 Relative azimuth and elevation

To further reduce the impact of the uncertainty of the system input produced by the GPS (see Section 4) and the system error given by the imprecision of the heading estimation (see

[1] Heading is the term used to describe the angle between the torso of the human body and the geographical north [5]

Section 5.1.1) it is convenient to find the relative position of each camera unit with regards to the rest of the camera units.

The information about the geographical position of all camera units as obtained from the GPS sensors is distributed across the whole system. Each pair of camera units then automatically performs the following:

1. Set the azimuth and elevation of the manipulator so that the optical axis of the camera would intersect the expected location of the LED target of the other camera unit.
2. Using the visual clue adjust the azimuth and elevation so that the optical axis of the camera would intersect the center of the LED target of the other unit (see Figure 5.1).
3. Save the current azimuth and elevation values of both camera units and use those values to update the model of the system.

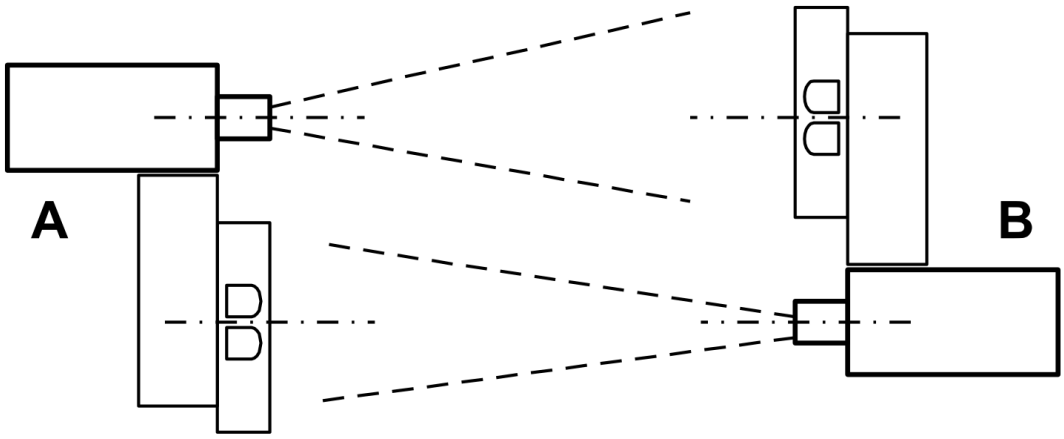


Figure 5.1: The schema of stationing process where two camera units attempt to align the optical axes of their cameras so that they would intersect the LED target of the other unit.

5.1.3 Horizontality

Since the camera unit is expected to be placed in an unknown outdoor terrain, it will never stand on an ideally horizontal surface. Thus it is necessary to either ensure that the unevenness of the surface is compensated by the suitable setting of the camera unit's stand or both the side tilt and front tilt angles of the stand must be estimated and integrated to the model of the given camera unit. For these purposes the inclinometer attached to the base plane of the camera unit (see Section 3) is used.

5.2 Rectification

The process of rectification serves the purpose of reducing the system error caused by the imprecise attachment of the camera to the manipulator. The model of the camera unit assumes that the camera is precisely attached to the manipulator so that the camera image sensor is positioned perpendicular to the azimuthal axis and the rows of the image sensor are parallel to the elevation axis (i.e. the camera is not rotated along the optical axis).

Regarding these requirements the rectification consists of three parts: rotation along the optical axis, rotation along the azimuthal axis, finding the default elevation angle.

5.2.1 Rotation along the optical axis

A custom made metal mount is attached to the bottom side of the camera. The mount is then attached to the manipulator using two opposing round tenons enabling for the rotation of the mount (together with camera) along the axis parallel to the optical axis of the camera (see Figure 5.2).

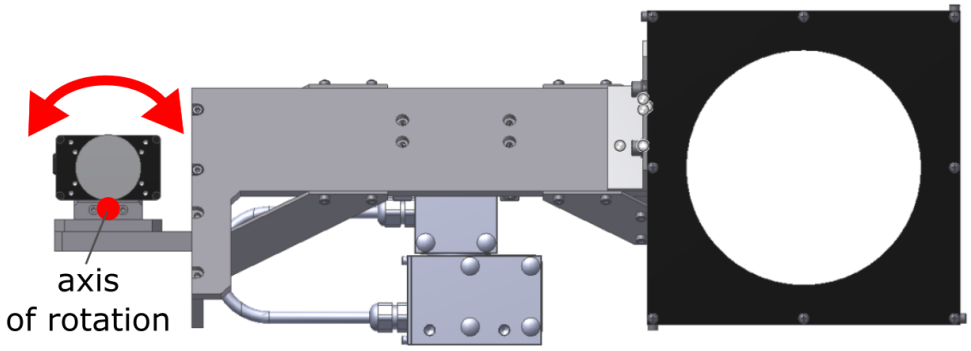


Figure 5.2: Front view of the top part of the camera unit. The red arrow shows the possible rotation of the camera along the axis parallel to the optical axis.

In this part the rectification target with three parallel horizontal black lines is used. Each line has different width so that the operator can select the most suitable one (given the distance of the target, ambient lighting conditions, etc.) As the first step a surveying automatic level is used to rotate the target so that the black lines become horizontal. Then the camera is pointed approximately to the center of the target. The camera image stream is blended with the same stream mirrored across the vertical axis. The operator then manually rotates the camera so that the black lines in this blended image stream appear visually aligned (see Figure 5.3). Once set, the mount with the camera is fixed to the manipulator using two set screws.

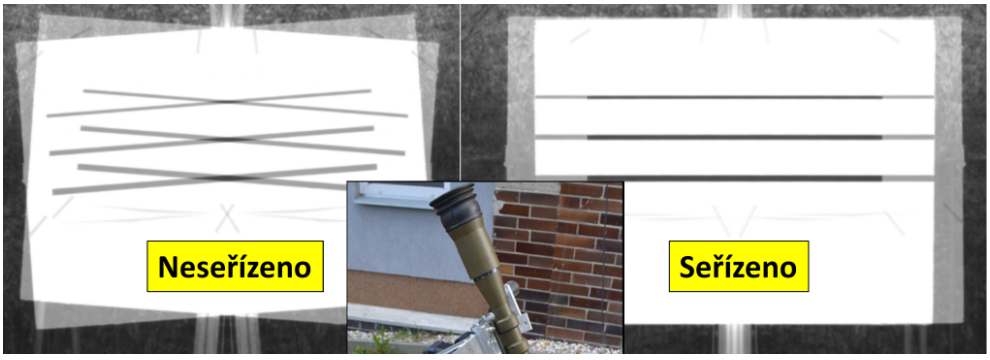


Figure 5.3: A blended image stream from the camera before (left) and after (right) rotating the camera along the optical axis to the correct position.

5.2.2 Rotation along the azimuthal axis

The mount can still rotate along the axis parallel to the azimuthal axis (see Figure 5.4). It is necessary to ensure that the optical axis of the camera is perpendicular to the elevation axis.

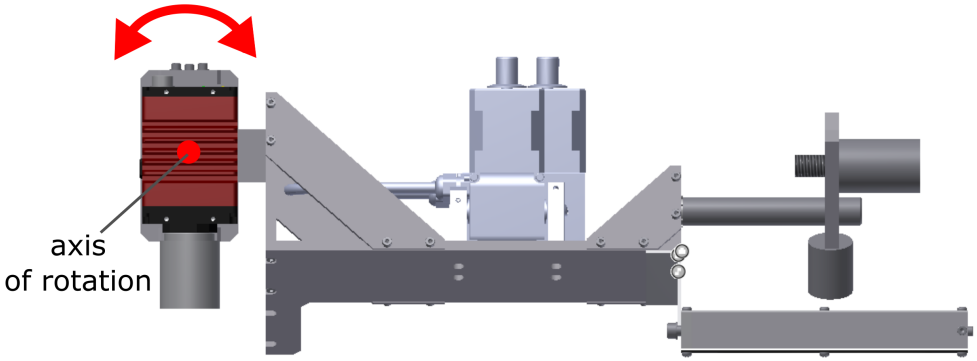


Figure 5.4: Top view of the top part of the camera unit. The red arrow shows the possible rotation of the camera along the axis parallel to the azimuthal axis.

The same target from the first part of the rectification is used, but two black crosses are added to the selected horizontal black line. The distance d_{ao} m between two crosses equals to the distance between the azimuthal and optical axis (which is known from the engineering design, see Figure 5.6).

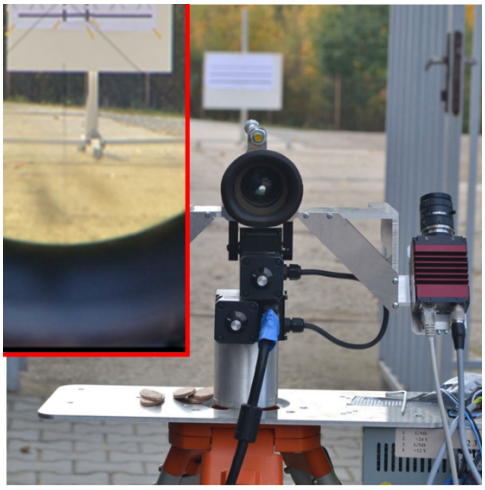


Figure 5.5: A telescope mounted on top of the manipulator. A person looking through a telescope sees a crosshair - a tip of a triangle.

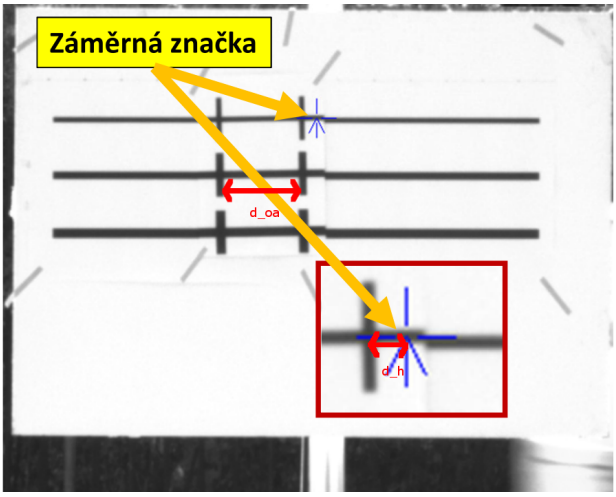


Figure 5.6: Rectification target with the pairs of black crosses. The two crosses in a pair are d_{ao} m appart. A digital corsshair is displayed in order to find the horizontal offset d_h.

A military optical monocular telescope (see Figure 5.5) is mounted on top of the manipulator. The optical axis of the telescope intersects the azimuthal axis, it is perpendicular to it and it intersects the left cross of a given pair on the rectification target. The camera is rotated so that its optical axis (represented by the digital crosshair) intersects the right cross on the target and then is fixed using set screws. As the screws are tightened

the camera is unintentionally rotated a bit again which causes the visual offset between the crosshair and the cross on the target. The offset expressed in pixels is recorded and transformed to the default angle β expressed in milliradians (see Figure) of rotation along Z-axis of the joint `camera` in the camera unit model (see Section 3):

$$\beta = \arccos \frac{focal\_length}{offset},$$
$$offset = pixel\_offset * pixel\_size$$

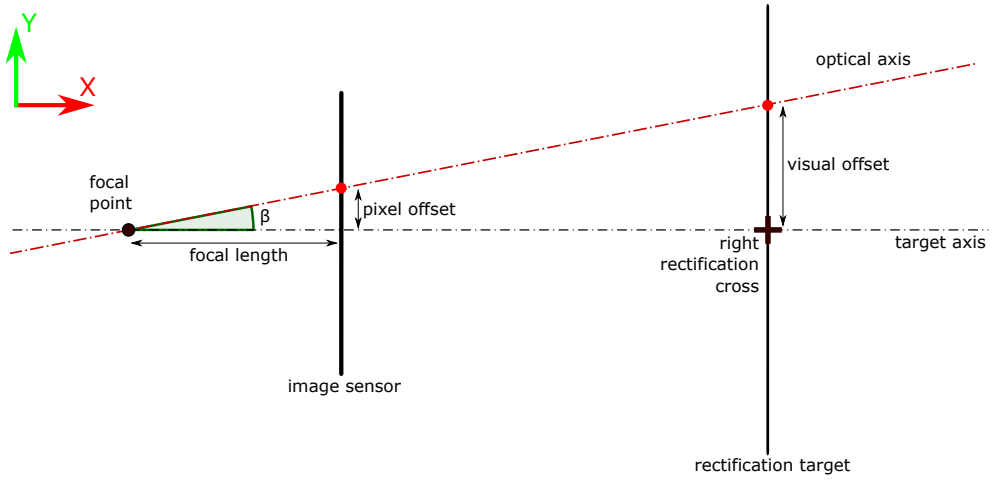


Figure 5.7: The top view schema of a rectification target being projected to the image sensor of the camera. The value of the angle β is one of the output of the rectification process.

5.2.3 Finding the default elevation angle

Given the application of the system the camera is expected to mainly observe the sky. Considering the limited elevation range of the manipulator `Flir PTU D46-70` (see Section 3), the camera must be mounted to the manipulator with the default elevation angle approximately $-60°$. However, after fixing the camera it is necessary to find default elevation angle precisely.

For this purpose a pair of rectification targets, which consist of vertical black and white lines representing the marks of a ruler. The targets are positioned in a row with the distance of a few meters so that the front target would overlap approximately half of the rare target when observed from the camera. Both targets must be rotated so that the lines become horizontal, then the operator manually adjusts the elevation of the manipulator so that the digital crosshair would intersect the same mark on both targets where the two marks form a straight line (see figure 5.8). Once found the current elevation angle is recorded and integrated to the model of the camera unit as an angle of rotation along the Y-axis of the joint `camera` (see Section 3).

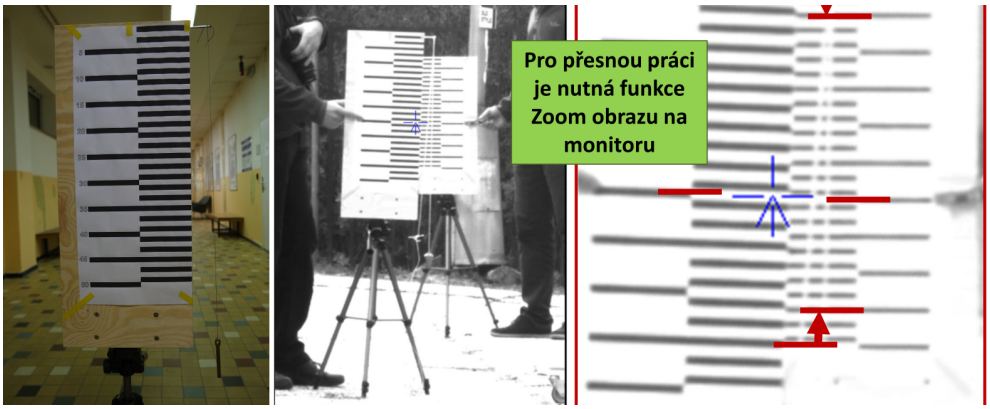


Figure 5.8: Front target of a pair of the rectification targets used to find a default elevation angle (left). A screenshot from the image stream of the camera with the crosshair focused on a row where the marks of the rulers align (right).

Chapter 6

Detection and tracking

6.1 Object model

Choosing a suitable representation of an object of interest is is a first step when designing a tracker. It is a crucial task since the choice of the features determines the type of the final tracker due to the strong relationship between the algorithm and the representation of the object and it can significantly impact the overall performance of the tracker. There are two main representations, the *shape* and the *appearance* [19].

shape representation This category covers the use of points, geometric shapes, silhouette and contour, articulated models and skeletal models. The point representation is not suitable for OLS since the distant aerial objects appear relatively small in the image and might not provide enough distinctive points. The geometric shapes seem suitable as long as the aerial object is far enough so that it could be approximated by a primitive geometric shape. Silhouettes and contours are mostly used for tracking non rigid objects (which is not the case of OLS), neither articulated model is needed to describe an UAV. Considering the small size of the object of interest, a skeletal model would most probably degrade to a single line or a point.

appearance representation The second category covers the use of templates, active appearance models and multiview appearance models. Active appearance models seem promising since they simultaneously model the object shape and appearance, however there is a need for an offline training phase. The templates combine the spatial and appearance information and can scale well to support varying object size (UAV approaching or flying away). A multiview appearance model might perform even better since it is designed to be robust against variation of the appearance of one object due to the change of its orientation.

6.2 Detection

Before a tracking of the object can be initiated the object of interest must first be discovered, which is the main task of object detectors. Even though the OLS allows a human operator to interfere and manually select an object for tracking, the system should be able to perform fully autonomously as well. The approaches to detect an object can be divided into two categories based on the appearance of the object, where the first group covers the cases where the system already posses a strong information about the objects since

they incorporate artificial landmarks, while the second group relies merely on the natural appearance and has only limited or no prior knowledge about the objects [11]. Since the OLS aims on tracking unknown UAVs the first group of approaches is out of question.

A couple of approaches can be distinguished among the detectors using natural appearance [19]:

detection of points This approach goes with the shape representation of the object given by the keypoints. Among others the Harris corner detector or SIFT and SURF descriptors are widely used.

background modeling Under the assumption an observed scene does not change rapidly its appearance can be learned resulting in a background model. The task of the object detector is then to estimate for each subregion or even each pixel whether it belongs to a background or to a foreground (i.e. the object of interest). A widely used approach is to model each pixel as a mixture of Gaussians (in order to support a periodically changing background). Another possible solution is to model intensity variations of separate pixels as the states (including both *background* and *foreground* state) and then with the help of Hidden Markov Models to estimate the state a given pixel is currently in. Since it is possible to enhance these algorithms to allow for the camera motion (resulting in the change of the camera view) they seem suitable for the OLS system.

supervised classification If the (coarse) class of the objects to be tracked is known beforehand and a proper dataset is available, a classifier can be trained to detect the objects. Among others the neural networks or adaptive boosting methods are widely used. Since the OLS aims to localize UAVs, the utilization of a classifier seems like a suitable solution.

6.3 Tracking

The main purpose of the tracker is to iteratively estimating the trajectory of the tracked object from frame to frame and regarding the approaches the trackers can be divided into three main categories [19]:

point tracking The tracker detects the keypoints in each frame and, select those representing the object and computes the transformation from the previous frame.

kernel tracking This approach corresponds to a representation of an object using a template, where kernel is the description of both shape and appearance.

silhouette tracking Silhouettes representing the object can be tracked either by shape matching or contour evolution.

Since the representation by a template (or multiple templates) suits conditions of the OLS the best, an approach Tracking Learning Detection belonging to the *kernel tracking* category was chosen to perform the tracking of the object of interest. TLD is one of the best performing state-of-the-art algorithms developed for tracking general objects and its

capabilities seem promising for the needs of OLS. For detailed description of the algorithm see Section 6.3.1.

6.3.1 Tracking Learning Detection

The Tracking Learning Detection (TLD) [8] is an algorithm designed for performing so called long-term tracking, a robust tracking of an object which might change its appearance, be temporarily occluded by closer objects or temporarily completely disappear from the scene. Since this task cannot be achieved solely by a tracker nor by a detector, the TLD aims to combine the strengths of the detection and tracking algorithms by combining their results. Furthermore the algorithm incorporates the online adaptation subsystem capable of learning the new appearances of the tracked object in the course of the tracking.

A conceptual diagram of the TLD algorithm is shown in Figure 6.1. The `tracking` component tracks the object and for each frame produces the new position. It expects that object does not disappear (occlusion, out of FOV) from the scene and if it does the tracker fails. The `detection` component performs full scanning of the image for each frame. It detects the object and if needed it reinitializes the tracker. The `learning` component is capable of generating new appearances of the tracked object and thus improving the performance of the detector.

The object itself is modeled as a set of patches, each patch being already learned appearance represented by the rectangular bounding box around the object rescaled to a normalized resolution of 15 x 15 pixels. The similarity of the patches is given by the normalized cross-correlation.

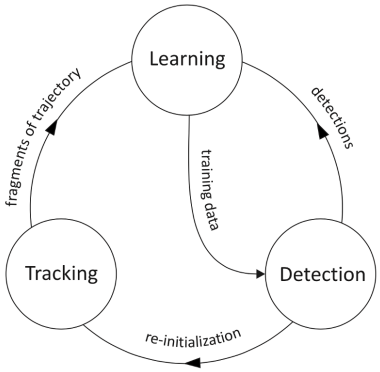


Figure 6.1: A diagram of the main TLD components. Image is adopted from [8].

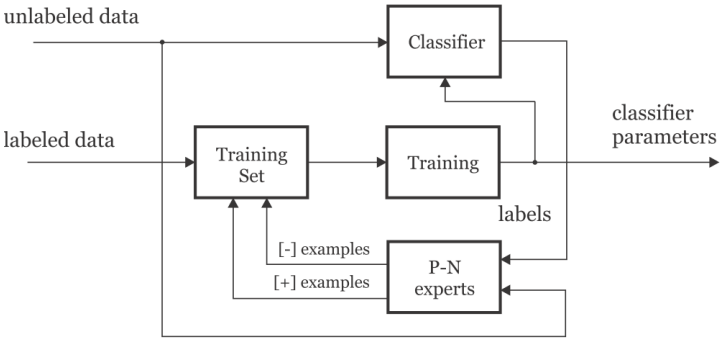


Figure 6.2: A diagram of the PN learning process. Image is adopted from [8].

Detection Detector treats each frame as an independent one and scans a full image. A scanning window is used which is gradually scaled (in order for the detector to achieve scale invariance) and iteratively shifted along a regular grid of candidate positions. Since this task is computationally intensive a cascade classifier is used so that the detector could quickly decide whether a given subregion contains the object. Therefore a cascade classifier is used. In case of TLD it consists of three sequential stages specifically ordered so that earlier the stage is the more subregions it should reduce while being computationally less expensive. Should the sub-region be rejected by any stage, later stages ignore it completely.

Tracking The tracking sub-system is based on thew algorithm called the Median-Flow tracker. A 10×10 grid is used to select the positions within the bounding box representing the object. For each position a given point is tracked between the consecutive frames using pyramidal Lucas-Kanade tracker and eventually the tracker only accepts a 50% of the most reliable displacements to estimate a new position of the target.

Learning Since the classifier used in the detection phase is initially trained using only one positive patch (the initial bounding box selected by a user) it tends to make errors as a video stream progress since the moving object of interest change its appearance due to the transformation caused by its motion. Therefore an online so called `P-N learning` component is incorporated in the system which gradually extends the training sets. Two experts are used, a `P-expert` identifies only false negatives while `N-expert` identifies only false positives. Once a wrongly classified patch is found, the experts extend the training set and the classifier is retrained (see Figure 6.2).

Chapter 7

Cooperation among camera units

- vyber druhe jednotky pro trackovani cile - najezd po opticke ose - predani cile

Chapter 8

Target localization

- triangulace - === LOKALIZACE OBJEKTU VE VICEKAMEROVYCH SYSTEMECH === - === ALGORITMY VHODNE PRO RESENI ULOHY ===

- Cameras are modeled as pinhole cameras. This is a simple model that describes the mathematical relationship between the coordinates of a 3D point in the camera coordinate system and its projection onto the image plane in an ideal camera without lenses through the expressions in

== (obrazek z rvizu (protnuti primek) ==

Given the extrinsic and intrinsic camera parameters, each image point defines a ray in three dimensional spaces, and in the absence of measurement errors, all rays from different cameras intersect in the same object point. But actually the four rays may not intersect in the same point due to the low-quality video cameras and other complicated reasons.

Chapter 9

Implementation

The whole system is built on the robotic framework Robot Operating System (for details see Section 9.1). Since the ROS defines multiple conventions, restrictions and best practices the whole system design including the selection of a programing language, a programing and a communication paradigm, a target platform and a tool for physical simulations is impacted by the possibilities of this framework.

As of writing this text a current state of the implementation mainly builds on the virtual environment provided by the physical simulator Gazebo (see Section 9.3). When confronted with the overview of all subsystems making up the whole system presented in Section 2 so far the following parts are already designed, implemented and/or integrated:

- manual control of the manipulator using peripheral devices
- manual selection of a target and distribution of its appearance to all CUs
- integration and utilization of a OpenTLD tracker
- triangulation of a 3D positions of a target within global frame
- integration of all subsystems

Furthermore a few additional tools were utilized and/or implemented as the necessary building blocks allowing for further development and testing:

- a functional model of a whole system in Gazebo environment
- a standalone application for rectification

9.1 Robot Operating System

Despite its name the ROS[1] is not an operating system but rather a collection of open source libraries, tools and conventions which serve the purpose of a middlewear running alongside a real operating system, however it provides the programmer with the hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [12].

Since the original motivation for developing ROS was to support the collaboration among the experts in the field of robotics in the means of a common software platform

[1]The official website of ROS: http://www.ros.org

[14], a huge developer community has formed around ROS which resulted in wide-scale penetration of this framework as well as the support for a huge range of hardware devices.

9.1.1 Application of ROS

The OLS is designed to become a relatively complex system, thus it exhibits non-trivial implementation requirements such as a need to distribute the computation among multiple computers, the real time performance, integration with physical simulator, etc. Since ROS is a mature framework satisfying the most of theses requirements (see Table 9.1), it was chosen as a main implementation platform.

Table 9.1: The table lists the most important requirements of the OLS and describes how the ROS framework addresses them.

OLS requirements	**ROS features**
native support for hardware such as Prosilica camera, manipulator Flir PTU D46-70, joystick, keyboard	nodes implementing image capture from Prosilica cameras, capturing events from keyboards and joysticks
modularity and reusability of source code	each subsystem is represented by a separate process (node), straightforward reusability
distribution of computation among multiple computers	provides abstraction layer for distributing nodes across devices
simple data exchange among subsystems	the publisher subscriber paradigm [12], support for custom message formats
real time performance	C++ implementation
modelling and simulating the robot	custom language `URDF`[2] for robot modeling, integration with Gazebo
specifying a kinematic chain, heavy 3D transformation computation	native support for computing transformation between frames using package `tf`
complex visualization, debugging	a visualization tool `rviz`[3] for visualizing frames, transformations, robot models, image streams etc.
physical simulation	integration with Gazebo

9.1.2 Standard ROS packages

ROS provides a wide range of standard packages for interaction with various hardware devices and performing various computations. The implementation of OLS utilizes following packages:

[3]Modeling language `URDF`: http://wiki.ros.org/urdf
[3]Vsisualization tool `rviz`: http://wiki.ros.org/rviz

avt_vimba_camera [4]
: This package wraps the Vimba GigE SDK provided by Allied Vision Technologies, the manufacturer of the Prosilica series cameras, and allows the programmer to subscribe to the topic `camera\image_raw` and easily access the image stream.

keyboard [5]
: The package process the keyboard events and expose them via `keydown` and `keyup` topics.

joy [6]
: This package process the events from a joystick and/or gamepad and expose them via `joy` topic.

tf [7]
: The package manages the distribution of the states of all joints within all robot models (in case of OLS the kinematic chains representing the camera units) among all nodes as well as it performs the transformation between given frames on demand [4].

9.2 System architecture

Given the modular concept of the ROS it is most convenient to divide the whole system into autonomous subsystems represented by separate ROS packages. From the perspective of OLS this division of computation work among packages should loosely correspond to the hardware units used.

9.2.1 Hardware

Considering the big picture of the system, OLS consists of four camera stations, a hardware devices necessary for network communication and the peripherals for manual control. There are two types of camera stations, a tracking station and an overview station (see Section 2). All stations communicate with each other vie Ethernet switch. The peripheral devices (keyboard, joystick) allowing the operator to manually control the system are connected to the overview station (see Figure 9.1).

A camera station itself consists of a camera unit (see Section 3) and a computer running the OLS software. The camera unit is controlled by the controller `STM32F4007` which communicates with the manipulator `Flir PTU D46-70`, a GPS module and a zoom lense. The camera `Prosilica GT 1290C` is connected directly to a computer running OLS system (for more details see Figure 9.2).

9.2.2 Software

The system is divided into multiple ROS nodes (processes running in the operating system) with the aim to loosely resemble the hardware components. Five namespaces are used, one `master` namespace and four `camera_unit_N` namespaces, where $N \in \{0, 1, 2, 3\}$ identifies a unique camera unit. The overview of the system architecture from the perspective of the ROS namespaces, nodes, messages and services is depicted on Figure 9.3.

[4] Package avt_vimba_camera: http://wiki.ros.org/avt_vimba_camera
[5] Package keyboard: http://wiki.ros.org/keyboard
[6] Package joy: http://wiki.ros.org/joy
[7] Package tf: http://wiki.ros.org/tf

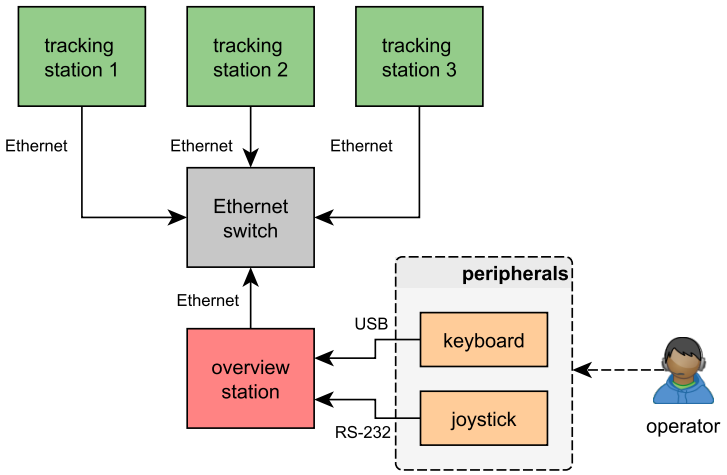


Figure 9.1: The big picture diagram of the main components of the OLS.

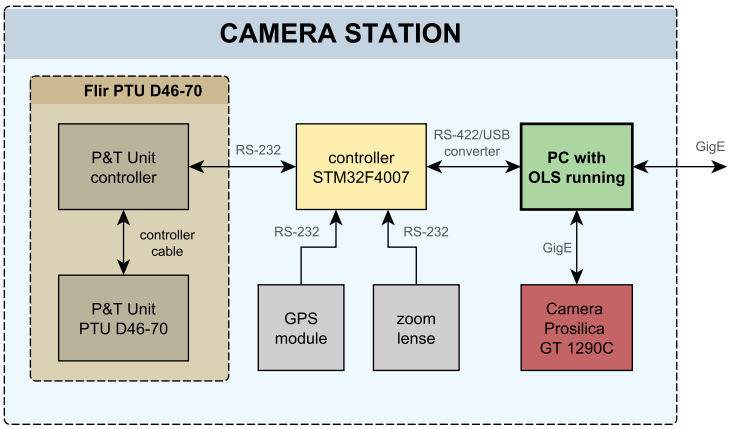


Figure 9.2: The diagram of the hardware components of the camera unit depicting both the hardware devices and the communication standards.

The nodes running within the `master` namespace serve the purpose of the access point for an user (an operator) as well as the main controller of the whole system. The node `controls` keeps the information about the tracked targets and distributes the workload to the separate camera units based on their position and orientation with regards to the newly discovered target. The node `position_estimation` calculates the triangulation and estimates the 3D position of the target. The node `GUI` implements the graphical user interface allowing the operator to control the system and the nodes `joy` and `keyboard` process the events from peripherals.

Each of the namespaces `camera_unit_N` control a separate camera unit. The namespace contains the standard node `avt_vimba_camera` processing the input image stream from Prosilica camera, and implements the node `manipulator` which controls the manipulator and publishes its state and the node `detection_and_tracking` which performs the computationally most expensive tasks of the object detection, learning its appearance and tracking.

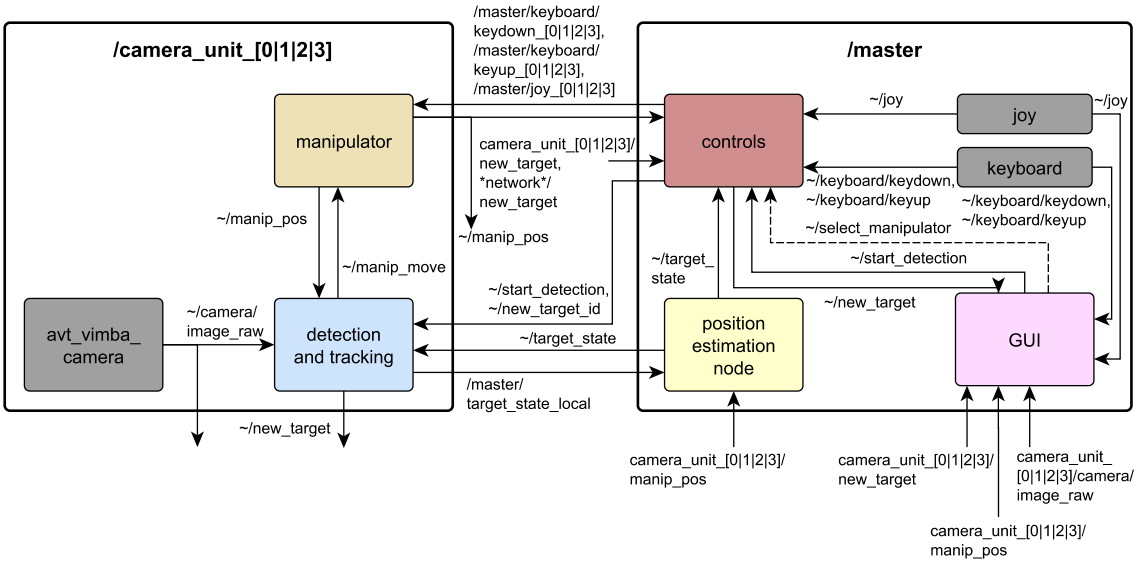


Figure 9.3: The diagram of the software components represented by the ROS nodes. The communication among topics is implemented using ROS topics (normal arrows) and ROS services (dashed arrows).

9.3 Application of Gazebo

Gazebo[8] is a physical simulator developed by the OSRF[9] providing the tools to model and simulate robots in both indoor and outdoor environment. The simulator has been developed since 2002 and today represents a mature system with wide penetration and support, while being distributed as open source and freeware.

Since the Gazebo is distributed also as one of the standard packages of the ROS framework it is straightforward to integrate the simulation environment with the already implemented ROS nodes. There are multiple advantages of using the simulator over a development using a real hardware, the main motivations were as follows:

testing a tracker The Gazebo provides the ROS plugin simulating an RGB camera, which captures the virtual scene and publishes a stream of rendered images. Thus it can be used to test an object tracking algorithm using arbitrarily complex environment and moving objects (see Figure 9.4).

testing a manipulator It is a good practice to include a real hardware in the simulation during the development [1]. Both actuators and sensors would be difficult to simulate properly, moreover a real manipulator is constrained in terms of the operational range (see Section 3), maximum acceleration and speed and communication throughput so it is necessary to thoroughly test its performance. This so called hardware-in-the-loop simulation reveals whether the implementation of motion control is correct and whether the possibilities of the manipulator suffice to track arbitrarily fast (simulated) objects.

[8]The official website of Gazebo: http://gazebosim.org
[9]Open Source Robotic Foundation: http://www.osrfoundation.org/

Figure 9.4: The screenshot of a Gazebo simulation (left) consisting of a virtual environment (the gas station), a flying object (the red sphere) and four manipulators. All four virtual camera streams are displayed real-time using `rviz` tool (right).

testing a triangulation Thanks to the Gazebo it is possible to simulate a flying object with a-priory set trajectory and evaluate the precision of a position estimation algorithm using comparison between the estimated target position and a ground-truth.

9.4 External libraries

Besides the framework ROS a few other publicly available libraries are used within the implementation.

OpenCV Open Source Computer Vision Library[10] is a free open source library providing algorithms for image processing, computer vision and machine learning. Version 2.4.11 is used as it is a component of ROS Indigo.

Eigen This open source C++ template library[11] implements the data structures and methods for fast and convenient solving of linear algebra problems.

OpenTLD The OpenTLD library[12] represents an open source C++ implementation the TLD tracking algorithm (see Section 6).

[10] The official website of OpenCV: http://opencv.org
[11] The official website of Eigen: http://eigen.tuxfamily.org
[12] The official website of OpenTLD: http://www.gnebehay.com/tld

Serial A cross-platform library Serial[13] implemented in C++ provides the API for interfacing with RS-232 serial like ports. It is used to control the manipulator.

[13]The official website of Serial: https://github.com/wjwwood/serial

Chapter 10

Experiments and results

- vlastni datova sada - experimenty na ziskena a porizene datove sade - === EXPERIMENTY NA REALNYCH DATECH ===

Chapter 11

Conclusion

- zhodnocení dosažených výsledků se zvlášť vyznačeným vlastním přínosem studenta - zhodnocení z pohledu dalšího vývoje projektu vzhledem k diplomové práci - moznosti dalsiho vyvoje: modelovani okoli ve 3D nebo vyuziti teto info z mapy, paralelizace narocnych casti na GPU - veci, co se zatim nebraly v uvahu - system je umisten na rozhrani dvou a vice UTM zon

- moznost sdileni info o appearance modelu mezi CUs - vylepseni TLD, protoze prepokladame fixni scenu (meni se pouze natoceni kamery)

Bibliography

[1] M. Bacic. On hardware-in-the-loop simulation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 3194–3198, Dec 2005.

[2] V. Chernyak. About the use of bistatic measurements for higher object localization accuracy in multisite radar systems. In *Radar Conference - Surveillance for a Safer World, 2009. RADAR. International*, pages 1–6, Oct 2009.

[3] Flir. Ptu-d46 miniature computer-controlled pan/tilt unit [online]. http://cvs.flir.com/ptu-d46-datasheet?_ga=1.141782651.854570727.1452364112, 2014.

[4] T. Foote. tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, pages 1–6, April 2013.

[5] Manne Henriksson. Estimation of heading using magnetometer and gps., 2013.

[6] Honeywell. Magnetic sensors product catalog - sensing earth's magnetic field [online]. https://aerospace.honeywell.com/~/media/Images/Plymouth%20Website%20PDFs/Magnetic%20Sensors/Technical%20Articles/Sensors_Product__Catalog.ashx, 2015.

[7] Texas Instruments. An-1728 ieee 1588 precision time protocol time synchronization performance [online]. http://www.ti.com/lit/an/snla098a/snla098a.pdf, 2013.

[8] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, July 2012.

[9] Joo-Ho Lee, Noriaki Ando, T. Yakushi, K. Nakajima, T. Kagoshima, and H. Hashimoto. Adaptive guidance for mobile robots in intelligent infrastructure. In *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, volume 1, pages 90–95 vol.1, 2001.

[10] Wei Liu, Chao Hu, Qing He, and M.Q.-H. Meng. A three-dimensional visual localization system based on four inexpensive video cameras. In *Information and Automation (ICIA), 2010 IEEE International Conference on*, pages 1065–1070, June 2010.

[11] Cristina Losada, Manuel Mazo, Sira Palazuelos, Daniel Pizarro, and Marta Marrón. Multi-camera sensor system for 3d segmentation and localization of multiple mobile robots. *Sensors*, 10(4):3261, 2010.

[12] Jason M. O'Kane. *A Gentle Introduction to ROS*. CreateSpace Independent Publishing Platform, 10 2013.

[13] D. Pizarro, M. Mazo, E. Santiso, M. Marron, and I. Fernandez. Localization and geometric reconstruction of mobile robots using a camera ring. *Instrumentation and Measurement, IEEE Transactions on*, 58(8):2396–2409, Aug 2009.

[14] Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[15] C.T. Reviews. *e-Study Guide for: A Short Course in Photography: Arts, Visual arts*. Cram101, 2013.

[16] Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Flying objects detection from a single moving camera. *CoRR*, abs/1411.7715, 2014.

[17] J.C. Toomay. *Radar Principles for the Non-Specialist*. Springer Netherlands, 2012.

[18] Allied Vision. Prosilica gt 1290 [online]. https://www.alliedvision.com/en/products/cameras/detail/1290-1/action/pdf.html, 2015.

[19] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), December 2006.