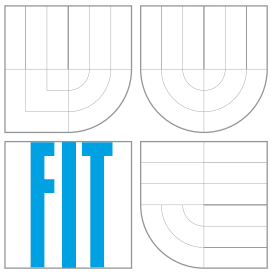
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



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 5, 2016

Contents

Chapter 1

Introduction

- napad: rozdelit uvod do pokapitol (podle https://uu.diva-portal.org/smash/get/diva2:648760/FULLTEX
- background - related work - problem formulation - system overview - thesis organization

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

Related work

- srovnani s ji existujicimi aktivnimi (pasivnimi) radary

Chapter 3

System overview

- koncept systemu (rozlozeni jednotek, pouzity hardware, propojeni, ...) - zmineni zakladniho knceptu behu celeho systemu od detekce cile az po jeho lokalizaci - zminit zakladnich problematiky celeho systemu: zastaniceni, rektifikace, detekce, tracking, vyber jednotek pro sledovani, najezd po opticke ose, predani cile, triangulace, ... - moznost rizeni jednotky pomoci joystiku/klavesnice - zminit problematiku zastaniceni a rektifikace a odkaz na prislusnou kapitolu - zminit pouziti ROS konvence co se tyka souradnych systemu

Chapter 4

Camera unit

- z ceho sestava camera unit (zminit manipulator i kameru) - návrh kinematického řetězce modelu manipulátoru - ukazat 2D diagramy, 3D model z rvizu/gazeba - fotka - schema, kde bude ukazano, co je azimuthal a elevation axis

- PTU Flir - HW/SW limity pohybu - krokove motory - proble proklouznuti kroku
- nejake info o kamere

Chapter 5

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 6

Stationing and rectification

As explained in section 5 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.

6.1 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.

6.1.1 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 5). 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 [3].

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.

6.1.2 Relative azimuth and elevation

To further reduce the impact of the uncertainty of the system input produced by the GPS (see Section 5) 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 [2]

Section 6.1.1) 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 6.1).
3. Save the current azimuth and elevation values of both camera units and use those values to update the model of the system.

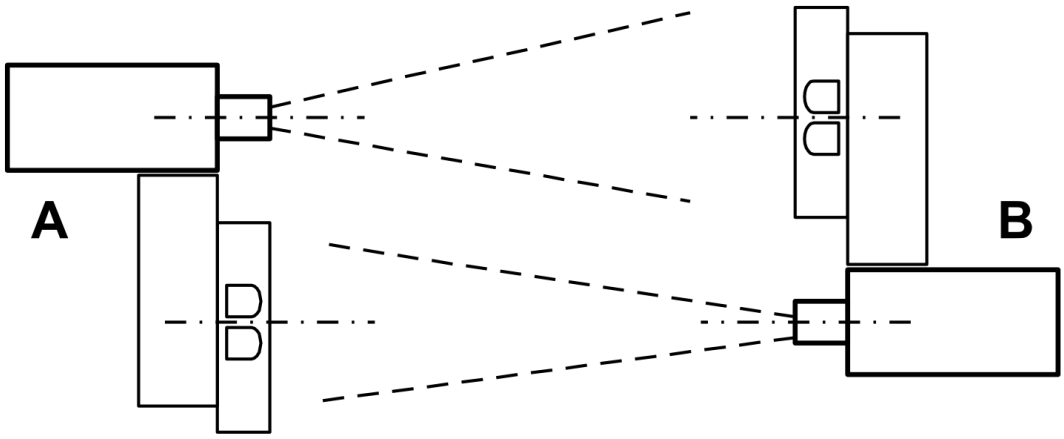


Figure 6.1: The schema of stationing process where two camera units attempt to align the optical axes of their cameras so that they would intersect the LED target of the other unit.

6.1.3 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 4) is used.

6.2 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.

6.2.1 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 6.2).

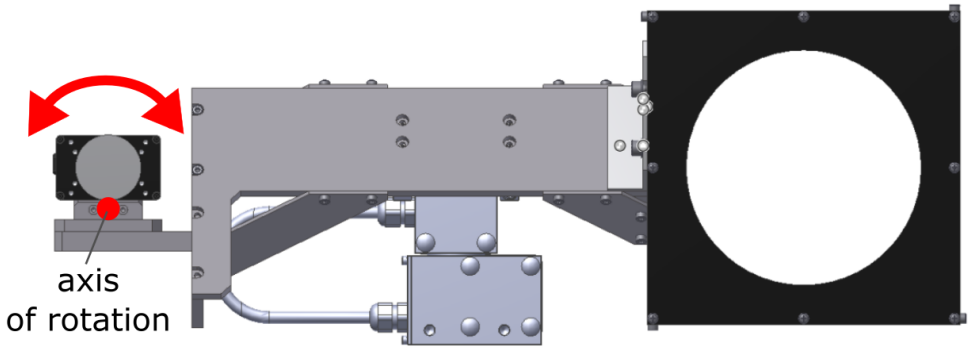


Figure 6.2: Front view of the top part of the camera unit. The red arrow shows the possible rotation of the camera along the axis parallel to the optical axis.

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 6.3). Once set, the mount with the camera is fixed to the manipulator using two set screws.

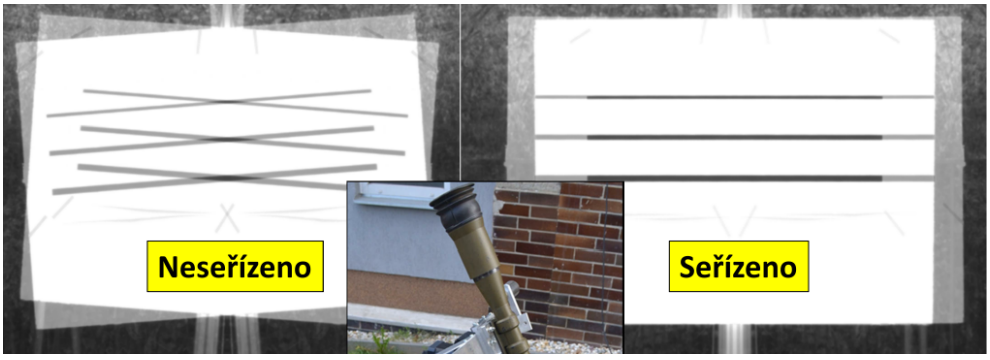


Figure 6.3: A blended image stream from the camera before (left) and after (right) rotating the camera along the optical axis to the correct position.

6.2.2 Rotation along the azimuthal axis

The mount can still rotate along the axis parallel to the azimuthal axis (see Figure 6.4). It is necessary to ensure that the optical axis of the camera is perpendicular to the elevation axis.

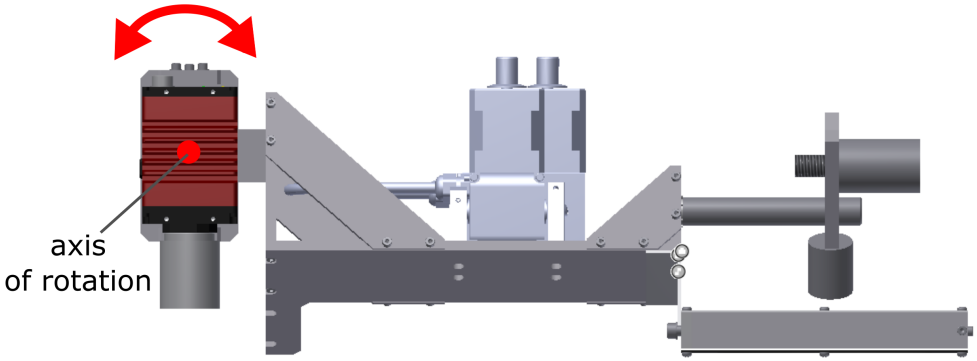


Figure 6.4: Top view of the top part of the camera unit. The red arrow shows the possible rotation of the camera along the axis parallel to the azimuthal axis.

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 6.6).

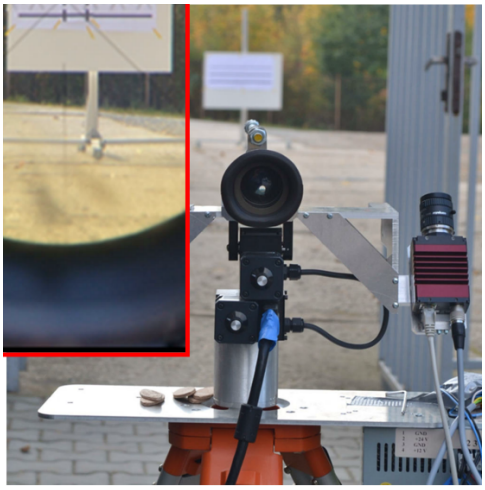


Figure 6.5: A telescope mounted on top of the manipulator. A person looking through a telescope sees a crosshair - a tip of a triangle.

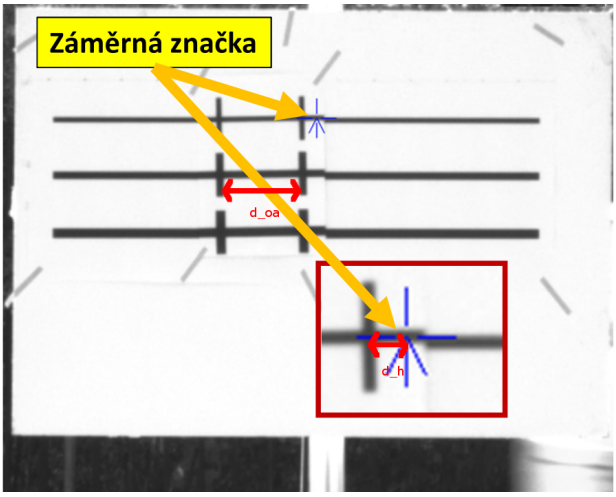


Figure 6.6: Rectification target with the pairs of black crosses. The two crosses in a pair are d_{ao} m appart. A digital corsshair is displayed in order to find the horizontal offset d_h.

A military optical monocular telescope (see Figure 6.5) 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 4):

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

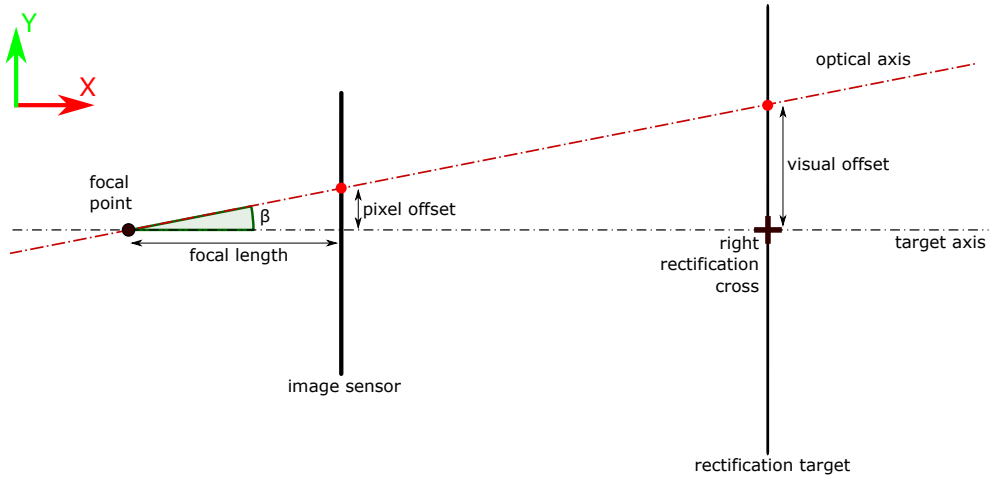


Figure 6.7: The top view schema of a rectification target being projected to the image sensor of the camera. The value of the angle β is one of the output of the rectification process.

6.2.3 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 4), 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 6.8). 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 4).

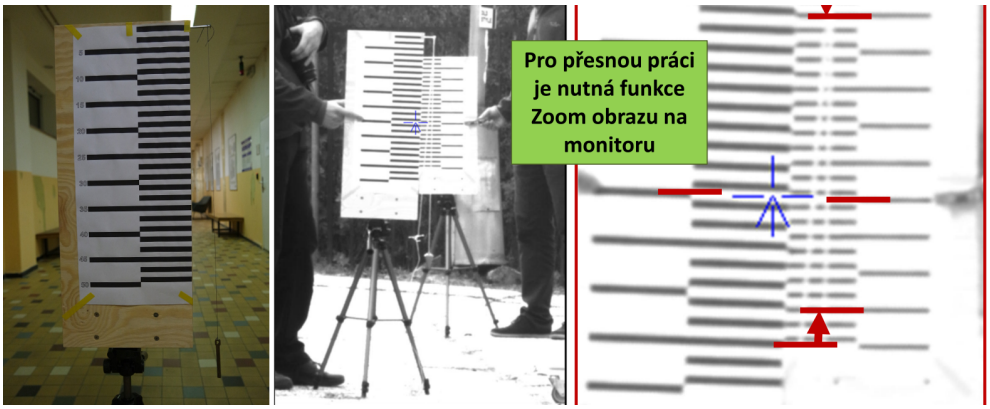


Figure 6.8: Front target of a pair of the rectification targets used to find a default elevation angle (left). A screenshot from the image stream of the camera with the crosshair focused on a row where the marks of the rulers align (right).

Chapter 7

Detection and tracking

- === DETEKCE VZDALENYCH OBJEKTU V OBRAZE === - === SLEDOVANI OBJEKTU VE VICEKAMEROVYCH SYSTEMECH === - === ALGORITMY VHODNE PRO RESENI ULOHY ===

  - TLD - moznost sdileni info o appearance modelu mezi CUs

Chapter 8

Cooperation among camera units

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

Chapter 9

Target localization

- triangulace - === LOKALIZACE OBJEKTU VE VICEKAMEROVYCH SYSTEMECH === - === ALGORITMY VHODNE PRO RESENI ULOHY ===
== (obrazek z rvizu (protnuti primek) ==

Chapter 10

Implementation

The whole system is built on the robotic framework Robot Operating System (for details see Section 10.1). 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 10.3). When confronted with the overview of all subsystems making up the whole system presented in Section 3 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

10.1 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 [4].

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

[5], 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.

10.1.1 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 10.1), it was chosen as a main implementation platform.

Table 10.1: 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 [4], 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

10.1.2 Standard ROS packages

ROS provides a wide range of standard packages for interaction with various hardware devices. 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.

10.2 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.

10.2.1 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 3). All stations communicate with each other vie Ethernet switch. The peripherals allowing the operator to manually control the system are connected to the overview station (see Figure 10.1).

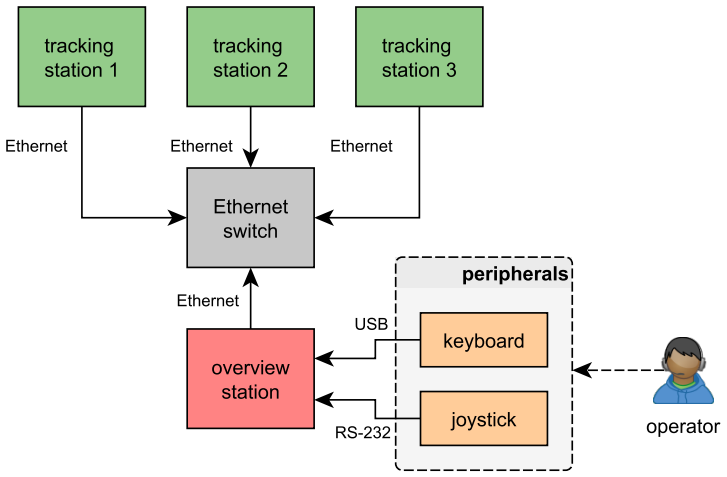


Figure 10.1: The big picture diagram of the main components of the OLS.

A camera station itself consists of a camera unit (see Section 4) 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 10.2).

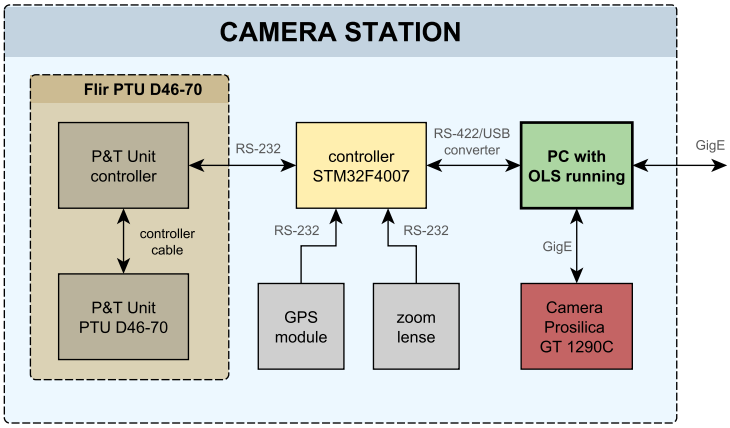


Figure 10.2: The diagram of the hardware components of the camera unit depicting both the hardware devices and the communication standards.

10.2.2 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 10.3.

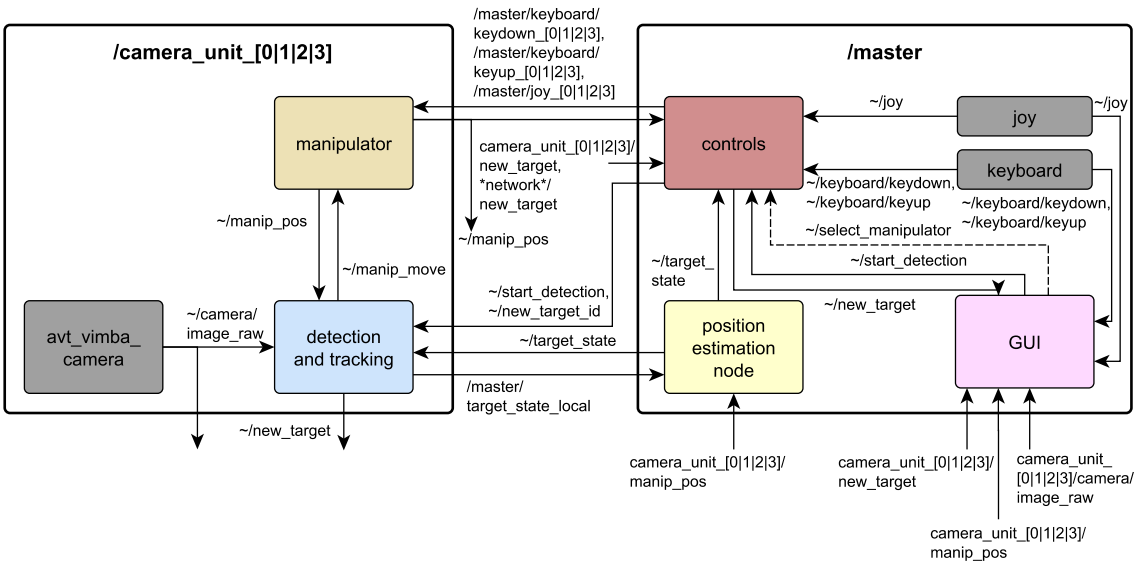


Figure 10.3: The diagram of the software components represented by the ROS nodes. The communication among topics is implemented using ROS topics (normal arrows) and ROS services (dashed arrows).

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.

10.3 Application of Gazebo

Gazebo[7] is a physical simulator developed by the OSRF[8] 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 10.4).

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 4), 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.

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.

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

Figure 10.4: 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).

10.4 External libraries

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

OpenCV Open Source Computer Vision Library[9] 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[10] implements the data structures and methods for fast and convenient solving of linear algebra problems.

OpenTLD The OpenTLD library[11] represents an open source C++ implementation the TLD tracking algorithm (see Section 7).

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

Chapter 11

Experiments and results

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

Chapter 12

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

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

[3] 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.

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

[5] 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.