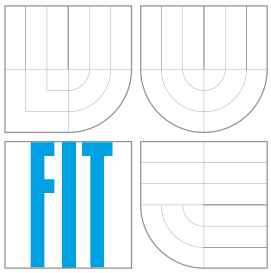
BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

OPTICAL LOCALIZATION OF VERY DISTANT TARGETS IN MULTICAMERA SYSTEMS

OPTICKÁ LOKALIZACE VELMI VZDÁLENÝCH CÍLŮ VE VÍCEKAMEROVÉM SYSTÉMU

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Bc. JAN BEDNAŘÍK

SUPERVISOR
VEDOUCÍ PRÁCE

prof. Ing. ADAM HEROUT, Ph.D.

BRNO 2016

Abstract

This work presents a system for semi-autonomous optical localization of distant moving targets using multiple P&T cameras. The cameras were calibrated and stationed using custom designed calibration targets and methodology with the objective to alleviate the main sources of error which were pinpointed in thorough precision analysis. The detection of the target is performed manually, while the visual tracking is automatic and builds on two state-of-the-art approaches. The estimation of the location in three-space is based on the N-view triangulation working with noisy measurements. A basic setup consisting of two camera units was tested against static targets and a moving terrestrial target, and the location estimation precision was compared to the theoretical model. The modularity and portability of the system allows fast deployment in a wide range of scenarios including perimeter monitoring or early threat detection in defense systems, as well as air traffic control in public space.

Abstrakt

Tato práce představuje semiautonomní systém pro optickou lokalizaci velmi vzdálených pohyblivých cílů za pomocí několika P&T kamer. Kamery byly kalibrovány a zastaničeny pomocí speciálně navržených kalibračních terčů a metodologie, jejímž účelem je minimalizovat hlavní zdroje chyb, jež byly objeveny během důkladné analýzy přesnosti. Detekce cíle probíhá manuálně, zatímco vizuální sledování je automatické a staví na dvou state-of-the-art přístupech. Odhad 3D lokace cíle je založen na triangulaci z N pohledů pracující s nepřesnými měřeními. Základní sestava o dvou kamerových jednotkách byla otestována na statických cílech a pohybujícím se pozemním cíli, přičemž byla přesnost odhadu lokace cíle porovnána s teoretickým modelem. Díky modularitě a přenosnosti je možné systém nasadit v široké škále situací, jako je například monitoring vytyčeného území, včasná detekce hrozby v bezpečnostních systémech nebo řízení vzdušeného provozu.

Keywords

optical localization, mutli camera localization, stereovision, object detection, object tracking, triangulation, motion prediction, UAV, physical simulation, ROS, Gazebo, kinematic chain, camera rectification

Klíčová slova

optická lokalizace, lokalizace více kamerami, stereovize, detekce objektů, sledování objektů, triangulace, odhadh pohybu, UAV, fyzikální simuace, ROS, Gazebo, kinematický řetězec, rektifikace kamery

Reference

BEDNAŘÍK, Jan. *Optical Localization of Very Distant Targets in Multicamera Systems*. Brno, 2016. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Herout Adam.

Optical Localization of Very Distant Targets in Multicamera Systems

Declaration

I hereby certify that this thesis is a presentation of my original research work and I have excercised reasonable care to ensure it does not to the best of my knowledge breach any law of copyright. Wherever contributions of others are involved, every effort is made to indicate this clearly, with due reference to the literature, and acknowledgement of collaborative research and discussions. The work was done under the guidance of Ing. Adam Herout, Ph.D. at the Brno University of Technology.

.....
Jan Bednařík
May 23, 2016

Acknowledgements

I thank my Master's thesis supervisor, prof. Ing. Adam Herout, Ph.D., for guidance of my work. This work is supported by company RCE systems s.r.o. which provided the development space, necessary hardware sources, and consultations and it is greatly appreciated. I thank my adviser, doc. Ing. Vladimír Čech, CSc., for providing his insight into the project.

Contents

Chapter 1

Introduction

An autonomous localization of arbitrary moving targets is an essential system component used in multiple domains, such as air traffic control, robotic workspaces or surveillance and defense systems. If the sensory data measured by the target are available it is straightforward to derive its location (by means of the GPS, radio multilateration, etc.). There are scenarios, however, were the target is unable (malfunctioning aircraft) or reluctant (UAV intruder) to expose its location. Then the localization estimation system is left with its own observations. Among others, nowadays, the air traffic control (ATC) and national defense systems are the most widely used applications of tracking and localizing the moving objects.

In case of the ATC, the airports mainly rely on the multilateration systems which specialize on surveillance and control of air traffic during all flight phases [18]. The design of such a system expects that the aircraft are equipped with a secondary surveillance radar transponders which periodically emit the signals to the ground receiving stations, however, primary radars are widely used as well [35].

In the military segment the object localization mainly serves the purpose of the early threat detection where it is necessary to identify and track the intruder which might pose a threat for a given area under protection. In this scenario only primary radars are used since the objects of interest are not expected to cooperate.

Besides the ATC and the national defense there are quite a lot other use cases where the autonomous localization systems are or could be employed. The road traffic security is one of such examples. The localization systems can be used by the police to measure the speed of vehicles and prospectively to pinpoint the exact locations where the speeding occurred. Autonomous traffic analysis is another well suited application for localization systems as the main requirement estimate the tracks of given vehicles with regards to the underlying geographical map.

Even though being widely used, radar based localization systems suffer from several drawbacks. Radar is a device based on the active emission of the radio signal [50]. In order to localize distant objects, a huge amount of energy must be radiated to make sure it would return from the target and a small amount of energy returned might be easily disrupted [35]. What is more, in defense applications it is not desirable that the tracked object would find out that it is being tracked, which is the condition an actively radiating system, such as a radar, cannot achieve. Last but not least, the professional class radars in use by both public segment and military are in general expensive, large and heavy devices not suitable for mobile deployment.

This work presents a semi-autonomous passive multi-camera system for tracking and localizing arbitrary objects — the Optical Localization System (OLS) — which is based

merely on ordinary RGB cameras. Since the system does not rely on active emission of a signal but rather captures the optical information from the environment, it can be used for secret localization of moving targets. The system is designed to be well suited for mobility and temporary deployment since each camera station used weigh no more than twenty kilograms and the whole system is inexpensive by comparison to radars as well.

The principle of the optical localization of objects, which is based on triangulation, is well known for quite a long time. The first devices which were designed to allow an operator to estimate a distance to a given object using mechanical and optical principles — optical range finders — emerged in the second half of the 18th century [7]. Ever since, the optical range finders had been mostly used for military operations in order to estimate the position of either naval, airborne or terrestrial targets until the World War II when radar was invented.

The OLS aims to utilize the same principle as the old optical range finders. However, instead of an optical device requiring the operator to aim on the target manually the OLS takes advantage of the RGB cameras and the image processing techniques capable of tracking and localizing the target autonomously without the need for a human to interfere. Furthermore, with the use of sensors capable of finding a geographical position of each observing unit the OLS system estimates a global geographical position of a given target represented in standard coordinate system (UTM and/or WGS84). A simple schema explaining the operation of the OLS is depicted in Figure 1.1.

This work is organized as follows: In Chapter 2 the existing approaches to visual tracking, multi-camera target localization and 3D environment reconstruction are summarized and their suitability for the OLS is discussed. Chapter 3 provides thorough analysis of the main sources of localization imprecisions in the OLS and proposes the methods to alleviate the error. Chapter 4 describes the design of the OLS from the perspective of system model, visual tracking, triangulation with noisy measurements and occlusion prediction. Chapter 5 explains the technical details of the implementation and Chapter 6 presents the results of conducted experiments. Finally, Chapter 7 concludes the work.

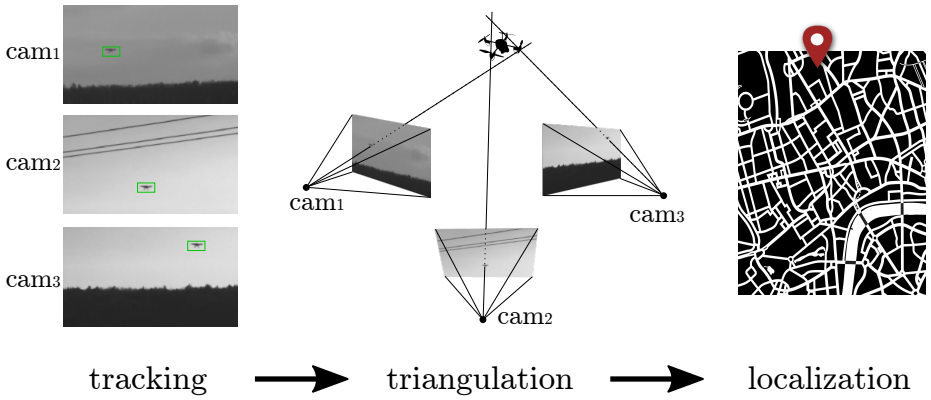


Figure 1.1: In the OLS a target is tracked by multiple cameras and triangulation is used in order to estimate its global location.

Chapter 2

Related Work

Being a complex and semi-autonomously working system the multi-camera optical localization system comprises problematics ranging over several different areas. Most importantly, robust visual tracking capable of long-term tracking of arbitrary target which might exhibit time varying appearance and which might move in a cluttered environment must be employed. Furthermore, a suitable approach for estimating the position of the target in 3-space given noisy measurements must be proposed. Finally, occlusion prediction using 3D environment reconstruction can ne incorporated as a possible extension. The overview of visual tracking approaches and the discussion of their suitability for the OLS is given in Section 2.1, the most widely used methods for localizing targets in 3-space given projective geometry are described in Section 2.2 and the problematics of 3D environment reconstruction is introduced in Section 2.3.

2.1 Object Tracking

This section discusses the various approaches to tracking of the objects using the computer vision techniques. First, the importance of the suitable object representation is explained and the properties of various object models as well as their advantages and disadvantages with regards to the requirements of the OLS system are discussed. Different categories of tracking algorithms are then described and two specific approaches which are most appropriate for the OLS – the *TLD tracker* and the *BGF tracker based on the particle filter framework* – are explained in more detail.

2.1.1 Object Model

The choice of how the targets are represented determines the domain of approaches used for visual tracking due to the strong relationship between the algorithm and the object model. Neither of the stat-of-the-art approaches is universal enough to cope with all the difficulties and disturbances, such as the illumination change, occlusion, cluttered background, motion blurring or appearance change due to deformation and/or transformation of the object [30] which might occur over the course of tracking. Therefore, the model should suit a priori known tracking conditions (e.g. size, speed, rigidity and motion model of the target, number of targets, camera motion, background model etc.). Yilmaz et al. classified the model representations into two categories in their review [56] – the *shape* and the *appearance*.

shape representation A *shape* model encompasses points [49], contours [27, 23] or articulated models [11, 33] (see Figure 2.2). The point representation is not suitable for the OLS since the distant objects appear relatively small in the image and might not provide enough distinctive points (see Figure 2.1). Both contours and articulated models are mostly used for tracking non-rigid deformable objects which is not the main concern of the OLS. Additionally, the accuracy of fitting a contour to a target strongly depends on the convergence criteria of the energy minimization function, thus they might be computationally expensive.

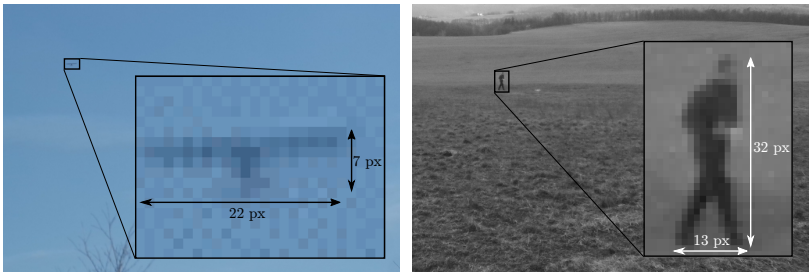


Figure 2.1: Due to their high distance the tracked objects might appear small as projected to the image plane. Therefore, a suitable object model must be chosen to avoid tracking failures.

appearance representation An *appearance* model is represented either by rectangular template [32, 21] or a weighted kernel [8, 13] (see Figure 2.2). The main advantage of both representations is the fact they contain both the spatial and appearance information, additionally they scale well to varying object size (approaching and receding object). The appearance model seem to suit the requirements of the OLS, thus the tracking approaches based on variants of both template and kernel model will be used (see Sections 2.1.3 and 2.1.4).

Figure 2.2: Various approaches to tracked object modeling. (a) Keypoints, (b) contour and (c) articulated model fall into the *shape representation* category whereas (d) template and (e) kernel belong to the *appearance representation* category. The intensity of a red color in case (e) denotes the weight of the given pixel in the given (ellipse shaped) kernel.

2.1.2 Tracking approaches

The main purpose of the tracker is to iteratively estimate the trajectory of the tracked object from frame to frame. There is a wide range of approaches to visual tracking and since they

usually combine multiple various methods in order to reinforce the tracker robustness they cannot be really divided into distinct categories in a straightforward manner. However, the approaches can be coarsely categorized by the selection of the object representation.

To reinforce the tracker robustness the motion models are often used. Kalman filter and particle filter are the most popular ones [9, 21].

Keypoint Tracking

Keypoint tracking represents one of the most common approaches [49, 36]. The *keypoint* is understood to be a single point in a an image which represents a small image region – a *point descriptor* – and which should be highly discriminative and invariant to various image transformations. There are many widely used keypoint detectors/descriptors e.g. SIFT, SURF, ORB, FREAK, etc. [31, 3, 42, 40] and they differ mainly in the means of matching precision and computation speed [44, 34]. What the tracker does is that it detects the keypoints and their descriptors in each frame, selects those representing the object, finds the correspondences and computes the transformation from the previous frame. Even though the keypoint tracking is well established approach it cannot be used in the OLS due to the insufficient size of the tracked objects as was explained in Section 2.1.1.

Kernel Tracking

Kernel approaches are based on so-called *kernel*. Basically, the feature target representation is spatially masked with an isotropic kernel (for illustration see Figure 2.2 (e)) which assigns the largest weights to the pixels in the middle while the weights decay in the directions towards the edges of the kernel. This enables a spatially-smooth similarity function to be defined. Consequently, this function can be optimized in the means of target position using traditional gradient based methods such as gradient descent [8]. To boost the robustness of the tracker multiple collaborative kernels might be used [13, 47]. The strategy to distinguish which pixels in the kernel/template are more or less reliable is also utilized in the BGFG tracker which the OLS is based on (see Section 2.1.4).

Tracking-by-Detection

This class of approaches heavily utilizes the detection principles in combination with motion based approaches to localize the object [6, 26]. Depending on the object model the detection might be performed either by detecting keypoints and matching them against the pretrained model [41, 38] or by dividing the image into individual patches in which the object is searched for. For each patch the template matching is performed [43, 21] (using SSD[1], SAD[2] or NCC[3] as a similarity measure) or feature set is extracted; consequently, the model presence probability is evaluated using the generative or the discriminative classifier [53, 58]. Since the exhaustive search within the whole image is computationally expensive, the cascade classifiers might be applied [51, 5]. The TLD approach which is used in the OLS utilizes the object detector in order to correct and/or reinitialize the tracker (see Section 2.1.3).

[1] Sum of Squared Differences.
[2] Sum of Absolute Differences.
[3] Normalized Cross Correlation.

Motion Based Tracking

This category of approaches attempts to extract the motion occurring between the consecutive images. The *optical flow* method which in general tries to find the motion of individual pixels in the image can be used to track the keypoints [57] or to produce the binary feature images and consequently the blobs corresponding to moving objects [1]. Alternatively, the moving object can be detected in the image regions yielding the highest response of *frame differencing* (known also as *background subtraction*) [37, 29]. The BGFG tracker which the OLS uses is based on the frame differencing to estimate the appearance model and most likely location of the tracked object (see Section 2.1.4).

Motion Modeling

The tracking can be approached through the model of a discrete-time dynamic system, where the aim is to estimate the current state for each incoming frame [8]. The state can be represented as a mere 2D position of a target (in pixel-coordinates) or other parameters such as velocity or acceleration of a target can be modeled a well. Thanks to the motion model the (computationally expensive) exhaustive search for the object can be reduced to the vicinity of the current target position estimate.

Kalman filter Kalman filtering is one of the widely used technique for recursively evaluating the current state of a target given the measurement corrupted by the measurement noise and the prediction of the next state corrupted by the process noise [54, 9]. It is based on the assumption that the state posterior density is Gaussian and thus can be parametrized by means and covariances. However, this assumption might not hold. In case of the OLS a camera used for tracking is often in motion and the sensory data about its position which could be used to stabilize the motion might be imprecise or not available at all (for illustration see Figure 2.3). Consequently, the position of a tracked object can change rapidly from frame to frame and thus to defy the assumption of the Gaussian distribution of the state posterior density. Furthermore the basic Kalman filter is based on unimodal Gaussian distribution which prevents it from keeping multiple hypothesis for a single target.

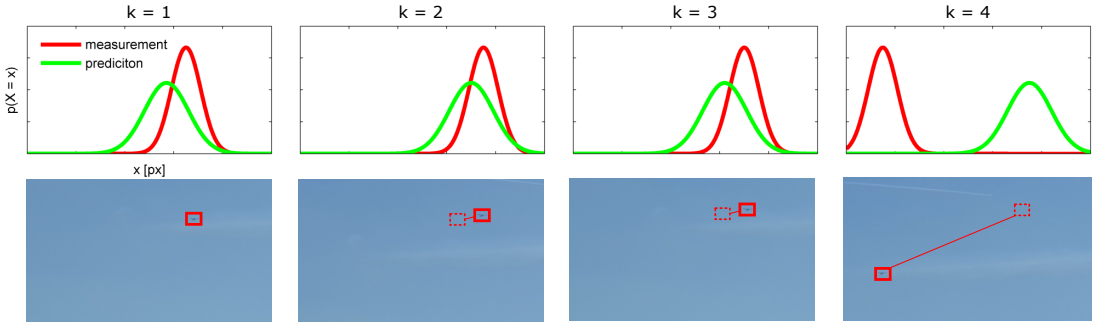


Figure 2.3: The evolution of measurement and prediction probability density of the tracked target in the Kalman filter framework. The target apparently moves along a line with constant velocity, however, in frame $k = 4$ it suddenly changes its position due to the rapid camera movement which is something that Kalman filter cannot cope with.

particle filter The particle filter represents the most general class of filters which can cope with non-Gaussian state and measurement processes as well as with tracking mul-

tiple hypothesis [4]. Current state of the system is represented as a *particle* $(\vec{x_t^i}, w_t^i)$ – a vector of parameters $\vec{x_t^i}$ describing the properties of tracked object (e.g. position, velocity, acceleration, etc.) and assigned scalar weight w_t^i. The suitable *fitness function* must be proposed to evaluate how well a particle fits to the observed data. Using finite number of particles the particle filter basically samples the fitness function (which might be arbitrarily complex and non-differentiable) in an attempt to find the optimum. A bootstrap particle filter (BPF) is a variant of particle filter widely used for visual tracking [25]. It follows the sequential importance sampling principle – in each iteration the particles with higher weights are duplicated while the particles with lower weights are discarded [12]. This enables higher resolution sampling of the fitness function only in the parts which are likely to contain the (local) optimum. The BPF iteratively performs four main steps – *resampling*, *prediction*, *update* and *weights normalization*. The detailed breakdown of all steps is depicted in Algorithm 1. Note that the function *predict*() in *update* step should correspond to the required motion model of the tracked object (a normal distribution $\mathcal{N}(\mu, \sigma)$ is given as an example) and it can be designed to allow for the rapid camera motion which suits the needs of the OLS. Therefore, the tracker based on BPF is utilized (see Section 2.1.4).

Algorithm 1: Tracking using BPF

Input: A measurements M, a set of particles $P = \{(\vec{x_t^1}, w_t^1), (\vec{x_t^2}, w_t^2), ..., (\vec{x_t^{n_p}}, w_t^{n_p})\}$

Output: The particle $(x_t^{i_{bestParticle}}, w_t^{i_{bestParticle}})$ representing the state estimation with best weight w_t^i

/* Resampling (importance sampling) */

1 $resampleParticles()$

/* Prediction */

2 **foreach** $(\vec{x_t^i}, w_t^i) \in P$ **do**

3 **foreach** *parameter* k_j *of* x_t^i **do**

4 $predict(k_j)$ /* (e.g. $k_j = k_j + x,\ x \sim \mathcal{N}(\mu_j, \sigma_j)$) */

5 **end**

6 **end**

/* Update */

7 **foreach** $(\vec{x_t^i}, w_t^i) \in P$ **do**

8 $w_t^i = fitnessFunction(M, (\vec{x_t^i}, w_t^i))$

9 **end**

/* Estimate final state e.g. using MAP */

10 $(x_t^{i_{bestParticle}}, w_t^{i_{bestParticle}}) = (\vec{x_t^i}, w_t^i) \in P : \nexists\ (\vec{x_t^j}, w_t^j) \in P : w_t^j > w_t^i$

/* Weights normalization */

11 **foreach** $(\vec{x_t^i}, w_t^i) \in P$ **do**

12 $w_t^i = \frac{w_t^i}{\sum_{j=1}^{n_p} w_t^j}$

13 **end**

2.1.3 Tracking-Learning-Detection

The Tracking Learning Detection (TLD) [26] 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. TLD is based on the appearance representation of the target, specifically a set of templates is stored and continuously updated. Additionally, TLD can be implemented to run in real-time. Such properties meet the requirements of the OLS (see Section 1), thus TLD is incorporated in OLS as one of the visual trackers and this section explains its design in more detail.

Since a long-term tracking cannot be easily achieved either by a mere 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 over the course of the tracking.

A conceptual diagram of the TLD algorithm is shown in Figure 2.4. 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 NCC.

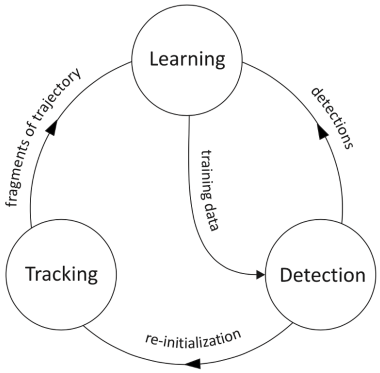


Figure 2.4: A diagram of the main TLD components. Image is adopted from authors Kalal, Mikolajczyk and Matas [26].

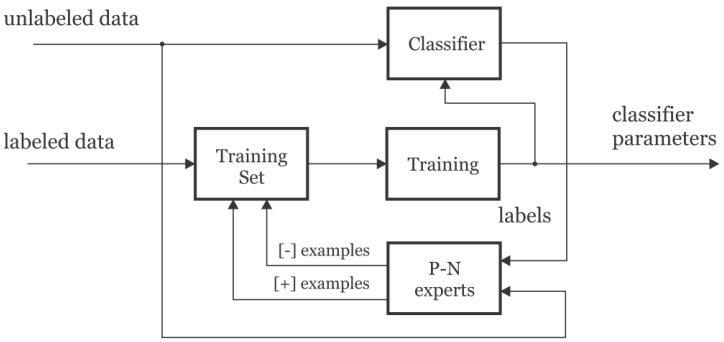


Figure 2.5: A diagram of the PN learning process. Image is adopted from authors Kalal, Mikolajczyk and Matas [26].

Detection The detector treats each frame as an independent one and scans a full image. A scanning window is used and it 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. In case of TLD the cascade

classifier 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 subregion be rejected by any stage later stages ignore it completely.

Tracking The tracking subsystem is based on the 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 user) it tends to make errors as a video stream progresses since the moving object of interest changes its appearance due to the transformation caused by its motion. Therefore, online component called *P-N learning* is incorporated in the system and it gradually extends the training sets. Two experts are used. *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 2.5).

2.1.4 Tracking Using Background/Foreground Modeling and Particle Filter

The autonomous tracking uses the implementation of the visual tracker combining the background subtraction, motion model and object model in the particle filter framework proposed in [21] (BGFG tracker). Thanks to both particle filter and inter frames homography computation this approach can even cope with the moving cameras. Partial occlusion is handled using foreground modeling and the tracker is capable of running in real-time. Therefore, this approach is suitable for the OLS as well and it is used as an alternative to TLD. The operation of the tracker is described below.

The target is represented as a rectangular template (consisting of gray-scale intensity values) which is normalized to the size 24×24 pixels. The template is created only once during the initialization, thus the tracker could fail if the target changed its appearance significantly during the course of tracking. However, in case of very distant targets only a slight change occurs.

The Bootstrap particle filter (BPF) [25] is used to generate and evaluate candidate positions of the target. Each particle (i.e. the state of the system) is represented as $\vec{x_n} = (x, y, v_x, v_y, h, w)$, where (x, y) represents the 2D position of the target, (v_x, v_y) represents the estimated speed of the target and (h, w) represents the bounding box size.

The perturbations in the observed position of the target caused by the moving camera are alleviated using the motion model which is applied in the *prediction* step of the BPF:

$$pos_{n+1} = pos_n + vel_n + \gamma_{pos} \sim \mathcal{N}(\mu, \sigma), \tag{2.1}$$

$$vel_{n+1} = vel_n + \gamma_{vel} \sim \mathcal{N}(\mu, \sigma), \tag{2.2}$$

$$bb_{n+1} = bb_n + \gamma_{bb} \sim \mathcal{N}(\mu, \sigma), \tag{2.3}$$

where scalar pos_n is the x or y position, scalar vel_n is the x or y velocity, scalar bb_n is the w or h size of the bounding box in time n, and γ is the noise drawn from the Gaussian distribution $\mathcal{N}(\mu, \sigma)$, where scalars μ and σ parameters are set empirically for each parameter.

In the *update* step, each particle is assigned a new weight w using the objective function reflecting the similarity of the template and the candidate patch, The function serves the purpose of the similarity measure and it is based on a SSD:

$$w = \sum_{(x,y)\in I} e^{min(M_t^{(x,y)}, M_c^{(x,y)})}(1 - |I_t^{(x,y)} - I_c^{(x,y)}|)^2, \tag{2.4}$$

where M_t and M_c are the foreground masks (FM) of the template and the current candidate respectively, I is the image, t, c subscripts denote template and candidate patch respectively, and (x, y) superscript denotes indexing 2D array (an image). The FMs are estimated by subtraction of the two images where the bounding boxes denoting the position of the target do not overlap (the FM M_t is estimated only once). The resulting estimate of the target position is chosen using the Maximum a posteriori approach.

In order to enable the motion of the camera the transformation between each pair of adjacent frames is estimated by detecting and tracking the keypoints using KLT tracker [49] and then estimating the homography using the RANSAC algorithm [19].

2.2 Multi-camera Target Localization

This section introduces the problematics of estimating the location of a target in 3-space given the arbitrary number of corresponding image points in 2-space. First, the suitable camera model is described, then the standard stereo setups are examined, next the notion of triangulation with noisy measurements is presented and finally the existing approaches to target location estimation as well as their suitability for OLS are discussed.

2.2.1 Camera Model

Localization of a target corresponds to a process of mapping image locations of a target in 2-space from multiple cameras to a location in 3-space. Thus it is necessary to define a model of a camera first. The *finite pinhole camera* based on the *central projection* is a standard approach to model cameras with CCD like sensors [19] and it is used to model hardware cameras in OLS as well. Finite pinhole cameras use projection matrix P which maps a point $\vec{X}$ in 3-space to an image location $\vec{x}$ in 2-space (see Figure 2.6):

$$\vec{x} = P\vec{X}, \tag{2.5}$$
$$P = KR[I|-C], \tag{2.6}$$
$$\tag{2.7}$$

where I is the identity matrix, R and C are the rotation and translation matrices representing the orientation and position of the camera frame with respect to the world frame and K is the *intrinsics matrix* (or *camera calibration matrix*) consisting of a focal length f, coordinates of the principal point $PP = (x_0, y_0)^T$ and skew parameter s:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.8}$$

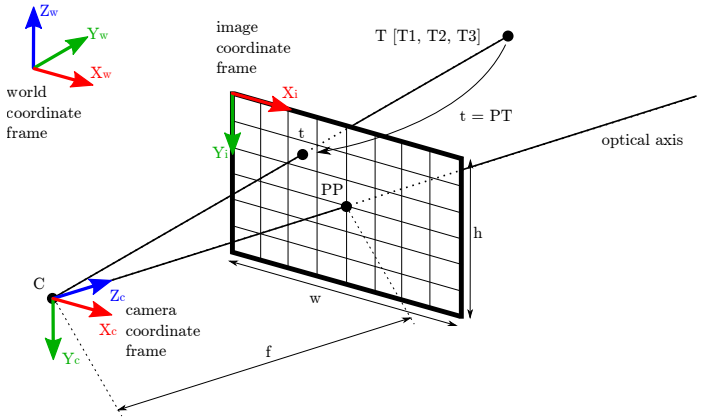


Figure 2.6: Perspective projection of pthe oint T in 3-space to the point t in 2-space located on the image plane in the finite pinhole camera model.

2.2.2 Stereo Setups

With two calibrated cameras (i.e. K, R and C matrices are known) observing the same portion of the environment it is possible to estimate the location of the given point/object in 3-space. The *canonical stereoscopic system* is one of the widely used setup capable of estimating the depth of points in the scene [10]. The optical axes of both cameras are collinear and the notion of *disparity* is introduced. Disparity refers to the difference in the image location of the same 3D point when projected under perspective to two different cameras [46] and the coordinates of a point in 3-space can derived using following equations (see Figure 2.7):

$$z = fb/(x_l - x_r) = fb/d, \tag{2.9}$$

$$x = x_l z/f = b + x_r z/f, \tag{2.10}$$

$$y = y_l z/f = y_r z/f, \tag{2.11}$$

where f is the focal length, b is the baseline, x_l and x_r are the horizontal distances between the principal points PP of the respective camera and the projection t_l and t_r respectively (the same applies for y_l and y_r in vertical direction), d is the disparity and $(x, y, z)^T$ are the 3-space coordinates of the target. Nevertheless, the OLS cannot be modeled as canonical stereoscopic system due to the fact that it is designed to work with arbitrary number of cameras and furthermore, the extrinsics of the cameras are not fixed (the cameras rotate freely in space, see Section 4).

In *general stereo setup* the collinearity of the optical axes is not required. However, location of a point in 3-space cannot be simply estimated through triangle similarity. On each camera a line in 3-space mapping to a point $\vec{p_i}$ in 2-space has to be first computed using *back-projection* (see Figure 2.7):

$$X(\lambda) = P^{+}\vec{x} + \lambda C, \tag{2.12}$$

where P^{+} is the pseudo-inverse of P ($P^{+} = P^T(PP^T)^{-1}$). The location of the target in 3-space is then given as an intersection of all back-projected lines. Given its nature the OLS can be modeled as a general stereo setup extended to use arbitrary number of cameras (instead of only two cameras).

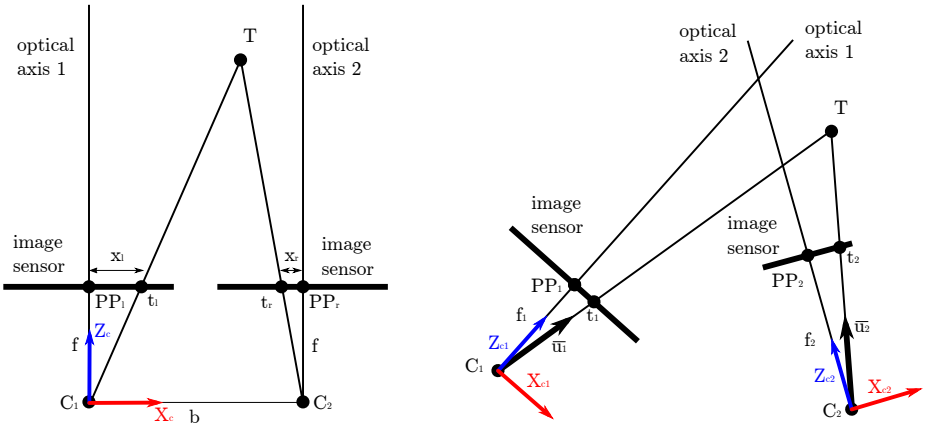


Figure 2.7: In canonical stereo setup (left) the 3D coordinates of the given point can be computed using the notion of disparity. On the other hand, in case of general stereo setup (right) the intersection of lines in 3-space defined by the target projected to the image space of each camera must be found.

2.2.3 Triangulation with Noisy Measurements

The process of finding the location of a target in 3-space as an intersection of back-projected lines is called *triangulation*. In ideal case the projection matrices P_i and calibration parameters are known precisely for both cameras and the lines in 3-space intersect. However, in real world the system is subject to both systematic and random error (see Chapter 3), consequently the lines in 3-space might become skew (i.e. they are not guaranteed to intersect anymore) since the stereo system does not satisfy the epipolar constraint [19] (more detailed explanation can be found in Section 2.3):

$$\vec{x'}^T F \vec{x} \neq 0. \tag{2.13}$$

In general, the same problem holds within each pair of cameras in N-view setup. Instead of the intersection the minimum distance between each pair of skew lines might be found as a line segment perpendicular to both skew lines (see Figure 2.8) via following equation [16]:

$$z_{ij}\vec{p_i} = \vec{T_{ij}} + z_{ij}R_{ij}\vec{p_j} + \lambda_{ij}(\vec{p_i} \times R_{ij}\vec{p_j}) \tag{2.14}$$

$$for\ i, j = 1, 2, ..., N \wedge i \neq j, \tag{2.15}$$

where $\vec{p_i}$ is the direction of back-projected line in camera i, z_{ij} gives the scale of vector p_i so as to define a point P_{ij} which is the closest to the line back-projected from camera j, T_{ij} and R_{ij} are the translation and rotation of the coordinate frame of the camera j with respect to the coordinate frame of the camera i and λ_{ij} defines the length of the line segment connecting both back-projected skew lines.

A similar approach to finding the closest distance between each pair of cameras is used in the OLS (see Section 4.3).

2.2.4 Estimation of Target Location

Since the back-projected lines are skew, there is not the only correct solution to the localization problem. Contrarily, the position of the target must be estimated. Hartley and

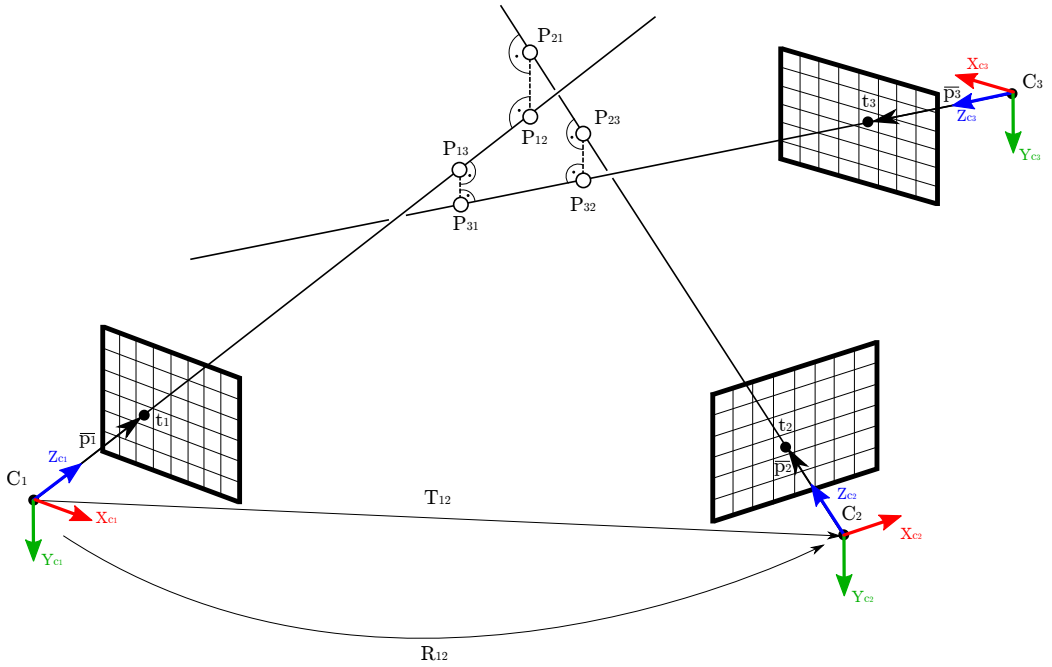


Figure 2.8: An example of a three-view setup and the target location estimation using triangulation. Since all cameras are subject to systematic and random error the back-projected lines are skew, thus there is no intersection. The closest distance between each pair of lines is given by the line segment perpendicular to both lines.

Zisserman [19] propose a couple of methods where the approaches basically boil down either to solving the overdetermined system of linear equations or to minimization of the geometric error. Both approaches are briefly described below.

Direct Linear Transformation (DLT) The DLT algorithm is based on the assumption that an overdetermined system of linear equations in the form $\mathbf{A}\vec{x} = \vec{0}$ (where $\mathbf{A}$ is the matrix of coefficients, $\vec{x}$ is the vector of unknowns and $\vec{0}$ is the zero vector) is available and that given the noise there is no exact solution. In case of a stereo setup where the image points $\vec{t_1}$ and $\vec{t_2}$ in 2-space on each camera correspond to a target $\vec{T}$ in 3-space the overdetermined system of linear equations can be defined with vector $\vec{x} = (t_1^x, t_1^y, t_2^x, t_2^y)$ and matrix $\mathbf{A}$:

$$A = \begin{bmatrix} t_1^x \vec{p_3^1}^T - \vec{p_1^1}^T \\ t_1^y \vec{p_3^1}^T - \vec{p_1^2}^T \\ t_2^x \vec{p_3^2}^T - \vec{p_2^1}^T \\ t_2^y \vec{p_3^2}^T - \vec{p_2^2}^T \end{bmatrix}, \tag{2.16}$$

where $\vec{p^r_{cam}}$ is the rth row of the projection matrix $\mathbf{P_{cam}}$ of the camera cam. The ultimate solution is found using the SVD[4] as the singular vector corresponding to the smallest singular value of $\mathbf{A}$.

[4]Singular Value Decomposition

Reprojection Error Minimization Similarly as DLT this method assumes that the correspondence between image points $\vec{t_1}$ and $\vec{t_2}$ does not meet the epipolar constraint. The aim thus is to estimate the position of the target $\vec{T}$ in 3-space which projects to image points $\vec{t_1'}$ and $\vec{t_2'}$ satisfying the epipolar constraint and which at the same time minimizes the reprojection error function ref:

$$ref(\vec{t_1}, \vec{t_2}) = d(\vec{t_1}, \vec{t_1'})^2 + d(\vec{t_2}, \vec{t_2'})^2, \tag{2.17}$$

where $d(\vec{x}, \vec{x'})$ is the Euclidean distance between the measurement $\vec{x}$ and reprojected image point $\vec{x'}$ (see Figure 2.9). The reprojection error function can be either minimized using any optimization method or the minimum can even be found non-iteratively in closed form.

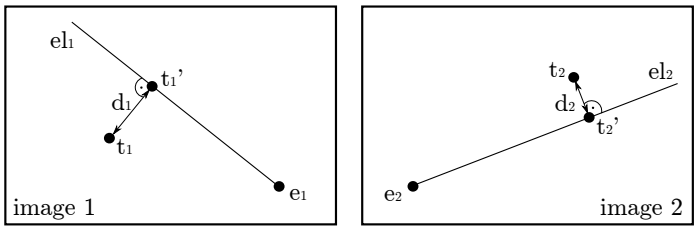


Figure 2.9: The demonstration of reprojection error in a stereo setup where e_i is the epipole, el_i is current estimation of epipolar line and d_i is the Euclidean distance between the initial noisy measurement t_i and the reprojection of the target estimate t_i' in the camera i.

Even though both DLT and reprojection error minimization could be extended so as to support multiple-view setup, neither of these approaches is suitable for the OLS since they do not consider any a priori known information about the reliability of the back-projected line in each camera. It has been shown that in the OLS the precision of target location estimation strongly depends on the mutual position of the target and the baseline of the given camera pair (see Chapter 3). Furthermore, it is possible to obtain the belief from individual visual trackers (i.e. the confidence that the object is tracked correctly). In order to exploit these prior information a specific location estimation method was proposed for the OLS (see Section 4.3).

2.3 3D Environment Reconstruction

One of the most challenging and still not fully solved problems of visual tracking algorithms is the occlusion [59, 28], i.e. the case where the tracked object becomes partially or fully overlapped by another object. Even though both visual trackers the OLS is based on (see Sections 2.1.3 and 2.1.4) are robust against partial occlusion and to the limited extent even against full occlusion, they detect the occlusion using the visual clues which might not be reliable.

However, if the 3D model of the surrounding scene is known and motion model in 3-space of the tracked target is estimated, it is possible to predict both start and end of full occlusion occurrence in advance and consequently to adjust the tracking algorithm temporarily so that it would not fail. If the 3D model of the environment is not known beforehand it can be reconstructed using mere visual information obtained from cameras.

Therefore, this section introduces the problematics of 3D reconstruction using visual cues in multi-camera system, namely the basics of epipolar geometry, the notion of bundle adjustment and a popular tool for performing both sparse and dense reconstruction in

multiple-camera setup – VisualSFM. Proposed algorithm of occlusion prediction based on the knowledge of sparse 3D model of the environment is presented in Section 4.4.

2.3.1 Reconstruction Pipeline

Assuming the most general scenario where multiple images of the scene taken from multiple uncalibrated cameras are available, a 3D reconstruction pipeline based on the *iterative bundle adjustment* can be utilized [45] (see Figure 2.10).

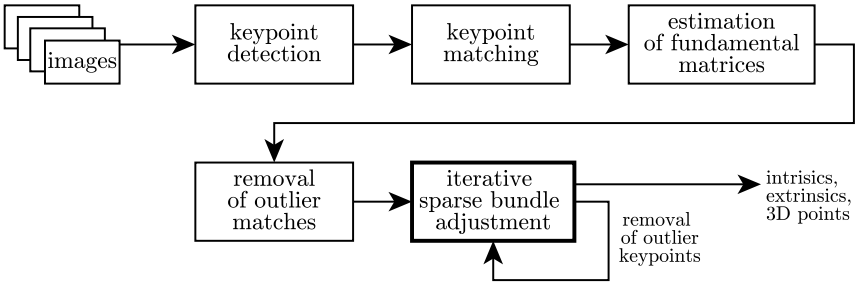


Figure 2.10: The 3D reconstruction pipeline taking multiple images obtained from uncalibrated cameras and producing estimated camera parameters and locations of matched points in 3-space.

The algorithm takes arbitrary number of images on the input and performs *SIFT keypoint detection* as the first step. Next, individual keypoints are matched across all images creating the *tracks*. A *fundamental matrix* $\mathbf{F}$ is estimated for each pair of images (see Section 2.3.2) and those matches which are outliers with regards to $\mathbf{F}$ are removed. Finally, the *iterative sparse bundle adjustment* (see Section 2.3.3) which produces the estimates of camera parameters (both intrinsics and extrinsics) and 3D locations of matched points is run.

In case of the OLS both extrinsics and intrinsics are known. However, they are correct only up to a systematic and random error caused by imprecise stationing and rectification (see Chapter 3) and a noise which the P&T unit orientation measurements are subject to. Therefore, in order to achieve a 3D reconstruction of the surrounding environment it is still reasonable to employ the bundle adjustment technique. In order to exploit the a priori known information the bundle adjustment could be for instance initialized with known camera calibration and pose parameters so as to make the optimization algorithm more likely to find the global optimum.

2.3.2 Epipolar Geometry

Epipolar geometry represents the relation between two projective pinhole cameras observing the same point (points) in 3-space [10] (see Figure 2.11). The line between both camera centers $\vec{C_i}$ is called the *baseline* and it delimits the *epipole* $\vec{e_i}$ in each projective plane Im_i. The projection of target $\vec{T}$ to both image planes defines image points $\vec{t_i}$ which in return back-project to lines in 3-space. The line el_i lying in image plane Im_i connecting $\vec{e_i}$ and $\vec{t_i}$ is called the *epipolar line*.

What the epipolar geometry relation says is that the observed target $\vec{T}$ must lie only in the plane Ep defined by the baseline and both back-projected lines (from the camera

center $\vec{C_i}$ through the image point $\vec{t_i}$). Alternatively, the epipolar line el_i is the projection of back-projected line $\vec{C_j}\vec{t_j}$ to the projection plane Im_i.

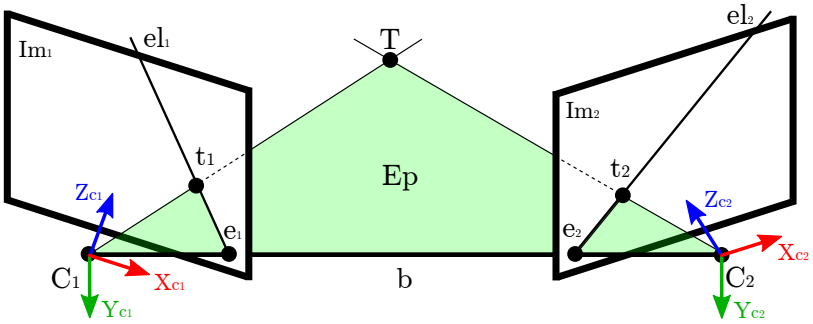


Figure 2.11: The epipolar geometry.

Ultimately, the *epipolar constraint* is defined as following: *Each image point* $\vec{t_i}$ *of a space point* $\vec{T}$ *lies in the image plane only on the corresponding epipolar line.* This can be stated numerically using the *fundamental matrix* $\mathbf{F}$:

$$\vec{t_1}^T \mathbf{F} \vec{t_2} = 0 \tag{2.18}$$

Fundamental matrix $\mathbf{F}$ can be estimated from mere image correspondences for instance using normalized 8-point algorithm within RANSAC framework [19, 10].

2.3.3 Bundle Adjustment

Bundle adjustment is the approach which aims to simultaneously refine the parameters of all involved cameras (intrinsics and extrinsics) and to minimize the reprojection error between the initially measured image point and reprojected target. The non-linear least squares error function E can be defined as [16]:

$$E = \frac{1}{mn} \sum_{i,j} [(x_{ij} - \frac{\vec{p_{i1}}\vec{T_j}}{\vec{p_{i3}}\vec{T_j}})^2 + (y_{ij} - \frac{\vec{p_{i2}}\vec{T_j}}{\vec{p_{i3}}\vec{T_j}})^2], \tag{2.19}$$

where m is the number of cameras, n is the number of target points in 3-space, (x_{ij}, y_{ij}) is the initially measured location of the projection of target T_j to the projective plane of camera i and $\vec{p_{ir}}$ is the rth row of projection matrix P_i. As for the minimization, *Levenberg-Marquardt* algorithm is mostly used.

In case of reconstruction approach proposed by Snavely, Seitz and Szeliski [45] (see Section 2.3.1) the iterative version of bundle adjustment is used. In this case the most suitable image pair which has enough matches and large baseline is selected and the camera parameters as well as the 3D locations of matched points are estimated. In each next iteration a new camera is added to the optimization algorithm.

2.3.4 VisualSFM

VisualSFM[5] is a well established application for end-to-end scene reconstruction using multiple cameras which follows the standard pipeline described in Section 2.10 and adds another

[5]Official website of VisualSFM: http://ccwu.me/vsfm/index.html

method for dense reconstruction. For sparse reconstruction the application uses parallel implementation of bundle adjustment [55] and for dense reconstruction the Patch-based Multi-view Stereo Software (PMVS) approach is utilized [17]. The demonstration of dense reconstruction is depicted in Figure 2.12. Upon successful reconstruction the VisualSFM enables the 3D point cloud to be exported in standard PLY format which could be imported to the OLS for instance with the use of Point Cloud Library (see Section 4.4).

Figure 2.12: The 3D reconstruction performed by VisualSFM from only two photographs taken by the same camera in district Štýřice (Brno, Czech Republic). Note that even mere two views suffice to reconstruct the church tower, however, the output data include a lot of noise.

Chapter 3

Precision of the Localization in Multi-camera System

The whole chapter is devoted to exploring the problematics of error analysis which must be performed prior to the system design. The objective is to find prospectively most severe sources of error already in the early stages of the system development and to take appropriate measures in order to eliminate them. First, the proposed topology of the OLS design so as to maximize the system precision is proposed and described in Section 3.1. Section 3.2 then discusses the main sources of error in the system, categorizes them and provides a detailed analysis of impact each error poses on the OLS. Finally, Sections 3.3 and 3.4 explain the processes of *stationing* and *rectification* which aim on eliminating certain types of errors.

3.1 System Topology

The main building block of the OLS is a *camera unit* (CU) – an independent collection of hardware modules including positionable camera and various sensors used for estimating the geographical coordinates of the CU itself (see Figure 3.1). Detailed description of the CU will be given in Section 4.1, however, for the purpose of precision analysis it suffice to remark that a camera which each CU is equipped with is free to rotate around azimuthal and elevation axes and it is used for the visual tracking of a target.

As was explained in Section 2.2.3 a system consisting of at least two cameras is capable of estimating the position of the target in 3-space using triangulation. The main strength of the OLS is the fact it is designed to work with an arbitrary number of CUs, therefore the estimated position of the target can be refined by incorporating multiple hypotheses. As will be explained in Section 3.2 the geometric limitations of the stereo-view systems significantly affect the localization precision. Therefore, the CUs should be positioned so that their positions projected to the horizontal plane would form approximately regular polygon. A sample setup of the OLS is depicted in Figure 3.1.

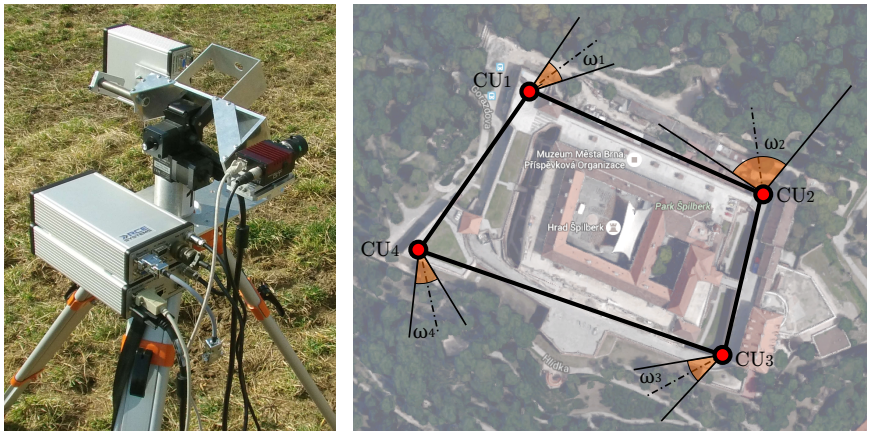


Figure 3.1: The photograph of the CU (left) and a use case scenario (right) showing four CUs (red dots) positioned so as to protect a real world area (castle Špilberk in Brno, Czech Republic). Note that in general each camera might have different FOV depicted as an angle ω.

3.2 Precision Analysis

The precision of the system can be defined in the means of the frame-by-frame Euclidean distance E between the estimated location $\vec{T_{est}}$ and the real (ground truth) location $\vec{T_{GT}}$ of the tracked target:

$$E = \|\vec{T_{GT}} - \vec{T_{est}}\|. \tag{3.1}$$

Since the precision can be affected severely by multiple independent factors it is essential to perform the error analysis in order to discover and prospectively alleviate the most prominent contributors of the overall error.

3.2.1 Sources of Error

The precision of estimating the target position is subject to various types of errors which have different impact on the overall deflection. Basically, there two main categories of errors can be distinguished, the *systematic error* and the *random error* [48].

Systematic error The main property of systematic error is the fact that it cannot be revealed by repeating the measurement. I.e. this error is intrinsicly integrated in the system itself and it always distorts the measurements the same way. In case of the OLS the sources of the systematic error are the discrepancy between the physical CU and its model (imprecise mechanical construction) and imprecise stationing (i.e. finding the global orientation and geographical coordinates of each CU). In order to eliminate the systematic error in the OLS the process of stationing (see Section 3.3) and rectification (see Section 3.4) were designed.

Random error This type of error can be usually described by a probability distribution i.e. it behaves in a random manner. As for the OLS the main source of random error is the visual tracker which typically outputs the measurements oscillating around the correct

target position. This phenomenon is caused jointly by non-maturity of the tracking algorithms and by the clutter present in the environment such as the atmospheric turbulences or refractive index fluctuations. Another source of the random error are the measurements of the current azimuth and elevation of the P&T unit and time synchronization among the cameras. The random error can be alleviated by making the visual tracker more robust (see Section 4.2).

Whether being systematic or random, in the OLS the error eventually reflects in undesirable rotation and/or translation of the CU model with regards to its real physical counterpart and consequently, when computing the triangulation the rays are back-projected in the wrong direction and/or from the wrong position. Note that given the nature of triangulation (see Section 4.3.1) the OLS is extremely sensitive to error angles (caused e.g. by wrong estimation of heading). Contrarily, the error translations (caused e.g. by faulty GPS measurement) are not of major concern (see Figure 3.2).

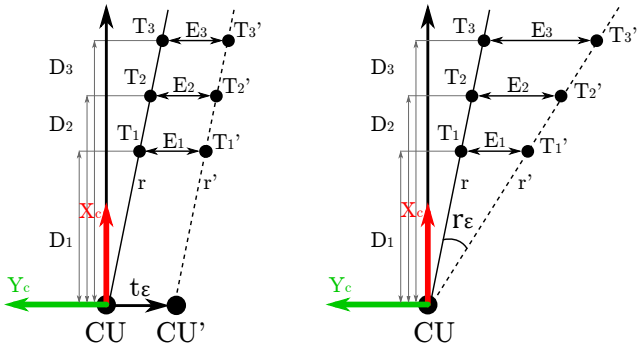


Figure 3.2: The impact of undesirable error translation t_ϵ (left) and error rotation angle r_ϵ (right) of the CU model which causes the back-projection of incorrect ray r' instead of correct r. Note that in case of t_ϵ the localization error E_i is invariant to the distance D_i of target T_i. Contrarily, in case of r_ϵ the more distant the target T_i is the higher value the localization error E_i reaches.

3.2.2 Error in Two-View System

Stereoscopic systems are affected by a phenomenon of diminishing accuracy of depth measurement with increasing distance of the target from the cameras [10]. The depth measurement resolution for canonical stereo setup is:

$$R = \frac{rd^2}{fB - rD}, \tag{3.2}$$

where f is the focal length, B is the baseline length, r is the horizontal size of one pixel and D is the target distance. By substituting r by pr, where p is the random error represented by integer number of pixels we obtain the *position estimation error function*

$$E = \frac{prD^2}{fB - prD}. \tag{3.3}$$

The OLS does not conform to the canonical stereo setup (all cameras can rotate freely) so the dependence of the error on the target distance is no longer quadratic (considering

the setup of two cameras where only one of them makes error):

$$E = B\tan(\arctan(\frac{D}{B}) + \arctan(\frac{r}{f})) - D. \tag{3.4}$$

The cameras setup as well as the error shown as the function of the baseline length and target distance is depicted in Figure 3.3. Note that the higher the distance of the target and at the same time the lower the baseline length the higher the precision error.

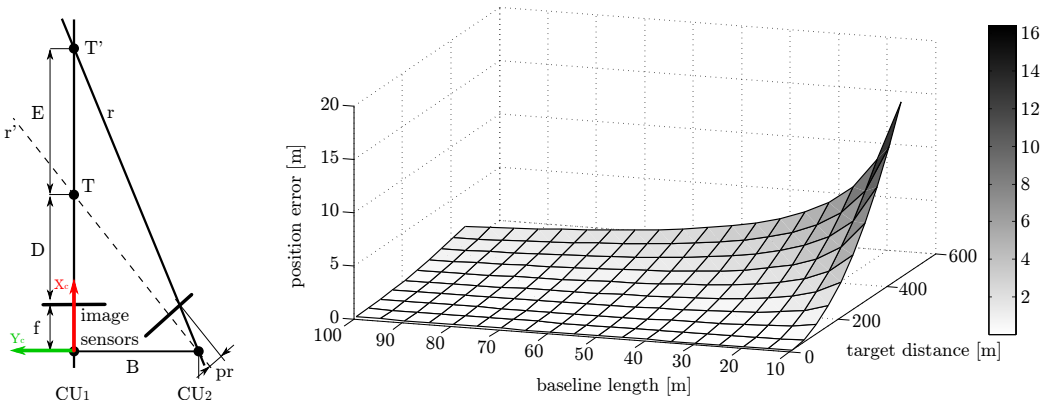


Figure 3.3: Left figure depicts the setup of two cameras CU_1 and CU_2 where only CU_2 makes an error worth p pixels. T represents the ground truth position of the target whereas T' is the wrongly estimated position. Right figure shows the position estimation error as the function of baseline length and the target distance (given the random error $p = 10\ px$ and following constants corresponding to the hardware used in the OLS: $r = 3.75e^{-6}\ m$, $f = 50e^{-3}\ m$).

3.2.3 Error in Multiple-View System

A more realistic scenario where each camera makes a random error p is depicted in Figure 3.5. A significant advantage of using multiple cameras is demonstrated — the geometrical limitations of the two-camera setup make it impossible to precisely evaluate the position of the target placed close to the line collinear with the baseline (see Figure 3.4). In the multi-camera setup, on the other hand, the subset of two cameras forming the baseline B_i is used for each position of the target, following the rule:

$$i = \underset{i}{\text{argmax}}\ \vec{t_i}\vec{n_i}, \tag{3.5}$$

where $\vec{t_i}$ is the direction of the line segment linking the center of the baseline CU_c and the target and n_i is the normal vector of the baseline B_i. Put in other way, the baseline B_i yielding the highest absolute value of angle γ is chosen to compute the position of the target.

The Figure 3.5 depicts the position estimation error as the function of the target's position with regards to the CUs for both two-camera and three-camera setup. In this scenario only horizontal position of the target is considered (i.e. its height is disregarded) and it is assumed that each tracker makes the random error worth $p = 10\ px$. For each position of the target the worst possible location estimation is considered (see Figure 3.4).

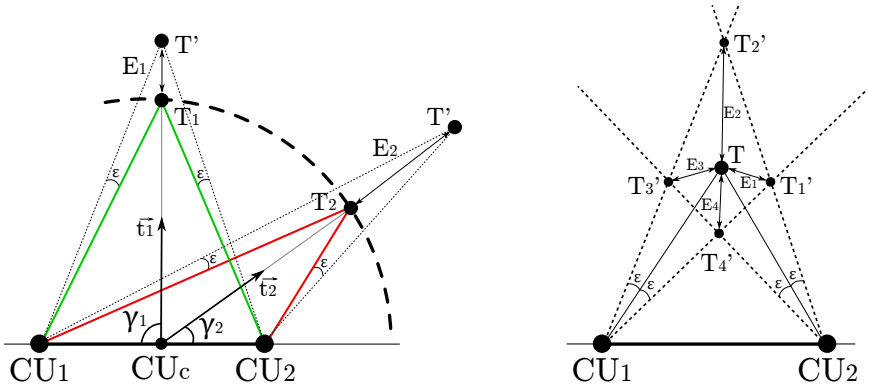


Figure 3.4: Left figure demonstrates the basic geometrical limitation of the two-view system which causes the position estimation error E to increase rapidly as the angle γ between the baseline and the target decrease up to the point where γ is zero and E becomes infinitely large. Note that both CUs make the same random error depicted as error angle ϵ. Right figure shows four possible intersections of the rays back-projected from both CUs making random error given as the angle ϵ. When analyzing the precision the worst error E_i is always chosen.

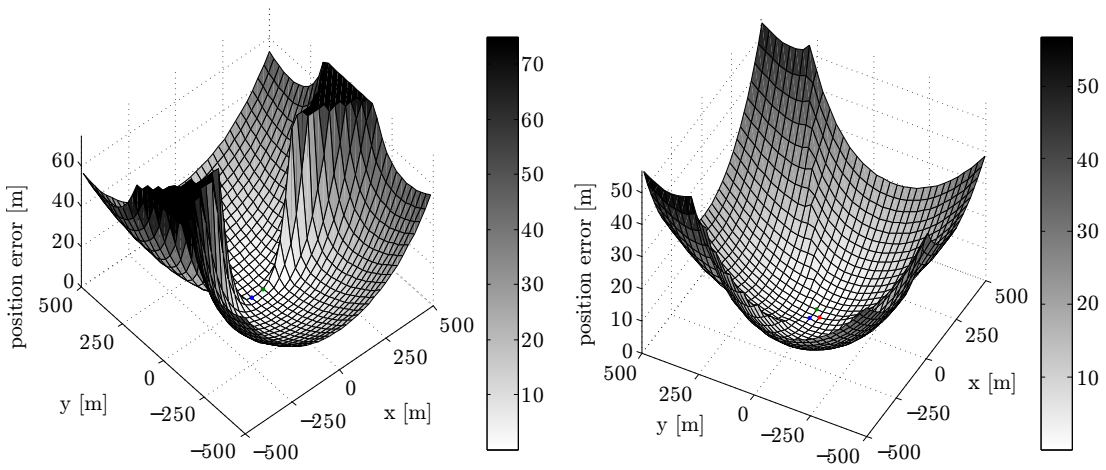


Figure 3.5: The position estimation error as the function of the horizontal position of the target. Two-camera (left) and three-camera (right) setup with $B = 20\ m$ are confronted where utilization of more cameras always yields lower errors. The first two cameras are placed on the X axis with the coordinate frame center in the middle of their baseline. The third camera is placed on the Y axis, so that all the cameras form a regular triangle.

3.3 Stationing

The main purpose of the *stationing* process is to find the positions and orientations of all CUs with regards to the global (world) coordinate system. The stationing consists of four subtasks: finding the global position (using DGPS sensor), ensuring horizontality, finding absolute heading[1] and finding relative heading. Great care should be taken when performing stationing since imprecise estimates of the position and/orientation are the main source of

[1] *Heading* is the term used to describe the angle between the torso of the human body and the geographical north [20].

the systematic error which significantly impacts the overall localization precision of the system.

3.3.1 Horizontality

Since the CU is expected to be placed in an unknown outdoor terrain it will never stand on an ideally horizontal surface. Therefore, it is necessary to either ensure that the unevenness of the surface is compensated by the suitable setting of the CU's stand or both the side tilt and front tilt angles of the stand must be estimated and integrated to the CU model. For these purposes the inclinometer attached to the base plane of the camera unit (see Section 4.1) is used.

3.3.2 Absolute Heading

Though it is a common practice to estimate the heading using a magnetometer, this device is unsuitable for the OLS since the accuracy of the current professional class magnetometers starts at ca 10 *mrad* [22] which is insufficient.

In order to find the orientation of each camera unit placed in the outdoor environment, distinctive landmarks with known geographical positions are used. For each such landmark the P&T unit is rotated so that the optical axis of the camera would intersect that landmark and the azimuth value is registered. Using triangulation the geographical position of the camera unit is derived.

A 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.

3.3.3 Relative Heading

To further reduce the impact of both random error produced by the GPS sensor and the systematic error given by the imprecision of absolute heading estimation it is reasonable to find the relative heading of each camera unit with regards to the rest of the camera units. Furthermore, relative heading estimation is inevitable in case the absolute heading cannot be measured at all. In case of the OLS, the absolute heading was not measured during testing (see Section 6.3) and the system relied only on the relative hading estimation.

The process of relative heading estimation is run separately for each pair of CUs. First, the azimuth and elevation of both P&T units is set so that the optical axis of the camera would intersect the expected location of the LED target of the other CU. Then the position is refined so that the camera would aim directly at the center of the LED target (see Figure 3.6). Current azimuth and elevation values of both CUs are saved and the whole model of the system is updated accordingly.

3.4 Rectification

The *rectification* process serves the purpose of reducing the systematic error caused by the imprecise attachment of the camera to the P&T unit. The objective is to fix the camera in such a position that the image sensor becomes parallel with both azimuthal and elevation axis. At the same time the rows of the image sensor must remain parallel with the elevation axis (i.e. the camera is not rotated around the optical axis). If it is not possible to fix the camera precisely in required position (mechanical limitations of the camera mount) the

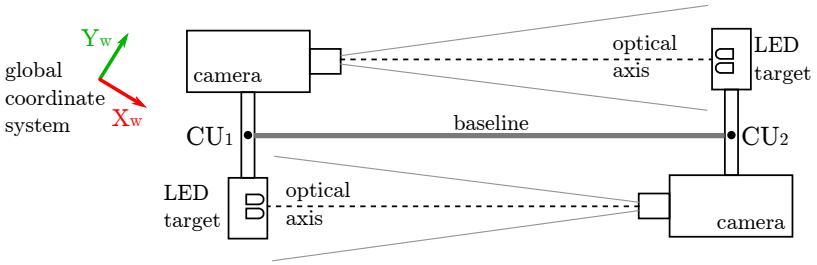


Figure 3.6: Schema of the stationing process where two CUs attempt to align the optical axes of their cameras.

error angles must be measured and integrated to the CU model. The rectification process consists of three parts: *eliminating rotation along the optical axis*, *measuring rotation along the azimuthal axis* and *finding the default elevation angle*.

3.4.1 Eliminating Rotation Along the Optical Axis

The camera is attached to a custom made metal mount. The mount itself is then attached to the P&T unit using two opposing round tenons enabling for the rotation around the axis parallel with the optical axis of the camera (see Figure 3.7).

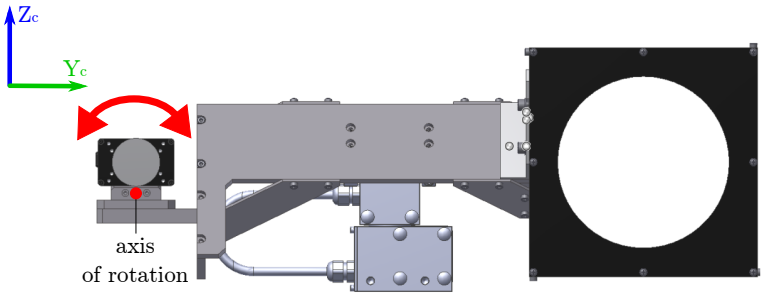


Figure 3.7: Front view of the top part of the CU. The red arrow denotes the possible rotation of the camera along the axis parallel to the optical axis. Image courtesy of company Oprox, a.s.

In this part the rectification target with three parallel horizontal black lines is used. As the first step a surveying automatic level is used to rotate the target so that the black lines become horizontal. The camera is then positioned so as to aim approximately at the center of the target. The camera image stream is blended with the same stream mirrored across the vertical axis. Finally, the objective is to manually rotate the camera so that the black lines in blended image stream appear visually aligned (see Figure 3.8). Once set, the mount with the camera is fixed to the manipulator using two set screws.

3.4.2 Measuring Rotation Along the Azimuthal Axis

The mount can still rotate around the axis parallel with the azimuthal axis (see Figure 3.9). It is necessary to ensure that the optical axis of the camera is perpendicular to the elevation 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

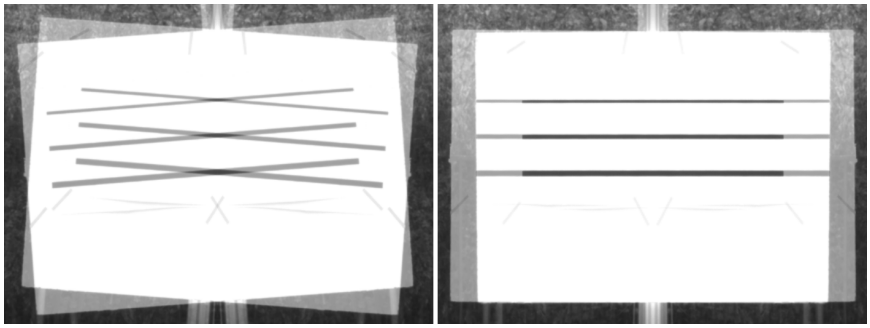


Figure 3.8: A blended image stream from the camera before (left) and after (right) rotating the camera along the optical axis to the correct position.

crosses equals to the distance between the azimuthal and optical axis (which is known from the engineering design, see Figure 3.11).

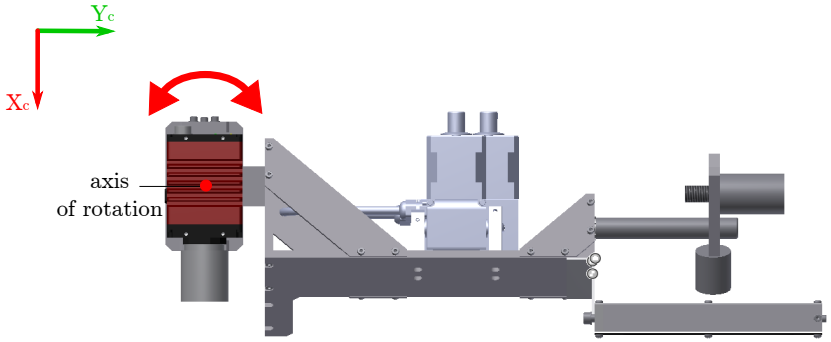


Figure 3.9: Top view of the top part of the CU. The red arrow shows the possible rotation of the camera along the axis parallel to the azimuthal axis. Image courtesy of company Oprox, a.s.

A military monocular telescope (see Figure 3.10) 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 intersects the right cross and then it 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 d_h expressed in pixels is transformed to the default rotation angle β (see Figure 3.12) of the joint CAMERA in the CU model:

$$\beta = \arctan \frac{d_h cs}{f}, \tag{3.6}$$

where f is the focal length and cs is the physical size of one pixel.

3.4.3 Finding the Default Elevation Angle

Considering the limited elevation range of the P&T unit `Flir PTU D46-70` (see Section 4.1) the camera must be mounted pre-rotated around the elevation axis by approximately $-60°$. However, after fixing the camera it is necessary to find this default angle precisely.

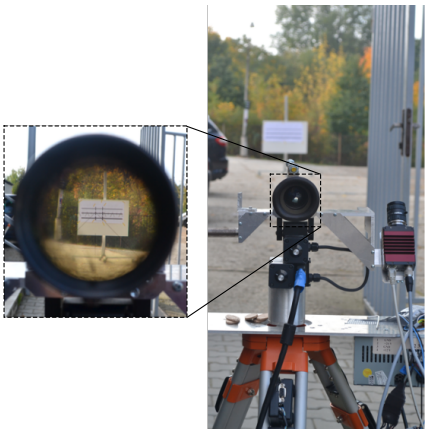


Figure 3.10: A telescope mounted on top of the P&T unit. A person looking through a telescope sees the black crosshair.

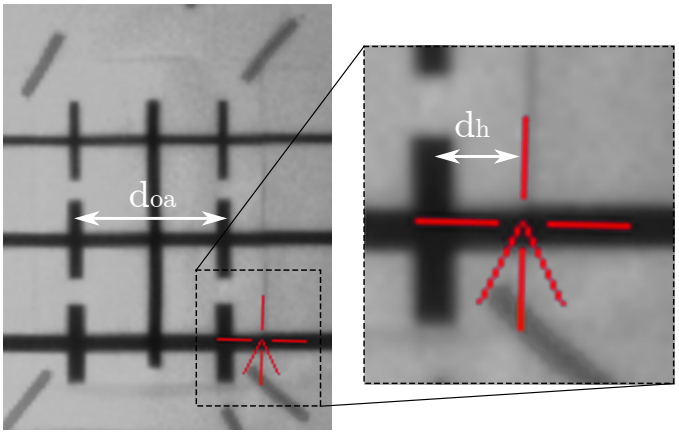


Figure 3.11: Rectification target with the pairs of black crosses. The two crosses in a pair are d_{ao} m apart. A digital crosshair is displayed in order to find the horizontal offset d_h.

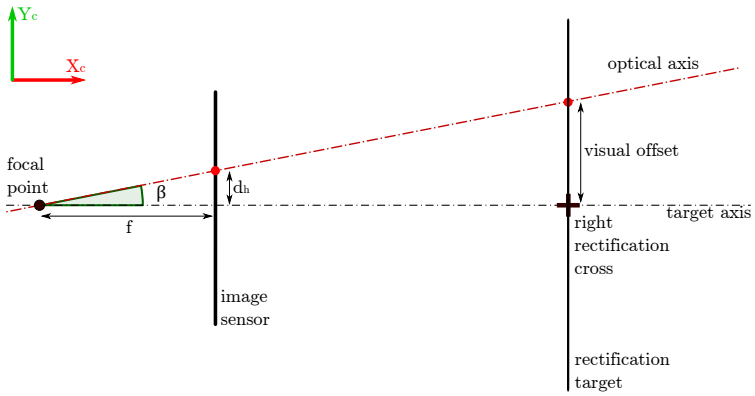


Figure 3.12: The top view schema of a rectification target being projected to the image sensor of the camera.

For this purpose a pair of rectification targets which consist of horizontal black and white lines representing the marks of a ruler are used. 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. The operator manually adjusts the elevation of the manipulator until the digital crosshair intersects the same mark on both targets where the two marks form a straight line (see figure 3.13). Once such an elevation is found the angle is recorded and integrated to the model of the camera unit as an angle of rotation around the Y-axis of the joint CAMERA (see Section 4.1).

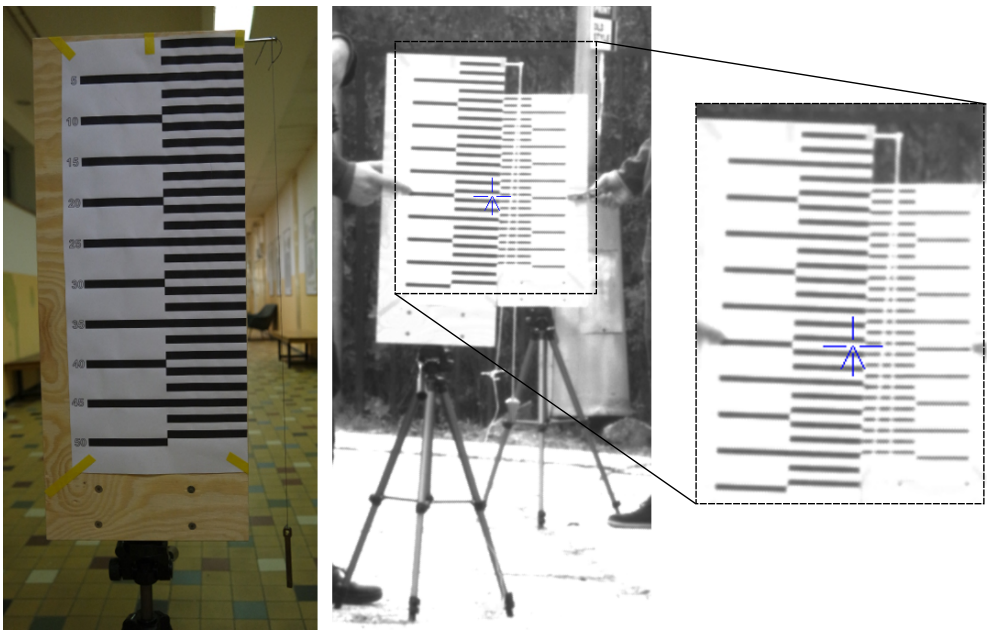


Figure 3.13: Front target of a pair of the rectification targets used to find a default elevation angle (left). A screenshot from the image stream of the camera with the crosshair focused on a row where the marks of the rulers align (right).

Chapter 4

Design of the Optical Localization System

This chapter describes the overall design of the OLS from both hardware and software perspective. First, the main physical component of the system – the camera unit – is presented. The hardware components which the camera unit consists of are described and the suitable model based on the kinematic chain is proposed in Section 4.1. Next Section 4.2 delves into the specific requirements and proposed solutions for the visual tracking. The Section 4.3 explains the proposed approach to localizing the target in multi-view scenario given the noisy measurements and Section 4.4 describes the proposed approach to utilize the 3D model of the environment in order to predict the occlusion. Finally, Section 4.5 presents both hardware and software architecture of the system.

4.1 Camera Unit

As stated in Section 3.1 the main component of the OLS is a camera unit (CU). Basically, the CU consists of hardware devices necessary for capturing the images, for manipulating the pose of the camera, for estimating absolute geographical position and orientation of the station as well as relative position and orientation with regards to the rest of the stations and for running the OLS software.

There are two type of CUs. The *overview unit* is designed to be controlled manually by the human operator and it is equipped with the zooming lens that allows achieving both a wider scanning range and a more detailed view of the farther objects. The *tracking unit* consists of the fixed lens and takes part in the autonomous tracking of the moving objects and continuous reporting of the estimated directions towards the target.

The hardware components used for tracking stations are described in greater detail in Section 4.1.1. In order to triangulate the target, the 3D location of the camera as well as the direction of the optical axis must be known for each captured frame, thus a suitable model corresponding to the real hardware mus be designed. Proposed model based on the kinematic chain is introduced in Section 4.1.2.

4.1.1 Devices and Components

The CU (see Figure 4.1) consists of a surveying tripod providing a solid base on which a manipulator (P&T unit[1]), a camera and all of the devices used for stationing (LED target, GPS sensor and inclinometer) are mounted. Each CU is equipped with its own desktop computer for processing the image data and calculating the 3D position estimates.

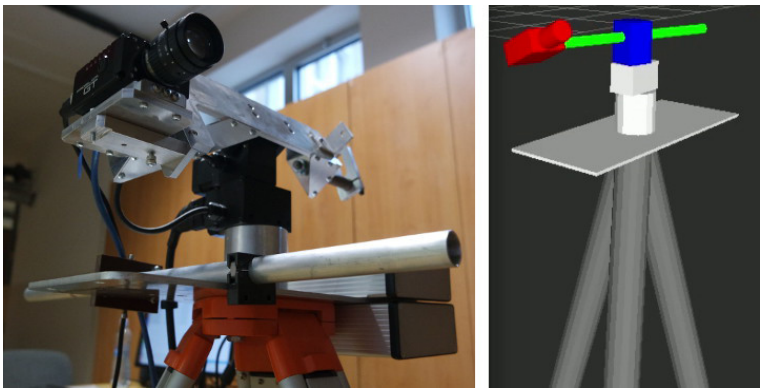


Figure 4.1: A photograph (left) of the upper part of the camera unit consisting of a manipulator Flir PTU-D46-70 with the aluminum mount carrying a camera Prosilica GT 1290C and a corresponding 3D model (right) created for *rviz* and *Gazebo* simulator (see Section 5.3.1).

Manipulator Flir PTU-D46-70 A manipulator PTU D46-70[2] produced by manufacturer Flir is used (see Figure 4.2). As compared to the other professional manipulators this is a lower middle class device consisting of two stepper motors (pan and tilt axes). The device is capable of maximal angular speed of $60°/s$ with the resolution of $0.003°$ while the payload must not exceed $4.08\ kg$ [14]. 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 Prosilica GT 1290C[3] is an industrial camera manufactured by the company Allied Vision (see Figure 4.2). It is an RGB camera equipped with CCD sensor (type 1/3") with the resolution of 1280×960 px and support for 33.3 FPS and it communicates through gigabit Ethernet [52]. What is important, the camera natively supports the Precision Time Protocol (PTP) for precise time synchronization which is a crucial feature in each application relying on stereo vision and it is capable of time synchronization among devices within the range of nanoseconds [24]. The manufacturer claims that Prosilica GT 1290C achieves the synchronization precision of 1 μs.

[1] From English Pan and Tilt.

[2] Website of the product Flir PTU-D46-70: http://www.flir.com/mcs/view/?id=53712

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

Lens Computar M5018-MP2 Each camera mounted on a tracking unit is equipped with a fixed-focus lens Computar M5018-MP2[4] (see Figure 4.2). The focal length 50 mm was chosen as a most suitable trade-off between the wide field of view and capability of imaging distant targets. Given the camera sensor type 1/3" the effective horizontal field of view fov_h is approximately 5.5°.

Figure 4.2: Product pictures of manipulator Flir PTU D46-70 (left), camera Prosilica GT 1290C (middle) and lens Computar M5018-MP2 (right).

4.1.2 Model

The model of the camera unit is based on the kinematic chain consisting of six joints and five links corresponding to the distance between separate parts of the surveying tripod and separate parts of the manipulator (see Figure 4.4). The starting joint GROUND itself is dependent on the reference location (let us call it ORIGIN) which represents the origin of the global coordinate frame. The transformation between ORIGIN and GROUND reflects the position and orientation of the given manipulator within the environment (which is estimated during the stationing process, see Section 3.3).

The kinematic chain is designed as the composition of transformation matrices where a single joint can be located as a solution to the *forward kinematics problem*, i.e. 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:

$$M_{cam} = M_{ele} T_{cam} R_{Z_{cam}} R_{X_{cam}} R_{Y_{cam}}, \tag{4.1}$$

where M_{ele} is the transformation matrix of the joint ELE which the joint CAMERA is dependent on, and T_{cam} and $R_{AXIS_{cam}}$ are transformation and rotation matrices describing transformation from the joint ELE to the joint CAMERA.

Note that from the implementation point of view the OLS is built on the ROS framework (see Chapter 5) which presents certain conventions, most importantly the orientation of the coordinate frame which is used throughout the document (see Figure 4.3).

4.2 Visual Tracking

The visual tracking subsystem of the OLS is based on the TLD tracker (2.1.3) and the BGFG tracker (2.1.4). As explained in Section 3 the system is subject to multiple different sources of error which influence the precision of localization. As far as the visual tracking is concerned, the random error caused mostly by the common difficulties such as the background clutter, varying illumination, target appearance change etc. (see Section 2.1.1)

[4]Website of the product Computar M5018-MP2: http://computar.com/product/556/M5018-MP2

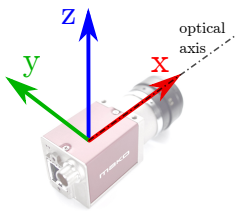


Figure 4.3: Frame orientation convention which is used throughout the work — a right handed coordinate system with X axis aiming forward, Y axis aiming left and Z axis aiming up.

is of the most significance. Furthermore, in the worst case scenario the tracker might fail completely and never recover unless the human operator interferes.

Therefore, the BGFG tracker was adjusted so as to assign a belief to each measurement which then enables the localization subsystem to combine the measurements in the weighted manner (see Section 4.2.1). Furthermore, the bootstrap particle filter (BPF) algorithm of the BGFG tracker was adjusted so as to alleviate the failures caused by the harsh camera motion (see Section 4.2.2). Finally, the most suitable strategy for regulating the motion of the manipulator based on predicting the 2D motion of the target is proposed (see Section 4.2.3).

4.2.1 Measurement Belief

Since the OLS is designed as multi-camera system, the random error or failure of a single tracker might be compensated by the rest of the units; consequently the total impact on the localization precision is alleviated. In order to denote the certainty of the measurements coming from individual trackers the notion of *belief* is introduced for the BGFG tracker (the operation of the BGFG tracker is described in Section 2.1.4).

The *belief* is calculated for each frame and it is based on the distance function (2.4) which BGFG tracker uses to evaluate the candidate states represented by the particles. Basically, the distance function corresponds to the visual similarity of two image patches – the target template and current candidate weighted by its foreground mask. Therefore, the particle weight is normalized by the maximum possible similarity so as to obtain a percent value *belief* representing the similarity (i.e. $belief \in [0, 1]$):

$$belief = \frac{\sum_{(x,y)\in I} e^{min(M_t^{(x,y)}, M_c^{(x,y)})}(1 - |I_t^{(x,y)} - I_c^{(x,y)}|)^2}{\sum_{(x,y)\in I} e^{min(M_t^{(x,y)}, M_c^{(x,y)})}}, \tag{4.2}$$

Therefore, the noisy measurements or a complete failure of the tracker can be detected and propagated to the rest of the system (see Figure 4.5).

4.2.2 Adjusting Prediction Step Using Frame Differencing

One of the most common reasons for failure of the BGFG tracker is sudden and harsh motion of the camera mounted to the P&T unit. In such a case the tracked target represented as a 2D location on the image plane of the camera abruptly changes its 2D position. The BGFG tracker continuously computes the homography [21] in order to detect the camera motion and incorporates it to the motion model of the target. However, the homography

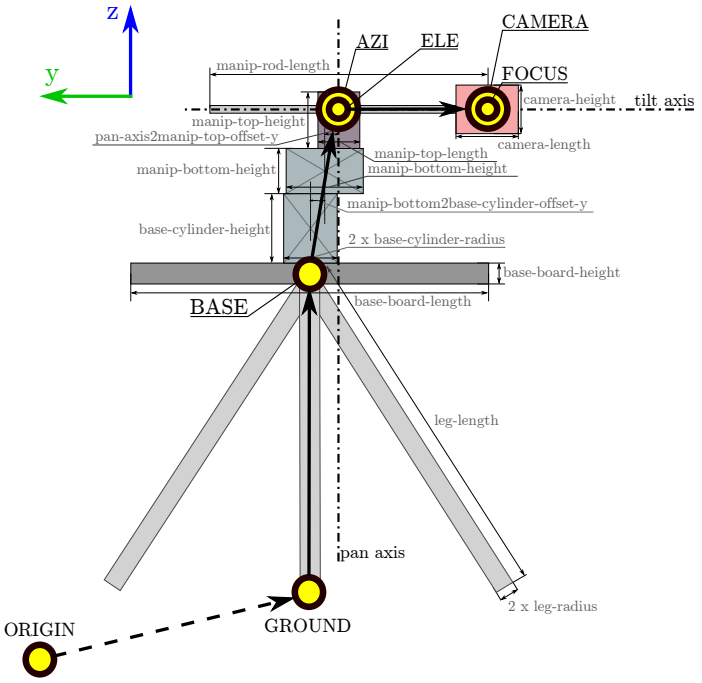


Figure 4.4: Schematic view of the kinematic chain of a camera unit with the joints depicted by the yellow circles. The sizes of all components necessary to specify the translation matrices between consecutive joints are shown as well. Note that this is the rear view, i.e. the camera is seen from behind. Thus the joints CAMERA and FOCUS overlap as they both lie on the optical axis (the joint FOCUS has a lower value of the Z coordinate).

computation often fails mostly when the background is too uniform (e.g. a monotone field, a sky, etc.).

To deal with such cases, the BGFG tracker was adjusted so that in the *prediction* step of the BPF a subset of particles would be forced to take the image positions yielding the highest response of the frame differencing computed for subsequent frames. Such locations are expected to contain the moving target of interest. As compared to the original *prediction* step (2.2), the parameters $pos^i_{dim_n}$ representing 2D position of the particle i in the dimension dim (i.e. x or y) in time n takes the following form:

$$pos^i_{dim_{n+1}} = \begin{cases} fd_{dim_n} + x, \ x \sim \mathcal{N}(\mu_{pos}, \sigma_{pos}), & \text{if } FD_{max_n} > T \wedge i < M \\ pos^i_n + x, \ x \sim \mathcal{N}(\mu_{pos}, \sigma_{pos}), & \text{otherwise,} \end{cases} \tag{4.3}$$

where fd_{dim_n} is the coordinate of maximal intensity F_{max_n} of the frame differencing in dimension dim in time n, T is the threshold which alleviates the ubiquitous non-zero frame differencing response caused by noise and M is the number of particles which are allowed to change their position. Note that the particles are sorted according to their current weights in the ascending order, i.e. the index i and constant M denote whether the particle is allowed to change its position. The impact of adjusted *prediction* step is depicted in Figure 4.6.

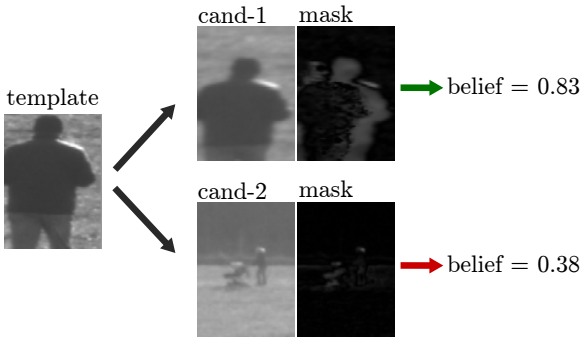


Figure 4.5: The demonstration of the belief computed for two different candidates (top, bottom) given the input target template (left).

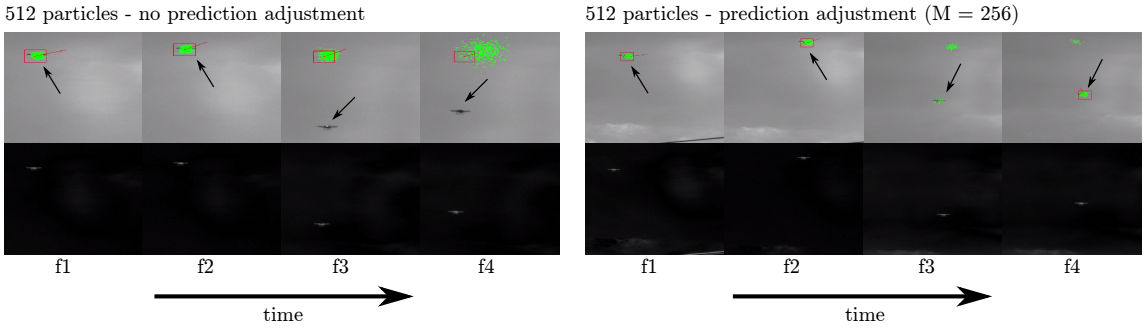


Figure 4.6: Four frames from two tracking sequences of the BGFG tracker are shown where the original (left) and adjusted (right) *prediction step* implementation is used. The distribution of the particles (green dots) is manifested in the top row, the response of the frame differencing is shown in the bottom row. In both cases the tilt of the camera changes abruptly between frames $f2$ and $f3$ which causes the target (denoted by black arrow) to appear in the significantly different position. In the first scenario the particles fail to follow the target whereas in second scenario the subset of particles are forced to the position of the highest frame differencing response and consequently the target is found again.

4.2.3 2D Motion Prediction and Regulation

In order not to loose the moving target from the field of view the manipulator has to rotate the camera around both azimuthal and elevation axis appropriately. A most straightforward approach would be to simply measure the angular distance between the current measurement (i.e. the projection of the target to the image plane) and the center of the image and instruct the manipulator to move so as to minimize this distance as fast as possible. However, this approach is not sufficient.

Contrarily, there are two main reasons why it is necessary to predict the future 2D position of the target. First, it is desirable to keep the target as close to the center of the image as possible so that there would be enough time to react in case of the sudden and rapid movement of the target. Second, the measurements are delayed (due to the tracking algorithm computation and communication delay) and the P&T unit manifests considerable latency.

Given the time stamped (t) states of the manipulator $\vec{m} = (\phi, \psi)$ where ϕ and ψ are the angular positions (given in *radians*) and tracker measurements $\vec{p} = (x, y)$, where x and y are the positions of the target (given in *pixels*), the 2D angular speed of the target $\vec{\omega_t}$

can be estimated as the moving average over N last measurements i as well as the angular difference $\vec{d} = (\phi_{diff}, \psi_{diff})$ between current and future 2D position in T seconds:

$$\vec{\omega_t} = \frac{\sum_{i=0}^{N-2} \alpha(\vec{p_{i+1}} - \vec{p_i}) + (\vec{m_{i+1}} - \vec{m_i})}{\sum_{i=0}^{N-2} t_{i+1} - t_i}, \tag{4.4}$$

$$\vec{d} = \vec{\omega_t} T, \tag{4.5}$$

where $\alpha = \arctan(\frac{cs}{f})$ is the angular difference given by one *pixel* which can be computed with the knowledge of the physical size of one pixel cs and focal length f.

Then the manipulator can be instructed to move so as to reach the given angular distance $\vec{d}$. It is necessary that the angular speed $\vec{\omega_m}$ of the manipulator decreases gradually and smoothly as the target position is approached since the harsh motion of the camera might causes the failure of the tracker (see Section 4.2.2).

Due to the fact the tracker measurements and consequently the motion commands sent to the manipulator are delayed, the regulation function must define a *deadzone*, i.e. the minimal angular distance d_{min} to the target from which the manipulator is instructed to stop, otherwise it would oscillate in the vicinity of the static target indefinitely. The angular distance d_{max} then defines the maximal desired *cutoff* angular speed $\omega_{m_{max}}$ of the manipulator. The regulation function is then given as follows:

$$\omega_m = \begin{cases} 0, & \text{if } d < d_{min} \\ f_{reg}, & \text{if } d_{min} \leq d < d_{max} \\ \omega_{max}, & \text{otherwise,} \end{cases} \tag{4.6}$$

where f_{reg} is either linear ($f_{lin} = ax + b$) or power ($f_{pow} = ax^k + b$) function. The example of a linear and power regulation function is depicted in figure 4.7.

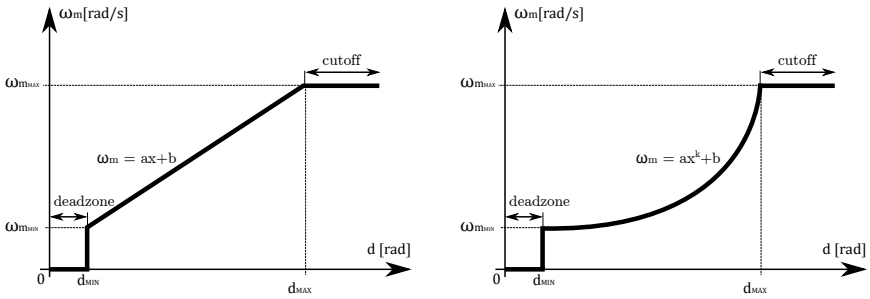


Figure 4.7: The linear and power regulation function used for controlling the angular speed of the manipulator in the given dimension (azimuth or elevation).

4.3 Target Localization Using Triangulation

This section describes the proposed triangulation method in two-view scenario (see Section 4.3.1), the incorporation of the a priori known relationship between the target-baseline orientation and the localization precision (see Section 4.3.2) and the strategy for estimating the speed and future locations of the target (see Section 4.3.3).

4.3.1 Triangulation

As explained in Section 2.2 the position of the target in 3-space as seen by the stereoscopic system with known cameras' parameters can be calculated using triangulation. In case od the OLS the intrinsics were estimated for each camera during calibration and extrinsics are known at each time thanks to the sensory data streamed from the manipulators. However, due to the random and systematic error the rays back-projected from each camera might not intersect in the 3D space (see Figure 4.8). Therefore, the common plane for both back-projected rays must first be found.

The estimation of the 3D position of the target consists of the following steps. First, back-projection is used to find the vectors $\vec{u}$ and $\vec{v}$ which form the planes C_1C_2U and C_1C_2V with the angle α between them. Both vectors are then rotated around the axis $\vec{m}$ so that they lie in the same plane C_1C_2W: $\vec{u'} = R(\beta_1)\vec{u}$, $\vec{v'} = R(\beta_2)\vec{v}$. The rotation angles β_1 and β_2 correspond to the beliefs bel_1 and bel_2 obtained from both trackers: $|\beta_1| = |\alpha|\frac{bel_2}{bel_1+bel_2}$, $|\beta_2| = |\alpha| - \beta_1$. Finally, the intersection W of the vectors $\vec{u'}$ and $\vec{v'}$ is found.

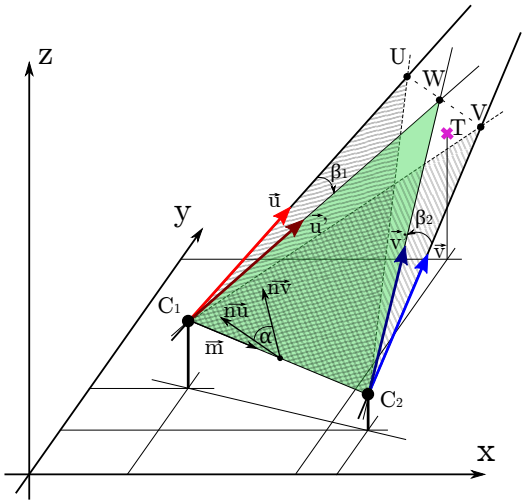


Figure 4.8: A schematic view of a problem of 3D position estimation using triangulation in two-camera scenario. The camera units C_1 and C_2 observe the target T in the directions $\vec{u}$ and $\vec{v}$. The plane C_1C_2W is used as a common plane where the projected vectors $\vec{u'}$ and $\vec{v'}$ intersect.

4.3.2 Incorporating Target-Base Geometry

If multiple CUs are used, the location of the target in 3-space can be estimated as the weighted centroid of the estimates T_i computed by each pair of the camera units forming the baseline b_i. The weights correspond to the angle γ between the baseline b_i and the line intersecting the initially estimated position of the target T' and the baseline center bc_i since this angle significantly affects the localization precision (see Section 3).

Basically, the location of the target in 3-space is estimated twice. First estimation T' corresponds to the standard centroid of the individual estimates T_i. The second estimation refines the position T' by using the weights corresponding to the target-baseline geometry (see Algorithm 2). The demonstration of the three-view scenario is shown in Figure 4.9.

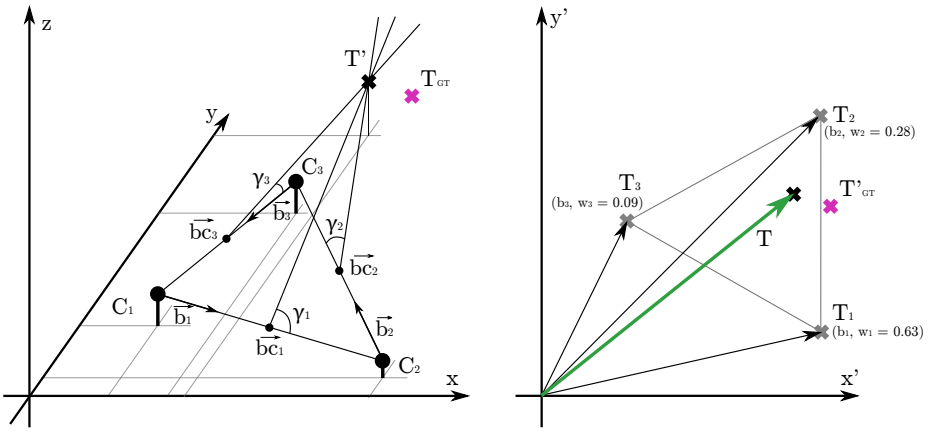


Figure 4.9: Left figure depicts the initial estimation of the target-baseline angles γ_i in three-view localization scenario with the target T_{GT}. The right figure demonstrates the final estimation of the target location using the weights $w_i \sim \gamma_i$.

Algorithm 2: Estimation of the 3D position from n-views

Input: Set of bases $B = b_1, b_2, ..., b_N$.
Output: 3D position estimate T.

/* 3D location estimate disregarding weights */
foreach $b_i \in B$ **do**
    $\vec{T_i} = Estimate3DPosFrom2Views(b_i)$
end
$\vec{T'} = \frac{1}{N}\sum_{i=1}^{N}\vec{T_i}$
/* Weighted estimation of the 3D location. $\vec{bc_i}$ represents center of the baseline b_i */
foreach $b_i \in B$ **do**
    $\gamma_i = |\vec{b_i}(\vec{T'} - \vec{bc_i})|$
    $w_i = \frac{e^{\kappa\gamma_i}}{\sum_{j=1}^{N} e^{\kappa\gamma_j}}$
end
$\vec{T} = \sum_{i=1}^{N} w_i\vec{T_i}$

4.3.3 3D Motion Prediction

The final estimates of the position of the target in 3-space are noisy and thus do not represent the motion of the tracked target well. In order to estimate current speed of the target and to predict its future locations (which can be used for occlusion prediction), the individual estimates must be smoothed. For this purpose a suitable model of the target motionmust be proposed.

The OLS is designed to localize arbitrary moving targets, therefore it is not possible to utilize one universal motion model. Nevertheless, it is assumed that the complex trajectory of a target can be locally approximated by a simple linear motion model with constant velocity and zero acceleration, i.e. the trajectory is modeled as a line in 3-space.

The method RANSAC is used in order to fit the line model to the observed time stamped measurements since it is fast and it can cope well with a large proportion of outliers [19]. First, the parameters of a line fitting to last N observations $\vec{o_i}$ with the tolerance range

d m are estimated. The inliers are then projected to the line and the speed $\vec{v}$ of a target is computed as the average weighted by the time durations t_i between consecutive observations $\vec{o_i}$ and $\vec{o_{i-1}}$:

$$\vec{v} = \sum_{i=1}^{N-1} \frac{\vec{o_i} - \vec{o_{i-1}}}{t_i - t_{i-1}} \tag{4.7}$$

The future position $\vec{p_t}$ of the target can be then simply estimated as $\vec{p} = \vec{p_0} + \vec{v}t$ given current position $\vec{p_0}$. The tolerance d was empirically set to 2 m and RANSAC is computed for $N = 30$ last observations (see Figure 4.10).

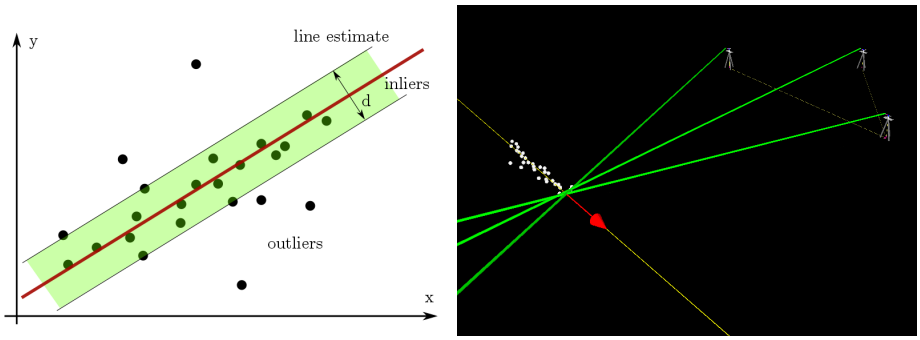


Figure 4.10: The RANSAC algorithm is capable of estimating the model parameters given the noisy data and it is robust even in the presence of outliers which are effectively disregarded (left). This method is used in the OLS in order to estimate the direction and speed of the target with the assumption of linear motion model (right). The three-view scenario where the 3D line (yellow) is fitted to the noisy measurements (white spheres) and where the speed is estimated (red arrow) is demonstrated in *rviz* environment.

4.4 Occlusion Prediction

As was explained in Section 2.3, if the 3D model corresponding to the environment is available and if the motion of the target can be predicted (see Section 4.3.3), it is possible to predict that the target will be occluded by another object. Since the OLS is composed of multiple cameras it is possible to reconstruct the environment using standard multi-view approaches such as the bundle adjustment. The problematics of obtaining the 3D model represented as a point cloud is described in Section 4.4.1 and the proposed approach to occlusion prediction is given in Section 4.4.2.

4.4.1 Obtaining the 3D Model of the Environment

Since the software tool VisualSFM (see Section 2.3.4) integrates the state-of-the-art approaches to 3D reconstruction its suitability for the OLS was tested. Even though it produces high-quality results in the scenarios where large amount of images with overlapping FOVs are provided (tens or hundreds of photographs), with just a few views (two to four) — which is the case of the OLS — the visual reconstruction performs rather unsatisfactorily.

The VisualSFM was tested with the task to reconstruct a couple of artificial environments created within the simulator Gazebo (see Section 5.3.1). The Figure 4.11 demonstrates the comparison of reconstruction output based on two and four views. It can be

seen that using mere two views results in imprecise estimates with sever noise while four views provide more reasonable results. Nevertheless, in both cases the relative positions and orientations of the cameras were not estimated correctly, hence the resulting model was wrongly positioned with regards to the ground truth position.

Unfortunately, the VisualSFM does not provide the option to set a priori known information about the intrinsics and extrinsics of each camera which are known in the OLS and which could help initialize the reprojection error optimization algorithm used by the bundle adjustment (see Section 2.3.3) so as to start near the (possibly) global minimum.

Therefore, VisualSFM does not suite the requirements of the OLS and different solution should be found in order to use the system to perform the 3D environment reconstruction on its own. The problematics of autonomous 3D reconstruction is out of scope of this work which rather focuses on how to exploit the 3D model to predict the occlusion (see Section 4.4.2).

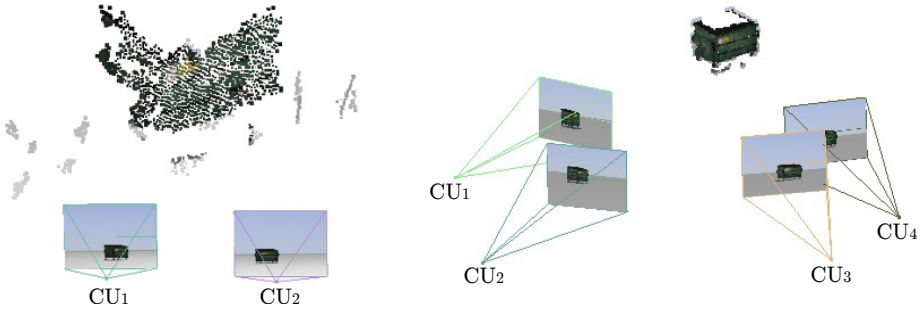


Figure 4.11: The 3D reconstruction of the rigid object given the two (left) and four (right) views captured by artificial cameras plugged-in the simulator Gazebo. Reconstruction based on four views produces far less noise, however, the position of the model is still estimated incorrectly with regards to the cameras.

4.4.2 Occlusion Prediction Algorithm and Application

A sample scenario where the full occlusion occurs for both cameras in a basic two-view setup is depicted in Figure 4.12 (a). The 3D model of the environment is represented as a point cloud $P = \{\vec{p_1}, \vec{p_2}, ..., \vec{p_N}\}$ where $\vec{p_i}$ is the i-th point $\vec{p} = [x, y, z]^T$. The function $occ(\vec{p}, \vec{T'})$ then determines for each part of the model (represented by the given point $\vec{p_i}$) whether it occludes the target T at the position T' predicted for time $t_i + \Delta t$ (where Δt is the time difference for which the future position is predicted):

$$occ(\vec{p}, \vec{T'}) = \begin{cases} 1, & \text{if } \frac{\vec{p}\vec{T'}}{|\vec{p}||\vec{T'}|} \leq \frac{\eta}{2} \wedge \|\vec{p} - \vec{CU}\| \leq \|\vec{T'} - \vec{CU}\| \\ 0, & \text{otherwise,} \end{cases} \tag{4.8}$$

where $\vec{CU}$ is the central point of the camera and eta is the required angle specifying how big surrounding of the target T' is considered (see Figure 4.12 (b)). Put in other words, only those points $\vec{p_i}$ that lie within the cone given by angle η back-projected from the camera center $\vec{CU}$ towards the predicted location of the target T' and which are closer to $\vec{CU}$ than the target T' itself.

If the occlusion is spotted, the output belief *bel* of the tacker (see Section 4.2.1) can be set 0 in order to disregard this measurement completely:

$$bel = \begin{cases} 0, & \text{if } \sum_{i=1}^{N} occ(\vec{p_i}, \vec{T'}) > T_{occ} \\ bel_{orig}, & \text{otherwise,} \end{cases} \tag{4.9}$$

where T_{occ} is an empirically set threshold used to disregard the small amount of points representing the reconstruction noise and bel_{orig} is the originally computed belief as shown in 4.2. A different softer solution is to use nonlinear function to scale the bel_{orig} to be inversely proportional to the number of points possibly causing the occlusion:

$$bel = e^{-\kappa \sum_{i=1}^{N} occ(\vec{p_i}, \vec{T'})} bel_{orig}, \tag{4.10}$$

where κ is the empirically set constant. Note that these concepts are only proposed as the possible approaches to predict the occlusion and were not implemented since the obtained datasets did not include any scenario where the occlusion would appear.

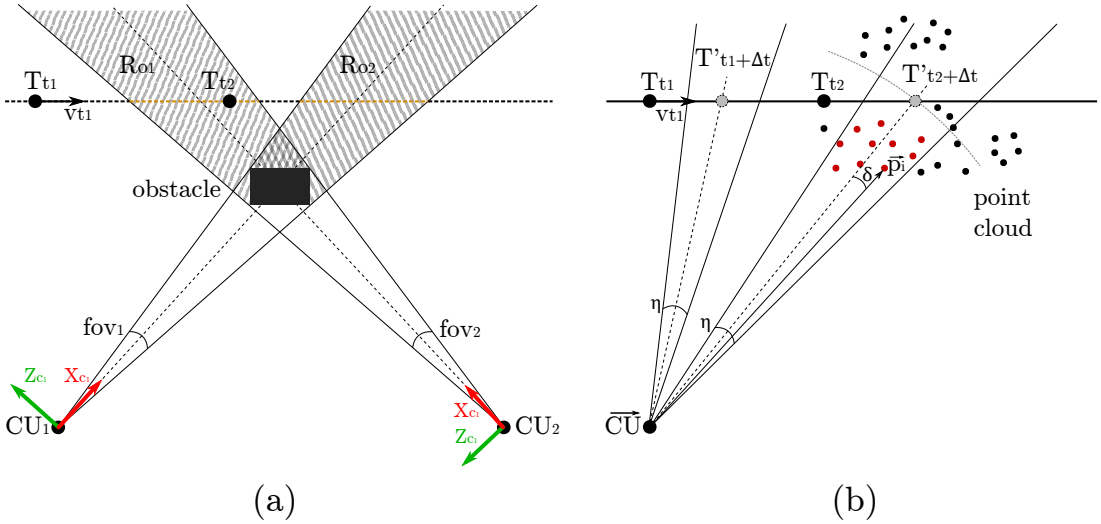


Figure 4.12: A sample schema demonstrating the occurrence of the occlusion caused by the *obstacle* (a). The demonstration of the 3D environment model represented by a pointcloud where a certain amount of points (depicted in red color) possibly cause the occlusion for target $T'_{t_2+\Delta t}$ (b).

4.5 Hardware and Software Architecture

In order to support scalability and to meet the computational demands, the OLS is designed as a distributed system where each camera unit is equipped with its own desktop computer. Sections 4.5.1 and 4.5.2 describe the hardware architecture of the system and internals of the camera unit. The section 4.5.3 then presents the main software pipeline from the input of the operator to the estimated position of the target in 3-space.

4.5.1 Hardware Topology

Considering the big picture of the system, OLS is designed as a simple two layer tree topology, where each node is represented as either tracking unit or overview unit (TU/OU) –

a standalone independent device continuously capturing the camera image stream, performing visual tracking and regulating the motion of the manipulator. All TU/OUs are interconnected via the Gigabit Ethernet switch.

One of the TU/OUs is selected as the MASTER – a root of the tree topology which is continuously receiving the tracking estimates from all the TU/OUs and estimating the 3D location of the target (see Figure 4.13). Additionally, MASTER serves the purpose of the interface between the OLS and the human operators and enables them to manually initialize the system and control each TU/OU (using keyboard and/or joystick) during runtime if necessary. All the TU/OUs are synchronized using PTP which is ensured by the hardware support provided by Prosilica cameras.

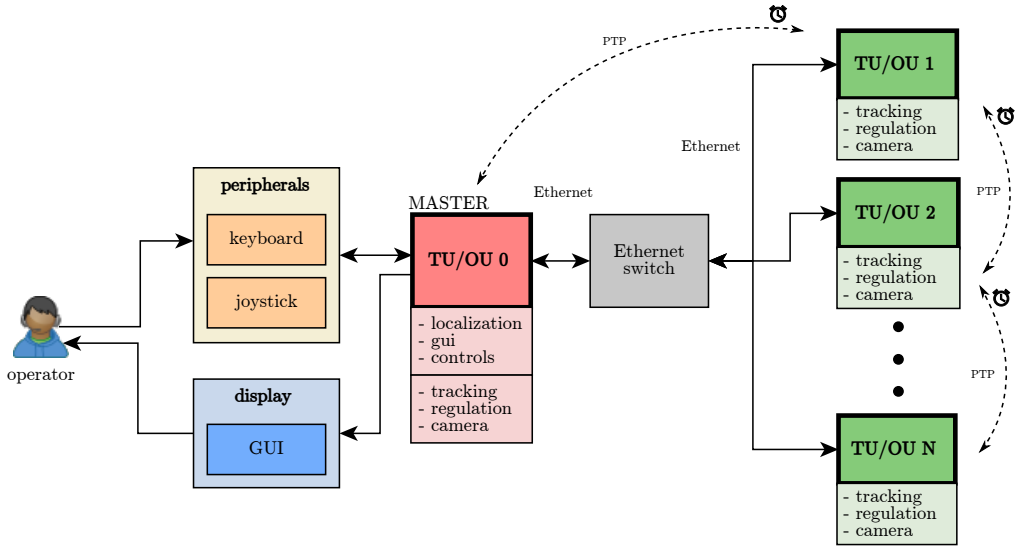


Figure 4.13: The hardware diagram of the OLS depicting the interconnection of the main building blocks – the tracking and/or overview units (TU/OU).

4.5.2 Components of the Camera Unit

As described in 4.1.1 a camera unit itself consists of a P&T unit, a camera, sensors used for stationing and a desktop computer running the OLS software. The P&T unit and all the sensors communicate through the RS-232 bus with the controller (based on μP STM32F4007) serving the purpose of the hub aggregating the communication with the desktop PC. The controller was designed and produced by the Department of Measurement at Faculty of Electrical Engineering, CTU in Prague. The camera is connected via GigE bus and finally the desktop computer is linked to the Ethernet Switch via Ethernet bus (see Figure 4.14).

4.5.3 Software Pipeline

The operation of the OLS in the means of data flow is depicted in Figure 4.15. The OLS is initialized by the human operator who directs the CUs to the desired target and starts the tracking process. All the selected tracking units then visually track the target, continuously stream current measurements to the MASTER and regulate the motion of the P&T unit so that the target would not disappear from their fields of view.

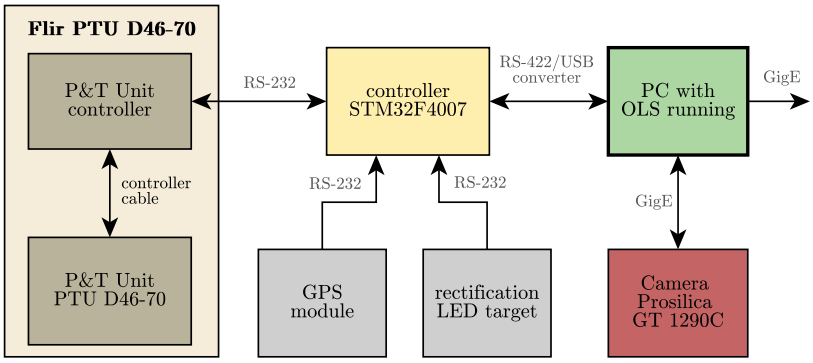


Figure 4.14: The diagram of the hardware components of the camera unit depicting both the hardware devices and the communication standards.

The localization component continuously estimates the location of the target in 3-space given the observations from the single tracking units. Given the knowledge of the target positions history and the suitable motion model the speed and future trajectory of the target can be estimated and used in order to refine the tracking. The final estimate of the current target location is continuously emitted and presented to the operator. Should the operators decide to interfere, the peripherals (keyboard/mouse/joystick) can be used to manually adjust the tracking and regulation.

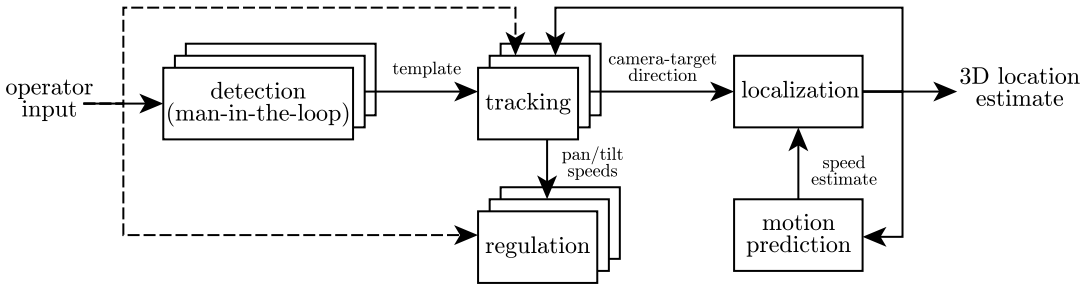


Figure 4.15: The end-to-end pipeline of the OLS in the means of data sent among the separate software components.

Chapter 5

Implementation

The implementation of the OLS is built on the robotic framework *Robot Operating System* and *C++* is used as the main programming language. Furthermore, the physical simulator *Gazebo* was used during both development and testing phase in order to prepare the system for real world environment tests. Due to the platforms support limitations of the ROS, the OLS can only be run within Linux distribution Ubuntu.

Section 5.1 provides the reasoning for why the ROS was chosen as the base framework, Section 5.2 focuses on the architecture of the OLS, shows the user interface and summarizes the external libraries integrated in the system. Finally, Section 5.3 demonstrates the necessity of using the simulator Gazebo and gives the technical background of how the ROS was used in order to obtain the real world environment datasets and to test the OLS.

5.1 Application of the ROS

The ROS[1] is a collection of open source libraries, tools and conventions which serve the purpose of a middlewear running alongside a real operating system. Among other features the ROS provides the programmer with hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing among processes, package management, a wide range of visualization and debugging tools, time synchronization or data capture and playback [39].

Utilization of ROS significantly simplified the development since it provides the tools and means meeting the requirements posed by the design of the OLS (see Section 5.1.1). Above that, the ROS encompasses multiple packages providing common algorithms and support for hardware devices which the OLS relies on (see Section 5.1.2).

5.1.1 ROS Properties Used in the OLS

The OLS is designed to become a relatively complex system, thus it exhibits non-trivial implementation requirements — among others, the distribution of the running components among multiple computers, the time synchronization or fast computation of transformations among coordinate frames. Table 5.1 lists all the main requirements of the OLS and corresponding features of the ROS.

[1] Official website of the ROS: `http://www.ros.org`
[2] Modeling language `URDF`: `http://wiki.ros.org/urdf`

Table 5.1: The table lists the most important requirements of the OLS and describes how the ROS framework addresses them.

OLS requirements	**ROS features**
support for camera Prosilica, P&T unit Flir PTU D46-70, joystick, keyboard	packages `avt_vimba_camera`, `keyboard`, `joy`
distribution of computation among multiple computers	abstraction layer for distributing nodes across devices
simple data exchange among subsystems	the publisher/subscriber paradigm [39], support for custom message formats
real time performance	C++ implementation, intrinsic OS level parallelism (each node runs as a process)
modeling and simulating the CUs	custom language `URDF`[2] for robot modeling, integration with Gazebo
modeling a kinematic chain, heavy 3D transformations computation	native support for computing transformation between frames using package `tf`
3D visualizations	tool `rviz` for visualizing transformations, robot models, image streams etc.
advanced debugging	multiple introspection tools such as `rqt_graph`, `rqt_tf_tree`, `rosparam`, `rostopic`, `rosmsg` etc.
physical simulation	integration with Gazebo
data streams synchronization	message filters for approximate time synchronization
data capturing and playback	custom container format `bag` and related tools (`rosbag`, `rqt_bag` etc.)
support for popular math and vision libraries	conversion API for easy integration of libraries OpenCV, PCL and Eigen

5.1.2 Standard ROS Packages Used in the OLS

ROS provides a wide range of packages for interaction with commonly used hardware devices and for performing various computations. Implementation of the OLS utilizes following packages:

avt_vimba_camera[3] 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` to easily access the image stream.

keyboard[4] The package processes the keyboard events and exposes them via `keydown` and `keyup` topics.

[3]Package avt_vimba_camera: http://wiki.ros.org/avt_vimba_camera
[4]Package keyboard: http://wiki.ros.org/keyboard

joy[5] This package processes the events from a joystick and/or gamepad and exposes them via `joy` topic.

tf[6] The package manages the distribution of the poses of all kinematic chain joints (representing the CUs) among all nodes and it computes the transformations between requested pair of frames [15].

5.2 System Architecture

The OLS is divided into multiple ROS nodes with the aim to loosely resemble the hardware components. Predefined ROS messages as well as the custom ones are used to exchange the data among nodes. During runtime the operator is presented with a set of windows serving the purpose of visualizing the current state and providing the interaction capabilities. The main output of the OLS is the continuous stream of the textual information representing the estimated information about the tracked target, most importantly its global location given in UTM coordinates.

5.2.1 Nodes Interaction Design

The overview of the system architecture from the perspective of the ROS namespaces, nodes, messages and services is depicted in Figure 5.1. The distribution of the computation tasks to the separate nodes loosely corresponds to the end-to-end data flow pipeline proposed in Section 4.5.3 (see Figure 4.15). The nodes running within the MASTER namespace serve the purpose of the access point for the operator as well as the main controller of the whole system while each of the namespaces `CU-N` controls a separate CU.

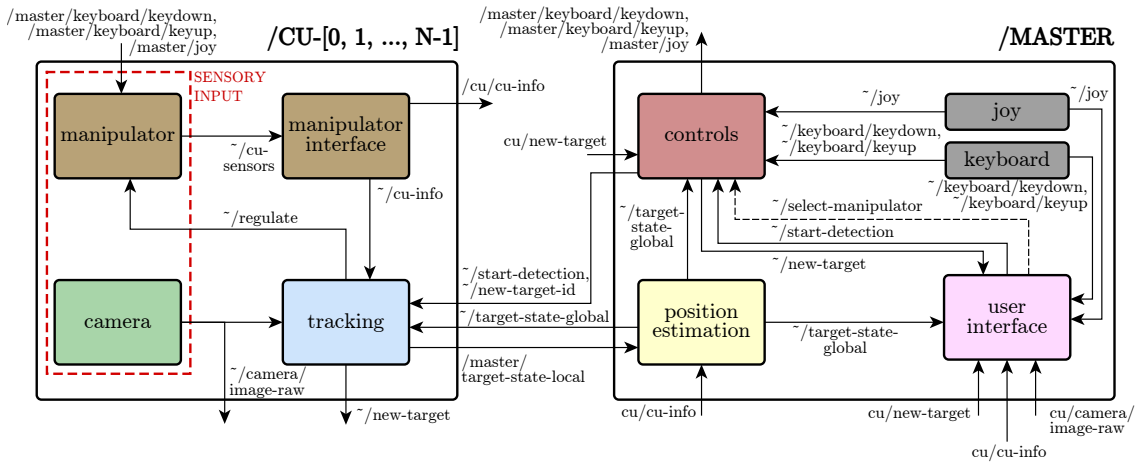


Figure 5.1: The diagram of the software components represented by the ROS nodes. The communication among topics is implemented using ROS messages (standard arrows) and ROS services (dashed arrows).

Required target is selected via the *user interface* node (using either of the peripherals controlled by the *joy* and *keyboard* nodes) and the information is immediately propagated

[5] Package joy: http://wiki.ros.org/joy
[6] Package tf: http://wiki.ros.org/tf

to the *controls* node which keeps the information about all tracked targets. The *tracking* node of the given CU is informed about the new target and the visual tracking is initialized. Furthermore, the *tracking* node continuously regulate the motion of the P&T unit — the *manipulator* node — and it publishes its estimates of the target position in two-space.

The *position estimation* node continuously receives the measurements from single trackers and it computes the final estimate of the target position in three-space. Additionally, it estimates the parameters of the target motion model. Finally, the output is presented to the operator via the *user interface* node and the textual data information is published via the *target-state-global* message.

5.2.2 User Interface

The OLS do not provide a single integrated graphical user interface. Contrarily, the user interface is composed of multiple windows, each having a different purpose (see Figure 5.2). The *camera_stream* window which is opened for each CU displays the image stream from corresponding camera and allows the user to select the target of interest (by drawing a bounding box). Furthermore, it displays the progress of tracking and estimated information about the tracked target. The *rviz* window displays the 3D visualization of all CUs and back-projected rays and the *maps* window displays the position and orientation of all CUs and the localized target.

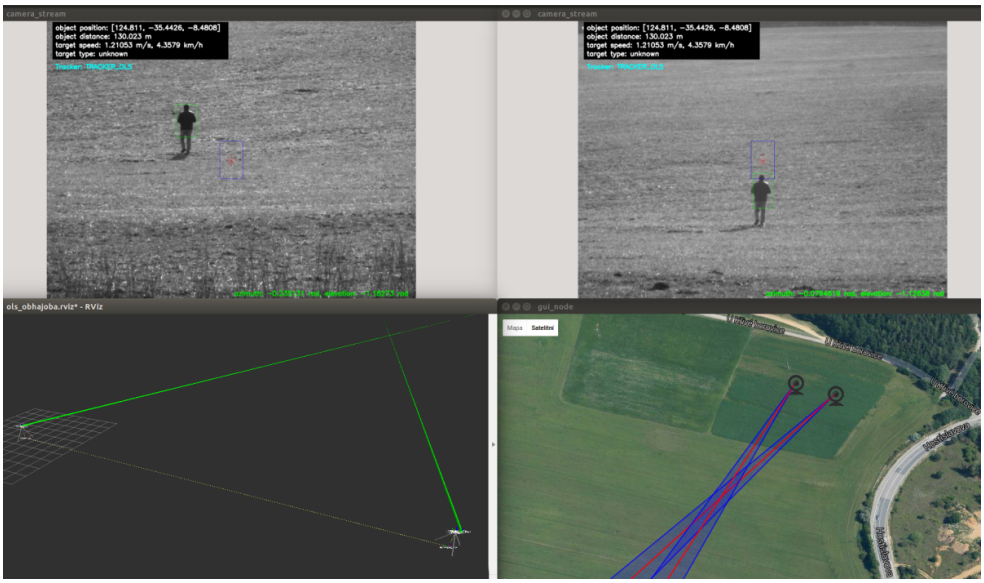


Figure 5.2: The graphical interface of the OLS containing the *camera_stream* windows (top), *rviz* window (bottom left) and *maps* window (bottom right).

5.2.3 Third Party Software Used in the OLS

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

OpenCV Computer vision library[7] providing algorithms for image processing, computer vision and machine learning. Version 2.4.11 compiled with the support for CUDA is used. In the OLS it is used by both visual trackers and by the *user interface* node.

Eigen Open source C++ template library[8] implementing the data structures and methods for fast solving of linear algebra problems. The OLS uses Eigen mostly for computations regarding the triangulation.

PCL An open-source library for 2D/3D image and point cloud processing[9]. Motion prediction based on fitting the line to the scattered data using RANSAC algorithm utilizes the PCL in the OLS.

TLD tracker The OpenTLD library[10] represents an open source C++ implementation the TLD tracking algorithm (see Section 2.1.3).

BGFG tracker The implementation of the BGFG tracker (see Section 2.1.4) which was provided by the authors of the tracker [21].

Serial A cross-platform library[11] implemented in C++ providing the API for interfacing with RS-232 serial like ports. It is used to control the P&T unit Flir PTU D46-70.

LatLong-UTM An open-source library[12] providing routines for coordinate conversion between WGS84 and UTM coordinate systems.s

5.3 Development and Testing

During both development and testing phase the physical simulator Gazebo was used. In order to test how the random error of the visual tracker impacts the localization precision the *ground truth tracker* was implemented. The real world environment datasets were obtained with the help of *bag* file format provided by the ROS.

5.3.1 Application of Gazebo

Gazebo[13] is a physical simulator providing the tools to model and simulate robots in both indoor and outdoor environment. Since the Gazebo is distributed as one of the standard packages of the ROS framework it is straightforward to integrate the simulation environment with the already implemented ROS nodes. Since the OLS is designed to track and localize very distant targets it is not possible to test the system during development in the laboratory. Therefore, the physical simulation is necessary for the development phase in order to prepare the system for the real world operation. Gazebo was used for multiple tasks:

[7] The official website of OpenCV: http://opencv.org
[8] The official website of Eigen: http://eigen.tuxfamily.org
[9] The official website of PCL: http://pointclouds.org/
[10] The official website of OpenTLD: http://www.gnebehay.com/tld
[11] The official website of Serial: https://github.com/wjwwood/serial
[12] The official website of LatLong-UTM: http://ereimer.net/programs/LatLong-UTM.htm
[13] The official website of Gazebo: http://gazebosim.org

Testing the tracker Gazebo provides the ROS plugin simulating an RGB camera which captures the virtual scene and publishes a stream of rendered images. Therefore, it can be used to test an object tracking algorithm using arbitrarily complex environment and moving objects (see Figure 5.3).

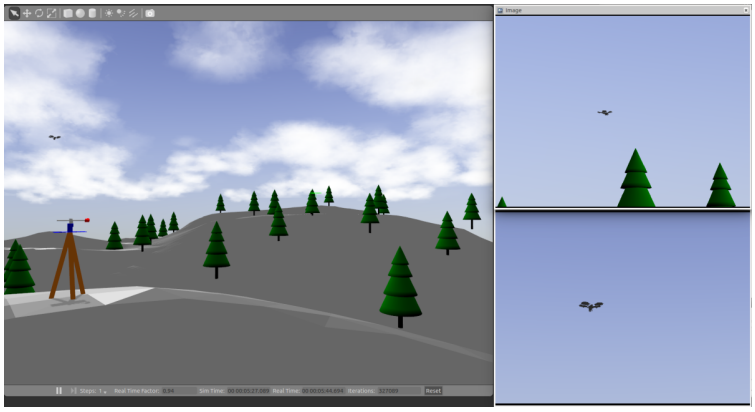


Figure 5.3: The screenshot of a Gazebo simulation (left) consisting of a virtual environment (terrain with trees), a flying object (the UAV) and two CUs. Both virtual camera streams are displayed real-time using `rviz` tool (right).

Testing the P&T unit It is a good practice to include a real hardware in the simulation during the development in the *hardware-in-the-loop* manner [2]. Both actuators and sensors would be difficult to simulate properly, moreover a real P&T unit is constrained in terms of the operational range (see Section 4.1), maximum acceleration and speed and communication throughput so it is necessary to thoroughly test its performance. The simulation reveals whether the implementation of motion control is correct and whether the possibilities of the P&T unit suffice to track arbitrarily fast (simulated) objects.

Testing the triangulation Thanks to the Gazebo it is possible to simulate a flying object with a-priory set trajectory (see Figure 5.3) and evaluate the precision of a position estimation algorithm using comparison between the estimated target position and a ground-truth. The visualization of the back-projected rays forming the skew-lines is done using visualization tool *rviz*[14] (see Figure 5.4).

5.3.2 Ground Truth Tracker

Both visual trackers employed by the OLS are expected to provide noisy and delayed measurements. The noise might be caused by either of the typical tracking difficulties (see Section 2.1) and the delay is inevitable due to the computational complexity of visual tracking algorithms. What is more, both the noise and the delay will vary across different environments and different devices running the OLS. As was explained in Section 3.2 the random error of the visual tracker can ha sever impact on the localization precision.

Therefore, in order to test the performance of the OLS utilizing the sub-optimal visual tracking algorithm, the *ground truth (GT) tracker* simulating both the noise and the delay

[14]The ROS package rviz: `http://wiki.ros.org/rviz`

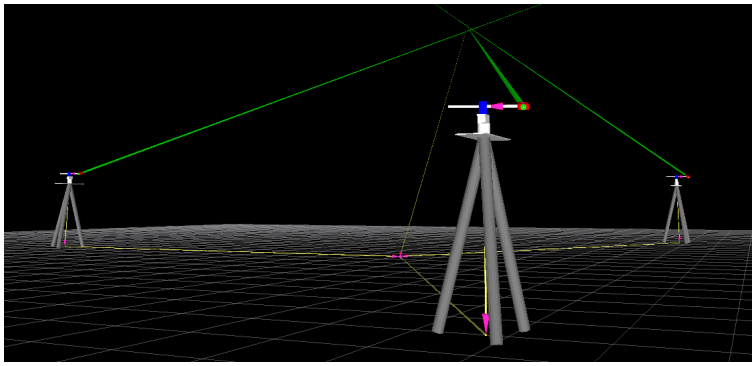


Figure 5.4: Three CUs tracking the target and the back-projected rays are visualized by the *rviz* tool.

was implemented. In time t the GT tracker outputs the estimated 2D position $o_{t-\vec{t_d}}$ of the target $O_{t-\vec{t_d}}$:

$$o_{t-\vec{t_d}} = \mathbf{P_{t-t_d}} O_{t-\vec{t_d}} + \vec{o_n}, \tag{5.1}$$

$$t_d \sim \mathcal{N}(\mu_{t_d}, \sigma_{t_d}), \tag{5.2}$$

$$o_{\vec{n_{dim}}} \sim \mathcal{N}(\mu_{n_{dim}}, \sigma_{n_{dim}}), \tag{5.3}$$

where t_d is the time delay, $\mathbf{P_{t-t_d}}$ is the projection matrix at time $t-t_d$, $\vec{o_n}$ is the (2D) noise, μ_{t_d} and σ_{t_d} are the parameters for the time delay normal distribution and $\mu_{n_{dim}}$ and $\sigma_{t_{dim}}$ are the parameters for the noise normal distribution for each dimension dim. By changing the constants μ_{t_d}. σ_{t_d}, $\mu_{n_{dim}}$ and $\sigma_{n_{dim}}$ it is possible to simulate various performance of the visual tracker.

5.3.3 Handling the Datasets

When testing the OLS in the real world environment it is necessary to record all the sensory data so that the experiments can be rerun later (with different implementations, different settings etc.). Since all the streams must be time synchronized, the time stamps must be recorded as well and the resulting dataset should be easy to playback.

For these purposes the ROS provides the data format *bag*[15] as well as the set of tools for both recording, editing and playing back the data. The OLS was designed so as to only record the sensory data coming from the P&T unit (the *manipulator* node) and the camera (the *camera* node).

Therefore, once recorded, both these nodes constituting the SENSORY INPUT (see Figure 5.1) can be easily replaced by the playback of the given *bag* representing the recorded data. This approach makes the system operate as if it was running in the real world environment, hence various tests can be rerun repeatedly.

[15]The bag format provided by the ROS: http://wiki.ros.org/Bags

Chapter 6

Experimental Results

The OLS was tested both in simulated and real world environment. The simulation was based on Gazebo and utilized hardware-in-the-loop approach. Section 6.1 describes the first test which compares various regulation functions and finds the most suitable parameters. Section 6.2 looks closely on the geometrical limitations of the OLS and verifies the proposed approach to localize the target using geometry based weights. Finally, Section 6.3 explains how the real world datasets were obtained and what precision the OLS achieves.

6.1 Regulation of the P&T Unit Motion

As explained in Section 4.2.3 it is desirable to keep the tracked target as close to the image center as possible by instructing the P&T unit to rotate the camera. Furthermore, the motion should be smooth and as fast as possible, however, it should cope with the delays imposed by the visual tracker computation, the communication latency and the latency of the hardware (P&T unit). Hardware-in-the-loop approach was taken in order to test the real P&T unit in the simulated environment and the *GT tracker* (see Section 5) was utilized.

First, both proposed regulation functions (linear and power, see Section 4.2.3) with various settings of parameters a and k were tested. In this scenario the objective was to position the camera so as to aim directly on the static target. The constants $\mu_d = 0.120\ s, \sigma_d = 0.010\ s, \mu_n = 0\ px, \sigma_n = 0\ px$ were selected for the *GT tracker* to simulate the worst case delay scenario of the visual tracker performance.

As can be seen in Figure 6.1, the linear function cannot be too steep (high value of derivative a), otherwise the P&T unit overshoots the target position and it must return (the cases of linear functions where $a = 1.0$ and $a = 5.0$) which is inadmissible behavior. Contrarily, if set correctly the power functions do not cause the overshoot and converge faster. As the result the power function with parameters $a = 2.26,\ k = 1.5$ is used in the OLS.

In order to keep the moving targets close to the center of the image plane, the prediction of future position of the target must be employed (see Section 4.2.3). A suitable value of variable T_{pred} specifying the time in the future for which the position of the target is predicted should be chosen. If T_{pred} is too small the P&T unit is not able to keep up with the motion of the target and the center of the image screen always falls behind the target. Contrarily, if T_{pred} is too high, the P&T unit overshoots to the *cutoff* region (see Figure 4.7) and anticipates the position of the target.

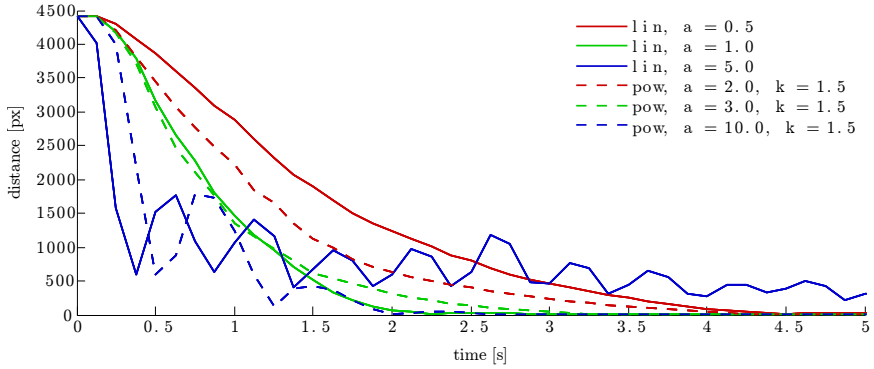


Figure 6.1: The figure depicts the Euclidean distance between the target and the center of the image plane (given in pixels) as the function of time for both linear and power regulation functions with various settings of parameters. The power function tends to converge more quickly towards the minimal distance.

Therefore, in the second test a simple scenario consisting of one P&T unit and the simulated target moving with a constant speed along a line between two boundaries was created. Different values for prediction time T_{pred} were set and the horizontal distance d_h between the center of the image PP and the projection t of the target T was measured (see Figure 6.2). As was expected, with increasing value of T_{pred} the overall distance d_h decreases up to the point where the P&T unit starts to overshoot. Note that it is not possible to select the best universal value since the performance depends on the distance, speed and motion model of the target. Therefore, it is necessary to choose the right value with regards to the given scenario.

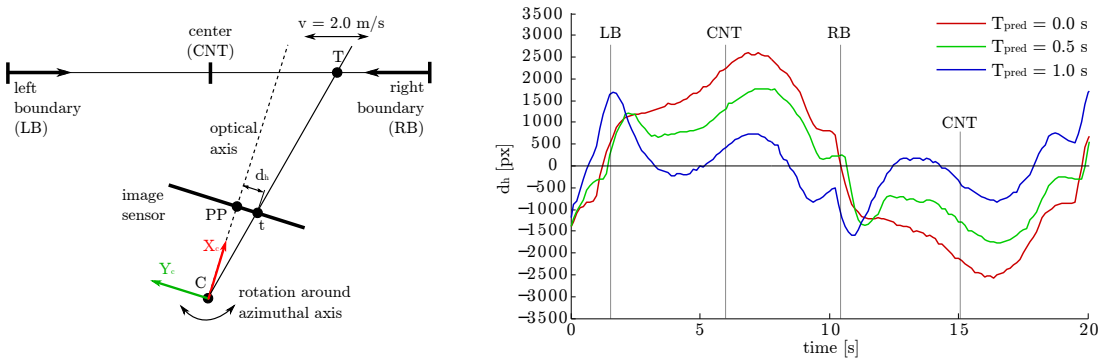


Figure 6.2: Left figure depicts the schema of the perpetual linear motion of the target moving to and fro between two boundaries. Right figure shows d_h as the function of time for different settings of prediction time T_{pred}.

6.2 Multi-camera Scenario

As was shown in Chapter 3.2 the precision of localizing the target in 3-space significantly depends on the angle γ between the baseline and the direction to the target. However, this geometry limitation can be alleviated by using multiple CUs which are appropriately placed within the environment so that at least one baseline which is oriented conveniently

with regards to the target could always be chosen .

In order to test the proposed approach of incorporating the target-base geometry by weighting the measurements from separate baselines (see Section 4.3.2) two scenarios comparing the localization precision of two and three CUs were created. In both cases the goal was to localize the target moving with constant speed along the circle. In order to simulate the random error of single visual trackers, the *GT tracker* with parameters $\mu_d = 0.120\ s,\ \sigma_d = 0.010\ s,\ \mu_n = 0\ px,\ \sigma_n = 15\ px$ was used.

In the first scenario only two CUs (CU_1, CU_2) are used and thus there are two critical regions CR_1 and CR_2 along the baseline within which the localization error is expected to be high. In the second scenario the third CU_3 was added so that all three CUs would form a regular triangle. It was expected that both critical regions would be covered by two newly created baselines. As can be seen in Figure 6.3 the expectations were met.

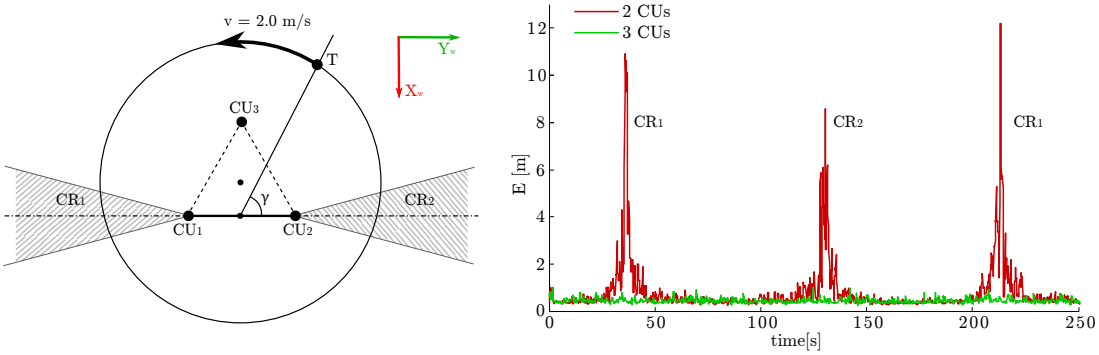
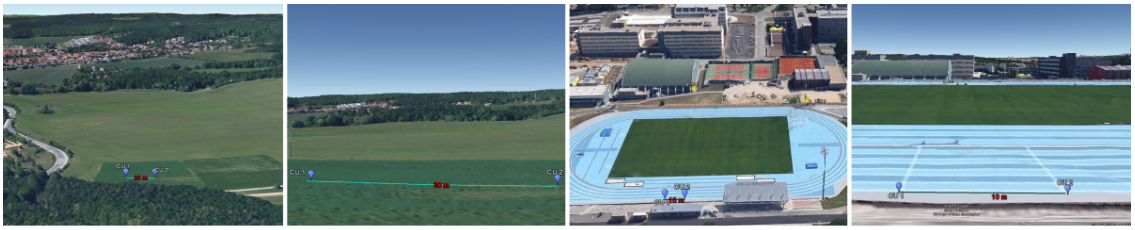


Figure 6.3: The figure to the left depicts the schematic view of two test scenarios where either two or three CUs are used. The target T moves along the circle with constant speed in the height of 2 m. The figure to the right depicts the localization error E as the function of time for both scenarios. It can be seen that in the two-camera scenario the error E increases rapidly once the target reaches either of the critical regions CR_1 and CR_2.

6.3 Real World Testing

In order to test the real world performance of the OLS two datasets were created using a basic two-camera setup. The first dataset (MEADOW) was obtained on the meadow located between districts Bystrc and Žebětín (Brno, Czech Republic). The CUs were precisely positioned using differential GPS sensor so that the baseline would be 30 m long. To create the second dataset (PITCH) the athletics stadium VUT SAPPV located in district Královo Pole (Brno, Czech Republic) was chosen since the DGPS sensor was no longer available and thus the marks measuring the distance painted on the running pitch were used to station the CUs which were placed 10 m apart (see Figure 6.4) and presumably in the same altitude.

In both datasets the local heading was estimated by aiming the units on each other as explained in Section 3.3 and the global heading was computed as the rotation of the baseline with regards to the UTM coordinate frame. Standalone digital inclinometer was used to level both stations by manually adjusting the lengths of surveying tripod legs.



(a) (b) (c) (d)

Figure 6.4: The overview and close-up look on the environments within which the MEADOW ((a), (b)) and PITCH ((c), (d)) datasets were created. The geographical positions of both CUs are denoted by the blue pinpoints.

6.3.1 Setup and Methodology

The system was tested in two different real-world environments and a basic two-camera setup was used in both cases. The first testing was performed on a meadow (MEADOW dataset) and the second one on the running pitch (PITCH dataset). Multiple static targets corresponding to the visually significant landmarks were chosen so that it would be possible to find their precise geographical coordinates using the cadastral map. In case of MEADOW dataset, a moving target equipped with the handheld GPS sensor was tracked and localized as well. During all measurements only horizontal position (i.e. northing and easting parameters of the UTM coordinate frame) was considered.

For each target, the expected *localization error* E_{est} was computed based on the theory presented in chapter 3 and it was compared to the real measured error. Note that the *localization error* E_{meas} is defined as the Euclidean distance between the estimated and ground truth geographical location of the given target given in UTM coordinates:

$$E_{meas} = \sqrt{(e_{gt} - e_e)^2 + (n_{gt} - n_e)^2} \tag{6.1}$$

where n and e are the northing and easting coordinates and subscripts gt and e denote the ground truth and the estimate.

6.3.2 Static Targets

Nine landmarks in case of MEADOW dataset and three landmarks in case of PITCH dataset were selected to test localization. Tables 6.1 and 6.2 summarize ground truth and measured UTM coordinates of the targets as well as the position of the targets with regards to the baseline (angle γ and distance d) which influences the total localization error (see Section 3).

Note that due to the fact the angle γ (see Figure 4.9) do not vary much the overall estimated error E_{est} is mainly influenced by the distance of the target d which varies significantly. Therefore the targets are sorted in ascending order with respect to the distance d. The estimated error E_{est} was calculated with assumption of the random error worth $p = 4\ px$ (see Section 3).

Figure 6.5 compares the estimated and measured localization error. Note that both E_{est} and E_{meas} follow the same trend of increasing value with increasing distance of the target from the baseline which is expected behavior caused by the phenomenon of diminishing accuracy of depth measurement (see Section 3).

Table 6.1: The results achieved for MEADOW dataset.

object	GT pos. [UTM]	estim. pos. [UTM]	d [m]	γ [$\circ$]	E_{est} [m]	E_{meas} [m]
pillar1	x: 608696.2 y: 5452998	x: 608695.05 y: 5452993.74	91.92	68.40	0.20	4.41
pillar2	x: 608714.13 y: 5452890.03	x: 608711.56 y: 5452885.16	199.14	63.91	0.95	5.51
pillar3	x: 608728.93 y: 5452804.81	x: 608731.78 y: 5452793.43	285.01	62.62	1.93	11.73
pillar4	x: 608713.61 y: 5452702.31	x: 608720.62 y: 5452686.65	386.81	66.61	3.39	17.16
tree1	x: 608687.5 y: 5452655.72	x: 608684.5 y: 5452652.72	433.88	70.31	4.13	17.20
person	x: 608481.06 y: 5452666.47	x: 608473.64 y: 5452643.75	479.96	84.09	4.65	23.90
hide	x: 608226.71 y: 5452875.99	x: 608216.86 y: 5452880.58	526.86	45.63	7.57	22.77
tree2	x: 608283.11 y: 5452618.41	x: 608287.11 y: 5452622.41	634.46	69.77	8.56	28.33
mast	x: 607816.03 y: 5452037	x: 607826.83 y: 5452069.46	1379.67	70.88	41.24	34.21

Table 6.2: The results achieved for PITCH dataset.

object	GT pos. [UTM]	estim. pos. [UTM]	d [m]	γ [$\circ$]	E_{est} [m]	E_{meas} [m]
pillar	x: 614563.07 y: 5453706.81	x: 614556.08 y: 5453703.41	125.54	52.93	1.14	7.77
hall	x: 614554.47 y: 5453794.96	x: 614547.67 y: 5453788.45	128.91	85.96	1.01	9.41
fsi	x: 614840.64 y: 5453589.38	x: 614752.41 y: 5453604.15	425.53	44.38	15.80	89.46

As for the MEADOW dataset the achieved precision is approximately 4–5 times worse than expected ranging from 4.41 to 34.21 m. Note that the targets were selected manually in the images, hence the random error plays insignificantly small role in the total imprecision. The main source of the deflections is the systematic error caused by both imprecise rectification and stationing.

In case of PITCH dataset the achieved precision is approximately 6–10 times worse than expected ranging from 7.77 to 89.46 m. The impact of the systematic error is even greater. One of the reason is the fact the CUs were not stationed using GPS, contrarily their geographical positions were estimated by finding the pitch marks (on which the CUs were placed) on the ortophotomap. It has been noted that all the wrong estimations tended to be placed in more less correct direction, however too close to the baseline which is another clue which points to the systematic error (i.e. the mounting of the cameras might have

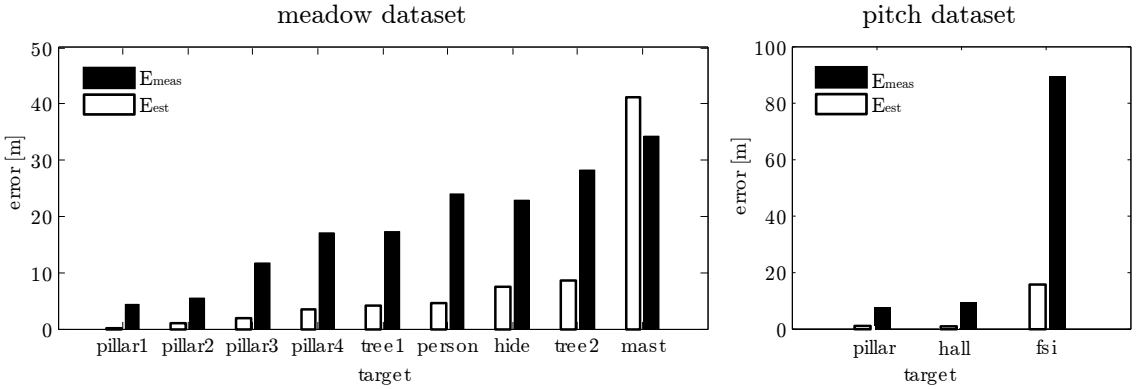


Figure 6.5: The plot displays both measured E_{meas} and estimated E_{est} localization error for all static targets, which are sorted in ascending order with respect to the distance (and consequently estimated error E_{est}).

moved slightly during the transportations).

Conclusion In order to achieve higher precision more thorough elimination of systematic error must be performed. In case of rectification the human error as well as non-robust construction of the camera mounting is the main limitation (see Section 3.4). In case of stationing more robust approach to leveling the stations as well as estimating the heading should be taken.

6.3.3 Dynamic Targets

In case of MEADOW dataset the system was tested against one dynamic terrestrial target equipped with a mobile GPS sensor – a walking person (see Figure 6.6). The target was tracked for 120 s and the estimated positions were captured and compared to the ground truth path (see Figure 6.7). On average the system achieved the precision of 6.25 m. Note that the position estimates oscillate around the ground truth trajectory, which is caused by the random error made by both trackers; the error, however, keeps in the specific range and reaches maximum of 13.35 m. The mean error is higher as compared to the estimated error (see Section 3), which is again caused by the systematic error (imprecise rectification and stationing).

An attempt was made to localize another moving target in PITCH dataset as well, however, it was not possible to evaluate the precision since the visual tracker was not able to follow the target properly. The main reason for faulty operation was the cluttered background of the target which prevented the tracking subsystem from providing correct measurements.

Conclusion Precision of localization of the dynamic target is limited in the same way as in case of static targets. Above that, the system greatly depends on the precision of the visual tracker which should be investigated more in order to avoid incorrect measurements.

Figure 6.6: The two-camera setup where the distant target is tracked by both CUs (left and right). The estimated position of the target is displayed in the map (center).

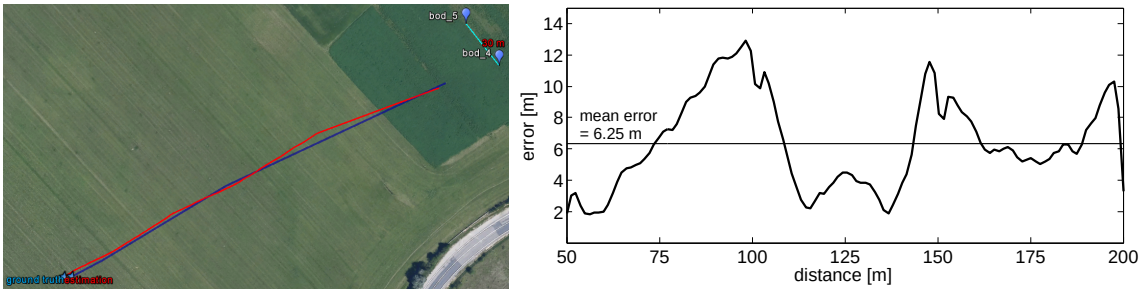


Figure 6.7: The comparison of the ground truth and estimated trajectory of a target moving in the distance range of ca 50–200 m (left). The error as the function of the distance of the target is also displayed (right). The system makes the average error of 6.25 m.

Chapter 7

Conclusion

A novel system for semi-autonomous optical localization of arbitrary distant targets was presented in this work. First, the existing state-of-the-art approaches to visual tracking were summarized and two techniques, the TLD tracker and BGFG tracker, were selected to be incorporated in the OLS. Then the problematics of multi-view triangulation was discussed and the approaches used to reconstruct the 3D environment were discussed.

Thorough precision analysis was performed so as to pinpoint the most severe sources of the error. The geometrical limitations of the OLS were demonstrated and the merits of multiple-view setup over the stereo setup in the means of localization precision were shown. The stationing and rectification procedures aiming on eliminating the systematic error were then proposed.

The kinematic chain based model of the camera was proposed and the BGFG tracker was adjusted so as to minimize the impact of the random error. Suitable functions for regulating the motion of the P&T unit were discussed and the approach to triangulate the target position with the knowledge of trackers' belief and target-base geometry was designed.

The OLS was implemented in C++ within the framework ROS and simulator Gazebo was used to test the system in hardware-in-the-loop manner. Real world datasets were obtained and the system was tested against both static and moving targets. It has been shown that the localization precision is 4-10 times lower than expected while most likely culprit is the imprecisely performed stationing and rectification.

As for the further development the tracking algorithm should be investigated so that it would perform well even with cluttered background and more thorough stationing and rectification should be performed. Furthermore, automatic detection and possibly classification of moving targets could be implemented and the approach to handoff the target among separate CUs should be proposed to make the OLS fully autonomous.

The work was conducted in collaboration with consortium of companies RCE systems s.r.o[1]., Oprox a.s.[2] and the Department of Measurement at Faculty of Electrical Engineering, Czech Technical University in Prague[3] as a real world application. The development will continue so that a standalone solution could be produced to potential customers.

[1]The official website of RCE systems s.r.o: http://www.rcesystems.cz/
[2]The official website of company Oprox a.s.: http://www.oprox.cz/
[3]The official website of Department of Measurement, FEL, ČVUT: http://measure.fel.cvut.cz

Bibliography

[1] Sepehr Aslani and Homayoun Mahdavi-Nasab. Optical flow based moving object detection and tracking for traffic surveillance. *International Journal of Electrical, Electronics, Communication, Energy Science and Engineering*, 7(9):789–793, 2013.

[2] 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.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[4] Alberto Del Bimbo and Fabrizio Dini. Particle filter-based visual tracking with a first order dynamic model and uncertainty adaptation. *Computer Vision and Image Understanding*, 115(6):771–786, 2011.

[5] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243 vol. 2, June 2005.

[6] Michael D. Breitenstein and Fabian Reichlin. Robust tracking-by-detection using a detector confidence particle filter. In *IEEE International Conference on Computer Vision*, October 2009.

[7] R. Bud and D.J. Warner. *Instruments of Science: An Historical Encyclopedia.* Garland encyclopedias in the history of science. Science Museum, London, and National Museum of American History, Smithsonian Institution, 1998.

[8] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, May 2003.

[9] Erik V Cuevas, Daniel Zaldivar, and Raul Rojas. Kalman filter for vision tracking. 2005.

[10] B. Cyganek. *An Introduction to 3D Computer Vision Techniques and Algorithms.* John Wiley & Sons, 2007.

[11] Quentin Delamarre and Olivier Faugeras. 3d articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3):328 – 357, 2001.

[12] A. Doucet, A. Smith, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice.* Information Science and Statistics. Springer New York, 2001.

[13] Zhimin Fan, Ying Wu, and Ming Yang. Multiple collaborative kernel tracking. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 502–509 vol. 2, June 2005.

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

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

[16] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall Professional Technical Reference, 2002.

[17] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[18] Ivan Antonio Mantilla Gaviria. *New strategies to improve multilateration systems in the air traffic control*. PhD thesis, Universitat Politecnica de Valencia, 2013.

[19] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

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

[21] David Herman, Filip Orság, and Martin Drahanský. Object tracking in monochromatic video sequences using particle filter. In *7th Scientific International Conference - Enviromental Protection of Population*, pages 120–128. Karel Englis College Inc., 2012.

[22] 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.

[23] W. Hu, X. Zhou, W. Li, W. Luo, X. Zhang, and S. Maybank. Active contour-based visual tracking by integrating colors, shapes, and motions. *IEEE Transactions on Image Processing*, 22(5):1778–1792, May 2013.

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

[25] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

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

[27] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

[28] Suha Kwak, Woonhyun Nam, Bohyung Han, and Joon Hee Han. Learning occlusion with likelihoods for visual tracking. In Dimitris N. Metaxas, Long Quan, Alberto Sanfeliu, and Luc J. Van Gool, editors, *ICCV*, pages 1551–1558. IEEE Computer Society, 2011.

[29] J. Lee, S. Lim, J. G. Kim, B. Kim, and D. Lee. Moving object detection using background subtraction and motion depth detection in depth image sequences. In *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on*, pages 1–2, June 2014.

[30] Xi Li, Weiming Hu, Chunhua Shen, Zhongfei Zhang, Anthony Dick, and Anton Van Den Hengel. A survey of appearance models in visual object tracking. *ACM Trans. Intell. Syst. Technol.*, 4(4):58:1–58:48, October 2013.

[31] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[32] X. Mei, S. K. Zhou, and F. Porikli. Probabilistic visual tracking via robust template matching and incremental subspace update. In *2007 IEEE International Conference on Multimedia and Expo*, pages 1818–1821, July 2007.

[33] Cyrille Migniot and Fakhreddine Ababsa. 3d human tracking in a top view using depth information recorded by the xtion pro-live camera. In *ISVC (2)*, volume 8034 of *Lecture Notes in Computer Science*, pages 603–612. Springer, 2013.

[34] Ondrej Miksik and Krystian Mikolajczyk. Evaluation of local detectors and descriptors for fast feature matching. In *ICPR*, pages 2681–2684. IEEE, 2012.

[35] Airways Museum and Civil Aviation Historical Society. Air traffic services surveillance systems [online]. http://www.airwaysmuseum.com/Surveillance.htm.

[36] Georg Nebehay and Roman Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *Winter Conference on Applications of Computer Vision*, pages 862–869. IEEE, March 2014.

[37] SeungJong Noh and Moongu Jeon. *Computer Vision – ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part III*, chapter A New Framework for Background Subtraction Using Multiple Cues, pages 493–506. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[38] Reza Oji. An automatic algorithm for object recognition and detection based on ASIFT keypoints. *CoRR*, abs/1211.5829, 2012.

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

[40] Raphael Ortiz. Freak: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society.

[41] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[42] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[43] Sanjay Kumar Sahani, G. Adhikari, and B. Das. A fast template matching algorithm for aerial object tracking. In *Image Information Processing (ICIIP), 2011 International Conference on*, pages 1–6, Nov 2011.

[44] Cameron Schaeffer. A comparison of keypoint descriptors in the context of pedestrian detection: Freak vs. surf vs. brisk, 2013.

[45] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM.

[46] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[47] M. Tang and J. Feng. Multi-kernel correlation filter for visual tracking. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3038–3046, Dec 2015.

[48] J.R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. A series of books in physics. University Science Books, 1997.

[49] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

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

[51] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

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

[53] K. Wang and Z. Ren. Enhanced gaussian mixture models for object recognition using salient image features. In *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pages 1229–1233, Aug 2007.

[54] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.

[55] Changchang Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 3057–3064, Washington, DC, USA, 2011. IEEE Computer Society.

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

[57] Jean yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker. *Intel Corporation, Microprocessor Research Labs*, 2000.

[58] Li Zhang and Ramakant Nevatia. Efficient scan-window based object detection using gpgpu. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 0:1–7, 2008.

[59] Shengping Zhang, Junpeng Wu, Yuan Tian, Shaohui Liu, and Xin Sun. Robust visual tracking based on occlusion detection and particle redistribution. In *Proceedings of the Second International Conference on Internet Multimedia Computing and Service*, ICIMCS '10, pages 159–162, New York, NY, USA, 2010. ACM.

List of Abbreviations

ATC	air traffic control
BGF	background/foreground tracker
BPF	Bootstrap particle filter
CCD	charge-coupled device
CU	camera unit
DGPS	Differential Global Positioning System
DLT	direct linear transform
FM	foreground mask
FOV	field of view
LED	light-emitting diode
OLS	Optical Localization System
P&T	pan and tilt
PCL	Point Cloud Library
PTP	Precision Time Protocol
ROS	Robot Operating System
SSD	sum of square differences
TLD	Tracking-Learning-Detection tracker
TU/OU	tracking unit/overview unit
UTM	Universal Transverse Mercator
WGS84	World Geodetic System 1984

Appendices

List of Appendices

Appendix A

DVD Contents

Following files can be found on the attached DVD:

/text	Source files of this work (.tex), the figures and the final document (.pdf).
/poster	The poster of this work presented at conference Excel@FIT 2016 (A1 format, .pdf).
/video	The video overviewing this work presented at conference Excel@FIT 2016 (.mp4, h264).
/paper	The paper published at conference Excel@FIT 2016 (.pdf).
/ols/src	Source code of the OLS divided into separate ROS packages (.cpp, .h, other auxiliary formats).
/ols/data	Part of the meadow dataset included (.bag).
/ols/doc	The generated documentation of the source code (.html).
/misc/photos	Various photographs of the OLS taken during development, rectification and testing (.jpg).
/misc/videos	Various videos documenting the development and testing of the OLS (.mp4, h264).
/README.txt	Text file including the description of the DVD contents, system requirements, building and running instructions and the usage manual.

Appendix B

Usage of the OLS

Table B.1 summarizes the controls used to interact with the OLS. It is expected that the *camera_stream* window (see Section 5.2.2) is in focus.

Table B.1: OLS controls.

KEYBOARD	
0-9 (alphanumeric)	Switch among CUs.
left/right arrows	Rotate the P&T unit around azimuthal axis.
down/up arrows	Rotate the P&T unit around elevation axis.
4, 6 (numeric)	Change the horizontal size of the bounding box.
2, 8 (numeric)	Change the vertical size of the bounding box.
Y	Select TLD tracker.
U	Select BGFG tracker.
I	select GT tracker.
Enter	Start/stop tracking.
T	Switch on/off regulation of the P&T motion.
S	Switch stepper mode on (rotate P&T unit by steps).
C	Switch continuous mode on (rotate P&T unit continusously).
Home	Rotate the P&T unit so as to reach azimuth = 0 rad.
End	Rotate the P&T unit so as to reach elevation = 0 rad.
MOUSE	
left button	Start tracking the selected image region.
JOYSTICK/GAMEPAD	
axis-0	Rotate the P&T unit around azimuthal axis.
axis-1	Rotate the P&T unit around elevation axis.
axis-2	Change the horizontal size of the bounding box.
axis-3	Change the vertical size of the bounding box.
button-9 (start)	Start/stop tracking.

Appendix C

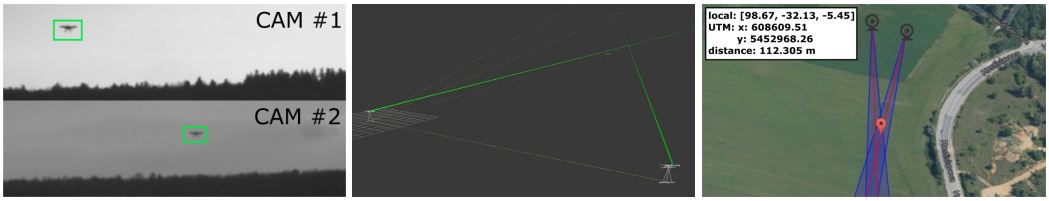
Paper - Excel@FIT 2016

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2016
http://excel.fit.vutbr.cz

Optical Localization of Very Distant Targets in Multi Camera System

Jan Bednařík*



Abstract
This paper presents a system for automatic optical localization of distant moving targets using multiple pan-tilt cameras. The cameras were precisely calibrated and stationed using custom designed calibration targets and methodology. The detection of the target is performed manually, while the automatic visual tracker combines the background/foreground modeling and motion model in the particle filer framework. The estimation of the 3D location is based on the N-view triangulation. A basic setup consisting of two camera units was tested against static targets and a moving terrestrial target, and the location estimation precision was compared to the theoretical model. The modularity and portability of the system allows fast deployment in a wide range of scenarios including perimeter monitoring or early threat detection in defense systems, as well as air traffic control in public space.

Keywords: multi-camera localization — visual object tracking — 2D motion prediction — particle filter based tracking — stationing and rectification — articulated model of PT unit — 3D localization using triangulation — physical simulation using Gazebo — robotic system design using ROS

Supplementary Material: Demonstration Video

*jan.bednarik@hotmail.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

An autonomous localization of arbitrary moving targets is an essential system component in multiple domains, such as air traffic control, robotic workspaces or surveillance and defense systems. If the sensory data measured by the target are available, it is straightforward to derive its location (by means of the GPS, radio multilateration, etc.). There are scenarios, however, were the target is unable (malfunctioning aircraft) or reluctant (UAV intruder) to expose its location. Then the localization estimation system is left with its own observations.

Radars, the most widely used devices for localizing distant targets, suffer from being unportable, energy-intensive and expensive. Furthermore, it might be desirable that the tracked object not find out that it is being tracked, which is the condition an actively radiating system cannot achieve.

This paper introduces a semi-autonomous passive multi-camera system for tracking and localizing the distant objects, which is based merely on ordinary RGB cameras — Optical Localization System (OLS). The system is designed to suit mobility and temporary deployment because each camera station weighs no more than twenty kilograms and the whole system is inexpensive by comparison to radars as well.

2. Related Work

The choice of how the targets are represented determines the domain of approaches used for visual detection and/or tracking. In general, two main representations are used [1]: a *shape* model which encompasses e.g. points [2], contours [3, 4] or articulated models [5, 6], and an *appearance* model which is represented by a template [7] or active appearance model [8].

Moving object detection Depending on the object model, the detection might be performed either by detecting keypoints and matching them against the pre-trained model [9, 10, 11], or by dividing the image into individual patches in which the object is searched for. For each patch, the template matching is performed [12, 7] or feature set is extracted; consequently, the model presence probability is evaluated using the generative or the discriminative classifier [13, 14]. Since the exhaustive search within the whole image is computationally expensive, the cascade classifiers are applied [15, 16]. Alternatively, the moving object can be detected in the image regions yielding the highest response of frame differencing [17, 18].

Object tracking There are multiple approaches to visual tracking. Keypoint tracking represents one of the most common ones [2, 19]. Kernel approaches are based on a weighted kernel used to derive smooth distance function which can be optimized in the means of target position using traditional gradient based methods such as gradient descent [20], or even multiple collaborative kernels might be used [21, 22]. Other approaches rely on tracking-by-detection concept which heavily utilizes the detection principles in combination with motion-aware approaches to localize the object [23, 24]. To reinforce the tracker robustness, the motion models are often used, Kalman filter and particle filter being the most popular ones [25, 7].

Multi-view optical localization Multi-camera localization is mostly used in the domain of robotics, where the *intelligent space* consisting of several cameras with a priory known and fixed intrinsics and extrinsics is utilized [26]. The centralized system uses either mere visual information or enhances the localization with the help of robots' sensory data [27, 28, 29, 30]. Bound to the predefined space and using fixed cameras, those systems do not need to deal with the imprecise estimates of a current camera pose.

3. System Overview

The main building block of the OLS is a camera station (CS), a standalone unit consisting of hardware modules

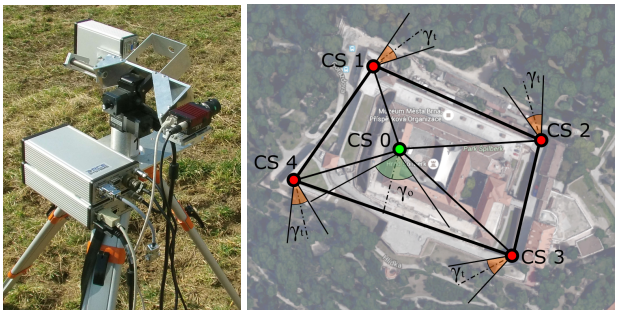


Figure 1. Tracking camera station (left) and a use case scenario (right) showing the positioning of four tracking stations (red dots) and one observation station (green dot) to protect a real world area.

necessary for capturing the images, manipulating the pose of the camera and estimating its own geographical coordinates. There are two type of CSs. The *overview station* is designed to be controlled manually by the human operator and is equipped with the zooming lens that allows achieving both a wider scanning range and a more detailed view of the farther objects. The *tracking station* consists of the fixed lens and takes part in the autonomous tracking of the moving objects.

The OLS is designed to work with an arbitrary number of CSs. Due to the geometric limitations of the multi-view systems, which affect the localization precision, the tracking stations should be positioned so as to form approximately regular polygon with long enough bases (see Section 4, see Figure 1).

The camera station itself consists of a surveying tripod, a P&T unit[1] Flir PTU-D46-70, a camera Prosilica GT 1290C (RGB, 1280×960 px, 33.3 FPS), an inclinometer and a GPS sensor. A camera unit is modeled as a kinematic chain consisting of six joints and five links corresponding to the distance between separate parts of the tripod and the manipulator (see Figure 2). The transformation between the `GROUND` and `ORIGIN` reflects the positioning and heading of the given manipulator within the local coordinate frame.

The kinematic chain is designed as composition of transformation matrices where a single joint can be located by applying the Euclidean transformation on the position of the joint which it is depending on:

$$M_{next} = M_{previous} T_{next} R_{Z_{next}} R_{X_{next}} R_{Y_{next}}, \qquad (1)$$

where M is the transformation matrix of the given joint, T is the translation between successive joints and R_a is the rotation around axis a.

4. Localization Precision

The precision of estimating the target position is subject to systematic error (miscalibration of the instru-

[1] Pan and Tilt.

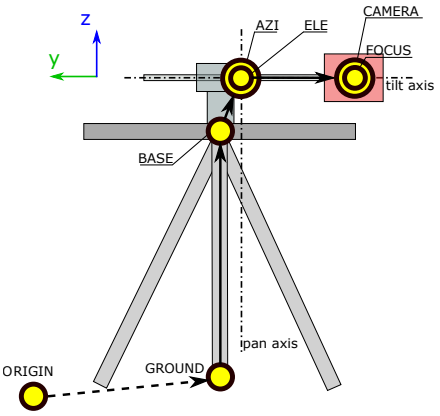


Figure 2. The model of a camera unit represented by a kinematic chain consisting of six joints (yellow dots) and 5 links (black arrows). The joints `AZI` and `ELE` share exactly the same position, the joint `CAMERA` is further along the X axis than the joint `FOCUS`.

ments) and random error (wrong measurements and disturbances in the environment) [31]. Atmospheric turbulence, refractive index fluctuations and uncertainty of the visual tracker are the main causes of the random error which is analysed in section 4.1. The systematic error was alleviated and/or measured using the custom designed stationing and rectification process.

4.1 Random Error Analysis

Stereoscopic systems are affected by a phenomenon of diminishing accuracy of depth measurement with increasing distance of the target from the cameras [32]. The depth measurement resolution for canonical stereo setup is $R = \frac{rZ^2}{fb - rZ}$, where f is the focal length, b is the base length, r is the horizontal size of one pixel and Z is the target distance. By substituting r by pr, where p is the random error represented by integer number of pixels we obtain the position estimation error function $E = \frac{prZ^2}{fb - prZ}$.

The OLS does not conform to the canonical stereo setup (all cameras can rotate freely), so the dependence of the error on the target distance is no longer quadratic (considering the setup of two cameras where only one of them makes error): $E = B\tan(\arctan(\frac{D}{B}) + \arctan(\frac{r}{f})) - D$. The cameras setup as well as the error shown as the function of the base length and target distance is depicted in Figure 3.

A more realistic scenario, where each camera makes a random error p, is depicted in Figure 4. A significant advantage of using multiple cameras is demonstrated — geometrical limitations of the two-camera setup make it impossible to precisely evaluate the position of the target placed close to the line collinear with the baseline. In the multi-camera setup, on the other hand, the subset of two cameras forming the baseline B_i is used for each position of the target, following the rule:

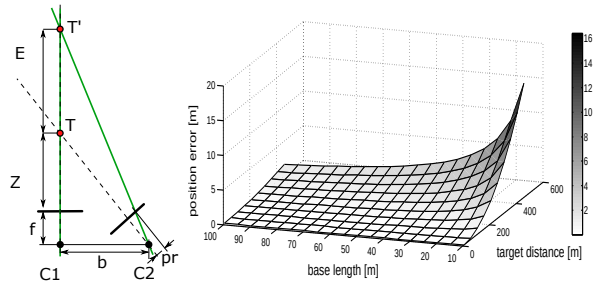


Figure 3. Left figure depicts the setup of two cameras $C1$ and $C2$, where only $C2$ makes an error worth p pixels. T represents the ground truth position of the target whereas T' is the wrongly estimated position. Right figure shows the position estimation error as the function of base size and target distance (given the random error $p = 10$ px and following constants: $r = 3.75e^{-6}$ m, $f = 50e^{-3}$ m).

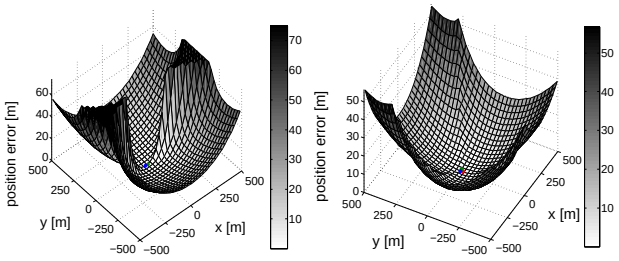


Figure 4. The position estimation error as a function of the horizontal position of the target. Two-camera (left) and three-camera (right) setup with $b = 20$ m are confronted, where utilization of more cameras always yields lower errors. The first two cameras are placed on the X axis with the coordinate frame center in the middle of their baseline. The third camera is placed on the Y axis, so that all the cameras form a regular triangle.

$i = \underset{i}{\text{argmax}}\ \vec{t_i}\vec{n_i}$, where $\vec{t_i}$ is the direction of the line segment linking the center of the baseline and the target and n_i is the normal vector of the baseline B_i.

4.2 Stationing and Rectification

The stationing procedure alleviates two types of systematic error: wrong heading estimation and undesirable tilt of the camera station. Since the accuracy of the commercial magnetometers is too low (hundreds to thousands of milliradians), the precise heading must be estimated visually by observing the distinctive landmarks. To achieve the horizontality of the station a digital inclinometer can be used.

The imprecision of the camera-manipulator attachment causes slight undesirable rotation of the camera coordinate frame. Three horizontally leveled rectification targets are used to alleviate and/or measure all rotation angles (around X, Y and Z axis): 5):

Rotation around optical axis The target contains parallel horizontal lines and the camera displays the blend of the original and vertically mirrored streams. The aim is to rotate the camera manually so that the

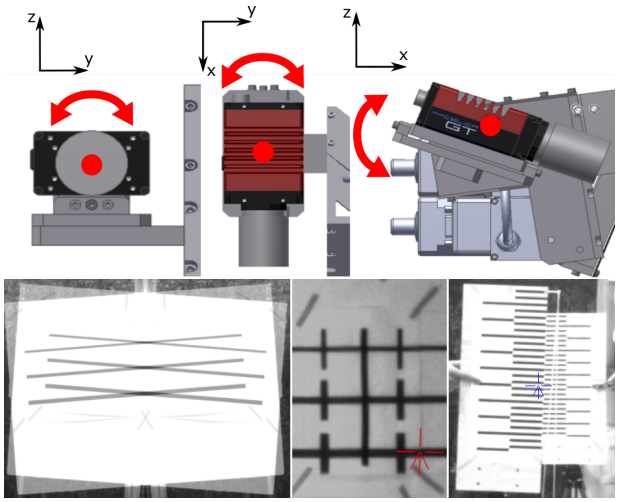


Figure 5. Three rectification targets (bottom) used to alleviate and/or measure the undesirable rotation angles of the cameras (top).

lines would appear aligned.

Rotation around azimuthal axis The target contains parallel horizontal lines and a pair of crosses whose distance equals the distance between ELE and CAMERA joint (see Figure 2). The aim is to measure the distance between the right cross and the intersection of the optical axis with the target, which translates to an error angle in the azimuth.

Default elevation angle Two targets which contain black and white lines representing a ruler are positioned in a row. The aim is to adjust the tilt of the camera so that the optical axis would intersect the same mark on both targets and the resulting elevation angle could be measured.

5. Object Tracking

The detection itself has been performed manually so far in the man-in-the-loop manner, while the autonomous tracking uses the implementation of the visual tracker combining the background subtraction, motion model and object model in the particle filter framework [7]. This approach can even cope with the moving cameras and thus is suitable for the OLS. The operation of the tracker is described below.

The target is represented as a rectangular template (consisting of gray-scale intensity values), which is normalized to the size 24×24 pixels. The advantage of the template representation is that it contains both spatial and appearance information. The template is created only once during the initialization, and thus the tracker could fail if the target changed its appearance significantly during the course of tracking. However, for very distant targets, no or merely small change is expected.

The Bootstrap particle filter (BPF) — the variant of a particle filter following the sequential importance sampling approach [33] — is used to generate and evaluate candidate positions of the target. Each particle (i.e. the state of the system) is represented as $\vec{x_n} = (x, y, v_x, v_y, h, w)$, where (x, y) represents the 2D position of the target, (v_x, v_y) represents the estimated speed of the target and (h, w) represents the bounding box size.

The perturbations in the observed position of the target caused by the moving camera are alleviated using the motion model which is applied in the *prediction* step of the BPF:

$$pos_{n+1} = pos_n + vel_n + \gamma_{pos} \sim \mathcal{N}(\mu, \sigma), \quad (2)$$

$$vel_{n+1} = vel_n + \gamma_{vel} \sim \mathcal{N}(\mu, \sigma), \quad (3)$$

$$bb_{n+1} = bb_n + \gamma_{bb} \sim \mathcal{N}(\mu, \sigma), \quad (4)$$

where scalar pos_n is the x or y position, scalar vel_n is the x or y velocity, scalar bb_n is the w or h size of the bounding box in time n, and γ is the noise drawn from the Gaussian distribution $\mathcal{N}(\mu, \sigma)$, where scalars μ and σ parameters are set empirically for each parameter.

In the *update* step, each particle is assigned a new weight w using the objective function reflecting the similarity of the template and the candidate patch:

$$w = \sum_{(x,y) \in I} e^{min(M_t^{(x,y)}, M_c^{(x,y)})} (1 - |I_t^{(x,y)} - I_c^{(x,y)}|)^2, \quad (5)$$

where M_t and M_c are the foreground masks (FM) of the template and the current candidate respectively, I is the image, t, c subscripts denote template and candidate patch respectively, and (x, y) superscript denotes indexing 2D array (an image). The FMs are estimated by subtraction of the two images where the bounding boxes denoting the position of the target do not overlap (the FM M_t is estimated only once). The resulting estimate of the target position is chosen using the Maximum a posteriori approach.

In order to enable the motion of the camera, the transformation between each pair of adjacent frames is estimated by detecting and tracking the keypoints using KLT tracker [2] and then estimating the homography using the RANSAC algorithm [34].

The homography might not be found, which often occurs if the airborne target with the uniform background of the sky is tracked or if the manipulator moves the camera too harshly. To deal with such cases in OLS, the tracker was adjusted so that in a *prediction* step of the BPF a small subset of particles would be forced to take the image positions yielding the highest response of adjacent frame differencing which is expected to contain the moving target of interest.

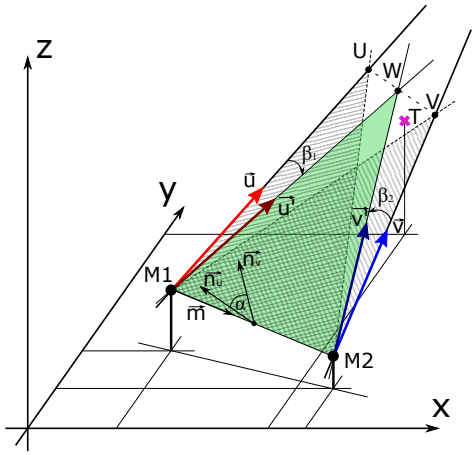


Figure 6. A schematic view of a problem of 3D position estimation using triangulation in two-cameras scenario. The camera units $M1$ and $M2$ observe the target T in the directions $\vec{u}$ and $\vec{v}$. The plane M_1M_2W is used as a common plane where the projected vectors $\vec{u'}$ and $\vec{v'}$ intersect.

6. Target Localization Using Triangulation

The hardware cameras are modeled as finite pinhole cameras based on the projection matrix P [34]:

$$P = KR[I|-C],$$
$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix},$$

where K is the intrinsics matrix and R and C are the rotation and translation matrices representing the orientation and position of the camera frame. The 3D point $\vec{X}$ projects to the 2D image point $\vec{x}$ via $\vec{x} = P\vec{X}$. If only the projection $\vec{x}$ is observed, the 3D line mapping to $\vec{x}$ can be computed using *back-projection*:

$$X(\lambda) = P^{+}\vec{x} + \lambda C,$$

where P^{+} is the pseudo-inverse of P ($P^{+} = P^{T}(PP^{T})^{-1}$).

In OLS, the intrinsics were estimated for each camera during calibration and extrinsics are known at each time due to the sensory data streamed from the manipulators. However, the rays backprojected from each camera might not intersect in the 3D space due to both systematic and random errors (see Figure 6).

The estimation of the 3D position of the target consists of the following steps. First, back-projection is used to find the vectors $\vec{u}$ and $\vec{v}$ which form the planes M_1M_2U and M_1M_2V with the angle α between them. Both vectors are then rotated around the axis $\vec{m}$ so that they lie in the same plane M_1M_2W: $\vec{u'} = R(\beta_1)\vec{u}$, $\vec{v'} = R(\beta_2)\vec{v}$. The rotation angles might be of the same value $\beta_1 = \beta_2 = \alpha/2$; however, to achieve higher precision the angles might be weighted by the trackers' beliefs b: $\beta_1 = \alpha\frac{b_2}{b_1+b_2}$, $\beta_2 = \alpha - \beta_1$. Finally, the intersection W of the vectors $\vec{u'}$ and $\vec{v'}$ is found.

If multiple camera units are used, 3D location can be estimated as the weighted centroid of the estimates computed by each pair of the camera units forming the base b_i (see Algorithm 1). The weights correspond to the angle between the baseline and the line intersecting the (estimated) position of the target and the baseline center, since this angle significantly affects the precision (see Section 4).

Since the 3D position estimation might be completely wrong occasionally, the position estimates are smoothed by the moving average computed over h consecutive estimates (h was empirically set to 10).

Algorithm 1: Estimation of the 3D position from n-views

Input: Set of bases $B = b_1, b_2, ..., b_N$.
Output: 3D position estimate T.

/* 3D location estimate disregarding weights */
foreach $b_i \in B$ **do**
  $\vec{t_i} = Estimate3DPosFrom2Views(b_i)$
end
$\vec{T'} = \frac{1}{N}\sum_{i=1}^{N}\vec{p}_i$
/* Weighted estimation of the 3D location. $\vec{bc_i}$ represents center of the baseline b_i */
foreach $b_i \in B$ **do**
  $w_i = \frac{e^{\vec{n_i}(\vec{T'}-\vec{bc_i})}}{\sum_{j=1}^{N} e^{\vec{n_j}(\vec{T'}-\vec{bc_j})}}$
end
$\vec{T} = \sum_{i=1}^{N} w_i\vec{t_i}$

7. Implementation and Experimental Results

The implementation is built on a robotic framework ROS[2] and a physical simulator Gazebo[3]. ROS was chosen for its wide support of hardware components and a seamless way to implement multi-process distributed system. The whole system was modeled and simulated in Gazebo (see Figure 7), which facilitated hardware-in-a-loop testing of the manipulators [35].

The system was tested in the real-world environment in the basic two-camera setup. The CSs were precisely positioned using the differential GPS sensor (achieving accuracy of ca 0.01 m) so that the base would be exactly 30 m long. The local heading was estimated by aiming the units on each other. The system

[2]Robot Operating System: http://www.ros.org
[3]Gazebo: http://gazebosim.org

Figure 7. A sample scene captured within the Gazebo simulator. The scenario consists of four CSs and one moving object (red ball). The simulated image streams are displayed on the right.

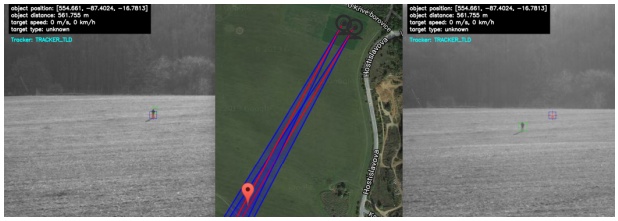


Figure 8. The two-camera setup, where a distant target is tracked by both CSs (left and right). The estimated position of the target is displayed in the map (center) in real time.

was tested against both static and dynamic targets, and in both cases only horizontal position was considered.

As for the static targets, nine landmarks with a priori known UTM coordinates (obtained from the cadastral map) and one target carrying an ordinary mobile GPS sensor were chosen (see Figure 8). The localization error, given as the Euclidean distance between the ground truth and the estimated locations, was compared with the estimated error (see Table 1). Note that both measured and estimated error follow the same trend (see Figure 9); however, the measured error is higher mainly due to the insufficient precision of calibration, stationing and rectification.

Table 1. The table shows the position as well as the localization error for each static target. The estimated error **est.** Δ is affected by the distance of the target and the angle α between the target and the base, and it was computed for the scenario where each CU makes random error $p = 4$ *px* (see Section 4). See also Figure 9 for graphical comparison of the estimated and the measured error.

object	dist. [m]	α [rad]	est. Δ [m]	Δ [m]
pillar1	91,92	0,38	0,20	4,41
pillar2	199,14	0,46	0,95	5,51
pillar3	285,01	0,48	1,93	11,73
pillar4	386,81	0,41	3,39	17,16
tree1	433,88	0,34	4,13	17,20
person	479,96	0,10	4,65	23,90
hide	526,86	0,77	7,57	22,77
tree2	634,46	0,35	8,56	28,33
mast	1379,67	0,33	41,24	34,21

The system was tested against one dynamic terrestrial target equipped with a mobile GPS sensor (a

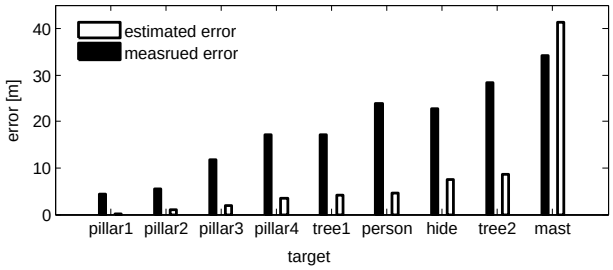


Figure 9. The plot displays both measured and estimated localization error for all static targets, which are sorted in ascending order with respect to the estimated error. The measured error is higher due to imprecise calibration, rectification and stationing.

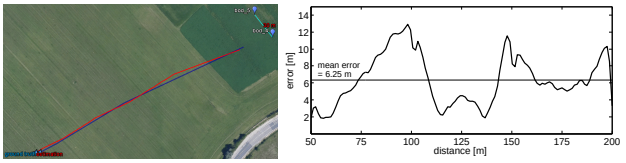


Figure 10. The comparison of the ground truth and estimated trajectory of a target moving in the distance range of ca 50–200 *m* (left). The error as the function of the distance of the target is also displayed (right). The system makes the average error of 6.25 *m*.

walking person). The target was tracked for 120 *s* and the estimated positions were captured and compared to the ground truth path (see Figure 10). On average the system achieved the precision of 6.25 *m*. Note that the position estimates oscillate around the ground truth trajectory, which is caused by the random error made by both trackers; the error, however, keeps in the specific range and reaches maximum of 13.35 *m*. The mean error is higher as compared to the estimated error (see Section 4), which is again caused by the systematic error (imprecise calibration, rectification and stationing).

8. Conclusion

This paper introduced a novel system capable of autonomous tracking and localization of distant moving targets using multiple cameras. The paper proposes precision analysis which aims on finding and alleviating the most prominent sources of error, as well as the methodology to calibrate and station all camera units.

The system utilizes a visual tracker based on the Bootstrap particle filter framework combining both visual and motion model of the target and positionable camera. The localization of the target uses the principle of triangulation, where both the belief of the tracker and the geometrical limitations given by the angle between the base and the target are incorporated into the final weighted estimate.

The system was tested in real world conditions against static and dynamic targets whose position was

known either from the cadastral map or captured by the GPS sensor. The localization precision follows the trend of diminishing accuracy of depth measurement and reaches slightly higher error then the theoretical model, namely due to the insufficiently precise calibration, rectification and stationing. This, however, can be improved by using more reliable hardware components and by performing the rectification procedure more thoroughly.

Though still in early development, the OLS system has great potential for being widely used as a passive, modular and highly portable substitute for the recently widely used radars for the applications ranging from automatic traffic control to national defense systems protecting the sensitive perimeters.

In the near future, the OLS system will be extended by the 3D environment reconstruction subsystem which should make the tracker predict occlusion and estimate more accurately the motion of the tracked target. Furthermore, more thorough tests will be carried out in order to spot the sources of error and reinforce the overall precision.

Acknowledgements

This work is supported by RCE systems s.r.o.[4] which provided the development space, necessary hardware sources, and consultations. I thank my Master's thesis supervisor, prof. Ing. Adam Herout, Ph.D., for guidance, and my adviser, doc. Ing. Vladimír Čech, CSc., for providing his insight into the project.

References

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

[2] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[3] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 1(4):321–331, 1988.

[4] W. Hu, X. Zhou, W. Li, W. Luo, X. Zhang, and S. Maybank. Active contour-based visual tracking by integrating colors, shapes, and motions. *IEEE Transactions on Image Processing*, 22(5):1778–1792, May 2013.

[5] Quentin Delamarre and Olivier Faugeras. 3d articulated models and multiview tracking with physical forces. *Computer Vision and Image Understanding*, 81(3):328 – 357, 2001.

[6] Cyrille Migniot and Fakhreddine Ababsa. 3d human tracking in a top view using depth information recorded by the xtion pro-live camera. In *ISVC (2)*, volume 8034 of *Lecture Notes in Computer Science*, pages 603–612. Springer, 2013.

[7] David Herman, Filip Orság, and Martin Drahanský. Object tracking in monochromatic video sequences using particle filter. In *7th Scientific International Conference - Enviromental Protection of Population*, pages 120–128. Karel Englis College Inc., 2012.

[8] Iain Matthews and Simon Baker. Active appearance models revisited. *Int. J. Comput. Vision*, 60(2):135–164, November 2004.

[9] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[10] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.

[11] Reza Oji. An automatic algorithm for object recognition and detection based on ASIFT keypoints. *CoRR*, abs/1211.5829, 2012.

[12] Sanjay Kumar Sahani, G. Adhikari, and B. Das. A fast template matching algorithm for aerial object tracking. In *Image Information Processing (ICIIP), 2011 International Conference on*, pages 1–6, Nov 2011.

[13] K. Wang and Z. Ren. Enhanced gaussian mixture models for object recognition using salient image features. In *Mechatronics and Automation, 2007. ICMA 2007. International Conference on*, pages 1229–1233, Aug 2007.

[14] Li Zhang and Ramakant Nevatia. Efficient scan-window based object detection using gpgpu. *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 0:1–7, 2008.

[15] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.

[4] Company RCE systems s.r.o. website - http://www.rcesystems.cz/

[16] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243 vol. 2, June 2005.

[17] SeungJong Noh and Moongu Jeon. *Computer Vision – ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part III*, chapter A New Framework for Background Subtraction Using Multiple Cues, pages 493–506. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[18] J. Lee, S. Lim, J. G. Kim, B. Kim, and D. Lee. Moving object detection using background subtraction and motion depth detection in depth image sequences. In *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on*, pages 1–2, June 2014.

[19] Georg Nebehay and Roman Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *Winter Conference on Applications of Computer Vision*, pages 862–869. IEEE, March 2014.

[20] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575, May 2003.

[21] Zhimin Fan, Ying Wu, and Ming Yang. Multiple collaborative kernel tracking. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 502–509 vol. 2, June 2005.

[22] M. Tang and J. Feng. Multi-kernel correlation filter for visual tracking. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 3038–3046, Dec 2015.

[23] Michael D. Breitenstein and Fabian Reichlin. Robust tracking-by-detection using a detector confidence particle filter. In *IEEE International Conference on Computer Vision*, October 2009.

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

[25] E.V. Cuevas, D. Zaldivar, and R. Rojas. *Kalman Filter for Vision Tracking*. Freie Universität Berlin, Fachbereich Mathematik und Informatik / B: Fachbereich Mathematik und Informatik. Freie Univ., Fachbereich Mathematik und Informatik, 2005.

[26] Joo-Ho Lee, Noriaki Ando, and T. Yakushi. 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.

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

[28] 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.

[29] Mariana Rampinelli and Vitor Buback Covre. An intelligent space for mobile robot localization using a multi-camera system. *Sensors*, 14(8):15039, 2014.

[30] K. Morioka and H. Hashimoto. Appearance based object identification for distributed vision sensors in intelligent space. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 199–204 vol.1, Sept 2004.

[31] J.R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. A series of books in physics. University Science Books, 1997.

[32] B. Cyganek. *An Introduction to 3D Computer Vision Techniques and Algorithms*. John Wiley & Sons, 2007.

[33] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[34] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[35] 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.

Appendix D

Poster - Excel@FIT 2016

31

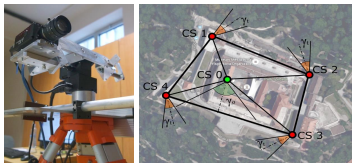
Optical Localization of Very Distant Targets in Multi Camera System

Jan Bednařík

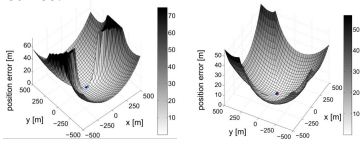
FIT, BUT: jan.bednarik@hotmail.cz
RCE systems s.r.o.: jan.bednarik@rcesystems.cz

System Overview

This poster presents a semi-autonomous passive multi-camera system for tracking and localizing the distant objects, which is based merely on ordinary RGB cameras. The main building block is a camera station, a standalone unit consisting of a surveying tripod, a camera, a Pan Tilt unit and hardware components for estimating geographical coordinates. The system is designed to work with an arbitrary number of stations which should be positioned so as to form approximately regular polygon with long enough bases.

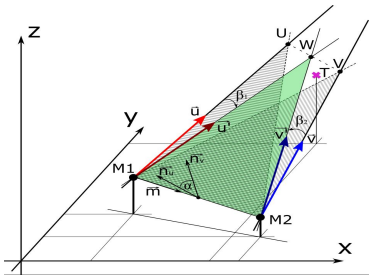


Stereoscopic systems are affected by a phenomenon of diminishing accuracy of depth measurement (known from the domain of stereoscopic systems) with increasing distance of the target. However, the OLS does not conform to the canonical stereo setup as all cameras can rotate freely. Furthermore multiple cameras can be utilized which alleviates the geometrical limitations of the two-camera setup. In the multi-camera setup the subset of two cameras yielding the lowest geometrical error can be chosen or weighted estimates of all cameras can be utilized.



Target Localization

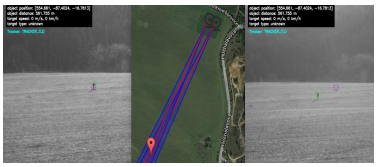
The autonomous tracking uses the implementation of the visual tracker combining the background subtraction, motion model and object model in the particle filter framework. This approach can even cope with the moving cameras and thus is suitable for the OLS. The target is represented as a rectangular template (consisting of gray-scale intensity values) The Bootstrap particle filter is used to generate and evaluate candidate positions of the target.



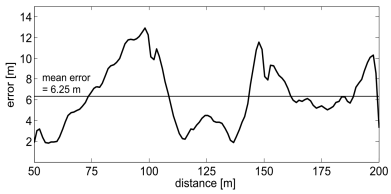
In OLS, the intrinsics were estimated for each camera during calibration and extrinsics are known at each time due to the sensory data streamed from the manipulators. The estimation of the 3D position of the target uses back-projection to find the 3D direction vectors (aiming on the target). Due to the random error the vectors do not intersect, thus they are projected to a common plane and the intersection is computed. If more then three camera units are used, 3D location can be estimated as the weighted centroid of the estimates computed by each pair of the camera units forming a unique base.

Experiments

The system was tested in the real-world environment in the basic two-camera setup. The camera stations were precisely positioned using the differential GPS sensor (achieving accuracy of ca 0.01 m) so that the base would be exactly 30 m long. The local heading was estimated by aiming the units on each other. The system was tested against both static and dynamic targets, and in both cases only horizontal position was considered.



As for the static targets, nine landmarks with a priori known UTM coordinates (obtained from the cadastral map) and one target carrying an ordinary mobile GPS sensor were chosen. The system was tested against one dynamic terrestrial target equipped with a mobile GPS sensor (a walking person) as well. The target was tracked for 120 s and the estimated positions were captured and compared to the ground truth path; on average the system achieved the precision of 6.25 m.



Excel@FIT 2016 RCE SYSTEMS