Czech Technical University in Prague

Faculty of Electrical Engineering

Bachelor Thesis

Tomáš Nouza

Localization of mobile robots for exploration using cloud services

Department of Cybernetics

Supervisor: **Ing. Michal Reinštein, Ph.D.**

Prague, 2012

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 1. 6. 2012

[illegible]
Podpis autora práce

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student:	Tomáš N o u z a
Study programme:	Cybernetics a Robotics
Specialisation:	Robotics
Title of Bachelor Project:	Localization of Mobile Robots for Exploration Using Cloud Services

Guidelines:

The aim is to analyze the *state of the art* of the currently available cloud services used for online localization of mobile robots for exploration.

Based on this analysis, a robust algorithm for localization of the mobile robot developed as part of the NIFTI project (http://www.nifti.eu) will be proposed.

Localization must include all the targets detected by the robot and their proper placement into the map. The algorithm will then be implemented for Robot Operating System (ROS) and its functionality and reliability will be experimentally verified under real outdoor conditions.
For the actual localization of the robot following sensors are available: GPS receiver, robot odometry, MTI-G Xsens inertial measurement unit.

Bibliography/Sources:

[1] The NIFTi project (October 3, 2011), http://www.nifti.eu
[2] Hlaváč, V., Sedláček, M.: Zpracování signálů a obrazů 1. vyd. Praha, ČVUT, 2000
[3] Thrun, S., Burgard, W., Fox, D., Probabilistic robotics, 2005

Bachelor Project Supervisor: Ing. Michal Reinštein, Ph.D.

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

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
Head of Department

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

[illegible]

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, December 13, 2011

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student:	Tomáš N o u z a
Studijní program:	Kybernetika a robotika (bakalářský)
Obor:	Robotika
Název tématu:	Lokalizace mobilních průzkumných robotů s využitím cloudových služeb

Pokyny pro vypracování:

Cílem práce je provést analýzu *state of the art* v současné době dostupných cloudových služeb využitelných pro online lokalizaci mobilních průzkumných robotů.

Na základě této analýzy bude navržen robustní algoritmus pro lokalizaci pásového robota vyvíjeného v rámci projektu NIFTi (http://www.nifti.eu).

Lokalizace musí zahrnovat i cíle detekované robotem a jejich zanesení do mapy. Algoritmus bude následně implementován pro Robot Operating System (ROS) a jeho funkčnost a spolehlivost bude experimentálně ověřena v reálných venkovních podmínkách. Pro lokalizaci robota je možné využít data z následujících senzorů: přijímač systému GPS, odometrie robota, MTI-G Xsens inerciální měřicí jednotka.

Seznam odborné literatury:

[1] The NIFTi project (October 3, 2011), http://www.nifti.eu
[2] Hlaváč, V., Sedláček, M.: Zpracování signálů a obrazů 1. vyd. Praha, ČVUT, 2000
[3] Thrun, S., Burgard, W., Fox, D., Probabilistic robotics, 2005

Vedoucí bakalářské práce: Ing. Michal Reinštein, Ph.D.

Platnost zadání: do konce zimního semestru 2012/2013

[illegible]

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

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

[illegible]

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 13. 12. 2011

Abstract

The aim of this thesis is to analyze the state-of-the-art of the currently available cloud services used for online localization of mobile robots performing urban search and rescue missions. Based on this analysis, a robust algorithm for localization of the mobile robot developed as a part of the Natural human-robot cooperation in dynamic environments (NIFTi) project was implemented in the Robot Operating System (ROS). The localization includes mainly the trajectory information in global world coordinates as well as markers corresponding the detected objects of interest. The result is visualized in the online map application.

To provide this localization algorithm with the suitable navigation data, a landmark based navigation algorithm was introduced. This relies on the computer vision and is based on the matching feature points between the actual camera view and the previously composed reference cloud of landmarks created locally (extension to the global navigation is discussed). For this purpose, the state-of-the-art techniques such as Structure from Motion, Speeded-Up Robust Features, and the Fast Approximate Nearest Neighbors were used.

Abstrakt

Cílem této práce je analýza state-of-the-art v současné době dostupných cloudových služeb využitelných pro online lokalizaci mobilních průzkumných robotů, používaných při pátracích a záchraných misích v obydlených oblastech. Na základě této analýzy byl implementován robustní algoritmus pro lokalizaci pásového robota, vyvíjeného v rámci projektu NIFTi (Přirozená spolupráce robotů a lidí v proměnlivých prostředích), pro prostředí ROS (Robot Operating System). Tato lokalizace zahrnuje převážně trajektorii robota ve světových souřadnicích a robotem detekované objekty. Výsledek je zobrazen do online mapové aplikace.

Aby byla tomuto lokalizačnímu algoritmu poskytnuta náležitá navigační data, byl navržen navigační systém založený na orientačních bodech. Tento systém, založený na metodách počítačového vidění, spočívá v přiřazení charakterických bodů mezi aktuálním pohledem kamery a předem spočítaným referečním mračnem orientačních bodů (rozšíření pro globální využití je diskutováno). Pro tento účel byly využity nejmodernější techniky jako je Structure from Motion, Speeded-Up Robust Features a Fast Approximate Nearest Neighbors.

Acknowledgements

I would gladly like to thank my supervisor Ing. Michal Reinštein, Ph.D. for his knowledge support and permanent flogging me to work. I also want to express special thanks to Mgr. Michal Havlena for his magics with SfM, which saved me a lot of time in dead ends. I would also like to thank my family and girlfriend for their patience, soothing and moral support.

Contents

List of Figures

Chapter 1

Introduction

In this work we will look closer to some state of the art cloud services and their application to the map localization of the search and rescue robot developed as a part of the NIFTi project. NIFTi (Natural human-robot cooperation in dynamic environments) [2] is an European project FP7-ICT-247870 focusing on Urban Search & Rescue (USAR) missions. It concerns human-robot cooperation such as the localization of victims trapped in confined spaces due to natural disasters, building or mine collapse, transportation accidents, etc. The emphasis is put on minimizing task load for human and optimizing workflow. Sharing of information via various interfaces is crucial for any human-robot cooperation and hence map visualization of the robot trajectory and other objects of interest, that robot has detected during mission (cars, victims, etc.), is key for the robot operators.

For this reason this thesis aims to: first, provide these information on the internet to be available independently on where the user is and what device he or she uses (PC, Mac, tablet, etc.), and second, suggest and evaluate one of the possible ways of determining robot position with respect to the global world map. In the section 2.1 there is closely described a robotic platform and its sensor equipment, that is used for robot localization (orientation and position estimation), mapping and for the detection of objects. For this purpose I have implemented a client side Python script, that records these data and uploads them via a web application to a cloud storage, from which they are used for visualization in an on-line map application as described in the section 4.1.

These days the internet is full of map applications which differ in area coverage, amount of details and functions of user API (Application Interface). I had compared several of them and then decided to use the Google Maps due to its user-friendly documentation, worldwide coverage and direct connection to other Google products like a database storage. More information about Google products used in this work can be found in the section 2.2.3.

The same method as mentioned above is used in Google My Tracks [4] where data from user's GPS (Global Position System) receiver are uploaded as tracks to a cloud storage and visualized in Google Maps. Tracks can be shared with other users and also information like average speed, elevation profile, etc. are provided.

This product is not used in this work because it does not allow to visualize detected objects and any other important non-trajectory markers.

Visualization of the robot position is useless if the robot position is unknown. There are several possible ways how to determine it. First is the dead reckoning navigation based on data fusion of proprioceptive sensors (odometry, inertial measurements). Second, a large group of computer vision based approaches using either camera images (visual odometry, VSLAM, etc.) or laser point cloud measurement. Regrettably these localization systems provide only relative positions (translation and rotation) with respect to the initial position. It also has an accumulative error, which increases with traveled distance. For drawing anything on maps, the position with respect to a global geodetic system is necessary. This can be acquired by any GNSS (Global Navigation Satellite System) like GPS. Unfortunately, satellites are unreachable in many USAR mission locations (e.g. narrow street, building, tunnels, etc.). Fourth solution is to use an image matching based navigation, which uses a database of images with known positions and compares it with an actual camera view.

Regarding the image based navigation, Robertson and Cipolla [5] announced a system that allows to navigate in an urban environment using just a single camera and a database containing building facades. The accuracy was in the street scale and it had problems in locations with similar buildings. Different system [6] tries to match camera images to an aerial image depending on specific road features like road marking. Senlet and Elgammal [7] introduced a system combining stereo visual odometry, satellite images and route map working even on rural routes with no marking but still working only on the routes. After reading [8] I decided to use landmark based navigation as described in [9] and in [10].

This solution uses a 3D cloud of visual landmarks and compares them with landmarks detected in the actual robot camera view. For comparison the Fast Approximate Nearest Neighbors (FLANN) [11] algorithm is used, which is in orders of magnitude faster than exact linear search, while still providing sufficient near optimal accuracy. More about FLANN is in the section 3.3. If there are three or more credible matches, robot position can be calculated as described in the section 3.5. For describing landmarks is advantageous to use Speeded-Up Robust Features (SURF) [3] because it is faster than other detectors, while its performance is comparable or sometimes even better. For more informations about SURF see the section 3.2. To generate the 3D cloud in this work is used Structure from Motion (SfM) algorithm presented in [12]. Benefit is that the 3D position of every SURF in the cloud is transformable to the global navigation frame if at least three points have known positions. This transformation is described in the section 3.4. Disadvantage is that the location has to be visited a priori to create this reference 3D cloud. Whole implementation of this localization method is in the section 4.2.

In the section 6.2 the improvement of this method is mentioned. It uses an online database Google Street View [13] as a source for reference cloud generator, which works even in locations previously not visited by the robot, but provides worse accuracy in comparison to the solution mentioned above due to low density of images per distance.

Chapter 2

Resources

In order to provide the reader with some important background information, this chapter briefly describes the robotic platform and software used in this work.

2.1 Hardware

Fig. 2.1 shows an Unmanned Ground Vehicle (UGV) [1] designed by BlueBotics[1], which is currently used in the NIFTi project. For the better imagination of the robot size, Fig. 2.2 shows dimensioned blueprint of the robotic platform. For its localization the robot uses the rotating SICK LMS-151 laser scanner, which can provide a 3D point cloud composed of rotated planar scans, the Point Grey Ladybug 3 omnicamera, and the X-sens MTI-G inertial measurement unit (IMU) with a GPS module. Finally, every motor has a position encoder for odometry measurement. This means that the speed of both belts is measured independently. Combination of all these sensors provides sufficient position and orientation accuracy of the robot. Experiments showed that the GPS module has a very poor precision and reliability. It has accuracy of 5 to 30 meters (the longer time spent on one place the more accurate measurement), but in urban environment signal precision often drops due to multipath error or signal outages.

Regarding networking and communication the robot is equipped with two wi-fi antennas. The large one provides connection to a base station, from which the robot can be teleoperated. It has maximum performance of 500 mW and enough capacity to transfer up to 10 images per second from all 6 cameras in 2 Mpix resolution each. The other one is a standard USB wi-fi module that provides an internet network connection which is crucial for cloud services. Unfortunately, this is often not available in USAR missions so in future it will be replaced or extended by GSM module.

Inside the robot there is a embedded Kontron® PC equipt with the Intel® Core™2 Quad Mobile processor (Penryn) Q9100, which has sufficient computing power to realtime image processing. Powered on battery the robot can operate

[1]Swiss robotics company http://www.bluebotics.ch/

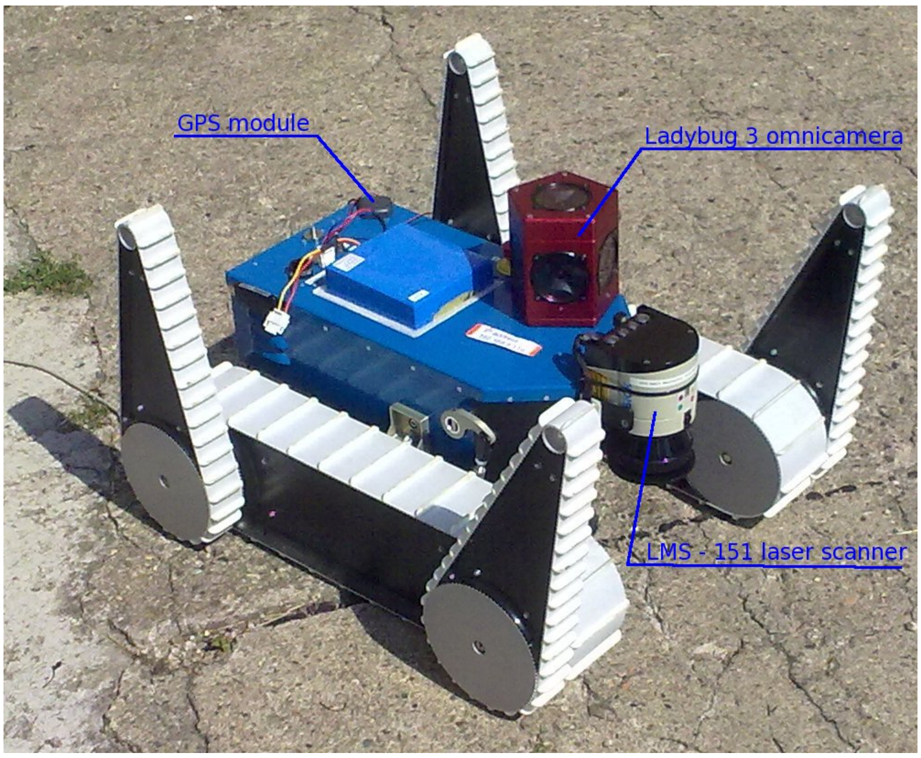


Figure 2.1: BlueBotics UGV [1] used in NIFTi project [2]

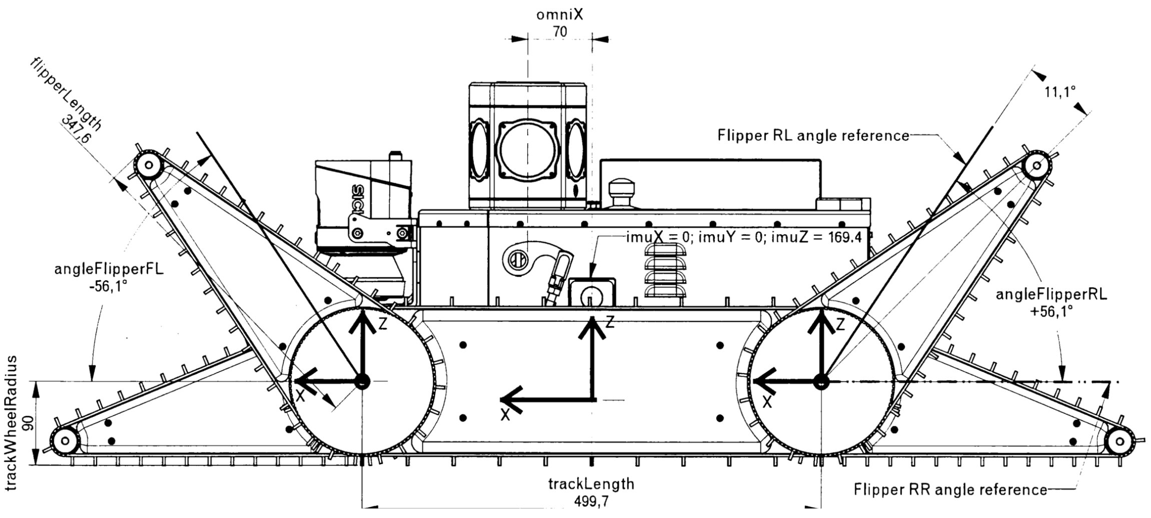


Figure 2.2: Side view of the robot

nearly 4 hours. For longer missions there is a possibility of a hot-swap battery replacement allowing uninterrupted operation.

2.2 Software

In this section the third party software used in this work is described. It is divided to open source software developed by Willow Garage [14], software developed by my colleges on CTU in Prague, and freely available software developed by Google.

2.2.1 Willow Garage software

Willow Garage is a robotics research lab and technology incubator devoted to developing hardware and open source software for personal robotics applications. It is a part of the open-source robotics community and is currently determining the state-of-the-art in personal robotics. Recently they founded the Open Source Robotics Foundation (OSRF) [15] which is maintaining the ROS (see bellow). The OSRF was awarded by software contract with the recently announced DARPA Robotics Challenge[2].

Robot Operating System (ROS) [16] is a multiplatform meta-operating system for robots. ROS provides an automatic management system for message-passing between nodes [17] using TCP/IP protocol. A node is an executable that uses the ROS to communicate with other nodes using topics [18]. Topics are named buses over which nodes exchange the data in the form of messages. When using a computer network connected to the robot(s), the ROS automatically passes messages, so any node running on one computer can interact with all the other nodes in that network. No code implementation is needed since it is open source package based finished solution.

OpenCV (Open Source Computer Vision Library) [19] is a cross-platform library of programming functions aimed at real time computer vision, originally developed by Intel, currently maintained by the Willow Garage. It has more than 2500 optimized algorithms covering all developer's needs. In this work is mainly used for image manipulation, SURF detection and FLANN matching.

2.2.2 CTU software

The Inertial navigation system aided by odometry (INSO) [20] is a ROS node for the Extended Kalman filter based data fusion of odometry and inertial data (acceleration and angular rates) provided by IMU. This node provides a reliable dead reckoning navigation and hence is used in this work as a standard data input for position information. In the future it will be extended by data from the visual odometry which will even improve the long-term accuracy.

[2]Challenge specification: https://www.fbo.gov/utils/view?id=74d674ab011d5954c7a46b9c21597f30

Computer vision based detectors are implemented as ROS nodes, which process the camera images in realtime and provide the position of detected objects in the robot coordinate frame. Every object, that is being detected, has unique ID enabling tracking of changes such as change in position. In this work, only the car detector [21] is used but the code is capable to use any detector that provides data mentioned above (position of unique ID labeled objects). There are currently two detectors available, the car detector (shown in the Fig. 2.3) and the victim detector. Other detectors such as the sign detector are under development.[3]

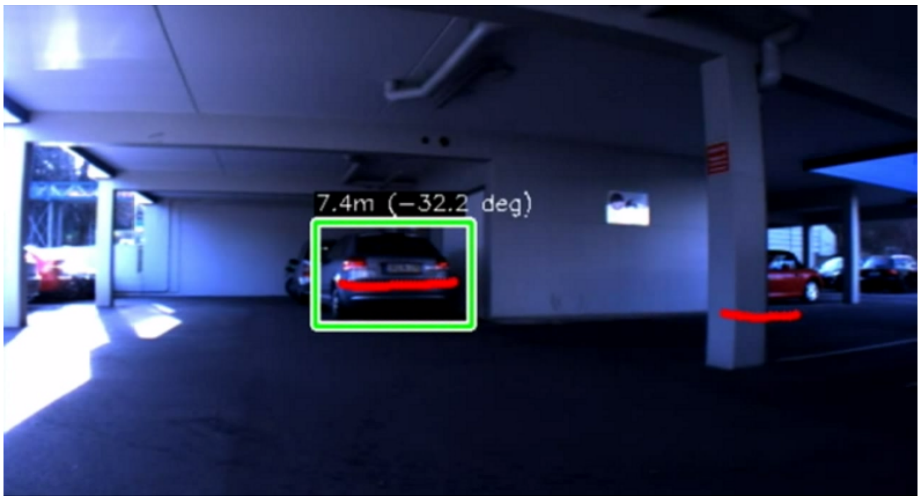


Figure 2.3: Example of car detector. Red line represents rangefinder laser beam, which is used to filter false positive detections.

SfM Web Service [23] is a web application developed by a team lead by Tomas Pajdla that provides a remote access to the 3D reconstruction systems developed in Center for Machine Perception, FEE, CTU Prague [24].

The system provides tools like:

- Large-Scale Sequential or Unordered Data Processing for SfM reconstruction from datasets containing thousands of perspective, fish-eye, or panoramic images
- Multi-View Dense 3D Reconstruction for creating colored or textured surface models of the scenes from perspective datasets once the cameras are calibrated

In this work, the SfM Web Service is used for generating a 3D reference cloud of SURF descriptors from panoramic images taken by the robot. Colored example of this cloud is shown on Fig. 2.4. For comparison Fig. 2.5 shows the same area taken from Google Maps. Red dots are reconstructed camera positions of each image. Although computation time required for generating this cloud is in order of

[3]More demos of detectors can be found in [22]

hours, in the final implementation the result is used for relatively fast localization algorithm.

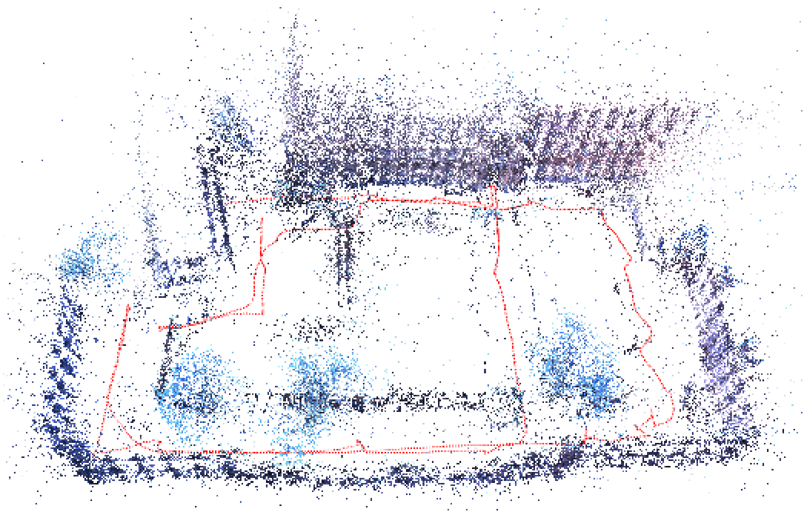


Figure 2.4: CTU quad represented by a colored 3D cloud of landmarks composed from panoramic images. Red dots represent reconstructed camera positions of each image.

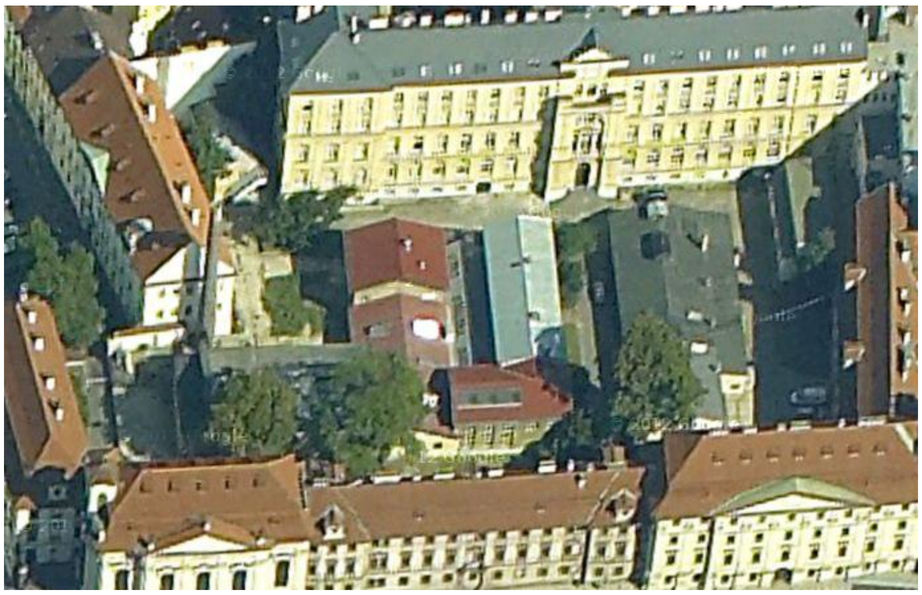


Figure 2.5: CTU quad from Google Maps.

2.2.3 Google products

The Google Maps API [25] is a powerful tool for visualizing almost everything into online maps. It has worldwide coverage, many layers and wide support for many devices. Data structures are described in KML (Keyhole Markup Language)

[26], which is XML like format for describing geographical data. It is an international standard maintained by the Open Geospatial Consortium, Inc. (OGC) [27]. This standard is used in all Google services for geographical input.

The Google Fusion Tables (GFT) [28] is a modern data management and publishing web application that makes it easy to host, manage, collaborate on, visualize, and publish data tables online. Data are accessed in a similar way like standard SQL database or from a web page. If they are in the KML format, they can be visualized through the Google Maps. Tables can be shared the same way as any other document using Google Drive, where they are implicitly stored.

Google Earth (GE) [29] is a free virtual globe, map and geographical information program. It maps the Earth by the superimposition of images obtained from satellite imagery, aerial photography and GIS 3D globe. A strong feature of the GE is the ability to open and process KML files which could also be linked to other KML files (for example placed on some web storage) that can even change dynamically. Although GE downloads map data from internet, it uses relatively large cache which enables off-line visualization of KML files on once visited locations.

Chapter 3

Theory

In this chapter is described theoretical analysis of methods used in this work. It contains description of two main coordination frames used for navigation, and a conversion between them (see section 3.1), also as brief introduction to the state-of-the-art matching method based on the SURF (section 3.2) and FLANN (section 3.3). Last but not least is the transformation of a point cloud from coordination frame with unknown scale to the global navigation frame based on the knowledge of exact position of few points from given point cloud (see the section 3.4). Finally, there is described an estimation of observer's position from the given view angles between points with known position (see the section 3.5).

3.1 Conversion of coordinates

Most of robot software works in the robot navigation frame with the:

- origin at the robot starting location
- X-axis oriented in the direction of the robot's forward motion at the moment of start
- Y-axis points to the left of the X-axis (perpendicular to the X-axis)
- Z-axis points up (perpendicular to the X-Y plane)

meanwhile the mapping software uses the Earth-centered Earth-fixed (ECEF) frame where the Earth surface is approximated by the WGS84 ellipsoid [30, p. 51-52]. This ellipsoid is defined by its origin located at the Earth's center of mass, semi-major axis $a = 6\,378\,137\,\mathrm{m}$ and semi-minor axis $b = 6\,356\,752,314\,\mathrm{m}$. Every point on the ellipsoid is determined by its latitude (angle from equator with positive value to the north), longitude (angle from International Reference Meridian with positive value to the east) and altitude. As only planar maps are used in this work, it is sufficient to labour only with the latitude and the longitude.

To allow conversions between these frames it is necessary to assign two points from both frames, or only one point and a rotation between coordinate systems. In this work, the latter variant, which depends on the robot's default heading to the east, is used. The east orientation was selected because the default orientation of the CTU robot in the laboratory is exactly to the east. The starting position is set manually by the robot operator (for example from a map or using GPS device) in the ECEF frame (latitude, longitude pair) as parameters O_{lat} and O_{lon}.

Point $\mathbf{P} = (P_{lat}, P_{lon})$ is then calculated from point $\mathbf{P}' = \left(P'_x, P'_y\right)$ as:

$$P_{lat} = O_{lat} + \frac{P'_y}{b} \cdot \frac{180}{\pi} \tag{3.1}$$

$$P_{lon} = O_{lon} + \frac{P'_x}{a \cdot \cos\left(O_{lat} \cdot \frac{180}{\pi}\right)} \cdot \frac{180}{\pi} \tag{3.2}$$

3.2 Speeded-Up Robust Features

SURFs are in this work used for describing keypoints in landmark based navigation. It is a scale and in-plane rotation invariant detector and descriptor with strong repeatability and robustness. The common application of SURF works as follows:

1. "interest points" (such as corners, blobs and T-junctions) are selected (detection),
2. the neighbourhood of every interest point is represented by a feature vector (description),
3. keypoints with closest Euclidean distances (above a given threshold) are matched.

Concerning the photometric deformations, a simple linear model with the bias (offset) and contrast (scale factor) change is assumed. Neither the detector nor the descriptor use colour information. The detection is based on the Hessian-matrix [31] approximation using integral images allowing fast computation of box type convolution filters, which drastically reduce the computation time.

As the SURF is used for the image matching based on the Euclidean distance between the descriptor vectors, the dimension of the descriptor has a direct impact on the computational time. Here lies the main benefit of using SURF against still widely used SIFT (Scale Invariant Feature Transform) [32]. SIFT is a predecessor of SURF based on the local oriented gradients around the point of interest, which uses 128-dimensional vector meanwhile the SURF, based on the first order Haar wavelet [33] responses in horizontal and vertical direction, provides descriptor of dimension 64. In [3] is the comparison that shows similar results in precision for nearest neighbour matching for both SURF and SIFT. Nearest neighbour is an optimization problem for finding closest points in n-dimensional spaces. In many cases, the distance is measured by Euclidean or Manhattan distance [34, p. 94, 313].

3.3 Fast Approximate Nearest Neighbors

As mentioned above, SURF landmarks are detected in the camera view and matched with the reference cloud of landmarks. Simple linear search, necessary to do the matching, is too costly since the cloud contains approximately 40 000 keypoints. Fortunately, there is no need for accurate Euclidean distance, in case only selection of points, whose distance is closer than given threshold, is required. The FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. For this purpose it contains a collection of algorithms, such as multiple randomized kd-trees [35] or hierarchical k-means trees with a priority search order [11]. Algorithm and its optimal parameters are selected automatically depending on the dataset. In this work, the FLANN implementation from the OpenCV was used.

3.4 Transformation of 3D map to the global coordinate frame

For full usage of the reference cloud of SURF landmarks gathered using the SfM Web Service (see 2.2.2), transformation of every point from this cloud to the global coordinate frame is crucial. To perform this transformation, at least three points must have known position in both frames. Even this is the minimum (other points are interpolated/extrapolated), the more points are known, the more precise transformation is obtained.

As this problem is encountered in many applications of computer vision, Shinji Umeyama [36] provided an optimal solution, which always gives the correct transformation parameters $\boldsymbol{R}$ (rotation), $\boldsymbol{t}$ (translation) and c (scaling) even when the data are corrupted. Let the $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots \boldsymbol{x}_n\}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots \boldsymbol{y}_n\}$ be corresponding point patterns in m-dimensional space. The minimum value ε^2 of the mean square error

$$e^2(\boldsymbol{R}, \boldsymbol{t}, c) = \frac{1}{n} \sum_{n}^{i=1} \|\boldsymbol{y}_i - (c\boldsymbol{R}\boldsymbol{x}_i + \boldsymbol{t})\|^2 \tag{3.3}$$

of these two point patterns with respect to the similarity transformation parameters ($\boldsymbol{R}$, $\boldsymbol{t}$ and c) is given as follows [36, p. 378]:

$$\epsilon^2 = \sigma_y^2 - \frac{\operatorname{tr}(\boldsymbol{D}\boldsymbol{S})^2}{\sigma_x^2} a = b \tag{3.4}$$

$$\boldsymbol{\mu}_x = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i \tag{3.5}$$

$$\boldsymbol{\mu}_y = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y}_i \tag{3.6}$$

$$\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{x}_i - \boldsymbol{\mu}_x\right\|^2 \tag{3.7}$$

$$\sigma_y^2 = \frac{1}{n}\sum_{i=1}^{n}\left\|\boldsymbol{y}_i - \boldsymbol{\mu}_y\right\|^2 \tag{3.8}$$

$$\sum_{\boldsymbol{xy}} = \frac{1}{n}\sum_{i=1}^{n}\left(\boldsymbol{y}_i - \boldsymbol{\mu}_y\right)\left(\boldsymbol{x}_i - \boldsymbol{\mu}_x\right)^T \tag{3.9}$$

and let a singular value decomposition [37] of $\sum_{\boldsymbol{xy}}$ be $\boldsymbol{UDV}^T$ where $\boldsymbol{D} = \operatorname{diag}(d_i), d_1 \geq d_2 \geq \cdots \geq d_m \geq 0$, and

$$\boldsymbol{S} = \begin{cases} \boldsymbol{I} & \text{if} \quad \det\left(\sum_{\boldsymbol{xy}}\right) \geq 0 \\ \operatorname{diag}(1, 1, \cdots, 1, -1) & \text{if} \quad \det\left(\sum_{\boldsymbol{xy}}\right) \leq 0 \end{cases} \tag{3.10}$$

$\sum_{\boldsymbol{xy}}$ is the covariance matrix of $\boldsymbol{X}$ and $\boldsymbol{Y}$, $\operatorname{tr}()$ is the trace of a square matrix (sum of the elements on the main diagonal), $\det()$ is determinant of a matrix, $\operatorname{diag}()$ is a diagonal of a matrix, $\boldsymbol{\mu}_x$ and $\boldsymbol{\mu}_y$ are mean vectors of $\boldsymbol{X}$ and $\boldsymbol{Y}$, and σ_x^2, σ_y^2 are variances around the mean vectors of $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively.

The singular value decomposition is a process of factoring $m \times n$ real or complex matrix $\boldsymbol{M}$ to the form of $\boldsymbol{M} = \boldsymbol{UDV}^*$ where $\boldsymbol{U}$ is an $m \times m$ real or complex unitary matrix[1], $\boldsymbol{D}$ is an $m \times n$ rectangular diagonal matrix with nonnegative real numbers on the diagonal and $\boldsymbol{V}^*$ (conjugate transpose of $\boldsymbol{V}$) is an $n \times n$ real or complex unitary matrix.

When $\operatorname{rank}\left(\sum_{\boldsymbol{xy}}\right) \geq m - 1$, the optimum transformation parameters are determined uniquely as follows:

$$\boldsymbol{R} = \boldsymbol{USV}^T \tag{3.11}$$

$$\boldsymbol{t} = \boldsymbol{\mu}_y - c\boldsymbol{R}\boldsymbol{\mu}_x \tag{3.12}$$

$$c = \frac{1}{\sigma_x^2}\operatorname{tr}(\boldsymbol{DS}) \tag{3.13}$$

where $\boldsymbol{S}$ in (3.11) must be chosen as:

$$\boldsymbol{S} = \begin{cases} \boldsymbol{I} & \text{if} \quad \det(\boldsymbol{U})\det(\boldsymbol{V}) = 1 \\ \operatorname{diag}(1, 1, \cdots, 1, -1) & \text{if} \quad \det(\boldsymbol{U})\det(\boldsymbol{V}) = -1 \end{cases} \tag{3.14}$$

[1] $\boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{UU}^* = \boldsymbol{I}$

when rank $\left(\sum_{\boldsymbol{xy}}\right) = m - 1$.

Using the parameters obtained above ($\boldsymbol{R}$, $\boldsymbol{t}$ and c) point $\boldsymbol{X}$ is calculated from the point $\boldsymbol{Y}$ as:

$$\boldsymbol{X} = \begin{bmatrix} c\boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \boldsymbol{Y} \tag{3.15}$$

where $\begin{bmatrix} c\boldsymbol{R} & \boldsymbol{t} \end{bmatrix}$ has dimension 3x4 and $\boldsymbol{Y}$ is in homogenous coordinates.

I have implemented this transformation in Python for the purpose of this work. $\boldsymbol{Y}$ represent points in the reference cloud frame, $\boldsymbol{X}$ represent points in the global coordinate frame.

3.5 Robot localization in a 3D map

Final step for performing the localization is to determine the robot position from matched SURF points between actual camera view and the 3D cloud. Fig. 3.1 shows an example of this situation from the robot perspective, where the red lines determine region of interest for the SURF search. Bellow this region is mainly the robot's body, above this region is too much distortion for credible matching. As we can see, 18 SURFs (green dots) were matched with their correspondences in the reference cloud.

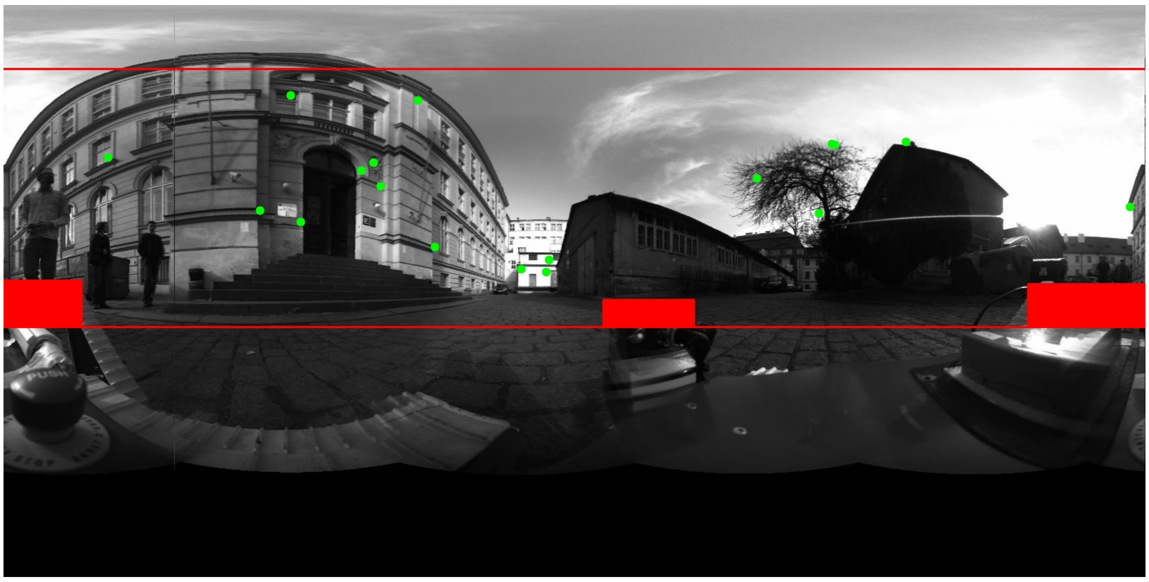


Figure 3.1: Matched SURF [3] points (green dots) from 3D cloud

This spherical image was composed from 6 cameras, which original image is on the Fig. 3.2. Spherical projection benefit is that the image width is equal to the 2π radian, image height to the π radian, and hence the distance between two points in pixels is their parallax in radians. This is useful in computing the camera position from known position of selected points and their parallax. This problem is encountered in many applications of computer vision and is called the Perspective-Three-Point Problem (P3P). A simple and straightforward solution to the P3P problem is the direct linear transformation (DLT) [38]. The description and implementation of this solution is above the scope of this bachelor thesis.

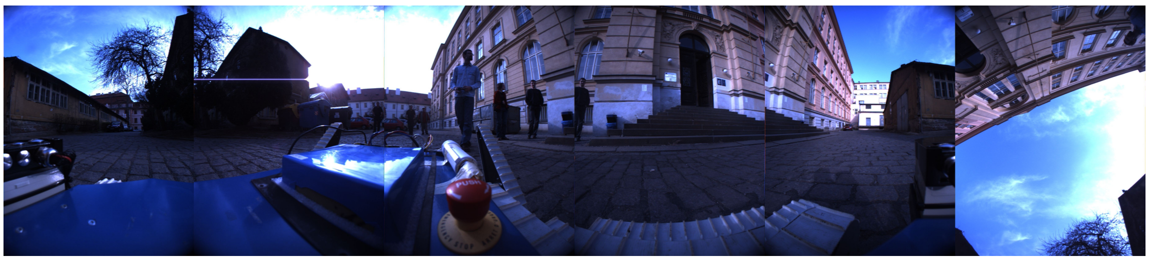


Figure 3.2: Original robot camera image

Chapter 4

Implementation

In this chapter an implementation of algorithms proposed in this work is introduced. The first is a ROS node called Googlemaps node 4.1, which is used for visualization of the robot trajectory and other detected objects of interest, that the robot has met during its mission, to the Google Maps. The second is a web application for the Google App Engine [39], which serves as a cloud buffer for the first node to optimize data flow over the network. In the section 4.2 is closely described the navigation method based on the reference 3D cloud of landmarks, which was generated from a sequence of panoramic images.

4.1 Visualization using Googlemaps node

Visualization of data gathered by the robot using web application is a complex task. The whole process can be split into three layers as shown on the Fig. 4.1. Position data from the INSO node and the data from detectors (both described in 2.2.2) are uploaded from the robot to a cloud buffer (for details see 4.1.2), where they are processed and then stored in the Google Fusion Tables, where they can be visualized trough the Google Maps.

This application requires internet connection to assure proper functionality. Since the internet connection during USAR mission is of questionable reliability, the loss of the network connection must be expected and accounted for such that the impact of this outage is minimized. This is done by parallel saving of the uploaded data to files, from which they can be restored and uploaded to the GFT using a recovery script also implemented in this work.

4.1.1 Robot applications

Object detectors and the navigation node (both described in 2.2.2) are running locally on the robot. Every detected object is evaluated by its unique ID whenever it was previously detected and if not, its position is saved in the global navigation frame. As the detectors provide positions of objects in the robot navigation

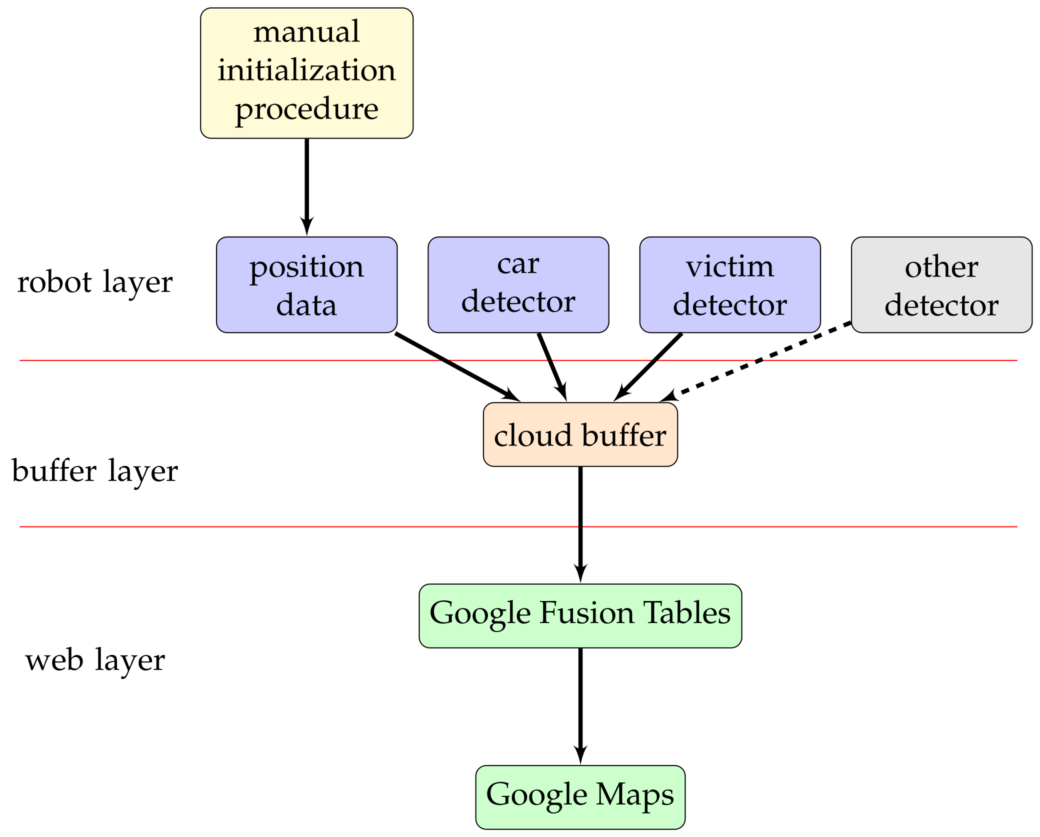


Figure 4.1: Process diagram showing the division of the visualization process into different computational layers.

coordinate frame, the conversion (described in the section 3.1) is necessary. The same conversion must be done to the position data when using dead reckoning navigation system. It is important, that the robot starting position and orientation in the global navigation frame is entered manually before the program starts to perform correct conversion.

The INSO node (described in 2.2.2) provides position messages up to 100 times per second, which means that the relative position changes are very low with respect to the motion dynamic of the robot. Since the Google maps have limited resolution, I have implemented an artificial constraint such that only the position change greater then 0.00001 degree in latitude or longitude (about 0.5m) is uploaded.

First of all, the Googlemaps node creates a fusion table on the Google Drive (accessible using robot's Gmail account), where every data will be stored. Data uploads are initiated by the human operator, for example by pressing a button on a rumblepad. Anytime the internet connection is not available and there is a request to upload the data (including request to create the fusion table), requests are queued and wait for the next data upload.

4.1.2 Cloud buffer

To lower the network load I have implemented a cloud buffer[1] running on the Google App Engine. Data from the robot are uploaded into this buffer and processed to provide better user experience when using the map. For example the trajectory gets its arrow style, which triples the number of points. Because the GFT has a limitation to amount of data inserted to one cell, the data are also automatically split to comply with these limits. Next step is the upload to the GFT over the Google's high speed infrastructure. Mostly, the data are just moved from one server to another in one datacenter.

4.1.3 Web interface

Once the data are in the GFT they are accessible as any other document stored on Google Drive. They can be visualized through the GFT web interface in rows or in a map or they are also accessible via SQL commands in the same manner as in a database. The web interface is described in the section 5.1.

4.2 Landmarks based navigation

To provide the robot with an absolute position navigation software, I have proposed an algorithm for landmarks based navigation. Fig. 4.2 shows the whole process. As a first step, the area of interest, where the robot navigation is demanded, is documented by a series of images using calibrated cameras or better by a panoramic camera. The best performance is achieved by using the same camera mounted on the same platform in the documentation phase as later in the detection phase. Reason is that cameras placed in the same altitudes above the terrain see for example buildings facade in the same angle. These images create a dataset, which is as a second step uploaded to the SfM web service (described in section 2.2.2), where it serves as a source for the Large-Scale Data Processing tool. This reconstructs a 3D point cloud of documented area composed from a SURF (see the 3.2) described landmarks. The third step is to transform this 3D point cloud to a global coordination frame using the algorithm described in the section 3.4. For this transformation at least 3 corresponding points must have known position, but the more points are provided, the more accurate reference is obtained.

After this preprocessing stage, the 3D point cloud is ready to serve as a reference point cloud for robot navigation. The robot position is obtained by iterating the following sequence:

1. SURF descripted landmarks are detected in the actual panorama view using OpenCV (see 2.2.1) functions. For detection, only the region of 1600x360 from the original image of 1600x800 with masked robot body as on the Fig. 3.1, is used.

[1]hosted on http://gmapscloudbuffer.appspot.com/

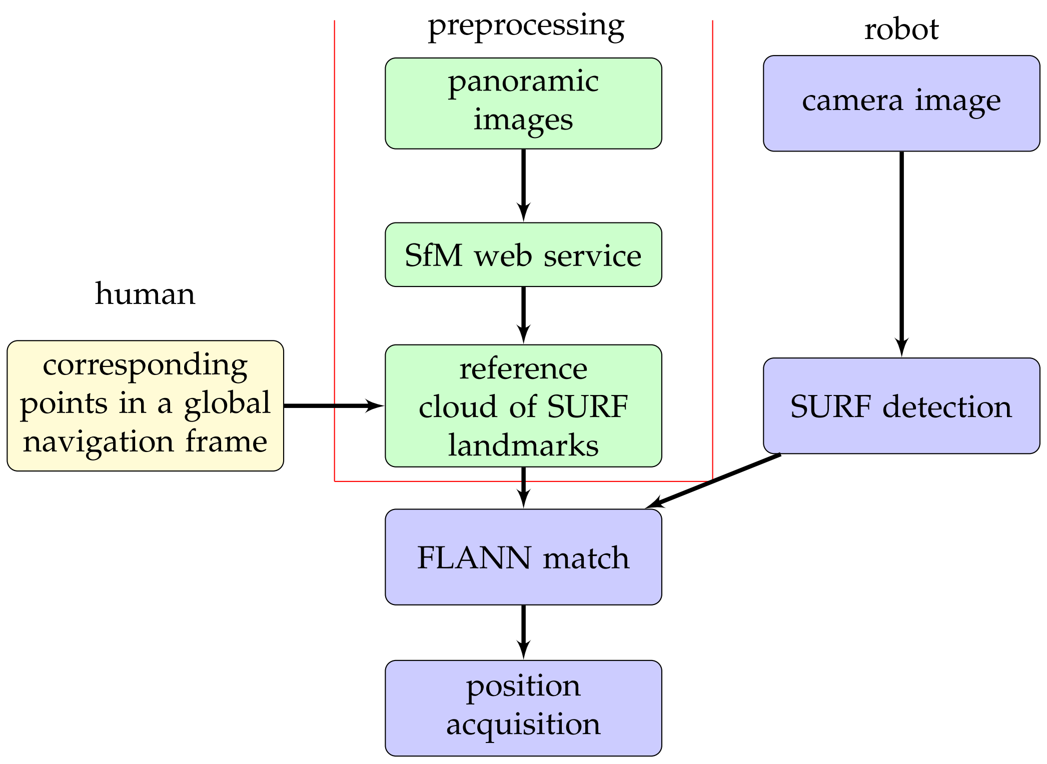


Figure 4.2: Process diagram describing the landmarks based navigation. Green part is made only once before placing to the robot. Blue part is running online and is repeated for every new image.

2. These descriptors are matched with descriptors obtained in preprocessing stage using the FLANN (introduced in the section 3.3).
3. Robot's position is calculated using the algorithm described in 3.5.

Chapter 5

Evaluation

5.1 Evaluation of the Googlemaps node

To demonstrate the basic functionality of the Googlemaps node and its cloud buffer a simple demo was created and is stored on the attached CD. Besides the core Googlemaps script, this demo consists of two files: a bagfile `demo.bag` which contains about 12 minutes long record of robot's trip at CTU and executable launchfile `demo.launch`. Bagfile is a file created by ROS program called Rosbag [40]. It saves all messages on selected topics (such as odometry data) to the file and also enables replaying of these messages. It is a strong feature of ROS which provides running several experiments on once recorded data. Launching the launchfile `demo.launch` will replay this bagfile and launch the Googlemaps node.

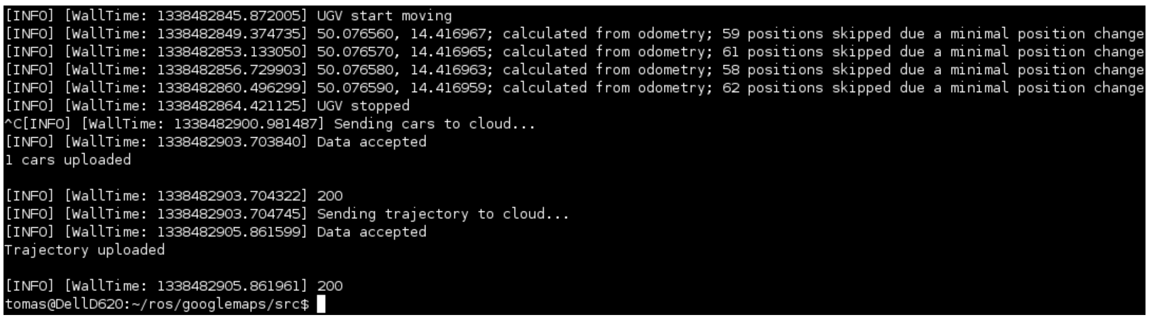
Running instructions

1. To launch the `/demo/demo.launch` type in the terminal:
   `roslaunch demo/demo.launch`

2. In order to upload the trajectory data as well as detected objects in to the Google Fusion Tables, first, let the program run to collect desired amount of data from replaying bagfile, then terminate the program (e.g. by Ctrl+C). Fig. 5.1 shows sample terminal output of the programme terminated by Ctrl+C.

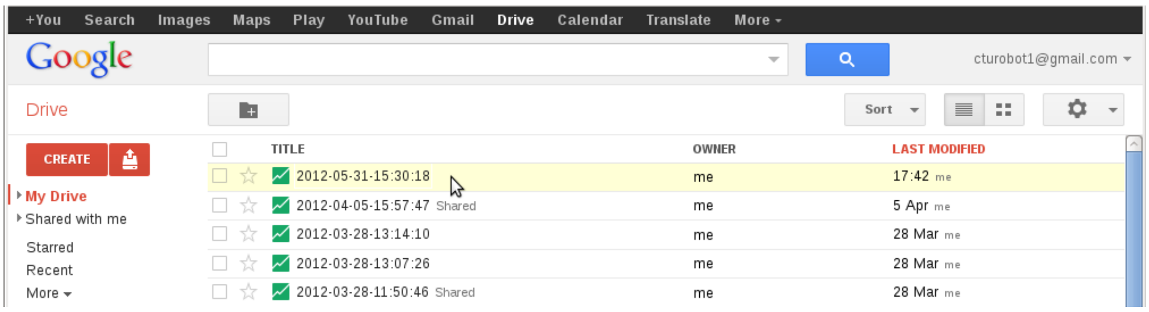
3. The data are now stored in the newly created fusion table (highlighted on the Fig. 5.2) in the Google Drive [41].

4. By default, data are visualized in the rows. A map visualization is obtained by clicking on the *Visualize* button and then selecting the *Map*. Fig. 5.3 shows the location of this button.



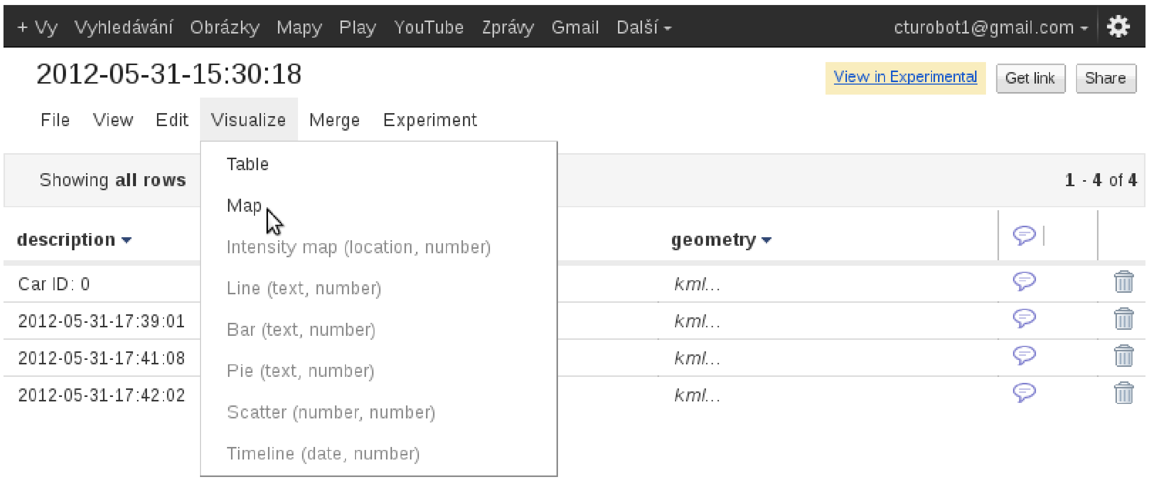
```
[INFO] [WallTime: 1338482845.872005] UGV start moving
[INFO] [WallTime: 1338482849.374735] 50.076560, 14.416967; calculated from odometry; 59 positions skipped due a minimal position change
[INFO] [WallTime: 1338482853.133050] 50.076570, 14.416965; calculated from odometry; 61 positions skipped due a minimal position change
[INFO] [WallTime: 1338482856.729903] 50.076580, 14.416963; calculated from odometry; 58 positions skipped due a minimal position change
[INFO] [WallTime: 1338482860.496299] 50.076590, 14.416959; calculated from odometry; 62 positions skipped due a minimal position change
[INFO] [WallTime: 1338482864.421125] UGV stopped
^C[INFO] [WallTime: 1338482900.981487] Sending cars to cloud...
[INFO] [WallTime: 1338482903.703840] Data accepted
1 cars uploaded

[INFO] [WallTime: 1338482903.704322] 200
[INFO] [WallTime: 1338482903.704745] Sending trajectory to cloud...
[INFO] [WallTime: 1338482905.861599] Data accepted
Trajectory uploaded

[INFO] [WallTime: 1338482905.861961] 200
tomas@DellD620:~/ros/googlemaps/src$
```

Figure 5.1: Terminal output of just terminated Googlemaps node



TITLE	OWNER	LAST MODIFIED
2012-05-31-15:30:18	me	17:42 me
2012-04-05-15:57:47 Shared	me	5 Apr me
2012-03-28-13:14:10	me	28 Mar me
2012-03-28-13:07:26	me	28 Mar me
2012-03-28-11:50:46 Shared	me	28 Mar me

Figure 5.2: Google Drive [41] interface showing the newly created fusion table



description	geometry
Car ID: 0	kml...
2012-05-31-17:39:01	kml...
2012-05-31-17:41:08	kml...
2012-05-31-17:42:02	kml...

Figure 5.3: Example of fusion table generated using Googlemaps node

5. Map as on the Fig. 5.4 contains the robot trajectory and detected cars. As we can see from the trajectory, the robot ended on the same place as started. Since these two points coincides, this serves as a demonstration of the INSO node accuracy.

6. Clicking on the *Get embeddable link* button will show the form with the direct link to the map[1] and a HTML tag as on the Fig. 5.5. This code can be inserted

[1]To use this feature the sharing of the document must be set as *Unlisted* or *Public*. By default every document is created as *Private* for security reasons

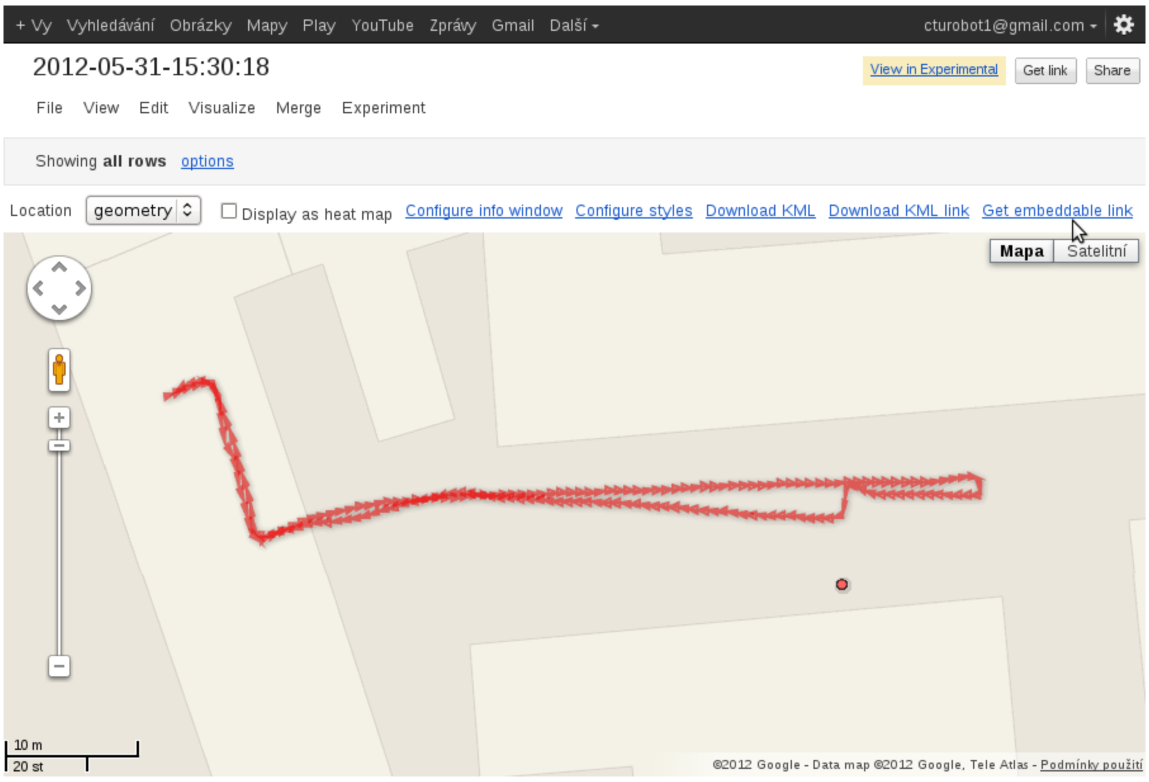
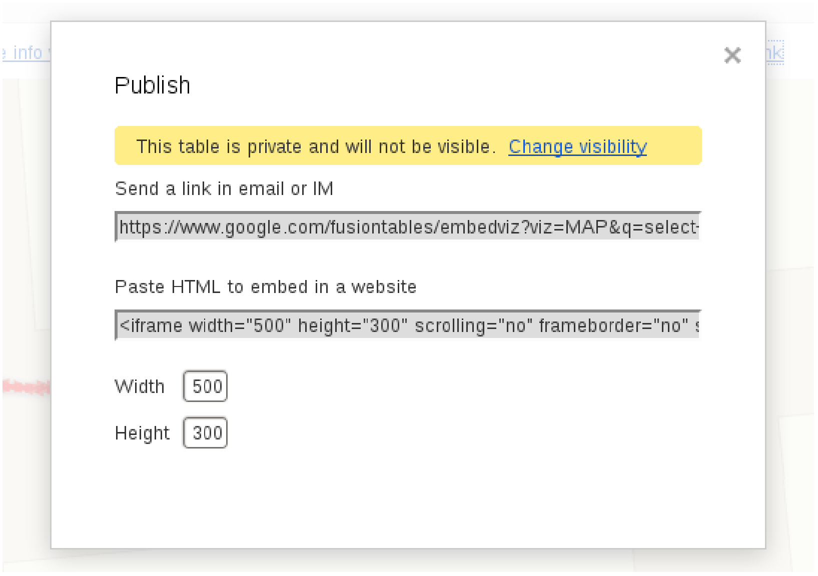

+ Vy Vyhledávání Obrázky Mapy Play YouTube Zprávy Gmail Další
cturobot1@gmail.com
2012-05-31-15:30:18
View in Experimental
Get link
Share
File View Edit Visualize Merge Experiment
Showing all rows options
Location geometry
Display as heat map
Configure info window Configure styles Download KML Download KML link Get embeddable link
Mapa Satelitní
10 m
20 st
©2012 Google - Data map ©2012 Google, Tele Atlas - Podmínky použití

Figure 5.4: Visualization of the robot trajectory in Google Maps. Detected car is displayed as a red dot.

anywhere on the web pages to have there a copy of this map. Structure of the tag is logical and can be simply generated by any program or script. The only unique parameter, which must be set manually, is the table ID, which is obtained during the creation of the table.


Publish

This table is private and will not be visible. Change visibility

Send a link in email or IM

https://www.google.com/fusiontables/embedviz?viz=MAP&q=select

Paste HTML to embed in a website

<iframe width="500" height="300" scrolling="no" frameborder="no"

Width 500

Height 300

Figure 5.5: Form for publishing the map

Chapter 6

Discussion

This chapter highlights some of the issues and unsolved problems, which are yet interesting but exceed the scope of this bachelor thesis.

6.1 Visualization using Google Maps

The Googlemaps node strongly depends on the node, which provides the navigation data, i.e. position and orientation in general. Currently used INSO node suffers from inaccurate measurements in heading rotation, which error accumulates with a traveled distance. One solution, which is currently under development, is to provide the INSO node with the data from visual odometry.

Other solution is usage of a triaxial magnetometer, which is also a part of the IMU, but unlike the acceleration and angular rates is not a data source for the INSO node. The problem is that the Earth's magnetic field is much more weaker than distortions coming both from the robot body and the surrounding environment. The robot body distortions come mainly from 7 motors, 3 fans and a differential locker. If the robot magnetic field model would be created, these disturbances should be filtered and the magnetometer should improve the accuracy of the navigation. Unfortunately this is very difficult task, moreover the robot can operate in magnetically strongly distorted environment like a transformer station where the filtering is almost impossible. Just a vicinity of a metal object such as girder or heater is sufficient to turn the magnetometer axis over 90 degrees.

The functionality of the Googlemaps node can also be extended by the navigation data from the introduced landmarks based navigation. However the Googlemaps node must expect that the reference cloud of landmarks does not cover the area, where the robot currently is and hence the fusion with the INSO node navigation is necessary.

6.2 Future work on the landmarks based navigation

Achilles heel of the proposed landmarks based navigation is the long preprocessing phase, which requires large amount of images to be captured as well as long computational time (up to several days). One of the possibilities how the landmarks based navigation could be improved is by using an on-line database containing images with their location. One example of a large on-line database is the Google Streetview [13], which contains panoramic images of streets all around the world and is being permanently updated. These images have a density of one image per 10 meters of route, which is not sufficient for the SfM web service (3D reconstruction software mentioned in the 2.2.2) and can cause problems such as false positive match of two similar windows on the building facade. However, every Google Streetview image has a sparse depth map, which can help with the image matching. This is an interesting topic for a future work.

Solution to the long computation time lies in the on-demand computing of the close surrounding of the robot assumed position. It means that instead of computing the large area of size 100x50 m using over 1600 images, only a small region composed of tens images would be processed. Center of this region would be the panoramic image with most corresponding landmarks. The reference point cloud of landmarks then expands dynamically with every new image.

Chapter 7

Conclusion

The first aim of this work was to analyze the state of the art of the currently available cloud services used for online localization of mobile robots performing urban search and rescue missions. Based on this analysis, the Google products were used for their user-friendly documentation of the API and direct connection to other cloud services.

As the next step, the ROS node for visualization of robot trajectory and detected cars was created. This node reports and publishes data on the internet at the end of the mission or whenever the robot operator requires and also when the internet connection is available. For optimizing the data flows, the cloud buffer web application was also implemented. The data published are stored on the Google Drive using the concept of fusion tables and can be shared this way. Also there is a possibility to add generated Google map to any web page. Specific implementation then depends on the web sites and web server services. The function of this software was evaluated in outdoor experiments (one example is in the section 5.1).

For this localization software it is sufficient to use the current implementation of the navigation algorithm based on inertial data fusion with odometry measurements provided by the INSO node (mentioned in the section 2.2.2). Since this navigation algorithm is a dead reckoning only (i.e. no external reference is taken into account), with the traveled distance it suffers from cumulative error in heading and the initial robot position has to be set manually. When seeking possible extensions and improvements to the dead reckoning navigation a new navigation method based on the database of landmarks was proposed and introduced in this work. Original aim was to use a cloud services like the Google Street View as a data source but in account of issues mentioned in the section 6.2 and for the reason, that this services are not covering every place (e.g. CTU quad), the creation of a custom data source was necessary and essential. This data was obtained during a one hour outdoor experiment with the NIFTi robot (section 2.1) and then processed by the SfM web service (section 2.2.2). Since this navigation method exceeds the scope of this thesis, there was not sufficient time for implementation and experimental evaluation.

Although the thesis assignment was accomplished, there are a lot of ideas for future work. It is mainly the minimizing the human factor in the whole localization process. In the case of Googlemaps node, the robot operator must manually set the robot starting location and orientation. In the case of the landmarks based navigation, the operator has to do manual assignment of the reference cloud of landmarks to the global navigation frame. The accuracy of both these operations has a direct impact on the accuracy of the whole system. At this point I see the weakest part because every human can make (and from time to time does) a mistake.

Bibliography

[1] The new NIFTi robot platform. Accessed: 27/01/2012. Available from: http://www.nifti.eu/news/the-new-nifti-robot-platform.

[2] Natural human-robot cooperation in dynamic environments. Accessed: 25/05/2012. Available from: www.nifti.eu.

[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. Available from: http://dx.doi.org/10.1016/j.cviu.2007.09.014, doi:10.1016/j.cviu.2007.09.014.

[4] Google My Tracks. Accessed: 12/05/2012. Available from: http://www.google.com/mobile/mytracks/.

[5] Duncan Robertson and Roberto Cipolla. An image-based system for urban navigation. In *IN BMVC*, pages 819–828, 2004.

[6] Masafumi Noda, Tomokazu Takahashi, Daisuke Deguchi, Ichiro Ide, Hiroshi Murase, Yoshiko Kojima, and Takashi Naito. Vehicle ego-localization by matching in-vehicle camera images to an aerial image. In *Proceedings of the 2010 international conference on Computer vision - Volume part II*, ACCV'10, pages 163–173, Berlin, Heidelberg, 2011. Springer-Verlag. Available from: http://dl.acm.org/citation.cfm?id=2040739.2040759.

[7] Turgay Senlet and Ahmed M. Elgammal. A framework for global vehicle localization using stereo images and satellite and road maps. In *ICCV Workshops*, pages 2034–2041, 2011.

[8] H. Uchiyama, D. Deguchi, T. Takahashi, I. Ide, and H. Murase. Ego-localization using streetscape image sequences from in-vehicle cameras. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 185 –190, june 2009. doi:10.1109/IVS.2009.5164275.

[9] Margrit Betke and Leonid Gurvits. Mobile robot localization using landmarks. *IEEE Transactions on Robotics and Automation*, 13:135–142, 1995.

[10] Stephen Se, David G. Lowe, and James J. Little. Vision-based global localization and mapping for mobile robots. *IEEE Transactions on Robotics*, 21:364–375, 2005.

[11] Marius Muja and David G. Lowe. FAST APPROXIMATE NEAREST NEIGHBORS WITH AUTOMATIC ALGORITHM CONFIGURATION. In Ranchordas, A and Araujo, H, editor, *VISAPP 2009: PROCEEDINGS OF THE FOURTH INTERNATIONAL CONFERENCE ON COMPUTER VISION THEORY AND APPLICATIONS, VOL 1*, pages 331–340. Inst Syst & Technologies Informat, Control & Commun; ACM SIGGRAPH, 2009. 4th International Conference on Computer Vision Theory and Applications, Lisbon, PORTUGAL, FEB 05-08, 2009.

[12] Akihiko Torii, Michal Havlena, and Tomas Pajdla. From Google Street View to 3D city models. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 2188–2195. IEEE, September 2009. Available from: http://dx.doi.org/10.1109/ICCVW.2009.5457551, doi:10.1109/ICCVW.2009.5457551.

[13] Google Street view. Accessed: 30/01/2012. Available from: www.google.com/streetview.

[14] Willow Garage. Accessed: 26/01/2012. Available from: http://www.willowgarage.com/pages/about-us/overview.

[15] Open Source Robotics Foundation. Accessed: 08/05/2012. Available from: http://www.osrfoundation.org/.

[16] Robot Operating System. Accessed: 16/01/2012. Available from: http://www.ros.org/wiki/ROS.

[17] ROS node. Accessed: 27/01/2012. Available from: http://www.ros.org/wiki/Nodes.

[18] ROS topic. Accessed: 27/01/2012. Available from: http://www.ros.org/wiki/Topics.

[19] Open Source Computer Vision Library. Accessed: 08/05/2012. Available from: http://opencv.willowgarage.com/wiki/.

[20] Michal Reinstein Vladimir Kubelka. Complementary filtering approach to orientation estimation using inertial sensors only, 2011.

[21] Tomáš Svoboda Karel Zimmermann, David Hurych. Improving cascade of classifiers by sliding window alignment in between. In *Proceedings of the Fifth International Conference on Automation, Robotics and Applications*, pages 196–201, Private Bag 11 222, Palmerston North , 4442, New Zealand, December 2011. School of Engineering and Advanced Technology, Massey University, Massey University. CD-ROM. Available from: ftp://cmp.felk.cvut.cz/pub/cmp/articles/hurycd1/icara2011.pdf.

[22] NIFTi object detection demos. Accessed: 23/05/2012. Available from: https://cw.felk.cvut.cz/doku.php/misc/projects/nifti/demos/object_detection.

[23] SfM Web Service. Accessed: 21/05/2012. Available from: http://ptak.felk.cvut.cz/sfmservice/.

[24] Center for Machine Perception at Czech Technical University in Prague. Accessed: 21/05/2012. Available from: http://cmp.felk.cvut.cz.

[25] Google Maps API Family. Accessed: 27/01/2012. Available from: http://code.google.com/apis/maps/index.html.

[26] KML Reference. Accessed: 27/01/2012. Available from: http://code.google.com/apis/kml/documentation/kmlreference.html.

[27] Open Geospatial Consortium, Inc. Accessed: 27/01/2012. Available from: http://www.opengeospatial.org/standards/kml.

[28] Google Fusion Tables. Accessed: 16/01/2012. Available from: http://www.google.com/fusiontables.

[29] Google Earth. Accessed: 27/01/2012. Available from: http://www.google.com/earth/index.html.

[30] D. Titterton and J. Weston. *Strapdown Inertial Navigation Technology*. The American Institute of Aeronautics and Astronautics, second edition, 2004.

[31] Hessian. Accessed: 28/05/2012. Available from: http://mathworld.wolfram.com/Hessian.html.

[32] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. Available from: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94, doi:10.1023/B:VISI.0000029664.99615.94.

[33] Haar Function. Accessed: 28/05/2012. Available from: http://mathworld.wolfram.com/HaarFunction.html.

[34] Michel Marie Deza and Elena Deza. *Encyclopedia of Distances*. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-00234-2_1.

[35] Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977. Available from: http://doi.acm.org/10.1145/355744.355745, doi:10.1145/355744.355745.

[36] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:376–380, 1991. doi:http://doi.ieeecomputersociety.org/10.1109/34.88573.

[37] Singular Value Decomposition from Wolfram MathWorld. Accessed: 25/05/2012. Available from: http://mathworld.wolfram.com/SingularValueDecomposition.html.

[38] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[39] Google App Engine. Accessed: 22/05/2012. Available from: https://developers.google.com/appengine/.

[40] rosbag. Accessed: 24/05/2012. Available from: http://ros.org/wiki/rosbag.

[41] Google Drive. Accessed: 22/05/2012. Available from: https://drive.google.com/.

CD content

Attached CD contains the source codes of the software presented in this work. Namely the Google maps node and some python scripts for the landmarks based navigation. It also contains a text of this bachelor thesis in the PDF format and a source codes of the whole text for the LaTeX.

The directory tree is in the next table:

Table 1: Directory tree of the attached CD

Directory	Label
`Googlemaps_node`	source files for the Googlemaps node
↪ `demo`	demo files
↪ `src`	source codes
↪ `googlemaps_manual.pdf`	short documentation to the Googlemaps node
`Landmarks`	source files for the Landmarks based navigation
↪ `dvorana.wrl`	WRL model of the CTU quad
↪ `find_in_pointcloud.py`	python script for matching in the model
↪ `pointassgmnt.point`	point assignment for the model
↪ `simtls.py`	python script for the transformation from 3.4
↪ `surf.desc`	file containing points descriptors
↪ `surf.point`	file containing points coordinates
↪ `transformed.point`	file containing points global coordinates
`LaTeX`	source codes for the text of the thesis
↪ `thesis.pdf`	CD version of the thesis