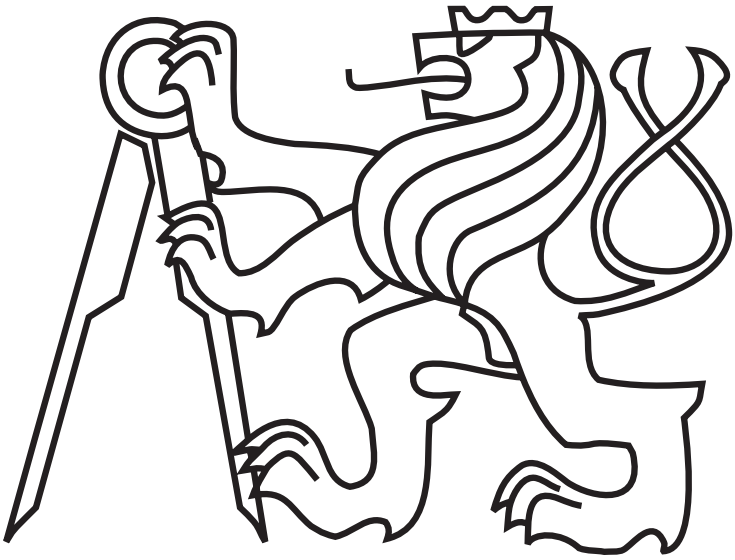
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

Bachelor's thesis



Martin Fadrhons

Software for embedded module for image processing

Department of Cybernetics

Thesis supervisor: **Ing. Tomáš Krajník Ph.D.**

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Martin F a d r h o n s

Studijní program: Kybernetika a robotika (bakalářský)

Obor: Robotika

Název tématu: Software pro vestavný modul pro zpracování obrazu

Pokyny pro vypracování:

1. Seznamte se s vestavným modulem pro zpracování obrazu využívaným v rámci katedry kybernetiky.
2. Seznamte se s metodami typu 'teach and repeat' pro navigaci mobilních robotů.
3. Upravte software FPGA modulu tak, aby mohl implementovat vybranou metodu.

Seznam odborné literatury: Dodá vedoucí práce.

Vedoucí bakalářské práce: Ing. Tomáš Krajník, Ph.D.

Platnost zadání: do konce zimního semestru 2013/2014

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

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

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2013

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

BACHELOR PROJECT ASSIGNMENT

Student: Martin F a d r h o n s

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Bachelor Project: Software for Embedded Module for Image Processing

Guidelines:

1. Get to know the embedded module for image processing used by Department of Cybernetics.
2. Get to know the 'teach and repeat' methods for mobile robot navigation.
3. Modify FPGA module's software so that it can implement the chosen method.

Bibliography/Sources: Will be provided by the supervisor.

Bachelor Project Supervisor: Ing. Tomáš Krajník, Ph.D.

Valid until: the end of the winter semester of academic year 2013/2014

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

L.S.

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 10, 2013

Prohlášení

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 2. 1. 2014 .....

.....

Podpis autora práce

Acknowledgement

I would like to thank my family for support during my studies at the Czech Technical University in Prague. I also would like to thank supervisor of my thesis Tomáš Krajník for his help and during work on this thesis.

Abstrakt

Tato práce popisuje úpravy provedené na zařízení zvaném SCHVAB MiniModule, které zatím slouží jen jako nástroj pro zpravování obrazu. Cílem úprav je, aby modul mohl implementovat navigační systém SURFNav. Modul má naimplementovaný detektor význačných bodů v obraze Speeded Up Robust Feature (SURF). Součásti úprav je i implementace nového deskriptoru detekovaných význačných bodů Binary Robust Independent Elementary Features (BRIEF). Na začátku práce se věnuji obecně metodam v mobilní robotice. Po té následuje diskuze navigačních algoritmů. Další kapitola se věnuje algoritmu SURF a deskriptorům význačných bodů, následována kapitolou o MiniModulu. Na závěr jsou vyhodnoceny provedené změny řadou experimentů.

Abstrakt

This thesis describes changes made on device called SCHVAB MiniModule which now only works as image processing tool. Goal of modifications is implementation of SURFNav navigation algorithm. MiniModule have implemented feature extractor algorithm Speeded Up Robust Features (SURF). Part of modifications is implementation of new feature descriptor Binary Robust Independent Elementary Features (BRIEF). Ad the beginning of thesis I speak about methods in mobile robotics. After that follows discussion of navigational methods. Next chapter is dedicated to SURF algorithm and feature descriptors, followed with chapter about SCHVAB MiniModule. It the end are modifications evaluated with experiments.

Contents

List of Figures

List of Tables

List of Source codes

List of Algorithms

1 Introduction

If we want to make mobile robot autonomously move there are three main tasks that must be solved. Even for the simplest tasks as moving from location A to location B robot needs to know location of itself in the environment. Therefore *localization* is a must. But it is not the only need, *mapping* creates map from robot's sensor data. Finally, moving the robot is called *motion planning*.

In other words navigation of robot is complicated task. But with growing computational power it is possible to invent more sophisticated ways of robot navigation. However sometimes we want robot to be small, that means that it can't carry powerful computer. Exactly for that case was developed SCHVAB MiniModule. It is a small Field Programmable Gate Array (FPGA), with PowerPC processor. This device is small and with relatively low power consumption, which means longer battery life. The FPGA is configured to process images from camera. Processed images are about to be used for visual navigation. Visual navigation may be used as cheaper version of navigation systems based on laser radars that are scanning environment. Application of this robot system may be found in military and manufacturing industry.

Aim of this thesis is to continue on work started by Jan Šváb. Whole module concept was designed to become a autonomously navigated robot. However not everything is ready to make it happen. At the beginning we will take a look on global view of methods used in mobile robotics. After that comes discussion of navigation methods suitable for SCHVAB MiniModule. MiniModule will implement SURFNav navigation method and it is compared to Simultaneous Localization and Mapping (SLAM), because it is very popular method used in various applications. Afterwards will be SURFNav algorithm described.

In next chapter I will describe SURF and feature description methods. This is on of the key components in visual navigation. Extracting features or landmarks is needed to create visualize relations between robot an surrounding environment. Later in paper will reader find description of SCHVAB MiniModule and information about done modifications. End of the paper is dedicated to experiments and evaluation of made modifications.

2 Methods in mobile robotics

For navigation purposes three main tasks need to be solved. We, the people, solve this tasks every now and then and in the context of robot navigation these tasks remain the same. These tasks are

- localization,
- motion planning,
- mapping.

All these parts of navigational process are closely related to each other, as we can see on following diagram 1

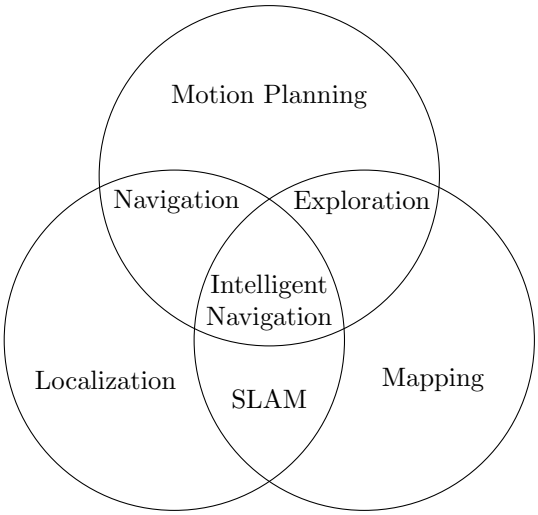


Figure 1: Basic navigation tasks interconnections. Courtesy of [1]

2.1 Localization

If we want to be able to navigate robot we need to determine its position, this problem solving is called localization. We can distinguish two major types of localization by absolute and relative reference[5]. Localization may be continuous, where the initial position of vehicle is a priori known or global localization, where the initial position of robot is not known. Very special case is called *kidnapped robot problem*. In this case robot is suddenly moved to another location.

2.1.1 Dead reckoning

Dead reckoning (also known as deduced reckoning) is technique, which estimates current position of vehicle from previously known position, speed over elapsed time and course[5]. Speed and course are measured with sensors. This is where the main disadvantage of dead reckoning comes from. Even with very precise sensors there will be cumulative error. This error is integrated over time which makes dead reckoning not suitable for long term localization. But still dead reckoning is often used for estimating position in a context of complex localization systems[6].

Odometry is the most frequently used dead reckoning method. It calculates speed and position from signals of sensors connected to actuators. Most common method is IRC (incremental rotary sensor) linked to wheel axis. From sensors's signals is computed speed or distance traveled by wheel, from this information we can estimate course. In this method error can be increased due to wheel slip. Another method uses optical mice to estimate position from speed and direction[7] or doppler sensors[8].

In aviation there are often used methods, that make use of accelerometers[9] or gyroscope[10]. These methods are called inertial localization. Inertial methods are used to supplement GPS localization, it can be found in many new smart phones.

2.1.2 Beacon based localization

This method of localization is based on beacons, artificial objects, which are added to the environment. There are two possible ways of approach. First is placing beacons in the environment with known position and robot carries detection system. Second approach locate detection system in the environment and beacon on the robot[11]. For determining position of robot is used triangulation or trilateration[5]. Moreover these methods also can provide heading of robot.

The most spread beacon method is GPS (Global Positioning System). Active beacons are floating on the orbit of the Earth. Position computation is based on distances from each satellite, whose positions is known. Disadvantage of GPS is that its signal is easily absorbed or reflected. This is being solved by combining GPS with dead reckoning methods, usually inertial ones, as mentioned before.

2.1.3 Map based localization

Map based localization is based on very clear procedure. Data from sensors are compared to map. Map based localization is dependent on the interpretation of map, but the main difference can be found in the ways of sensor data processing. In one case all data from sensors create local map, which is consequently compared to global map[12]. This approach is very conditional on sensitivity to changes of environment.

In the other case are first detected well distinguishable patterns, these are called features or landmarks[13]. Patterns are being matched with corresponding patterns contained in map[14]. For better results there is a possibility to use beacons as landmarks. With this approach you can get rid of noise from sensors.

2.2 Motion planning

Under the term motion planning we can find two major tasks: path planning and motion control. The goal of the whole process is to move robot through path. The path planning task finds suitable path without collisions. We can have other requirements such as the cheapest way or if we want to explore to move only on unexplored places. Motion control task just sends signal to actuators (e.g. wheels, legs or tracks).

2.3 Mapping

Maps are used as a tool for robots to establish their position in environment and they are also used for motion planning. Maps are being built from robots sensors. Due to many types of localization and motion planning, different types of maps are used in robotics[15]. We can divide maps to two groups - metrical and topological. Metrical maps have fixed coordinates and are represented by sensor, geometrical, grid, meshes and landmark maps. Topological maps contain only spatial relations.

Sensor maps

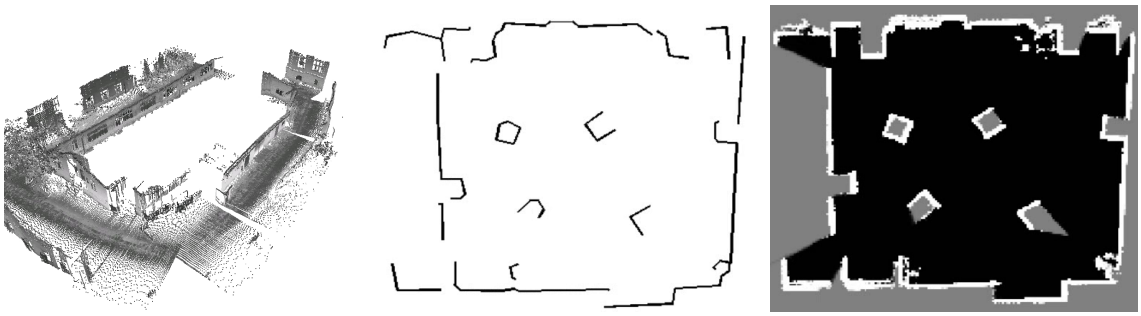
Sensor map is a recording of all sensory measurements. Collected data are not further processed, however sensory maps are usually created for further processing of measured data. Very popular method for gathering environment data is through laser scanners, also radar or sonar are commonly used for gathering information. Typical representative of sensor map is point cloud 2a.

Geometrical maps

These maps represent information about environment with the use of basic geometrical shapes. For two-dimensional space it is usually lines (example in figure 2b) or polygons and planes for three-dimensional space. Geometrical maps are suitable for use in environment with high amount of flat surfaces such as indoor and urban ones. In comparison with sensory maps significantly lesser amount of data needs to be stored into memory.

Grid maps

We can also think of environment as grid of cells. Then we can assign to each cell information about its inner space. One of the most common types of information is whether cell is occupied or not in probabilistic way. It is called occupancy grid [16], [17]. Example of two-dimensional occupancy grid can be seen in figure 2c. White colour represents cell with high probability of being occupied, in other words it shows where obstacles are, black colour represents free to roam area. Grey represents unexplored area.



(a) Sensory map. [4] (b) Geometrical map. [18] (c) Occupancy grid map. [18]

Figure 2: Maps examples

Meshes

In mobile robotics it is usually required to create map with as little as possible memory consumption but with as many as possible information. But in other applications of map building we might want to build exact models of environment. Purpose of these maps may be for visualizing areas of interest. Mesh map can be built from point cloud sensory map using triangulation [19]. Example of this technique is shown in figure 3a, original point clouds can be seen in figure 2a.

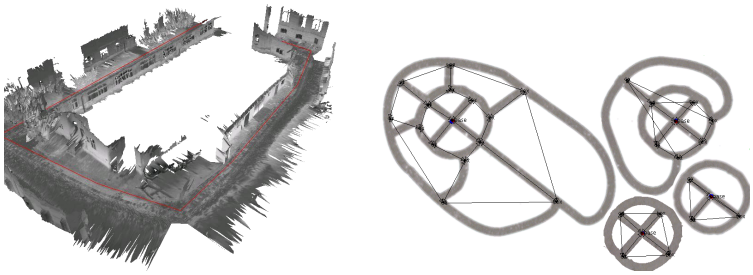
Landmark maps

Landmark map consists of well distinguishable salient objects with known position. This type of map is very popular in visual localization/navigation systems [20], [15]. In this case landmarks are detected in images compared with ones already stored in the map and afterwards the location is determined.

Topological maps

Topological map is graph representing environment. In this graph nodes stand for distinctive locations and edges link these locations. Additional information can be bound with

node and edges, most common case is adding path to edges. Topological maps are suitable for easy path planning. Well known example of topological is public transport scheme. Another instance of topological map can be seen in figure 3b.



(a) Mesh map. [4] (b) Topological maps. [4]

Figure 3: Maps examples

2.4 Navigation

One of the key tasks in mobile robotics is navigation. Similar to using GPS navigation in a car, when man is told where to ride, robots need to be told the same way. Basically navigation tasks methods can be split into three groups: map based, map-less based and map-building based navigation[21].

Map based navigation have two possible approaches. One are systems that need complete map of environment before starting the navigation. The other type is called teach and repeat method. Principle of those methods is described in chapter 3.

Map-less navigation systems only reacts on current observation of surroundings of robot. They don't use any map for navigation. Typical representative of last group map-building navigations is SLAM which is also more described in chapter 3.

2.5 Exploration

Process of building mapping surroundings is called exploration. Most known exploring application is connected with mapping of extraterrestrial surfaces [22]. We can describe exploring process as a sequence of several repeating steps. First robot maps envinronment and builds a map. Afterwards path is planned on the edge between known and unknown. After that robot moves to given location and starts mapping again. Exploring can be also performed in cooperative manner such as in [23].

3 Teach and repeat navigation methods

The aim is to find suitable navigation method for SCHVAB MiniModule device. For that purpose it was decided to use teach and repeat navigation method which is providing a simple form of telling the robot which way to go. This method divides navigation in two phases:

teach phase in which robot is driven along the path. Robot is recording signals from its sensors and builds a map.

repeat phase in which robot compares sensor inputs with the map and moves along the taught path.

There are many various teach and repeat navigation methods. In paper by Paul Furgale and Timothy D. Barfoot [24] is used stereovision camera to store and postprocess images to submaps. Since the SCHVAB MiniModule is processing images I was looking for navigation methods based on image processing. Mainly we can find two different approaches. One option is to build three-dimensional map of features [25] or create topological maps [26, 27]. A very popular solution is Simultaneous Localisation and Mapping (SLAM) [28]. For our use would be most suitable so-called MonoSLAM [25] which uses single camera. There are other versions of SLAM using stereo vision [29]. Another method that seems suitable for our purpose would be SURFNav [26]. To decide which method would be better to implement i will briefly describe each method and compare them. Subsequently chosen method will be described in detail.

3.1 Methods discussion

MonoSLAM algorithm is building probabilistic 3D map of features. This map is used to acquire position of robot. The process of map building uses feature extraction algorithms. For each feature is estimated its position and uncertainty of estimation. Generated map is evolving and updated using Extend Kalman Filter (EKF). During update features can be even added or removed from the map. To gain data for update estimates of position and uncertainty camera motion is needed. With growing number of features N grows size of map in order $O(N^2)$.

SURFNav algorithm is building topological map, where edges of map contain information about detected features. This map is only used to correct heading direction of robot. Traveled distance is measured with odometric measurements. Map is just a set of features with information of their location and distance where they were detected. In repeat phase are compared features detected from camera image with the ones on the map and horizontal difference of matched features is calculated. This difference is used to correct robot's heading.

To decide which method would be better we have to take into consideration computational demands. Since SCHVAB MiniModule have only PowerPc 440 processor unit we will choose method with lower computational demands. In this case SLAM based algorithm has a disadvantage. With mapping of large environments would extremely grow computational time due to great number of detected features. This can be partially solved by dividing maps into smaller segment with known mutual position [30]. Still SLAM contains computation of estimated position and uncertainty. The SURFNav is in repeat phase comparing only two sets of features and computing modus of horizontal differences. Thus it was decided that SURFNav will be suitable for implementation.

3.2 SURFNav

As mentioned before SURFNav creates topological map of surroundings, that is created for later autonomous navigation. SURFNav uses proprioceptive sensors to estimated length of traveled segment and camera sensor for building map and heading control. Images from camera are processed with Speeded Up Robust Features (see Chapter 4) algorithm in order to identify landmarks in the image. Those detected landmarks are used to build map.

Map is created when robot is driven by operator, this is teaching phase of this algorithm. Robot is guided so that it moves straight forward or turn on spot. Generated topological map edges consist of local landmark map build of detected salient features. This local map provides information about location of feature in image and traveled distance where it was detected. And of course it provides feature descriptor.

Mathematical proof of robot position uncertainty having its bound can be found in paper [27].

3.2.1 Teaching phase

Procedure that takes action during teaching phase is described in algorithm 1. Described procedure is repeated for each segment of map. After creation map of one segment robot is turned and whole procedure repeats. This algorithm is based on working with three sets of landmarks: L is map of landmarks, T is set of tracked landmarks and S is set of currently detected landmarks. The set S is updated each time new image from camera is processed. For each landmark from set T are found two best matching landmarks in set S. More about matching features is written in chapter 4.2. In case of one feature is significantly similar than the other feature, tracked landmark information is updated. In other case tracked landmark is moved to set L. Remaining landmarks from set S are added to set of tracked landmarks T.

Algorithm 1: Learn one segment. Courtesy of [4].

Input: α – initial robot orientation (compass value)

Output: (α, s, L) – associated data to segment, where s is traveled distance and L is set of landmarks, a landmark is senary (e, k, u, v, f, g), where e is a SURF descriptor, k is counter of feature detection, u and v is position of feature in the image (at the moment of first, resp. last occurrence), f and g denotes distance from segment start according to u, resp. v.

$L \leftarrow \emptyset$ // set of learned landmarks

$T \leftarrow \emptyset$ // set of tracked landmarks

$\alpha \leftarrow$ *compas value* // robot orientation at the beginning of segment mapping

repeat

  $d \leftarrow$ *current traveled distance from the segment start*

  $S \leftarrow$ *extracted features,* $(u, e) \in S$*, u position, e feature descriptor*

  foreach $\boldsymbol{t_i} = (e_i, k_i, u_i, v_i, f_i, g_i) \in T$ **do**

    $(u_a, e_a) \leftarrow \text{argmin}\{\|e_i, e(s)\| | s \in S\}$ // select the best matching descriptor from S to e_i

    $(u_b, e_b) \leftarrow \text{argmin}\{\|e_i, e(s)\| | s \in S \setminus \{(u_a, e_a)\}\}$ // select the next best matching descriptor

    if $\|(e_i, e_a)\| \ll \|(e_i, e_b)\|$ **then**

      $\boldsymbol{t_i} \leftarrow (e_i, k_i + 1, u_i, u_a, f_i, d)$ // update matched landmark

      $S \leftarrow S \setminus \{(u_a, e_a)\}$ // remove matched feature from current set of detected landmarks

    else

      $T \leftarrow T \setminus \{\boldsymbol{t_i}\}$ // remove $\boldsymbol{t_i}$ from set of tracked landmarks

      $L \leftarrow L \cup \{\boldsymbol{t_i}\}$ // add $\boldsymbol{t_i}$ to set of learned landmarks

  foreach $(u, e) \in S$ **do**

    $T \leftarrow T \cup \{(e, 1, u, u, d, d)\}$ // add new feature to set of tracked landmarks

  set robot steering velocity to ω

until *robot is in the mapping mode*

$s \leftarrow d$ // total travelled distance along segment

$L \leftarrow L \cup T$ // add current tracked landmarks to the learned landmarks

3.2.2 Repeat phase

Autonomous navigation process is based on retrieving set of landmarks from map and matching them with landmarks retrieved from processed image to estimate their current position in the image. Mode of differences between current and retrieved position is used to correct robot's heading. Whole procedure for traversing one segment of map is described in algorithm 2.

Algorithm 2: Traverse one segment. Courtesy of [4].

Input: (α, s, L) – associated data to segment, where α is initial angle of robot orientation at segment start, s is traveled distance and L is set of landmarks, a landmark is senary (e, k, u, v, f, g), where e is a SURF descriptor, k is counter of feature detection, u and v is position of feature in the image (at the moment of first, resp. last occurrence), f and g denotes distance from segment start according to u, resp. v.

Input: c – steering gain parameter

turn(α) `// turn robot in direction` α

$d \leftarrow$ *current traveled distance from the segment start*

while $d < s$ **do**

- $T \leftarrow \emptyset$ `// set of current tracked landmarks`
- $H \leftarrow \emptyset$ `// histogram of horizontal position differences`
- $d \leftarrow$ *current traveled distance from the segment start*
- $S \leftarrow$ *extracted features,* $(u, e) \in S$*, u position, e feature descriptor*
- **foreach** $\boldsymbol{l_i} = (e_i, k_i, u_i, v_i, f_i, g_i) \in L$ **do**
  - **if** $g_i \geq d \geq f_i$ **then**
    - $T \leftarrow T \cup \{\boldsymbol{l_i}\}$ `// add landmark to tracked landmarks according to traveled distance`
- **while** $|T| > 0$ **do**
  - $(e_i, k_i, u_i, v_i, f_i, g_i) \leftarrow \text{argmax}_{\boldsymbol{t} \in T}\, k(\boldsymbol{t})$ `// get landmark with maximal number of occurrences` k
  - $p \leftarrow \frac{d - f_i}{g_i - f_i}(v_i - u_i) + u_i$ `// estimate its image coordinates`
  - $(u_a, e_a) \leftarrow \text{argmin}\{||e_i, e(s)|| \; | s \in S\}$ `// select the best matching descriptor from` S `to` e_i
  - $(u_b, e_b) \leftarrow \text{argmin}\{||e_i, e(s)|| \; | s \in S \setminus \{(u_a, e_a)\}\}$ `// select next the best matching descriptor`
  - **if** $||(e_i, e_a)|| \ll ||(e_i, e_b)||$ **then**
    - $H \leftarrow H \cup \{p_x - u_{ax}\}$ `// add horizontal difference to the histogram`
  - $T \leftarrow T \setminus \{(e_i, k_i, u_i, v_i, f_i, g_i)\}$ `// discard used landmark`
- $\omega \leftarrow c \cdot \text{mode}(H)$ `// determine new robot steering velocity`
- *set robot steering velocity to* ω

At the beginning of each loop are retrieved landmarks S from processed picture and traveled distance d from odometry. After that is set of tracked landmarks T filled with landmarks from map L corresponding with traveled distance. That means that distance of first detection f of landmark must be lower than currently traveled distance d and distance of last detection of landmark g must be higher then d. For all tracked landmarks is computed supposed position of landmark in image. The linear interpolation of u, v, f, g

and d is used for that purpose. Now two sets of landmarks T and S are paired the same way as in teaching phase. For each pair is added difference between horizontal coordinates to histogram H. Histogram H consist of 31 bins. A mode of differences from histogram is then used to correct robot's heading. To make sure that the computed mode is trustworthy it is needed to evaluate histograms quality with the notion of information $I(H)$ defined as

$$I(H) = log_2 n + \sum_{i=1}^{n} \frac{h_i}{h} log_2 \frac{h_i}{h} [bit], \tag{1}$$

where h_i is value of bin, h is sum of all bin values, n is number of bins. Mode of histogram with low information value such as flat histogram is more likely to provide wrong ω value. When robot reaches end of a segment it stops and loads map of next segment.

4 Speeded Up Robust Features

For comparing images in meaning of translation, rotation or scale we need to detect well distinguishable points, region or edges, these are called features. Basic demand for feature extraction is repeatability, which means that features should be detected repeatedly. Extracting features consist of two parts:

Interest point detection is process with goal to localize salient point in the image. Together with location of points of interest additional information may be provided. Such as Hessian determinant value, scale, Laplacian, etc.

Interest point description process is allowing matching features across several images. Descriptor is generated by describing surroundings of point of interest.

In this thesis I will focus only on Speeded Up Robust Features (SURF) [31] algorithm as it is implemented in Schvab Minimodule. Detecting features with SURF algorithm is composed of three phases: integral image generation, interest point detection, descriptor generation. First two phases are part of Feature detection and are described in following section.

4.1 Feature detection

4.1.1 Integral image generation

Integral image is calculated according to equation 2. Where I represents luminance component of the image.

$$I_{\Sigma}(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(i,j) \tag{2}$$

Purpose of integral image generating is to lower computational and memory requirements. For obtaining integral above part of image only four memory reads are performed. For example given in figure 4 the integral over gray part of integral image will be equal to equation 3.

$$I_{\Sigma}(ABCD) = I_{\Sigma}(A) + I_{\Sigma}(D) - I_{\Sigma}(B) - I_{\Sigma}(C) \tag{3}$$

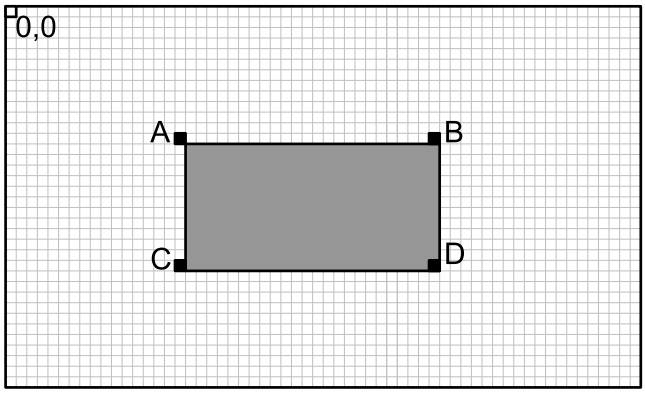


Figure 4: Integral image usage. Courtesy of [2]

4.1.2 Interest point detection

In this phase we compare local maxima across Hessian matrix determinants. Hessian matrix consists of second order partial derivatives of multi-dimensional scalar function. The determinant of Hessian matrix for two-dimensional scalar function $f(x, y)$ is given by equation 4.

$$\mathcal{H}(x,y) = det\left(H(x,y)\right) = \begin{vmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{vmatrix} = \frac{\partial^2 f}{\partial x^2}\frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 \tag{4}$$

In order to use equation 4 in computer vision is two-dimensional function is replaced by luminance and second order partial derivatives are replaced with approximation made with use of convolution of image with derivatives of corresponding Gaussian kernels. Then Hessian matrix in equation 4 changes to following form.

$$H(x,y,\sigma) = \begin{pmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{xy}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{pmatrix} \tag{5}$$

Matrix element $L_{xx}(x, y, \sigma)$ stands for convolution of second order partial derivatives of two-dimensional Gaussian $/frac\partial^2 f(\sigma)/partial x^2$, where σ is the variance parameter of Gaussian kernels. These second order partial Gaussian derivatives have to be approximated in order to be applied on integral image. Approximation is made by box filters. Figure 5 displays discretised second order derivatives of Gaussian kernels on upper line and their representation using box filters on lower line.

If we establish D_{xx}, D_{yy}, D_{xy} as box filter approximations then equation 4 can be modified into form shown in equation 6.

$$\mathcal{H} \sim D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{6}$$

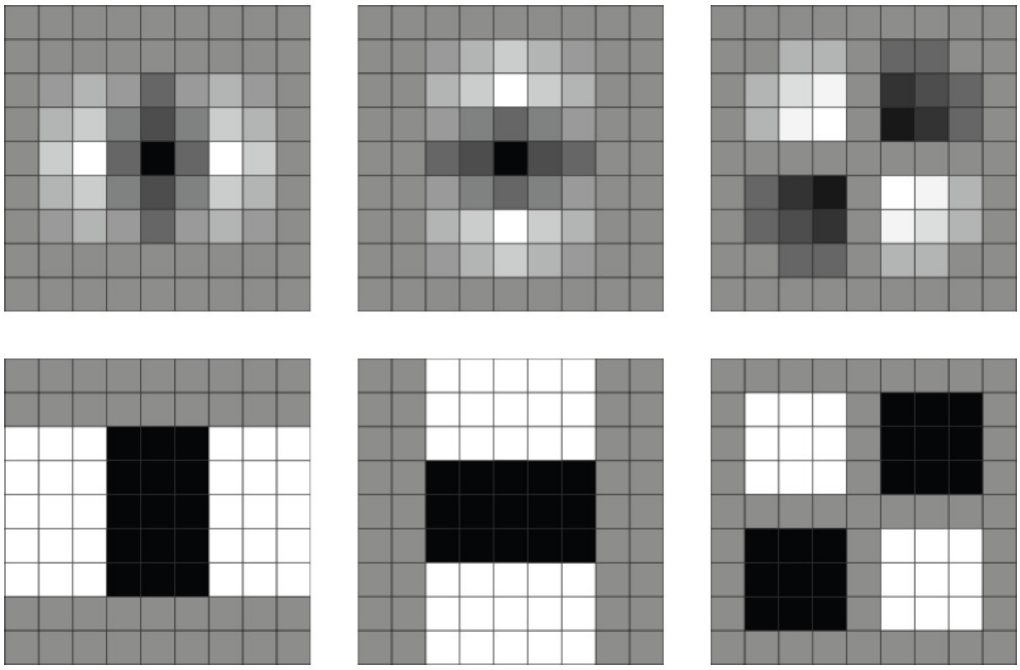


Figure 5: Discretised Gaussian kernel and their 2D approximations. Courtesy of [2]

Because of distortion due to approximation there is weighting mechanism compensating this distortion. It is constant equal to 0.9. The approximation and integral image usage brings SURF algorithm the biggest computation performance optimization.

Scale invariance of SURF algorithm is achieved by building scale space. To build scale space we use box filters with different sizes, which builds three-dimensional space of Hessian determinants. With built scale space the next step is comparing determinants with threshold value. If determinant surpasses given threshold it is compared with all it neighbours in scale space to ensure that it is local maxima. Passing both comparisons we have established image feature.

4.2 Feature description

With established feature it is necessary to somehow describe its surroundings in order to be able to match features. I will describe original SURF descriptor and BRIEF descriptor since it was implemented in SCHVAB MiniModule.

4.2.1 SURF descriptor

SURF descriptor generation is original part of SURF algorithm. Output is defined as vector of 64 floating point numbers. Surroundings of interest point is divided into 16 squares. If rotation invariance is required these squares are oriented in dominant direction.

Dominant direction is generated using Haar wavelet responses weighted with Gaussian centered at interest point. Responses are represented as points in two dimensional space. Points are then summed creating vector of orientation window with defined width $\frac{\pi}{3}$. Rotation step of orientation window is $\frac{\pi}{18}$. Orientation with the biggest summed vector is then declared as dominant direction.

Each square is sampled with Haar Wavelet responses. Afterwards wavelet responses are weighted with Gaussian centered at interest point with result of d_x for horizontal direction and d_y for vertical direction. Terms vertical and horizontal are used with relation to dominant orientation of interest point. The resulting values are united in form of a vector $[\sum dx, \sum dy, \sum |dx|, \sum |dy|]$. Those vectors from all 16 squares are then joined together and normalized to achieve euclidean length equal to one. This gives us the main part of SURF descriptor. Nevertheless there is yet another one useful component of SURF descriptor. It is the sign of Laplacian, trace of the Hessian matrix, which distinguishes if blob is bright on the dark background or the opposite case. This is very useful in matching features stage, where Euclidean distance between two vectors is computed.

4.2.2 BRIEF descriptor

The Binary Robust Independent Elementary Features (BRIEF) is binary string feature point descriptor which was proposed in paper [32]. This descriptor is based on comparison of pixel intensities within image patch surrounding feature point. In other words on given patch $\mathbf{p}$ with given coordinates of points to compare $\mathbf{x}, \mathbf{y}$ the test τ result is defined as

$$\tau(\mathbf{p}, \mathbf{x}, \mathbf{y}) := \begin{cases} 1 & \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0 & otherwise \end{cases}, \tag{7}$$

where $\mathbf{p}(\mathbf{x})$ is pixel intensity on given position. Number of comparisons denotes length of generated bitstring. In original paper they mention three version BRIEF-16, BRIEF-32 and BRIEF-64. The number denotes bytes count used to generate descriptor. That means BRIEF-64 length is 512 bits (512 tests). In comparison with SURF descriptor which consumes 256 bytes of memory. Very important part of BRIEF descriptor generation is a set of points which pixel intensities should be compared. Various types of set had been tested in original paper with conclusion that the best result is provided with randomly generated set of points. Example of this kind of set is shown on figure 6, where lines connects two point whose intensities are compared.

Original BRIEF descriptor does not provide scale and rotation invariance. However some modifications to BRIEF were made in order to implement scale and rotation invariance. Result is Binary robust invariant scalable keypoints (BRISK) descriptor [33]. Another upgrade of BRIEF is called D-BRIEF [34] made by authors of original BRIEF. D-BRIEF uses discriminative projections and is long only 32 bits. Another binary descriptor is BinBoost [35], which outperforms other binary descriptors and has comparable accuracy to floating point descriptors with only 64 bits of length.

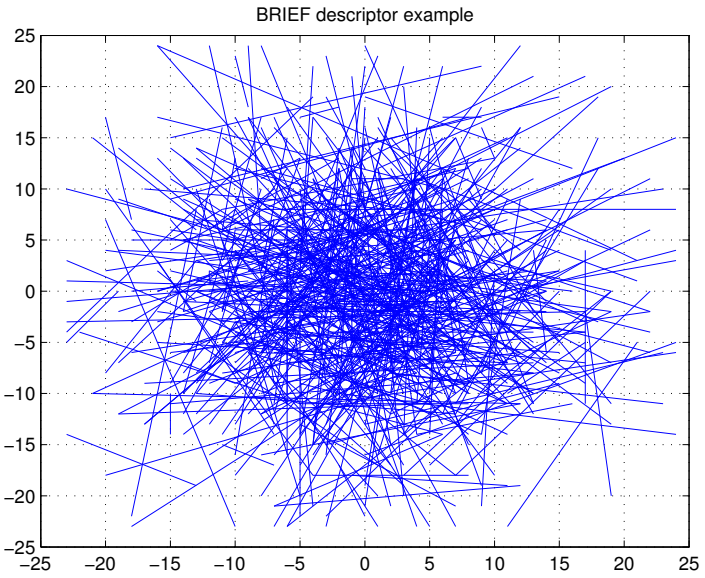


Figure 6: Example of BRIEF descriptor pairs.

Matching features using BRIEF descriptor is done by computing Hamming distance. This is one of the factors why it was decided to implement BRIEF descriptor on SCHVAB MiniModule. It was expected that it will be faster than computing euclidean distances, results of experiments can be found in chapter 6.

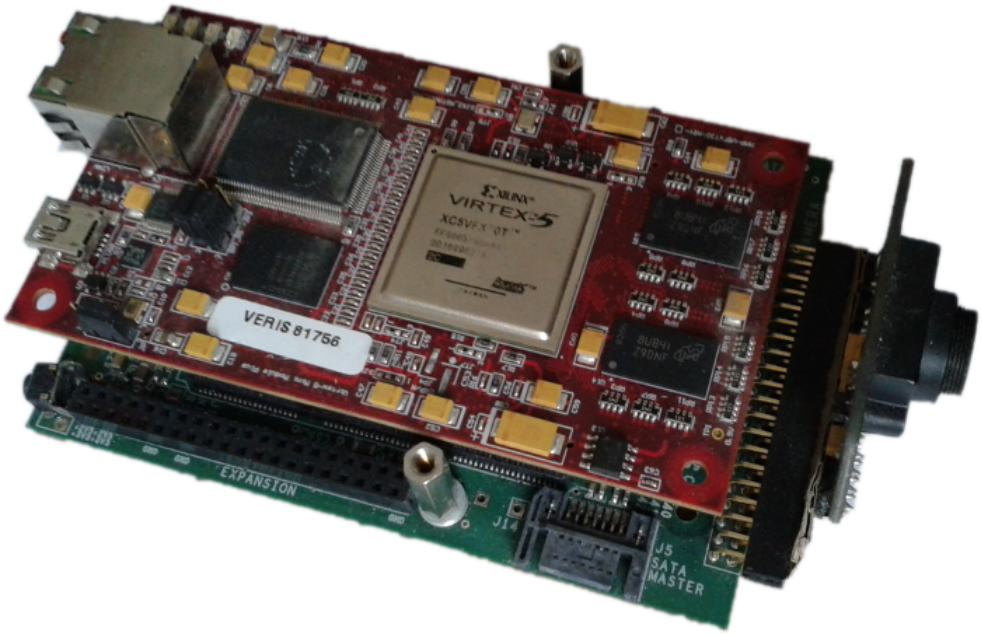


Figure 7: SCHVAB MiniModule

5 Device description

I was working with the Schvab FPGA-based Computer Vision Embeded Module (SFCVEM) designed by Ing. Jan Šváb [2]. It is based on AES-MMP-V5FXT70-G Avnet MiniModule Plus board which is attached to custom designed Schvab Mini-Module Small BaseBoard.

5.1 Hardware

AES-MMP-V5FXT70-G Avnet MiniModule Plus board is populated with Xilinx Virtex-5 XC5VFX70T-2FF665 FPGA. Following list summarizes the most important features of Avnet MiniModule. I just would like to point out that there is ARM architecture based PowerPC processor unit along with the FPGA.

- XC5VFX70T-2FF665

  PowerPC 440 Processor Core

  Auxiliary Processor Unit (APU) (automatically decode PowerPC floating-point instructions)

- 64MB DDR2 SDRAM
- 32MB FLASH

- USB 2.0 PHY
- 10/100/1000 Ethernet PHY (RJ-45)

Avnet provides baseboards for their minimodules but the original board is big and with unnecessary circuits thus custom board was built by Jan Šváb called Schvab Mini-Module Small BaseBoard. This board was built with respect to lower power consumption and fit needs of computer vision application. Schvab Mini-Module Small BaseBoard provides:

- JTAG Connector
- 4MB SSRAM
- SD Card Slot
- 2x SATA Connector
- Camera Module Connector
- Expansion Connector (23 pins)
- Power Supply

5.2 FPGA configuration

I will just briefly describe configuration and how data are processed. Main building blocks are called intellectual property (IP) cores. Majority of IP cores are designed by Jan Šváb and are fully described in [2]. Rest of IP cores are Standard Xilinx IP cores.

Schvab Pixel Bus (SPB) – Custom bus with single-master, multi-slave specification. Creates connection between blocks that contributes to image processing.

SPB_PDMAB – Creates bridge between SPB and PLB (Processor Local Bus). Allows user to store data (e.g. integral image, Fast-Hessian response) into system memory.

SPB_IIG – This IP core is generating the integral image. It has configurable bus widths with maximum output width 32-bits.

SPB_SAFHG – Provides computation of Fast-Hessian responses from integral image. This is on of the fundamental tasks while generating SURFs.

SPB_LMF – Purpose of this core is thresholding and non-maxima suppression of incoming data from SPB. This core buffers data and performs comparison sequences when buffer is full.

SPB_RCSIC – Subsamples and/or crop images processed by SPB subsystem without any delay.

SPB_DCFG – Interface from CMOS camera interface to SPB. Also contains counter of rows/columns for debugging purposes.

SSIO – GPIO core with 32 inputs and 32 outputs.

All these IP cores creates *processing chain* which is pipelined IP cores to perform some task. During my work i was using the same design with one slight change, second `UART 16550` for communication with MMP-5 Mobile Robot Platform [36] was added.

5.3 Linux implementation

To provide user-friendly interface GNU/Linux operating system was ported for this device. Original Linux implemented is based on Linux kernel of version 2.6. With this version I was working almost all the time, however I also tried to port latest version of Linux kernel available at Xilinx git server[1]. To be able of cross-compiling Linux kernel it is needed to build cross-compilation environment which consist of cross-compilation toolchain, buildroot for creation of filesystem and kernel source code. I used same toolchain as in original work described in [2], but i was also using ELDK that was recommended by Xilinx.

Most important part of Linux implementation is `spb_subsys.ko` kernel driver module. This driver module sets up all IP cores that are part of *processing chain* correspondingly to information about chain composition provided by user during the runtime. Part of the driver is also device tree parser, it means that it is possible to control all configured *processing chains* through device nodes in filesystem. Recognized *processing chains* are: the SURF accelerator chain (`chsa`) and the frame grabber chain (`chfg`). Driver also contains a memory manager. It is needed because image data need to be stored in continuous memory block of size of approximately 4MB. To be sure that this memory will be available it is allocated during the system boot and then distributed.

Processing chains

Detailed description of *processing chains* interfaces can be found in original thesis [2]. The frame grabber chain can be controlled through device nodes `/dev/spbss/chfg%d`. This chain is used to grab images from camera. For my work was important the other chain – SURF accelerator processing chain. Controlling device nodes can be found in `/dev/spbss/chsa%d`. This chain controls the cores responsible for image feature extraction and proceed in feature description. In order to implement BRIEF descriptor `ssa_get_bd`

[1] https://github.com/Xilinx/

and `i_sum` functions was added. First mentioned function `ssa_get_bd` is called instead of `ssa_get_sd` function to create BREIF descriptor. To generate BRIEF descriptor i used set of points from OpenCV 2.4.2 examples it can be found on figure 6. Function `i_sum` is just a C code representation of equation 3. It computes luminance of area in surroundings of input coordinates. Size of the area is defined as a `KERNEL_SIZE` constant. Currently is set to 9 pixels. This is done to reduce influence of noise in the image.

Other work

To device tree file was added second `UART 16550` IP cores and it was tested with MMP-5 Mobile Robot Platform. Great number of time was spent in attempt to make memory card slot working, however the `spi` core which is responsible for communication with memory cards was not responding in expected manner. As another try to make it work I upgraded to up-to-date linux kernel version 3.9.0. Since there are differences in driver management original drivers had to be updated as well. Nevertheless it did not help in the matter of memory card access, but `sps_subsys.ko` driver was working fine.

6 Experiments

6.1 BRIEF execution time

I have done some measurements to find out if computing hamming distance would be really faster than computing Euclidean distance, which includes multiplication of floating point numbers. Results are summed up in table 1. For computing hamming distance I used algorithm presented in [37]. Implementation of this algorithm in C language is shown in source code 1. This method execution time is dependent on Hamming distance between compared bitstrings, thus it was measured for different cases. Each test was comparing two descriptors for million times.

SURF	
Euclidean dist.	23.33s
BRIEF	
Hamming dist.	time[s]
512	25.79
384	20.31
256	14.82
128	9.33
0	3.68

Table 1: Executing time comparison.

```
int hammingDist(int * a, int * b){
    int xor = *a^*b;
    int dist = 0;
    while(xor){
        dist++;
        xor = xor & (xor - 1);
    }
    return dist;
}
```

Source code 1: Hamming distance computation function.

As we can see only the worst case scenario of computing Hamming distances is more time consuming than computing Euclidean distance. To be sure I also created program for matching descriptors between following images on SCHVAB MiniModule and I gathered data of all computed Hamming distances. On figure 8 we can see histogram for Hamming distances from 0 to 512 and fitted normal distribution. Based on this data we can assume that matching features with BRIEF descriptors is faster than with original SURF descriptor.

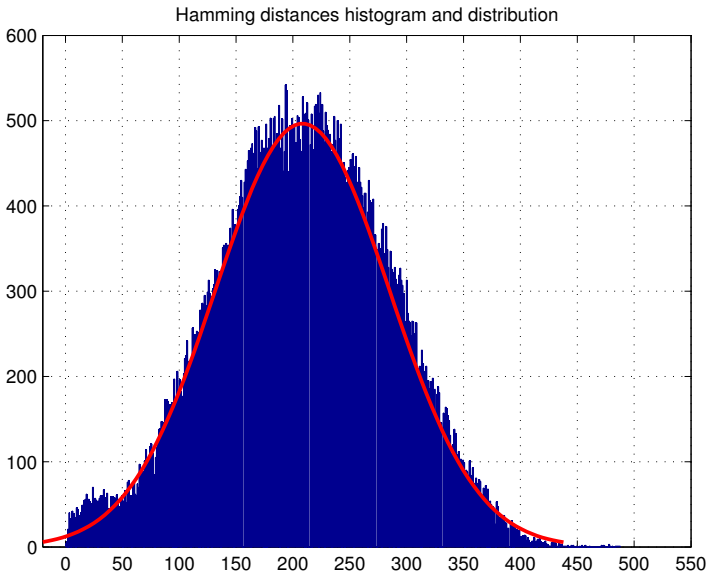


Figure 8: Hamming distances histogram and distribution.

6.2 Long-term navigation test

This test is done to measure performance of feature detector in long-term scenario of outdoor navigation. Methods of this test are described in [3]. The aim is to track results of various feature detectors in matching algorithm to establish difference in heading. This test basically uses methods described in chapter about SURFNav. The dataset consist of images taken in five locations and through whole year each month. In figure 9 is shown example of seasonal changes recorded in dataset. Figure 10 shows remaining locations used in dataset.

(a) November 2009 (b) January 2010 (c) May 2010 (d) Semptember 2010

Figure 9: Example of seasonal changes. Courtesy of [3].

(a) Location I. (b) Location III. (c) Location IV. (d) Location V.

Figure 10: The dataset locations. Courtesy of [3].

Measured data are compared with data provided with thesis supervisor. We can see in figure 11 that implemented BRIEF descriptor is only outperformed by BRIEF combined with STAR feature detector [38]. And if we look for SURF we can see that there is a significant gap between SURF and FADR, which is representing my BRIEF implementation on SCHVAB MiniModule.

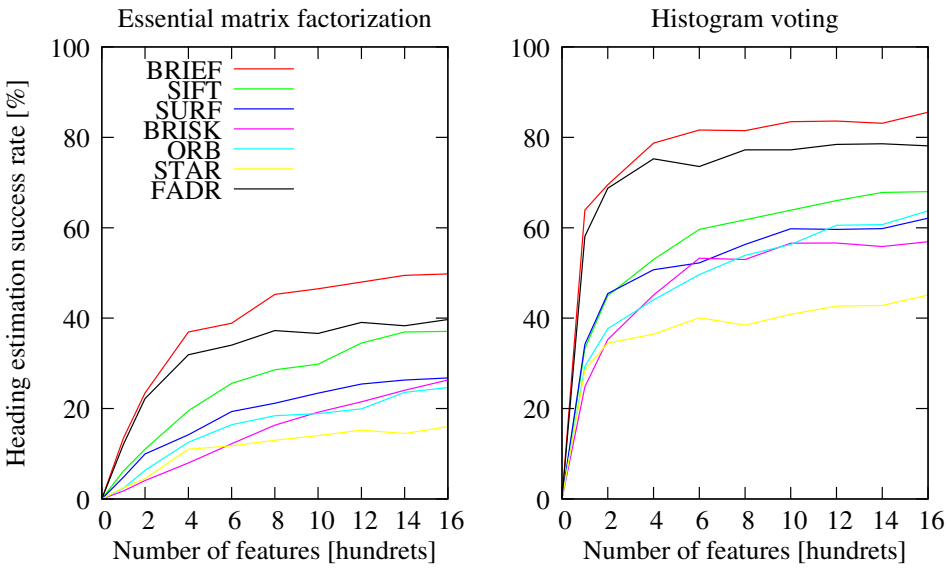


Figure 11: Heading estimation success rates.

6.3 Distinguishability measurements

Distinguisability is understood as a chance of establishing valid correspondence. It was measured on dataset counting 5000 images taken from drive of robot. Two following images are always processed. At first are created tentative correspondences based on hamming distance. After that is established fundamental matrix with the use of RANSAC [39]. Fundamental matrix is used to project points from tentative correspondences and those point that are projected closer than 3 pixels to epipolar line are considered valid. Measured value is ratio between tentative correspondences and correct ones. I have measured it for two cases. In first case it is used also Laplacian sign to determine tentative correspondences, in second one just Hamming distances.

Laplacian used	Ratio
Y	0.98
N	0.97

Table 2: Distinguishability

7 Conclusion

In this thesis I have shown some improvement of originally implemented image processing chain. SURF descriptor along with BRIEF descriptor provide much better results. Moreover in feature matching phase will be BRIEF descriptor on most cases faster then matching SURF descriptors.

Another serial port communication was added in order to be able control MMP-5 robot. What bothers me is not working memory card slot. I have tried many things, but nothing worked. This would be very useful for storing/loading maps for SURFNav. I have also decided that that SURFNav is the most suitable navigation algorithm for SCHVAB MiniModule.

During work on this thesis i learnt a lot of things about GNU/Linux and how it works which i consider very useful experience.

References

[1] Hana Szücsová. Computer Vision-based Mobile Robot Navigation. Master's thesis, Czech Technical University, Czech Republic, 2011.

[2] Jan Šváb. FPGA-based Computer Vidoin Embedded Module. Master's thesis, Czech Technical University, Czech Republic, 2011.

[3] Tomáš Krajnık1 Pablo Cristóforis2 Jan Faigl, Hana Szücsová1 Matıas Nitsche2 Libor Preucil, and Marta Mejail. Image features for long-term mobile robot autonomy. 2013.

[4] Tomáš Krajník. *Large Scale Mobile Robot Navigation and Map Building*. PhD thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2011.

[5] Johann Borenstein, HR Everett, Liqiang Feng, and David Wehe. Mobile robot positioning-sensors and techniques. Technical report, DTIC Document, 1997.

[6] George J Geier, Ardalan Heshmati, Kelly G Johnson, and Patricia W McLain. Position and velocity estimation system for adaptive weighting of gps and dead-reckoning information, May 16 1995. US Patent 5,416,712.

[7] Lenka Mudrová, Jan Faigl, Jaroslav Halgašík, and Tomáš Krajník. Estimation of mobile robot pose from optical mouses. In *Research and Education in Robotics-EUROBOT 2010*, pages 93–107. Springer, 2011.

[8] Johann Borenstein, HR Everett, and Liqiang Feng. Where am i? sensors and methods for mobile robot positioning. *University of Michigan*, 119:120, 1996.

[9] Chin-Woo Tan and Sungsu Park. Design of accelerometer-based inertial navigation systems. *Instrumentation and Measurement, IEEE Transactions on*, 54(6):2520–2530, 2005.

[10] Hakyoung Chung, Lauro Ojeda, and Johann Borenstein. Accurate mobile robot dead-reckoning with a precision-calibrated fiber-optic gyroscope. *Robotics and Automation, IEEE Transactions on*, 17(1):80–84, 2001.

[11] Ubisense Real time Location Systems (RTLS). Rtls solutions - ubisense. `http://www.ubisense.net/en/rtls-solutions`. Visited on 2013-11-04.

[12] Bernt Schiele and James L Crowley. A comparison of position estimation techniques using occupancy grids. *Robotics and autonomous systems*, 12(3):163–171, 1994.

[13] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[14] Margrit Betke and Leonid Gurvits. Mobile robot localization using landmarks. *Robotics and Automation, IEEE Transactions on*, 13(2):251–263, 1997.

[15] Sebastian Thrun. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1:1–35, 2003.

[16] Thomas Collins, JJ Collins, and Conor Ryan. Occupancy grid mapping: An empirical evaluation. In *Control & Automation, 2007. MED'07. Mediterranean Conference on*, pages 1–6. IEEE, 2007.

[17] Sv Noykov and Ch Roumenin. Occupancy grids building by sonar and mobile robot. *Robotics and autonomous systems*, 55(2):162–175, 2007.

[18] Miroslav Kulich. *Localization and Map Building for Intelligent Robots*. PhD thesis, Czech Technical University in Prague, Faculty of Electrical Engineering, 2004.

[19] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.

[20] Stephen Se, David Lowe, and Jim Little. Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks. *The international Journal of robotics Research*, 21(8):735–758, 2002.

[21] Francisco Bonin-Font, Alberto Ortiz, and Gabriel Oliver. Visual navigation for mobile robots: A survey. *Journal of intelligent and robotic systems*, 53(3):263–296, 2008.

[22] RE Arvidson, SW Squyres, RC Anderson, JF Bell, D Blaney, J Brückner, NA Cabrol, WM Calvin, MH Carr, PR Christensen, et al. Overview of the Spirit Mars exploration rover mission to Gusev crater: Landing site to Backstay rock in the Columbia Hills. *Journal of Geophysical Research: Planets (1991–2012)*, 111(E2), 2006.

[23] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21(3):376–386, 2005.

[24] Paul Furgale and Tim Barfoot. Visual path following on a manifold in unstructured three-dimensional terrain. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 534–539. IEEE, 2010.

[25] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[26] T. Krajník and L. Přeučil. A Simple Visual Navigation System with Convergence Property. In *European Robotics Symposium 2008*, pages 283–292, Heidelberg, 2008. Springer.

[27] T. Krajník, J. Faigl, M. Vonásek, V. Kulich, K. Košnar, and L. Přeučil. Simple yet stable bearing-only navigation. *J. Field Robot.*, 2010.

[28] MWM Gamini Dissanayake, Paul Newman, Steven Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (slam) problem. *Robotics and Automation, IEEE Transactions on*, 17(3):229–241, 2001.

[29] Miguel Angel Garcia and Agusti Solanas. 3d simultaneous localization and modeling from stereo vision. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 1, pages 847–853. IEEE, 2004.

[30] Michael Bosse, Paul Newman, John Leonard, and Seth Teller. Simultaneous localization and map building in large-scale cyclic environments using the atlas framework. *The International Journal of Robotics Research*, 23(12):1113–1139, 2004.

[31] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[32] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: binary robust independent elementary features. In *Computer Vision–ECCV 2010*, pages 778–792. Springer, 2010.

[33] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[34] Tomasz Trzcinski and Vincent Lepetit. Efficient discriminative projections for compact binary descriptors. In *Computer Vision–ECCV 2012*, pages 228–242. Springer, 2012.

[35] Tomasz Trzcinski, Christos Marios Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. 2013.

[36] Inc. The Machine Lab. Mmp-5 mobile robot platform. `http://www.themachinelab.com/MMP-5.html`. Visited on 2013-12-15.

[37] Peter Wegner. A technique for counting ones in a binary computer. *Communications of the ACM*, 3(5):322, 1960.

[38] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *Computer Vision–ECCV 2008*, pages 102–115. Springer, 2008.

[39] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Appendix

CD Content

In table 3 are listed names of all root directories on CD

Directory name	**Description**
bp	bachelor thesis in pdf format.
benchmark	source code of benchmarks used to compare computing Euclidean and Hamming distance
matchtest	source code of picture matching test to obtain Hamming distances
2.6-driver	spb_subsys driver for 2.6 linux kernel version
3.9-driver	spb_subsys driver for 3.9 linux kernel version
virtex440-sfcvem.dts	updated device tree file

Table 3: Obsah CD