Stereo camera pose estimation to enable loop detection
- Detecting previously visited scenes for relocalization

Estimering av kamera-pose i stereo för att återupptäcka besökta platser

Viktor Ringdahl

Supervisor : Mikael Persson
Examiner : Per-Erik Forssén
Upphovsrätt


Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances. The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/her own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility. According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Viktor Ringdahl
Abstract

Visual Simultaneous Localization And Mapping (SLAM) allows for three dimensional reconstruction from a camera’s output and simultaneous positioning of the camera within the reconstruction. With use cases ranging from autonomous vehicles to augmented reality, the SLAM field has garnered interest both commercially and academically.

A SLAM system performs odometry as it estimates the camera’s movement through the scene. The incremental estimation of odometry is not error free and exhibits drift over time with map inconsistencies as a result. Detecting the return to a previously seen place, a loop, means that this new information regarding our position can be incorporated to correct the trajectory retroactively. Loop detection can also facilitate relocalization if the system loses tracking due to e.g. heavy motion blur.

This thesis proposes an odometric system making use of bundle adjustment within a keyframe based stereo SLAM application. This system is capable of detecting loops by utilizing the algorithm FAB-MAP. Two aspects of this system is evaluated, the odometry and the capability to relocate. Both of these are evaluated using the EuRoC MAV dataset, with an absolute trajectory RMS error ranging from 0.80 m to 1.70 m for the machine hall sequences.

The capability to relocate is evaluated using a novel methodology that intuitively can be interpreted. Results are given for different levels of strictness to encompass different use cases. The method makes use of reprojection of points seen in keyframes to define whether a relocalization is possible or not. The system shows a capability to relocate in up to 85% of all cases when a keyframe exists that can project 90% of its points into the current view. Errors in estimated poses were found to be correlated with the relative distance, with errors less than 10 cm in 23% to 73% of all cases.

The evaluation of the whole system is augmented with an evaluation of local image descriptors and pose estimation algorithms. The descriptor SIFT was found to perform best overall, but demanding to compute. BRISK was deemed the best alternative for a fast yet accurate descriptor.

Conclusions that can be drawn from this thesis is that FAB-MAP works well for detecting loops as long as the addition of keyframes is handled appropriately.
Acknowledgments

I would like to extend my gratitude to SAAB for giving me the opportunity to do this thesis, and in particular to Fredrik Lundell for all the guidance and help when things behaved strangely. I would also like to thank Jimmy Jonsson at SAAB for the helpful pointers of where to find things in the code base and the additional help with calibration.

At LiU I would like to extend my thanks to my supervisor, Mikael Persson at Computer Vision Laboratory, who gave me several valuable pointers toward algorithms and additions to consider for the system.
Contents

Abstract iii
Acknowledgments iv
Contents v
List of Figures vi
List of Tables ix
1 Introduction 1
  1.1 Motivation ........................................... 1
  1.2 Aim .................................................. 2
  1.3 Research questions .................................... 2
  1.4 Delimitations ......................................... 3
2 Related Research 4
  2.1 Theoretical background ............................... 4
  2.2 Method ............................................... 12
  2.3 Summary and rationale behind a priori choices .......... 14
3 Method 15
  3.1 Evaluation phase ..................................... 15
  3.2 Implementation & the system ........................ 17
  3.3 Final evaluation phase ............................... 22
4 Results 24
  4.1 Evaluation phase ..................................... 24
  4.2 Final evaluation ..................................... 28
5 Discussion 37
  5.1 Results ............................................... 37
  5.2 Method ............................................... 45
  5.3 The work in a wider context .......................... 48
6 Conclusion 50
  6.1 Future work ......................................... 51
Bibliography 52
A Appendix 59
  A.1 Additional Results ................................... 59
  A.2 Varying keyframe insertion frequency ................. 62
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A 3D reconstruction using the proposed system. The reconstruction was created from a sequence walking around a house. The pair of cameras are about to return to the initial position, and thus creating a loop. Red markers indicate estimated camera poses.</td>
</tr>
<tr>
<td>2.1</td>
<td>An illustration of a pinhole camera model, axis convention is the same as used by the OpenCV library. A point with coordinates ((X, Y, Z)) is projected onto ((u, v)) given the intrinsic parameters of the model, focal length and principal point. Recreation of an image that is a part of the BSD-3 licensed OpenCV documentation.</td>
</tr>
<tr>
<td>2.2</td>
<td>An illustration of a camera pose graph with camera poses represented as triangles. Dashed line with text indicates that the cameras can see common features, thus should trigger a loop detection.</td>
</tr>
<tr>
<td>2.3</td>
<td>Detector model used by FAB-MAP. Location (L_i) generates latent words (e_1, e_2, \ldots, e_n). Latent words are are not observed directly and instead generate the visual word detections, (z_1, z_2, \ldots, z_n) (modeling e.g. false positives).</td>
</tr>
<tr>
<td>3.1</td>
<td>A block diagram overview of the system, where each block represents a collection of functionality relating to a particular task. Dashed lines indicate things that sometimes but not always occur while solid lines always occur.</td>
</tr>
<tr>
<td>3.2</td>
<td>To the left, ANMS filtered AGAST key points. To the right, the original key points. Do however note that some of the furthest away detected key points have been removed by a separate process due to being too far away.</td>
</tr>
<tr>
<td>3.3</td>
<td>Flowchart describing the selection of navigation frame. Dashed outlines of boxes indicate things occurring elsewhere (i.e. later) in the system that impact whether the its state is lost or not. The circle indicate the start point with each new frame.</td>
</tr>
<tr>
<td>3.4</td>
<td>Flowchart describing the keyframe insertion scheme. The circle indicate the start point with each new frame.</td>
</tr>
<tr>
<td>4.1</td>
<td>Precision plotted against recall for a frame undergoing both translation and rotation ((\text{the sequence DESK1}), \text{frame two}). Translation and rotation relative to reference frame is 0.05 m and 0.07 rad respectively.</td>
</tr>
<tr>
<td>4.2</td>
<td>Precision plotted against recall for a frame undergoing both translation and rotation ((\text{the sequence DESK1}), \text{frame four}) with a larger transformation than for Fig. 4.1, frame four. Translation and rotation relative to reference frame is 0.10 m and 0.15 rad respectively.</td>
</tr>
<tr>
<td>4.3</td>
<td>Precision plotted against recall for a frame undergoing both translation and rotation ((\text{the sequence DESK1}), \text{frame six}) with a larger transformation than for Fig. 4.2, frame six. Translation and rotation relative to reference frame is 0.16 m and 0.21 rad respectively.</td>
</tr>
<tr>
<td>4.4</td>
<td>Precision plotted against recall for a frame undergoing both translation and rotation ((\text{the sequence DESK1}), \text{frame eight}) Translation and rotation relative to reference frame is 0.23 m and 0.21 rad respectively.</td>
</tr>
<tr>
<td>4.5</td>
<td>Precision-recall curves for the test image “+3 Overexposed”.</td>
</tr>
<tr>
<td>4.6</td>
<td>Precision-recall curves for the test image “-4 Underexposed”.</td>
</tr>
</tbody>
</table>
4.7 Precision-recall curves for the test image “Directional + 0.4 Uniform” ................................. 27
4.8 Precision-recall curves for the test image “Directional illumination only” .............................. 27
4.9 Translational errors as percentage of translational movement in a sequence moving back and forth, the sequence XYZ1 ................................................................. 30
4.10 Rotational errors as percentage of the angle rotated in a sequence rotating while stationary, the sequence RPY-ROLL ................................................................. 30
4.11 Translational errors as percentage of translational movement in a sequence both rotating and translating the camera, the sequence DESK1 .............................................. 30
4.12 Rotational error as a percentage of the angle rotated in a sequence rotating and translating the camera, the sequence DESK1 ...................................................... 30
4.13 The estimated trajectory for proposed and original system along X-Y axis overlaid with ground truth. The sequence is MH01 ................................................................. 32
4.14 The estimated trajectory for proposed and original system along X-Y axis overlaid with ground truth. The sequence is MH02 ................................................................. 32
4.15 The estimated trajectory for proposed and original system along X-Y axis overlaid with ground truth. The sequence is MH03 ................................................................. 32
4.16 The estimated trajectory along X-Y axis overlaid with ground truth. The sequence is MH03. Here AKAZE is used as key point detector and descriptor extractor as opposed to AGAST and BRISK, respectively ................................................................. 32
4.17 Distance error between ground truth and estimated translations between a matched keyframe and test frame. The sequence is MH01. A pose is estimated in 28 out of the 1000 test frames, yielding an estimation percentage of 2.8%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is ≈ 64% ................................................................. 34
4.18 Distance error between ground truth and estimated translations between a matched keyframe and test frame. The sequence is MH02. A pose is estimated in 11 out of the 500 test frames, yielding an estimation percentage of 2.2%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is ≈ 55% ................................................................. 34
4.19 Distance error between ground truth and estimated translations between a matched keyframe and test frame. The sequence is MH03. A pose is estimated in 94 out of the 600 test frames, yielding an estimation percentage of approximately 15.7%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is ≈ 23% ................................................................. 34
4.20 Distance error between ground truth and estimated translations between a matched keyframe and test frame. Here, AKAZE detector and extractor is used instead of AGAST and BRISK. The sequence is MH03. A pose is estimated in 97 out of the 600 test frames, yielding an estimation percentage of approximately 16.2%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is ≈ 53% ................................................................. 34
4.21 The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. The sequence is MH01 ........................................................................................................................................ 35
4.22 The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. The sequence is MH02 ........................................................................................................................................ 35
4.23 The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. The sequence is MH03 ........................................................................................................................................ 35
4.24 The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. AKAZE key points and descriptors are used instead of AGAST and BRISK respectively. The sequence is MH03.

4.25 A 3D plot showing the ground truth trajectory, colorized depending on whether the mapping was active or if not. Green dashed lines represent the detection of a keyframe that could be used to successfully estimate a pose. The sequence is MH01.

4.26 A 3D plot showing the ground truth trajectory, colorized depending on whether the mapping was active or if not. Green dashed lines represent the detection of a keyframe that could be used to successfully estimate a pose. The sequence is MH02.

4.27 A 3D plot showing the ground truth trajectory, colorized depending on whether the mapping was active or if not. Green dashed lines represent the detection of a keyframe that could be used to successfully estimate a pose. The sequence is MH03.

5.1 Reconstruction from running SLAM on a recorded sequence walking around a house while looking at it. Red markers are estimated poses.

5.2 Reconstruction from running SLAM on a recorded sequence walking around a house while looking at it, focused on the mesh where overlap between start and end exists. Note they do not perfectly overlap, indicating loop closing is required.

5.3 AKAZE key points extracted from an “RGB-D SLAM dataset” [sturm12iros] frame. Original frame licensed under Creative Commons 4.0 Attribution License (CC BY 4.0). The key points are in most cases fairly well separated. The circles have a radius of 3.0 pixels.

A.1 Precision-recall curve for the test image “+1 Overexposed”.
A.2 Precision-recall curve for the test image “+2 Overexposed”.
A.3 Precision-recall curve for the test image “+4 Overexposed”.
A.4 Precision-recall curve for the test image “-1 Underexposed”.
A.5 Precision-recall curve for the test image “-2 Underexposed”.
A.6 Precision-recall curve for the test image “-3 Underexposed”.
A.7 Precision-recall curve for the test image “Directional + Uniform”.
A.8 Precision-recall curve for the test image “Directional + 0.8 Uniform”.
A.9 Precision-recall curve for the test image “Directional + 0.6 Uniform”.
A.10 Precision-recall curve for the test image “Directional + 0.2 Uniform”.
A.11 Distance error between ground truth and estimate translations between a matched keyframe and test frame. The sequence is MH03 and the setting is “More”. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is $\approx 38\%$.
A.12 Distance error between ground truth and estimate translations between a matched keyframe and test frame. The sequence is MH03 and the setting is “Most”. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is $\approx 35\%$.
A.13 Distance error between ground truth and estimate translations between a matched keyframe and test frame. The sequence is MH01 and the setting is “Most”. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is $\approx 73\%$.
A.14 The recall for various threshold values, given different $d_{max}$. The sequence is MH03 and the setting is “More”.
A.15 The recall for various threshold values, given different $d_{max}$. The sequence is MH03 and the setting is “Most”.
A.16 The recall for various threshold values, given different $d_{max}$. The sequence is MH01 and the setting is “Most”.
List of Tables

4.1 Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence XYZ1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top three of time in bold. .................................................. 27

4.2 Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence RPY-ROLL, additionally the fraction of successful estimations are shown as $f_{ok}$. Top three values in bold, unless tied. ................................. 27

4.3 Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top three in bold. .......................................................... 28

4.4 Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK2, additionally the fraction of successful estimations are shown as $f_{ok}$. Top two in bold, unless tied. ........................................ 28

4.5 Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence XYZ1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top two in bold, unless tied. ........................................ 29

4.6 Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top two in bold, unless tied. ........................................ 29

4.7 Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK2, additionally the fraction of successful estimations are shown as $f_{ok}$. Top two in bold, unless tied. ........................................ 29

4.8 The result of the odometry on the three sequences, using proposed system. .................................. 31

4.9 The result of the odometry on the three sequences, using original system. .................................. 31

4.10 Mean computation times per frame for different parts of the system. ........................................... 32

4.11 Detailed timing statistics for processing a whole frame .............................................................. 33

4.12 The sequences used for relocalization evaluation. ................................................................. 33

4.13 Results when relaxing keyframe insertion constraints. All errors are in meters. Recall is given for $\tau_{side} = 0.9$ and $d_{max} = 0.5$ m. .................................................. 33

4.14 Parameter values for different keyframe insertion frequency settings. Distances are in meters and angles in radians. ................................................................. 33

5.1 Absolute trajectory RMSE comparison. ......................................................................................... 42
Introduction

Computer vision has many applications in today’s society – from medicinal applications that help save lives to immersive Augmented Reality (AR) applications and games. Computer vision encompasses tasks such as object detection and 3D reconstruction using photographs or video sequences, with much in between.

State-of-the-art video 3D reconstruction systems are able to navigate, reconstruct, and augment worlds created in real time using cameras with impressive results. These kinds of applications perform what is referred to as Simultaneous Localization and Mapping (SLAM), or more specifically visual SLAM. SLAM, as the name implies, simultaneously navigates and constructs a map. A visual SLAM system estimates the current camera position and orientation, i.e. the camera pose, in order to incorporate new information into the map, thus extending it. The act of tracking the camera pose over time using visual data is referred to as performing visual odometry.

The camera pose has six degrees of freedom when treating the camera as a rigid body under unconstrained motion. Three components describe the camera translation ($x$, $y$, and $z$) and three describe the rotation (e.g. yaw, pitch, and roll).

1.1 Motivation

The odometry is rarely perfect, e.g. noise gives rise to errors in the poses estimated which accumulate over time. For SLAM this means that once we return to a place previously mapped objects might appear elsewhere compared to where they were originally seen. The map has “drifted”, and minimizing this drift is desirable, but eliminating it completely is unfortunately not a realistic proposition.

Knowing if the system is observing a place that has already been seen before provides useful information. If the system knows where it should be then errors can be compensated for retroactively. The detection of returning to a previously seen place is referred to as loop detection and the act of fixing the errors is referred to as loop closing. Fig. 1.1 shows an example of a loop about to occur.

Detecting loops is useful in order to facilitate “relocalization” when tracking failure occurs, e.g. due to excessive motion blur. In such cases the system can continue if and when a previously visited place can be identified.
1.2. Aim

Performing visual odometry for an existing SLAM system so that loops can be detected, and subsequently closed, is the focus of this thesis. This also includes being capable of relocating.

1.2 Aim

Investigating ways of finding a camera’s pose for the use in a visual SLAM application is as mentioned the main goal of this thesis. Detecting loops and estimating pose should be done as efficiently and accurately as possible. Efficient here refers to computation time and accurate refers to the quality of the estimated poses.

SLAM systems often makes use of keyframes to navigate against. Exactly what they are used for depends on the SLAM method – but often it is these who are drawn in visualization applications. Keyframes can also be used to detect loops, even if drift has caused the system to consider them apart. Keyframes are used in this system as well and these will be utilized as reference states when detecting loops.

1.3 Research questions

The question to which an answer is sought is

- How can six-degrees of freedom camera pose estimation be performed in order to allow for relocalization and loop detection?

This is a very open-ended question, one without an explicitly correct answer. There are many ways to tackle and answer this question depending on requirements. As such, multiple smaller research questions are formulated based on this – using approaches established in literature. Visual SLAM systems are large systems, where even subsystems can be large and complex. Some choices are hence made a priori based on existing studies and with inspiration
taken from other successful systems. A broad adoption made is that the system utilizes keyframes.

*Local image features* are utilized for odometry and loop detection. These are points in images considered *distinct*, from which a *feature descriptor* can be extracted that describes this point. These points can then be used to both estimate the pose of the camera and detect already seen places.

The algorithm FAB-MAP \[20\] is used as basis to be able to detect if the system is observing a previously seen place or not.

More detailed research questions are formulated based on these a priori choices. These research questions aim to investigate parts of the system in more detail. Other research questions are formulated from choices left open, with evaluations deciding how to best proceed. These more detailed research questions in focus are:

- **What feature descriptor should be used that can both be accurate enough and be fast enough to compute?**
- **What pose estimation method should be used that achieves a balance between accuracy and computation time?**
- **How often should keyframes be inserted to provide sufficiently many for localization, yet not impose penalties? And what are these penalties?**

Note that limits are omitted from the research questions to be open to trade-offs between different parts in the system. The objective is real-time or near real-time capabilities.

The whole goal of this project is to propose and implement a visual odometry system that can deal with relocalization and detect loops. As such it is also relevant to answer the question

- **How well does the proposed system perform odometry and relocalization?**

### 1.4 Delimitations

This thesis focuses specifically on topics related to loop detection and relocalization, and some aspects of a SLAM system are thus ignored. Notably this means that closing loops once they have been detected is not investigated. The major reason for this delimitation is due to another master thesis occurring at the same time that is investigating this issue in particular.

As hinted at when formulating the research questions, two choices delimits this thesis. These two choices are a keyframe based system and utilizing local image features. Direct methods instead of feature based methods have been proposed, such as *Direct sparse odometry* (DSO) \[22\], but are thus not considered here.

It is also assumed here that stereo images have been rectified and that the depth of any valid image point is available or calculable. By extension it is thus assumed that the cameras are calibrated and the images distortion free.

Only visual data from two cameras is used, no motion estimation using an inertial measurement unit (IMU) is performed.

Evaluation is only done for some algorithms, and is in no way exhaustive. Pose estimation methods are focused on 2D-3D methods and 3D-3D, 2D-2D methods are thus ignored (e.g. estimating a homography). This delimitation is made to leverage the fact that the depth is available for points in each frame. Refer to e.g. \[1\] for an evaluation of 2D-2D methods.
2 Related Research

This chapter begins with a brief overview of the necessary background in Sec. 2.1. Following this, in Sec. 2.2, are related research and methodology that this work is based on. This chapter concludes with a short summary and rationale behind a priori choices made during a pilot study phase.

2.1 Theoretical background

Finding where a camera is located and how it is oriented within a known “map” can be considered a six degrees of freedom (6DoF, rigid) pose estimation problem. The map in this context is a collection of three dimensional landmark points, \( X \in \mathbb{R}^3 \). The camera pose can be estimated using these points. For certain calculations it is convenient to represent poses with a homogeneous \( 4 \times 4 \) transformation matrix, denoted

\[
(R, t) = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix},
\]

where \( R \) is a \( 3 \times 3 \) rotation matrix and \( t \) is a \( 3 \times 1 \) translation matrix (i.e. a vector).

All cameras are not equal, and different camera models are suited for different types of cameras. Examples include pinhole camera models and fisheye camera models, with the former being used in this thesis. Fig. 2.1 illustrates the model, which can be described by a calibration matrix \( K \). As calibration is omitted in this thesis (see Delimitations) and for the sake of brevity, \( K \) will here be assumed to be a normalized calibration matrix, i.e.

\[
K = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.
\]

The focal lengths \( f_u \) and \( f_v \) will thus be assumed to be one and the principal point \( (c_u, c_v) \) in the image center, at \((0, 0)\).

\( K \) together with the camera pose form the camera matrix \( P = K(R, t) \). Projection of a 3D point \( X \) to a camera as \( x \in \mathbb{R}^2 \) is here denoted with \( f(X) = x \), assuming that \( X \) is expressed in the camera coordinate system.
2.1. Theoretical background

The inverse, projecting the 2D point $x$ to 3D is denoted with the inverse projection function, $f(x, Z)^{-1} = X$. This function relies on the depth at $x, Z$, being available, e.g. from a depth map or by calculating depth from disparity.

With a normalized pinhole camera model $f(X)$ and $f(x, Z)^{-1}$ can in the homogeneous case be written as

$$f(X) = x \Rightarrow f(X) = \begin{pmatrix} \frac{x_1}{X_3} \\ \frac{x_2}{X_3} \\ 1 \end{pmatrix}^T$$  \hspace{1cm} (2.3)

and

$$f(x, Z)^{-1} = X \Rightarrow f(x, Z)^{-1} = \begin{pmatrix} x_1Z \\ x_2Z \\ Z \end{pmatrix}^T.$$  \hspace{1cm} (2.4)

Given a known correspondence between points in an image, $x_1, ..., x_k$, and the 3D points $X_1, ..., X_k$ the 6DoF pose can be recovered by one of numerous algorithms. Examples include using a perspective three-point algorithm (P3P) [29] which uses three points ($k = 3$) to estimate poses.

Methods such as P3P are sensitive to outliers and noise and something more robust is thus needed [79, Ch. 6]. Additionally, the true correspondence between image points, $x_i$, and map points, $X_j$, is generally not known with certainty.

Points both in frame images and 3D points have been assumed to be known a priori. It stands to reason that not all points with neighboring region within an image provide useful information (consider for example an image featuring a uniformly white wall). Points in an image that are “distinct” can be found using a key point detector, aka interest point detector [71]. Regions around these points can then be used to extract descriptors to obtain a “signature” for each of the detected key points.

With stereo or RGB-D, 3D points can be added to a map by projecting a 2D point into the map using the depth at the point using Eq. 2.4 followed by transforming the points with the camera’s ($R, t$). The depth can be calculated either by triangulation of the point between the two cameras (using a known stereo configuration) for stereo or using the depth directly for RGB-D.

3D map points are added to the map based on detected key points. This means that associations between 2D and 3D can be done using a descriptor extracted from an image and the descriptor the 3D point originally had when it was identified.

Numerous detectors and descriptors have been proposed [40, 30], with some being binary to enable fast distance calculations and hence matching (using the Hamming distance). Descriptors are presented in more detail shortly.

The type of visual SLAM used here makes use of keyframes, “snapshots” of frame poses and other information that relates to them [95, 58]. Detecting if the currently observed scene (i.e. image) corresponds to a previously observed keyframe is the act of detecting a loop. The
2.1. Theoretical background

keyframe’s descriptors and the corresponding 3D points can when such a detection is made be used to estimate the pose relative to that keyframe. This can achieve either relocation or provide a destination for loop closing. This detection, like other topics, is presented in more details in the following sections.

Keyframes as a graph

Consider keyframes, identified by a number of keypoints, their descriptors and a camera pose, and let the keyframes be represented by nodes in a graph. Let edges between nodes be represented by some kind of similarity measure between nodes, e.g. sharing a certain number of feature descriptors, similarity in appearance, or a transition was made from one frame to another.

This or a similar representation of a map (i.e. as a topological graph, whether explicitly or implicitly) is common for localization tasks as it means advantage can be taken from well established document/image retrieval methods. It also conceptually provides an easy way of representing and closing loops (with e.g. pose graph optimization) by connecting two nodes with an edge. An illustration of this can be found in Fig. 2.2, where two of the poses should trigger a loop detection and thus be connected with an edge.

Nodes represent keyframes, and if the keyframe’s descriptors are used to “define” the node an intuitive way to perform loop detection is to flag a detection whenever a frame sufficiently matching the descriptors of a keyframe is detected. This type of detection means identifying that an edge should exist between the current node and some other, previously unconnected, node in the graph. Conceptually this means continuously searching the whole graph for nodes matching the criteria for an edge, something not feasible for large maps or in real-time. Luckily numerous ways of speeding up this search exists in variants of approximate nearest neighbor (ANN) algorithms. Popular examples include inverted indices, various hashing algorithms such as locality sensitive hashing (LSH), k-d trees among others.

Bag of (visual) words (BoW) methods are useful in order to reduce the dimensionality of a set of feature vectors (i.e. the descriptors representing a frame) to allow for faster retrieval. This is typically done by clustering descriptor vectors extracted from training data to form a vocabulary of words. Descriptors extracted from new images can be partitioned based on the vocabulary to calculate a compact vector with one element for each word, each element indicating to what degree that word is present. Perceptual aliasing issues can become more pronounced with BoW methods, i.e. inability to distinguish similar but different scenes.
Examples of algorithms for localization based on a graph representation are FAB-MAP \[20\], CAT-SLAM \[57\], the algorithm later employed in ORB-SLAM \[55\] based on DBoW\[6\] among others – see \[3\] for a comprehensive survey.

SLAM using only a graph representation like described above is sometimes used (“appearance only SLAM”), but exact positioning is not trivial from this kind of map. It is thus often used for loop detection as a first, coarse, step in a hybrid system before a more exact algorithm refines the pose \[45\]. This step will be referred to as “keyframe estimation”.

**FAB-MAP**

The focus here is on FAB-MAP \[20\], an algorithm for appearance-only SLAM originally proposed by Cummins and Newman. As such the goal here is to fuse FAB-MAP with “dense” stereo SLAM system to form a hybrid system.

The FAB-MAP algorithm is formulated as a Recursive Bayes estimation problem (similar to e.g. the Kalman filter). It thus estimates the probability of a location, \(L_i\), being observed given the set of all observations, \(Z^k\), as

\[
p(L_i|Z^k) = \frac{p(Z_k|L_i, Z^{k-1})p(L_i|Z^{k-1})}{p(Z_k|Z^{k-1})} \tag{2.5}
\]

where \(p(Z_k|L_i, Z^{k-1})\) is the observation likelihood, i.e. the probability of observing the words \(Z_k\) given the location \(L_i\) and all previous sets of observations \(Z^{k-1}\). \(p(L_i|Z^{k-1})\) is the location prior. \(p(Z_k|Z^{k-1})\) is a normalization constant that also takes into consideration whether the seen place is a new or an already seen place.

Determining whether a place is new or previously seen is in FAB-MAP done in one of two ways, either a mean field approximation or by sampling location models. The latter is seen place is a new or an already seen place.

The observation likelihood, \(p(Z_k|L_i, Z^{k-1})\), makes use of Chow-Liu trees. These are Bayesian network approximations of discrete distributions of \(n\) variables so that each variable in the approximation may only be dependent on one other variable. An algorithm devised by Chow and Liu \[16\] is used to obtain the best such approximation (in the sense of Kullback-Leibler divergence). This simplifies training of the model significantly as it means that counting frequency of variable conditionally together is limited to two variables. The Chow-Liu algorithm has to do this counting to calculate the joint probability function, \(p(x, y)\), for two random variables \(X\) and \(Y\), which in turn is used to calculate the mutual information. Chow-Liu trees are generally faster to learn than Bayesian networks that features higher conditionals as learning these are NP-hard \[14\] and requires heuristics for large enough networks.

The Chow-Liu tree is trained on words extracted from training images to represent the probabilities of word conditionals. Clustering of descriptors to form words thus means that the FAB-MAP is a bag of words based algorithm.

The Chow-Liu tree is used with a detector model, see Fig. 2.3, to reduce the observation likelihood, \(p(Z_k|L_i, Z^{k-1})\), with application of approximations to

\[
p(z_{\text{node}}|z_{\text{parent}}, L_i) = \sum_{s_{e, \text{node}} \in (0, 1)} p(z_{\text{node}}|e_{\text{node}} = s_{e, \text{node}}, z_{\text{parent}}) p(e_{\text{node}} = s_{e, \text{node}}|L_i) \tag{2.7}
\]

\[\text{See } \text{https://github.com/dorian3d/DBoW}^1\]
2.1. Theoretical background

Figure 2.3: Detector model used by FAB-MAP. Location $L_i$ generates latent words $e_1,2,...,n$. Latent words are are not observed directly and instead generate the visual word detections, $z_1,2,...,n$ (modeling e.g. false positives).

for a node, $z_{node}$, in the Chow-Liu tree with parent $z_{parent}$, evaluated for location $L_i$. $p(z_{node}|z_{node}; z_{parent})$ in turn can be expressed purely in values from training, meaning only $p(e_{node}|L_i)$ needs to be computed online.

The last component of the Recursive Bayes formulation is the location prior, $p(L_i|Z^{k-1})$. Being a prior, there’s some choice here on how to set it up depending on what prior information is available. A motion model could be used, or a uniform distribution over all places if no prior information exists.

For a more detailed explanation of the FAB-MAP algorithm, refer to the original paper [20]. Multiple variations of FAB-MAP have been proposed, most notably FAB-MAP 2.0 [19]. The largest change to the original algorithm in FAB-MAP 2.0 is an approximation of the observation likelihood in order to be able to utilize inverted indices, reducing execution time and allowing for better scalability.

Pose estimation & odometry

The pose estimate retrieved from graph search is only based on appearance and will at most give an estimate in the form of the nearest keyframe’s pose. This is not a sufficiently accurate estimate and some algorithm for recovering a more accurate pose is thus needed.

Various methods for dealing with this exists, with one of the most popular for robust pose estimation using 2D-3D correspondences being Random SAmple Consensus (RANSAC) [25], or variations thereof. RANSAC is a generic algorithm for finding inliers to a model. A typical way of using it for pose estimation is with a P3P algorithm as the minimal sample set kernel, followed by an algorithm estimating the pose using all the inliers found [58].

RANSAC in this context draws random 2D-3D correspondence samples from all correspondences and uses them to estimate poses using the kernel. Using a threshold on reprojection error to define inliers the algorithm returns the collection of points with highest support.

Pose estimation can also be done with 3D point correspondences. This can be done by solving the absolute orientation problem with for example Horn’s method [42] or Umeyama’s algorithm [87]. Horn’s method (with orthonormal matrices) estimates the translation as the difference between the centroids of two sets of 3D points. Estimation of the rotation makes use of eigenvalue-eigenvector decomposition of a $3 \times 3$ matrix with the rotation matrix formed from the decomposed matrix, see [42] for more information. Umeyama’s algorithm is based on similar principles but uses singular-value decomposition (SVD) instead and disallows reflection. Lorusso, Eggert, and Fisher note that SVD based methods perform slightly better in [53].
Sometimes estimates from one of the methods mentioned are not good enough. Local (or windowed) bundle adjustment can be appropriate to use in order to incorporate information from previous frames and update them with the information of a new one. Bundle adjustment is a non-linear problem where the goal is to minimize the reprojection error of 3D points reprojected to multiple cameras with respect to camera poses and the positions of the points. The cost function for a particular pose and a particular point is
\[
E_{\text{partial}}(R_j, t_j, p_i) = \rho \left( \frac{\partial f}{\partial R_j} \Delta R_j + \frac{\partial f}{\partial t_j} \Delta t_j + \frac{\partial f}{\partial p_i} \Delta p_i - r_i \right), \tag{2.8}
\]

with the full cost function formed as
\[
E(R_1, R_2, \ldots, R_{\text{N\_cameras}}, t_1, t_2, \ldots, t_{\text{N\_cameras}}, p_1, p_2, \ldots, p_{\text{N\_points}}) = \\
\sum_{j=1}^{\text{N\_cameras}} \sum_{i=1}^{\text{N\_points}} E_{\text{partial}}(R_j, t_j, p_i) \tag{2.9}
\]
where \( R_j \) and \( t_j \) are rotation and translation of camera \( j \) respectively, out of a total of \( \text{N\_cameras} \) cameras. \( p_i \) is the 3D point with index \( i \), out of \( \text{N\_points} \) points in total. \( r_i \) is the residual between the point’s measured position and the position predicted by the model used, \( f \). \( f \) projects the point to the camera, and depends on the projection model used. Here \( f \) is based on Eq. 2.4 with a \((R, t)\) transformation applied to the point to move it by the camera pose. \( \rho \) is an optional loss function to better handle outliers. A non-linear solver is used to minimize the cost function \( E \), using the Levenberg–Marquardt algorithm or similar.

SLAM and odometric systems using bundle adjustment include ORB-SLAM2 \cite{64}, PTAM \cite{49}, and optionally SVO \cite{25}.

The methods initially investigated here are mostly from OpenCV, with the focus on robust estimators, thus mainly using RANSAC. A short description of each algorithm is given here, with the interested reader referred to the original papers.

**Non-linear optimization of reprojected points.** A similar optimization problem as for bundle adjustment, Eq. 2.8 and 2.9, can be formulated with only one camera, \( \text{N\_cameras} = 1 \), and fixed points, \( p_i \). Non-linear minimization with e.g. a Levenberg-Marquardt solver can be used to obtain the pose estimate.

**Efficient Perspective n-Point (EPNP)** \cite{50}. EPNP is a closed-form, linear in complexity, solution to the PnP problem that reformulates the problem so that the \( n \) points are reduced to four “control points”. The \( n \) points are reconstructable from the control points using eigendecomposition of a matrix formed by the control points. These are formed using a weighted sum of the eigen vectors, and are thus representable of all \( n \) points. The pose is estimated using the four control points and a Gauss-Newton solver.

**Perspective 3-Point** uses three points (the minimum required) to algebraically find a pose given the three points. Different variations exists such as Lambda Twist \cite{69}, AP3P \cite{45}, and the P3P algorithm by Gao et al. \cite{29}. The latter two are available in OpenCV. As P3P can yield multiple solution a fourth point is sometimes used to select which of the solutions to use.

**Iterative Closest Point (ICP)** \cite{11}. With RGB-D or a stereo pair of cameras it is possible to perform ICP to further refine a pose, a method more typically employed for point cloud registration. As ICP only finds the local optimum, and one of the aforementioned methods should be ran prior to ICP to ensure convergence to the right pose. Systems that use ICP in this way includes \cite{82} and \cite{35}.
2.1. Theoretical background

Umeyama’s algorithm [87] is another algorithm that can be used when 3D points can be calculated from just one frame, e.g. with a stereo pair of cameras. It can be used both as a RANSAC kernel and to estimate the pose using all inliers. A minimum number of 3 points required for Umeyama’s algorithm when the points are three-dimensional [87].

OpenCV uses predefined combinations of RANSAC kernels and algorithms that calculates a final solution using all the inliers found for its RANSAC method. The available are combinations are:

- **Iterative** using EPNP as the minimal sample set kernel and non-linear optimization of the inliers with a Levenberg-Marquardt solver,
- **EPNP** which uses EPNP for both the minimal sample set kernel and the final solution using all inliers,
- **AP3P** using AP3P for the minimal sample set kernel and final solution using all inliers estimated with EPNP, and
- **P3P** which uses the P3P method by Gao et al. as the minimal sample set kernel and again EPNP for the final solution using all inliers.

For non-RANSAC based pose estimation, OpenCV uses the named algorithm directly.

Descriptors

This thesis makes use of local image features, combinations of a key point detector and a descriptor extractor. As previously hinted at, a key point detector aims to find regions or points in an image that can be repeatably detected in other images of the same scene subject to e.g. change in viewpoint. A descriptor extractor calculates a descriptor vector (or descriptor for short) that characterizes the region around a key point. The principle behind these descriptors are that similar regions of an image (e.g. the same point from a slightly different perspective) are close in descriptor distance. This distance could for example be the Euclidean distance, or in the case of binary descriptors, the Hamming distance. A special note for Hamming distance is that it can efficiently be computed on modern CPU:s using exclusive-or and bit count [13].

Descriptors selected as candidates are available in OpenCV 3.4.1 and its contrib modules. A short description of them is given of here, with interested readers referred to the original papers for more details. Many of the descriptors presented below also have a corresponding key point detector. These are omitted from the brief descriptions here, instead refer to the original articles for information about these.

Scale-invariant features (SIFT) [55, 54], introduced by Lowe in 1999, is one of the most well known local image features to date. Its descriptor is extracted for a key point by sampling precomputed gradients around the point. The magnitudes of these gradients are then weighted by a Gaussian function to attribute lesser importance to gradients further from the key point. Groups of $4 \times 4$ gradient samples are then used to calculate orientation histograms with eight bins which forms the descriptor vector. Lowe recommend $4 \times 4$ of these groups, giving a descriptor size of 128 elements. Invariance to rotation is achieved by orienting the gradients relative the key point’s orientation and invariance to uniform illumination is achieved by normalizing the magnitudes.

Speeded-up robust features (SURF) [10, 9] aims to provide faster to compute features than SIFT, while based on similar concepts. Its descriptor is computed from vertical and horizontal Haar wavelet responses from regions in a $4 \times 4$ grid around the keypoint, oriented by the keypoint. The responses for each region are computed from $5 \times 5$ regularly spaced samples within each. The sums for the vertical and horizontal axis responses as well as the
Theoretical background

The sum of their absolute values are collected to form part of the descriptor vector. The complete descriptor vector is formed by concatenating the result for all of the $4 \times 4$ regions, bringing the total size of the descriptor to 64 elements. As is similarly done in SIFT, the responses are weighted by a Gaussian centered at the interest point in order to increase robustness to geometric deformation. Invariance to contrast is also much like in SIFT done by normalizing the descriptor vector to be a unit vector.

**Binary robust independent elementary features (BRIEF)** is a binary descriptor, and can take advantage of fast matching using Hamming distance. The descriptor works by comparing intensities between predetermined pairs of points around the key point, after having applied a smoothing kernel to a patch around the key point.

**Oriented FAST and rotated BRIEF (ORB)** is based on BRIEF and thus shares much in common with it. A major difference is that the sample pattern of the BRIEF descriptor is oriented according to the orientation of the keypoint. The main motivation behind ORB was to achieve rotational invariance.

**Binary Robust Invariant Scalable Keypoints (BRISK)** is a binary descriptor with much in common with BRIEF. Brightness is compared between pairs of points to construct the bit vector that the descriptor consists of. These sampling points are selected evenly in circles around the key point, with a Gaussian blurring kernel applied around each test point to mitigate aliasing.

**Fast Retina Keypoint (FREAK)** is another binary descriptor similar to BRISK and BRIEF and is inspired by the human eye. The descriptor represents one-dimensional differences of Gaussians, computed between pairs of “receptive fields”, areas defined by Gaussian kernels around the key point, larger in size and more separated further from the key point.

**Accelerated KAZE (AKAZE)** is an accelerated version of the KAZE descriptor. Its descriptor works in a similar way to BRIEF. It is thus also binary but instead of comparing single points AKAZE compares averages around points. Like many of the rotationally tolerant descriptors, AKAZE also uses the orientation around the key point to orient the sampling pattern and thus achieve rotational invariance.

**DAISY** was initially proposed to efficiently handle dense stereo matching. It works by computing “orientation maps”, defined from the gradient along a specified direction, these are convolved and then sampled at specific points around the key point to form the descriptor. DAISY is not a binary descriptor.

**Boosting Binary Descriptors (BoostDesc/BinBoost)** is a binary descriptor that is trained using a number of weak binary learners. This descriptor’s problem formulation closely follows that of the well established AdaBoost machine learning algorithm (hence the boost in the name), with the aim of learning a compact representation of a binary descriptor.

Complete systems

Many systems capable of performing visual SLAM have been proposed. One of the earliest successful such system was PTAM, which in 2007 demonstrated the ability to run tracking and mapping in parallel threads. The mapping thread uses bundle adjustment of key frames with the tracking thread handling real-time camera pose estimation.

A number of solutions for camera tracking that allows for relocalization have been also been proposed as individual algorithms. Williams, Klein, and Reid present a monocular...
RGB relocalization module based on a binary “randomized list” classifier, where likelihood of a given class is trained online. The classifier tries to correctly label image patches in a frame to the “landmarks” that was learned online, each with a known 3D position. It does this by comparing pixel intensities of points within the patch for every “element” of the list of binary classifiers, with the result of all comparisons determining what class to label the patch as. The classification gives a match to a landmark, and these are used with RANSAC and P3P (and further optimized with all inliers) to find the pose. PTAM makes use of an older version of this relocalization system, presented in [92].

Lynen et al. [56] describe a system for global localization relative a previously mapped and heavily compressed world. Binary descriptors are used in conjunction with bag-of-words retrieval based on inverted multi-indices [7]. A covisibility graph is used to narrow down the matching keyframes which are then selected and refined using preemptive RANSAC. After this, an additional refinement step is done using a nonlinear least square PnP solver. Tracking is subsequently handled by an extended Kalman filter.

More recently Mur-Artal and Tardós [63, 64] introduced “ORB-SLAM”, a complete SLAM system that handles global relocalization and loop detection. Both of these are handled using a pre-trained bags of words vocabulary, using inverted indices to speed up retrieval. Like in the system proposed by Lynen et al., a covisibility graph is maintained and is here used to group keyframes to more efficiently handle local tracking using bundle adjustment. A second version of the algorithm, ORB-SLAM2, was released as open-source.

2.2 Method

Various different methods for evaluating specific parts and complete systems have been proposed in literature, and this section presents such methodologies.

Descriptor & pose estimation evaluation

Mikolajczyk and Schmid [59] proposed a methodology for descriptor evaluation in 2005 and evaluated a number of key point detectors and descriptors. Many later studies have adopted the same or a heavily inspired methodology such as in [13, 80, 10]. Mikolajczyk and Schmid compare the matching power of descriptors extracted from one image with another, a “reference image” and a “test image”.

Concretely this is performed by calculating precision and recall of the matches given a discriminatory threshold, \( \tau \). This threshold is varied from the minimum value (no matches selected) to the maximum (every match selected) in order to obtain a precision-recall curve (PRC) for each descriptor extractor given the two images. A particular recall hence yields a precision value and vice versa.

Precision and recall for a set of descriptors matches, \( M \), extracted from a test image and matched to descriptors from another image are defined as

\[
\text{Precision}(\tau) = \frac{\sum_{m \in M} \begin{cases} 1 & \text{if selected}(m; \tau) \text{ and relevant}(m) \\ 0 & \text{otherwise} \end{cases}}{\sum_{m \in M} \begin{cases} 1 & \text{if selected}(m) \\ 0 & \text{otherwise} \end{cases}},
\]

\[
\text{Recall}(\tau) = \frac{\sum_{m \in M} \begin{cases} 1 & \text{if selected}(m; \tau) \text{ and relevant}(m) \\ 0 & \text{otherwise} \end{cases}}{\sum_{m \in M} \begin{cases} 1 & \text{if relevant}(m) \\ 0 & \text{otherwise} \end{cases}}.
\]
where selected\((m, \tau)\) determines if a descriptor match is selected or rejected based on a threshold \(\tau\) and relevant\((m)\) denotes whether a descriptor match should have been selected or not.

Mikolajczyk and Schmid test three different ways of performing matching. One of these is the *Nearest Neighbor Distance Ratio* test \([54]\) (NNDR), testing whether

\[
D_1 < \tau D_2,
\]

holds or not, with \(D_1\) denoting the descriptor distance between the descriptors best matching and \(D_2\) the descriptor distance for second best match. The fraction \(D_2\) has to be of \(D_1\) for an accepted match, \(\tau\), is the threshold varied in order to obtain the PRC curves.

Whether a point is relevant or not is by Mikolajczyk and Schmid determined by checking if the area overlap between the matched descriptors’ key points (reprojected if images are under transformation) is above a threshold.

Pose estimation is commonly evaluated by comparing rotational and translational error of estimates against ground truth \([50,45]\).

**Odometry evaluation**

Odometry is often evaluated by comparing estimated trajectories against a ground truth trajectory \([32,76,39]\). Sturm et al. present two different metrics for evaluating the quality of the trajectory in \([76]\), *absolute trajectory error* (ATE) and *relative pose error* (RPE). To calculate ATE both the estimated and ground truth trajectories need to be aligned, this is done with Horn’s method \([42]\). Both trajectories are first translated to be centered at zero, i.e. by subtracting the means. ATE for one estimated pose, \(i\), from the trajectory is then defined as

\[
E_{ATE,i} = \|S_{est,i} - t_{truth,i}\|_2, \tag{2.13}
\]

that is, as the error between an estimated position vector, \(t_{est,i}\), and ground truth position vector for the same time instance, \(t_{truth,i}\). The estimated trajectory is aligned in the least-square sense with the ground truth trajectory by the rotation matrix \(S.\) \(S\) is obtained using Horn’s method for finding an orthonormal matrix aligning two coordinate systems \([42]\). Rotation for estimated poses are not directly a part of the error metric, but Sturm et al. note that errors in orientation will yield errors in position.

RPE is calculated as the difference in movement between estimated and ground truth trajectory when comparing movement over \(\Delta\) frames, and thus measures “drift per \(\Delta\)”.

For both ATE and RPE Sturm et al. propose to calculate the root mean squared error. Handa et al. too use ATE in \([39]\).

The quality of loop detection is typically assessed after loop closing has been performed by computing an error metric such as ATE \([50]\).

**Datasets**

No evaluation can be carried out without data. While recording a sequence to use for testing is simple, specifically for pose estimation, defining the error is not trivial. To this end, released and available datasets are used instead, featuring ground truth poses so that an error is easily defined without bias.

Using datasets allows for direct comparison with other algorithms, something that is not possible in a quantitative way otherwise.

For visual odometry and SLAM the “KITTI” dataset \([32]\) is among the most popular. It features eleven stereo sequences (and laser) with ground truth recorded by a car driving through real world streets.

Being recorded from a driving car however means that the sequences do not exhibit truly unconstrained motion. Various other datasets exist that feature this. The “RGB-D SLAM” dataset \([44]\) features sequences recorded from a handheld Microsoft Kinect with ground truth...
poses recorded by an external motion-capture system. The lengths traversed during these sequences range from roughly 1.5 to 40 meters and are thus relatively short.

Another dataset is the “EuRoC” dataset \[12\] where a drone, an AscTec Firefly MAV, flies through indoor environments. The EuRoC dataset was recorded for the European Robotics Challenge to test the capabilities of SLAM systems participating in the contest. The dataset comes in batches, one batch of sequences recorded in a machine hall (MH0x) and second batch collected in a room in two configurations (Vicon room, V10x and V20x). The latter batch was mostly designed to evaluate scene reconstruction and provides 3D point clouds of the two configurations of the room.

The first batch, MH0x, features large spatial extents of various difficulties. The position of the drone was for this batch captured using a millimeter accurate laser tracker, and the orientation estimated from an IMU. Both intrinsic and extrinsic calibration parameters are provided.

Datasets explicitly for testing key point detectors and descriptors are also available, with one of the most popular being the “affine covariant regions dataset” \[59\]. Others exists as well, with some specifically for testing specific aspects. One such example is the “Phos” dataset \[88\] that is specifically designed for testing robustness to illumination.

### 2.3 Summary and rationale behind a priori choices

3D points observed by a camera can be used to estimate its pose. These points can be detected and described using local image features and subsequently be detected again in later camera frames. The descriptors can also be used to describe a frame, which with an algorithm like FAB-MAP means previously visited places can be identified.

These descriptors need to be invariant to e.g. perspective changes in order to be able to reliably detect them and different pose estimation algorithms work differently well. These, together with the complete system, will be evaluated and the next chapter, Ch. 3, presents how this evaluation is carried out.

Some choices were made before any evaluation was conducted. Most notably this includes the use of FAB-MAP for finding matching keyframes. The rationale behind FAB-MAP stems from the theoretical benefits the algorithm has (e.g. alleviating perceptual aliasing) and the fact that it has successfully been used in other SLAM systems such as CAT-SLAM \[57\], CD SLAM \[72\], and Rittums \[68\].

The methodology presented in the next chapter makes use of previous methods to a large extent as to be able to compare results. For selection of what datasets to use for evaluations, similarity to the eventual application was given large weight (e.g. unconstrained motion). Novelty was also considered when conducting evaluations already performed in previous studies, i.e. using a different dataset to contradict or support findings from these studies.
3 Method

The thesis is split into a number of different phases, the evaluation phase, the implementation phase, and the final evaluation phase. This chapter and the previous, the Related Research chapter, present results obtained pertaining to how these phases should be performed.

The system is, where applicable, implemented in a modular way to facilitate exchanging algorithms for other ones to allow many different configurations to be used.

The phases are each presented in more depth in coming sections, with a brief overview of them given here;

The evaluation phase, where evaluations of proposed algorithms are conducted to find the most likely candidates to select for use as “modules” (descriptors, keypoint detectors, search methods etc). Extensive use of previous studies are taken to narrow down the list of potential methods if they are not explicitly evaluated.

The implementation phase, where the modules are integrated as a system, with capabilities to switch between them.

The final evaluation phase, where the modules found performing best individually are evaluated as a complete system.

3.1 Evaluation phase

Evaluation is performed on individual modules in order to gauge if a satisfactory performance can be achieved when they are combined. The goal is not to perform an exhaustive evaluation, instead it is meant to investigate if aspects of a particular module can achieve adequate performance. Refer to evaluations such as [40, 18, and 80] for complementary evaluations of feature descriptors and detectors.

The modules investigated here are feature descriptor and pose estimation. What is being evaluated is different for each module, presented shortly. For all evaluations in the initial evaluation phase the dataset used is the “RGB-D SLAM dataset” [76]. The subset of the dataset used was gathered using a handheld Microsoft Kinect sensor in various indoor environments. Being handheld, the non-debug sequences in the dataset exhibit 6-DoF unconstrained motion.
3.1. Evaluation phase

The dataset was chosen due to the unconstrained motion in addition to being an RGB-D dataset, meaning depth is available without stereo rectification (see Delimitations).

**Key points & descriptors**

For the local image features, the desire is sufficiently many, good, key points. This means key points are repeatably detectable between images subject to perspective transformation and phenomena such as illumination change. The descriptors need to be sufficiently invariant under the same circumstances so that matching between frames can be done.

Descriptors are used for both keyframe and pose estimation. When using FAB-MAP for the former, descriptors are classified into words from a vocabulary so that a frame can be represented by a collection (bag) of such words. This means the use differs slightly between both estimations, what is best for one is not necessarily best for the other.

As a heuristic, the initial evaluations omit any effects from word extraction and focuses on the feature descriptors descriptiveness. Chiefly the evaluation is performed following the methodology outlined by Mikolajczyk and Schmid in [59], with precision and recall of descriptor matching between one frame and another. Precision and recall are calculated from one frame ("reference frame") to another ("test frame") by selecting key points in the reference frame and extracting the descriptor corresponding with the keypoint using the feature extractor currently evaluated. This is then repeated for the test frame. The last step before matching is reprojecting points extracted from the reference frame onto the test frame using the ground truth pose given by the dataset, this is later used to determine whether matching was done correctly or not. Unlike Mikolajczyk and Schmid, a fixed maximum pixel distance, ϵ, is used when determining if a match is correct or not instead of area overlap.

Descriptor types can be compared with one another for any given precision or recall value, with a general performance for each descriptor type given by the area under curve (AUC).

A few notes is in order for this evaluation. Ideally key points should be chosen randomly on the reference frame and then reprojected onto the test frame as to not let the key point detector influence. The dataset used here is however not synthetic, which means that some errors exists for both the ground truth pose and the depth values in the depth map. Adding these factors together mean that the reprojected point can not be trusted to truly correspond with the point in the reference frame. This is also the reason as to why a small ϵ deviation is allowed.

As it is also desirable that descriptors are invariant to illumination changes, this is specifically evaluated using the Phos [88] dataset. This dataset features a static scene viewed from the same position illuminated in a number of different ways, both uniformly (by varying exposure time) and non-uniformly (using a strong directed light source). The evaluation method is again based on that of Mikolajczyk and Schmid, but as the viewpoint stays static the key points extracted from the reference image ("baseline illumination") can be reused for each of the subsequent test images of different illumination. Correct matches is thus known a priori and no reprojection or ϵ threshold is needed.

The descriptor extractors are additionally evaluated by using them to perform pose estimation under increased distance and rotation from a fixed reference image. This test is performed to support the precision-recall evaluation and in order to gauge how large transformations can be considered reasonable to recover from.

The pose estimation evaluation is done by projecting 2D points extracted from a reference frame to 3D using Eq. 2.3, then using these points and matches from a later (in time) test frame and estimating the pose with a fixed RANSAC based PnP algorithm. Using these points to perform PnP pose estimation yields the transformation (\( R, t \)) from the reference frame to the test frame.

The estimated pose is then compared against the ground truth pose available in the dataset in terms of translation and rotation. An simple outlier rejection method for points with NNDR τ less than 0.9 and point correspondences that are not symmetric between test and reference
frame is employed. A minimum number of inliers, $N_{\text{min}}$, has to be found for the pose to be considered valid. This is done for multiple frames with larger and larger temporal distance to the reference frame. No information from intermediate frames is used.

The pose error evaluation include additional parameters and is not as generally applicable as the PRC evaluation. On the other hand it more closely resembles the use case for pose estimation, being near identical. A similar method is for example used in [40, 75].

**Pose estimation**

Evaluation of pose estimation algorithms is by and large performed as the pose error evaluation for the feature descriptors. Instead of the estimation method being kept fixed it is the descriptor algorithm that is. As again no information from intermediate frames is used, it is not odometry that is evaluated but the capability to find a valid transformation between two images.

Similar to the pose evaluation test for descriptors, a minimum number of inliers are required for a pose to be considered valid, $N_{\text{min}}$. This works for RANSAC based methods, which are the focus, but not for PnP without outlier rejection – one of which is evaluated as reference. To reject poses with this method, a maximum distance, $d_{\text{disqualify}}$ from the first (“keyframe”) and second (“test frame”) is used to disqualify poses estimated further away than reasonable. This distance is set to be larger than the movement occurring in any of the test sequences.

The implementation of RANSAC used throughout is from OpenCV, except for Umeyama’s algorithm which uses a modified version of the RTL library [15]. The OpenCV implementation of RANSAC uses “jump out”, and this feature was added to the RTL version of RANSAC to keep both implementations comparable. Jump out terminates the execution of RANSAC based on the support for the best solution found. When better solutions are found, the number of iterations required is recalculated based on the new inlier ratio together with a confidence threshold and the kernel sample size [84].

The implementation of Umeyama’s algorithm used for evaluation is the one available in Eigen.

### 3.2 Implementation & the system

The implementation was initiated alongside the evaluation phase, extending until the final evaluation phase. The major guideline followed was to build a system that facilitates exchanging parts, modules, for others. OpenCV is used where applicable, with the decision to use OpenCV being based on the maturity and popularity of the library [2].

The implementation process was guided by results obtained from the initial evaluations, and during implementation.

An outline of the system can be seen in Fig. 3.1, where each block roughly corresponds to a base class or a group of them tasked with performing the various named assignments.

The first part is a conversion step from the existing SLAM system, and can here be mostly ignored. Tasks done in this step includes reprojecting the images to a normalized camera as the SLAM system uses calibration parameters expressed in a slightly different way than OpenCV conventions. Another task is creating OpenCV matrices out of the images read from the existing system, namely rectified frame images, disparity image and a validity binary mask for the disparity image. The disparity image is then converted to instead represent the distance metrically.

The succeeding block is the first proper step in the system presented here. This block is responsible for detecting key points and extracting descriptors. As bag of words methods

---

[1] https://github.com/sunglok/rtl

likely benefit from greater number of descriptors (up to a certain degree, of course) the key point detectors are configured to return a large number of them. This is on the other hand not preferable for descriptor matching, where fewer but good key points are desired. To solve this, the original, plentiful, key points are subject to a filtering step using Adaptive Non-Maximal Suppression (ANMS), implemented as a slight variation of the Suppression via Disk Covering (SDC) algorithm \[31\]. ANMS algorithms finds and keeps the $N$ best key points around the neighborhoods of the selected key points, thus finding a balance between the spatial distribution of key points and their quality \[4\]. The resulting key points are used for odometry, with the original key points used for word extraction. The same descriptor type is not necessarily used for both word extraction and odometry.

Once descriptors are extracted, words are extracted using pre-trained cluster centers and compared to existing keyframes using FAB-MAP (discussed in more detail in theory Sec. \[2.1\]). In the case of FAB-MAP, this gives the probability of a new place or a previously inserted key frame, given the pre-trained clusters, training data, and the Chow-Liu trees.

When the system is not lost and the probability of the most likely key frame is greater than a set threshold, the key frame is compared using a frame counter against the last $N$ frames. If the selected best matching is not within the $N$ frames, the system flags the new frame as a potential loop, pending verification.

What navigation frame is selected to be used for the next step, descriptor matching and pose estimation, depends on whether the system is lost or not. If not lost, the last frame is selected unless it with a very simple descriptor matching scheme too poorly matches the current frame. If lost on the other hand, the output from the keyframe estimation algorithm is used if this has enough probability of being a match according to FAB-MAP. This descriptor matching scheme uses NNDR (as for evaluation) with a fixed threshold. Whether the system is lost or not is determined by successively failing pose estimation or obtaining results based on too few key points (i.e. inliers). A few frames where pose can not be determined in succession is thus allowed (due to e.g. temporary excessive but short-lasting motion blur).

The navigation frame selected is then used in the next block, pose estimation and descriptor matching. How these are handled depends on what algorithm is implemented, but they all

---

\[3\] Conjectured, and slightly supported by \[67\], where 420000 features performs better than 50000, although the application differ and even 50000 is well above what is used here. A simple experiment conducted comparing when using stricter thresholds for keyframe extraction (thus behaving ANMS-like) supported this, although Bailo et al. conjecture the opposite \[8\].

\[4\] As ranked by some score metric, e.g. Harris score.
take the selected frame as input, with their key points, descriptors, associated 3D points, and optionally the same for all frames deemed covisible from the reference frame.

After the pose estimation, world matching and bundle adjustment takes place. World matching exists in order to add associations between multiple frames observing a single point. The inlier 2D-3D correspondences are used to do this association, effectively associating detected (image) points to points detected by any of the frames in the navigation frame’s covisibility list. Descriptor matching guided by reprojection can also be used to perform the association.

Points associated and therefore originally seen in another frame are updated with a reference to the new frame. Points without any association are tentatively added until either the frame is discarded as too old or an association with the same point is done with a later frame.

Points updated are used to define covisibility. If a frame shares at least \( K \) points with another frame, the two frames are connected with an edge in the implicit covisibility graph. Concretely, this is done by adding a reference to the other frame to both of the frames’ covisibility list.

Once a pose has been determined and matches to points in other frames are known, an additional optimization step running bundle adjustment is (optionally) used. This further refines the current pose and the pose of any frame deemed covisible.

The number of frames since last keyframe insertion and the relative transformation between last insertion is taken into consideration when determining if a new keyframe should be created from the current frame. The probability of a new frame as determined by FAB-MAP is also used.

The next frame is then ready to be processed, repeating all of the steps above. Some additional details regarding each module is given in the following subsections.

**Key points, descriptors & ANMS**

Key points and descriptors are, as mentioned, from OpenCV. The choice of what descriptors to use was simply a case of availability. An additional descriptor was however initially implemented, reusing the OpenCV’s `Feature2d` interface – but it was ultimately not used.

The implementation of ANMS key point filtering in large follows the SDC algorithm as proposed by Gauglitz et al. [31]. The algorithm was implemented mostly as presented, with the addition of allowing termination of the algorithm when the number of found key points is within an interval, and not just when \( N \) points have been found. The response of the key point and its location both affects whether the key point is filtered or not. See Fig. 3.2 for an example of ANMS filtered key points.

SDC is a heuristic method used to speed up the computation time. There however exists alternatives to the SDC algorithm, for example Suppression via Square Covering (SSC) and ANMS based on Tree Data Structure, both presented in [8]. The former being very similar to SDC and the latter making use of a tree structure to allow for fast retrieval of points within a range. As SDC and SSC computation time are comparable for lower number of input points, which is used here, SDC was chosen due to approximating distance more closely to the real distance – even if SSC is slightly faster. The ANMS based on the Tree Data Structure method get its speed up with reuse of key points, something not applicable here.

**Keyframe estimation**

Keyframe estimation uses FAB-MAP, with the implementation being OpenFABMAP [36]. A modified version of OpenFABMAP was used that works with OpenCV 3, with the extra addition of being able to use the Hamming distance for binary descriptors when clustering. FAB-MAP2.0 is the version used throughout. The FAB-MAP location prior was set as uniform and sampling was used to determine the likelihood of a new place.
3.2. Implementation & the system

Figure 3.2: To the left, ANMS filtered AGAST key points. To the right, the original key points. Do however note that some of the furthest away detected key points have been removed by a separate process due to being too far away.

Figure 3.3: Flowchart describing the selection of navigation frame. Dashed outlines of boxes indicate things occurring elsewhere (i.e. later) in the system that impact whether the its state is lost or not. The circle indicate the start point with each new frame.

For a keyframe to be accepted, the probability output from FAB-MAP has to exceed a predefined threshold value. How this information is used depends on whether the system is lost or not, see Fig. 3.3 for a flowchart of the logic.

If the system is lost the keyframe is selected for pose estimation. If not, the system checks whether the keyframe output from FAB-MAP is among the $N_{\text{loop, recent}}$ most recent frames. If it is the system ignores the result, otherwise it is flagged as an potential loop, pending verification.

One of the $N_{\text{navigate}}$ last frames is then selected as reference frame for pose estimation. This frame is in the vast majority of cases the most recent frame, but can be an older if the last frame is deemed too poor. This is determined as the case if too few of the tracks (“good points” seen in more than one previous frame) are found on it using a simple descriptor matching. The frame selected is referred to as the “navigation frame”.

20
New keyframe decision

The FAB-MAP algorithm can estimate a frame as a new “place”, and this is one condition for deciding whether to add the current frame as a new keyframe or not. See Fig. 3.4 for a flowchart of the logic. A keyframe is not necessarily inserted if FAB-MAP deems a frame as new. A minimum number of frames must also have elapsed since last insertion and the translational movement or rotation relative to the last inserted keyframe must have exceeded a distance or angle respectively. If these criteria are met, a keyframe is inserted after bundle adjustment.

There is an additional, separate, condition that may force insertion of keyframes if translational movement or the rotational angle since last insertion exceeds an upper limit. This is to ensure keyframes are not too far between.

Pose estimation

Another pose estimation method was implemented in addition to those previously evaluated. This method is inspired by the one used by Persson et al. in [70], and uses a two step process to estimate the pose and good inliers.

The method makes use of 3D point reprojection to guide descriptor matching. The first step is searching for correspondences between descriptors of the frame and the tracks of the navigation frame. This process is guided by where the tracks were seen in the navigation frame, searching in a radius around this point for descriptor matches. If the descriptor distance of a match is low enough and no similar enough matches exist within the same radius (using NNDR) the correspondence is accepted. Additionally, the distance to the track (i.e. depth) and the depth at the descriptor position may not differ too much or the correspondence is deemed void.

These track correspondences are then used to estimate an initial pose, using RANSAC with a P3P kernel followed by EPNP to provide a pose for the second step.

The second step repeats the same procedure as the first step, but now using all points found on the navigation frame, as well as points of frames covisible from the navigation frame. The pose estimated from the first step is used to reproject the points to the current frame.
where a more narrow search radius is used to redo descriptor correspondences. Inliers found by RANSAC using P3P are saved so that they can be used for bundle adjustment in the next step and provide a final estimate of the current frame’s pose.

Pose estimation works slightly differently if the system is lost (or when checking for loop). Instead of using reprojection in the first step, descriptor matching with NNDR is used and a constraint on symmetric matches. The constraint on points not moving too far is thus omitted, which is an assumption for normal frame-to-frame navigation.

**World matching & bundle adjustment**

The points determined as inliers by the pose estimation are then used to determine which of the recent frames that will be considered as sharing visible features, i.e. are covisible. This is determined by tallying up how many of the inliers were seen in other frames, with a fixed threshold required for a given frame to be deemed covisible.

Each point keeps a reference to the frames where it was an inlier, i.e. which frames it was seen in. This structure makes the tallying trivial. A maximum number of frames that may be added as covisible is used, if this number is exceeded then the frames with most matches are chosen.

The inliers and the covisible frames are then used to run local bundle adjustment using Ceres Solver. Two versions are available, one where points may be moved and one where they may not, but due to execution time the latter is used. A Cauchy loss function is used for robustification (i.e. $\rho(x) = \log(1 + x)$ in Eq. 2.8).

**Notes on original system**

The original system estimates pose using Umeyama’s method, implemented in such a way that outliers can be dealt with. Previous applications of the system did not require the capability to handle truly unconstrained motion nor relocate. Presented as comparison in the evaluation section is a version of the original tracker that only performs frame-to-frame motion estimation. This is opposed to the slightly more accurate “map navigation”, which utilizes the keyframes to keep the trajectory more consistent. This version was not used due to it not functioning properly with the motion in the evaluation sequences. Frame-to-frame navigation is also what is used in the proposed system, but with consistency handled by bundle adjustment.

### 3.3 Final evaluation phase

The final, complete, system is evaluated on the dataset presented in (“EuRoC”). As mentioned in Sec. 2.2, the dataset consists of two batches. It is the first batch that is used for this phase. Two different evaluations are done using the sequences of the first batch, one testing the quality of the odometry and one testing the relocalization capabilities.

**Odometry**

The first evaluation metric is a pose error for the full estimated trajectory, where the pose for every frame is compared against the ground truth pose. A method to align the ground truth and estimated trajectories was used, as the result should be independent of the coordinate system (and units) used to express the trajectories. This method is the same as proposed by Sturm et al. in [76]. Absolute Trajectory Error (ATE) is used, again following the methodology of Sturm et al.
3.3. Final evaluation phase

Relocalization

The secondary evaluation metric focuses on the quality of loop detections made and the capability to relocate. As defining frames where a loop detection “should” be made is not trivially decided upon without subjective choices, it is instead typically evaluated as the pose error after loops have been closed, see e.g. [90]. This is outside the scope of this thesis, see Delimitations (Sec. 1.4), and thus the evaluation is instead done by testing the relocalization capabilities.

Relocalization is evaluated by giving the system a sequence of frames from which keyframes are decided. This sequence of frames, the training sequence, is a subsequence from a long benchmark sequence featuring large scale loops. After the sequence has been processed mapping is turned off, and individual frames are given that are from a later part of the whole sequence. This is the test subsequence, and evaluation is done by letting the system find keyframes added during mapping, and estimate the pose if a confident enough match is found.

If such a keyframe is found, the estimated relative translation between the test frame and the found keyframe is calculated, $t_{rel,est}$, and marked as selected. Decision of whether the estimate is deemed sufficiently correct or not is application dependent, with the desire of estimates as close as possible to the true position.

Deciding whether it is sufficiently close is done with a threshold value, $d_{max}$, which governs whether an estimated pose is true or false. This threshold is used in

$$\|St_{rel,est} - t_{rel,truth}\|_2 \leq d_{max},$$

where $S$ is the aligning rotational matrix for the sequence from Eq. 2.13, $t_{rel,est}$ and $t_{rel,truth}$ are the relative translations between the test frame and the keyframe as estimated and according to ground truth, respectively. As $d_{max}$ is application dependent, results are given for a few different thresholds.

This will however only give a sense of the performance in regards to how well the key frames that are found are estimated. Frames for which no keyframe is found with enough confidence do not impact the result. In reality it is also desirable to detect as many keyframes as possible where poses can successfully be estimated, given that it is possible at all.

Determining that such a keyframe exists is done by utilizing reprojection of points identified during mapping. Using ground truth frame translation, the points from the three nearest keyframes to the test frame are reprojected onto it (ignoring any viewing in an angle differing more than 90° and further than five meters). If $N_{visible}/N_{keypoints} \geq \tau_{inside}$ evaluates as true it is deemed that a pose estimate should be obtainable. Here $N_{visible}$ is the number of reprojected points visible inside the test frame, $N_{keypoints}$ the total number of points reprojected, and $\tau_{inside}$ a minimum fraction. Do note, visible here does not take occlusion into consideration. The fraction deemed as “sufficiently many points” is not straightforward to determine, and the result is hence given when letting $\tau_{inside}$ vary between 0 and 1.

True/false estimated poses result in a “precision”, given thresholding $d_{max}$ distances. “Recall” is given by the fraction of true estimated poses and the number of frames for which enough points (determined by $\tau_{inside}$) lies within the frames. A set $d_{max}$ threshold yields a value of precision, with an associated recall curve given a varying $\tau_{inside}$.
This chapter presents the results obtained, split into two sections for each of the evaluation phases, Evaluation phase (in Sec. 4.1) and Final evaluation (in Sec. 4.2).

4.1 Evaluation phase

Multiple smaller evaluations are here presented. The test sequences used are here briefly described, all using sequences from the “RGB-D SLAM Dataset” \cite{76}:

**XYZ1** A sequence moving the camera forward then backwards.
- **Origin:** fr1/xyz, **Start index:** 0, **Length:** 50

**XYZ2** A sequence moving from side to side.
- **Origin:** fr1/xyz, **Start index:** 500, **Length:** 60

**RPY-SIDE** A sequence yawing (as in yaw, pitch, and roll) the camera from side to side while looking towards the ground.
- **Origin:** fr1/rpy, **Start index:** 0, **Length:** 30

**RPY-ROLL** A longer sequence rolling (as in yaw, pitch, and roll) the camera, stationary in place.
- **Origin:** fr1/rpy, **Start index:** 367, **Length:** 50

**DESK1** A sequence where the camera is both rotated and moved while facing roughly the same direction.
- **Origin:** fr1/desk1, **Start index:** 60, **Length:** 45

**DESK2** A sequence where the camera is both rotated and moved while facing roughly the same direction.
- **Origin:** fr1/desk2, **Start index:** 317, **Length:** 35

**FLOOR** A sequence where the camera is moved forward observing a wooden floor.
- **Origin:** fr1/floor, **Start index:** 46, **Length:** 30

For brevity, only a few are presented in this chapter.
4.1. Evaluation phase

Figure 4.1: Precision plotted against recall for a frame undergoing both translation and rotation (the sequence DESK1), frame two. Translation and rotation relative to reference frame is 0.05 m and 0.07 rad respectively.

Figure 4.2: Precision plotted against recall for a frame undergoing both translation and rotation (the sequence DESK1), with a larger transformation than for Fig. 4.1, frame four. Translation and rotation relative to reference frame is 0.10 m and 0.15 rad respectively.

Descriptor evaluation

Descriptor evaluation is done in two different ways, with precision-recall curves and pose error. Figures 4.1, 4.2, 4.3, and 4.4 show the precision-recall curves with increasing transformation between the reference and test frame, skipping two intermediate frames between each. The pixel radius for a point being considered correct and hence relevant, $\epsilon$, is here set to 3.0 pixels.

The result of the illumination evaluation for one scene can be seen in Fig. 4.5 to 4.8. The illumination settings shown are “+2 Overexposed”, “-4 Underexposed” “Directional + 0.4 Uniform”, and “Directional illumination only” respectively, all of these on the excessive side. The first two are uniform while the last two are non-uniform. Precision-recall curves for the other illumination images for this test scene can be found in appendix A.1.

Mean error for translation and rotation, in meters and radians respectively, and their standard deviation can be seen in the tables 4.1, 4.2, and 4.4 for three of the test sequences. Along with the translational and rotational errors, the mean and standard deviation of the time used for extracting descriptors is given. $f_{ok}$ indicates the fraction of all poses which was deemed as valid with enough inliers as support. This threshold is here $N_{min} = 25$. Do note that the errors are only calculated for frames with valid poses. This means $f_{ok}$ has to be considered the primary metric and the pose errors a secondary metric for when $f_{ok}$ values are close. RANSAC using EPNP as the minimal sample set kernel and the Levenberg–Marquardt based iterative algorithm in OpenCV is used for estimating pose.

All of the descriptors are as implemented in OpenCV 3.4.1, using default parameter values.

Pose estimation evaluation

Similar to the descriptor evaluation, errors for different algorithms are shown in the tables 4.5, 4.6 and 4.7 for three test sequences. The same AKAZE descriptors and key points are used for all algorithms. Maximum movement distance allowed, $d_{disqualify}$, is here set to 5.0 meters.

---

1Evaluated on a laptop with an Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz processor.
4.1. Evaluation phase

Figure 4.3: Precision plotted against recall for a frame undergoing both translation and rotation (the sequence DESK1), with a larger transformation than for Fig. 4.2, frame six. Translation and rotation relative to reference frame is 0.16 m and 0.21 rad respectively.

Figure 4.4: Precision plotted against recall for a frame undergoing both translation and rotation (the sequence DESK1), larger than for Fig. 4.3, frame eight. Translation and rotation relative to reference frame is 0.23 m and 0.21 rad respectively.

Figure 4.5: Precision-recall curves for the test image “+3 Overexposed”.

Figure 4.6: Precision-recall curves for the test image “-4 Underexposed”.
4.1. Evaluation phase

![Graph showing precision-recall curves](image1)

Figure 4.7: Precision-recall curves for the test image “Directional + 0.4 Uniform”.

![Graph showing precision-recall curves](image2)

Figure 4.8: Precision-recall curves for the test image “Directional illumination only”.

Table 4.1: Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence XYZ1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top three values in bold.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>$\mu_t$</th>
<th>$\sigma_t$</th>
<th>$\mu_R$</th>
<th>$\sigma_R$</th>
<th>$\mu_{time}$</th>
<th>$\sigma_{time}$</th>
<th>$f_{ok}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKAZE</td>
<td>0.018</td>
<td>0.007</td>
<td>0.016</td>
<td>0.006</td>
<td>29.967</td>
<td>0.460</td>
<td>1.000</td>
</tr>
<tr>
<td>BRIEF</td>
<td>0.017</td>
<td>0.007</td>
<td>0.015</td>
<td>0.006</td>
<td>2.688</td>
<td>0.083</td>
<td>1.000</td>
</tr>
<tr>
<td>BRISK</td>
<td>0.017</td>
<td>0.007</td>
<td>0.016</td>
<td>0.006</td>
<td>7.238</td>
<td>0.178</td>
<td>1.000</td>
</tr>
<tr>
<td>BoostDesc</td>
<td>0.016</td>
<td>0.006</td>
<td>0.014</td>
<td>0.006</td>
<td>84.559</td>
<td>3.093</td>
<td>0.980</td>
</tr>
<tr>
<td>DAISY</td>
<td>0.016</td>
<td>0.006</td>
<td>0.014</td>
<td>0.006</td>
<td>31.383</td>
<td>3.093</td>
<td>0.980</td>
</tr>
<tr>
<td>FREAK</td>
<td>0.017</td>
<td>0.007</td>
<td>0.015</td>
<td>0.006</td>
<td>4.467</td>
<td>0.232</td>
<td>1.000</td>
</tr>
<tr>
<td>ORB</td>
<td>0.017</td>
<td>0.007</td>
<td>0.015</td>
<td>0.006</td>
<td>4.027</td>
<td>0.220</td>
<td>1.000</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.016</td>
<td>0.007</td>
<td>0.014</td>
<td>0.005</td>
<td>26.978</td>
<td>0.755</td>
<td>1.000</td>
</tr>
<tr>
<td>SURF</td>
<td>0.016</td>
<td>0.007</td>
<td>0.014</td>
<td>0.005</td>
<td>9.180</td>
<td>0.254</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 4.2: Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence RPY-ROLL, additionally the fraction of successful estimations are shown as $f_{ok}$. Top three values in bold, unless tied.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>$\mu_t$</th>
<th>$\sigma_t$</th>
<th>$\mu_R$</th>
<th>$\sigma_R$</th>
<th>$\mu_{time}$</th>
<th>$\sigma_{time}$</th>
<th>$f_{ok}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKAZE</td>
<td>0.013</td>
<td>0.006</td>
<td>0.034</td>
<td>0.018</td>
<td>30.033</td>
<td>0.517</td>
<td>0.857</td>
</tr>
<tr>
<td>BRIEF</td>
<td>0.010</td>
<td>0.004</td>
<td>0.016</td>
<td>0.005</td>
<td>2.436</td>
<td>0.214</td>
<td>0.082</td>
</tr>
<tr>
<td>BRISK</td>
<td>0.013</td>
<td>0.006</td>
<td>0.033</td>
<td>0.018</td>
<td>6.499</td>
<td>0.804</td>
<td>0.878</td>
</tr>
<tr>
<td>BoostDesc</td>
<td>0.018</td>
<td>0.019</td>
<td>0.033</td>
<td>0.020</td>
<td>73.424</td>
<td>8.917</td>
<td>0.857</td>
</tr>
<tr>
<td>DAISY</td>
<td>0.009</td>
<td>0.003</td>
<td>0.014</td>
<td>0.006</td>
<td>31.225</td>
<td>1.254</td>
<td>0.143</td>
</tr>
<tr>
<td>FREAK</td>
<td>0.012</td>
<td>0.006</td>
<td>0.035</td>
<td>0.020</td>
<td>3.948</td>
<td>0.421</td>
<td>0.898</td>
</tr>
<tr>
<td>ORB</td>
<td>0.012</td>
<td>0.005</td>
<td>0.035</td>
<td>0.020</td>
<td>3.769</td>
<td>0.190</td>
<td>0.898</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.013</td>
<td>0.006</td>
<td>0.034</td>
<td>0.019</td>
<td>25.104</td>
<td>1.899</td>
<td>0.878</td>
</tr>
<tr>
<td>SURF</td>
<td>0.013</td>
<td>0.007</td>
<td>0.032</td>
<td>0.020</td>
<td>8.298</td>
<td>0.935</td>
<td>0.796</td>
</tr>
</tbody>
</table>
4.2 Final evaluation

Table 4.3: Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top three in bold.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>$\mu_t$</th>
<th>$\sigma_t$</th>
<th>$\mu_R$</th>
<th>$\sigma_R$</th>
<th>$\mu_{time}$</th>
<th>$\sigma_{time}$</th>
<th>$f_{ok}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKAZE</td>
<td>0.069</td>
<td>0.157</td>
<td>0.080</td>
<td>0.182</td>
<td>29.892</td>
<td>0.537</td>
<td>0.727</td>
</tr>
<tr>
<td>BRIEF</td>
<td>0.038</td>
<td>0.027</td>
<td>0.047</td>
<td>0.032</td>
<td>2.647</td>
<td>0.045</td>
<td>0.614</td>
</tr>
<tr>
<td>BRISK</td>
<td>0.029</td>
<td>0.013</td>
<td>0.036</td>
<td>0.019</td>
<td>7.109</td>
<td>0.111</td>
<td>0.614</td>
</tr>
<tr>
<td>BoostDesc</td>
<td>0.090</td>
<td>0.204</td>
<td>0.120</td>
<td>0.275</td>
<td>81.540</td>
<td>0.666</td>
<td>0.432</td>
</tr>
<tr>
<td>DAISY</td>
<td>0.034</td>
<td>0.019</td>
<td>0.043</td>
<td>0.021</td>
<td>33.182</td>
<td>0.721</td>
<td>0.500</td>
</tr>
<tr>
<td>FREAK</td>
<td>0.070</td>
<td>0.154</td>
<td>0.061</td>
<td>0.072</td>
<td>4.693</td>
<td>1.098</td>
<td>0.727</td>
</tr>
<tr>
<td>ORB</td>
<td>0.050</td>
<td>0.065</td>
<td>0.063</td>
<td>0.077</td>
<td>3.911</td>
<td>0.151</td>
<td>0.682</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.031</td>
<td>0.020</td>
<td>0.035</td>
<td>0.023</td>
<td>28.237</td>
<td>3.136</td>
<td>0.864</td>
</tr>
<tr>
<td>SURF</td>
<td>0.043</td>
<td>0.038</td>
<td>0.050</td>
<td>0.041</td>
<td>10.464</td>
<td>1.885</td>
<td>0.386</td>
</tr>
</tbody>
</table>

Table 4.4: Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK2, additionally the fraction of successful estimations are shown as $f_{ok}$. Top three values in bold.

<table>
<thead>
<tr>
<th>Descriptor</th>
<th>$\mu_t$</th>
<th>$\sigma_t$</th>
<th>$\mu_R$</th>
<th>$\sigma_R$</th>
<th>$\mu_{time}$</th>
<th>$\sigma_{time}$</th>
<th>$f_{ok}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKAZE</td>
<td>0.034</td>
<td>0.043</td>
<td>0.033</td>
<td>0.021</td>
<td>32.839</td>
<td>4.075</td>
<td>0.559</td>
</tr>
<tr>
<td>BRIEF</td>
<td>0.021</td>
<td>0.012</td>
<td>0.025</td>
<td>0.010</td>
<td>2.786</td>
<td>0.134</td>
<td>0.441</td>
</tr>
<tr>
<td>BRISK</td>
<td>0.023</td>
<td>0.008</td>
<td>0.026</td>
<td>0.009</td>
<td>7.242</td>
<td>0.462</td>
<td>0.441</td>
</tr>
<tr>
<td>BoostDesc</td>
<td>0.026</td>
<td>0.010</td>
<td>0.023</td>
<td>0.005</td>
<td>80.881</td>
<td>2.546</td>
<td>0.353</td>
</tr>
<tr>
<td>DAISY</td>
<td>0.018</td>
<td>0.005</td>
<td>0.024</td>
<td>0.005</td>
<td>41.780</td>
<td>9.120</td>
<td>0.412</td>
</tr>
<tr>
<td>FREAK</td>
<td>0.023</td>
<td>0.012</td>
<td>0.026</td>
<td>0.010</td>
<td>4.874</td>
<td>0.750</td>
<td>0.471</td>
</tr>
<tr>
<td>ORB</td>
<td>0.024</td>
<td>0.010</td>
<td>0.027</td>
<td>0.009</td>
<td>4.585</td>
<td>0.747</td>
<td>0.471</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.041</td>
<td>0.058</td>
<td>0.042</td>
<td>0.054</td>
<td>28.911</td>
<td>1.769</td>
<td>0.647</td>
</tr>
<tr>
<td>SURF</td>
<td>0.026</td>
<td>0.018</td>
<td>0.026</td>
<td>0.014</td>
<td>9.693</td>
<td>0.646</td>
<td>0.324</td>
</tr>
</tbody>
</table>

All methods use the same method for determining whether a point is an inlier or not, based on reprojection, and uses the same reprojection threshold, 4.0 pixels.

RANSAC with Umeyama’s algorithm as kernel is evaluated in two different ways, one using Umeyama’s algorithm with all inliers at the end, denoted “RANSAC-Ume”, and one version that skips this step and uses the resulting model given by RANSAC directly, denoted “RANSAC-Ume2”.

Plots showing these errors can be seen in Fig. 4.9 through 4.12 for the test sequences XYZ1, RPY-ROLL, and DESK1 (note that these are different sequences to the tables).

4.2 Final evaluation

This section presents the results obtained during the final evaluation, where the performance of the whole system is evaluated. This evaluation is done in two ways, partly with the odometry to gauge the quality of the whole trajectory and another metric specifically for evaluating relocalization, as outlined in the Method chapter.

The sequences for evaluation comes from the “EuRoC MAV” dataset, using the sequences recorded in a machine hall. These sequences are Machine Hall 01 (MH01, easy), Machine Hall 02 (MH02, easy), and Machine Hall 03 (MH03, medium). The sequences Machine Hall 04-05 (MH04 and MH05, both difficult) feature a near black section, designed to either test relocalization capabilities or navigation using the onboard IMU. The techniques here does not make use of the IMU, and with a specific evaluation step for relocalization, these two sequences are ignored.
### 4.2. Final evaluation

Table 4.5: Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence XYZ1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top two in bold, unless tied.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\mu_t$</th>
<th>$\sigma_t$</th>
<th>$\mu_R$</th>
<th>$\sigma_R$</th>
<th>$\mu_{time}$</th>
<th>$\sigma_{time}$</th>
<th>$f_{ok}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PnP-Iter.</td>
<td>0.281</td>
<td>0.175</td>
<td>0.233</td>
<td>0.132</td>
<td>5.404</td>
<td>0.387</td>
<td>0.388</td>
</tr>
<tr>
<td>RANSAC-AP3P</td>
<td>0.017</td>
<td>0.007</td>
<td>0.016</td>
<td>0.006</td>
<td>5.113</td>
<td>0.447</td>
<td>1.000</td>
</tr>
<tr>
<td>RANSAC-EPNP</td>
<td>0.017</td>
<td>0.007</td>
<td>0.016</td>
<td>0.006</td>
<td>7.923</td>
<td>2.921</td>
<td>1.000</td>
</tr>
<tr>
<td>RANSAC-ICP</td>
<td>0.016</td>
<td>0.027</td>
<td>0.025</td>
<td>0.028</td>
<td>3084.145</td>
<td>283.042</td>
<td>1.000</td>
</tr>
<tr>
<td>RANSAC-Iter.</td>
<td>0.016</td>
<td>0.007</td>
<td>0.016</td>
<td>0.006</td>
<td>5.208</td>
<td>0.545</td>
<td>1.000</td>
</tr>
<tr>
<td>RANSAC-P3P</td>
<td>0.017</td>
<td>0.007</td>
<td>0.016</td>
<td>0.006</td>
<td>5.208</td>
<td>0.545</td>
<td>1.000</td>
</tr>
<tr>
<td>RANSAC-Ume</td>
<td>0.042</td>
<td>0.025</td>
<td>0.038</td>
<td>0.021</td>
<td>1398.624</td>
<td>349.637</td>
<td>0.980</td>
</tr>
<tr>
<td>RANSAC-Ume2</td>
<td>0.021</td>
<td>0.009</td>
<td>0.016</td>
<td>0.006</td>
<td>1387.902</td>
<td>347.722</td>
<td>1.000</td>
</tr>
</tbody>
</table>

Table 4.6: Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK1, additionally the fraction of successful estimations are shown as $f_{ok}$. Top two in bold, unless tied.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\mu_t$</th>
<th>$\sigma_t$</th>
<th>$\mu_R$</th>
<th>$\sigma_R$</th>
<th>$\mu_{time}$</th>
<th>$\sigma_{time}$</th>
<th>$f_{ok}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PnP-Iter.</td>
<td>0.340</td>
<td>0.293</td>
<td>0.179</td>
<td>0.177</td>
<td>5.518</td>
<td>0.241</td>
<td>0.327</td>
</tr>
<tr>
<td>RANSAC-AP3P</td>
<td>0.080</td>
<td>0.164</td>
<td>0.175</td>
<td>0.676</td>
<td>9.436</td>
<td>2.005</td>
<td>0.878</td>
</tr>
<tr>
<td>RANSAC-EPNP</td>
<td>0.063</td>
<td>0.187</td>
<td>0.048</td>
<td>0.128</td>
<td>29.523</td>
<td>11.608</td>
<td>0.673</td>
</tr>
<tr>
<td>RANSAC-ICP</td>
<td>0.242</td>
<td>0.176</td>
<td>0.349</td>
<td>0.472</td>
<td>16021.314</td>
<td>47081.859</td>
<td>0.694</td>
</tr>
<tr>
<td>RANSAC-Iter</td>
<td>0.064</td>
<td>0.184</td>
<td>0.049</td>
<td>0.126</td>
<td>29.348</td>
<td>11.780</td>
<td>0.694</td>
</tr>
<tr>
<td>RANSAC-P3P</td>
<td>0.081</td>
<td>0.152</td>
<td>0.177</td>
<td>0.648</td>
<td>9.807</td>
<td>2.350</td>
<td>0.878</td>
</tr>
<tr>
<td>RANSAC-Ume</td>
<td>0.027</td>
<td>0.033</td>
<td>0.040</td>
<td>0.042</td>
<td>927.927</td>
<td>577.091</td>
<td>0.980</td>
</tr>
<tr>
<td>RANSAC-Ume2</td>
<td>0.028</td>
<td>0.026</td>
<td>0.041</td>
<td>0.033</td>
<td>982.081</td>
<td>566.161</td>
<td>0.980</td>
</tr>
</tbody>
</table>

Table 4.7: Mean and standard deviation of translation ($t$, in meters) and rotation error ($R$, in radians), along with average time (in milliseconds) taken and the standard deviation for test sequence DESK2, additionally the fraction of successful estimations are shown as $f_{ok}$. Top two in bold, unless tied.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$\mu_t$</th>
<th>$\sigma_t$</th>
<th>$\mu_R$</th>
<th>$\sigma_R$</th>
<th>$\mu_{time}$</th>
<th>$\sigma_{time}$</th>
<th>$f_{ok}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>PnP-Iter.</td>
<td>0.409</td>
<td>0.244</td>
<td>0.112</td>
<td>0.084</td>
<td>5.082</td>
<td>0.236</td>
<td>0.000</td>
</tr>
<tr>
<td>RANSAC-AP3P</td>
<td>0.116</td>
<td>0.186</td>
<td>0.198</td>
<td>0.636</td>
<td>8.682</td>
<td>2.081</td>
<td>0.765</td>
</tr>
<tr>
<td>RANSAC-EPNP</td>
<td>0.026</td>
<td>0.015</td>
<td>0.028</td>
<td>0.011</td>
<td>28.230</td>
<td>11.125</td>
<td>0.529</td>
</tr>
<tr>
<td>RANSAC-ICP</td>
<td>0.189</td>
<td>0.090</td>
<td>0.197</td>
<td>0.112</td>
<td>4376.203</td>
<td>12867.462</td>
<td>0.353</td>
</tr>
<tr>
<td>RANSAC-Iter</td>
<td>0.026</td>
<td>0.016</td>
<td>0.028</td>
<td>0.011</td>
<td>28.481</td>
<td>11.058</td>
<td>0.529</td>
</tr>
<tr>
<td>RANSAC-P3P</td>
<td>0.116</td>
<td>0.186</td>
<td>0.198</td>
<td>0.636</td>
<td>9.119</td>
<td>2.370</td>
<td>0.765</td>
</tr>
<tr>
<td>RANSAC-Ume</td>
<td>0.157</td>
<td>0.195</td>
<td>0.137</td>
<td>0.203</td>
<td>777.801</td>
<td>442.333</td>
<td>0.618</td>
</tr>
<tr>
<td>RANSAC-Ume2</td>
<td>0.113</td>
<td>0.143</td>
<td>0.212</td>
<td>0.741</td>
<td>753.502</td>
<td>423.240</td>
<td>0.882</td>
</tr>
</tbody>
</table>
### 4.2. Final evaluation

#### Figure 4.9: Translational errors as percentage of translational movement in a sequence moving back and forth, the sequence \textit{XYZ1}.

#### Figure 4.10: Rotational errors as percentage of the angle rotated in a sequence rotating while stationary, the sequence \textit{RPY-ROLL}.

#### Figure 4.11: Translational errors as percentage of translational movement in a sequence both rotating and translating the camera, the sequence \textit{DESK1}.

#### Figure 4.12: Rotational error as a percentage of the angle rotated in a sequence rotating and translating the camera, the sequence \textit{DESK1}.
4.2. Final evaluation

Table 4.8: The result of the odometry on the three sequences, using proposed system.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RMSE [m]</th>
<th>Mean [m]</th>
<th>Median [m]</th>
<th>Std. dev. [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH01</td>
<td>0.842823</td>
<td>0.782310</td>
<td>0.782512</td>
<td>0.313595</td>
</tr>
<tr>
<td>MH02</td>
<td>0.804023</td>
<td>0.718006</td>
<td>0.735866</td>
<td>0.361829</td>
</tr>
<tr>
<td>MH03</td>
<td>1.696580</td>
<td>1.480941</td>
<td>1.265126</td>
<td>0.827765</td>
</tr>
<tr>
<td>MH03 AKAZE</td>
<td>1.969838</td>
<td>1.736493</td>
<td>1.572269</td>
<td>0.929975</td>
</tr>
</tbody>
</table>

Table 4.9: The result of the odometry on the three sequences, using original system.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RMSE [m]</th>
<th>Mean [m]</th>
<th>Median [m]</th>
<th>Std. dev. [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH01</td>
<td>1.963259</td>
<td>1.877619</td>
<td>1.922815</td>
<td>0.573529</td>
</tr>
<tr>
<td>MH02</td>
<td>1.934054</td>
<td>1.835816</td>
<td>1.735050</td>
<td>0.608560</td>
</tr>
<tr>
<td>MH03</td>
<td>3.517432</td>
<td>3.161920</td>
<td>2.694119</td>
<td>1.540970</td>
</tr>
</tbody>
</table>

The training data for FAB-MAP is very different from the scenarios tested here, being recorded in an apartment and uses relatively small vocabularies of roughly 5000 words. A minimum probability threshold of 0.97 for accepting a keyframe output from FAB-MAP as likely is employed throughout. This threshold is lower than the threshold used in the original paper (at 0.99 [20]), and is set as such due to the relatively small vocabulary and short training sequence. The training data was collected using a completely different camera to the one used in evaluation (a mobile phone for training), and with little similarity in scene—a machine hall for evaluation and as mentioned an apartment for training. The training sequence was a roughly three minutes long sequence featuring no loops. Every fourth frame was used from the recording to not have too much overlap between frames (yielding 1148 training frames), and is thus relatively short.

BRIEF-32, AKAZE, ORB, and BRISK vocabularies and Chow-Liu trees was trained, but unless otherwise stated the descriptor used for keyframe estimation is BRIEF-32. FAB-MAP with BRIEF-32 gave best results when used to recover from tracking loss due to motion blur on a recorded sequence, and was hence chosen. The descriptor for frame to frame navigation is BRISK, and key points are extracted using AGAST with a threshold of 50, unless otherwise stated, followed by ANMS filtering.

When relocating, a minimum number of inliers must be found with RANSAC for the pose to be deemed trustworthy enough. This threshold is here set to 30 inliers.

Odometry

The ground truth and estimated trajectory for three different sequences are shown in Fig. 4.13 through 4.15. The trajectory of the original tracker used in the system is also shown. The trajectories are plotted in the same coordinate system and aligned using the method used in [42] (as per Sec. 3.3). Fig. 4.16 shows the MH03 sequence with the slower AKAZE key points and descriptors. All of the sequences are started at frame 100, to skip the initial frames where no poses are available, and runs until all other frames with poses have been processed.

Error statistics are shown in Tab. 4.8 (for AGAST and BRISK), with the statistics for the original implementation shown in Tab. 4.9.

Computation times per frame can be seen in Tab. 4.10 and 4.11.

Relocalization

The goal of this test is to check how well relocalization is handled by the system. The quality of the relocalization is in nature somewhat scenario dependent and consequently, the results are here presented using different levels of strictness. The sequences used and how they where used can be seen in Tab. 4.12, with how many of the initial frames where skipped (due to the

\[2\] Evaluated on a computer with an Intel Xeon CPU E5-2637 v4 @ 3.5GHz processor.
4.2. Final evaluation

Figure 4.13: The estimated trajectory for proposed and original system along X-Y axis overlaid with ground truth. The sequence is MH01.

Figure 4.14: The estimated trajectory for proposed and original system along X-Y axis overlaid with ground truth. The sequence is MH02.

Figure 4.15: The estimated trajectory for proposed and original system along X-Y axis overlaid with ground truth. The sequence is MH03.

Figure 4.16: The estimated trajectory along X-Y axis overlaid with ground truth. The sequence is MH03. Here AKAZE is used as key point detector and descriptor extractor as opposed to AGAST and BRISK, respectively.

Table 4.10: Mean computation times per frame for different parts of the system.

<table>
<thead>
<tr>
<th>Task</th>
<th>AGAST and BRISK mean computation time [ms]</th>
<th>AKAZE mean computation time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keypoint detection, ANMS, and descriptor extraction</td>
<td>21.0</td>
<td>127.0</td>
</tr>
<tr>
<td>FAB-MAP, loop detection</td>
<td>29.6</td>
<td>46.6</td>
</tr>
<tr>
<td>Initial pose estimation and 2D-3D correspondence</td>
<td>20.8</td>
<td>18.7</td>
</tr>
<tr>
<td>Map matching and bundle adjustment</td>
<td>92.6</td>
<td>67.6</td>
</tr>
</tbody>
</table>
4.2. Final evaluation

Table 4.11: Detailed timing statistics for processing a whole frame.

<table>
<thead>
<tr>
<th>Version</th>
<th>Mean [ms]</th>
<th>Median [ms]</th>
<th>Std. dev. [ms]</th>
<th>Min [ms]</th>
<th>Max [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>44.6</td>
<td>44</td>
<td>5.8</td>
<td>24</td>
<td>65</td>
</tr>
<tr>
<td>AGAST, BRISK</td>
<td>164.1</td>
<td>166.2</td>
<td>31.9</td>
<td>82.5</td>
<td>302.3</td>
</tr>
<tr>
<td>AKAZE</td>
<td>259.9</td>
<td>261.6</td>
<td>30.4</td>
<td>161.4</td>
<td>349.1</td>
</tr>
</tbody>
</table>

Table 4.12: The sequences used for relocalization evaluation

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Skipped frames</th>
<th>Build map (#frames)</th>
<th>Pure localization (#frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH01</td>
<td>850</td>
<td>1500</td>
<td>1000</td>
</tr>
<tr>
<td>MH02</td>
<td>850</td>
<td>1500</td>
<td>500</td>
</tr>
<tr>
<td>MH03</td>
<td>200</td>
<td>900</td>
<td>600</td>
</tr>
</tbody>
</table>

Table 4.13: Results when relaxing keyframe insertion constraints. All errors are in meters. Recall is given for $\tau_{\text{inside}} = 0.9$ and $d_{\text{max}} = 0.5$ m.

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Setting</th>
<th>Num. poses</th>
<th>Mean error</th>
<th>Std. dev.</th>
<th>Min</th>
<th>Max</th>
<th>Recall [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH03</td>
<td>Normal</td>
<td>94</td>
<td>0.250</td>
<td>0.165</td>
<td>0.015</td>
<td>0.662</td>
<td>72</td>
</tr>
<tr>
<td>MH03</td>
<td>More</td>
<td>165</td>
<td>0.186</td>
<td>0.184</td>
<td>0.010</td>
<td>0.786</td>
<td>85</td>
</tr>
<tr>
<td>MH03</td>
<td>Most</td>
<td>82</td>
<td>0.224</td>
<td>0.177</td>
<td>0.006</td>
<td>0.649</td>
<td>57</td>
</tr>
<tr>
<td>MH01</td>
<td>Normal</td>
<td>28</td>
<td>0.106</td>
<td>0.123</td>
<td>0.007</td>
<td>0.474</td>
<td>37</td>
</tr>
<tr>
<td>MH01</td>
<td>Most</td>
<td>45</td>
<td>0.109</td>
<td>0.142</td>
<td>0.006</td>
<td>0.638</td>
<td>51</td>
</tr>
</tbody>
</table>

axis test in the beginning of them), how many frames were used to build the map, and for how many frames pure localization was ran, i.e. navigation against only the keyframes.

Figures 4.17 through 4.20 shows the L2-distance error (when aligned) between the ground truth pose and estimated pose as a histogram. Below each histogram is a stem plot, showing the estimated movement between the test frame and the selected keyframe.

How many of the frames where a pose “should” have been found based on reprojection is shown in the figures 4.21 through 4.24, using four different maximum absolute distance differences thresholds. If the absolute distance difference is above these threshold values, $d_{\text{max}}$, it is considered a false positive, otherwise a true positive. Varying the fraction of a frame’s points visible in a keyframe, $\tau_{\text{inside}}$, yields the various percentages of recall. Note that the graphs omit $\tau_{\text{inside}} < 0.5$. Conceptually $\tau_{\text{inside}}$ can be thought of as deciding the fraction of the current frame’s points that needs to be visible in a keyframe before it is deemed reasonable that the system can recover the pose.

The result of relaxing the constraints for keyframe insertion is shown in Tab. 4.13. Original and modified keyframe insertion values are given in Tab. 4.14. Error histograms and recall graphs for these tests can be found in Appendix A.2.

As a qualitative result, Fig. 4.25 through 4.27 show the trajectories visually, denoting in color whether the system was mapping or purely localizing. The trajectories are the ground truth trajectories. Note that the UAV does not always face the same direction as it is moving in, meaning what is in view can not be read from the plots.

Table 4.14: Parameter values for different keyframe insertion frequency settings. Distances are in meters and angles in radians.

<table>
<thead>
<tr>
<th>Setting</th>
<th>min dist.</th>
<th>min angle</th>
<th>max dist.</th>
<th>max angle</th>
<th>min frames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>0.1</td>
<td>$\frac{\pi}{18}$</td>
<td>0.8</td>
<td>$\frac{\pi}{18}$</td>
<td>10</td>
</tr>
<tr>
<td>More</td>
<td>0.05</td>
<td>$\frac{\pi}{36}$</td>
<td>0.4</td>
<td>$\frac{\pi}{36}$</td>
<td>5</td>
</tr>
<tr>
<td>Most</td>
<td>0.01</td>
<td>$\frac{\pi}{72}$</td>
<td>0.2</td>
<td>$\frac{\pi}{72}$</td>
<td>3</td>
</tr>
</tbody>
</table>
4.2. Final evaluation

Figure 4.17: Distance error between ground truth and estimated translations between a matched keyframe and test frame. The sequence is MH01. A pose is estimated in 28 out of the 1000 test frames, yielding an estimation percentage of 2.8%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is \( \approx 64\% \).

Figure 4.18: Distance error between ground truth and estimated translations between a matched keyframe and test frame. The sequence is MH02. A pose is estimated in 11 out of the 500 test frames, yielding an estimation percentage of 2.2%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is \( \approx 55\% \).

Figure 4.19: Distance error between ground truth and estimated translations between a matched keyframe and test frame. The sequence is MH03. A pose is estimated in 94 out of the 600 test frames, yielding an estimation percentage of approximately 15.7%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is \( \approx 23\% \).

Figure 4.20: Distance error between ground truth and estimated translations between a matched keyframe and test frame. Here, AKAZE detector and extractor is used instead of AGAST and BRISK. The sequence is MH03. A pose is estimated in 97 out of the 600 test frames, yielding an estimation percentage of approximately 16.2%. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is \( \approx 53\% \).
4.2. Final evaluation

Figure 4.21: The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. The sequence is MH01.

Figure 4.22: The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. The sequence is MH02.

Figure 4.23: The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. The sequence is MH03.

Figure 4.24: The recall for various threshold values, given four different maximum error allowed, $d_{max}$. The threshold $\tau$ define the fraction of a keyframe’s points that must be projectable inside the current view for it to be deemed relevant. AKAZE key points and descriptors are used instead of AGAST and BRISK respectively. The sequence is MH03.
Figure 4.25: A 3D plot showing the ground truth trajectory, colorized depending on whether the mapping was active or if not. Green dashed lines represent the detection of a keyframe that could be used to successfully estimate a pose. The sequence is MH01.

Figure 4.26: A 3D plot showing the ground truth trajectory, colorized depending on whether the mapping was active or if not. Green dashed lines represent the detection of a keyframe that could be used to successfully estimate a pose. The sequence is MH02.

Figure 4.27: A 3D plot showing the ground truth trajectory, colorized depending on whether the mapping was active or if not. Green dashed lines represent the detection of a keyframe that could be used to successfully estimate a pose. The sequence is MH03.
This chapter examines the results obtained and presented in Ch. 4, Results, and puts them in relation to prior studies of similar nature as this work. In addition to the results, the overarching methodology employed is analyzed, and the last section, section 5.3, takes a look at the work in a wider context, taking ethical, environmental, and societal aspects into consideration.

5.1 Results

Evaluations were conducted at two different stages of the project, and this section discusses them both in separate sections. The first section (Initial evaluations - Descriptors) handles the first sets of evaluations made, analysing them and commenting on choices made due to the results. The second and last section (Final evaluations - Relocalization) focuses on the evaluation of the whole system, setting it in relation to already proposed systems.

Initial evaluations - Descriptors

The precision-recall evaluation shows that none of the descriptors are perfect. This is especially evident at larger perspective changes where the descriptors give very low AUC values with the use of the simple NNDR matching scheme. Motion blur also exists within the tests, meaning that it is not only the transformation invariance that is tested. Coping with e.g. motion blur nevertheless has to be dealt with in a SLAM system, so the descriptors have to be somewhat tolerant to reasonable levels of motion blur as well.

SIFT performs above the rest of the descriptors with generally higher AUC values. SIFT on the other hand is more demanding to compute than for example BRISK and BRIEF, and being non-binary means matching is slower.

BRISK, DAISY, FREAK and ORB too perform relatively well, being about on par with each other. This does make some sense as, DAISY aside, they by and large are based on similar premise, e.g. pairwise intensity comparisons of points surrounding the key points. As DAISY is not binary and does take a considerable time to compute, one of the other three is for this application preferred.

AKAZE performs slightly below these descriptors at small translations and rotations, but shows an increase in performance at greater differences to be on par or slightly above BRISK, FREAK, and ORB.
5.1. Results

BRIEF is not rotationally invariant, and this fact reflects poorly on its performance. BRIEF is on the other hand the fastest descriptor to compute, and as such is of special interest for bag of word matching, where each individual descriptor matters less. The version of BRIEF tested is BRIEF-32, and not BRIEF-64 which likely yields slightly better results at the expense of computation time and storage requirements.

While results are omitted for brevity, these trends hold true for the other test sequences as well. Differences exist in the ordering between descriptors with AUC values close, e.g. BRISK performing better than FREAK for certain tests and vice versa.

The quality of the pose estimation by and large mirrors the results from the precision-recall evaluation, with some details being more evident. Observing the result of the RPY-ROLL test sequence, that rolls in place, which descriptors are rotationally invariant become evident. Both BRIEF and DAISY fail to estimate a valid pose for most of the frames. AKAZE shows an increase in performance compared to the precision-recall curves, successfully estimating poses for more frames than other descriptors excluding SIFT on DEK1 and DESK2.

These findings are comparable to those found by Cowan et al. [18] who deemed BRISK, FREAK, and ORB the best to use for tracking due to their performance in relation to their computation time. Cowan et al. do however not clearly indicate which “secondary” image of the “Affine Covariant Regions” dataset [59] they performed the precision-recall evaluations with. It is thus not evident how large transformations and distortions that yield the PRC curves presented.

Figat, Kornuta, and Kasprzak [24] also evaluated descriptors using the “Affine Covariant Regions” dataset, focusing on binary descriptors. While fewer descriptors were examined, they conclude that FREAK (with ORB detector, variation on FAST) is the most appropriate for their use case, but note that BRISK (with ORB detector) perform better in presence of a combination of rotation and zooming.

Focusing instead on the illumination evaluation and it is clear that most descriptors can deal with illumination changes well. Many descriptors are able to cope with even the extreme illumination changes shown in Fig. 4.7 through 4.8. AKAZE is the descriptor consistently performing best, achieving AUC score of above 0.99 for all the harder test cases (0.94 in appendix A.1 on easier sequences, but note AUC calculation caveat given). AKAZE key points was used for this evaluation due to the AKAZE descriptor requiring them, and the result might thus be biased. Other top performing descriptors in regards to illumination change are ORB, BRISK, and SIFT – all with AUC scores over 0.90 for all test cases.

The DAISY descriptor shows an inability to cope with more extreme illumination changes, having the lowest or among the lowest AUC for all test images. BRIEF and SURF too generally performs worse than the rest, DAISY aside.

In presence of more moderate illumination changes, all descriptors perform acceptably to exceptionally. The “Affine Covariant Regions” feature a sequence for illumination invariance (“leuven”), but to the author’s knowledge this is the first more comprehensive such evaluation performed with modern descriptors. While not commented on, the evaluation in [18] found FREAK and BRISK the most performing descriptors, followed by SIFT and LATCH. AKAZE was not evaluated. The results are relatively consistent with the findings here, although notably the more extreme illumination changes shows FREAK suffering in performance.

Taking in to account all of these results, the best descriptor had to be chosen to be used in the system. SIFT overall performed best, but have some disadvantages - it is slow to compute and non-binary, meaning matching is slower than a binary descriptor where distance can be calculated using the Hamming distance. AKAZE is binary and performs fairly well, but it is also slow to compute due to the requirement of using AKAZE key points.

ORB, BRISK, and FREAK were deemed the best candidates – all binary and relatively on par in terms on performance and time to compute. FREAK performs best in terms of precision-recall under transformation, followed by BRISK and last ORB. FREAK however suffers when it comes to illumination, being well behind both BRISK and ORB. These factors
combined meant that BRISK became the choice as initial descriptor used in the system for performing odometry. Even BRIEF could be sufficient for pure odometry. In the evaluation performed by Hartmann, Klussendorff, and Maehle [40] BRIEF was even found to work best for odometry. Pure odometry is not of interest here however, as we wish to be able to find the poses with larger transformations than exhibited frame-to-frame when detecting loops or relocating.

BRIEF was however chosen as the descriptor for BoW extraction. This choice came down to the faster extraction of BRIEF descriptors for the more numerous key points used for FAB-MAP. The less rotational invariance of BRIEF can potentially be considered an advantage in this case. This is due to keyframes being more distinguishable in cases when multiple keyframes exists that observe the same “landmarks”.

Ultimately, the choice of descriptor comes down to the requirements of the system. Hopefully these evaluations can serve as pointers towards the correct choice for other SLAM applications as well.

Initial evaluations - Pose estimation

Evident is the need for a robust pose estimation method as illustrated by the fact that the “PnP Iterative” method fails in 60% of the frames for even the most simplistic sequence, XYZ1. Failure in this case means estimating a distance over \( d_{\text{disparity}} = 5 \) m. A robust estimation method is for this application required.

Out of all methods the two based on Umeyama’s algorithm yield the most number of successfully estimated poses. The poses estimated using the variant with no refinement, “RANSAC-Ume2”, have a larger error than the 2D-3D based methods for small translations and rotations, suggesting that the method might not be as well suited for odometry. At larger changes in perspective, and especially under both translation and rotation, it estimate poses closer to ground truth than any other method.

Using Umeyama’s algorithm with the inliers found (“RANSAC-Ume”) generally leads to worse performance compared to “RANSAC-Ume2”. This is most apparent in the sequences XYZ1 and DESK2 at greater displacement relative to the first frame. It is possible this worse performance is due to the sensitivity to noise, either from mismatches within the RANSAC threshold or the position of the points themselves. The computation time is the primary downside with both Umeyama based methods, being two orders of magnitude slower than the fastest 2D-3D methods.

AP3P and P3P are the two best performing out of the 2D-3D methods, successfully estimating a pose from more frames than the rest of the methods. The OpenCV RANSAC setting AP3P and P3P use these methods as the RANSAC kernel. Inliers found are then used with EPNP to estimate the resulting pose. Additionally, these two methods are also the fastest to execute out of all RANSAC based methods. Both these methods use a minimal sample set size of four \(^1\) compared to the five points EPNP uses, which is used as kernel in OpenCV for all other methods \(^2\). Fewer points used by the kernel means that fewer samples have to be drawn for a given probability of success, \( P \), as characterized with

\[
N_{\text{iterations}} = \frac{\log (1 - P)}{\log (1 - p^k)} \tag{5.1}
\]

given the inlier ratio, \( p \), and the number of data points used by the kernel, \( k \) [79, p. 282]. Both \( P \) and \( p \) are fixed, meaning more iterations need to be run with the increased \( k \), and the result is consistent with this.

\(^1\)Three points to estimate a pose, and one to select which pose to pick in case of multiple solutions

“RANSAC-Iterative” followed by ICP proved in this evaluation to not improve the estimates, rather generally deteriorate them. As the dataset this method was tested on is RGB-D, it is unlikely that the point clouds where too sparse, each frame having a point cloud constructed from every pixel with valid depth. It is thus likely the results are due to noisy and/or poor depth for a number of these pixels, something ICP deals poorly with. Downsampling was also tried, but it did not improve the results. Using a more modern technique for point cloud registration is likely to improve results, but this was not pursued further as bundle adjustment instead was selected for improving poses, commonly used in odometric and SLAM systems (see examples in Pose estimation & odometry, Ch. 2).

A note is however in order for the computation time of ICP, it was implemented in a quick and dirty way, and is by no means representative of the actual computation time which ICP would take. Each frame meant a point cloud was rebuilt for both frames, then converted to a PCL point cloud - where ICP was then performed.

OpenCV implementation uses the standard RANSAC algorithm, but multiple extensions and variations to RANSAC have been proposed, M-estimator Sample Consensus (MSAC), Maximum Likelihood Estimation Sample Consensus (MLESAC), Progressive Sample Consensus (PROSAC) to name a few. Depending on which is used, these promise to either speed up the process or yield better inliers compared to standard RANSAC. Using one of these methods can thus be beneficial as well.

Out of the OpenCV algorithms available for pose estimation, these tests indicate that there seem to be little reason to use anything other than AP3P or P3P. This is slightly surprising given that reprojection optimization with the iterative solver is the default and preferred OpenCV method. It seems likely that this difference is due to fewer points being considered inliers when using EPNP in the minimal sample step as given that it is used for both EPNP and the iterative solver, as can be seen by the identical $f_{ok}$ values of both these methods.

A concluding note is that RANSAC based PnP is limited in OpenCV. The minimal sample set kernel and the final non-linear optimization step with all inliers can not be controlled independently. OpenCV will for example always use EPNP, either as the minimal sample set kernel or the final step. This was a major reason for moving away from purely OpenCV based pose estimation methods and instead only use P3P inliers and a rough estimate for guided descriptor matching with a final refinement handled by bundle adjustment (see Sec. 3.2).

Final evaluations - Odometry

The odometric evaluation shows that the odometry is not perfect and exhibits drift. All sequences start and end at roughly the same position, but none of the estimated trajectories fully return to the start position. This is expected, and illustrates the need for loop detection so that the trajectory subsequently can be refined. Loop closing, as mentioned in Delimitations, is not done here.

A caveat is that the visual system where this sub-system is implemented utilizes a different strategy for calibration than EuRoC. This meant that undistortion of images was not optimal, and frames exhibited some scale distortion near the upper left and lower right of images. Though both the original odometry system and the proposed use the same calibration, a risk exists that the trajectories are needlessly worse.

Observing the trajectories in detail for MH01 and MH02, Fig. 4.13 and 4.14 respectively, show that the general trajectory is followed relatively faithfully, but as noted, not to perfection. MH02 exhibits the least drift, ending nearly back at the beginning. This lower drift for MH02 can partly be attributed to it being a shorter sequence, being roughly half a minute shorter than MH01.

---

3 As it is called in OpenCV, RANSAC with an EPNP kernel followed by non-linear optimization using the inliers.

4 See OpenCV documentation.
5.1. Results

Figure 5.1: Reconstruction from running SLAM on a recorded sequence walking around a house while looking at it. Red markers are estimated poses.

The trajectory estimated from the MH03 sequence shows a worse result. The faster and more aggressive movement of the MAV means the trajectory is not as closely followed as in MH01 and MH02. A reason could be due to the last frame’s pose being used to initialize searching of tracks. The higher speeds in the sequence means a higher risk of features moving outside the circular maximum search radius employed. While outside of the scope, it is likely that using inertial measurement data to obtain a better initial guess for the pose would improve the odometry in general, and in particular in fast moving sequences like MH03.

Using AKAZE instead of AGAST and BRISK does not improve the odometric results, instead it deteriorates them. RMSE increases by roughly 0.3 m with AKAZE instead of AGAST and BRISK. AKAZE was found to outperform BRISK when estimating poses but not in terms of precision and recall, meaning the worse odometry both supports and contradicts the results from the descriptor evaluation. It is possible that the discrepancy is due to the key point detection and would thus support the findings by Cowan et al. [18], where it was found that AGAST outperforms AKAZE on images with blur and viewpoint change – both present in the test sequences. It is possible that changing parameter values for the system makes up for all or some of this difference. Black-box parameter-optimization to verify this is however well out of scope here.

Comparing the trajectories to those of the original tracker used it can be seen that the proposed tracker diverges more slowly from ground truth as more frames are processed. This is also mirrored in the RMS errors in Tab. 4.8 and 4.9, where the proposed system has more than halved the error compared to the original implementation in each of the sequences.

The original implementation is a fair bit faster than the proposed system, approximately 3.7 times faster. The proposed system can be optimized, but this is not pursued further in this proof-of-concept.

An example of SLAM reconstruction of a sequence recorded while moving a stereo pair of cameras around a house can be seen in Fig. 5.1 and 5.2 using the proposed system. Notice in
5.1. Results

Figure 5.2: Reconstruction from running SLAM on a recorded sequence walking around a house while looking at it, focused on the mesh where overlap between start and end exists. Note they do not perfectly overlap, indicating loop closing is required.

Table 5.1: Absolute trajectory RMSE comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Proposed RMSE [m]</th>
<th>Original RMSE [m]</th>
<th>ORBSLAM2 RMSE [m]</th>
<th>DSO RMSE [m]</th>
<th>S-MSCKF RMSE [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH01</td>
<td>0.842823</td>
<td>1.963259</td>
<td>0.035</td>
<td>1.003719</td>
<td>N/A</td>
</tr>
<tr>
<td>MH02</td>
<td>0.804023</td>
<td>1.934054</td>
<td>0.018</td>
<td>2.171987</td>
<td>N/A</td>
</tr>
<tr>
<td>MH03</td>
<td>1.696580</td>
<td>3.517432</td>
<td>0.028</td>
<td>1.760647</td>
<td>~0.4</td>
</tr>
</tbody>
</table>

The wall added when getting close to the start position does not end where it began, resulting in part of the short side wall sticking out.

A comparison to some other odometric or SLAM systems evaluated on EuRoC can be seen in Table 5.1. These systems are ORB-SLAM2 [64], DSO [22], and S-MSCKF [78], but a comparison cannot be done straightforwardly between them as a number of caveats exists for each of the different algorithms. ORB-SLAM2 is also a stereo system, but actively closes loops detected, and this is in the proposed system currently not done.

DSO is a monocular odometric method that does not close loops. It was also not evaluated on the whole EuRoC sequences, and does not give a pose estimate for every frame. S-MSCKF is a Kalman-filter based approach and uses inertial measurements as well as visual. S-MSCKF was only evaluated on the MH03 sequence, and the results are presented in graph form, without exact numbers given.

When taking differences in systems into consideration, it can be seen that the proposed system compares favorably or on par with DSO and S-MSCKF. DSO, being monocular, has a higher risk of suffering from scaling issues which in part explains the worse odometry. S-MSCKF

---

5See the supplementary material at [https://vision.in.tum.de/research/vslam/dso](https://vision.in.tum.de/research/vslam/dso)
MSCKF is hard to properly comment on, but seems to perform slightly better than the proposed system. It, as mentioned, utilizes inertial measurements which are not used by the proposed system.

The great performance of ORBSLAM2 in comparison to the other systems on the other hand illustrates the benefit of being able to detect and close loops, something this system aims to enable.

**Final evaluations - Relocalization**

The main rationale behind the proposed system is not improving odometry. While this too is desired, the main goal is to be able to recover from tracking loss and detect loops which can then be closed.

As the evaluation methodology for relocalization has not been applied to other systems, true comparisons can not be done. Much of the results here are application dependent, and discussion of them is hence done rather generally. This is also the reason for the multiple $d_{max}$ thresholds used when presenting the results.

Observing the histograms of translational errors and the estimated movements (Fig. 4.17 to 4.19), it is evident that poses found are not always excellent - with errors in translation sometimes reaching towards a meter. The error is generally increasing with the estimated distance between tested frame and the keyframe. This information is available to the system, and a simple solution to improve the accuracy of poses estimated is thus to disallow relocalization when the estimated movement is above a set threshold. This imposes the penalty of fewer valid poses being found, and might not be needed with later loop closing.

The sequence MH03 (medium) yields a higher percentage of frames where relocalization can be done compared to MH01 and MH02. This is expected as the movement in the MH03 sequence is more concentrated, observing features already seen in the machine hall more frequently. This is also reflected in the recall graphs, with overall a higher resulting recall at lower fractions of reprojected points visible ($\tau_{inside}$). This is despite the fact that the quality of this harder sequence yields worse poses, as shown by higher errors and consequently lower precision.

Comparing MH03 when using AGAST and BRISK compared to AKAZE key point detector and descriptors it is evident that the choice impacts the performance. The percentage of estimated poses with error less than 10 cm when relocating with AGAST and BRISK is approximately 23%, compared to 53% for AKAZE. A few more frames also yield estimates, 16.2% compared to the 15.7% of AGAST and BRISK. This is despite the fact that BRIEF descriptors are used for keyframe estimation with FAB-MAP in both the cases, implying that pose estimation with AKAZE works better, consistent with the pose estimation results of the initial evaluations. There is however a caveat here, which we will get back to shortly.

Turning the attention to the recall graphs, Fig. 4.21-4.23, and it is clear that many potential poses are missed in MH01 and MH02. There are multiple potential reasons to this, one being FAB-MAP not being confident enough or miss-predicting, another that that too few keyframes are inserted. Too few keyframes and there is risk a keyframe inserted is poor, e.g. blurry, making the pose estimation more difficult despite an overlap of points between the keyframe and the testframe. The effect of a greater overlap between training sequence (mapping) and test sequence (pure localization) can be seen when comparing the result of MH03 to the MH01 and MH02 sequences, with a higer recall obtained in MH03.

Comparing MH03 when using AGAST and BRISK to the result of using AKAZE for both key point detector and descriptor extractor shows that AGAST and BRISK exhibit higher recall at higher values of $d_{max}$. This is in contrast to the distance error where AKAZE yielded lower errors and an increased number of poses successfully estimated. All other parameters are fixed between these two configurations the most likely reason is due to the AKAZE key point detector. As FAB-MAP uses the non-ANMS filtered key points for constructing the BoW these key points differ in the two runs, despite both using BRIEF-32 as the descriptor for BoW extraction.
The minimum and maximum distance and angles for keyframe insertion is the same for both configurations. A subtlety is that FAB-MAP might want keyframes inserted at different frames during the grace window between allowing and forcing keyframe insertions due to the different keypoints used. This in turn means that comparisons between both methods must be done with care as relocalization was done under slightly different circumstances. With this in mind, the increased recall despite the fewer and worse estimates of AGAST and BRISK indicate that BRIEF-32 seem to work better with FAB-MAP when using AGAST key points. This can be part of the reason for the increased number of estimates mentioned earlier for AKAZE compared to AGAST and BRISK.

The increased recall in MH03 over MH01 and MH02 suggests increasing the frequency at which keyframes may be inserted at might improve the results, something evaluated and shown in Tab. 4.13. The results indeed support that increasing the keyframe insertion frequency improve the result, at least in moderation. The “More” setting on the MH03 sequence shows an increase in recall and at the same time decreases the mean error. The only cost is an increased max error, something which can be mitigated by using a maximum relocalization threshold (see Appendix A.2 for results).

Relaxing the insertion constraints even further has an adverse effect on performance for the MH03 sequence. A lower recall and fewer number of valid poses detected than the baseline “Normal” setting shows that FAB-MAP is not immune to aliasing issues common for bag-of-words based techniques. This effect seems to be exacerbated by the many laps over the same area compared to that of MH01, where the most extreme relaxation still increases the recall at a very slight cost in mean error and max error. This is again mitigable by disallowing too large movements.

Both these results combined can also be taken as an indication that a more adaptive keyframe insertion scheme might be of benefit. Nearest neighbor search of all keyframes is possible with few of them, but impractical with large number of them. A heuristic is thus the most promising idea, a simple example being bucketing keyframe poses (e.g. with locality sensitive hashing).

The results indicate that the frequency of keyframe insertions is an important decision. The simplistic scheme used here is not enough when repeatedly mapping the same general area. The inverted index structure used in FAB-MAP 2.0 means that the algorithm is not limited in the number of keyframes that can be added, searching stays relatively constant regardless of the number of them [19]. Naturally, insertion frequency is also dictated by memory availability.

Note that the performance of FAB-MAP can likely be improved on with a better trained vocabulary [94] and statistical model for it. Better in this context means training in an environment more similar to the sequences and using more training data to produce larger vocabularies with support from clusters of larger sizes. FAB-MAP was however found to perform well in cases where the training and test environments differ in terms of precision, but at the cost of recall [20].

With care, some comparisons can nonetheless be made with other systems performing relocalization, for example the strategy proposed by Straub et al. [75], based on locality sensitive hashing and binary descriptors. The tracking loss evaluation was done when walking down a corridor facing the same direction as the path traversed. Previously seen features was thus present for many of the subsequent frames once tracking loss was simulated. The trajectory was also relatively straight meaning that rotational invariance of features was less necessary than in the evaluation carried out here. The system proposed by Straub et al., using ORB descriptors, manages to relocate above 90% of all the frames up to a distance of 12 m, with an average error of 40 cm from ground truth.

With average errors ranging from 9.4 cm for MH02 to 25.1 cm for MH03, the proposed system yield better estimates than that of the strategy proposed by Straub et al. but with a considerably lower estimation percentage. This is partly due to the test sequences as some frames in the EuRoC sequences contain no visual overlap with previously seen and mapped frames. When instead considering recall given a $t_{inside}$ the results are more comparable. With
a $\tau_{\text{inside}} = 0.9$ (that is 90% of a keyframe’s points can be reprojected inside a test frame) and a $d_{\text{max}} = 0.5$ recall range from roughly 30% (MH02, lowest recall) to 70% (MH03, highest recall). The recall is even higher for MH03 with increased keyframe insertion frequency, at approximately 85% with $\tau_{\text{inside}} = 0.9$.

Another evaluation is done by Glocker et al. [35] using the “7 Scenes” dataset, specifically testing relocalization capabilities of their system. Poses are proposed by their system using one of various methods, from which poses are estimated and validated using ICP. For the evaluation, pose are accepted as correct if it, according to ground truth, is within 2 cm and the rotational difference is less than 2 degrees. With these thresholds, their best proposed method manages to relocate in between 37.0% and 85.8% (with an average of 67.1%) of all frames. Large overlap do however exists between the training and test data for this dataset, which makes relocalization for every frame feasible.

Comparably, using $d_{\text{max}} = 2$ cm yields a recall of between 10% and 30% (see Appendix A.2) when 90% of points can be reprojected from the one of the three nearest keyframes to the current frame (although that keyframe is not necessarily selected by FAB-MAP). Given that attempt is not made to relocate for every frame unless we are certain enough we’ve found a matching keyframe, while worse, the result is promising. Considerably higher recall is obtained if allowing $d_{\text{max}}$ to be slightly higher.

More recently and less similarly Kendall and Cipolla [46] used deep learning using convolutional neural networks to perform relocalization. Based on PoseNet [47] the relocalization engine directly estimates the pose from a frame. The method was evaluated on two datasets, one moving in smaller space than EuRoC and one using a larger space, “7 Scenes” [35] and “Cambridge Landmarks” [47] respectively. On the 7 Scenes dataset the method achieves a 0.47 m average median error over the test sequence. This is a larger error than the proposed system, but poses are on the other hand given for each of the frames. The Cambridge Landmarks dataset has an average median relocalization error of 1.96 m, with spatial extents ranging from 35 by 25 meters to 500 by 100 meters. The obvious downside of this strategy is that the net must be pre-trained with true poses for the environment where relocalization may occur, rendering it only selectively applicable.

While not conclusive nor comprehensive, these comparisons suggest that the system performs at least close to on par with other existing systems. Given the differences in method for all evaluations compared with, deciding an overall best method is not possible. It is certain that the proposed system is capable of detecting loops and relocate, and depending on the requirements, to a fairly exact degree. Granted, additional verification is likely needed to ensure no bad estimates get through, such as exhibited with large estimated movements.

5.2 Method

This section discusses the method employed throughout this thesis, touching on scenarios evaluations fail to cover, potential problems with the methodologies, etc.

Initial evaluations

Using precision and recall as performance metrics for descriptors is commonly done, works using it includes [33], [18], [10], [22], and [44]. Definitions of how true and false positives and negatives are defined do however vary. [33] for example lets each point (“landmark”) which to feature match be represented by a cluster of descriptors extracted from the point from several images under different degree of the characteristic evaluated (viewpoint, illumination, etc). The method employed here is more akin to [33] and [18]. Both these papers define correct correspondences based on overlap between key point radii of points in the pair of images tested. This radius is not necessarily the same between two observations of the same point and was thus scrapped in favor of a simple maximum distance threshold instead. This is guaranteed
5.2. Method

Figure 5.3: AKAZE key points extracted from an “RGB-D SLAM dataset” frame. Original frame licensed under Creative Commons 4.0 Attribution License (CC BY 4.0). The key points are in most cases fairly well separated. The circles have a radius of 3.0 pixels.

to be the same regardless of whether a descriptor making use of key point radius to scale the extraction area or not.

There are however details to note with this evaluation. Firstly, AKAZE key points was used to allow AKAZE to be used in the evaluation. Ideally random points should be used on the reference frame (“keyframe”), which then can be reprojected using ground truth poses used as key points for the test frame. The choice of using AKAZE is not unmerited however, as it is usually among the top performing key point detectors, although not unanimously.

Reprojection of points and using these as key points is not possible (per OpenCV) with the AKAZE descriptor. It is not reliable either with a non-synthetic dataset as both the poses and depth map are subject to noise – meaning descriptors might be extracted from the “wrong” position due to the noise and not match due to this fact. The same noise also applies to the reprojection performed in order to find correct matches.

A tolerance radii of the reprojected point was used to mitigate this effect, but this in turn means that there is higher risk a single point in the reference frame causes multiple points in the test to be flagged as relevant, due to it (when reprojected) being within the radius of two or more points. This is however also the case with defining a match based on circle overlap.

Luckily AKAZE key points exhibit a slight ANMS-like behaviour and tend to be fairly well separated, and was another major reason as to why using AKAZE was deemed acceptable (see Fig. 5.3 for an example). ANMS could have been ran before extracting descriptors, but was at the time of the evaluation phase not implemented.

The illumination evaluation has correspondences known a priori and hence does not exhibit this problem.

The pose estimation evaluation is harder to assess. It performs a step which in the system has to be done and as such is very indicative of what kind of relocation capabilities can be expected out of the system. There are however more parameters to consider with this kind of evaluation and the results should by no means be deemed conclusive, and instead more in support or contradiction to the main evaluation with precision-recall curves.

The “Iterative” OpenCV RANSAC method was used to perform the pose estimation, and it is thus indicative of the performance of it when a NNDR scheme is used for descriptor matching. NNDR is used in the actual system for relocalization, but this is mainly due to the
threshold value at which a descriptor pair are accepted as being from the same point is less dependent on the choice of descriptor for NNDR.

The increased number of parameters means an exhaustive evaluation is not practical. RANSAC is also a non-deterministic algorithm, and is not guaranteed to give the same solution every time. Multiple attempts was not used per frame as the main evaluation criteria was not primarily the quality of the pose, instead it was if a pose of enough inliers could be found at all given the correspondences.

Pose estimation algorithms were evaluated in the same way, only this time keeping the descriptor fixed and instead using different pose estimation algorithms. The same caveats exist for this evaluation, but slightly exaggerated by the fact that no other evaluation of them was performed.

The main purpose of the pose estimation evaluation is for this study to gauge from how far a pose can be recovered using different methods. Results gained are meant to guide where further effort can yield best results or are most needed. The limited evaluation was for this end deemed sufficient, but this means that the results are not necessarily general, e.g. applicable to how well odometry can be performed using the algorithms.

Final evaluations

Odometry evaluation using the absolute trajectory error (ATE) was carried out to measure the quality of estimated trajectories – following the benchmark method suggested by Sturm et al. They also propose an additional metric, relative pose error (RPE), more appropriately used to evaluate odometry when no algorithm to ensure global consistency is used (i.e. closing loops). RPE evaluates “drift per n frames” and is thus well suited for evaluating the quality of a system more independently of the length of the sequence. The reason this metric was not used is due to the lack of systems evaluated using this metric on EuRoC, only (and only on MH04) and , neither published in a journal and only available in pre-print. As they might thus not be peer-reviewed these were ignored for comparison in favor of publications.

Sturm et al. however do note that both the ATE and the RPE metrics are strongly correlated . Weighing in both these factors (lack of systems evaluating RPE for EuRoC and the correlation) and ATE was deemed suitable as the metric for gauging the quality of the odometry.

As mentioned in Ch. Related Research, the most common way loop detection is evaluated is after optimizations have been done. RMSE of the trajectories can then be used as a measure of the performance. Closing loops is not done here (see Sec. Related Research), and this evaluation metric (ATE) is thus not applicable for evaluating the quality of loop detections. Evaluation of how well the system can detect a previously observed place thus had to be done differently.

Detecting such a place is of interest for both loop detection and relocalization, and the evaluation was focused on the latter. A system becoming lost is often seen as a reason for disqualification , which for evaluation purposes makes sense but on the other hand omits the real world application where the ability to recover from failure is needed. Ideally, a system never gets lost – but when and if it does, it needs to be handled. Another reason for testing relocalization is that it also tests how well the relative pose between the two frames is estimated, something often useful when closing a loop with e.g. pose graph optimization .

A novel methodology for this evaluation was thus conceived. Given the novelty, care was taken to ensure that the metric had a clear connection to intuitive concepts. Recall is defined as “given a fraction $\tau_{\text{inside}}$ of points being reprojectable from a keyframe to the current frame, what is the percentage of frames that yield correct pose estimates”. “Correct” is defined based on the error in relative distance between a frame and a keyframe, using a maximum distance ($d_{\text{max}}$). This connection to intuitive concepts is crucial for being able to draw any conclusions with confidence from the results.

There are however a number of caveats with this method, most importantly the vector $\mathbf{t}_{\text{est,diff}} = \mathbf{t}_{\text{est, keyframe}} - \mathbf{t}_{\text{est, frame}}$ not correctly being aligned when using $\mathbf{t}_{\text{est,align}} = S \mathbf{t}_{\text{est,diff}}$. $S$
5.3 The work in a wider context

is calculated as the rotation that best fits the data to the model (ground truth) in the least squares sense over all poses. This transformation is not correct for every pose unless the data is a rotated version of the model. This in turn means that the relative vector can still be slightly wrong due to the rotation and thus yield larger errors with vectors of greater magnitude.

While not perfect this was deemed as the best compromise as aligning the estimated movement vector and the ground truth movement vector (thus considering the error as the difference in movement) creates another problem. The error in this case would completely ignore the directional component of the error, something deemed not acceptable. Thus the error metric was defined based on the (aligned) estimated and ground truth poses.

Another way to mitigate this effect could have been using two metrics, one for the difference in distance (difference in the vector magnitudes) and one using the vector angle, e.g. cosine similarity,

\[ \frac{t_{\text{truth, diff}} \cdot t_{\text{estimate, diff}}}{\|t_{\text{truth, diff}}\| \|t_{\text{estimate, diff}}\|} \]

between aligned estimate and truth vectors or indeed the angle difference directly. This does not remove the issue with the rotation being slightly wrong due to the least square fitting of rotation, but penalizes smaller and larger translations equally. A single metric for the error between ground truth and estimate is however arguably more intuitive.

Methodologies of similar nature to the one employed here have been used, for example the method used in [34], where the percentage of frames where a successful relocalization occurs is the metric. They use a fixed allowed distance from ground truth of 2 cm for deciding a relocalization as successful or not (equivalent to \(d_{\text{max}} = 2 \text{ cm}\)) and a maximum angle difference of 2 degrees. Relocalizations are considered possible for all frames which is reasonable in their case as the overlap between training and test data is large (the 7 Scenes dataset).

The differences from the method in [34] is thus not only using one fixed \(d_{\text{max}}\) and defining frames as relevant based on reprojection instead of assuming relocalization is possible for them all. This assumption of relocalization being possible for every frame could have been used in this study too, but this would have meant the results are more dependent on the sequence used. This in turn would subsequently make comparisons between sequences harder – a decrease in the reliability of the results – and even more so when comparing the results to other systems, potentially using other datasets.

Moteki et al. uses an evaluation method similar to [34] in [60], but without using a maximum allowed error and thus “successfully” estimates a pose for every frame. They instead use the difference to ground truth pose as the error metric. A caveat is that they use the same synthetic sequence for both training and test, and is hence not comparable to other evaluations.

5.3 The work in a wider context

As with nearly all fields within science and technology, they collectively and in isolation can have a profound impact on society. This, to different degrees, is as true for chemistry (e.g. medicine versus chemical weapons) as it is for computer science (e.g. environmental impact \([77]\) versus aiding people \([62]\)). This is equally true for the sub-discipline computer vision, where the applications range from augmented reality games and autonomous vehicles to large scale surveillance. Wherever you personally stand on either of these topics, it is clear they can have a clear societal impact.

This work in particular is within visual SLAM, with applications such as aiding inspection as done in \([48]\) to automating tasks using drones, either for civilian or military use. The former example uses an underwater vehicle that runs a SLAM application to aid inspection of large ships. An example of the latter is “MedizDroids” \([3]\), a project that aims to use small drones and SLAM to control mosquitoes (vectors for e.g. malaria). The same principle could of course

\[\text{Rotate two vectors, } a \text{ and } b, \text{ by the same angle } \alpha, \text{ if } \|a\|_2 < \|b\|_2 \text{ holds, then } \|a - a_{\text{rotated}}\|_2 < \|b - b_{\text{rotated}}\|_2 \text{ will always be true unless } \alpha \mod 2\pi \equiv 0.\]
also be used in military application, e.g. indoor recon in war zones without putting human lives directly at risk.

SLAM applications aside, tracking and loop detection in particular are most prevalently of interest in applications where autonomous behavior is desired, especially when GPS is not precise enough or not available (e.g. tunnels, underwater). Use cases include autonomous cars and other vehicles. Widespread use of autonomous cars has the potential to impacts the society to a large degree, conjectured to reduce (potentially fatal) vehicle accidents by at least 40% \cite{23}. Automatisation is also of interest commercially, for example in warehouses \cite{93}, where robots can replace humans – promising an increase in efficiency at the expense of human jobs.
This thesis proposes a system for odometry such that loop detection and relocalization can be performed. Estimated trajectories was evaluated for three different long sequences from the dataset EuRoC. The proposed system was found to perform about on par with systems evaluated on the same dataset that do not close loops. A system performing loop closing performed far better than the rest of the systems, showcasing the benefit of being able to detect and subsequently close loops. ATE RMSE of the proposed system range between 0.80 m and 1.69 m on these three sequences.

Additionally the proposed system’s capability to relocate was evaluated using a novel evaluation scheme with the intent to allow intuitive interpretation of the results. This evaluation scheme can be used with datasets not explicitly designed for relocalization evaluation as long as they feature one or more loops.

This evaluation showed that the system is capable of relocating with a pose error less than 10 cm in 23% to 73% of cases, depending on test sequence and settings. The system manages to find a pose in 37% to 85% of all frames when a keyframe exists that can reproject 90% of its points onto the frame.

The result of both the odometry and relocalization evaluation answer the question “How well does the proposed system perform odometry and relocalization?”

The proposed system makes use of FAB-MAP to detect loops, with pose estimation made using 2D-3D correspondences and RANSAC using P3P as the minimal sample set kernel. Both the current frame and previous frames are then refined using local bundle adjustment using the inliers found by RANSAC.

The effect of different keyframe insertion frequencies was investigated in more detail. The most valuable conclusion drawn from this investigation is that the frequency can significantly impact performance, both beneficial and adversive depending on to what degree keyframes has been added in near proximity. Too many keyframes and risk perceptual aliasing, too few and relocalization will be successful less frequently. An adaptive keyframe insertion scheme is proposed to be used in systems where perceptual aliasing can become an issue, such as when using FAB-MAP. Although the question “How often should keyframes be inserted to provide sufficiently many for localization, yet not impose penalties?” is not answered explicitly, the answer to the follow up question “And what are these penalties?” is thus answered for the proposed system.
In addition to presenting the proposed system, two aspects of descriptor performance were looked at in detail - one focused on invariance to viewpoint and one focused on illumination invariance. SIFT was found to perform best. SIFT is slow to compute, which for real time applications meant that BRISK, ORB, or FREAK was found to be most performing in relation to their computational cost. These results augment already existing studies, which should all be taken into account when answering “What feature descriptor should be used that can both be accurate enough and be fast enough to compute?” The true answer depends on the use case, for this particular application BRISK was deemed most appropriate.

The descriptor evaluation was followed by a brief evaluation of mainly OpenCV pose estimation methods, focusing on estimation using 2D-3D or 3D-3D correspondences together with RANSAC. The 2D-3D methods with kernel size of “3 + 1” was found to perform best relative to their computation time, i.e. AP3P and P3P in OpenCV. 3 + 1 refers to the fact that three points are used to estimate the pose, with one point used to select among solutions if multiple exists. Umeyama’s algorithm as the RANSAC kernel yielded most successful estimates, but is prohibitively slow to compute. AP3P and P3P were fastest to compute, while still yielding results with a greater number of inliers compared to all other 2D-3D methods tested.

The question “What pose estimation method should be used that achieves a balance between accuracy and computation time?”, like the descriptor related question, also has an application dependent answer. None of the algorithms are perfect, which led to the implementation of another algorithm for odometry that calculates the pose in two steps. This algorithm does not fully work for relocalization, where using something more advanced is likely beneficial as well. An explicit answer is thus not given to the question, but the results provide data to guide the selection of algorithm for other systems.

6.1 Future work

This thesis opens up a certain number of new questions that are of interest to investigate.

Firstly, seeing as a novel method was used for assessing relocalization quality. Evaluating other systems using this methodology is desirable, as it provides a better picture of this capability of already proposed systems, a detail which is otherwise often glossed over in research.

Secondly, already from the beginning the idea was to eventually fuse this work with the result of another thesis where loop closing was examined in detail. This is a step that remains to be done, and should when implemented provide far better trajectories if the trajectories feature detectable loops.

And third, there are a number of additions to the system found during evaluation that could increase performance that are worth evaluating. Most chiefly this is a more adaptive keyframe insertion scheme and more advanced pose estimation for relocalization, for example based on EVSAC [27]. Another thing to investigate is making use of a more appropriate prior for FAB-MAP, based on the topological map that is implicitly built in the system.
Bibliography


A.1 Additional Results

Some additional results are given here. Due to the quantity, not all results are given as it would yield a far too long appendix.

Descriptor evaluation - Illumination

All of the additional results for one test scene are presented here. A note is however in order for the simpler test cases, e.g. “+1 Overexposed” and “-1 Underexposed”, AUC values for these sequences is a fairly poor approximation as they were so simple that recall never reached very close to zero. This in turn means that the trapezoid integration does not integrate over the whole range of values leading to a smaller AUC. This is partly why these are not present in the thesis. A small mitigation strategy is used, manually inserting 0 precision and 0 recall when calculating the AUC as this, by definition, is always achieved with $\tau = 0$ (or rather, a precision value of undefined due to $0/0$). This is however still not perfect.
A.1. Additional Results

Figure A.1: Precision-recall curve for the test image “+1 Overexposed”.

Figure A.2: Precision-recall curve for the test image “+2 Overexposed”.

Figure A.3: Precision-recall curve for the test image “+4 Overexposed”.

Figure A.4: Precision-recall curve for the test image “-1 Underexposed”.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AKAZE</td>
<td>0.94</td>
</tr>
<tr>
<td>BRIEF</td>
<td>0.76</td>
</tr>
<tr>
<td>BRISK</td>
<td>0.98</td>
</tr>
<tr>
<td>BoostDesc</td>
<td>0.94</td>
</tr>
<tr>
<td>DAISY</td>
<td>1.00</td>
</tr>
<tr>
<td>FREAK</td>
<td>0.97</td>
</tr>
<tr>
<td>ORB</td>
<td>0.92</td>
</tr>
<tr>
<td>SIFT</td>
<td>0.97</td>
</tr>
<tr>
<td>SURF</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Recall [%] 94 95 96 97 98 99 100

Precision [%]
AKAZE, AUC=0.94
BRIEF, AUC=0.76
BRISK, AUC=0.98
BoostDesc, AUC=0.94
DAISY, AUC=1.00
FREAK, AUC=0.97
ORB, AUC=0.92
SIFT, AUC=0.97
SURF, AUC=0.99
A.1. Additional Results

Figure A.5: Precision-recall curve for the test image “-2 Underexposed”.

Figure A.6: Precision-recall curve for the test image “-3 Underexposed”.

Figure A.7: Precision-recall curve for the test image “Directional + Uniform”.

Figure A.8: Precision-recall curve for the test image “Directional + 0.8 Uniform”.
A.2 Varying keyframe insertion frequency

This section presents corresponding graphs of errors and recall given fractional threshold when varying the keyframe insertion frequency.

Figure A.9: Precision-recall curve for the test image “Directional + 0.6 Uniform”.

Figure A.10: Precision-recall curve for the test image “Directional + 0.2 Uniform”.

A.2 Varying keyframe insertion frequency

This section presents corresponding graphs of errors and recall given fractional threshold when varying the keyframe insertion frequency.
A.2. Varying keyframe insertion frequency

Figure A.11: Distance error between ground truth and estimate translations between a matched keyframe and test frame. The sequence is MH03 and the setting is “More”. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is $\approx 38\%$.

Figure A.12: Distance error between ground truth and estimate translations between a matched keyframe and test frame. The sequence is MH03 and the setting is “Most”. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is $\approx 35\%$.

Figure A.13: Distance error between ground truth and estimate translations between a matched keyframe and test frame. The sequence is MH01 and the setting is “Most”. Below is the distances estimated between the two frames plotted against the error. The percentage of estimates with error less than 10 cm is $\approx 73\%$. 
A.2. Varying keyframe insertion frequency

Figure A.14: The recall for various threshold values, given different $d_{max}$. The sequence is MH03 and the setting is “More”.

Figure A.15: The recall for various threshold values, given different $d_{max}$. The sequence is MH03 and the setting is “Most”.

Figure A.16: The recall for various threshold values, given different $d_{max}$. The sequence is MH01 and the setting is “Most”.