Artificial Intelligence Group — Robotics

Master Thesis

Implementation and Evaluation of Image Sequence Based Place Recognition Utilizing a Humanoid Robot

Benjamin Aschenbrenner
Matr. 4292264

Supervisor: Prof. Dr. Raúl Rojas
Assisting Supervisor: Prof. Dr. Manfred Hild

Institute of Computer Science, Freie Universität Berlin, Germany
January 30, 2015

I hereby declare to have written this thesis on my own. I have used no other literature and resources than the ones referenced. All text passages that are literal or logical copies from other publications have been marked accordingly. All figures and pictures have been created by me or their sources are referenced accordingly. This thesis has not been submitted in the same or a similar version to any other examination board.

Berlin, January 30, 2015

(Benjamin Aschenbrenner)

Contents

1. Introduction **2**
1.1. Motivation . . . 2

2. Related Work **4**

3. Approach **6**
3.1. Contribution . . . 6
3.2. Underlying Theory . . . 7
3.2.1. Mean Absolute Difference . . . 7
3.2.2. Tangent Distance . . . 9
3.2.3. OpenSeqSLAM . . . 15

4. Offline Implementation **21**
4.1. DreamViewer - Software Requirements . . . 21
4.2. Software Architecture and used Libraries . . . 22
4.3. Experiments and Results . . . 26
4.3.1. Experiments using Nordland dataset . . . 31
4.4. Offline Implementation Experimental Results . . . 36
4.4.1. Performance observations and discussion . . . 46

5. Online Implementation **50**
5.1. Motivation and Reasons for another Algorithm . . . 50
5.2. Myon . . . 51
5.3. Algorithm . . . 53
5.3.1. Description of the Localization Algorithm . . . 57
5.4. Experiments and Results . . . 63
5.5. Results of Heuristic Online Algorithm Experiments . . . 63
5.5.1. Conclusion and Result Evaluation . . . 77
5.6. Using saved Sensory Data and Active Movements . . . 78

6. Conclusions **80**
6.1. Future Work . . . 80
6.2. Conclusion . . . 81

A. Komische Oper Image Sequences **85**

B. Experimental Results **89**

Abstract

In the domain of image based localization, alternatives to image feature based algorithms have been developed making use of similarity metrics that operate directly on pixel intensity values. These approaches can work robustly and efficiently in cases when the recognition process needs to deal for example with varying lighting conditions or changes in scenery detail. In addition to employing such a pixel based similarity metric a recent algorithm called *OpenSeqSLAM* processes image sequences instead of single images to improve the recognition. However *OpenSeqSLAM* is not robust in cases of perspective change. This makes it problematic to use in many robotic applications when the camera perspective is not fixed to certain positions with fixed orientations. In this contribution an approach is developed and evaluated that aims to mitigate the effect of perspective change on recognition performance by combining the concept of *OpenSeqSLAM* with an alternative similarity metric called tangent distance. It was further analyzed if the algorithm can be suitably designed to run on a humanoid robotic platform and how it can utilize the robots capabilities. To enable evaluation a test application called *DreamViewer* for image sequence based localization algorithms has been developed. In result of the first tests an adapted and heuristic version of the algorithm was developed and evaluated as well. This algorithm aims to run on hardware performance constraint robotic embedded systems. Developed algorithms were tested offline with recorded image data as well as online on a humanoid robot platform called *Myon*. Results indicated that the developed algorithms using tangent distance can perform superior in terms of recognition performance compared to the standard *OpenSeqSLAM* algorithm in the tested cases of perspective change.

1. Introduction

It is fair to assume that one of the first and most pressing questions that will come to mind of a person regaining consciousness in a different place than the person last remembers to be in will be: *Where am I?* Furthermore this person will probably start looking around in the hope of recognizing some parts of the place he is in. Evidently it is very important to us to get an idea of where we are, using this information as a basis for planning our next steps or getting memories into context. The above mentioned question and associated behaviour are central aspects of this thesis, its main focus is to explore and implement an image sequence based place recognition approach utilizing a humanoid robot. The used robotic platform is the modular *Myon* humanoid robot, developed in the *Neurorobotics Research Laboratory* (NRL), but the general approach can be applied to other platforms as well and is not tailored to the specific hardware.

In image based recognition and localization many implementations rely on using so called *feature* based techniques. In general these approaches first detect features in an image, like for example an intersection area of two edges. Afterwards these image areas are numerically transformed into the final feature descriptors, that are stored. These descriptors are designed to be robust against possible image transformations, so that for example the same image scene can be recognized despite a change in perspective or contrast. These approaches have proven to work well in a number of use cases, especially when detailed high resolution input images are used, that are nowadays easily obtainable using low-cost hardware. However some drawbacks, still exist:

- detector needs to be tuned to image scene characteristics (e.g. indoor vs. outdoor)
- possible failure due to change in lighting (e.g. day and night)
- might not tolerate changed details in scene (e.g. removed objects)

It is therefore not just an academic exercise to evaluate other non feature based localization methods. Such approaches will be evaluated in this thesis. Instead on extracting features the main underlying idea is to determine similarity of images based on a pixel based difference. This concept will be briefly introduced in the following part.

1.1. Motivation

In this contribution a localization algorithm is developed operating on the basis of image sequences and aimed to run on a humanoid robotic system. As mentioned above this thesis is focused on algorithms that directly operate on pixel based intensity values of images without any further extraction of features. The concept of such image sequence based localization algorithms alone are not a novelty and related research will be presented in the following section. However, these kind of state-of-the-art algorithms often fail to recognize places when the input images, that should be recognized as a known location, show the place from a different perspective than in the original image that should be recognized. As later explained this problem even occurs when the perspective change is small. The key reason for that is that one component of these algorithms, the image similarity metric, is not well adjusted to compensate perspective changes. In other words such similarity metrics are not perspective change invariant. Since this contribution is aimed to enable a robotic system to recognize a location by using image sequence based localization it would be necessary to make the approach more perspective change invariant. For example it should still be possible for the robot to recognize a room when the robot's input camera images are tilted by nine degrees compared to the images stored in memory. A similarity metric that aims to achieve such properties is the *tangent distance* and for this contribution it was attempted to combine it with a present image sequence based localization algorithm. Further it was necessary to evaluate the algorithms regarding their expected performance when running online on a robotic system and also considering constraints regarding reactiveness and hardware resources.

In addition, the present algorithms do not make any use of abilities that humanoid robotic systems potentially offer. For example active movements of the robot can be used to achieve a different perspective and thereby improving the recognition performance. That aspect will be considered as well in this contribution.

2. Related Work

Even when leaving aside place recognition approaches that rely on non image based visual data like laser scanner readings, still plenty of research has accumulated in recent years. Therefore in this section only a selection of topics that appear to be the most closely related to the subject of image sequence based localization is brought into context. Image sequence based localization can be categorized as a subproblem of visual localization. Visual localization approaches use two dimensional image data as input or sometimes three dimensional shape data. It is possible to divide this category further into feature based and non feature based approaches. Feature based approaches make use of the input images by applying image feature extractor and descriptor techniques like *SIFT* and *SURF* on them. A recent example for this is given by the approach of Andreasson et al. in [ADL08]. It will be explained in greater detail later, but in order to recognize a locally obtained image in a set of stored images an algorithm needs to determine how similar two images are. This is referred to as similarity metric and the discriminating aspect of the vision based localization algorithms. Andreasson et al. present a Simultaneous Localization And Mapping (SLAM) algorithm that determines the similarity between input images and stored images by making use of *SIFT*. More closely related to the approach presented in this thesis however are localization approaches that work directly on the image pixel values. They are also sometimes referred to as *holistic* approaches. A biologically inspired approach is OpenRatSLAM presented by Milford et al. in [MJCW13] and [MW10]. This approach is modeled around neurological structures in the mammalian brain called *grid cells*. These cells play an important role in the spatial orientation for example of the rat. OpenRatSLAM uses an image similarity metric called Sum of Absolute Differences (SAD) and this similarity metric is closely related to the metric used in the OpenSeqSLAM algorithm presented by Milford et al. in [MW12]. OpenSeqSLAM provides a basis for the algorithms developed in this contribution. It makes use of image sequences to recognize a location. Milford et al. further investigate OpenSeqSLAM's localization performance for various types of downsampled input images in order to evaluate how much visual data is actually required to perform reliable localization. They present their results in [Mil13] and experiment for example with different image resolutions and intensity ranges. Major features of these tests are reflected in the experiments performed in this contribution. The similarity metric used and evaluated for the developed place recognition algorithms is the tangent distance. This metric was developed by Simard et al. and they evaluated it in the context of numeric digit classification tasks in [SLCDV00]. The tangent distance is a similarity metric that aims to be transformation invariant for a set of given transformations. Other metrics with a similar aim were developed. For example the *Min-warping* method designed to provide illumination robustness. It is developed and evaluated by Möller et al. in

[MHF14]. Another example is the *Image Euclidean Distance* (IMED), which enhances the euclidean distance to make it more robust against pixel pertubations. This method was developed by Wang et al. in [WZF05]. They also compare this metric to the tangent distance and find the latter to be superior in terms of recognition performance, but see the challenge to integrate it efficiently into image recognition algorithms.

3. Approach

In this chapter the basic contributions and the general theory of the employed approaches are presented. First in section 3.1 aspects specifically addressed by this thesis are summarized. References to the general structure of this thesis are presented there also. In section 3.2 theoretical concepts, that are key to the contributed algorithms are presented.

3.1. Contribution

In this section main contributions that this thesis aims to achieve are presented. They can be structured in different parts and listed sequentially according to the order they were approached. This is done in the following enumeration and although most listed topics will be presented in later following chapters they are listed here to give a complete overview. Each of them will be shortly addressed in this section.

1. Image Sequence Localization Algorithms
   - Tangent Distance
   - Mean Absolute Difference
2. Data Collection (online and offline)
3. First Tests and Implementation Verification
   - Testing Application Development *DreamViewer*
   - Algorithm Verification with Online Data
4. First Experiments and Evaluation
5. Development and Implementation of adapted Online Algorithm
6. Online Algorithm Experimental Verification on Myon
7. Final Experiments and Evaluation of all Algorithms
8. Assessment of Future Modifications

For the reasons presented in the motivation section one of the main goals is to combine the tangent distance similarity metric with an image sequence based localization algorithm. The first step is therefore to implement the localization algorithms along with the tangent distance and mean absolute difference similarity metrics. Main aspects of that are presented in chapter 3. Once that is achieved first tests can be performed with the similarity metrics and the localization algorithms. In order to do that first test data is needed, so the second step is to collect data. This can be done using directly the target robotic system to record image sequences. To obtain larger image sets existing offline datasets were gathered in addition. The third step is then to test and verify the implemented algorithms. For that purpose it is desirable to be able to select input data in a comprehensible and flexible way for the localization algorithms and also to simply view input image sequences. Therefore an application called *DreamViewer* was developed to enable that. The word *Dream* was used because recorded and saved image sequences are similar to the experience of human dreaming, that can be in general described as a series of images as well. This is described in chapter 4. Following that, the fourth contribution is the setting up and performing of experiments that allow evaluation and first comparisons between the algorithms. On the basis of these first results it became obvious that the development of an algorithm more suitable for the target platform was necessary. This is listed as the fifth aspect and described in chapter 5. The sixth part summarizes the validation and first test of this algorithm that was implemented on the target robotic platform. Following that experiments similar to those of the fourth listed aspect were performed. All developed algorithms are compared there.

In context of the last listed aspect future modifications and assumptions about the online algorithm are assessed. Important theoretical concepts that are the basis and therefore preconditions for the developed contributions are described in the following section.

3.2. Underlying Theory

In this section theoretic concepts that are fundamental to the developed localization algorithms are summarized. Starting with descriptions of related image similarity metrics the section is concluded with the presentation of the later modified localization algorithm OpenSeqSLAM. In the following descriptions the term localization is used. It means that a location, for example defined by x-y coordinates on a map, is obtained. Synonymous to the term location is the position. In the context of this contribution localization is performed by recognizing a place by using input image sequences and comparing those to stored images. Each stored image is associated with a position, so localization is possible in combination with correct recognition. In addition to that, images that are recorded at a certain location can be associated with the orientation of the camera at the moment the image is shot. This is referred to as a pose, consisting of position and camera orientation and makes is possible to distinguish different perspectives obtained at the same location.

3.2.1. Mean Absolute Difference

The *Mean Absolute Difference* (MAD) is a metric indicating similarity of two same sized grayscale images A and B. This metric is detailed here because it plays a major role in the OpenSeqSLAM localization algorithm described in section 3.2.3. This metric operates in the spatial image domain, meaning it is working directly on pixel intensity values. It simply sums up the absolute differences for corresponding pixels in images A and B and divides it by the total number of pixels. This is defined as:

$$MAD = \frac{1}{xy} \sum_{r=1}^{y} \sum_{c=1}^{x} |A_{rc} - B_{rc}| \tag{3.1}$$

Where in equation 3.1 x is the number of pixels in horizontal direction and y is the number of pixels in vertical direction for both same sized images A and B. The notation A_{rc} represents for the grayscale image A the intensity of the pixel in row r and column c. The question might be asked why the MAD metric has been chosen for OpenSeqSLAM instead of other simple intensity based metric like the similar *Mean Squared Difference* (MSD) metric defined as:

$$MSD = \frac{1}{xy} \sum_{r=1}^{y} \sum_{c=1}^{x} (A_{rc} - B_{rc})^2 \tag{3.2}$$

It can easily be observed that the MAD metric does not promote outliers, meaning larger differences in intensity values between A and B as much as the MSD metric does because of the square. Especially in the image based localization use cases this is of advantage because often two compared images depict the same location but in a slightly altered way. Alteration can be caused for example by a combination of change in perspective, changed light, change of objects in the scene, noise and other things. In any case the difference metric should be able to tolerate a certain amount of these alterations or at least not exaggerate them like the MSD might do, as is argued by Brock et al. in [Bro13, Chapter 16].

As mentioned before the MAD however also does not tolerate a wide range of image transformations. The problem can be easily illustrated which is done in in figure 3.1.

There, two similar images are shown and the only difference between them is a vertical translation by one pixel. Notwithstanding that seemingly small difference a pixel based image difference like the MAD, or even worse the MSD, would be large because the dark pixels of the left and the right image are subtracted with largely unequal intensity values represented by white pixels. If there would be a way to transform at least one of the images before the pixel difference is calculated the result would be more close to the expected value that indicates great similarity. This problem has been addressed in OpenSeqSLAM in a limited manner by modifying the MAD to tolerate a certain degree of translation in horizontal direction. This is described by Milford et al. in [Mil13] by defining the MAD as follows:

$$MAD = \min_{\Delta x \in \sigma} g(\Delta x, A, B) \tag{3.3}$$

$$g(\Delta x, A, B) = \frac{1}{xy} \sum_{r=1}^{y} \sum_{c=1}^{x} |A_{r(c+\Delta x)} - B_{rc}| \tag{3.4}$$

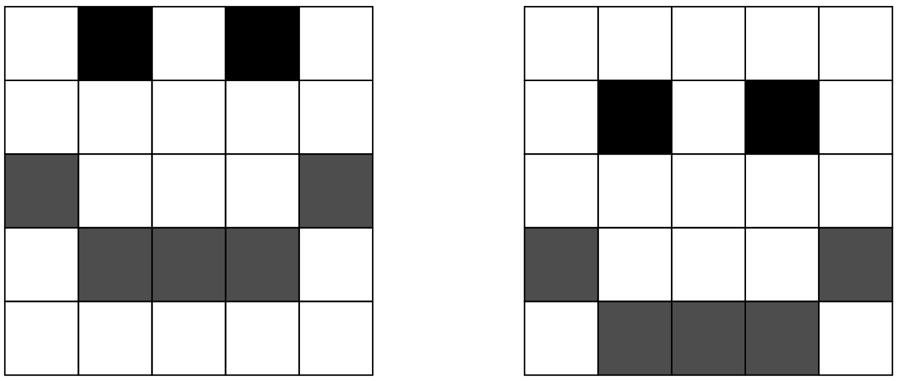


Figure 3.1.: A simple corresponding pixel based image difference between the left image and a vertically translated version of itself, represented by the right image, would result in a large pixel based difference because all non white pixels in the left and right image are compared to a white pixel instead of a dark one.

In equation 3.3 the MAD is calculated for different offsets in horizontal direction and then the minimum is selected as a result. The range of offsets that is considered can be controlled by the parameter σ. The intent for this definition is described in [Mil13] as a way to increase OpenSeqSLAM's performance regarding camera yaw movements and horizontal offsets.

3.2.2. Tangent Distance

It is shown in section 3.2.1 the MAD pixel difference based similarity metric is not invariant to most image transformations that are related to perspective changes like rotation or scaling. As described in section 1.1 this thesis will evaluate a different image similarity metric, the tangent distance. This metric is defined so that its results are more invariant in the presence of some anticipated image transformations. In the following it will be described how this metric is defined and how it performs compared to the MAD and other distance metrics.

The concept of tangent distance was first introduced by Simard et al. in [SLCDV00] in the context of pattern recognition. The application context that the authors evaluate in this paper is the problem of handwritten digit recognition. This means that a classification algorithm has to decide for an input image which number out of the 10 possible is depicted in it. Since the input images are handwritten digits the vast majority of them was subject to a range of transformations. In result these images differ significantly on a pixel basis from a single representative of each digit class. Transformations that frequently happen in this context are for example small rotations, variations in stroke thickness, scaling and translations. Pixel based similarity metrics like the MAD, described in section 3.2.1, or the similar euclidean distance are not invariant to these transformations. In the context of pattern recognition this often leads to wrong classifications. The authors describe that by using the tangent distance in the case of handwritten digit recognition they outperform most other approaches that are used in this context. Since the tangent distance performs well in the case of transformation invariant digit recognition it is worth evaluating if it can be used for transformation invariant place recognition or localization, a key question of this thesis.

If given an image A of a location, we can define a set of transformations that when applied to A can be interpreted as a change in perspective. An example of one possible transformation is the rotation operation. This transformation occurs for example when the vision system of a robot is tilted. A rotation operation can be described by a single parameter, the rotation angle α. Further we can describe the set S_{rot} containing all rotated versions of the image A as:

$$S_{rot} = \{x | \exists \alpha : x = rotation(A, \alpha)\} \tag{3.5}$$

Where $rotation(A, \alpha)$ is A rotated by angle α and $A = rotation(A, 0)$. Each element of S_{rot} can be understood as a vector with a dimension equal to the number of pixels in A.

$$dim(A) = A_{width} \times A_{height} = A_{pixels} \tag{3.6}$$

If we assume for example A to be 25 pixels in width and 20 pixel in height then A and each transformed image in S_{rot} can be interpreted as a vector of dimension 500.

The set S_{rot} is a manifold in this 500 dimensional image space and since it depends only on the single parameter α it has dimension 1. Of course definitions analogous to the definition of S_{rot} can be made for other image transformations as well including non linear transformations like image translations. In reality we can expect a combination of transformations to happen. For example transformations that can be considered closely resembling a change of perspective are rotation by angle α_1, vertical translation in number of pixels α_2, horizontal translation in number of pixels α_3 and image scaling by factor α_4. Now again we can define a set S_A that includes all images around A that are transformed by these four transformations:

$$S_A = \{x | \exists \vec{a_i} : x = t(A, \vec{a_i}) \tag{3.7}$$

$$\vec{a_i} = \begin{bmatrix} \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \end{bmatrix} \tag{3.8}$$

The four transformation parameters are now grouped into the vector $\vec{a_i}$ and $t(A, \vec{a_i})$ performs the mentioned image transformations on A according to the entries in $\vec{a_i}$. The manifold resulting from this has dimension 4. If we consider a second image B and wish to determine how similar it is to A when only the mentioned four transformations can be applied to both of the images. What needs to be done to get an exact solution is to find the minimum distance between the manifolds S_A and S_B. Where S_B is the manifold containing all transformed versions of B analog to the definition of S_A. However since non linear image transformations are involved and therefore S_A and S_B are not linear subspaces of the high-dimensional image space it is complicated to analytically determine this distance. The general idea of the tangent distance is therefore to find the tangential planes that approximate the manifolds around A and B and then to determine the minimal distance between these linear subspaces. The concept is also visualized in figure 3.2. There the two manifolds S_A coloured in red and S_B coloured in green stretch around the two images A and B. Although it is difficult to convey the fact in a figure, it should be noted again that all the elements in S_A and S_B are of high dimension and therefore the shape of S_A and S_B can be highly complex. Also in the figure there are three different distances highlighted. One of them is the tangent distance, marked by a solid line that connects the two lines tangential to the points A and B. These lines represent the linear approximations of the manifolds around the original images A and B and the tangent distance is the shortest distance between these approximations. In the following descriptions this linear approximation is referred to as tangent plane. Marked by a dotted line that directly connects A and B is the euclidean distance of the untransformed images. Since A and B are in the tangent plane it can be seen that the euclidean distance can not be smaller than the tangent distance. In the upper right corner, marked by a dashed line is the actual distance between S_A and S_B.

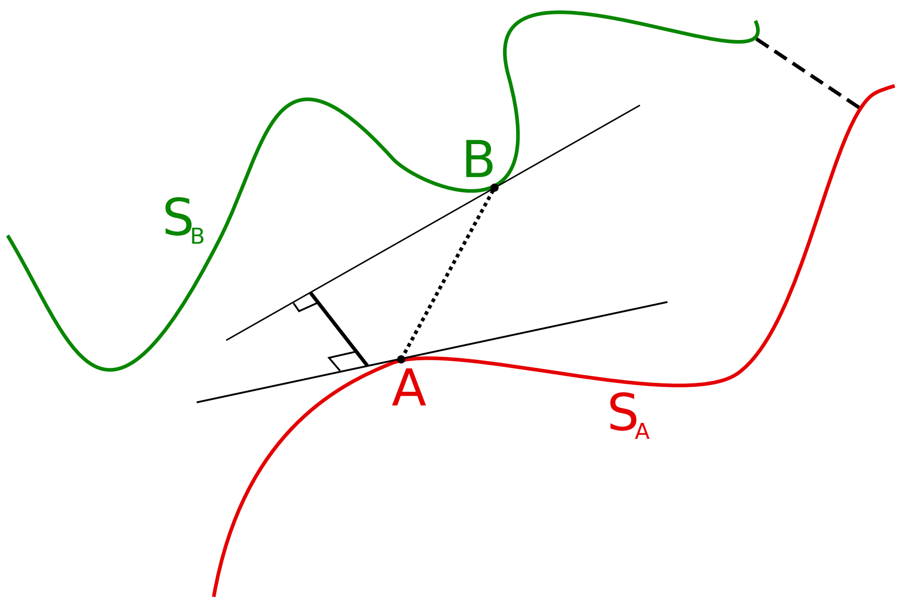


Figure 3.2.: Visualization of transformation manifolds S_A and S_B around two images A, B and distances. The euclidean distance between A and B is marked by a dotted line directly connecting the images. The solid line connecting the tangent planes is the tangent distance and represents the shortest distance between the tangent planes around A and B. The shortest distance between S_A and S_B is represented by a dashed line.

To obtain the tangent distance the following steps can now be summarized:

1. linearize and obtain tangent planes
2. find approximations on tangent planes that minimize distance
3. get euclidean distance between minimizing approximations

In equation 3.7 the manifolds are defined by the function $t(A, \vec{a_i})$. To get the tangent plane this function is linearized by performing a first order Taylor expansion centered at the input image A, that is $t(A, 0)$. This linearization is expressed as:

$$t(A, \vec{a}) = t(A, 0) + \frac{\partial t(A, \vec{a})}{\partial \vec{a}} + H.O.T. \approx A + T\vec{a} \tag{3.9}$$

$$T = \frac{\partial t(A, \vec{a})}{\partial \vec{a}}\Big|_{\vec{a}=0} \tag{3.10}$$

It is known that the full Taylor expansion involves an infinite summation of terms where each term includes a higher order derivative of the approximated function. Since in case of the tangent distance the aim is to gain only a linear approximation that is the tangential subspace, the higher order terms ($H.O.T.$) are omitted which leaves $A + T\vec{a}$. The matrix T contains now the tangent vectors, a basis of the linear subspace, which is the tangent plane. Tangent vectors are the first order partial derivatives of $t(A, \vec{a})$ evaluated at $\vec{a} = 0$. Since in this thesis we considered four different transformations, T is defined as:

$$T = \frac{\partial t(A, \vec{a})}{\partial \vec{a}}\Big|_{\vec{a}=0} \tag{3.11}$$

$$= \left[\frac{\partial t(A, \vec{a})}{\partial a_1}, \frac{\partial t(A, \vec{a})}{\partial a_2}, \frac{\partial t(A, \vec{a})}{\partial a_3}, \frac{\partial t(A, \vec{a})}{\partial a_4}\right]_{\vec{a}=0} \tag{3.12}$$

Each element in T is a tangent vector and it will be now shown how the tangent vectors are obtained. To give a demonstrative example a sample image is used from which some tangent vectors are derived. This is illustrated in figure 3.3. Considering there first just the left column of images, the top image depicts the original input image, a grayscale version of the flag of Sweden. From this input image the two tangent vectors for horizontal and vertical translation are obtained. The partial derivative of the horizontal translation operation of the image can be understood as an image in that each pixel value represents the value of pixel intensity change in a horizontal direction of the original image at this pixel position. This is shown in the middle row image and it can be seen there that change in intensity occurs for the flag of Sweden only at the borders of the vertical stripe. Similar the vertical translation operation is gained and the only difference is that vertical contrasts are considered as can be seen in the bottom image. Naturally $t(A, \vec{a})$ has to be derivable at least once but digital images are technically not continuous because of the discrete intensities. As can be seen also in the example flag of Sweden tangent vector images that this can be problematic, because the contrast changes at the borders of the stripes are very narrow and the resulting tangent vectors operational range is very narrow itself and not continuous. The tangent vectors are used, as described hereafter, in a weighted summation to form the linear approximation of a transformed image. In this regard it should be noted that only those tangent vector image parts can have an effect on the weighted summation that are non zero. In the example tangent vector images therefore the only range of translation transformations that can be approximated is within the range of the narrow bright lines visible at the borders of the stripes and the rest is zero. Because of these problems Simard et al. present the tangent vector creation in combination with a Gaussian blur operation on the image. Results of the input image convolved with a Gaussian blur kernel are shown in the right column of images in the example figure. The resulting tangent vectors, shown below the blurred input image, are in result wider in range of operation and intensity change between contrasting intensity regions is mitigated. Ways to obtain the horizontal and vertical image derivatives as well as the Gaussian blur are well known and are in detail described for example by Gonzalez and Woods in [GW10, Chapter 3]. As the authors also point out blurring reduces the original structural image details but can also be beneficial when noise is present so it should be always considered how much the images should be blurred. In the example two tangent vectors types have been presented. For other transformations like image scaling and rotation it is described by Simard et al. in [SLCDV00] how they are created. All of them are formed by a combination of the horizontal and vertical image derivatives.

The tangent vectors are also called lie operators so each entry in T containing a single tangent vector is labeled L_i and filled into a matrix L, so that in column i of L the lie operator L_i is found. In case of image A the resulting matrix is labeled L_A and each column is formed by one tangent vector. This is shown in equation 3.13.

$$L_A = [L_1|L_2|L_3|L_4] \tag{3.13}$$

In this thesis four types of image transformations are considered, the linear rotation and scaling transformation in combination with the non linear translation transformations in vertical and horizontal direction. Without restricting general validity we can say that L_1 shall be the lie operator of the translation operation in horizontal direction, L_2 the lie operator of translation in vertical direction, L_3 the lie operator of rotation transformation and L_4 the lie operator of the scaling transformation.

Each element of the tangential plane can be expressed as a linear combination of the lie operators. This linear combination is solely defined by the scalar values in $\vec{a}$ so that the equation of the tangential plane becomes:

$$\begin{aligned} A'(\vec{a}) &= A + L_A\vec{a} && (3.14)\\ &= A + a_1L_1 + a_2L_2 + a_3L_3 + a_4L_4 && (3.15) \end{aligned}$$

From equation 3.14 it can be seen how the linearization of the four transformation operations works. In case for example we imagine an image B that is a highly rotated and

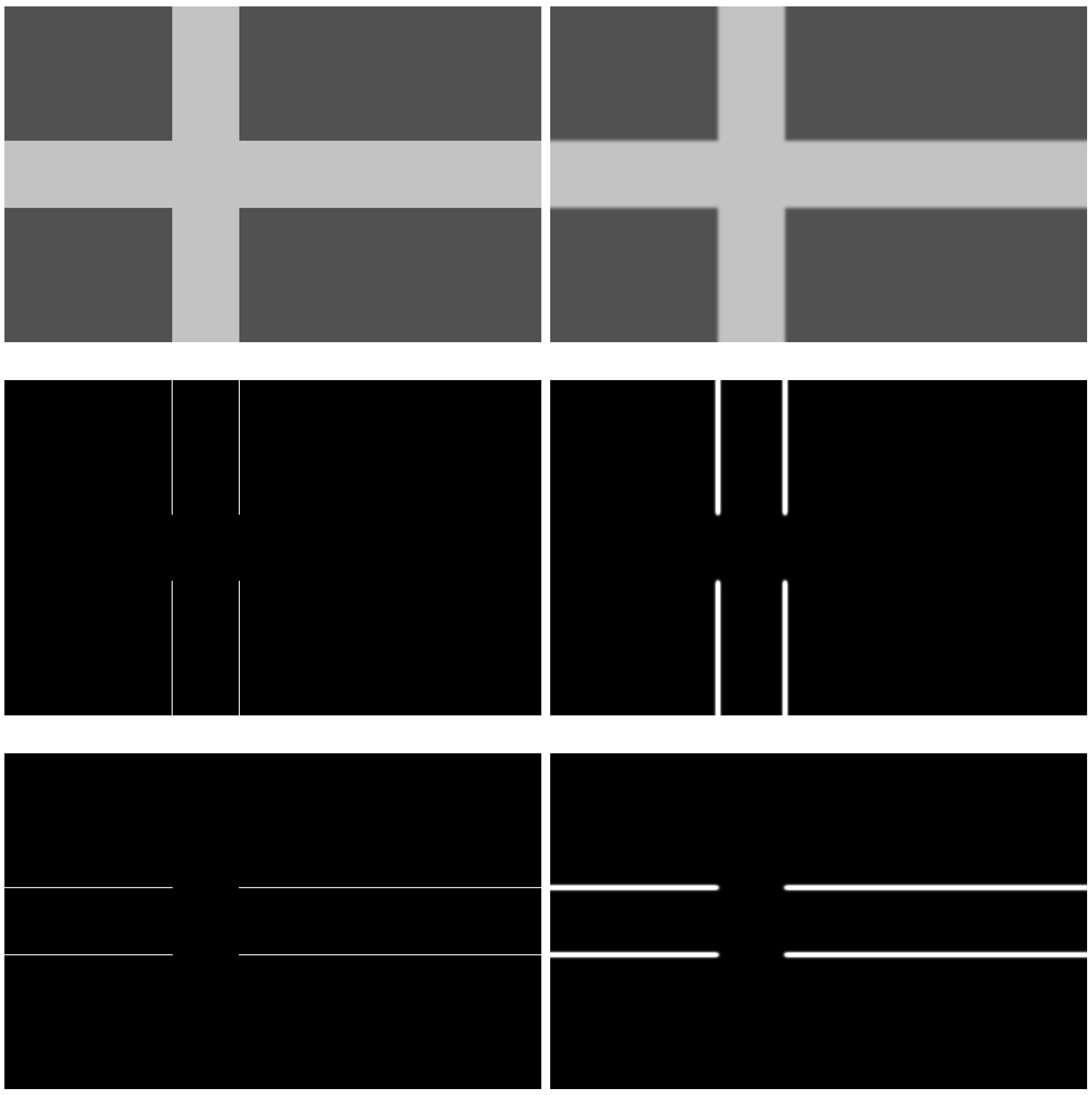


Figure 3.3.: A grayscale image of the Flag of Sweden and its horizontal and vertical derivatives original (top left) and after applied Gaussian blurring (top right). Below these are the corresponding tangent vectors for horizontal (middle row) and vertical translation (bottom row).

horizontally translated version of the original image A. Then the linear approximations of B would reflect these transformations by larger values a_3 and a_4 in the linear combination since these factors influence the rotation and translation lie operators.

Given the formal definition of the tangent planes it is now possible to define the tangent distance between two images A and B. Let us assume the tangent planes of images A and B, following the definition of equation 3.14, are $A'(\vec{a})$ and $B'(\vec{b})$. The tangent distance TD can now be defined by the minimizing expression:

$$TD(A,B) = \min_{\vec{a},\vec{b}} ||A'(\vec{a}) - B'(\vec{b})||_2 \tag{3.16}$$

So in both tangent planes approximations with the smallest euclidean distance needs to be found. In [SLCDV00] Simard et al. achieve this by defining the normed expression as a function d that depends on the two parameters $\vec{a}$ and $\vec{b}$. The idea is then to setup an equation where the partial derivatives of this function d are set equal to zero because this is the condition for which a minimum is expected. More details on the derivation are given in the named paper but in result the solution to the minimization problem is given as:

$$\vec{a} = (L_{AA} - L_{AB}L_{BB}^{-1}L_{BA})^{-1}(L_{AB}L_{BB}^{-1}L_B^T - L_A^T)(A-B) \tag{3.17}$$

$$\vec{b} = (L_{BA}L_{AA}^{-1}L_{AB} - L_{BB})^{-1}(L_{BA}L_{AA}^{-1}L_A^T - L_B^T)(A-B) \tag{3.18}$$

With $L_{AA} = L_A^T L_A$, $L_{BA} = L_B^T L_A$, $L_{AB} = L_A^T L_B$ and $L_{BB} = L_B^T L_B$.
This is the form that has also been used for the implementation of the tangent distance in the developed localization algorithms that will be described in the following sections.

3.2.3. OpenSeqSLAM

In this section the image sequence based simultaneous localization and mapping (SLAM) algorithm OpenSeqSLAM will be analyzed. This algorithm provides the general concept for the algorithms created in this thesis. OpenSeqSLAM was introduced by Milford and Wyeth in [MW12] and is described also in [Mil13].

As mentioned above OpenSeqSLAM uses image sequences to attempt a localization. The images the algorithm operates on are grayscale images. This means that for each pixel in the used images exactly one intensity value is available. In the context of the algorithm two types of image sequences are distinguished, these are:

- database sequence
- localization sequence

In the database sequence all images are stored that will be compared to the locally acquired input images. So in the robotic context the database sequence can be understood as the visual memory of the robot. A localization sequence is a sequence of locally acquired images and that is the input for the localization algorithm. In the case of OpenSeqSLAM for this input there is then a sub-sequence of equal length in the database to be found that is the most similar to all regarded subsequences in the database sequence. Later in this section it is explained what kind of metric is used in the context of OpenSeqSLAM to determine the similarity.

An important attribute of both the database sequence and also the localization sequence is that each element of them is semantically linked to its surrounding elements. To show what that exactly means the two sequences will be defined more formally. Lets assume the database to be labeled M with elements m_i and the localization sequence labeled L containing elements l_j where i and j mark the position of these elements in each sequence. Then a total order $<$ for both sets can be defined using a *was memorised before relation*. So that $m < m'$ for $m, m' \in M$ if and only if m was memorised before m'. An equal definition can be given for elements of L by $l < l'$ for $l, l' \in L$ if and only if l was memorised before l'. The order of the elements in each sequence is then defined by:

$$\forall m_i, m_j \in M \wedge m_i < m_j \Leftrightarrow i < j \tag{3.19}$$

$$\forall l_i, l_j \in L \wedge l_i < l_j \Leftrightarrow i < j \tag{3.20}$$

This order is obtained when images are recorded successively and added to the sequence by simply appending each newly acquired image to the end while moving through an arbitrary scenery. From this structure two characteristics become obvious. One is that there is a temporal link between neighbouring images because neighbouring images where taken successively through time and it can be said that the temporal link between neighbouring images m_i and m_{i+1} becomes stronger the shorter the time difference between successive recordings is. Because of the latter there is also a semantic link regarding the location where neighbouring images were memorised. This also means that there is a possibility that two neighbouring images m_i and m_{i+1} show at least partly the same location and this possibility increases if there is a strong temporal link between them. But this of course also depends on the movement speed through the scenery.

The main data structure that the OpenSeqSLAM algorithm operates on is called image difference matrix, labeled D. This matrix consists of entries that encode the similarity of all possible image pairs that can be formed by taking one image of the database sequence and another of the localization sequence. In sections 3.2.2 and 3.2.1 different pixel based metrics have been shown to determine image similarity and as mentioned in the listed section OpenSeqSLAM relies on the mean absolute difference. So for each pair the mean absolute difference is calculated and entered into the matrix.

As also shown in the figure 3.4 the results of the pairwise image similarity comparison are organized in the following way. Each row i in the matrix corresponds to similarity comparisons using image m_i of the database sequence M. In direction of a column j the similarity comparisons are calculated using image l_j of the localization sequence L. So it follows that the element d_{ij} of D is the result of image similarity calculation between m_i and l_i. In case of the OpenSeqSLAM that is the mean absolute difference $MAD(m_i, l_i)$. Since the MAD metric is actually a metric that is monotonically rising with pixel difference for the two compared images the resulting entry d_{ij} is inversely related to similarity. To summarize a small value d_{ij} corresponds to a low difference and therefore to greater similarity, whereas a large value marks the opposite. In both figures the values of entries in D are colour-coded, so that a light gray corresponds to

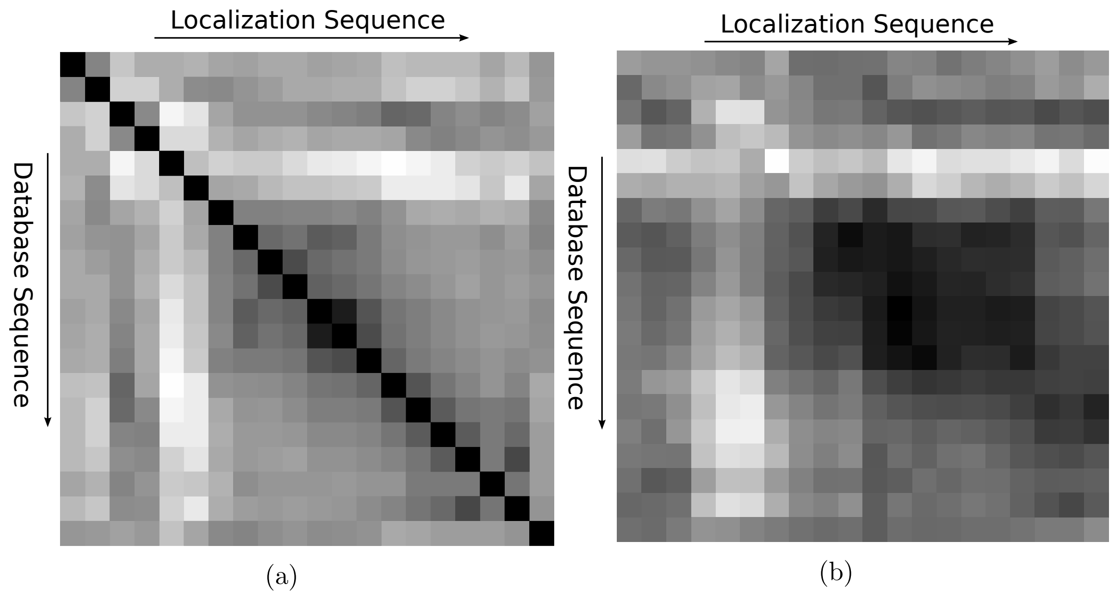


Figure 3.4.: Figure 3.4a shows the difference matrix for a case of identical database and localization sequences using MAD. Entries on the diagonal have maximum similarity because identical images are compared. Figure 3.4b is a difference matrix for the more typical case where the database sequence is not equal to the localization sequence.

a large difference value and therefore low similarity. Darker shades of gray correspond to low difference and greater similarity. In figure 3.4 two examples of image difference matrices are given. Figure 3.4a shows the result of creating the image difference matrix with identical database and localization sequences. Typical for this case is the zero valued diagonal where the similarity of equal images is computed. The right image, figure 3.4b resulted from computing the matrix using two differing sequences, which is the more usual case. It should be noted that as mentioned in the previous section the similarity metric of OpenSeqSLAM is not invariant regarding image transformations that are related to perspective changes. In the contribution of this thesis an alternative metric, is integrated into OpenSeqSLAM and analyzed. This metric is the tangent distance and it is used instead of the MAD. However the following continued description of how OpenSeqSLAM operates after creation of the image difference matrix is indifferent to the used similarity metric. A general description of what the algorithm does next on the basis of the image difference matrix is to look for stretches of large similarity. However before the image difference matrix is traversed there is a preprocessing step applied to its entries. The authors of [Mil13], Mildford et al., call it image difference matrix normalization. They found that using this processing step for localization sequences that contain more than one element, a better localization performance is achieved. Their argument in favour is that this normalization partly can negate bias effects like changes in lighting. It is performed on each element in D using a number of surrounding elements in the same column. The definition is given by:

$$\hat{d_{ij}} = \frac{d_{ij} - \overline{d_k}}{max(\sigma_d, \sigma_{min})} \tag{3.21}$$

Where $\hat{d_{ij}}$ is the normalized difference value that is entered at position i, j in D. Value $\overline{d_k}$ is the mean calculated over a range of k entries around d_{ij} in the column j. The nominator is divided by the standard deviation σ_d of the k elements. To avoid devision by zero a σ_{min} is defined and used for the devision in case σ_d gets close to zero. In figure 3.5 the resulting effect can be observed, there the right figure shows the normalized version of the left image difference matrix. It can be observed that after normalization for some image regions the contrast appears to be enhanced. However as Milford et al. also note, for single image based localization, meaning using a localization sequence of length one, the normalization process would be counterproductive. The reason for that is that in such a case the image difference matrix would be reduced to a single column and the best guess for a localization match one could make is to look for the minimal element in that column. Then normalization is simply unnecessary and could obscure the true minimum.

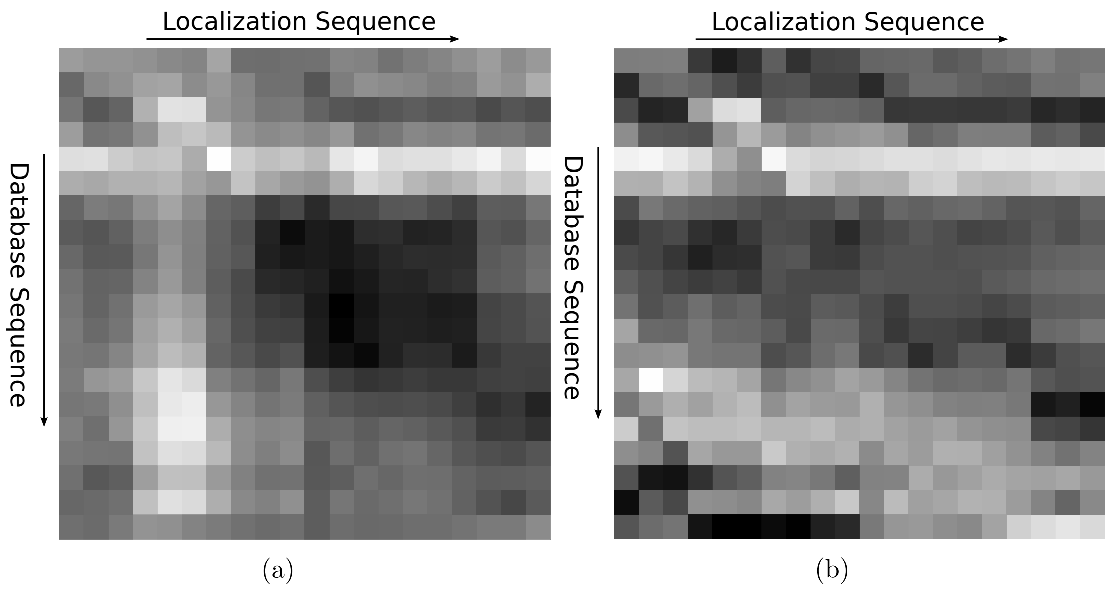


Figure 3.5.: The two figures visualize the effect of the difference matrix normalization. The computed image difference matrix before normalization is shown in figure 3.5a. Figure 3.5b represents the normalized version of this image difference matrix according to equation 3.21.

After building the image difference matrix and performing the step of normalization the OpenSeqSLAM algorithm proceeds with searching for so called minimizing sub-routes within the image difference matrix. These sub-routes can be generally described as connected regions of low difference in the image difference matrix, that resemble line-like shapes. A most obvious form of such a sub-route can be observed in the before mentioned figure 3.4a on the diagonal of the image difference matrix. For that example OpenSeqSLAM should find the slope traversing the diagonal as being the most minimizing one and return its point of origin in the left upper corner. Minimizing in this context means that the average of all image difference matrix entries that are traversed by the slope is smaller than the sum of other possible slopes. The process of finding the minimal slope is exemplified in figure 3.6.

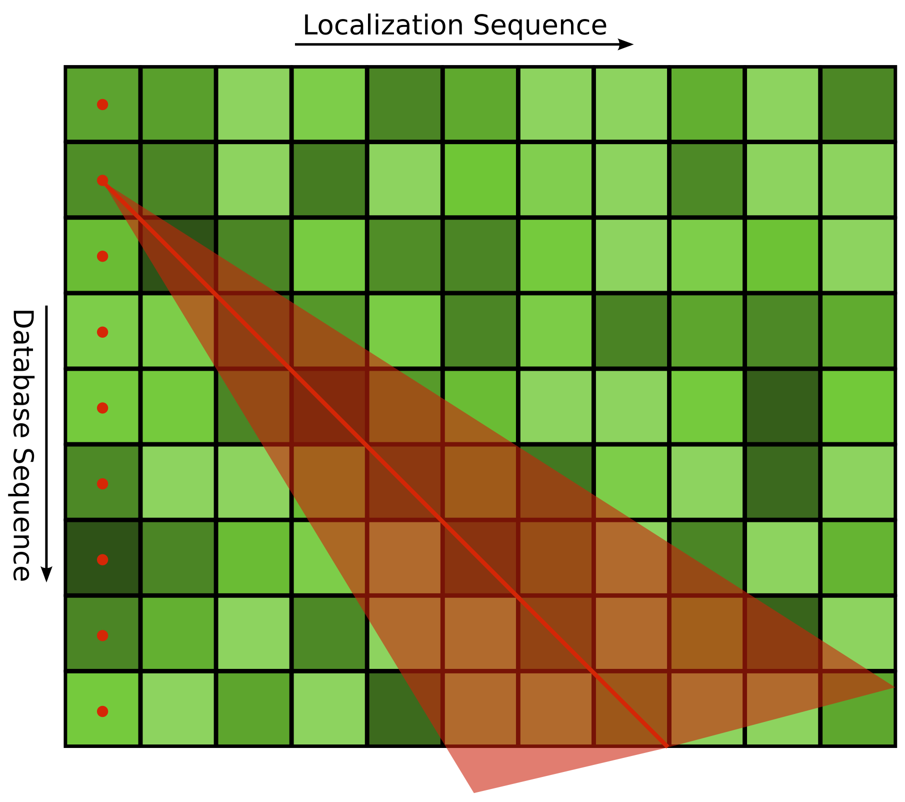


Figure 3.6.: The figure shows an image difference matrix with highlighted minimal slope search. For each starting search position in the left column, marked by a red dot, a range of slopes is traversed. For the first position this is highlighted by the semi-transparent red area. Shades of green represent image similarity values, the darker the color, the more similar are the two compared images.

There an arbitrary image difference matrix is shown. The search for the minimal slope always starts at an element in the left most column. In the example image this is emphasized by red dots. Starting from these points a number of possible sub-routes originating there are followed through the matrix as shown in the example by the solid red line and red semi-transparent area. The semi-transparent red area symbolizes possible other sub-routes that will be traversed for other slopes. For each matrix entry that is traversed during a sub-route traversal its difference value is added to an accumulative value that is initialized with zero in the beginning. This value is called sub-route score. Since there is a range of possible slopes for each starting point there is an equal number of scores for each sub-route starting point. When all scores are calculated for one starting point the minimal score of them is selected and saved. After this has been done of all starting points the one with the smallest score is chosen. This is the candidate that is finally returned by the algorithm as best matching memory to the input sequence.

The described process returns a database index d_{min} and can be defined as follows:

$$d_{\mathrm{min}} = \arg \min_{1 \leq i \leq m} s(i) \tag{3.22}$$

Where $s(i)$ is the minimum score over the range of k slope possibilities at starting position i in the database. The minimizing expression in equation 3.23 gives the definition of $s(i)$ as:

$$s(i) = \min \hat{d_i} \tag{3.23}$$

Where $\hat{d_i}$ contains all the k scores generated for each slope starting at i. Each element in $\hat{d_i}$ is obtained by summing up the elements traversed for the range of possible slopes k. The possible variety of slopes can be constrained so that each sub-route contains the same number of elements, this would be typically be the number of elements in the localization sequence, which is equal to the number of columns in the image difference matrix. If the number of elements is not the same for the sub-routes a normalization step is required, so that the final accumulated score is divided by the number of elements in the sub-route. However the implementation of the algorithm contributed by this thesis uses equal length sub-routes and therefore score normalization was not needed. This concludes the fundamental description of the OpenSeqSLAM algorithm. In the following chapter this basis will be used to describe the realization of the evaluation software for the OpenSeqSLAM and modified version using the tangent distance.

4. Offline Implementation

As mentioned in the first sections, one goal of this thesis was to explore the properties of a localization algorithm that combines OpenSeqSLAM with a more transformation invariant image similarity metric, the tangent distance. Although it would have been possible to simply implement the standard OpenSeqSLAM algorithm as well as to design and implement the modified OpenSeqSLAM using tangent distance this approach would not have been very flexible regarding evaluation, understanding and testing. Instead a GUI application software was realized that embeds the localization algorithm implementations and enables the user to interactively choose and view input sequences that are passed then as inputs to the algorithms. This software was named *DreamViewer*. The application name should reflect partly the intended data structures on which the localization algorithms should run in the context of Myon. What is referred to are *dream sequences*. These are the actual data structures that were designed for the Myon to hold, in addition to other data, the image sequences. The focus in this chapter is on the image sequences, what kind of other information and what purpose they might serve is discussed in a later chapter. In the following sections first the software is described regarding its architecture and features. After that, in section 4.3, some experiments and results are presented that also include tests using the DreamViewer software. In the last section of this chapter conclusions are drawn, gained from these tests. It should be noted that the primary question that is addressed in this chapter is: Can the tangent distance be beneficial to the image sequence based localization process under the influence of perspective changes? However there was first no constraint given regarding memory usage and processing time. These algorithmic attributes are of course very important especially considering the algorithm should perform on a robotic embedded system. So these aspects and how they were addressed is described in a separate chapter, following this one.

4.1. DreamViewer - Software Requirements

The aim of the *DreamViewer* application was to make it possible to combine the implementation of the image sequence based localization algorithms with a simple GUI that enables the user to select inputs for the algorithms in an intuitive and flexible way. Since the primary data on which the algorithms that are evaluated in this chapter operate are sequences of images the main requirement was to visualize these sequences and select sub-sequences. An important aspect to deal with was, that these image sequences can be represented in different ways. For example as mentioned in the introduction to this chapter there is the representation in the form of *dream sequences*, that is the used format for image sequences that are recorded and used on the Myon. Since the image data is embedded together with other information in a binary format, it became necessary to build a parser to extract the images saved in this format. In addition to the *dream sequences* format it is desirable to use and test the algorithms with arbitrary collections of saved images interpreted as image sequences, so DreamViewer was built to support these as well. In addition to that the software was intended to perform with reasonable performance, since the memory and computational demands were anticipated to be high for larger image sequences and greater image resolution. The requirements can finally be summarized as follows:

- flexible testing of image sequence localization algorithms
- possible to select sequence parts
- visualize results and selections
- modular and easily extensible
- reasonable performance
- able to be build and run on multiple platforms

4.2. Software Architecture and used Libraries

In this section a brief overview on how the DreamViewer software is structured from the viewpoint of software architecture is presented. Furthermore, reasons will be given why certain software libraries and frameworks were chosen. The DreamViewer software is structured in very modular way, so that the functionality of the localization algorithms is as isolated as possible from the GUI modules. This has been done to implement a maximum reusability of the modules and makes it more simple to extend and maintain the software. Because of the requirements summarized in the previous section 4.1 the OpenCV image processing library and the cross platform GUI framework Qt have been chosen for this implementation. OpenCV and Qt are natively available in the C++ programming language. Another reason why C++ has been chosen for the implementation is its object oriented capabilities and interoperability with the C language. This was of advantage because the recorded *dream sequences* were obtained on the Myon with a C implementation in plain C style structures. In the following paragraphs the libraries that the implementation relied on are briefly presented.

OpenCV

OpenCV is a BSD licensed open source computer vision library. It is implemented with the aim to be computationally efficient and utilizes a wide range of hardware acceleration capabilities. These include support of CPU SIMD operations like SSE4 and NEON instruction sets as well as parallel computation usage via OpenCL on GPUs and multi-core CPUs. Since DreamViewer was aimed to be runnable on not just one platform another positive aspect is that OpenCV builds are available for Linux, Android, Windows and iOS. The library is available, among other programming languages, in C++, so it could be used without any obstacles for this implementation. The used API version was 2.4.8. Naturally, OpenCV was used to process the images but was used isolated from the GUI functionality. A more detailed introduction and overview of OpenCV is presented by Pulli et al. in [PBKE12] and also well presented [1] and documented [2] on the OpenCV website. .

Qt

Qt is an open source application programming framework including GUI creation functionality. It offers cross platform capabilities so that the same Qt implementation can be used without modification on Linux/X11, Android, Mac OS, iOS, Windows, WindowsCE and Blackberry. As mentioned before, it is natively available in C++ and offered for the named reasons a good choice for the visualization. For this implementation the Qt version 5.3.3 has been used. More detailed information about this version and Qt in general can be obtained from the Qt website [3]. Although Qt offers a vast range of features only a very small part of them has been used for this implementation.

In figure 4.1, an overview of software components in connection with DreamViewer is given. In the figure the GUI module, labeled DreamViewer, is separated from the localization algorithm implementations and the image similarity metrics module. Only the GUI is dependant on the Qt libraries. The image similarity metrics module contains the implementations of the tangent distance, mean absolute difference and the euclidean distance. This module is used by the implemented localization algorithms, that are the OpenSeqSLAM and the OpenSeqSLAM with tangent distance. Further it can be seen that the DreamViewer implementation relies on the implemented localization algorithms. All implemented modules depend at least to a small degree on the OpenCV module, because the data type that is used for common image data access between the modules is the OpenCV *Mat* matrix type. This type is also used for the main processing calculations in the image similarity and localization algorithms modules.

Since all used and implemented software components are open source every step of execution can be traced and understood.

In the following paragraph the main features of DreamViewer will be shown by presenting a typical use case scenario. In figure 4.2 a screenshot of the running DreamViewer application is shown. As mentioned before the main input data the evaluated algorithms operate on are image sequences. Therefore the first step when using DreamViewer would be to load such a sequence. DreamViewer at the moment supports three different image sequence types. These are:

1. binary format dream sequence

[1]`http://opencv.org`
[2]`http://docs.opencv.org`
[3]`https://qt-project.org`

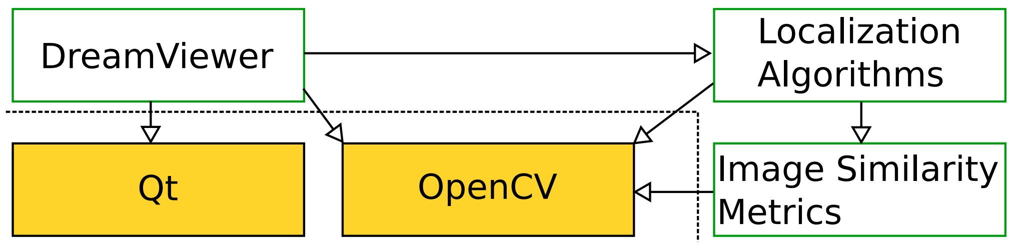
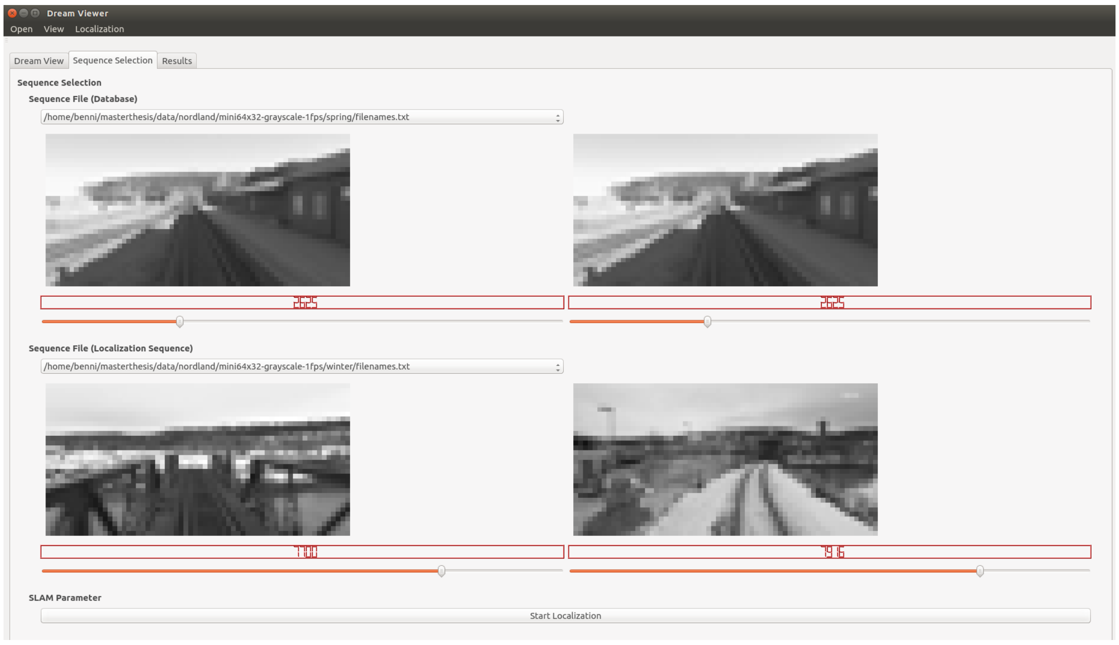


Figure 4.1.: DreamViewer application and associated modules. Dependencies are indicated by arrows. The dotted line separates software components developed in this contribution from used libraries and frameworks. The former are shown by rectangles with green borders and the latter by rectangles with a yellow background.

2. binary format reduced dream sequence
3. arbitrary image collections

As mentioned before in this chapter, the dream sequence type is used for image sequence recordings on the Myon robot. It is a binary format and not only contains plain image data but also additional data like for example various sensory outputs. However in its current implementation state the DreamViewer application only parses the grayscale image data parts in the dream sequence. Although the implemented localization algorithms only use one channel grayscale images the dream sequence format holds the full color images format which might be useful for future localization algorithms or are just more favourable to the user when viewing the sequence. All details of the dream sequence datatype can be examined in the C header file containing the structs definitions. This header file is provided along with the rest of the code for this thesis. The second type, reduced dream sequence, is similar to the first one regarding that it is also used for saving data on the Myon and is binary, but it is a thinned out and trimmed version of the first sequence type. For example it does not hold the full color image data but instead single channel 8-Bit grayscale image data. This was done to decrease memory usage and to lower the writing to memory time when recording the sequence. The additional information are largely reduced to sensory data that holds orientation information of the robot head, that is the visual center of the Myon. Again the precise definition is given in the C-struct definition and DreamViewer currently only uses the visual data part. Lastly the third data sequence type are arbitrary collections of single images. These are passed to DreamViewer by referencing an index file. This index file contains all paths to the images that should be used for this sequence. These paths are stored as human readable strings and there is one image path defined per line. The order of the paths defines the position of the image in the resulting sequence, so that the path in the first line points to the image that will be first in the sequence, the second line to the second image in the sequence and so on.

After loading a few image sequences of interest the user can review them and select parts of the sequences as input for the localization algorithm. In figure 4.2 a view



Sequence Selection

Sequence File (Database)
/home/benni/masterthesis/data/nordland/mini64x32-grayscale-1fps/spring/filenames.txt
2625 2625

Sequence File (Localization Sequence)
/home/benni/masterthesis/data/nordland/mini64x32-grayscale-1fps/winter/filenames.txt
7700 7916

Figure 4.2.: Sequence Selection window of the DreamViewer application.

of the selection process is given. Two sequences are selected, these are the database sequence and the localization sequence. The database sequence is chosen by a drop down menu in the upper part of the application window. This menu lists all image sequences that have been loaded so far. Similar in function there is a second drop down menu, by using it the localization sequence is chosen. After both sequences are selected four image display areas are visible. Two are grouped with the upper drop down menu of the database sequence and another two with the localization sequence drop down menu. These display areas show single frames of the sequences selected by the drop down menus and each one is connected to a selection slider positioned below them. They are intended to select a sub image sequence that is then used as input for the localization algorithms. This is done for the localization sequence by positioning the left bottom slider to the start frame and the right bottom slider to the end frame. The sequence number of the currently selected image is displayed in a red LCD-style numeric indicator between the image and the slider. In the example figure the sub sequence chosen as localization sequence input starts at frame 7700 and ends at frame 7916. Naturally the sliders can only be moved to form reasonable selections, so that the start sequence number is always less or equal to the associated end sequence number. After a selection has been made the localization algorithms can be executed with the given input by pressing the *Start Localization* button. The implemented algorithms use a multi-threaded approach to perform the image difference matrix calculations and the search in the image difference matrix so that multiple CPU cores are used. This has the advantage of reduced processing time for larger image sequences or higher image resolutions. However more working memory is consumed during processing. In many cases multiple sub localization sequences of a larger localization sequence are tested in DreamViewer, but it would be desirable to avoid for each of these sub-sequences the time consuming calculations of new image difference matrices. This is done by first calculating for the complete database sequence and the complete localization sequence the full image difference matrix. If then a sub-sequence is selected by using the sliders a sub-matrix of the complete image difference matrix is referenced and therefore any new image similarity calculations are avoided. So although there is initially a greater processing time to be expected for the construction of the complete image difference matrix, subsequent test runs complete much faster, which has been confirmed testing the application.

4.3. Experiments and Results

In the following section a detailed description of experiments conducted by using DreamViewer is given. The aim of the experiments was to find out if there are situations where the OpenSeqSLAM algorithm using tangent distance performs better than the original OpenSeqSLAM algorithm using the mean absolute difference. This has been done by conducting an experiment using dream sequence datasets recorded on the Myon robot in a real world indoor environment. Later experiments use larger datasets in combination with the separately applied image transformations rotation, scaling and translation in horizontal and vertical direction as well as different image resolutions.

DreamViewer experiment with KOB dataset

The dataset used for this experiment is named KOB dataset in reference to its recording location in a room at the Komische Oper Berlin. All used sequences where recorded in a single room but from different positions. For the following experiment three different sequences are used. For each sequence, the Myon head, that includes the recording camera, was setup at a fixed position. After starting the sequence recording a slow panning shot from the left to the right was captured. In total 20 images were recorded during the complete movement and an approximate 180 degree range was covered. The used recorded images had a resolution of 25 pixels in width and 20 pixels in height.

The data recording setup is outlined in figure 4.3. It shows a layout view on the recording room and some landmark objects that can be identified in the recorded images. In total, three different recording locations were used, these are indicated by colored circles in the figure. The first recording position for the first dream sequence is marked by a red circle. The Myon head was located at this position and 20 images were recorded while panning from left to right. The capture direction of each recorded image during this pan movement is indicated by the dotted lines originating in the center of the circle. At the remaining two positions similar recordings were performed. In the figure the green and blue circles mark the locations of the second and third sequence recordings respectively. The center of each recording location is separated by equal distances of

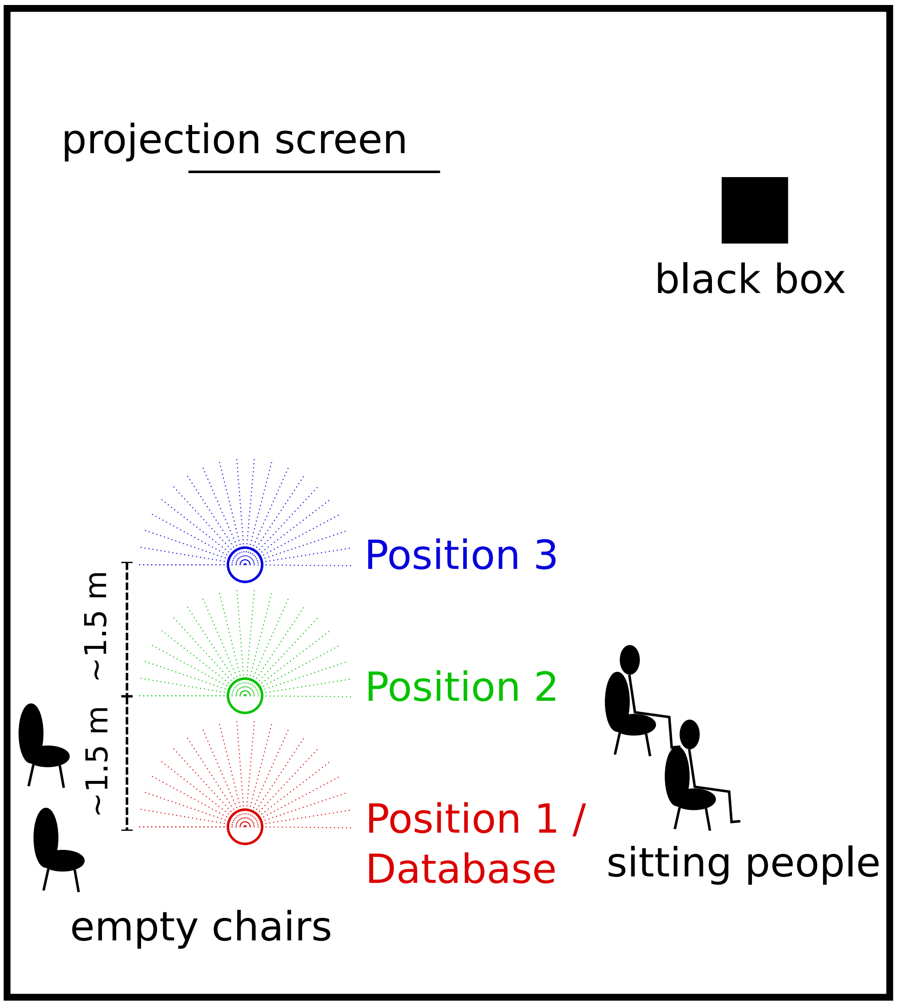


Figure 4.3.: Outline of the location used for KOB dataset recording. The first recording position is indicated by a red circle, the second by a green and the third by a blue circle. Colored rays indicate directions of image recordings. The position of the projection screen, identifiable in the recoded images, is represented by the black line. Positions of other identifiable objects are indicated by black symbols as well. These are a group of sitting people to the right of the recording positions, a set of empty chairs to the left and a black, box shaped object, to the right of the projection screen.

approximately 1.5 meters. This kind of position change had different effects on the recorded images. In figure 4.3 it can be seen that the recording position was moved effectively in parallel to the left and right wall or towards the projection screen. Images that were recorded in direction of the movement show predominantly a scaling effect between different recording locations. For example the projection screen is visible in images recorded at the green position but also in images that are recorded at the blue position only that the projection screen will appear larger viewed from the blue position. This can be verified by comparing the two images Another effect that can be observed is translation in horizontal direction. This effect will become largely visible for images shot in orthogonal direction to the recording position movement. These are the images facing the left and right wall. Though not actively provoked, but also not willingly prevented, small rotations and translations in vertical direction are also present between the different sequences. In conclusion a mix of the following image transformations is summarized that can be present between images of the sequences:

- scaling
- horizontal translation
- small rotation and vertical translation

In the following experiment a test will be conducted if the OpenSeqSLAM with tangent distance and the standard OpenSeqSLAM implementation with mean absolute difference are able to recognize an image subsequence recorded at the colored green and blue position respectively when the image sequence recorded at the red position is used as image database sequence. Each of the three sequences holds 20 images but for the sake of clarity in the following description only a smaller selection of images is shown. However, the complete sequences are included in the appendix A. Each sequence was recorded with a resolution of 25 pixels in width and 20 pixel in height. Although at first impression this seems to be a low resolution, for humans it is still possible to identify many objects and characteristic features of a room by looking at images of this type.

The results of the experiment are first summarized in tabular form and will be discussed with illustrating figures afterwards. In the first column the local input image indices are listed from top to bottom in ascending order. The second column takes reference to this column. Each entry in the second column lists images of the database sequence that should be recognized as being similar to the local input image that is listed in the same row in the first column. Similar in this case means that a human recognized the depicted scene of the local image to be closely related to the listed database images. Since the data was recorded in a real indoor environment, in some images, objects that are visible in one direction and close to the recording position, like for example a chair, can cover large parts of the scene. In these cases it was, even for a human, not possible to recognize the scene and no matching indices in table are listed. In this experiment this was the case for images of the beginning and the end of the sequences. Of course the problem of these non recognizable images is usually bypassed by using a larger localization sequence length. In this first experiment in total only 20 images were recorded and used. The localization sequence length was chosen in proportion to that number and set to a value of two. The reason for that was to focus more on the image similarity metric instead of gaining too much information by choosing a larger sequence size. The last two columns in the table list the results of the OpenSeqSLAM algorithm variants using mean absolute difference and tangent distance. To summarize, each row lists the localization results of the two OpenSeqSLAM variants in the last two columns where the first local input image for both algorithms is listed in the first column and the acceptable result images are listed in the second column. This means a localization result of an algorithm is considered correct if it is listed in the second column. Since the matching database images in the second column were composed by a human and therefore can be considered subjective and also the overall database sequence length is relatively small it should be noted that these first tests are not intended to prove superiority of one algorithm over the other but instead should show that localization is possible with the contributed implementations and to give some example of certain tendencies and challenges. In later tests each algorithm will be challenged with a greater emphasis on precise localization performance.

First the algorithms were presented with the input images recorded at the second recording location highlighted green in figure 4.3.

Table 4.1.: Results of image sequence recorded at position 2 using position 1 sequence as database.

Indices Position 2	Matching Database Indices	OpenSeqSLAM MAD	OpenSeqSLAM TD
3	2	6	6
4	3	7	6
5	3	7	7
6	4, 3	0	7
7	5, 4	0	9
8	5, 6	6	6
9	7	7	7
10	9, 10, 11	1	11
11	13, 12	12	12
12	13, 14	13	14
13	14	13	13
14	15, 14	7	7
15	16	8	8
16	17, 16	9	9

Again the experimental results are listed in tabular form for the third recording position highlighted in blue in figure 4.3

Naturally, the numeric values in the presented tables alone are not much intuitively comprehensible, so in the following, the results will be interpreted and summarized by using underlying example images taken from the input sequences and the database sequence. In general two observations can be made from the results. First the algorithms recognize the short local image sequences taken in direction of the presentation screen fully in case of OpenSeqSLAM with tangent distance and partly in case of the mean average distance variant. This is reflected by the middle rows in table 4.1 and table 4.2. Secondly in this experiment they seem not to recognize local images recorded orthogonally in direction to the presentation screen. These results are reflected by the top and

Table 4.2.: Results of image sequence recorded as position 3 using again position 1 images as database.

Indices Position 3	Matching Database Indices	OpenSeqSLAM MAD	OpenSeqSLAM TD
4	3	7	7
5	4	8	7
6	5	6	9
7	5, 6	0	6
8	6	7	6
9	7	0	7
10	8	1	10
11	9, 10, 11	11	11
12	11, 12, 13	11	11
13	13, 14	10	10
14	14, 13	10	10

bottom rows.

The results in the middle rows of the tables can be better understood by looking at the images in figure 4.4. The three images show the presentation screen centered in the view. This object is easily recognizable also for a human because of its rectangular shape and relatively homogeneous intensity. Differences between the images are present because of scaling and small translations. The OpenSeqSLAM with tangent distance correctly matches this view for recording position 2 shown in figure 4.4b and position 3 shown in figure 4.4c to the database image, recorded at position 1 and shown in figure 4.4a.

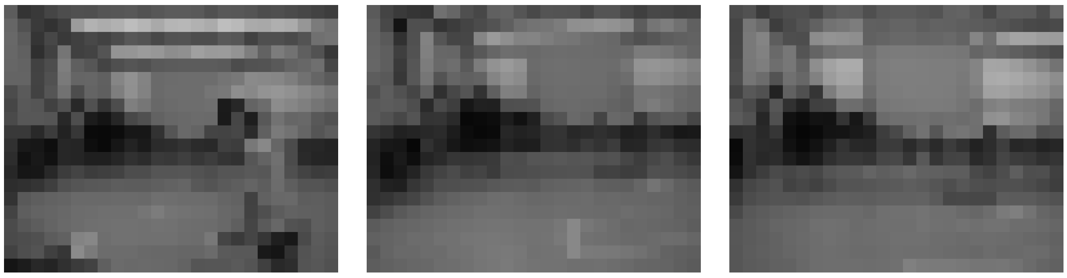
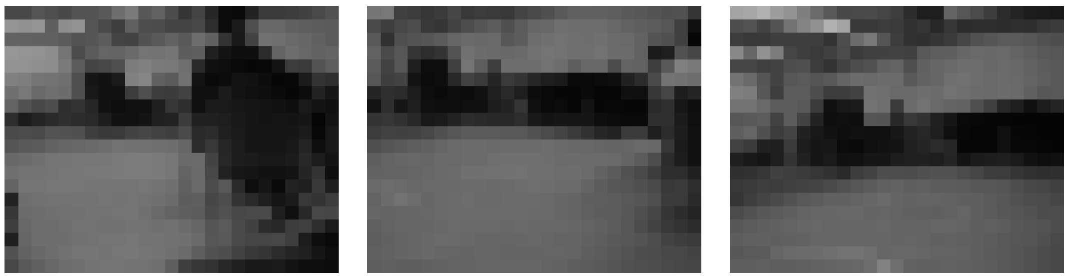

(a) Database (Position 1) (b) Position 2 (c) Position 3

Figure 4.4.: The three images show the presentation screen centered in view for all three recording positions.

The same images are matched by the OpenSeqSLAM algorithm using mean absolute difference correctly in the case of recording position 2 but incorrectly for the view at position 3. However, further tests showed that the MAD variant matched the position correctly as well when using localization sequence sizes greater than two.

As mentioned above, both algorithms were in error when facing localization images facing the left and right wall. The poor results in direction of the left and right wall can be at least partly attributed to objects positioned close to the Myon head and thereby causing large differences in the image when the recording position was changed. They also hid most other features visible in that direction of the room. In the figure 4.5 an example is given where such an object is visible and might have had an adverse effect on recognition performance. Again the three images were shot from the three recording positions. The first image was taken from position 1 and used in the database. The view is directed to the right wall and in the left part of the image the far right corner of the room and the black box object can be identified. Both other images in figure 4.5 were recorded facing roughly the same direction. What is different is that in the database image a person sitting on a chair is fully visible in the center right part of the image, covering the rest of the view partly. In the middle image this person is not visible, only partly a chair and a person can be identified at the outer right of the image border. In the third image nothing of the aforementioned is visible.



(a) Database (Position 1) (b) Position 2 (c) Position 3

Figure 4.5.: The images show the view towards the corner right of the presentation screen for all three recording positions. In one case the view is partly blocked by a chair.

This concludes the first experiment and the following aspects can be summarized. Both algorithms were able to perform correctly scene recognition despite perspective changes. However objects relatively close to the recording position can cover major parts of the image and possibly contribute to wrong recognitions.

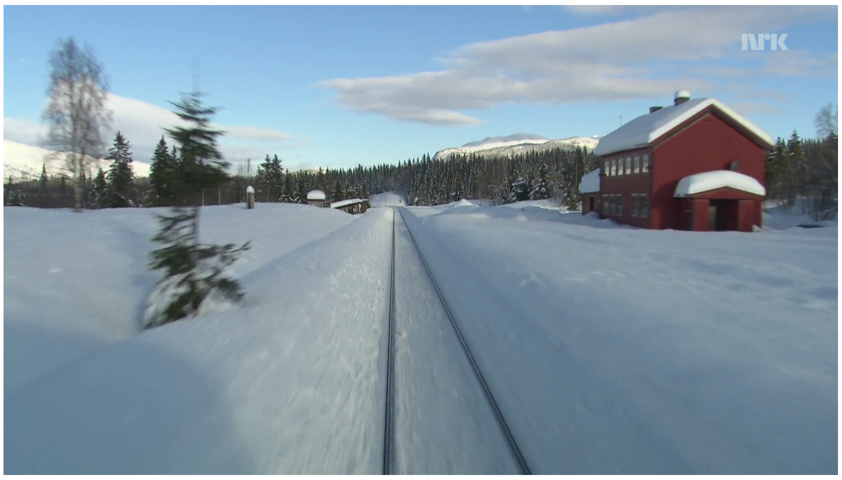
4.3.1. Experiments using Nordland dataset

In this section, a second experiment for testing the OpenSeqSLAM variants using mean absolute difference and tangent distance will be described. The focus is set on using larger databases and precisely setup image transformations to explore the localization performance of both algorithms operating on transformed input image data. The used dataset is called *Nordland* dataset and was used in the context of OpenSeqSLAM by Sünderhauf et al. in [SNP13] and [SNP14] to determine localization precision. This dataset is a collection of images obtained from four HD movies of the Nordland Line filmed at all four seasons of the year. These movies are produced by the Norwegian broadcaster *NRKbeta* and made available under a creative commons license [4]. They can be obtained online from *NRKbeta* at their website [5]. The Nordland Line is a railway line connecting the two Norwegian cities Trondheim in the south and Bodø in the north. So each movie shows the same railway line at a different season. With a total rail length of 729 kilometers it provides a large variety of natural scenery and different artificial objects like houses and bridges along the track. The movies are set apart by the influence of the seasons, for example snow cover in winter or different vegetation periods. Originally these movies were produced by the Norwegian broadcaster *NRK* and licensed under Creative Commons. Each movie was recorded by mounting a front facing camera into the cockpit of same type locomotives. All recordings were shot in an original resolution of 1920 pixels in width and 1080 pixels in height. Furthermore, a continuous GPS tracking was performed. Afterwards, each frame of the four different movies was synchronized using the GPS data, so that in effect for each movie the frame number k is equal in location and perspective to the frame number k of all other sequences. Sünderhauf et al. modified these movies to a downsampled version of 64 pixels in width and 32 pixels in height and obtained per movie one set of grayscale images by saving every second the current frame. So in result there is one image set for each season. The dynamic range for each pixel in the downsampled grayscale images is 0 to 255 at 8-Bit. Further details on how the downsampling was performed are explained in [SNP13]. Some examples of the downsampled frames can be seen in figure 4.6. There, a frame obtained from the original Nordland winter recording is shown together with its matching image in the nordland winter dataset. It can be seen that for humans characteristic objects like for example houses, mountain formations, single standing trees and the bending pattern of the rails can still be distinguished at the lower image resolution variant.

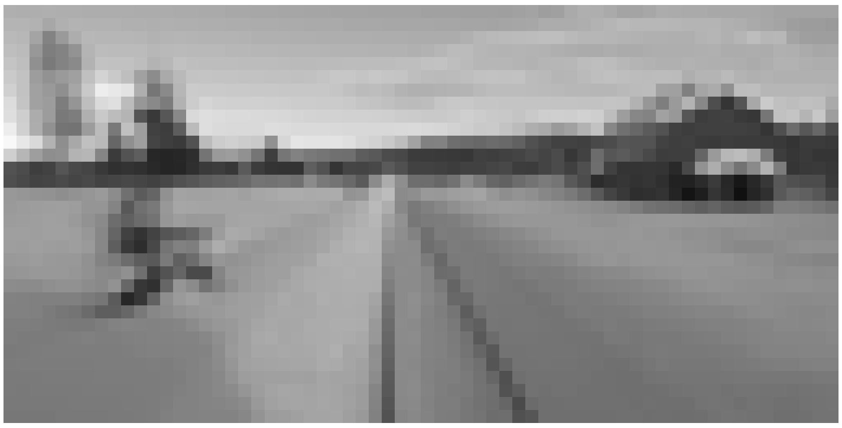
The following localization experiments involve a number of image transformations being applied to the original images to various degrees and tests with different image resolutions to see how these parameters affect the localization performance of both OpenSeqSLAM algorithms using mean abolute difference and tangent distance. In the experiment a part of the Nordland winter dataset was used. This dataset includes 800 images corresponding to a 13 minutes and 20 seconds clip of the original movie. These 800 original images were used as database sequence for all here described experiments. Each localization algorithm was confronted with transformed versions of the database sequence as local input. This was done by performing 800 separate localizations for each transformed image set for each algorithm so that of one transformed image set each image was used as starting point for a localization run. The expected correct result for each of these runs is, that for a given transformed local image k, used as starting input, the localization algorithm should return k as localization result for a correct answer, because the image k is, except for the applied transformation, identical to the image k in the database. The parameter localization sequence length has been set to 10 images, meaning that after the starting local input image 9 more directly succeeding images of the transformed image set were passed to the algorithms. To clarify this an example

[4]`https://creativecommons.org/licenses/by/3.0/`

[5]`https://nrkbeta.no/2013/01/15/nordlandsbanen-minute-by-minute-season-by-season/`



(a) Original Nordland frame



(b) Nordland frame processed

Figure 4.6.: The two image depict a single frame of the Nordland dataset before (figure 4.6a) and after pre processing (figure 4.6b). Before processing the image is colored and has a resolution of 1920 pixels in width and 1080 pixels in height. The processed image is a greyscale image with 64 pixels in width and 32 pixels in heigt.

for one specific image transformation experiment can be given as follows. One transformation that was applied to the original images was rotation and an image set that consists of all the images of the database sequence rotated by 15 degrees was created. At beginning the first image of the rotated image set and its nine successors were passed as local input to the OpenSeqSLAM with mean absolute difference and OpenSeqSLAM with tangent distance. In case of a correct answer the algorithms need to return the first database image index as a result. This experiment is then repeated for the image localization sequence beginning at the second image of the rotated image set and so on until each of the 800 transformed images was once used as a starting point. As mentioned before a number of different transformations have been used with different parameters. What kind of images in addition to the original Nordland sets were exactly created is detailed in table 4.3. There, each row lists what kind of data sets were generated for one of the used image transformation types. The last column specifies the unit of the used parameter and the second to third column represent the applied range of this parameter. For example from the third row it can be understood that 141 different rotation image datasets were created, using a rotation in degrees out of the range $-35, -34.5, -34, .., 35$ degrees for each of them.

Table 4.3.: Parameters used for localization experiments.

Transformation type	Range Start	Range End	Step	Comment and Unit
Translation x	-18	18	1	translated Pixel
Translation y	-18	18	1	translated Pixel
Rotation	-35	35	0.5	angle in Degrees
Scaling	1	3	1	zoom Factor

To create these datasets, a Python script was created that parses for a given directory all subdirectories and transforms the image files therein according to passed command line parameters similar to those specified in table 4.3. In addition to that each image set was created in different image resolutions. Meaning that beginning with the base resolution of 64 pixels in width and 32 pixels height downscaled versions of the images were created and used for testing. Inducement for experimenting with different image resolutions in the context of image sequence based localization was the paper by Michael Milford called "Vision-based place recognition: how low can you go" [Mil13], which was one of the idea giving cornerstones for this contribution as mentioned in the introduction section. In this work Milford et al. look, in the context of OpenSeqSLAM using MAD, at how localization performance develops when reducing the information content of the database and input images for example in terms of total number of pixels per image and the dynamic range per pixel. One of the main results of said paper is that image sequence based localization can be precise at high recall rates when using a surprisingly low number of pixels and further finds that localization performance can suffer in some cases when image resolutions is chosen too high. Because of these findings it became imperative to look in addition to the different translations at a number of different image resolutions for the described experiment. Another important reasons why lowering image resolutions should be considered is that it can be expected advantageous regarding processing time and memory consumption, which will be elaborated in a later chapter. The used resolutions were:

- 64 pixel width, 32 pixel height
- 32 pixel width, 16 pixel height
- 25 pixel width, 20 pixel height

- 18 pixel width, 16 pixel height
- 12 pixel width, 10 pixel height
- 8 pixel width, 6 pixel height

In case of the 12 pixels in width and 10 pixels in height images and certainly for 8 pixels in width and 6 pixels in width image resolution, even larger objects become reduced to a few pixels and can not be recognized by humans. It is then hopeless to identify a correct match on the basis of a single input image in combination with a larger image database. However, due to the use of a sequence of input images, as shown by Milford, even in those cases correct recognition becomes possible because of the combination of multiple single images that only faintly hint the original depicted location due to the reduced resolution. Of course based on the presumption this is only possible if the sequence in the database is at least partly similar to the input sequence. The base resolution being 64 pixel in width and 32 pixel in height the image transformations were applied before resizing the images. This was done to ensure that the amount of transformation is comparable between the different resolutions. Otherwise for example a translation applied to an image in 12x10 resolution by one pixel in horizontal direction would correspond to a significantly larger lateral perspective variation than a translation by the same amount on the 64x32 images.

In the above paragraph it has been said that in the experiment a localization result was considered correct if for an input image sequence the index of the correctly matching database image was returned. Due to the strong semantic link between images of the dataset it is however also of interest to note how far off the results returned by each localization algorithm were. If for example for all performed localization runs an algorithm returns an index that differs from the correct image position by two, this algorithm can still be considered performing better than an algorithm that only exactly once returns a perfectly correct result but differs with a maximum offset for all other localization results. In conclusion for each transformed image set different error measures were looked at. First there is the mean absolute error (MAE), defined in equation 4.1 as:

$$MAE := \frac{1}{n}\sum_{i=1}^{n} |\hat{k}_i - k_i| \tag{4.1}$$

Where n is the number of performed localizations, that is in this case $n = 800$, $\hat{k}_i$ is the expected correct database index for the localization i and k_i is the result that was returned by the algorithm. From this definition it can be seen that this error measure takes into account how close on average the algorithm was to the original image location in the database. It should be noted however that it is possible that in some cases the transformed version of the input image might indeed be more reminiscent of another database image instead of its untransformed original in the database. The MAE however is of course oblivious to these situations and might punish such a result of a localization algorithm even if a human would consider it to be correct. The possibility for that increases for a larger database but in case of the used 800 images sets should not be a dominant influence.

Another used localization error measure was the precision. In this context the precision is defined as:

$$p := \frac{cl}{cl + fl} \tag{4.2}$$

Where cl is the number of correct localizations and fl is the number of false localizations. A localization is considered correct only if the algorithm returns a database index that is either exactly the index of the original untransformed image or differs only by one from that position. This is in contrast to the MAE a more binary measure in the sense that a result is either correct or false with nothing in between and can give an idea how often in relation to all localization runs the algorithm hits the nail on the head precisely. A definition of precision can not go without giving reference to the recall rate, sometimes referred to as sensitivity. Because in the performed experiments the algorithms were confronted exclusively with input images that are represented in the database and were not allowed to opt out by indicating that the input is unknown to them, the recall rate is 100 percent. This means the algorithms were forced for every input to return a best effort solution.

Other related measures that were computed are standard deviation and mean squared error (MSE). In the following the results of the experiments will be presented and later discussed.

4.4. Offline Implementation Experimental Results

In this section the results of the experiment described in section 4.3.1 are presented. To visualize them a set of figures is presented. However the numeric results that these figures are based on are provided separately with this contribution. The results are separated by the six different image resolutions. To be able to focus on the main characteristics of the results mainly one of the four used image transformation types, that is the rotation operation, will be presented in this section. Notwithstanding that some connections to the other image transformations are outlined and the results in full can be found in the appendix B.

In the following results at least two figures for each tested image resolution are presented. One of them is a plot of the mean absolute error over the applied image rotation in degrees. The other figure is a plot of the calculated precision value according to equation 4.2 and the definition given in the previous section 4.4. The precision is also presented in dependence to the applied image rotation in degrees.

To make the results of the different resolutions more comparable a set of attributes has been picked out. Those attributes will be compared for all the resolution experiments. These are:

1. general difference between OpenSeqSLAM and OpenSeqSLAM-TD

2. points of result graphs intersections indicating performance inversion
3. mean absolute error at and between -10 and 10 degrees of rotation
4. mean absolute error around -20 and 20 degrees of rotation
5. mean absolute error tendency around -30 and 30 degrees of rotation
6. precision tendency around -30 and 30 degrees of rotation

With larger degrees of image rotation it was expected and confirmed by the results, that the precision of all algorithms monotonically gets smaller. Because of that it became possible to analyze for each resolution the following questions regarding the precision of the algorithms:

- When is the precision is 1?
- When is the precision at least 0.98?
- When is the precision at least 0.95?
- When is the precision at least 0.90?
- When is the precision at least 0.80?

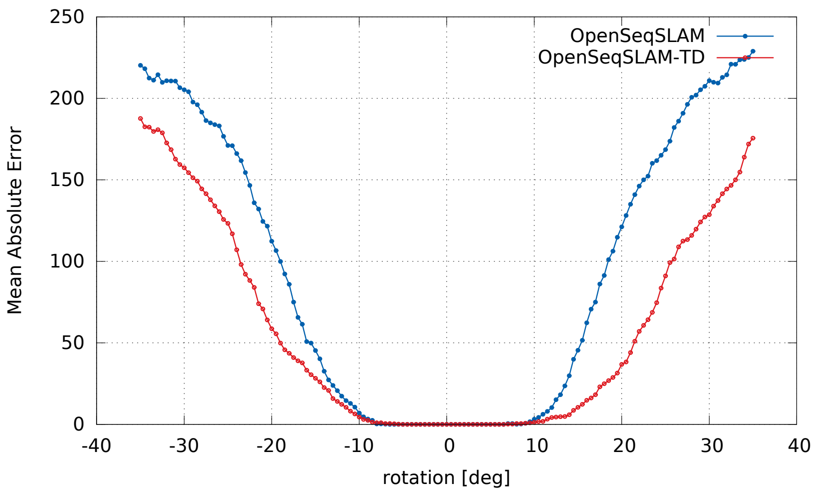
Results of the experiment with the smallest image resolution of eight pixels in width and six pixels in height are presented in figures 4.7. It can be seen in the upper figure, that outside the interval -10 to 10 degrees in rotation, the mean absolute error of the OpenSeqSLAM-TD algorithm is smaller than the error of the original OpenSeqSLAM algorithm for the tested range. With increasing rotation in either direction the difference between the OpenSeqSLAM-TD and OpenSeqSLAM-MAD error and precision further increases.

Selected numeric results of the mean absolute error for the smallest image resolution are presented in table 4.5. It can be seen there that the OpenSeqSLAM-TD had a lower error for this experiment at all sampled positions than the algorithm using the mean absolute distance. It is also observable that the difference in error between the two increases steadily for increased degrees of rotation. A comparison regarding precision is detailed in table 4.4. From the numeric values in this table it becomes clear, that the OpenSeqSLAM-TD algorithm in the majority of cases maintains for larger ranges of rotation a higher precision value than the mean absolute difference variant.

For images with a resolution of 12 pixels in width and 10 pixels in height the experimental results are presented in graphical form in figure 4.8 and numeric excerpts of these are presented in the table 4.6 and table 4.7 sampling precision and MAE respectively.

The results show that for this resolution in case of the OpenSeqSLAM-TD over the rotation angle range from -20 to 20 degrees higher precision is achieved than for the smallest image resolution. This can be seen in table 4.6 where in the before mentioned rotation range, in case of OpenSeqSLAM-TD, the precision not drops below a value of

(a) 8x6 rotation MAE



(b) 8x6 rotation precision

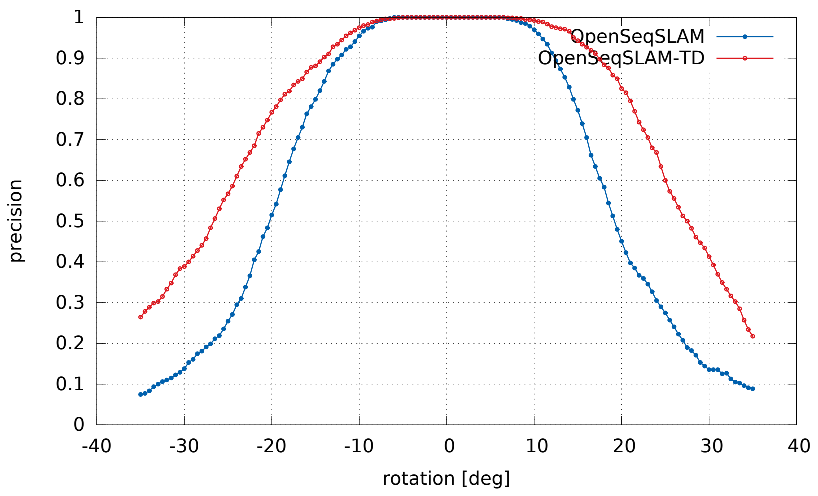


Figure 4.7.: Results of localization experiment using Nordland winter 8x6 images.

0.80. For the same algorithm clearly the mean absolute error is also smaller in the -10 to 10 degrees rotation range than it was for the smallest resolution which can be seen in table 4.7. The OpenSeqSLAM-MAD also improves slightly but the changes are not as large. However for rotations beyond -20 degrees or 20 degrees the precision drops more sharply than in the case of the lower resolution image and the mean absolute error is also greater.

The next larger image resolution that was experimented with were images with 18 pixels in width and 16 pixels in height. The results are presented in figure 4.9 and numeric excerpts are listed again in tabular form. This is done for the precision in

Table 4.4.: Excerpt of the precision results gained from the rotation experiment with images of 8 pixels in width and 6 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
$= 1.00$	$[-6.0, 6.5]$	$[-5.0, 6.5]$
≥ 0.98	$[-8.0, 9.0]$	$[-9.0, 11.5]$
≥ 0.95	$[-10.0, 10.5]$	$[-11.5, 14.5]$
≥ 0.90	$[-12.0, 12.0]$	$[-14.0, 17.0]$
≥ 0.80	$[-14.5, 14.0]$	$[-18.5, 20.5]$

Table 4.5.: Excerpts of mean absolute error results from the rotation experiment using images with 8 pixels in width and 6 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
-30.0	205.28	157.39
-20.00	112.33	58.66
-10.00	6.96	4.45
10.00	3.10	1.22
20.00	121.15	36.79
30.00	210.86	128.65

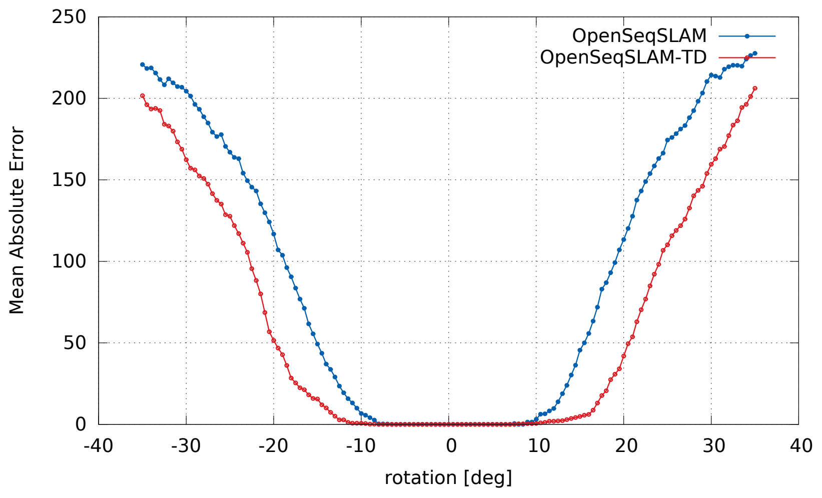
table 4.8 and mean absolute error results can be found in table 4.9. Over the complete rotation range the OpenSeqSLAM-TD variant achieved for this experiment a greater precision value than the OpenSeqSLAM-MAD variant. For rotation degrees between -30 and 30 the mean absolute error of OpenSeqSLAM-TD is lower for larger degrees of rotation the mean absolute error was comparable for both algorithm variants.

It can be seen in table 4.8 that the rotation degree range in which a precision value of 1 is achieved is slightly wider for the OpenSeqSLAM-TD algorithm than it was for smaller resolutions. Apart from the rotation range in which a precision value of 1 is achieved, the OpenSeqSLAM-TD precision drops faster than in case of the smaller resolutions. The OpenSeqSLAM-MAD algorithms performs equally or slightly worse compared to images with smaller resolution. This can be seen for the mean absolute error in table 4.9 and also for the achieved precision listed in table 4.8.

For images with a resolution of 25 pixels in width and 20 pixels in height according to the results in figure 4.10 it can be seen that the OpenSeqSLAM-TD generally achieved higher precision than the OpenSeqSLAM-MAD algorithm. This becomes more clear when looking at table 4.4. However, while the precision of the OpenSeqSLAM-MAD appears to be similar to results achieved at images with 18 pixels in width and 16 pixels in height the tangent distance variant precision appears to have been slightly dropped so that smaller rotation angle ranges are tolerated.

The mean absolute error has also increased for the regarded image resolution. In

(a) 12x10 rotation MAE



(b) 12x10 rotation precision

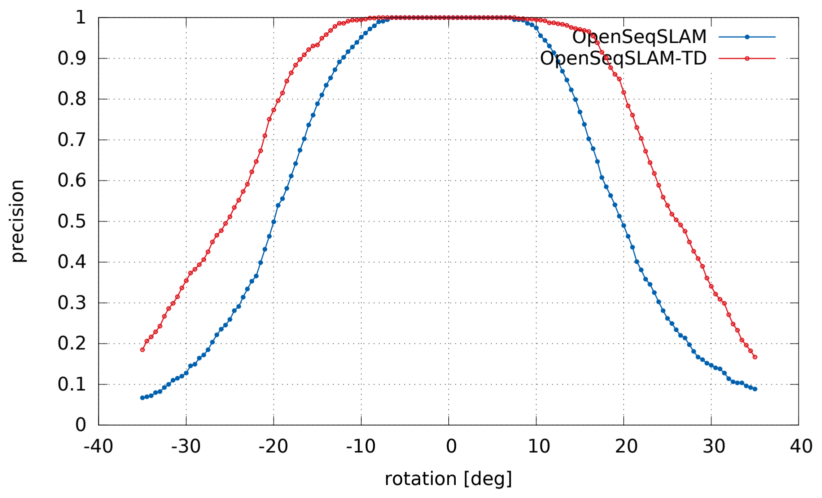


Figure 4.8.: Results of localization experiment using Nordland winter 12x10 images.

table 4.11 it can be seen that the OpenSeqSLAM-TD has a smaller error for rotation up to -20 or 20 degrees. In contrast to the lower image resolution results, for rotations around -30 degrees, the mean absolute error of OpenSeqSLAM-TD was higher than the OpenSeqSLAM-MAD error for this experiment. Similar for 30 degrees and above performance of OpenSeqSLAM-MAD seemed to be comparable or superior compared to OpenSeqSLAM-TD.

Images with a resolution of 32 pixels in width and 16 pixels in height show similar performance to the images with 25 pixels in width and 20 pixels in height and the results are presented in figure 4.11. However the difference in error between OpenSeqSLAM-TD

Table 4.6.: Excerpt of the precision results gained from the rotation experiment with images of 12 pixels in width and 10 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
$= 1.00$	$[-6.5, 7.0]$	$[-8.0, 7.5]$
≥ 0.98	$[-8.0, 9.5]$	$[-12.5, 13.5]$
≥ 0.95	$[-10.0, 10.5]$	$[-14.0, 16.5]$
≥ 0.90	$[-12.0, 12.0]$	$[-16.5, 18.0]$
≥ 0.80	$[-14.5, 14.0]$	$[-19.0, 20.0]$

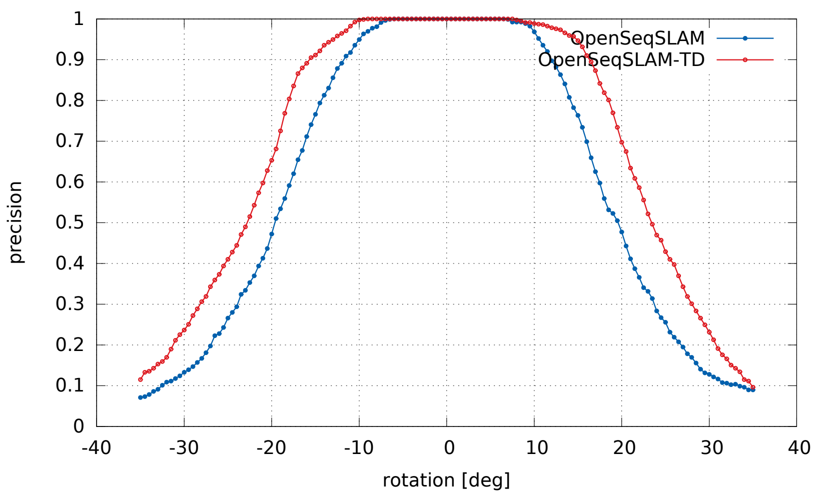
Table 4.7.: Excerpts of mean absolute error results from the rotation experiment using images with 12 pixels in width and 10 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
-30.0	204.40	162.34
-20.00	116.79	51.48
-10.00	6.68	0.64
10.00	3.14	0.55
20.00	113.37	41.93
30.00	214.31	159.55

and OpenSeqSLAM-MAD for rotations larger than -20 or 20 degrees becomes slimmer.

The precision results in numerical form for the 32 pixels in width and 16 pixels in height images are presented in table 4.12. Compared to the results of table 4.10 for OpenSeqSLAM-TD the difference of achieved precision in the selected intervals is only marginal but was slightly greater for the 32 pixels in width and 16 pixels in height image resolution. The same applies for the OpenSeqSLAM-MAD algorithm. From table 4.13 it can be seen that the mean absolute error increases for both algorithms compared to the lower image resolutions for rotations angles above -20 or 20 degrees. Between -10 and 10 degrees it is slightly lower than in the case of 25 pixels in width and 20 pixels in height images.

(a) 18x16 rotation precision



(b) 18x16 rotation MAE

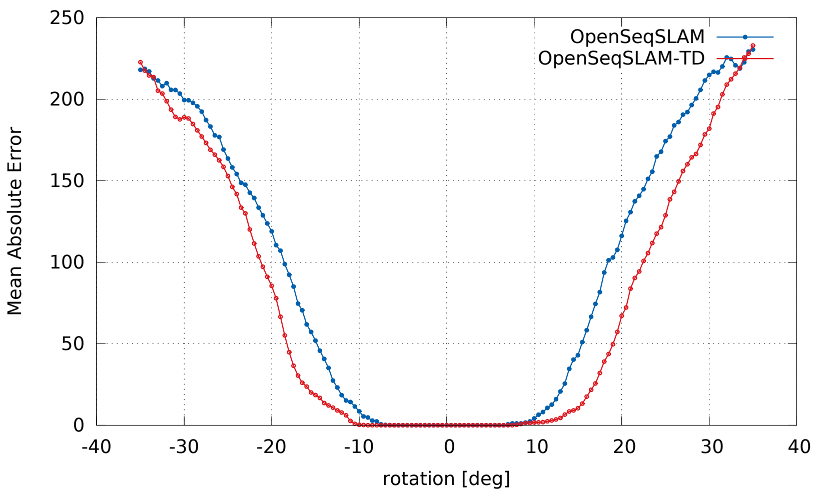


Figure 4.9.: Results of localization experiment using Nordland winter 18x16 images.

Table 4.8.: Excerpt of the precision results gained from the rotation experiment with images of 18 pixels in width and 16 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
$= 1.00$	$[-6.0, 6.5]$	$[-9.0, 7.5]$
≥ 0.98	$[-8.0, 9.5]$	$[-11.0, 11.5]$
≥ 0.95	$[-9.5, 10.5]$	$[-12.5, 14.5]$
≥ 0.90	$[-11.5, 11.5]$	$[-15.5, 16.0]$
≥ 0.80	$[-14.0, 14.0]$	$[-18.0, 18.5]$

Table 4.9.: Excerpts of mean absolute error results from the rotation experiment using images with 18 pixels in width and 16 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
-30.0	199.49	188.94
-20.00	118.96	85.49
-10.00	8.52	0.31
10.00	4.17	1.69
20.00	116.22	67.10
30.00	214.91	181.95

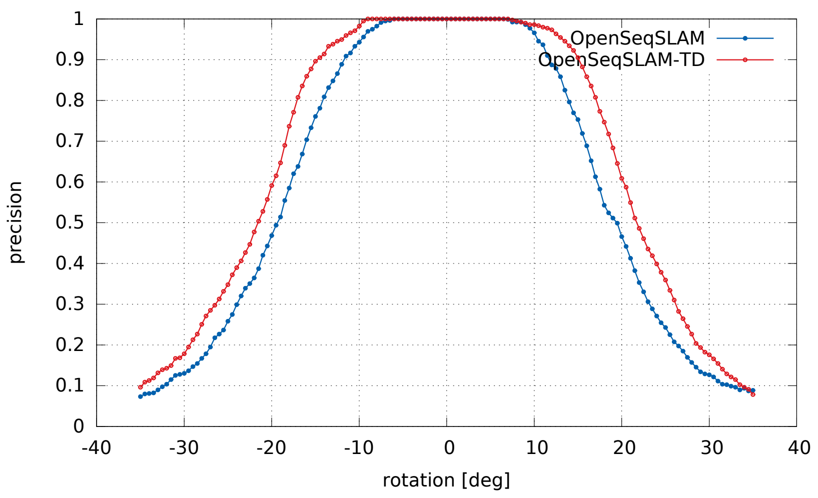
Table 4.10.: Excerpt of the precision results gained from the rotation experiment with images of 25 pixels in width and 20 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
$= 1.00$	$[-6.0, 6.5]$	$[-9.0, 7.0]$
≥ 0.98	$[-8.0, 9.0]$	$[-10.0, 10.5]$
≥ 0.95	$[-9.5, 10.0]$	$[-11.5, 13.0]$
≥ 0.90	$[-11.5, 11.5]$	$[-14.5, 15.0]$
≥ 0.80	$[-14.0, 13.5]$	$[-17.0, 17.0]$

Table 4.11.: Excerpts of mean absolute error results from the rotation experiment using images with 25 pixels in width and 20 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
-30.0	195.11	202.21
-20.00	116.67	97.96
-10.00	10.16	3.31
10.00	4.66	1.93
20.00	116.95	89.03
30.00	211.27	200.46

(a) 25x20 rotation precision



(b) 25x20 rotation precision

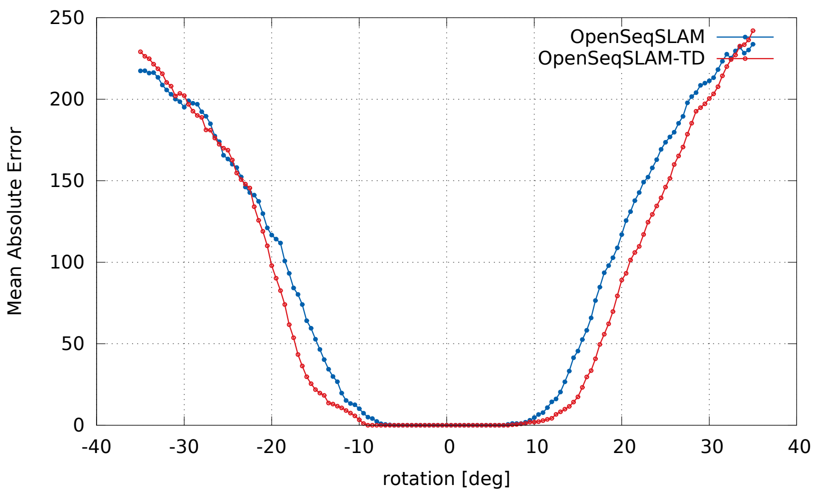
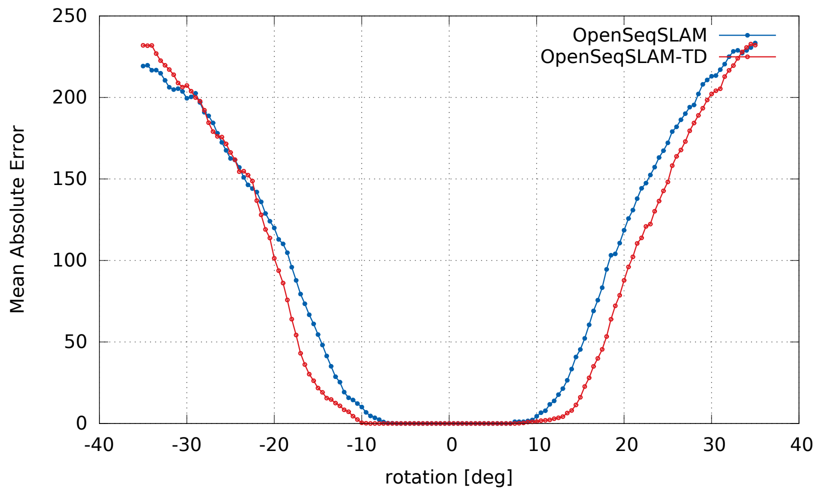


Figure 4.10.: Results of localization experiment using Nordland winter 25x20 images.

(a) 32x16 rotation MAE



(b) 32x16 rotation precision

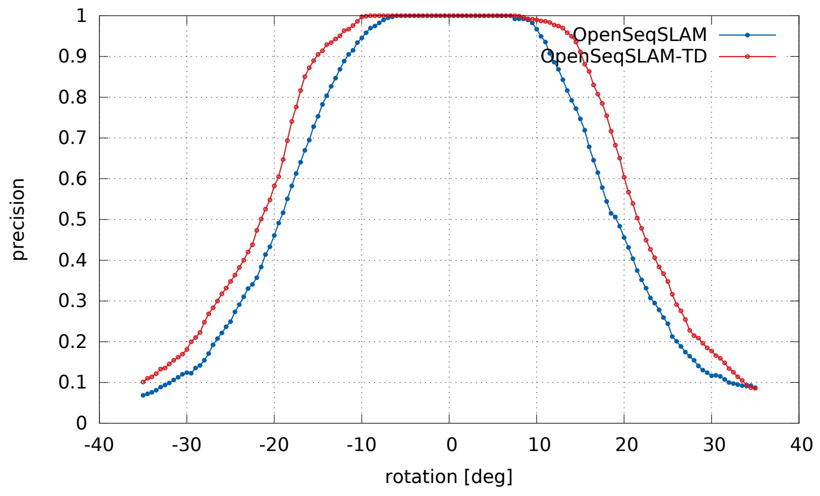


Figure 4.11.: Results of localization experiment using Nordland winter images 32x16.

Table 4.12.: Excerpt of the precision results gained from the rotation experiment with images of 32 pixels in width and 16 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
$= 1.00$	$[-6.0, 7.0]$	$[-9.0, 7.5]$
≥ 0.98	$[-8.0, 9.5]$	$[-10.5, 11.5]$
≥ 0.95	$[-9.5, 10.0]$	$[-12.5, 13.5]$
≥ 0.90	$[-11.5, 11.5]$	$[-15.0, 15.0]$
≥ 0.80	$[-14.0, 13.5]$	$[-17.0, 17.0]$

Table 4.13.: Excerpts of mean absolute error results from the rotation experiment using images with 32 pixels in width and 16 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
-30.0	199.51	207.34
-20.00	119.95	101.37
-10.00	10.11	0.44
10.00	4.40	1.17
20.00	118.52	87.76
30.00	213.00	202.15

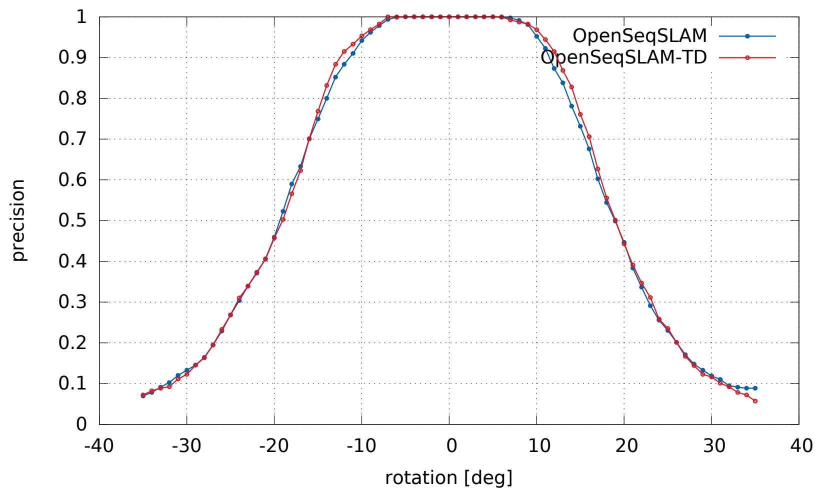
The highest image resolution tested had 64 pixels in width and 32 pixels in height. The related results are visualized in figure 4.12. It can be seen that regarding precision both algorithms achieve similar results.

The numeric results of the experiments are given in excerpt for the achieved precision in table 4.14. It can be seen there that the OpenSeqSLAM-TD algorithm does maintain the listed precision values for slightly larger rotation ranges than OpenSeqSLAM-MAD. However overall the precision of both algorithms is lower in any case than for the smaller resolutions. The same applies for the mean absolute error which is listed in table 4.15

4.4.1. Performance observations and discussion

Most sections of both OpenSeqSLAM variants, mean absolute difference and the tangent distance computations can be processed in parallel. Both algorithms were implemented with threads to make use of all CPU-cores, offered by the machine performing the experiments. However especially for larger databases when testing with DreamViewer and performing other experiments it became clear that performing a single localization takes noticeable time. The most processing intensive calculations happen when generating the image difference matrix. As explained in previous sections, to create this matrix the similarity of each local input image in combination with each database image needs to be calculated. For example in case of the experiment described in the previous sec-

(a) 64x32 rotation precision



(b) 64x32 rotation MAE

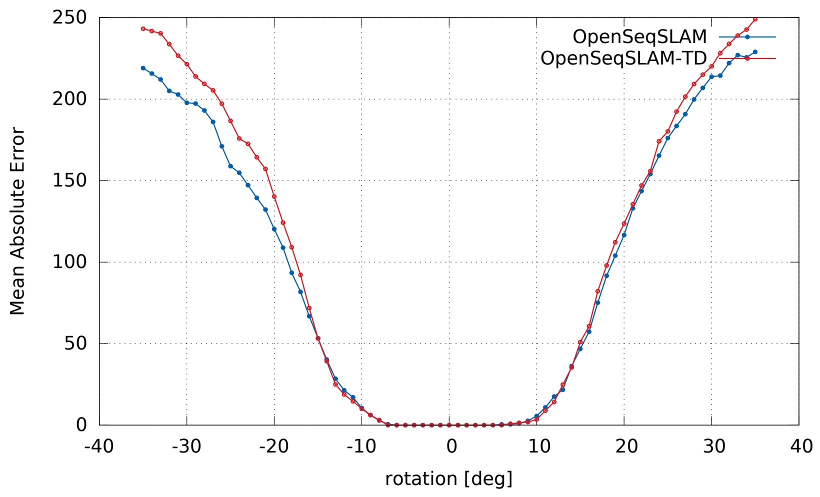


Figure 4.12.: Results of localization experiment using Nordland winter 64x32 images.

tion, using the 800 image Nordland database, for each transformed image set alone, over 640000 image similarity calculations were performed per algorithm. Of course in addition, execution of all calculations performed after creating the image difference matrix also scale with its size. The time to find the minimizing path in the image difference matrix behaves also proportionally to the range of considered slopes.

The results of both similarity metrics are real valued, so in terms of memory consumption, the image difference matrix when representing the results as floating point numbers, has a considerable size. That is because those data types typically use between four and eight Bytes per value. So for the used image dataset, one image difference matrix took

Table 4.14.: Excerpt of the precision results gained from the rotation experiment with images of 64 pixels in width and 32 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
$= 1.00$	$[-5.0, 5.0]$	$[-7.0, 6.0]$
≥ 0.98	$[-7.0, 9.0]$	$[-8.0, 9.0]$
≥ 0.95	$[-9.0, 10.0]$	$[-10.0, 10.0]$
≥ 0.90	$[-11.0, 11.0]$	$[-12.0, 12.0]$
≥ 0.80	$[-14.0, 13.0]$	$[-14.0, 14.0]$

Table 4.15.: Excerpts of mean absolute error results from the rotation experiment using images with 64 pixels in width and 32 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD
-30.0	197.73	221.46
-20.00	120.26	140.26
-10.00	10.52	10.02
10.00	5.63	3.45
20.00	116.62	123.55
30.00	213.70	220.11

up 5120000 Bytes in the experiment. This might be considered negligible on a modern desktop machine but on robotic systems most system resources are heavily sought after and precious so even this single matrix could already be too much to keep in working memory.

In any case the tangent distance requires more computations than the mean absolute difference metric because the tangent distance includes as final step a pixel based difference on top of all other performed operations. In the context of handwritten digit recognition LeCun et al. compare various digit classifiers, including the tangent distance, in [LJB+95]. They evaluate the metrics regarding their classification performance but also analyze processing time and memory usage. Although for their example classification performance is considered good, the tangent distance used up most memory of all classifiers and was slowest in all calculations. Parameters that are mostly affecting its computation time are the two factors

- number of used tangent vectors
- number of pixels

It has been shown at the end of section 3.2.2 that the calculations involve the inversion of a matrix with row and column size equal to the number of tangent vectors. The number of required calculations for the matrix inversion however increases more than quadratically when increasing the matrix size. Another aspect is that each additional tangent vector needs to be produced two times in the resolution of the two images that are compared. This is also a reason why the number of pixels per image largely contributes to the processing time of the tangent distance.

In terms of memory usage the mean absolute difference metric also is much more leaner than the tangent distance. The reason for that is that during the tangent distance calculations not only the original two images need to be kept in memory but also all tangent vectors and intermediate results for each image when solving for the scalar parameters in equations 3.17.

In conclusion the computational burden of creating and searching in the image difference matrix can be considered to be too high when intending to use it in combination with the tangent distance on a robotic system. For implementation on the Myon an alternative algorithm was developed and is detailed in the following chapter.

5. Online Implementation

In the previous chapter the tangent distance based localization algorithm showed promising results for the online experiments. Its use on the robotic platform was therefore considered. However for reasons that are detailed in the following section 5.1 the exact algorithm could not be just simply deployed without modification to the robot. Instead an adapted algorithm was developed based on a heuristic approach. Its concept and design is described in the following sections and thereafter evaluated and compared to the previously presented algorithms.

5.1. Motivation and Reasons for another Algorithm

As described in the introduction chapter 1 an important reason for the development of the *OpenSeqSLAM* using tangent distance was to enable the *Myon*, or other humanoid robots, to localize themselves through image sequences. The results and evaluation of the experiments, detailed in section 4.4, make it evident that the OpenSeqSLAM-TD requires considerable large amounts of memory and processing power during its computations. This however is problematic in the context of getting the algorithm to run on typical humanoid robotic systems like the *Myon* where different tasks need to be performed concurrently and such a demanding task could intolerable cripple overall system performance. Especially when real-time performance and great reactiveness is desired. As for many humanoid robots *Myon's* current processing hardware can be categorized as an embedded system. In general, that also brings some constraints regarding general purpose processing power and working memory availability with it, at least when compared to a modern desktop or laptop computer. So when designing an adapted algorithm at least some characteristics of the underlying hardware should be considered which is done in the following section 5.2. In conclusion, it can be said an adapted algorithm is needed because of the following arguments:

- need to cut processing time to get a real-time like answer
- need to reduce memory usage to fit embedded hardware capabilities

To achieve this, instead of the analytical, processing intense solution, a heuristic approach was developed that incorporates a greedy strategy to answer the localization problem. There is evidence that heuristic approaches can work well solving real life situations and day-to-day decision problems. In fact, one can argue that we humans use heuristic behaviour on a daily basis like when guessing the general mood of a person simply based only on facial expression or predicting the winner of a football match. Of course how well a heuristic strategy works depends on the complexity of the problem and the available cues to make the heuristic decision. It can also go wrong e.g. when choosing the supposedly fastest queue in the supermarket based on queue length.

In the field of artificial intelligence heuristic search algorithms are widely adopted for example in the field of path planning. An overview is presented by Russell and Norvig in [RN10, Chapter 3]. Heuristic based decision making has also been studied in other problem domains like psychology, finance and economy. Many different heuristic approaches for those areas are presented at length for example by Gigerenzer and Gaissmaier in [GG11]. Gigerenzer et al. also developed the *Take The Best* (TTB) heuristic, described in [GG96]. This heuristic is in many ways similar to the heuristic developed for the online algorithm described in this chapter. A number of the mentioned heuristic strategies will be briefly characterised and summarized in section 5.3 and there also the general concepts of the devised heuristic used in the online algorithm are presented.

In the next section the robotic hardware platform, used in context of the performed experiments and for collecting data, will be described. This robot is called Myon. The available hardware capabilities needed to be considered in the design of the online algorithm because Myon was the intended target platform for testing the algorithm.

5.2. Myon

In this section a brief overview of the main characteristics of the robotic platform used for experimentation and data collection is given. The robot is called *Myon*. *Myon* is a 1,25 meters high humanoid robot weighting about 15 kilograms. It is developed and used by the members of the *Neurorobotics Research Laboratory* (NRL). The robot's design and functioning, that is only partly summarized here, is presented in greater detail by the developers, Hild et al., for example in [HSB$^+$12, Chapter 2] and [HSB$^+$11]. To get a more complete description of the robot more pictures and related research are also available at the NRL website [1]. One of the distinctive features of Myon is that it is composed of different body part modules that can be attached in various ways to each other to act as a combined unit, but each of them can also operate autonomously when separated. This behaviour is possible because of processing capabilities and energy supplies contained in each separable body part. Also locally in each module there is a neuronal network that can control actuator behaviour independently of any central behaviour processing unit. This way it is even possible to combine or detach modules during runtime. The separable parts are two legs, one torso, two arms, two hands, and a head. In each body part there are also different types of sensory data available, these include currently multiple three dimensional accelerometer readings, motor position readings, angles, currents, voltages and forces. This data of every connected body part is published in the robot's body by a data bus named *SpinalCord*. Updated sensory data is provided by the *SpinalCord* every 10 ms. Some of the published sensory information bear the potential to be used for improving the performance of place recognition. This aspect however is out of the scope what can be fully evaluated in this contribution but is outlined in section 5.6.

[1] `http://www.neurorobotics.eu/robots/myon_en.php`

Image Sequence Recording on Myon

The main body part that was used in the context of this thesis is Myon's head. The hardware module integrated in it is called the *BrainModule*. This module includes a camera that was used for recording the image sequences data for the experiments. The camera model is a 21K15XDIG and single images are provided in a full color image resolution of 346 pixels in width and 271 pixels in height. This camera is connected to a Field Programmable Gate Array (FPGA) board. A range of modules, instantiated on the FPGA, handle the image data reading from the camera, audio processing and perform other crucial tasks for example generating *SpinalCord* events. The precise details regarding the functioning and features of the *BrainModule* are also elaborated by Hild et al. in [HSB^{+}12]. Camera image data updates are read out every 20 ms and pixel values are saved into a pixel buffer. This pixel buffer can be seen as a memory bank on the FPGA that is addressable like an array and can store a maximum of 512 pixel values. Of course 512 pixel values means, that the original camera resolution can not be fully buffered. The mapping from the original resolution into the buffer is performed by transforming each pixel coordinate of the original image into an address of the pixel buffer. To do this first a two by two dimensional transformation matrix A is used. This matrix maps the x and y coordinates of each pixel in the input image to a new coordinate pair x' and y'. After a pixel coordinate pair is transformed by the transformation matrix A two offsets b_x and b_y are added, so that the final coordinates are $x'+b_x$ and $y'+b_y$. It is then checked if the generated coordinate pair is within a range of two defined threshold values x_{max} and y_{max}. These values can be understood as the dimensions of the image that will be stored in the pixel buffer and because of the size of the pixel buffer they must be set to values so that $x_{max}y_{max} \leq 512$. Only if $0 \leq x'+b_x < x_{max}$ and $0 \leq y'+b_y < y_{max}$ holds the pixel value at pixel position x and y in the original image is saved in the pixel buffer at position $x_{max}(y'+b_y)+(x'+b_x)$. For the recording of the image sequences different resolutions were tested. In result the transformation matrix A was setup to scale down the original input images to a resolution of 25 pixels in width and 20 pixels in height. This offers the advantage that every part of the input image is mapped into the pixel buffer for every new camera image update. In effect the full field of view of the camera is preserved and this resolution still offers the possibility to recognize characteristic objects in the scene as was demonstrated in the example images of section 4.3. The above described image buffering is performed in dedicated modules implemented on the FPGA. In addition there is a general purpose soft processor unit instantiated on the FPGA. This unit is a so called *MicroBlaze* microprocessor and is designed and provided by the manufacturer *Xilinx*. More detailed information about this unit can be acquired from the online *Xilinx* documentation [2]. In general terms the *MicroBlaze* is used to control the above described modules instantiated on the FPGA, handles events like *SpinalCord* updates, triggers capturing of images, has access to the pixel buffer and can execute arbitrary programs. For this contribution a program was written in *C* and executed on the *MicroBlaze*. It performed the recording of image sequences in the *dream sequences*

[2] `http://www.xilinx.com/tools/microblaze.htm`

formats as described in the section 4.1. This software relied on already provided libraries and interfaces to configure the pixel buffer parameters, trigger the image capturing and write the sequence data to a SD card. Most importantly this processor was the intended target to test the implementation of the heuristic online localization algorithm on the Myon. Although as of the time of writing a new version of the *BrainModule* is developed, that uses completely different processors, for testing the heuristic algorithm on the Myon the performance limits of the *MicroBlaze* had to be considered. The used *MicroBlaze* had a memory capacity of 65456 Bytes, operated at a processing frequency of 72 MHz and was instantiated without an additional Floating Point Unit (FPU). In addition to the required system resources in terms of memory usage and processing time that could be anticipated for the developed localization algorithm, memory usage and processing capacity for the Myon libraries also needed to be considered. Also it was anticipated that because of the involved tangent distance calculations floating point operations would be involved. Hence when designing and implementing the contributed algorithm these constraints needed to be confronted. However as noted before the implementation of the algorithm itself is independent of the hardware. The heuristic localization algorithms will be presented in detail in the following sections.

5.3. Algorithm

The experiments performed in section 4.3.1 showed that the OpenSeqSLAM algorithm using tangent distance recognition performance is in many cases superior to the mean absolute distance variant of the algorithm. In the same context it was recognized that the runtime behaviour regarding processing time and working memory requirements can not be neglected especially when considering it in context of a robotic platform. In the introduction of this chapter therefore the case was made for developing a different algorithm using a heuristic approach. In this section it will be reasoned why a heuristic might be a valid approach for the problem of image sequence based place recognition and what kind of related heuristic approaches it draws its inspiration from. In addition further assumptions about the execution of the algorithm will be made.

Heuristic algorithms are widely used in the field of artificial intelligence. Luger describes them in [Lug05, Chapter 4] by describing the heuristic idea in the context of search. According to Luger a heuristic algorithm is designed in a way so that the search for a solution is directed by a set of rules that aim to prefer at each search step a subset of all possibilities in the search space that appear to be most promising to lead to a desirable solution. Thereby it is possible to reduce the search space by only considering candidates that are promising according to the heuristic rules. A prominent example of heuristics in the field of artificial intelligence and robotics is the path planning problem, meaning to search for a path with minimum cost between two locations represented by nodes in a graph. Different heuristic strategies that are established for this problem are for example presented by Russell and Norvig in [RN10, Chapter 3]. The path planning approaches presented by Russell and Norvig face all the common challenge that for larger graphs the search space of possible paths can exceed what is feasible or desirable to compute, so heuristics are used to reduce or prioritize regions in the search space. Heuristic path planning algorithms that not only prioritize but are actually reducing the size of the search space on the basis of what possibilities look best at the current step are also referred to as greedy algorithms. For those the following general attributes can be summarized:

- result is not always optimal
- reduced computation time
- reduced memory usage

The last two of the listed points are desirable for the online place recognition algorithm. The computation time and memory requirements are reduced simply because only a true subset of all possibilities in the search space is considered. However at the same time it is possible for some problems that a more optimal solution is overlooked by the heuristic algorithm. This infers the necessity to evaluate how the heuristic algorithm performs in terms of localization performance compared to the algorithm variants that do not reduce the search space. That was done experimentally in section 5.4.

Another example, originally from the field of psychology is the so called *Take the best heuristic* (TTB). As mentioned in the introduction of this chapter this heuristic was introduced by Gigerenzer et al. and is described in [GG96]. The heuristic aims to help deciding between alternatives. The decision is made on the basis of a set of cues. A cue is an information that can be attributed to each alternative. The TTB makes the decision based on a single cue that is most promising to make a correct decision in the general case. In the context of image sequence recognition we can assume the cue to be based on the image similarity metric. For example with the help of the tangent distance for an input image we can select a set of most similar candidate images in the database that can then be analyzed more closely. Since this approach eliminates a number of possible candidates that are not considered any further it falls also in the category of greedy algorithms.

Concepts and observations from the path planning problem heuristics and the heuristics described by Gigerenzer can be transferred to the image sequence recognition problem that is of concern in this contribution.

Clearly the image sequence recognition problem can be described as a search problem. For a given input image sequence a match in the database is searched for. The search space consists of all possible database image sequences that can be compared to the input image sequence. In case of the OpenSeqSLAM offline algorithms, evaluated and described in the previous chapter, the search space is analyzed using the image difference matrix data structure. This matrix completely maps the search space. It is kept in memory and the OpenSeqSLAM algorithms traverse all parts of this structure at least once in search for a match. The heuristic concept of reducing the search space was applied in the design of the online algorithm. In effect the image difference matrix is not created completely but a heuristic rule determines points of interest. The detailed functioning of the algorithm is described in section 5.3.1 but the heuristic concept involves performing a greedy step to reduce the search space. This step is very similar to the idea of the *take the best - leave the rest* heuristic and can be summarized as follows. For the first input image that is locally acquired the image similarity to all database images is calculated and afterwards a set of candidate images that seem to be most promising in terms of image similarity are determined. Afterwards exclusively the surrounding images of these candidates are considered in further calculation steps of the algorithm. From that it can be seen that the search space is reduced in a greedy manner, because solely the supposedly best candidates in the first step are selected for further processing and the rest of the search space is ignored.

Software Design and Implementation

In the introducing sections of this chapter the motivation for the heuristic algorithm is laid out. Further in section 5.2 it was detailed on what kind of target system the algorithm needs to run. But of course it should not be tailored to this platform alone. From all aspects mentioned there a set of general software requirements can be distilled that influence the software design in a major way. In summary the implementation should have the following attributes:

- portable and flexible
- extendable
- maintainable
- efficient
- reactive

Especially the last two points are to a great extend influenced by the design of the heuristic localization algorithm. The way this algorithm operates is detailed separately in section 5.3.1. However in general the implementation of the heuristic localization algorithm is capsuled in a separate module and this section describes the way this module is integrated with the other components. A full overview of all contributed modules and how they interact with the underlying system is shown in figure 5.1:

The heuristic algorithm is designed in a way so that it never needs to directly communicate with the underlying system hardware or system libraries for example to obtain input data. In the figure the module containing the algorithm is shown in the upper right part of the diagram as a rectangular box labeled *Heuristic Online Localization.* All other modules are symbolised by a labeled rectangle each as well. Arrows between boxes mark dependencies between modules. For example the heuristic algorithm module depends on the *linear algerbra*, *parameters* and image *provider module.* A single dashed line separates modules that were implemented in this contribution and modules that where provided in case of the Myon system. The following descriptions explain the functionality of each module and how they interact with other components. In addition related software design decisions are explained:

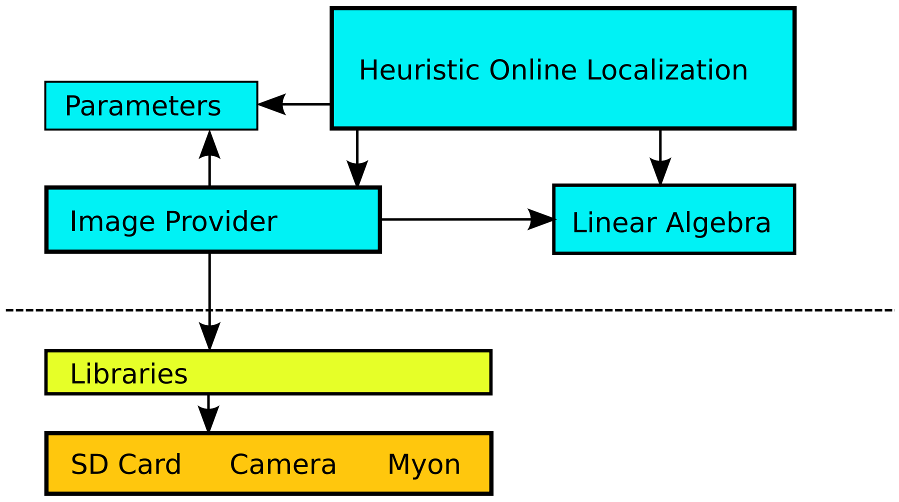


Figure 5.1.: Module structure of the heuristic algorithm. The dashed line separates the modules developed for this contribution and the used software and hardware of the Myon. Arrows indicate dependencies between the modules.

Heuristic Algorithm As mentioned above this module contains the online algorithm. It performs localization on the basis of local acquired input image sequences and a stored image database. However there is no functionality that implements access to the images themselves. Instead whenever the localization algorithm requires a new local image or database image it demands access to these resources through the interface of the *Image Provider* module. The idea behind this is to separate the localization algorithm implementation from platform specific functionality. If this was not done then for example the system calls to capture, load and store images would need to be integrated into the heuristic algorithm module and require adaptation each time the platform interface changes. Also regarding the aspects of maintainability and extendability the implemented separation is beneficial. Another module the Heuristic algorithm module depends on is the *Linear Algebra* module. It is used to perform all involved matrix calculations for example when calculating the tangent distance and various image convolution functions that are needed for the tangent distance calculations. Another dependency of the *Heuristic Localization* module is the *Parameters* module. The latter provides definitions of constants and defines parameter values that have an influence on the algorithm. For example the image database size and the number of used tangent vectors is defined here.

Image Provider This module handles requests from the *Heuristic Algorithm* module. It is the only module that includes platform specific functionality. In that sense it can also be regarded as an abstraction layer that handles calls to the robotic system libraries. In case of the Myon it conducts for example interactions with the camera and the SD card memory through the provided system libraries. That is highlighted in figure 5.1 by marked dependency to the *Libraries* module. In case the robotic hardware changes, *Image Provider* is the only module that needs to be modified when the interface of the *Libraries* module changes as well. These adaptations can be tuned to the available hardware resources. For example it is possible to implement the functionality in a way that all database images, requested by the *Heuristic Localization* module, are kept in the working memory. This will minimize the time needed to serve a request of the *Heuristic Algorithm*, but would consume considerable large amounts of working memory. Larger image databases might not fit completely. In these cases the Image provider can be implemented so that it only loads parts, or even single images, of the database while other parts that are currently unused are kept on a mass storage like the SD card. The *Image Provider* depends also on definitions in the *Parameters* module.

Linear Algebra This module implements a collection of common matrix operations like, addition, subtraction, multiplication, scaling, trace and matrix inversion. It also offers a pixel based image difference. All operations were implemented with the intent to minimize working memory usage. For example stack size was kept minimal by reusing local variables and offering the possibility to interpret matrices as transposed version by a flag instead of keeping a transposed version in memory.

Libraries and Myon Components To show how the developed modules interact with the provided components of the robot, figure 5.1 also includes a *Libraries* module. Its function is to offer an interface for using the hardware components of the robot. Some of these components that are used by the implementation for Myon are depicted also in the bottom part of the figure and include the SD-card and camera.

Parameters This module holds definitions of parameters used by the *Heuristic Algorithm* module and the *Image Provider*. For more details see also the descriptions given before.

All of the described modules were implemented in ANSI C. In addition it was possible to avoid the usage of C-library functions, so that the memory footprint of the software components on the target system was further reduced.

5.3.1. Description of the Localization Algorithm

In this section it will be described in detail how the heuristic algorithm is designed to work. Similar to the cases of OpenSeqSLAM and OpenSeqSLAM-TD the heuristic algorithms main goal is to match a sequence of locally acquired images to a stored set of database images so that the matched database image set is the most similar one to the local images. In the best case the matched database and local images were taken in the same location and so the localization is successful. In order to determine how similar two images are the implemented heuristic algorithm makes use of the in previous section 3.2.2 described tangent distance metric.

The algorithm can be divided into three main steps:

1. Take The Best Leave The Rest - greedy step (find k many TD minima)
2. follow the trail step (and sum up)
3. return the minimum

Each step is shown in more detail in pseudo code algorithm 1 and an example is outlined in figure 5.2. The first step, referred to as greedy step, is shown from line 1 to line 17 in the pseudo code. The general aim of this step is to find the $k \in N^+$ most similar images to the first local input image in the database. The algorithm will then consider only these k database positions in the following steps. That is the reason this step is called greedy step, because it behaves as described in [CLRS09, Chapter 16] classical greedy algorithms do. They are characterised by taking only into account the most promising sub-result and try to deduce an optimal solution from it.

The first step of the heuristic algorithm is exemplified in the upper part of the figure 5.2. There a box filled with the label L1 is shown. This shall represent the first input image. Beneath that there is an array like structure, which represents the image database. In the example we set $k = 3$, meaning a maximum number of three different positions in the image database will be searched for in the greedy step as the most promising locations. To determine these three locations first the similarity of every database image to the input image L1 is determined using tangent distance. The results of this step are represented in the figure by the second top row array that has greenly colored entries. Each shade of green represents the result of the tangent distance calculation of L1 and the corresponding database image at that index. More darker shades of green shall indicate a low tangent distance result and therefore greater similarity, whereas greater luminance indicates a larger tangent distance. In the same array three arrows mark the three currently most promising database locations and for each array field the tangent distance result value is given as a number below. So the database images at indices 0, 4 and 8 are selected with their respective tangent distance values 6, 5 and 3. After the three most promising database images are selected the second stage of the algorithm is performed by using the next input image, marked by the box labeled with L2. In the second step, using tangent distance, L2 is now compared to three images that are the direct successors of the images found in the greedy step. Each tangent distance result is accumulated by adding it to the result of the previous step. So in the example from left to right the new tangent distance values are 5, 4 and 6. Added to the values of the previous step that gives an accumulated value of 11 for the first and 9 for the other considered positions. The localization sequence length parameter in this example is set to three. So in this case the second stage of the algorithm is only repeated once and the final input image is labeled L3 as shown in the bottom part of the figure. Again the tangent distance results are added to the accumulated values of the previous results. The final scores for all three positions from left to right are 18, 12 and 16. Therefore the finally returned minimum is the second sequence associated with the first database image at index 4.

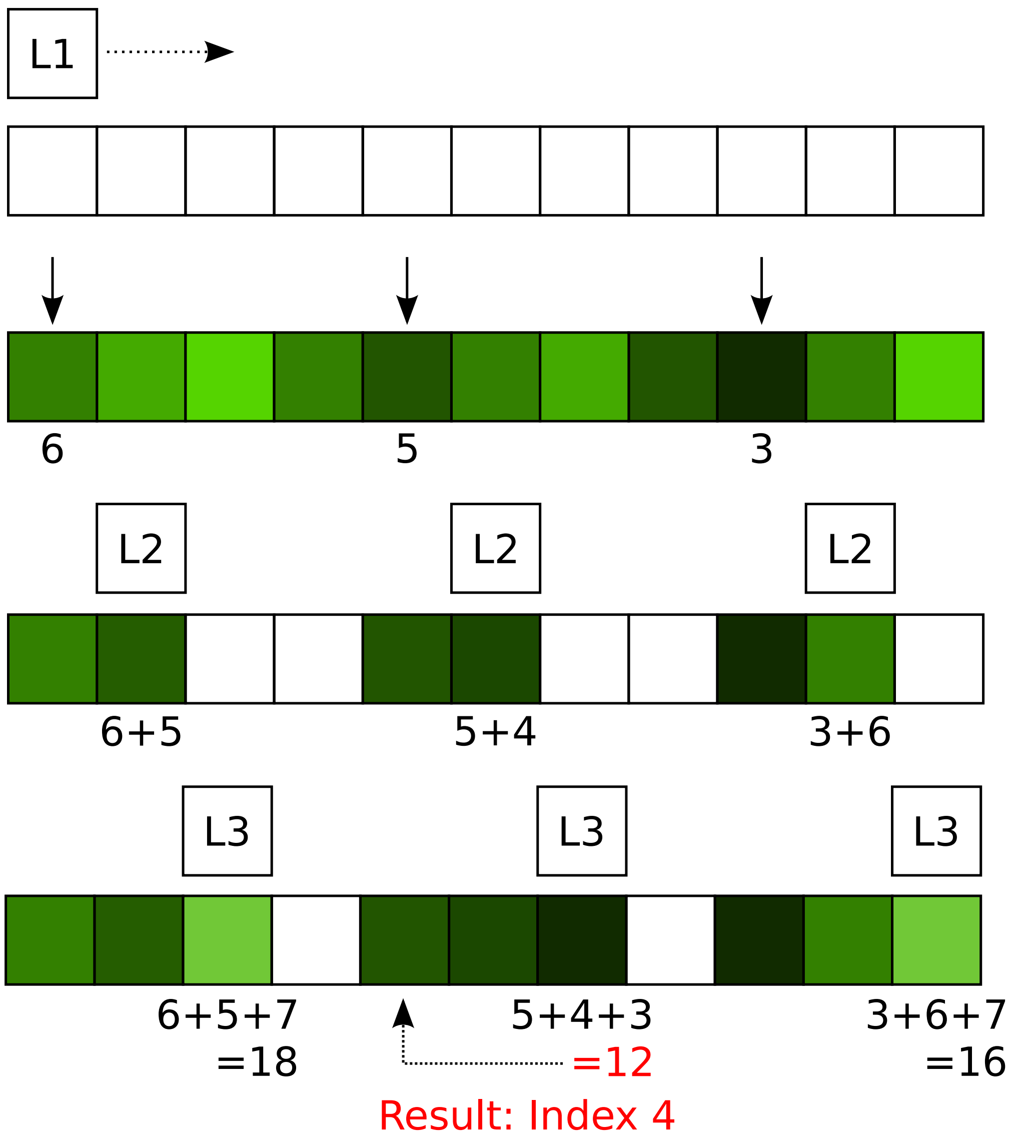


Figure 5.2.: Example heuristic localization execution using three points of entry and a localization sequence of length three.

In order to select these k images first the tangent distance for each database image and the first input image is calculated. Then it is decided if this database image should be included in the set of the k most promising images. However it is only included if all of the following criteria are met:

1. the tangent distance does not exceed a fixed defined threshold value
2. if k previous images are already selected, it needs to be more similar to the local image, than at least one image in the set

3. the index of the newly found database image is not too close to a more similar previous found image index

The third criteria is ensured by a guard window mechanism and its size is a configurable parameter of the algorithm. The motivation for using this guard window is to avoid the collection of more than one of the k database images near the same tangent distance minimum. This becomes more clear when looking at figure 5.3. There, the image database is represented by an array like structure and each cell of this array is colored in a shade of green to represent the tangent distance of the local image and the database image at this position. A dark green represents a low tangent distance whereas more light tones represent higher tangent distance values. Typically two similar images will be marked with a dark green. For example array cell 4 appears to have a low tangent distance in the figure. However, it can also be seen that the surrounding cells of a minimum appear also in a darker tone. The reason for that is of course the spatial semantic context between neighbouring image database entries. In the example that means that cell images 2, 3, 5 and 6 will probably show only a slightly altered view of the same location that is shown in the database image associated with cell 4 and therefore have a lower tangent distance as well. Without the guard window it could happen that during the first stage of the algorithm cell images 3, 4 and 5 are included in the k greedy set but the database image associated with cell 11 is not. This is not desirable because we want to keep k as small as possible to reduce the problem size but at the same time be able to follow different separated positions in the image database that represent different locations and not just slightly altered versions of the same place. In figure 5.3 it is also shown that the guard window is moved as long as the tangent distance values are decreasing in the area covered by the guard window, so for example until it is placed at array cell 4. When the algorithm leaves the guard window the previous found minimum will be inserted into the greedy set. In the example this is position 4.

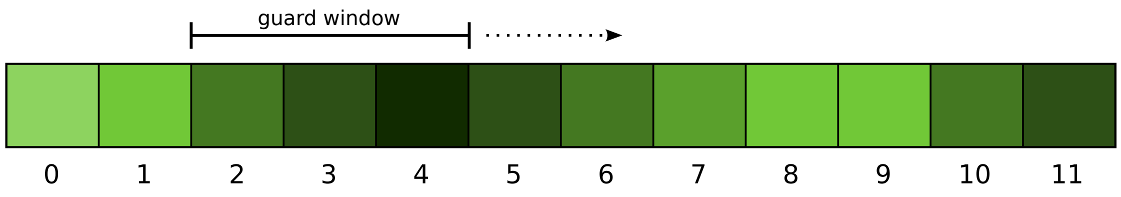


Figure 5.3.: Greedy step of heuristic online algorithm in progress. The guard window is intended to get a better spacial distribution of database elements that are considered for further sequence comparison. Shades of green indicate image similarity.

The second step of the algorithm is summed up in pseudo code from line 18 to line 25. For each of the k positions found in step one we look at the immediately following database image and calculate the tangent distance of it to the second localization image. This tangent distance value is then added to the tangent distance value calculated in step one and added to an accumulator. This process is repeated for all following local images. So for each of the k positions the tangent distance values are accumulated.

The third and last step of the algorithm looks for the minimal value of the k accumulated values and returns the original database position found in step one that is associated with it.

It is of course possible that no position in the database resulted in a tangent distance below the defined threshold value and in this case no database position can be returned. This offers the possibility to indicate locations that are yet unknown to the robot or with other words locations or perspectives that are not yet represented in the image database.

Experimental Implementation on Myon

In the previous sections of this chapter the design and workings of the heuristic online algorithm have been described. There it has been stated that a main goal was to design a heuristic algorithm that can be implemented and run on the Myon robot, described in section 5.2. In order to demonstrate that the implemented algorithm can indeed work on the Myon a small demonstration was created. The main procedure is that Myon's head is setup in a room. Then through the camera in Myon's head three images of different perspectives of that room are recorded and saved in the image database. Intended were for example three images of different surrounding walls. After the image database is filled with these three images the program enters an infinite loop. In that loop first a current camera image is obtained and then secondly the heuristic online algorithm is presented with that image as local input image. It should return the database image, or to be more precise, its database index, that is recognized as being sufficiently similar to the local input. If none is sufficiently similar that should be noted too. Then the loop repeats itself. Tests with that setup showed that the algorithm works but a number of glitches exist. For example when images are presented to the algorithm that are not much similar to the images in the database, they are sometimes matched nonetheless and often to a wrong index. This can happen when the head is turned from one perspective represented by a database image to another, because in between the matching fails. This problem might be reduced by adjusting the recognition threshold or waiting for multiple loop circles to complete, until a recognition is confirmed. However, this could damage responsiveness of the localization process. Admittedly this experiment could not be much more simple in its approach considering the database only holds three images and the input image sequence has a length one. Nevertheless, in addition to the limited time frame of this thesis and constrained system resources, both not allowing for more thorough tests of larger scale in this way, the described experiment was not intended to build a detailed picture of the algorithm's recognition performance. Instead it was a proof of concept for the implementation and can be used as a basis for further development, experiments and integration into the Myon system. Notwithstanding the above the performance for larger scale image databases and input sequences at different image resolutions and image transformatinos was analyzed for the heuristic approach as it was done in the previous chapter for the OpenSeqSLAM offline algorithms. These results and conclusions are presented in the following sections.

```
Data: used variables
image database, $Images$
constant value recognition threshold, threshold
local image, initialized with first input image, $l \leftarrow GetNextLocalImage()$
minimal distance in guard window, minimum $\leftarrow$ threshold
greedy candidate list $G \leftarrow \emptyset$ , later filled with
tuple elements $t(index, distscore)$
Result: best matching index $result$ in image database to local images OR -1 if
        not recognized
foreach $i \in Images$ do
    distance $\leftarrow$TangentDistance($i, l$)
    if $inside\ guard\ window$ $\wedge$ distance $<$ minimum then
        minimum $\leftarrow$ distance
        index $\leftarrow$ index of $i$ in $Images$
        center guard window at index
    end
    else
        if $outside\ guard\ window$ then
            AddToGreedyList($t$(index, minimum))
        end
        if distance $<$ threshold $\wedge$($G$ $not\ full$ $\vee$(distance $<$ $maximal\ distscore\ in\ G$)
        then
            minimum $\leftarrow$ distance
            index $\leftarrow$ index of $i$ in $Images$
        end
    end
end
foreach $offset \in$ $1$ .. $localization\ sequence\ length$ - $1$ do
    $l \leftarrow$ GetNextLocalImage ()
    foreach $t(index, distscore) \in G$ do
        $i \leftarrow Images[index + offset]$
        distance $\leftarrow$TangentDistance($i, l$)
        update $t(index, distscore)$ with $t(index, distscore +$ distance)
    end
end
if $|G|$ $==$ $0$ then
    return -1;
end
$t(minindex, minscore)$ = GetMinimalscoreTuple($G$)
return minindex
```

Algorithm 1: Pseudocode of heuristic online algorithm. It returns the index of the database image best matching the local input images or -1 in case no recognition was possible.

5.4. Experiments and Results

In this section experimental results for the heuristic online algorithm are presented. As described in the previous sections of this chapter the developed heuristic online algorithm makes use of the tangent distance to determine image similarity. Of course it is well possible to use other image similarity metrics instead of the tangent distance like the mean absolute difference. In the previous chapter the tangent distance variant of the OpenSeqSLAM algorithm was compared to the OpenSeqSLAM algorithm using mean absolute difference. This was done to evaluate potential benefits of the tangent distance in connection with this localization method and so for similar reasons the following experiments not only include the tangent distance variant of the heuristic online localization but also a mean absolute difference variant was created and tested.

To be comparable with the offline algorithm's experimental results obtained in the previous chapter the same experiments as described in section 4.3.1 were performed for the heuristic online algorithm using tangent distance and the heuristic online algorithm using mean absolute distance. The results are described in the following section.

5.5. Results of Heuristic Online Algorithm Experiments

To make the results of the experiments more easily comparable to the experimental results of the offline implementations presented in section 4.4 a similar form of presentation as utilized there is used. For the same reason the offline implementation results are included into the figures and tables presented hereafter. However the offline results will not be described again in detail: They will only be compared to the results of the heuristic online algorithms.

For the smallest tested image resolution, that are 8 pixels in width and 6 pixels in height, the results are presented in figure 5.4. It can be seen there that in general the heuristic algorithms perform worse than the offline algorithms for this resolution. However this is not unexpected and possible reasons for this are given in the following section 5.5.1 concluding the experiments. Further the tangent distance variance has a lower precision than the mean absolute difference variant for a large part of the rotation angle range.

Excerpts of the precision results are numerically presented in table 5.1. It can be seen again that precision for both heuristic algorithms drops significantly faster for increasing degrees of rotation than it does for the offline OpenSeqSLAM variants. Similar observations can be made regarding the mean absolute error. This is sampled in table 5.2. Only for rotations larger than 25 degrees in both directions the mean absolute error of the heuristics was smaller.

The second lowest resolution tested was 12 pixels in width and 10 pixels in height. Results of the experiments are visualized in figure 5.5. It can be seen that the precision of the heuristic online algorithm using tangent distance was again slightly smaller for the selected rotation range than the mean absolute difference heuristic. However, it can be also observed that the difference in precision is smaller compared to the results obtained

Table 5.1.: Excerpt of the precision results gained from the rotation experiment with images of 8 pixels in width and 6 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
$= 1.00$	$[-6.0, 6.5]$	$[-5.0, 6.5]$	$[-2.0, 2.5]$	$[-3.0, 3.0]$
≥ 0.98	$[-8.0, 9.0]$	$[-9.0, 11.5]$	$[-4.0, 4.5]$	$[-5.5, 5.5]$
≥ 0.95	$[-10.0, 10.5]$	$[-11.5, 14.5]$	$[-5.5, 5.5]$	$[-6.5, 7.0]$
≥ 0.90	$[-12.0, 12.0]$	$[-14.0, 17.0]$	$[-6.5, 6.5]$	$[-8.0, 8.0]$
≥ 0.80	$[-14.5, 14.0]$	$[-18.5, 20.5]$	$[-8.0, 8.0]$	$[-9.0, 9.0]$

Table 5.2.: Excerpts of mean absolute error results from the rotation experiment using images with 8 pixels in width and 6 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
-30.0	205.28	157.39	147.16	119.00
-20.00	112.33	58.66	127.26	110.37
-10.00	6.96	4.45	49.10	22.11
10.00	3.10	1.22	41.54	12.41
20.00	121.15	36.79	130.76	110.89
30.00	210.86	128.65	164.90	141.58

for images with 8 pixels in width and 6 pixels in height and the same is true for the mean absolute error.

Again numeric results are summarized in tabular form. Intervals of rotation degrees, where certain precisions are achieved are listed in table 5.3. Mean absolute error results are listed in table 5.4. It can be seen from the numeric precision values that the rotation invariance for both algorithms has slightly increased when compared to results of the 8 pixels in width and 6 pixels in height image results. However it should be noted that these results only assess the rotation operation but as mentioned in the experiment description three other transformations where tested. Transformations like for example translation in horizontal direction show at this resolution quite different results and the tangent distance seems to be more stable for those transformations. To make this observation more clear, results for the horizontal translation transformation in the same image resolution of 12 pixels in width and 10 pixels in height are presented in figure 5.6. It can be seen in this figure that for the complete horizontal translation argument range the heuristic online algorithm using tangent distance has a greater precision than the mean absolute difference variant and even outperforms for large translation ranges the non-heuristic approaches. Without going into further details for this transformation type the rest of the analysis will focus on the rotation invariance. The experiments for the other transformation types are listed in the appendix B.

For the 18 pixels in width and 10 pixels in height images the results are presented in

Table 5.3.: Excerpt of the precision results gained from the rotation experiment with images of 12 pixels in width and 10 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

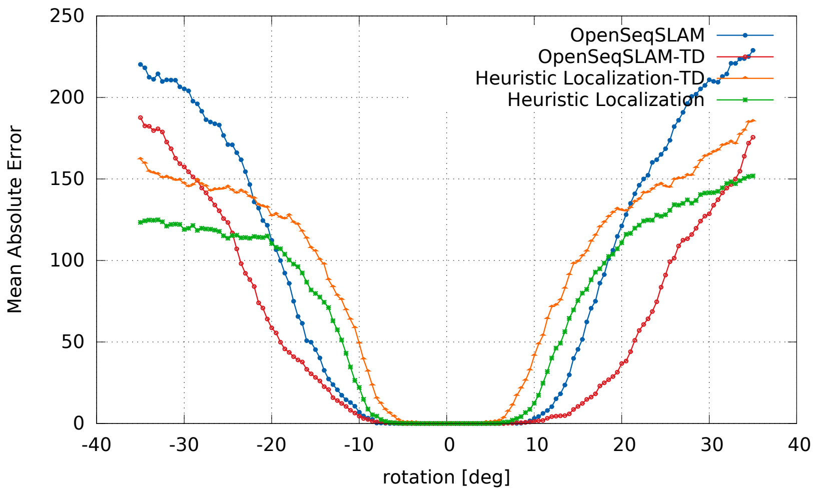
Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
$= 1.00$	$[-6.5, 7.0]$	$[-8.0, 7.5]$	$[-3.0, 3.0]$	$[-3.5, 4.0]$
≥ 0.98	$[-8.0, 9.5]$	$[-12.5, 13.5]$	$[-5.0, 5.0]$	$[-6.0, 6.0]$
≥ 0.95	$[-10.0, 10.5]$	$[-14.0, 16.5]$	$[-6.5, 6.5]$	$[-7.0, 7.0]$
≥ 0.90	$[-12.0, 12.0]$	$[-16.5, 18.0]$	$[-7.5, 7.5]$	$[-8.0, 8.0]$
≥ 0.80	$[-14.5, 14.0]$	$[-19.0, 20.0]$	$[-9.0, 9.0]$	$[-9.0, 9.5]$

Table 5.4.: Excerpts of mean absolute error results from the rotation experiment using images with 12 pixels in width and 10 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
-30.0	204.40	162.34	158.20	113.61
-20.00	116.79	51.48	125.62	103.33
-10.00	6.68	0.64	27.15	25.95
10.00	3.14	0.55	19.47	14.72
20.00	113.37	41.93	117.63	111.74
30.00	214.31	159.55	150.06	144.25

figure 5.7. As it was the case for the lower image resolutions, these results indicate that the heuristic approaches in total have a lower localization precision at this resolution than the OpenSeqSLAM variants. In contrast to previous results, however, the heuristic online algorithm using tangent distance overtakes the mean absolute error variant for some rotation degrees.

(a) 8x6 rotation MAE



(b) 8x6 rotation precision

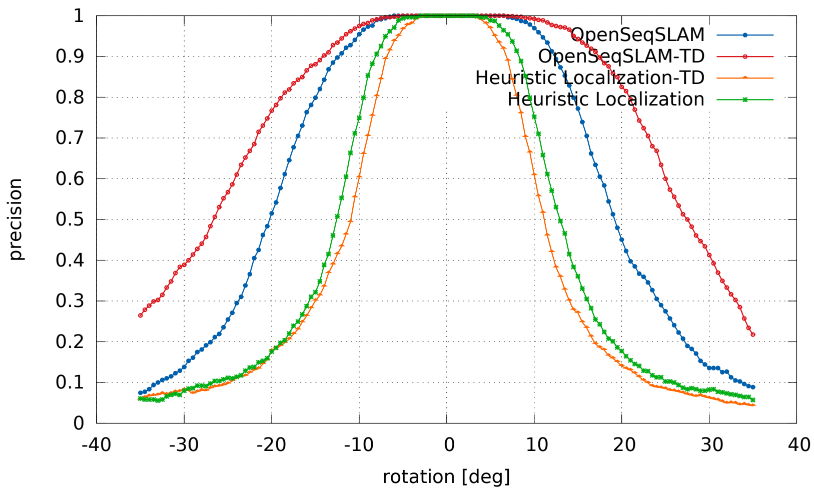
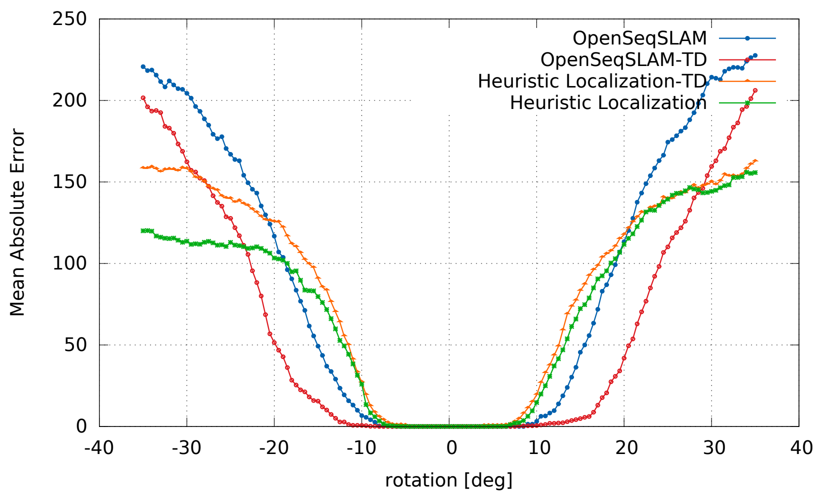


Figure 5.4.: Results of localization experiment using Nordland winter 8x6 images.

(a) 12x10 rotation MAE



(b) 12x10 rotation precision

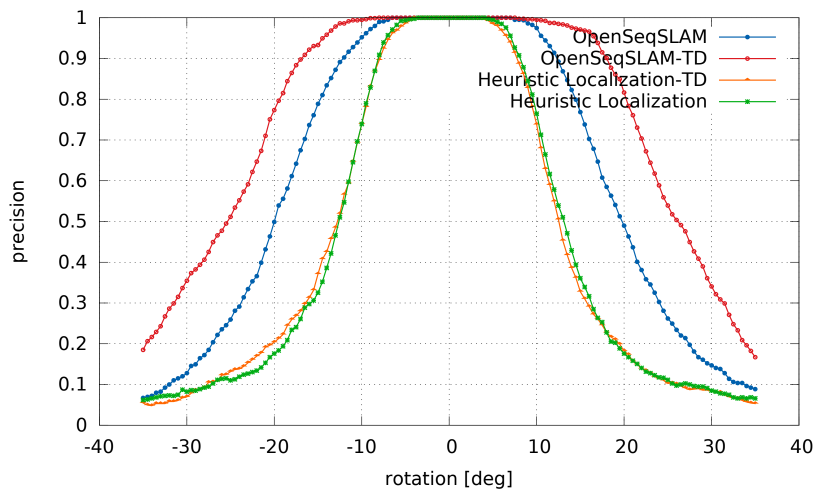


Figure 5.5.: Results of localization experiment using Nordland winter 12x10 images.

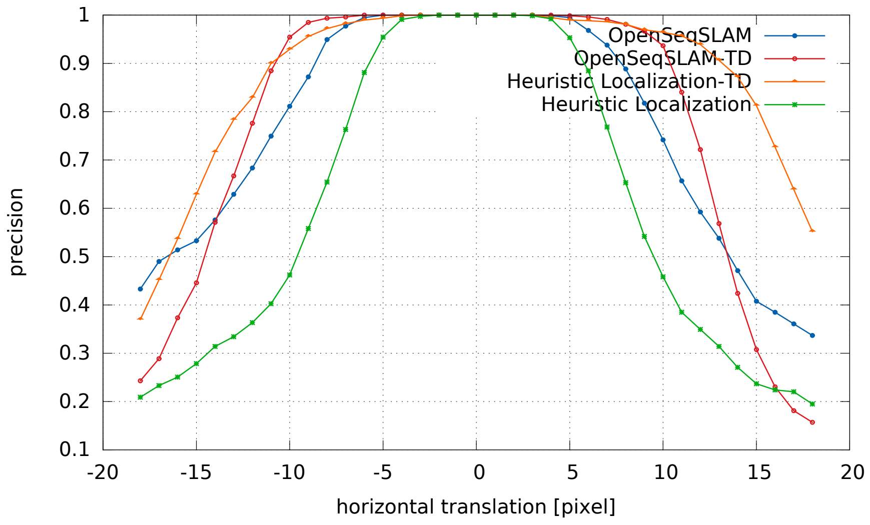
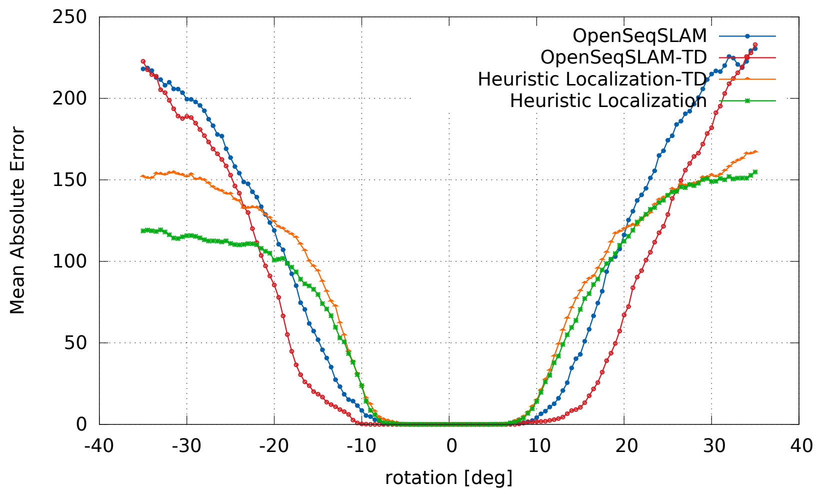


Figure 5.6.: Precision results for images with 12 pixels in width and 10 pixels in height for the horizontal translation transformation type. The heuristic online algorithm's precision is greater than that of the mean absolute difference algorithm.

(a) 18x16 rotation MAE



(b) 18x16 rotation precision

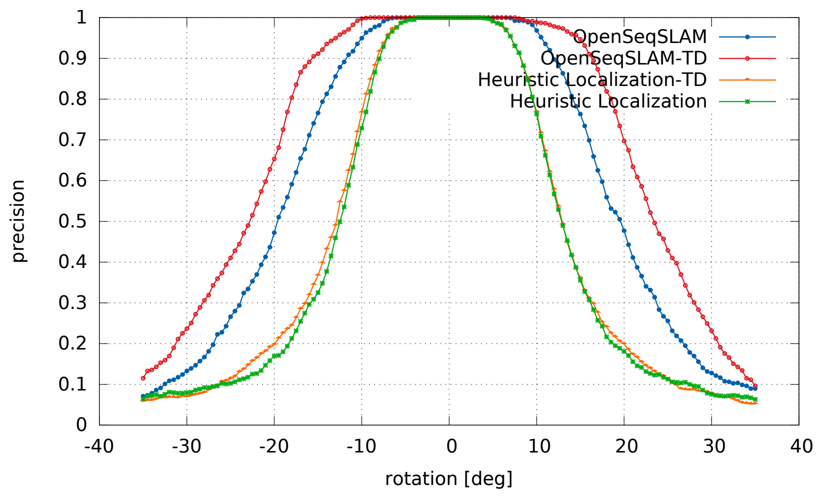


Figure 5.7.: Results of localization experiment using Nordland winter 18x16 images.

The numeric precision results listed in table 5.5 show that the rotation invariance has increased when compared to results of the lower image resolutions. All sampled precision intervals are larger than for lower image resolutions in case of the heuristic online algorithms using tangent distance. Also in case of the latter the mean absolute error has decreased in a similar manner, that is shown by the values listed in table 5.6.

Experimental results regarding the rotation transformation for images with 25 pixels in width and 20 pixels in height are shown in figure 5.8. It can be seen that for the all tested rotation angles the heuristic algorithm using tangent distance achieved greater precision. Also the mean absolute error of the tangent distance algorithm was smaller for rotation angles not exceeding 13 degrees in either direction compared to the heuristic

Table 5.5.: Excerpt of the precision results gained from the rotation experiment with images of 18 pixels in width and 16 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
$= 1.00$	$[-6.0, 6.5]$	$[-9.0, 7.5]$	$[-3.0, 3.5]$	$[-3.5, 3.5]$
≥ 0.98	$[-8.0, 9.5]$	$[-11.0, 11.5]$	$[-5.5, 5.5]$	$[-5.5, 6.0]$
≥ 0.95	$[-9.5, 10.5]$	$[-12.5, 14.5]$	$[-7.0, 7.0]$	$[-7.0, 7.0]$
≥ 0.90	$[-11.5, 11.5]$	$[-15.5, 16.0]$	$[-8.0, 8.0]$	$[-7.5, 8.0]$
≥ 0.80	$[-14.0, 14.0]$	$[-18.0, 18.5]$	$[-9.5, 9.5]$	$[-9.0, 9.5]$

Table 5.6.: Excerpts of mean absolute error results from the rotation experiment using images with 18 pixels in width and 16 pixels in height.

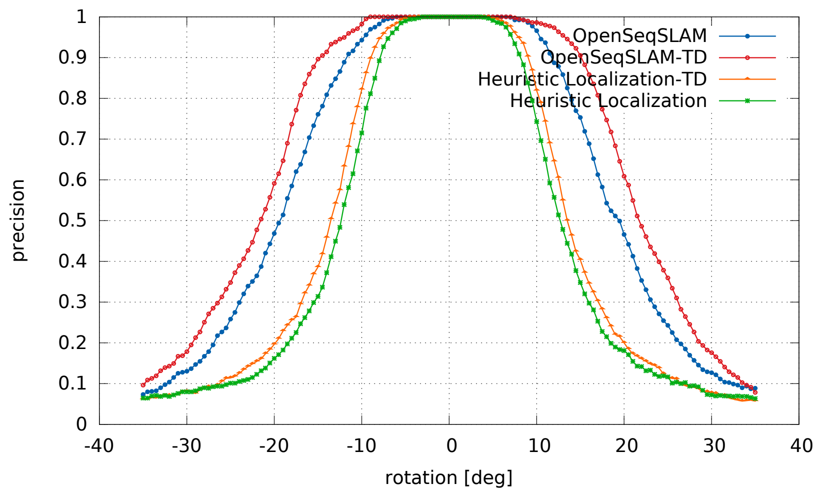
Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
-30.0	199.49	188.94	151.79	115.77
-20.00	118.96	85.49	124.18	100.77
-10.00	8.52	0.31	23.59	23.76
10.00	4.17	1.69	14.77	14.09
20.00	116.22	67.10	119.84	112.36
30.00	214.91	181.95	152.83	148.91

online algorithm that used the mean absolute difference.

The numeric results highlighting the achieved precision in table 5.7 show that the rotation invariance of the heuristic online algorithm using tangent distance has again increased compared to the lower image resolutions. For all listed precision values listed in the table the heuristic online algorithm using tangent distance does outperform the mean absolute difference variant. Also the precision of the heuristic online algorithm using the mean absolute difference is slightly reduced compared to the lower image resolution results. Similar observations can be made regarding the mean absolute error. Some numeric examples of this are listed in table 5.8 and it can be observed there that for the rotation angles between -10 and 10 degrees the tangent distance variant of the heuristic online algorithm has a lower mean absolute error compared to the lower image resolution results.

Experiments with images at a resolution of 32 pixels in width and 16 pixels in height proved to yield the best results for the heuristic online algorithm. This can be observed in figure 5.9. In case of the rotation transformation and combined with this image resolution the precision of the heuristic online algorithm using tangent distance exceeded the precision of the OpenSeqSLAM-MAD algorithm for rotation angles between -13 and 13 degrees. For the majority of tested rotation angles the mean absolute error of the heuristic online algorithm was lower than the error of the OpenSeqSLAM-MAD algorithm. The heuristic online algorithm using mean absolute difference remains nearly

(a) 25x20 rotation precision



(b) 25x20 rotation precision

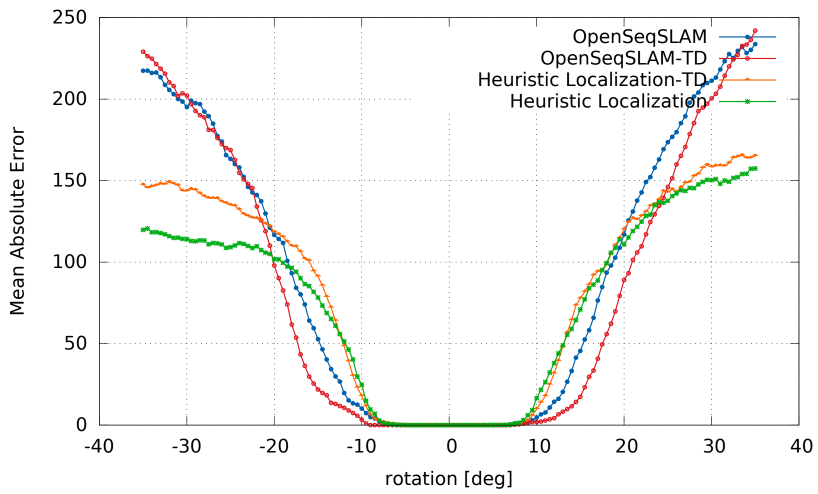


Figure 5.8.: Results of localization experiment using Nordland winter 25x20 images.

unchanged compared to the 25 pixels in width and 20 pixels in height resolution.

For images with 32 pixels in width and 16 pixels in height excerpts of the numeric precision results can be found in table 5.9. It can be seen that for the heuristic online algorithm using tangent distance the rotation angle interval in which a precision of 1 is achieved has doubled compared to the 25 pixels in width and 20 pixels in height resolution. The other listed intervals for smaller precision values also increased. Table 5.10 lists samples of the mean absolute error. A significant feature of the listed values is that in the rotation angle range of -10 to 10 degrees the heuristic online algorithm using tangent distance has the smallest mean absolute error of all tested algorithms.

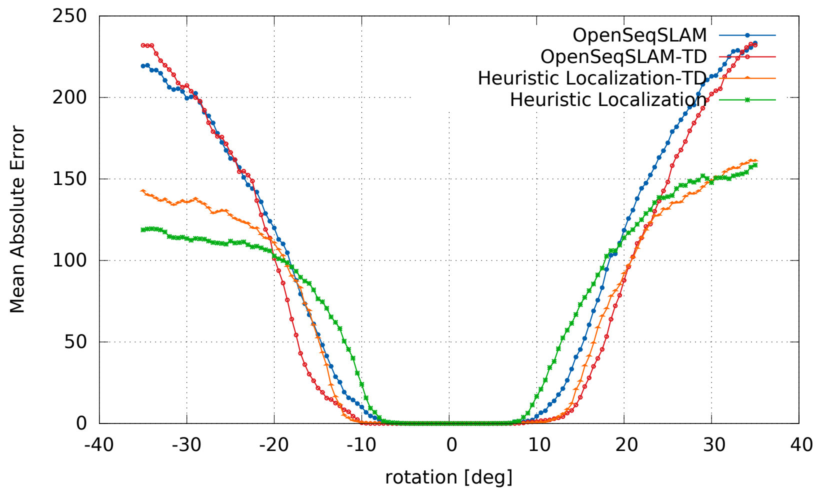
Table 5.7.: Excerpt of the precision results gained from the rotation experiment with images of 25 pixels in width and 20 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
$= 1.00$	$[-6.0, 6.5]$	$[-9.0, 7.0]$	$[-3.5, 4.0]$	$[-3.0, 3.5]$
≥ 0.98	$[-8.0, 9.0]$	$[-10.0, 10.5]$	$[-6.0, 6.5]$	$[-5.5, 6.0]$
≥ 0.95	$[-9.5, 10.0]$	$[-11.5, 13.0]$	$[-7.5, 7.5]$	$[-6.5, 7.0]$
≥ 0.90	$[-11.5, 11.5]$	$[-14.5, 15.0]$	$[-8.5, 9.0]$	$[-7.5, 8.0]$
≥ 0.80	$[-14.0, 13.5]$	$[-17.0, 17.0]$	$[-10.0, 10.0]$	$[-9.0, 9.0]$

Table 5.8.: Excerpts of mean absolute error results from the rotation experiment using images with 25 pixels in width and 20 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
-30.0	195.11	202.21	143.62	114.07
-20.00	116.67	97.96	118.47	101.81
-10.00	10.16	3.31	17.91	24.82
10.00	4.66	1.93	9.96	16.57
20.00	116.95	89.03	120.13	110.87
30.00	211.27	200.46	158.33	150.26

(a) 32x16 rotation MAE



(b) 32x16 rotation precision

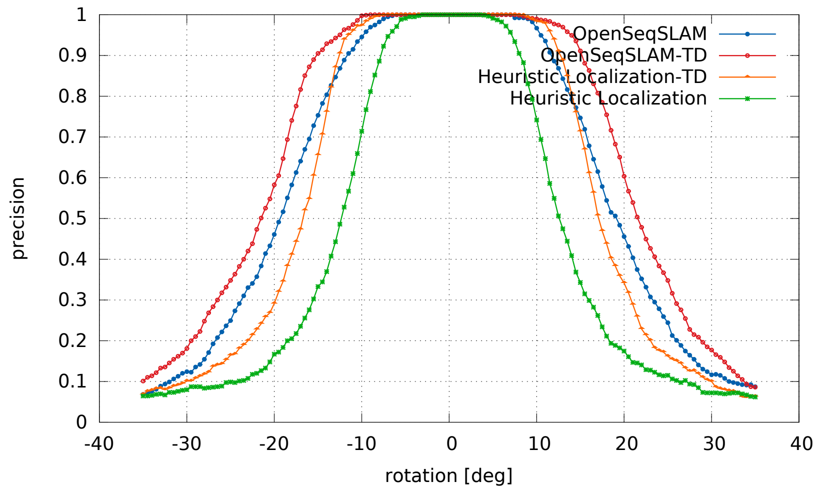


Figure 5.9.: Results of localization experiment using Nordland winter images 32x16.

Table 5.9.: Excerpt of the precision results gained from the rotation experiment with images of 32 pixels in width and 16 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
$= 1.00$	$[-6.0, 7.0]$	$[-9.0, 7.5]$	$[-7.0, 7.5]$	$[-3.0, 3.5]$
≥ 0.98	$[-8.0, 9.5]$	$[-10.5, 11.5]$	$[-9.5, 10.5]$	$[-5.5, 6.0]$
≥ 0.95	$[-9.5, 10.0]$	$[-12.5, 13.5]$	$[-11.5, 11.5]$	$[-6.5, 7.0]$
≥ 0.90	$[-11.5, 11.5]$	$[-15.0, 15.0]$	$[-12.5, 12.5]$	$[-7.5, 8.0]$
≥ 0.80	$[-14.0, 13.5]$	$[-17.0, 17.0]$	$[-13.5, 14.0]$	$[-9.0, 9.0]$

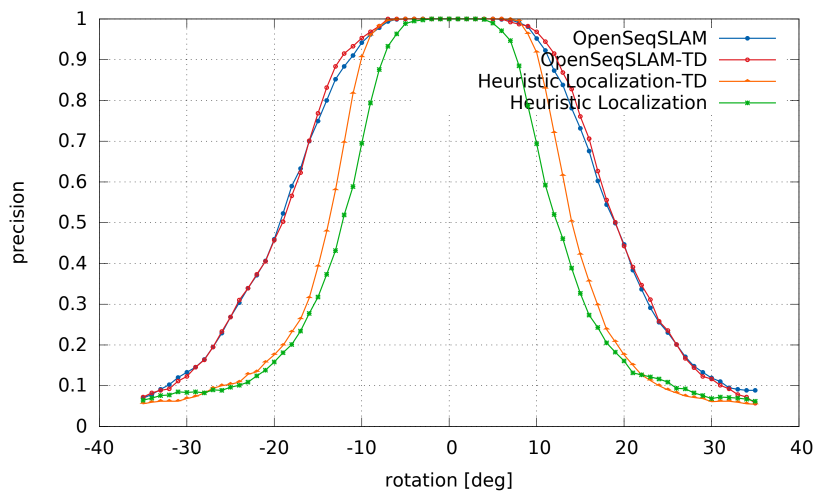
Table 5.10.: Excerpts of mean absolute error results from the rotation experiment using images with 32 pixels in width and 16 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
-30.0	199.51	207.34	135.38	113.45
-20.00	119.95	101.37	110.52	102.76
-10.00	10.11	0.44	0.44	24.03
10.00	4.40	1.17	0.35	16.63
20.00	118.52	87.76	91.68	113.91
30.00	213.00	202.15	148.63	147.75

The results of the experiments using the resolution of 64 pixels in width and 32 pixels in height are presented in figure 5.10. Like in the case of the OpenSeqSLAM algorithms for this resolution the overall precision dropped slightly compared to the 32 pixels in width and 16 pixels in height image results. Also the mean absolute error increased for all tested algorithms.

Numeric excerpts can help to illustrate the difference to the other image resolution results. These are listed in table 5.11 and table 5.12. The former lists again rotation angle intervals in which at least a stated precision is achieved and the latter gives samples of the mean absolute error at different rotation angles. Regarding the precision it can be said that the rotation angle intervals of the algorithms in which the listed precision values are not as wide compared to the results obtained at an image resolution of 32 pixels in width and 16 pixels in height. The developed heuristic online algorithm maintained a precision of 1 for a wider rotation angle interval than the OpenSeqSLAM-MAD and the Heuristic-MAD algorithm. From the values in table 5.12 it can be seen that it also achieved around -10 degrees and 10 degrees the lowest mean absolute error of all tested algorithms.

(a) 64x32 rotation precision



(b) 64x32 rotation MAE

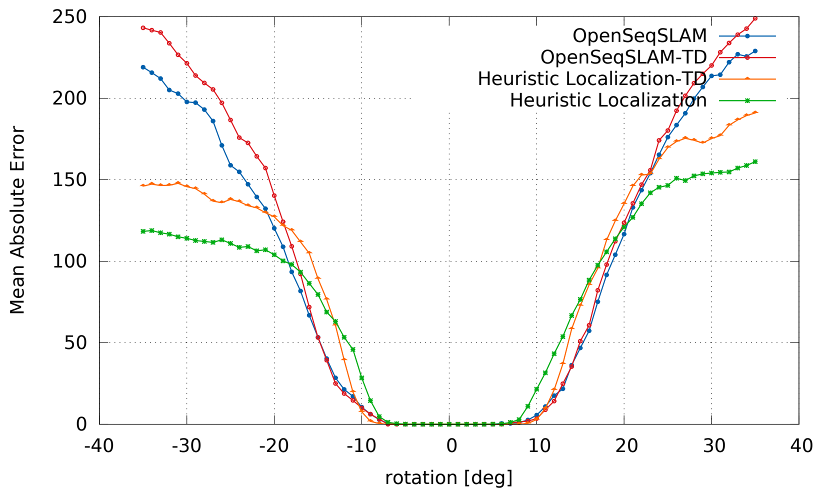


Figure 5.10.: Results of localization experiment using Nordland winter 64x32 images.

Table 5.11.: Excerpt of the precision results gained from the rotation experiment with images of 64 pixels in width and 32 pixels in height. The interval entries mark the rotation degrees range where the precision specified in the first column is achieved.

Precision	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
$= 1.00$	$[-5.0, 5.0]$	$[-7.0, 6.0]$	$[-5.0, 6.0]$	$[-2.0, 3.0]$
≥ 0.98	$[-7.0, 9.0]$	$[-8.0, 9.0]$	$[-8.0, 8.0]$	$[-5.0, 5.0]$
≥ 0.95	$[-9.0, 10.0]$	$[-10.0, 10.0]$	$[-9.0, 9.0]$	$[-6.0, 6.0]$
≥ 0.90	$[-11.0, 11.0]$	$[-12.0, 12.0]$	$[-10.0, 10.0]$	$[-7.0, 7.0]$
≥ 0.80	$[-14.0, 13.0]$	$[-14.0, 14.0]$	$[-11.0, 11.0]$	$[-8.0, 8.0]$

Table 5.12.: Excerpts of mean absolute error results from the rotation experiment using images with 64 pixels in width and 32 pixels in height.

Angle [°]	OpenSeqSLAM-MAD	OpenSeqSLAM-TD	Heuristic-TD	Heuristic-MAD
-30.0	197.73	221.46	145.71	114.03
-20.00	120.26	140.26	127.28	103.98
-10.00	10.52	10.02	7.59	28.42
10.00	5.63	3.45	2.93	21.49
20.00	116.62	123.55	135.25	120.92
30.00	213.70	220.11	175.29	154.13

5.5.1. Conclusion and Result Evaluation

In the previous section results of the experiments performed with all implemented localization algorithms are presented and compared. In this section a few general observations about the characteristics of these results are carved out. Also some possible explanations for these observations are attempted.

When looking at the results obtained at different image resolutions a general difference in recognition performance between algorithms that use the mean absolute difference and those that use the tangent distance can be observed. It appears that the algorithms using tangent distance are more affected by a change in resolution than the mean absolute difference algorithms. This can be observed by comparing the rate of change of the precision values per algorithm at different image resolution experiments. One example for this is the change in precision of the heuristic online algorithm between the 25 pixels in width and 20 pixels in height resolution and the 32 pixels in width and 16 pixels in height resolution when looking at the rotation transformation. In this case suddenly the rotation angle interval in which a perfect precision is achieved doubles. A possible explanation for this could be that the tangent distance only has a limited transformation range at which it effectively operates. The reason for this is laid out in section 3.2.2 but an extreme example can be given to demonstrate this again. Lets say we have a completely black image with one vertical, one pixel wide, white line in the middle going from the upper image border down to the bottom. If we now only consider the tangent vector for the horizontal translation operation. This is the horizontal image gradient and as result we get two lines parallel to the original line, immediately left and right of it marking the change in intensity. These two lines are also one pixel wide. If we want to create an image using the tangent vector that is like the original image only horizontally translated by 12 pixels to the left this cant really be done because the tangent vector is zero at these columns and no information of the original white line is present there. This is why, as described in the named tangent distance section, the Gaussian blur is applied to the tangent vectors to reduce this problem. In effect the Gaussian blur acts like an low pass filter on the image. This happens also when the image resolution is reduced. In the experiments for all resolutions the same Gaussian blur was applied but future experiments should consider testing a range of the performed Gaussian blur to optimize according to the image resolution.

Another observation that was made is that especially for the 8 pixels in width and 6 pixels in height image resolution, recognition performance was poor for the heuristic algorithm. A reason could be that the heuristic online algorithm relies to a great extend on the *first impression* of the first input image, because only a few candidates are selected according to this one. However once the details are reduced too far at such an image resolution, too many candidates of database images become similar and it would be better to rely more on image sequence information. However in the previous section it became also evident that the best recognition performance is not achieved with the highest tested image resolution that was 64 pixels in width and 32 pixels in height. Also it depended on the transformation type and the tested algorithm.

Surprisingly for some of the tested image resolutions and transformation types the heuristic approach using tangent distance outperforms the OpenSeqSLAM-TD algorithm so evidently sometimes *less is more.* In general the OpenSeqSLAM-TD achieved the best recognition performance. To summarize the highlighted aspects:

- heuristic approach fails when image resolution drops too low
- image resolution influences TD algorithm results more than MAD results
- heuristic outperformed in some cases the non heuristic approach
- best overall precision for OpenSeqSLAM-TD

5.6. Using saved Sensory Data and Active Movements

In this section further possibilities will be briefly outlined on how the tangent distance can be used in combination with sensory data and the abilities of the Myon or similar humanoid robots to possibly improve the place recognition performance. However the limited scope of this thesis did not allow to fully develop or experimentally evaluate the presented ideas. So this section merely details the general ideas and possible advantages that future research in this area might pick up on. In the previous section 4.1 it was mentioned that the image sequences recorded on the Myon are enriched with sensory data. This means that each single image of the image sequence is saved in combination with data obtained for example from sensors like accelerometers or readings from the robot's motors. This data can be used to infer the orientation of the robot's camera at the moment the associated image was recorded. This is possible at least up to a certain degree of precision depending on what type of sensor readings are available, the measurement error of these sensors and other influencing factors that can not be compensated when inferring the pose. However we assume for the following descriptions and examples that the orientation of the robot's camera can be sufficiently restored from saved sensor values, so that for a used image resolution no perspective difference between a pair of images, recorded assuming the real pose and the inferred pose, is noticeable. Also we assume the orientation of the robot's head to be defined by the three Euler angles roll, pitch and yaw, around the axes of a coordinate system originating in the middle of the robot's camera. It was shown in section 3.2.2 that when calculating the tangent distance of two images A and B, intermediate results, calculated right before applying the pixel based difference, are two sets of scalar values. Each of the scalar values per set is associated with a tangent vector. The scalar's magnitude indicates how much of the associated tangent vector should be applied in the total sum of tangent vectors that forms the approximated image. If we say the scalar values for the tangent vectors of the image A are organized in the vector $\vec{\alpha_A}$ and similar, the scalar values of the tangent vectors associated with image B are in $\vec{\alpha_B}$, then each scalar in $\vec{\alpha_A}$ indicates how to transform A to get an image more similar to B and conversely the scalar values of $\vec{\alpha_B}$ indicate how to transform the image B to get an image that is more similar to image A. For example if we say A is a rotated version of B by 15 degrees in clockwise direction then it can be expected that the scalar value associated with the tangent vector of the rotation operation is proportionally larger than the other scalar values. Fabrizio and Dubuisson examine the tangent distance in a similar fashion in order to estimate relative motion of objects depicted in different images. They describe this approach in [FD07] and successfully use the scalar values to determine the optical flow and the movement of objects. Building upon the example given above so that A is a rotated version of B, we can in addition say that A shall be a continuously updated live image that is obtained from the robot's camera and B is stored in the robots memory. This means the head is rolled by 15 degrees in difference to the pose that would result in a situation where the recorded image A equals B. By analyzing the resulting scalar values when calculating the tangent distance of A and B it is now possible to induce a slight rolling head movement in proportion to the scalar of the rotation tangent vector and in result this should decrease the difference between the current pose of the head and the pose associated with image B. If this behaviour is repeated in a control loop until the tangent vector scalar value of rotation drops below a certain threshold, the final pose of the head can be expected to be more close to the pose associated with image B. Also it can be expected that the tangent distance between A and B is smaller in the final position than in was at the starting position. A smaller tangent distance means of course greater similarity and recognition of input images should become more reliable. The above given example uses the rotation scalar value to adjust the roll of the robot's head camera. It should be possible to apply similar loops for the translations in vertical and horizontal direction by adjusting the pitch and yaw of the head respectively. The above stated properties of the tangent vector scalar values can be exploited for the use in the heuristic localization algorithm at least in two ways, these will briefly described in the following. After performing the greedy stage of the heuristic localization approach described in section 5.3.1 of this chapter, we have a set of k most promising candidate images identified in the database sequence and these images. Also for each of these candidate images and the input image the scalar values of the tangent distance is available. First it can be checked if a head movement according to these scalar values would result in a head pose that is more close to the pose that can be inferred from the saved sensory data of the candidate database images. According to the results the recognition score of each candidate images can be influenced, penalizing it if its not more similar. The second use is that the head can be actively moved according to the scalar values in a looping manner as described above and it can then be observed if the tangent distance decreases for the candidate images and this also should be reflected in the recognition score of each candidate image. After this refinement of the candidate image scores and before each following input image is recorded, the pose that can be inferred from the sensory data of the database image after the last candidate image can be assumed. This should result in a camera pose that increases the similarity of each following recorded input image. However it necessary to move the head in a separate pose for each candidate. The scheme is repeated until the end of the localization sequence is reached and this concludes the basic outline of possible uses for the tangent distance in combination with an image databased enriched with sensory information.

6. Conclusions

In the following sections, the findings and results of this thesis are concluded. Possible related future work is summarized.

6.1. Future Work

The experiments show that the tangent distance can indeed have a positive influence on the localization performance. The performed experiments however form only a limited set of possible test cases. While the performed tests analyze the four transformation types rotation, horizontal translation, vertical translation and scaling, other image transformation types should be included in future experiments to test if better transformation invariance and place recognition performance can be achieved. An example of one transformation is image shearing. Also experiments that include mixing of multiple image transformation types applied on the input images should be carried out. For example testing localization performance when rotation and scaling transformations are present at the same time. In addition to the transformation types other input datasets could be tested. These might include images with a greater variation in lighting conditions or change in scenery. On the other hand images recorded in environments that offer a small degree of variation in scenery should be tested. Good example images of the latter kind can be obtained for example in indoor office environments with long corridors in which humans would also struggle to recognize different places because walls, doors and edges look all very similar. Since in case of the experiments performed for this contribution, recognition performance variations were observed depending on image resolution, all future tests should be carried out with input image datasets differing in resolution as well. Also it might be possible that the optimal image resolution depends largely on the conditions or appearance of the environment in which the images are recorded.

Other parameters that can influence the performance of the algorithms using the tangent distance similarity metric are:

- local sequence size
- image database size
- Gauss convolution kernel size
- recognition threshold
- different tangent vector types

Moreover the heuristic algorithm should be tested with a number of item greedy candidate set sizes at different image resolutions. Another future aim is to adapt the integration of the heuristic online algorithm implementation into a concurrent processing environment where different tasks are executed. For example on the Myon tasks like audio processing or high level motion coordination tasks are running concurrently and context switches appear between them. Therefore rather than blocking the system until the complete localization algorithm has run through it would be desirable to offer a function in the module that returns the current best candidate of a recognized location for the current image input without any large processing overhead. This seems to be easily possible to implement since the current best location candidate can be determined simply by comparing the accumulated similarity scores of the candidate images collected during or after the greedy step of the heuristic algorithm. To increase the performance of the heuristic online localization algorithm the adaptive head or body movements according to the saved sensory data and results of the tangent distance should be correlated. Benefits and basic concepts of that are elaborated in section 5.6 of the previous chapter.

To test new or modified localization algorithms it is probably a good thing to use data from the target system. For example record image sequences directly on the Myon as was done for this contribution. To load and view this data and select parts of it for processing with these algorithms, the *DreamViewer* application can be extended to include these algorithms.

6.2. Conclusion

In this contribution new image sequence based place recognition approaches, based on the OpenSeqSLAM localization algorithm, were developed and experimentally evaluated. This has been done by incorporating the tangent distance image similarity metric to achieve greater perspective invariance. A main goal was to test the localization algorithm in an embedded human robotic system. To achieve this heuristic concepts were used to develop an algorithm more adapt to the capabilities of such systems. The results of the experiments showed that indeed the tangent distance did for many of the evaluated cases of perspective changes improve the place recognition or localization process performance. The OpenSeqSLAM algorithms were tested by using different data sets recorded directly on the Myon robot and in addition larger image sequences were used, obtained from openly licensed movies suitable for the purpose of testing. In addition a multi platform application software called *DreamViewer* was developed. This software was designed to test image sequence based localization algorithms by enabling the user to load various image sequences and select from these some arbitrary parts as inputs for the localization algorithms. This software was used to test the implemented OpenSeqSLAM algorithm variants. It is designed so that it can be easily extended to include other image sequence based localization algorithms as well. The developed heuristic algorithm variants were tested comparably to the OpenSeqSLAM variants with offline image data. In addition an implementation of the heuristic online algorithm using tangent distance was implemented in a small experiment on the Myon. By that the general concept was verified and tested. From the obtained results and the evaluation it can be concluded, that the tangent distance in case of the tested image transformations and image resolutions was superior for notable circumstances especially for cases of medium or small degrees of transformation. For the rotation transformation type, at an image resolution of 32 pixels in width and 16 pixels in height, the heuristic online algorithm using tangent distance achieved up to 8 percent higher precision than the OpenSeqSLAM-MAD algorithm for rotation angles between -15 and 15 degrees. So this image resolution can be recommended when using the heuristic algorithm. The OpenSeqSLAM-TD algorithm had greatest of precision of all tested algorithms over the complete rotation angle range. Further tests and investigations regarding the use of the tangent distance for image sequence based place recognition are therefore promising to achieve also desirable results. For that purpose a modest basis is provided by this contribution.

Bibliography

[ADL08] Henrik Andreasson, Tom Duckett, and Achim J Lilienthal. A minimalistic approach to appearance-based visual slam. *Robotics, IEEE Transactions on*, 24(5):991–1001, 2008.

[Bro13] K.K. Brock. *Image Processing in Radiation Therapy.* Imaging in medical diagnosis and therapy. Taylor & Francis, 2013.

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* The MIT Press Cambridge, 3rd edition, 2009.

[FD07] Jonathan Fabrizio and Séverine Dubuisson. Motion estimation using tangent distance. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 1, pages I–489. IEEE, 2007.

[GG96] Gerd Gigerenzer and Daniel G Goldstein. Reasoning the fast and frugal way: models of bounded rationality. *Psychological review*, 103(4):650, 1996.

[GG11] Gerd Gigerenzer and Wolfgang Gaissmaier. Heuristic decision making. *Annual review of psychology*, 62:451–482, 2011.

[GW10] R.C. Gonzalez and R.E. Woods. *Digital Image Processing.* Pearson Education, Limited, 2010.

[HSB$^+$11] Manfred Hild, Torsten Siedel, Christian Benckendorff, Matthias Kubisch, and Christian Thiele. Myon: Concepts and design of a modular humanoid robot which can be reassembled during runtime. In *Proceedings of the 14th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, Paris, France, September 2011.

[HSB$^+$12] Manfred Hild, Thorsten Siedel, Christian Benckendorff, Christian Thiele, and Michael Spranger. *Myon, a New Humanoid.* Springer, 2012.

[LJB$^+$95] Yann LeCun, LD Jackel, L Bottou, A Brunot, C Cortes, JS Denker, H Drucker, I Guyon, UA Muller, E Sackinger, et al. Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, 1995.

[Lug05] George F Luger. *Artificial intelligence: structures and strategies for complex problem solving.* Pearson education, 2005.

[MHF14] Ralf Möller, Michael Horst, and David Fleer. Illumination tolerance for visual navigation with the holistic min-warping method. *Robotics*, 3(1):22–67, 2014.

[Mil13] Michael Milford. Vision-based place recognition: how low can you go? *The International Journal of Robotics Research*, 32(7):766–789, 2013.

[MJCW13] Michael Milford, Adam Jacobson, Zetao Chen, and Gordon Wyeth. Ratslam: using models of rodent hippocampus for robot navigation and beyond. *International Symposium on Robotics Research*, 2013.

[MW10] Michael Milford and Gordon Wyeth. Persistent navigation and mapping using a biologically inspired slam system. *The International Journal of Robotics Research*, 29(9):1131–1153, 2010.

[MW12] Michael J Milford and Gordon Fraser Wyeth. Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1643–1649. IEEE, 2012.

[PBKE12] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. Real-time computer vision with opencv. *Communications of the ACM*, 55(6):61–69, 2012.

[RN10] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach.* Prentice Hall, Upper Saddle River, N.J. [u.a.], 3. ed. edition, 2010.

[SLCDV00] Patrice Y Simard, Yann A Le Cun, John S Denker, and Bernard Victorri. Transformation invariance in pattern recognition: Tangent distance and propagation. *International Journal of Imaging Systems and Technology*, 11(3):181–197, 2000.

[SNP13] Niko Sünderhauf, Peer Neubert, and Peter Protzel. Are we there yet? challenging seqslam on a 3000 km journey across all four seasons. In *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, page 2013, 2013.

[SNP14] Niko Sünderhauf, Peer Neubert, and Peter Protzel. Predicting the change–a step towards life-long operation in everyday environments. *Robotics Challenges and Vision (RCV2013)*, 2014.

[WZF05] Liwei Wang, Yan Zhang, and Jufu Feng. On the euclidean distance of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1334–1339, 2005.

A. Komische Oper Image Sequences

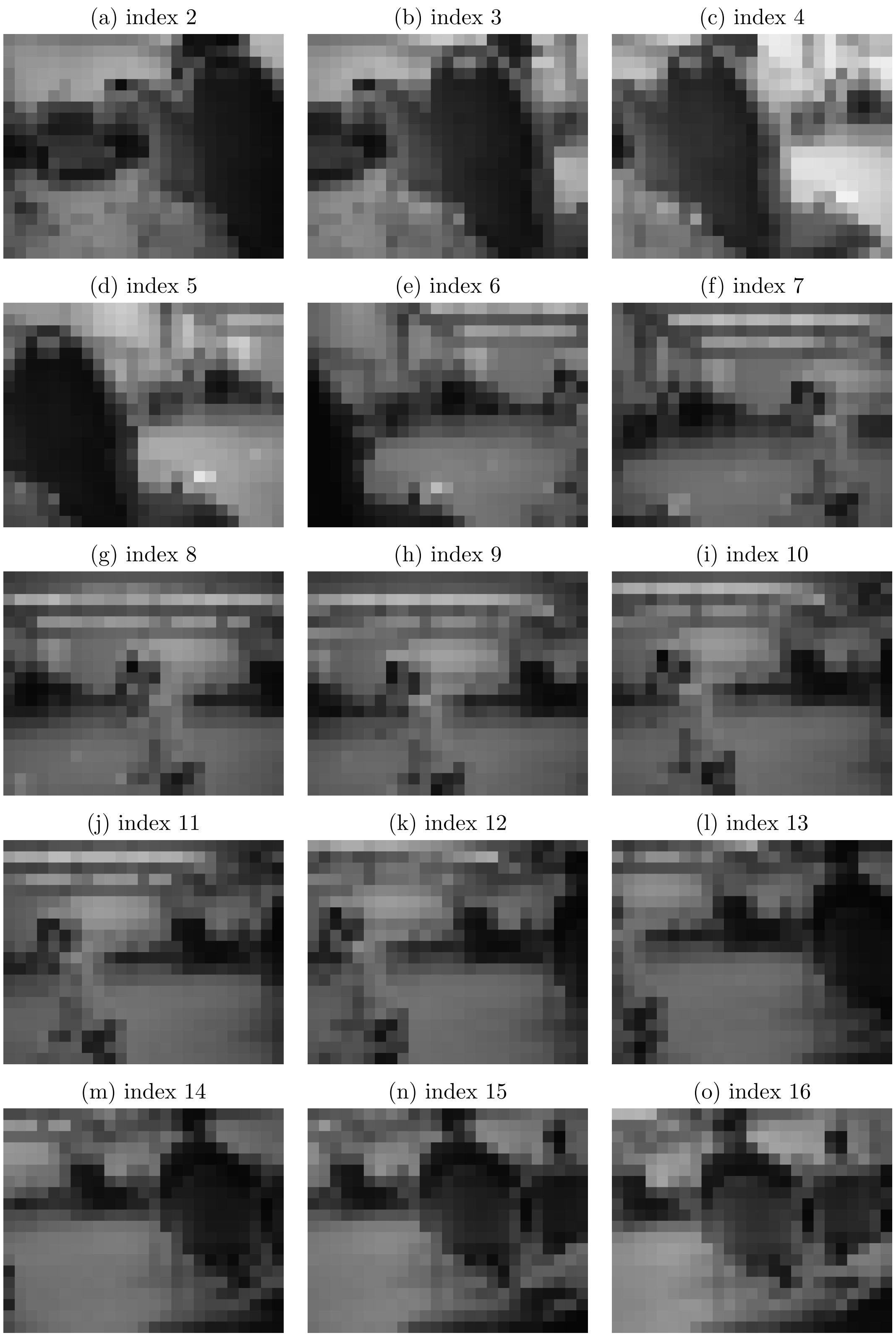


Figure A.1.: Database sequence images or *Pose 1* sequence.

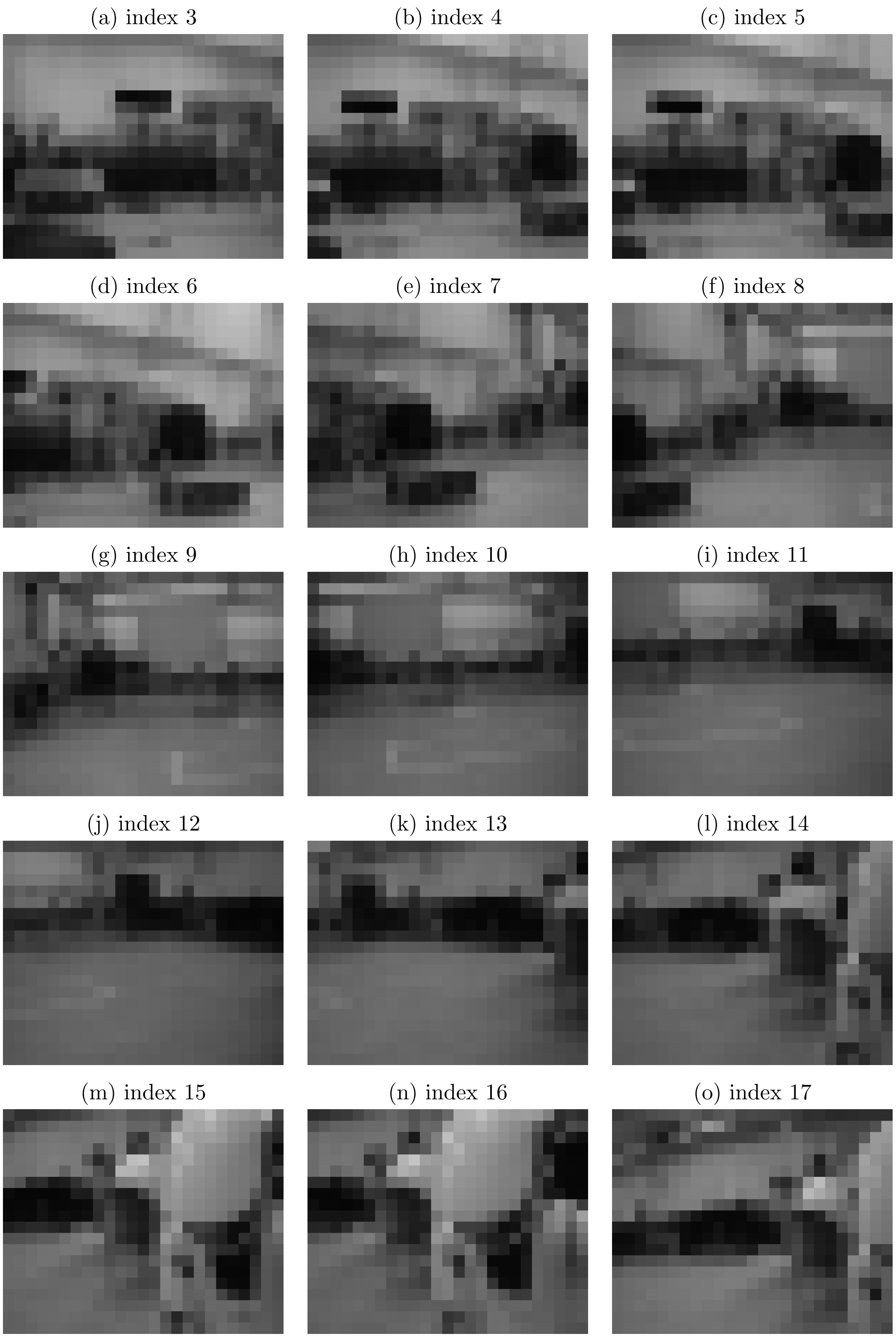


Figure A.2.: *Pose 2* image sequence.

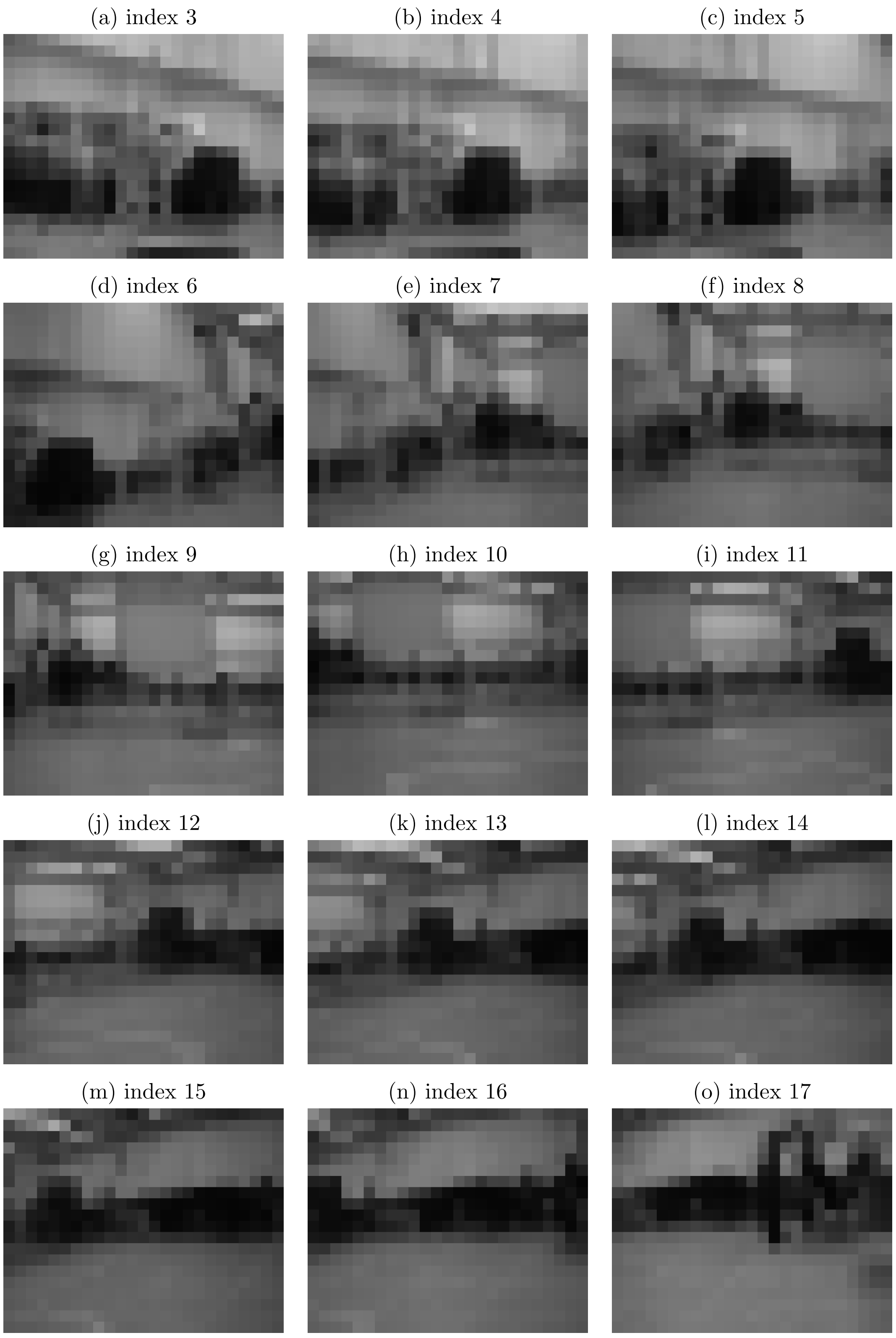
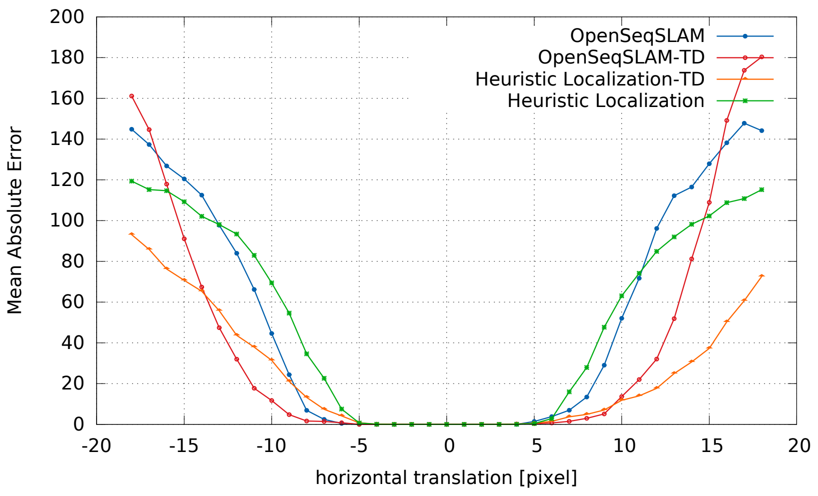


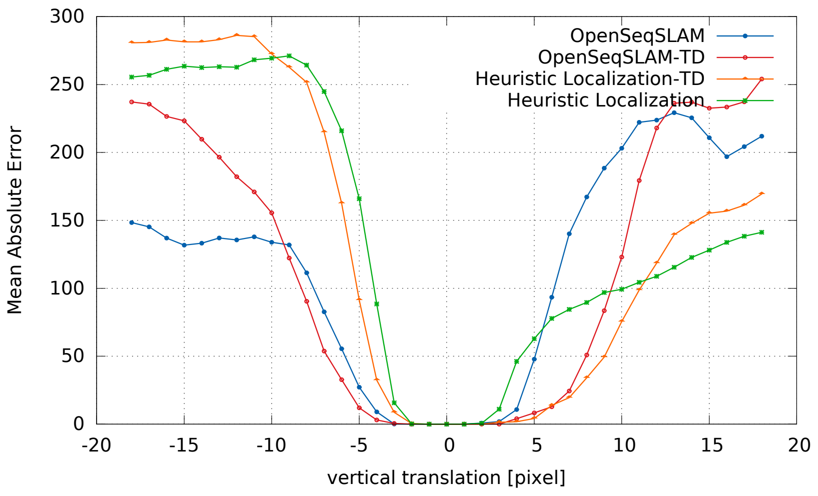
Figure A.3.: *Pose 3* image sequence.

B. Experimental Results

(a) 8x6 - translation x



(b) 8x6 - translation y



(c) 8x6 - scaling

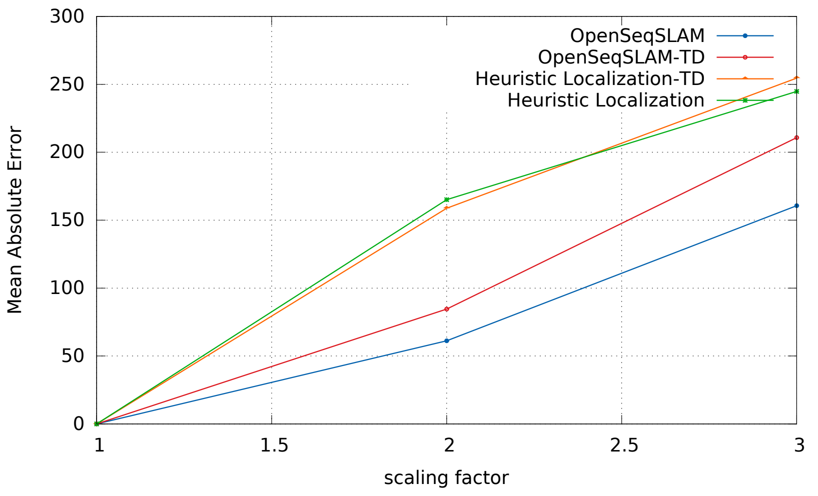
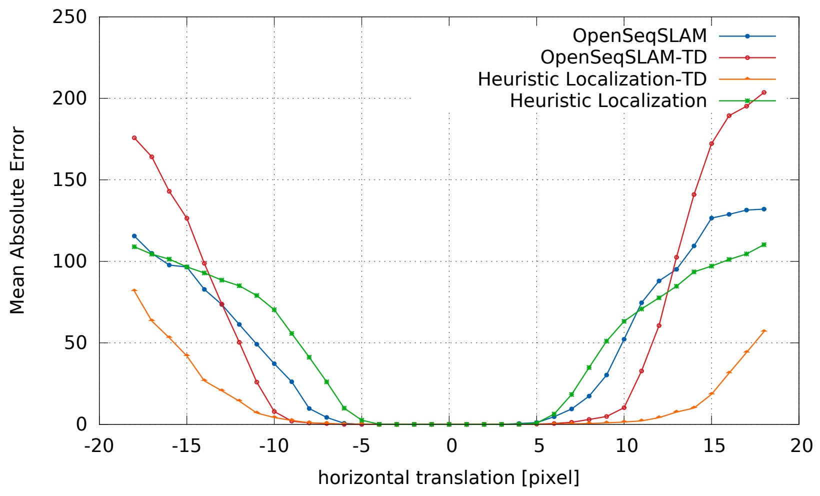
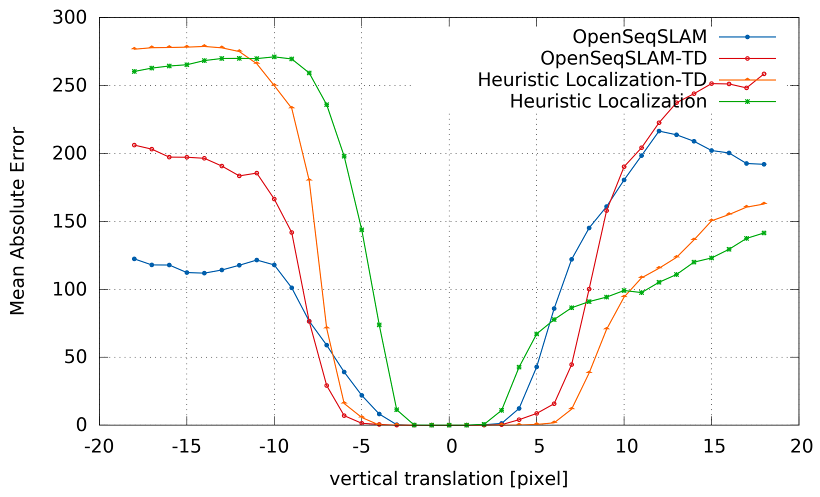


Figure B.1.: Experimental results mean absolute error for 8x6 images.

(a) 12x10 - translation x



(b) 12x10 - translation y



(c) 12x10 - scaling

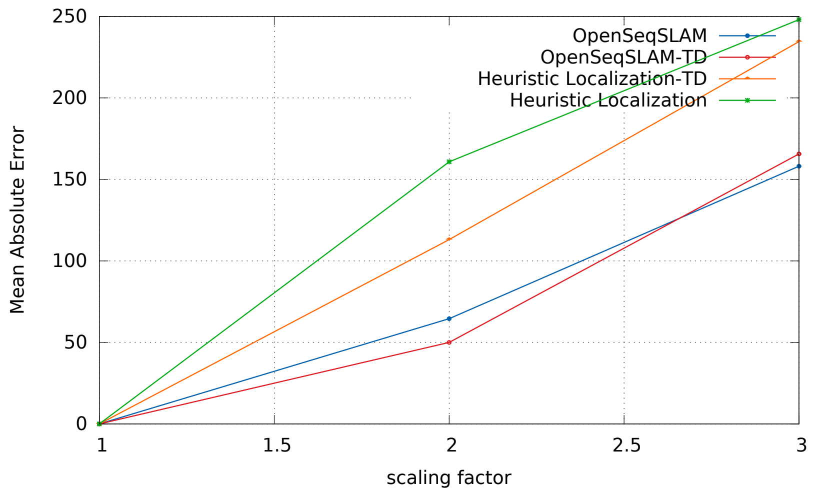
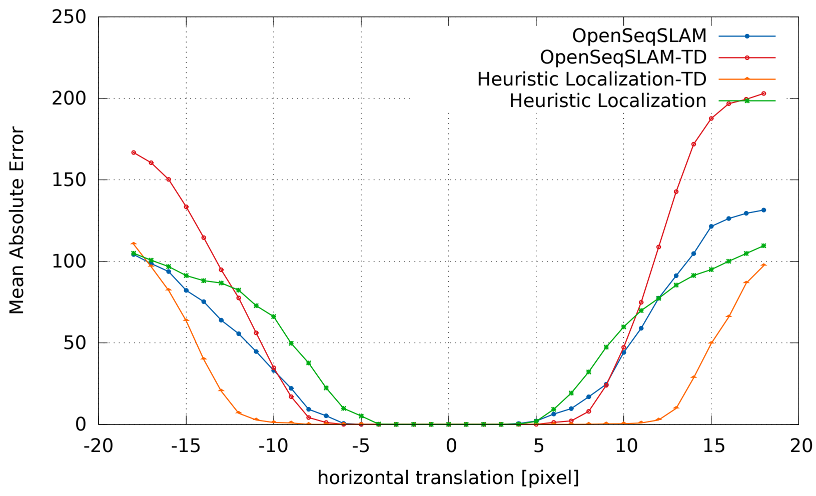
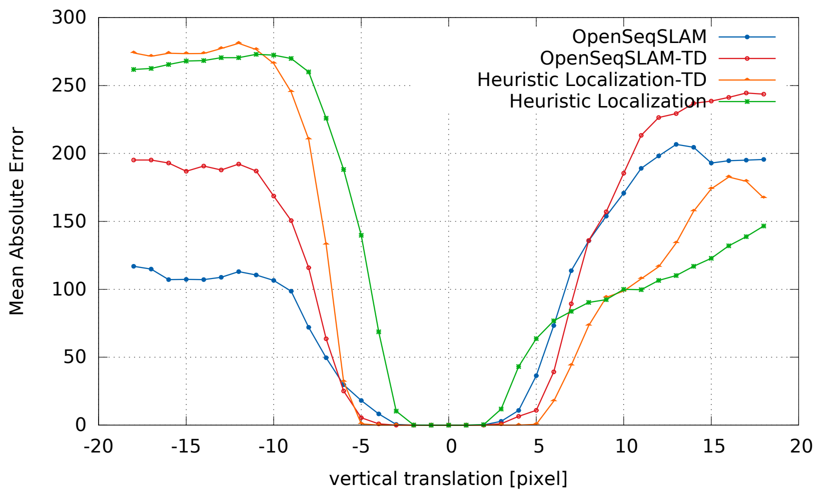


Figure B.2.: Experimental results mean absolute error for 12x10 images.

(a) 18x16 - translation x



(b) 18x16 - translation y



(c) 18x16 - scaling

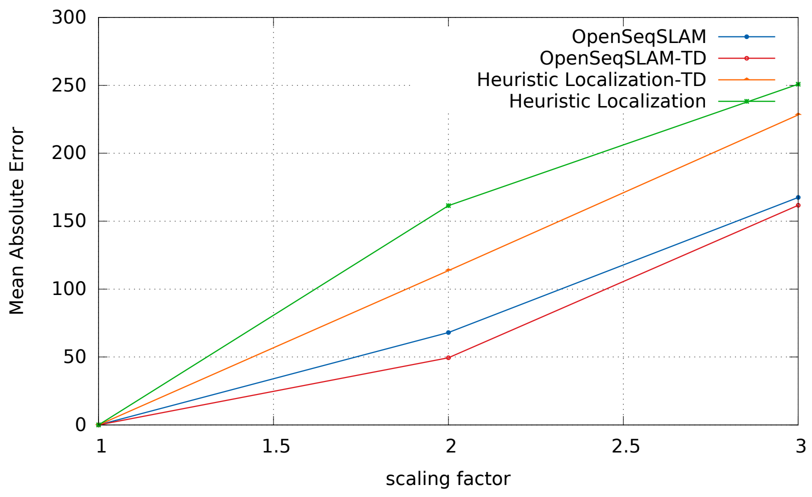
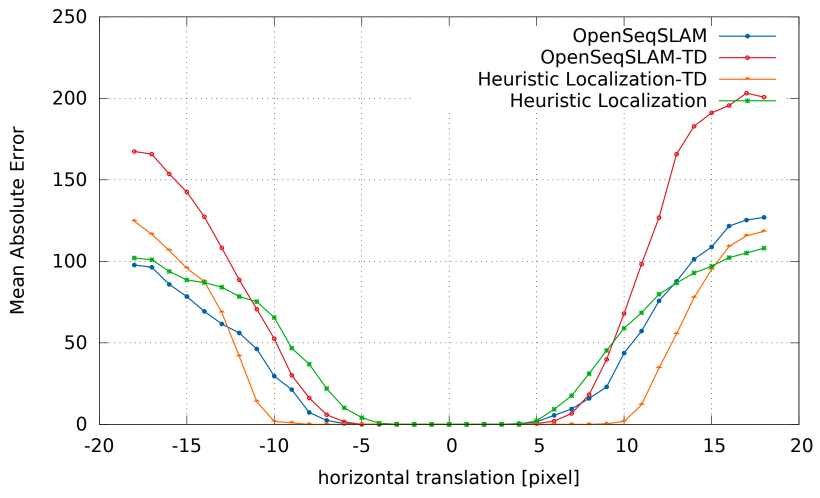
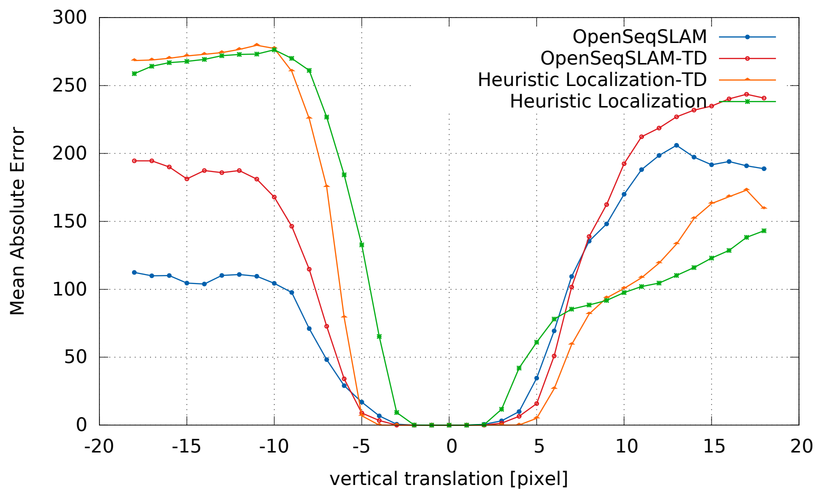


Figure B.3.: Experimental results mean absolute error for 18x16 images.

(a) 25x20 - translation x



(b) 25x20 - translation y



(c) 25x20 - scaling

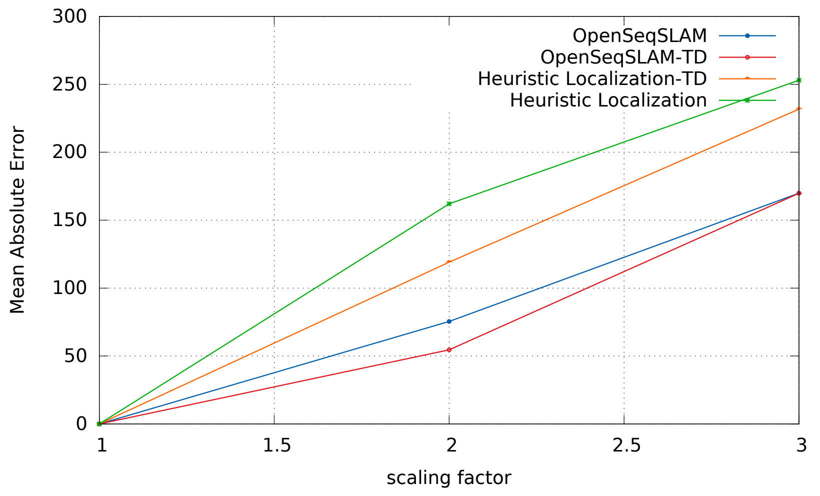
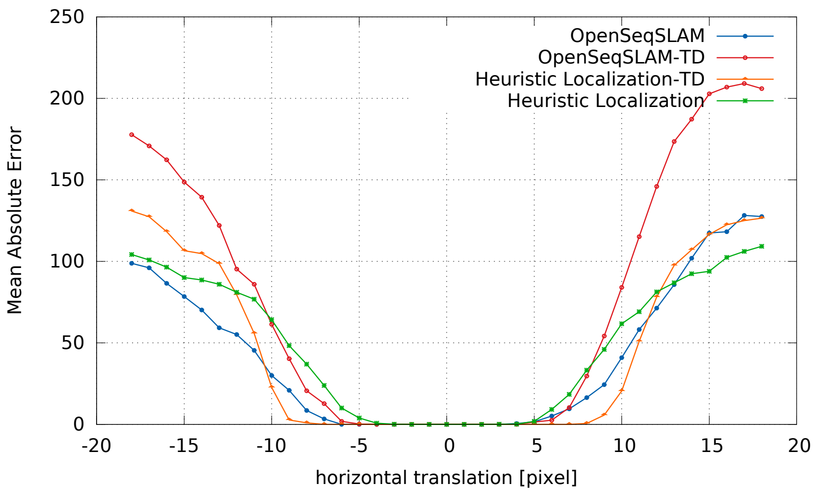
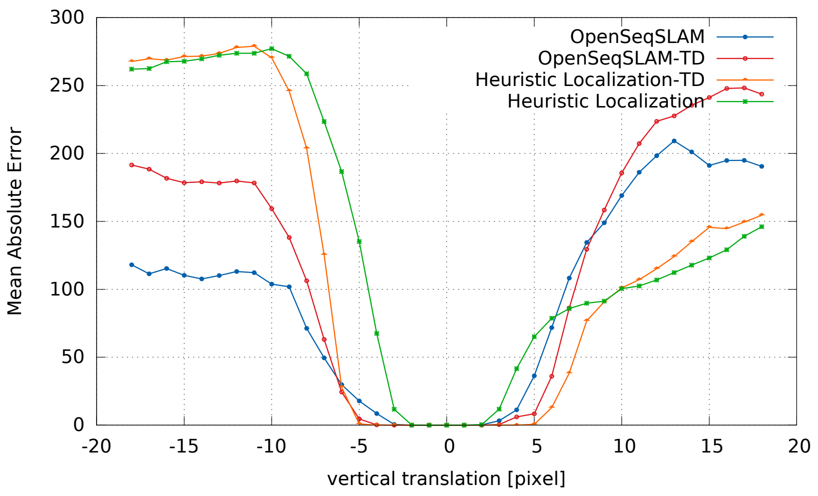


Figure B.4.: Experimental results mean absolute error for 25x20 images.

(a) 32x16 - translation x



(b) 32x16 - translation y



(c) 32x16 - scaling

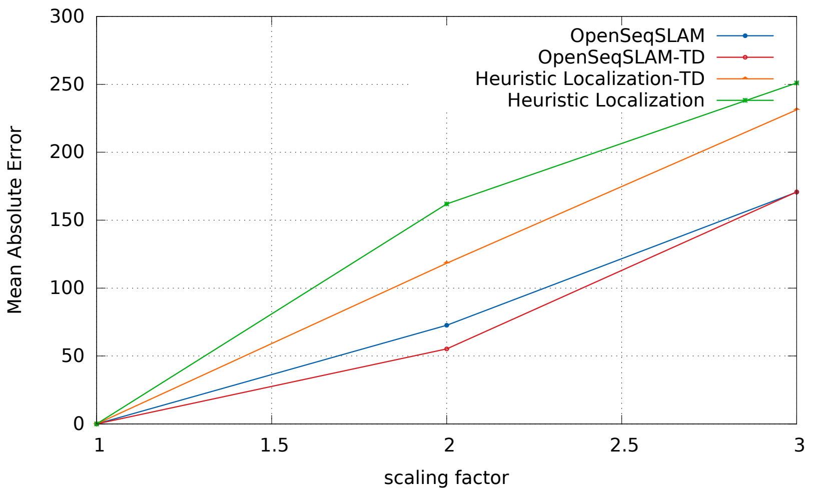
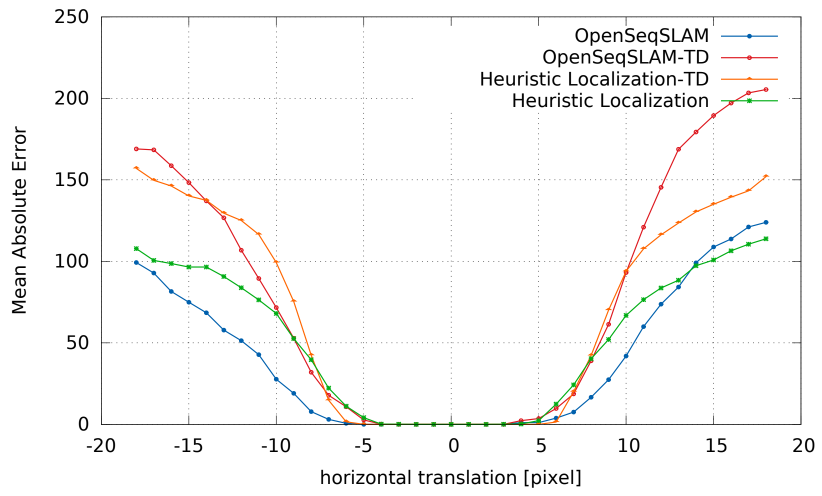
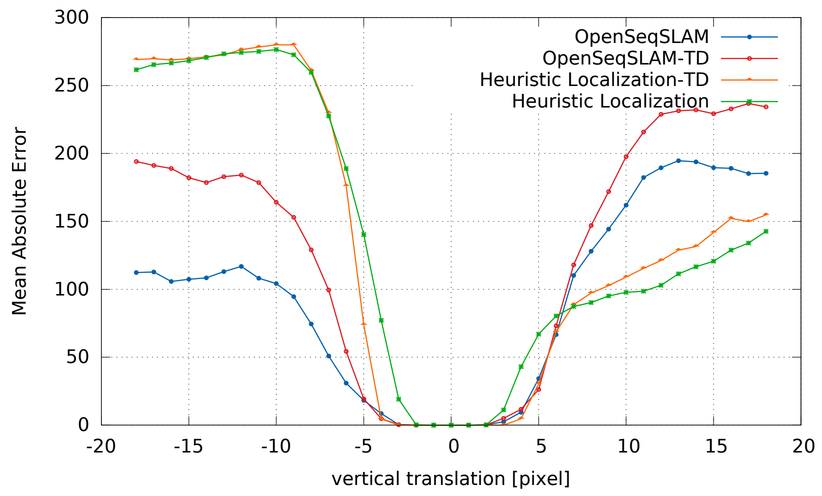


Figure B.5.: Experimental results mean absolute error for 32x16 images.

(a) 64x32 - translation x



(b) 64x32 - translation y



(c) 64x32 - scaling

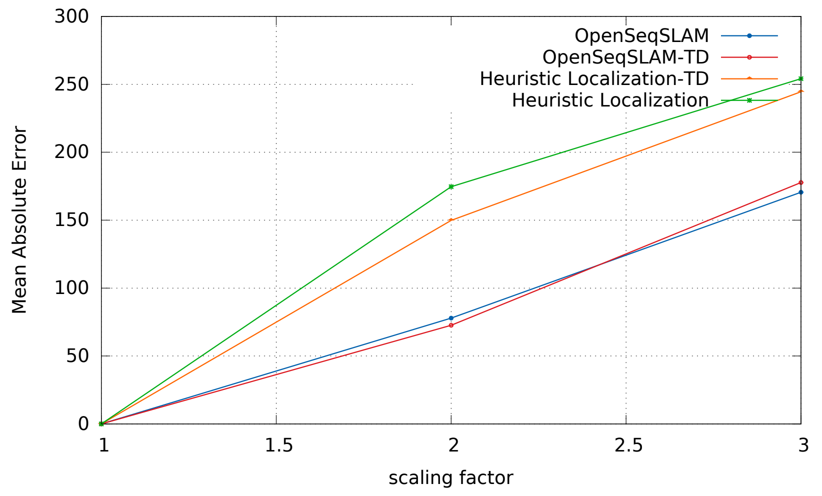


Figure B.6.: Experimental results mean absolute error for 64x32 images.