Institute for Computer Science
Knowledge-Based Systems

Bachelor's Thesis

Visual Features to Help Close the Loop in 6D-SLAM

Lars Kunze

October 2006

First supervisor: Prof. Dr. Joachim Hertzberg
Second supervisor: Prof. Dr. Martin Riedmiller

Abstract

One fundamental problem in autonomous mobile robotics is SLAM (*Simultaneously Localization and Mapping*): A robot has to localize itself in an unknown environment and at the same time generate a map of the surrounding area. Loop closing, i.e., detecting whether the robot has reached an area that it has visited before, is a substantial sub-problem of SLAM.

In this thesis, which is inspired by the work of Newman and Ho [23], visual features from camera data are used to help closing the loop in an existing 6D-SLAM architecture [31]. For the visual feature extraction two detection methods are combined, namely, salient region detection and maximally stable extremal region detection. The detected regions are then encoded using SIFT descriptors and stored in a database. The descriptors of different images are matched in order to detect a loop. To assess the approach, the feature detection and description will be compared to other methods within the framework developed by Mikolajczyk and Schmid [22].

Zusammenfassung

Ein fundamentales Problem in der autonomen mobilen Robotik ist das gleichzeitige Lokalisieren und Kartieren (engl.: SLAM; *Simultaneously Localization and Mapping*): Ein Roboter muß gleichzeitig in einer für ihn unbekannten Umgebung seine Position bestimmen als auch eine Karte dieser Umgebung erzeugen. Das Schleifenschließen ist ein Teilproblem vom SLAM. Hierbei soll der Roboter erkennen, ob er sich in einer Umgebung befindet, welche er zuvor bereits besucht hat.

In dieser Arbeit, die durch einen Aufsatz von Newman und Ho [23] inspiriert wurde, werden visuelle Merkmale aus Kameradaten extrahiert und dazu verwendet, das Schleifenschließen in einer existierenden 6D-SLAM Architektur [31] zu unterstützen. Bei der Extraktion visueller Merkmale werden zwei Methoden kombiniert, die Erkennung von interessanten Regionen (engl.: salient regions) und die Erkennung von maximal stabilen extremalen Regionen. Die erkannten Regionen werden mit Hilfe von SIFT Deskriptoren in einer Datenbank gespeichert. Die Deskriptoren von verschieden Bildern werden verglichen, um eine Schleife zu detektieren. Um den Ansatz zu beurteilen, werden sowohl die Merkmalserkennung als auch die Merkmalsbeschreibung in einem Evaluationssystem, welches von Mikolajczyk und Schmid [22] entwickelt wurde, getestet.

Contents

List of Figures

Chapter 1

Introduction

Loop Closing is the problem in the process of simultaneous localization and mapping where a robot comes to a place in the environment where it has been before. More precisely, there are two different problems in the task of loop closing: place recognition and knowledge integration. In place recognition, a robot recognizes if it has visited an area under inquiry before. Knowledge integration is the task of incorporating the extra information, that a robot gains from recognizing a place, into the localization and mapping data structures. Figure 1.1 makes the division in two problems clearer. On the left-hand side the path of a robot just before loop closing is shown. The two pictures in the figure are recognized as the same place. In the next step this extra information is integrated in the knowledge base of the robot. That means that the robot position and its corresponding uncertainty, which is illustrated with the gray ellipses, will be adjusted. This adjustment is shown on the right-hand side of the figure where the updated uncertainty of the robot position is illustrated with blue ellipses. This view on loop closing is not completely general, but it is a useful distinction criterion.

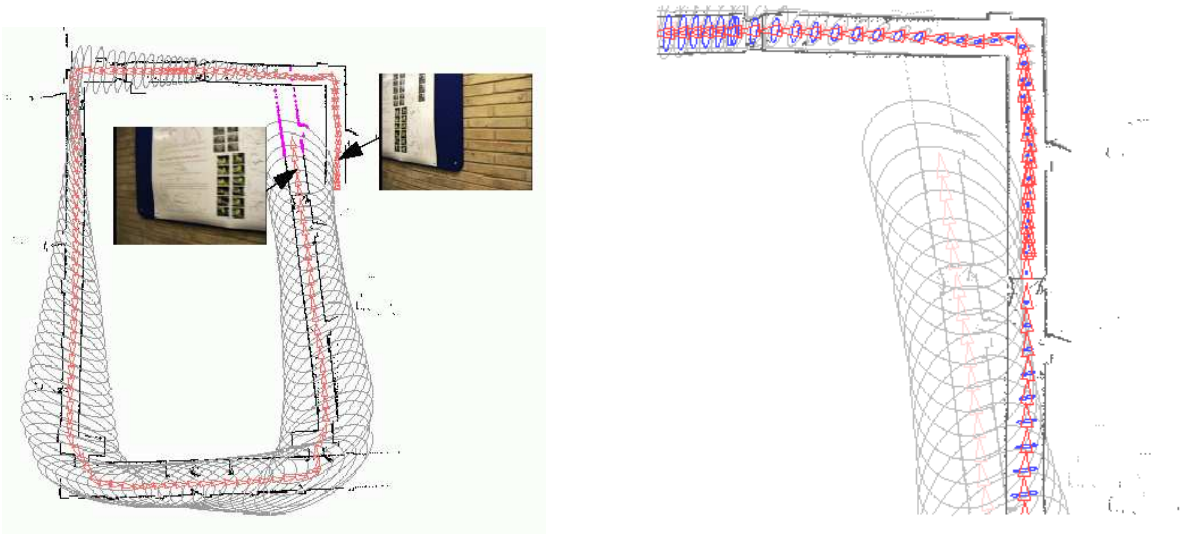
In this thesis only the problem of place recognition is addressed. The thesis is not concerned with the problem of knowledge integration. To recognize a place, a robot has to rely on some kind of sensor data. Visual information from camera data is examined here.

In the following section the thesis is put into relation to other present work. The scientific contribution of this thesis is explained in detail in Section 1.2.

1.1 Related Work

There is a variety of approaches that tackle the SLAM problem; some methods work in 2D – see [32] for an overview – while others do it in 3D [1,31]. The use of sensors is also different, but most approaches rely either on laser scanners or cameras. In this thesis, both kinds of sensors are used to combine their respective strengths. Here laser data is used as described in [31] to generate the maps of the environment. Visual features are extracted from the camera data and processed to improve the loop detection.

In order to make loop detection robust, Newman and Ho [23] suggest an approach that does not rely on the pose estimation of the robot to decide whether a loop closure is possible. Data for pose estimation, for example based on odometry, is typically erroneous. Therefore



(a) (b)

Figure 1.1: A robot trajectory before and after loop closing [23]. On the left: A robot path just before loop closing. The state vector of the robot positions is illustrated in red. The gray ellipses around these positions indicate the uncertainty of the position. The two pictures show the same scene from different viewpoints. On the right: A close-up on the robot path after loop closing. The state vector is adjusted and the uncertainty of the robot position is updated (blue ellipses).

loop closing is improved by not considering this kind of measurement. Searching over all poses means more or less solving the *kidnapped robot problem*[1], one of the three kinds of localization distinguished by [26][2]. Using only visual information as proposed by [23] shows difficulties, for example in environments with recurring structures; therefore Newman and Ho [5] combined visual and spatial information acquired from camera and laser data, respectively, to enhance their developed procedure. This is an interesting approach, but beyond the scope of this work. Here only visual data is considered for the loop detection task. Laser data is only used for the purpose of map building.

Camera data for loop detection is used by extracting features from the captured images and processing them. This is normally done in three steps: detecting the features, encoding them using feature descriptors and finally matching them against each other. Many region detectors available [8, 15–17, 19, 21]. There are also quite a number of feature descriptors, which differ in their properties [3,4,10–13,28]. Important issues are invariance to scale, rotation, transformation or changes in illumination. For an overview over both detectors and descriptors see [20]. In the present thesis, salient regions are detected and encoded with SIFT descriptors as presented by [14]. For comparison, other feature descriptors have been tested as well.

[1] A robot is moved to an unknown position in a known environment, where it has to localize itself

[2] Just for completeness the others are *tracking* and *global localization*. But it is the *kidnapped robot problem* that is mostly related to this thesis.

1.2 Objective and Structure of the Thesis

The main aim of the thesis is to implement and evaluate the procedure suggested by [23] which tries to solve loop closing by using information from camera data, or more specific by using visual features. The proposed feature detector is compared with other detectors within the framework developed by [22]. Furthermore the procedure is integrated into the existing 6D-SLAM architecture [31] on the Kurt3D robot platform to assist the present algorithms regarding loop closing. The intentions above are leading to three general parts in this thesis:

Developing the procedure for feature extraction and loop detection. The procedure is implemented as described in [23]. The authors suggest a feature detector with two criteria, namely saliency and wide-baseline stability. The features are encoded using a SIFT descriptor [14] and stored in a database. For the loop detection the features of a new image are processed and compared with the features already stored in the database. If there is an overlap of features, a loop hypothesis is generated. The methods used for the visual feature extraction, the feature matching and the application of loop closing are described in Chapter 2.

Evaluation of the feature detection and description within a given framework [22]. The combined feature detection of salient regions [9] and *maximal stable extremal regions* (MSERs) [15] is compared with other detection methods. Especially the exclusive usage of salient regions or MSERs is interesting. Other feature descriptors apart from SIFT are tested, too. The data set that is used, the evaluation criteria and the evaluation results are presented in Chapter 3.

Integrating the loop detecting procedure into the existing 6D-SLAM architecture. The loop detecting procedure assists the robot, while it is exploring the environment (indoor and outdoor). If a loop is detected, the robot tries to verify it based on independent sensor data, e.g., 3D scans, or self-localization. The robot architecture and the testing results are explained in Chapter 4.

Chapter 5 discusses the results, draws some conclusions and addresses open issues.

Chapter 2

Loop Closing

Loop closing means to detect if a robot reaches a position in the environment where it was before. In general, the choice of sensors to tackle this problem is not limited. Camera sensors have the advantage that they are quite cheap, handy and widely-used. Therefore this implementation focuses on visual information for the task of loop closing. Visual features are extracted from images made by the robot while driving around. These features are stored in a database and matched against features extracted from previously taken images.

2.1 System Overview

The principal design of the loop closing procedure is depicted in Fig. 2.1. Images are taken from the robot in an incremental fashion. These images are applied to the saliency-MSER-SIFT pipeline one at a time and one after another. The result of the pipeline, the extracted features, are stored in a database. After the first image is processed, the resulting features of each following image are matched against all features that are already stored in the database to detect a loop. The matching of the features is equivalent to the measurement of the distance between vectors in a high dimensional vector space. A loop closing hypothesis is generated if similar features are found in two images, that means that their distance is below a certain threshold. In other words, it is likely that the images were taken at the same or neighboring location.

There are three steps needed in the visual feature extraction pipeline to generate the feature representation that is stored in a database and used for the matching process: the saliency detection, the MSER detection and the SIFT description. Two criteria are used to detect the features in the image, namely saliency and wide-baseline stability (MSER detection). The detection of the features is fulfilled by steps one and two. Each detection process generates a list of feature regions. The overlapping feature regions are considered for further processing. An example of the feature regions detected by its corresponding method as well as the resulting overlapping regions is shown in Figure 2.2. The sample images in the figures of this chapter are all taken from a data set that is publicly available on the website `http://www.robots.ox.ac.uk/~vgg/research/affine/`.

These regions are encoded in the third step with the SIFT description algorithm. The repre-

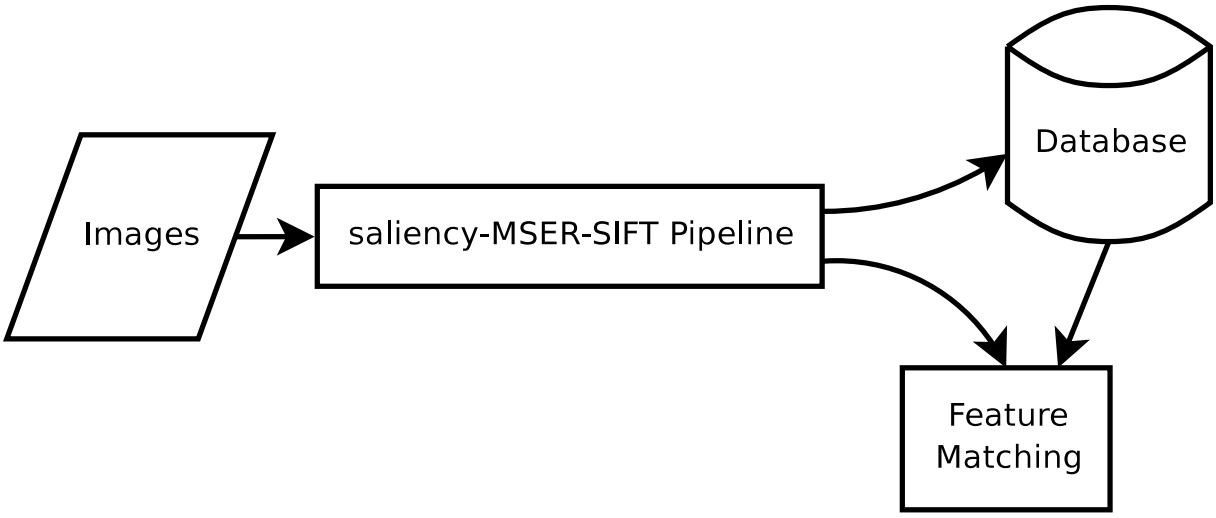


Figure 2.1: The principal design of the loop closing procedure. Images are applied to the saliency-MSER-SIFT pipeline. Salient and wide-baseline stable features are detected and encoded using the SIFT algorithm. The resulting feature descriptors are stored in the database and matched against previously generated descriptors.

sentation of visual features with SIFT descriptors is chosen because they are highly distinctive and at the same time very efficient for comparisons.

The algorithms of the visual feature extraction pipeline, i.e., the feature detection and description, are presented in Section 2.2. Furthermore the matching process is explained in more detail in Section 2.3 and finally the application of loop closing is described in Section 2.4.

2.2 Visual Feature Extraction

Here the task of visual feature extraction is to represent an image by a set of features. This problem consists of two sub-problems: feature detection and feature description. Feature detection finds regions in an image that meet some criteria. Feature description encodes these regions to have a representation that is used for further processing, e.g., feature matching.

As mentioned above, in this implementation three general steps are needed to generate the representation of features for an image. These steps, the saliency detection (2.2.1), the MSER detection (2.2.2) and the SIFT description (2.2.3), are described in the following. It is noteworthy that this implementation combines two different feature detection methods to make the approach more robust.

2.2.1 Salient Region Detection

Saliency is the first criterion for feature detection. It can be best understood as characteristic for 'interesting' regions. Presumably those regions are relatively sparse in an image. That makes this metric useful for loop detection, because features are more or less unique in an image and,

Figure 2.2: The original scene and the results for the saliency, the MSER and the combined saliency-MSER feature detection for the same image. The image shows a structured graffiti scene from the data set that is shown in Figure 3.1. The detected visual features are highlighted: Salient regions in the top right image and MSERs in the bottom left image. These regions are found by their corresponding feature detectors. The overlapping regions from these both images are shown on the bottom right image. Regions are considered overlapping if the number of the intersecting pixels is at least 10% of the number of unified pixels. The ellipses in the right images are calculated from the convex hulls of the MSER regions.

accordingly, for a location.

The scale-saliency algorithm developed by Kadir and Brady [9] was used by Newman and Ho for loop closing [23]. It detects image regions having a local complexity or, in other words, are distinct from their neighborhood. This complexity or distinctiveness is measured as the entropy H_D for a certain region D. A region is a scaled circular window around a center pixel $\vec{x}$. The window size or scale s is bound to a range between a minimum and a maximum scale value. Pixels and their values within the region are denoted with d_i. The probability density function for region D is $P_D(s, \vec{x})$, it returns the probability for a certain value of d_i in its corresponding region. The following equation is used to calculate the distinctiveness for each image pixel and for different scales:

$$H_D(s,\vec{x}) = -\int_{i\in D} P_D(s,\vec{x})\log_2 P_D(s,\vec{x})d_i. \tag{2.1}$$

In order to select only these scales that contribute most to the result, the entropy measure is weighted. The weight puts more emphasis on scales where the entropy significantly changes in respect to their next neighbor scales. The rate of change of the probability density function $P_D(s,\vec{x})$, multiplied with the scale s, meets the needs as weighting factor:

$$\mathcal{W}_D(s,\vec{x}) = s\int_{i\in D}\left|\frac{\partial}{\partial_s}P_D(s,\vec{x})\right| d_i. \tag{2.2}$$

Thus, the overall metric for salient regions $\mathcal{Y}_D(S,\vec{x})$ is the described entropy H_D multiplied with the weighting factor $\mathcal{W}_D(s,\vec{x})$:

$$\mathcal{Y}_D(S,\vec{x}) = H_D(S,\vec{x}) \times \mathcal{W}_D(S,\vec{x}). \tag{2.3}$$

Here is an informal description of the scale-saliency algorithm:

First, regions of interest (ROIs) for all scales are pre-calculated. The algorithm works over different scales, therefore for each scale a set of pixels is calculated that is used as a template of the neighborhood of an arbitrary pixel.

Second, the principal part of the scale-saliency algorithm is performed: What follows is done for each pixel in the image. For each scale the local descriptor values are sampled with the help of the pre-calculated ROIs, the probability density function is estimated using histograms and the entropy H_D and weights $\mathcal{W}_D$ are calculated. Then the weights are smoothed and the entropy peaks are determined. For each peak the salient metric $\mathcal{Y}_D(S,\vec{x})$ is calculated and stored in a list of salient pixels.

Finally the salient pixels are clustered and only the centroids of the clusters remain in the list. Figure 2.3 shows the regions that are labeled salient by scale-saliency algorithm.

Here is the scale-saliency algorithm in pseudo-code [7]:

Algorithm 1 Scale-saliency algorithm

```
for all pixel locations, (x,y), in the image I do
    for all scales, s, between Smin and Smax do
        IS = Sample local descriptor values at I(x,y) in a window of size s
        P(d,s) = Estimate the local PDF from IS {e.g. using histograms}
        HD(s) = Calculate the entropy of P(d,s)
        WD(s) = Calculate inter-scale saliency between P(d,s) and P(d,s-1)
    end for
    Run smoothing filter over WD(s) {e.g. 3-tap average}
    for all scales for which the entropy attains a peak, SP do
        YD(SP,x,y) = HD(SP) x WD(SP)
    end for
end for
```

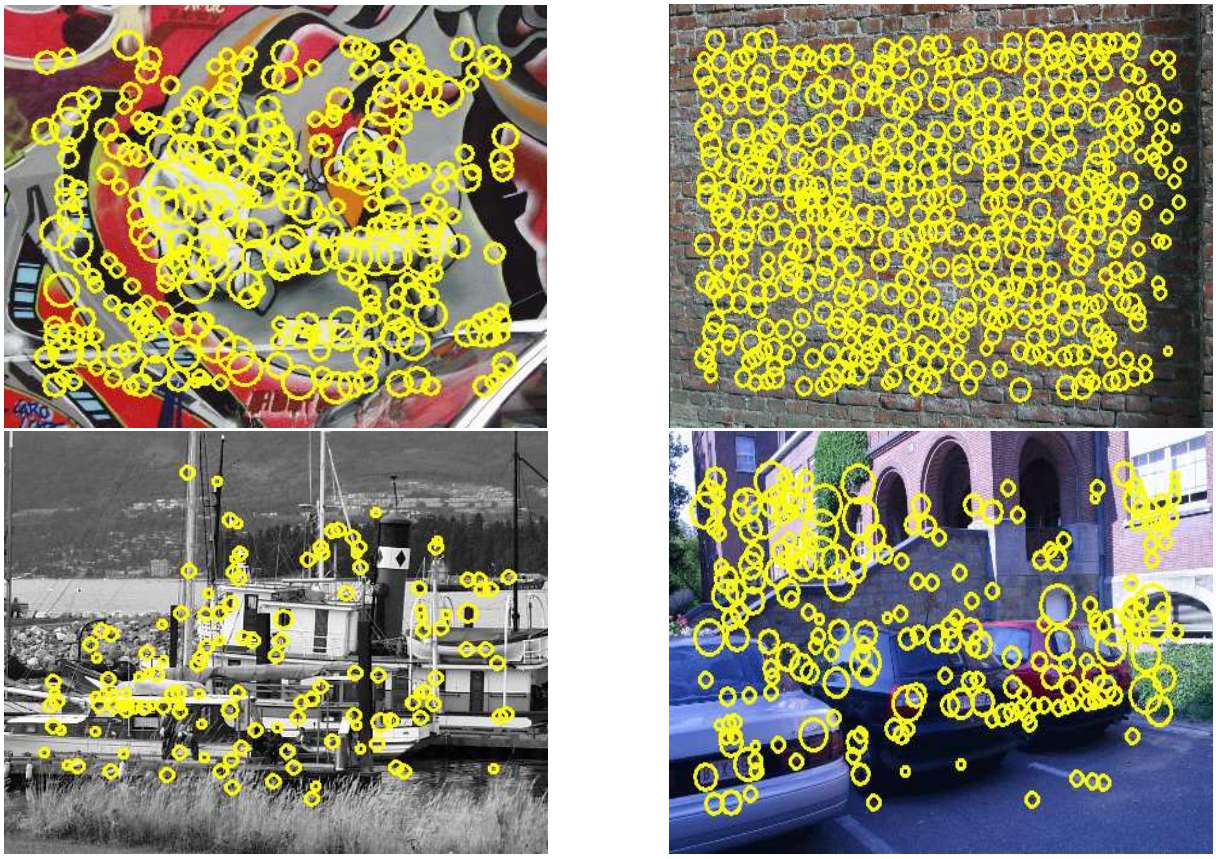


Figure 2.3: Salient regions in various scenes. The images are from the data set that is illustrated in Figure 3.1. The highlighted regions are detected by the scale-saliency algorithm that is described in Section 2.2.1.

2.2.2 Maximal Stable Extremal Region (MSER) Detection

The second criterion for the feature detection is wide-baseline stability. The benefits of these features are that they are robust against monotonic changes of image intensities as well as continuous transformations of image coordinates. The last property is useful for loop detection because the robot will barely reach precisely the same pose like before. For instance, the viewpoint of the robot changes. Figure 2.4 shows a series of images taken from different viewpoints. The regions that are highlighted are detected by the methods that are described next. It is interesting to note that most regions of the scene are detected in all images.

The detection algorithm that was developed by Matas et al. [15] fits the needs of wide-baseline stability mentioned above. Before it is explained in more detail, some definitions that are used by algorithm are given:

Image I is a mapping $I : \mathcal{D} \subset \mathbb{Z}^2 \rightarrow \mathcal{S}$. Extremal regions are well defined on images if:

1. $\mathcal{S}$ is totally ordered, i.e. reflexive, antisymmetric and transitive binary relation $\leq$

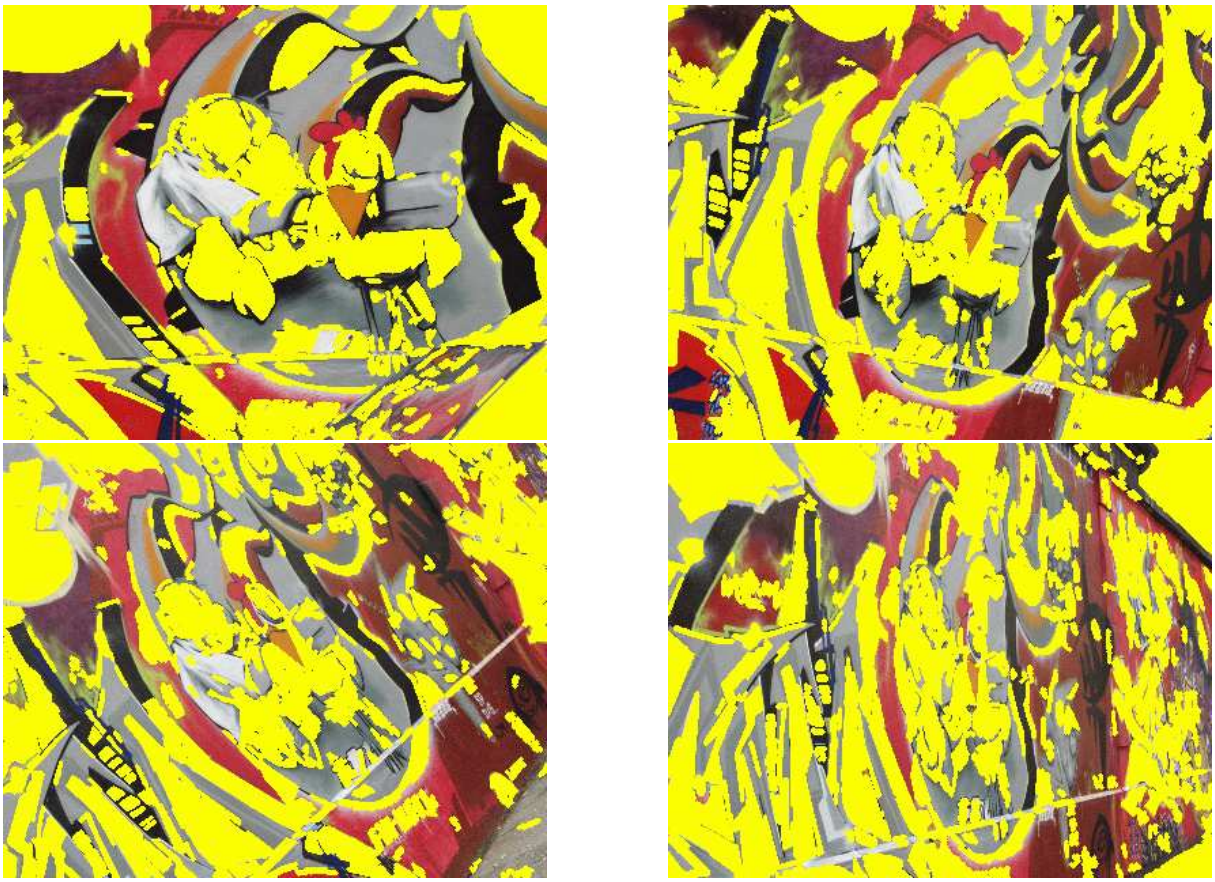


Figure 2.4: A series of images taken from different viewpoints. The images are from the data set that is illustrated in Figure 3.1. The detected maximally stable extremal regions are represented as ellipses and highlighted in each image. This series shows that many MSERs that represent one location in the scene are detected in all images although the viewpoint is changing.

exists. In this paper only $\mathcal{S} = \{0, 1, \ldots, 255\}$ is considered, but extremal regions can be defined on e.g. real-valued images ($\mathcal{S} = \mathcal{R}$).

2. An adjacency (neighborhood) relation $A \subset \mathcal{D} \times \mathcal{D}$ is defined. In this paper 4-neighborhoods are used, i.e. $p, q \in \mathcal{D}$ are adjacent (pAq) iff $\sum_{i=1}^{d} |p_i - q_i| \leq 1$[1].

Region $\mathcal{Q}$ is contiguous subset of $\mathcal{D}$, i.e. for each $p, q \in \mathcal{Q}$ there is a sequence $p, a_1, a_2, \ldots, a_n, q$ and $pAa_1, a_iAa_{i+1}, a_nAq$.

(Outer) Region Boundary $\partial\mathcal{Q} = \{q \in \mathcal{D} \setminus \mathcal{Q} : \exists p \in \mathcal{Q} : qAp\}$, i.e. the boundary $\partial\mathcal{Q}$ of $\mathcal{Q}$ is the set of pixels being adjacent to at least one pixel of $\mathcal{Q}$ but not belonging to $\mathcal{Q}$.

Extremal Region $\mathcal{Q} \subset \mathcal{D}$ is a region such that for all $p \in \mathcal{Q}, q \in \partial\mathcal{Q} : I(p) > I(q)$ (maximum intensity region) or $I(p) < I(q)$ (minimum intensity region).

[1] d denotes the number of dimensions. Since $\mathcal{D}$ is a subset of $\mathbb{Z}^2$, the dimension for d is 2.

Figure 2.5: MSERs in various scenes. The images are from the data set that is illustrated in Figure 3.1. In contrast to the salient regions, which are shown in Figure 2.3, the shape of a MSER is arbitrary.

Maximally Stable Extremal Region (MSER). Let $\mathcal{Q}_1, \ldots, \mathcal{Q}_{i-1}, \mathcal{Q}_i, \ldots$ be a sequence of nested extremal regions, i.e. $\mathcal{Q}_i \subset \mathcal{Q}_{i+1}$. Extremal region $\mathcal{Q}_{i^*}$ is maximally stable iff $q(i) = |\mathcal{Q}_{i+\Delta} \setminus \mathcal{Q}_{i-\Delta}|/|\mathcal{Q}_i|$ has a local minimum at i^* ($|\cdot|$ denotes cardinality). $\Delta \in \mathcal{S}$ is a parameter of the method.

It is crucial that the pixel values are ordered. In this implementation only gray-scale images with values from the set $[0, \ldots, 255]$ are considered. The neighborhood relation of the image pixels is depicted in Fig. 2.6. The extremal regions are of an arbitrary shape and have properties that are stable against changes in the image. They are either minimum or maximum intensity regions. The algorithm that is described here detects the minimum intensity regions. To find the other ones, the input image only needs to be inverted. The maximally stable extremal regions are a subset of the extremal regions. They have the following properties:

- Invariant to monotonic changes of image intensities.
- The neighborhood of the regions are preserved under transformation.

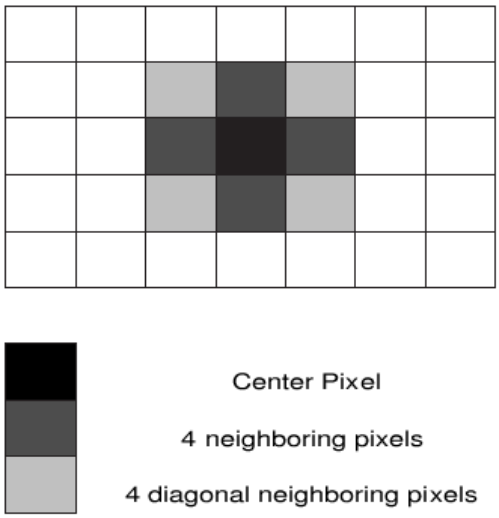


Figure 2.6: Neighborhood relation of image pixels [6]. The figure shows the neighborhood relation that is described in Table **??**. By $\sum_{i=1}^{d} |p_i - q_i| \leq 1$ the neighboring pixels are specified. This excludes the diagonal neighbors from the neighborhood.

- The regions are stable, because they stay unchanged over an interval of thresholds.
- Both very small and very large regions are detected.

First all pixels of the gray-scale image are sorted in $O(n)$ into bins with regard to their intensities using the BINSORT algorithm [29] in order to find all pixels that belong to one intensity. For each intensity from 0 to 255 a list of connected pixels (or extremal regions) below the current intensity is maintained. To visualize the process imagine a binary image where all pixels below the current intensity are black and those above are white. At first, the image is white. As the intensity increases, tiny black spots grow to large regions, and finally the complete image is black. With the union-find algorithm [29] the extremal regions are efficiently determined and stored in a data structure with their corresponding intensity levels and sizes. If two regions merge, the smaller is subsumed by the larger one. In a last step those intensities of the regions are chosen from the data structure where the rate of change of the regions size have a local minimum. The regions with these selected intensities form the maximally stable extremal regions.

2.2.3 SIFT Description

After the features or feature points are detected using both methods above, they are encoded with SIFT descriptors as developed by Lowe [14]. In order to create descriptors that are invariant to the image rotation the descriptors are calculated relative to the orientation of the feature points.

To determine the orientation of the feature points some pre-computation has to be done: For all pixels of the image the orientations and magnitudes of the pixels are determined. This is done with the following equations, where $m(x, y)$ is the magnitude and $\theta(x, y)$ is the orientation

of image pixel with the coordinates x and y. Here $L(x, y)$ is simply the the pixel value of the gray-scale image[2]:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}. \quad (2.4)$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))). \quad (2.5)$$

After the pre-computation the orientations are assigned to keypoints that are generated from the feature points in the following way: An orientation histogram is built that covers 360 degrees around the feature point. For this, 36 bins are used. The orientations θ of the pixels in the region of the feature point that contribute to the histogram are weighted by their magnitude m and a circular Gaussian window with σ 1.5. The highest peak of the histogram is the significant direction of the feature point. Additionally, other directions that lie within 80% of the peak are considered as influential. That means, one or more keypoints with different orientations are generated for each and every detected feature point. To gain a better approximation of the orientation of the keypoints the three values next to each peak are used to interpolate the value.

Then the descriptors are calculated from the generated keypoints which exhibit a location and an orientation. How this is done is exemplarily shown in Fig. 2.7. On the right side of Fig. 2.7 a descriptor with a 2×2 array is illustrated with an 8 bin orientation histogram in each of its fields. In this implementation a 4×4 array is used instead. This leads to a 128 dimensional vector as descriptor ($4 \times 4 \times 8 = 128$). The descriptor is determined by the Gaussian weighted gradients of the image pixels around the keypoint, which are shown on the left hand side in Fig. 2.7. The Gaussian function[3] assures that the gradients of pixels that lie next to the keypoint have more influence on the outcome than pixels that are more distant from the center. These weighted gradients contribute to the array of orientation histograms relative to their position. To deal with boundary effects between two orientation histograms another weighting factor is used for interpolation for each dimension. This factor is $1 - d$, with d the distance of the current orientation value to the central histogram bin value. Finally the descriptor vector is normalized, all values greater than 0.2 are thresholded[4], and the vector normalized again. These three steps are made in order to make the descriptor more robust to changes in illumination. While the normalizing is concerned with linear illumination changes, the thresholding deals with non-linear changes, because it stresses more the orientations than the magnitudes of the gradient histograms.

2.3 Visual Feature Matching

After detecting and encoding the visual features of a recently taken image by the robot the feature descriptors and the capture time of the image are stored in the database. To detect a loop the descriptors of a query image I_q are compared or matched with the feature descriptors of all candidate images I_c's that are already in the database. A successful matching of a graffiti scene is illustrated in Figure 2.8.

[2]Note, this is different from the procedure presented by Lowe [14] where $L(x, y)$ has another meaning.
[3]The σ of the function corresponds to one half of the descriptor window.
[4]The value 0.2 was determined experimentally by Lowe [14].

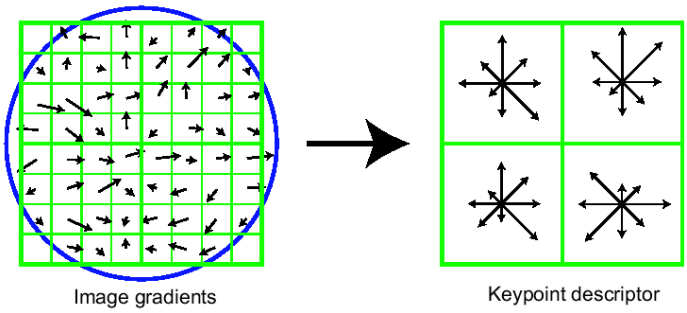


Figure 2.7: A keypoint descriptor. It is calculated from weighted gradients in a certain area around the keypoint [14]. On the left side, the gradients of each pixel and the Gaussian weighting function are illustrated. The right side of the figure shows a 2×2 array of orientation histograms. The histograms are built from the weighted gradients of the left hand side. In this implementation a 4×4 descriptor array is used.

Figure 2.8: Examples for a successfully matched graffiti scene. The images are from the data set that is illustrated in Figure 3.1.

The similarity measure in the matching process for two feature descriptors is the Euclidean norm of the distance between a descriptor V_q of the query image and a descriptor V_c of a candidate image. In this implementation the descriptors are 128 dimensional vectors:

$$\parallel V_q - V_c \parallel . \tag{2.6}$$

For two images each query descriptor is matched with each candidate descriptor. The resulting norms are stored in a $n_q \times n_c$ matrix m where n_q is the number of descriptors generated for the query image and n_c is the number of descriptors for one candidate image. for all candidate images such a m matrix is created. Finally the matrices are thresholded and only norms below a certain threshold are considered for further processing. These norms correspond to similar descriptors. For each image pair (I_q,I_c), n_{qc} norms are the result of the feature matching process. How these matched features are used for loop closing is explained in the next section.

2.4 Application of Loop Closing

Loop closing uses visual features as follows to detect a loop in the path of the robot. For a query image I_q:

1. Generate n_q features descriptors V_q from the image I_q.
2. Store feature descriptors and capture time of the image in the database.
3. For each candidate image I_c in the database:
   (a) Retrieve all n_c candidate feature descriptors V_c from the database.
   (b) Build a $n_q \times n_c$ matrix $M_{q,c}$ where the $(i,j)^{th}$ entry $M_{q,c}(i,j)$ is the Euclidean norm $\parallel V_q(i) - V_c(j) \parallel$.
   (c) Threshold the distances result in n_{qc} matched descriptors.
4. After all candidate images are processed the candidate images with the largest number of n_{qc} matched descriptors are selected, if the number is higher than a certain threshold.
5. The capture times of the selected images are compared with a separate journal of temporal and spatial information in order to determine the location where the candidate image was made. Finally, a loop hypothesis for the assumed location is generated.

This algorithm detects a loop if some descriptors of two images are similar. Similar means that the norm of two descriptors is below a certain threshold. For each image pair the similar descriptors are counted. If two images have, for example, three or more of these similar descriptors, it is possible that they were taken in the same location, and a loop hypothesis is generated. The number of needed similar descriptors in order to detect a loop was determined while testing.

It is interesting to note that no spatial information of the robot pose is used during that process. At first it sounds strange not to use this freely available information, but on the other hand this has the advantage that the loop detection is not misguided by erroneous geometrical data. If loop detection depends on the estimated robot pose, the robot potentially misses situations where it has been before only because the pose estimation was incorrect. To overcome this problem, here the application of loop closing is independent from the estimated robot pose.

Chapter 3

Evaluation of the Visual Feature Extraction

The presented algorithms for feature detection in Sections 2.2.1 and 2.2.2 and the description in Section 2.2.3 are compared with other methods within the framework developed by Mikolajczyk and Schmid [22]. The used test data is described in Section 3.1. How the detection and description is evaluated is explained in Section 3.2. These results are presented in detail in Section 3.3.

3.1 Data Set

To produce comparable results, the described detectors and descriptors are tested not only in the same framework but also on the same data set. The data set that is used here is publicly available on the website `http://www.robots.ox.ac.uk/~vgg/research/affine/`. This test set of images contains small subsets where different image transformations were applied. The subsets can be grouped in five categories: image blur, viewpoint change, zoom and rotation, light change, and JPEG compression. In addition to these transformations the type of the scene is another influential factor for feature detection. Therefore, for some of the transformation categories, two image sets exist with different scene types, namely, structured versus textured scenes. In total, there are eight image sets of six images each. Examples of the different test sets can be seen in Fig. 3.1. How the data is acquired is explained in more detail on the website mentioned above and in [22].

When evaluating the detection methods on the test data, the question arises, how to know whether the matched features are really describing the same location in two images or not. The answer is that some ground truth information is needed that describes how the two images are related. This information is given by a homography and comes together with the test data. These homographies are calculated as follows: In a first manual step, image correspondences are set by hand. Then the two images are aligned using this information. In the second and final step, the homography of the two images is refined using an out-of-box algorithm that brings automatically detected features together. The combination of the two steps makes this calculated homography very accurate.

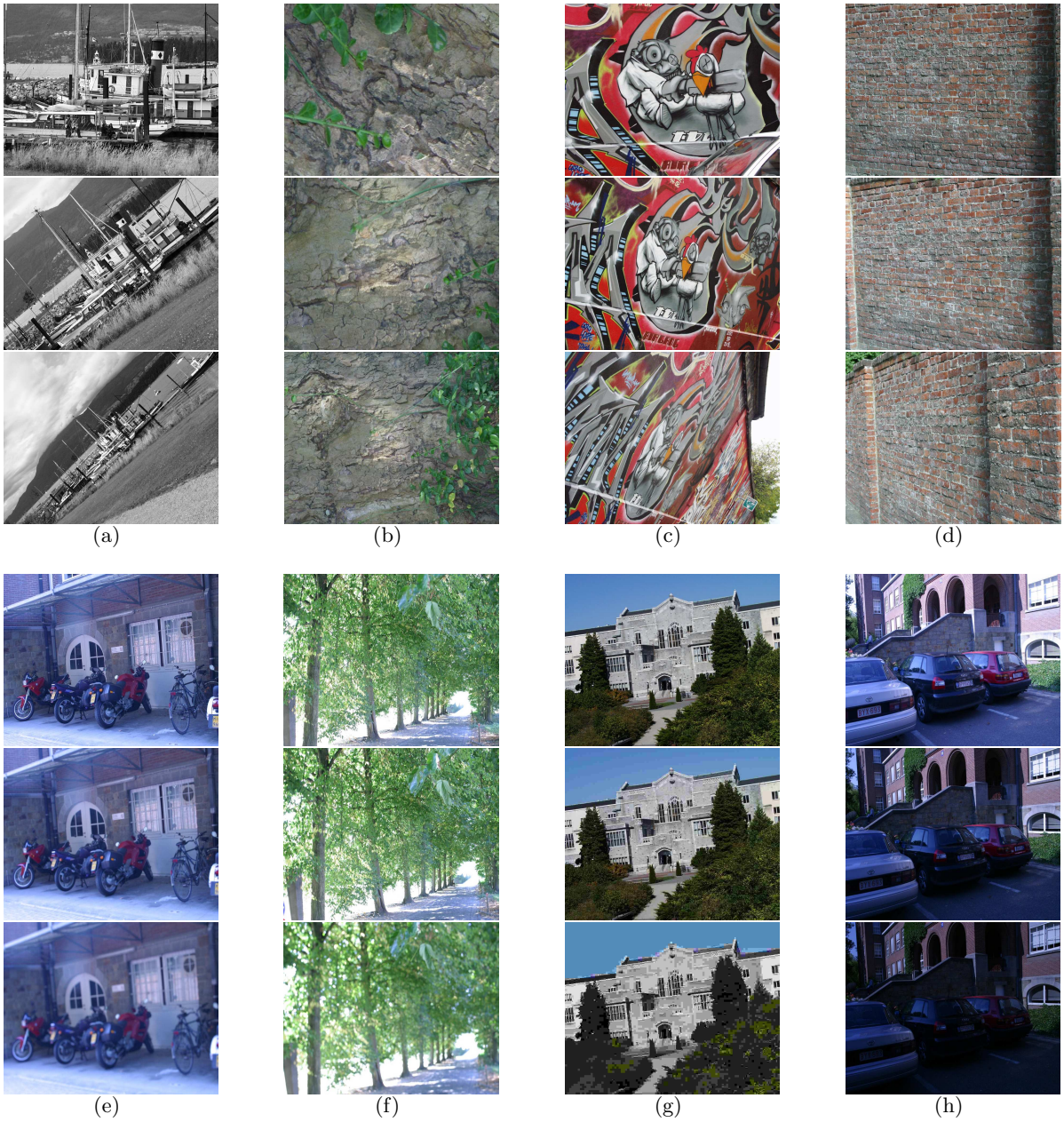


Figure 3.1: Examples of the evaluation data set. The data set is publicly available on the website `http://www.robots.ox.ac.uk/~vgg/research/affine/`. It can be distinguished in five categories: (a) and (b) zoom and rotation, (c) and (d) viewpoint change, (e) and (f) blur, (g) JPEG compression, and (h) light change.

3.2 Evaluation Criteria

There are two steps in the visual feature extraction process (see Section 2.2) that can be distinguished: First, the combined feature detection of the salient and wide-baseline stable regions. And second, the feature encoding using SIFT descriptors. Both of these steps are evaluated separately. What criteria are used to evaluate each of these steps is described in the respective subsections.

3.2.1 Detector Evaluation

The detector evaluation is of special interest, the proposed combination of the saliency and the wide-baseline stability criteria for feature detection by Newman and Ho [23] is compared to other available region detectors. Especially the comparison between the combined saliency-MSER detector and the exclusive saliency or MSER detector is one of the main objectives of this thesis.

In order to use the evaluation framework the detected regions have to be represented as ellipses [22]. A region is described by the parameters u, v, a, b, c that fulfill the following equation, where the upper left corner of the image is represented with $(0, 0)$:

$$a(x-u)^2 + 2b(x-u)(y-v) + c(y-v)^2 = 1. \tag{3.1}$$

The evaluation framework provides the tester with two measures that can be used for analyzing the feature detectors. These measures are the repeatability and the matching score. Both measures are calculated for a given image pair.

The repeatability score is the number of corresponding regions with respect to the smaller number of detected regions in an image:

$$repeatability\ score = \frac{\#\ corresponding\ regions}{\#\ detected\ regions}. \tag{3.2}$$

The number of region-to-region correspondences is calculated using the ground truth information. The regions from one image are projected to the other. Two regions correspond if the overlap of the regions is sufficiently large. To determine the overlap of two regions an overlap error is computed. The overlap error is calculated using the ratio between the intersection and the union of the regions $(1 - intersection/union)$. The error of two regions A and B can be computed as follows where H is the homography between the two images:

$$\epsilon_O = 1 - \frac{A \cap H^T B H}{A \cup H^T B H}. \tag{3.3}$$

If the overlap error ϵ_O is smaller than a threshold, A and B are counted as corresponding regions. Figure 3.2 shows some examples of overlapping regions together with their overlap error in percent. The overlap error is influenced by the size, the orientation and the position of the regions. For the evaluation different thresholds for the overlap error are used.

The second measure, the matching score, is the relative number of correctly matched regions compared with the smaller number of detected regions in one image:

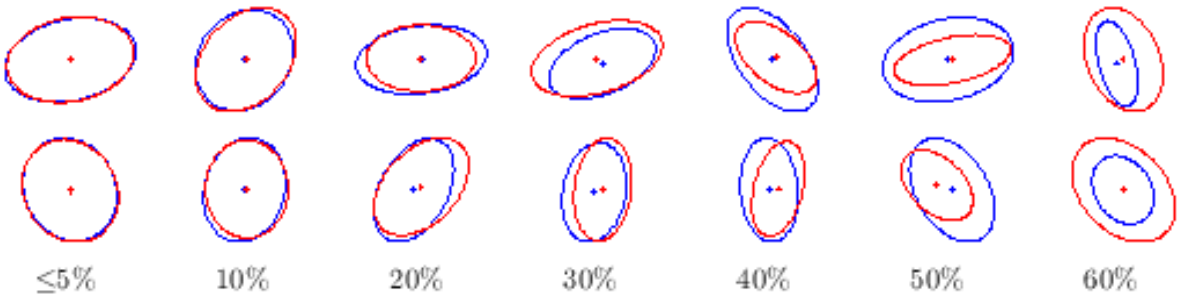


Figure 3.2: Examples of overlapping regions and their respective overlap error in percent.

$$matching\ score = \frac{\#\ correct\ matches}{\#\ detected\ regions}. \tag{3.4}$$

The matching is done on the basis of descriptors, therefore the regions have to be encoded for the calculation. In this evaluation SIFT descriptors are used. The descriptors are compared using the Euclidean distance. A match is the nearest neighbor in the SIFT feature space. The correctness of the matching is determined by the ground truth homographies (see Section 3.1) and the overlap error ϵ_O that is explained above.

The results of the presented Salient-MSER detector are compared with six other detectors: the Salient region detector, the MSER detector, the Harris-Affine detector, the Hessian-Affine detector, the Intensity extrema based detector (IBR), and the Edge based detector (EBR).

3.2.2 Descriptor Evaluation

In the descriptor evaluation, the SIFT descriptor that was proposed by Newman and Ho [23] is compared with other descriptors. All descriptors are calculated from the feature regions that are detected by the combined Salient-MSER detector in order to see if the proposed SIFT descriptor is the best choice for the detection method under inquiry.

The matching score is one performance figure for this evaluation, it was already discussed in Section 3.2.1.

Another figure, which indicates distinctiveness of the descriptors, is the detection rate with respect to the false positive rate. The detection rate is the ratio between the number of correct matches and the total number of corresponding regions. The false positive rate is the ratio of the false matches and all possible false matches. This figure is very useful for a practical application, because it measures the quality of the feature matching.

Besides the SIFT descriptor the other methods that are used are: GLOH, Shape Context, PCA, Moments, Cross correlation and Steerable filters.

3.3 Evaluation Results

The evaluation results are presented in this section. As mentioned above, the region detection and the region description are evaluated separately. Therefore the structure of this section has two parts: The results of the detector evaluation are presented in Section 3.3.1, the results of the descriptor evaluation are presented in Section 3.3.2.

3.3.1 Detector Evaluation

The following feature detectors are tested within the framework: the Salient-MSER detector, the MSER detector, the Salient region detector, the Harris-Affine detector, the Hessian-Affine detector, the Intensity extrema based detector (IBR), and the Edge based detector (EBR). While the first three detectors are implemented for this thesis, the binaries of the other detectors are taken from the website mentioned above. The images from the test data set are applied to the detectors. The detector output, the detected regions, is used as input for the evaluation framework. Before the results for each category of the various image transformations (i.e., scale change, viewpoint change, blur, JPEG compression and light change) are presented, some general remarks are stated.

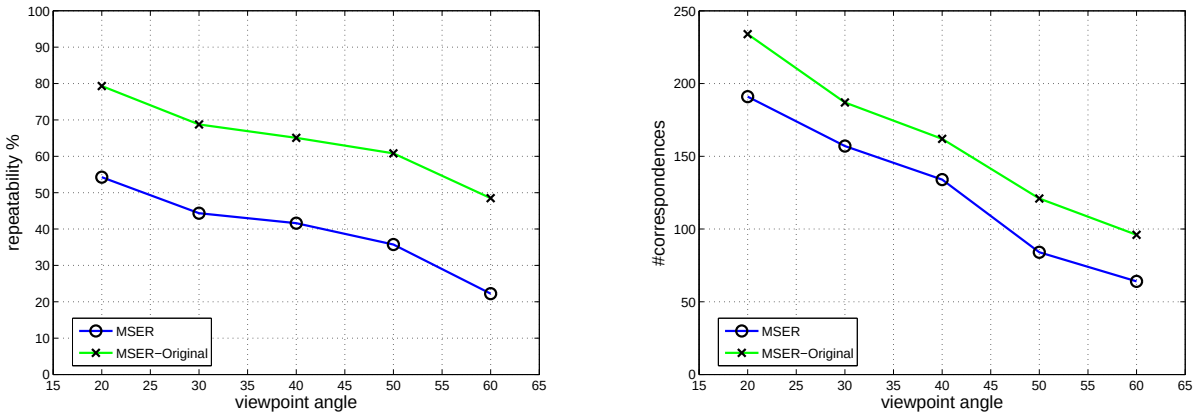
General Remarks

Different Results. The first thing to note is that the results for the salient region detector and the MSER detector are different from those that are obtained in [18]. For the other tested detectors the results are comparable.

In the case of the salient region detector this is not surprising because the detected regions in [18] are described as ellipses while in this implementation circles are used.

In the case of the MSER detector the difference is presumably due to the fact that other parameters are used in this and the original implementation by Matas et al. [15]. The parameters are not clearly laid out in the original paper, so the parameters in this implementation were defined during testing. Although the results are different and the performance measured here is not as good as in [18], a comparison between the Salient-MSER, the MSER, and the Salient region detector perfectly makes sense since the regions are detected by this implementation. This difference is depicted for the structured graffiti scene from Figure 3.1(c) in Figure 3.3. It is noteworthy that the shapes of both curves are similar, only the run of the curves is shifted. The performance for other scenes is comparable. While studying the evaluation results one should bear in mind that the original MSER implementation [15] is reported to perform roundabout 15–30% better than this implementation. So the performance of this implementation can possibly be improved in future work. The evaluation results should be viewed in consideration of this prior knowledge.

Computation Time. The issue of complexity and computation time of the detectors is shortly discussed in [18]. In the setting of the evaluation, the computation times are not that important, but they are crucial when it comes to an on-line application such as loop closing. Similar computation times that are listed in [18] are obtained from the experiments: Most detectors



(a) (b)

Figure 3.3: Differences between this and the original MSER implementation on a structured graffiti scene. (a) Repeatability score. (b) Number of corresponding regions.

process an image of size 800×640 in a reasonable time, less or equal than 10 seconds. The exception is the salient region detector which processes the image in more than 30 minutes. It is clear that a method with such a computation time is not applicable for an on-line application. Therefore the parameters of the salient region detection need to be adjusted in order to integrate the loop closing procedure in the 6D-SLAM robot platform, which is presented in Section 4.

Region Density and Region Size. Both the region density and the region size has an effect on the performance measure of the detector.

The number of detected regions depends on the detector but also on the scene type. In Table 3.1 the numbers of detected regions for a structured scene are illustrated for each tested detector. For comparison, the table shows the numbers for a textured scene.

It is noteworthy that all detectors, except the Hessian-Affine detector, detect less regions in the structured scene than in the textured scene. The difference for the Salient-MSER detector is particularly noticeable. The differences in region numbers are plausible because the detectors are receptive to different image structures, what makes them complementary. Since the number of detected regions varies among the different detectors, the performance measures for the evaluation results, presented in the next section, are reported in absolute and relative terms in order to give the reader a better understanding.

The size of the detected region depends also on the detector. While some detectors detect mostly small regions, others detect larger ones. But the region size has an influence on the matching process. Larger regions are more likely to be matched. In order to achieve a better comparison between the detectors the regions are normalized.

Table 3.1: Number of detected regions for various detectors. The numbers are reported for two different scene types, namely a structured and a textured scene.

Detector	Structured scene	Textured scene
Salient-MSER	368	2289
MSER	528	2871
Salient	1025	2033
Harris-Affine	1758	2267
Hessian-Affine	2454	1375
IBR	679	783
EBR	1265	3748

Results for Various Image Transformations

The results are grouped in five categories: scale change, viewpoint change, blur, JPEG compression and light change. In each of these categories one or two scene types are tested. For each scene a set of six images is applied to the detectors. One image is the reference image. The others show the same scene under increasing image transformations. For the evaluation, the reference image is pairwise processed with each of the other images. The repeatability score, the absolute number of corresponding regions, the matching score and the number of correct matches are reported for these transformations in figures: 3.4, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10 and 3.11. The overlap error threshold ϵ_O is fixed to 40% for these tests.

An optimal curve for the repeatability score is a horizontal line at 100%. A horizontal line indicates the stability of the detector under increasing image transformations. A repeatability score at 100% means that for all detected regions in one image a corresponding region is found in other image. None of the tested detectors perform that optimally. In the most cases a curve starts at its own maximum value and then decreases as the image transformation increases. For some cases the curves are nearly horizontal. The maximum value of 95% for the repeatability score is measured for the JPEG compression category.

The figures that display the matching score and the number of correct matches are in general more interesting for a practical analysis. The matching score is an indication on how distinctive the detected regions are. At first glance the figures look similar to those of the repeatability score, but at some figures the ranking of the detectors changes. This means that the regions of some detectors are more distinctive than others.

Scale Change. Figure 3.4 shows the results for the structured boat scene from Figure 3.1(a). In Figure 3.5 the results for the textured bark scene from Figure 3.1(b) are presented.

For the repeatability score the Hessian-Affine detector yields the best results for both scene types. In the textured scene the curve runs nearly horizontal. The MSER detector performs slightly better than the Salient-MSER for both scenes. It is noteworthy that for the textured scene there is a huge gab between the number of correspondences between for these two detectors.

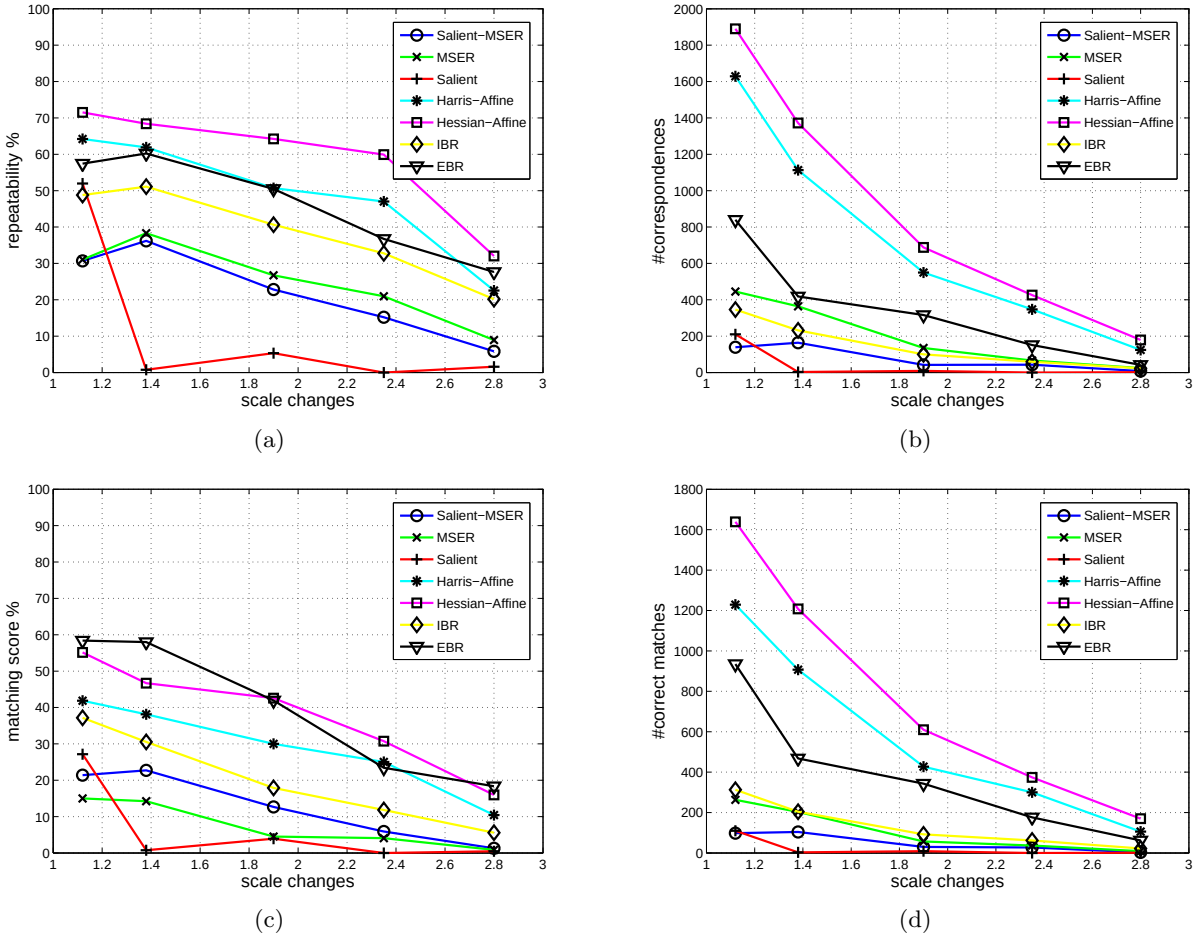


Figure 3.4: Scale change transformations for the structured boat scene from Figure 3.1(a). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

The performance of the Salient region detector is noticeable; larger amounts of zoom and rotation transformations yield a performance near 0%.

For the structured scene the Hessian-Affine and the EBR detector have the highest matching score. Interestingly the Salient-MSER detector performs better than the MSER detector, which is in opposite to the results of the repeatability test. The performance of all detectors is significantly worse in textured scene.

Viewpoint Change. The influence of a changing viewpoint for a structured graffiti scene from Figure 3.1(c) is shown in Figure 3.6. The results for a textured scene of a brick wall from Figure 3.1(d) are illustrated in Figure 3.7.

In the structured scene the performance for the repeatability score of the detectors looks similar. Only the Salient region detector shows a different shape of the curve. The Salient-

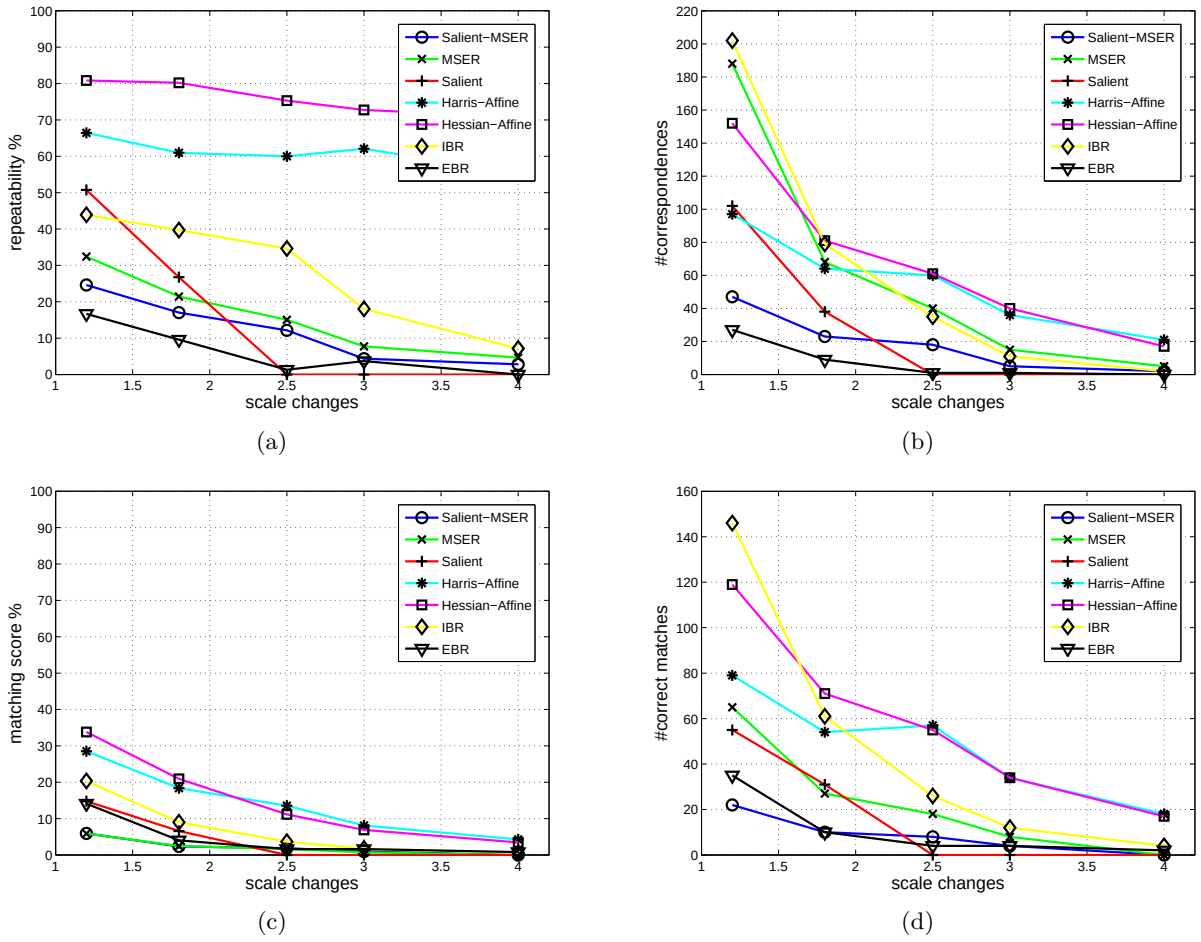


Figure 3.5: Scale change transformations for the textured bark scene from Figure 3.1(b). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

MSER and the MSER detector are nearly identical. In the textured scene the Salient region detector performs best in terms of repeatability score, except for the last transformation. Again the performance of the Salient-MSER and the MSER detector are comparable.

The EBR detector performs substantially better in the matching test than in the repeatability test. This is true for the structured and the textured scene. The ranking of the other detectors does not change significantly. The Salient-MSER detector obtains better scoring as the MSER detector for the structured scene. This is the same result as in the scale change setting.

Blur. Figure 3.8 displays the results for the structured bike scene from Figure 3.1(e), while Figure 3.9 displays the results for the textured tree scene from Figure 3.1(f).

For the repeatability score nearly all curves run horizontal for both scene types. The Hessian-Affine detector performs best in both tests. The Salient region detector is slightly better for the

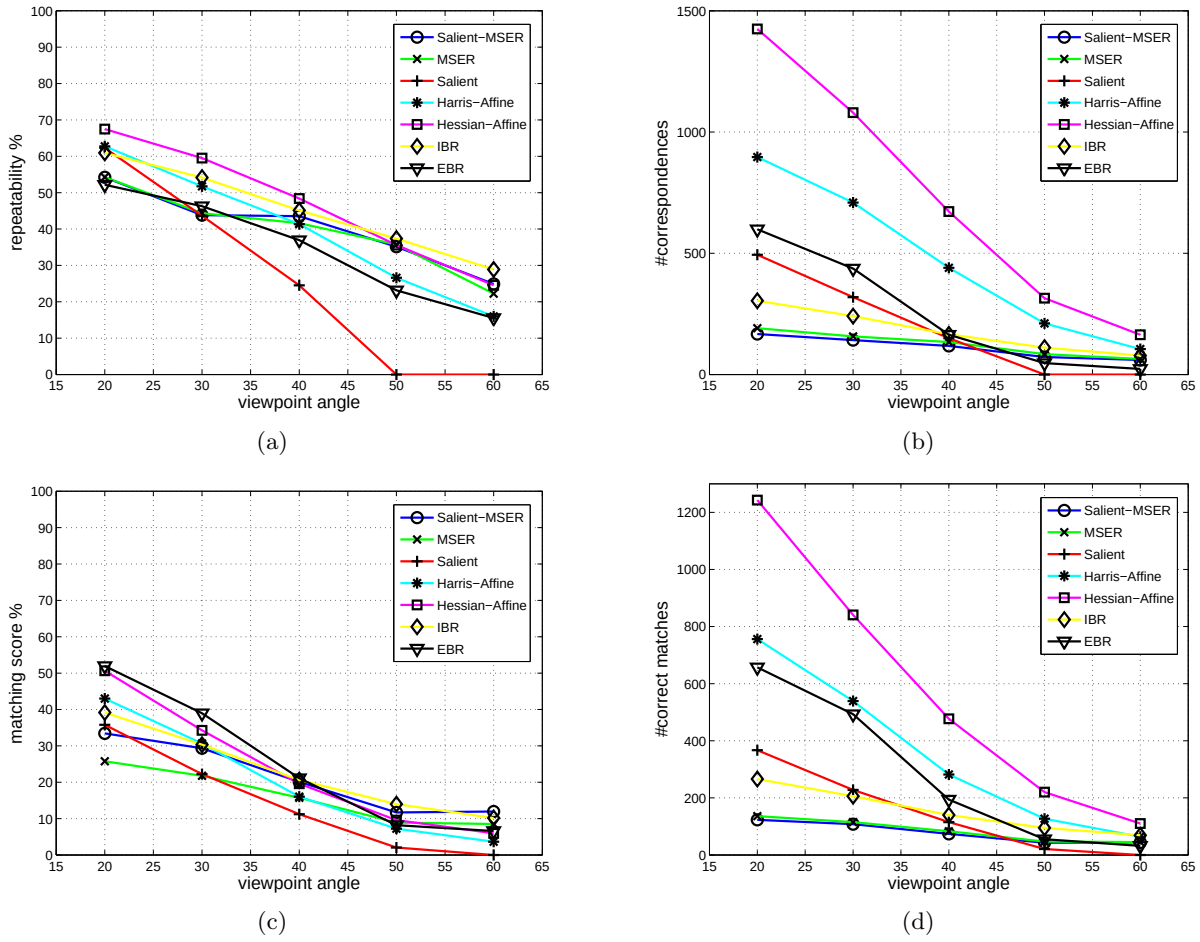


Figure 3.6: Viewpoint change transformations for the structured graffiti scene from Figure 3.1(c). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

textured than for the structured scene. The Salient-MSER and the MSER detector are almost identical for the textured scene. For the structured scene the Salient-MSER is around 5% better than the MSER detector.

In general the matching score is higher in the structured than in the textured scene for all detectors. In the EBR detector obtains the highest scores for the structured scene. Again the Salient-MSER outperforms the MSER detector in the structured scene, while the score is nearly identical for the textured scene. The Salient region detector yields higher scores than both detectors based on MSERs.

JPEG Compression. Figure 3.10 shows the effects for different levels of JPEG compression for the image set from Figure 3.1(g).

The repeatability score of 95% that is reached for the JPEG compression category is the

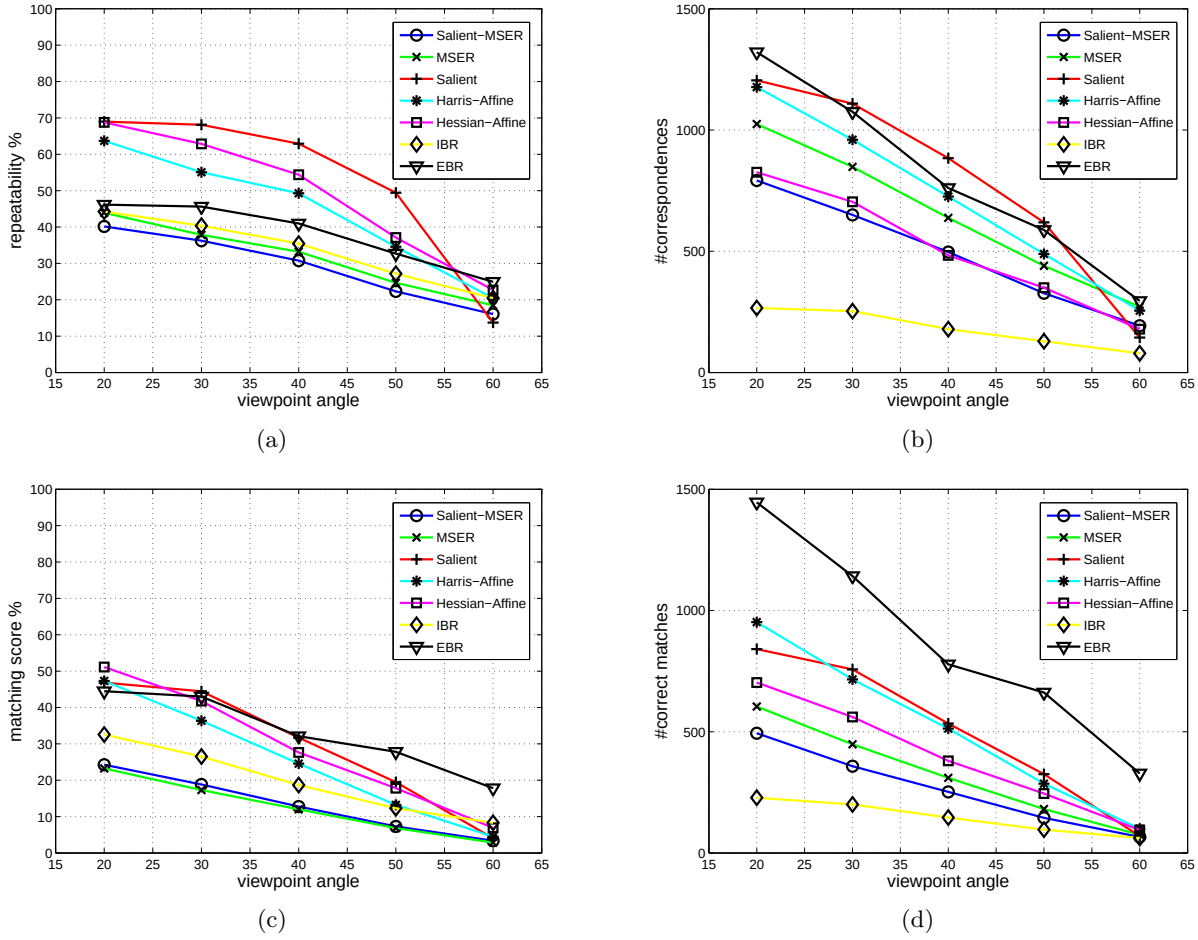


Figure 3.7: Viewpoint change transformations for the textured wall scene from Figure 3.1(d). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

maximum value that is obtained for all tests. Again, it is the Hessian-Affine region detector that outperforms the others. The Harris-Affine detector comes second. The worst performance is measured for the Salient-MSER and MSER detectors. The Salient region detector is approximately in the middle of the best and the worst detector.

The curves for the matching score are similar to those of the repeatability score, except for the Salient region detector, which performance for the matching test is not as good as for the repeatability test.

Light Change. Effects of illumination changes in the images from Figure 3.1(h) are presented in Figure 3.11.

All detectors show a stable repeatability score. The Hessian-Affine performs best. The MSER detector comes second and shows an around 10% better performance than the Salient-MSER

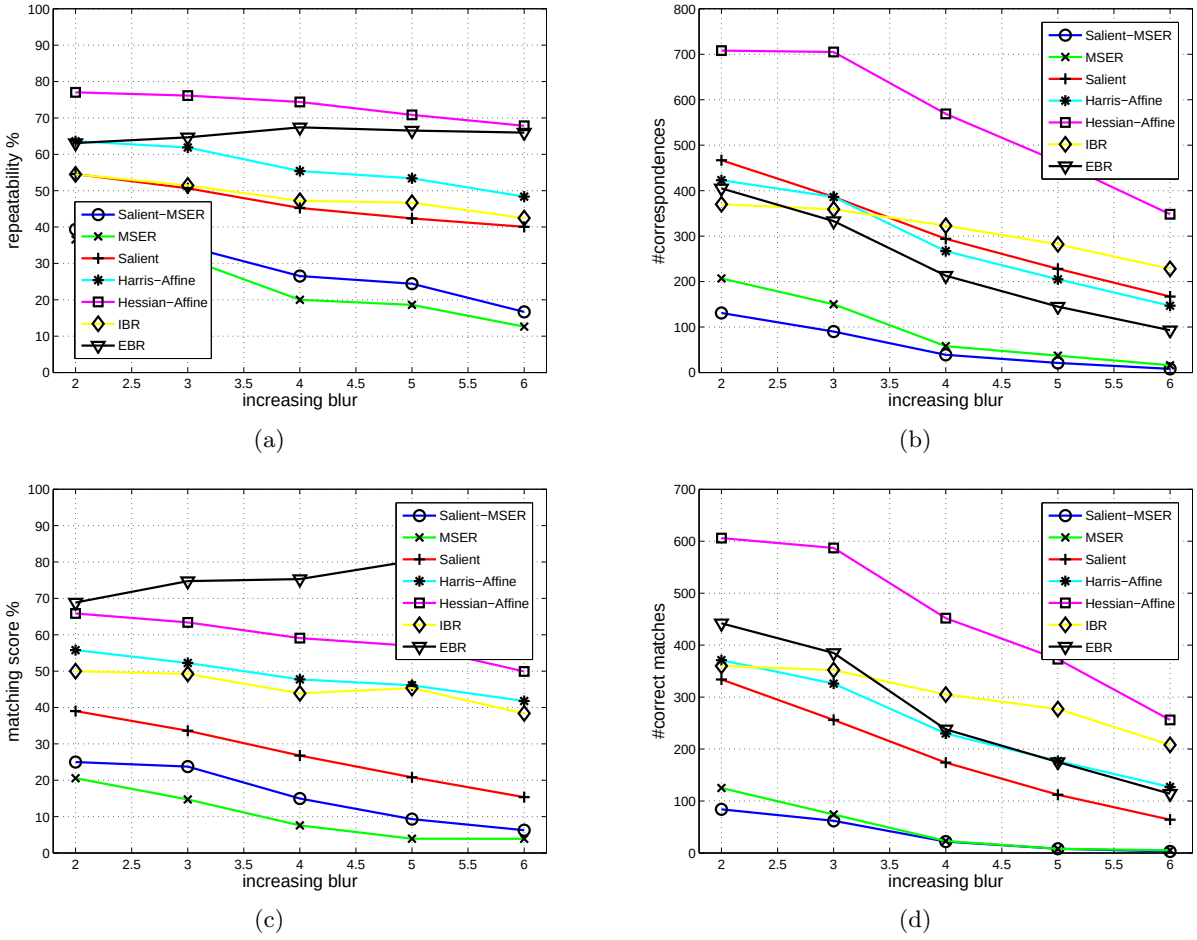


Figure 3.8: Blur transformations for the structured bike scene from Figure 3.1(e). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

detector. The Salient region detector show a slightly worse performance for larger amounts of decreasing light than the other detectors.

The best matching score is obtained by the EBR detector. What is interesting here is that the order between the Salient-MSER and the MSER changes. While the MSER detector has higher scores for the repeatability test, it is the Salient-MSER detector which performs slightly better.

Conclusions

The results show the differences between the tested detectors in respect to the various image transformations as well as to the two scene types.

In general the change of the viewpoint seems to be the most difficult setting for all detectors, followed by the change of the scale. For increasing blur and decreasing light changes nearly all

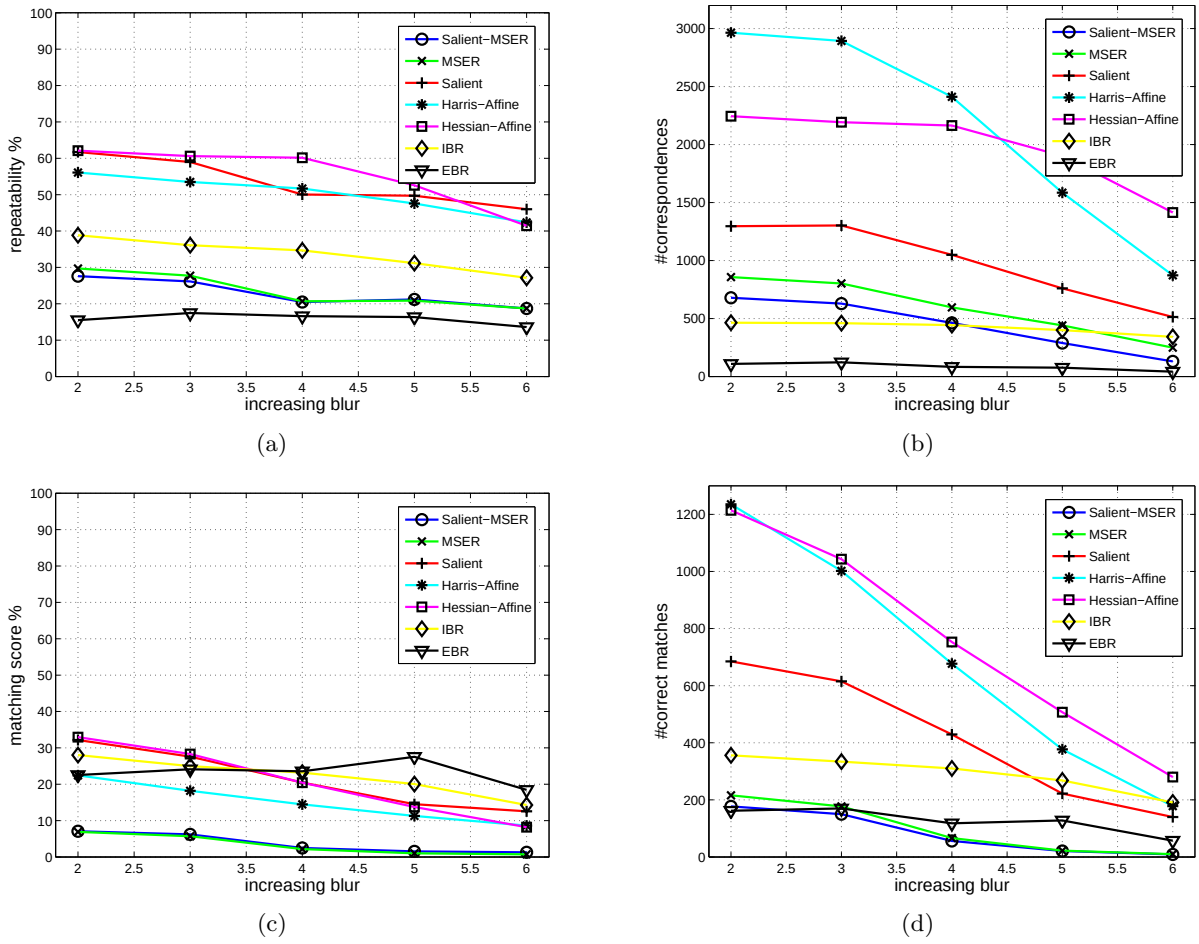


Figure 3.9: Blur transformations for the textured tree scene from Figure 3.1(f). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

detectors are relatively robust and show almost horizontal curves. Another point to mention is that the matching of feature region is better on structured than on textured scenes.

The Hessian-Affine detector shows mostly the best performance for both repeatability and matching score, it also has almost the highest number of correspondences. The Harris-Affine detector shows also good results. It is often ranked on the second or third place. While the performance of the IBR detector is on average, the performance of the EBR detector changes from best to worst depending on the image transformation and scene type.

The main focus of this evaluation is on the Salient-MSER detector and how it performs different from the MSER and the Salient region detector. The Salient region detector performs better on textured scenes than on structured scenes. For the MSER detector the opposite is true, its performance is better on structured scenes. These two results sound promising for the combined Salient-MSER detector. The Salient-MSER detector obtains slightly higher scores

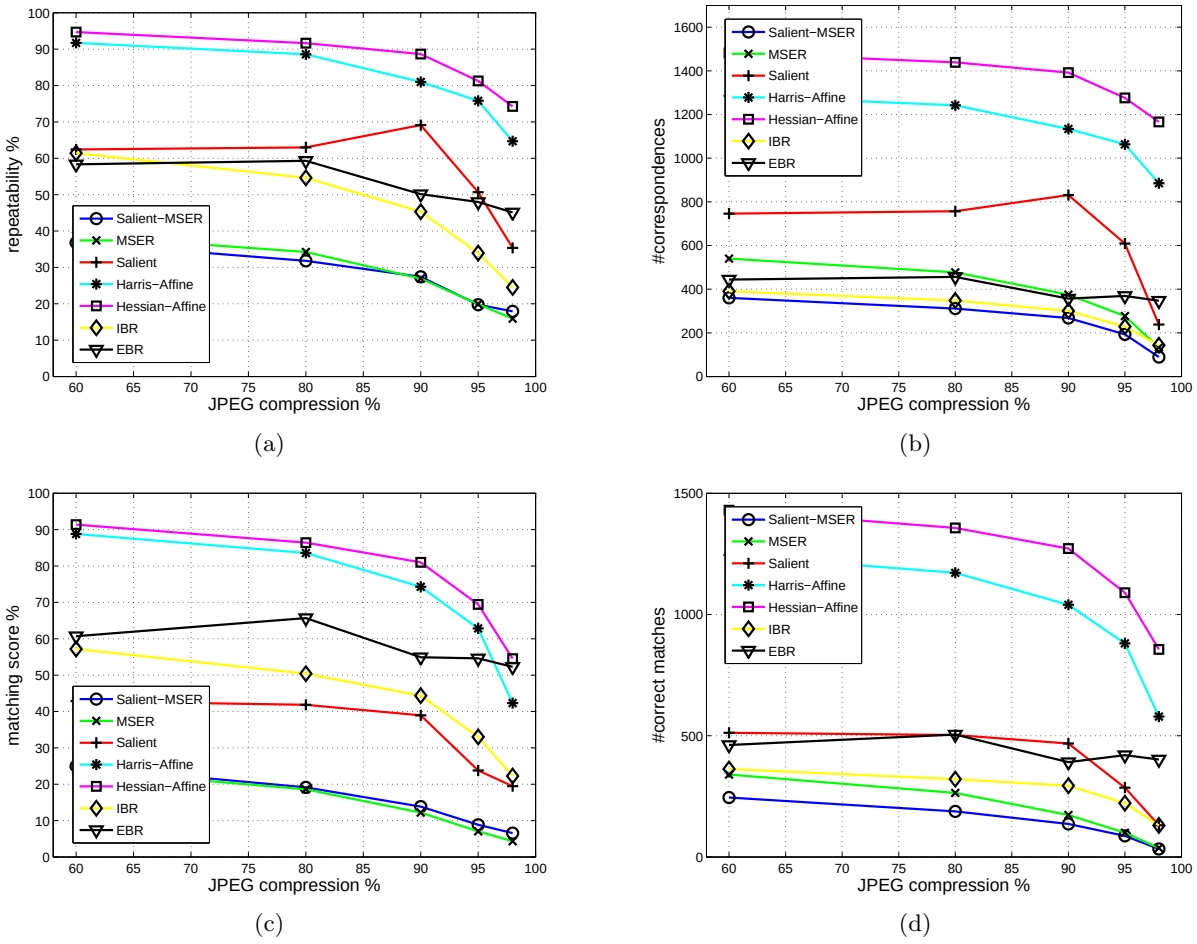


Figure 3.10: JPEG compression transformations on images from Figure 3.1(g). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

than MSER detector for structured scenes. But for the textured the performance is similar. In total the performance of the Salient-MSER detector is not significantly different from the MSER detector. This conclusion is somewhat humbling since the purpose of the combination of the two different criteria, namely saliency and wide-baseline stability, is to create a more robust detection method. So the results obtained from the evaluation do not show a substantial advantage of the combination of the two detectors. But for the task of loop closing a combination is nevertheless reasonable. In their work [23] Newman and Ho suggest a combination of these detectors to increase the robustness for the task of loop closing. The purpose of the Salient region detector is to select interesting regions in one image while the task of the MSER detector is to find regions that are robust to viewpoint changes. For an application of loop closing it makes sense to combine these both detectors for mainly two reasons: First, using only salient regions leads to many regions not be matched from different viewpoints. Second, using only

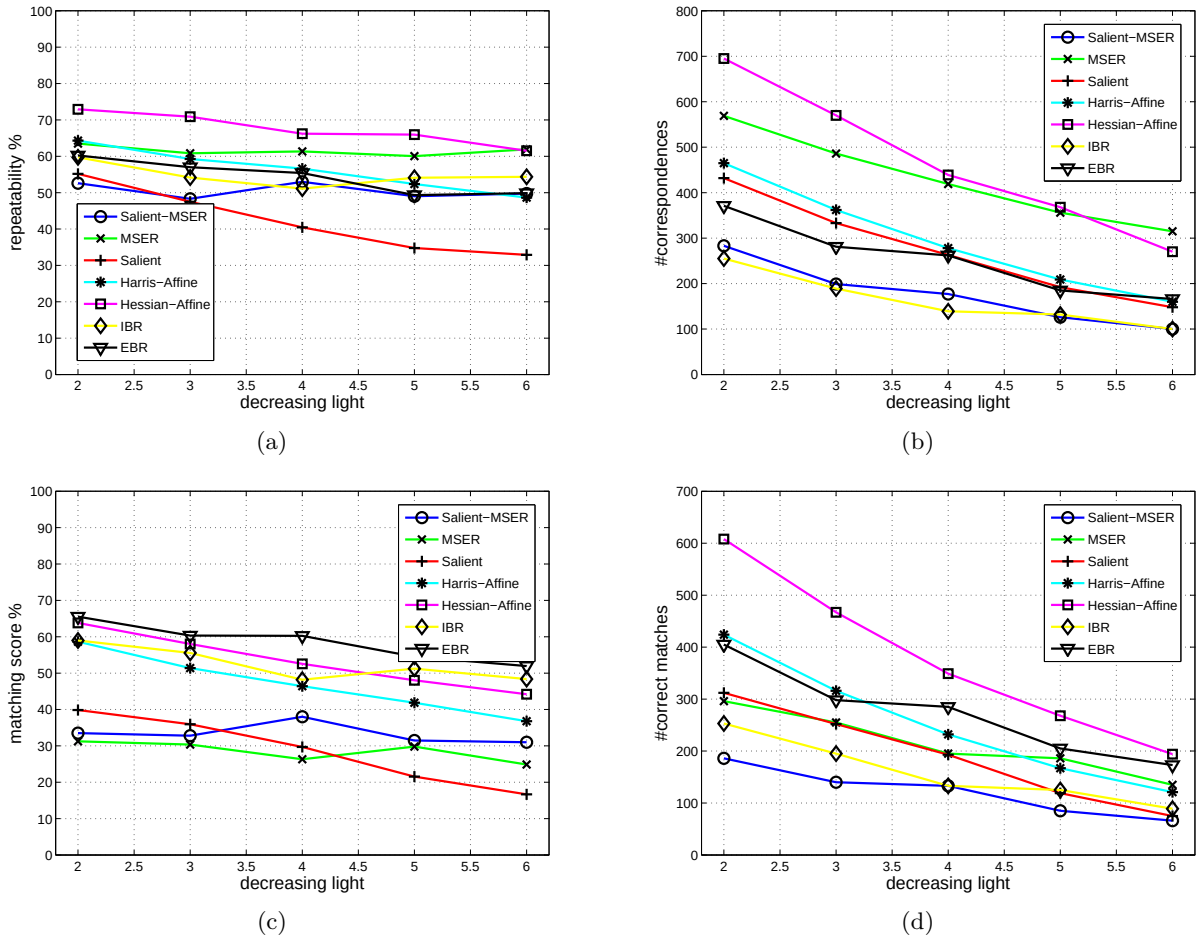


Figure 3.11: Light change transformations on images from Figure 3.1(h). (a) Repeatability score. (b) Number of corresponding regions. (c) Matching score. (d) Number of correct matches.

MSER regions leads to less distinctive regions. Hence a combination leads to more distinctive regions that are detectable from different viewpoints.

3.3.2 Descriptor Evaluation

For the following tests all regions are detected by the Salient-MSER detector. At first glance this sounds odd, since other detectors outperform this detector. But as discussed in Section 3.3.1 the results for MSER and therefore also for Salient-MSER detector are potentially better. Another reason for the selection of the Salient-MSER detector is that the choice of Newman and Ho [23] for the SIFT description method should be verified.

The regions are described using the following methods: GLOH, SIFT, Shape Context, PCA, Moments, Cross correlation and Steerable filters. These described regions are the input for the evaluation framework. As for the detector evaluation in Section 3.3.1 the results are presented

for various image transformations, i.e., scale change, viewpoint change, blur, JPEG compression and light change.

Results for Various Image Transformations

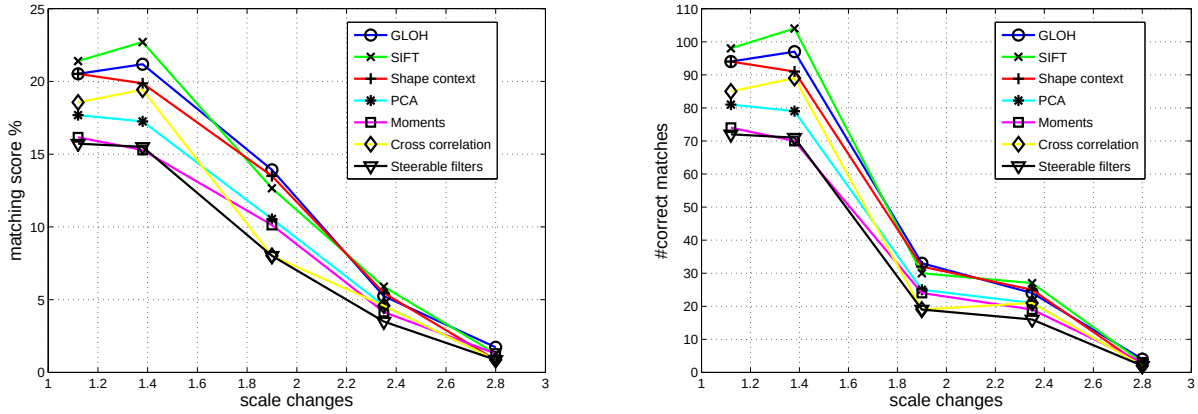
For the descriptor evaluation the matching score and the number of correct matches are reported for the various image transformations and the different scene types. Like in the detector evaluation, the overlap error threshold is fixed to 40%. Regarding the computation times there are only minor differences between the descriptors.

Whereas the results for the detector evaluation show great differences between the several detectors, the results for the tested descriptors do not. In general all curves run more or less in parallel. It is noticeable that for small amounts of image transformations, i.e., on the left side of the figures, there is a greater difference between the descriptors than for larger amounts. This is nearly true for all transformations, except light change. In the light change scene all descriptors perform relatively stably, i.e., the curves are almost horizontal. So the differences between the detectors remain also stable. Also interesting in the case of the light change scene is that although the number of correct matches decreases for larger amounts of decreasing light, the matching score does not. This means that the number of detected regions decreases in the same way as the correct matches.

The best performance is achieved by SIFT descriptors, followed by the GLOH and the Shape context description method. In the middle are Cross correlation and PCA, where Cross correlation is slightly better than PCA. Moments and Steerable filters show the worst performance. In the performance evaluation of local descriptors by Mikolajczyk and Schmid [22] similar results were obtained, although the results were more discriminative among the descriptors. SIFT and GLOH descriptors performed best, but the ordering was reversed. Shape context descriptors showed also a good performance. The descriptors were tested on Harris points, Harris-Laplace regions, Hessian-Laplace regions, Harris-Affine regions and Hessian-Affine regions.

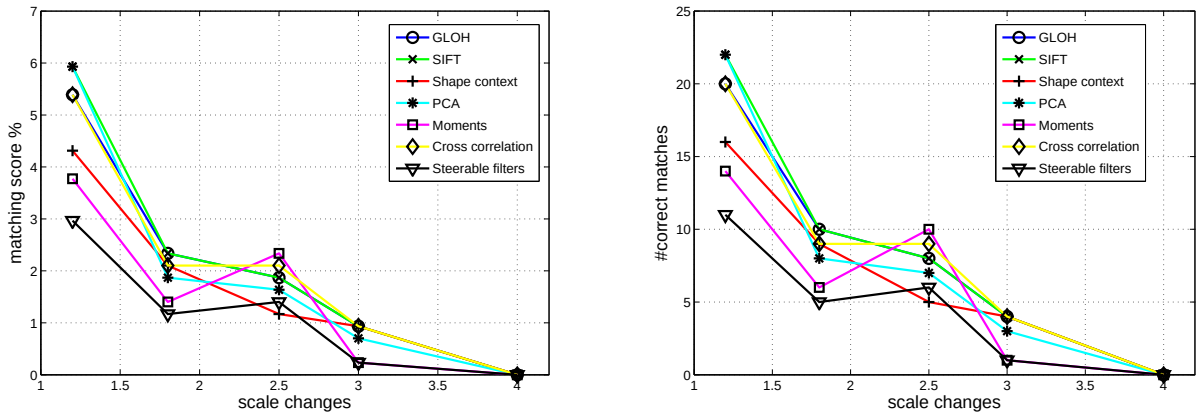
Conclusions

The results of the descriptor evaluation show that no description method is outstanding. This means that the choice of the descriptor is not as influential as the choice of the detector where there are greater differences between the several methods. The performance of all descriptors are relatively similar but nevertheless imply a ranking of the description methods. The explicit ordering of the descriptors, where the best results are obtained by the SIFT description method, support the choice for SIFT descriptors by Newman and Ho [23].



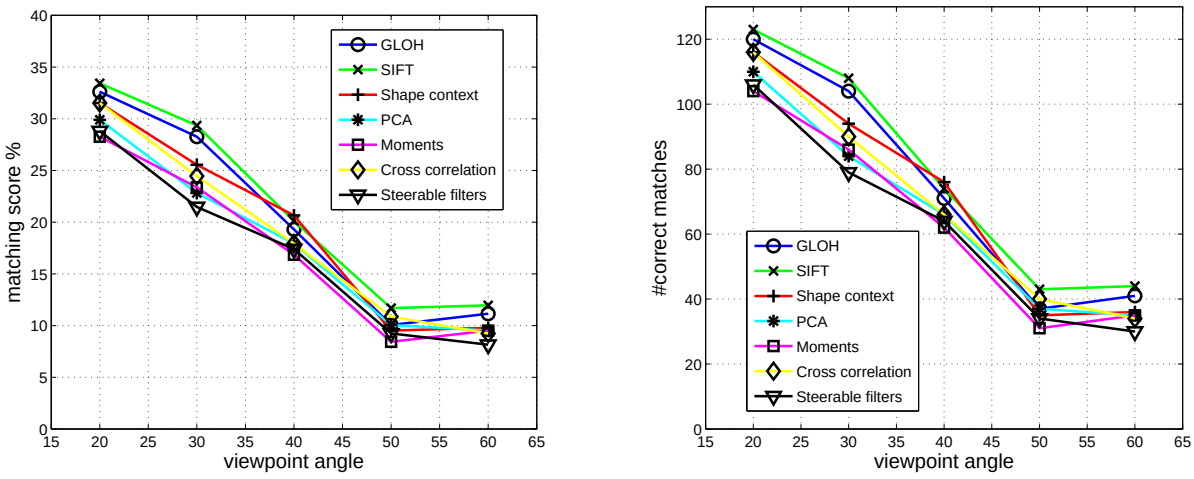
(a) (b)

Figure 3.12: Scale change transformations for the structured boat scene from Figure 3.1(a). (a) Matching score. (b) Number of correct matches.



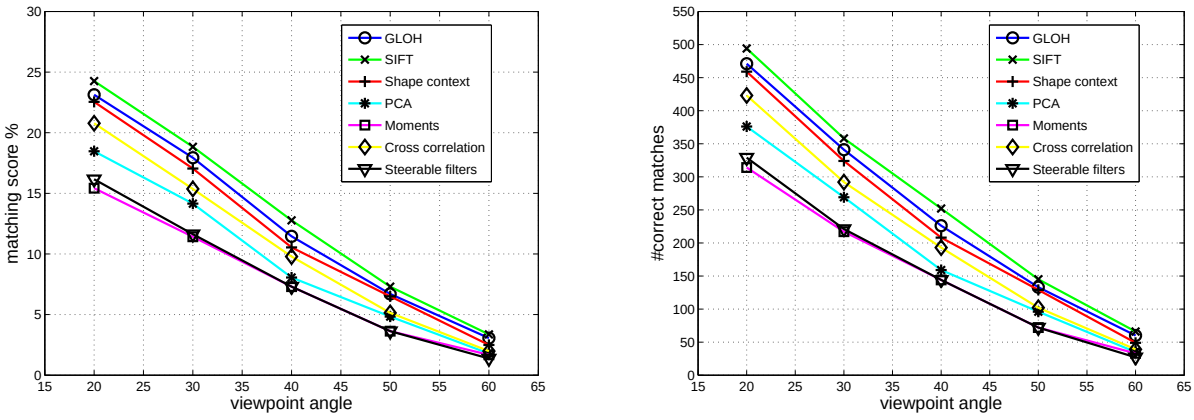
(a) (b)

Figure 3.13: Scale change transformations for the textured bark scene from Figure 3.1(b). (a) Matching score. (b) Number of correct matches.



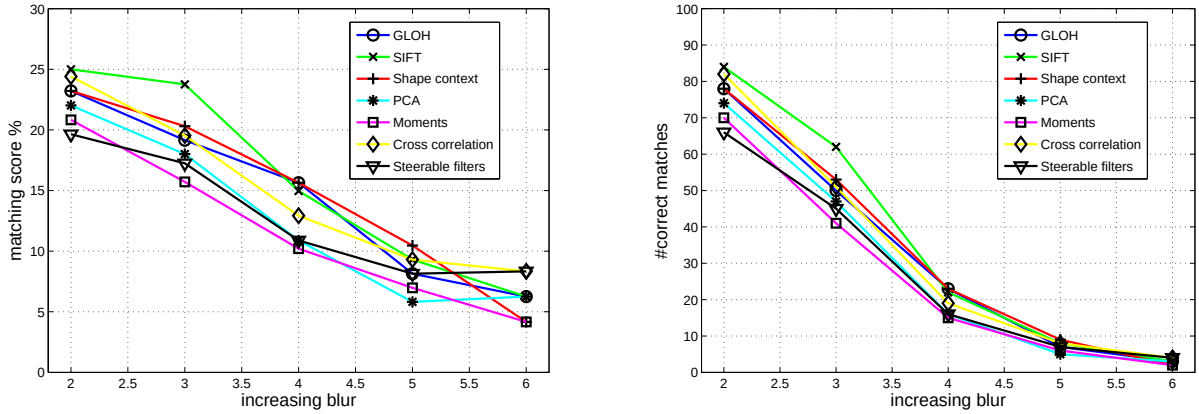
(a) (b)

Figure 3.14: Viewpoint change transformations for the structured graffiti scene from Figure 3.1(c). (a) Matching score. (b) Number of correct matches.



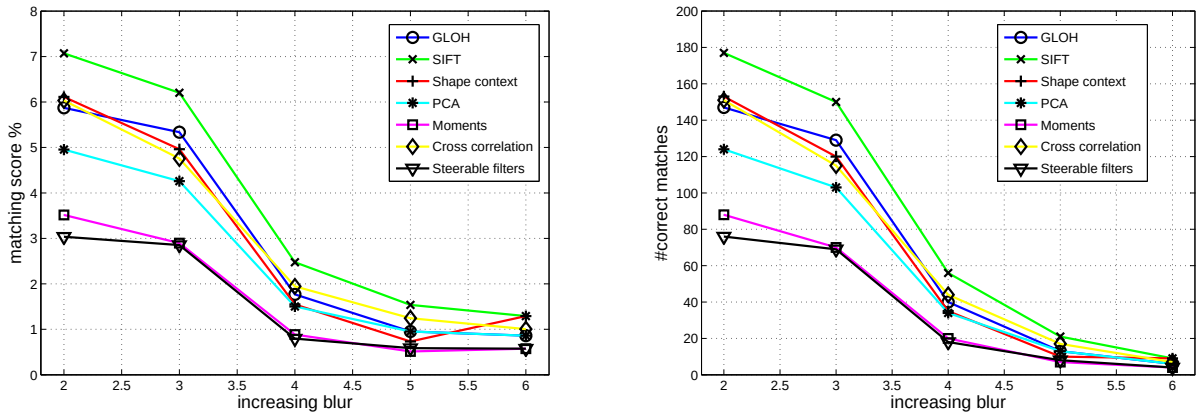
(a) (b)

Figure 3.15: Viewpoint change transformations for the textured wall scene from Figure 3.1(d). (a) Matching score. (b) Number of correct matches.



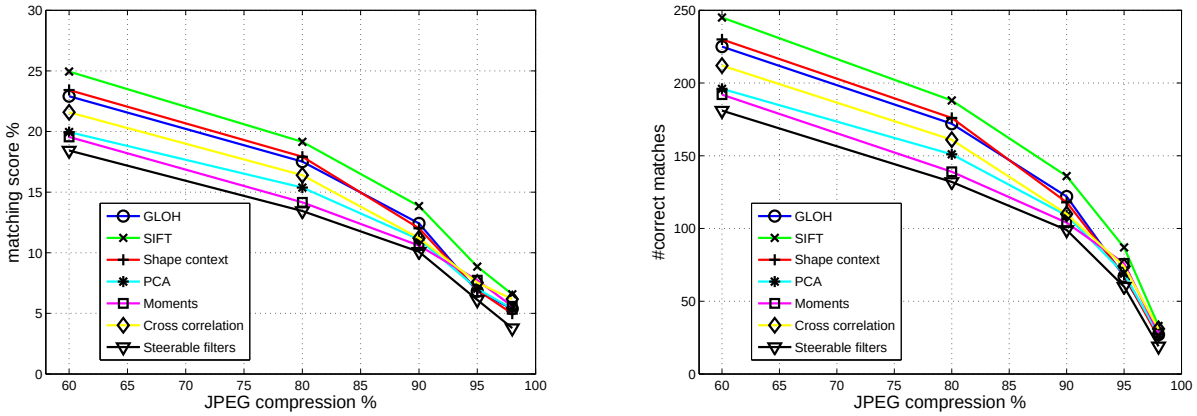
(a) (b)

Figure 3.16: Blur transformations for the structured bike scene from Figure 3.1(e). (a) Matching score. (b) Number of correct matches.



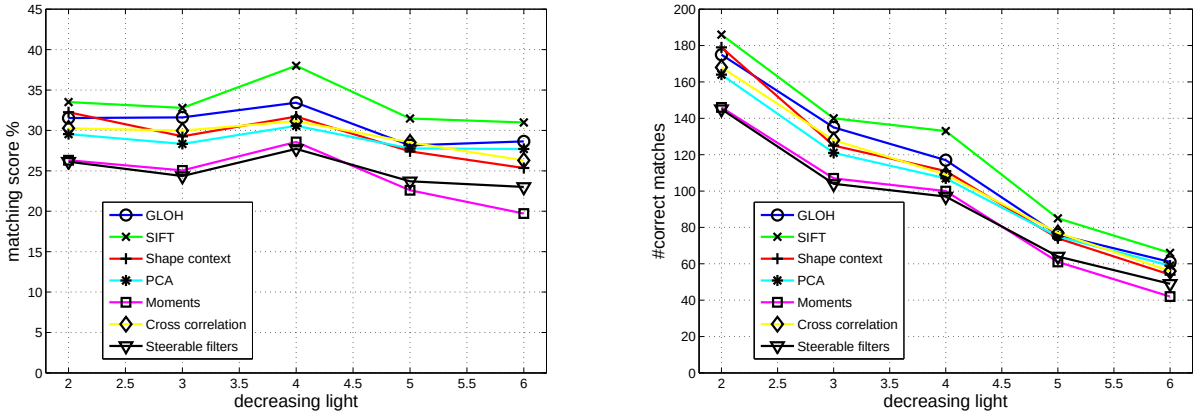
(a) (b)

Figure 3.17: Blur transformations for the textured tree scene from Figure 3.1(f). (a) Matching score. (b) Number of correct matches.



(a) (b)

Figure 3.18: JPEG compression transformations on images from Figure 3.1(g). (a) Matching score. (b) Number of correct matches.



(a) (b)

Figure 3.19: Light change transformations on images from Figure 3.1(h). (a) Matching score. (b) Number of correct matches.

Chapter 4

Experiments on a 6D-SLAM Robot Platform

Applying the loop closing procedure on a robot platform is another objective of this thesis. As an experiment, the loop closing application that is shortly described in Section 2.4 is integrated and tested on the existing 6D-SLAM robot platform Kurt3D. The robot platform is described in Section 4.1. The experimental setup and the results are presented in Section 4.2 and Section 4.3, respectively.

4.1 6D-SLAM Robot Platform Kurt3D

Figure 4.1 shows two images of the robot platform Kurt3D. The 6 wheels are powered by a 90W motor. The maximal possible speed is 1.2. The platform has a length of 45 cm, a width of 33 cm, a height of 26 cm and a weight of 15.6 kg. The main sensor of the robot is a 3D laser scanner which increases the height of the robot to 47 cm and the weight to 22.6 kg. The scanner is built on the basis of a 2D Sick laser scanner that is rotatable. Different resolutions are adjustable for the scanning area of $180\,^{\circ}$(h) $\times$ $120\,^{\circ}$(v). A horizontal scan of 181 data points is acquired in 13 ms. For example, a scan with 181×256 data points takes 3.4 seconds. As additional sensors the robot is equipped with 2 cameras. While the data of laser scanner is mainly used for the generation of a 3D map, the camera data is used to help close the loop. The robot is controlled by software that runs on a Linux-based laptop.

4.2 Experimental Setup

Basically the Kurt3D robot drives around and takes pictures of the environment with its camera. These images are processed with the feature detection and description algorithms explained in Section 2.2. The resulting feature descriptors are stored in a database and matched against descriptors from previously taken images. If the matching of several feature descriptors of two images is successful, a loop hypothesis is generated. In the following, some aspects of the description above are explained in more detail.

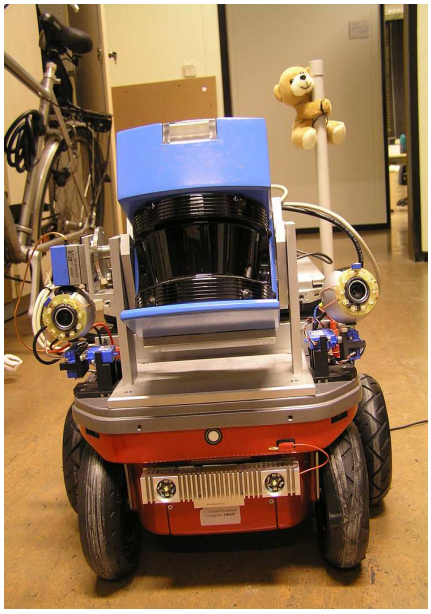


Figure 4.1: The robot platform Kurt3D.

Cameras. As described in Section 4.1 the Kurt3D robot is equipped with two cameras. Obviously the presented loop closing application from Section 2.4 is adaptable so that two or more cameras are used simultaneously. More cameras are clearly an advantage because the field of vision is augmented. On the other hand, the feature matching possibly needs to be adjusted, since all cameras contribute their described features to the same database. Of course, it is desired that the described features of all cameras are compared among each other, but if there is an overlap in the visual field of at least two cameras, it is not desired that features of these cameras that are acquired at the same time are matched. Since the usage of multiple cameras is not of main interest in this thesis, for reasons of simplicity only one camera is used in the experiments. The image size is set to 320×240.

Capture Times. A crucial factor for the success of the application of loop closing is the time interval between the capture of two images. If the robot only takes a picture every half an hour, it is clearly insufficient for the task of loop detection, because it barely takes two pictures of the same location. On the other hand, a robot that takes thirty pictures a second does not succeed either. In the last case the application detect one loop after another since the scene is not changing significantly. Hence a successful application uses a well-balanced interval time between the capture of two images. For an on-line application on an autonomous mobile robot the minimal time interval is bounded by the processing time of one image. In the experiments different time intervals are tested in order to determine an efficient one. All experiments are done offline, in order to test different time intervals and different matching thresholds, as discussed in the next paragraph, on the same input data.

Matching Thresholds. There are two influential thresholds in the process of feature matching. The first one is the threshold for the Euclidean distance measure. That means, for which distance are two matched feature descriptors considered as similar. The second is the minimum number of similar features descriptors that are needed to generate a loop hypothesis. In their work [23] Newman and Ho say nothing about the first threshold, the second one was fixed to three in order to deal with false positives. In these experiments, different values are tested for both thresholds.

Computation Time. As mentioned in Section 3.3.1 the computation time of the salient region detector is more than 30 minutes for an image of size 800×640. This means, that the detector is clearly not applicable on-line on an autonomous mobile robot. Therefore the parameters of the scale-saliency algorithm are adjusted. The minimum scale is set to 5 and the maximum scale is set to 10. The configuration for the evaluation was 3 for the minimum and 30 for the maximum scale. Due to this adjustment and due to the fact that smaller images are processed in the experiments the computation time is reduced to 10-15 seconds per image.

4.3 Experimental Results

In this section the experimental results are reported. In the experiments the robot was driven twice round a loop in an office environment. The principal goal of this scenario was to test the application of loop closing in general.

The robot took pictures as fast as possible, i.e., round about 10 pictures per second. Most pictures were acquired while driving. From these tons of data every 30th picture was processed for the task of loop detection. Since the robot did not drive in a stop-sense-go sequence, but rather in a sense-while-driving sequence, such a high processing rate makes sense, although it is not applicable on-line.

In the matching process, the minimum number of similar feature descriptors were varied between 2 and 3. For each number, different thresholds for the Euclidean distance measure were tested. The number of generated loop hypotheses are reported for different thresholds in terms of true and false positives in Tables 4.1 and 4.2. A successful loop detection counts as true positive, whereas a wrong hypothesis count as false positive. The ground truth information was provided manually, that means, that an operator decided whether two images showed the same scene. Since the robot was driven twice round the loop, there were many potential scenes for a true positive. Figure 4.2 shows two examples of successful loop detections. In some cases a true positive was counted, but not all features were matched correctly. An example for the last case is shown on the right side of Figure 4.3. On the left side of the same figure a false positive is depicted. While in most cases a false positive can relatively easy be identified by an operator, it is really hard to tell here.

Table 4.1 shows the results for at least 3 similar matched feature descriptors. The minimal distance between the feature descriptors were varied between 250, 220 and 200. The number of processed images was 53. The ratio between true and false positives changes significantly for the tested thresholds. While there are more false positives than true positives for a distance of 250, for 220 and 200 the opposite is true. Here, the smallest threshold leads to the best performance

Figure 4.2: Examples for two true positive loop hypotheses.

Table 4.1: Results with at least 3 matched feature descriptors. The threshold for the distance is denoted with d.

	$d = 250$		$d = 220$		$d = 200$	
# Images	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.
53	41	67	27	21	18	5

Table 4.2: Results with at least 2 matched feature descriptors. The threshold for the distance is denoted with d.

	$d = 200$		$d = 170$		$d = 150$	
# Images	True Pos.	False Pos.	True Pos.	False Pos.	True Pos.	False Pos.
53	30	30	17	3	8	1

Figure 4.3: The left picture shows an example for a false positive. Although the scenes looks similar, the images show two different locations. On the right picture a true positive is shown, but one feature descriptor was matched incorrectly.

of the loop closing application.

The results for at least 2 matched feature descriptors are reported in Table 4.2. Since the number of required matched features is reduced, the thresholds for the distance are reduced also in order to avoid false positives. In general the characteristic of the numbers for the tested distances are similar. As the ratios between the true positives and false positives suggest the best performance is achieved for the two smallest distances. In comparison to the test run that required 3 matched features there are less features that were matched incorrectly.

Chapter 5

Discussion and Open Issues

In this chapter, the evaluation results of Chapter 3 and the experimental results of the previous Chapter 4 are discussed and conclusions are drawn. The tasks of feature detection and feature description are considered in Section 5.1 and Section 5.2, respectively. The application of loop closing is analyzed in Section 5.3. Aspects that are noteworthy for future work are presented in Section 5.4.

5.1 Feature Detection

In [23] Newman and Ho suggest two criteria for the feature detection, namely saliency and wide-baseline stability. In this thesis the Salient-MSER detector fulfills the above criteria. It is tested in the evaluation framework developed by Mikolajczyk and Schmid [22].

The results of the evaluation show that the performance of the Salient-MSER detector is in general not different from the performance of the MSER detector. The reason is that the combined detection method uses only a subset of MSERs. The subset consists of the regions that are salient and wide-baseline stable. So MSERs that are not salient are not considered for further processing. As the results show, the additional criteria of saliency neither improve nor worsen the performance of the MSER detector significantly. For the matching test, the results for Salient-MSER detector are only slightly better than for the MSER detector.

This leads to the question whether the performance of a detector can simply be improved by building the intersection between its detected regions and the regions detected by another feature detector. Since most feature detectors are complementary, a concurrent operation clearly make sense. The question is how the various qualities of each feature detector can be deployed in order to build a robust method. Instead of taking the intersection of the detected regions of two or more feature detectors, optimizing each detector's method and unifying the results is an alternative approach. In this way the strengths of each detector are preserved. For example, a detector that has a good performance for increasing blur on structured scenes like the EBR detector can be combined with the Salient region detector that shows good results for increasing blur on textured scenes.

How to determine and enhance the performance of a combined feature detection method is an interesting question, but beyond the scope of this thesis. Here it is shown that the combination

of the two criteria, saliency and wide-baseline stability, does not lead to a significant performance improvement.

5.2 Feature Description

SIFT descriptors are used in [23] to encode the detected feature regions and store them in a database. Despite the SIFT description method other approaches are tested on the regions detected by the Salient-MSER detector in the publicly available evaluation framework [22].

The descriptor evaluation shows a similar performance for all tested descriptors. Nonetheless a ranking between the description methods is observable. The SIFT description method performs best on nearly all tested scenes. A similar result was obtained in the performance evaluation on various descriptors by Mikolajczyk and Schmid [22]. So using SIFT descriptors is a sensible choice for the feature description.

In addition to the good performance, SIFT descriptors have the desired properties that they are encoded in a compact way and that they are highly distinctive. These aspects make them very attractive and popular in recent work [30].

5.3 Loop Closing

The loop closing is implemented as it is described in Section 2.4. In general the performance of the application in the experiments was good. In the following some general considerations about the application of loop closing are stated.

Loop closing that only relies on temporal information as it is proposed by Newman and Ho [23] is different from other methods that use information about the robot position to figure out when closing a loop is a possibility. At first glance, it seems attractive to use information about the position of the robot because it is often available for no extra cost, since it is used for robot localization, it does not have to be acquired in an expensive manner. Obviously this is not true for approaches that use, for example, a hand-held video camera like it is used in [2,24,25,27], but that is not the point here. The point is rather that methods that use the estimated robot position have to deal with errors made at earlier stages in the process of map building. These errors do not have to be large. Many small errors can sum up to one gross error. If a robot does not take the possibility of loop closing into account because of its wrongly estimated position, it is likely that it maps a previously visited area incorrectly, because no loop is detected. So, the idea of the exclusive usage of temporal information tries to overcome the potential problem of a self-made error by a wrong position estimation.

The problem in the example above is that there is a potential loop but it is not detected. The other problematic scenario is that the robot is at a position in the environment where it has not been before, but nevertheless the robot generates a loop hypothesis. One reason for this is recurring structures in the environment. For example, in an office environment doors in a hallway often look the same. Taking a detected loop for granted possibly results in an incorrect map. To deal with this kind of problem, a verification of the loop hypothesis with additional information is an alternative. Geometrical information from a laser range scanner is used additionally to support a generated loop hypothesis in [5].

To recapitulate, there are two problematical cases in the application of loop closing: a loop is detected where there is no loop and no loop is detected although the robot is at a position where is has been before (false positives and false negatives). In their work [23] Newman and Ho tackle the problem of the the second case. Ignoring the information about the robot position reduces the chance of making a wrong decision based on erroneous map data. The other case is no less important. As the results of the experiments from the previous chapter suggest, the number of false positives is reducible by adjusting the thresholds for the matching process.

5.4 Open Issues

There are several issues that can be addressed in future work on loop closing. First, the threshold setting in the experiments was done by hand. This is an awkward task, a solution to this problem are thresholds that can be learned automatically from processed data. Second, the search over the feature descriptors is linear. Efficient algorithms and data structures can be used to decrease the complexity. Third, the computation time of the scale-saliency algorithm can be reduced by analyzing the influence of the scale on the task of loop closing. Finally, the implementation of loop closing proposed here is based on single visual features only. It is possible to incorporate other methods such as object recognition to achieve a more robust performance.

Bibliography

[1] David M. Cole, Alastair R. Harrison, and Paul M. Newman. Using Naturally Salient Regions for SLAM with 3D Laser Data. *IEEE International Conference on Robotics and Automation*, 18-22 April 2005.

[2] A. W. Fitzgibbon and A. Zisserman. Automatic 3D Model Acquisition and Generation of New Images from Video Sequences. In *Proceedings of European Signal Processing Conference (EUSIPCO '98), Rhodes, Greece*, pages 1261–1269, 1998.

[3] W. T. Freeman and E. H. Adelson. The Design and Use of Steerable Filters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[4] Luc J. Van Gool, Theo Moons, and Dorin Ungureanu. Affine/ Photometric Invariants for Planar Intensity Patterns. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 642–651, London, UK, 1996. Springer-Verlag.

[5] K. Ho and P. Newman. Combining Visual and Spatial Appearance for Loop Closure Detection. *Proceedings of European Conference on Mobile Robotics*, September 2005.

[6] Kin L. Ho. Using Visual Saliency and Geometric Sensing for Mobile Robot Navigation. 2004.

[7] T. Kadir, D. Boukerroui, and J. Brady. An analysis of the Scale Saliency algorithm, 2003.

[8] T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *Proceedings of the 8th European Conference on Computer Vision, Prague, Czech Republic.* sp, May 2004.

[9] Timor Kadir and Michael Brady. Saliency, Scale and Image Description. *Int. J. Comput. Vision*, 45(2):83–105, 2001.

[10] Yan Ke and Rahul Sukthankar. Pca-sift: A more distinctive representation for local image descriptors. *CVPR*, 02:506–513, 2004.

[11] J J Koenderink and A J van Doom. Representation of Local Geometry in the Visual System. *Biol. Cybern.*, 55(6):367–375, 1987.

[12] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. A sparse texture representation using local affine regions. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 27(8):1265–1278, 2005.

[13] David G. Lowe. Object Recognition from Local Scale-Invariant Features. In *Proc. of the International Conference on Computer Vision ICCV, Corfu*, pages 1150–1157, 1999.

[14] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[15] J Matas, O Chum, U Martin, and T Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *Proceedings of the British Machine Vision Conference*, volume 1, pages 384–393, London, 2002.

[16] K. Mikolajczyk and C. Schmid. An Affine Invariant Interest Point Detector. In *ECCV (1)*, pages 128–142, 2002.

[17] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.

[18] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A Comparison of Affine Region Detectors. *Int. J. Comput. Vision*, 65(1-2):43–72, 2005.

[19] Krystian Mikolajczyk and Cordelia Schmid. Indexing based on scale invariant interest points. In *International Conference on Computer Vision*, volume 1, pages 525–531, 2001.

[20] Krystian Mikolajczyk and Cordelia Schmid. Comparison of affine-invariant local detectors and descriptors. In *12th European Signal Processing Conference, Austria*, 2004.

[21] Krystian Mikolajczyk and Cordelia Schmid. Scale & Affine Invariant Interest Point Detectors. *Int. J. Comput. Vision*, 60(1):63–86, 2004.

[22] Krystian Mikolajczyk and Cordelia Schmid. A Performance Evaluation of Local Descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005.

[23] Paul M. Newman and Kin L. Ho. SLAM - Loop Closing with Visually Salient Features. *IEEE International Conference on Robotics and Automation*, 18-22 April 2005.

[24] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual Modeling with a Hand-held Camera. *International Journal of Computer Vision*, 59:207–232, 2004.

[25] M. Pollefeys, R. Koch, M. Vergauwen, and L. Van Gool. Hand-Held Acquisition of 3D Models with a Video Camera. *3dim*, 00:0014, 1999.

[26] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

[27] Tomokazu Sato, Masayuki Kanbara, Naokazu Yokoya, and Haruo Takemura. Dense 3-D Reconstruction of an Outdoor Scene by Hundreds-Baseline Stereo Using a Hand-Held Video Camera. *Int. J. Comput. Vision*, 47(1-3):119–129, 2002.

[28] Frederik Schaffalitzky and Andrew Zisserman. Multi-view Matching for Unordered Image Sets, or "How Do I Organize My Holiday Snaps?". In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 414–431, London, UK, 2002. Springer-Verlag.

[29] Robert Sedgewick. *Algorithms, 2nd Edition.* Addison-Wesley, 1988.

[30] R. Sim, P. Elinas, M. Griffin, A. Shyr, and J. J. Little. Design and analysis of a framework for real-time vision-based SLAM using Rao-Blackwellised particle filters. In *Proceedings of the 3rd Canadian Conference on Computer and Robotic Vision (CRV)*, Québec City, QC, June 2006. CIPPRS, IEEE Press. 9 pages, electronic proceedings. **Best Robotics Paper**.

[31] Hartmut Surmann, Andreas Nüchter, Kai Lingemann, and Joachim Hertzberg. 6D SLAM - Preliminary Report on Closing The Loop in Six Dimensions. In *Proceedings of the 5th Symposium on Intelligent Autonomous Vehicles (IAV '04), Lissabon, Portugal*, June 2004.

[32] S. Thrun. Robotic Mapping: A Survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium.* Morgan Kaufmann, 2002.

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Osnabrück, October 2006