Loop Closure Detection using Depth Images

Sebastian A. Scherer* Alina Kloss* Andreas Zell*
*Department of Computer Science, University of Tuebingen, Tuebingen, Germany

***Abstract*— We investigate the question whether loop closure detection using depth images is feasible using currently available depth features. For this reason, we collected a benchmark dataset consisting of a total number of 15 logfiles with several loops in various environments, implemented a modular and easily extensible loop closure detector and used this to evaluate the adequacy of state-of-the art depth features on our benchmark dataset. To allow for a fair comparison, we determined the best values for the sometimes large number of user-chosen parameters using a large-scale grid search. Since our benchmark dataset contains both depth and RGB images, we can compare the performance relying on depth features with the performance achieved when using intensity image features.**

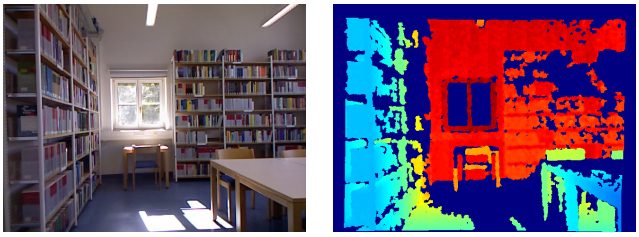
I. INTRODUCTION

Starting with the introduction of Microsoft's Kinect, high-resolution depth cameras became affordable and thus widely-used in robotics and robot vision. While the Kinect and its predecessor also included a regular color camera (see Fig. 1 for an example), there are now more compact devices like the Asus Xtion PRO that only consist of a depth camera.

Using depth cameras as main devices for low-cost mobile robots navigating in unknown environments and solving the SLAM (simultaneous localization and mapping) problem immediately comes to mind. There already is some interesting work on fast registration for egomotion estimation using depth images [1], but the still missing component towards a full SLAM system using depth images is loop closure detection.

There are, however, a plethora of attempts to visually detect closed loops based on regular (intensity) camera images [2], [3], [4]. The general consensus about the approach chosen for visual loop closure detection seems to contain first finding the previous images that are most similar to the current image (based on a content-based image retrieval scheme) and afterwards discarding those that cannot be loops due to heuristics or geometry consistency checks. The most commonly used image retrieval scheme is Bag of Visual Words (see sect. III for more details).

There are some alternatives to using Bag of Words for loop closure detection that rely on using raw image descriptors [4] or locality-sensitive hashing (LSH) [5]. In both cases, the authors report to obtain much better results using alternative approaches than using Bag of Words. We are not so sure how representative that comparison is, however, due to very small datasets used and the fact that Bag of Words was used with very small vocabulary sizes: at most 2500 words were used in [5] and at most 8000 in [4], as opposed to 1 million in e.g. [6].



(a) RGB image (b) Depth image

Fig. 1. Example RGB and Depth Image Pair

II. LOCAL FEATURES FOR DEPTH IMAGES

We evaluate various interest point detectors and local descriptors for depth images. Some algorithms contain both interest point detection and local descriptor extraction, e.g. Normal Aligned Radial Features (NARF) [7] or SIFT [8]. In general, there are two classes of algorithms: Those which were developed for depth images exclusively and those which are adaptations of known algorithms for intensity images.

A. Interest Points

Interest points or keypoints are image locations in regions that contain a relatively high amount of information (i.e. texture in intensity images). They are usually desired to be distinctly located, which is typically the case for local extrema or corners, and robust to changes of perspective.

1) Harris: The Harris corner detector [9] is a well known interest point detector for intensity images. It evaluates the variance of the intensity around each point. For a corner and therefore a stable interest point, intensity variation in both x and y direction needs to be high. This corresponds to both eigenvalues of the so called *Harris matrix* H of the patch having large positive values. The response C is usually computed without explicitly determining the eigenvalues of H:

$$H = \begin{bmatrix} \widehat{I_x^2} & \widehat{I_x I_y} \\ \widehat{I_x I_y} & \widehat{I_y^2} \end{bmatrix} \tag{1}$$

$$C = |H| - k \cdot \text{trace}(H) \tag{2}$$

Where I_x denotes the x-component of the intensity gradient and $\widehat{\cdot}$ is a shorthand for the weighted sum over a certain influence window. C above is the so-called Harris response. An alternative response measure is the Shi and Tomasi response: $C = \min(\lambda_1, \lambda_2)$, where λ_i are the eigenvalues of H. When using depth images instead of intensity images,

the image gradients are roughly proportional to the x and y components of the surface normals.

Keypoints detected by the Harris detector tend to lie directly on the boundaries of objects. This works fine in continuous 2D images but can easily become a problem with depth images, as the border regions are often unstable with regard to both surface normal estimation and depth measurement.

2) FAST: Like Harris, the FAST keypoint detector [10] was originally developed for intensity images. Its main idea is to test the intensities of pixels on a Bresenham circle of 16 pixels diameter around each keypoint candidate p. If the patch around p contains a corner, there should be at least $n = 12$ consecutive pixels on that circle that are either all darker or all brighter than p by some threshold t. The order of pixel tests is optimized such that non-corners can be eliminated as soon as possible. Instead of using FAST on intensity values, we can also easily "abuse" it and apply it to depth values.

3) NARF: [7] NARF interest point detection begins with border detection: For each point it tests whether there is a border of an object above, below, to the left or to the right of it. If a point has a border in its neighbourhood it is assigned the direction of the border as its dominant direction of surface change, otherwise the first principal component of the curvature is used. The interest value of a point is then influenced by two aspects:

- To encourage keypoints on stable surfaces, the interest value of a point is decreased if there are points with strong surface changes nearby.
- Points with pairs of neighbours with different directions of surface have their interest values increased.

As a result, NARF interest points can be found near the borders of objects but not directly on these borders, which is usually good for depth images or point clouds.

B. Local Descriptors

Local descriptors are designed to describe an image patch (usually located around an interest point) in a compact but distinctive way. Similar to the location of interest points, a point's descriptor is usually desired to be invariant to changes in scale and perspective.

1) NARF: The main idea of the NARF descriptor is similar to 2D algorithms like SIFT: For each keypoint, a normal aligned range value patch is computed that is orthogonal to the surface normal of the point. Changes in depth values are computed along 36 directions around the keypoint and weighted by their distance to it. A unique orientation is obtained by computing a direction histogram over the descriptor values. The bin with the maximum value is selected as the dominant orientation and the descriptor can be made rotational invariant by rotating (shifting) the histogram by this orientation. It should be noted that the algorithm will compute multiple descriptors at a single keypoint if there is more than one bin within the histogram with values exceeding 80% of the maximum. Because of this, NARF will often produce more descriptors than there are keypoints for a given image.

2) Kernel Descriptors: Bo et al. in [11] introduce a kernel view of SIFT and HOG features and demonstrate that comparing HOG descriptors can be interpreted as computing a linear match kernel that combines two subkernels comparing gradient orientation and magnitude of all pixel pair combinations. They propose a slightly different orientation kernel and add a gaussian position kernel to arrive at what they call gradient kernels. The problem of these kernel features is the fact that they are generally infinite-dimensional. Bo et al. propose sampling basis vectors over a fine grid to obtain finite-dimensional features. Those are still too high-dimensional to be of any use, so a kernel PCA is used to compact them to a 200-dimensional feature space. They show that this notion of kernel descriptors can also be applied to come up with completely new descriptors, e.g. color or shape kernels and kernel descriptors can successfully be applied to depth images [12].

3) BRIEF: [13] BRIEF is a very simple descriptor in the form of a binary string. It was developed for intensity images to allow for fast computation, efficient storage and also fast comparison by using the Hamming distance instead of the common L_2 norm. To obtain the descriptor, the intensity values of several point pairs in the neighbourhood of a keypoint are compared after smoothing the patch to reduce noise. The descriptor cell is then either 1 or 0, depending on which of the points had the higher intensity.

III. BAG OF WORDS

Bag of Visual Words (BoW), also referred to as Bag of Keypoints ([14]), is a technique widely used in computer vision to compute a single global descriptor of an image, given an arbitrary number of descriptors of local features found within this image. The general idea is rooted in document processing, where documents of arbitrary length can be represented using a global descriptor by counting occurrence of a finite number of n keywords (i.e. the vocabulary). The resulting histograms, which we will call BoW vectors from now on, can be interpreted as an n-dimensional global descriptor of a document and efficiently compared using any vector norm.

In computer vision, representative feature descriptors are used as visual words. A visual vocabulary, i.e. a set of representative feature descriptors, can be computed from a large set of example features by clustering, typically k-means clustering. Computing the BoW vector of a given image consists of extracting all local features, determining the corresponding visual word for each feature (e.g. using fast linear approximate nearest neighbour search), and finally counting the occurrence of each visual word.

The utility of Bag of Visual Words heavily depends on the vocabulary size. The authors of [6] propose vocabularies that contain on the order of millions of visual words and claim that using such large vocabularies, image retrieval works accurately even without considering the geometric layout of visual words. In order to cope with this large vocabulary sizes, they suggest hierarchical vocabulary trees: During vocabulary creation, k-means clustering is performed

recursively on multiple levels up to a maximum tree depth, clustering the set of all sample features that belong to one cluster in the previous level in turn. Looking up the corresponding visual word for a given feature involves traversing the vocabulary tree down to a leaf node, finding the closest representative descriptor on each level.

IV. LOOP CLOSURE DETECTION

Since the main focus of this work is evaluating different features for loop closure detection, we employ only a very basic loop closure detection method. For each image, it will do the following steps:

1) Query the database of previous images to find and return the most similar ones using Bag of Words as described in sect. III.
2) Disregard all resulting candidates that belong to the 10 latest keyframes in order to prevent detecting loops already when the robot is still at the same location.
3) Compute the similarity s of the current image with each resulting candidate based on their BoW vectors v_1, v_2 and disregard those whose similarity is below a user-chosen threshold α. We use the similarity measure described in [15], which is based on the L_1 norm:

$$s(v_1, v_2) = 1 - \frac{1}{2} \left| \frac{v_1}{|v_1|_1} - \frac{v_2}{|v_2|_1} \right|_1$$

4) Return the up to k images with the highest similarity as loop closure candidates, where k can be chosen by user. The choice of k will depend on how many candidates can be further verified by checking the geometric consistency within a reasonable amount of time.
5) Finally, the current image is added to the database so it can later be found as a loop closure candidate.

One could think of many more heuristics to filter out more false positives in loop closure detection. Galvez-Lopez and Tardos in [16] propose many more heuristics, e.g. enforcing temporal consistency, i.e. requiring detection of the same loop multiple times in a row before it is actually reported.

The final step within a real SLAM system, however, should always be a geometric consistency check. This could involve trying to register loop closure candidates (e.g. using ICP) and determining their goodness of fit, or matching local features within a RANSAC scheme and counting the number of inliers. Also evaluating registration methods, however, would go beyond the scope of a single paper, so we evaluate the performance of the returned loop closure detection candidates without checking their geometric consistency.

V. IMPLEMENTATION

We implemented a highly modular (polymorphic) and easily configurable loop closure detector in C++. The user can choose among any of the interest point detectors and local descriptors mentioned in sect. II and arbitrarily set all of their parameters. Adding more interest point detectors or local descriptors is easy and only involves writing one more derived wrapper class, which will register with the corresponding object factories.

We use the DBoW2 library [15], a very efficient open-source implementation of the hierarchical vocabulary tree approach mentioned in sect. III.

For SIFT, FAST, and BRIEF, we rely on their implementations found in OpenCV[1]. For Harris and NARF their implementations in PCL[2] are used. As the C-Version of Kernel Descriptors[3] only allows the computation of dense Kernel Descriptors, i.e. sampled over a grid of overlapping patches, we modified the source code to allow computation of sparse Kernel Descriptors at given keypoint locations.

VI. BENCHMARK DATASET

In order to evaluate the performance of loop detection algorithms, we need sequences of depth images with ground truth pose information in environments as diverse as possible. To our knowledge, there is currently only one applicable dataset publicly available, published in [17], which features sequences of RGBD images with ground truth 6D pose information obtained by an external tracking system. The number of loops in this dataset, however, is not high enough for a proper evaluation of loop closure detection, so we additionally recorded our own data.

We used a Scitos G5 robot by Metralabs with a forward-looking Microsoft Kinect mounted on its top. Its 2D pose is obtained by localizing it within a previously built map using odometry and its laser range finder. We drove this robot on multiple loops through various environments available in our building. Our final set of sequences consists of:

- 5 runs within our robots laboratory and on the corridor just outside of it,
- 4 runs within our department's library,
- one run in which the robot enters different offices several times
- one run within our department's computer museum,
- one run within our kitchen,
- and 4 runs containing loops taken from the freiburg dataset[4] [17].

Instead of keeping full sequences of all these runs, we significantly reduce the number of images to a subset of keyframes, as it is usually sufficient to find loops for new keyframes only in keyframe-based SLAM.

VII. EXPERIMENTS AND RESULTS

A. Evaluation

1) Determining True Loops: In order to evaluate the performance of the loop closure detection the ground truth must be established first. Depending on the dataset, poses with three or six degrees of freedom were given for each image. We determined whether two images were taken at the same place by considering the euclidean distance of the

[1] http://opencv.org/

[2] http://pointclouds.org/

[3] http://www.cs.washington.edu/ai/Mobile_Robotics/projects/kdes/

[4] We use the logfiles named "freiburg1_360", "freiburg1_room", "freiburg2_large_with_loop", and "freiburg3_long_office_household"

(a) office (b) corridor

(c) library (d) laboratory

(e) kitchen (f) museum

Fig. 2. Examples of different scenes encountered by the robot.

translation and the angle between the orientations of both poses: Two images are counted as a loop if their translation is less than $2\,\mathrm{m}$ and their angle is less than $30°$.

As we allow a distance of up to $2\,\mathrm{m}$ between two images belonging to a loop, it must be ensured that the images were not taken on the same visit to this place. Therefore, there must be at least 10 keyframes between two images in order for them to be considered a loop.

2) Evaluating Classification Performance: At first glance, loop closure detection is a binary classification problem: Given a place described by an image and a database of images, decide whether the place has been visited before. Therefore it seems natural to evaluate the performance in terms of *sensitivity* (SE) and *specificity* (SP) and e.g. calculate a ROC curve (*receiver operating characteristic*). The problem at hand, however, is not only to decide whether a place has been visited before, but also to identify the place correctly by retrieving a matching image from the database. This makes the definition of *true positives* (TP), *false positives* (FP), *true negatives* (TN) and *false negatives* (FN) more difficult: If a place has been visited before and no image from the database is retrieved, its a clear *false negative*. If a correct image is found it is obviously a true positive. But if a wrong image is retrieved for a previously visited place this could be counted as *false positive* (the retrieved image was not taken at the current position) as well as *false negative* (as the loop was not detected correctly). We called these cases *wrong positives* (WP) and decided to count them towards the *false negatives*, as we consider an undetected loop a graver mistake than a false loop candidate. The reason for this is simply that false loop candidates could be filtered out by further checks as discussed in section IV whereas a missed loop is final. A diagram of the definitions can be seen in Fig. 3. Based on these numbers, we can compute the sensitivity (SE) and specificity (SP) in the following way:

$$SE = \frac{TP}{TP + FN^*}$$
$$SP = \frac{TN}{FP + TN}$$

Where we apply the sum of both, false negatives and wrong positives, instead of the classical false negative count:

$$FN^* = FN + WP$$

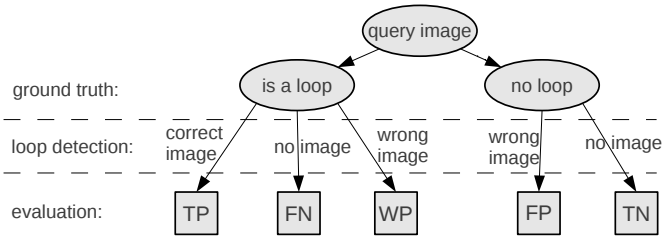


Fig. 3. Classification of loop candidates into categories for evaluation.

B. Parameter Optimization

The performance of each of the algorithms employed in loop closure detection (i.e. interest point detectors, local descriptors, Bag of Words, and the actual decision about loop closure detection) heavily depends on a certain number of parameters. Finding universally good values for these parameters is often impossible as the optimal values depend on the problem, the input data provided to the algorithm, and how its output is further processed. To allow a fair comparison, however, we need to find the best parameter values for the problem at hand, i.e. test the full system including vocabulary creation and loop closure detection for various parameter values.

Vocabulary creation involves k-means clustering, which is usually implemented using a randomized algorithm. In order to get meaningful results, we run it at least five times with the same parameters and compute the mean over all runs.

This means we need to evaluate our loop detection system on the benchmark dataset thousands of times, when a single run can take up to a few hours for computationally expensive local features. We utilized grid computing on a cluster of 8 nodes with 16 CPU cores each to run up to 128 evaluations in parallel.

Conceptually, the task of loop closure detection using Bag of Words consists of two major steps. The first step involves using local features and Bag of Words to suggest loop closure candidates. In this first step, our main objective is a high sensitivity: If there is a loop, we want it to be among the suggested candidates. In a second step, the actual loop closure detection is performed based on the candidates obtained in the first step. Here, we want to obtain a good tradeoff between specificity and sensitivity by tuning the receiver operator characteristic (ROC) curve.

1) Vocabulary size: We first try to determine a good vocabulary size using default parameter values of various methods used. The optimal vocabulary parameters of NARF and FAST interest points with kernel descriptors were a *branching factor* (i.e. the number of clusters during each clustering step) of $b = 20$ and *level* (i.e. the maximum tree depth of the resulting vocabulary tree) of $l = 3$, which corresponds to an efficient vocabulary size of $20^3 = 8000$. Using FAST with BRIEF descriptors, the optimal vocabulary size is smaller with $b = 5$ and $l = 5$, whereas the optimal vocabulary size of SIFT is orders of magnitude bigger with $b = 20$ and $l = 4$. Since we, however, wanted to focus on depth features and use a consistent vocabulary size, we decided to use $b = 20$ and $l = 3$ for further experiments, which seems to work well for all methods.

2) Reference: SIFT on Intensity Images: Since our benchmark dataset consists of RGBD image pairs with both color and depth images, we first try to detect loops using SIFT features computed on the intensity images as a reference.

We tested various values for the parameters *contrast threshold* $ct \in \{0.01, 0.02, 0.04, 0.08\}$, *edge threshold* $et \in \{2, 5, 10, 15\}$ and *sigma* $\sigma \in \{1.2, 1.4, 1.6, 2.0, 2.5\}$ and found that $ct = 0.01$, $et = 10$ and $\sigma = 2.5$ obtained the best sensitivity of 94.6%.

3) FAST & BRIEF: FAST has only one parameter (its threshold) and we decided to keep BRIEF's amount of bytes constant at 32 bytes. We thus only had to vary one parameter: The best FAST threshold $t \in \{6, 8, 10, 12, 14, 16, 18, 20, 22\}$ turned out to be $t = 6$, which resulted in the best sensitivity of 87.5%. The choice of the lowest value for the threshold t does not come as a surprise, since this basically means it will consider a rather high number of interest points.

4) NARF: NARF features require the user to choose a large number of features, which makes large-scale search on the full high-dimensional grid computationally intractable. We decided to always enable rotation invariance and fix the angular resolution to $ar = 0.3$, as this appeared to be the highest resolution for which we can still compute NARF features within a reasonable amount of time. We tested various values for the parameters support size $ss \in \{0.1, 0.2, 0.4\}$, minimum keypoint distance $mkd \in \{0.125, 0.25, 0.5\}$, optimal surface distances $osd \in \{0.125, 0.25, 0.5\}$, minimum interest values $miv = \{0.225, 0.45, 0.9\}$, minimum surface changes $msc = \{0.1, 0.2, 0.4\}$, and optimal patch size $ops = \{51020\}$. This search lead to the optimal values $ss = 0.1$, $mkd = 0.125$, $osd = 0.125$, $miv = 0.225$, $msc = 0.10$, $ops = 5$, with which we obtained a maximum sensitivity of $SE = 76.1\%$.

Interest Points	Descriptor	best Sensitivity
FAST	BRIEF	87.5 %
HARRIS	BRIEF	88.1 %
NARF	NARF	76.1 %
HARRIS	NARF	79.8 %
FAST	KDES	89.6 %
HARRIS	KDES	89.6 %
NARF	KDES	92.0 %
SIFT	SIFT	94.6 %

TABLE I

BEST SENSITIVITIES OBTAINED FOR $k = 3$ LOOP DETECTION CANDIDATES USING DIFFERENT INTEREST POINTS AND DESCRIPTORS

5) Harris Interest Points: We evaluated Harris interest points combined with various descriptors from other methods with their respective best parameters, optimizing the parameters radius $r \in \{0.1, 0.05, 0.025, 0.01\}$, threshold $t \in \{0.025, 0.01, 0.005, 0.0025\}$, and method $m \in \{Harris, Curvature, Tomasi, Noble\}$. Using BRIEF descriptors, the best sensitivity 88.1% was achieved for $r = 0.01$, $t = 0.0025$ and $m = Tomasi$. In combination with NARF descriptors, the best sensitivity 79.8% was also obtained for the same values $r = 0.0025$, $t = 0.01$ and $m = Tomasi$.

6) Sparse Kernel Descriptors: Since kernel descriptors do not provide their own interest point locations, we need to combine these with an interest point detector. Unfortunately, the source code of Kernel Descriptors does not include the functionality to train new kernel descriptors, so we only used the pre-trained ones that come bundled with the source code. They use a fixed patch size of 16×16 pixels. The only parameter we can modify directly is *low contrast*: The gradient magnitude for each image patch is normalized by dividing it by its L2-norm or *low contrast* if it is below *low contrast*. This protects from noise artifacts being amplified too much when there is not much structure. The original source code of kernel descriptors computes these descriptors on heavily downscaled versions of input images. We therefore introduced a new parameter *scale factor* by which images (and keypoints) are scaled before computing kernel descriptors.

We evaluated the performance of kernel descriptors in combination with FAST, Harris, and NARF interest points using their respective optimal parameters as determined before. We tried different values for $lc \in \{0.75, 0.9, 1.0, 1.1, 1.25\}$ and $rf \in \{0.5, 0.25, 0.125, 0.1, 0.05\}$ for each combination.

In all three cases, the same parameters $lc = 0.9$ and $rf = 0.25$ turned out as optimal, resulting in sensitivities of 89.6% for FAST and Harris interest points and 92.0% using NARF interest points.

7) Varying the Loop Closure Detection Parameter: Since we are typically not only interested in a high sensitivity but also a high specificity, we evaluated the loop closure detection method described in sect. IV using various values for α. This leads to one receiver operating characteristic

(ROC) curve for each interest point detector/descriptor pair, which are drawn in Fig. 4.

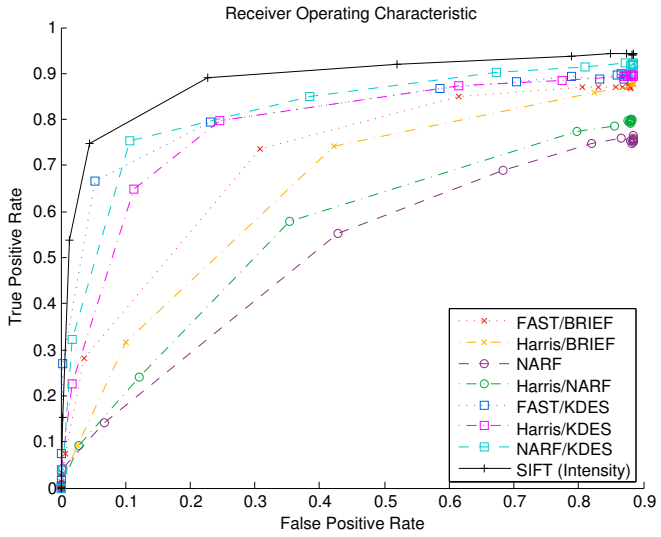


Fig. 4. Receiver operating characteristic.

8) Varying the Number of Considered Loop Closure Candidates: In the previous experiments, we considered $k = 3$ loop candidates for each image. If a correct loop is among these k, it counts as detected. The choice of $k = 3$ was made by us rather arbitrarily. In practise this parameter should be chosen depending on a combination of the desired sensitivity and how computationally expensive it is to verify a loop candidate geometrically (e.g. using iterative closest point or based on feature matches). The influence of k on the sensitivity is shown in Fig. 5. For this experiment, we ran each method 10 times instead of 5 times to obtain more meaningful estimates of the standard deviation, which is shown using error bars.

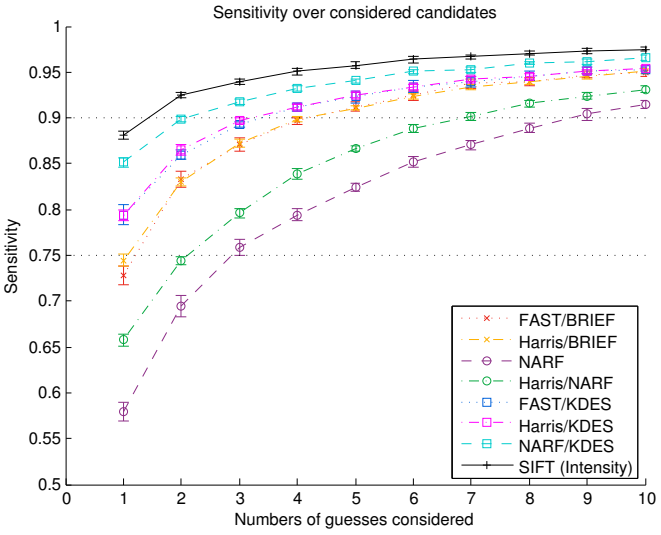
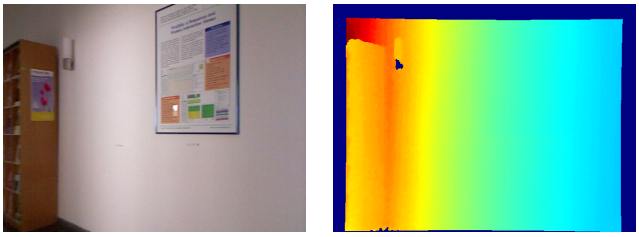


Fig. 5. Sensitivity and its standard deviation vs. the number of considered loop closure candidates.

VIII. CONCLUSION

A. Discussion

The results show that the performance of loop closure detection using depth images with any kind of depth features is always worse than when using intensity of color images. This should not come as a surprise, since intensity images typically contain more information useful for loop closure detection: When the camera is pointed towards a planar wall on a corridor for example, we cannot expect to reliably detect any loop using depth images alone. The intensity image in this case, however, might capture distinct texture information, e.g. of a unique poster on the wall, which makes detecting this loop much easier. An example of this scenario from our dataset is shown in Fig. 6.



(a) RGB image (b) Depth image

Fig. 6. RGB and depth image pair from the dataset. The RGB image clearly contains much more relevant information since the highly textured poster is not "visible" to the depth camera.

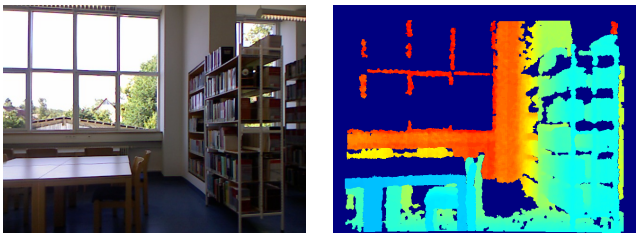
But the results also show that loop closure detection using depth images works surprisingly well. Even if we only consider the single most similar image obtained using the best combination of all methods tried, we can find close to 85% of all loops using NARF interest points with kernel descriptors or roughly 73% using the computationally very inexpensive combination of FAST interest points with BRIEF descriptors. This should be more than enough for a typical SLAM setting: When a robot enters a room for the second time, we can expect it to detect one of usually several possible loops eventually.

Considering the performance of the various individual methods tested, we can make the following interesting observations:

1) The choice of interest points is not as important as the choice of the descriptor: This is especially obvious from Fig. 5, where we find the curves of different interest points in combination with the same descriptor to be nearly identical. This is in parts due to how we posed our optimization problem in sect. VII-B: Since we try to find the best sensitivity and do not care about the number of interest points, we usually end up with low thresholds and thus high numbers of interest points. Comparing results using a fixed number of interest points might have been more fair, but enforcing a fixed limit is not trivial. Also, we usually do not care so much about the number of interest points as long as this does not make it computationally prohibitively expensive.

2) "Abusing" 2D methods originally intended for intensity images works surprisingly well compared to proper

3D features: The performance of the computationally least expensive combination of FAST interest points and BRIEF descriptors is only surpassed by kernel descriptors with different interest points, but works better than NARF. It appears that for loop closure detection, we do not need the additional information encoded in 3D features like NARF. This can be explained by the fact that true loops in our dataset consist of a pair of images that are captured by roughly the same pose. This means there is usually no considerable change in scale, orientation, or perspective in general. Additionally, regions with invalid depth measurements are usually reproducible, i.e. invalid depth measurements also contain useful information (see Fig. 7), whereas 3D features that rely on 3D points to recover surface normals will disregard these regions completely. Finally, 3D methods like Harris



(a) RGB image (b) Depth image

Fig. 7. Example image pair from our dataset: Borders between valid and invalid depth values contain useful information about the scene.

and NARF might work rather badly in our dataset because they suffer from many rather high depth values for which there is considerable depth inaccuracy.

B. Future Work

Since even the computationally very inexpensive methods FAST and BRIEF seem to obtain reasonable results, it might be interesting to combine loop closure detection with a fast registration technique to implement an computationally inexpensive full SLAM system that relies on depth images alone.

C. Summary

We evaluated several methods of interest point detection and descriptor extraction for the task of loop closure detection using Bag of Words on depth images. As expected, the achieved sensitivity is lower than what can be obtained on intensity images, but still high enough that it should be usable for SLAM based on depth images alone. It turns out that basic 2D features known from intensity images work surprisingly well for loop closure detection, which might be mainly due to a combination of important information being contained in invalid depth readings and no big changes in scale, orientation or perspective in general for real loops.

REFERENCES

[1] W. Lui, T. Tang, T. Drummond, and W. H. Li, "Robust egomotion estimation using ICP in inverse depth coordinates," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 1671–1678.

[2] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[3] K. Konolige and M. Agrawal, "FrameSLAM: From Bundle Adjustment to Real-Time Visual Mapping," *IEEE Trans. Robotics*, vol. 24, no. 5, pp. 1066–1077, 2008.

[4] H. Zhang, "BoRF: Loop-closure detection with scale invariant visual features," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 3125–3130.

[5] H. Shahbazi and H. Zhang, "Application of Locality Sensitive Hashing to realtime loop closure detection," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 1228–1233.

[6] D. Nister and H. Stewenius, "Scalable Recognition with a Vocabulary Tree," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 2161–2168.

[7] B. Steder, R. Rusu, K. Konolige, and W. Burgard, "Point feature extraction on 3D range scans taking into account object boundaries," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011, pp. 2601–2608.

[8] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[9] C. Harris and M. J. Stephens, "A combined corner and edge detector," in *Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[10] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *In European Conference on Computer Vision*, 2006, pp. 430–443.

[11] L. Bo, X. Ren, and D. Fox, "Kernel Descriptors for Visual Recognition," in *Advances in Neural Information Processing Systems 23*, J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds., 2010, pp. 244–252.

[12] ——, "Depth kernel descriptors for object recognition," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 2011, pp. 821–826.

[13] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a Local Binary Descriptor Very Fast," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1281–1298, 2012.

[14] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[15] D. Galvez-Lopez and J. D. Tardos, "Bags of Binary Words for Fast Place Recognition in Image Sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, October 2012.

[16] ——, "Real-time loop detection with bags of binary words," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, sept. 2011, pp. 51 –58.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.