A Rotation Invariant Feature Descriptor O-DAISY and its FPGA Implementation

Jan Fischer, Alexander Ruppel, Florian Weißhardt, Alexander Verl

Abstract— **State-of-the-art local feature descriptors like SIFT or SURF require a significant amount of computational power which prevents their usage in applications with real time constraints. Despite recent efforts to simplify the calculation of feature descriptors, a faster computation comes often to the disadvantage of weakening the invariance to rotation or scale. Recently, Tola et al. introduced DAISY, a new local feature descriptor for wide-baseline matching across stereo image pairs. It is shown that DAISY outperforms SIFT in terms of matching accuracy while being computed significantly faster. This paper takes on the idea of DAISY by proposing a rotational invariant extension of the descriptor, called O-DAISY, and outlining its implementation on FPGA to achieve real time performance. The results are benchmarked against its original version and against the widely used descriptors BRIEF and SURF on a standardized image set.**

I. INTRODUCTION

Object detection is a vital capability of modern service robots like Care-O-bot® 3 [1] to assist humans in flexible and challenging household environments. Over the last years, local feature point descriptors proved to be efficient for reliable object description even under visual transformations like changes in scale, rotation or illumination.

Currently, the well known SIFT algorithm [2] is the most used local 2D feature detector and descriptor algorithm. However, due to its computational complexity it is unsuitable for real time object detection. To overcome the computational burden of SIFT, Bay et al. introduced SURF [3] which performs comparable to SIFT in terms of matching accuracy while greatly reducing computation time. Inspired by the developments of SIFT and SURF, Tola et al. developed DAISY [4], a novel feature point descriptor for wide-baseline matching across stereo image pairs which outperforms SIFT and SURF in terms of matching accuracy and computation costs.

However, even with the improvements in computation time of DAISY and SURF, up to this date most robust object detection systems are still not capable to perform in real time. Recent developments in feature point descriptors extensively elaborate possibilities to simplify descriptor calculation, reducing it to pure intensity comparisons and representing the descriptor as a binary string enabling fast matching [5], [6], [7]. One prominent example is BRIEF from Calonder et al. [8], which randomly compares intensity values of image point pairs within the local neighbourhood of a feature point. However, the significant reduction of the computational complexity of the proposed feature descriptors is often traded for poor invariance against visual transformations like scale or rotation.

For real time object detection it is desired to have an algorithm that is as strong as SIFT, but much faster to compute. Therefore, this paper takes on the idea of Tola et al. and proposes an extension to the DAISY descriptor enabling it to explicitly handle variations in rotation. We term this rotation invariant descriptor O-DAISY and propose an FPGA implementation of the feature descriptor to achieve real time performance. Results show that O-DAISY generally outperforms SURF on the presented benchmarks and the fast keypoint descriptor BRIEF in terms of rotational invariance. As a platform for the FPGA implementation serves an Avnet Virtex-6 FPGA DSP Kit, with a Virtex 6 LX240 that has over 240.000 logic cells, 768 DSP and 832 BRAM16 slices.

This paper is organized as follows. Section II discusses existing FPGA implementations of feature detectors and evaluates feature descriptors based on an estimation of the necessary resources and control signal complexity for FPGA implementation and the possibilities for parallel and pipelined computation. Then, section III presents the novel rotation-invariant O-DAISY descriptor and benchmarks it against the original DAISY, BRIEF and SURF. Following, section IV explains the proposed FPGA design of O-DAISY. Finally, section V summarizes the paper and gives an outlook for future work.

II. RELATED WORK

To our knowledge, research mainly focused on the FPGA implementation of feature detectors rather than the implementation of feature descriptors. Prominent examples are given by the implementation of the Harris corner detector on an FPGA board by Tippetts et al. [9], an FPGA realization of the difference of Gaussian (DOG) interest point detectors used for SIFT by Yao et al. [10], Bonato et al. [11] and Chati et al. [12] and an implementation of the Fast-Hessian corner detector by Svab et al. [13].

Considering feature descriptors, SIFT proves to be unsuitable for FPGA implementation because of its rotation schema which requires a high number of control signals for the rotation of the input patch. Additionally, the weighting schema for filling the histograms, which is applied after the patch rotation, would require over 1000 multiplications and additions within one clock. Even large FPGAs like the Virtex 6 do not provide enough resources for this.

This work was conducted in the department of Robot Systems at the Fraunhofer Institute for Manufacturing Engineering and Automation (IPA), 70569 Stuttgart, Germany, jan.fischer@ipa.fraunhofer.de, ARuppel1@gmx.de, florian.weisshardt@ipa.fraunhofer.de, alexander.verl@ipa.fraunhofer.de

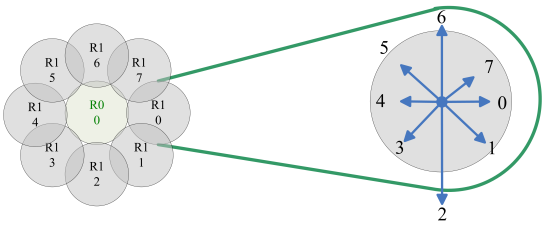
Mikolajczyk and Schmid [14] were the first to introduce a log polar grid instead of SIFT's regular grid for gradient-histogram sampling. The resulting descriptor GLOH (Gradient location-orientation histogram) outperforms SIFT in terms of distinctiveness and robustness. However, it still uses SIFT-like bilinear weighting for the creation of the histograms, which results in the same downside for FPGA implementation as SIFT.

Winder and Brown [15] split the descriptor creation into four pipeline stages and experimented with various instantiations of them. To estimate appropriate algorithm parameters, they apply a machine learning approach. The feature descriptor is created by pooling local pixel information like image gradients into histograms and concatenating all selected histograms to a single large vector, which represents the first version of the descriptor. All schemes, for which the machine learning algorithm is allowed to approximate log-polar arrangement of the local histogram areas creation perform equally well and outperform the standard SIFT pooling scheme.

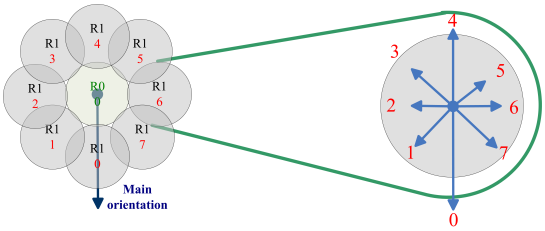
Tola et al. [4] use the log-polar grid with Gaussian weights from [15] and speed up computation by applying Gaussian convolutions for the histogram binning. They named their descriptor *DAISY*, due to the flower like arrangement of the local image regions for histogram creation. Daisy is much faster to compute than SIFT, which allows its descriptors to be calculated densely for every image pixel, enabling wide-baseline stereo matching. They compare DAISY with SIFT, SURF, NCC and pixel difference by creating dense depth maps from stereo images. Ground truth was obtained by using a laser-scanner depth-map. DAISY and SIFT perform equally well and outperform the others. The speed up of DAISY in relation to SIFT is due to the replacement of the weighted sums from SIFT by fast-to-compute Gaussian convolutions, which prove to be advantageous for FPGA implementation as well. Calculating the DAISY descriptor for different orientations reduces to shifting values of histograms, which is much better suited for FPGA design than the patch rotation of SIFT. The process of histogram rotation is visualized in figure 1 and explained in depth in section III-A.2.

Winder et al. [16] explored the possibility to quantize each entry of the DAISY descriptor and found that in general 4 bits per entry are sufficient when using PCA and 2 bits without prior PCA dimension reduction. Advice is given for the best DAISY setup for the applications in object recognition, real time mobile devices and for large data bases. Their paper shows that even the lowest dimensional DAISY descriptors with very simple computation has matching capabilities comparable to SIFT, while higher dimensional descriptors even outperform SIFT.

In summary, DAISY natively supports its implementation on FPGA, because it treats every pixel in the same way. Also, the usage of Gaussian masks for weighting reduces the necessary number of multiplications and additions because Gaussian convolutions are separable and symmetric, thus reducing the required amount of resources. Additionally,



(a) Unrotated DAISY descriptor. Left: Center histogram and 8 histograms on ring 1, with the standard numbering. Right: The eight entries of the single histogram R1-0.



(b) Rotated DAISY descriptor. Left: New numbering of the histograms. Right: New numbering of the entries of one histogram (previously R1-0, now R1-6).

Fig. 1. Descriptor alignment for rotation invariance.

the DAISY setup allows to make the descriptor rotation invariant by rotating the descriptor itself, instead of the full input patch, which requires less control signals and therefore simplifies the design. It should be noted, though, that the descriptor can only be rotated to discrete orientations. However, it has been shown in [15] that DAISY performs at least as good and often better than SIFT on pre-rotated patches. The effect of the limitation to discrete rotation angles is evaluated in section III-C.

III. O-DAISY ALGORITHM

This section briefly describes the original DAISY algorithm. It is not intended to give full insight and the interested reader is referred to [4] or [17] for more details. We follow the same nomenclature as Tola et al. in [4] and [17]. Beginning with a short introduction of DAISY, this chapter explains how the rotation invariant O-DAISY descriptor is derived. Finally, O-DAISY is benchmarked against state-of-the-art feature descriptors.

DAISY samples local gradient information in the way visualized by figure 2. Each circle represents one histogram region, which is part of the descriptor vector. Each histogram represents the gradient orientations within this region. The gradient is split into H discrete orientations, so each single histogram has H entries. The algorithm uses Q rings in log-polar arrangement around the center on which histograms are sampled. T represents the number of histograms (circles in figure 2) on each ring. Therefore, the resulting descriptor has $D_s = (Q * T + 1) * H$ entries.

Our DAISY implementation uses $H = 8$ discrete gradient orientations, $Q = 2$ rings and $T = 8$ histograms on each ring, which means that the resulting feature vector has in total $D_s = 136$ entries.

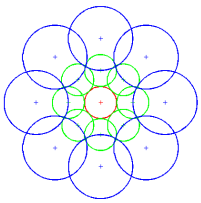


Fig. 2. DAISY descriptor with 2 rings and 8 angular elements: Each circle represents one histogram region footprint. The radius of each circle is one time the standard deviation σ. The center of each circle represents the sample position of this specific histogram. H=8, Q=2, T=8.

A. Rotational Invariance

A local image descriptor is called rotation invariant, if it is invariant to rotations around the camera's optical axis. Such a behaviour is desirable in the case of object detection, because the relative pose of objects to the camera is unknown. Many image descriptors, like SIFT, rotate the image patch around an interest point, according to the patches main orientation. This is usually done before the descriptor itself is calculated. Therefore, the descriptor algorithm is not affected by the main orientation, only its input changes. However, this is a costly process since the patch rotation requires bilinear interpolation for many pixels. The DAISY feature descriptor does not provide rotational invariance. However, the properties of DAISY allow to achieve rotational invariance relatively easy. We approximate the image patch pre-rotation at discrete angles without much overhead, by rotating the descriptor instead of the input patch. For this purpose, the interest points' main orientation is calculated. In *1)* we propose a novel, fast assignment of this main orientation. Inspired by [16], the main orientation is used to align the descriptor, as described in *2)*. The rotation invariance of this approach is evaluated and further improvements are presented in *3)*.

1) Main Orientation Assignment: The original SIFT algorithm [2] calculates gradient orientation θ and magnitude m at each pixel of the feature point area to form an orientation histogram with 36 bins. Each bin corresponds to 10° of the full 360°. The gradient orientation of each pixel within a given radius around the interest point is weighted by its magnitude and a circular Gaussian window, with a sigma of 1.5 times the detected scale. Peaks in the histogram correspond to dominant orientations. SIFT allows to assign multiple dominant orientations, for histogram peaks within 80% of the highest peak.

The original DAISY algorithm already calculates values very similar to SIFT's main orientation histogram. DAISY usually discretizes the gradient orientations into 8 directions, which means that every histogram bin corresponds to 45° of the full 360°. The gradient magnitude is calculated for each discrete direction i, resulting in one gradient image per direction, the so called orientation maps G_{o_i}. Each orientation map is now sequentially smoothed with Gaussian masks Σ_k, resulting in eight smoothed orientation maps $G_{o_i}^{\Sigma_k}$, one for each orientation i. The magnitude of the smoothed gradients are the entries that will be written into the final descriptor.

Now, we will show that the entries of the smoothed orientation maps $G_{o_i}^{\Sigma_k}(x,y)$ at a interest point location (x,y) are similar to the entries of SIFT's orientation histogram for the same interest point: Each orientation map $G_{o_i}(x,y)$ contains the magnitude of the gradients in the discrete direction i, so it is a discrete version of SIFT's orientation θ and magnitude m calculation. The convolution of the orientation maps with a Gaussian mask is similar to creating the SIFT orientation histogram with entries weighted by a Gaussian mask and their gradient magnitude. In summary, $G_{o_i}^{\Sigma_k}(x,y)$ is similar to SIFT's orientation histogram for a point (x,y), with the main differences being the larger angle discretization.

To assign the orientation based on the gradient information of the largest possible area around the interest point, we use the convolved orientation maps with the largest sigma, $G_{o_i}^{\Sigma_2}(x,y)$, at a interest point location (x,y). Our O-DAISY implementation searches for the maximum values in the convolved orientation map $max_i(G_{o_i}^{\Sigma_2}(x,y))$, just as SIFT searches for peaks in the orientation histogram. The corresponding direction i of these maxima define the main orientations. We allow up to three main orientations if the 2nd and 3rd maxima are within 80% of the highest peak. Imagine $G_{o_i}^{\Sigma_2}(x,y)$ to be a histogram equivalent to the one shown on the right side of figure 1. In that figure, obviously orientation number 2 constitutes the maximum.

Note that even if we explained the orientation assignment for a specific keypoint position, the main orientation can be computed without much overhead for every pixel, because the convoluted orientation maps already exist for every pixel location.

2) Descriptor Rotation: After calculating the main orientation, the descriptor itself is made rotation invariant by aligning it with the main orientation. First, every single histogram, i.e. circle in figure 2, is reordered internally, so that histogram bin 0 now represents the magnitude of the main orientation within its local area. After each single histogram is internally aligned with the main orientation, the whole descriptor needs to be rotated. The center histogram stays on the same position within the descriptor. The numbering of the histograms on ring one are changed in the same way as the internal histograms. The same applies for all following rings.

To illustrate the process, we assume the main orientation is 2 (90°). Then, within each single histogram the old bin 2 becomes the new rotated bin 0. The old bin 3 becomes bin 1 and so forth, until the old bin 1 becomes the new bin 7. In the same way, the histogram numbering on the rings is changed. The process is visualized by figure 1. If there is more than one main orientation the descriptor is sequentially aligned to all of them and up to three descriptors are created. We found, similar to Lowe in [2], that the multiple oriented descriptor increased the matching accuracy significantly.

B. Test Scenario

We benchmark the proposed O-DAISY descriptor in the same way as [8], where Calonder et al. compare their BRIEF descriptor with SURF. The test data originates from the

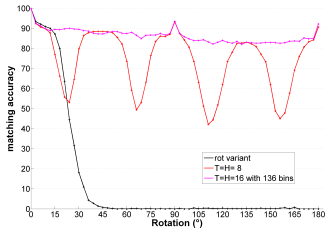


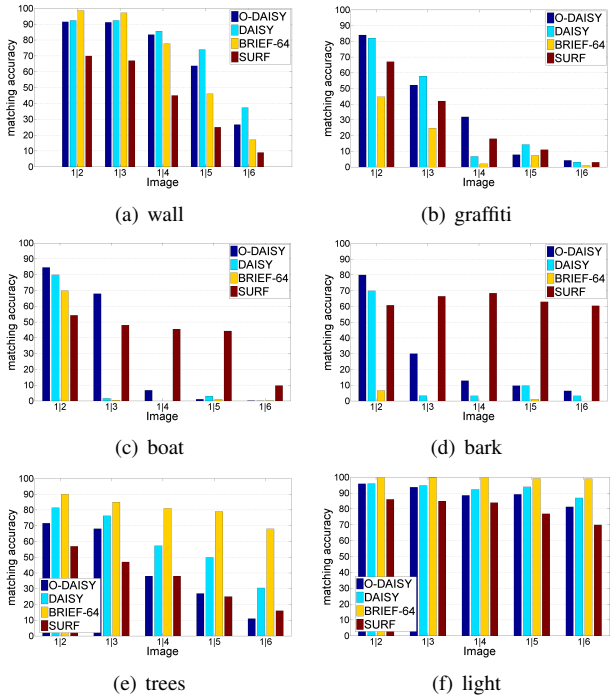
Fig. 3. **Main Orientation Assignment**: Matching accuracy as a function of the rotation angle for an artificially rotated wall image using the rotation variant original DAISY (black), DAISY with rotational invariance using T=H=8 (red), and rotation invariant DAISY with T=H=16, with descriptor size reduced to 136bins (magenta)

Oxford's Visual Geometry Group[1] and consists of original and transformed image pairs from different scenes.
A SIFT like Difference of Gaussian (DOG) interest point detector generates a set of keypoints on the first image of the investigated test set. The keypoints are projected into each transformed image in the set using the provided homographies. Descriptors for all keypoints are calculated. The keypoint descriptors of the original image constitute the *original descriptor database*, while the descriptors of the transformed image create the *test descriptor database*. For the rotation invariant case, up to three descriptors are calculated per keypoint for the original descriptor database, if there are multiple maxima in the main orientation histogram. For the test descriptor database, only one descriptor per keypoint is allowed, because the detected orientation in the test image is most likely to fit one of the three detected orientations in the original database, if the keypoints orientation is ambiguous.
For each descriptor in the test database, the closest neighbour in the original database is searched and classified as a match. We apply a brute force search using the FLANN library [18]. A match is considered correct, if its two points are correlated by the homography. The matching accuracy is calculated as the amount of correct matches over the number of all transformed keypoints that lie within the image.

C. Results

Figure 3 shows the evaluation results of the matching accuracy in proportion to the rotation angle between original and artificially rotated versions of itself. By comparing the black curve representing the original DAISY descriptor and the red curve showing the proposed O-DAISY descriptor with the modifications described above, it is obvious that the described orientation assignment and descriptor rotation greatly improve the invariance of DAISY against rotation. However, the red curve also shows that using eight discrete gradient directions with an enclosing angle of 45° each, results in a significant matching accuracy drop down, when the test image's rotation is right in the middle between two discrete orientations. To compensate this behaviour we doubled the number of discrete angles. This comes with two major downsides. First, increasing both T and H from 8 to 16 the descriptor's size increases from 136 to 528. Second, the single histograms would now overlap significantly and describe one and the same area multiple times. This leads to a lot of redundant information which might even decrease the matching accuracy. To avoid these two problems, we propose to first convolve the orientation maps for all 16 orientations, but to use only every second value within each histogram and only every second histogram from the rings for descriptor creation. This results in a feature descriptor with only 136 bins, still having the shape shown in figure 2, now only aligned more accurately to the main orientation. Its performance is shown by the magenta curve of figure 3 which clearly does not exhibit the dramatic performance drop down when using just 8 discrete gradient directions.

We will term the resulting descriptor O-DAISY, marking its invariance to orientation changes. O-DAISY's rotation invariance is also significantly better than the one of SURF. This can be seen by comparing figure 3 with the equivalent test of [8], where SURF's matching accuracy drops from 83% for 3° down to 63%, when the rotation angle is between 30° and 60°. In comparison, O-DAISY's accuracy stays between 90% and 85% for all angles between 0° and 180°. It should be noted that the computational complexity for calculating the convolved orientation maps doubles with increasing the number of discrete orientations from 8 to 16. However, in the FPGA design this does not increase the run time, since all orientation maps can be smoothed in parallel as enough resources are available.



(a) wall (b) graffiti (c) boat (d) bark (e) trees (f) light

Fig. 4. Evaluation of the rotation invariant O-DAISY, the original DAISY, SURF and BRIEF-64

Additionally to the rotation test, we benchmark O-DAISY using the Oxford library and comparing it to the original

[1] http://www.robots.ox.ac.uk/~vgg/research/affine/

DAISY, the rotation invariant SURF and the strongest BRIEF descriptor, BRIEF-64 for which rotation invariance is not specifically addressed. Figure 4 visualizes the results.

The rotation variant DAISY and BRIEF-64 outperform the two rotation invariant descriptors O-DAISY and SURF in scenes, where the camera's rotation around the optical axis is small. This is the case for the trees and light scenes, where the camera did not move and also for the first two wall images. It should be noted that rotation invariance usually not only induces higher computational complexity, but also results in lower matching accuracies in cases where rotational invariance is not needed. This is already pointed out in [8] and our results confirm this.

The wall and graffiti scenes are viewpoint change evaluations. In the wall scene the camera is not rotated around its optical axis and despite its rotational invariance, O-DAISY performs almost as good as DAISY and BRIEF. SURF is the weakest descriptor for this scene. The graffiti scene combines large viewpoint changes of up to 60° with rotation around the optical axis up to 45°. The performance of the original DAISY for the first two test images shows that the log-polar arrangement makes it invariant to small camera rotations. BRIEF does not have this behaviour and its matching accuracy is significantly below the others. O-DAISY outperforms SURF in this scene, but fails to provide good accuracies for the last images, where the viewpoint change is large.

The boat and bark scenes combine large camera rotations with large optical zoom factors. As expected, the two rotation variant descriptors BRIEF and DAISY fail in these scenes. For images with zoom factors smaller than 2, i.e. the two first boat images and the first bark image, O-DAISY outperforms SURF, but with larger zoom factors the explicit scale invariance of SURF makes it the better choice. Remember that scale invariance is not yet explicitly addressed in O-DAISY. This will be considered in future work.

SURF and O-DAISY perform equally well in the trees and light scenes, but are outperformed by the rotation variant descriptors, as mentioned above.

In summary, O-DAISY proves to be a useful descriptor with better rotation invariance than SURF, while exhibiting drawbacks in terms of scale invariance.

IV. FPGA IMPLEMENTATION

The DAISY algorithm is implemented on an Avnet Virtex-6 DSP Kit. Via Ethernet the image is transferred from the PC to the FPGA and resulting descriptors are returned from the FPGA to the PC. The main part of the FPGA design is done using Xilinx' System Generator toolbox for Matlab/Simulink. Figure 5 gives an overview of the FPGA implementation. First, the image stream is pre-smoothed by a Gaussian mask. After the gradients in x and y direction are calculated, the discrete gradient orientations are calculated. The blocks for smoothing the orientation maps are implemented as subsequent Gaussian filters. The blocks take the separation and symmetry possibilities into account, which are described in [11]. The 1D filters are implemented using the

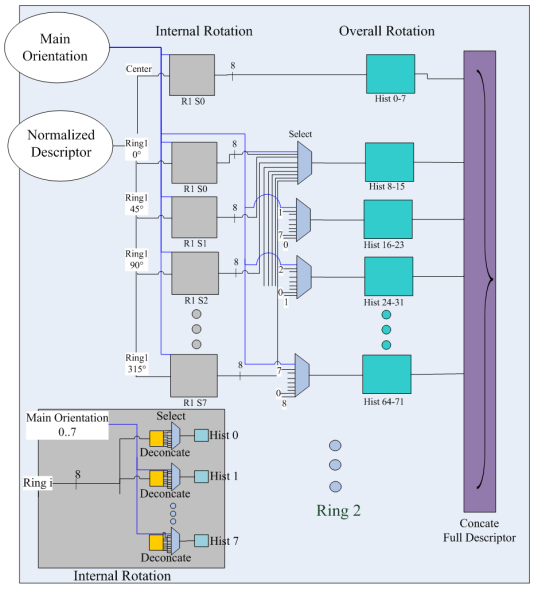


Fig. 6. FPGA Design: Descriptor rotation according to the main orientation. The grey blocks for the internal rotation are detailed in the bottom left corner

FIR Compiler from System Generator that uses DSP slices. After all three smoothed orientation maps have been calculated, the single histograms are read out from the correct positions around the current middle pixel. This is fulfilled by the light blue block, by delaying the stream of smoothed orientation maps using block RAMs as line buffers. The values on the correct positions from the appropriate orientation map are concatenated to the feature vector. This feature vector is normalized in the green block and finally, it is rotated according to the main orientation in the last grey-blue block. The orientation alignment follows the algorithm described above. First every single histogram is aligned with the main orientation, as it can be seen in the bottom left corner of figure 6. Then the histograms exchange positions within the vector which is shown in the main part of figure 6.

A. Quantization Results

Floating point operations on FPGAs are very resource intensive, therefore algorithms need to be quantized to a fixed bit accuracy. For this reason, we used the wall test scene to evaluate the necessary number of bits for the critical intermediate results of the algorithm.

The *unsmoothed orientation maps* are critical, since they build the basis for the costly Gaussian convolutions. Also, the *smoothed orientation maps* are critical, because they need to be buffered for the later histogram sampling. Finally, the *normalized descriptor* are transferred over Ethernet and stored on the PC and should therefore be as small as possible. Figure 7(a) shows that using less than three bits for the unsmoothed orientation maps G_{o_i} results in impossible matching. More than five bits do not improve the accuracy, so 5 bits are used for G_{o_i}.

Figure 7(b) presents the evaluation for the smoothed orientation maps $G_{o_i}^{\Sigma_k}(x,y)$. The matching accuracy is almost zero for up to five bits. The steady grow of accuracy from

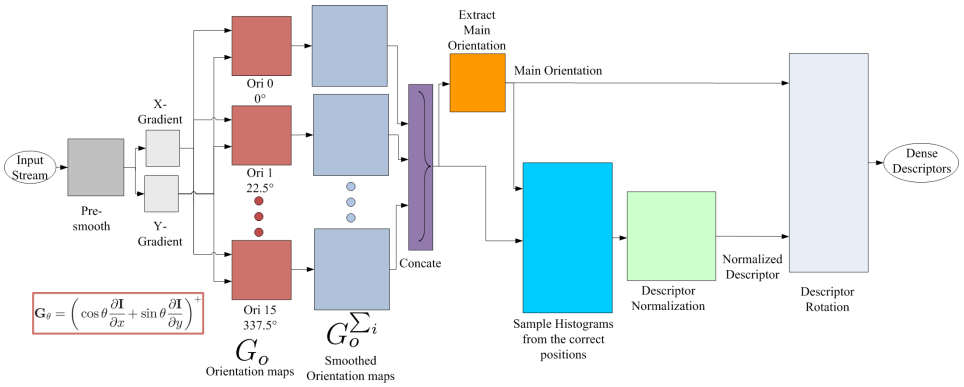
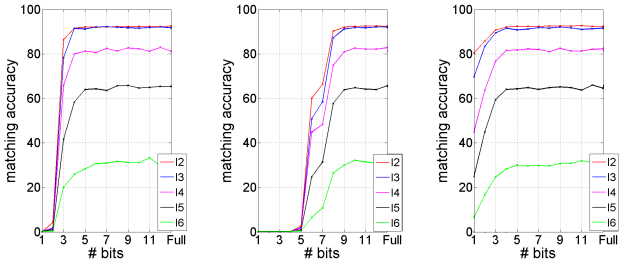


Fig. 5. DAISY Algorithm FPGA Design: Overview



(a) Unsmoothed orientation map: 5 bits (b) Smoothed orientation maps: 9 bits (c) Descriptor after normalization:4 bits

Fig. 7. **Quantization** Matching accuracy as a function over the number of bits for using the wall scenario

thereon stops when 9 bits are reached, so 9 bits are used for $G_{o_i}^{\Sigma_k}(x,y)$. By increasing the number of bits for each entry of the normalized descriptor from 1 to 4, the matching accuracy grows continuously for all test images, as can be seen in figure 7(c). Higher accuracies do not improve the results, therefore 4 bits are used for each normalized descriptor entry, so that a descriptor has 544 bits in total.

V. Conclusion and Outlook

This paper presented O-DAISY, a rotation invariant extension of the novel feature descriptor DAISY. Experiments on a standardized test set showed that the adopted descriptor is able to outperform current state-of-the-art fast-to-compute descriptors, like SURF and BRIEF. Additionally, we described an FPGA implementation of O-DAISY, which enables real time performance of the descriptor calculation. Future work will address the problem of scale invariance. The original DAISY descriptor proved to be invariant to scale changes up to a factor of 2. To further improve scale invariance one could scale the sampling schema of the local histogram areas with a scale determined by a feature detector or coming from information of a RGB-D camera device like Microsoft's Kinect.

References

[1] C. Parlitz, M. Haegele, P. Klein, J. Seifert, and K. Dautenhahn, "Care-o-bot® 3 - rationale for human-robot interaction design," in *Proceedings of 39th International Symposium on Robotics (ISR), Seoul, Korea*, 2008.

[2] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[4] E. Tola, V. Lepetit, and P. Fua, "A fast local descriptor for dense matching," in *Proc. CVPR*, Citeseer, 2008.

[5] G. Shakhnarovich, *Learning task-specific similarity*. PhD thesis, Massachusetts Institute of Technology, 2006.

[6] M. Oezuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE transactions on pattern analysis and machine intelligence*, pp. 448–461, 2009.

[7] S. Taylor, E. Rosten, and T. Drummond, "Robust feature matching in 2.3s," *Computer Vision and Pattern Recognition Workshop*, 2009.

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *European Conference on Computer Vision*, September 2010.

[9] B. Tippetts, S. Fowers, K. Lillywhite, D. Lee, and J. Archibald, "Fpga implementation of a feature detection and tracking algorithm for real-time applications," in *Proceedings of the 3rd international conference on Advances in visual computing-Volume Part I*, pp. 682–691, Springer-Verlag, 2007.

[10] L. Yao, H. Feng, Y. Zhu, Z. Jiang, D. Zhao, and W. Feng, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," in *Field-Programmable Technology, 2009. FPT 2009. International Conference on*, pp. 30–37, IEEE, 2009.

[11] V. Bonato, E. Marques, and G. Constantinides, "A parallel hardware architecture for scale and rotation invariant feature detection," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 12, pp. 1703–1712, 2008.

[12] H. D. Chati, F. Muhlbauer, T. Braun, C. Bobda, and K. Berns, "Hardware/software co-design of a key point detector on fpga," *Field-Programmable Custom Computing Machines, Annual IEEE Symposium on*, vol. 0, pp. 355–356, 2007.

[13] J. Svab, T. Krajnik, J. Faigl, and L. Preucil, "FPGA based Speeded Up Robust Features," in *Technologies for Practical Robot Applications, 2009. TePRA 2009. IEEE International Conference on*, pp. 35–41, IEEE, 2009.

[14] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE transactions on pattern analysis and machine intelligence*, pp. 1615–1630, 2005.

[15] S. Winder and M. Brown, "Learning local image descriptors," in *IEEE Conference on Computer Vision and Pattern Recognition, 2007. CVPR'07*, pp. 1–8, 2007.

[16] S. Winder, G. Hua, and M. Brown, "Picking the best daisy," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR09)*, (Miami), June 2009.

[17] E. Tola, V. Lepetit, and P. Fua, "DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 815–830, May 2010.

[18] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pp. 331–340, INSTICC Press, 2009.