

A feature descriptor for texture-less object representation using 2D and 3D cues from RGB-D data

Jan Fischer¹, Richard Bormann¹, Georg Arbeiter¹ and Alexander Verl¹

Abstract—At the core of every object recognition system lies the development and integration of distinct feature descriptors to create object representations robust against varying perspectives or lightning conditions. Recent work has primarily focused on the development of distinct point features. While these features achieve impressive recognition results, point features fail to capture the shape and appearance of an object with less or even without texture. This paper proposes a novel method for the rapid and dense computation of 2D and 3D image cues from RGB-D data to target the recognition of objects without rich texture and a global histogram-based descriptor for the distinct description of object models.

I. INTRODUCTION

Robust object recognition is a key functionality and challenging problem towards the vision of a fully autonomous acting service robot like Care-O-bot[®] 3 [1]. Especially changing conditions in lightning, scale, rotation or occlusion significantly influence the recognition performance. Most vision systems divide object recognition into a training and recognition stage. During training, object images from different viewpoints are recorded and processed to create an object representation. Stable feature points like corners provide the basis for the later recognition. They are extracted from the individual object views and saved within the object model. A distinct representation of the image data from the local vicinity of the feature points is constructed by feature descriptors.

A major advantage of point features is their locality that enables the descriptor to accurately describe an object even under partial occlusion. However, 2D point feature detectors and descriptors rely on texture in order to robustly locate and describe a feature point. Objects without textured elements are not suited for point feature based recognition. Here, often global features are used to describe the complete object by a single descriptor, diminishing the robustness against occlusion.

This paper presents a novel method for the rapid computation of a local point feature descriptor that estimates 2D and 3D image cues on a per-pixel basis. The method makes use of dynamic programming techniques in order to reduce the computation time. Even though the focus of the paper is on the point feature computation, it further presents a global descriptor that aggregates the point feature cues into histograms to enable object training and recognition. The computation of the local point feature descriptor is evaluated in terms of computation time. Its performance in terms of



(a) Car

(b) Cup

Fig. 1: Recognition results of the proposed feature descriptor for texture-less object recognition

recognition rate is evaluated using recall-precision plots and L2 distance metrics to measure the separability of positive and negative descriptor samples. A visual impression of the recognition performance is given by Figure 1.

The paper is organized as follows. At first, related work concerning local and global feature points is briefly described in Section II. Afterwards, the proposed descriptor and its different processing steps are presented (Section III) and evaluated (Section IV). The paper concludes with a summary and an outlook on future work.

II. RELATED WORK

Probably, the most prominent and widely used 2D point feature descriptor is SIFT which has been presented by Lowe et al. [2]. The authors describe a robust point feature detector and a descriptor that uses a histogram over gradient directions to describe the local neighborhood of the feature point. Bay et al. present with SURF [3] an accelerated version of SIFT by using several approximations for feature point detection and description. Fischer et al. present in [4] a rotation invariant extension of the DAISY descriptor originally proposed by Tola et al. [5]. The authors claim to outperform SIFT, however, due to its complexity an FPGA implementation is needed to achieve real-time performance. Calonder et al. introduce the binary feature descriptor BRIEF in [6] which has been extended by Rublee et al. [7] to achieve rotational invariance. It is a simple feature point descriptor that compares individual intensity values against each other. The result of the comparison (smaller or greater) is saved in a single bit. The binary representation enables a fast matching of the feature point descriptor while still achieving high recognition rates. Hinterstoisser et al. [8], [9], [10] propose a template matching approach explicitly addressing the recognition of untextured 3D objects. The authors apply

¹J. Fischer, R. Bormann, G. Arbeiter and A. Verl are with the Fraunhofer Institute for Manufacturing Engineering and Automation IPA, Stuttgart, Germany

a sliding-window based search on the whole image data by comparing gradient based image templates against (on-line) trained object templates. The presented work is strongly related to the work of Hinterstoisser et al.. It proposes an alternative method for 2D and 3D gradient estimation using dynamic programming that further decreases computation.

Recent frameworks for 3D object detection have been presented by Grundmann et al. [11] or Collet et al. [12] both systems build upon 2D point features that have been augmented with 3D information about their position relative to the object center.

III. METHOD

This section outlines the proposed point feature descriptor and its application for object training and object recognition. The proposed descriptor operates on single shot RGB-D images originating from a RGB-D camera device like the Microsoft Kinect. Initially, the computation of the local point feature descriptor captures 2D image and 3D depth cues with a binary descriptor. This per-pixel information is aggregated into a histogram over the object related image area. The histogram is reordered and normalized in order to achieve rotation and scale invariance.

A. Point feature descriptor

Inspired by the recent rise of binary point feature descriptors and their superior performance considering computation time [6] [7], the paper makes use of binary descriptor computations. Following the approach of Hinterstoisser et al., the local point feature descriptor captures quantized 2D gradient information from the color image and quantized 3D gradient information from the range image. However, the dense computation of the descriptor values for each image pixel is based on the principles of dynamic programming to break down the estimation of 2D and 3D gradients into simple comparisons that are solved only once and reused afterwards to compute the remaining descriptors. This results in a significant speed up in descriptor computation compared to the original approach.

Figure 2 shows the layout of the binary point feature descriptor. In general, a real-valued function $f(x, y)$ is evaluate on different pixel-pairs as indicated by the dashed lines and the results are thresholded to compute binary values which are concatenated to a local descriptor.

The layout of the feature descriptor has been chosen to compare pixels of opposing squared regions around the feature point p as indicated by the green and red squares in Figure 2. The arrangement of the opposing squares is rotated about the feature point p through an angle of 45° in anticlockwise direction (Figure 2a - 2d). This results in a total of 4×2 different opposing squared regions around the feature point p that contribute to the value of the binary descriptor. The computation of the descriptor value is the same for each alignment of the opposing rectangles, therefore the following sections refer to Figure 2a without loss of generality if not stated otherwise.

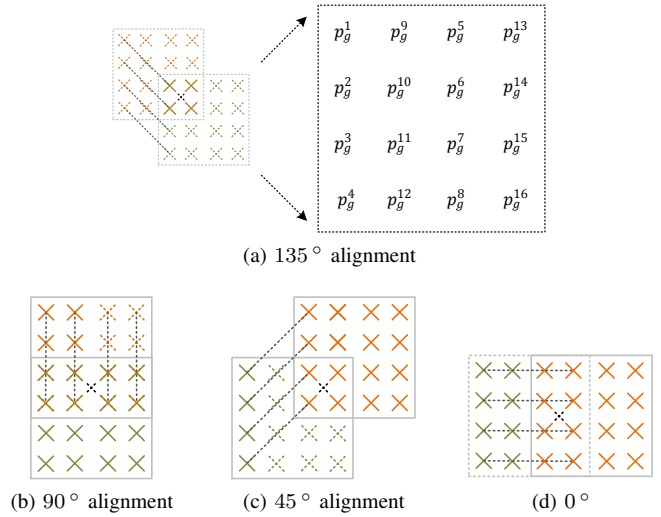


Fig. 2: Layout of the feature descriptor to compute the binary descriptor values for the feature point p located at the black cross in the center of the colored squares. The dashed lines indicate corresponding pixel positions p_r^i and p_g^i from the red and green square at which a function $f(p_g^i, p_r^i)$ is evaluated. The value of f is thresholded to compute a binary value $\tau(f(p_g^i, p_r^i)) \in \{0, 1\}$.

Within each of the squared regions 16 equally spaced pixel positions $p_s^i, i \in \{0..15\}, s \in \{r, g\}$ are selected, where s refers to the red or green square in Figure 2a. For each of the 16 pixel positions p_g^i a counterpart p_r^i is attributed as indicated by the dashed lines resulting in 16 pixel-pairs per alignment. The RGB-D values corresponding to the pixel-pairs are jointly evaluated to compute the necessary information for the later 2D and 3D gradient estimation and the results are binarized which allows the application of the Hamming operator in order to speed up further computations. The number of 16 pixel-pairs for each distinct alignment of the square regions has been chosen to increase the robustness of the proposed descriptor. It reduces the effect of noisy measurements and avoids the need for a preceding smoothing operation on the image data due to an averaging of the evaluation results among all 16 pixel-pairs which is described in section III-A.1 and III-A.2.

1) *2D gradient estimation:* The 2D gradients are computed on the RGB channels of the color image. Following the approach of Hinterstoisser et al., the gradient direction is omitted to avoid the influence of bright or dark object-backgrounds on the 2D gradient direction and the gradients are computed on each channel individually. The algorithm computes quantized gradient orientations ranging from 0° to 135° in steps of 45° . At first, the 2D gradient magnitudes for the orientation corresponding to the current arrangement of the opposing rectangles ($0^\circ, 45^\circ, 90^\circ$ or 135°) are computed in (1) by subtracting and maximizing the intensity differences of the R, G and B values at all 16 pixel-pairs.

$$f_{2D}(p_g^i, p_r^i) = \max_{j \in \{R, G, B\}} |i_j(p_g^i) - i_j(p_r^i)| \quad (1)$$

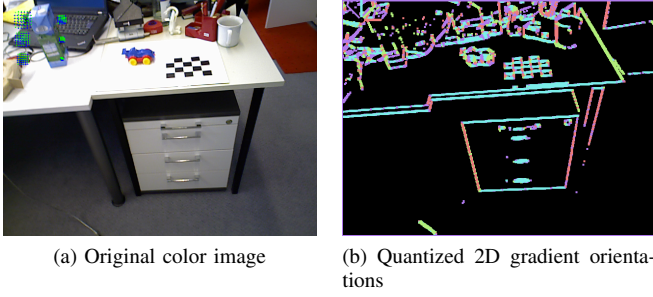


Fig. 3: Output of the proposed method for 2D gradient computation using discrete angles of 0° , 45° , 90° and 135°

The three RGB color channels are denoted by $j \in \{R, G, B\}$ and $i_j(p)$ returns the intensity value for channel j at the pixel p . Then, the result of f_{2D} is binarized as given in (2).

$$\tau_{2D}(f_{2D}(p_g^i, p_r^i)) = \begin{cases} 1, & \text{if } f_{2D}(p_g^i, p_r^i) < t_{2D} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

τ_{2D} compares the value of f_{2D} against a predefined fixed threshold $t_{2D} \in \{0 \dots 255\}$ expressed in pixels. An optimal value of $t_{2D} = 70$ has experimentally been determined according to Section IV. Finally, the binary values computed from all 16 pixel-pairs in Figure 2 are concatenated to a 16-bit descriptor d_{2D}^k .

In order to assign a single 2D gradient orientation to the feature point, the discretized 16-bit descriptors $d_{2D}^k, k \in \{0..3\}$ of all four alignments of the opposing square regions from Figure 2 are evaluated in (3) - (5).

$$x = \sum_{k=0}^3 \left(\text{Ham}(d_{2D}^k, 0) \cos\left(\frac{45\pi}{180}k\right) \right) \quad (3)$$

$$y = \sum_{k=0}^3 \left(\text{Ham}(d_{2D}^k, 0) \sin\left(\frac{45\pi}{180}k\right) \right) \quad (4)$$

$$\text{ori}_{2D}(p) = \text{atan2}(y, x) \quad (5)$$

$\text{Ham}(h_1, h_2)$ computes the Hamming distance between the binary values of h_1 and h_2 . The values of the \sin , \cos and atan2 functions have been implemented using a look-up table in order to speed up computations. Finally, the discretized gradient orientation closest to the value of $\text{ori}_{2D}(p)$ is selected and saved as the 2D gradient orientation at feature point p . A visual impression of the computed and quantized 2D gradient orientation is given by Figure 3.

2) *3D gradient estimation*: The idea of the 3D gradient estimation is to estimate the gradient of the tangent plane going through the feature point p . Therefore, the function f_{3D} for 3D normal estimation measures the gradient on the range image at the current pixel-pair as given in (6).

$$f_{3D}(p_g^i, p_r^i) = z(p_g^i) - z(p_r^i) \quad (6)$$

$z(p)$ denotes the measured range value at the pixel coordinate p . Compared to (1) the gradient direction is not omitted as it specifies the orientation of the tangent plane. f_{3D} estimates the 3D gradient orientation at the current feature point for the eight discretization angles ranging from 0° to 315° in steps of 45° . The binarization of f_{3D} is given in (7).

$$\tau_{3D}^+(f_{3D}) = \begin{cases} 1, & \text{if } f_{3D}(z_g^i, z_r^i) > t_{3D} \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

$$\tau_{3D}^-(f_{3D}) = \begin{cases} 1, & \text{if } f_{3D} < -t_{3D} \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

The value of f_{3D} is compared against two thresholds t_{3D} and $-t_{3D}$ expressed in meters in order to keep the sign of the gradient directions. Compared to t_{2D} in (2), t_{3D} is not fixed but modified proportional to the measured distance at the current feature point location in order to compensate for the lower spatial resolution of the input data at increasing distances. A value of $t_{3D} = 0.002$ at 0.5 m has experimentally been determined to yield best results. Finally, the binary values from (7) and (8) computed from all 16 pixel-pairs are concatenated to separate 16-bit descriptor d_{3D}^k and d_{3D}^{k+4} . Here, the index $k+4$ indicates that the results of the descriptor d_{3D}^{k+4} contribute to the gradient direction that points in the opposite direction than the entries of d_{3D}^k .

In order to assign a single 3D gradient orientation to the feature point, the discretized 16-bit descriptors $d_{3D}^k, k \in \{0..7\}$ of all four alignments of the squared regions from Figure 2 are evaluated in (9) - (11).

$$x = \sum_{k=0}^7 \left(\text{Ham}(d_{3D}^k, 0) \cos\left(\frac{45\pi}{180}k\right) \right) \quad (9)$$

$$y = \sum_{k=0}^7 \left(\text{Ham}(d_{3D}^k, 0) \sin\left(\frac{45\pi}{180}k\right) \right) \quad (10)$$

$$\text{ori}_{3D}(p) = \text{atan2}(y, x) \quad (11)$$

A visual impression of the computed and quantized 3D gradient orientations is given by Figure 4.

B. Layout of the point feature descriptor

Due to the dense calculation of the descriptor for each image pixel it is of utmost importance to enable a fast feature descriptor computation. Therefore, the descriptor layout of the pixel comparisons has been selected to enable a speed up using dynamic programming. Figure 5 illustrates how the results from the pixel-pair evaluations may be reused for the computation of the current descriptor. Initially, the descriptor computation begins with the top left pixel and proceeds by moving from left to right and from top to bottom.

Only the first descriptor values must be calculated from scratch. All successive descriptors may use the results from their predecessors reducing the number of pixel-pair evaluations for the descriptors of the first row from $4 * 16$ to

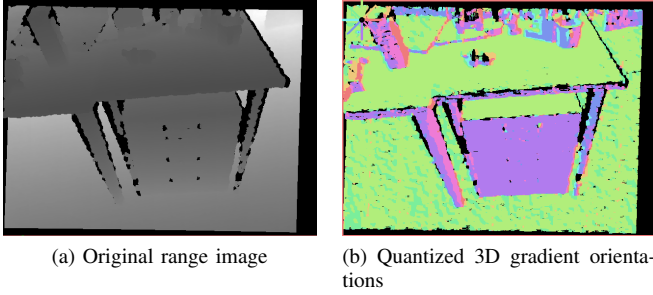


Fig. 4: Output of the proposed method for 3D gradient computation using discrete angles ranging from 0° to 315° in steps of 45°

4×4 (Figure 5a). Once the descriptors of the first row have been computed an additional reduction in the number of evaluations from 4×4 to 4×1 (Figure 5b) is achieved. An additional speed up of the descriptor computation results from the binarization as explained in (2), (7) and (8). It limits the amount of data that needs to be copied and enables the usage fast descriptor assignments by simple bit-shifts of the appropriate descriptor entries to compute the binary descriptor for the next feature point.

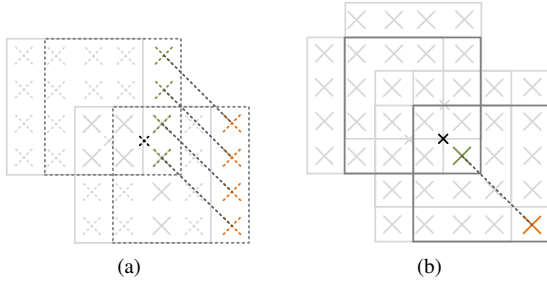


Fig. 5: Computation of the binary descriptor values for the pixel location at the black cross in the center of the colored squares. When computing the descriptor values of the first image row, 4×4 new pixel pairs must be evaluated for each new descriptor indicated by the dashed lines in (a). For all successive rows only 4×1 pixel pairs need to be evaluated all others are reused (b)

C. Object training

The setup for object training is visualized in Figure 6. In order to teach a new object to the system, the object is placed on a predefined position at a checker board. Then the camera is moved around the object to record images from different viewpoints. The training procedure detects the pose of the checker board and infers the position of the object relative to it. By the use of a virtual 3D box with a predefined size depending on the object's dimensions, the image data corresponding to the object is determined and the background is removed. Finally, local point feature descriptors are computed according to section III-A for all remaining pixels.

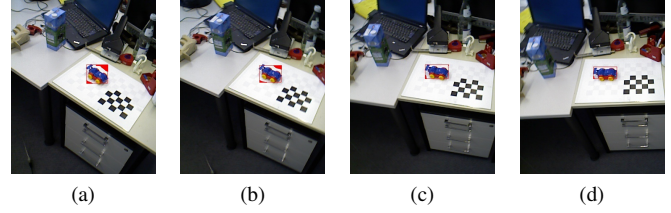


Fig. 6: Recorded training sequence for the car object. The ground truth pose of the object is given by the checker board. A virtual box in 3D space is placed around the car to separate the object data (native color values within the red rectangle) from the background

Even it would have been possible to follow the approach of Hinterstoisser et al. by applying a brute force template matching approach for object description and recognition on the 2D and 3D image cues, the proposed method makes use of a more classical approach by applying a global descriptor for object description that exhibits scale and rotation invariance. This approach has the advantage that the number of templates necessary for a robust object description is significantly reduced. The proposed procedure for object training and recognition is slower compared to the approach of Hinterstoisser, however, in the present implementation no optimizations concerning SSE accelerations or cache optimizations have been conducted.

The basic idea of the global descriptor is to capture the local 2D and 3D gradient distribution of the object. Therefore, the training data is divided into smaller spatial regions according to Figure 7 and the global descriptor is computed by counting the frequency of 3D and 2D gradient directions over these regions. The final object descriptor is constructed by concatenating the histogram values from all local descriptor regions. In order speed up the computation of the histograms, we make use of integral images [13]. Therefore, for each of the resulting gradient directions from (5) and (11) a separate gradient map is created where each pixel of the gradient map corresponds to a pixel in the original image and a value other than 0 indicates the presence of the gradient direction at that pixel. This results in a total of 12 (four 2D gradient directions and eight 3D gradient directions) gradient maps. By the use of one integral image for each gradient map, it is feasible to compute the frequency of a single gradient direction for a rectangular area with only 4 operations.

1) *Rotation invariance*: Invariance against rotation is achieved by computing the most dominant 3D gradient orientation and reordering the values of the histograms accordingly. Assuming that the gradient with the strongest magnitude has an angle of 45° then the first histogram entry will be the gradient frequency of direction 45° , followed by direction 90° , 135° , and so on until the gradient frequency 0° is appended to the end. The dominant 3D gradient orientation is computed based on the image area indicated by the red rectangle in Figure 7. The rectangle is centered on

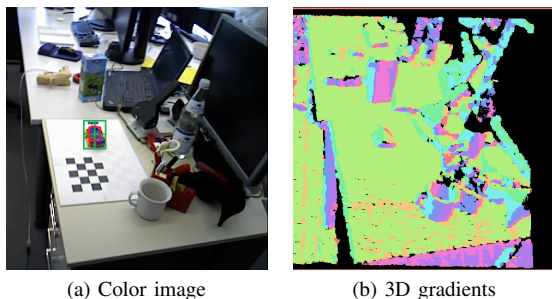


Fig. 7: One local histogram of gradient orientations is computed for each of the four areas surrounded by the green rectangles and one local histogram for the area surrounded by the red rectangle. The green rectangles are of the same size as the red rectangle. The main 3D gradient orientation from the histogram describing the red area is used for descriptor alignment in order to achieve rotation invariance

the object and of smaller size than its outline in order to keep the influence of the background for orientation estimation to a minimum.

2) *Scale invariance*: In order to achieve scale invariance the histogram entries are normalized. Additionally, the dimensions of the training rectangles (Figure 7) are stored together with the measured range value at the center of the object. Compared to a pure histogram normalization the proposed approach preserves the object dimensions as the search window for a later recognition is resized in order to fit to the stored dimensions of the training rectangle e.g. assume that the dimension of the training rectangles is 10 x 20 pixels at a distance of 1m, than the search window for recognition will be resized to 20 x 40 pixels at a distance of 2m or to 5 x 10 pixels at a distance of 0.5m.

D. Object recognition

Object recognition is based on a sliding window approach that adapts the window size according to the measured range value from the RGB-D camera as described in Section III-C.2. At each pixel a descriptor is computed according to Section III-C and compared against all descriptors created during the training procedure. Based on comparing the L2 distance to the closest descriptor with a fixed object-based threshold, the pixel is either classified as a positive or negative sample.

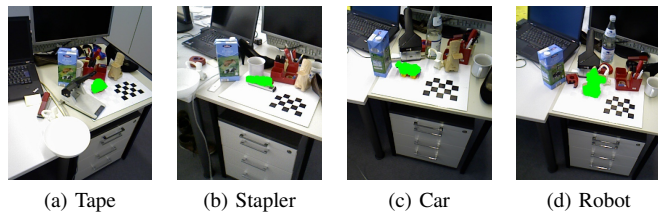


Fig. 8: Recognition results for different objects. The ground truth pose of the object is given by the checker board in order to evaluate the recognition rate

IV. RESULTS

The proposed descriptor has been tested on a dataset of four different objects. Each dataset consists of 1000 testing images recorded from strongly varying viewpoints and 200 training images. The ground truth position of the object is given by a checker board that is placed in a fixed position relative to the object. Positive samples denote the descriptor of pixels that are not more than 4 pixels away from the object center. Negative samples denote the descriptor of pixels that have a distance to the object center that is larger than the object's dimensions. Figure 6 and 8 show an excerpt from the training and testing dataset. The evaluation has been conducted with regard to the true positive rate ρ_{tp} (recall) and the precision ρ_{pr} as given in (12).

$$\rho_{\text{tp}} = \frac{h_{\text{tp}}}{h_{\text{tp}} + h_{\text{fn}}} \quad \rho_{\text{pr}} = \frac{h_{\text{tp}}}{h_{\text{tp}} + h_{\text{fp}}} \quad (12)$$

Here we denote the number of true positives h_{tp} , the number of false positives h_{fp} , the number of false negatives h_{fn} and the number of true negatives h_{tn} . Furthermore, the distribution of L2 descriptor distances from positive and negative samples compared to the closest training descriptors is evaluated for different training objects.

At first an optimal parameter of t_{2D} from (2) has been determined by computing and evaluating the recall-precision curves for different values of t_{2D} . Figure 9 gives the results for values ranging from 6 to 70 pixels. It clearly shows,

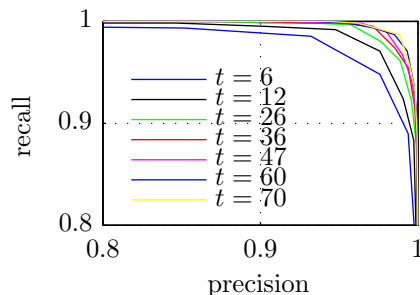


Fig. 9: Close-up view on the recall-precision plot for different values of t_{2D} from (2)

that the performance gradually increases with increasing values of t_{2D} until an optimal value of 70 is reached. The performance does not further improve for larger values of t_{2D} , therefore $t_{2D} = 70$ has been used for all following evaluations. The same evaluation has been conducted for t_{3D} from (7) and (8) based on which a value of $t_{3D} = 0.002$ at 0.5m has been selected.

The distribution of the L2 distances when comparing the descriptor of positive and negative samples with their closest matching descriptor from the training data is shown in the left column of Figure 10, whereas the right column shows the resulting recall-precision plot. The distribution of the L2 distances are approximately bell-shaped and clearly show that a separation of positive and negative samples based on thresholding on the L2 distance is feasible. Referring to the recall-precision plot the car, robot and tape object

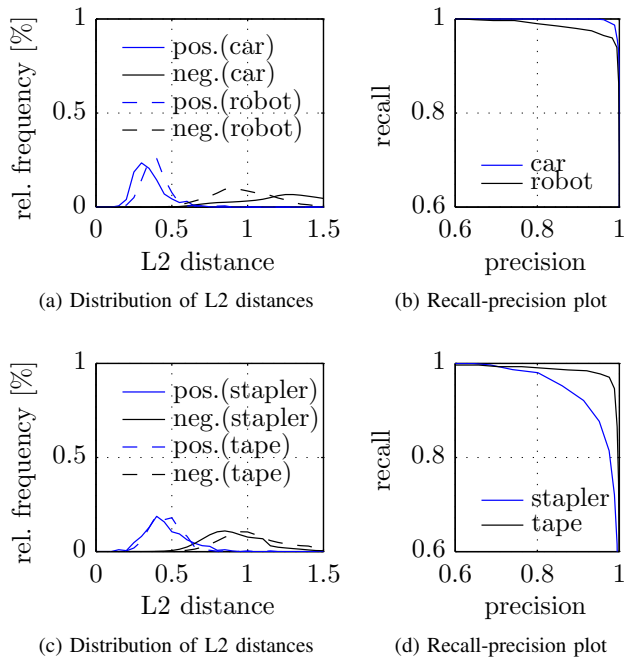


Fig. 10: Distribution of L2 distances (first column) and recall-precision plots (second column) for different objects

reach a precision more than 99% while having a recall rate above 95%. This means that we can recognize 95% of all objects when accepting a false positive rate of 1%. Due to its more common shape, the performance of the stapler object is slightly worse compared to the other objects. However it still achieves a recall rate of 74% for a precision of 99%.

Our target platform is a Intel® Core™ i7-2860QM with 2,5 GHz and 8 GB RAM. The computation time for 2D and 3D gradient estimation is compared to the original approach of Hinterstoisser et al. as it was given in the OpenCV [14] git repository from 09/2012. Table I shows that the proposed approach performs faster while still exhibiting a similar recognition performance as shown above even without the usage of any SSE or cache optimizations.

	Original approach	Proposed approach
2D gradient estimation	0.029	0.027
3D gradient estimation	0.039	0.036

TABLE I: Average computation time on a 640 x 480 image in seconds for 2D and 3D gradient estimation of the original approach by Hinterstoisser et al. and the proposed approach using dynamic programming

V. CONCLUSION

This paper presents a method that speeds up the computation of 2D and 3D gradient information using dynamic programming techniques which enables the recognition of objects even in the absence of rich texture. The proposed

method performs an initial dense descriptor computation to capture 2D and 3D gradient information and aggregates the point-based information into local histograms to generate a rotation and scale invariant object descriptor. Evaluation results show that the proposed procedure enables the recognition of untextured objects even under larger viewpoint changes.

Future work will address the integration of the proposed work into a point-feature based object recognition approach in order to achieve a reasonable recognition rate even under partial occlusion.

ACKNOWLEDGEMENTS

This research was partly funded by the ARTEMIS Joint Undertaking under grant agreement no. 100233 and the ECHORD (EU-FP7-ICT) project under grant no. 231143 within the experiment HERMES.

REFERENCES

- [1] U. Reiser, C. Connette, J. Fischer, J. Kubacki, A. Bubeck, F. Weisshardt, T. Jacobs, C. Parlitz, M. Hägele, and A. Verl, "Care-o-bot 3 - creating a product vision for service robot applications by integrating design and technology," in *International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, USA: IEEE, Oct. 2009, pp. 1992–1998. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5354526>
- [2] D. G. Lowe, *Distinctive Image Features from Scale-Invariant Keypoints*, 2003.
- [3] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, 2006, pp. 404–417.
- [4] J. Fischer, A. Ruppel, F. Weißhardt, and A. Verl, "A rotation invariant feature descriptor o-DAISY and its FPGA implementation," in *International Conference on Intelligent Robots and Systems (IROS)*, San Francisco/Calif, 2011, p. 2365–2370.
- [5] 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, no. 5, pp. 815–830, May 2010. [Online]. Available: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4815264>
- [6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: binary robust independent elementary features," in *ECCV (4)*, 2010, pp. 778–792.
- [7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *International Conference on Computer Vision (ICCV)*, Barcelona, Nov. 2011.
- [8] S. Hinterstoisser, C. Cagniard, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, pp. 876–888, 2012.
- [9] S. Hinterstoisser, S. Holzer, C. Cagniard, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," *IEEE International Conference on Computer Vision (ICCV)*, 2011.
- [10] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for Real-Time detection of Texture-Less objects," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010.
- [11] T. Grundmann, R. Eidenberger, M. Schneider, M. Fiebert, and G. v. Wichert, "Robust high precision 6D pose determination in complex environments for robotic manipulation," in *IEEE International Conference on Robotics and Automation*. IEEE, 2010.
- [12] A. C. Romea, M. M. Torres, and S. Srinivasa, "The MOPED framework: Object recognition and pose estimation for manipulation," *International Journal of Robotics Research*, vol. 30, no. 1, pp. 1284 – 1306, Sept. 2011.
- [13] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–511 – I–518 vol.1.
- [14] G. Bradski, "The OpenCV library," *Dr. Dobbs Journal of Software Tools*, 2000.