Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects

Stefan Hinterstoisser[1], Vincent Lepetit[2], Slobodan Ilic[1], Pascal Fua[2], Nassir Navab[1]

[1]Department of Computer Science, CAMP, Technische Universität München (TUM), Germany
[2]École Polytechnique Fédérale de Lausanne (EPFL), Computer Vision Laboratory, Switzerland
{hinterst,slobodan.ilic,navab}@in.tum.de, {vincent.lepetit,pascal.fua}@epfl.ch

Abstract

We present a method for real-time 3D object detection that does not require a time consuming training stage, and can handle untextured objects. At its core, is a novel template representation that is designed to be robust to small image transformations. This robustness based on dominant gradient orientations lets us test only a small subset of all possible pixel locations when parsing the image, and to represent a 3D object with a limited set of templates. We show that together with a binary representation that makes evaluation very fast and a branch-and-bound approach to efficiently scan the image, it can detect untextured objects in complex situations and provide their 3D pose in real-time.

1. Introduction

Currently, the dominant approach to object recognition is to use statistical learning to build a classifier offline, and then to use it at run-time for the recognition [17]. This works remarkably well but is not applicable for all scenarios, for example, a system that has to continuously learn new objects online. It is then difficult, or even impossible, to update the classifier without losing efficiency.

To overcome this problem, we propose an approach based on real-time template recognition. With such a tool at hand, it is then trivial and virtually instantaneous to learn new incoming objects by simply adding new templates to the database while simultaneously maintaining reliable real-time recognition.

However, we also wish to keep the advantages of statistical methods, as they learn how to reject unpromising image locations very quickly, which increases their real-time performance considerably. They can also be very robust, because they can generalize well from the training set. For these reasons, we also designed our template representation based on some fast to compute image statistics that provide invariance to small translations and deformations, which in turn allows us to quickly yet reliably search the image.

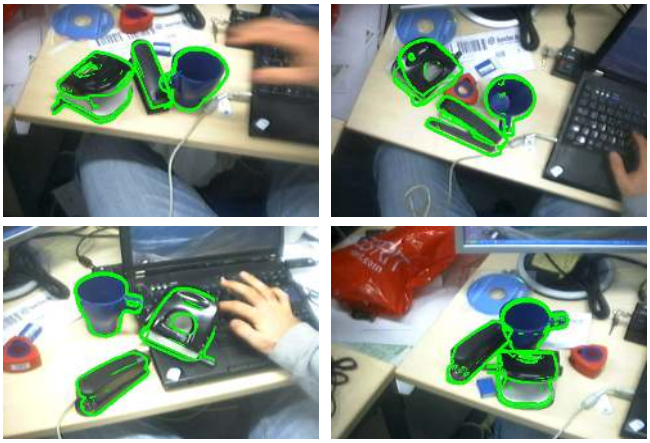


Figure 1. Overview. Our templates can detect non-textured objects over cluttered background in real-time without relying on feature point detection. Adding new objects is fast and easy, as it can be done online without the need for an initial training set. Only a few templates are required to cover all appearances of the objects.

As shown in Figure 1, in this paper we propose a template representation that is invariant enough to make search in the images very fast and generalizes well. As a result, we can almost instantaneously learn new objects and recognize them in real-time without requiring much time for training or any feature point detection at runtime.

Our representation is related to the Histograms-of-Gradients (HoG) based representation [1] that has proved to generalize well. Instead of local histograms, it relies on locally dominant orientations, and is made explicitly invariant to small translations. Our experiments show it is in practice at least as discriminant as HoG, while being much faster. Because it is explicitly made invariant to small translations, we can skip many locations while parsing the images without the risk of missing the targets. Moreover we developed a bit-coding method inspired by [16] to evaluate an image location for the presence of a template. It mostly uses simple bit-wise operations, and is therefore very fast on modern CPUs. Our similarity measure also fulfills the requirements for recent branch-and-bound exploration techniques [10],

speeding-up the search even more.

In the remainder of the paper we first discuss related work before we explain our template representation and how similarity can be evaluated very fast. We then show quantitative experiments and real world applications of our method.

2. Related Work

Template Matching is attractive for object detection because of its simplicity and its capability to handle different types of objects. It neither needs a large training set nor a time-consuming training stage, and can handle low-textured objects, which are, for example, difficult to detect with feature points-based methods.

An early approach to Template Matching [13] and its extension [3] include the use of the Chamfer distance between the template and the input image contours as a dissimilarity measure. This distance can efficiently be computed using the image Distance Transform (DT). It tends to generate many false positives, but [13] shows that taking the orientations into account drastically reduces the number of false positives. [9] is also based on the Distance Transform, however, it is invariant to scale changes and robust enough against perspective distortions to do real-time matching. Unfortunately, it is restricted to objects with closed contours, which are not always available.

But the main weakness of all Distance Transform-based methods is the need to extract contour points, using Canny method for example, and this stage is relatively fragile. It is sensitive to illumination changes, noise and blur. For instance, if the image contrast is lowered, contours on the object may not be detected and the detection will fail.

The method proposed in [15] tries to overcome these limitations by considering the image gradients in contrast to the image contours. It relies on the dot product as a similarity measure between the template gradients and those in the image. Unfortunately, this measure rapidly declines with the distance to the object location, or when the object appearance is even slightly distorted. As a result, the similarity measure must be evaluated densely, and with many templates to handle appearance variations, making the method computationally costly. Using image pyramids provides some speed improvements, however, fine but important structures tend to be lost if one does not carefully sample the scale space.

Histogram of Gradients [1] is another very popular method. It describes the local distributions of image gradients as computed on a regular grid. It has proven to give reliable results but tends to be slow due to the computational complexity.

Recently, [2] proposed a learning-based method that recognizes objects via a Hough-style voting scheme with a non-rigid shape matcher on the contour image. It relies on statistical methods to learn the model from few images that are only constraint with a bounding box around the object. While giving very good classification results, the approach is neither appropriate for object tracking in real-time due to its expensive computation nor it is exact enough to return the correct pose of the object. Moreover, it holds all the disadvantages of Distance Transform based methods as mentioned previously.

Grabner and Bischof [4, 5] developed another learning based approach that put more focus on online learning. In [4, 5] it is shown how a classifier can be trained online in real-time, with a training set generated automatically. However, [4] was demonstrated on textured objects, and [5] cannot provide the object pose.

The method proposed in this paper has the strength of the similarity measure of [15], the robustness of [1] and the online learning capability of [4, 5]. In addition, by binarizing the template representation and using a recent branch-and-bound method of [10] our method becomes very fast, making possible the detection of untextured 3D objects in real-time.

3. Proposed Approach

In this section, we describe our Dominant Orientation Templates, and how they can be built and used to parse images to quickly find objects. We will start by deriving our similarity measure, emphasizing the contributions of each aspect of it. We then show how to use a binary representation to compute the similarity using efficient bit-wise operations. We finally demonstrate how to use it within a branch-and-bound exploration of the image.

3.1. Initial Similarity Measure

Our starting idea is to measure the similarity between an input image $\mathcal{I}$, and a reference image $\mathcal{O}$ of an object centered on a location c in the image $\mathcal{I}$ by comparing the orientations of their gradients.

We chose to consider image gradients because they proved to be more discriminant than other forms of representations [11, 15] and are robust to illumination change and noise. For even more robustness to such changes, we use their magnitudes only to retain the orientations of the strongest gradients, without using their actual values for matching. Also, to correctly handle object occluding boundaries, we consider only the orientations of the gradients, by contrast with their directions (two vectors with a $180\mathrm{deg}$ angle between them have the same orientation).. In this way, the measure will not be affected if the object is over a dark background, or a bright background. Moreover, as in SIFT or HoG [1], we discretize the orientations to a small number n_o of integer values.

Our initial energy function $\mathcal{E}_1$ counts how many orienta-

tions are similar between the image and the template centered on location c, and can be formalized as:

$$\mathcal{E}_1(\mathcal{I}, \mathcal{O}, c) = \sum_r \delta\Big(\text{ori}(\mathcal{I}, c+r) = \text{ori}(\mathcal{O}, r)\Big) , \quad (1)$$

where

- $\delta(P)$ is a binary function that returns 1 if P is true, 0 otherwise;
- $\text{ori}(\mathcal{O}, r)$ is the discretized gradient orientation in the reference image $\mathcal{O}$ at location r which parses the template. Similarly, $\text{ori}(\mathcal{I}, c+r)$ is the discretized gradient orientation at c shifted by r in the input image $\mathcal{I}$.

3.2. Robustness to Small Deformations

To make our measure tolerant to small deformations, and also to make it faster to compute, we will not consider all possible locations, and will decompose the two images into small squared regions $\mathcal{R}$ over a regular grid. For each region, we will consider only the dominant orientations. Such an approach is similar to the HMAX pooling mechanism [14]. Our similarity measure can now be modified as:

$$\mathcal{E}_2(\mathcal{I}, \mathcal{O}, c) = \sum_{\mathcal{R} \text{ in } \mathcal{O}} \delta\Big(\text{do}(\mathcal{I}, c+\mathcal{R}) \in \text{DO}(\mathcal{O}, \mathcal{R})\Big) , \quad (2)$$

where $\text{DO}(\mathcal{O}, \mathcal{R})$ returns the set of orientations of the strongest gradients in region $\mathcal{R}$ of the object reference image. In contrast, $\text{do}(\mathcal{I}, c+\mathcal{R})$ returns only one orientation, the orientation of the strongest gradient in the region $\mathcal{R}$ shifted by c in the input image.

The reason why we chose each region in $\mathcal{O}$ to be represented by the strongest gradients is that the strongest gradients are easy and fast to identify and very robust to noise and illumination change. Moreover, to describe uniform regions, we introduce the symbol $\perp$ to indicate that no reliable gradient information is available for the region. The $\text{DO}(.)$ function therefore returns either a set of discretized gradient orientations of the k strongest gradients in the range of $[0, n_o - 1]$ or $\{\perp\}$, and can be formally written as:

$$\text{DO}(\mathcal{O}, \mathcal{R}) = \begin{cases} S(\mathcal{O}, \mathcal{R}) & \text{if } S(\mathcal{O}, \mathcal{R}) \neq \emptyset, \\ \{\perp\} & \text{otherwise} \end{cases} \quad (3)$$

with

$$S(\mathcal{O}, \mathcal{R}) = \{\text{ori}\,(\mathcal{O}, l) : l \in \text{maxmag}_k(\mathcal{R}) \wedge \text{mag}(\mathcal{O}, l) > \tau\} \quad (4)$$

where

- l is a pixel location in $\mathcal{R}$,
- $\text{ori}(\mathcal{O}, l)$ is the gradient orientation at l in image $\mathcal{O}$, and $\text{mag}(\mathcal{O}, l)$ its magnitude,

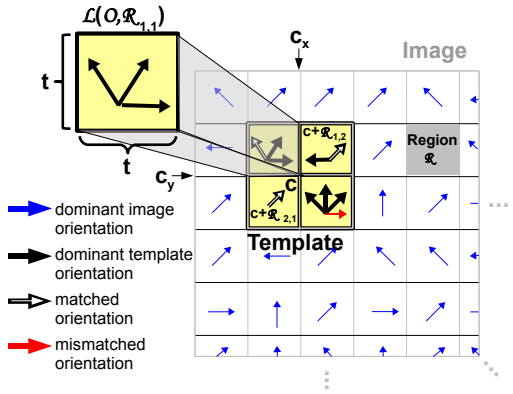


Figure 2. Similarity measure $\mathcal{E}_4$. Our final energy measure $\mathcal{E}_4$ counts how many times a local dominant orientation for a region $\mathcal{R}$ in the image belongs to the corresponding precomputed list of orientations $\mathcal{L}(\mathcal{O}, \mathcal{R})$ for the corresponding template region. Each list is made of the local dominant orientations that are in the region $\mathcal{R}$ when the object template is slightly translated.

- $\text{maxmag}_k(\mathcal{R})$ is the set of locations for the k strongest gradients in $\mathcal{R}$. In practice we take $k = 7$ but the choice of k does not seem critical.
- τ is a threshold on the gradient magnitudes to decide if the region is uniform or not.

The function $\text{do}(\mathcal{I}, c+\mathcal{R})$ is computed similarly in the input image $\mathcal{I}$. However, to be faster at runtime, in $\text{do}(\mathcal{I}, c+\mathcal{R})$, k is restricted to 1, and therefore $\text{do}(\mathcal{I}, c+\mathcal{R})$ returns only one single element.

3.3. Invariance to Small Translation

We will now explicitly make our similarity measure invariant to small motions. In this way, we will be able to consider only a limited number of locations c when parsing an image and save a significant amount of time without increasing the chance of missing the target object. To do so, we consider a measure that returns the maximal value of $\mathcal{E}_2$ when the object is slightly moved, which can be written as:

$$\begin{aligned} \mathcal{E}_3(\mathcal{I}, \mathcal{O}, c) &= \max_{M \in \mathcal{M}} \mathcal{E}_2(\mathcal{I}, \mathbf{w}(\mathcal{O}, M), c) \\ &= \max_{M \in \mathcal{M}} \sum_{\mathcal{R} \text{ in } \mathcal{O}} \delta\Big(\text{do}(\mathcal{I}, c+\mathcal{R}) \in \text{DO}(\mathbf{w}(\mathcal{O}, M), \mathcal{R})\Big) , \end{aligned} \quad (5)$$

where $\mathbf{w}(\mathcal{O}, M)$ is the image $\mathcal{O}$ of the object warped using a transformation M. In practice, we consider for M only 2D translations as it appears sufficient to handle other small deformations, and $\mathcal{M}$ is the set of all (small) translations in the range $[-t; +t]^2$.

There is of course a limit for the range t. A large t will result in high speed-up but also in a loss of discriminative power of the function. In practice, we found that $t = 7$ for 640×480 images is a good trade-off.

3.4. Ignoring the Dependence between Regions

Our last step is to ignore the dependence between the different regions $\mathcal{R}$. This will simplify and significantly speed-up the computation of the similarity. We therefore approximate $\mathcal{E}_3$ as given in Eq.(5) by:

$$\begin{aligned} & \mathcal{E}_4(\mathcal{I}, \mathcal{O}, c) \\ = & \sum_{\mathcal{R} \text{ in } \mathcal{O}} \max_{M \in \mathcal{M}} \delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \text{DO}(\mathbf{w}(\mathcal{O}, M), \mathcal{R})\Big) \,. \end{aligned} \tag{6}$$

The speed-up comes from the fact that, for each region $\mathcal{R}$, we can precompute a list $\mathcal{L}(\mathcal{O}, \mathcal{R})$ of the dominant orientations in $\mathcal{R}$ when $\mathcal{O}$ is translated over $\mathcal{M}$. As illustrated by Figure 2, the measure $\mathcal{E}_4$ can thus be written as:

$$\mathcal{E}_4(\mathcal{I}, \mathcal{O}, c) = \sum_{\mathcal{R} \text{ in } \mathcal{O}} \delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \mathcal{L}(\mathcal{O}, \mathcal{R})\Big) \,, \tag{7}$$

and $\mathcal{L}(\mathcal{O}, \mathcal{R})$ can formally be written as:

$$\begin{aligned} & \mathcal{L}(\mathcal{O}, \mathcal{R}) \\ = & \{o : \exists M \in \mathcal{M} \text{ such that } o \in \text{DO}(\mathbf{w}(\mathcal{O}, M), \mathcal{R})\} \,. \end{aligned} \tag{8}$$

The collection of lists over all regions $\mathcal{R}$ in $\mathcal{O}$ forms the final object template.

3.5. Using Bitwise Operations

Inspired by [16], and as shown in Figure 3, we efficiently compute the energy function $\mathcal{E}_4$ using a binary representation of the lists $\mathcal{L}(\mathcal{O}, \mathcal{R})$ and of the dominant orientations $\text{do}(\mathcal{I}, c + \mathcal{R})$. This allows us to compute $\mathcal{E}_4$ with only a few bitwise operations.

By setting n_o, the number of discretized orientations, to 7 we can represent a list $\mathcal{L}(\mathcal{O}, \mathcal{R})$ or a dominant orientation $\text{do}(\mathcal{I}, c + \mathcal{R})$ with one byte i.e. a 8-bit integer. Each of the 7 first bits corresponds to an orientation while the last bit stands for $\perp$.

More exactly, to each list $\mathcal{L}(\mathcal{O}, \mathcal{R})$ corresponds a byte L whose i^{th} bit with $0 \leq i \leq 6$ is set to 1 iff $i \in \mathcal{L}(\mathcal{O}, \mathcal{R})$, and whose 7^{th} bit is set to 1 iff $\perp \in \mathcal{L}(\mathcal{O}, \mathcal{R})$. A byte D can be constructed similarly to represent a dominant orientation $\text{do}(\mathcal{I}, c + \mathcal{R})$. Note that only one bit of D is set to 1. Now the term $\delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \mathcal{L}(\mathcal{O}, \mathcal{R})\Big)$ in Eq.(7) can be evaluated very quickly. We have:

$$\delta\Big(\text{do}(\mathcal{I}, c + \mathcal{R}) \in \mathcal{L}(\mathcal{O}, \mathcal{R})\Big) = 1 \text{ iff } L \otimes D \neq 0 \,, \tag{9}$$

where $\otimes$ is the bitwise AND operation.

3.6. Using SSE Instructions

The computation of $\mathcal{E}_4$ as formulated in Section 3.5 can be further speeded up using SSE operations. In addition to bitwise operations, which are already very fast, SSE technology allows to perform the same operation on 16 bytes in parallel. Thus, by using the function given in Listing 1, the similarity score for 16 regions can be computed with only 3 SSE operations and one lookup-table with 16-bits entries. Thus, if n denotes the number of regions $\mathcal{R}$, we only have to use $3\left\lceil \frac{n}{16} \right\rceil$ SEE instructions, $\left\lceil \frac{n}{16} \right\rceil$ uses of a lookup table with 16-bits entries and additional $\left\lceil \frac{n}{16} \right\rceil - 1$ "+" operations if the number of regions n is larger than 16. Assuming that each operation has the same computational cost we need $5\left\lceil \frac{n}{16} \right\rceil - 1$ operations for n regions which results in only ≈ 0.3 operations per region.

This method is extremely cache friendly because only successive chunks of 128 bits are processed at a time which holds the number of cache misses low. This is very important because SSE technology is very sensitive to optimal cache alignment. This is probably why, although our energy function is slightly more computationally expensive in theory than [16], we found that our formulation performed 1.5 times faster in practice.

Another advantage of our algorithm, however, is that it is very flexible with respect to varying template sizes without loosing the capability of using the computational capacities very efficiently. In our method, the optimal processor load is reached by multiples of 16 in contrast to [16] that needs

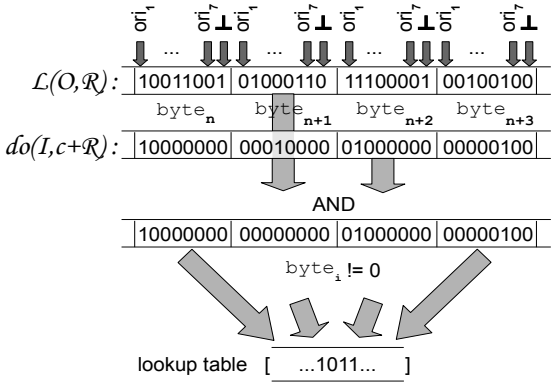
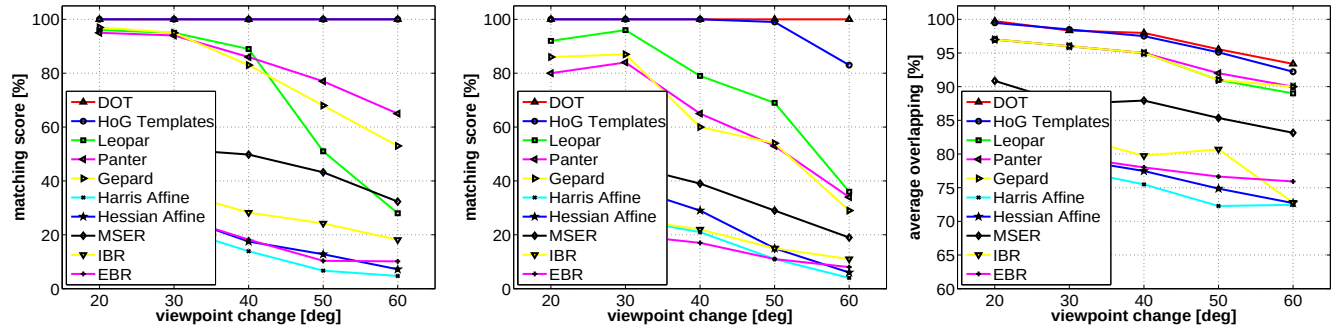


Figure 3. Computing the similarity $\mathcal{E}_4$ using bitwise operations and a lookup table that counts how many terms $\delta()$ as in Eq.(9) are equal to 1.

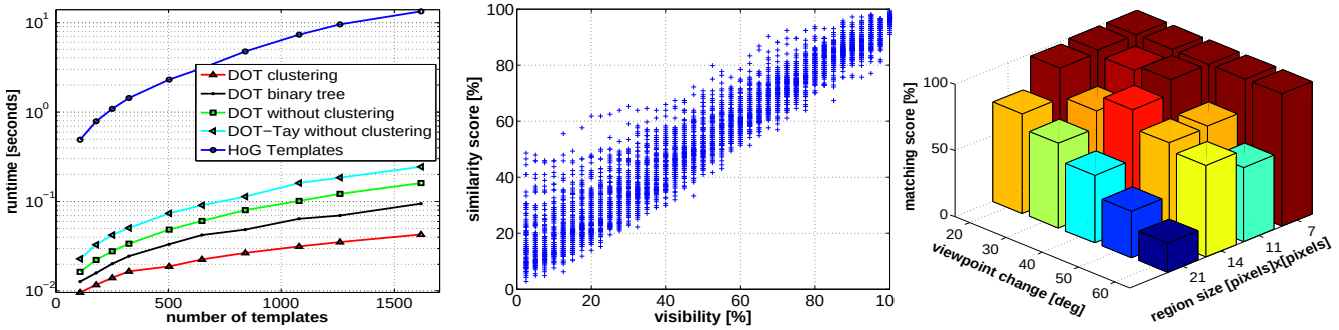
```
int energy_function4( __m128i lhs, __m128i rhs )
{
    __m128i a = _mm_and_si128(lhs,rhs);
    __m128i b = _mm_cmpeq_epi8(a);

    return lookuptable[_mm_movemask_epi8(b)];
}
```

Listing 1. C++ Energy function for 16 regions with 3 SSE instructions and one look-up in a 16-bit-table. Since in SSE there is no comparison on non-equality for unsigned 8-bit integers we have—in contrast to Figure 3—to compare the AND'ed result to zero and count the "0" instead.



(a) (b) (c)

Figure 4. Methods comparisons on the Graffiti and Wall Oxford datasets. (a-b): Matching scores for Graffiti and Wall sets when increasing the viewpoint angle. Our method is referred as "DOT", and reaches a 100% score on both sets for every angle. These results are discussed in Section 4.1. (c) shows the overlaps between the retrieved and expected regions as an accuracy measure for Graffiti. These results are discussed in Section 4.2.



(a) (b) (c)

Figure 5. (a) Comparison of different methods and cluster schemes with respect to speed. Our method with our cluster scheme performs superior over all other methods and cluster schemes as discussed in Section 4.3. (b) In Section 4.4 we discuss the linear behavior of our method with respect to occlusion. (c) $t = 7$ is a good trade-off between speed and robustness (Section 4.5).

multiples of 128 in a possible dynamic SSE implementation. The probability of wasting computational power is therefore much lower using our approach.

3.7. Clustering for Efficient Branch and Bound

We can further improve the scalability of our method by exploiting the similarity between different templates representing different objects under different views. The general idea is to build clusters of similar templates—each of them being represented by what we will refer to as a *cluster template*. A cluster template is computed as a bitwise OR operation applied to all the templates belonging to the same cluster. It provides tight upper bounds and can be used in a branch and bound constrained search as in [10]. By first computing the similarity measure $\mathcal{E}_4$ between the image and the cluster templates at run-time, we can reject all the templates that belong to a cluster template not similar enough to the current image.

We use a bottom-up clustering method: To build a cluster, we start from a template picked randomly among the templates that do not yet belong to a cluster. We then look for the template the most similar to it according to the Hamming distance, and not picked yet. We proceed this way using the Hamming distance between the current cluster template and the remaining templates, until the cluster has a given number of templates assigned. We then continue building clusters until every template is assigned to a cluster.

For our approach, this clustering scheme allows faster runtime than the binary tree clustering suggested in [16], as will be shown in Section 4.3.

4. Experimental Results

In the experiments, we compared our approach called DOT (for *Dominant Orientation Templates*) to Affine Region Detectors [12] (Harris-Affine, Hessian-Affine, MSER, IBR, EBR), to patch rectification methods [8, 7, 6] (Leopar, Panter, Gepard) and to the Histograms-of-Gradients (HoG) template matching approach [1].

For HoG, we used our own SSE optimized implementation. In order to detect the correct template from a large template database we replaced the Support Vector Machine mentioned in the original work of HoG by a nearest neighbor search since we want to avoid a training phase and to look for a robust representation instead.

We did the performance evaluation on the Oxford Graffiti and on the Oxford Wall image set [12]. Since no video

sequence is available, we synthesized a training set by scaling and rotating the first image of the dataset for changes in viewpoint angle up to 75 degrees and by adding random noise and affine illumination change.

4.1. Robustness

The matching scores of the different methods is shown in Figure 4(a) for the Graffiti dataset, and in Figure 4(b) for the Wall dataset. As defined in [12], this score is the ratio between the number of correct matches and the smaller number of regions detected in one of the two images.

For the affine regions, we first extract the regions using different region detectors and match them using SIFT. Two of them are said to be correctly matched if the overlap error of the normalized regions is smaller than 40%. In our case, the regions are defined as the patches warped by the retrieved transformation. For a fair comparison, we used the same numbers and appearances of templates for the DOT and HoG approaches. We also turned off the final check on the correlation for all patch rectification approaches (Leopar, Panter, Gepard) since there is no equivalent for the affine regions.

DOT and HoG clearly outperform the other approaches by delivering optimal matching results of 100% on the Graffiti image set. For the Wall image set, DOT performs optimal again with a matching rate of 100% while HoG performs worse for larger viewpoint changes.

These very good performances can be explained by the fact that DOT and HoG scan the whole image while the affine regions approach is dependent on the quality of the region extraction. As it will be shown in Section 4.3, even if it parses the whole image, our approach is fast enough to compete with affine region and patch rectification approaches in terms of computation times.

4.2. Detection Accuracy

As it was done in [7], in Figure 4(c), we compare the average overlap between the ground truth quadrangles and their corresponding warped versions obtained with DOT, HoG, the patch rectification methods and with the affine regions detectors. We did the experiments for overlap and accuracy on both image sets but due to the similarity of the results and the lack of space we only show the results on the Graffiti image set. Since the Affine Region Detectors deliver elliptic regions we fit quadrangles around these ellipses by aligning them to the main gradient orientation as it was done in [7].

The average overlap is very close to 100% for DOT and HoG, about 10% better than MSER and about 20% better than the other affine region detectors.

4.3. Speed

Although performing similar in terms of robustness and accuracy, DOT clearly outperforms HoG in terms of speed by several magnitudes. In order to compare both approaches, we trained them on the same locations and appearances on a 640×480 image with $|\mathcal{R}| = 121$. The experiment was done on a standard notebook with an Intel Centrino Processor Core2Duo with 2.4GHz and 3GB RAM where unoptimized training of one template took 1.8ms and the clustering of about 1600 templates 0.76s. As one can see in Figure 5(a), when using about 1600 templates our approach is about 310 times faster at runtime than our SSE optimized HoG implementation. The reason for this is both the robustness to small deformations that allows DOT to skip most of the pixel locations and the binary representation of our templates that enables a fast similarity evaluation.

We also compared our similarity measure to a SSE optimized version of Taylor's version [16]. Our approach is constantly about 1.5 times faster than Taylor's. We believe it is due to the cache friendly formulation of $\mathcal{E}_4$ where we successively use sequential chunks of 128 bits at a time while [16] has to jump back and forth within 1024 bits (in case $|\mathcal{R}| = 121$) for successively OR'ing pairs of 128 bit vectors and accumulating the result (for a closer explanation of Taylor's similarity measure please refer to [16]) in a SSE register.

We also did experiments with respect to the different clustering schemes. We compared the approach where no clustering is used to the binary tree of [16] and our clustering described in Section 3.7. Surprisingly, our clustering is twice as fast as the binary tree clustering at runtime. Although the matching should behave in $O(log(N))$ time, our implementation of the binary tree clustering behaves linearly up to about 1600 templates as it was also observed by [16]. As the authors of [16] claim, the reason for this might be that there are not enough overlapping templates to fully exploit the potential of their tree structure.

4.4. Occlusion

Occlusion is a very important aspect in template matching. To test our approach towards occlusion we selected 100 templates on the first image of the Oxford Graffiti image set, added small image deformation, noise and illumination changes and incrementally occluded the template in 2.5% steps from 0% to 100%. The results are displayed in Figure 5(b). As expected the similarity of our method behaves linearly to the percentage of occlusion. This is a desirable property since it allows to detect partly occluded templates by setting the detection threshold with respect to the tolerated percentage of occlusion.

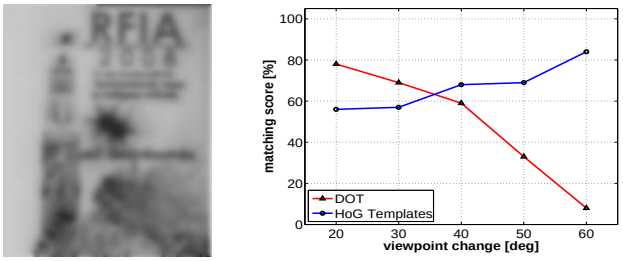


Figure 6. Failure Case. When the object does not exhibit strong gradients, like the blurry image on the left, our method performs worse than HoG.

4.5. Region Size

The size of the region $\mathcal{R}$ is another important parameter. The larger the region $\mathcal{R}$ gets the faster the approach becomes at runtime. However, at the same time as the size of the region increases the discriminative power of the approach decreases since the number of gradients to be considered rises. Therefore, it is necessary to choose the size of the region $\mathcal{R}$ carefully to find a compromise between speed and robustness. In the following experiment on the Graffiti image set we tested the behavior of DOT with respect to the matching score and the size of the region $\mathcal{R}$. The result is shown in Figure 5(c). As the matching score is still 100% for regions of 7×7 pixels, one can see that the robustness decreases with increasing region size. Although dependent on the texture and on the density of strong gradients within one region $\mathcal{R}$, we empirically found on many different objects that a region size of 7×7 gives very good results.

4.6. Failure Cases

Figure 6 shows the limitation of our method: To obtain such optimal results as in Figure 4, the templates have to exhibit strong gradients. In case of too smooth or blurry template images, HoG tends to perform better.

4.7. Applications

Due to the robustness and the real-time capability of our approach, DOT is suited for many different applications including untextured object detection as shown in Figure 8, and planar patches detection as shown in Figure 9. Although neither a final refinement nor any final verification, by contrast with [7] for example, was applied to the found 3D objects, the results are very accurate, robust and stable. Creating the templates for new objects is easy and illustrated by Figure 7.

5. Conclusion

We introduce a new binary template representation based on locally dominant gradient orientations that is invariant to small image deformations. It can very reliably detect untextured 3D objects using relatively few templates from many different viewpoints in real-time. We have shown that our approach performs superior to state-of-the-art methods with respect to the combination of recognition rate and speed. Moreover, the template creation is fast and easy, does not require a training set, only a few exemplars, and can be done interactively.

Acknowledgment: This project was funded by the BMBF project AVILUSplus (01IM08002).

References

[1] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[2] V. Ferrari, F. Jurie, and C. Schmid. From images to shape models for object detection. *IJCV*, 2009.

[3] D. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *ICCV*, 1999.

[4] M. Grabner, H. Grabner, and H. Bischof. Tracking via Discriminative Online Learning of Local Features. In *CVPR*, 2007.

[5] M. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008.

[6] S. Hinterstoisser, S. Benhimane, V. Lepetit, P. Fua, and N. Navab. Simultaneous recognition and homography extraction of local patches with a simple linear classifier. In *BMVC*, 2008.

[7] S. Hinterstoisser, S. Benhimane, N. Navab, P. Fua, and V. Lepetit. Online learning of patch perspective rectification for efficient object detection. In *CVPR*, 2008.

[8] S. Hinterstoisser, O. Kutter, N. Navab, P. Fua, and V. Lepetit. Real-time learning of accurate patch rectification. In *CVPR*, 2009.

[9] S. Holzer, S. Hinterstoisser, S. Ilic, and N. Navab. Distance transform templates for object detection and pose estimation. In *CVPR*, 2009.

[10] C. H. Lampert, M. B. Blaschko, and T. Hofmann. Beyond Sliding Windows: Object Localization by Efficient Subwindow Search. In *CVPR*, June 2008.

[11] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 20(2):91–110, 2004.

[12] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 2005.

[13] C. F. Olson and D. P. Huttenlocher. Automatic target recognition by matching oriented edge pixels. *IP*, 6, 1997.

[14] T. Serre and M. Riesenhuber. Realistic modeling of simple and complex cell tuning in the hmax model, and implications for invariant object recognition in cortex. TR, MIT, 2004.

[15] C. Steger. Occlusion Clutter, and Illumination Invariant Object Recognition. In *IAPRS*, 2002.

[16] S. Taylor and T. Drummond. Multiple target localisation at over 100 fps. In *BMVC*, 2009.

[17] P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 2001.

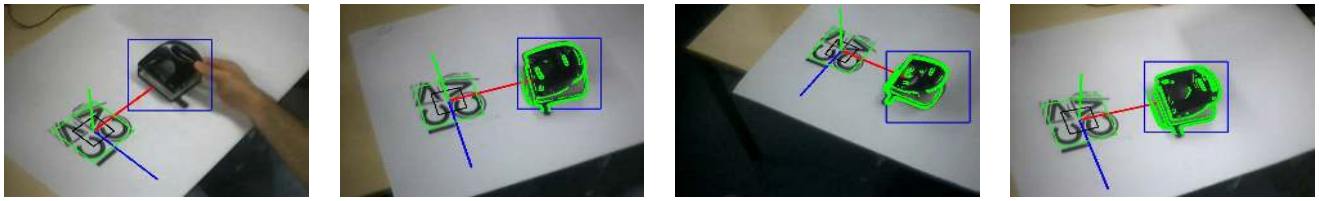


Figure 7. Templates creation. To easily define the templates for a new object, we use DOT to detect a known object—the ICCV logo in this case—next to the object to learn in order to estimate the camera pose and to define an area in which the object to learn is located. A template for the new object is created from the first image, and we start detecting the object while moving the camera. When the detection score becomes too low, a new template is created in order to cover the different object appearances when the viewpoint changes.

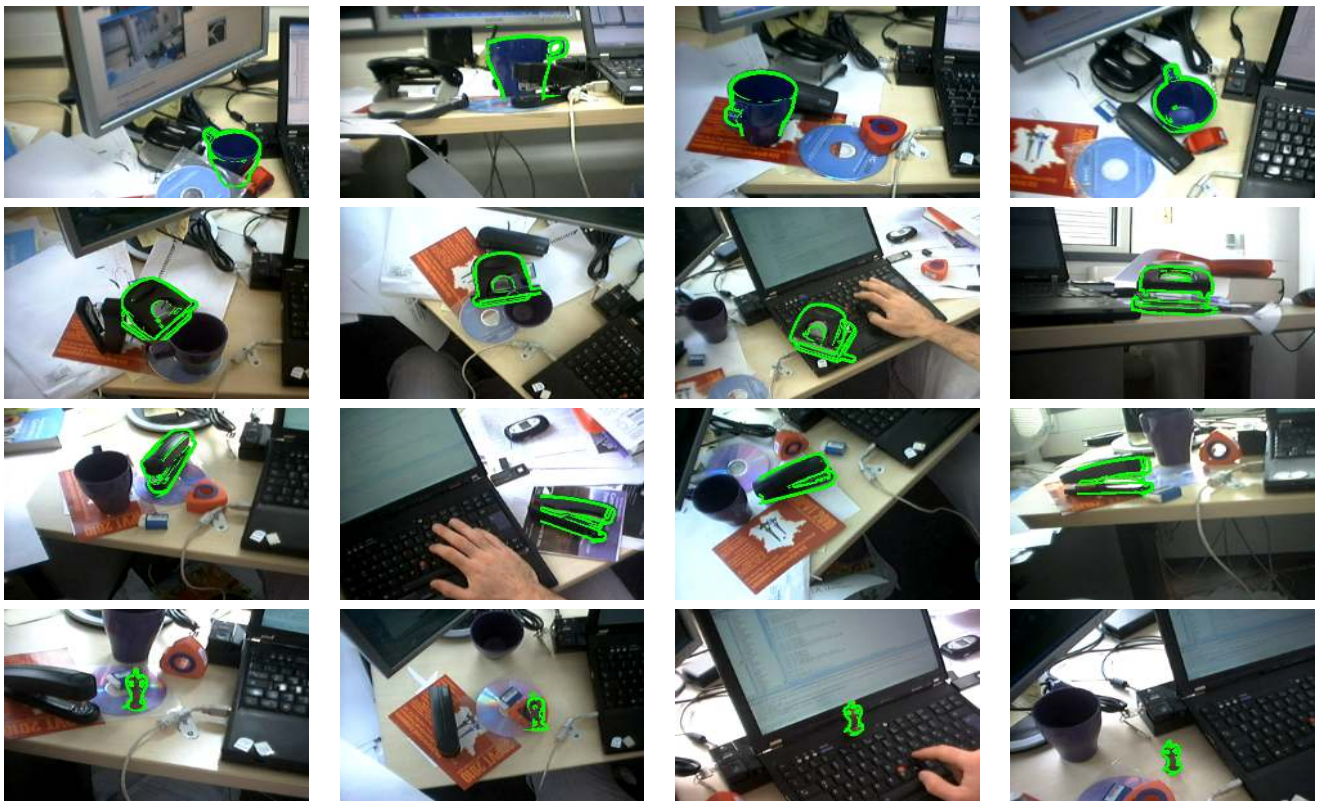


Figure 8. Detection of different objects at about 12 fps over a cluttered background. The detections are shown by superimposing the thresholded gradient magnitudes from the object image over the input images. *The corresponding video is available on http://campar.in.tum.de/Main/StefanHinterstoisser.*

Figure 9. Patch 3D orientation estimation. Like Gepard [8], DOT can detect planar patches and provide an estimate of their orientations. DOT is however much more reliable as it does not rely on feature point detection, but parses the image instead. *The corresponding video is available on http://campar.in.tum.de/Main/StefanHinterstoisser.*