Improving Keypoint Orientation Assignment

BMVC 2011 Submission # 346

Abstract

Detection and description of local image features has proven to be a powerful paradigm for a variety of applications in computer vision. Often, this process includes an orientation assignment step to render the overall process invariant to in-plane rotation. In this paper, we review several different existing algorithms and propose two novel, efficient methods for orientation assignment. The first method exhibits a very good speed-performance trade-off; the second is capable of multiple orientations and performs comparable to SIFT's orientation assignment while being significantly cheaper. Additionally, we improve one of the existing orientation assignment methods by generalizing it. All algorithms are evaluated empirically under a variety of conditions and in combination with six keypoint detectors.

1 Introduction

Keypoint detection, description and matching has proven to be a powerful paradigm for a variety of applications in computer vision, including image retrieval, object recognition, and visual tracking. In many frameworks, this paradigm includes an orientation assignment step [2, 3, 7, 13] to make the overall process invariant to in-plane rotation. While this approach seems to work well and is widely accepted, the orientation assignment is frequently presented as a mere "add-on" to a descriptor, and little work has been devoted to the orientation assignment algorithms themselves. In particular, we are not aware of any studies that evaluate and compare different orientation assignment algorithms directly.

Our contributions in this paper include:

- We propose two novel, very efficient algorithms for orientation assignment; one if a single dominant orientation is sought, and the second capable of detecting multiple dominant orientations.
- We present a detailed quantitative evaluation and analysis of our two algorithms as well as four competing algorithms. In this process, we generalize one existing method (Taylor and Drummond [13]) and, by doing so, significantly improve its robustness. Our results entail observations about the orientation assignment problem in general as well as observations about individual algorithms.

After defining the problem in the remainder of this section, we review different existing algorithms for orientation assignment (Section 2). We then propose two novel, efficient algorithms, dubbed Center of Mass (Section 3) and Histogram of Intensities (Section 4). We empirically evaluate all these approaches under a variety of conditions and in combination with six different keypoint detectors, and we generalize one of the existing methods (Section 5). We discuss the outcomes and implications in Section 6 and conclude in Section 7.

Problem Statement. Given a keypoint $p = (x_p, y_p)$ in an image I, derive an angle $\theta_p \in [-\pi, \pi]$ based on the local neighborhood of p that captures the orientation of the local texture.

θ_p is arbitrary in that there is no correct or incorrect value given any individual image I. However, if the same real world point X is seen in two different images, the obtained orientations should be consistent, that is, identical relative to the orientation of X in the real world. Therefore, θ_p must be robust to image noise affecting keypoint location and image intensities, and, as far as possible, invariant to changes in viewpoint and lighting.

The purpose of orientation assignment is that it enables any subsequently derived image features to be invariant to in-plane image rotation by rotating the local coordinate frame in which it operates relative to θ_p. Alternatively, one can use a feature descriptor that is inherently rotation invariant [9, 11]. This approach, however, limits the low-level image features that can be used and is not compatible with many existing descriptors that are known to be distinctive. In contrast, an orientation assignment step is relatively easy to insert into any processing pipeline and can be used in conjunction with any image descriptor or other subsequent step.

In many keypoint detection frameworks, keypoints are also assigned a scale s [2, 3, 7]. In this case, the orientation assignment is typically done after the scale assignment, and I refers to the image sampled at the appropriate scale.

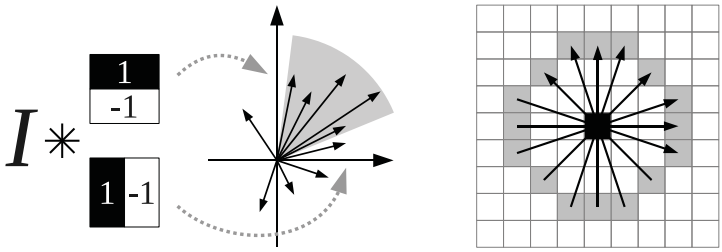
Notation. Following the naming convention in many programming languages, we denote the angle between the positive x-axis and the vector from the origin to a point $p = (x, y)$ with $\operatorname{arctan2}(y, x)$, or $\operatorname{arctan2}(p)$ for convenience. $\operatorname{Arctan2}(y, x)$ is equivalent to $\arctan(y/x)$ for $x > 0$ and appropriately shifted by $\pm\pi$ otherwise.

2 Related Work

SIFT's orientation assignment. Lowe [7] first computes the gradient orientation $\theta(x, y) = \operatorname{arctan2}(\nabla I(x, y))$ at each pixel (x, y) within a region R around the keypoint. Then, a histogram is formed of all $\theta(x, y)$. Each sample that is added to the histogram is weighted by the strength of the local gradient ∇I and a Gaussian around the keypoint location (x_p, y_p) The highest peak in this histogram and up to three further peaks (if higher than 80% of the highest peak) are refined by fitting a parabola to the histogram values around the respective location, and are then taken as keypoint orientations $\theta_{p,\mathrm{SIFT}}$.

SURF's orientation assignment. The orientation assignment used in SURF [2] is illustrated in Fig. 1(a). First, the image is convoluted with two first-order Haar wavelets. The filter responses at certain sampling points around the keypoint are represented as a vector in a two-dimensional space. Then, a rotating window of $\frac{\pi}{3}$ (gray circular sector in Fig. 1(a)) is used to sum up all vectors within its range, and the longest resulting vector determines the orientation $\theta_{p,\mathrm{SURF}}$.

Taylor's method. Taylor and Drummond's orientation assignment [13] is based on intensity differences of eight pixel pairs in a circle around the keypoint location. Each of the eight vectors shown in Fig. 1(b) is scaled by the intensity difference between the respective two pixels. The vectors are concatenated and the direction of the resulting vector determines the the keypoint orientation $\theta_{p,\mathrm{Taylor}}$. This method has the advantage of being extremely fast,



(a) SURF (b) Taylor

Figure 1: Orientation assignments in (a) SURF [2] and (b) Taylor and Drummond [13].

but it is susceptible to image noise and assumes existence of strong intensity differences in a relatively small neighborhood.

Smoothed gradient. Brown et al. [3] take the orientation from the local gradient of the image smoothed with a Gaussian G_σ:

$$\theta_{p,\nabla} = \arctan2(\nabla(G_\sigma * I)\big|_{(x_p,y_p)}) \tag{1}$$

A large σ ensures smooth variation of the gradient across the image and thus a certain robustness to errors in the keypoint location (x_p, y_p), however, the cost of this method also increases quadratically with σ.

3 "Center-of-Mass" Orientation Assignment

To derive a single orientation, we propose the following simple and very efficient algorithm:

Consider a circular neighborhood R around the keypoint (x_p, y_p) and compute the following sums:

$$c_x = \frac{1}{S}\sum_{(x,y)\in R} w(r)\cdot(x-x_p)\cdot I(x,y) \quad ; \quad c_y = \frac{1}{S}\sum_{(x,y)\in R} w(r)\cdot(y-y_p)\cdot I(x,y) \tag{2}$$

where $r = \|(x,y)-(x_p,y_p)\|$, $w(r)$ is a radial weighting function, and $S = \sum w(r)\cdot I(x,y)$. (c_x, c_y) is the (weighted) "center of mass" (CoM) of the neighborhood if each pixel's intensity is interpreted as its "mass." Its direction relative to the keypoint location is a function of the local intensities that is robust to small shifts in the keypoint location and image noise, and is thus well-suited as the keypoint's orientation. Since inception of this algorithm we have learned of at least one other group being interested in similar ways of orientation assignment [5].

We experimented with different radial weighting schemes $w(r)$ (such as uniform, linear, Gaussian), and finally used a quadratic weighting $w(r) = 1-(r/r_{\max})^2$ if $r \le r_{\max}$, 0 otherwise, which seems to be a good balance between creating a large "mass" S (which is important for robustness) and creating a smooth "fading out" towards the border.

The distance d between the keypoint and the CoM is an indicator of how stable the orientation assignment is. For example, in a rotation-symmetric (e.g., uniform) neighborhood R, d is (close to) 0 and the orientation is determined by noise. In our experiments, the average precision of keypoint matches increased from below 0.2 for the smallest d to above 0.6 for the largest d.

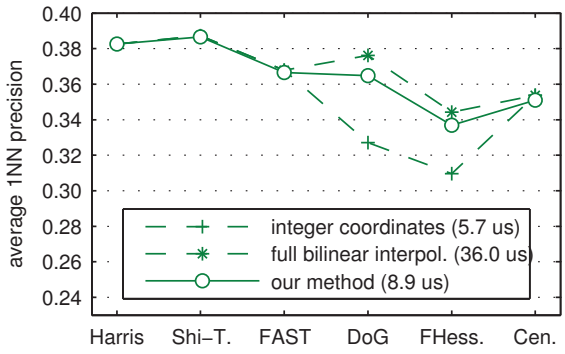


Figure 2: Average first nearest neighbor (1NN) precision for different interpolation methods for CoM, results broken down by the six different keypoint detectors. (For description of dataset, evaluation setup and performance metric cf. Section 5.) Our method is significantly faster than full bilinear interpolation for all coordinates (average timings see figure legend), but achieves similar performance.

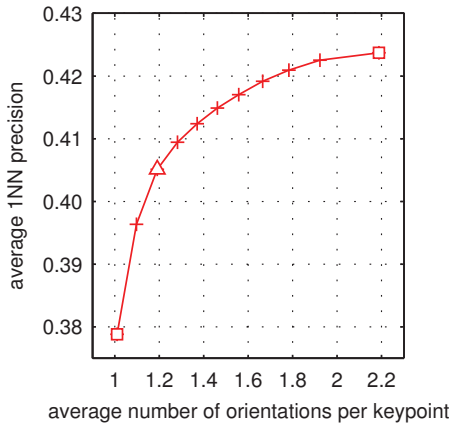


Figure 3: Precision of SIFT's orientation assignment vs. average number of orientations per keypoint. The minimum height (relative to the global maximum) for a peak in the histogram to be accepted was varied (0.99, 0.9, 0.8, ... 0.1, 0). For dataset and evaluation setup cf. Section 5. SIFT's default configuration (0.8) is marked by the triangle; for the evaluations below, we use the two configurations marked by the squares: a single orientation (for comparison with the other single-orientation methods) and accepting all peaks.

Subpixel accuracy. Several keypoint detectors provide subpixel-accurate keypoint locations, i.e., x_p and y_p are not necessarily integers. All of the aforementioned orientation assignment methods can be adapted to incorporate this by sampling pixels at non-integer locations using (e.g.) bilinear interpolation. However, interpolation is costly, requiring more multiplications per pixel than our CoM algorithm given by Eq. (2).

For CoM, almost the same performance can be achieved without interpolation as follows: In the summation, we iterate over integer coordinates x, y. However, we pre-compute a look-up table for $w(r)$ with sub-pixel granularity, and when computing $r = \|(x,y)-(x_p,y_p)\|$ to retrieve the correct weight $w(r)$, we use the subpixel-accurate values for x_p and y_p. This has a similar effect as interpolating the image at non-integer locations x,y and is significantly cheaper. The difference in performance is shown in Fig. 2.

4 "Histogram-of-Intensities" Orientation Assignment

One significant advantage of SIFT's histogram-based approach is that *multiple* dominant orientations can be extracted. Lowe [7] reports that this improves matching precision significantly, and we confirm this finding in Fig. 3 and in our evaluations in Section 5.

By combining the intensity-based computation of CoM with SIFT's histogram concept, we arrived at the following algorithm dubbed "Histogram of Intensities" (HoI): Each pixel (x,y) in a region R around the keypoint (x_p,y_p) is entered into a histogram based on the angle $\theta(x,y) = \arctan2(y-y_p, x-x_p)$ between the two points, weighted by its intensity $I(x,y)$ and a radial weighting function $w(\|(x,y)-(x_p,y_p)\|)$. From this histogram, dominant orientations can be extracted in the same fashion as done in SIFT: Each peak above a certain threshold is accepted as keypoint orientation $\theta_{p,\mathrm{HoI}}$ and refined by quadratic interpolation.

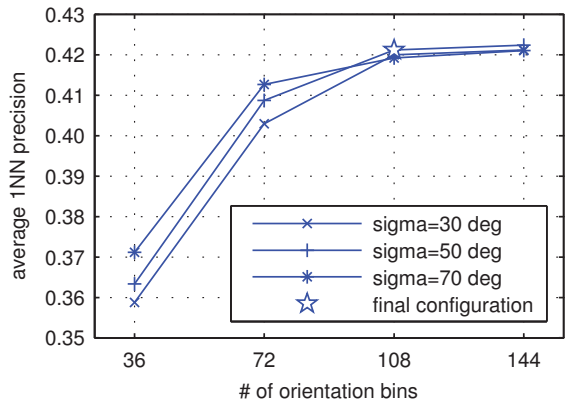


Figure 4: Precision of HoI for various parameter configurations. We evaluated the number of orientations bins (along x-axis), size of smoothing kernel (different lines), as well as the minimum peak height and the maximum number of orientations (not shown, values here are 0.9 and 5, respectively). The configuration that is used in Section 5 is marked by the star. For dataset and evaluation setup cf. Section 5.

Despite the structural similarity to SIFT's orientation assignment, it is important to note one fundamental difference: In SIFT, the histogram bin into which a value enters depends on the *direction of the local gradient*, while in our HoI approach, the histogram bin is computed from the relative orientation between pixel and keypoint. HoI may be understood as radially projecting all the intensities into a histogram fanned out around the keypoint.

There are two main factors that make our orientation assignment algorithm significantly faster than that of SIFT: (1) we do not require computation of the image gradients, and (2) we do not need to execute arctan2 and the subsequent computation of the corresponding bin index at every pixel: since the pixel offsets $y-y_p, x-x_p$ reoccur at every keypoint, we can pre-compute and store the corresponding bin index for each offset. Thus, the pixel-wise iteration over the region R consists of fast access into two look-up tables (the bin index and the radial weight w) and a single multiplication.

Furthermore, instead of using bilinear interpolation upon each entry into the histogram, we use a relatively large number of bins (such that each bin is "accurate" enough so that no interpolation is needed), and then smooth the filled histogram with a relatively wide Gaussian filter. While the smoothing operation is costly, it has to be performed only a single time and is thus cheaper than bilinear interpolation for every pixel if the number of pixels in region R is large compared to the number of bins and the size of the filter.

We tested various parameter configurations (cf. Fig. 4) and chose the following configuration: The same $w(r)$ as for CoM (cf. Section 3), 108 orientation bins (=3.33°/bin), smoothing of the histogram with a Gaussian filter with $\sigma = 50°$, a requirement for peaks to be within 90% of the global maximum, and a maximum of 5 orientations. With this configuration, HoI creates 3 orientations per keypoint on average.

Subpixel accuracy. We incorporate subpixel accuracy in a similar fashion as for CoM (cf. Section 3): instead of using bilinear interpolation for each intensity value, we use subpixel-accurate look-up tables. For a keypoint (x_p, y_p), we first extract its fractional components $x_f = x_p - \text{round}(x_p)$, $y_f = y_p - \text{round}(y_p)$ and use them to access look-up tables that were created using this fractional offset, that is, where the histogram bins and weights for each pixel were computed using an appropriate fractional shift. Using these look-up tables, we can iterate over R using integer coordinates without interpolation and achieve "floating point accuracy" with "integer effort." We found that 5×5 subpixel "cells" are sufficient to achieve the same robustness as with floating point coordinates.

5 Evaluation

To evaluate the aforementioned algorithms, we used them in a keypoint matching framework and measured the time to compute the orientation as well as the average precision achieved by the different algorithms under a variety of conditions. We first describe the setup in detail, then we present the experimental results.

Evaluation setup. We used the dataset published by Gauglitz et al. [4]. It contains videos of six different planar textures under several camera paths each as well as the ground truth position of the target in each frame. Here, we use the part of the dataset which features in-plane rotation only (6x 50 frames), as well as (for Fig. 7) the "unconstrained" paths featuring unconstrained camera motion around the object (6x 500 frames).

After detecting keypoints with one of six popular keypoint detectors (Harris [6], Shi-Tomasi [10], FAST [8]; Difference-of-Gaussians [7], Fast Hessian [2], and CenSurE [1]), each keypoint gets assigned one or (for SIFT and HoI) multiple orientations using the orientation assignment algorithm under evaluation, and finally, we sample an accordingly rotated 11x11 pixel image patch as image descriptor. From each sequence, we take 500 random pairs of images (random frame pairs as suggested by [4]). Each descriptor from the first image is paired with the descriptor with the smallest sum of squared differences (i.e., its first nearest neighbor, 1NN) among all descriptors from the second image. The match is classified as correct if they stem from the same location on the target within a 2 pixel tolerance. The average 1NN precision is thus the average fraction of correct matches per frame pair.

All figures show average results for all textures, image pairs, and detectors, except where broken down by one of these dimensions. In total, every data point shown in (e.g.) Fig. 6(b) aggregates 3,416,945 1NN keypoint pairs.

The precision is, of course, limited not only by inconsistent orientation assignments, but also by the repeatability of the detector (sometimes there is no correct match because the keypoint was not detected in both images) and the discriminativeness of the image descriptor. One could eliminate these effects by simulating the perfect detector and/or directly testing the assigned orientations for consistency, but this approach leads to unrealistic conditions and does not not adequately test the robustness to noise (such as, for example, small shifts in the keypoint location). Instead, we added an "oracle" which derives orientations directly from the provided ground truth; its results indicate an upper bound on what precision could be achieved with perfect orientation assignment given a particular condition. For further comparison, we also show the precision if no orientation assignment is used.

Implementation. Algorithms and evaluation framework were implemented in C++ using OpenCV and libCVD. We used the compiled SURF library provided by Bay et al. [2], and the open SIFT implementation by Vedaldi and Fulkerson [14], and implemented the other algorithms ourselves. For Taylor and Drummond's method [13], all pixels are sampled with bilinear interpolation. For Brown et al.'s method [3], we compute the convolution with the gradient and Gaussian filters only at the pixels immediately surrounding the keypoint's location (to avoid the overhead of filtering the entire image and get a more accurate per-keypoint timing) and then use bilinear interpolation of the convolution results. With 1x3 gradient kernels `[-1 0 1]`, the convolution has to be computed at 12 pixel locations per keypoint.

All evaluations were run on a PC with a 3.4 GHz Dual Core CPU (only one core used to run the algorithms) and 2 GB RAM running Ubuntu 10.4.

Evaluating the impact of the size of the neighborhood. All of the aforementioned orientation assignment algorithms are based on a local neighborhood R around the keypoint. Like most local feature descriptors, they assume that this neighborhood is approximately rigid and planar. It is intuitively clear that the robustness to small keypoint shifts and image noise increases with an increasing size of R as long as these assumptions are fulfilled. Therefore, we first evaluated all algorithms for different sizes of the neighborhood region R that is used to derive the orientation.[1]

Taylor's method as presented in [12] and [13] only supports one fixed neighborhood size. We generalize their method as follows: Instead of always using the ring depicted in Fig. 1(b), we implemented six different rings with diameters 7 (the original), 11, 15, 21, 29, and 41 as depicted in Fig. 5. As in the original algorithm, we use the intensity difference between opposite pixels to scale the respective vector and concatenate all resulting vectors.

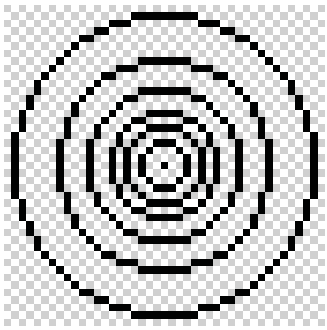


Figure 5: Ring geometries that we used to generalize Taylor's method.

The results shown in Fig. 6(b) confirm the intuition above: the performance of all algorithms increases with the size of the neighborhood. Interestingly, this is true for Taylor's method as well, which we significantly improved by using larger rings.

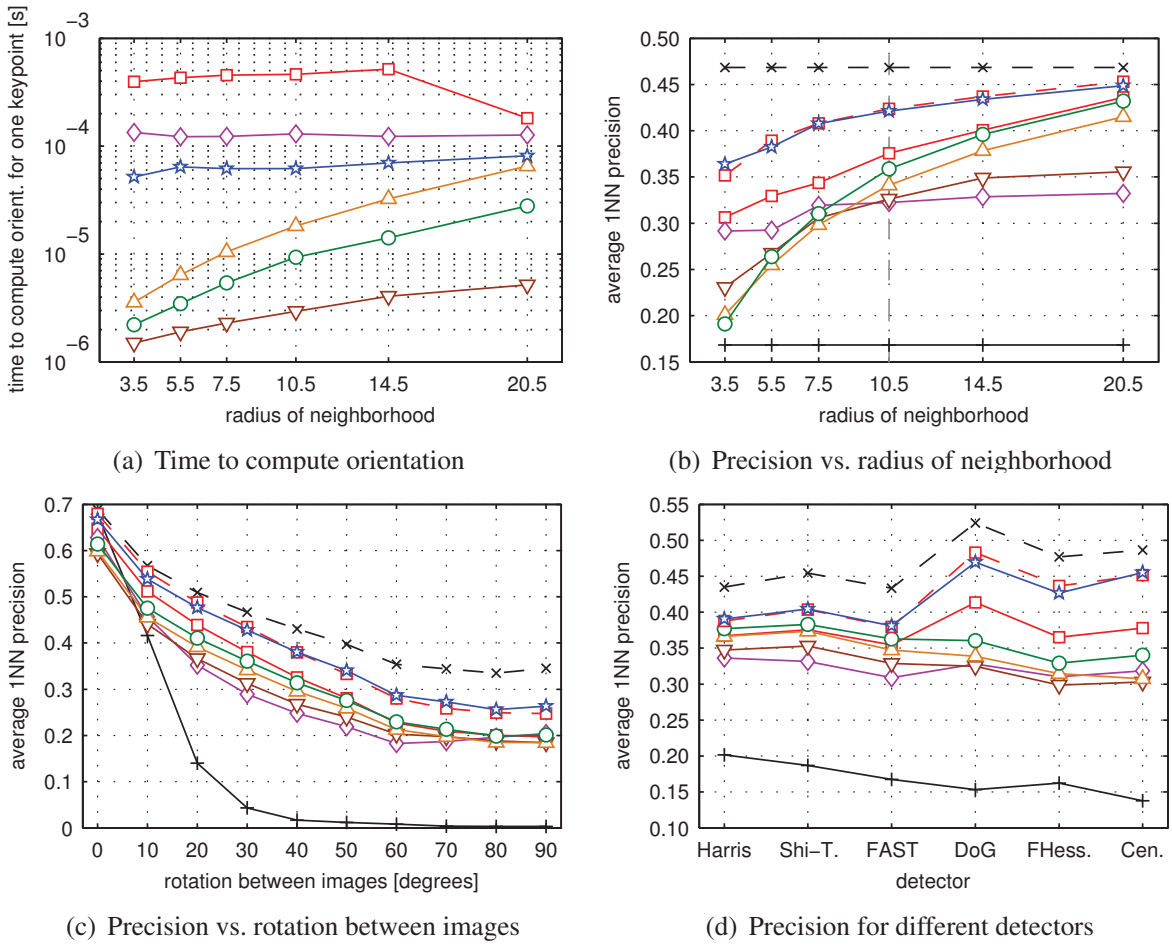
The multiple-orientation algorithms SIFT and HoI perform significantly better than the single-orientation methods and very similar to each other, while HoI is significantly cheaper than SIFT (Fig. 6(a)). Among the single-orientation algorithms, SIFT's single orientation assignment performs best, however, it is also by far the most expensive algorithm (Fig. 6(a)). Our algorithm CoM is the second-fastest and, for mid-size to large neighborhoods, second-best algorithm.

In the subsequent figures, we use a medium-sized neighborhood (radius of 10.5) for all algorithms (indicated by the dashed gray line in Fig. 6(b)).

Evaluating the performance by image rotation and by keypoint detector. Fig. 6(c) shows the performance broken down by the rotation between the two images in each pair (according to ground truth). Since the video sequences are recordings of manual motion (cf. description in [4]), the shown decline is to be expected: a frame's angle of rotation is directly correlated with its position in the sequence, hence image pairs with larger rotations between them are more strongly affected by changes due to lighting effects, motion blur and deviations from the perfect (rotation only) camera path. (Note that the performance with the orientation "oracle" also decreases significantly.) However, adding any of the orientation assignment algorithms significantly improves the precision over the no-orientation condition for more than 10° of rotation.

Fig. 6(d) shows that the performance of the orientation assignment algorithms also depends on the keypoint detector. Taylor's method, the smoothed gradient, and CoM all work better with the corner detectors (Harris, Shi-Tomasi, FAST) than with the "blob" detectors (DoG, Fast Hessian and CenSurE). For the corner detectors, CoM performs best among the single-orientation methods. Also, the advantage of the multi-orientation methods is significantly smaller for the corner detectors.

[1] For SIFT and SURF, this can be done via manipulating the keypoint's scale information, even if the algorithm is provided only as compiled library (as it is the case for SURF here).



(a) Time to compute orientation

(b) Precision vs. radius of neighborhood

(c) Precision vs. rotation between images

(d) Precision for different detectors

Figure 6: (a) Time to compute orientation for a single keypoint, *not* including the per-image precomputation steps needed for SIFT and SURF (building the scale space pyramid and the integral image, respectively), since this cost is per image rather than per keypoint. (b-d) Average precision of first nearest neighbor (1NN) when matching an image patch rotated according to the assigned orientation: (b) as function of the size of the neighborhood R (average for all detectors); (c) broken down by actual rotation between the images (radius 10.5); (d) for different detectors (radius 10.5).

6 Discussion & Implications

Size of the neighborhood. Judging from Fig. 6(b), the bigger the neighborhood R the better. Unfortunately, increasing the size of R not only increases the computation time for most algorithms (cf. Fig. 6(a)), but more importantly, enlarges the region which is assumed to be planar and rigid. Since this assumption is true for the testbed we used, Fig. 6(b) shows such a clear trend. Generally, however, it is desirable to confine this assumption to a small area. (Having stated this, we suggest that R might often be chosen too small, and some applications could be made more robust by increasing it: since local feature descriptors rely on the same assumption and highly non-planar feature points result in mismatches either way, it does not make sense to make R too small.) Since this assumption is the same for all algorithms, it is important to compare them along vertical "slices" of Fig. 6(b).

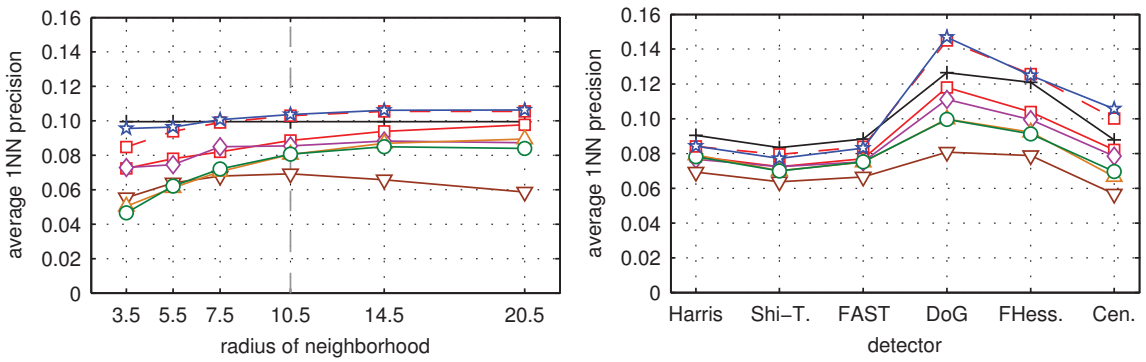


Figure 7: "Unconstrained" motion: Precision as function of the size of the neighborhood R (left), and broken down by for different detectors (right; radius 10.5). Legend as in Fig. 6.

Performance of SURF. In Fig. 6(b), it is notable that the performance of SURF stays fairly constant beyond a radius of 7.5 pixels and thus falls considerably behind the other algorithms despite very good performance for small neighborhoods. One possible explanation is that SURF uses a constant number of samples from within the neighborhood R (only sampling step and filter size depend on the size of R, but not the number of samples (illustrated as vectors in Fig. 1(a))). This works well for small neighborhoods, but seems to fail to capture all information available in larger neighborhoods. We suggest that the orientation assignment of SURF could be substantially improved for large R by increasing the number of samples.

Evidence that orientation assignments can be counterproductive. Fig. 7 shows results for the same experimental setup as Figs. 6(b) and 6(d), but for a different set of video sequences: Here, we use the "unconstrained" paths of the dataset, which feature unconstrained "moving around" the object. Due to perspective distortions and heavy motion blur, matching of random frames within these video sequences is very challenging, and thus the average precision is rather low. Of special interest, however, is the fact that the precision *without* orientation assignment is better than with many of the orientation assignment algorithms; only SIFT with multiple orientations and HoI can outperform it slightly and only for large neighborhoods. These sequences contain in-plane rotation only infrequently, so that being invariant to it turns out to be a disadvantage (since the original orientation within the source image is lost). This confirms the results of Gauglitz et al. [4] on the same dataset.

7 Conclusions

In this paper, we presented an extensive evaluation of orientation assignment algorithms and proposed two novel methods.

The first method, CoM, is very easy to implement, fast and effective. We suggest that CoM is a worthwhile alternative especially if (a) computation time is critical; (b) a single-orientation solution is preferred (due to algorithm design or to keep the size of the database small), and/or (c) a corner detector is used — here, the benefit of using multiple orientations was shown to be very small and CoM performed best among all single-orientation methods.

Our second algorithm, HoI, is capable of extracting multiple orientations and performs very similar to SIFT's orientation assignment (after the latter has been optimized for the metric used) while being significantly cheaper to compute. However, SIFT achieves this perfor-

mance with fewer orientations per keypoint on average. Therefore, HoI might be preferable for applications in which feature extraction time is critical, but the size of the database is limited (as in the case of tracking-by-detection or initialization of tracking of a known object), while SIFT would be preferable for image retrieval from a large database, in which the database retrieval is the bottleneck. In general, the ability to create multiple orientations significantly increases robustness.

Furthermore, we discussed implications about the problem in general as well as for existing algorithms: We analyzed the performance/number of orientation trade-off for SIFT, improved Taylor and Drummond's very fast method [13] by using larger rings, and we suggest that the orientation assignment of SURF [2] could be improved by using more samples.

References

[1] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. CenSurE: Center surround extremas for realtime feature detection and matching. In *Proceedings of the European Conference on Computer Vision*, volume 5305, pages 102–115, 2008. doi: 10.1007/978-3-540-88693-8_8.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110:346–359, June 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014.

[3] Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, volume 1, pages 510–517, 2005. doi: 10.1109/CVPR.2005.235.

[4] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *Intl. Journal of Computer Vision*, 2011. doi: 10.1007/s11263-011-0431-5.

[5] Jens Grubert (ICG TU Graz). Personal communication.

[6] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the 4th ALVEY Vision Conference*, pages 147–151, 1988.

[7] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Intl. Journal of Computer Vision*, 60(2):91–110, November 2004.

[8] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Proceedings of the IEEE European Conference on Computer Vision*, volume 1, pages 430–443, May 2006. doi: 10.1007/11744023_34.

[9] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.

[10] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994. doi: 10.1109/CVPR.1994.323794.

[11] Gabriel Takacs, Vijay Chandrasekhar, Sam S. Tsai, David M. Chen, Radek Grzeszczuk, and Bernd Girod. Unified real-time tracking and recognition with rotation-invariant fast features. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 934–941, 2010. doi: 10.1109/CVPR.2010.5540116.

[12] Simon Taylor and Tom Drummond. Multiple target localisation at over 100 fps. In *Proceedings of the British Machine Vision Conference*, September 2009.

[13] Simon Taylor and Tom Drummond. Binary histogrammed intensity patches for efficient and robust matching. *Intl. Journal of Computer Vision*, 2011. doi: 10.1007/s11263-011-0430-6.

[14] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.