19th Computer Vision Winter Workshop
Zuzana Kúkelová and Jan Heller (eds.)
Křtiny, Czech Republic, February 3–5, 2014

A Few Things One Should Know About Feature Extraction, Description and Matching

Karel Lenc, Jiří Matas, and Dmytro Mishkin

CMP CTU in Prague, Czech Republic

Abstract *We explore the computational bottlenecks of the affine feature extraction process and show how this process can be speeded up by 2-3 times with no or very modest loss of performance. With our improvements the speed of the Hessian-Affine and MSER detector is comparable with similarity-invariant SURF and DoG-SIFT detectors.*

The improvements presented include a faster anisotropic patch extraction algorithm which does not depend on the feature scale, a speed up of a feature dominant orientation estimation and SIFT descriptor computation using a look-up table.

In the second part of the paper we explore performance of the recently proposed first geometrically inconsistent nearest neighbour criterion and domination orientation generation process.

1 Introduction

Extraction of local image features is an important part of a wide variety of computer vision algorithms and significant effort has been put into speeding up this process. Speed up efforts have been usually directed at similarity covariant feature detectors or even only translation invariant detectors. Affine covariant detectors have been avoided because they are considered, the paper indicates somewhat unfairly, too computationally expensive (see Table 1).

Method	SIFT	SURF	HesAff	MSER	**HesAff+**	**MSER+**
Avg. NFeats	1719.23	2192.33	3181.81	1028.02	3787.60	1348.71
Avg. Time [s]	0.90	0.72	5.38	2.83	**1.80**	**0.69**

Table 1: Average processing time (without image IO) and number of features per an image of commonly used feature extractors and their variants implementing the proposed improvements. We compare similarity invariant SIFT (VlFeat implementation) and SURF (OpenSURF implementation) and standard implementations of affine invariant Hessian Affine (HesAff) (implementation by [18]) and MSER [11] with the improved affine invariant HesAff+ and MSER+. Values are computed on all images from Mikolajczyk's dataset [15] with average image resolution of 0.7MPx.

In the first part of this work, we show that with careful implementation the processing time of the traditional affine covariant detectors (such as Hessian-Affine and MSER) is comparable to similarity covariant detectors. We show that by loosening the demands on correctness of the patch extraction process from the signal processing point of view, the processing time can be significantly reduced with only a modest loss in the scale invariance of the extracted descriptors.

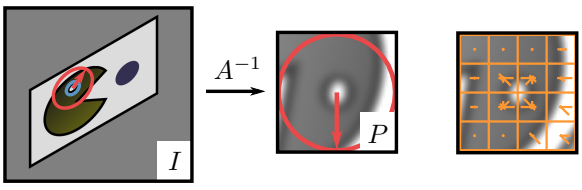


Figure 1: The classical local image feature extraction pipeline. Measurement region (red) of a detected feature (blue) is warped from image I to a patch P normalising the region based on the local affine shape of the feature described by matrix A. The image data in the patch are used to compute the SIFT descriptor.

In the second part, we investigate another bottleneck which is the conversion of gradients from cartesian to polar coordinates and we examine existing and propose a new $\arctan 2$ approximation which speed up this process.

Finally we examine strategies for generating tentative correspondences. The traditional method for SIFT matching is based on the second (to first) nearest neighbour (SNN) distance ratio. We confirm that using the first geometric inconsistent nearest neighbour [16] can improve the performance even on two view matching without view synthesis. We investigate the tentative correspondences between distinguished regions (DRs) and number of their matched dominant orientations. We show that the standard second nearest neighbour criterion is able to remove most of the inconsistent correspondences (e.g. two dominant orientations of a single DR matched to two different DRs in the tested image) and that the knowledge of the multiple orientation correspondences can help to avoid some degenerate hypotheses in RANSAC.

2 Related Work

The feature extraction process consists of several stages that are visualised in Figure 1. In the first step, distinguished regions (DRs) are detected using a feature detector. There are several ways how to perform that: Scale-space detectors (DoG [10], Hessian or Harris-Affine [13]) start with building a scale space pyramid and each layer of the pyramid is anal-

ysed for local maxima of some differential operator; in the Fast Hessian, which is how we call the detector part of the SURF system [2], the pyramid consists of integral images, which are used for computing an approximated Hessian operator with Haar wavelets. In the case of Hessian and Harris Affine detector, the structure tensor is used to estimate an affine shape of each DR. Another affine invariant detector, MSER [11], detects DRs finding maximally stable extremal regions where the affine shape is defined by their interior's second moments.

In order to obtain rotationally invariant descriptions, dominant orientations of each DR are detected. This is usually done by collecting a weighted histogram of finite derivatives in the feature's measurement region, which has m-times bigger measurement scale. This measurement region is used for descriptor computation and is then normalised to a small patch (usually of size 41 pixels) used to form invariant descriptor such as SIFT [10].

The process differs for the SURF descriptor, which uses Haar wavelets for the computation of both the dominant orientations and for the SURF descriptor; therefore, no patch needs to be extracted. The main advantage of the SURF detector is its speed, but like the DoG detector, it does not offer Affine invariance. However, this can be achieved by using ASIFT [17] or MODS [16] matching startegy which synthesise views so that images can be matched across significant viewpoint changes without a substantial increase in the processing time.

Another speed-aware approach to feature detection is the FAST detector [20], which is a classifier learned for corner detection. However, it examines a neighbourhood of constant size; therefore, it is not scale invariant. This restriction was addressed in BRISK detector [9].

The computationally expensive part of estimating dominant orientations or computing SIFT descriptor is the conversion of gradients from Cartesian to Polar coordinates. This operation can be accelerated in several ways, e.g. in [8], a circuit design is proposed to accelerate in hardware this operation. The $\arctan 2$ function can be approximated well by Taylor series expansions, which is often used in speed-aware implementations [19], [12]. Indeed, this is used in practice e.g. in *VLFeat* library [21], where the 3rd order Taylor polynomial is used to approximate the function $\arctan((1-r)/(1+r))$. Similarly, in *OpenCV* library[1], the $\arctan 2$ is approximated with a 7th degree Taylor series expansion. Moreover, it is implemented using SSE instructions.

The extracted local image features are usually used for image based matching, e.g. in wide baseline stereo problems. Feature extraction is followed by generation of tentative correspondences, usually using the Lowe's Second Nearest Neighbour criterion [10]. Then the tentative correspondences are verified against two-view geometry constraints in a RANSAC framework [5].

[1] `http://opencv.org/`

3 Speeding up patch extraction process

We start with investigation of the main bottlenecks of the feature extraction process. In order to compare existing feature extraction algorithms fairly, we measure time needed for processing hypothetical 1MPx image where a detector finds 2000 Distinguished regions and extract 3600 descriptors as each distinguished region has 1.8 dominant orientations on average. Because the processing time can also depend on the size and shape of the detected features, we compute an average over 16 images of various scenes.

Figure 2 shows the execution time of each stage of the VLFeat[2] DoG detector [21], OpenSURF[3] implementation of SURF detector [2], improved implementation of Hessian-Affine [13] by Perdoch et. al. [18] and MSER [11].

All detectors have been configured in such a way that they detect features within the same range of scales; the OpenSURF algorithm was used in two configurations: FHes+SURF where the initial sampling is set to default 2pxs (i.e. the response is computed for every second pixel) and FHes-1px+SURF where the sampling is set to 1px. All measurements in this article are done with measurement scale $m = 3\sqrt{3}$.

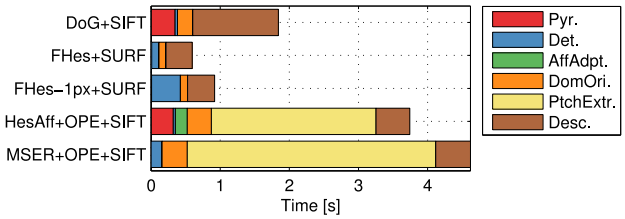


Figure 2: The processing time of feature extraction stages of commonly used algorithms (VLFeat DoG, OpenSURF, HessianAffine by Perdoch [18] and authors' implementation of MSER [11] which uses same SIFT implementation as the HessianAffine). The time was measured on a set of 16 1MPx images. *Pyr.* stands for the time needed for building a pyramid (the scale space or the integral images for the SURF detector), *Det.* is the duration of the feature detection stage. Both of these stages depend mainly on the image size. *AffAdapt.* is the duration of iterative affine adaptation and *DomOri.* is the time needed for detection of dominant orientations. *AffAdapt.* and and *DomOri.* are normalised to 2000 DRs. The last two stages, *PtchExtr.*, the time needed to extract patches used for the description (*Desc.*) are normalised to 3600 features.

Figure 2 clearly shows that the most expensive stage of the existing affine invariant feature detectors is the patch extraction process, note that this stage takes longer for MSER as it generally detects regions with higher scale [15] and uses the same patch extraction algorithm as Hessian-Affine, which means that the patch extraction time depends on the feature size.

In the following, we propose novel algorithms which reduce significantly the dependence of patch extraction time

[2] `http://www.vlfeat.org/`

[3] `http://www.chrisevansdev.com/computer-vision-opensurf.html`

on the local image feature scale. Then, several improvements which speed up the feature extraction process are presented.

3.1 The standard patch extraction process

In order to obtain invariant description of a co-variant local image feature, the image region corresponding to a feature is normalised. The local derivatives, used to obtain gradients for the SIFT descriptor, are computed using a Gaussian kernel of the size determined by differentiation scale σ_D [13] and the extracted patches should have $\sigma_D = \text{const}$ in order to gain full scale invariance.

In the case of the original Lowe's [10] SIFT feature detection and description framework, the patch used for feature description is extracted from the scale-space layer where the feature has been located. This means that for a feature which was found in octave o and in a layer l of a scale space with O octaves and L layers, the feature can have a scale in the input image in interval

$$s \in \left[\sigma_i 2^{o+\frac{l-1}{L}}, \sigma_i 2^{o+\frac{l+1}{L}}\right] \quad (1)$$

where σ_i is the initial scale (the prior smoothing used for building the scale-space pyramid) [10]. Then the descriptor is computed from a measurement region with scale $s_{mr} = s \cdot m$ in the original image where m is the magnification factor. This method can be used only for similarity-invariant features as it does not handle anisotropy of affine co-variant features.

In patch extraction implementations [13] and [18], at first, an affine image feature is normalised with $A^{-1} \cdot s$ from the input image where $A \in GL(2)$ is the de-normalisation matrix and $s = \sqrt{A}$ is the feature scale. In the next step, the extracted patch is blurred with an isotropic Gaussian kernel with variance $\sigma_B = 2\sigma_D ms/p$ in order to obtain differentiation scale σ_D in the down-sampled descriptor patch.

The disadvantage of the method ([13], [18]) is its computational complexity as it works with the original image data even for features which may cover the whole image. This can be seen in Figure 3 which clearly shows that even though there is only a fraction of detected features in higher octaves, the patch extraction process still takes a significant amount of time. All the computation times were measured on a machine with Intel® Core™i7-3517U CPU with 4MB cache.

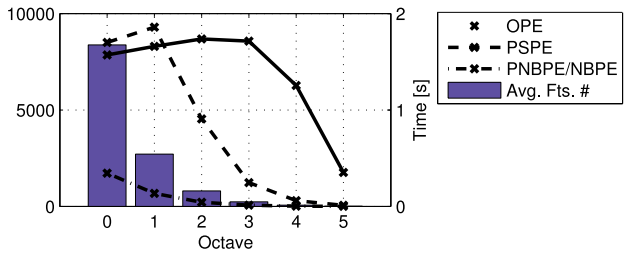


Figure 3: Time needed for extraction of patches found at different scale space octaves. OPE is the original patch extraction method [13], [18]. Results obtained with the Hessian Affine detector. Average on 18 various 3MPx images.

3.2 Speeding up the affine patch extraction process

We propose a novel algorithm for patch extraction which uses the pre-blurred layers of the isotropic Gaussian scale space pyramid. We will refer this algorithm as Pyramid-Smoothing Patch Extraction (PSPE). Based on the required σ_D it operates on the Gaussian scale space pyramid layer where the differentiation scale is small enough that after the extraction it would not exceed σ_D in any direction.

Then, the measurement region of the feature is extracted to the intermediate patch in its original scale in such a way that the eigenvectors of the affine transformation are aligned with the image axes. This transformation is found using the singular value decomposition. Then, equal differentiation scale in all directions can be achieved by a convolution with an anisotropic, separable Gaussian kernel, which has the same computational complexity as isotropic Gaussian blurring. Finally, the patch is downscaled and rotated to the final patch of size p. Details are given in Algorithm 1.

Algorithm 1 PSPE Pyramid-Smoothing Patch Extraction

Require: I – Input image; L_{o_i,s_j} – Gaussian Scale-Space pyramid with initial image scale σ_0, octaves $0 < o_i < O$ and octaves' layers $0 \leq s_j < S$; A – Local affine feature; σ_D – required diff. scale, p – a patch size, m - a measurement region multiplier.

Ensure: P – Extracted patch.

Get feature scale $s = \sqrt{A}$
Get patch to extr. feature scale $\rho = \frac{2\,ms}{p}$
Compute Singular Value Decomposition $A = U\ D\ V^T$,
Set $l_1 = D_{1,1}/s,\ \ l_2 = D_{2,2}/s$
Differentiation blur in l_2 direction in I is $\sigma_{l_2} = \sigma_D\ \rho\ l_1$
if $\sigma_{l_2} < \sigma_0$ **then**
  $\hat{\sigma} = 0.5,\ \ \hat{d} = 1$
  $\hat{P}(\mathbf{x}) = I(U\ \text{diag}(l_1, l_2)\ \mathbf{x})$
else
  $o = \lfloor \log_2(\sigma_{l_2}/\sigma_0) \rfloor,\ \ v = \lfloor \log_{2^{1/S}}(2^{-o}\sigma_{l_2}/\sigma_0) \rfloor$
  $\hat{\sigma} = \sigma_0\ 2^{o+v/S},\ \ \hat{d} = 2^o$
  $\hat{P}(\mathbf{x}) = L_{o,s}(2^{-o}\ U\ \text{diag}(l_1, l_2)\ \mathbf{x})$
end if
Ensure correct diff. scale in both x and y directions
$\hat{\sigma}_x = l_2\ \hat{\sigma}/\hat{d},\ \ \hat{\sigma}_y = l_1\ \hat{\sigma}/\hat{d},\ \ \hat{\sigma}_D = \sigma_D\ \rho/\hat{d}$
$\hat{\sigma}_{dx} = \sqrt{\hat{\sigma}_D^2 - \hat{\sigma}_x^2},\ \ \hat{\sigma}_{dy} = \sqrt{\hat{\sigma}_D^2 - \hat{\sigma}_y^2}$
Blur $\hat{P}_B(\mathbf{x}) = \hat{P}(\mathbf{x}) * g(\mathbf{x}, \Sigma)$, $\Sigma = \text{diag}(\hat{\sigma}_{dx}, \hat{\sigma}_{dy})$
where $g(\mathbf{x}, \Sigma)$ is 2D Gaussian filter
Sub-sample to patch: $P(\mathbf{x}) = \hat{P}_B(\rho\ V^T\ \mathbf{x})$

We propose a faster variant of the PSPE algorithm, PNBPE (Pyramid, No-Blur Patch Extraction), which ignores the anisotropic blurring step. This variant simply warps the selected pyramid layer to the patch without any intermediate steps. Unlike the PSPE, the pyramid layer is not selected according to the affine shape of the feature, but solely based on the feature scale. With this simplification, the PNBPE method gets several times faster than the former PSPE variant. Computation time of this method depends only on the SIFT patch size (usually $p = 41$). Details

Algorithm 2 PNBPE Pyramid, No-Blur Patch Extraction

Require: I – Input image; L_{o_i,s_j} – Gaussian Scale-Space pyramid with initial image scale σ_0, octaves $0 < o_i < O$ and octaves' layers $0 \leq s_j < S$; A – Local affine feature; σ_D – required diff. scale, p – a patch size, m - a measurement region multiplier.

Ensure: P – Extracted patch.

Get feature scale $s = \sqrt{A}$
Get patch to extr. feature scale $\rho = \frac{2\,ms}{p}$
Differentiation blur in I is $\sigma = \sigma_D\,\rho$
if $\sigma < \sigma_0$ **then**
  $P(\mathbf{x}) = I(\rho\,A\,\mathbf{x})$
else
  $o = \lfloor \log_2(\sigma/\sigma_0) \rfloor,\ \ v = \lfloor \log_{2^{1/S}}(2^{-o}\sigma/\sigma_0) \rfloor$
  $P(\mathbf{x}) = L_{o,s}(2^{-o}\,\rho\,A\,\mathbf{x})$
end if

of this variant are given in Algorithm 2.

We also test a method, called NBPE (No-Blur Patch Extraction), which simply warps the feature measurement region to the patch using the *original image*, not the pyramid. The speed of NBPE and PNBPE is equal.

With PSPE, PNBPE and NBPE it generally holds that the time needed for patch extraction is proportional to the number of the features and does not depend on feature size, as can be seen in Figure 3. The independence of patch extraction time on feature size is demonstrated in Figure 4.

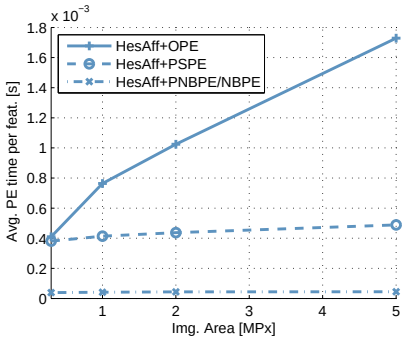


Figure 4: Average time needed for extraction of a single patch. The values are computed as an average over all features from 18 different images of a given resolution and is computed with HessianAffine detector.

3.3 Matching performance of patch extraction variants

In Figure 7, the comparisons of matching scores of different patch extraction methods for the Hessian-Affine and MSER detector are shown. The matching score is measured as defined by [15], with a difference that one-to-one correspondences are computed for descriptors only. In the original implementation of the matching score benchmark, match is deemed correct when it is a one-to-one match both based on descriptor distances and in ellipse overlaps. Though, this is not usable for DRs with multiple orientations as the ellipse overlap does not take into consideration the dominant orientations. We use matching score instead of [14] as we want to measure performance under various geometric transformations. Matching score has been computed with RootSIFT [1] descriptor normalisation.

Tests have been performed on datasets from [15] or their variants from [4] with more precise ground truth where available. The GRAF and WALL dataset test invariance to viewpoint change, BOAT and BARK to zoom and rotation and BIKES are used to test invariance to image blur. In order to show the invariance to scale changes we have measured average matching score using 32 images which have been resized to scales in $\langle 1, 0.2 \rangle$ (SYNTH. SCALE). Similarly, to simulate invariance to anisotropic deformations, we have generated datasets of the same images but scaled only in the y-axis direction (SYNTH. ANIS. SCALE).

It can be observed that PSPE method obtains in general the same performance as the original patch extraction method. The PNBPE and NBPE methods have similar performance when the features which need to be matched are relatively small, though they get worse performance on the Bark and Bikes dataset and with MSER detector which detects bigger features. This is additionally confirmed with the tests on synthetic images. On synthetic scale dataset, it can be seen that the PNBPE method has slightly better scale invariance as the NBPE method is ignoring Nyquist–Shannon sampling theorem and in case of bigger scale changes, the aliasing becomes an issue.

The reason why PNBPE and NBPE has got the same number of matches is that these methods detects more dominant orientations (1.9 for PNBPE and 2 for NBPE) as the extracted patch contains higher frequencies. But those orientations are less stable, thus these patch extraction variants have worse matching score.

4 Speeding up the SIFT

HesAff+OPE+SIFT	HesAff+NBPE+SIFT
Convolution (30%)	Interpolation (19%)
Interpolation (20%)	SIFT sampling (16%)
SIFT sampling (10%)	arctan2 (14%)
arctan2 (8%)	Gradient comp. (10%)

Table 2: The most time consuming functions (self-cost, percentage of the whole program runtime) by profiling Wall-1 [15] feature extraction.

As a significant bottleneck of the feature extraction process is the arctan2 function (see Table 2), we have investigated the precision and speed of existing implementations in *VLFeat* and *OpenCV* and proposed a new algorithm which outperforms these approximations in speed.

We have created a method which approximates the arctan2 function using look-up table (LUT). This method divides the interval $\langle 0, 2\pi \rangle$ into octants, and is using a LUT of 256 bins accordingly for each octant as $\arctan(x/y) = \pi/2 - \arctan(y/x)$, if $x > 0$ and similar rule is for $x < 0$.

The comparison of different methods in single precision floating point numbers is given in Table 3 where the error

Impl.	RMS Err. [rad$\times 10^{-3}$]	Max Err. [rad$\times 10^{-3}$]	Avg. time [ns]
ARCTAN2	0	0	139.1
VLFEAT	4.278	6.136	95.3
OPENCV	0.073	0.167	99.5
LUT-256	1.815	3.922	89.9

Table 3: Speed of different arctan2 implementations and approximations in single floating point precision. Values are computed over 2×10^6 measurements and the error is compared against the arctan2 , standard reference.

is compared against the standard implementation[4]. It can be seen that our LUT-256 method outperform the investigated methods in speed and has smaller error than the method used in VLFeat library. The error does not have any influence on descriptor performance. However, as the speed of LUT approximation depends mostly on memory access speed, for processors with smaller CPU cache it may be needed to reduce the number of bins, partially scarifying the precision.

Method	DomOri [μs] (Speed-up)	SIFT [μs] (Speed-Up)
Standard	110.35	159.26
LUT-256	47.12 (2.3)	95.06 (1.7)
SSE-OpenCV	38.75 (2.8)	73.38 (2.2)

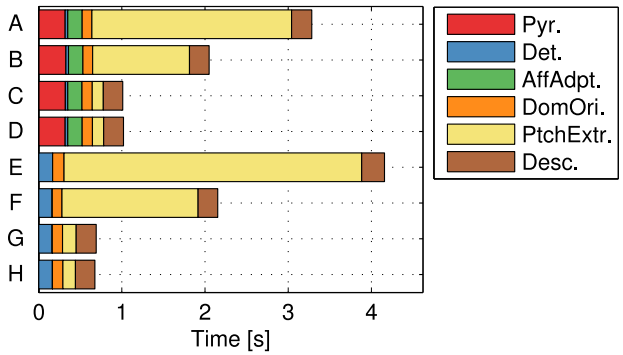
Table 4: The average speed-up per a single MSER feature using the arctan2 approximation and SSE instructions in different patch extraction stages.

However the OPENCV approximation is implemented using SSE instructions, and in addition it uses SSE for gradient magnitude computation where the SSE instruction for the square root (SQRTPS) is already an approximation with rel.err. $\leq 1.5 \times 10^{-12}$ [6] which is more than sufficient for the feature extraction purposes.

The overall speed-up of the feature extraction stages in comparison to the original implementation is shown in Table 4. It can be seen that even speeding up the arctan2 function with a look-up table can bring a significant improvements in the processing time. The advantage of the LUT-256 is that it has tunable precision by varying the number of bins which can be chosen in such a way that it would fit to the CPU cache of the target architecture. However, in the following experiments we use the SSE-OpenCV variant.

The processing time of Hessian-Affine and MSER detectors using the proposed improvements are shown in Figure 5. It can be seen that with the PSPE or NBPE algorithm together with the SSE-OpenCV (referred as SIFT+), the feature extraction process takes around half the time of the original implementation. In Table 5 we show the average processing time per an image from the Mikolajczyk's dataset [15] without normalisation to a constant number of features, i.e. the expected time needed to extract features from a single image. From this table it is clear that for example using the improved MSER+NBPE+SIFT+ for feature extraction is similarly time consuming as using SURF algorithm.

[4]Defined by IEEE Std 1003.1, particularly used GNU C Library 2.15 implementation



Variant	Detector	Patch Extr.	Descriptor
A	HesAff	OPE	SIFT+
B	HesAff	PSPE	SIFT+
C	HesAff	PNBPE	SIFT+
D	HesAff	NBPE	SIFT+
E	MSER	OPE	SIFT+
F	MSER	PSPE	SIFT+
G	MSER	PNBPE	SIFT+
H	MSER	NBPE	SIFT+

Figure 5: Processing time of particular feature extraction stages using the proposed improvements. In all feature extractors here, the SSE-OpenCV method is used for computing arctan2 . The values in the graph are measured in the same way as in Figure 2.

Method	Avg. NFeats	Avg. Time
DoG+SIFT	1719.23	0.90
FHes+SURF	2192.33	0.72
HesAff+OPE+SIFT	3181.81	5.38
HesAff+PSPE+SIFT+	3201.98	3.56
HesAff+PNBPE+SIFT+	3714.40	1.78
HesAff+NBPE+SIFT+	3787.60	1.80
MSER+OPE+SIFT	1028.02	2.83
MSER+PSPE+SIFT+	1035.04	2.15
MSER+PNBPE+SIFT+	1266.79	0.69
MSER+NBPE+SIFT+	1348.71	0.69

Table 5: Average processing time and number of features per an image (without image IO) of commonly used extractors and feature extractors with the proposed speed improvements on all images from Mikolajczyk's dataset [15]. The values are computed in the same way as in Table 1.

5 Matching features in multiple-orientation context

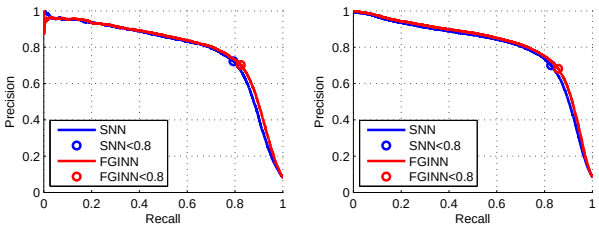
The output of a detection algorithm on input image I is a set of distinguished regions (DR). Afterwards, dominant orientations $\mathcal{A}$ for each region are detected in order to obtain rotation invariance, which creates several local affine features for each DR. Usually, the number of dominant orientations is limited to $1 < |\mathcal{A}_u^I| \leq 4$ and for each dominant orientation, one descriptor is extracted.

In image matching task, the fact that a single DR generates more descriptors is usually ignored and it simply matches all descriptors from a reference image to the matched image. This has several consequences, e.g. for the SNN ratio (SNNr) criterion. This criterion is used to filter correspondences $\mathcal{C}$ of the reference image descriptors to the matched image descriptors and generates a set of tentative correspondences $\mathrm{TC} \subset \mathrm{C}$. It computes the distance ratio

of the first closest to the second closest reference image descriptor and when this ratio is higher than 0.8, the correspondence is rejected (i.e. that the correspondence is not distinguishable enough). On the other hand, in case of symmetric DRs, the second nearest neighbour can be located on the same DR in the matched image as the first nearest neighbour. This issue was tackled by [16] where the SNN criterion has been revisited. The authors added a new condition that the SNN ratio is not computed with the second closest descriptor but with the *First Geometrically Inconsistent Nearest Neighbour* (FGI-NN). Two descriptors are geometrically inconsistent when their centres of gravity are farther than a given threshold (in our case set to $10px$).

In [16], it was used in context of synthesised views, but we have observed that it has some influence on simple two view matching as well. In our tests we have matched various 85 image pairs (image pairs introduced by [7], [15] and 65 pairs selected from clusters generated by [3]) and we estimated a homography between them using LO-RANSAC algorithm [7]. Descriptors are extracted using the Hessian-Affine+OPE+SIFT. Inliers $\mathrm{I} \subset \mathrm{TC}$ are TC with symmetric reprojection error [5] smaller than $4px$, and similarly, valid correspondences $\mathcal{VC} \subset \mathrm{C}$ are all geometrically correct correspondences. We compute precision and recall as:

$$\text{precision} = \frac{|\mathcal{I}|}{|\mathcal{TC}|} \quad \text{recall} = \frac{|\mathcal{I}|}{|\mathcal{VC}|} \tag{2}$$



(a) MSER (b) HessianAffine

Figure 6: Precision and recall of SNN and FGINN for MSER and HessianAffine with QRT-SIFT. Results for 85 image pairs.

Method	AP	Prec [%]	Rec [%]	$\lvert\mathcal{TC}\rvert$	$\lvert\mathcal{I}\rvert$
MSER SNN	78.42	72.25	79.38	9408	6797
MSER FGI-NN	79.60	70.22	82.43	10039	6940
HesAff SNN	80.12	70.04	82.74	29817	20883
HesAff FGI-NN	81.63	68.12	85.73	31919	21397

Table 6: Precision and recall for $\mathrm{SNNr} < 0.8$ with average precision (area under the PR curve) for methods for generating tentative correspondences and two selected detectors which use SQRT-SIFT as descriptor. Results for 85 image pairs

We have measured that using the FGI-NN method improves the performance of the system even in the case of two-view matching without view synthesis. The particular values of precision and recall for $\mathrm{SNNr} < 0.8$ with average precision are shown in Table 6. Though it slightly lowers the precision, it increases the recall and is able to obtain more inliers which are important for the accuracy. The FGI-NN criterion also increases the average precision means that FGI-NN is a better classifier of tentative correspondences than SNN without being dependent on the particular SNNr threshold. The correspondences missed by the SNN method are usually caused by symmetric features where the SNN may be on the same DR but with a different orientation. The precision-recall curve, computed varying the SNNr threshold, is shown in Figure 6.

# dom.orientations $\lvert\mathcal{A}\rvert$	1	2	3	4
% of DRs	44.97	43.09	10.72	1.22

Table 7: Percentage of detected distinguished regions with a certain number of dominant orientations. Values are computed out of 1.8×10^5 DRs.

In the next experiment we investigate how dominant orientations of the DRs are matched across the images. At first, in Table 7, we show the distribution of number of dominant orientations per a DR detected in the reference images. On average, each DR is assigned 1.8 dominant orientations.

Dominant orientations $\lvert\mathcal{A}\rvert$				
Matched DRs	Matched $\lvert\mathcal{A}_\mathcal{D}\rvert$			
	1	2	3	4
1	62.49%	30.70%	4.63%	0.38%
2	0.00%	1.37%	0.39%	0.03%

Table 8: Distribution of the DRs in tentative correspondences $\mathcal{TC}$ according to their number of matched Dominant orientations (column) and number of matched *unique* DRs from the tested image (row). E.g. a DR which is in the second row and third column has three dominant orientations in $\mathcal{TC}$ where two are matched to the same DR in the tested image. $\mathcal{TC}$ are generated using the FGI-NN criterion.

There is clearly a lot of DRs which have more than one descriptor. But what happens when correspondences are generated? In Table 8 we show the distribution of the DRs in $\mathcal{TC}$ according to the number of their dominant orientations in $\mathcal{TC}$ and number of matched DRs in the second image. It can be seen that for many DRs with multiple dominant orientations, only some of them passed the FGI-NN criterion.

More than 30% of the DRs have 2 dominant orientations where both of them are matched against a single DR in the tested image (row 1 column 2). Those 30% of DRs are actually generating more than 42% of correspondences which are passed to RANSAC algorithm. This also means that in our dataset, more than 22% of the correspondences are duplicates.

This can be exploited by improving the speed of RANSAC algorithm by passing less tentative correspondences as sampling two correspondences of same image regions leads to a degenerate solution. This issue is handled using "duplicate filtering" procedure in [17], [16]. However, if the double correspondence is found as inlier, it may be counted twice as the fact that two dominant orientation have been matched may bear a prior of the correspondence quality. Though, we have not investigated those issues.

From Table 8 it can be observed that most of the incoherent correspondences, i.e. ref. image DRs matched to different DRs in the matched image, does not pass to the list of $\mathcal{TC}$, thus the Lowe's (FGI)SNN ratio works well for the multiple orientation matching by itself.

6 Conclusions

It has been shown shown that a careful implementation of existing affine invariant feature detectors has a speed comparable to existing similarity covariant detectors. Furthermore, using a simplified patch extraction method the speed of the affine feature extraction becomes comparable to their approximations, such as SURF. The speed leads to slight decrease in the scale invariance of the extracted patches.

The process of patch description is made faster by a simple approximation of the arctan2 function. We have created a simple approximations of arctan2 which uses a look-up table and the terms of speed outperforms the approximations used in different computer vision libraries.

We have investigated the Lowe's second nearest neighbour criterion in the context of multiple orientation matches. We confirmed that using the first geometrically inconsistent nearest neighbour increases the number of inliers as it allows to match symmetric features. Furthermore, we have investigated the way how the second nearest neighbour works in case of multiple orientations and proposed some improvements which can speed up RANSAC.

Acknowledgement

The authors were supported by EC project FP7-ICT-270138 DARWIN by the Technology Agency of the Czech Republic program TE01020415 (V3C – Visual Computing Competence Center).

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *IEEE Computer Vision and Pattern Recognition*, 2012.

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[3] Ondrej Chum and Jirí Matas. Large-scale discovery of spatially related images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(2):371–377, 2010.

[4] Kai Cordes, Bodo Rosenhahn, and Jörn Ostermann. Increasing the accuracy of feature evaluation benchmarks using differential evolution. In *IEEE Symposium on Differential Evolution*, pages 1–8. IEEE, 2011.

[5] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge University Press, 2000.

[6] Intel Corporation. *Intel® 64 and IA-32 Architectures Software Developer's Manual*, volume 2. June 2013.

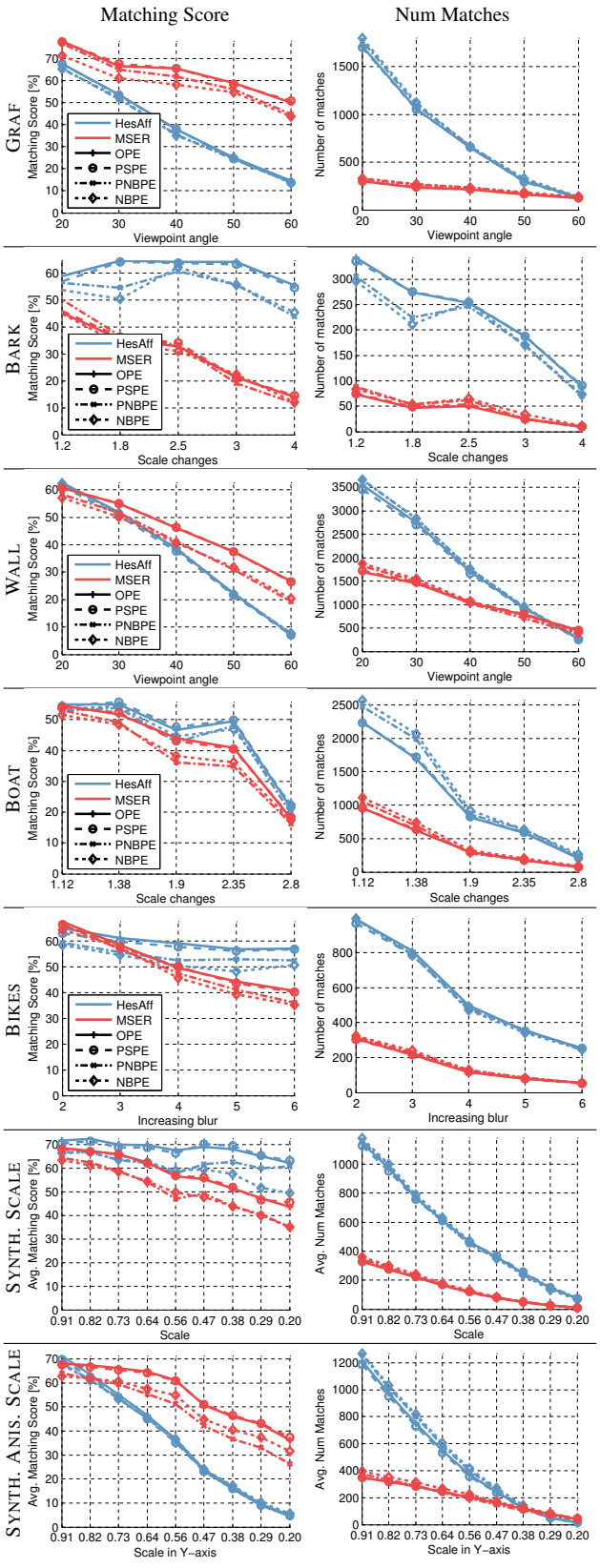


Figure 7: Comparison of the performance of different patch extraction algorithms using Mikolajczyk's matching score protocol [15]. Measurements on synthetic datasets are computed as an average over generated image pairs from 32 various images. Line colour distinguishes different detectors and line style signifies the patch extraction algorithm.

[7] Karel Lebeda, Jiri Matas, and Ondrej Chum. Fixing the locally optimized RANSAC. In *British Machine Vision Conference*, 2012.

[8] Sung-Won Lee, Ki-Seok Kwon, and In-Cheol Park. Pipelined cartesian-to-polar coordinate conversion based on srt division. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 54(8):680–684, 2007.

[9] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. BRISK: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555. IEEE, 2011.

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

[11] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *British Machine Vision Conference*, pages 384–393, 2002.

[12] Herbert A Medina. A sequence of polynomials for approximating arctangent. *The American Mathematical Monthly*, 113(2):156–161, 2006.

[13] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision*, pages 128–142, 2002.

[14] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[15] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

[16] Dmytro Mishkin, Michal Perdoch, and Jiri Matas. Two-view matching with view synthesis revisited. In *Proceedings of the 28th Conference on Image and Vision Computing New Zealand*, 2013.

[17] Jean-Michel Morel and Guoshen Yu. Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, 2(2):438–469, 2009.

[18] M. Perďoch, O. Chum, and J. Matas. Efficient representation of local geometry for large scale object retrieval. In *IEEE Computer Vision and Pattern Recognition*, pages 9–16. IEEE, 2009.

[19] Sreeraman Rajan, Sichun Wang, Robert Inkol, and Alain Joyal. Efficient approximations for the arctangent function. *Signal Processing Magazine, IEEE*, 23(3):108–111, 2006.

[20] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

[21] Andrea Vedaldi and Brian Fulkerson. VLFeat: an open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469–1472, 2010.