Affine Adaptation of Local Image Features using the Hessian Matrix

Ruan Lakemond, Clinton Fookes, Sridha Sridharan
Image and Video Laboratory, Queensland University of Technology
GPO Box 2434, 2 George St, Brisbane, Queensland 4001
{r.lakemond, c.fookes, s.sridharan}@qut.edu.au

Abstract

Local feature detectors that make use of derivative based saliency functions to locate points of interest typically require adaptation processes after initial detection in order to achieve scale and affine covariance. Affine adaptation methods have previously been proposed that make use of the second moment matrix to iteratively estimate the affine shape of local image regions. This paper shows that it is possible to use the Hessian matrix to estimate local affine shape in a similar fashion to the second moment matrix. The Hessian matrix requires significantly less computation effort to compute than the second moment matrix, allowing more efficient affine adaptation. It may also be more convenient to use the Hessian matrix, for example, when the Determinant of Hessian detector is used. Experimental evaluation shows that the Hessian matrix is very effective in increasing the efficiency of blob detectors such as the Determinant of Hessian detector, but less effective in combination with the Harris corner detector.

1. Introduction

Local image features are patterns in an image that are distinguishable from the surrounding image in some way. Invariant local feature detectors attempt to consistently find features in an image and localize them accurately, despite significant changes in viewing conditions. These features are then matched across multiple views and the matches used for tasks such as computing scene geometry, object recognition and many others. A comprehensive review of local image feature detectors was recently published in [1].

This paper is concerned with a class of feature detector that computes a saliency map of the image and locates the extrema of the saliency map. The saliency map shows the regions of the image that have high information content or high curvature and is a function of image partial derivatives. Examples include detectors based on the Harris detector [2], Determinant of Hessian detector [3], Laplacian of Gaussian and Difference of Gaussians detector [4]. The saliency map based feature detectors are inherently sensitive to changes in scale and projective deformations and do not produce features that are covariant under these changes directly. Additional steps are required to adapt the features so that they are covariant under various transformations. Evaluations of a number of feature detectors presented in [5, 6] showed that the most successful method for adapting Harris and Determinant of Hessian features is to use the affine adaptation algorithm given in [7] in combination with the scale selection algorithm given in [8].

Although other detectors are available that require significantly less time to process each image, Harris and Hessian based affine detectors are still a superior choice for difficult wide base-line scenarios such as those found in automatic calibration of surveillance cameras, for example. Fast detectors such as SURF [9] and FAST [10] do not generate affine covariant features and the popular MSER [11] detector may not produce sufficient numbers of correspondences if the scene does not contain many planar features [5, 6]. Improving the speed of Harris and Hessian based detectors would therefore be a useful improvement.

To date, the second moment matrix has been the dominant method for estimating the affine shape of local image regions detected using the saliency map approach. The purpose of this paper is to show that the Hessian matrix may also be used for estimating local affine shape in certain circumstances and to compare the performance of the Hessian matrix with the commonly used second moment matrix. The Hessian matrix is simpler to compute than the second moment matrix and may enable a reduction in computation time. The Hessian matrix may also be more convenient to compute, for example, when using the determinant of Hessian detector. It is shown through experimentation that using the Hessian matrix allows for substantial gains in computational efficiency over the second moment matrix.

2. Previous Work

Saliency map based feature detectors are sensitive to changes in scale and viewing angle. This problem has been addressed by applying scale and affine adaptation to the features. An affine transform consists of rotation, translation, scaling and sheering. Discussion in this paper will focus on adapting the sheering component of the affine transform, while ignoring scale, rotation and translation, as these are handled separately in most practical algorithms (see [7, 5] for examples).

Existing affine adaptation methods are based around using the second moment matrix to iteratively estimate local affine shape. The scale normalized second moment matrix $\mu(\mathbf{x}, \sigma_D, \sigma_I)$ is defined as,

$$\mu(\mathbf{x}, \sigma_D, \sigma_I) = \sigma_D^2 g(\mathbf{x}, \sigma_I) * \mathbf{D}, \quad (1)$$

$$g(\mathbf{x}, \sigma) = \frac{1}{\sigma^2 2\pi} \exp\left(\frac{-\mathbf{x}^\top \mathbf{x}}{2\sigma^2}\right), \quad (2)$$

$$\mathbf{D} = \begin{bmatrix} L_x^2(\mathbf{x}, \sigma_D) & L_x L_y(\mathbf{x}, \sigma_D) \\ L_x L_y(\mathbf{x}, \sigma_D) & L_y^2(\mathbf{x}, \sigma_D) \end{bmatrix}, \quad (3)$$

$$L(\mathbf{x}, \sigma) = g(\mathbf{x}, \sigma) * I(\mathbf{x}), \quad (4)$$

where $g(\mathbf{x}, \sigma)$ is the Gaussian scale-space operator (see [12] for background on scale-space), $L_i(\mathbf{x}, \sigma)$ is the first partial derivative along dimension i of an image $I(\mathbf{x})$ at scale σ, σ_D is referred to as the derivative scale and σ_I is referred to as the integration scale. The convolution with the Gaussian function $g(\mathbf{x}, \sigma_I)$ is equivalent to windowed integration of the derivative images. The second moment matrix is a positive definite symmetric matrix and can be seen as a (local) estimate of the covariance matrix of a two dimensional variable.

The second moment matrix was first used for affine adaptation in [13]. The scale-space concept was extended to affine scale space, so that the Gaussian operator is of the form,

$$g(\mathbf{x}, \Sigma) = \frac{1}{2\pi \det(\mathbf{\Sigma})} e^{\frac{-\mathbf{x}^\top \mathbf{\Sigma}^{-1} \mathbf{x}}{2}}. \quad (5)$$

Here $\mathbf{\Sigma}$ is a positive definite symmetric matrix known as the covariance matrix and $\det(\mathbf{\Sigma})$ is the determinant of $\mathbf{\Sigma}$. The affine second moment matrix computed in affine scale-space is then defined as,

$$\mu(\mathbf{x}, \mathbf{\Sigma}_D, \mathbf{\Sigma}_I) = \det(\mathbf{\Sigma}_D)\, g(\mathbf{x}, \mathbf{\Sigma}_I) * \mathbf{D}_A, \quad (6)$$

$$\mathbf{D}_A = \begin{bmatrix} L_x^2(\mathbf{x}, \mathbf{\Sigma}_D) & L_x L_y(\mathbf{x}, \mathbf{\Sigma}_D) \\ L_x L_y(\mathbf{x}, \mathbf{\Sigma}_D) & L_y^2(\mathbf{x}, \mathbf{\Sigma}_D) \end{bmatrix}, \quad (7)$$

with $\mathbf{\Sigma}_D$ and $\mathbf{\Sigma}_I$ differing only in scale. It was shown that if the second moment matrix is iteratively computed as,

$$\mathbf{M}_i = \mu(\mathbf{x}, k_D \mathbf{M}_{i-1}, k_I \mathbf{M}_{i-1}), \quad (8)$$

where i is the iteration number and $\mathbf{M}_0 = \mathbf{I}$, then $\mathbf{M}$ converges such that for sufficiently large n,

$$\mu(\mathbf{x}, k_D \mathbf{M}_n, k_I \mathbf{M}_n) \approx \mathbf{M}_n. \quad (9)$$

It was shown that the resulting matrix $\mathbf{M}$ is covariant under affine transformations of the image.

Instead of adapting the parameters of affine scale space, the method presented in [7] applies a normalizing affine transform to a local image region. Integration scale and differentiation scale are set proportional to the scale at which the feature is detected. At each iteration the second moment matrix is computed from the normalized image region using radially symmetric Gaussian kernels and is normalized to have a determinant of 1. A local image region around the selected feature is then transformed using the inverse square root of the normalized second moment matrix. This continues until the measured normalized second moment matrix is sufficiently close to the identity matrix. The method of [7] is also applied to both Harris and Determinant of Hessian features in [5].

A more complete algorithm is presented in [8] that updates the integration scale, differentiation scale and feature location at each iteration, before computing the second moment matrix. A measure of local shape isotropy is defined based on the eigenvalues λ_{min}, λ_{max} of the second moment matrix as,

$$Q = \frac{\lambda_{\text{min}}(\mu)}{\lambda_{\text{max}}(\mu)}. \quad (10)$$

Adaptation concludes when Q is sufficiently close to 1.

3. Hessian Matrix based Affine Adaptation

The second moment matrix has to date been the dominant affine shape estimator used in affine adaptation. This section explores the novel approach of using the Hessian matrix as an affine shape measure for affine adaptation. The main motivation for using the Hessian matrix is that it is simpler to compute (see Section 4) and only requires one scale parameter. Furthermore, the Hessian matrix is closely related to the determinant of Hessian function and the Laplacian function, which are both commonly used for feature extraction and scale selection. On the other hand, the Hessian matrix is predicted to be less stable than the second moment matrix due to the use of second order derivatives instead of first order derivatives and due to the absence of an averaging window applied after differentiation.

3.1. Estimation of the Parameters of a 2D Affine Gaussian Function

In this section it is demonstrated how the Hessian matrix can be used to find a linear transformation that transforms a 2D affine Gaussian function to an isotropic Gaussian.

The 2D affine Gaussian function centered on the coordinate origin is defined in equation (5). The matrix $\mathbf{\Sigma}$ may be decomposed as,

$$\mathbf{\Sigma} = \sigma_\alpha \mathbf{\Sigma}^{'} = \sigma_\alpha \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix}, \tag{11}$$

so that $\det(\mathbf{\Sigma}) = \sigma_\alpha^2$, $\det(\mathbf{\Sigma}^{'}) = 1$, $\sigma_{xx} > 0$ and $\sigma_{yy} > 0$. Rewriting g with arbitrary gain k in terms of $\mathbf{\Sigma}^{'}$ and σ_α yields,

$$kg\left(\mathbf{x}, \mathbf{\Sigma}\right) = ke^{\frac{-\mathbf{x}^\top \mathbf{\Sigma}^{'-1}\mathbf{x}}{2\sigma_\alpha^2}}. \tag{12}$$

In the following discussion the second order partial derivatives of g will be indicated as,

$$\begin{aligned} g_{xx}\left(\mathbf{x}, \mathbf{\Sigma}\right) &= \frac{\partial^2 g}{\partial x^2}, \\ g_{yy}\left(\mathbf{x}, \mathbf{\Sigma}\right) &= \frac{\partial^2 g}{\partial y^2}, \\ g_{xy}\left(\mathbf{x}, \mathbf{\Sigma}\right) &= \frac{\partial^2 g}{\partial x \partial y} = \frac{\partial^2 g}{\partial y \partial x}. \end{aligned} \tag{13}$$

Substituting $\mathbf{x} = \mathbf{0}$ into each second partial derivative of equation 12 gives,

$$\begin{aligned} kg_{xx}\left(\mathbf{0}, \mathbf{\Sigma}\right) &= k\frac{-\sigma_{yy}}{\sigma_\alpha^2} &= -k^{'}\sigma_{yy}, \\ kg_{yy}\left(\mathbf{0}, \mathbf{\Sigma}\right) &= k\frac{-\sigma_{xx}}{\sigma_\alpha^2} &= -k^{'}\sigma_{xx}, \\ kg_{xy}\left(\mathbf{0}, \mathbf{\Sigma}\right) &= k\frac{\sigma_{xy}}{\sigma_\alpha^2} &= k^{'}\sigma_{xy}. \end{aligned} \tag{14}$$

This set of equations can be arranged in the form of the covariance matrix in (11),

$$\mathbf{\Sigma}^{'} = k^{'-1}\begin{bmatrix} -g_{yy}\left(\mathbf{0}, \mathbf{\Sigma}\right) & g_{xy}\left(\mathbf{0}, \mathbf{\Sigma}\right) \\ g_{xy}\left(\mathbf{0}, \mathbf{\Sigma}\right) & -g_{xx}\left(\mathbf{0}, \mathbf{\Sigma}\right) \end{bmatrix}. \tag{15}$$

The matrix on the right is the negative inverse Hessian matrix evaluated at the center of the Gaussian function $\left(-\mathbf{H}^{-1}\left(g\right)\left(\mathbf{0}, \mathbf{\Sigma}\right)\right)$. The Hessian matrix of a function is defined as the matrix of second order partial derivatives of the function,

$$\mathbf{H}(f)(x, y) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}. \tag{16}$$

Hence $\mathbf{\Sigma}^{'}$ can be recovered by evaluating the inverse Hessian matrix at the center of the Gaussian function. Since $\det\left(\mathbf{\Sigma}'\right) = 1$, the scale of $\mathbf{H}$ may be discarded by normalizing its determinant to 1. The affine parameters of a Gaussian blob in an image can therefore be recovered if the center of the blob is known.

To normalize the anisotropic blob to an isotropic blob, simply transform the image by the square root of $\mathbf{\Sigma}'$,

$$\begin{aligned} kg\left(\mathbf{\Sigma}'^{\frac{1}{2}}\mathbf{x}, \mathbf{\Sigma}\right) &= ke^{\frac{-\mathbf{x}^\top \mathbf{\Sigma}'^{\frac{1}{2}\top}\mathbf{\Sigma}^{'-1}\mathbf{\Sigma}'^{\frac{1}{2}}\mathbf{x}}{2\sigma_\alpha^2}} \\ &= ke^{\frac{-\mathbf{x}^\top \mathbf{I}\mathbf{x}}{2\sigma_\alpha^2}}. \end{aligned} \tag{17}$$

If the sign of the Gaussian is negated, then the Hessian matrix will also be negated. If the sign of the Gaussian is not known, the Hessian should be negated whenever it is not positive definite.

3.2. Application to Affine Shape Measurement of Image Regions

The above proof shows how to directly measure the affine shape of a Gaussian function of which the center point is known. This method may be applied to local image features. Image structures are in general not proper affine Gaussian functions, however the characteristic scale representations of blob-like structures are sufficiently close to affine Gaussian blobs for the Hessian matrix to be of use. A characteristic scale blob detector, such as the Laplacian or determinant of Hessian detector, can be used to find the blob scale and center point. Unfortunately, neither scale selection, nor blob center location is highly accurate unless the shape of the detected blob is already isotropic. The Hessian matrix can therefore be used for affine normalization in conjunction with a characteristic scale blob detector, however this system does not provide a direct solution and must be applied in an iterative fashion, similar to how the second moment matrix is used.

The Hessian matrix is predicted to be less stable than the second moment matrix due to the use of second order derivatives instead of first order derivatives, and due to the absence of an averaging window applied after differentiation. For this reason it is recommended that a small averaging filter is applied to improve stability. Implementations presented in this paper make use of a Gaussian filter with scale parameter $\sigma = 1$ and truncated to a 3×3 pixel window to average the Hessian matrix over a small area.

4. Comparison of the Complexity of the Hessian and Second Moment Operators

Both the Hessian matrix and the Second Moment matrix can be computed from an image by applying a series of filters to the image. The scale operator and windowed integration operation can be implemented as Gaussian filters, which are most efficiently realized as IIR filters [14, 15]. The differentiation operations can be implemented as 3×3 kernel filters. The complexity of all these filters is linear in the number of image pixels processed. The only difference between computing the Hessian matrix and the Second Moment matrix is the arrangement of the above filters.

To compute these operators at a single point in an image (as is done repeatedly during affine adaptation), the filters need only be applied to the image region around the point that has significant influence on the filter output. Once one filter has been applied, its support region can be discarded, leaving a smaller area to process for subsequent filters. The

Filter	Label	Op's per pixel
$\partial I/\partial x$	c_x	6
$\partial I/\partial y$	c_y	6
$\partial^2 I/\partial x^2$	c_{xx}	9
$\partial^2 I/\partial y^2$	c_{yy}	9
$\partial^2 I/\partial x \partial y$	c_{xy}	4
$g(\mathbf{x}, \sigma)$	$c_{g\sigma}$	k_g
$g(\mathbf{x}, 1)$ truncated to 3×3	c_{g1}	9

Table 1. Number of operations per pixel for derivative kernels.

derivative kernels require support area of width 3 pixels. IIR Gaussian filters technically require infinite support, though the region within a radius of 3σ accounts for 98% of the filter response. The total support width for a Gaussian filter is therefore specified as 6σ.

The number of operations per pixel required by each filter is listed in Table 1. Each derivative kernel requires a different number of operations per pixel, due to the presence of zero elements in the kernels. The constant k_g depends on the specific Gaussian filter implementation.

The Hessian operator consists of filtering to the feature scale σ with a Gaussian filter, followed by computing second order partial derivatives and finally a 3×3 averaging filter to each of the three derivative images. The number of operations required to compute the Hessian matrix at one pixel is,

$$\begin{aligned} c_H &= c_{g\sigma}(6\sigma + 6) + 3(c_{xx} + c_{yy} + c_{xy}) + 3c_{g1} \\ &= k_g(6\sigma + 6) + 93. \end{aligned} \tag{18}$$

The Second Moment operator consists of filtering to the differentiation scale σ_D, computing first order partial derivative images L_x and L_y, multiplying the derivatives to produce images L_x^2, L_y^2 and $L_x L_y$ and finally filtering each derivative product image with a Gaussian with scale σ_I to compute windowed integration. The number of operations required to compute the Second Moment matrix at one pixel is,

$$\begin{aligned} c_S &= c_{g\sigma}(6\sigma_I + 6\sigma_D + 3) + 6\sigma_I(c_x + c_y) \\ &\quad + 3 \cdot 6 \cdot \sigma_I + 3c_{g\sigma}6\sigma_I \\ &= k_g(24\sigma_I + 6\sigma_D + 3) + 90\sigma_I \end{aligned} \tag{19}$$

Given that for the same feature both σ_I in the Second Moment operator and and σ in the Hessian operator are set to the characteristic scale of the feature, it is clear that the Hessian matrix requires significantly less computational effort than the Second Moment matrix.

5. Experimental Evaluation

5.1. Evaluation Method

To compare the performance of the Hessian matrix and the second moment matrix when used in the task of affine adaptation, the following four detectors were implemented:

1. Determinant of Hessian detector using the Hessian matrix for affine adaptation (He_h).
2. Determinant of Hessian detector using the second moment matrix for affine adaptation (He_s).
3. Harris detector using the Hessian matrix for affine adaptation (Ha_h).
4. Harris detector using the second moment matrix for affine adaptation (Ha_s).

The scale and affine adaptation algorithm is based on the adaptation algorithm applied in the Harris Affine and Hessian Affine detectors defined in [5] and incorporates the scale selection method from [16]. The algorithm is summarized as follows:

1. Detect multi-scale features in isotropic scale-space.
2. Use the feature locus clustering based scale selection method in [16] to quickly select characteristic scale features (using the Laplacian).
3. Apply iterative affine adaptation while keeping the scale and position of features fixed as in [7].

All four detectors used the above adaptation framework with only the saliency function and affine shape estimation matrix implemented differently for each detector.

The repeatability test described in [5] was used to determine the repeatability of detectors and the number of correspondences produced when applied to sequences of test images. Additionally, the processing time of the feature extraction process was recorded and analyzed to assess detector efficiency.

The repeatability test functions by projecting features detected in a test image to a base image using a ground truth homography. Correspondence counts are determined according to the overlap between the projected and base image features. For a detailed description of the test, refer to [5]. The test software and sequences were sourced from http://www.robots.ox.ac.uk/∼vgg/data/data-aff.html.

In order to measure the relative efficiency of the detectors, the rate at which corresponding features were produced, r, in number of correspondences per second, was computed as,

$$r = \frac{c_i}{t_i + t_0}, \tag{20}$$

where c_i is the number of correspondences between image 0 and i, and t_i and t_0 are the time taken to extract features from image i and 0, respectively. The test sequences used were *graffiti* (varying viewing angle), *wall* (varying viewing angle), *bark* (varying scale and rotation), *boat* (varying scale and rotation), *leuven* (varying lighting) and *trees* (varying image blur).

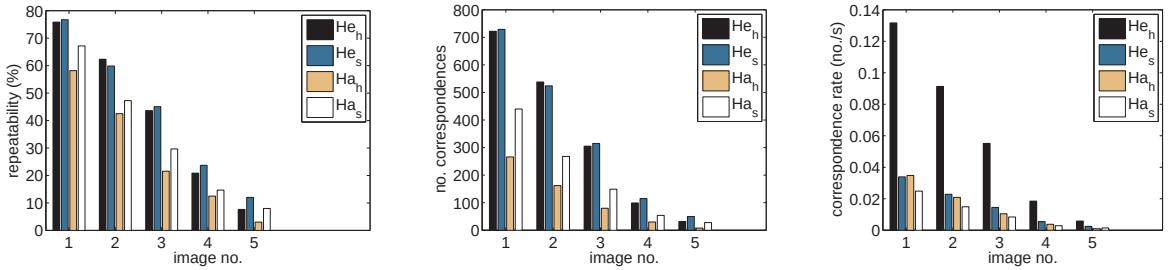

Figure 1. Results for the *graffiti* sequence. From left to right: repeatability, correspondence count, correspondences produced per second.

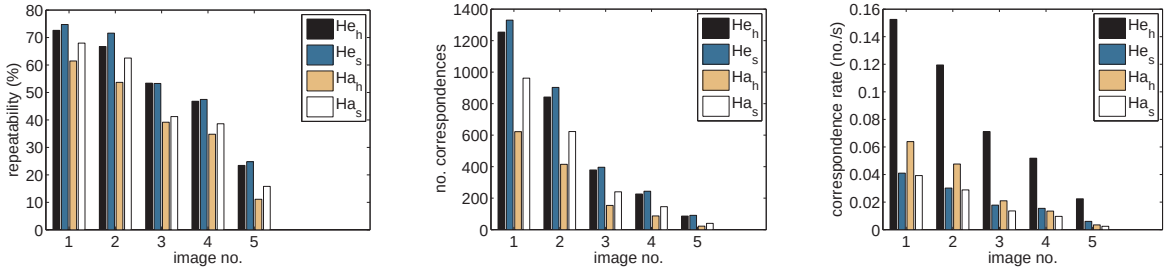

Figure 2. Results for the *boat* sequence. From left to right: repeatability, correspondence count, correspondences produced per second.

5.2. Results

Selected test results are presented in Figures 1–4 (not all results are presented in figures due to space constraints). Each figure presents the results of testing the four detectors on one of the test sequences. The figures show repeatability, correspondence counts and the correspondence rate (r). The horizontal axis indicates the test image number. Test images are arranged in order of increasing severity of the transformation or distortion tested in the image sequence.

5.3. Discussion

For the Determinant of Hessian detectors the repeatability results are similar for both the Hessian and second moment matrices. The He_s detector performed marginally better than He_h in *graffiti*, *bark* and *boat*, and He_h performed significantly better in *leuven* and *trees* and marginally better in *wall*. The number of correspondences followed the same trend, but the differences were less pronounced. Correspondence rate results showed that the Hessian matrix was more efficient across all tests by as much as 400% and 336% on average. The Hessian matrix is also less affected by image blur or lighting conditions.

For the Harris detectors the repeatability and correspondence results show that the Hessian matrix performed significantly worse than the second moment matrix. Despite this, the Hessian matrix still performed significantly better than the second moment matrix in terms of efficiency. On average, a 144% improvement in efficiency was measured.

This difference in behavior of the Hessian matrix between the Harris and Hessian detectors may be attributed to the fact that the Harris detector selects corner features more than blob features, whereas the Hessian detector extracts blobs with a high degree of reliability. The Hessian matrix should be computed at the center of a blob, as outlined in Section 3.1, and may be unstable or inaccurate otherwise. The results indicate that the Hessian matrix is very effective and efficient in affine adaptation of blob-like features, but is much less effective for corner features.

6. Conclusion

The Hessian matrix may be used for estimating local affine deformation in an affine adaptation process, similar to how the second moment matrix is commonly used. When combined with a characteristic scale blob detector the Hessian matrix provided a considerable reduction in computation time compared to the second moment matrix at no cost to the repeatability of the detector or the number of correspondences produced. Efficiency gains of up to 400% and 336% on average were observed. This reduction in computation time is primarily attributed to the fact that the Hessian matrix is simpler to compute than the second moment matrix. The detector using the Hessian matrix is also more robust in terms of changing lighting conditions and varying focus or blur.

When combined with a corner detector the Hessian matrix produced significantly fewer correspondences and lower repeatability. This is due to the fact that the Hessian matrix is less stable and accurate when not computed at the center of a blob-like image region. Despite the reduction in correspondence counts, a 144% efficiency improvement was measured.

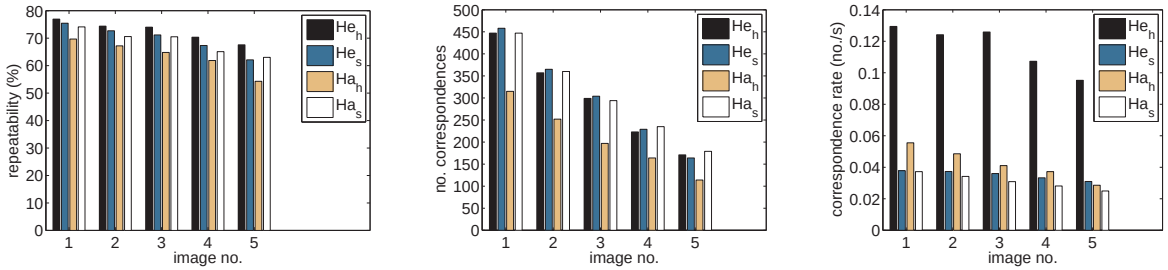

Figure 3. Results for the *leuven* sequence. From left to right: repeatability, correspondence count, correspondences produced per second.

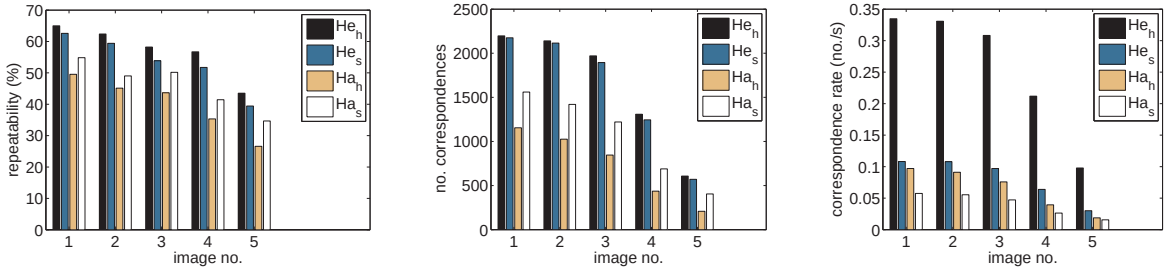

Figure 4. Results for the *trees* sequence. From left to right: repeatability, correspondence count, correspondences produced per second.

7. Acknowledgments

This project was supported by the Australian Government Department of the Prime Minister and Cabinet.

References

[1] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008. 1

[2] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 189–192, 1988. 1

[3] P.R. Beaudet. Rotationally invariant image operators. In *In International Joint Conference on Pattern Recognition*, pages 579–583, Kyoto, Japan, 1978. 1

[4] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1

[5] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. 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. 1, 2, 4

[6] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3d objects. *International Journal of Computer Vision*, 73(3):263–284, 2007. 1

[7] A. Baumberg. Reliable feature matching across widely separated views. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 774–781, Hilton Head Island, South Carolina, USA, 2000. 1, 2, 4

[8] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60:63–86, 2004. 1, 2

[9] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer Vision – ECCV 2006*, pages 404–417, Graz, Austria, 2006. 1

[10] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. volume 3951/2006, pages 430–443. Springer, 2006. 1

[11] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In P.L. Rosin and D.Marshall, editor, *British Machine Vision Conference, BMVC 2002*, volume 1, pages 384–393, Cardiff, UK, 2002. 1

[12] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic, Boston, 1 edition, 1994. 2

[13] T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-d depth cues from affine distortions of local 2-d brightness structure. In *Computer Vision – ECCV '94*, volume 800/1994 of *Lecture Notes in Computer Science*, pages 389–400. Springer, Berlin / Heidelberg, 1994. 2

[14] Rachid Deriche. Recursively implementing the gaussian and its derivatives. Technical report, INRIA, April 1993 1993. 3

[15] I.T. Young and L.J. van Vliet. Recursive implementation of the gaussian filter. *Signal Processing*, 44(2):139–151, 1995. 3

[16] R. Lakemond, D.N.R. McKinnon, C. Fookes, and S. Sridharan. A feature clustering algorithm for scale-space analysis of image structures. In *International Conference on Signal Processing and Communication Systems, ICSPCS'2007*, Gold Coast, Australia, 2007. 4