Maximally Stable Colour Regions for Recognition and Matching

Per-Erik Forssén
Department of Computer Science
University of British Columbia
perfo@cs.ubc.ca

Abstract

This paper introduces a novel colour-based affine covariant region detector. Our algorithm is an extension of the maximally stable extremal region (MSER) to colour. The extension to colour is done by looking at successive time-steps of an agglomerative clustering of image pixels. The selection of time-steps is stabilised against intensity scalings and image blur by modelling the distribution of edge magnitudes. The algorithm contains a novel edge significance measure based on a Poisson image noise model, which we show performs better than the commonly used Euclidean distance. We compare our algorithm to the original MSER detector and a competing colour-based blob feature detector, and show through a repeatability test that our detector performs better. We also extend the state of the art in feature repeatability tests, by using scenes consisting of two planes where one is piecewise transparent. This new test is able to evaluate how stable a feature is against changing backgrounds.

1. Introduction

In recent years there has appeared many vision algorithms, that use local covariant feature detection and description [1, 5, 7, 8, 9, 10, 14, 15]. Such methods allow fast 3D object recognition from single images in cluttered environments, under partial occlusion. They are also applicable to the related problems of view-matching, and wide-baseline stereo.

From an abstract point of view, all such systems work in three steps. They start with a *detection step* that detects reference frames in an image. These frames are covariant with image changes under either a *similarity transform* (translation, scale, and rotation) [7, 15], or an *affine transform* (also includes skew and different scalings of axes) [5, 8, 10, 14]. This is followed by a *description step*, where a local image patch is sampled in the reference frame, and converted to a descriptor vector that can be compared with a set of reference descriptors in the final *matching step*. This allows the system to find a sparse set of correspondences between scene and memory (3D object recognition), or between two views of a scene (view-matching and wide-baseline stereo). Figure 1 shows 428 such correspondences found between two views, using the detector proposed in this paper.

Figure 1. View correspondence. Left to right: Frontal view of scene, frontal view with 428 corresponding features painted in (as white ellipses), 40° view with corresponding features painted in.

Even though most cameras these days use colour, most covariant feature detectors have been based on grey-scale images. In the descriptor step however, the use of colour has been common [8, 14, 16]. The reason why colour has not been used in the detection step is that it normally increases the computational load three times, but unless care is taken, the performance improvement is negligible.

We have found three prior attempts at colour based detection of local covariant features. Corso and Hager [1] find extrema in DoG responses defined from three linear projections of the RGB space. They also compute such extrema on two variance images computed with fixed window sizes (thus non-scale invariant). The detected regions are represented as image-axis-aligned ellipses, and they are thus only scale (x and y) and translation invariant. Their detector gives a lower repeatability than the the grey-scale SIFT detector [7] in their own experiments [1] (unless they halve the aspect ratio). Unnikrishnan and Herbert [15] test two illuminant invariant scalar functions, one invariant to independent scalings of the RGB channels, and one invariant to a full 3×3 perturbation of the RGB space. They proceed by detecting scale and rotation invariant LoG points. They find that the version only invariant to independent scalings of the RGB channels performs best under rotation and scale changes, and more importantly it performs better than LoG

1-4244-1180-7/07/$25.00 ©2007 IEEE

on a plain grey-scale image. They do not evaluate their detector under view changes. Forssén and Moe [5] detect affine covariant regions by approximating regions from a scale-space segmentation algorithm as ellipses. We have downloaded their algorithm implementation [17] and compare it with ours in the experiment section.

We propose an extension of the popular MSER covariant region detector [8] to colour. The original MSER detector finds regions that are stable over a wide range of thresholdings of a grey-scale image. We instead detect regions that are stable across a range of time-steps in an agglomerative clustering of image pixels, based on proximity and similarity in colour. In addition to extending the MSER algorithm to non-scalar functions, the algorithm contains two important theoretical contributions: First, we derive a novel colour edge significance measure from the Poisson statistics of pixel values. Second, we introduce a way to stabilise the selection of clustering time-steps against intensity scaling and image blur.

In the experiment section, we conduct a repeatability test to demonstrate that our detector improves over the intensity based MSER algorithm [8], and the colour based blob feature detector [5]. We also show that our detector is actually faster than most intensity based detectors (not MSER though). We also show the benefits of using our novel edge significance measure, and of changing the amount of spatial neighbours considered in the agglomerative clustering.

2. Maximally Stable Regions

The concept of maximally stable regions (MSR) was originally defined in [8], by considering the set of all possible thresholdings of a intensity image, I, to a binary image, E_t:

$$E_t(\mathbf{x}) = \begin{cases} 1 & \text{if } I(\mathbf{x}) \geq t \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

A maximally stable extremal region (MSER) is then a connected region in E_t with little size change across several thresholdings. An evolution that successively increases the threshold t in (1) detects only dark regions (called MSER+), bright regions (called MSER-) are obtained by inverting the intensity image. The number of thresholds for which the region is stable is called the *margin* of the region. Figure 2, left, shows the output of the MSER detector when a minimum margin of 7 has been chosen. We have chosen to show the approximating ellipses of the regions instead of the actual regions, in order to better illustrate that the regions are often nested.

A straightforward way to detect more regions with MSER, if a colour image is available, is to run the detector on the grey-scale image, and on red-green and yellow-blue channels, as was done with DoG in [1]. This will cause some duplicate detections. We remove duplicates in the

Figure 2. Comparison of MSER and MSCRs. Ellipses show approximating ellipses for regions. Left: White is MSER+, Black is MSER-, in total 34 regions. Right: MSCR output, in total 42 regions.

added channels if the region centroid distance is below 4 pixels, and the areas differ by less than 10%. We will refer to this method as MSER3.

MSER has earlier been extended to colour for tracking by Roth *et al*. [13]. Their approach requires as input an RGB-space Gaussian model of the object to be tracked, and consequently it does not do bottom up feature detection (which we do).

2.1. Agglomerative clustering

To extend the MSR concept to colour images, we define an *evolution process* over the image $\mathbf{I} : \Omega \mapsto \mathbb{R}^3$, where $\Omega = [1 \dots L] \times [1 \dots M] \subset \mathbb{Z}^2$ is the set of all image positions. This process successively clusters neighbouring pixels with similar colours. We will consider two sets of neighbouring relations, one with horizontal and vertical relations, and one which also includes diagonal relations:

$$\mathcal{N}_1 = \{(\mathbf{x}, \mathbf{y}) \in \Omega^2 : \mathbf{y} = \mathbf{x} + \tbinom{0}{1} \vee \mathbf{y} = \mathbf{x} + \tbinom{1}{0}\}\,. \quad (2)$$

$$\mathcal{N}_2 = \mathcal{N}_1 \cup \{(\mathbf{x}, \mathbf{y}) \in \Omega^2 : \mathbf{y} = \mathbf{x} + \tbinom{1}{1} \vee \mathbf{y} = \mathbf{x} + \tbinom{-1}{1}\}\,. \quad (3)$$

The positions, $\mathbf{x}$, $\mathbf{y}$, and the colour distance (defined in section 2.2) are stored in a list. For an $L \times M$ image the length of this list becomes either $|\mathcal{N}_1| = 2LM - L - M$ or $|\mathcal{N}_2| = 6LM - 3L - 3M + 2$. Such neighbour distances are sometimes called *crack edges* [11] to emphasise that the output actually corresponds to a position in-between the pixels. Note that the diagonal edges in $\mathcal{N}_2$ should be normalised by $1/\sqrt{2}$ to compensate for their larger spatial distance.

For each *time step* $t \in [0 \dots T]$, the evolution is a map $E_t : \Omega \mapsto \mathbb{N}$ of labels. Each unique label defines a contiguous region $\mathcal{R} \subseteq \Omega$. Any two positions $\mathbf{x}, \mathbf{y} \in \mathcal{R}$ are connected by a path of distances which are all smaller than $d_{\text{thr}}(t)$. The design of the function $d_{\text{thr}}(t)$ is the topic of section 2.3. Figure 3 shows a sample of evolutions E_t of the image in figure 2.

The label image E_t is the generalisation of the thresholded images (1) in the MSER algorithm. The label image E_0 is all zeroes, and E_{t+1} is constructed from E_t by assigning new regions to all pairs of pixels with a distance smaller

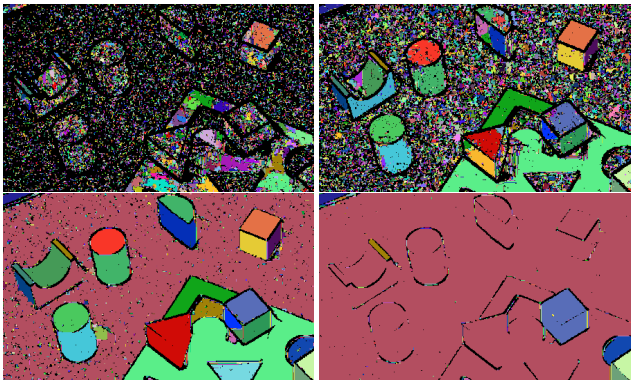

Figure 3. Illustration of evolution used in colour MSER detector. Left to right, top to bottom: $d_{\text{thr}} = 0.0065, 0.011, 0.023, 0.038$. Each region is painted in a different, random colour.

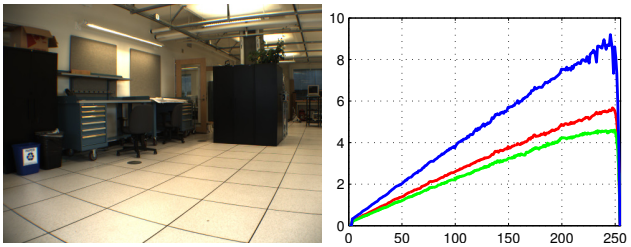

Figure 4. Image variance as function of intensity. Left: Mean image of 100 identical frames. Right: Variance from mean as function of intensity for the three colour bands.

than $d_{\text{thr}}(t)$. Alternatively, if one of the pixels in the pair already belongs to a region, the non-assigned pixel is appended to the region, and if both pixels belong to regions the corresponding regions are merged.

2.2. Edge significance

Most digital cameras in use today have sensors that essentially count the number of photons, n, falling onto the detector over a period of time. This implies that the image noise should follow the discrete *Poisson distribution*. For high values of n, a good continuous approximation is a Gaussian with mean and variance equal to the expected intensity [3]. This is easily verified by the simple experiment reproduced in figure 4. Here the variance as a function of intensity has been estimated from 100 frames of a static scene. As can be seen, the variance is indeed linear in the expected intensity. This motivates us to model the measured intensity, I, given the expected intensity μ, as $p(I|\mu) = \mathrm{g}(\mu, \sqrt{a\mu})$, where a is the *camera gain* that converts the photon count to a pixel value, and $g(\mu, \sigma)$ is a Gaussian with mean μ and variance σ^2. As can be seen in figure 4, the camera white balance has resulted in different gain factors for the three colour bands.

In order to derive a measure of edge significance, we now

Figure 5. Edge significance PDF. Left: Input image, Centre: Image blurred with a Gaussian ($\sigma = 1.2$). Right: PDFs for the edge significance measure. Solid curve is the PDF for the original image, dashed curve is the PDF after Gaussian blur.

consider the probability that a pixel location $\mathbf{x}$ has a larger mean than its neighbour $\mathbf{y}$:

$$P(\mu(\mathbf{x}) > \mu(\mathbf{y})) = \int_0^\infty g(t, \mu, \sigma) dt \\ = 1 - \Phi(-\mu/\sigma) = \Phi(\mu/\sigma)\,, \quad (4)$$

where Φ is the CDF of the standardised normal distribution, and $\mu = \mu(\mathbf{x}) - \mu(\mathbf{y})$ and $\sigma = \sqrt{a(\mu(\mathbf{x}) + \mu(\mathbf{y}))}$. Since Φ is monotonic, the absolute value of the argument of Φ gives us an ordering of edges according to statistical significance:

$$d = |\mu/\sigma| = |\mu(\mathbf{x}) - \mu(\mathbf{y})| / \sqrt{a(\mu(\mathbf{x}) + \mu(\mathbf{y}))}\,. \quad (5)$$

Assuming that the three colour bands are independent, we should multiply their PDFs, and this instead gives us:

$$d^2 = \sum_{k=1}^{3} \frac{(\mu_k(\mathbf{x}) - \mu_k(\mathbf{y}))^2}{a_k(\mu_k(\mathbf{x}) + \mu_k(\mathbf{y}))}\,. \quad (6)$$

When we do not have access to the gains, we can simply set $a_k = 1$. We also have to replace each mean $\mu_k(\mathbf{x})$ with its maximum likelihood estimate given one image, *i.e.*, the pixel value $I_k(\mathbf{x})$. This measure is sometimes referred to as the *Chi-squared distance*. To demonstrate that (6) is indeed an improvement, we will compare it with the Euclidean distance

$$d^2 = \sum_{k=1}^{3} (I_k(\mathbf{x}) - I_k(\mathbf{y}))^2\,, \quad (7)$$

in the experiment section.

2.3. Distribution of Edge Significance Magnitudes

The distribution of the edge significance measure (6) between neighbouring pixels in an RGB image is far from uniform. Due to the high degree of spatial correlation in an image we tend to have many small values, and increasingly fewer large ones. Figure 5 shows an estimated *probability density function* (PDF) of (6) for an image, and also how the PDF is affected by applying Gaussian blur to the image. The Euclidean distance (7) follows a similar, but slightly more jagged distribution.

If we were to linearly increase the threshold d_{thr} with the time-step, we would end up with an image evolution

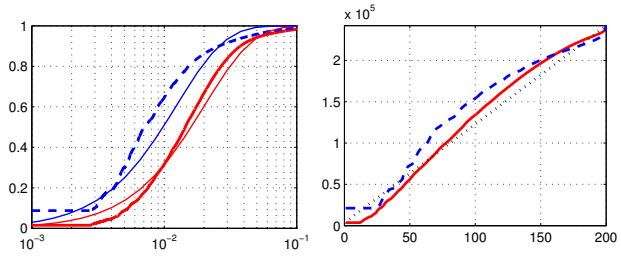

Figure 6. Left: Empirical CDFs and the approximating $\mathcal{X}_3^2$ CDFs on a logarithmic scale. Solid curve is empirical distance CDF for the original image, dashed curve is distance CDF after Gaussian blur. Thin curves are approximating $\mathcal{X}_3^2$ CDFs. Right: The resultant number of edges processed after each time step. Dotted straight line is desired curve.

where changes are very fast in the beginning, and very slow at the end of the evolution. In order for the image evolution to progress at a similar rate for different time steps in a given image, we should instead change the distance threshold according to the inverse of the *cumulative distribution function* (CDF), $c(x) = P(d < x)$. This will also have the benefit of stabilising the evolution under image blur and contrast changes. In order to stabilise the evolution with respect to small image content changes, we should not use the exact inverse of the CDF, but a regularised version. If we assume that the normalised difference between two pixels is Gaussian, the edge significance measure (6) will follow a scaled Chi-squared distribution [3]. For grey-scale images we should use $\mathcal{X}_1^2$, and for colour images $\mathcal{X}_3^2$:

$$c_1(x) = \text{erf}(\sqrt{x/\lambda}) \quad \text{and} \tag{8}$$

$$c_3(x) = -\sqrt{\frac{4x}{\lambda\pi}}e^{-x/\lambda} + \text{erf}(\sqrt{x/\lambda})\,. \tag{9}$$

For $\mathcal{X}_1^2$, we have $\mu = \lambda/2$ and for $\mathcal{X}_3^2$, $\mu = 3\lambda/2$, where μ is the mean, which we can estimate by the sample average. Figure 6, left, shows the estimated distance CDFs for the blurred and non-blurred images in figure 5.

Note that gradient magnitudes, i.e. (7), have previously been shown to follow the Weibull distribution [6]. We settled for the $\mathcal{X}_1^2$ and $\mathcal{X}_3^2$ distributions here, since they gave a good enough fit, and are easier to estimate.

After estimation of the mean, evolution thresholds can be computed as $d_{\text{thr}}(t) = c^{-1}(t/T)$ for $t \in [0, T]$. For speed, we store values of c^{-1} in a lookup table. The number of time-steps T is heuristically set to $T = 200$. In figure 6, right, we plot the actual number of edges processed after each evolution time-step. This should ideally be a straight line, and as can be seen we come reasonably close. One could also imagine sorting the distances and picking $L \times M/T$ new edges in each time-step. This turns out to be a bad idea, since such an approach would inevitably cause some edges with the same distance to be split across two time-steps.

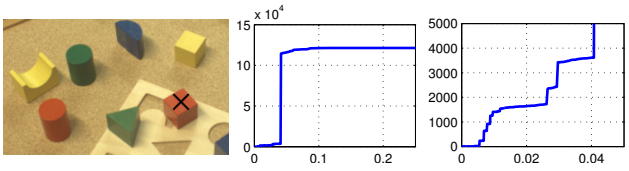

Figure 7. Evolution for a single pixel. Left: The pixel is indicated by a cross. Centre: Region area as function of distance threshold. Right: Detail of region area function.

2.4. Detecting Stable Regions

An alternative illustration of region evolution is shown in figure 7. Here the area of the region to which a selected pixel belongs is plotted as a function of the current merger threshold d_{thr}. As can be seen in this plot, the area function has plateaus of little or no area change, and it is these plateaus we want to detect as stable regions.

For each region in the label image the algorithm remembers the area a_* and distance threshold d_* at which the region appeared. Whenever the area increases more than a given percentage between two time steps (*i.e.*, $a_{t+1}/a_t > a_{\text{thr}}$), a_* and d_* are re-initialised. Typical value is $a_{\text{thr}} = 1.01$. For each new time-step the algorithm computes the slope, s, of the curve between d_* and d_t:

$$s = \frac{a_t - a_*}{d_t - d_*}\,. \tag{10}$$

If s is the best (*i.e.*, smallest) since the previous re-initialisation, the region is stored, possibly replacing an earlier region. For stability reasons, we also avoid choosing the first distance as the optimal. The *margin* of the region is set to the distance between the initialisation point and point before the next initialisation point, *i.e.*, $m = d_\dagger - d_*$. For each region, now represented by a binary mask $v : \Omega \mapsto \{0, 1\}$, we also compute and store the raw moments $\mu_{k,l}$ up to order 2, and the colour moments π of order 0:

$$\mu_{k,l}(v) = \sum_x \sum_y x^k y^l v(x, y) \quad \text{and} \tag{11}$$

$$\pi(v) = \sum_x \sum_y \mathbf{I}(x, y) v(x, y)\,. \tag{12}$$

Although the above is the definition of the moments, no integration is required in practise. All moments can be updated incrementally, as new pixels join a region, or two regions merge. Since the moments are linear functions of the region mask v, a region merger simply means an addition:

$$v = v_1 + v_2 \quad \Rightarrow \quad \begin{cases} \mu_{k,l}(v) = \mu_{k,l}(v_1) + \mu_{k,l}(v_2)\,, \\ \pi(v) = \pi(v_1) + \pi(v_2)\,. \end{cases} \tag{13}$$

From the raw moments we then compute the region area a, centroid $\mathbf{m}$, inertia matrix $\mathbf{C}$, and average colour $\mathbf{p}$, see e.g. [11]. These measures define an approximating ellipse for the detected region as:

$$\mathcal{R}(\mathbf{m}, \mathbf{C}) = \{\mathbf{x} : (\mathbf{x} - \mathbf{m})^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m}) \leq 4\}. \quad (14)$$

2.5. Restricting the Number of Regions

The above described procedure produces a huge number of regions, which are pruned by requiring that the margin, $m = d_\dagger - d_*$ is above a given threshold $m > m_{\text{min}}$ (typically set to 0.003). After this procedure, we are still left with some small unstable regions in textured areas. We deal with this simply by also requiring that the area is above a threshold $a > a_{\text{min}}$ (typically set to 60). Finally we require that the minor axis of the approximating ellipse is larger than 1.5 pixels. This last constraint removes regions that are very long and narrow, since such regions tend to appear more often when aligned to the image grid, and this makes them highly unstable. The output of the *maximally stable colour region* (MSCR) algorithm after pruning is shown in figure 2, right. Compare this to the output of the MSER detector in figure 2, left. We can observe that the MSER detector sometimes combines object and shadow, while this is less often the case for MSCR.

2.6. Improving the Distance Computation

The distance measurements that control the evolution of the image (6) are a very crude form of measurement, potentially sensitive to image noise. We have tried two ways of improving the distance computation: **Image smoothing**. Here each colour band of the input image is blurred with a Gaussian filter, as is common in many segmentation algorithms, see *e.g.* [4]. **Edge smoothing**. Here each of the four edge directions (see (2) and (3)) are treated as edge images, and each is separately smoothed. These two approaches will be compared in a repeatability test in section 3.4.

We let the standard deviation of the Gaussian filter be specified by the side of the spatial support, as $\sigma = \sqrt{N/5}$, *e.g.*, for a 5×5 filter we get $\sigma = 1.0$. This will make the Gaussians similar to binomial filters of the same size.

Note that image smoothing compresses the distance PDF (see figure 5). This is also the case for edge smoothing. This implies that the margin threshold, m_{min} (see section 2.5) should be lowered accordingly.

2.7. Performance Issues

To speed up re-labeling after region mergers, we incrementally maintain the bounding boxes of the active regions. Even better performance could probably be obtained by using the union/find algorithm [2]. Table 1 gives average computation time (of 10 runs) for the first 800×640 image in the 'Graffiti' test set on a 3GHz Pentium 4 CPU. The grey-scale methods are downloaded at [18]. The blob detector is downloaded at [17]. As can be seen in table 1, our detector is significantly faster than the other colour based detector, and also faster than all grey-scale based methods except MSER.

grey-scale method	computation time (seconds)
MSER	0.26
Harris affine	2.63
Hessian affine	3.50
IBR	13.5
EBR	144.9
colour method	computation time (seconds)
MSCR($\mathcal{N}_2$)+edge blur	1.67
MSCR($\mathcal{N}_1$)+edge blur	1.01
Blob detector	4.95

Table 1. Average computation times on 'Graffiti1' image.

3. Repeatability Test

In order to demonstrate the stability of the developed feature, we make use of the affine covariant feature evaluation software available at [18]. This software compares detected features in two views of a scene that differ by a known view change (a homography), and checks whether corresponding features are extracted in both views. It was used in a recent performance evaluation of affine covariant region detectors [10], where the original MSER detector compared well to the other detectors in all of the sequences, and was actually the winner in several of them. For clarity of presentation, we will thus only include the MSER curves from [10], and ask the interested reader to look up the results for other detectors. We will show repeatability graphs of the following methods:

1. MSER detector, downloaded at [18]
2. MSER3 detector, see section 2.
3. Blob detector, downloaded at [17]
4. MSCR with edge smoothing ($N = 7$, $m_{\text{min}} = 0.0015$).

See section 2.6 for details on MSCR smoothing. The images in the test sets are available on the web at [18], see also figure 1 for an illustration of the view change experiment.

3.1. Repeatability Measure

The repeatability measure used is based on the overlap error of the ellipses μ_a and μ_b in view 1 and 2 respectively:

$$\epsilon_o(\mu_a, \mu_b) = 1 - \frac{|\mathcal{R}(\mu_a) \cap \mathcal{R}(\mathbf{H}^T \mu_b \mathbf{H})|}{|\mathcal{R}(\mu_a) \cup \mathcal{R}(\mathbf{H}^T \mu_b \mathbf{H})|}. \quad (15)$$

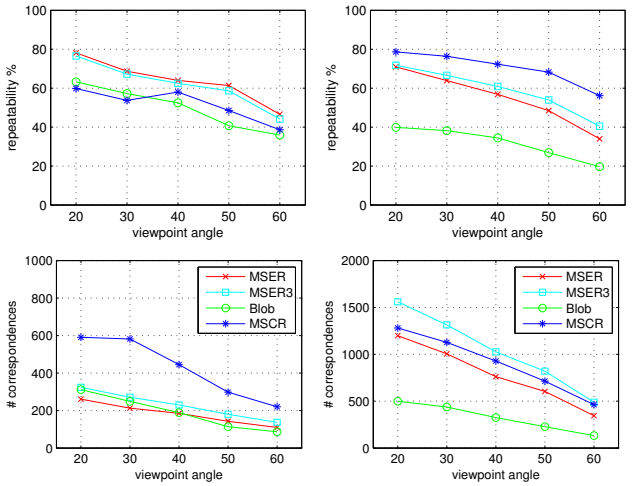

Figure 8. Repeatability rates and number of correspondences for viewpoint change. Top row shows repeatability rates, and bottom row shows number of correspondences. Left column is 'Graffiti' sequence, right column is 'Wall' sequence.

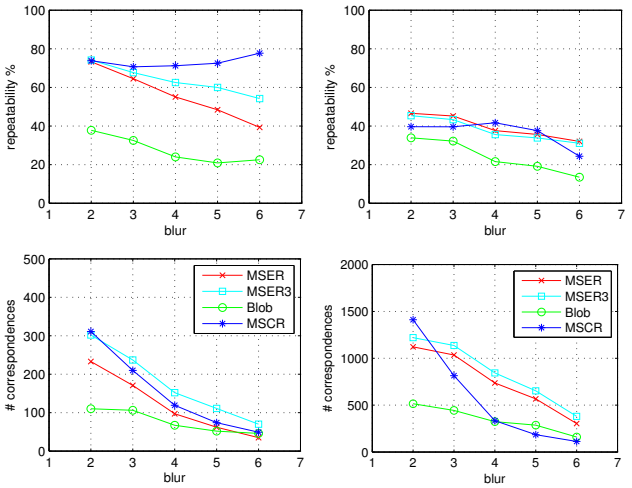

Figure 9. Repeatability rates and number of correspondences for de-focus blur. Top row shows repeatability rates, and bottom row shows number of correspondences. Left column is 'Bikes' sequence, right column is 'Trees' sequence.

Here $\mathbf{H}$ is the ground truth homography that transforms features between the two views, $\cap$ and $\cup$ are set (*i.e.*, region) intersection and union respectively, $|\cdot|$ is the area of a region, and $\mathcal{R}(\mu)$ is the set of image positions enclosed by the ellipse:

$$\mathcal{R}(\mu) = \{\mathbf{x} : (\mathbf{x}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m}) \leq 4\} \quad \text{with} \quad (16)$$

$$\mu = \frac{1}{4}\begin{pmatrix} \mathbf{C}^{-1} & -\mathbf{C}^{-1}\mathbf{m} \\ -\mathbf{m}^T\mathbf{C}^{-1} & \mathbf{m}^T\mathbf{C}^{-1}\mathbf{m}-4 \end{pmatrix}. \quad (17)$$

Here $\mathbf{m}$ and $\mathbf{C}$ are the centroid and inertia matrix of the region. Two regions are counted as corresponding if $\epsilon_o < 40\%$, and the repeatability score is computed as the ratio between the number of region-to-region correspondences and the smaller of the number of regions in the pair of images. Only regions located in the part of the scene present in both images are considered [10].

3.2. Sequences

The first two sequences (called 'Graffiti' and 'Wall') test stability across view-point change. Figure 8 shows the repeatability rates and number of correspondences found for the different methods.

The following two sequences (called 'Bikes' and 'Trees') test stability across de-focus blur. Figure 9 shows the repeatability rates and number of correspondences found for the different methods.

As can be seen, the MSCR detector has a better repeatability than MSER in sequences 'Wall' and 'Bikes', they tie in 'Trees', and the MSER detector does better in the 'Graffiti' sequence. The MSER3 method adds more correspondences to MSER, but the repeatability is not affected (it improves in two sequences, but degrades in two). The MSCR detector also gives more correspondences, except in the 'Trees' sequence. We can also see that the other colour based detector, the Blob feature detector, does significantly worse. With the exception for the 'Graffiti' sequence, we do consistently better. The reason for this is probably that the Blob feature detector is based on a segmentation. Thus, whenever the choice arises whether to merge two regions or not, the algorithm has to choose. The MSER and MSCR algorithms, on the other hand, will output both alternatives.

Four more sequences are available at [18]. These test stability across zoom+rotation, shutter changes, and JPEG compression. They are not included here for lack of space. The results for those sequences are similar, with the exception for the JPEG artifacts, which the MSCR algorithm appears to be more sensitive to than the other methods.

3.3. Two layer scenes

The repeatability tests performed by Mikolajczyk *et al.* [10] are quite useful for comparing detectors that are used to match planar, and near planar scenes. However, the test does not reveal how sensitive a detector is to background interference. A low background interference is important for 3D scene matching, and recognition of non-planar objects. Since we suspect that the use of colour would help discriminate foreground from background, we will now extend the repeatability test to two layer scenes, see figure 10.

We generate a two layer scene, by controlled blending of the two scenes 'Graffiti' and 'Wall', which are available online at [18]. First, we generate an opacity mask for the foreground layer, starting from a 100×75 image composed of uniformly distributed random numbers in range $[0, 1]$. This image is first blurred with a Gaussian kernel, with $\sigma = 2.4$,

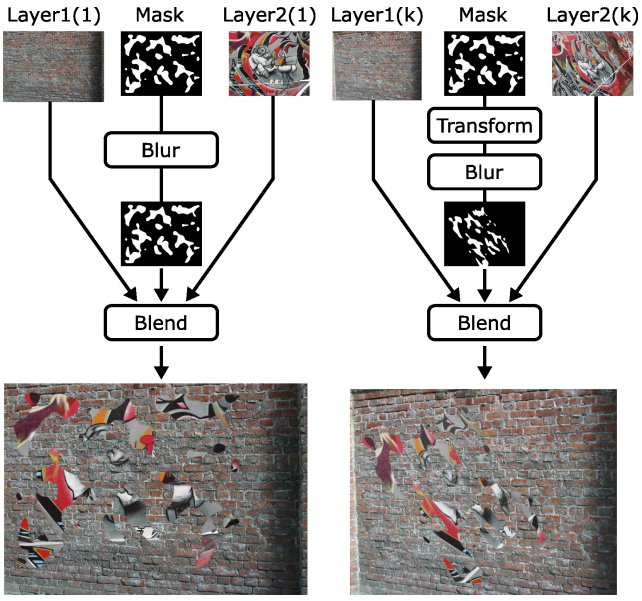

Figure 10. Generation of a two layer scene with ground truth.

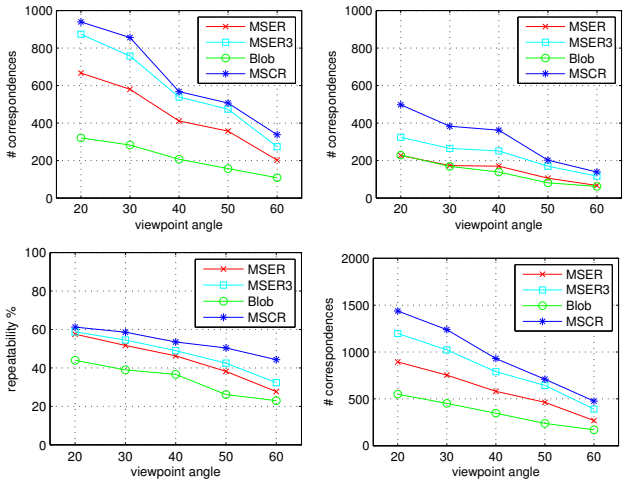

Figure 11. Repeatability rates and number of correspondences for viewpoint change on two-layer scene. Top row shows number of correspondences in background and foreground layers respectively. Bottom row shows repeatability and total number of correspondences.

and then up-sampled to the resolution of the foreground layer, and finally thresholded such that 25% of the pixels are ones. For all images except the first one, we then transform the mask using the ground truth homography for the foreground layer, and blur it with a Gaussian with $\sigma = 1.0$, see figure 10. Finally we blend each of the colour bands in the two layers using a weighted sum:

$$m_k(\mathbf{x}) = (1-\text{mask}(\mathbf{x}))L_{1,k}(\mathbf{x}) + \text{mask}(\mathbf{x})L_{2,k}(\mathbf{x})\,. \quad (18)$$

Figure 11 shows the repeatability rates and number of correspondences for the two-layer scene shown in figure

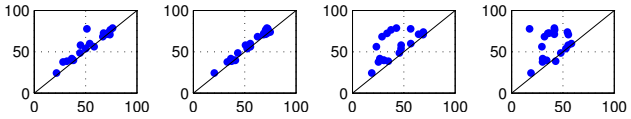

Figure 12. Discrimination plots for repeatability. Each point is the result of two repeatability tests on the same image pair, the vertical axis has the optimal feature enabled, and the horizontal has the feature disabled or replaced with a competing feature. Left to right: edge measure, neighbour set, image blur, and no blur.

10. The total repeatability has been computed as the sum of correspondences for the two layers, divided by minimum of the number of features in the overlapping regions of the two images. It is interesting to compare these results to the sequences in figure 8, since the two-layer scene is composed of these sequences. We can see that the number of correspondences for MSCR relative to MSER is now significantly higher, it is now also consistently higher than for the MSER3 method. We can also see that the the foreground layer gives almost as many correspondences as the 'Graffiti' sequence. This is probably because the opacity mask actually creates many regions in the foreground by cutting up large regions of uniform colour. To conclude, these graphs show that MSCR gives a small but significant improvement in repeatability, and increases the number of correspondences by more than 50% compared to MSER.

3.4. MSCR variants

We have tried a number of different variants of the MSCR algorithm, and here we compare these to the edge blur method. In figure 12 we show discriminative repeatability graphs for the following cases:

1. edge measure (6) vs. (7)
2. neighbour set $\mathcal{N}_2$ vs. $\mathcal{N}_1$
3. edge blur vs. image blur.
4. edge blur vs. no blur

Each point in figure 12 corresponds to one of the 20 image pairs from the four sequences in section 3.2. As can be seen, most points are above the diagonal, and thus the chosen algorithm variant has a consistently higher repeatability than the other variants. We can also see that the choice of blur is the most crucial improvement, while the use of $\mathcal{N}_1$ instead of $\mathcal{N}_2$ can be recommended for speed, see table 1.

3.4.1 Edge Significance Measure

The two edge significance measures (6) and (7) differ slightly in which regions they allow the detector to find. These differences are best visible in images with both dark regions and highlights, as in figure 13. Here we have used a margin threshold of $m = 0.0015$ with (7), which gave 611 regions. For (6) we then picked a threshold such that

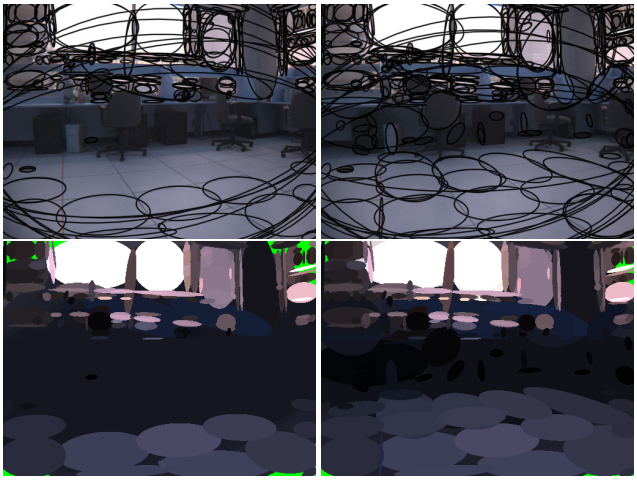

Figure 13. Comparison of Euclidean and normalised colour distances. Top row: detected blobs overlaid on a gamma-corrected input image. Bottom row: Detected regions (using a non gamma-corrected input) painted with their average colours. Left column: Euclidean distances, Right column: Normalised distances.

611 regions were also detected. Thus this figure should give an idea of in which *order* regions are ranked with the two measures. As can be seen in the figure, the Euclidean distance measure tends to generate more regions in highlight areas, while the normalised distance gives more regions in the shadows.

4. Discussion

In this paper we have implemented and tested a colour based feature detector, and compared it with the corresponding intensity-based detector, and a competing colour based detector. We have shown that our method has better repeatability in most scenes, and consistently gives a higher number of correspondences (more than a 50% increase in the two layer scene), while still being faster than most grey-scale based detectors. We have also designed a novel repeatability test that evaluates a detector's ability to separate foreground and background. In figure 2 we have also hinted at another possible advantage with the use of colour: better separation between object and shadow. An evaluation of this effect will however require a test scheme based on a full 3D scene, such as the one in [12].

Although we have chosen to describe the MSCR feature detector as a detector for finding affine-covariant regions, its use should not be limited to defining the affine frame for local image descriptors [9, 10]. As can be gathered from the illustration in figure 13, many scenes are actually recognisable in the blob representation, indicating that the detector output could actually be used as it is, perhaps as one of the features in a perception system of a robot.

Acknowledgements

The author thanks Jim Little and David Lowe for helpful discussions and comments. This work was supported by the Swedish Research Council through a grant for the project *Active Exploration of Surroundings and Effectors for Vision Based Robots*.

References

[1] J. J. Corso and G. D. Hager. Coherent regions for concise and stable image description. In *CVPR*, pages 184–190, 2005. 1, 2

[2] M. Couprie, L. Najman, and G. Bertrand. Quasi-linear algorithms for the topological watershed. *Journal of Mathematical Imaging and Vision*, 22(2-3):231–249, May 2005. 5

[3] M. Evans, N. Hastings, and B. Peacock. *Statistical Distributions*. John Wiley & Sons, Inc., 3rd edition, 2000. 3, 4

[4] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004. 5

[5] P.-E. Forssén and A. Moe. View matching with blob features. In *2nd CRV*, pages 228–235, May 2005. 1, 2

[6] J.-M. Geusebroek and A. Smeulders. Fragmentation in the vision of scenes. In *ICCV'03*, 2003. 4

[7] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1

[8] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *13th BMVC*, pages 384–393, September 2002. 1, 2

[9] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *TPAMI*, 27(10):1615–1630, 2005. 1, 8

[10] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005. 1, 5, 6, 8

[11] V. H. Milan Sonka and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Brooks/Cole Publishing Company, 2nd edition, 1999. 2, 5

[12] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3D objects. *IJCV*, 73(3):263–284, July 2007. 8

[13] P. M. Roth, M. Donoser, and H. Bischof. Tracking for learning an object representation from unlabeled data. In *CVWW*, pages 46–51, 2006. 2

[14] T. Tuytelaars and L. V. Gool. Wide baseline stereo matching based on local, affinely invariant regions. In *BMVC2000*, Bristol, September 2000. Invited Paper. 1

[15] R. Unnikrishnan and M. Herbert. Extracting scale and illuminant invariant regions through colour. In *17th British Machine Vision Conference*, September 2006. 1

[16] J. van de Weijer and C. Schmid. Coloring local feature extraction. In *ECCV*, 2006. 1

[17] Web-site. `http://www.isy.liu.se/~perfo/software/`, 2005. 2, 5

[18] Web-site. `http://www.robots.ox.ac.uk/~vgg/research/affine/`, 2005. 5, 6