

# Autonomous Learning of Object Appearances using Colour Contour Frames

Per-Erik Forssén  
Laboratory for Computational Intelligence  
Department of Computer Science  
2366 Main Mall  
BC, V6T 1Z4 Canada  
perfo@cs.ubc.ca

Anders Moe  
Computer Vision Laboratory  
Department of Electrical Engineering  
Linköping University  
SE-581 83 Linköping, Sweden  
moe@isy.liu.se

## Abstract

*In this paper we make use of the idea that a robot can autonomously discover objects and learn their appearances by poking and prodding at interesting parts of a scene. In order to make the resultant object recognition ability more robust, and discriminative, we replace earlier used colour histogram features with an invariant texture-patch method. The texture patches are extracted in a similarity invariant frame which is constructed from short colour contour segments. We demonstrate the robustness of our invariant frames with a repeatability test under general homography transformations of a planar scene. Through the repeatability test, we find that defining the frame using ellipse segments instead of lines where this is appropriate improves repeatability. We also apply the developed features to autonomous learning of object appearances, and show how the learned objects can be recognised under out-of-plane rotation and scale changes.*

## 1. Introduction

During recent years there has been great progress in the field of 3D object recognition. Appearance based techniques such as SIFT[13] and local affine frames[17, 20] now allow automatic modelling and recognition of a wide range of 3D objects at real time speed. These methods essentially rely on finding correspondences between textured patches defined around intensity extrema, and thus have problems with texture-free objects where few extrema per object are found, and the resultant frames will tend to cover both object and nearby background, and thus be sensitive to interference.

In our work we recently encountered a need for automatic modelling and recognition of objects of uniform

colour and simple geometrical shape (see figure 1). Recognition of this class of objects has traditionally been accomplished using model based approaches, see e.g. [12, 11]. In the model based approach a geometrical object model is postulated and fitted to the image data. We would like to have the “model free” property of the appearance based approaches, but need to deal with texture-free objects. This lead us the use of invariant frames constructed from short colour contour segments. Our approach gives us several invariant frames per object, and the obtained frames will tend to have a smaller scale than the objects. A related approach is presented in [19], where contours are also used to define invariant frames. They look for the *longest* contour chains in an image, and define their frames using the two end-points of each contour chain.

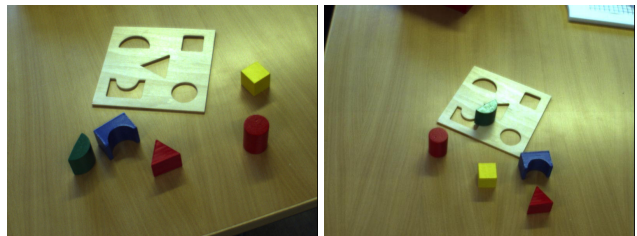


Figure 1. Scene examples.

We intend to use our system in a *developmental robotics* setting [14], where the robot shall adapt to its environment through experimentation. This setting allows us to have our robot prodding or poking at the environment and make use of rigidity of objects to decide what belongs together. A collection of image features that reside in the rigid motion region can then be associated with current object type and position. This way to autonomously discover objects and learn their appearances was introduced by Metta and Fitzpatrick in [15]. We extend their work here, by replacing

their colour histogram features with a collection of similarity invariant texture patches defined around contour segments.

### 1.1. Organisation

This paper is organised as follows. We start by giving an overview of the proposed feature detection algorithm. Then we describe each of the computational steps in detail. We then demonstrate the stability of the detection process with respect to scale and view angle changes, using a repeatability test. Finally we employ the developed features for learning of object appearances, and demonstrate that objects which have been manipulated by the system can successfully be recognised in single frames throughout an image sequence.

## 2. Algorithm overview

The algorithm for generating contour features consists of the following steps:

1. Detect colour edges and their orientations
2. Extract curve segments
3. Split curve segments into lines
4. Combine compatible line segments into ellipse segments where appropriate
5. Define a similarity invariant frame from each line or ellipse segment
6. Extract a gradient patch using the invariant frame.

### 2.1. Colour edge detection

For edge detection we make use of a slightly modified Canny edge detector [1]. Instead of applying Gaussian derivative filters to a grey-scale image, we make use of the colour information available as suggested by Jähne [9]. First we compute gradient vectors in each of the three colour-bands separately

$$\mathbf{z}_k(\mathbf{x}) = \begin{pmatrix} g_x * I_k \\ g_y * I_k \end{pmatrix}(\mathbf{x}), \text{ for } k \in \{1, 2, 3\}. \quad (1)$$

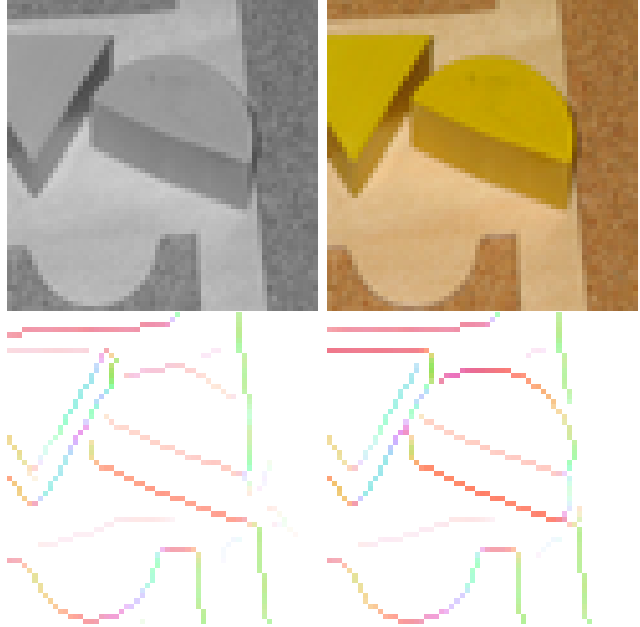
then we combine them into a *structure tensor*

$$\mathbf{T}(\mathbf{x}) = \sum_{k=1}^3 \mathbf{z}_k(\mathbf{x})\mathbf{z}_k(\mathbf{x})^T. \quad (2)$$

Finally, the orientation information, now in *double angle representation* is obtained as [6]:

$$\mathbf{z}(\mathbf{x}) = \begin{pmatrix} t_{11}(\mathbf{x}) - t_{22}(\mathbf{x}) \\ 2t_{12}(\mathbf{x}) \end{pmatrix}. \quad (3)$$

To demonstrate that we really benefit from the use of colour edges, we show a comparison between colour and grey-scale Canny output in figure 2.



**Figure 2. Top row: grey-scale and colour images. Bottom row: detected edges in grey-scale, and colour images. Double angle orientation is shown in smooth transitions between red for horizontal, green for vertical, and light blue and yellow for the diagonals.**

### 2.2. Curve segment extraction

Curve segments are extracted from the Canny output, by connecting nearby pixels in the edge image, if they have compatible positions and gradient directions. The extraction process works sequentially, starting in positions with large gradient magnitude, making use of the gradient direction information to decide in which directions to proceed.

### 2.3. Split curve segments into lines

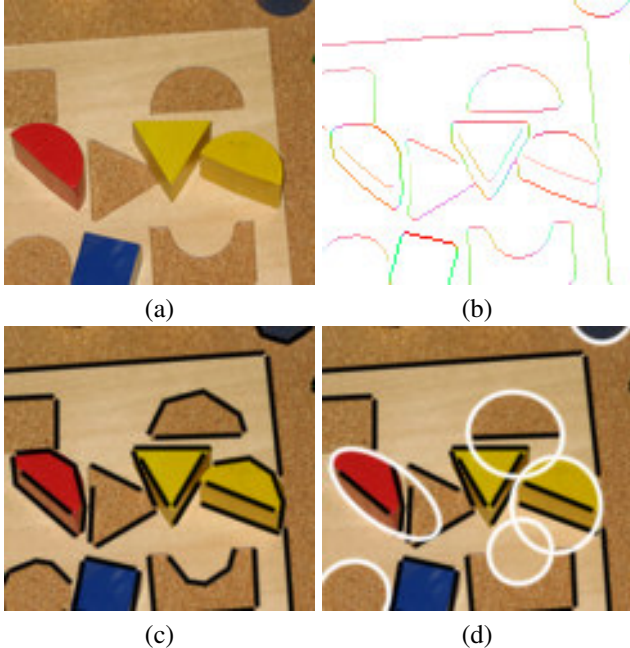
The next step is to split the curve segments into a number of short line segments. This is done by trying to fit line equations to runs of consecutive contour points. The inliers are then removed from the list and the process is repeated.

The output of this step is shown in figure 3c. Each line segment is defined by its centre point  $\mathbf{m}_k$ , length  $l_k$ , and normal angle  $\varphi_k$

$$\mathcal{L}_k = \langle \mathbf{m}_k, l_k, \varphi_k \rangle. \quad (4)$$

For each line segment, we also store the list of edge pixel coordinates that defines it. This data will be used later in the ellipse segment estimation step.

An earlier account of a similar method for line segment extraction, as well as an overview of related methods is given by Nelson in [16]. Our line extraction code could most likely be replaced by the one by Nelson with similar results.



**Figure 3. Steps in feature extraction. (a) colour image, (b) detected colour edges, (c) detected line segments superimposed on input image, (d) detected line and ellipse segments superimposed on input image.**

### 2.4. Combine line segments into ellipse segments

In order to find ellipses, we first generate a list of line segment pairs. The list only contains adjacent line segments (smallest end-point distance below 4 pixels) with orientations that differ by between  $8^\circ$  and  $80^\circ$ .

We then try to combine each line segment pair with each of the remaining line segments with a distance below 100 pixels<sup>1</sup> from the average line centre. The two lines form a corner, and we can restrict the search for a matching segment to the semi-circle which the corner is pointing away from. Using the three line segments we then try to fit an ellipse to the set of edge points that define them. The least squares fit is made using the Halir and Flusser method [7] which is a modification of the Fitzgibbon method [3].

<sup>1</sup>This speeds up the algorithm considerably, and essentially puts a limit on how large ellipses we will find.

For each ellipse estimate, the residuals of the constituent points are estimated. We then compute the residuals for the remaining line segments, and add those with small residual. Finally we re-estimate the ellipse using all points in the inlying line segments.

From the Halir and Flusser method we obtain a conic matrix  $\mathbf{C}$ , which we convert into a centroid  $\mathbf{m}$  and inertia matrix  $\mathbf{I}$ :

$$\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix}^T \mathbf{C} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = 0 \quad \Leftrightarrow \quad (\mathbf{x} - \mathbf{m})^T \mathbf{I}^{-1} (\mathbf{x} - \mathbf{m}) = 4. \quad (5)$$

Each ellipse segment is now defined by:

$$\mathcal{E}_k = \langle \mathbf{m}_k, \mathbf{I}_k \rangle. \quad (6)$$

### 2.5. Invariant frame construction

For each ellipse and line segment, we construct a feature vector  $\mathbf{a}$ , by sampling the double angle gradient (3) in an invariant frame. The frame is defined using a *similarity transform*  $\mathbf{S}$  [8], which maps a point  $\mathbf{p} \in \mathbb{R}^2$  to a point  $\tilde{\mathbf{p}} \in \mathbb{R}^2$  as

$$\tilde{\mathbf{p}} = \mathbf{S} \begin{pmatrix} \mathbf{p} \\ 1 \end{pmatrix}. \quad (7)$$

For a line  $\mathcal{L}_0 = \langle \mathbf{m}, l, \varphi \rangle$ , we define

$$\mathbf{S} = \frac{1}{l} (\mathbf{R} \quad -\mathbf{R}\mathbf{m}) \quad \text{where } \mathbf{R} = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}. \quad (8)$$

For an ellipse  $\mathcal{E}_0 = \langle \mathbf{m}, \mathbf{I} \rangle$ , we construct a line between the two points on the perimeter that lie along the major axis, see figure 4, left.

$$\mathcal{E}_0 \rightarrow \mathcal{L}_0 \equiv \langle \mathbf{m}, 4\sqrt{\lambda_1}, \arg(\mathbf{e}_1) \rangle$$

where  $\mathbf{I} = \lambda_1 \mathbf{e}_1 \mathbf{e}_1^T + \lambda_2 \mathbf{e}_2 \mathbf{e}_2^T$ ,  $\lambda_1 \geq \lambda_2$  is the eigenvalue factorisation of  $\mathbf{I}$ . Such a line will of course be quite arbitrary when the ellipse is close to being a circle. Except for this degenerate case, the frame defined from such a line will be quite useful for constructing a feature vector.

Since the direction of a line is ambiguous, both line directions  $\varphi$  and  $\varphi + \pi$  are used during the prototype generation, giving two prototypes for each reference line.

### 2.6. Feature vector construction

Once we have the similarity mapping, the feature vectors can be obtained by sampling the image orientations according to the similarity transforms, see figure 4, right. To avoid aliasing the sampling is made using a scale pyramid. Each orientation sample  $\mathbf{z}$  (see (3)) is rotated to the coordinate

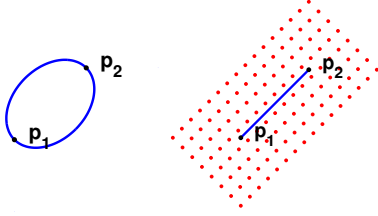


Figure 4. Left: Two points on an ellipse define a reference line. Right: Sampling grid around reference line.

system of the reference line and then *channel encoded* (see next section) with 8 channels to form a vector  $\mathbf{o}$ :

$$\mathbf{o} = |\mathbf{z}| \text{enc}(\arg(\mathbf{z})). \quad (9)$$

By stacking such vectors for the entire patch we get a  $8 \times 16 \times 8 = 1024$  element feature vector

$$\mathbf{a} = \begin{pmatrix} \mathbf{o}_1 \\ \mathbf{o}_2 \\ \vdots \\ \mathbf{o}_{16 \times 8} \end{pmatrix}. \quad (10)$$

### 2.7. Channel encoding

The *channel representation* is a way to represent single and multiple values with associated confidences, in a unified manner [4, 5]. The channel encoding mentioned at (9) converts a single scalar value  $x$  into a vector  $\mathbf{x}$ , by passing  $x$  through a set of regularly placed kernel functions  $B_k(\cdot)$ , optionally weighting them with a confidence  $r$ ,

$$\mathbf{x} = r \cdot \text{enc}(x) = r \cdot (B_1(x) \ B_2(x) \ \dots \ B_K(x))^T. \quad (11)$$

See figure 5 for an example of a periodic 12-channel set. Note that the kernels continue on the other side of the discontinuity, and thus avoid wrap-around problems in the representation. A more extensive discussion on channel representations is out of the scope of this paper; the reader is instead directed to [4] or [5]. We channel encode our feature vectors to change the feature space metric. After channel encoding, features more than one kernel width apart will have zero correlation.

### 3. Repeatability test

We will now demonstrate how the repeatability of the features is affected by view changes. To do this in a controlled fashion, we make use of a high resolution photograph ( $2592 \times 1944$  pixels) of a shape-sorter puzzle. We place a synthetic camera with focal length  $f = 200$  above

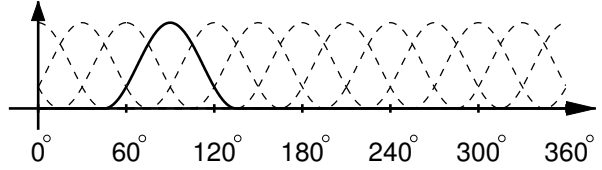


Figure 5. Example of a periodic arrangement of channel kernel functions. Kernel at  $90^\circ$  is shown in solid, other kernels are dashed.

the image, at varying angles and distances (scale). We then compute feature representations of both the synthesised view ( $300 \times 300$  pixels), and of the photograph, see figure 6.

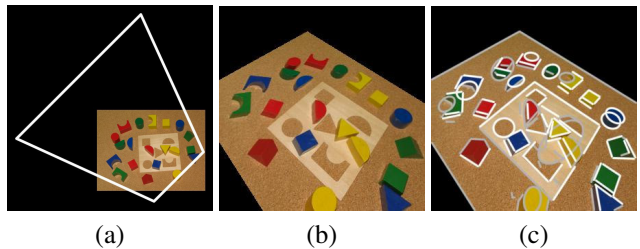


Figure 6. Experiment setup. (a) Photograph with homography quadrangle overlaid. (b) Synthesised view. (c) Synthesised view with detected features painted in. White features are valid correspondences, grey features are discarded.

We then transform the features in the synthetic view to the coordinate system of the photograph, and vice versa, and compute the position and shape distances between features. Correspondences falling below a threshold in position and shape are counted, and are used to determine the *repeatability rate* of the detector.

The *repeatability rate* [18] concept was originally used to compare interest point detectors, by computing the frequency with which an interest point detector gives repeated detection of the same 3D point in different 2D projections of the scene. We now extend the repeatability rate definition to our features, by also considering the parameters of the detected features. For the ellipse and line segment features we compute the repeatability rates as:

$$r_{\text{ellipse}}(\sigma_p, \sigma_s) = \frac{N_c(\sigma_p, \sigma_s)}{\min(N_p, N_s)} \quad \text{and} \quad r_{\text{line}}(\sigma_p, \sigma_s) = \frac{M_c(\sigma_p, \sigma_s)}{\min(M_p, M_s)}. \quad (12)$$

Here  $N_p$  and  $N_s$  are the number of detected ellipses in the overlapping parts of the photo and the synthesised view respectively, and  $N_c$  is the number of correspondences found.

Similarly the number of detected lines in the images are given by  $M_p$  and  $M_s$ , and  $M_c$  is the number of correspondences found.

The explicit formula for  $N_c$  is

$$N_c(\sigma_p, \sigma_s) = |\{ \langle \mathcal{E}_i, \mathcal{E}'_j \rangle | d(\mathbf{m}_i, \mathbf{m}'_j)^2 / \sigma_p^2 + d(\mathbf{I}_i, \mathbf{I}'_j)^2 / \sigma_s^2 < 1 \}|. \quad (13)$$

The parameters  $\sigma_p$  and  $\sigma_s$  are position and shape distance thresholds. We have used  $\sigma_p = 7$  and  $\sigma_s = 0.3$ , which gives visually quite similar ellipses after transformation.

The explicit formula for  $M_c$  is

$$M_c(\sigma_p, \sigma_s) = |\{ \langle \mathcal{L}_i, \mathcal{L}'_j \rangle | d(\mathbf{m}_i, \mathbf{m}'_j)^2 / \sigma_p^2 + d(\varphi_i, \varphi'_j)^2 / \sigma_s^2 < 1 \}|. \quad (14)$$

The parameters  $\sigma_p$  and  $\sigma_s$  are centre position and normal direction distance thresholds. We have used  $\sigma_p = 4$  and  $\sigma_s = 5^\circ$ , which gives visually quite similar lines after transformation.

### 3.1. Distance measures

For completeness, we will here define the distance measures used in (13) and (14). For positions  $\mathbf{m}_i$  and  $\mathbf{m}'_j$ , we define the distance as

$$d(\mathbf{m}_i, \mathbf{m}'_j)^2 = (\mathbf{m}_i - \tilde{\mathbf{m}}'_j)^T (\mathbf{m}_i - \tilde{\mathbf{m}}'_j) + (\tilde{\mathbf{m}}_i - \mathbf{m}'_j)^T (\tilde{\mathbf{m}}_i - \mathbf{m}'_j). \quad (15)$$

Here  $\sim$  denotes that an element has been transformed to the other view (the photograph, or the synthesised view) by the appropriate homography mapping, see [8]. Note that we apply the transformation both ways in order to remove possible bias favouring one direction of the transform.

For two inertia matrices  $\mathbf{I}_i$  and  $\mathbf{I}'_j$ , we define the distance as:

$$d(\mathbf{I}_i, \mathbf{I}'_j) = \frac{\|\mathbf{I}_i - \tilde{\mathbf{I}}'_j\|}{\|\mathbf{I}_i\| + \|\tilde{\mathbf{I}}'_j\|} + \frac{\|\tilde{\mathbf{I}}_i - \mathbf{I}'_j\|}{\|\tilde{\mathbf{I}}_i\| + \|\mathbf{I}'_j\|}. \quad (16)$$

Mapping of an inertia matrix through a homography is accomplished by mapping the entire conic form, see [8].

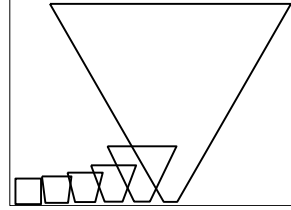
For two normal angles  $\varphi_i$  and  $\varphi'_j$ , we define the distance as

$$d(\varphi_i, \varphi'_j)^2 = \arg \left\{ e^{i2\varphi_i} e^{-i2\tilde{\varphi}'_j} \right\}^2 + \arg \left\{ e^{i2\tilde{\varphi}_i} e^{-i2\varphi'_j} \right\}^2. \quad (17)$$

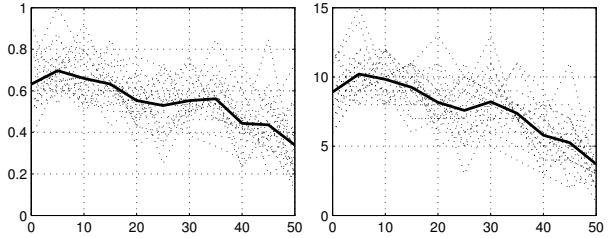
Note the doubling of the angle in order to avoid the modulo  $\pi$  ambiguity.

### 3.2. Repeatability results

Figure 7b and 7d show how the repeatability is affected by the inclination angle. The angle is varied in the range  $0 \dots 50^\circ$  which results in homography quadrangles according to figure 7a. In figures 7d and 7e we also see a comparison between keeping and removing line segments that are used to define ellipses. As can be seen, and as should be intuitively evident, detected line segments that actually are parts of an ellipse are not as stable as those that represent actual lines.

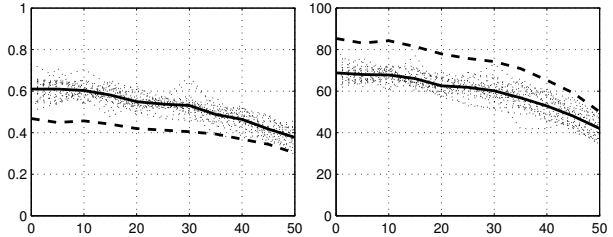


(a) Quadrangles



(b) Ellipse repeatability

(c) #Correspondences



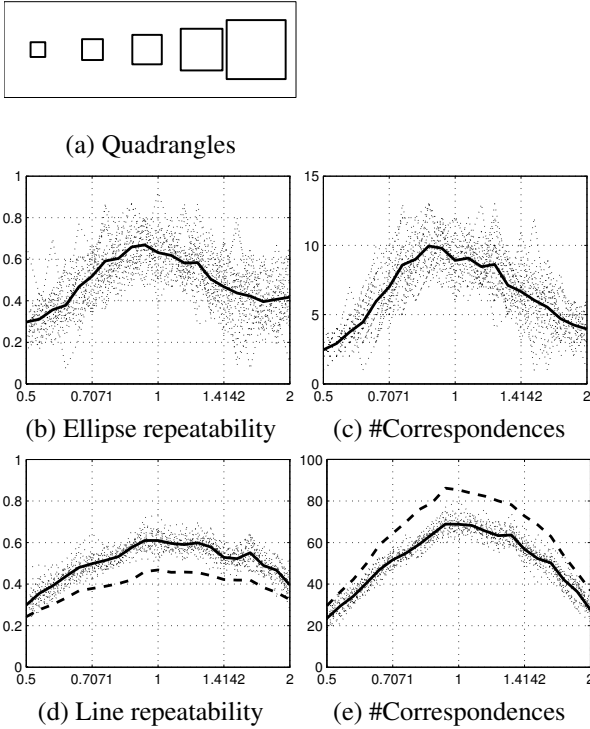
(d) Line repeatability

(e) #Correspondences

**Figure 7. (a) Homography quadrangles for inclination angles  $0 \dots 50^\circ$ . (b,d) Repeatability as function of inclination angle. (c,e) Number of correspondences as function of inclination angle. Solid curves are averages over all in-plane rotations. Dashed curves in (d) and (e) show results when line segments that make up the ellipses are also included.**

Figure 8b and 8d show how the repeatability is affected by scale change. The scale is varied in the range  $0.5 \dots 2$  which results in homography quadrangles according to figure 8a. In figures 8d and 8e we again show a comparison between keeping and removing line segments that are used

to define ellipses. As can be seen, the results are the same as in the inclination angle experiment: detected line segments that represent parts of an ellipse are not as stable as those that represent actual lines. The repeatability across scale could probably be improved by replacing Canny with a multi-scale edge detection algorithm.



**Figure 8.** (a) Homography quadrangles for scales 0.5 to 2. (b,d) Repeatability as function of scale. (c,e) Number of correspondences as function of scale. Solid curves are averages over all in-plane rotations. Dashed curves in (d) and (e) show results when line segments that make up the ellipses are also included.

## 4. Learning object recognition

Learning to recognise three dimensional objects is greatly enhanced by the ability to manipulate the objects yourself, especially by means of prodding and later grasping. Metta and Fitzpatrick demonstrate this for the robot COG [15]. By allowing the robot to prod at objects it can directly find out what parts of the scene move together, and thus obtain an object/background segmentation. In order to do this in a controlled fashion we settle for a non-robotic equivalent here: We take a sequence of still images of the

shape-sorter puzzle in figure 1, and move one of the pieces between each pair of frames, see figure 9.

### 4.1. Figure/ground segmentation

For each training image pair (see figure 9a,b) we then perform a colour image segmentation and measure the amount of changes within each region. All regions with more than a given percentage of changing pixels are said to belong to a *change mask*. Note that most segmentation algorithms will work here, provided that they can be tuned to over-segment rather than under-segment the image.

We now want to remember the appearance of the object within the change mask. This will allow us to recognise the moved object also without prodding it.

### 4.2. Learning phase

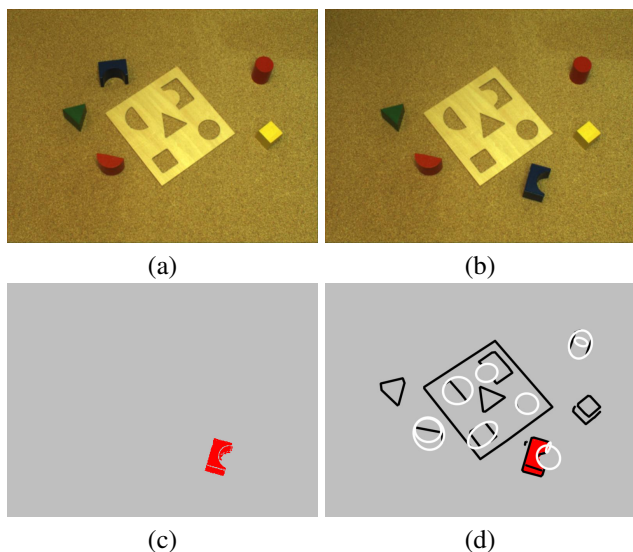
Learning in this system consists of generation and storage of a set of prototype vectors for each object. To generate the prototype vectors we move and rotate each object we want to recognise, see figure 9. Lines and ellipses are estimated from the images and a change mask is constructed between each consecutive image pair in the sequence (see figure 9c,d). For each line and ellipse segment inside the change mask a feature vector  $\mathbf{a}_{2p}$ , and a  $180^\circ$  rotated copy  $\mathbf{a}_{2p+1}$  are constructed and stored in a matrix  $\mathbf{A}$ . The two copies are needed to resolve the  $180^\circ$  ambiguity in the invariant frame construction, see section 2.5. Then a vector pointing from the line centre to the object centre together with the object type is stored in a matrix  $\mathbf{U}$ . The object centre is defined by the centre of gravity of the change mask in the line's coordinate system, and the object type is the number (1 . . . 5) of the puzzle piece that is moved. These under the learning phase constructed vectors are called prototype vectors and the vectors constructed during recognition are called query vectors.

### 4.3. Recognition phase

Now, we want to detect and recognise the trained objects in new images without a change mask. This is done by calculating a feature vector  $\mathbf{a}_q$  for each line segment in the image and match these with the prototype features. The *match score*  $M$  is computed as a normalised correlation:

$$M(\mathbf{a}_p, \mathbf{a}_q) = \frac{\mathbf{a}_q^T \mathbf{a}_p}{\|\mathbf{a}_q\| \|\mathbf{a}_p\|}. \quad (18)$$

For each query vector  $\mathbf{a}_q$  the  $N$  best matches are picked (we typically use  $N \in \{3 \dots 10\}$ ), and the stored object positions in  $\mathbf{U}$  corresponding to these matches are transformed to the new line's coordinate system. This gives  $N$  votes on



**Figure 9. Training data example.** Arch piece has moved between frames (a) and (b). (c) shows the resultant change mask, and (d) shows the change mask with features from (b) superimposed.

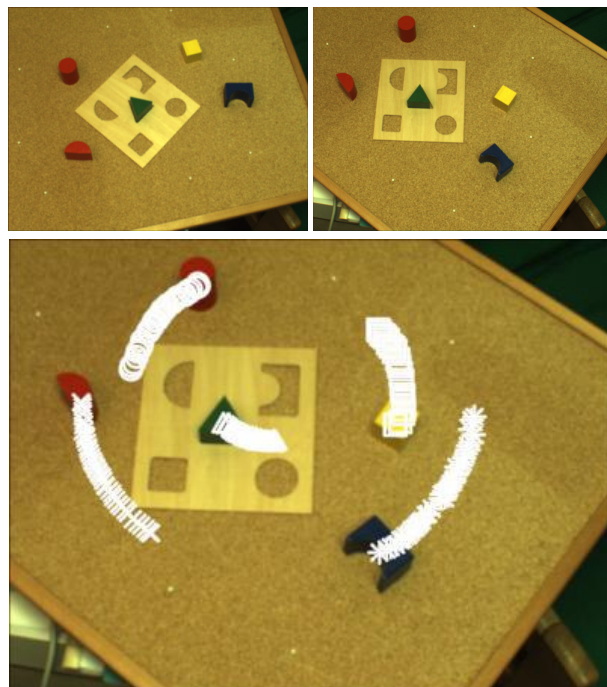
object position and type for each line segment in the image, the votes are weighted with the match scores  $M$ . The votes are then clustered using mean-shift clustering [2]. Significant clusters indicate recognised objects and the cluster positions give the position and type of the recognised objects.

Since the pieces are moved by hand in the used training sequence we have no information about the object's 3D rotation and scale. Had the robot moved the pieces itself, this would be available as internal (or *proprioceptive*) sensory information, and could be included in the clustering to obtain the pose (3D position and orientation) of the objects [10].

#### 4.4. Recognition evaluation

During training, each puzzle piece is moved and rotated ten times. Thus, we have about  $36^\circ$  rotation between two adjacent views. This gave a feature matrix  $\mathbf{A}$  with 420 feature vectors, which means about  $420/(10 \times 5) \approx 8$  features on average for an object in an image. Note that in a real robotic setting the robot should decide for itself when it has enough prototypes to reliably recognise an object.

As a simple way to evaluate the recognition system (without designing an explicit ground truth) we now place the pieces on a board, and move and rotate it in a continuous manner. For each frame in the resultant sequence we perform the recognition step, 56 frames in total. The result of this is shown in figure 10. As can be seen in the figure,

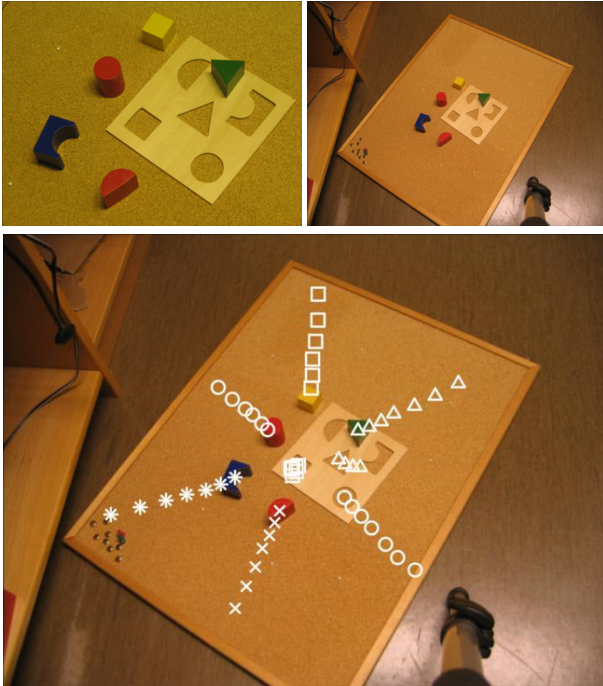


**Figure 10. Recognition results for out-of-plane rotation.** Top row: First and last frame in sequence. Bottom: Last frame with detection results. Detections with confidence above a threshold are plotted with a symbol corresponding to the object type.

most objects are recognised in most of the frames. A similar sequence with zoom instead of rotation and translation is shown in figure 11. This sequence (7 frames) is taken with a different camera and with different illumination and the total zoom in the sequence is about 1.5 octaves. The scaling between the last frame and the training images is about 0.2 octaves.

#### 4.5. Concluding remarks

We have introduced a novel way of computing invariant frames from colour contour descriptions and shown that they can be used in object recognition. We have demonstrated the robustness of the features with respect to scale and inclination angle changes in a repeatability test. We have also designed a simple object recognition system that makes use of the features to recognise some simple 3D shapes. This system was a first test of the features before using them for full 3D-pose object recognition on a robotic platform. For the system to be useful in more general object recognition settings we believe it is necessary to include other classes of invariant frames, such as SIFT and MSER as well. This is indeed the next step on our agenda.



**Figure 11. Recognition results for zoom. Top row: First and last frame in sequence. Bottom: Last frame with detection results. Detections with confidence above a threshold are plotted with a symbol corresponding to the object type.**

#### 4.6. Acknowledgements

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*, and the European Community through EC Grant IST-2003-004176 COSPAL. It reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein.

#### References

[1] J. Canny. A computational approach to edge detection. *IEEE TPAMI*, PAMI-8(6):255–274, November 1986.

[2] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE TPAMI*, 17(8):790–799, August 1995.

[3] A. Fitzgibbon, M. Pilu, and R. Fisher. Direct least square fitting of ellipses. *IEEE TPAMI*, 21(5):476–480, May 1999.

[4] P.-E. Forssén. Low and medium level vision using channel representations, March 2004. Dissertation No. 858, ISBN 91-7373-876-X.

[5] G. H. Granlund. An associative perception-action structure using a localized space variant information representa-

tion. In *Proceedings of Algebraic Frames for the Perception-Action Cycle (AFPAC)*, Kiel, Germany, September 2000.

[6] G. H. Granlund and H. Knutsson. *Signal Processing for Computer Vision*. Kluwer Academic Publishers, 1995. ISBN 0-7923-9530-1.

[7] R. Halir and J. Flusser. Numerically stable direct least squares fitting of ellipses. In *Proc. of the 6th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG'98)*, volume 1, pages 125–132, Plzen, Czech Republic, 1998.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[9] B. Jähne and H. Haussecker, editors. *Computer Vision and Applications*. Academic Press, 2000. ISBN 0-12-379777-2.

[10] B. Johansson and A. Moe. Patch-duplets for object recognition and pose estimation. In *2nd Canadian Conference on Robot Vision*, pages 9–16, Victoria, BC, Canada, May 2005. IEEE Computer Society.

[11] H. Kollnig and H.-H. Nagel. 3D pose estimation by directly matching polyhedral models to gray value gradients. *International Journal of Computer Vision*, 23(3):283–302, 1997.

[12] D. Lowe. Fitting parametrized three-dimensional models to images. *IEEE TPAMI*, 13(5):441–450, 1991.

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

[14] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics: A survey. *Connection Science*, 15(4):151–190, December 2003.

[15] G. Metta and P. Fitzpatrick. Early integration of vision and manipulation. *Adaptive Behavior*, 11(2):109–128, June 2003.

[16] R. C. Nelson. Finding line segments by stick growing. *IEEE TPAMI*, 16(5):519–523, May 1994.

[17] S. Obdržálek and J. Matas. Object recognition using local affine frames on distinguished regions. In *13th BMVC*, pages 113–122, September 2002.

[18] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *Int. Journal of Computer Vision*, 37(2):151–172, 2000.

[19] A. Selinger and R. C. Nelson. A perceptual grouping hierarchy for appearance-based 3d object recognition. *Computer Vision and Image Understanding*, 76(1):83–92, October 1999.

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