Appearance-based Keypoint Clustering

Francisco J. Estrada[†] Pascal Fua[‡] Vincent Lepetit[‡] Sabine Süsstrunk[‡]
[†]University of Toronto at Scarborough
1265 Military Trail, M1C 1A4, Toronto, On.
Canada
[‡]Ecole Polytechnique Fédérale de Lausanne
EPFL - I&C, CH 1015, Lausanne
Switzerland
strider@cs.utoronto.ca, [pascal.fua,vincent.lepetit,sabine.susstrunk]@epfl.ch

Abstract

We present an algorithm for clustering sets of detected interest points into groups that correspond to visually distinct structure. Through the use of a suitable colour and texture representation, our clustering method is able to identify keypoints that belong to separate objects or background regions. These clusters are then used to constrain the matching of keypoints over pairs of images, resulting in greatly improved matching under difficult conditions. We present a thorough evaluation of each component of the algorithm, and show its usefulness on difficult matching problems.

1. Introduction

The fundamental motivation behind the continued development of low-level perceptual grouping algorithms is the belief that a correct organization of the scene into meaningful regions or feature groups will lead to more efficient and reliable algorithms for object detection, tracking, and scene reconstruction. Existing algorithms, however, have limited success. Current state-of-the-art methods produce good results on simple scenes, but fail on more realistic ones.

In this paper, we present a clustering algorithm that enables standard keypoint matching techniques to recover useful matches under very challenging conditions. We show that through the use of a well designed representation of colour and texture, a spectral embedding on a set of arbitrarily distributed image samples yields a meaningful set of clusters that captures the structure of the scene, typically separating objects of interest from the background and from each other. When applied to image patches that correspond to the location of detected interest points, the resulting clusters provide strong constraints for typical interest point matching algorithms, resulting in a significant reduction in the number of false matches. This process is illustrated in Fig. 1

*This work was partially supported by the PHAROS project funded by the European Commission under the 6th Framework Programme (IST Contract No. 045035).

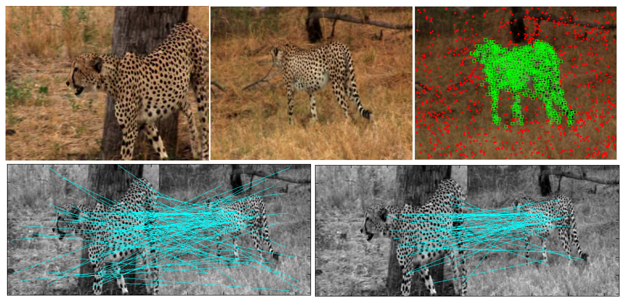
Figure 1. **Top row** from left to right: Reference image of a cheetah, target frame from a video sequence, and clusters of keypoints detected by our algorithm. bf Bottom row: The left side shows standard SIFT matching illustrating the difficulty of this problem. The right side shows the result of using our method to constrain SIFT matches to keypoints within specific clusters as described in the text. Our algorithm makes matching possible under very challenging conditions.

To support the case for our proposed method, each component of the algorithm is evaluated separately on images from the Berkeley Segmentation Database (BSD) [16]. We also present a clustering comparison with the normalized cuts method [22], a standard graph-based clustering algorithm that is closely related to spectral embedding. The evaluation brings additional insights about the nature of the problem of clustering keypoints by appearance, and provides solid evidence that the proposed framework is sound.

Having evaluated the performance of our algorithm on a standard image database, we demonstrate its usefulness for object detection and tracking on a series of difficult matching problems characterized by clutter, non-uniform illumination changes, and non-rigid transformations. It is shown that the clustering process leads to significantly better matches, allowing for the tracking of lightly texture objects in heavily cluttered backgrounds, matching under conditions of significant perspective and non-rigid deformations, and matching under non-uniform illumination changes.

2. Feature grouping in context

The algorithm we propose is related to a wider class of methods in two fields of computer vision: perceptual organization and image segmentation. Perceptual organization (perceptual grouping) deals with the general problem of grouping a set of observed features into meaningful sets. It is based on the Gestalt school of psychology [25], which studies how human observers organize visual stimuli. In computer vision, perceptual grouping has addressed the fundamental problem of extracting complete boundaries and surfaces from fragmented data [10], [26]. Image segmentation, on the other hand, has explored the problem of deciding which parts of the image have a consistent appearance. While it is often considered a separate field within computer vision, one could think of image segmentation as a special case of perceptual grouping, namely, that of grouping pixels or image patches according to their similarity of appearance.

The performance of state-of-the-art methods in both fields has improved over the years. Existing algorithms use complex appearance descriptors and powerful optimization and inference procedures (see [22], [7], and [19] to name just a few). However, current algorithms are still limited in their ability to deal with typical natural images depicting heterogeneous or textured objects against complex backgrounds. These limitations explain why, outside of medical imaging problems, grouping and segmentation methods have found few direct applications. It is arguably the case that the increasing complexity of object recognition and tracking algorithms arises in part because of the limitations of existing low-level vision methods.

Instead of focusing on the segmentation problem of extracting accurate object boundaries, we look at the grouping of a much smaller set of features: The keypoints detected on a given image. This has two principal advantages, first, it makes the clustering task simpler and less computationally demanding. Secondly, the clustering does not have to be exceedingly accurate in order to be useful. As long as the clustering method manages to reject most background clutter for a given object, standard robust matching techniques such as RANSAC will recover a useful match.

The framework we propose here is related to spectral clustering algorithms, such as normalized cuts [22] and its many extensions; to existing methods that take advantage of the steady-state properties of random walks [9]; and to previous work on cue combination [15] together with recent work in texture similarity measures [23]. We also find related work in the clustering method proposed in [8] for feature grouping and boundary extraction, and in recent patch-based frameworks for image representation [11], [12]. While this is not an exhaustive list, thorough reviews of image segmentation and perceptual grouping are available from [2]. Finally, we note that while the grouping of keypoints by appearance has not been properly studied before, the reverse problem (namely, the use of keypoint based object detection to improve segmentation results) is the topic of recent research efforts [27].

3. Representing Local Appearance

A critical component of any clustering method is the similarity measure that will govern the behaviour of the algorithm and the quality of the clusters. Even a good clustering algorithm working on weak similarity information can be expected to produce poor results. This section describes the colour and texture representation for local image patches, the similarity measure that provides suitable information for the clustering algorithm, and an evaluation of the proposed measure on test images from the BSD. An important observation derived from the evaluation is that the problem of grouping keypoint patches is in general harder than that of grouping pixels or uniformly distributed image patches.

For the rest of the paper we use the following notation: I is an input colour image, $\mathcal{P}$ is the set of local image patches extracted from I, $\vec{p}_i$ is the ith patch in $\mathcal{P}$ associated with image coordinates (x_i, y_i). The patch contains the pixel values for a square region of a specified width centered at (x_i, y_i) on image I. We assume that the patches have been detected through the use of a standard keypoint detector such as SIFT [13], but we note that the algorithm does not depend on a specific detector. In fact, it does not require square patches either. Arbitrarily shaped and sampled patches derived, for example, from superpixels could be used as well without modifications to the algorithm described below. For an extensive review and evaluation of keypoint detectors please see [18].

3.1. Colour and Texture Representation

The appearance representation we describe here was chosen for its simplicity and compactness. It does not require convolution of the image with large filter banks, nor the additional step of computing a suitable number of texton channels. Despite its simple form, we will show at the end of this section that it provides results that are competitive with regard to current (significantly more complex) state-of-the-art approaches.

We use standard colour histograms to represent the colour appearance of a local image patch. Histograms have the advantage of being easy to generate and having well studied similarity measures [21]. For simplicity, we use the RGB colourspace, but we note that the same procedure can be used in any suitable colourspace and can be easily extended to the analysis of multi-spectral imagery. We denote the colour histogram for a patch $\vec{p}_i$ as $\vec{H}_i$. This histogram is formed by computing 1-dimensional histograms for each available colour channel in patch $\vec{p}_i$, and then concatenating the individual histograms for each colour channel into a single vector. The size of the patch and the number of bins in the histogram are parameters of the algorithm.

To represent image texture, we use a recently proposed texture descriptor based on covariance matrices of image derivatives at a specific scale [23]. This formulation compares favourably to current state-of-the-art filter bank approaches, such as [24]; more importantly, it has been shown to provide good performance for object categorization. The texture descriptor C_i corresponds to the covariance matrix

$$C_i = \frac{1}{(N-1)} \sum_{j \in \vec{p}_i} (\vec{v}_j - \bar{\vec{v}})^T (\vec{v}_j - \bar{\vec{v}}), \tag{1}$$

where

$$\vec{v}_j = \left[\frac{\partial I}{\partial x} \;\; \frac{\partial I}{\partial y} \;\; \frac{\partial^2 I}{\partial x^2} \;\; \frac{\partial^2 I}{\partial y^2} \;\; \frac{\partial^2 I}{\partial xy} \right] \tag{2}$$

is the vector of first and second image derivatives for some pixel $j \in \vec{p}_i$ computed using standard finite difference approximations, $\bar{\vec{v}}$ is the mean vector over the patch, and N is the number of pixels in $\vec{p}_i$. While this measure has been used for full-image object matching and for texture discrimination before, it has, to our knowledge, not been incorporated into existing multi-cue visual classification algorithms.

3.2. Similarity Computation

Given the appearance descriptors detailed above, we can compute the similarity in terms of colour and texture between any two patches. The problem here is to determine an appropriate way to combine the two image cues of colour and texture. The chosen form of the similarity measure is common for image segmentation tasks, the novelty here is in replacing large filter-banks by the covariance-based texture measure described above, which achieves almost equal performance at a lower computational cost. Additionally, while common similarity measures typical of image segmentation research employ some form of edge energy [15], [5], we do not incorporate any edge-based component in our own similarity measure. This is due to the fact that keypoints typically occur on or near image boundaries and so the probability that the patch that contains the keypoint straddles an image edge is high.

We use a simple exponential combination of the available cues, and show that this simple measure yields good results on standard precision/recall discrimination tests over images from the BSD. The first step is to define proper similarity values for the colour and texture components of our appearance model. Colour similarity is straightforward. Based on the analysis and evaluation results for histogram similarity measures presented in [21] within the context of content based image retrieval, we use the χ^2 histogram distance given by

$$\chi^2_{ij} = \chi^2(\vec{H}_i, \vec{H}_j) = \sum_k \frac{(\vec{H}_i(k) - \hat{\vec{H}}(k))^2}{\hat{\vec{H}}(k)}, \tag{3}$$

where $\hat{\vec{H}}(k) = (\vec{H}_i(k) + \vec{H}_j(k))/2, k = 1 \ldots n$ is the average histogram of $\vec{H}_i$ and $\vec{H}_j$.

Texture similarity is measured as described in [23]. The idea behind this similarity measure is that the covariance matrix C_i describes a hyper-ellipsoidal distribution of points in a an m-dimensional space. The shape and orientation of this hyper-ellipsoid is described by the eigenvectors and eigenvalues of C_i. Texture similarity can be estimated by comparing the shape and orientation of the hyper-ellipsoids corresponding to two covariance matrices C_i and C_j. More specifically, similarity between C_i and C_j is estimated by solving the generalized eigenvalue problem $\lambda_k C_i \vec{u}_k - C_j \vec{u}_k = 0$ where $\lambda_k, k = 1 \ldots 5$ are the generalized eigenvalues and $\vec{u}_k \neq \vec{0}$ are the generalized eigenvectors of C_i and C_j. Texture similarity is then defined as

$$\rho_{ij} = \rho(C_i, C_j) = \sqrt{\sum_{k=1}^{5} \ln^2 \lambda_k} \tag{4}$$

which satisfies $\rho_{ij} \geq 0$, $\rho_{ij} = 0$ only if $C_i = C_j$, and also satisfies the triangle inequality.

Finally, we include an additional term in our similarity function that depends on the Euclidean distance between the centers of two patches. The final similarity function follows a common formulation from image segmentation research [15]. We write

$$a_{ij} = e^{-(\frac{\rho^2_{ij}}{\sigma^2} + \frac{(\chi^2_{ij})^2}{\sigma^2} + \frac{d^2_{ij}}{\sigma^2})}, \tag{5}$$

where d_{ij} is the Euclidean distance between patches $\vec{p}_i$ and $\vec{p}_j$. There is one free parameter, namely σ used to weight the cues. While we have elected to use a single σ in our affinity measure, particular tasks or problem domains may benefit from individually weighting each cue with a different σ. The values for these parameters should be learned from training data as discussed below.

To set the value of our parameters (σ, number of histogram bins, and patch size) and to evaluate the performance of the similarity measure, we follow the framework proposed in [5]. We choose $\sigma = .3$, patches of size 11×11, and 15 histogram bins as the values that maximize the *F-measure* over a set of 50 randomly selected training images from the BSD. The *F-measure* is given by $F(p, r) = (pr/(.5 * p + .5 * r)$ where p is the precision defined as the probability that two features declared to be in the same group are indeed in the same group, and r is the recall defined as the probability that a same-group pair is detected.

We then evaluate the performance of the similarity measure on the remaining 250 BSD images. We obtain $F = .5728$ for pixel classification with our formulation. In comparison, the optimized similarity measure proposed in [5], which is based on a large set of filters and a carefully selected group of texton channels achieves $F = .602$ using patch information only (we note that the latter value was computed over a significantly larger set of images than what the public distribution of the BSD contains, and that cues were computed using the CIE-Lab colourspace). We conclude that with regard to patch-based features our simpler similarity measure performs almost as well as the state-of-the-art measure proposed in [5]

Testing over the BSD also reveals that the task of grouping keypoint patches is significantly harder than that of clustering pixels. We processed the 250 BSD images not used during training with the SIFT keypoint detector [14], and recomputed the *F-measure* using only keypoint patches. The result yields $F = .5247$. This decrease can be explained by noting that keypoints are by definition located close to, or directly on top of object boundaries and areas of large image gradient. Patches obtained from these locations often contain data from more than one visually distinct region, making classification harder. In conclusion: keypoint clusters have soft boundaries (a fact that motivates the need for a good clustering method), and the reduced *F-measure* scores reflect this fact.

We end this section with two important observations regarding the evaluation above. First, in [5] a significantly better $F = .652$ is obtained by using an intervening contour cue based on boundary energy. Such a contour-based cue is appropriate for pixel segmentation, but not so for keypoint clustering since many keypoints are expected to span object boundaries. The second observation pertains the use of spatial distance as a cue for grouping. Results in [5] and our own tests show no meaningful improvement from the use of spatial information for pixel-wise classification. However, the distance term makes a significant difference for keypoint grouping. Without the distance term the *F-measure* for keypoints decreases from .5247 to only $F = .5054$.

4. Clustering Algorithm

We propose a two-stage clustering method that takes advantage of the good performance of our similarity measure. The first step in our method is based on the spectral embedding algorithm [4], which works on the principle that similarity between elements that belong in the same cluster can be strengthened through a random walk process, and that this process can be carried out efficiently in a low-dimensional subspace computed from the matrix of pairwise similarities between data points.

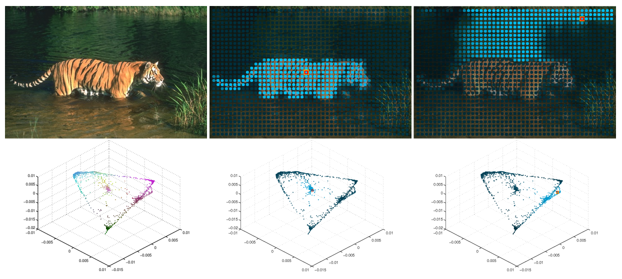
Figure 2. Left: Original image and first 3 dimensions of the embedding. Second and third columns: similarity of sampled image patches (blue dots) with regard to a selected patch shown in red, brightness is proportional to similarity. Proximity in the embedding space is equivalent to image-space similarity.

This property of enhancing the similarity of elements that belong in the same cluster is what differentiates this algorithm from standard spectral clustering methods such as Ncuts, and other dimensionality reduction techniques like PCA. The spectral embedding technique has been shown to yield better segmentations in a benchmark study of image segmentation algorithms [3], and our results here confirm its ability to produce better clusters than related techniques such as Ncuts.

The mathematical formulation and details of the spectral embedding process are given in the Appendix. The result of the embedding is a 5D representation of our input patches as a set of feature vectors. This representation is such that patches in the image that belong to the same object or appearance class form tight clusters. Figure 2 shows an input image, and the first 3 dimensions of the 5D embedding computed for uniformly sampled patches over this image. There are two things to note in the plots of the embedding: first, one can see a few dense regions with many feature vectors separated by sparsely populated areas. Each dense region corresponds to one of the appearance classes of the image (e.g. tiger, grass, water). Secondly, similarity of appearance is encoded in the embedding as spatial proximity. These two properties are typical of spectral embedding and contribute to making the clustering of feature vectors much easier.

Once we have a set of low-dimensional feature vectors from the embedding, we find clusters and assign a label to each patch. In the original embedding formulation, this was accomplished by proposing a set of seed regions corresponding to small sets of highly similar feature vectors, followed by a series of min-cut computations to determine the optimal cluster boundaries. This approach has two limitations: the amount of computation required by the many min-cut operations, and the fact that clusters could be arbitrarily split if more than one seed region was generated for a single image object.

In our case, given that we have a good similarity measure, clusters are often very tightly packed. We can thus detect them more easily and at a lower computational expense by using the standard mean-shift algorithm [6], [1]. Mean-shift receives as input a set of feature vectors and the radius of the local window used to compute the mean-shift updates. This radius is directly related to the size of clusters output by the algorithm. The issue of determining an appropriate radius is in general unsolved for mean-shift. However, we note that given the relatively small number of input features, (usually less than a couple of thousand, compared to typically millions of pixels) performing the mean-shift process is not too expensive in terms of computation. Thus we can perform mean-shift with iteratively larger radius until the resulting clustering has a specified number of groups. For all our tests we fixed the number of output clusters to 7, and allowed the clustering method to determine the proper mean-shift radius.

The complete clustering process is as follows: Detect keypoints on an input image using a suitable interest point detector, generate colour and texture features for each patch, compute the matrix of pairwise similarities and perform spectral embedding, and finally, use mean-shift to detect groups of keypoints in the resulting embedding. Clustering results for several BSD test images are shown in Fig. 3. The figure shows that the method yields good clusters that correspond to individual objects or visually distinct background components.

To quantify the performance of the algorithm and to provide a baseline for comparison, we conducted a systematic evaluation of clustering performance over the 250 images of the BSD that we previously used to evaluate the performance of our similarity measure. We compared our results against the well know normalized cuts (Ncuts) [22] algorithm. This algorithm was chosen because it is representative of current graph-based clustering methods, and because it is closely related to spectral embedding, providing the closest alternative to the framework described above. To ensure that we evaluate only the clustering component of both algorithms, we use exactly the same similarity function, the same affinity matrix, and the same eigenvectors for both methods.

The criteria for clustering quality is simple. For each image we generate clusters using our method as well as Ncuts (note that Ncuts is used to cluster the keypoints directly, the clusters are not generated from an Ncuts segmentation of the image). Both algorithms were set to produce 7 clusters per image or as close to this as possible given the eigenvectors. We then compute the percentage of keypoints for each image that have been correctly clustered with regard to the human ground-truth from the BSD.

To evaluate correct classification, we generate ground truth labels for keypoints from the human segmentation in the BSD that has the closest number of regions to our expected 7. For each ground-truth cluster we find the keypoint cluster in the automatic grouping results that has the largest intersection (*i.e.* the automatic cluster in best agreement with this ground-truth cluster). We then mark keypoints common to both as correctly classified. We repeat this process with each ground-truth cluster, and compute at the end the percentage of correctly classified keypoints. This simple measure of correct classification will be low if clusters have been merge too much, or split too much. It provides a good indicator of clustering performance.

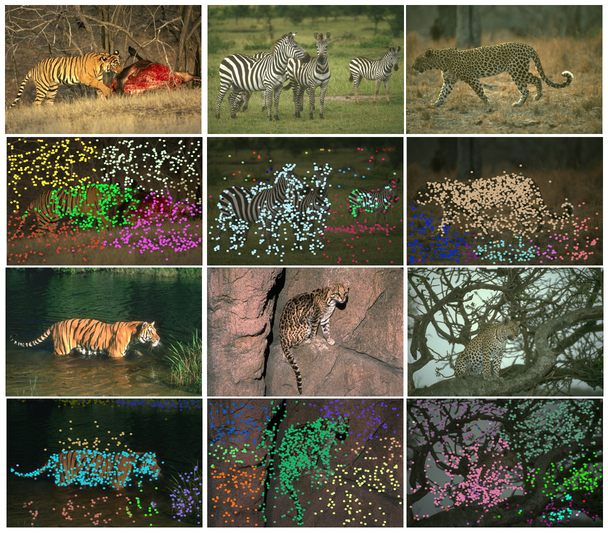

Figure 3. Clustering results on several images from the BSD. Each cluster is indicated by a different colour. The clustering algorithm is able to detect clusters that correspond to the object of interest despite the large amount of background features.

The mean correct-classification rate for our algorithm over the 250 BSD images used for testing is $R_{SE} = .6613$. The correct classification rate for Ncuts is only $R_{NCuts} = .5279$. Fig. 4 shows the correct classification results for the 250 BSD images used for testing. The results above, both visual and quantitative, indicate that our clustering framework and affinity function are sound and lead to good clustering results. We now show that the keypoint clusters produced by our algorithm provide constraints for typical interest point matching algorithms that result in significantly improved matching under difficult conditions.

5. Experimental Results and Applications

We first demonstrate the use of keypoint clustering for tracking. Fig. 5 shows several consecutive frames of a sequence with a van. This is a very difficult problem for keypoint based approaches because the van is lightly textured, and because it is imaged against a heavily cluttered back-

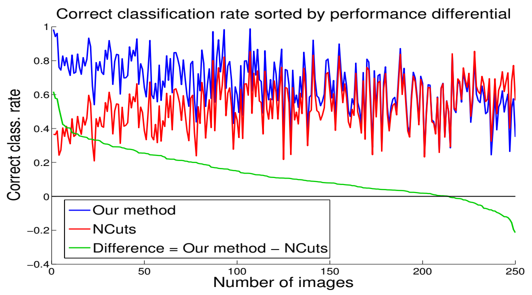

Figure 4. Correct classification rate on 250 BSD images for our method and NCuts. Results are sorted by the difference in performance to make the plot easier to understand. We observe that our method outperforms NCuts on more than 200 out of the 250 test images.

ground. The number of keypoints detected on the van is only a very small fraction of the total set of keypoints detected on any given frame of the sequence. Typical SIFT matching fails to detect and track the object. Our algorithm, on the other hand, is able to separate the keypoints coming from the van from those originating in the background. Once the user selects a cluster to be tracked on the initial frame, that cluster is automatically detected on successive frames. Constrained SIFT matching, where matches are only allowed to occur between the reference cluster and the automatically detected corresponding group in successive frames, is able to recover enough useful matches to provide a good estimate of pose for the duration of the sequence.

Our second example, illustrated in Fig. 6 concerns matching for an object characterized by non-rigid deformation. The reference image is not part of the sequence and in fact may depict a different animal than the one shown in the sequence. There is perspective distortion, a significant amount of background clutter, and the background is moving. Under this conditions we find once more that standard SIFT matching fails. Our method is able to separate keypoints corresponding to the cheetah from those on the background, both for the reference image and the frames in the sequence. After the user selects the reference object to be tracked, the cheetah cluster is easily detected over the entire sequence. Constrained matching is able to provide meaningful matches. Conversely, standard matching is typically confounded by similarity between background and foreground features.

A final example concerns matching under non-uniform illumination changes and perspective distortion. Fig. 7 shows two image pairs depicting two objects of interest. Due to non-uniform illumination changes, typical amounts of clutter, and viewpoint changes, standard matching produces poor results. Constrained matching on the other hand yields a much cleaner set of correspondences. The results clearly show that our algorithm yields high-quality clus-

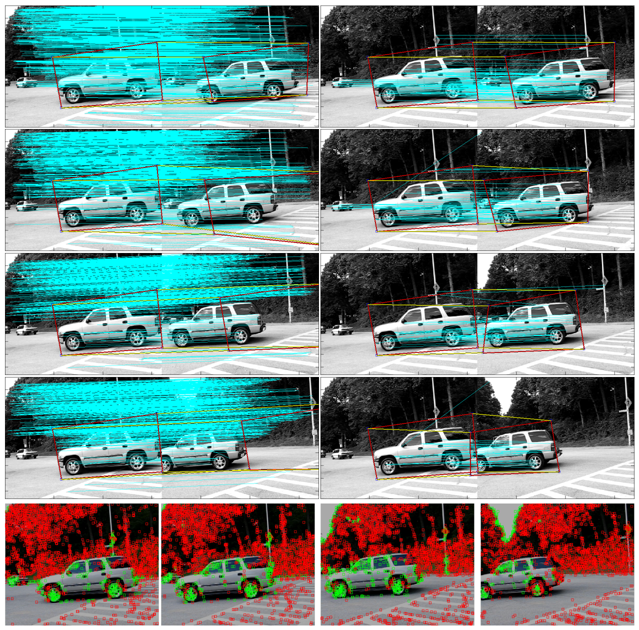

Figure 5. Tracking of a lightly textured object on a heavily cluttered background. **Left Column:** Unconstrained matching between image pairs. The red rectangles on the reference image are hand drawn and those on the target image are transformed by the homography computed using the matching keypoints. Because of large amounts of background clutter and erroneous correspondences the homography is bad. **Right Column:** Our approach to matching. The correspondences are much better and background is mostly eliminated. This is evidenced by the fact that the transformed rectangles now match much more accurately the car's location. **Bottom Row:** Keypoint clusters produced by our algorithm for the four target images. Keypoints assigned to the foreground are shown in green and the rest in red.

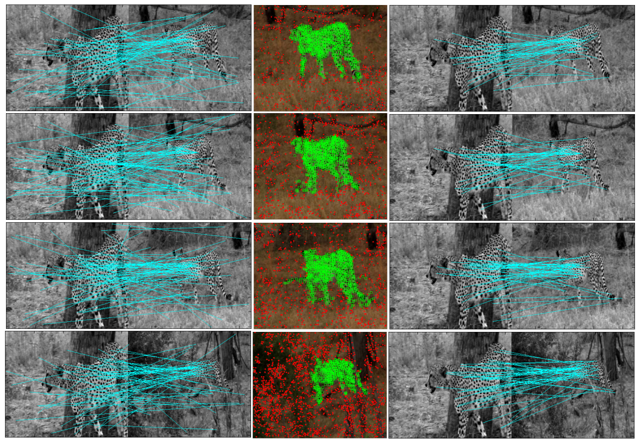

Figure 6. Tracking a cheetah against a complex background. **Left column:** Standard unconstrained matching between a video frame and the reference image that appears in the top-left corner of Fig. 1. **Middle column:** Foreground/background clusters produced by our method for the target video frame. **Right column:** Constrained matching results.

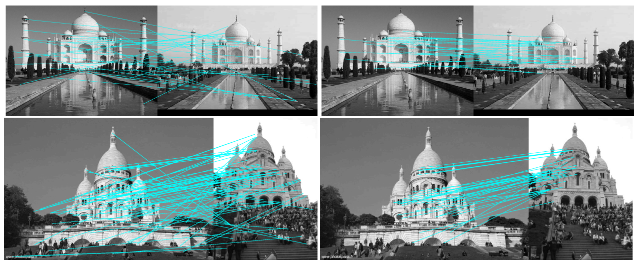

Figure 7. Additional examples of constrained SIFT matching. Top row: conventional SIFT matching. Bottom row: constrained matching using the clusters detected by our method.

ters even on difficult images, this leads to greatly improved matching results. This is not a trivial accomplishment given the degree or complexity of the objects and their background, as well as the fact that the results are produced without the benefit of any shape model, dynamical model, or predictive stage to help tracking the cluster across frames.

While pose can't be properly estimated using a simple homography for deformable objects, or in general for non-planar objects, the clustering process nonetheless provides strong constraints for algorithms that rely on feature matching for 3D reconstruction, bags of features approaches to object recognition, and automated model learning, which currently requires flat backgrounds (see for example [20]) but which could be extended to use keypoint clusters instead. These wider applications remain a matter for future research.

6. Conclusion

We have presented a sound clustering framework that organizes a set of keypoint patches into groups with similar appearance. We proposed a compact affinity measure for colour and texture and showed that it achieves almost the same performance as more complex, and computationally expensive filterbank-based measures. We described a clustering process based on spectral embedding and mean-shift, and showed it produces significantly better clusters than an alternate graph-based method on a large image database. Finally, we presented results that illustrate the usefulness of keypoint clusters. The results show that our algorithm allows for matching under conditions that are otherwise too difficult to handle using unconstrained matching alone.

References

[1] D. Comaniciu and P. Meer. A robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[2] F. Estrada. Advances in computational image segmentation and perceptual grouping, 2005. Ph.D. Thesis, Dept. of Computer Science, University of Toronto.

[3] F. Estrada and A. Jepson. Quantitative evaluation of a novel image segmentation algorithm. In *CVPR*, pages 1132–1139, 2005.

[4] F. Estrada, A. Jepson, and C. Chennubhotla. Spectral embedding and min-cut for image segmentation. In *BMVC*, pages 317–326, 2004.

[5] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In *CVPR*, volume 2, pages 54–61, 2003.

[6] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function with applications in pattern recognition. In *IEEE Trans. in Information Theory*, volume IT-21, pages 32–40, 1975.

[7] M. Galun, E. Sharon, R. Basri, and A. Brandt. Texture segmentation by multiscale aggregation of filter responses and shape elements. In *ICCV*, pages 716–723, 2003.

[8] Y. Gdalyahu, D. Weinshall, and M. Werman. Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *PAMI*, 23(10):1053–1074, 2001.

[9] L. Grady. Random walks for image segmentation. *PAMI*, 28(11):1768–1783, 2006.

[10] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *IJCV*, 13(9):920–935, 1996.

[11] N. Jojic, B. J. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *ICCV*, pages 34–41, 2003.

[12] A. Kannan, J. Winn, and C. Rother. Clustering appearance and shape by learning jigsaws. In *NIPS*, pages 657–664, 2006.

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

[14] D. Lowe. Demo software: Sift keypoint detector, 2005. `http://www.cs.ubc.ca/~lowe/keypoints/`.

[15] J. Malik, S. Belongie, J. Shi, and T. K. Leung. Textons, contours and regions: Cue integration in image segmentation. In *ICCV*, volume 2, pages 918–925, 1999.

[16] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, pages 416–423, 2001.

[17] M. Meila and J. Shi. Learning segmentation by random walks. In *NIPS*, pages 873–879, 2000.

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

[19] B. Mičušík and T. Pajdla. Multi-label image segmentation via max-sum solver. In *CVPR*, pages 1–6, 2007.

[20] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature targeting for tracking by detection. In *ECCV*, pages 592–605, 2006.

[21] Y. Rubner, J. Puzicha, C. Tomasi, and J. M. Buhman. Empirical evaluation of dissimilarity measures for color and texture. *CVIU*, 84(1):25–43, 2001.

[22] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[23] O. Tuzel, F. Porikli, and P. Meer. Region covariance: a fast descriptor for detection and classification. In *ECCV*, volume 2, pages 589–600, 2006.

[24] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *IJCV*, 62(1-2):61–81, 2005.

[25] M. Wertheimer. *A Sourcebook of Gestalt Psychology*. Routledge and Kegan Paul, 1938.

[26] L. R. Williams and D. W. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. *Neural Computation*, 9(4):837–858, 1997.

[27] L. Yang, P. Meer, and D. J. Foran. Multiple class segmentation using a unified framework over mean-shift patches. In *CVPR*, pages 1–8, 2007.

Appendix

We follow the formulation of the spectral embedding presented in [4] in the context of image segmentation. First, we represent the set $\mathcal{P}$ of image patches as a graph $G(V, E)$ where V is a set of nodes, with one node per patch, and E is a set of undirected edges connecting nodes to one another. The weight of the edge $E(i, j)$ is simply the similarity a_{ij} (see Eq. 5) between the two corresponding patches. The graph is conveniently represented as a symmetric affinity matrix A where $A(i, j) = a_{ij}$. From this matrix, it is easy to generate a proper Markov matrix M by normalizing each column of A so that the values in the column add up to 1. We can define an arbitrary probability distribution $\vec{d}$ as a column vector with one entry per patch, which satisfies $\sum_i \vec{d}(i) = 1$. The value of each component in this vector represents how much probability mass is currently stored at the corresponding image patch.

The matrix M defines a random walk on the graph $G(V, E)$. Consider what happens to an initial probability distribution $\vec{d}_0$ that has all its probability mass concentrated at a single node of G. After t steps of the random walk, the new probability distribution $\vec{d}_t$ is given by $\vec{d}_t = M^{\,t}\vec{d}_0$. The effect of M upon $\vec{d}_0$ is that of diffusing the initial probability distribution, but this diffusion is not isotropic. Since the probability of traveling between any two nodes is directly proportional to their similarity, the resulting diffusion favours visiting nodes that are similar to the starting point.

In [4], the patterns of diffusion resulting from simulating the random walk at specific nodes are called blur kernels since they can be used for performing anisotropic image smoothing. Here we refer to them simply as diffusion kernels. Formally, we define $\vec{d}_t^j = M^{\,t}\vec{d}_0^j$ where $\vec{d}_t^j$ is the diffusion kernel obtained by simulating t steps of a random walk with initial probability distribution $\vec{d}_0^j$, and $\vec{d}_0^j$ has all its probability mass concentrated at the node corresponding to patch p_j.

The key property to be noted from the above process is that the diffusion kernels for image patches belonging to the same object are typically very similar even when the initial distributions are localized at different points in the image. In fact, this will be the case even if the initial similarity between the patches is small. The information provided by the diffusion kernel goes beyond what is encoded in the pairwise patch similarities, it provides global similarity information that relates two patches by taking into account the configuration of the entire graph. As we should expect, diffusion kernels for patches from different-looking regions of the image yield entirely different distributions. The fundamental principle behind the clustering process is that clustering using diffusion kernels should be much easier since the diffusion process enhances the similarity of elements that should be grouped together.

While we could compute the diffusion kernels for each patch, and try to cluster them directly, this may be complicated because of the high-dimensionality of the kernels (equal to the number of patches in the image). Instead of using the long vectors that encode each diffusion kernel, we compute a low-dimensional approximation of each kernel using the spectral properties of M. Formally speaking, we compute a projection (embedding) of each diffusion kernel onto the leading eigenvectors of M (i.e. the eigenvectors with largest, real eigenvalues). The projected diffusion kernels are given by

$$\vec{w}_t^j = \Delta_n^t V_n^T \vec{d}_0^j, \tag{6}$$

where $[U, \Delta, V]$ is the SVD of M, Δ_n is a diagonal matrix containing the largest n eigenvalues of M, and V_n contains the n columns of V with the corresponding eigenvectors. From $\vec{w}_t^j$ we can easily reconstruct an approximation of the diffusion kernels $\vec{d}_t^j \approx \hat{\vec{d}}_t^j = U_n \vec{w}_t^j$. U_n contains the n columns of U corresponding to the n largest eigenvalues. We choose $n = 5$, and set the value of t as described in [4] so that the contribution of the last eigenvector of the embedding is small, and the resulting reconstruction of the full diffusion kernels is reasonably free of artifacts. A sample embedding is shown in Fig. 2.

It is worth pointing out the fundamental difference between the spectral embedding formulation described above and standard spectral clustering techniques. Typical spectral clustering techniques use individual eigenvectors of a Laplacian matrix to determine partitions of the dataset directly. This provides good partitions only when the data have strongly separated clusters, but has a tendency to leak across weak cluster boundaries. Spectral embedding, on the other hand, uses a subset of the eigenvectors of M only to (efficiently) compute a low dimensional approximation of the diffusion kernels. The random walk process together with the embedding effectively map similar elements to small neighborhoods in the low-dimensional space. The partitions do not come from the eigenvectors, instead, the projected diffusion kernels $\vec{w}_t^j$ are clustered directly using a suitable algorithm (mean shift in our case). A study of the relationship between random walks and standard spectral clustering techniques is presented in [17].