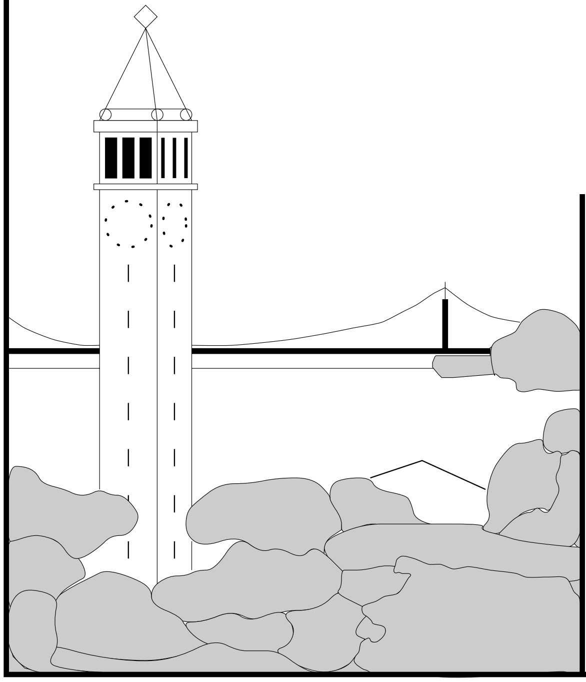
Normalized Cuts and Image Segmentation

Jianbo Shi and Jitendra Malik



Report No. UCB/CSD-97-940

May 1997

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Normalized Cuts and Image Segmentation

Jianbo Shi*† and Jitendra Malik*
Computer Science Division
University of California at Berkeley, Berkeley, CA 94720
{jshi,malik}@cs.berkeley.edu

Abstract

We propose a novel approach for solving the perceptual grouping problem in vision. Rather than focusing on local features and their consistencies in the image data, our approach aims at extracting the global impression of an image. We treat image segmentation as a graph partitioning problem and propose a novel global criterion, the normalized cut*, for segmenting the graph. The* normalized cut *criterion measures both the total dissimilarity between the different groups as well as the total similarity within the groups. We show that an efficient computational technique based on a generalized eigenvalue problem can be used to optimize this criterion. We have applied this approach to segmenting static images as well as motion sequences and found results very encouraging.*

*Supported by (ARO) DAAH04-96-1-0341
†NSF Graduate Fellowship

1 Introduction

Nearly 75 years ago, Wertheimer[20] launched the Gestalt approach which laid out the importance of perceptual grouping and organization in visual perception. For our purposes, the problem of grouping can be well motivated by considering the set of points shown in the figure (1).

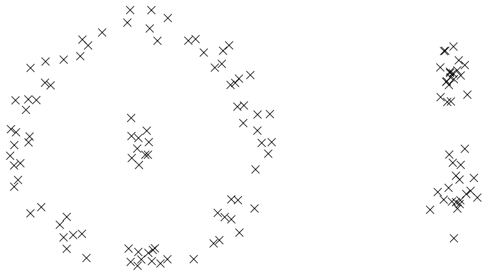


Figure 1: How many groups?

Typically a human observer will perceive four objects in the image–a circular ring with a cloud of points inside it, and two loosely connected clumps of points on its right. However this is not the unique partitioning of the scene. One can argue that there are three objects–the two clumps on the right constitute one dumbbell shaped object. Or there are only two objects, a dumb bell shaped object on the right, and a circular galaxy like structure on the left. If one were perverse, one could argue that in fact every point was a distinct object.

This may seem to be an artificial example, but every attempt at image segmentation ultimately has to confront a similar question–there are many possible partitions of the domain D of an image into subsets D_i (including the extreme one of every pixel being a separate entity). How do we pick the "right" one? We believe the Bayesian view is appropriate– one wants to find the most probable interpretation in the context of prior world knowledge. The difficulty, of course, is in specifying the prior world knowledge– some of it is low level such as coherence of brightness, color, texture, or motion, but equally important is mid- or high- level knowledge about symmetries of objects or object models.

This suggests to us that image segmentation based on low level cues can not and should not aim to produce a complete final "correct" segmentation. The objective should instead be to *use the low-level coherence of brightness, color, texture or motion attributes to sequentially come up with candidate partitions.* Mid and high level knowledge can be used to either confirm these groups or select some for further attention. This attention could result in further repartitioning or grouping. The key point is that image partitioning is to be done from the big picture downwards, rather like a painter first marking out the major areas and then filling in the details.

Prior literature on the related problems of clustering, grouping and image segmentation is huge. The clustering community[11] has offered us agglomerative and divisive algorithms; in image segmentation we have region-based merge and split algorithms. The hierarchical divisive approach that we are advocating produces a tree, the *dendrogram.* While most of these ideas go back to the 70s (and earlier), the 1980s brought in the use of Markov Random Fields[8] and variational formulations[15, 2, 13]. The MRF and variational formulations also exposed two basic questions (1) What is the criterion that one wants to optimize? and (2) Is there an efficient algorithm for carrying out the optimization? Many an attractive criterion has been doomed by the inability to find an effective algorithm to find its minimum–greedy or gradient descent type approaches fail to find global optima for these high dimensional, nonlinear problems.

Our approach is most related to the graph theoretic formulation of grouping. The set of points in an arbitrary feature space are represented as a weighted undirected graph $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$, where the nodes of the graph are the points in the feature space, and an edge is formed between every pair of nodes. The weight on each edge, $w(\boldsymbol{i}, \boldsymbol{j})$, is a function of the similarity between nodes $\boldsymbol{i}$ and $\boldsymbol{j}$.

In grouping, we seek to partition the set of vertices into disjoint sets $\mathbf{V}_1, \mathbf{V}_2, ..., \mathbf{V}_m$, where by some measure the similarity among the vertices in a set $\mathbf{V}_i$ is high and across different sets $\mathbf{V}_i$,$\mathbf{V}_j$ is low.

To partition a graph, we need to also ask the following questions:

1. What is the precise criterion for a good partition?
2. How can such a partition be computed efficiently?

In the image segmentation and data clustering community, there has been much previous work using variations of the minimal spanning tree or limited neighborhood set approaches. Although those use efficient computational methods, the segmentation criteria used in most of them are based on local properties of the graph. Because perceptual grouping is about extracting the global impressions of a scene, as we saw earlier, this partitioning criterion often falls short of this main goal.

In this paper we propose a new graph-theoretic criterion for measuring the goodness of an image partition– the *normalized cut*. We introduce and justify this criterion in section 2. The minimization of this criterion can be formulated as a generalized eigenvalue problem; the eigenvectors of this problem can be used to construct good partitions of the image and the process can be continued recursively as desired(section 3). In section 4 we show experimental results. The formulation and minimization of the normalized cut criterion draws on a body of results, theoretical and practical, from the numerical analysis and theoretical computer science communities–section 5 discusses previous work on the spectral partitioning problem. We conclude in section 6.

2 Grouping as graph partitioning

A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ can be partitioned into two disjoint sets, A, B, $A \cup B = \mathbf{V}$, $A \cap B = \emptyset$, by simply removing edges connecting the two parts. The degree of dissimilarity between these two pieces can be computed as total weight of the edges that have been removed. In graph theoretic language, it is called the *cut*:

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v). \tag{1}$$

The optimal bi-partitioning of a graph is the one that minimizes this *cut* value. Although there are exponential number of such partitions, finding the *minimum cut* of a graph is a well studied problem, and there exist efficient algorithms for solving it.

Wu and Leahy[21] proposed a clustering method based on this minimum cut criterion. In particular, they seek to partition a graph into k-subgraphs, such that the maximum cut across the subgroups is minimized. This problem can be efficiently solved by recursively finding the minimum cuts that bisect the existing segments. As shown in Wu & Leahy's work, this globally optimal criterion can be used to produce good segmentation on some of the images.

However, as Wu and Leahy also noticed in their work, the minimum cut criteria favors cutting small sets of isolated nodes in the graph. This is not surprising since the *cut* defined in (1) increases with the number of edges going across the two partitioned parts. Figure (2) illustrates one such case. Assuming the edge weights are inversely proportional to the distance between the two nodes, we see the cut that partitions out node n_1 or n_2 will have a very small value. In fact, any cut that partitions out individual nodes on the right half will have smaller cut value than the cut that partitions the nodes into the left and right halves.

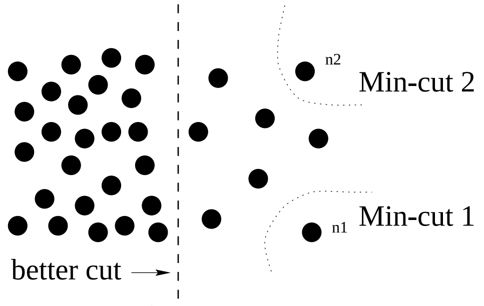


Figure 2: A case where minimum cut gives a bad partition.

To avoid this unnatural bias for partitioning out small sets of points, we propose a new measure of disassociation between two groups. Instead of looking at the value of total edge weight connecting the two partitions, our measure computes the cut cost as a fraction of the total edge connections to all the nodes in the graph. We call this disassociation measure the *normalized cut* (*Ncut*):

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)} \tag{2}$$

where $asso(A, V) = \sum_{u \in A, t \in V} w(u, t)$ is the total connection from nodes in A to all nodes in the graph, and $asso(B, V)$ is similarly defined. With this definition of the disassociation between the groups, the cut that partitions out small isolated points will no longer have small *Ncut* value, since the *cut* value will almost certainly be a large percentage of the total connection from that small set to all other nodes. In the case illustrated in figure 2, we see that the cut_1 value across node n_1 will be 100% of the total connection from that node.

In the same spirit, we can define a measure for total normalized association within groups for a given partition:

$$Nasso(A, B) = \frac{asso(A, A)}{asso(A, V)} + \frac{asso(B, B)}{asso(B, V)} \tag{3}$$

where $asso(A, A)$ and $asso(B, B)$ are total weights of edges connecting nodes within A and B respectively. We see again this is an unbiased measure, which reflects how tightly on average nodes within the group are connected to each other.

Another important property of this definition of association and disassociation of a partition is that they are naturally related:

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)} \\ &= \frac{asso(A,V) - asso(A,A)}{asso(A,V)} \\ &\quad + \frac{asso(B,V) - asso(B,B)}{asso(B,V)} \\ &= 2 - (\frac{asso(A,A)}{asso(A,V)} + \frac{asso(B,B)}{asso(B,V)}) \\ &= 2 - Nasso(A,B) \end{aligned}$$

Hence the two partition criteria that we seek in our grouping algorithm, minimizing the disassociation between the groups and maximizing the association within the group, are in fact identical, and can be satisfied simultaneously. In our algorithm, we will use this *normalized cut* as the partition criterion.

Having defined the graph partition criterion that we want to optimize, we will show how such an optimal partition can be computed efficiently.

2.1 Computing the optimal partition

Given a partition of nodes of a graph, V, into two sets A and B, let $\boldsymbol{x}$ be an $N = |\boldsymbol{V}|$ dimensional indicator vector, $x_i = 1$ if node i is in A, and -1 otherwise. Let $\boldsymbol{d}$(i) $= \sum_j w(i,j)$, be the total connection from node $\boldsymbol{i}$ to all other nodes. With the definitions $\boldsymbol{x}$ and $\boldsymbol{d}$ we can rewrite $Ncut(A,B)$ as:

$$\begin{aligned} Ncut(A,B) &= \frac{cut(A,B)}{asso(A,V)} + \frac{cut(B,A)}{asso(B,V)} \\ &= \frac{\sum_{(\boldsymbol{x}_i>0,\boldsymbol{x}_j<0)} -w_{ij}\boldsymbol{x}_i\boldsymbol{x}_j}{\sum_{\boldsymbol{x}_i>0}\boldsymbol{d}_i} \\ &\quad + \frac{\sum_{(\boldsymbol{x}_i<0,\boldsymbol{x}_j>0)} -w_{ij}\boldsymbol{x}_i\boldsymbol{x}_j}{\sum_{\boldsymbol{x}_i<0}\boldsymbol{d}_i} \end{aligned}$$

Let $\mathbf{D}$ be an $N \times N$ diagonal matrix with $\boldsymbol{d}$ on its diagonal, $\mathbf{W}$ be an $N \times N$ symmetrical matrix with W(i,j) $= w_{ij}$, $k = \frac{\sum_{x_i>0}\boldsymbol{d}_i}{\sum_i \boldsymbol{d}_i}$, and $\mathbf{1}$ be an $N \times 1$ vector of all ones. Using the fact $\frac{\mathbf{1}+\boldsymbol{x}}{2}$ and $\frac{\mathbf{1}-\boldsymbol{x}}{2}$ are indicator vectors for $x_i > 0$ and $x_i < 0$ respectively, we can rewrite $4[Ncut(\boldsymbol{x})]$ as:

$$\begin{aligned} &= \frac{(\mathbf{1}+\boldsymbol{x})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}+\boldsymbol{x})}{k\mathbf{1}^T\mathbf{D}\mathbf{1}} + \frac{(\mathbf{1}-\boldsymbol{x})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}-\boldsymbol{x})}{(1-k)\mathbf{1}^T\mathbf{D}\mathbf{1}} \\ &= \frac{(\boldsymbol{x}^T(\mathbf{D}-\mathbf{W})\boldsymbol{x}+\mathbf{1}^T(\mathbf{D}-\mathbf{W})\mathbf{1})}{k(1-k)\mathbf{1}^T\mathbf{D}\mathbf{1}} + \frac{2(1-2k)\mathbf{1}^T(\mathbf{D}-\mathbf{W})\boldsymbol{x}}{k(1-k)\mathbf{1}^T\mathbf{D}\mathbf{1}} \end{aligned}$$

Let $\alpha(\boldsymbol{x}) = \boldsymbol{x}^T(\mathbf{D}-\mathbf{W})\boldsymbol{x}$, $\beta(\boldsymbol{x}) = \mathbf{1}^T(\mathbf{D}-\mathbf{W})\boldsymbol{x}$, $\gamma = \mathbf{1}^T(\mathbf{D}-\mathbf{W})\mathbf{1}$, and $M = \mathbf{1}^T\mathbf{D}\mathbf{1}$, we can then further expand the above equation as:

$$
\begin{aligned}
&= \frac{(\alpha(\boldsymbol{x})+\gamma)+2(1-2k)\beta(\boldsymbol{x})}{k(1-k)M} \\
&= \frac{(\alpha(\boldsymbol{x})+\gamma)+2(1-2k)\beta(\boldsymbol{x})}{k(1-k)M} - \frac{2(\alpha(\boldsymbol{x})+\gamma)}{M} + \frac{2\alpha(\boldsymbol{x})}{M} + \frac{2\gamma}{M}
\end{aligned}
$$

dropping the last constant term, which in this case equals 0, we get

$$
\begin{aligned}
&= \frac{(1-2k+2k^2)(\alpha(\boldsymbol{x})+\gamma)+2(1-2k)\beta(\boldsymbol{x})}{k(1-k)M} + \frac{2\alpha(\boldsymbol{x})}{M} \\
&= \frac{\frac{(1-2k+2k^2)}{(1-k)^2}(\alpha(\boldsymbol{x})+\gamma)+\frac{2(1-2k)}{(1-k)^2}\beta(\boldsymbol{x})}{\frac{k}{1-k}M} + \frac{2\alpha(\boldsymbol{x})}{M}
\end{aligned}
$$

Letting $b = \frac{k}{1-k}$, and since $\gamma = 0$, it becomes,

$$
\begin{aligned}
&= \frac{(1+b^2)(\alpha(\boldsymbol{x})+\gamma)+2(1-b^2)\beta(\boldsymbol{x})}{bM} + \frac{2b\alpha(\boldsymbol{x})}{bM} \\
&= \frac{(1+b^2)(\alpha(\boldsymbol{x})+\gamma)}{bM} + \frac{2(1-b^2)\beta(\boldsymbol{x})}{bM} + \frac{2b\alpha(\boldsymbol{x})}{bM} - \frac{2b\gamma}{bM} \\
&= \frac{(1+b^2)(\boldsymbol{x}^T(\mathbf{D}-\mathbf{W})\boldsymbol{x}+\mathbf{1}^T(\mathbf{D}-\mathbf{W})\mathbf{1})}{b\mathbf{1}^T\boldsymbol{D}\mathbf{1}} \\
&\quad + \frac{2(1-b^2)\mathbf{1}^T(\mathbf{D}-\mathbf{W})\boldsymbol{x}}{b\mathbf{1}^T\boldsymbol{D}\mathbf{1}} \\
&\quad + \frac{2b\boldsymbol{x}^T(\mathbf{D}-\mathbf{W})\boldsymbol{x}}{b\mathbf{1}^T\mathbf{D}\mathbf{1}} - \frac{2b\mathbf{1}^T(\mathbf{D}-\mathbf{W})\mathbf{1}}{b\mathbf{1}^T\boldsymbol{D}\mathbf{1}} \\
&= \frac{(\mathbf{1}+\boldsymbol{x})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}+\boldsymbol{x})}{b\mathbf{1}^T\boldsymbol{D}\mathbf{1}} \\
&\quad + \frac{b^2(\mathbf{1}-\boldsymbol{x})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}-\boldsymbol{x})}{b\mathbf{1}^T\boldsymbol{D}\mathbf{1}} \\
&\quad - \frac{2b(\mathbf{1}-\boldsymbol{x})^T(\mathbf{D}-\mathbf{W})(\mathbf{1}+\boldsymbol{x})}{b\mathbf{1}^T\boldsymbol{D}\mathbf{1}} \\
&= \frac{[(\mathbf{1}+\boldsymbol{x})-b(\mathbf{1}-\boldsymbol{x})]^T(\mathbf{D}-\mathbf{W})[(\mathbf{1}+\boldsymbol{x})-b(\mathbf{1}-\boldsymbol{x})]}{b\mathbf{1}^T\boldsymbol{D}\mathbf{1}}
\end{aligned}
$$

Setting $\boldsymbol{y} = (\mathbf{1}+\boldsymbol{x}) - b(\mathbf{1}-\boldsymbol{x})$, it is easy to see that

$$
\boldsymbol{y}^T\boldsymbol{D}\mathbf{1} = \sum_{x_i>0}\boldsymbol{d}_i - b\sum_{x_i<0}\boldsymbol{d}_i = 0 \tag{4}
$$

since $b = \frac{k}{1-k} = \frac{\sum_{x_i>0}\boldsymbol{d}_i}{\sum_{x_i<0}\boldsymbol{d}_i}$, and

$$
\begin{aligned}
\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y} &= \sum_{x_i>0}\boldsymbol{d}_i + b^2\sum_{x_i<0}\boldsymbol{d}_i \\
&= b\sum_{x_i<0}\boldsymbol{d}_i + b^2\sum_{x_i<0}\boldsymbol{d}_i \\
&= b(\sum_{x_i<0}\boldsymbol{d}_i + b\sum_{x_i<0}\boldsymbol{d}_i) \\
&= b\mathbf{1}^T\boldsymbol{D}\mathbf{1}.
\end{aligned}
$$

Putting everything together we have,

$$min_{\boldsymbol{x}} Ncut(\boldsymbol{x}) = min_{\boldsymbol{y}} \frac{\boldsymbol{y}^T(\boldsymbol{D}-\boldsymbol{W})\boldsymbol{y}}{\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{y}}, \tag{5}$$

with the condition $\boldsymbol{y}_i \in \{1, -b\}$ and $\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{1} = 0$.

Note that the above expression is the Rayleigh quotient[9]. If $\boldsymbol{y}$ is relaxed to take on real values, we can minimize equation (5) by solving the generalized eigenvalue system,

$$(\mathbf{D}-\mathbf{W})\boldsymbol{y} = \lambda\mathbf{D}\boldsymbol{y}. \tag{6}$$

However, we have two constraints on $\boldsymbol{y}$, which come from the condition on the corresponding indicator vector $\boldsymbol{x}$. First consider the constraint $\boldsymbol{y}^T\boldsymbol{D}\boldsymbol{1} = 0$. We can show this constraint on $\boldsymbol{y}$ is automatically satisfied by the solution of the generalized eigensystem. We will do so by first transforming equation (6) into a standard eigensystem, and show the corresponding condition is satisfied there. Rewrite equation (6) as

$$\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\boldsymbol{z} = \lambda\boldsymbol{z}, \tag{7}$$

where $\boldsymbol{z} = \mathbf{D}^{\frac{1}{2}}\boldsymbol{y}$. One can easily verify that $\boldsymbol{z}_0 = \mathbf{D}^{\frac{1}{2}}\boldsymbol{1}$ is an eigenvector of equation (7) with eigenvalue of 0. Furthermore, $\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}$ is symmetric semi-positive definite, since $(\mathbf{D}-\mathbf{W})$, also called the *Laplacian* matrix, is known to be semi-positive definite[16]. Hence $\boldsymbol{z}_0$ is in fact the smallest eigenvector of equation (7), and all eigenvectors of equation (7) are perpendicular to each other. In particular, $\boldsymbol{z}_1$ the second smallest eigenvector is perpendicular to $\boldsymbol{z}_0$. Translating this statement back into the general eigensystem (6), we have 1) $\boldsymbol{y}_0 = (0, \boldsymbol{1})$ is the smallest eigenvector, and 2) $0 = \boldsymbol{z}_1^T\boldsymbol{z}_0 = \boldsymbol{y}_1^T\mathbf{D}\boldsymbol{1}$, where $\boldsymbol{y}_1$ is the second smallest eigenvector of (6).

Now recall a simple fact about the *Rayleigh quotient*[9]:

Let $\mathbf{A}$ *be a real symmetric matrix. Under the constraint that* $\boldsymbol{x}$ *is orthogonal to the j-1 smallest eigenvectors* $\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{j-1}$*, the quotient* $\frac{\boldsymbol{x}^T\mathbf{A}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}}$ *is minimized by the next smallest eigenvector* $\boldsymbol{x}_j$*, and its minimum value is the correspoding eigenvalue* $\boldsymbol{\lambda}_j$*.*

As a result, we obtain:

$$\boldsymbol{z}_1 = arg.min_{\boldsymbol{z}^T\boldsymbol{z}_0=0} \frac{\boldsymbol{z}^T\mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{W})\mathbf{D}^{-\frac{1}{2}}\boldsymbol{z}}{\boldsymbol{z}^T\boldsymbol{z}}, \tag{8}$$

and consequently,

$$\boldsymbol{y}_1 = arg.min_{\boldsymbol{y}^T\mathbf{D}\boldsymbol{1}=0} \frac{\boldsymbol{y}^T(\mathbf{D}-\mathbf{W})\boldsymbol{y}}{\boldsymbol{y}^T\mathbf{D}\boldsymbol{y}}, \tag{9}$$

Thus the second smallest eigenvector of the generalized eigensystem (6) is the real valued solution to our *normalized cut* problem. The only reason that it is not necessarily the solution to our original problem is that the second constraint on $\boldsymbol{y}$ that $\boldsymbol{y}_i$ takes on two discrete values is not automatically satisfied. In fact relaxing this constraint is what makes this optimization problem tractable in the first place. We will show in section (3) how this real valued solution can be transformed into a discrete form.

A similar argument can also be made to show that the eigenvector with the third smallest eigenvalue is the real valued solution that optimally sub-partitions the first two parts. In fact this line of argument can be extended to show that one can subdivide the existing graphs, each time using the eigenvector with the next smallest eigenvalue. However, in practice because the approximation error from the real valued solution to the discrete valued solution accumulates with every eigenvector taken, and all eigenvectors have to satisfy a global mutual orthogonal constraint, solutions based on higher eigenvectors become unreliable. It is best to restart solving the partitioning problem on each subgraph individually.

In summary, we propose using the *normalized cut* criteria for graph partitioning, and we have shown how this criteria can be computed efficiently by solving a generalized eigenvalue problem.

3 The grouping algorithm

As we saw above, the generalized eigensystem in (6) can be transformed into a standard eigenvalue problem. Solving a standard eigenvalue problem for all eigenvectors takes $O(n^3)$ operations, where n is the number of nodes in the graph. This becomes impractical for image segmentation applications where n is the number of pixels in an image. Fortunately, our graph partitioning has the following properties: 1) the graphs often are only locally connected and the resulting eigensystems are very sparse, 2) only the top few eigenvectors are needed for graph partitioning, and 3) the precision requirement for the eigenvectors is low, often only the right sign bit is required. These special properties of our problem can be fully exploited by an eigensolver called the Lanczos method. The running time of a Lanczos algorithm is $O(mn) + O(mM(n))$[9], where m is the maximum number of matrix-vector computations allowed, and $M(n)$ is the cost of a matrix-vector computation. In the case where $(\mathbf{D} - \mathbf{W})$ is sparse, matrix-vector takes only $O(n)$ time. The number m depends on many factors[9]. In our experiments on image segmentations, m is typically less than $O(n^{\frac{1}{2}})$.

Once the eigenvectors are computed, we can partition the graph into two pieces using the second smallest eigenvector. In the ideal case, the eigenvector should only take on two discrete values, and the signs of the values can tell us exactly how to partition the graph. However, our eigenvectors can take on continuous values, and we need to choose a splitting point to partition it into two parts. There are many different ways of choosing such splitting point. One can take 0 or the median value as the splitting point, or one can search for the splitting point such that the resulting partition has the best $Ncut(A, B)$ value. We take the latter approach in our work. Currently, the search is done by checking l evenly spaced possible splitting points, and computing the best *Ncut* among them. In our experiments, the values in the eigenvectors are usually well separated, and this method of choosing a splitting point is very reliable even with a small l.

After the graph is broken into two pieces, we can recursively run our algorithm on the two partitioned parts. Or equivalently, we could take advantage of the special properties of the other top eigenvectors as explained in previous section to subdivide the graph based on those eigenvectors. The recursion stops once the *Ncut* value exceeds certain limit.

We also impose a stability criterion on the partition, rather analogous to a localization criterion in edge detection. In edge detection, we can distinguish a real

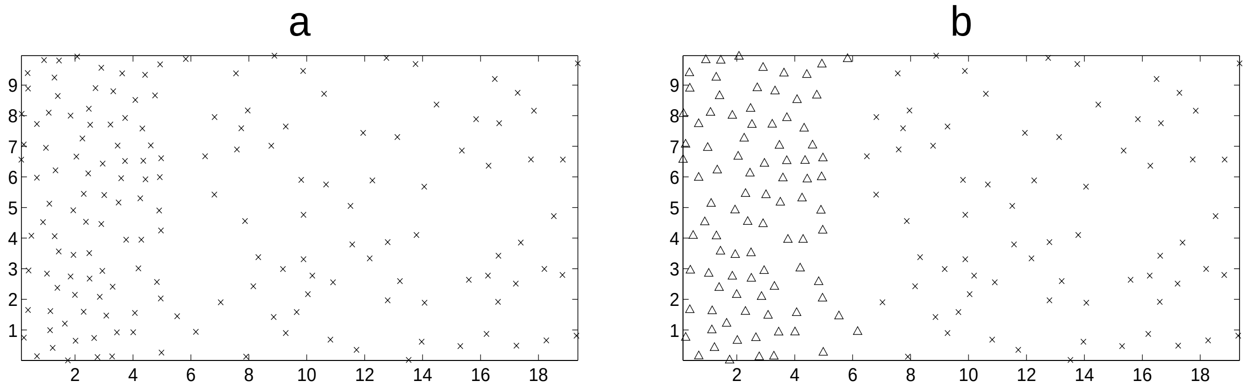


Figure 3: (a) Point set generated by two Poisson processes, with densities of 2.5 and 1.0 on the left and right cluster respectively, (b) $\triangle$ and $\times$ indicates the partition of point set in (a). Parameter settings: $\sigma_X = 5, r = 3$.

edge from a region of high shading gradient by the criterion that varying the position of a true edge changes its strength, while in a smoothly shaded region varying the position of the putative edge does not effect its strength. In the current context, we regard a cut as unstable if varying the set of graph edges forming the cut, the *Ncut* value does not change much. To compute this stability measure, we vary the value of splitting point around the optimal value, and induce two different partitions, $P1 = (A1, B1)$ and $P2 = (A2, B2)$. The stability measure is the ratio $\frac{\delta cut(P1,P2)}{\delta D(P1,P2)}$, where $\delta D(P1,P2) = \sum_{i \in (A1/A2)} \boldsymbol{d}_i$

Our grouping algorithm can be summarized as follows:

1. Given a set of features, set up a weighted graph $G = (V, E)$, compute the weight on each edge, and summarize the information into $\mathbf{W}$, and $\mathbf{D}$.
2. Solve $(\mathbf{D} - \mathbf{W})\boldsymbol{x} = \lambda \mathbf{D} \boldsymbol{x}$ for eigenvectors with the smallest eigenvalues.
3. Use the eigenvector with second smallest eigenvalue to bipartition the graph by finding the splitting point such that $Ncut$ is maximized,
4. Decide if the current partition should be sub-divided by checking the stability of the cut, and make sure $Ncut$ is below pre-specified value,
5. Recursively repartition the segmented parts if necessary.

The number of groups segmented by this method is controlled directly by the maximum allowed $Ncut$.

4 Experiments

We have applied our grouping algorithm to monocular image segmentation based on brightness, color, or texture information. In each case, we construct the graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ by taking each pixel as a node, and define the edge weight w_{ij} between node i

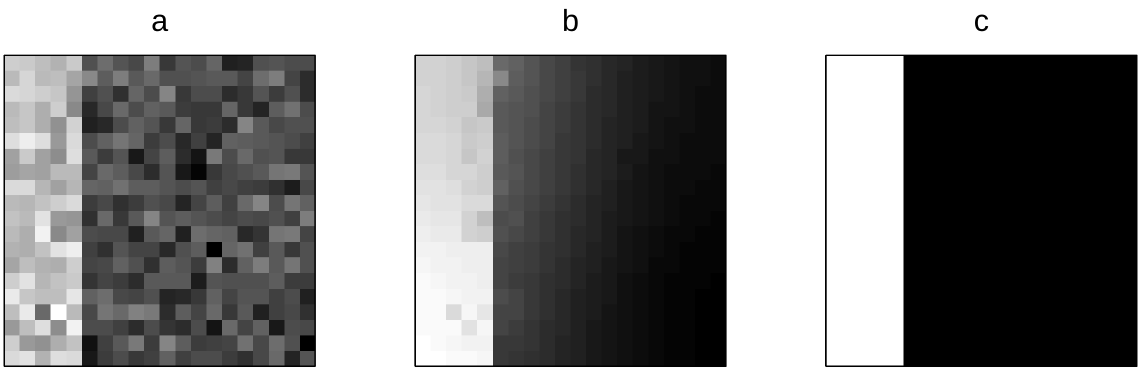


Figure 4: A synthetic image showing a noisy "step" image. Intensity varies from 0 to 1, and Gaussian noise with $\sigma = 0.2$ is added. Subplot (b) shows the eigenvector with the second smallest eigenvalue, and subplot (c) shows the resulting partition.

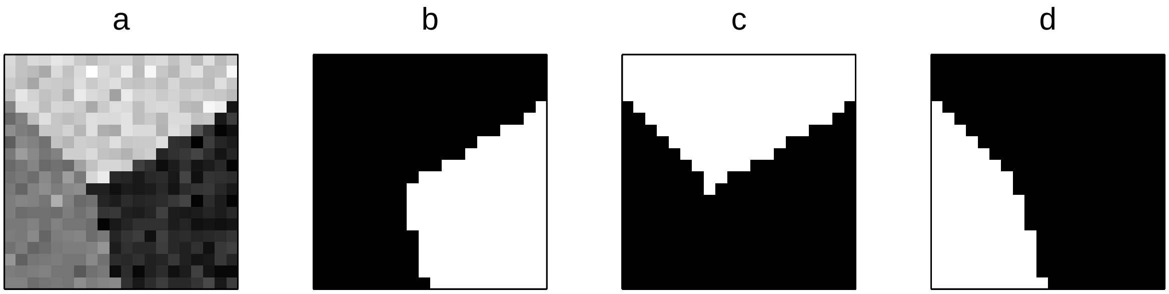


Figure 5: (a) A synthetic image showing three image patchs forming a junction. Image intensity varies from 0 to 1, and Gaussian noise with $\sigma = 0.1$ is added. (b)-(d) shows the top three components of the partition.

and j as the product of a feature similarity term and spatial proximitiy term:

$$w_{ij} = e^{\frac{-\|\boldsymbol{F}(i)-\boldsymbol{F}(j)\|_2}{\sigma_I}} * \begin{cases} e^{\frac{-\|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|_2}{\sigma_X}} & \text{if} \\ & \|\boldsymbol{X}(i)-\boldsymbol{X}(j)\|_2 < r \\ 0 & \text{otherwise} \end{cases}$$

where $\boldsymbol{X}(i)$ is the spatial location of node i, and $\boldsymbol{F}(i)$ is the feature vector based on intensity, color, or texture information at that node defined as:

- $\boldsymbol{F}(i) = 1$, in the case of segmenting point sets,
- $\boldsymbol{F}(i) = \boldsymbol{I}(i)$, the intensity value, for segmenting brightness images,
- $\boldsymbol{F}(i) = [v, v \cdot s \cdot sin(h), v \cdot s \cdot cos(h)](i)$, where h, s, v are the HSV values, for color segmentation,
- $\boldsymbol{F}(i) = [|\boldsymbol{I} * f_1|, ..., |\boldsymbol{I} * f_n|](i)$, where the f_i are DOOG filters at various scales and orientations as used in[14], in the case of texture segmentation.

Note that the weight $w_{ij} = 0$ for any pair of nodes i and j that are more than r pixels apart.

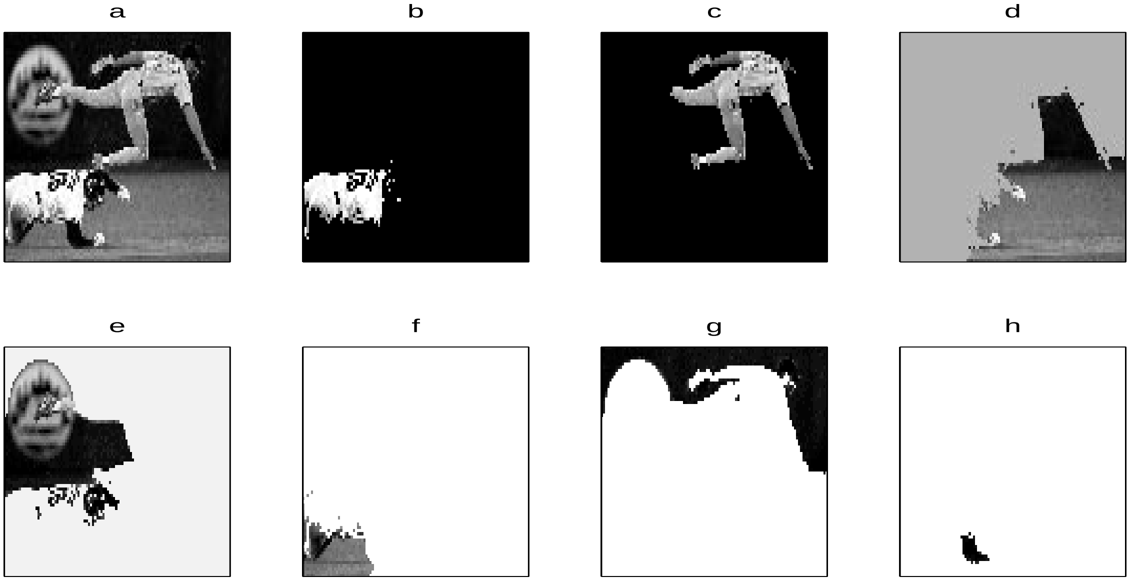


Figure 6: (a) shows a 80x100 baseball scene, image intensity is normalized to lie within 0 and 1. (b)-(h) shows the components of the partition with $Ncut$ value less than 0.04. Parameter setting: $\sigma_I = 0.01$, $\sigma_X = 4.0$, $r = 5$.

We first tested our grouping algorithm on spatial point sets similar to the one shown in figure (2). Figure (3) shows the point set and the segmentation result. As we can see from the figure, the *normalized cut* criterion is indeed able to partition the point set in a desirable way as we have argued in section (2).

Figures (4), (5), (6), and (7) shows the result of our segmentation algorithm on various brightness images. Figure (4), (5) are synthetic images with added noise. Figure (6) and (7) are natural images. Note that the "objects" in figure (7) have rather ill-defined boundary which would make edge detection perform poorly. Figure (8) shows the segmentation on a color image, reproduced in gray scale in these proceedings. The original image and many other examples can be found at web site *http://www.cs.berkeley.edu/~jshi/Grouping.*

Note that in all these examples the algorithm is able to extract the major components of scene, while ignoring small intra-component variation. As desired, recursive partitioning can be used to further decompose each piece.

Finally, we conclude with preliminary results on texture segmentation for a natural image of a zebra against a background, see figure (9). Note that the measure we have used is orientation-variant, and therefore parts of the zebra skin with different stripe orientation should be marked as seperate regions.

4.1 3D Structure-Motion Segmentation

Our grouping algorithm can also be extended to the problem of segmenting independent moving objects from image sequences. The key to our motion segmentation algorithm lies in the connection between the *shape interaction matrix*, **Q**, developed by [3] and our *normalized cut* graph partition algorithm. Without going into the details of [3]'s work,

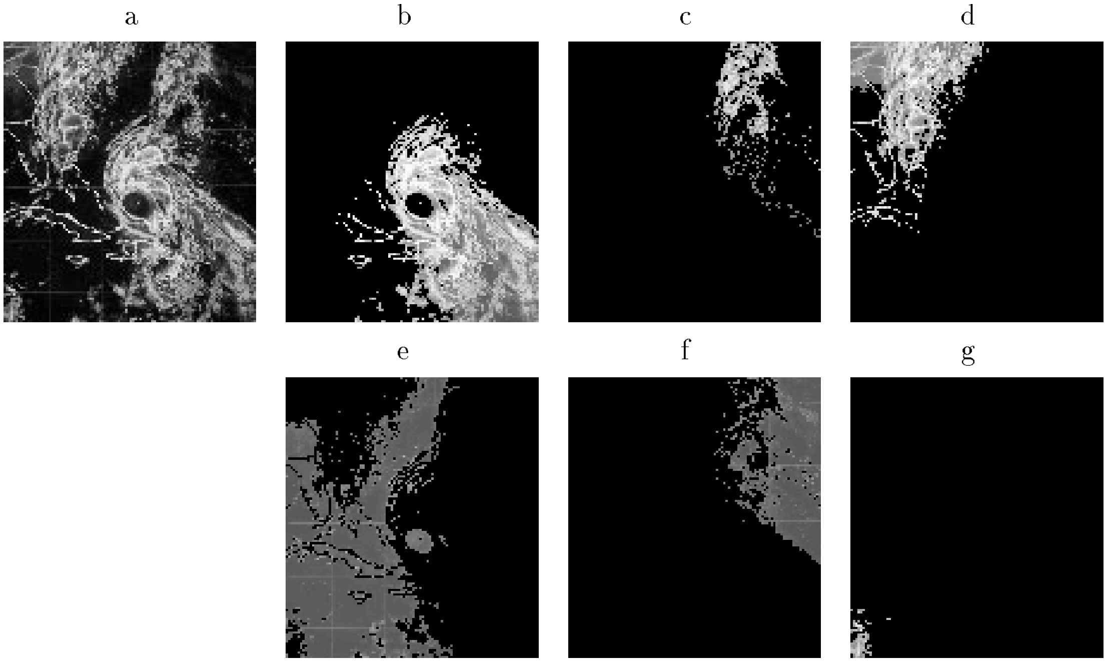


Figure 7: (a) shows a 126x106 weather radar image. (b)-(g) show the components of the partition with $Ncut$ value less than 0.08. Parameter setting: $\sigma_I = 0.005$, $\sigma_x = 15.0$, $r = 10$

we will only point out some of the important conclusions of their paper. First, the $\mathbf{Q}$ matrix can be computed directly from the image data. From a set of N tracked image features, we can set up a measurement matrix $\mathbf{W}$ of size $2N \times F$, where F is the number of frames. Factorizing $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ using SVD as in [19]. $\mathbf{Q}$ is defined as $\mathbf{Q} \equiv \mathbf{V}\mathbf{V}^T$. Second, if we have the right ordering of the elements in the measurement matrix $\mathbf{W}$, $\mathbf{Q}$ can be shown to be a block diagonal matrix, with only nonzero elements in square blocks on the diagonal. The number of blocks is exactly the number of independent moving objects. Third, wrong ordering of the elements in $\mathbf{W}$, will only permute the rows and columns of $\mathbf{Q}$. Hence if we know how to permute any given $\mathbf{Q}$ matrix back to the block diagonal form, we can solve the motion segmentation problem.

Treating each tracked feature point as a node in a complete graph, $\mathbf{Q}$ can be used to define a weight on an edge of this graph by setting $w_{ij} = \mathbf{Q}(i,j)$. For any permutation of $\mathbf{Q}$ into a block diagonal form, we can define an equivalent partitioning of the graph by taking nodes whose indexes define a block on the diagonal as one group. Since off block diagonal entries of permuted $\mathbf{Q}$ correspond to exactly those edges connecting nodes from different groups, and entries in diagonal blocks correspond to the edges connecting elements within groups, the condition that all off block diagonal entries should be as small as possible and block diagonal entries should be as large as possible translates directly into our graph partition criterion: connections between different partitions should be minimized, while connections within a partition should be maximized. As we saw early this can be achieved by simultaneously by minimizing the *normalized cut* of the graph.

We have applied our grouping algorithm from section 3 to simulated image sequences such as one shown in figure (10). Our simulation procedure is similar to that

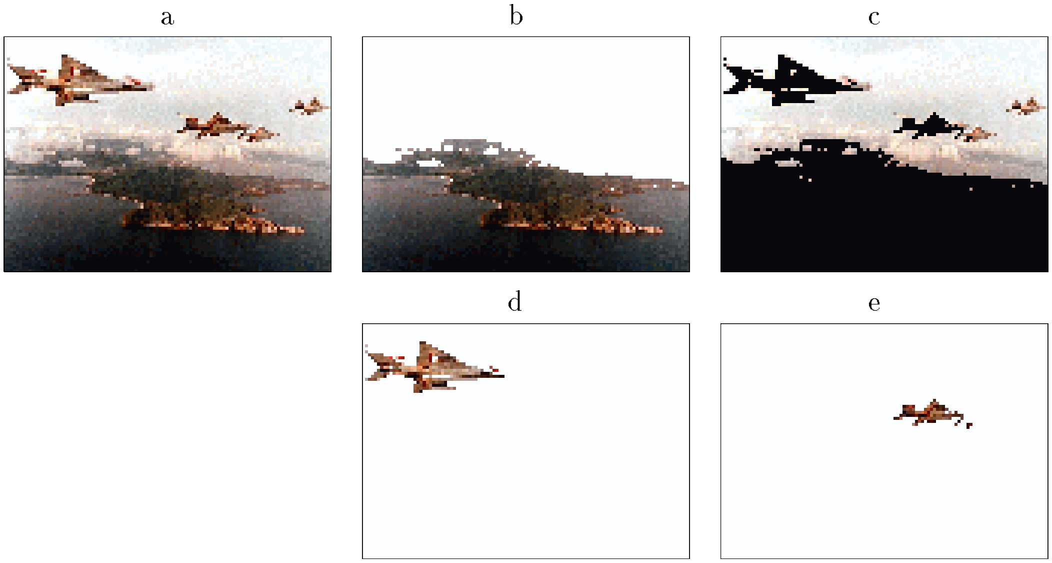


Figure 8: (a) shows a 77x107 color image. (b)-(e) show the components of the partition with $Ncut$ value less than 0.04. Parameter settings: $\sigma_I = 0.01$, $\sigma_X = 4.0$, $r = 5$.

of [3]. As shown in figure (10), 3 independent moving objects, each of which contains about 50 feature points, are created in a sequence of 20 image frames. Small amount of Gaussian noises are added to the feature positions. As shown in figure (10), all three objects are correctly separated from each other despite the image noise, and resulting $\mathbf{Q}$ is in the right block diagonal form. Experiments on real image sequences are currently planned.

5 Related graph partition algorithms

The idea of using eigenvalue problems for finding partitions of graphs originated in the work of Donath & Hoffman[5], and Fiedler[7]. Fiedler suggested that the eigenvector with the second smallest eigenvalue of the system $(\mathbf{D} - \mathbf{W})\boldsymbol{x} = \lambda\boldsymbol{x}$ could be used to split a graph. In fact the second smallest eigenvalue is called the Fiedler value, and corrsponding eigenvector the Fiedler vector. This spectral partitioning idea has been revived and further developed by several other researchers, and recently popularized by the work of [16], particularly in the area of parallel scientific computing.

In applications to several different areas, many authors have noted that the spectral partition method indeed provides good partitions of graphs [10]. Most of the theoretical work done in this area has been focused on the connection between the *ratio of cut* and the Fiedler value. A *ratio of cut* of a partition of V, $P = (A, V - A)$ is defined as $\frac{cut(A,V-A)}{min(|A|,|V-A|)}$. It was shown that if the Fiedler value is small, partitioning graph based on the Fiedler vector will lead to good *ratio of cut*[1][18]. Our derivation in section 2.1 can be adapted (by replacing the matrix $\mathbf{D}$ in the denominators by the identity matrix $\mathbf{I}$) to show that the Fiedler vector is a real valued solution to the problem of $min_{A \subset V} \frac{cut(A,V-A)}{|A|} + \frac{cut(V-A,A)}{|V-A|}$, which we can call the *average cut*.

Although *average cut* looks similar to the *normalized cut*, *average cut* does not have the important property of having a simple relationship to the *average association*, which can be analogously defined as $\frac{asso(A,A)}{|A|} + \frac{asso(V-A,V-A)}{|V-A|}$. Consequently, one can not simultaneously minimize the disassociation across the partitions, while maximizing the association within the groups. When we applied both techniques to the image segmentation problem, we found that the *normalized cut* produces better results in practice.

The generalized eigenvalue approach was first applied to graph partitioning by [6] for dynamically balancing computational load in a parallel computer. Their algorithm is motivated by [12]'s paper on representing a hypergraph in a Euclidean Space.

In the computer vision community, there are a few related approaches for image segmentation. Wu&Leahy[21] use the *minimum cut* criterion for their segmentation. Cox *et.al.* [4] seek to minimize the ratio $\frac{cut(A,V-A)}{weight(A)}, A \subset V$, where $weight(A)$ is some function of the set A. When $weight(A)$ is taken to the be the sum of the elements in A, we see that this criterion becomes one of the terms in the definition of *average cut* above. Cox *et. al.* use an efficient discrete algorithm to solve their optimization problem assuming the graph is planar.

Sarkar & Boyer[17] use the eigenvector with the largest eigenvalue of the system $\mathbf{W}\boldsymbol{x} = \lambda\boldsymbol{x}$ for finding the most coherent region in an edge map. Although their eigensystem is not directly related to the graph partitioning problem, using a similar derivation as in section (2.1), we can see that their system approximates $min_{A \subset V} \frac{asso(A,A)}{|A|}$.

6 Conclusion

In this paper, we developed a grouping algorithm based on the view that perceptual grouping should be a process that aims to extract global impressions of a scene, and that provides a hierarchical description of it. By treating the grouping problem as a graph partitioning problem, we proposed the *normalized cut* criteria for segmenting the graph. *Normalized cut* is an unbiased measure of disassociation between subgroups of a graph, and it has the nice property that minimizing *normalized cut* leads directly to maximizing the *normalized association* which is an unbiased measure for total association within the sub-groups. In finding an efficient algorithm for computing the minimum *normalized cut*, we showed that a generalized eigenvalue system provides a real valued solution to our problem.

A computational method based on this idea has been developed, and applied to segmentation of brightness, color, and texture images. Results of experiments on real and synthetic images are very encouraging, and illustrate that the *normalized cut* criterion does indeed satisfy our initial goal of extracting the "big picture" of a scene.

References

[1] N. Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[2] A. Blake and A. Zisserman. *Visual Reconstruction.* The MIT Press, 1987.

[3] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *ICCV 95*, pages 1071–1076, 1995.

[4] I.J. Cox, S.B. Rao, and Y. Zhong. Ratio regions: a technique for image segmentation. working paper, NEC Research Institute, 1996.

[5] W.E. Donath and A.J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, pages 420–425, 1973.

[6] R. Van Driessche and D. Roose. An improved spectral bisection algorithm and its application to dynamic load balancing. *Parallel Computing*, 21:29–48, 1995.

[7] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory. *Czech. Mathematical Journal*, 25(100):619–633, 1975.

[8] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *PAMI*, 6:721–741, November 1984.

[9] Golub and Van Loan. *Matrix computations.* John Hopkins Press, 1989.

[10] B. Hendrickson and R. Leland. Domain mapping of parallel scientific computations. Technical Report SAND 92-1461, Sandia National Laboratories, Albuquerque, NM, 1992.

[11] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[12] K.Fukunaga, S. Yamada, H.S. Stone, and T. Kasai. A representation of hypergraphs in the euclidean space. *IEEE Trans. Comput.*, C-33:364–367, April 1984.

[13] Y.G. Leclerc. Constructing simple stable descriptions for image partitioning. *International Journal of Computer Vision*, 3:73–102, 1989.

[14] J. Malik and P. Perona. Preattentive texture discrimination with early vision mechanisms. *Journal of Optical Society of America*, 7(2):923–932, May 1990.

[15] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions, and associated variational problems. *Comm. Pure Math.*, pages 577–684, 1989.

[16] A. Pothen, H.D. Simon, and K.P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Anal. Appl.*, 11:430–452, 1990.

[17] S. Sarkar and K.L. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. In *CVPR 96*, 1996.

[18] D.A. Spielman and S.H. Teng. Disk packings and planar separators. In *Proceedings of 12th ACM Symposium on Computational Geometry*, May 1996.

[19] C. Tomasi and T. Kanade. Shape from motion from image streams under orthograph. *IJCV*, 9(2), 1992.

[20] M. Wertheimer. Laws of organization in perceptual forms(partial translation). In W.B. Ellis, editor, *A Sourcebook of Gestalt Psycychology*, pages 71–88. Harcourt, Brace and Company, 1938.

[21] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *PAMI*, 11:1101–1113, November 1993.

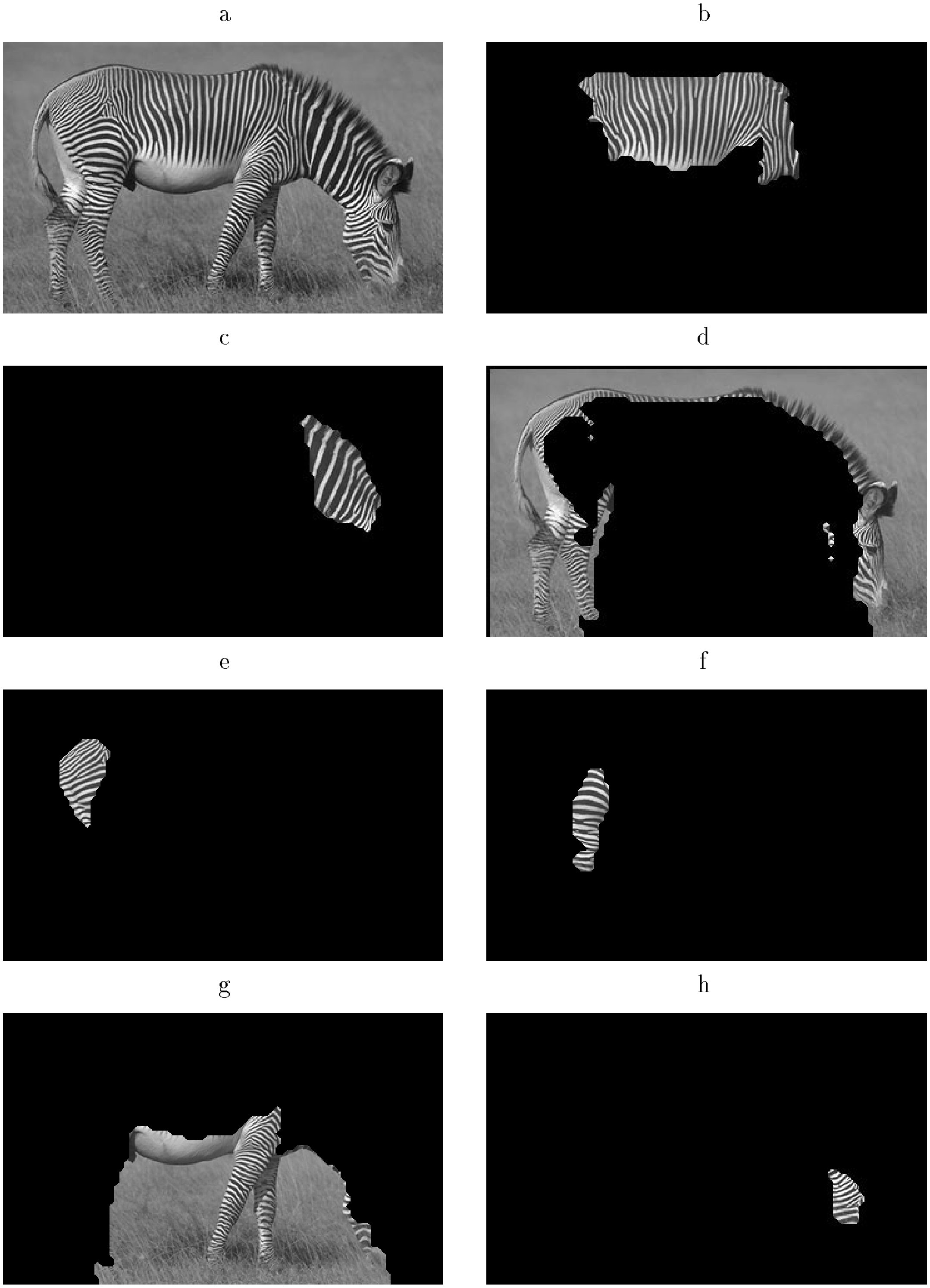


Figure 9: (a) shows an image of zebra. The remaining images show the major components of the partition. The texture features used corresponde to convolutions with DOOG filters at 6 orientations and 5 scales.

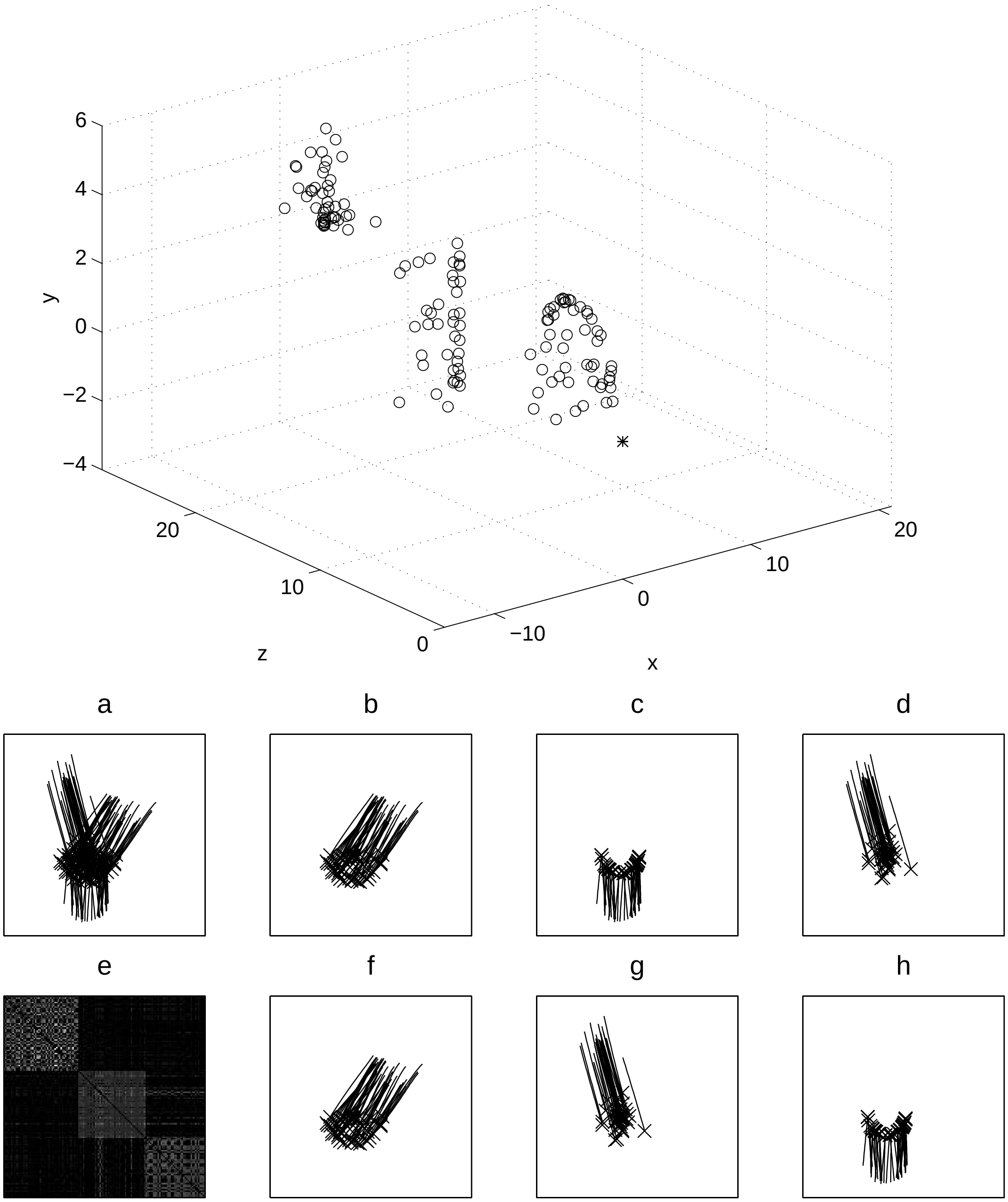


Figure 10: Top figure shows the 3D object positions('o') and the location of camera('*'). In the figure below, subplot (a) shows the feature tracks for all three objects, while individual tracks are shown in (b)-(d). (f)-(h) show the top three segmented feature tracks. The resulting permuted Q is shown in (e)