Piecewise Flat Embedding for Image Segmentation

Yizhou Yu[†] Chaowei Fang[†‡] Zicheng Liao[‡]
[†] Department of Computer Science, The University of Hong Kong
[‡] College of Computer Science and Technology, Zhejiang University

Figure 1. Given an input image (left), our piecewise flat embedding transforms the original color image into a set of embedding images (middle left to middle right), each capturing a subset of the characteristics of the input image. These characteristic images are similar to the results of other embedding methods, such as Laplacian Eigenmaps, with a main distinction that our embedding tends to be piecewise flat. This property facilitates more robust image segmentation. On the right we show a segmentation result using our embedding.

Abstract

Image segmentation is a critical step in many computer vision tasks, including high-level visual recognition and scene understanding as well as low-level photo and video processing. In this paper, we propose a new nonlinear embedding, called piecewise flat embedding, for image segmentation. Based on the theory of sparse signal recovery, piecewise flat embedding attempts to identify segment boundaries while significantly suppressing variations within segments. We adopt an L_1-regularized energy term in the formulation to promote sparse solutions. We further devise an effective two-stage numerical algorithm based on Bregman iterations to solve the proposed embedding. Piecewise flat embedding can be easily integrated into existing image segmentation frameworks, including segmentation based on spectral clustering and hierarchical segmentation based on contour detection. Experiments on BSDS500 indicate that segmentation algorithms incorporating this embedding can achieve significantly improved results in both frameworks.

1. Introduction

Image segmentation remains a critical step in many computer vision tasks, which not only include high-level visual recognition and scene understanding tasks, but also low-level photo and video processing operations, such as localized photo/video editing, enhancement and compositing. Superpixels, small image regions resulted from oversegmentation, are also valuable for accelerating a variety of computer vision and image processing algorithms.

Many mainstream image segmentation methods measure the similarity between pairs of pixels or regions before determining the boundaries of image segments. Both of these steps are highly nontrivial. An important reason is that pixel attributes (with high-frequency textures suppressed) could have smooth but relatively large variations over a single object surface due to a variety of factors including spatially varying illumination and shading. In the event of neighboring pixels sharing similar attributes, small differences between neighboring pixels can accumulate and give rise to significant differences between distant pixels even when they perceptually belong to the same object. As a result, differences between distant pixels on the same object could exceed the differences between nearby pixels on different objects, making it hard to decide where the object boundary should be.

To tackle the aforementioned challenges, a common practice computes an embedding to map pixels to a new feature space, where pixels with similar attributes are positioned closer to each other than those with dissimilar attributes. Grouping pixels in this new feature space often gives rise to improved segmentation results. Nevertheless, existing methods typically generate smooth embeddings, making it not easy to have clear-cut decisions on object boundaries. Ideally, pixel grouping would be made much easier if an embedding could map original pixels on the same object to a point cloud tightly distributed around a single point in the new feature space while points corresponding to pixels on different objects are kept apart from

each other. However, solving such an embedding requires the knowledge of object boundaries in the first place.

In this paper, we propose a new nonlinear embedding, called piecewise flat embedding (PFE), to solve the above dilemma in image segmentation. PFE attempts to identify segment boundaries while suppressing variations within segments. It is based on the theory of sparse signal recovery [10, 9] since segment boundaries consist of a sparse subset of the original pixels. We adopt an L_1-regularized energy term in the formulation to promote sparse solutions. A sparse solution in our context implies that there only exists a sparse subset of point pairs whose distances in the new feature space are sufficiently large, and the distances between the rest of the point pairs are almost zero. Such an embedding essentially form well-separated clusters, each of which is a point cloud tightly distributed around a single center. We further develop an effective numerical solution based on Bregman iterations [5] to our L_1-regularized objective function.

The proposed piecewise flat embedding can be easily integrated into existing image segmentation frameworks, including segmentation based on spectral clustering [19] and hierarchical segmentation based on contour detection [1]. Experiments on a popular benchmark (BSDS500) indicate that segmentation algorithms incorporating this embedding can achieve significantly improved results in both frameworks.

In summary, this paper has the following contributions:

- We propose piecewise flat embedding based on an L_1-regularized objective function. The formulation is analogous to that of Laplacian Eigenmaps. But the L_1-norm in the formulation makes the embedding piecewise flat instead of piecewise smooth.
- We devise a two-stage numerical algorithm based on Bregman iterations as well as effective initialization schemes to solve the proposed embedding.
- We further integrate piecewise flat embedding into two popular image segmentation frameworks. Experiments on BSDS500 confirm the effectiveness of this embedding in image segmentation tasks.

2. Related Work

Image Segmentation Image segmentation provides building blocks for higher level vision inference and perception [20], such as object detection, figure/background analysis, and scene understanding. By treating pixels in an image as a set of unordered data points, traditional clustering algorithms, such as K-means, GMM and Mean Shift [7], can be directly applied to segmentation. By modeling an image as a function defined on a 2D space, the Level Set Method [22, 6] extracts image segments as implicit regions whose boundary curve is an isocontour of the function. Another widely used method, the Graphcut algorithm [4], forms a Markov random field over the 2D pixel grid of an image. Efficient graphcut algorithms [14, 4, 17] have been developed to solve binary or multi-label image segmentation. Yet another method is based on spectral embedding [27, 3]. Spectral embedding projects pixels of an image into a low dimensional space, where similar pixels are closer to each other than dissimilar pixels. The embedded coordinates thus can be used for producing better segmentation with K-means than the original pixel values (e.g. RGB) [19]. The spectral images also provide clearer boundary information. Therefore, another segmentation approach extracts global contours from spectral images, and then generates hierarchical segmentations from integrated local and global contours[1].

Our image segmentation algorithms are closely related to the above two spectral segmentation methods. The idea is to replace the spectral images with our piecewise flat embeddings.

Manifold Embedding We first summarize previous work on manifold embedding in the context of unsupervised learning. There are two basic types of manifold embedding, local methods and global methods. Local methods, such as LLE [25] and Laplacian Eigenmap [3], attempt to preserve local relationships among points. LLE seeks an embedding that unfolds the global structure of a manifold using linear models fit to local neighborhoods. Normalized cuts [27], or the mathematically equivalent Laplacian Eigenmaps, seeks a mapping such that originally similar data points stay close in the embedding space. Global methods, such as Isomap [28] and structure preserving embedding [26], attempt to preserve local and global relationships among all data points. Isomap is based on the idea of preserving pairwise distances. However, instead of the Euclidean distance, it preserves geodesic distances (the length of the shortest paths) between pairs of points over a manifold.

Note that most of the above embedding methods attempt to minimize an objective function formulated using the squared L_2 norm, and solutions are found by solving an eigendecomposition problem. Our method differs from them in that we minimize an objective function formulated using L_1 regularization. Because of the L_1 norm, closed-form solutions do not exist any more. Instead, we use a two-stage algorithm to minimize the objective function.

The above methods were developed in the context of unsupervised learning where label information is absent. Embeddings obtained with label supervision (e.g. LDA [12]), defined as linear projections (e.g. Locality Preserving Projection [15]), or based on kernelization are also related, but out of the scope of this paper. Sparsity has been consid-

Figure 2. Embedding examples. Left column: input images; Others: four embedding images for the input image in each row.

ered in [21], which simultaneously performs dimension reduction and dictionary learning. It tries to represent high-dimensional signals in a lower-dimensional space using sparse coding.

3. Piecewise Flat Embedding

3.1. Problem Definition

As discussed earlier, when the input data is noisy and diverse, inter-cluster variations may be buried in intra-cluster variations, which makes accurate grouping (clustering) a challenging goal to accomplish. Thus, it is much desired to have an embedding that maps the original data into a different space, where intra-cluster variations are significantly suppressed while inter-cluster variations are still preserved with respect to the underlying ground truth. Since data points lying on cluster boundaries are relatively sparse in comparison to the rest of the data, we apply the theory of sparse signal recovery [10, 9] and devise a new data embedding technique that facilitates the discovery of sparse cluster boundaries while suppressing intra-cluster variations.

Given n data points $\mathbf{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ in R^m, we would like to transform them into a new d-demensional space. Let $\mathbf{Y}$ be a $n \times d$ matrix, each row in $\mathbf{Y}$ has d elements, and they represent the d coordinates of a data point in the new space. We use Y_i to represent the i-th row of $\mathbf{Y}$. Each column in Y has n elements. They represent the same coordinate in the new space across all data points. Our goal is to find an embedding $\mathbf{Y}$ that minimizes the following objective function,

$$\min_{\mathbf{Y}} \sum_{ij} W_{ij} \left\| Y_i - Y_j \right\|_1 \quad s.t. \;\; \mathbf{Y}^T\mathbf{D}\mathbf{Y} = \mathbf{I} \qquad (1)$$

where $\mathbf{W}$ is known as the affinity matrix, and $\mathbf{D}$ is a diagonal weight matrix with $D_{ii} = \sum_j W_{ji}$. Our embedding formulation is inspired by Normalized Cuts [27] and Laplacian Eigenmap [3], which is defined as follows.

$$\min_{\mathbf{Y}} \sum_{ij} W_{ij} \left\| Y_i - Y_j \right\|_2^2 \quad s.t. \;\; \mathbf{Y}^T\mathbf{D}\mathbf{Y} = \mathbf{I} \qquad (2)$$

where $\mathbf{W}$ and $\mathbf{D}$ are defined in the same way.

Although problem (1) and problem (2) are similar. The most obvious difference is that we use the L_1 norm in problem (1) while the formulation of Laplacian eigenmap inherits a common trait of all mainstream embedding methods, which is the use of the squared L_2 norm in the objective function. In fact, such a small difference has the following important implication. According to the theory of sparse signal recovery [10, 9], the L_1 norm promotes sparse solutions while the L_2 norm does not. A sparse solution in our context implies that there only exists a sparse subset of point pairs whose distances in the new space are sufficiently large, and the distances between the rest of the point pairs are almost zero. Such a sparse solution suggests points with a large distance in the new space should belong to different clusters and points with a very small distance in the new space should belong to the same cluster. Therefore, performing data clustering in the new space becomes a straightforward process. Although the affinity matrix in problem (2) attempts to move points with similar attributes closer in the new space, the resulting pairwise distances in the new space still follows a relatively smooth distribution, and the distinction between similar points and dissimilar points is not as clear as in our results.

3.2. Numerical Solution

Solving problem (1) is challenging because it has both L_1 regularization and an orthogonality constraint while most of existing numerical methods can handle one of them only. Here we develop a numerical solution to problem (1) by effectively nesting two existing methods that handle L_1 regularization and orthogonality constraints respectively.

A. *Solution to the Orthogonality Constraint* First of all, a numerical solver that is capable of enforcing the orthogonality constraint is required. Since the objective function in (1) is convex, we apply the Splitting Orthogonality Constraint (SOC) algorithm in [18]. Following the derivation of the SOC algorithm, we define $\mathbf{P} = \mathbf{D}^{1/2}\mathbf{Y}$, and rewrite

problem 1 as

$$\min_{\mathbf{Y}} \sum_{ij} W_{ij}||Y_i - Y_j||_1 \quad s.t.\ \mathbf{D}^{1/2}\mathbf{Y} = \mathbf{P},\ \mathbf{P}^T\mathbf{P} = \mathbf{I}, \tag{3}$$

which can be iteratively solved using Bregman iterations [5] as follows:

(a)

$$\mathbf{Y}^{(k+1)} = \arg\min_{\mathbf{Y}} \sum_{ij} W_{ij}||Y_i - Y_j||_1 + \frac{r}{2}||\mathbf{D}^{1/2}\mathbf{Y} - \mathbf{P}^{(k)} + \mathbf{B}^{(k)}||_2^2; \tag{4}$$

(b)

$$\mathbf{P}^{(k+1)} = \arg\min_{\mathbf{P}} ||\mathbf{P} - (\mathbf{D}^{1/2}\mathbf{Y}^{(k+1)} + \mathbf{B}^{(k)})||_2^2, \quad s.t. \quad \mathbf{P}^T\mathbf{P} = \mathbf{I}; \tag{5}$$

(c)

$$\mathbf{B}^{(k+1)} = \mathbf{B}^{(k)} + \mathbf{D}^{1/2}\mathbf{Y}^{(k+1)} - \mathbf{P}^{(k+1)}, \tag{6}$$

where $\mathbf{B}$ is an auxiliary matrix, and $\mathbf{B}^{(0)} = 0$.

According to Theorem 2.1 in [18], the spherically constrained problem in step (b) has the following closed form solution:

$$\begin{aligned} \mathbf{D}^{1/2}\mathbf{Y}^{(k+1)} + \mathbf{B}^{(k)} &= \mathbf{U}\mathbf{\Sigma}_{n\times d}\mathbf{V}^T, \\ \mathbf{P}^{(k+1)} &= \mathbf{U}\mathbf{I}_{n\times d}\mathbf{V}^T, \end{aligned} \tag{7}$$

where $\mathbf{U}$ and $\mathbf{V}$ are matrices with orthogonal columns from the SVD decomposition in the first step.

B. *Solution to the L_1-norm Problem* Note that the subproblem in step (a) of the above numerical solution inherits the L_1-regularized energy term from problem 1. There exist many numerical solutions for optimization problems with L_1 regularization, and most of them can be applied here. To be consistent with the top-level solution presented above, we apply the Split Bregman algorithm in [13] to solve the subproblem in step (a) efficiently. The Split Bregman algorithm also relies on Bregman iterations. By introducing auxiliary variables, the Split Bregman algorithm solves an L_1 regularized optimization problem iteratively by transforming the original optimization into a series of differentiable unconstrained convex optimization problems. The definition of any convex optimization in the series depends on the auxiliary variables passed from the previous iteration, and convergence can be achieved within a relatively small number of iterations.

For notational convenience in the rest of this section, let $Y_v^{(k)}$, $P_v^{(k)}$ and $B_v^{(k)}$ be vectors obtained by flattening matrices $\mathbf{Y}^{(k)}$, $\mathbf{P}^{(k)}$ and $\mathbf{B}^{(k)}$ respectively, where flattening means concatenating matrix columns. Further, let $\mathbf{M} = \{M_{ij}\}$ be a $(n(n-1)/2) \times n$ matrix, where $M_{ki} = w_{ij}, M_{kj} = -w_{ij}$ if p_i and p_j are data points that form the k-th pair, and define two new matrices, $\mathbf{L}$ and $\tilde{\mathbf{D}}$ as follows,

$$\mathbf{L}_{(dn(n-1)/2)\times(dn)} = \begin{pmatrix} \mathbf{M} & & & \\ & \mathbf{M} & & \\ & & \dots & \\ & & & \mathbf{M} \end{pmatrix},$$

$$\tilde{\mathbf{D}}_{(dn(n-1)/2)\times(dn)} = \begin{pmatrix} \mathbf{D} & & & \\ & \mathbf{D} & & \\ & & \dots & \\ & & & \mathbf{D} \end{pmatrix}.$$

Then the problem in step (a) can be rewritten as follows,

$$Y_v^{(k+1)} = \arg\min_{Y_v} ||\mathbf{L}Y_v||_1 + \frac{r}{2}||\tilde{\mathbf{D}}^{1/2}Y_v - P_v^{(k)} + B_v^{(k)}||_2^2, \tag{8}$$

which can be solved by iterating the following steps until $||Y_v^{(k,l+1)} - Y_v^{(k,l)}|| \le \epsilon$:

(a.1)

$$Y_v^{(k,l+1)} = \arg\min_{Y_v} \frac{\lambda}{2}||\mathbf{L}Y_v + b^l - d^l||_2^2 + \frac{r}{2}||\tilde{\mathbf{D}}^{1/2}Y_v - P_v^{(k)} + B_v^{(k)}||_2^2; \tag{9}$$

(a.2)

$$d^{l+1} = \text{Shrink}(\mathbf{L}Y_v^{(k,l+1)} + b^l, \frac{1}{\lambda}); \tag{10}$$

(a.3)

$$b^{l+1} = b^l + \mathbf{L}Y_v^{(k,l+1)} - d^{l+1}. \tag{11}$$

Note that in the above steps, b and d are two auxiliary vectors, $b^0 = d^0 = 0$, $Y_v^{(k,0)} = Y_v^{(k)}$, and ϵ is a predefined error tolerance. The problem in (a.1) is a least-squares problem and can be easily minimized using its normal equation. In (a.2), suppose $\mathbf{z} = \text{Shrink}(\mathbf{y}, \gamma)$. Then $z_i = \text{sign}(y_i)\max(|y_i| - \gamma, 0)$.

C. *Two-Stage Implementation* In practice, we devise the following two-stage implementation to obtain a high-quality solution efficiently.

Stage I This stage implements the full numerical solution with nested Bregman iterations. A large penalty coefficient for the orthogonality constraint is used. The iterations in the outer loop make different dimensions of the embedded data orthogonal to each other to remove redundancy among them. This is important in avoiding naive solutions with

Figure 3. Initialization. The top two rows show pixelwise density defined by the 16 components of the Gaussian Mixture Model (GMM) learned from an input image. Images in the bottom row show a GMM-based initialization of the 4 dimensions of our embedding. Each image in the bottom represents the mixed density of 8 components of the GMM. Specifically, the first image in the bottom row records the mixed density of the 8 components in the first row; the second image in the bottom records the mixed density of the first 4 components in the top two rows; the third image in the bottom records the mixed density of the 1st, 2nd, 5th and 6th components in the top two rows; and the fourth image in the bottom records the mixed density of the 1st, 3rd, 5th and 7th components in the top two rows.

highly redundant or even duplicate dimensions. However, orthogonality is a highly non-convex constraint that prevents the objective function in problem (1) settling into a truly low-energy state; and more importantly, it is not absolutely necessary for us to pursue an embedding whose dimensions are strictly orthogonal as long as there is not too much redundancy across different dimensions. Therefore, following a few iterations of the full numerical solution, we optionally relax the orthogonality constraint in a second stage.

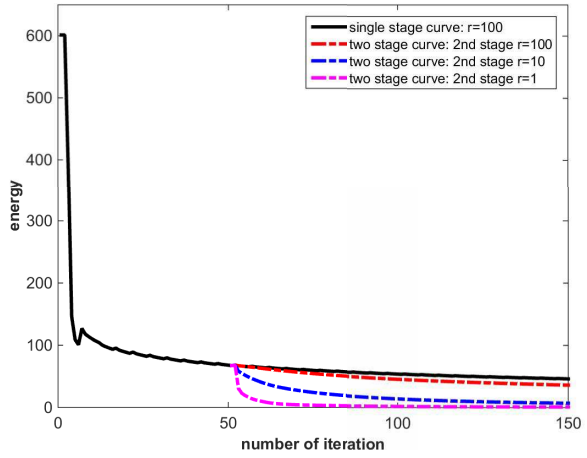

Figure 4. Energy curves. The black curve shows how the energy decreases with regular single-stage Bregman iterations with $\lambda = 1000, r = 100$. The dashed curves show how energy evolves with our 2-stage algorithm. The first stage is the same as the black curve. The second stage starts from iteration 50. The three dashed curves show varying convergence rates using different parameter values ($r = 100, 10, 1$ respectively for the red, blue and pink curves.)

Stage II This stage only executes the Bregman iterations in the inner loop to minimize the L_1-regularized objective function without performing the SVD in the outer loop to strictly enforce orthogonality. Such relaxation of the orthogonality constraint allows the L_1-regularized objective function to reach a lower energy.

Figure 4 shows an example of the energy curve during Stage I and Stage II. It can be verified in this figure that our two-stage scheme can reach a lower energy than a single-stage scheme.

D. ***Initialization and Parameters*** Our iterative numerical solution requires an initialization. When the number of dimensions of the new space is at most 3, $\mathbf{Y}$ is simply initialized with some or all of the mean-subtracted color channels of the input image. When the number of dimensions of the new space is larger than 3, we rely on GMM clustering to initialize $\mathbf{Y}$. Suppose the number of dimension is d. We first perform GMM clustering with 2^d Gaussians on pixelwise color channels. The probability densities resulted from GMM clustering are then encoded into the d dimensions of the new space. Suppose the Gaussians in the GMM have been ordered into a linear sequence. Each dimension of the new space records the mean-subtracted mixed probability density of half of the Gaussians in this sequence. For example, the first dimension records the mixed probability density of the first half of the Gaussians; then we divide the sequence in the middle into two equal subsequences, and the second dimension records the mixed probability density of the first half of the Gaussians in both subsequences. Once the initial $\mathbf{Y}$ has been set, $\mathbf{P}$ is initialized with an orthogonalized version of $\mathbf{D}^{1/2}\mathbf{Y}$ using (7).

In Stage I, we set the maximum number of outer iterations to 10 and the number of inner iterations to 5. In Stage II, the maximum number of inner iterations is set to 100. It takes around 15 minutes for our prototype MATLAB code to compute an embedding for an input image on an Intel 3.3GHz processor. The parameter setting for λ and r will be given in the following section.

Figure 2 shows a few examples of embeddings.

4. Segmentation via Piecewise Flat Embedding

An image with resolution $p \times q$ gives rise to a set of $n = p \times q$ data points. By computing an affinity value between neighboring pixels, we have an $n \times n$ sparse affinity matrix W. Using our method, we can obtain a piecewise flat embedding of the input data in a new d-dimensional space. We have integrated our embedding results in two popular image segmentation frameworks.

4.1. Segmentation by Clustering

The first segmentation framework we have considered is based on spectral clustering [19]. Spectral clustering first computes a set of eigenvectors. The values in the eigenvectors corresponding to the same pixel form a d-dimensional feature vector at that pixel. It then runs standard clustering, such as K-means, on the pixelwise feature vectors. A variant of spectral clustering, named weighted spectral clustering in the rest of the paper, reweighs the i-th feature coordinate by $1/\sqrt{\lambda_i}$, where λ_i is the i-th eigenvalue.

Our revised clustering-based segmentation simply replaces the eigenvectors with our piecewise flat embedding of the pixels. That is, each row of the matrix $\mathbf{Y}$ becomes the d-dimensional feature vector of the corresponding pixel. We do not have a version corresponding to weighted spectral clustering because our embedding is not a spectral one. During the computation of our embedding, the parameters λ and r are set to 10000 and 100 respectively in stage I, and r is reduced to 10 in stage II.

Since our embedding is piecewise flat, pixels of the same region are tightly distributed in the feature space. In contrast, existing embedding techniques, such as Laplacian Eigenmaps, do not have such a property. A Laplacian eigenmap is obtained by minimizing the squared L_2 distance between neighboring pixels. The resulting eigenmaps are piecewise smooth but not piecewise flat. Therefore pixels from the same region may still have reasonably large distances among them in the feature space, and may not be grouped together into the same cluster. This is the reason why large or elongated regions often break up in the middle in segmentation results based on spectral clustering.

4.2. Contour-Driven Hierarchical Segmentation

One strategy to avoid the drawback of spectral clustering is deriving contour information from the eigenmaps, and then form segments with respect to contours using hierarchical clustering [1]. To make the algorithm robust, their method also combines local edges with global contours when constructing the contour probability map. It also builds a hierarchical clustering tree from watershed segmentation results. This makes it more flexible to choose a segmentation granularity. Our contour-oriented segmentation follows the same pipeline, except that global contour information is extracted from our embedding results instead of the eigenmaps. When computing our piecewise flat embedding, we set the parameters λ and r to 1000 and 100 respectively in stage I, and reduce r to 10 in stage II.

Again, since our embedding is piecewise flat, coordinates from the embedding have almost zero gradients everywhere except at region contours where gradients have a large magnitude. Contour maps extracted from our embedding are both clear and clean without many spurious edges.

Figure 5 shows our segmentation pipeline. A gallery of segmentation results from various algorithms can be found in Figure 6.

5. Experimental Results

In this section, we evaluate the performance of our piecewise flat embedding (PFE) for segmentation on the BSDS500 dataset [1]. We have conducted experiments within the two aforementioned segmentation frameworks: (a) segmentation by clustering and (b) contour-driven hierarchical segmentation, and compared revised segmentation algorithms incorporating our embedding against the original and other existing algorithms.

We first incorporate piecewise flat embedding into the clustering-based segmentation approach, and perform im-

Table 1. Comparison of Normalized Cut (NCut), spectral clustering (SC), weighted spectral clustering (WSC) and our method (PFE+K-means) on BSDS500 using the affinity in the original Normalized Cut.

method	Covering		PRI		VI	
	fixed	dynamic	fixed	dynamic	fixed	dynamic
NCut	0.33	0.40	0.75	0.76	2.77	2.39
SC	0.36	0.44	0.75	0.77	2.68	2.24
WSC	0.36	0.44	0.75	0.77	2.63	2.21
Ours	**0.46**	**0.52**	**0.77**	**0.79**	**2.21**	**1.91**

Table 2. Comparison between existing segmentation-by-clustering methods (spectral clustering, weighted spectral clustering) and our method (PFE+K-means) on BSDS500 using the affinity based on local contour information (mPb in [1]).

method	Covering		PRI		VI	
	fixed	dynamic	fixed	dynamic	fixed	dynamic
SC	0.35	0.45	0.76	0.77	2.66	2.17
WSC	0.35	0.44	0.76	0.77	2.67	2.20
Ours	**0.45**	**0.56**	**0.78**	**0.81**	**2.26**	**1.77**

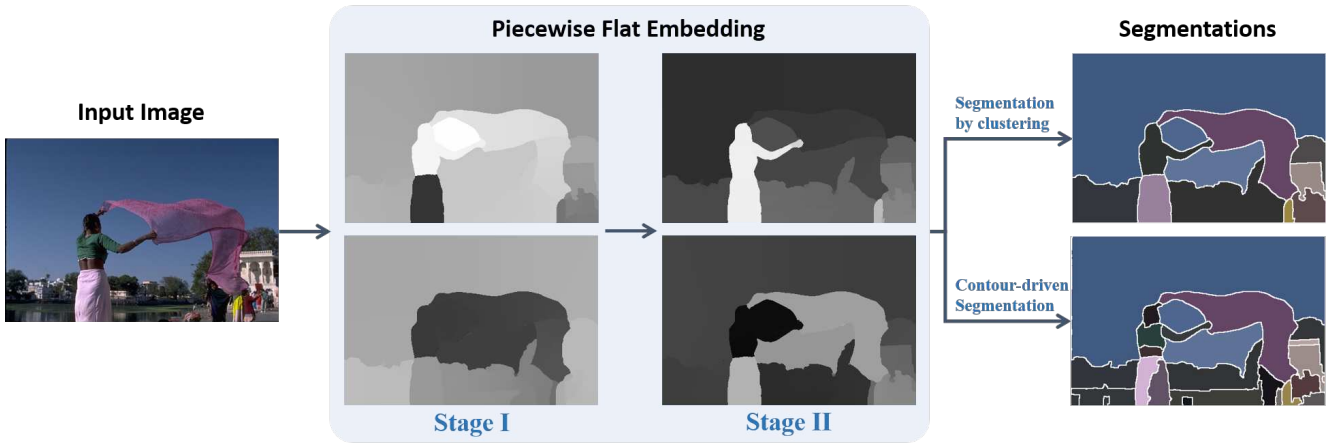

Figure 5. Pipeline. Given an input image, our method generates a piecewise flat embedding of the image in a two-stage optimization. The images corresponding to the embedding are used for image segmentation with a clustering-based method, or a contour-driven method.

age segmentation using two types of affinity matrices: the affinity used in the original Normalized Cut algorithm [27], and the affinity computed using local contour information (mPb in [1]). We have compared against three existing spectral methods using the affinity in the original Normalized Cut: spectral clustering; weighted spectral clustering; and Normalized Cut. Spectral clustering and the original Normalized Cut use the same eigenvectors, but differ in the way they use these eigenvectors. Spectral clustering assigns each pixel a d-dimensional feature vector taken from d eigenvectors and runs standard K-means clustering on these feature vectors afterwards, while the original Normalized Cut recursively partitions an image into smaller regions using the eigenvector with the second smallest eigenvalue. Weighted spectral clustering divides every eigenvector by the square root of its corresponding eigenvalue before running K-means clustering. Our clustering-based segmentation performs K-means clustering on the coordinates from piecewise flat embedding. We have also compared against spectral clustering and weighted spectral clustering using the affinity computed using local contours. Final segmentation results are evaluated using three criteria, Covering, PRI and VI, discussed in BSDS segmentation benchmarks [1].

All the clustering-based methods need to be given the number of segments at the beginning. We design the following two schemes: the *fixed* scheme and the *dynamic* scheme. The fixed scheme uses the number of segments in the groundtruth images. If there are multiple groundtruth images, we run every algorithm multiple times, once for the number of segments in each groundtruth image, and finally average the performance from these multiple runs. The dynamic scheme chooses the number of segments that falls between 5 and 25, and produces the best performance. In our embedding, we use four channels each initialized with joint probabilities of a distinct subset of Gaussians from a Gaussian Mixture Model, as discussed earlier, in all experiments (except indicated otherwise). For other methods that require multiple NCut eigenvectors, we run each method three times using 4, 8, and 16 eigenvectors respectively, and the best performance from these three runs is reported.

Table 1 shows comparison results using the affinity in the original Normalized Cut. Table 2 shows comparison results using the contour-based affinity. Segmentation based on our piecewise flat embedding produces the best result in all scenarios. In particular, our method achieves 0.52 and 0.56 on Covering (dynamic scheme) in these tables, which are respectively 18.2% and 24.4% higher than the second best.

We have also incorporated piecewise flat embedding into contour-driven hierarchical segmentation methods named gPb [1] and MCG [2], which integrate contour information derived from both local features and global eigenmaps. In our revised algorithms, we simply replace global eigenmaps in these methods with our piecewise smooth embedding, which is computed using locally derived contour maps in these methods. Then we compare the segmentation performance of our revised algorithms with the original algorithms and other existing algorithms. The comparison results are shown in Table 3, which clearly demonstrates hierarchical segmentation based on our embedding achieves the best results among all considered algorithms. Although the algorithm in [23] achieves state-of-the-art results on contour detection, integrating its local contours with the segmentation algorithm in [1] does not lead to good segmentation performance.

The contours extracted from our embedding is a type of global contours. To verify the effectiveness of our global contours, we perform contour-driven hierarchical segmentation using global contours only by skipping local contour information. The results are shown in Table 4, which compares our results against the results obtained using the original global contours (in gPb) computed from Laplacian

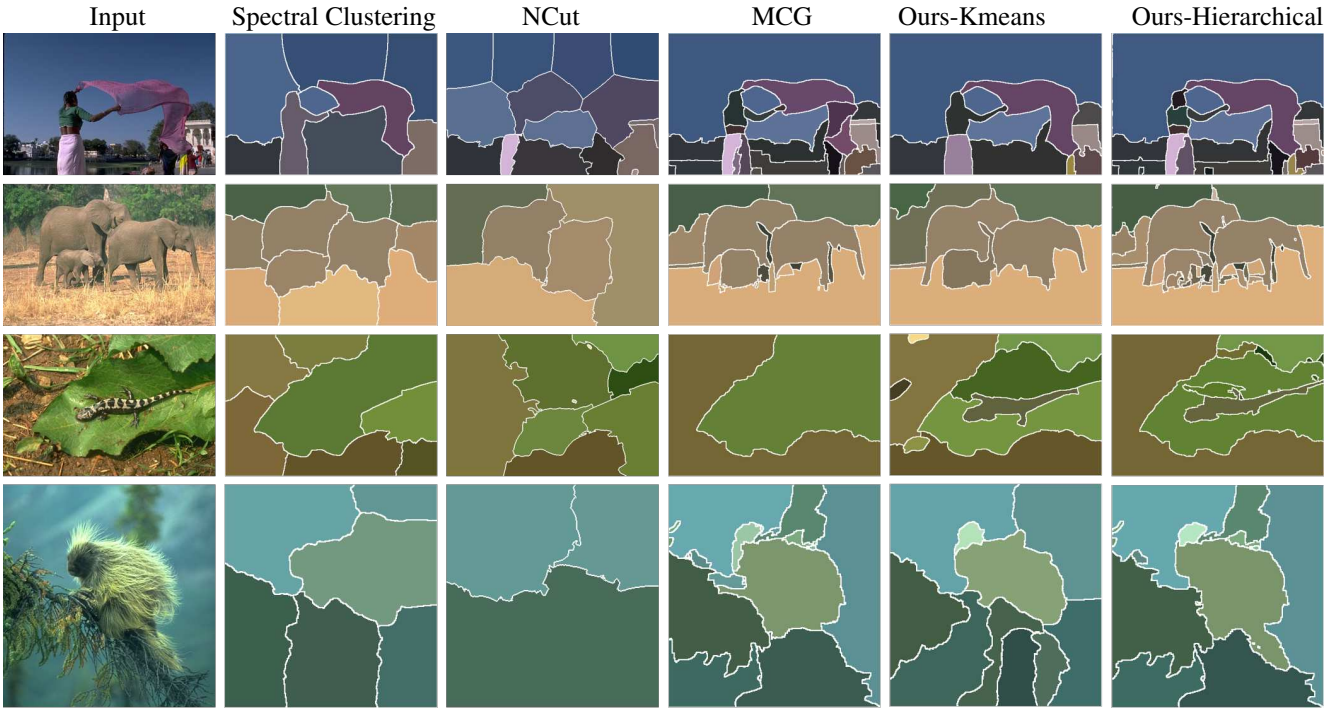

Figure 6. Comparison of segmentation results. Spectral clustering uses the mPb affinity; Normalized Cut and MCG are based on their original configuration; Our result by K-means clustering uses the Normalized Cut affinity; Our hierarchical segmentation follows the MCG pipeline but uses our piecewise flat embedding to compute global contours.

Table 3. Segmentation performance on BSDS500. 'PFE+mPb' denotes results generated from the segmentation algorithm in [1] using global contours from our piecewise flat embedding. 'PFE+MCG' denotes results generated from the segmentation algorithm in [2] using global contours from our piecewise flat embedding.'SCG-owt-ucm' means integrating the segmentation algorithm in [1] and the local contours from [23].

method	Covering			PRI		VI	
	ODS	OIS	Best	ODS	OIS	ODS	OIS
MS-NCut [8]	0.45	0.53	0.67	0.78	0.80	2.23	1.89
Felz-Hutt [11]	0.52	0.57	0.69	0.80	0.82	2.21	1.87
SCG-owt-ucm	0.51	0.56	0.66	0.78	0.83	1.98	1.84
Mean Shift [7]	0.54	0.58	0.66	0.79	0.81	1.85	1.64
Hoiem [16]	0.56	0.60	-	0.80	0.77	1.78	1.66
gPb-owt-ucm [1]	0.59	0.65	0.74	0.83	0.86	1.69	1.48
ISCRA [24]	0.59	0.66	-	0.82	0.85	1.60	1.42
MCG [2]	0.61	0.66	0.76	0.83	0.86	1.57	1.39
Ours (PFE+mPb)	**0.62**	0.67	0.76	**0.84**	0.86	1.61	1.43
Ours (PFE+MCG)	**0.62**	**0.68**	**0.77**	**0.84**	**0.87**	**1.56**	**1.36**

eigenmaps. It can be seen that segmentation results based on global contours derived from our piecewise flat embedding are not only clearly better than those from eigenmap-based global contours, but also better than the results from the complete gPb-owt-ucm algorithm, shown in Table 3. In this comparison, our piecewise flat embeddings are obtained using the local contour information computed in [1].

Table 4. Comparison of contour-driven hierarchical segmentation using global contours only. 'sPb-owt-ucm' means segmentation results from the algorithm in [1] using global contours from NCut only; our results (PFE-owt-ucm) are generated with the algorithm in [1] using global contours from piecewise flat embedding only.

method	Covering			PRI		VI	
	ODS	OIS	Best	ODS	OIS	ODS	OIS
sPb-owt-ucm	0.58	0.64	0.73	0.82	0.85	1.71	1.49
PFE-owt-ucm	**0.61**	**0.66**	**0.74**	**0.83**	**0.86**	**1.64**	**1.46**

6. Conclusions

We have presented a new nonlinear embedding called piecewise flat embedding for image segmentation. We adopt an L_1-regularized energy term in the formulation to promote sparse solutions. We further devise an effective two-stage numerical algorithm based on Bregman iterations to solve the proposed embedding. Piecewise flat embedding can be easily integrated into existing image segmentation frameworks. Experiments on BSDS500 indicate that segmentation algorithms incorporating this embedding can achieve significantly improved results.

Acknowledgment

This work was partially supported by Hong Kong Research Grants Council under General Research Funds (HKU719313) and Zhejiang Natural Science Foundation (ZJNSF Q15F020006).

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, May 2011. 2, 6, 7, 8

[2] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 7, 8

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001. 2, 3

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE PAMI*, 23(11):1222–1239, 2001. 2

[5] L. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex optimization. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967. 2, 4

[6] T. Chan and L. Vese. Active contours without edges. *IEEE IP*, 2001. 2

[7] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 2002. 2, 8

[8] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *CVPR*, 2005. 8

[9] D. Donoho and B. Logan. Signal recovery and the large sieve. *SIAM J. Appl. Math.*, 52(2):577–591, 1992. 2, 3

[10] D. Donoho and P. Stark. Uncertainty principles and signal recovery. *SIAM J. Appl. Math.*, 49(3):906–931, 1989. 2, 3

[11] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 2004. 8

[12] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936. 2

[13] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM J. Img. Sci.*, 2(2):323–343, Apr. 2009. 4

[14] D. Greig, B. Porteous, and A. Seheult. Exact maximum a posteriori estimation for binary images. *Journal o the Royal Statistical Society, Series B*, 51(2):271–279, 1989. 2

[15] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems 16*, 2003. 2

[16] D. Hoiem, A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. *International Journal of Computer Vision*, 91:328–346, 2011. 8

[17] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE PAMI*, 2004. 2

[18] R. Lai and S. Osher. A splitting method for orthogonality constrained problems. *J. Sci. Comput.*, 58(2):431–449, Feb. 2014. 3, 4

[19] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *IJCV*, 2001. 2, 6

[20] D. Marr. *Vision: a computational investigation into the human representation and processing of visual information*. MIT Press, 1982. 2

[21] H. V. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa. Sparse embedding: A framework for sparsity promoting dimensionality reduction. In *ECCV*, 2012. 3

[22] S. Osher and J. Sethian. Fronts propagation with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 1988. 2

[23] X. Ren and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012. 7, 8

[24] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2018, 2013. 8

[25] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323 – 2326, 2000. 2

[26] B. Shaw and T. Jebara. Structure preserving embedding. In *International Conference on Machine Learning*, 2009. 2

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, Aug. 2000. 2, 3, 7

[28] J. B. Tenenbaum, V. Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. 2