Spectral Segmentation with Multiscale Graph Decomposition

Timothée Cour[1] Florence Bénézit[2] Jianbo Shi[3]
[1,3]Computer and Information Science [2]Applied Mathematics Department
University of Pennsylvania Ecole Polytechnique
Philadelphia, PA 19104 91128 Palaiseau Cedex, FRANCE
timothee@seas.upenn.edu florence.benezit@polytechnique.org

Abstract

We present a multiscale spectral image segmentation algorithm. In contrast to most multiscale image processing, this algorithm works on multiple scales of the image in parallel, without iteration, to capture both coarse and fine level details. The algorithm is computationally efficient, allowing to segment large images. We use the Normalized Cut graph partitioning framework of image segmentation. We construct a graph encoding pairwise pixel affinity, and partition the graph for image segmentation. We demonstrate that large image graphs can be compressed into multiple scales capturing image structure at increasingly large neighborhood. We show that the decomposition of the image segmentation graph into different scales can be determined by ecological statistics on the image grouping cues. Our segmentation algorithm works simultaneously across the graph scales, with an inter-scale constraint to ensure communication and consistency between the segmentations at each scale. As the results show, we incorporate long-range connections with linear-time complexity, providing high-quality segmentations efficiently. Images that previously could not be processed because of their size have been accurately segmented thanks to this method.

1. Introduction

There are two things you could do to make image segmentation difficult: 1) camouflage the object by making its boundary edges faint, and 2) increase clutter by making background edges highly contrasting, particularly those in textured regions. In fact, such situations arise often in natural images, as animals have often evolved to blend into their environment.

Several recent works have demonstrated that multiscale image segmentation can produce impressive segmentation results under these difficult conditions. Sharon, et. al. [9] uses an algebraic multi-grid method for solving the normalized cut criterion efficiently, and uses recursive graph coarsening to produce irregular pyramid encoding region based grouping cues. Yu [11] constructs a multiple level graph encoding edge cues at different image scales, and optimizes the average Ncut cost across all graph levels. Zhu et. al. [1] explicitly controls the Markov chain transitions in the space of graph partitions by splitting, merging and re-grouping segmentation graph nodes.

We argue that there are in fact three orthogonal issues in multiscale image segmentation: 1) *multiscale signal processing* to enhance faint contours [6, 2]; 2) *image region similarity cues* at multiple scales provides texture/shape cues for larger regions; 3) *propagation* of local grouping cues across multiple ranges of spatial connections allows us to detect coherent regions with faint boundary.

Sharon [9], Zhu [1]'s approaches focus on the last two issues, and Yu [11] focuses on the first and third. The primary motivation underlying all these approaches is that local information propagates faster with long range connections across image regions, and computation converges faster both in graph partitioning and MRF probabilistic formulation. Both Sharon and Zhu advocated *data driven adaptive* coarsening of an image region/segmentation graph as an essential step in multiscale segmentation. This conclusion is partly justified by the failure of most multiscale segmentation algorithms [8, 7, 4] which use simple geometric coarsening of images: typically fine level details along object boundaries are lost due to coarsening error.

We focus on the third issue of multiscale propagation of grouping cues *in isolation*. We show that simple geometric coarsening of the image region/segmentation graph *can* work for multiscale segmentation. The key principle is that segmentation across different spatial scales should be processed in *parallel*. We specify the constraint that segmentation must be self-consistent across the scales. This constraint forces the system to seek an "average" segmentation across all scales. We show our multiscale segmentation algorithm can precisely segment objects with both fine and coarse level object details.

The advantage of graphs with long connections comes with a great computational cost. If implemented naively, segmentation on a fully connected graph G of size N would require at least $O(N^2)$ operations. This paper develops an efficient computation method of multiscale image segmentation in a constrained Normalized cuts framework. We show that multiscale Normalized cuts can be computed in *linear time*.

This paper is organized as follows. In section 2, we review the basics of graph based image segmentation. In section 3 and 4, we show how to compress a large fully connected graph into a multiscale graph with $O(N)$ total graph weights. In section 5, 6, we demonstrate efficient optimization of multiscale spectral graph partitioning with $O(N)$ running time. We conclude with experiments in section 7.

2. Graph based image segmentation

Given an image **I**, we construct a graph $G = (V, E, W)$, with the pixels as graph nodes V, and pixels within distance $\leq G_r$ are connected by a graph edge in E. A weight value $W(i, j)$ measures the likelihood of pixel i and j belonging to the same image region. Partitioning on this graph provides image regions segmentation.

2.1. Encoding Graph Edges

The overall quality of segmentation depends on the pairwise pixel affinity graph. Two simple but effective local grouping cues are: intensity and contours.

Intensity Close-by pixels with similar intensity value are likely to belong to one object:

$$W_I(i, j) = e^{-||X_i - X_j||^2/\sigma_x - ||I_i - I_j||^2/\sigma_I} \quad (1)$$

where X_i and I_i denote pixel location and intensity. Connecting pixels by intensity is useful to link disjoint object parts. But because of texture clutter, the intensity cue alone often gives poor segmentations.

Intervening Contours Image edges signal a potential object boundary. It is particularly useful when background clutter has similar intensity value with object body. We evaluate the affinity between two pixels by measuring the magnitude of image edges between them:

$$W_C(i, j) = e^{-max_{x \in line(i,j)} ||Edge(x)||^2/\sigma_C} \quad (2)$$

where $line(i, j)$ is a straight line joining pixels i and j, and $Edge(x)$ is the edge strength at location x. Fig. 1 shows graph weights $W(i, :)$ for a fixed pixel i. At coarse image scales, texture edges tend to be blurred out and suppressed, while at fine image scales faint elongated edges are more likely to be detected, together with texture edges. To define the affinity between two pixels i and j we look at the edges across multiple scales.

We can combine the two cues with $W_{Mixed}(i, j) = \sqrt{W_I(i, j) \times W_C(i, j)} + \alpha W_C(i, j)$.

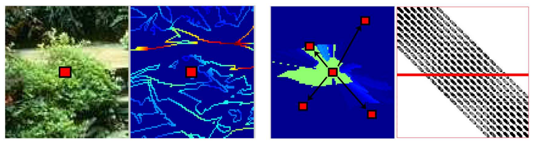

Figure 1: Column 1 and 2: image and image edges. Column 3 and 4: segmentation graph encoding intervening contour grouping cue. Two pixels have high affinity if the straight line connecting them does not cross an image edge. Column 3 displays one row $W_C(i, :)$ of the graph connection matrix reshaped as an image, for the central pixel i. The row corresponds to the red line on column 4.

2.2. Computing Optimal Normalized Cuts

For a bipartition of the graph $V = A \bigcup B$, the Normalized Cuts [10] cost is defined as: $Ncut(A, B) = \frac{Cut(A,B)}{Volume(A) \times Volume(B)}$. We can rewrite it using binary group indicator function $X_l \in \{0, 1\}^N$, with $X_l(i) = 1$ iff pixel i belongs to segment l. Let $X = [X_1, X_2]$, D be a diagonal matrix where $D(i, i) = \sum_j W(i, j)$. The segmentation criterion amounts to the following:

$$\text{maximize } \epsilon(X) = \frac{1}{2} \sum_{l=1}^{2} \frac{X_l^T W X_l}{X_l^T D X_l} \quad (3)$$

subject to $X \in \{0, 1\}^{N \times 2}$ and $X 1_2 = 1_N$ (1_N is a vector of N ones). A generalized K-way Ncut cost function can be similarly defined using $X = [X_1, \ldots, X_K]$. Finding the optimal Ncut graph partitioning is NP hard. A spectral graph partitioning technique allows us to solve this problem using a continuous space solution by computing the K eigenvectors corresponding to the K largest eigenvalues in:

$$WV = \lambda DV \quad (4)$$

To discretize V into X, we first normalize the rows of V into V', and then search for the rotation R that brings V' the closest possible to a binary indicator vector X.

3. How large of a graph connection radius?

The construction of our image segmentation graph thus far has focused on encoding grouping cues to compute the graph weights $W(i, j)$. We turn our attention to the graph topology. Recall that two pixels are connected in a graph if they are within distance G_r. How big should the graph connection radius G_r be ?

A larger graph radius G_r generally makes segmentation *better*. Long range graph connections facilitate propagation of local grouping cues across larger image regions. This effect allows us to better detect objects with faint contours in a cluttered background, as shown in fig. 2.

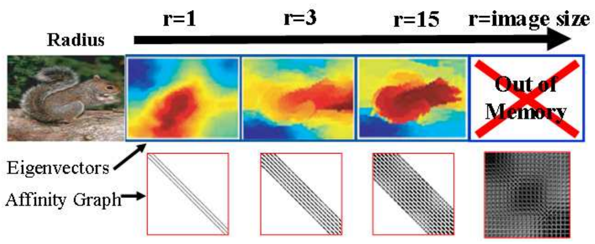

Figure 2: The Ncut segmentation eigenvector of the left image for increasingly large graph connection radius G_r. With larger G_r, the squirrel with faint contours pops out more clearly, but the graph affinity matrix becomes denser. The bottom row shows zoomed out versions of the affinity matrices.

Smaller G_r generally makes segmentation *faster*. The graph weight matrix grows rapidly with rate of $O(G_r^2)$. In computing the Ncut eigenvectors, the overall running time is dominated by two factors: 1) the cost of matrix-vector multiplication $y := Wx$ (which can be thought of as a local anisotropic diffusion on partitioning function x), and 2) the number of iterations, or number of matrix-vector multiplications until convergence. For faster computation, we want to minimize the number of iterations and make each iteration faster. However, as the following experiment (fig. 3) shows, setting G_r small does *not* necessarily make the overall Ncut optimization faster. As we see in fig. 3, there is a tradeoff between the number of eigensolver iterations, and the size of G_r. It appears there is a minimum required connection radius. A graph with a too small connection radius requires a lot of diffusion operations $y := Wx$ to propagate local grouping cues across larger neighborhood.

4. Compression of long range connection graphs

The ideal graph connection radius G_r is a tradeoff between the computation cost, and segmentation result. We will show that we can alleviate this tradeoff by providing an efficient segmentation algorithm which can effectively have a very large G_r. We do so by decomposing the long range connection graph into independent subgraphs.

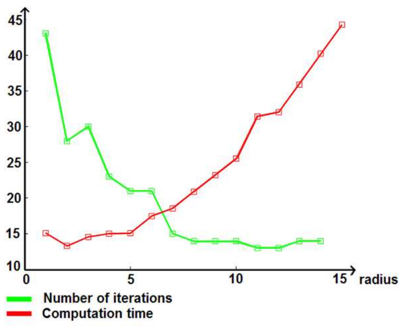

Figure 3: We compute Ncut segmentation eigenvectors for graphs with increasing connection radius G_r for the squirrel image in Fig. 2. The number of eigensolver iterations and total running time (sec) as a function of graph radius G_r. The number of eigensolver iterations is high for small G_r, and decreases steadily until $G_r = 7$. The total running time remains constant until $G_r = 5$ despite rapid increase in the cost of $y := Wx$.

4.1. Statistics of Segmentation Graph Weights

Consider the following experiment. We extract 60×60 patches from 200 randomly selected images. For each image patch P_k, we use the intervening contour cue to compute $W^{P_k}(i,j) = W_C^{P_k}(i,j)$ for the central pixel i and all possible j. We estimate the following statistical measures:

1) Average of graph weights across images: $Ave[W(i,j)] = \frac{1}{N}\sum_{k=1}^{N} W^{P_k}(i,j)$, shown in Fig. 4(a). As expected, the average affinity between two pixels i and j at distance $r_{ij} = ||X_i - X_j||$ decreases exponentially fast as a function of r_{ij}. This can be explained. If p_{edge} is the probability that an edge does not fall between two adjacent pixels, the probably that i and j are *not* separated by an image edge is theoretically $p_{edge}^{r_{ij}}$.

2) Variance of graph weights across images: $Var[W(i,j)] = \frac{1}{N}\sum_{k=1}^{N} |W^{P_k}(i,j) - Ave[W(i,j)]|^2$, shown in Fig. 4(b). Overall, as a function of r_{ij}, the graph variance approximates a Laplacian. For short range connections ($r_{ij} \leq 3$), the pair-wise affinity has low variance across image patches. As pair-wise pixel separation r_{ij} increases, the variance in graph affinity increases quickly until $r_{ij} = 13$. For long range connections, the variance drops back to zero. This implies that for very short and very long range connections, the $W(i,j)$ are more predictable between the images, than those of mid-range connections. Therefore, the mid-range connections contain most information of the image structure.

3) Variance of graph weights across small neighborhood. For a pair of pixels i, j with r pixels apart, we take two balls of radius R, B_i and B_j around i and j. We measure the variance of graph weights $W^{P_k}(i',j')$ for all $(i',j') \in B_i \times B_j$, denoted as $VarW^{P_k}(B_i,B_j)$, and average it across all image patches and all i,j with r pixels apart: $VarW(r) = \frac{1}{N}\sum_{k=1}^{N} Ave_{i,j:r_{ij}=r} VarW^{P_k}(B_i,B_j)$, as

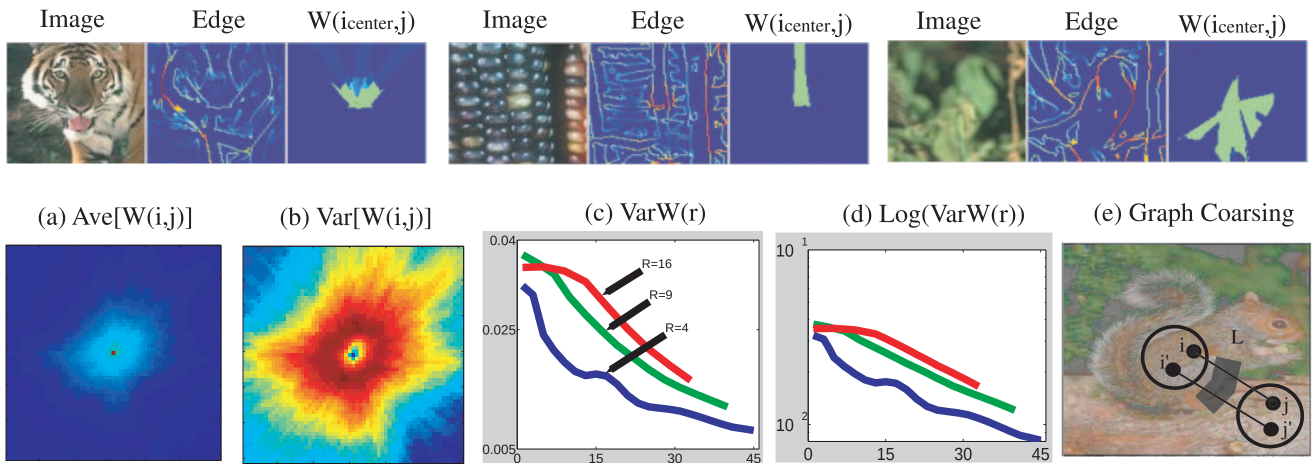

Figure 4: Statistics of graph weights on natural images. Top row: we use intervening contour cue to compute graph weights for randomly selected image patches across 200 images. For a fixed pixel i, we estimate average graph weight $W(i,j)$ in (a), variance of $W(i,j)$ across images in (b), and variance of each graph edge $W(i,j)$ across a small neighborhood R of the edge as a function of spatial separation $r = r_{ij}$, in (c), (d). Distant pixels are more likely to be disconnected by an image contour far from both pixels. Together, these facts allow us to create a multiscale decomposition of large radius graph connections by "condensing" close-by edge weights (i,j) and (i',j'), (e).

shown in Fig. 4(c,d). $VarW(r)$ decreases *exponentially* fast as a function of spatial separation r! For short range graph edges, it is hard to predict neighboring affinities around graph edge (i,j) from a particular affinity $W(i,j)$. As spatial separation increases, the affinity variations decrease quickly, indicating one can potentially predict graph edge weights in its neighborhood using one representative edge connection.

In summary, statistical measurements of the *intervening contour* image segmentation graph reveal three facts: 1) graph edge weights decrease exponentially fast with pairwise pixel separation; 2) across the images, the mid-range graph edges are most un-predictable, therefore contain most relevant grouping information; 3) the variations in graph weights across a small neighborhood of graph edge (i,j) decrease exponentially fast with the pixel separation r_{ij}, implying that longer range graph connections have more redundant information with their nearby connections, therefore can be compressed.

4.2. Decomposition of graph into multiple scales

Our empirical estimation of *Graph mean* and *Graph variance* indicates that at different ranges of spatial separations r_{ij}, the graph affinity $W(i,j)$ exhibits very different characteristics. Therefore, we can separate the graph links into different *scales* according to their underlying spatial separation:

$$W = W_1 + W_2 + ... + W_S, \tag{5}$$

where W_s contains affinity between pixels with certain spatial separation range: $W_s(i,j) \neq 0$ only if $G_{r,s-1} < r_{ij} \leq G_{r,s}$. This decomposition allows us to study behaviors of graph affinities at different spatial separations.

Furthermore, affinity variation $VarW(r)$ decreases quickly, implying that, at a given scale, one can potentially "condense" pixels in a small neighborhood into representative pixels, and store only the affinity between those pixels. More importantly, the exponential decay of $VarW(r)$ implies we can condense pixels very aggressively, at exponential rate, as we increase the graph connection radius. Therefore even though the non-zero elements in W_s grow quadratically with s, we can represent W_s more efficiently with fewer representative connections.

How do we determine representative pixels at graph scale W_s? For the first graph scale W_1, we take every pixel as graph node, and connect pixels within r distance apart by a graph edge. For the second graph scale W_2, there are no short graph connections, we can sample pixels at distance $2r + 1$ apart in the original image grid as representative nodes. Applying this procedure recursively, at scale s, we sample representative pixels at $(2r+1)^{s-1}$ distance apart on the original image grid, as shown in Fig. 5. We will denote the representative pixels in each scale by I_s, and denote W_s^c as a compressed affinity matrix with connections between the representative pixels in I_s. The different scales of the graph are defined on different layers of the image pyramid, each a sub-sample of the original image.

Note we create W_s^c by simply sub-sampling the original graph W_s which encodes combined intervening-contour/brightness cues. There are several alternatives: one can average the graph connections within the sampled image region, or use region based grouping cues to define the graph weights between the representative pixels. We choose not to use these alternatives, and focus on purely the effect of multiscale graph decomposition itself.

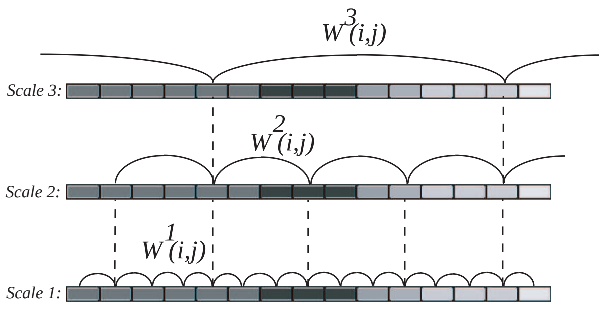

Figure 5: 1D view of multiple-scale graph decomposition with $r = 1$. Large radius graphs can be decomposed into different scales, each containing connections with specific range of spatial separation: $W = W_1 + W_2 + ... + W_S$. At larger scales, the graph weights vary slowly in a neighborhood, we can sample them using representative pixels at $(2 \cdot r + 1)^{s-1}$ distance apart.

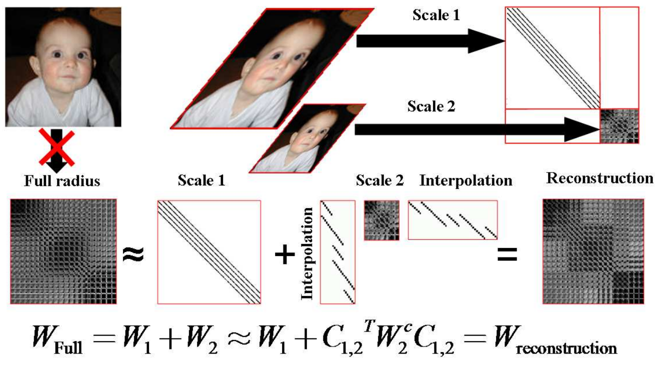
Scale 1
Scale 2
Full radius
Scale 1
Scale 2
Interpolation
Reconstruction
Interpolation
$W_{\text{Full}} = W_1 + W_2 \approx W_1 + C_{1,2}{}^T W_2^c C_{1,2} = W_{\text{reconstruction}}$

Figure 6: Multiscale graph compression. With a maximal graph connection radius G_r, the affinity matrix W_{Full} probably doesn't fit in memory. We can decompose it into short-range and long-range connections: $W_{Full} = W_1 + W_2$, and compress W_2 with a low-rank approximation: $W_2 \approx C_{1,2}^T W_2^c C_{1,2}$. W_2^c can be computed either directly on a sub-sampled image, or by sampling values from W_1. The interpolation matrix $C_{1,2}$ from scale 2 to scale 1 will be introduced later on to couple segmentations at each scale.

Computational saving. Using the above mentioned multiscale graph decomposition and compression, at compressed graph scale s we have $N/\rho^{2(s-1)}$ nodes, where N is the number of pixels, and ρ is the sampling factor, in our case $\rho = 2r + 1$. Summing across all the scales, we have a total of $N/(1 - \frac{1}{\rho^2})$ nodes. Since at each scale nodes are connected with only $(2r + 1)^2$ nearest neighbors, we can compress a fully connected graph with $N(2r + 1)^2/(1 - \frac{1}{\rho^2})$ graph weights. Take a typical value of $\rho = 3$, $r = 1$, the total number of multiscale graph connections is about $10N$ which is a very small fraction of the original N^2 connections. As Fig. 6 illustrates, such a small number of connections can have virtually the same effect as a large fully connected graph.

In summary, we have proposed a decomposition of a segmentation graph W into disjoint scales: $(W_s)_{s=1..S}$, where each W_s can be compressed using a recursive sub-sampling of the image pixels. This compression of W is not perfect, compared to more accurate data-driven graph compression schemes such as algebraic multi-grid. However, as we will explain in the following section, we can still achieve precise and efficient graph partitioning using this simple multiscale graph decomposition.

5. Multiscale Graph Segmentation

Principle We process the multiscale graph in *parallel* so that information propagates from one scale to another. We achieve this by specifying *constraints* on the multiscale graph partitioning criterion. This is in contrast to most existing multiscale segmentation algorithms where the different scales are processed *sequentially*. The main difficulty is in specifying information flow across scales, which is the topic of the next section.

5.1. Parallel segmentation across scales

Let $X_s \in \{0,1\}^{N_s \times K}$ be the partitioning matrix at scale s, $X_s(i,k) = 1$ iff graph node $i \in I_s$ belongs to partition k. We form the multiscale partitioning matrix $\mathbf{X}$ and the bloc diagonal **multiscale affinity matrix W** as follows:

$$X = \begin{pmatrix} X_1 \\ \vdots \\ X_S \end{pmatrix}, \; W = \begin{pmatrix} W_1^c & & 0 \\ & \ddots & \\ 0 & & W_S^c \end{pmatrix} \qquad (6)$$

We seek a multiscale segmentation optimizing the Ncut criterion on W defined in Sec. 2.2.

Direct partitioning of graph W gives the trivial segmentation, grouping all the nodes in a given scale as one segment. For multiscale segmentation, we need segmentation costs to propagate across the different scales. At the finest graph scale, the segmentation should take into account graph links at all coarser levels. We need to seek *one* consistent segmentation across all scales. The cross-scale consistency we seek is simple: the coarse-scale segmentation (X_{s+1}) should be locally an average of the fine-scale segmentation (X_s). This is done by constraining the multiscale partitioning vector X to verify: for all node i in layer I_{s+1}, $X_{s+1}(i) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} X_s(j)$. The neighborhood $\mathcal{N}_i$

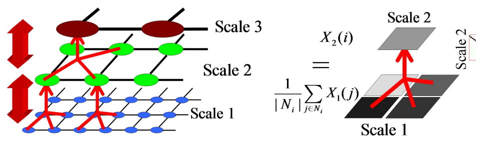

Figure 7: Left: 2D view of a three-layer graph with connection radius $r = 1$. The three scales communicate through cross-scale interpolation matrices $C_{1,2}$ and $C_{2,3}$. Middle: cross-scale constraint between scale 1 and scale 2 for partitioning vector X. $X_2(i)$ is the average of $X_1(j)$ for nodes j below i. Stacking those equations together, we get the cross-scale constraint $CX = 0$, here for two scales. We see the upper triangular structure of $C = [C_{1,2}, -I_2]$.

specifies the projection of $i \in I_{s+1}$ on the finer layer I_s, and is simply defined on a regularly spaced grid of size ρ, the sampling factor.

Define matrix $\mathbf{C_{s,s+1}}$ (of size $N_{s+1} \times N_s$) as the **cross-scale interpolation matrix** between nodes in layer I_s and those in coarser layer I_{s+1}, as shown in Fig. 7:

$$C_{s,s+1}(i,j) = \begin{cases} \frac{1}{|\mathcal{N}_i|} \text{ if } j \in \mathcal{N}_i, \\ 0 \text{ else} \end{cases} \tag{7}$$

We define the **cross-scale constraint matrix C**:

$$C = \begin{pmatrix} C_{1,2} & -I_2 & & 0 \\ & \ddots & \ddots & \\ 0 & & C_{S-1,S} & -I_S \end{pmatrix}, \tag{8}$$

and the cross-scale segmentation constraint equation:

$$\boxed{CX = 0} \tag{9}$$

As illustrated in Fig.7, the cross-scale constraint is a key concept in our multiscale segmentation algorithm. With this constraint, the segmentation cost is forced to propagate across the scales to reach a consistent segmentation at all scales.

Multiscale segmentation criterion The segmentation criterion we will use is the constrained multiscale Normalize Cut:

$$\text{maximize } \varepsilon(X) = \frac{1}{K}\sum_{l=1}^{K} \frac{X_l^T W X_l}{X_l^T D X_l} \tag{10}$$

$$\text{subject to } CX = 0, X \in \{0,1\}^{N^* \times K}, \; X1_K = 1_{N^*}, \tag{11}$$

where $N^* = \sum_s N_s$. The problem being set, we will now show how to handle it in an efficient way.

6. Computational solution and running time

We transform this NP-complete combinatorial problem into its counter part in a continuous space. After some algebra, the problem becomes:

$$\text{maximize } \varepsilon(Z) = \frac{1}{K}\text{tr}(Z^T W Z) \tag{12}$$

$$\text{subject to } CZ = 0, Z^T D Z = I_K, \tag{13}$$

This constrained optimization problem has been addressed in [12], we adapt the main result below. Let $P = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ be the normalized affinity matrix, and Q be the projector onto the feasible solution space:

$$Q = I - D^{-\frac{1}{2}} C^T (C D^{-1} C^T)^{-1} C D^{-\frac{1}{2}}. \tag{14}$$

Let $V = (V_1, ..., V_K)$ be the first K eigenvectors of matrix QPQ. Then the solutions to (12) are given by scaling any rotation of the K eigenvectors $V = (V_1, ..., V_K)$:

$$\arg\max_Z \varepsilon(Z) = \{D^{-\frac{1}{2}} V R : R \in O(K)\}. \tag{15}$$

The proof, given in [12], uses Lagrange multipliers to get rid of the constraint. The optimal solution is a subspace spanned by the K largest eigenvectors, but this time the matrix is $QD^{-\frac{1}{2}} W D^{-\frac{1}{2}} Q$ instead of $D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$. The final algorithm is summarized in the box below.

1. Given a $p \times q$ image I, for $s = 1..S$ (S=# scales):
   (a) sample $\frac{p}{\rho} \times \frac{q}{\rho}$ pixels $i \in I_s$ from I_{s-1} on a regular grid, where ρ is the sampling factor.
   (b) compute constraint $C_{s-1,s}(i,j) = \frac{1}{|\mathcal{N}_i|}$ $\forall j \in \mathcal{N}_i$ sampling neighborhood of i.
   (c) compute affinity W_s^c on I_s with small radius r, using image edges at scale s.
2. compute W, C from $(W_s^c, C_{s,s+1})_s$ as in (6),(8)
3. Compute Q using (14), compute $\overline{V}$, the first K eigenvectors of $QD^{-\frac{1}{2}} W D^{-\frac{1}{2}} Q$. Compute $V = D^{-\frac{1}{2}}\overline{V}$ and discretize.

6.1. Running time analysis

We show that the complexity of this algorithm is *linear* in the number of pixels. Fix the sampling factor ρ between the scales, and the connection radius r to compute W_s at each scale s. Suppose we use all possible scales, i.e. $S = \log_\rho(\max(p,q))$ for a $N = p \times q$ image. Denoting $nnz(A)$ the number of non-zero elements of a matrix A, we have $nnz(W) = \sum_s nnz(W_s) = \sum_s \frac{N}{\rho^{2(s-1)}}(2r+1)^2 = O(N)$.

We show the constrained multiscale Ncut can also be computed in $O(N)$ time. The complexity of the eigensolver is dominated by the running time of the matrix-vector multiplication $y := QPQx$, where Q defined in (14) could be full. Instead of computing Q explicitly, we expand out the terms in Q, and apply a chain of smaller matrix-vector operations. The only time consuming term is computation of $y := (CD^{-1}C^T)^{-1}x$, which has $O(N^3)$ running time. However, because we chose non-overlapping grid neighborhoods, we can order the graph nodes to make C (and hence $CD^{-\frac{1}{2}}$) upper triangular. We then compute $y := (CD^{-1}C^T)^{-1}x$ by solving 2 triangular systems with $nnz(C) = O(N)$ elements. Overall, the complexity of $y := QPQx$ is $O(N)$. We verified empirically this linear running time bound, and the results in Fig. 8 show a dramatic improvement over state of the art implementations.

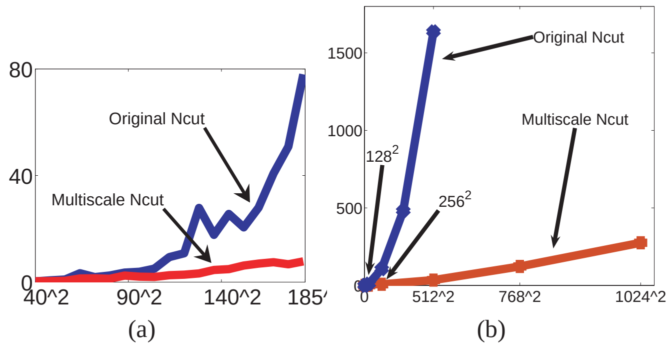

Figure 8: Running time in seconds of original Ncut vs. Multiscale Ncut as a function of image pixels N. In original Ncut, we scale connection radius with image size:$G_r = \frac{\sqrt{N}}{20}$, and running time is $\geq O(NG_r^2) = O(N^2)$. In Multiscale Ncut, we construct a multiscale graph with same effective connection radius. Its running time is $O(N)$.

6.2. Comparison with other multi-level graph cuts

It is important to contrast this method to two other successful multilevel graph partitioning algorithms: METIS [5] and Nystrom approximation [3]. In both cases, one adaptively coarsens the graph into a small set of nodes, and compute segmentation on the coarsened graph. The fine level segmentation is obtained by *interpolation*. Both algorithms require correct initial graph coarsening [3]. Nystrom works quite well for grouping cues such as color. However for intervening contour grouping cues, graph weights have abrupt variations making such precise graph coarsening infeasible.

7. Results

Sanity check. We verify Multiscale Ncut segmentation with a simple "tree" image shown in Fig. 9. We create two scales, with sampling rate = 3. The first level graph has radius =1, the second level has radius = 9. We test whether Multiscale Ncut is able to segment coarse and fine structures at the same time: the large trunk as well as the thin branches. For comparison, we computed Ncut eigenvectors of coarse and fine level graphs in isolation. As we see in Fig.9, multiscale segmentation performs correctly, combining benefits of both scales.

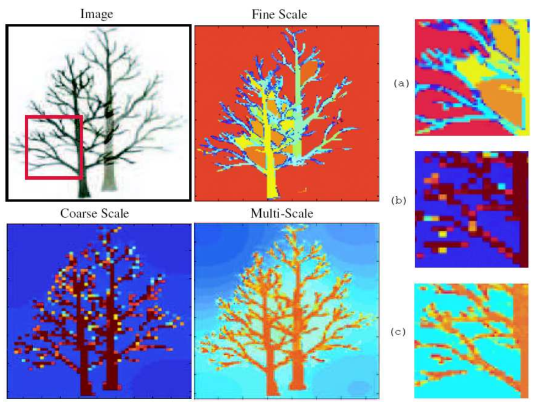

Figure 9: Top middle: fine level segmentation fails in cluttered region; Bottom left, coarse level segmentation alone fails to provide detailed boundary; Bottom middle multiscale segmentation provides correct global segmentation with detailed boundary. Right: zoom portion of the segmentation in fine level (a), coarse level (b), and multiscale (c).

Effect of sampling error in coarse graph construction. We purposely kept construction of multiscale graph extremely simple with geometric sampling. This sampling could have a bad effect on pixels near an object boundary. We study if Multiscale Ncut can overcome this sampling error. Fig. 10 shows the final segmentation can overcome errors in coarse grid quantization, with a small decrease in boundary sharpness (defined as eigenvector gap across the object boundary) in worst case.

Effect of image clutter and faint contours We argue multiscale segmentation can handle image clutter and detect objects with faint contours. Such a problem is particularly important for segmenting large images. Fig. 11 provides one such example with a 800×700 image. The segmentation is both accurate (in finding details), robust (in detecting faint but elongated object boundary), and fast.

We have experimented with the multiscale Ncut on a variety of natural images, shown in Fig. 12. We observed that compressed long range graph connections significantly improve running time and quality of segmentation. More quantitative measurement is currently underway.

References

[1] Adrian Barbu and Song-Chun Zhu. Graph partition by swendsen-wang cuts. In *Int. Conf. Computer Vision*, pages 320–327, Nice, France, 2003.

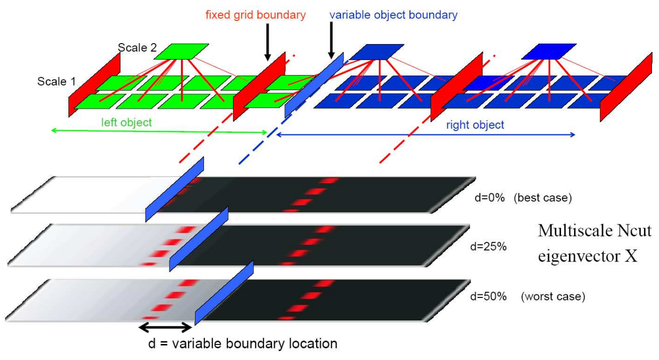

Figure 10: Non-adaptive grid can produce precise object boundaries and recover from errors in grid quantization. Top: a two level graph, with coarse nodes spaced on a regular grid (boundaries in red). An object boundary (in blue) specifies two regions (green vs. blue) with low mutual affinity. Its location d (as % of grid size) w.r.t. the grid varies from $d = 0\%$ (best case, grid and object boundaries agree) to $d = 50\%$ (worst case, object boundary cuts the grid in half). The central coarse node is linked either to left or right coarse nodes depending on d. Bottom: multiscale Ncut eigenvector for $d = 0\%, 25\%, 50\%$. In all cases, multiscale segmentation recovers from errors in coarse level grid. Notice that the gap in Ncut eigenvector across the object boundary remains high even in worst case.

[2] Peter J. Burt and Edward H. Adelson. The laplacian pyramid as a compact image code. COM-31(4):532–540, April 1983.

[3] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. 2003.

[4] J-M. Jolion and A. Rosenfeld. *A Pyramid Framework for Early Vision*. Kluwer Academic Publishers, Norwell, MA, 1994.

[5] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. 1995.

[6] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. pages 465–470, 1996.

[7] M. Luettgen, W. Karl, A. Willsky, and R. Tenney. Multiscale representations of markov random fields. pages 41:3377–3396, 1993.

[8] Patrick Perez and Fabrice Heitz. Restriction of a markov random field on a graph and multiresolution image analysis. Technical Report RR-2170.

[9] Eitan Sharon, Achi Brandt, and Ronen Basri. Fast multiscale image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–7, 2000.

[10] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[11] Stella X. Yu. Segmentation using multiscale cues. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 70–7, 2004.

[12] Stella X. Yu and Jianbo Shi. Grouping with bias. In *Advances in Neural Information Processing Systems*, 2001.

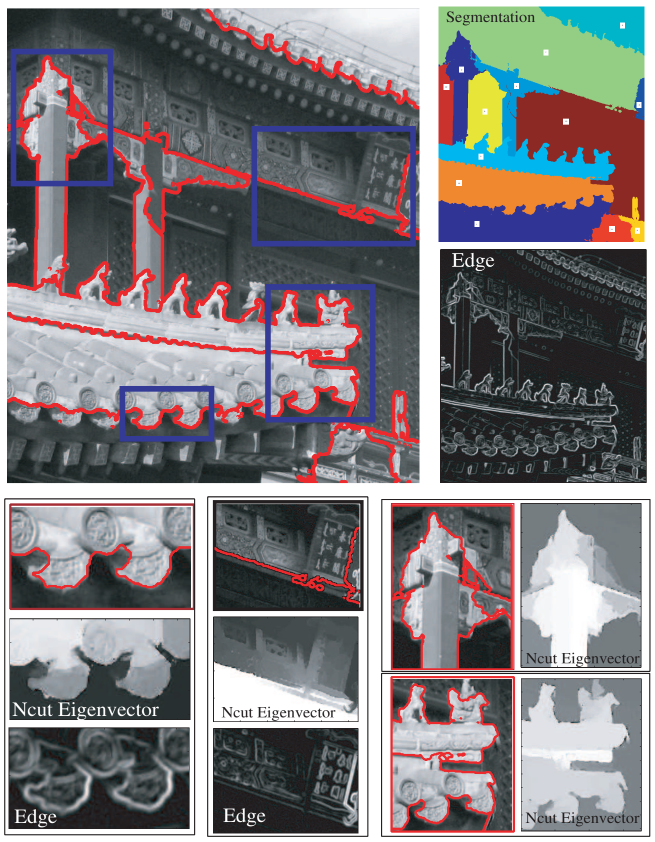

Figure 11: Multiscale Ncut segmentation of a 800×700 image. Top left, image with detected object boundary. Top right, segmentation and input edge map. Bottom: zoom in details. Note the faint roof boundary is segmented clearly.

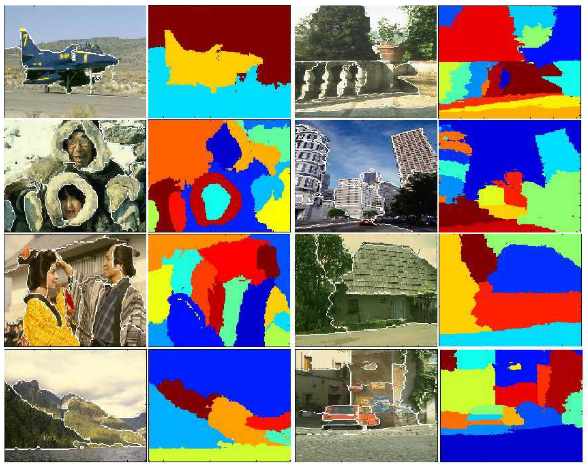

Figure 12: Multiscale Ncut prevents braking large uniform image regions into smaller parts due to its efficient use of long range graph connections.