Improved Cut-Based Foreground Identification

Sharat Chandran Satwik Hebbar Vishal Mamania Abhineet Sawa
Computer Science & Engineering Department
Indian Institute of Technology, Bombay
India 400076
http://www.cse.iitb.ac.in/~{sharat,satwik,vishalm,sawa}

Abstract

Automatic content based schemes, as opposed to those with human endeavor, have become important as users attempt to organize massive data presented in the form of multimedia data such as images, and home or movie videos. One important goal, be it in shot understanding, or scene detection, or compression, is the ability to find foreground pixels. This higher level task is best realized using a graph-based description of the input image or video.

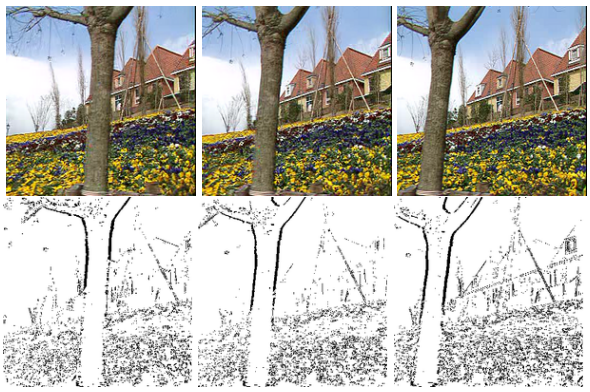
The normalized cut framework is appealing because it looks at an image or an image sequence from a global perspective. Unfortunately due to quadratic storage and time complexity, the algorithm appears to be infeasible to use on medium and large datasets. In this paper, we show how to make graph based schemes tractable and useful.

1 Introduction

An increasing amount of video is engulfing computer disks with information. An estimated half a terabyte or 9,000 hours of motion pictures are produced around the world every year. Furthermore, 3,000 television stations broadcasting for twenty-four hours a day produce eight million hours of video per year. Counting a few hours of amateur video efforts produced by camcorder owners, several gigabytes of video start residing on typical end user computers. We quickly come to the conclusion that tools must be provided to store, organize, and retrieve video data. One central cognitive task is therefore the question: "What is the foreground object in static or moving picture data?"

This task is difficult if only a single frame is given. When it comes to meaningful extraction of semantic information ("where is the actor in this frame?") in a video, a very common strategy is to build a model of the background, and "subtract" it from the frame under interest.

However, background segmentation techniques are problematic when camera motion is involved. The basic reason for this arises from the fact that they are fundamentally based on computing frame differences of some form. Consider the *flower garden* sequence



where the interesting part is the prominent tree in the foreground. The results of a simple thresholded frame difference are also shown below the sequence. It is evident that almost everything seems to be moving; dynamically building a model of the background is difficult. A similar problem manifests itself in the video sequence below where the

camera is continuously moved at a rapid pace to keep the driver of a car in the center of the frame. In other words, both the camera and the scene are moving.

In fact, a number of factors such as camera noise, shadows, reflections, and the like force an error prone trial-and-check post-processing using, say, morphological methods to eliminate false positives. Lost foreground, and thus a true measure of the shape, is irrecoverable in post-processing. Therefore while background subtraction is a very important tool, and *must* be used where applicable, there are cases where an alternative is required.

Graph based methods are attractive; graphs are highly malleable but simple data structures and the approach takes

a global approach on deciding how to classify foreground pixels.

One reason for the lack of universal popularity of graph algorithms has been the enormous computational cost, both in terms of space and time. Even after making some assumptions on the sparsity of certain matrices, the core routine in the N-cut algorithm takes $O(n^{1.5})$ time for a graph with n nodes (usually the $O(n)$ pixels). A segmentation scheme requires repeated use of this core super-linear algorithm, and therefore the time requirements can make certain approaches intractable. The space complexity is $O(n^2)$, which precludes working with videos more than a few minutes long.

1.1 Our Contributions

In this paper, we consider the problem of obtaining the important "foreground" in either individual images, or a set of frames in a video. In the second case, we suppose that either enough frames are not available to build a model of the background, or the scene is such that background subtraction methods might fail.

Our first naive attempt of using the N-cut algorithm to obtain two clusters within a graph was computationally infeasible. One key idea reported in this paper is to make the problem tractable by using a fast segmentation algorithm to "feed" the N-cut algorithm with a smaller sized, but meaningful data. This is not a simple "pre-processing" step such as bottom-up region merging to reduce input size; we provide a ready-made matrix of *relevant* data for the linear algebra basis of the N-cut algorithm. In particular, for video data with n pixels in a frame, and considering, say, 3 adjacent frames, a naive N-cut algorithm would require $9n^2$ matrix space; we use n matrix space and $O(n \log n)$ time. We demonstrate our algorithm with qualitative and quantitative experimental data.

The rest of this paper is organized as follows. After a discussion of previous work in Section 2, we give details of our approach in Section 3. Sample results are shown in Section 4 and we end with some concluding remarks in the last section.

2 Previous work

Several [5, 13] graph theoretic approaches to segmentation have been proposed in the recent past. A general clustering algorithm based on flows has been proposed in [23]. Techniques using local information as in [5, 2], and those using region-growing and clustering are typically fast. Those which use global information, usually apply some sort of graph cuts to identify (dis)similar regions. Nested cuts [21], average cuts [15], and normalized cuts [19], are various techniques aimed at partitioning an undirected graph using a global cost function. In [22] another metric called the cut ratio is used to perform efficient iterated region-based segmentation. The approach in [17] is to perform hierarchical aggregation using an irregular pyramid of an image.

As in the work of [19], we believe it is important that segmentation algorithms take into account non-local properties of the image, because local properties fail to capture perceptually important differences. There have been several attempts made at utilizing global information without incurring heavy computational overheads. In [8], the features of region and boundary based approaches are combined by transforming region energy to boundary energy. The global optimization problem is then solved as a minimum mean weight cycle problem on a directed graph. In [9], an SVD approximation method based on probabilistic sampling has been proposed in an attempt to make normalized cuts computationally tractable.

Clustering pixels in an image has also been used to considerably reduce the complexity of certain techniques. Authors in [11] use image segmentation to obtain "superpixels." These superpixels guide their recognition algorithm in the recovery of human body configurations. In [7], static segmentation using the watershed algorithm is applied to obtain small locally coherent segments and then apply the dominant motion model to separate the background. In [12], a method is presented which works with an oversegmented video frame and performs merging based on motion similarity to separate moving objects. In[24] intra-frame segmentation is performed to obtain contours of objects and merge object regions using motion information to get a binary object image.

In [20], edges are tracked in image sequences to guide the segmentation of frames based on colour and the labeling of the regions based on their motion model. A simple framework with two frames, to give two motions (foreground and background) is presented and extended for more motion layers. A comprehensive survey of various spatio-temporal grouping techniques can be found in [10]. *In the above techniques, the main goal is to produce several segments; we are primarily interested in only two segments.*

Image sequences (as opposed to static images) provide a different challenge. Because of 'prior' information, considerable work has been focused on separating the background from the foreground in image sequences. In [4], a non-parametric background model has been developed which adapts to high frequency variations in the background. A Gaussian mixture model has been adopted in [6] for classifying a pixel into a background, shadow or moving object pixel. In [1], each pixel is represented as a group of clusters which are then ordered according to the likelihood of them modeling the background, based on the history information of pixel values. A background registration technique is used in [3] to construct a reliable background image from the accumulated frame difference information. Authors in [16] adapt the permissible range of background image variations

using the co-occurrence of image variations between pairs of neighboring image blocks. Authors in [14] have stressed that different background models need to be applied for different scenes; they show how such models can be computed using spatio-temporal image processing.

However, most of these methods depend on having information from a dense set of past frames about the approximate background model itself. Alternately, the methods learn or have prior knowledge of the nature of the background in order to produce results. Our method works without a background model and when only a sparse set of frames is given. If only a single key frame (say in a shot) is given, we can still get satisfactory results.

3 Our Approach

Clustering is one of the best known problems in computer vision, and has been vigorously addressed for the last 30 years. Graph theory being a well studied branch of computer science has been exploited for various segmentation purposes. Specifically, vertices in the graph contain features such as color, texture, and motion profiles. Edges might correspond to how these vertices are associated with each other; a strong link might suggest that these vertices are similar. Graph-based methods may be classified into two approaches.

In the *top-down* approach, a minimal cut is made to partition the graph into two. Significantly, further segmentation is achieved recursively. In the *bottom-up* approach clusters are created, possibly in parallel, at several places and each cluster is represented by the weight of the minimum spanning tree. The cut between two clusters is considered, but this time to decide whether clusters should be accumulated or not. The top-down approach is of particular interest in the case of motion segmentation in videos where we need the most prominently moving "foreground" separated out from the background. Authors in [19] propose the notion of *normalized* cuts. This technique produces good results in segmentation but is expensive. As an example, consider Figure 1. Unfilled circles represent the foreground and occupy a small but significant portion of the image. Completely filled circles represent the background. A large portion of the image is occupied by noisy data and is shown using shaded circles. The normalized cut method succeeds in creating an intuitive segmentation. A bottom-up approach correctly produces three segments but the higher level process is confused as to which of the $\binom{3}{2}$ areas should be combined to be declared as a possible foreground. (Note that the method does not solve the figure-ground problem.)

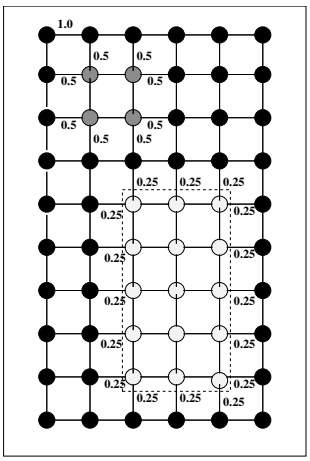


Figure 1. The normalized cut method is expensive, but can recover the intuitive foreground suggested by unfilled circles.

3.1 Our Idea

The central idea is to take advantage of a bottom-up algorithm we have developed earlier (Algorithm P) [2] which produces a *good* segmentation. Specifically, the speed of Algorithm P and the top-down global nature of Algorithm N-cut which produces the foreground data are combined. If we somehow feed an input of size $\sqrt{n}$ to Algorithm N-cut, it will result in manageable space complexity of $O(n)$. It will also result in one call running in time $n^{0.75}$. As a result, the net algorithm runs in $O(n \log n)$ time.

The delicate aspect of this pipelining procedure is to feed the *proper* "super pixels" to Algorithm N-cut. For example, the naive bottom-up strategy of grouping pixels in a quadtree fashion can indeed produce $O(\sqrt{n})$ nodes; however it might, in all probability and irrecoverably, feed improper pixels to Algorithm N-cut. On the other hand, the data driven output segmentation in Algorithm P may result in unpredictable number of segments. It may be $O(\sqrt{n})$ which is acceptable, or it might be $o(\sqrt{n})$ which is also acceptable. If however, the output size is $\Omega(\sqrt{n})$, we are in trouble!

Further, we also need to input (to Algorithm N-cut) an edge weight indicating the right measure of similarity between clusters. These two factors might spoil the running time of the overall algorithm, and therefore require further exploration of Algorithm P. The details of this have been omitted here due to space limitations; however, the pseudocode for the algorithm is given in the appendix.

3.2 Video Processing

The efficiency of our approach becomes more apparent when we depart from images and work with videos. The first approach would be to build a 3-dimensional graph just as we built the 2-dimensional graph for single images. Edges are thus between pixels as before within a frame (termed *spatial edges*) as well as between pixels in neighboring frames (termed *temporal edges*). The weights assigned to both temporal and spatial edges use motion profiles as discussed below.

The normalized cut on this graph results in two volumes representing what is globally the region of interest based on both intensity and motion profiles. An example of the

efficacy of this approach is reflected in Figure 4(a). The poor light in any one image, and the uneven lighting in the background in this sequence makes it impossible to detect the figure; motion profile with a cut based approach saves us. The projections on individual frames make it possible to obtain the foreground in image sequences. However, if there are n pixels in any frame, the storage for the input matrix for a sequence of 3 frames is $9n^2$ which makes the approach intractable.

In our approach we first cluster pixels together to obtain "superpixels." Edge weights now denote a similarity measure in terms of both spatial (intensity) and temporal (motion profile) features. Having used motion information to obtain "coherently moving" clusters, we avoid building a 3-dimensional graph and work with individual frames on a frame-by-frame basis. As a result, we use only $O(n)$ space for the input matrix to the N-cut subroutine.

3.2.1 Motion Profiles

Generally speaking, the motion profile MP_v of a pixel v represents the probability that a pixel is moving. It encodes both the direction and the amount of motion. A local calculation, it is represented as a matrix $\text{MP}_v[i,j], 0 \leq i,j < N$. These motion profiles are similar to those introduced in [18]. They differ, for example, in that our clustering algorithm needs a dissimilarity metric for the weights on the edge connecting two pixels in an image.

The motion profile is calculated using patch differences. A patch corresponding to a pixel v is a square neighborhood centered at the pixel and is denoted by $P_v[k,l], 0 \leq i,j < M$, where M is the patch-size. Now, the motion profile of pixel v^f, where f denotes the frame to which the pixel belongs, is given by :

$$\text{MP}_{v^f}[i,j] = \sum_{k,l} \left| P_{v^f}[k,l] - P_{v_1^{f+1}}[k,l] \right|$$

where v_1 is the pixel obtained after a displacement of $(i - \frac{N-1}{2}, j - \frac{N-1}{2})$ pixels on v. The net weight of an edge is a weighted combination of the intensity and motion profile differences.

4 Sample Results

Two types of results are shown in this section. We first show evidence that the qualitative performance of our algorithm is acceptable. Later we show that the time taken by our algorithm is substantially less. Our timing results are on a Pentium IV 1.8 GHz Linux based computer with 512MB RAM.

4.1 Qualitative Results on Images

First, we demonstrate that the results of our pruning procedure are comparable to the results obtained by other means, most notably, the original N-cut algorithm. Consider the input query image (Figure 2(a)) used in several [13, 19, 5] papers. Figure 2(b) shows the result of Algorithm-P. It runs fast, but produces too many clusters. Algorithm N-cut (Figure 2(d) reproduced from the original paper) does a reasonable job of identifying one of the players in the foreground. We were unable to produce this segmentation in a reasonable amount of time (our computer implementation ran out of memory also). Figure 2(e) shows our result, and it is evident that, as part of the foreground, we have managed to produce both the players *including the forearms missing in Figure 2(d)*. Note that a second pass on this segmented foreground can extract the two players separately. Figure 2(f) (reproduced from [17]) overlays on the input image the cut an irregular pyramid scheme would produce. Figure 2(c) shows the result obtained by a trial-and-check procedure using a popular tool applying fuzzy c-means clustering for obtaining two components and then using a region-growing process to get the two segments.

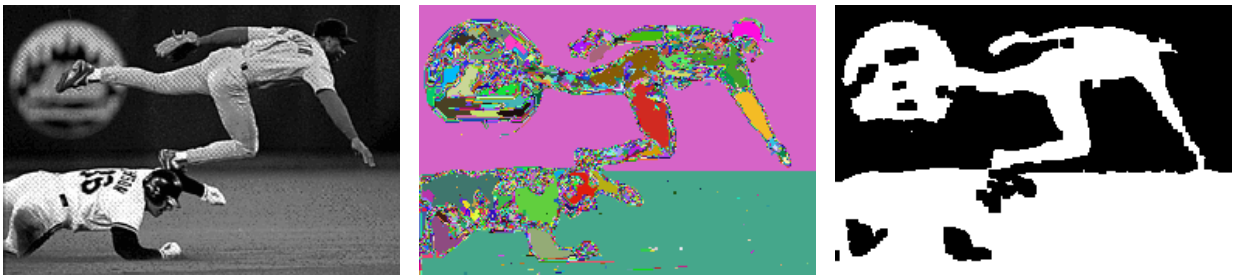
Figure 3 shows a set of images and the corresponding foreground region determined by the algorithm. The results demonstrate that the technique works well in many cases. For instance, there are a number of humans in the foreground in Figure 3(d). Figure 3(g) is a good example to illustrate the fact that the background need not always be uniform for our algorithm to perform handsomely. An immensely cluttered background is presented to the algorithm in Figure 3(h) and the results are still satisfactory. The range of images presented shows that the algorithm works well for both indoor and outdoor scenes. Figure 3(i) shows a result where we see that the algorithm works well for synthetic gradient images as well.

4.2 Qualitative Results on Videos

Figure 4 shows the results of our work for videos. We have again attempted to demonstrate that our method works for most of the typical scenarios.

Consider Figure 4(c) which shows a car moving across at a high velocity. Standard background segmentation techniques try to "learn" the background in the initial few frames; such techniques are defeated if the motion is high speed. Our approach uses motion information, along with the pixel intensities to cluster pixels together. This implies that, independent of whether there is camera motion, we will be able to group coherently moving pixels together. A higher level normalized cut on these clusters will then bring out the region of interest in the video adequately. This can be seen in our results where the car is identified as the foreground.

As discussed earlier, standard background segmentation techniques also are problematic when they are given videos with camera motion involved. In the *flower garden* sequence shown in Figure 4(e), the interesting part is the prominent tree in the foreground. The standard background



(a) Input image (b) Result from Algorithm-P (c) Semi-automatic segmentation results

(d) Result from [19] (e) Our result (f) Result from [17] at scale=11 (notice the black cut between the two players).

Figure 2. Regions created using various algorithms. Figures are best seen in color.

subtraction schemes will be distracted as there would be no way to differentiate between the motion of the tree and that of the house or the flowers. In our case, the motion profile scheme succeeds in eliminating certain parts of the frame (the sky, for example) as they do not appear to be moving at all. More important, the global nature of our scheme succeeds in differentiating the "faster motion" of the tree as compared to the flowers and the house.

Another interesting sequence is the *model* sequence (Figure 4(f)) where the camera is fixed on a car moving at a rapid pace. Both the camera and the scene are moving. Here too our method succeeds in getting the intuitive segmentation of the foreground, which is made up of the model and her car. Building a model of the background appears difficult.

Figure 4(b) shows our result for the *Hall Monitor* sequence which has high but uneven indoor lighting. In Figure 4(a) a man, occupying a very small region of the frame, is walking in front of a cluttered background with dim lighting. Our procedure identifies the man fairly clearly as the foreground in both cases. In Figure 4(d) there are two prominent motions in the foreground. We eliminate the background and retain both moving objects. All these cases show that even when only a few frames are available, foreground can be identified.

4.3 Running Time

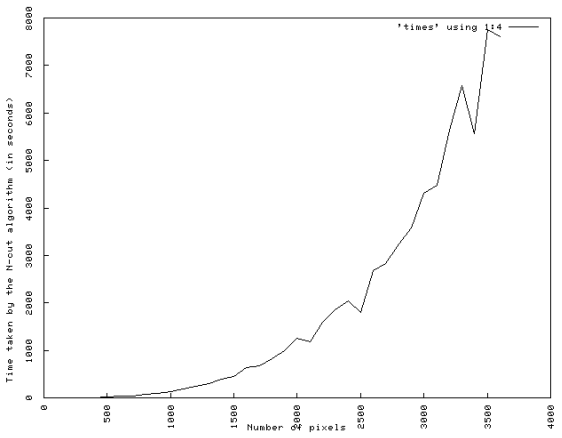


Figure 5. The time taken by our implementation of the original N-cut algorithm.

The experimental comparison between our algorithm and the normalized cut approach was performed on Figure 3(g) at different resolutions. Figure 5 shows the running time of our implementation of the N-cut algorithm. The x and y axes correspond to the number of pixels and the time

taken in seconds respectively. It can be observed that the running time increases dramatically as the size of the image increases.

Figure 6 shows the ratio of the running time of the original N-cut algorithm to our proposed algorithm. The x and y axes correspond to the number of pixels and the ratio of the time taken in seconds respectively. It can be observed that our algorithm is vastly superior when it comes to running time. (For large images or videos, after some point, the original version ran out of memory and would not continue; we could sustain ours.) The non-monotonic nature of the graph can be explained as follows. The time taken by our procedure depends directly on the number of clusters fed to it. As we work with small images at different resolutions, this number does not vary drastically from one resolution to the other and may even decrease with an increase in resolution. However, this is not true for the original procedure which is fed more pixels with every increase in resolution.

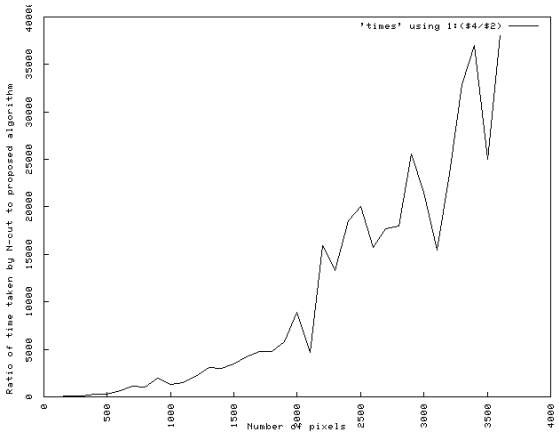


Figure 6. The ratio of the time taken by the original algorithm in [19] to ours. Any value more than 1 – all the cases – shows that our algorithm is faster.

5 Final Remarks

Identifying portions of an input image or video that qualify for the foreground (as opposed to whether it is the figure or the ground) is an important problem with various approaches. Even in video sequences (where there is the possibility of using past frames) identifying the foreground using background subtraction is often infeasible. A general framework is to represent the data as a graph and to partition the vertices into two components by finding a minimal cut. Normalized cuts (N-cut) ensure that there is cohesion within each component.

Our experiments indicated that the N-cut method produces classification of good quality, but can take an inordinate amount of time or space. In this paper, we have devised a region growing algorithm that can be integrated with the N-cut algorithm. Our entire algorithm runs in $O(n \log n)$ time. This attempt to reduce the computational complexity is also seen in [17] where the goal is to produce a complete segmentation of *images*. Our quantitative and qualitative results on videos indicate that the proposed method is successful. Once the foreground has been obtained, any suitable algorithm (such as the one in this paper) can be recursively applied to obtain finer details where necessary.

Acknowledgments

We thank Nithya and other members of ViGIL for their comments and suggestions.

References

[1] D. Butler, S. Sridharan, and J. V. Michael Bove. Real-time adaptive background segmentation. In *ICASP*, pages 341–344, 2003.

[2] S. Chandran and K. K. Madheshia. A fast segmentation algorithm revisited. In *Indian Conference on Computer Vision, Graphics, and Image Processing*, pages 496–501, 2002.

[3] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen. Efficient moving object segmentation algorithm using background registration technique. *IEEE Transactions on CSVT*, pages 577–586, 2002.

[4] A. Elgammal, D. Harwood, and L. Davis. Non-parametric model for background subtraction. In *ECCV*, pages 751–767, 2000.

[5] P. Felzenszwalb and D. Huttenlocher. Image segmentation using local variation. In *CVPR*, pages 98–104, 1998.

[6] N. Friedman and S. Russell. Image segmentation in video sequences: A probabilistic approach. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 175–181, 1997.

[7] Y. Huang, D. Paulus, and H. Niemann. Background-foreground segmentation based on dominant motion estimation and static segmentation. *Journal of Computing and Information Technology*, pages 349–353, 2000.

[8] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries. In *ICCV*, pages 904–910, 1999.

[9] J. Keuchel and C. Schnorr. Efficient graph cuts for unsupervised image segmentation using probabilistic sampling and SVD-based approximation. In *3rd International Workshop on Statistical and Computational Theories of Vision, Nice (France)*, 2003.

[10] R. Megret and D. DeMenthon. A survey of spatio-temporal grouping techniques. Technical report, University of Maryland, 2002.

[11] G. Mori, X. Ren, A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. *CVPR*, 2004.

[12] H. Nguyen, M. Worring, and A. Dev. Detection of moving objects in video using a robust motion similarity measure. *IEEE Transactions on Image Processing*, pages 137–141, 2000.

[13] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *CVPR*, pages 145–152, 2003.

[14] R. Pless, J. Larson, S. Siebers, and B. Westover. Evaluation of local models of dynamic backgrounds. In *CVPR*, pages 73–78, 2003.

[15] S. Sarkar and P. Soundararajan. Supervised learning of large perceptual organization: Graph spectral partitioning and learning automata. *IEEE Transactions on PAMI*, 22:504–525, 2000.

[16] M. Seki, T. Wada, H. Fujiwara, and K.Sumi. Background subtraction based on co-occurrence of image variations. In *CVPR*, pages II–65–II–72, 2003.
[17] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *CVPR*, pages 70–77, 2000.
[18] J. Shi and J. Malik. Motion segmentation using normalized cuts. *ICCV*, pages 1154–1160, 1998.
[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on PAMI*, 22(8):888–905, 2000.
[20] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *IEEE Transactions on PAMI*, pages 479–494, 2004.
[21] O. Veksler. Image segmentation by nested cuts. In *CVPR*, pages 339–344, 2000.
[22] S. Wang and J. M. Siskind. Image segmentation with ratio cut. *IEEE Transactions on PAMI*, pages 675–690, 2003.
[23] Z. Wu and R. M. Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Transactions on PAMI*, pages 1101–1113, 1993.
[24] J. Y. Zhou, E. P. Ong, and C. C. Ko. Video object segmentation and tracking for content-based video coding. *IEEE International Conference on Multimedia and Expo (III)*, pages 1555–1558, 2000.

procedure overall()
initQ$_2$()
initQ$_1$()
i $\leftarrow$ 0
h $\leftarrow$ new(handle)
while Q$_2 \neq \phi$ **do**
  s $\leftarrow$ findMin(Q$_2$)
  Q$_1$.dec(s, 0)
  grow(h, i)
  i $\leftarrow$ i+1
end while

Procedures initQ$_1$ and initQ$_2$ initialize the queues to be used in the algorithm.

procedure initQ$_2$
for all v $\in$ V **do**
  x $\leftarrow$ minAdjacent(v)
  Q$_2$.insert(v,x)
end for

procedure initQ$_1$
for all v $\in$ V **do**
  v.key $\leftarrow \infty$
  Q$_1$.insert(v, v.key)
end for

Procedure grow() creates the clusters by merging the neighboring nodes. It also creates the link between clusters which could not be merged due to the internal variation criteria.

procedure grow(h, i)
done $\leftarrow$ false
while not done **do**
  u $\leftarrow$ findMin(Q$_1$)
  if causeMerge(h, u) **then**
    if isLinkingNode(u) **then**
      h' = handle pointing to u
      Link handles h, h'
      linkWeight $\leftarrow \frac{\text{Int(h')}}{\text{edgeWeight}}$
    end if
    h.add(u)
    updateAdjacent(u)
    Q$_1$.remove(u)
    Q$_2$.remove(u)
  else
    u.setLinkingNode (true)
    u.setHandlePointingIt (h)
    done $\leftarrow$ true
  end if
end while

Procedure causeMerge(h,u) determines whether to merge the node u to cluster h or not and updateAdjacent(u) updates the neighbor's weights while growing the cluster.

procedure causeMerge(h, u)
if u.key $<$ (h.internalVariation + τ) **then**
  return true
else
  return false
end if
procedure updateAdjacent(u)
for all v $\in$ adjacent(u) **do**
  if (v $\in$ Q$_1$) $\wedge$ (v.key$<$w(u,v)) **then**
    v.key $\leftarrow$ w(u,v)
    Q$_1$.decreaseKey(v,v.key)
  end if
end for

procedure mergeClusters()
sort(edgeList) {sorted list of edges e_i's looks like $(C_x e_0 C_{x'})(C_y e_1 C_{y'}) \ldots$, where $e_i \geq e_{i+1}$}
i $\leftarrow$ 1
count $\leftarrow$ number of clusters
while (count $\geq$ n) $\wedge$ (i $\leq$ no of edges) **do**
  merge($C_x,\ C_{x'}$) {e_i joins $C_x,\ C_{x'}$}
  i $\leftarrow$ i+1
  count $\leftarrow$ count-1
end while

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figure 3. Sample results for input images, best viewed in color. See text for details.

(a) (b) (c)

(d) (e) (f)

Figure 4. Sample results for video data. Only representative frames from the video are shown.