LIVE VIDEO OBJECT TRACKING AND SEGMENTATION USING GRAPH CUTS

Zachary Garrett Hideo Saito

Department of Information and Computer Science, Keio University,
3-4-1 Hiyoshi, Kohoku-ku, Yokohama, 223-8522 JAPAN
E-mail: {zgarrett,saito}@ozawa.ics.keio.ac.jp

ABSTRACT

Graph cuts have proven to be powerful tools in image segmentation. Previous graph cut research has proposed methods for cutting across large graphs constructed from multiple layered video frames, resulting in an object being tracked across multiple frames. However, this research focuses on cutting graphs constructed from a prerecorded video sequence. In live video scenarios, frames cannot be layered to construct 3D volumes, since the contents of the subsequent frames are unknown. Instead, new graphs must be created and cut for each frame on demand. Resource limitations make this unfeasible on high-resolution videos. In addition, object tracking requires a method for incorporating the previous frame's object position and shape into the current graph. We propose a method for tracking and segmenting objects in live video that utilizes regional graph cuts and object pixel probability maps. The regionalization of the cuts around the tracked object will increase the speed of the tracker, and the object pixel probability maps will enable more flexible tracking.

Index Terms— Tracking, Image segmentation, Image processing, Real time systems

1. INTRODUCTION

Tracking objects across a sequence of images is often combined with segmentation in computer vision. Systems have been developed that track and segment objects for 3D reconstruction (such as visual hulls) [1], as well as systems that fit models for pose analysis used in human-computer interaction [2]. We plan to show that graph cuts are a method to achieve both tracking and segmentation simultaneously, and are applicable to live video scenarios and real-time conditions.

Applying graph cuts to layered frames of a video sequence has shown to be a robust method of object tracking and object segmentation. In such scenarios, the entire video sequence is treated as a single 3-D volume of layered frames, and a cut is performed across the entire sequence at once [3]. The large size of the graphs causes the performance of the cut algorithm to be a concern. To solve this problem, advances in the graph cut algorithm have produced dramatic performance improvements [4, 5].

However, in live video applications, there is no prior knowledge of subsequent frames. Techniques that build single graph across the entire video [3] become impossible in live video situations. Instead, a graph must be maintained or created for each new frame. When considering live real-time video, using faster cut algorithms [4, 5], is not sufficient. The creation of the graph, as well as the structures for utilizing the temporal data from previous frames must be reconsidered.

One method is to track only the contours of object, using graph cuts to find object boundaries [6]. Yet, objects that move or change shape rapidly will be difficult to track and segment, since the contours will likely exceed the small region being examined by the graph cuts. Approaches also exist for background segmentation using Markov Chains in live video; however tracking of objects in static and non-static backgrounds requires new techniques [7].

In this paper, we propose an object tracking and segmentation system for live video where the graph is dynamically adapted to the motion of the object. This paper briefly introduces graph cuts and provides a detailed explanation of the technique. An analysis of the system performance on an indoor and an outdoor scene is presented. Finally, the technique and future research are discussed.

2. RELATED RESEARCH

The graph cut method is a technique for reducing energy minimization problems to maximum flow problems. Using image data, a graph is constructed where each vertex corresponds to a pixel, and each vertex is connected to each neighboring vertex by a weighted edge. After construction, a maximum-flow (minimum-cut) algorithm is used to locate the set of edges that can be removed to create two distinct components in the graph, at minimal cost.

Maximum flow algorithms continue to be a popular research topic in graph theory and combinatorial mathematics, resulting in many polynomial algorithms such as Ford-Faulkerson *shortest augmenting path* algorithms [8], and Goldberg-Tarjan *push-relabel* algorithms [9].

3. LIVE VIDEO GRAPH CUTS

The proposed solution relies on a five-step system to track objects in real-time in live video feeds using regions of interest. Then algorithm proceeds as follows:

3.1. Initialize the system

The first step in our system is to initialize a standard graph over the complete video frame. The graph will be seeded with object and background pixels in the same manner as other graph cut techniques. Depending on the application, this could be done automatically using some other kind of object detection algorithm. Once the seeds have been placed, the region of interest is initially set to the entire frame.

3.2. Build the graph

Each edge in the graph requires an energy computation, and as the size of the image increases, the number of edges increases dramatically. For example, an 8-neighborhood graph on a 640x480-resolution video has almost 2 million edges. This results in poor performance on live video because 2 million edges must be recomputed for every frame. If we are tracking and segmenting an object that does not fill the entire frame, many of the pixels will not need to be computed, such as those far from the object, since they will not likely change unless the object moves closer. To reduce the number of computations required, we restrict the graph to a region of interest.

For the very first cut on the graph, the region of interest will cover the entire video frame. However, subsequent regions will be smaller, depending on the object size.

3.3. Cut the graph

The second step is to cut the graph. For this step, we utilize a freely available graph cut implementation available online [10]. Research has already been done to increase speed of the cut [4, 5], so we will not discuss improvements here.

3.4. Re-seed the graph

Seeding the graph for the next frame is not a simple matter of marking the previous frame's object pixels as object seeds, and all other pixels as background seeds. In traditional graph cuts, a pixel seeded as either "background" or "foreground" would automatically be segmented into that pixel class. In our case, if the object were to move within the next frame and the seeds did not, then some pixels may have been inappropriately seeded, causing the cut to incorrectly label some pixels. Therefore, we need a method that seeds probabilities, rather than binary labels. We propose a new method for seeding pixels based on probabilities of future pixel locations.

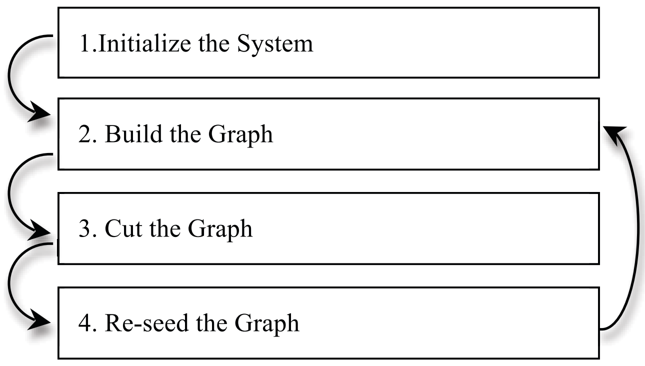

Figure 1. Diagram of the proposed system

When examining any two consecutive frames of a video sequence, many of the pixels of the object will overlap between frames. These will most likely be those pixels closer to the center of the object, as the object must travel farther before these pixels change from "object" to "background." Thus, a probably distribution can be created over the object pixels such that the pixels near the edge will have lower probability of remaining object pixels in the next video frame, whereas pixels near the center of the object have a high probability of remaining object pixels in the next video frame. This distribution is called the object pixel probability map.

To compute the pixel probability map, the binary mask of the object segmentation from the previous frame is computed. Then a distance transform is applied to the mask to obtain the pixel probability map. The distance transform computes the distance from a given white pixel to the closest black pixel. This gives us a gradient from the center of the object (greatest distance) to the cut edge (shortest distance).

Using the probability map, the *t*-links of the next frame's graph are computed using a modified version of Region Equation *R* by Boykov [3].

Table 1. t-Link Edge Weight Equations

Edge	Weight (cost)	For
$\{p,S\}$	$\min(\lambda+1, \lambda \cdot R_p(\text{"bkg"}) + \lambda \cdot D_p)$	$p \in P, p \notin O$
	0	$p \in O$
$\{p,T\}$	$\min(\lambda+1, \lambda \cdot R_p(\text{"obj"}) + \lambda \cdot D_p)$	$p \in P, p \notin B$
	0	$p \in B$

Where D_p is the value of the probability map for pixel p divided by the largest D, and thus $D_p \in [0, 1]$. The technique also requires that $R_p \in [0, 1]$, make sure that the maximum value of the *t*-link weight is $\lambda + 1$ (labeled as *K* [3]).

Finally, a region of interest is constructed around the object area of the binary image mask. The region of interest is then expanded to contain probable future locations of the object. Metrics for determining the potential size of for the region of interest include camera frame rate, previous object velocities, and object scaling speeds.

4. TESTING METHOD

To test our proposed system, we simultaneously used our system to track and segment an object on live video and recorded the output of the system to the hard disk for later comparison. This paper examines the results of two different tests performed at different resolutions and of different scenes. The first video [9] was at 352x240-resolution of an indoor scene with an object that moves and changes shape. The second video [10] was a 768x576-resolution outdoor traffic scene with low color variation in snowy conditions, making edge detection more difficult, and color modeling of the object and background more complex.

We ran the regionalized graph cuts along with the full image graph cuts and analyzed the frame rate, false positive pixels (uncut non-object pixels), and false negative pixels (cut object pixels). In the following section, we first present the findings for the indoor scene, followed by the outdoor scene. All tests were performed on a 2.4 GHz Intel Core notebook with 2 GB of RAM in a Mac OS X environment.

5. RESULTS & ANALYSIS

The indoor scene and the outdoor scenes exhibited dramatic performance improvements using regionalized cuts. The tracking ability of the graph cuts seeded with the object pixel probabilities proved to be very robust and accurate.

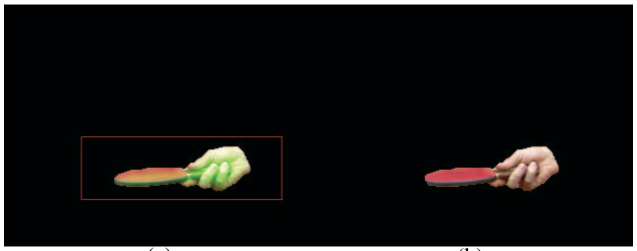

Figure 2. Indoor Scene: (a) region of interest (red) and pixel probability (green) overlaid on input image, (b) result of cut

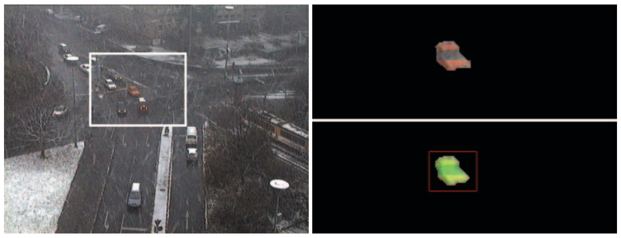

Figure 3. Outdoor Scene: (a) original scene (b) (top) cut result, (bottom) region-of-interest (red) and pixel probability (green) overlaid on input image

5.1. Indoor scene

Figure 2 shows a frame from the operation of the algorithm on the input video sequence. Figure 4 shows the regional cut outperformed the full image cut by 1.5x or more in speed. In addition, the performance of the regional cut is proportional to the size of the region of interest, as is expected. From frame 35 to frame 75, the frame rate decreases as the region of interest expands. On the other hand, the frame rate for the full image cut remains constant across the entire test, as expected, since the properties of the graph remain largely unchanged.

We also compared the balance of processing time spent on different portions of the system. Figure 5 shows the break down of time spent on each task for a graph cut on a single frame. By regionalizing the cuts to areas of interest, we have reduced graph construction and maintenance from requiring a majority of the processing time, to less than one fifth of the time. By consequence of the smaller graphs, the cut time was also reduced.

5.2. Outdoor scene

The outdoor scene displayed even more dramatic improvements than the indoor scene. While the full image cut on the 768x576-resolution image averaged 2 fps, the regional cut attained 18 fps on average, almost a 9x speed increase. This is attributed to the object size being smaller in comparison to the indoor scene. It is important to note that

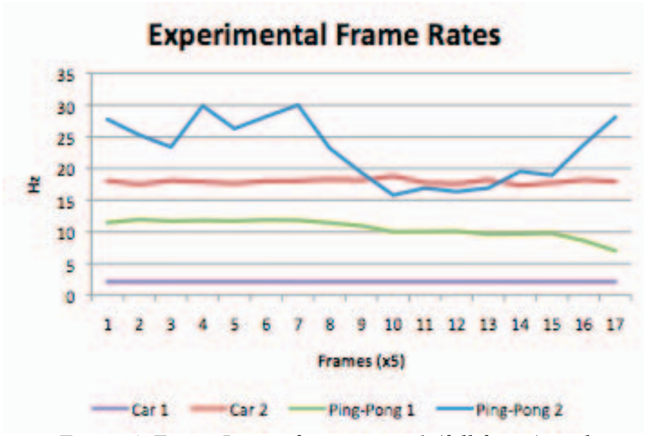

Figure 4. Frame Rates of experiment 1 (full frame), and experiment 2 (regionalized cuts)

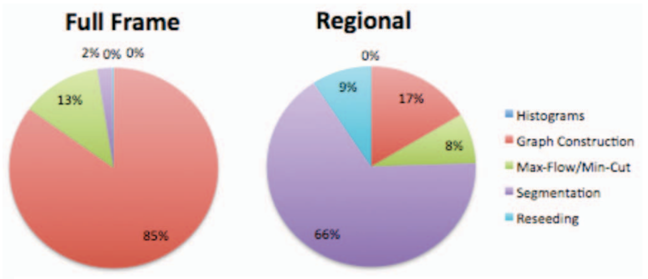

Figure 5. Comparison of graph cut performance based on size

Table 2. Indoor Scene: Accuracy of regionalize cut with probability map

	False Positives	**False Negatives**
Average	0.15%	0.15%
Worst	0.23%	0.34%

Table 3. Outdoor Scene: Accuracy of regionalize cut with probability map

	False Positives	**False Negatives**
Average	0.15%	0.15%
Worst	0.23%	0.34%

even due to the small size of the object and region of interest, the object was never lost during tracking, even though the wire mesh in the window partially occluded the object being tracked. In Figure 4, the efficiency increase of the regionalized cut is apparent. The object to be tracked is much smaller than the video frame, making graph cuts over the entire image inefficient.

5.2. Accuracy

A mask of the ground truth object was manually created for every other frame of the video segment. The number of object pixels that were cut (false negatives) and the number of background pixels that were not cut (false positives) were divided by the input image size to obtain the accuracy of the regional cut. Table 2 and Table 3 have the average and worst results for the both scenes. Even though many of the edges in the indoor scene were not clearly defined, the pixel probability map has performed extremely well in seeding the graphs and tracking the face in the video.

In addition, despite the smaller size of the graph, accuracy of image tracking and segmentation remains very high. As the entire object remains contained within the region of interest, the graph cut built using the pixel probability map correctly segments object in the image.

6. DISCUSSION AND FUTURE WORK

We have introduced a new graph seeding method and a region of interest based graph-minimizing technique as a system capable of tracking and segmenting objects using graphs cuts in live video sequences. Empirical analysis has shown the system to be capable of performing within real-time constraints on commonly available hardware.

While this method requires the user to initially seed the image with object and background locations, this method could easily be combined with other techniques to automate the initial seeding task, such as finding skin color to track human faces. Many other applications exist, such as traffic monitoring on highways, since the cars are usually small compared to the size of the image.

Further research in incorporating multiple-object tracking into the system is being considered. Possible techniques include using a single graph cut with multiple pixel probability fields and multiple graph cuts.

7. REFERENCES

[1] Laurentini, A., "The Visual Hull Concept for Silhouette-Based Image Understanding." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 2, pp. 150-162, February 1994.

[2] Ueda, E., Matsumoto, Y., Imai, M., and Ogasawara, T., "A Hand-Pose Estimation for Vision-Based Human Interfaces." *IEEE Transactions on Industrial Electronics*, vol. 50, no. 4, pp. 676-684, August 2003.

[3] Boykov, Y., and Funka-Lea, G., "Graph Cuts and Efficient N-D Image Segementation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109-131, 2006.

[4] Juan, O., and Boykoy, Y., "Active Graph Cuts," *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 1023, June 2006.

[5] Boykoy, Y., and Kolmogorov, V., "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124-1137, September, 2004.

[6] Mooser, J., You. S, and Neumann, U., "Real-Time Object Tracking for Augmented Reality Combining Graph Cuts and Optical Flow," *The Sixth International Symposium on Mixed and Augmented Reality,* pp. 145-152, November, 2007.

[7] Criminisi A., Cross, G., Blake, A., and Kolomogrov, V., "Bilayer Segmentation of Live Video," *Conference on Computer Vision and Pattern Recognition,* vol. 1, pp. 53 - 60, June 2006.

[8] Ford, L. and Fulkerson, D., "Flows in Networks," Princeton University Press, 1962.

[9] Goldberg, A.and Tarjan, R., "A New Approach to the Maximum Flow Problem," *Journal of the Association for Computing Machinery*, vol. 35, no. 4, pp. 921–940, 1988.

[10] *http://www.adastral.ucl.ac.uk/~vladkolm/software.html*

[11] Motion Imagery Database, Ohio State University. *http://sampl.ece.ohio-state.edu/database.htm*

[12] Image Sequence Server, Institut für Algorithmen und Kognitive Systeme. *http://i21www.ira.uka.de/image_sequences/*