Space-Time Multi-Resolution Banded Graph-Cut for Fast Segmentation

Tobi Vaudrey[1], Daniel Gruber[2], Andreas Wedel[3] and Jens Klappstein[3]

[1] The University of Auckland, New Zealand
[2] Universität Konstanz, Germany
[3] Daimler Group Research, Sindelfingen, Germany

Abstract. Applying real-time segmentation is a major issue when processing every frame of image sequences. In this paper, we propose a modification of the well known graph-cut algorithm to improve speed for discrete segmentation. Our algorithm yields real-time segmentation, using graph-cut, by performing a single cut on an image with regions of different resolutions, combining space-time pyramids and narrow bands. This is especially suitable for image sequences, as segment borders in one image are refined in the next image. The fast computation time allows one to use information contained in every image frame of an input image stream at 20 Hz, on a standard PC. The algorithm is applied to traffic scenes, using a monocular camera installed in a moving vehicle. Our results show the segmentation of moving objects with similar results to standard graph-cut, but with improved speed.

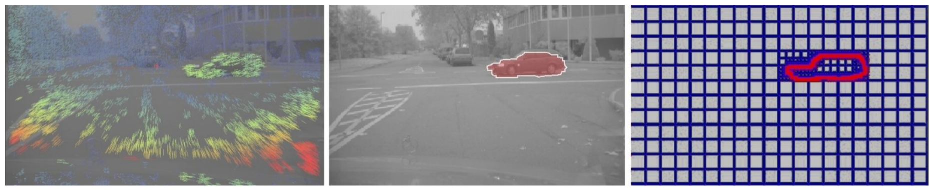


Fig. 1. The images show the flow field of image features in a traffic scene and a segmented moving object. The image on the right shows the multi-resolution graph structure obtained using the approach in this paper for real-time segmentation.

1 Introduction

Separating structures of interest from the background is a well known problem in the computer vision community. The goal of such segmentation is to reduce the representation of an image into meaningful data. Segmentation tasks include; identifying anatomical areas of interest in medical image processing [13], optimal contour finding of user defined foreground and background image areas [14, 19], and segmenting areas of similar motion for machine vision [20]. The goal of this paper is the separation of moving and stationary objects using a monocular camera installed in a moving vehicle. Due to the motion of the vehicle, simple background subtraction does not work in this case, because (apart from the focus of expansion) the whole image changes from frame to frame (see Figure 1).

This is also different to motion segmentation, namely the grouping of image regions which are similar in their motion, because for some constellations, such as motion of non-rigid objects, no parametric motion field exists. Last, the binary segmentation needs to be performed in real-time on video streams.

We formulate an energy function based on the boundary length, the number of foreground pixels, and the assigned probabilities of a pixel being foreground (moving object) or background (stationary world). Several techniques have been developed for image segmentation [15]. Two methods dominate the domain of segmentation by energy minimization in the image domain: differential methods, such as LevelSets [3, 12], and discrete methods. Among the discrete methods, the graph-cut algorithm [8] finds the global optimal energy in polynomial time. However, the time to compute the energy is still far from real-time for many applications. Real-time performance is important when working on image streams, where the information in every frame has to be processed. This is especially important in the field of motion segmentation where small time steps between consecutive images is important, as motion estimation is based on gradient descent and the displacement of pixels is crucial [4, 18].

The contribution of this paper is a modification of the input images, such that the graph-cut gains real-time performance on image sequences. We reduce the dimension of the search space for the segmentation, by executing the graph cut algorithm on different scales of the input image, in one single cut. The original image is segmented at a low resolution, then segmentation boundaries are refined throughout the image sequence. The cut is still optimal on the modified input images, such that no repeated iterations on the same image are required. Experimental results prove, that for most problems, where the segment boundary consists only of a small set of image pixels, this algorithm yields real-time performance on standard computer hardware.

The next section contains a literature overview, investigating approaches for speeding up the graph-cut algorithm. Our novel approach is described in Section 3. Results and comparisons to the algorithms described in literature are found in Section 4. The paper closes with conclusions and an outlook on future work.

2 Literature Overview

Graph-cut for image segmentation was first introduced by [5] for image restoration. Its impact on the computer vision community rapidly increased after the publication of [1], where an algorithm especially designed for the segmentation of image domains was proposed. The algorithm boosted average segmentation times from several minutes to a few seconds for usual image domains, where every pixel yields one node in the resulting graph. However, most research on graph-cut focuses on single images and not on real-time image streams, where computation time becomes crucial. In our experiments we use frame rates of up to 25 frames per second, demanding accordingly fast segmentations. In the remaining part of this section we will explain the graph-cut algorithm and review techniques to speed up graph-cut computation time.

2.1 Graph-Cut

The image being segmented is modelled as a weighted undirected graph. Each pixel corresponds to a node in the graph, and an edge is formed between two neighbouring pixels. We use an $\mathcal{N}_4$ neighbourhood where each pixel has four neighbours (upper, lower, left, and right) representing the Manhattan metric. The weight of an edge is a measure of similarity between end nodes (in this case pixels). We use the grey value difference of the end nodes as similarity measure. With the maximal grey value I_{max} and a scale factor c the weight of an edge connecting x and y, $e_{x,y}$ is:

$$w\left(e_{x,y}\right) = c \cdot \left(1 - \left(\frac{I(x) - I(y)}{I_{max}}\right)^2\right) . \qquad (1)$$

Additional edges connect nodes with a source and sink. They have weights corresponding to a node's foreground and background probability. To detect and segment moving objects we use the motion constraints described in [6]. The constraints are based on a motion analysis of individual tracked image features. We use a real time implementation of the tracking described in [17]. The tracked features are 3D reconstructed. A point is detected as moving if its reconstruction is identified as erroneous by checking whether it fulfils the constraints:

- **Epipolar Constraint:** This constraint ensures that the viewing rays (joining projection centres and the 3D point) from both cameras must meet. If this constraint is violated, no reconstruction is possible and the 3D position of the point has changed between both frames of the sequence.
- **Positive Depth Constraint:** If viewing rays intersect behind the camera, the 3D point must be moving.
- **Positive Height Constraint:** All static points in traffic scenes lie above the ground plane. If viewing rays intersect under the ground plane, the 3D point must be moving.

For details on how to exploit these constraints, refer to [6]. For every tracked point an error measure is computed, measuring the distance in pixels, to the next valid point fulfilling the above constraints. This error value serves as the weight for an edge from tracked pixels to the source (background) or sink (foreground). In our experiments we use $w = (1 + \exp(6 - 3x))^{-1}$.

The optimal s-t (source to sink) separating cut of the graph is one that minimizes the sum of the weights of the edges that are removed (referred to as energy), such that no more connections between source and sink exist. Figure 2 shows the model of a graph and its optimal cut.

2.2 Banded Cut on Image Pyramids

The simplest way to decrease computational time is using images of lower resolution, thus reducing the number of nodes and edges. However, this is at the expense of less segmentation accuracy, especially along the boundaries. In [9]

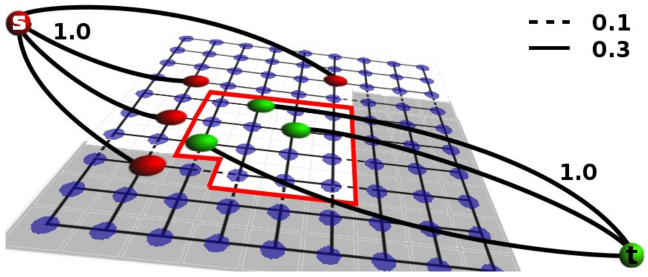


Fig. 2. The plot shows the segmentation of an image by performing an s-t cut. The pixels correspond to nodes of the graph (blue). Edges for foreground and background pixels connecting these pixels with the source and sink respectively have costs 1.0. Edges between neighbouring pixels, connecting their corresponding nodes in the graph have costs 0.3 or 0.1, depending on the grey value difference of the two pixels. The minimal s-t separating cut with cost 3.0 is given by the red boundary. Although we use sparse data representing the foreground and the background, one connected region with it's boundary along a grey value difference is found.

the authors propose the use of image pyramids and refine the boundaries on lower pyramid images as illustrated in Figure 3. When image features are small, they might not be detected on lower pyramid levels as they are smoothed out. One possibility to account for these changes was proposed in [16]. An additional Laplace pyramid is built, which represents the lost information from the image down-sampling process in the high frequency spectrum. If the values in the Laplace image exceed a certain threshold, additional pixels are introduced into the graph. The drawback of pyramid approaches is the inherent iterative process, where consecutive graph-cut steps on lower pyramid levels have to wait for the preceding step to finish. This causes the undesired effect of the graph-cut being carried out several times on the same input image, once for every pyramid resolution.

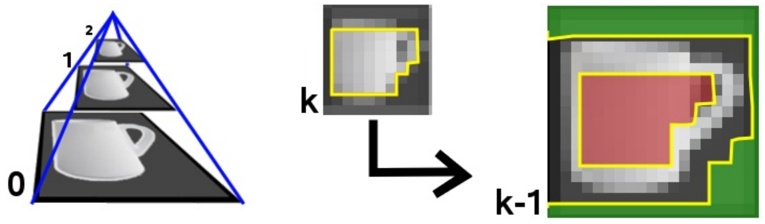


Fig. 3. The figure shows an example of an image pyramid (left). A full graph cut is performed on the highest pyramid level and the result is iteratively propagated downwards, then refined in a band around the result of the upper pyramid level.

2.3 Temporal Banded Cut

Recall that we are interested in detecting moving objects. If the frame rate is high, this implies that segmentation boundaries in consecutive images are similar. Instead of predicting the segmentation boundary from the highest to lowest pyramid level, the segmentation of the current image is used as a hint for the cut of the next image in the sequence, and refined in a band around the segmentation border. This is done by [10] and [11]. The major drawback of this method is that essentially only tracking is performed, therefore new objects that appear are only detected if they are within the band width of the boundary for existing objects. In the papers, a human initialization step is required. We propose to solve this problem by a temporal prediction on pyramids.

2.4 Temporal Prediction on Pyramids

Instead of propagating segmentation boundaries from a higher pyramid level onto a lower pyramid level of the same image, this propagation is done in the subsequent image of an image sequence (green arrows in Figure 4). Using the assumption that the segments motion in the image is roughly known, this combines the detection in the pyramid scheme and the temporal prediction (tracking) of segmentation boundaries. Another advantage of this method is that the graph-cut algorithms for the single pyramid levels can be executed in parallel, resulting in an improved computational time when using parallel processing computers.

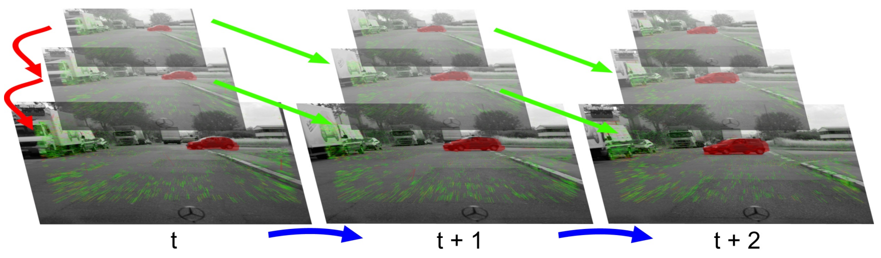


Fig. 4. This shows the temporal prediction on pyramids. Instead of propagating results down (red), or from one time instance to another on the same pyramid level (blue), a combined approach propagates results between frames and pyramid levels to combine both real-time detection and real-time tracking.

Other techniques to speed up the segmentation algorithm have been presented by [7]. The authors use the search tree of the old segmentation as a hint for the current segmentation. In [2], an approximation of the energy is proposed which yields faster convergence. Both approaches are applied on a graph without changing its topology. In the following section, a novel approach is presented related to temporal prediction on pyramids, but performing a single cut on an image with multiple resolutions. This is related to surface approximations, where primitives may consist of different scales, such that a good approximation is achieved with the constraint, to use a minimal number of primitives (e.g. 3D surface modelling).

3 Multi-Resolution Image

If parallel processing is not possible, the temporal prediction on pyramids yields no computation time gain compared to the plain vanilla pyramid implementation of the graph-cut algorithm. A closer look at the temporal prediction also reveals that situations occur, where the same segmentation boundary is found on every image level, such that a prediction onto the next time instance creates similar problems as using a full pyramid.

The idea to solve this issue is to take the highest pyramid image (lowest resolution) and iteratively refine the pixels into 4 sub pixels if they are within the segmentation border of the lower pyramid level (Figure 5 shows the result; see also Figure 6 for a schematic illustration). The resulting image consists of different resolutions representing different pyramid layers, thus different segmentation accuracies. A high resolution is obtained at segmentation boundaries (inhomogeneous background / foreground regions) and a low resolution at homogeneous image regions. In the next section we will show how to obtain a graph representation of a multi-resolution image. The advantage is obvious, as only a single cut has to be performed on the multi-resolution image.

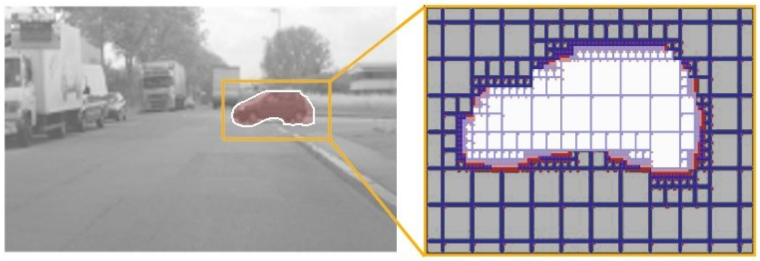


Fig. 5. The left image shows the original frame with a segmentation overlay. The right image shows a close-up of the calculated multi-resolution image. Areas around segmentation boundaries have high resolution and in the other areas, pixels are combined to provide a low resolution.

3.1 Multi-Resolution Graph-Cut

The mapping of the multi-resolution image onto the graph is straightforward. In the first frame t, a coarse grid is created (see Figure 6). Every grid section is a node, every node has an edge from the source and one to the sink. Pixels within the same grid are combined by adding their weights from the source and to the sink. The remaining edges, for each node, are created from the $\mathcal{N}_4$ boundary grid nodes (identified by any red pixel touching a blue grid line in Figure 6). The edge values are based on the energy function mentioned in Equation 1. The weights of all pixel edges connecting two grid cells are summed to obtain a single edge weight. The graph-cut is performed on the coarse grid, providing a rough segmentation.

In the next time frame $(t + 1)$, the grid sections on the boundaries of the segmentations are split into 4 subsections. The old node (grid section) and corresponding edges are removed from the graph and the new nodes are added.

Finally, the new edges are added to the graph. New edges can be seen in Figure 6 as any red node that is touching a blue grid line. This process is then refined again in the next frame $(t + 2)$. This will happen iteratively over time until the segmentation boundary is refined to single pixel detail.

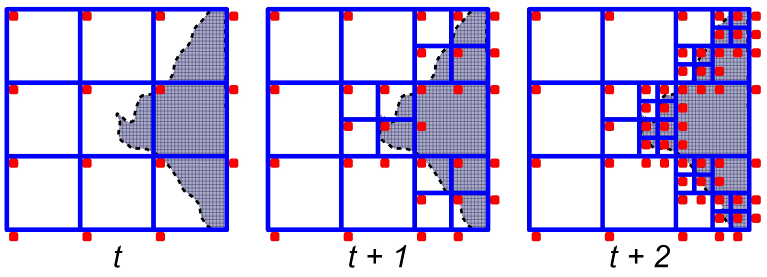
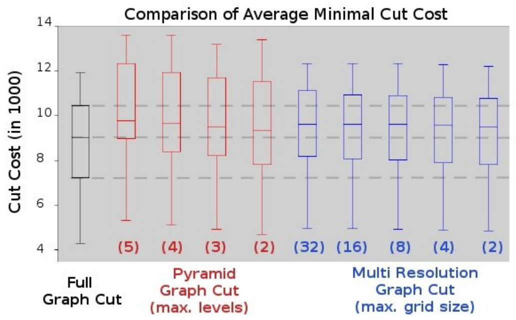


Fig. 6. The graph shows the iterative segmentation approach of the multi-resolution graph-cut. The dashed line shows the edge of the true segmentation. Blue grid lines identify graph sections (nodes). Red dots touching a blue grid line indicate edges.

As we are segmenting image sequences, we need to deal with the implicit movement within the images over time. The movement of the segmentation area can be estimated by using the tracked features used for the motion constraints. In this case, a segmentation boundary in frame $(t+1)$ is estimated from frame t using the motion constraint data. The segmentation in frame $(t+1)$ is performed as above, using the estimated boundary position. The graph-cut is performed over the entire image, still allowing new features to be detected.

Another issue that needs to be dealt with when using this approach is when there are refined areas in the image, that are not located on a segmentation boundary of the subsequent frame. When this happens, the process is reversed over the involved homogeneous grid cells, reverting to a lower resolution.

Figure 7 shows a comparison of our algorithm in terms of minimal cut costs, identifying a minor loss in minimization. Note: Do not mix up the cut costs, which deals with the energy minimum (see Figure 7), and the computational cost which is discussed in the results section.



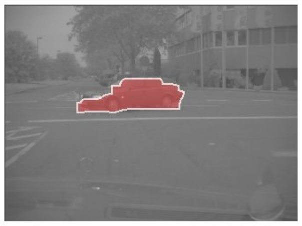


Fig. 7. The plot compares the calculated minimum cut costs for a VGA $(640 \times 480$ pixels) test sequence of 25 images. The minimum cut cost of the multi-resolution method increases slightly with larger grid size and is close to the cost of the full graph-cut.

4 Results

In this result section we show qualitative segmentation results of moving objects and a quantitative comparison of computation time for different speed improvement techniques for graph-cut segmentation. The segmentation and tracking results show that we are able to detect rigid objects, such as other vehicles (see Figure 8). The segmentation is meaningful and finds the moving object in the image. A close look reveals that vehicles driving away from the camera, at large distances, are not detected (Figure 8 upper right). This is due to the motion constraint, which is too weak for such situations. The error for each track is shown in the image demonstrating that sparse data is used for segmentation. Note that in the experiments, the camera is installed in a moving vehicle, so simple background subtraction is not possible.

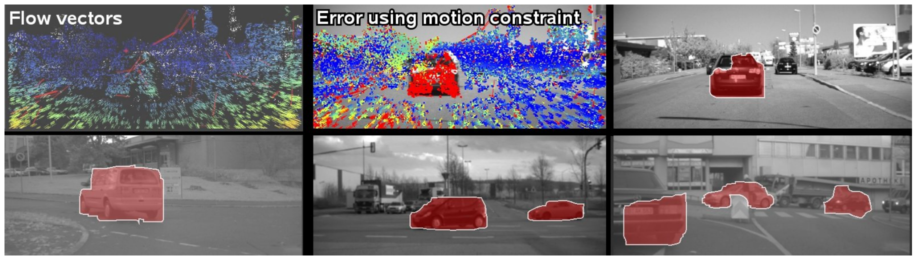


Fig. 8. Segmentation of moving objects using the multi-resolution graph-cut method and constraints on tracked features (20 frames per second on VGA images). The segmentation boundary proves to be accurate keeping in mind that sparse features are used. Outliers are rejected and the moving objects are verified.

While the example above could be solved by motion segmentation with parametric motion fields, the accurate segmentation of non-rigid motion would not be possible with such an approach. However, by using the motion constraint, we are able to segment a moving person quite accurately, as can be seen in Figure 9. The multi-resolution method generates segmentation results close to those obtained using the full graph-cut algorithm.

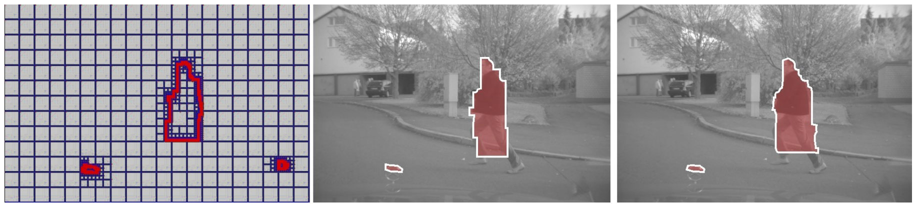


Fig. 9. Segmentation of non-rigid motion using the multi-resolution method in comparison to the full graph-cut method. The left image shows the multi-resolution graph used for segmentation, the middle image shows the segmentation overlay on the original image. The right image shows the results using original graph-cut, yielding similar results.

Figure 10 compares computation time for the multi-resolution method and full graph-cut. In addition, the pyramid version and the temporal prediction techniques (other techniques used to speed up graph-cut computation time) are included in the runtime comparison. As the temporal banded cut is used for tracking and not detection, the resulting boundary from the multi-resolution method was provided as an initialization in every frame. The sequence used had increased movement in the scene near the end, this explains the increase in computational time, especially noticeable using original graph-cut.

The multi-resolution method performed best on our test sequence. The results confirm that run times under 50 ms are achieved, such that real time segmentation becomes possible. The speed improvements using the multi-resolution approach, compared to the original graph-cut, lies between a factor of 10 and 20. The right hand graph in Figure 10 shows the different run times using decreasing maximum grid-size. Note that a max grid-size of 1 pixel corresponds to the original graph-cut.

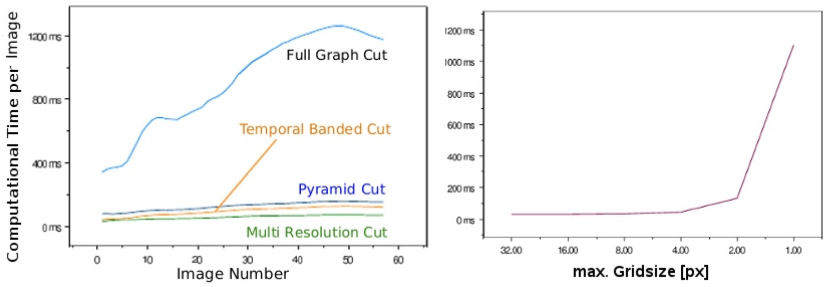


Fig. 10. The left plot shows the runtime comparison of different graph-cut implementations (on a Pentium IV 3.2 GHz). The right plot shows average runtime comparisons for the multi-resolution approach using different maximum grid sizes.

5 Conclusion and Future Work

In this paper we presented a novel method to combine tracking and detection in a real-time graph-cut approach. The multi-resolution method outperforms the standard graph-cut in computation time, on average, by a factor of 10. It also outperforms other speed improvement techniques, but by a smaller factor. The future ideas for this approach are to expand the algorithm to other areas, such as:

- The usage of the algorithm on other segmentation problems (e.g. stereo and general motion segmentation into segments of parametric (affine) motion).
- The generalization of the approach for 3D volumes.

References

1. Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 359–374, 2001.

2. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 1222–1239, 2001.
3. D. Cremers, M. Rousson, and R. Deriche. A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International Journal of Computer Vision*, 72(2):195–215, April 2007.
4. D. Cremers and S. Soatto. Motion competition: A variational framework for piecewise parametric motion segmentation. *International Journal of Computer Vision*, 62(3):249–265, 2005.
5. D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51:271–279, 1989.
6. J. Klappstein, F. Stein, and U. Franke. Monocular motion detection using spatial constraints in a unified manner. In *IEEE Intelligent Vehicles Symposium*, pages 261–267, 2006.
7. P. Kohli and P. H. S. Torr. Effciently solving dynamic markov random fields using graph cuts. In *ICCV*, pages 922–929, 2005.
8. V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 147–159, 2004.
9. H. Lombaert, Y. Sun, L. Grady, and C. Xu. A multilevel banded graph cuts method for fast image segmentation. In *ICCV*, pages 259–265, 2005.
10. J. Malcolm, Y. Rathi, and A. Tannenbaum. Multi-object tracking through clutter using graph cuts. In *ICCV*, pages 1–5, 2007.
11. J. Mooser, S. You, and U. Neumann. Real-time object tracking for augmented reality combining graph cuts and optical flow. In *ISMAR: Int. Symposium on Mixed and Augmented Reality*, pages 145–152, 2007.
12. S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging,Vision,and Graphics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.
13. D. L. Pham, C. Xu, and J. L. Prince. Current methods in medical image segmentation. In *Annual Review of Biomedical Engineering*, volume 2, pages 315–337. 2000.
14. C. Rother, V. Kolmogorov, and A. Blake. “GrabCut”: interactive foreground extraction using iterated graph cuts. 23(3):309–314, Aug. 2004.
15. L. G. Shapiro and G. C. Stockman. *Computer Vision*. New Jersey, Prentice-Hall, 2001.
16. A. K. Sinop and L. Grady. Accurate banded graph cut segmentation of thin structures using laplacian pyramids. In *MICCAI*, pages 896–903, 2006.
17. F. Stein. Efficient computation of optical flow using the census transform. In C. E. Rasmussen, H. H. Bülthoff, B. Schölkopf, and M. A. Giese, editors, *DAGM-Symposium*, volume 3175 of *Lecture Notes in Computer Science*, pages 79–86. Springer, 2004.
18. C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, 1991. Carnegie Mellon University Technical Report CMU-CS-91-132.
19. J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen. Interactive video cutout. *SIGGRAPH*, 24:585–594, 2005.
20. A. Wedel, T. Schoenemann, T. Brox, and D. Cremers. Warpcut - fast obstacle segmentation in monocular video. In *Proc. Pattern Recognition (DAGM)*, pages 264–273, 2007.