https://doi.org/10.2352/ISSN.2470-1173.2017.15.DPMI-066

A novel framework for fast MRF optimization

Gowri Somanath [1]***, Jiajie Yao*** [2]***, Yong Jiang*** [2]
[1] ***Computational Imaging Lab, Intel Labs, USA.*** [2] ***Intel Corporation, China.***

Abstract

Many computer vision tasks such as segmentation, stereo matching can be presented as a pixel labeling problem, which can be solved by optimizing a Markov Random Field modeling it. Most methods using this formulation treat every pixel as a node connected to its neighbors. Thus the compute requirements are directly proportional to the image size. For example a 720p image with 4-connectivity leads to 1 million nodes and 2 million edges. This is further scaled by the number of labels. With increasing resolution of cameras the traditional scheme does not scale well due to high compute and memory requirements, especially in mobile devices. Though methods have been proposed to overcome these problems, they still do not achieve high efficiency. In this paper we propose a framework for MRF optimization that significantly reduces the number of nodes through adaptive and intelligent grouping of pixels. This reduces the problem size in general and adapts to the image content. In addition we also propose a hierarchical grouping of labels, allowing for parallelization and thus suitable for modern processing units. We demonstrate this novel framework for the application of RGB-D scene segmentation and show up to 12X speed-up compared to the traditional optimization algorithms.

Introduction

Markov Random Fields (MRFs) have been widely used in low-level computer vision such as image segmentation [8], image restoration [3], and stereo matching [9]. These tasks can be presented as a pixel-labeling problem, where each pixel $p \in P$ must be assigned a label from known set L. The goal is to find a labeling f that assigns each pixel $p \in P$ a label $f_p \in L$, where f is both piecewise smooth and consistent with the observed data. It can be solved by minimizing a Markov Random Field modeling the specific problem [1]:

$$E(f) = E_{smooth}(f) + E_{data}(f) \tag{1}$$

Where E_{smooth} (smoothness term) measures the extent to which f is not piecewise smooth, E_{data} (data cost) measures the disagreement between f and the observed data. For two neighboring pixels p,q, the forms of E_{smooth} and E_{data} are typically [1]:

$$E_{data}(f) = \sum_{p \in P} D_p(f_p) \tag{2}$$

$$E_{smooth}(f) = \sum_{\{p,q\} \in N} V_{\{p,q\}}(f_p, f_q) \tag{3}$$

Among the various optimization methods, Graph-Cuts based approaches have become the mainstream in the last decade, since [1, 2, 3] proposed two algorithms ($\alpha - \beta$ swap and α expansion) to solve the multi-label assignment problems. The two algorithms can achieve a local minimum which is less than a constant factor times the global minimum.

Even though Graph-Cuts based methods are powerful in solving MRF, it does not scale well with increasing image resolution. The complexity of the optimization procedure, hence the runtime and memory usage, are determined by three main factors: the number of nodes in the graph, the connectivity or edges between nodes, and the formulation of the energy function using the costs. To the best of our knowledge, most currently known methods in literature treat every pixel of an image to be a node and the connectivity is 4 to 8 neighbors. Thus the computing required becomes directly proportional to the image size. For example a 720p image with 4-connectivity leads to graph of near 1 million nodes and 2 million edges. For high resolution images, the optimization procedure could occupy large amount of memory, which will lead to poor memory accessing performance. In addition the algorithms' complexity grows exponentially with the scale of the problem. In this paper we propose a framework for MRF optimization that significantly reduces the number of nodes through adaptive and intelligent grouping of pixels. This reduces the problem size in general and also adpats to the image content. In addition, we also propose a hierarchical grouping of labels, allowing for parallelization during the optimization and thus more suitable for modern processing units.

In order to make MRF or Graph-Cuts feasible for large scale problems, many approaches have been proposed to reduce the memory usage and speed up the optimization procedure. Multiresolution techniques [4, 5, 6] have been explored to reduce the graph size for interactive segmentation algorithms. The "banded graph-cuts" approach [5, 6] work on image pyramids where finer level graphs are only contructed around a narrow band between the binary label segmentation. The approach though effective for specific binary image segmentation is not a general framework for reducing the compute complexity. Lerme et.al [10] proposed a strategy for reducing graphs by testing each node during its creation if it is really useful to the max-flow computation. Although it reduced memory usage significantly, the speedup is only 1.7X on the given example. Delong and Boykov [7] developed a parallelized max-flow algorithm yielding near-linear speedup with the number of processor, however it's speedup depends on the number of cores on the device and it cannot reduce the memory usage of constructed graphs. Methods such as those proposed by Li et.al [11] uses super-pixels as nodes of the graphs, where the super-pixels are generated using the color information in the image. In our method nodes are non-uniform and are typically much larger than super-pixels, making the method adapt to the image complexity thus providing larger reductions for instance in images with large portions of similar regions (in color, depth, texture, etc). Towards our second contribution of hierachical label space for parallization, the closest work in literature is that pro-

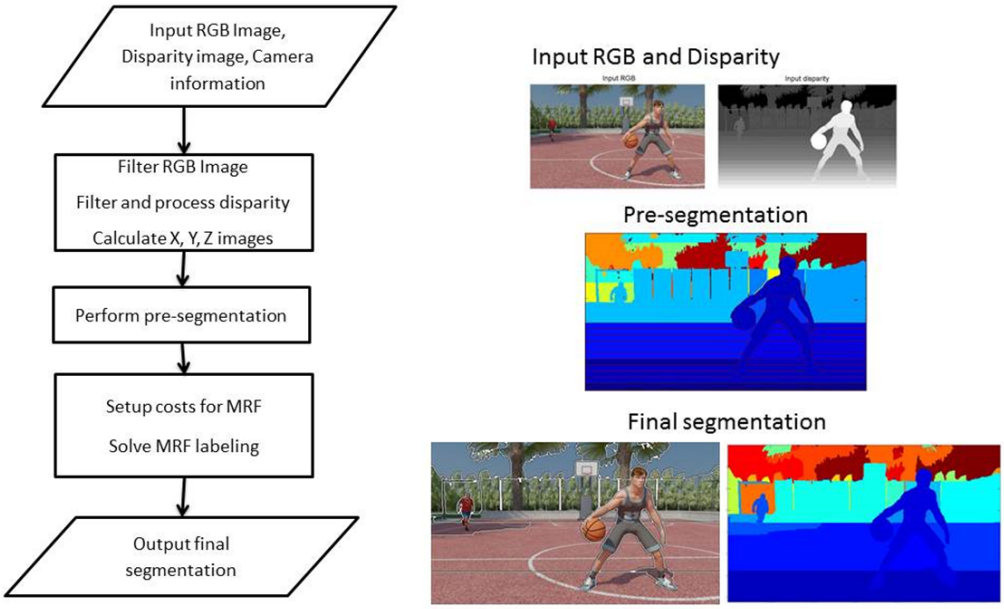


Figure 1: Overview of segmentation pipeline. The colors in the segmentation maps indicate the different labels/segments.

posed by Rastogi et.al [12]. However their formulation is still over the pixels in the image. The hierarchy in [12] is essentially a sub-sampling of the image resolution, while in our case it is in the label space as detailed later.

In this work we propose a novel way to formulate the graph structure for MRF optimization. It significantly reduces the number of nodes in MRF through adaptive and intelligent grouping of pixels. In addition, a hierarchical label space is constructed to allow for parallelization during the optimization. Our main contributions include:

- Using the constructed data cost in eq(2) and neighborhood costs in eq(3), we adaptively merge pixels into blocks, each block becomes a node in the graph. This step is detailed in section Adaptive Pixel Merging.
- Applying a hierarchical optimization strategy. In our current implementation we employ a 2-stage optimization. This reduces the label search space for each stage, and allow for parallelism. It is detailed in section Hierarchical Expansion.

We applied the proposed framework to RGBD image segmentation pipeline (modeled by MRF and solved by Graph-Cuts). We achieved up to 11X speedup for the optimization procedure, with no change in result quality. In section Traditional Pipeline, we will briefly introduce the traditional pipe line of Graph Cuts based RGBD image segmentation. And in section Adaptive Pixel Merging and section Hierarchical Expansion, we will introduce our two strategies to speed up the MRF optimization respectively. We provide quantitative comparisions in the Results section.

Traditional Pipeline

A common pipeline for segmentation using RGBD information is shown in Fig.1. The input to the algorithm includes the RGB color image, a disparity or depth map, and camera parameters of focal length and baseline from which the disparity was

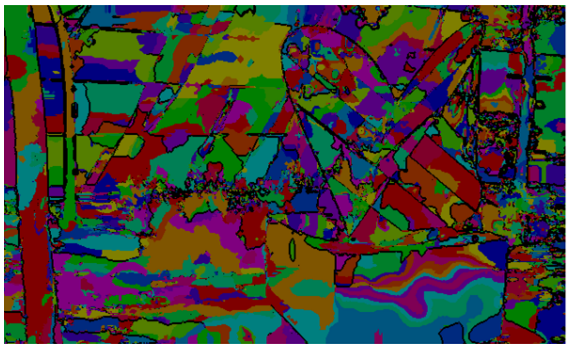
Figure 2: Overview of the proposed MRF optimization framework.

estimated. In the first stage, the input RGB image is filtered by Gaussian blur filter and pyramid based mean shift filtering. The operation helps to reduce noise and quantizes the color channel. The world X, Y, Z coordinate maps are also calculated from disparity map and calibration during this stage. In the next stage, pre-segmentation is performed to generate a rough segmentation (labeling) based on the depth distribution of the scene. We used adaptive clustering to generate bins of depth with various ranges, and these bins represent the labels which will be assigned to pixels. At the last stage, we set up the MRF and solve it using Graph Cuts to obtain an optimal segmentation.

In further discussions we denote P as the set of pixels in the image, $L = \{L_1, L_2, ..., L_K\}$ the set of labels corresponding to the depth bins, and f is the labeling/mapping that assigns a label $f_p \in L$ to each pixel $p \in P$. The term E_{data} is calculated based on the world X, Y, Z coordinates, position and color of each pixel. Intuitively it measures the cost of assigning a given label to a given pixel. The term E_{smooth} is calculated based on the differences of world X, Y, Z coordinates, and colors between two neighbor pixels. This term maintains smoothness (same label) over two neighboring pixels which have similar color and/or depth.

Once we solve the above optimization, we obtain the final segmentation or labeling as shown in Fig.1. In this approach: P, the set of nodes, is the same as the set of pixels in an image. Thus for a 1280x720 pixel image, it will contain nearly 1 million nodes. And as the neighborhood N is defined as edge between the pixels, with 4-Neighbor connecting, the graph will contain nearly 2

(a) the original image



(b) Merged blocks

Figure 3: Merging results

million edges. The alpha-expansion is iteratively applied until the convergence of Eq.1 (that is, change in total energy between two iterations is below a set threshold). Each iteration is a binary cut for a given label L_i and all other labels from set L, thus K expansions are done for K labels. This large formulation (K iterations for graph of millions of nodes and edges) contributes to the long runtime and memory consumption.

To handle the above problems, we designed and implemented two optimizing strategies (gray-background boxes in Fig.2), which will be introduced in the following sections.

Adaptive Pixel Merging

For a given application, data and smoothness cost are tuned to provide meaningful output without under or over segmentation. Thus to reduce the graphs to be constructed, we merge pixels with similar distribution of data cost $D_p(\cdot)$ and large neighborhood cost $V_{(p,q)}(\cdot)$ into blocks that become nodes for the graph. In our implementation the labels are generated through the pre-segmentation using RGB and disparity/depth data. Fig.3 shows an example of the merging result. Fig.3a is the input RGB image, Fig.3b shows the blocks generated from the merging process detailed further. Each colored block is a single node in MRF and the black pixels indicate the ones that haven't been merged. Note that these are of irregular shapes and sizes in contrast to other known methods in literature such as super-pixels[11]. Since super-pixels are typically generated by using a fixed desired number for all images, they cannot take advantage of larger homogenous regions that our scheme is able to do.

We use the following merging criteria:

a. Pixels to be merged should have similar D_p and large neighborhood cost, that is, pixels p1 and p2 can be merged only if there exists one label l such that $D_{p1}(l) = min(D_{p1}(\cdot))$ and $D_{p2}(l) = min(D_{p2}(\cdot))$. At the same time, $V_{(p1,p2)}(f_{p1}, f_{p2}) \geq threshold$ for $f_{p1} \neq f_{p2}$. This threshold is calculated based on neighborhood cost histogram over all pixels. In our implementation we choose the threshold to be the one which is larger than 70% of all neighboring pixels $V_{(p,q)}(f_p, f_q)$ values. This criteria is tuned to ensure we do not suffer from over-segmentation (that is, many nodes each of very few pixels).

b. Each block's size cannot exceed a maximum value to avoid under-segmentation. In our implementation, this value is set to 10000 for 720p images.

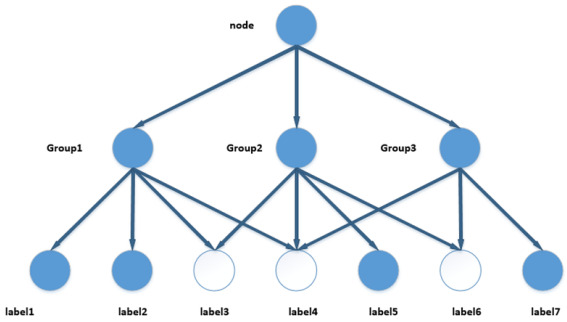


Figure 4: Label hierarchy.

After pixel merging, the resultant number of nodes in the constructed graphs is only about 25% of the original approach for the example in Fig.3. As the blocks now become the nodes in graphs, we need to update the original data term and smoothness term in eq(2) and eq(3): for each block the data terms of the included pixels will be summed to form the data term of the block. For each pair of neighboring blocks, the updated smoothness term is calculated by summing over the smoothness terms on the boundary of the two blocks.

Hierarchical Expansion

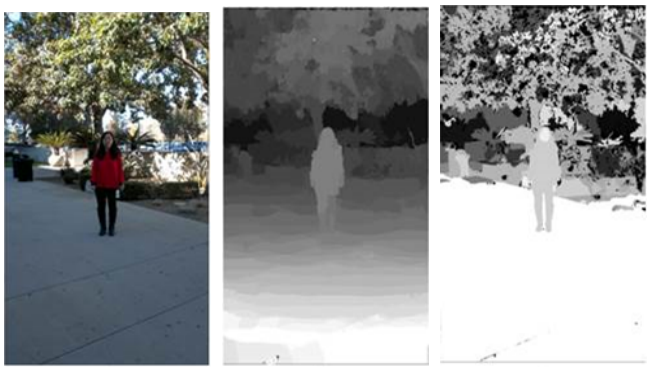
In the traditional MRF and Graph Cuts scheme, the optimization is run for all labels in each iteration. We propose a hierarchical expansion where the labels are grouped and the optimization is run first to determine the group and then at the lower levels till the final label set L is done. The number of levels in this hierarchy can be adapted for each application. In our implementation we use a 2 level hierarchy of the label space as shown in Fig.4.

In Fig.4, we have 7 labels shown at the lowest level and 3 groups. A label can belong to one or more groups as indicated by the non-colored labels (label 3,4 and 6 for example). The 2-level alpha-expansion is applied as following:

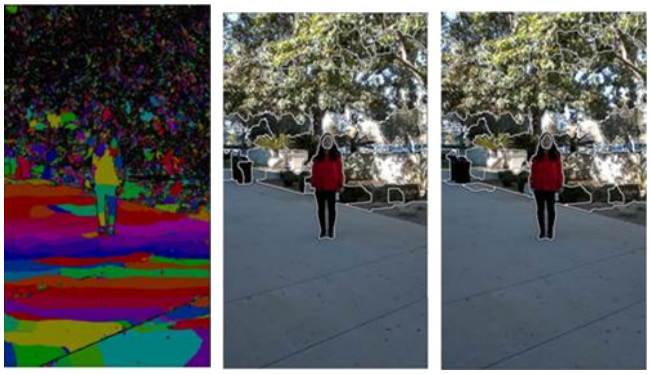
a. Apply global expansion: Segment the nodes into groups (groups 1-3 in our example)

b. Apply local expansion in parallel: Within each group, segment the nodes into individual labels.

Compared to flat label space this approach allows for parallelization. Thus our implementation can take advantage of the multiple threads available in most modern multi-core processors,



(a) input RGB data (b) input disparity data (c) pre-segmentation



(d) result of pixel-merging (e) result from traditional pipeline (f) result from the proposed framework

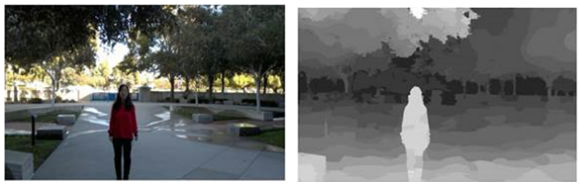
Image resolution	1280×720
Number of labels (label groups)	124 (67)
Number of nodes after merging	315,260 (34%)
Optimization time - traditional	200.434 s
Optimization time - proposed	16.81 s
Speedup	$11.92\times$

Figure 5: Results from case 1. Runtimes measured on Intel Core i5 PC.

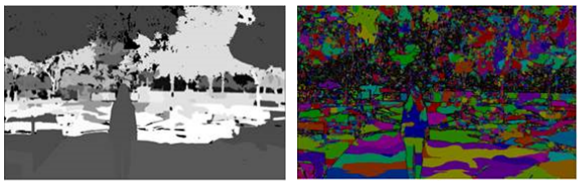
as the local expansions can be done in parallel. A simple way to group the labels would be to group them randomly (and evenly) into a specified number of groups. In this work, we employed a more robust strategy: First, a rough segmentation is estimated by a fast greedy approach, in which each node is assigned to the label with minimum data cost $D_p(\cdot)$. Then based on the rough segmentation, we build the label hierarchy using the following criteria: In the rough segmentation, each label will occupy one or more areas in the image. Assume that there two areas in the rough segmentation – A1 and A2, where label L1 is assigned to $A1$, and label $L2$ to $A2$. Whether $L1$ and $L2$ will be grouped is determined by the following procedure: we calculate the sum of neighborhood term between two labels $(L1, L2)$ as:

$$V_{sum} = \sum_{f_p=L1, f_q=L2, (p,q)\in N} V_{(p,q)}(f_p, f_q) \tag{4}$$

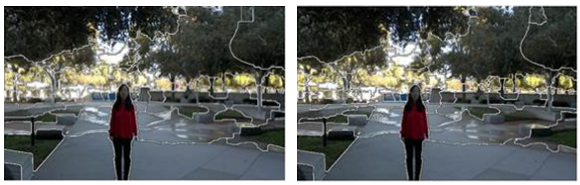
where N is the set of all neighboring pixel pairs, f_p and f_p are the labels of pixels p and q in the rough segmentation. Then the two labels ($L1$ and $L2$) can be combined in one group only when the



(a) input RGB data (b) input disparity data



(c) pre-segmentation (d) result of pixel-merging



(e) result from traditional pipeline (f) result from the proposed framework

Image resolution	1280×720
Number of labels (label groups)	76 (61)
Number of nodes after merging	189,080 (21%)
Optimization time - traditional	54.92 s
Optimization time - proposed	10.43 s
Speedup	$5.5\times$

Figure 6: Results from sample case 2. Runtimes measured on Intel Core i5 PC.

following equation holds:

$$\frac{V_{sum}}{A1+A2} \geq threshold \tag{5}$$

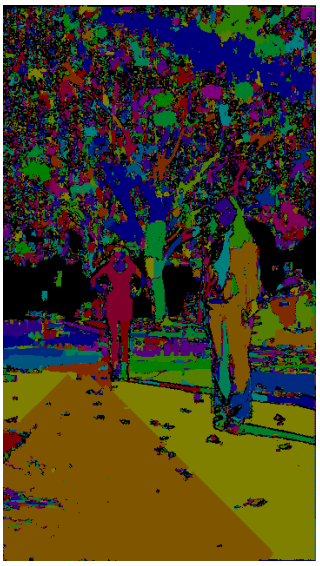
The threshold is calculated according to the histogram of neighborhood cost between all the labels (in the rough segmentation), in our implementation, we choose it to be the 75% to 90% quantile in the histogram. Initially, we create a new group for each label, and then add labels to groups when the above criteria meets. Groups with large intersections are combined. The intuition behind this procedure is that spatially non-interacting labels can be part of non-overlapping groups as they are unlikely to have effect on the alpha-expansion iterations. In addition, labels of pixels are likely to interact in the iterations with neighboring pixels and their labels, thus allowing for labels to belong to more than one group ensures that it doesnt cut that interaction.

Both data term and smooth term need to be updated after label grouping. For each node, its data term to one group is defined as the minimum data term of the included low-level labels. The smoothness term in eq(3) related to label groups $G1$ and $G2$ is defined as follows:

$$V_{(p,q)}(G1, G2) = \max_{f_p\in G1, f_q\in G2} V_{(p,q)}(f_p, f_q) \tag{6}$$

Once all the levels in the hierarchy are optimized, we obtain

(a) RGB with segmentation



(b) result of pixel merging

Image resolution	1280×720
Number of labels (label groups)	54 (27)
Number of nodes after merging	98,659 (10.7%)
Optimization time – traditional (PC)	26.02 seconds
Optimization time – proposed (PC)	3.73 seconds
Speedup on PC	6.9×
Optimization time - traditional (tablet)	40.62 s
Optimization time - proposed (tablet)	6.84 s
Speedup on tablet	5.9×

Figure 7: Results from sample case 3. Runtimes measured on Intel Core i5 PC and tablet with Intel Atom.

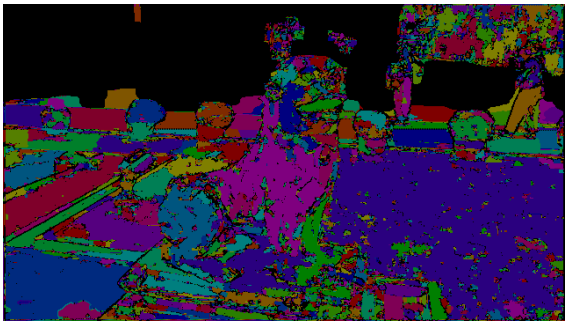
the final labeling for the input which determines the final segmentation. As will be shown in the results, it offers significant speedup compared to traditional formulation.

Results

We made quantitative comparison for a representative dataset containing over 30 images with 1280 x 720 resolutions, and average of about 105 labels. The tests were taken on two platforms: (i) A PC with Intel core i5 Haswell processor (1.3GHz X 4), 16G RAM; (ii) a tablet with Intel Atom processor (Dell Venue 8). A sample set of cases are detailed in Figures 5 - 8. The reported numbers were avergaged over multiple runs and the cost function used for traditional and our pipelines were the same. Overall for our datasets containing a wide variety of indoor and outdoor scenes with varying number of labels, we observed:

a. Up to 12X speedup up for Graph Cuts optimization, with 8.54X on average.
b. Peak memory usage is 71% of the original method on average. In the traditional approach the peak memory usage is over 900M,while in our framework peak memory usage is about 640M.
c. We also verifed that the final energy after optimization is almost same between the two approaches.
d. The runtime is reduced by 25% by using hierarchical expansion in addition to pixel merging.

(a) RGB with segmentation



(b) result of pixel merging

Image resolution	1280×720
Number of labels (label groups)	30 (15)
Number of nodes after merging	79,576 (8.6%)
Optimization time - traditional (PC)	15.02 s
Speedup on PC	5.1×
Optimization time - traditional (tablet)	20.48 s
Optimization time - proposed (tablet)	4.34 s
Speedup on tablet	4.7×

Figure 8: Results from sample case 4. Runtimes measured on Intel Core i5 PC and tablet with Intel Atom.

Conclusion

In this work we proposed a novel framework for fast MRF optimization in computer vision problems. Though popular and effective, graph-cuts based optimization for MRF do not scale well with increasing image resolution. Previously proposed schemes to reduce memory and/or time complexity for the optimization have been limited in their generality of application and the total gain in compute efficiency for both time and memory. To address the above shortcomings our framework applies two strategies to speedup the MRF optimization procedure: an adaptive way of pixel merging and a hierarchy of label space. The two contribute to significant reduction in memory requirements, up to 12X speedup on mobile and PC platforms, and parallization using multiple threads/cores in modern CPUs, without increasing the final optimal energy (eq(1)). The generalitity of our framework allows it to be applied to various computer vision problems and target processing platforms.

References

[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[2] Kolmogorov, Vladimir and Zabih, Ramin What Energy Functions Can Be Minimized via Graph Cuts? *European Conference on Computer Vision*, 2002.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.

[4] Patrick Prez and Fabrice Heitz Restriction of a Markov Random Field on a Graph and Multiresolution Statistical Image Modeling. (IEEE Transactions on Information Theory), 1996.

[5] C. Xu, Y. Sun, H. Lombaert and L. Grady A Multilevel Banded Graph Cuts Method for Fast Image Segmentation. *International Conference on Computer Vision (ICCV)*, 2005.

[6] A. K. Sinop and L. Grady. Accurate banded graph cut segmentation of thin structures using laplacian pyramids. *Medical image computing and computer-assisted intervention : MICCAI International Conference on Medical Image Computing and Computer-Assisted Intervention, 9(Pt 2):896–903*, 2006.

[7] A. Delong and Y. Boykov. A Scalable graph-cut algorithm for N-D grids. (IEEE Conference on Computer Vision and Pattern Recognition), 2008.

[8] Yuri Y. Boykov and et al. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. *IEEE International Conference on Computer Vision*, 2001.

[9] Scharstein, Daniel and Szeliski, Richard A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 2002.

[10] N. Lermé, F. Malgouyres, and L. Létocart. Reducing graphs in graph cut segmentation. *Proceedings - International Conference on Image Processing, ICIP*, pages 3045–3048, 2010.

[11] Z. Li and X.-m. Wu. Segmentation Using Superpixels : A Bipartite Graph Partitioning Approach. *IEEE International Conference on Computer Vision and Pattern Recognition*, 2012.

[12] A. Rastogi and B. Krishnamurthy. Localized hierarchical graph cuts. *Proceedings - 6th Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP 2008*, pages 163–170, 2008.

Author Biography

Gowri Somanath recieved her B.E. in Information Science & Engineering from PES Institute of Technology, India (2006), and her Ph.D. in Computer Science from University of Delaware (2012). She is currently a Research Scientist in Intel Labs focussing on computational photography and computer vision systems and algorithms.

Jiajie Yao received his B.S. (2011) and M.S. (2014) in Control Theory and Engineering from Zhejiang University. He is currently a Software Engineer in Intel Asia-Pacific R&D Ltd., focusing on image processing and computer vision.

Yong Jiang received his B.S. (1998) in Computer Science from Shanghai Jiao Tong University. He is currently a Software Engineer in Intel Asia-Pacific R&D Ltd., focusing on image processing and computer vision.