In IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 1998, Santa Barbara, CA.

Segmentation by Grouping Junctions *

Hiroshi Ishikawa Davi Geiger

Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012
{ishikawa, geiger}@cs.nyu.edu
http://simulation.nyu.edu/cvgroup.html

Abstract

We propose a method for segmenting gray-value images. By segmentation, we mean a map from the set of pixels to a small set of levels such that each connected component of the set of pixels with the same level forms a relatively large and "meaningful" region. The method finds a set of levels with associated gray values by first finding junctions in the image and then seeking a minimum set of threshold values that preserves the junctions. Then it finds a segmentation map that maps each pixel to the level with the closest gray value to the pixel data, within a smoothness constraint. For a convex smoothing penalty, we show the global optimal solution for an energy function that fits the data can be obtained in a polynomial time, by a novel use of the maximum-flow algorithm. Our approach is in contrast to a view in computer vision where segmentation is driven by intensity gradient, usually not yielding closed boundaries.

1. Introduction

Image segmentation is a prototypical problem in computer vision, where one needs to organize the image and separate figure from ground.

This problem incorporates and goes beyond edge detection, since the output of the system must be regions delineated by closed contours.

1.1. Background

Of course, the large clustering community [13] have discussed various distinct approaches to this problem, e.g., the merge and split techniques, the K-mean approach, and others. Usually, they are not described as solutions to clear optimization criteria.

A class of approaches to this problem is an extension of the edge-detection view of images [1, 4, 8, 9, 15, 16], where image contrast (intensity gradient information) with some grouping process yields the final image boundaries. These techniques do not always yield closed boundary contours as the output, and they are never guaranteed to be optimized in polynomial time on the size of the image. Gradient approaches have other difficulties. While it is true that usually large image gradient are perceived as region boundaries, small intensity changes can also yield region boundaries (illusory contours being an extreme example).

Region approaches are our main interest. Most of the work, however, is ad-hoc when predefining the number of regions or predefining values for the regions. Layers approaches have shown promises in motion segmentation (Weiss [21]), and they yield segmentation directly (Darrell and Pentland [6]). Their usual problems are to estimate the number of layers and the values associated with the layers, where ad-hoc methods or prohibitive optimization computations (e.g., reducing to EM algorithms) are employed.

Recently, Shi and Malik [20], in a related approach to the one by Sarkar and Boyer [19], have proposed an interesting method that uses graph partitioning techniques to find what they call the normalized cut. Because their approach is again computationally prohibitive to solve exactly, they use a generalized eingenvalue approximation technique. The maximization of total dissimilarity between different groups of pixels and similarity within groups is interesting and we also consider it in a different form.

The optimization step in our approach is in the spirit of graph partition, but we map our optimization problem (grouping criteria) to a minimum cut problem on a directed

*This work was supported by NSF CAREER award and AFOSR Grant No. F49620-96-1-0159.

graph. An advantage of our approach is that the globally optimal solution is obtained in a polynomial time by the use of the maximum-flow algorithm. Moreover, we propose a new way of estimating the number of layers and their values based on junction information.

1.2. Our approach

A segmentation is a classification of pixels into a small set of labels, which we call *levels* in this paper, because here each of them is associated with a gray value. Suppose we assign to each pixel the level with the nearest gray value. This is likely to yield an unsatisfactory result if the number of levels and their associated gray values are chosen arbitrarily. Thus, we wish to determine the optimal set of levels and gray values. We do this by detecting junctions in the image and choosing gray values that are needed to preserve the junctions in the resulting segmentation. Given a set of levels, we then assign one of the levels to each pixel. Though simply assigning the nearest level to each pixel may work in the noise-free case, in general we would also need some smoothing effect for the resulting segmentation to be useful. So we minimize a certain energy functional that balances the fitness to the data and the smoothness of the assignment map. The optimal solution is obtained by mapping the model to a maximum-flow problem in a directed graph, and solving it in a polynomial time.

Intuitively, we are proposing the use of the maximum-flow algorithm as a mechanism that group distinct regions of detected junctions into larger regions.

1.3. Related work

There are a number of recent work on application of network-flow algorithms to computer vision. For binary images, Greig, Porteous, and Seheult [10, 11] provided an efficient and optimal solution. Recent work [2, 7, 12, 18] extended this result to more than two levels in different ways. Since the problem is in general NP-hard, it is (at least) difficult to find an efficient exact solution to all of them. Approximate solution is one way: Boykov, Veksle, and Zabih [2] used an approximate multiway-cut algorithm to solve it approximately for a specific type of smoothing function, while Ferrari, Frigessi, and de Sá [7] used color coding. Limiting the applicable class of problems and exactly solving them is another way, including our approach: Roy and Cox [18] used maximum-flow algorithm for N-camera stereo correspondence problem, though they did not expressly mention the limitation. As a use of maximum-flow algorithm, our approach is most similar to [18]. Also, [12], which used directed graph maximum-flow for binocular symmetric stereo, used varied capacities (weights) for discontinuity penalties according to the context information, as [2] later but independently included in their method.

2. Formulation of the problem

We assume the input g to be an image corrupted by noise. We can typically raster scan an image and so g is represented by a vector in an N^2-dimensional vector space (for a square image of size $N \times N$.) Then, $g_k (k = 1, ..., N^2)$ represents the gray value at pixel k. Here we assume an 8-bit gray-scale image.

2.1. Junctions and levels

We segment an image by assigning to each pixel a *level* it belongs to. Each level has an associated gray value, and we wish to assign a level with the nearest gray value to each pixel, within a smoothness constraint.

We first determine the number of levels and their associated gray values. We take junctions (e.g., corners, T-junctions) as strong indicators of a region boundary. T-junctions and corners often arise from overlapping surfaces, which we wish to obtain as distinct segments in the image. Though sometimes they can arise from a mark on a surface, even in that case it is often appropriate to divide the mark from the rest as a distinct region. Let us assume that we have a junction detector (we have modified a junction detection model presented in Parida, Geiger and Hummel [17].) Each detected junction is a partition of a small disk in the image into K pie-shaped regions ($K = 2$ for corners, 3 for T- or Y-junctions, 4 for X-junctions, etc.), each with an assigned gray value. (See Figure 1 left.) The detector has several parameters we can use to filter junctions, including the contrast between partitions. We set the threshold for the contrast relatively high, so that only high-contrast junctions are detected.

Now, we want these junctions to survive our segmenting process, since we are supposing these junctions indicate region boundaries. Therefore, the set F of gray values should satisfy the following condition:

> *For each pair (e, e') of gray values assigned to neighboring regions in a junction, the nearest values in F to e and e' are always different.*

We look for the minimum set of threshold values that separates any two neighboring regions in detected junctions by gray values. Suppose a junction has four regions a, b, c, and d with gray values e_a, e_b, e_c, and e_d in this order, and that the relations $e_a < e_b$, $e_c < e_b$, $e_c < e_d$, and $e_d < e_a$ hold. We call (e_a, e_b), (e_c, e_b), (e_c, e_d), and (e_d, e_a) the intervals

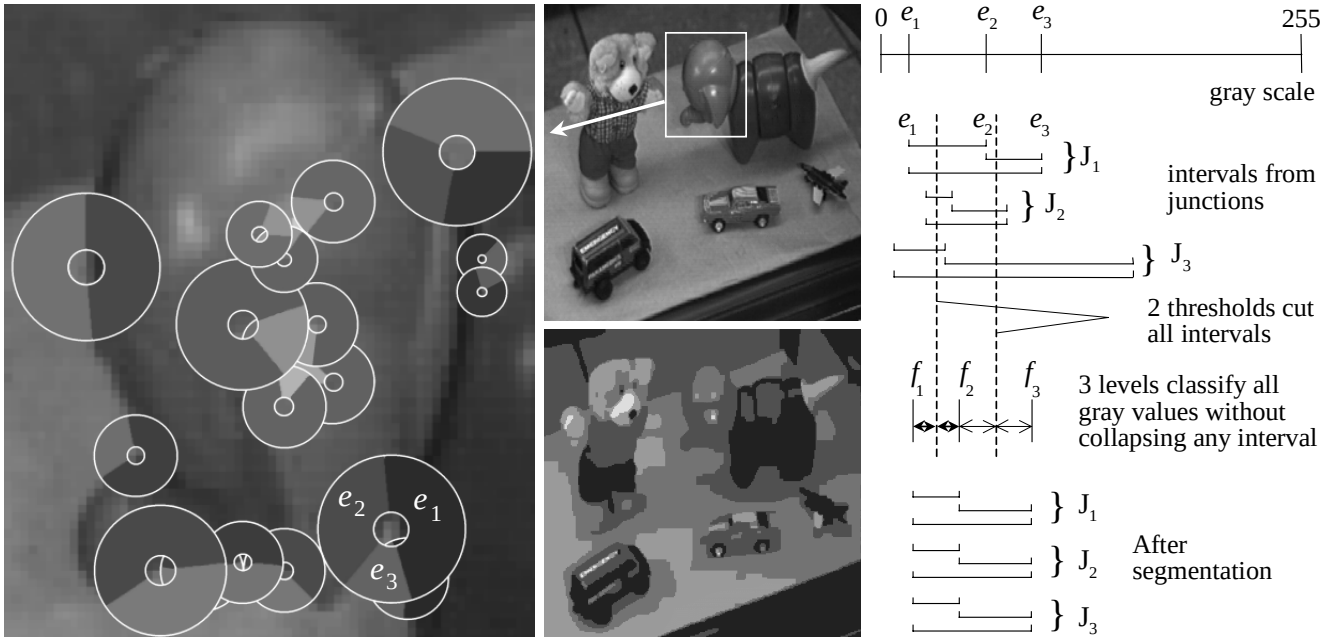

Figure 1. The segmentation process. First, junctions are detected, whose gray values are used to determine gray values for the levels. The map from pixels to levels are computed using the maximum-flow algorithm.

associated with the junction. Note that if we define thresholds so that each interval contains at least one threshold, all four gray values are divided by these thresholds. (See Figure 1 right.)

Let I be the set of all intervals that are associated with the detected junctions. We define T to be the smallest among the sets of gray values satisfying "for any $\iota \in I$, there exists $t \in T$ such that $t \in \iota$". Given such a set T, we define the set of gray values $F = \{f_1, f_2, \ldots, f_L\}$ for the levels to be the smallest set whose Voronoi diagram includes T. Then for each interval $(e, e') \in I$, the nearest values in F to e and e' are always different. We assume that the gray values are ordered in a natural manner:

$$f_a < f_{a+1}, \quad a = 1, \ldots, L-1. \tag{1}$$

2.2. The map

The next problem is to assign one of the possible levels to each pixel k, i.e., to find an assignment of a level $a(k) \in \{1, \ldots, L\}$ to each pixel k. We expect each grey value data g_k at pixel k to be close to the assigned grey value $f_{a(k)}$. This suggests a cost

$$\text{Error}(a,k) = \sum_{k=1}^{N^2} G(f_{a(k)} - g_k),$$

where $G(x)$ is some error measure for which a square function is usually employed. However, our approach can handle any function $G(x)$ as efficiently.

We also impose a smoothness constraint on the assignment function, where nearby pixels are encouraged to share the same level a. We consider the cost

$$\text{Smoothness}(a,k) = \sum_{k=1}^{N^2} \sum_{j \in N_k} F(a(k) - a(j)),$$

where N_k represents the neighborhood of pixel k. Typically, a four near neighbors is chosen, setting $N_k = (k-1, k+1, k-N, k+N)$. $F(x)$ is a smoothness function and we will show in Appendix that $F(x)$ must be a convex function for our method to guarantee an optimal solution with polynomial time in N. We point out that the smoothing penalty function does not depend on the specific grey values $f_{a(k)}$ but rather on the level assignments $a(k)$. It is important to stress that the penalty is given to different *assignments* rather than a change in gray value $f_{a(k)}$. It is still the case, for increasing monotonic $F(x)$, that the further away the levels the larger is the penalty, since the levels are ordered as in (1).

This smoothness function will encourage regions to grow (not to have too many small regions) and account for noise errors.

Our criteria for an optimal assignment $a(k)$ is the minimization of the following functional, the sum of both error and smoothness cost:

$$E(a(k)) = \sum_{k=1}^{N^2} \Big\{ G(f_{a(k)} - g_k) + \sum_{j \in N_k} F(a(k) - a(j)) \Big\}. \quad (2)$$

2.3. Brief comparison to MRF

The energy (2) looks similar to the one used in the Markov Random Field (MRF) model [1, 8, 9], where they search for a reconstruction function $f(k)$ that minimizes (MAP estimation)

$$E^G(f(k)) = \sum_{k=1}^{N^2} \Big\{ G(f(k) - g_k) + \sum_{j \in N_k} F(f(k) - f(j)) \Big\}. \quad (3)$$

Unfortunately, this optimization problem is in general NP-hard, and a problem with MRF is that the optimization is extremely slow. However, for a class of MRF problems in which the penalty function $F(x)$ is convex, our method can obtain a global optimum solution in a polynomial time. In fact, though the model (2) is not in general an MRF of type (3), it includes MRF model (3) as a special case where $F = \{f_1, f_2, \ldots, f_L\}$ is evenly distributed, and for convex $F(x)$, it can solve the problem efficiently. In addition, our method can solve a more general class of problems efficiently: Our model of discontinuities is based on the penalty for different levels $a \to b$ assigned to neighboring pixels, regardless of the value of $f_a - f_b$. Hence, our model (2) with convex $F(x)$ can have, viewed in terms of grey-value change, discontinuity penalty that is more like a step function, or multiple well.

3. Mapping the optimization problem to a maximum-flow algorithm

In this section we explain the segmentation assignment architecture that utilizes the maximum-flow algorithm to obtain the globally optimal assignment, with respect to the energy (2).

3.1. The directed graph

We devise a directed graph and let a cut represent an assignment function $k \mapsto a(k)$ so that the minimum cut corresponds to the optimal assignment. Let M be the set of all possible assignments, i.e.,

$$M = \{(a,k) \mid a \in [1 \ldots L], k \in [1 \ldots N^2]\}.$$

We define a directed graph $G = (V, E)$ as follows:

$$\begin{aligned} V &= \{u_{ak} \mid (a,k) \in M\} \cup \{s,t\} \\ E &= E_{\mathrm{D}} \cup E_{\mathrm{C}} \cup E_{\mathrm{P}}. \end{aligned}$$

In addition to the source s and the sink t, the graph has a vertex u_{ak} for each possible assignment $(a,k) \in M$. The set E of edges is divided into subsets E_{D}, E_{C}, and E_{P}, each one having a capacity with a precise meaning in terms of the model (2), which we will explain in the following subsections.

We denote a directed edge from vertex u to vertex v as (u,v). Each edge (u,v) has a nonnegative capacity $c(u,v)$. A *cut* of G is a partition of V into S and $T = V \setminus S$ such that $s \in S$ and $t \in T$ (see figure 2). We mean by a cut of an edge (u,v) that $u \in S$ and $v \in T$. This is the only case that the cost $c(u,v)$ of the edge contributes to the total cost $\sum_{u \in S, v \in T} c(u,v)$. We note that if the cut is through the edge (u,v) with $u \in T$ and $v \in S$ the cost is $c(v,u)$, which is in general different from $c(u,v)$. It is well known that by solving a maximum-flow problem one can obtain a *minimum cut*, a cut that minimizes the total cost over all cuts. (See [3] for details.)

Note in this formulation we use a directed graph in contrast to other application of network-flow algorithms to computer vision.

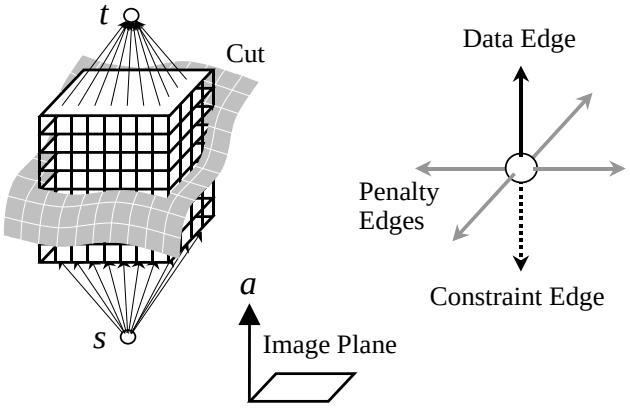

Figure 2. A directed graph. An edge is given by a tuple of graph nodes (u,v). A cut of the graph can be thought of as a surface that separates the two parts. The optimal cut is the one that minimizes the sum of the capacities associated to the cut edges. Each node has 6 outgoing edges (except for boundary).

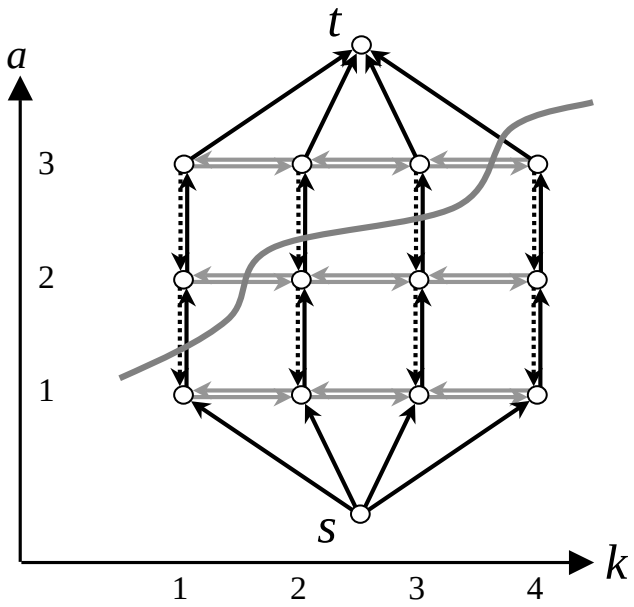

Figure 3. Data edges are depicted as black arrows. Four of them are cut here, representing an assignment a(1)=1, a(2)=2, a(3)=2, and a(4) = 3. Penalty edges are represented by gray arrows. By crossing consecutive penalty capacities, the cost is added linearly, accounting for the function F(x)=|x|. Constraint edges are depicted as dotted arrows. They ensure that the assignment a(k) is a function. These edges cannot be cut, preventing the cut from "going back".

Let us now analyze the different set of edges E_{D}, E_{P}, and E_{C}.

3.2. Data edges

From each vertex u_{ak}, there is one outgoing *data edge*: $(u_{ak}, u_{a(k+1)})$ if $k < L$, or (u_{ak}, t) otherwise. It has a capacity $\mathrm{Error}(a,k) = G(f_a - g_k)$. Thus, the capacities associated to these edges will contribute to account for the first term of (2). We denote the set of data edges by E_{D}. If a data edge originating from u_{ak} is cut, we interpret that this cut represents an assignment of level a to pixels k. Figure 3 shows the nodes and data edges. The cut shown represents a match $\{(a,k)\} = \{(1,1),(2,2),(3,2),(4,3)\}$. Also, edges (s, u_{1k}) are added for all k with infinite capacity. Note these edges are actually unnecessary and s and first layer vertices $\{u_{1k} | k = 1 \ldots N^2\}$ can be merged to one vertex, but for clarity are shown thus.

3.3. Penalty edges (discontinuity)

Penalty edges are defined as

$$E_{\mathrm{P}} = \{(u_{ak}, u_{aj}) \mid (a,k) \in M\,;\, j \in N_k\}$$

These edges are for paying for discontinuities (region boundaries). Edges in E_{P} are cut whenever a change in the level occurs. For instance, if level a is assigned to pixel k and level $a+2$ is assigned to pixel $k+1$, two edges will be cut, namely $(u_{a+1,k+1}, v_{a+1,k})$, and $(u_{a+2,k+1}, v_{a+2,k})$.

We set the capacity to be some constant value. By crossing consecutive penalty capacities the cost is added linearly, yielding a cost function $F(x) = |x|$. While we have used a simple connectivity and capacity setting here, we could seek more general connectivity of the form

$$E_{\mathrm{P}} = \{(u_{ak}, v_{bj}) \mid (a,k),\,(b,j) \in M\,;\, j \in N_k\}\,,$$

with arbitrary capacity.

By setting the capacity for these edges, we control the smoothness function $F(x)$ between levels. We prove in Appendix that for any general form of connectivity graph, where maximum-flow can be applied, the edge penalty must yield convex smoothing function $F(x)$. This follows from the requirement that the capacities must be non-negative. Conversely, it can be shown that any convex function $F(x)$ can be used as the smoothness function.

3.4. Constraint edges

Constraint edges are for enforcing that the assignment $a(k)$ is a function, i.e., that each pixel is assigned only one level.

$$E_{\mathrm{C}} = \{(u_{ak}, u_{a-1,k}) \mid (a,k) \in M\,, a > 1\}.$$

The capacity of each constraint edge is set to infinity. Therefore, any cut with a finite total flow cannot cut these edges. Note that, because the edges have directions, a constraint edge prevents only one of two ways to cut them. In Figure 3, constraint edges are depicted as dashed arrows, and none is cut.

4. Implementation and Results

We implemented the architecture explained in the last section. For maximum-flow algorithm we used standard push-relabel method with global relabeling [5].

Our system successfully segmented various images into only a few levels. Figure 4 shows the segmentation of the image **Gear** into two, three, and four gray values. Figure 5 shows the segmentation of the image **Squirrel** into three gray values.

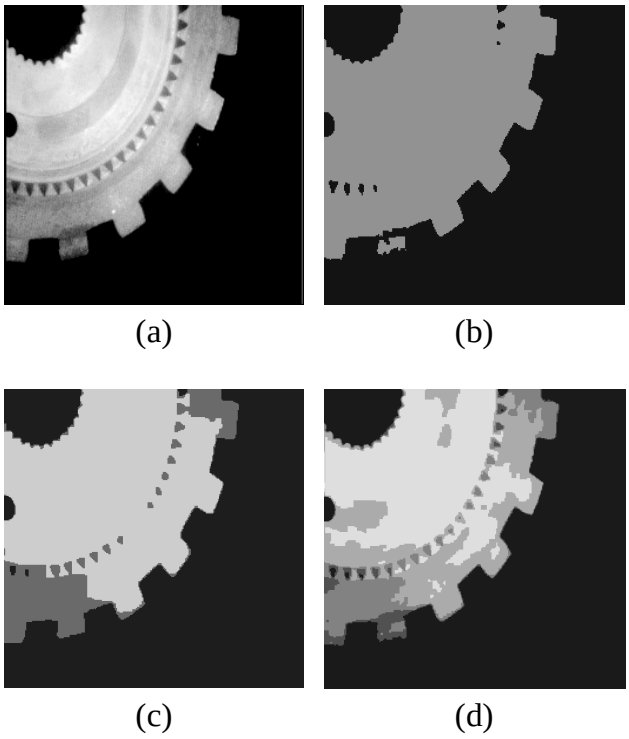

Figure 4. Segmentations of the image Gear (a) into two (b), three (c), and four (d) gray values.

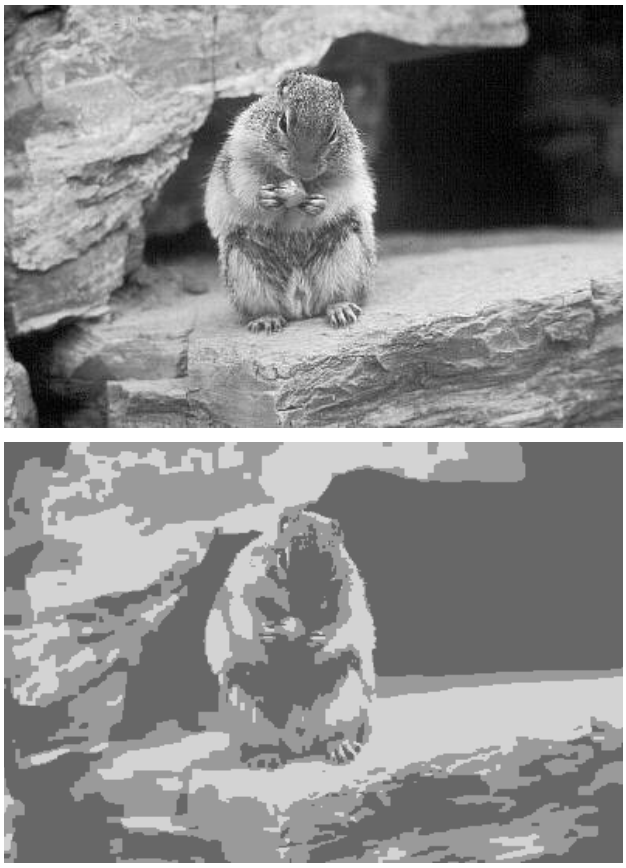

Figure 5. Segmentation of the image Squirrel.

Appendix: Maximum-flow optimization and convex prior models

In this appendix we prove that this use of the maximum-flow algorithm can only be applied to optimize Markov models of first order (neighborhood of size one) with prior models whose discontinuity penalties are convex. While this is a severe restriction on the class of functions $F(x)$, it is still of great use, especially because we are applying $F(x)$ to the assignment function $a(k)$ and not to the values $f_{a(k)}$, i.e., although $F(a)$ is convex on a, it is not always the case that $F(\tilde{a}(f))$ is convex as a function of f, where we denote by $\tilde{a}(f)$ the corresponding level a that is associated with a gray value f.

$F(x)$ must be convex

We briefly outline a proof that $F(x)$ has to be convex.

Each node u_{ak} has, in general, neighbors to all nodes in column $k-1$, i.e, has in general non-zero capacity at

$$\begin{aligned} c(u_{ak},u_{bj}) &\neq 0, \quad \text{where } j \in N_k, \quad \text{and} \\ c(u_{ak},u_{bj}) &= 0, \quad \text{otherwise} \end{aligned}$$

The smoothing function $F(x)$, at a pixel k, is the result of a cut through various edges bridging nodes u_{ak} and u_{bj}, where $\{j \in N_k = (k-1, k+1, k-N, k+N)\}$. Let us focus on the bridge between pixel k and $k-1$. In the directed graph, this cut goes through a column of hypothesis separating these two pixels k and $k-1$. Say hypothesis at k is $a(k)$ while at $k-1$ is $a(k-1)$. The smoothing cost depends only on $|a(k)-a(k-1)|$, and this is the sum of all capacities that are cut by the jump column. Thus, the capacities between any two nodes should depend only on the difference between the two node hypothesis, hence we assume $c(u_{a(k),k}, u_{a(k-1),k-1}) = c(a(k), a(k-1))$.

Given the assignment change from a at pixel $k-1$ to b at pixel k, we have the smoothness function to be the sum of all the capacities being cut. More precisely,

$$F(a-b) = \sum_{a'=1}^{a} \sum_{b'=b+1}^{L} c(a',b') + \sum_{a'=a+1}^{L} \sum_{b'=1}^{b} c(a',b')$$

Examining the second derivative, it follows

$$\frac{\partial^2 F(a-b)}{\partial a^2} = 2c(a,b) \geq 0\,,$$

since all the capacities are non-negative. Therefore, the discontinuity penalty cost $F(x)$ must be a convex function, i.e., second derivative is always non-negative. This is a limitation of the use of the maximum-flow algorithm. While there is this limitation, our experiments do yield good solution with the linear cost $F(x)$.

References

[1] A. Blake and A. Zisserman. *Visual reconstruction*. MIT Press, Cambridge, Mass, 1987.

[2] Y. Boykov, O. Veksle, and R. Zabih. Markov random fields with efficient approximations. In *Proc. IEEE Conf. CVPR*, Santa Barbara, California, 1998.

[3] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms.* McGrow-Hill, New York, 1990.

[4] J. F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[5] B. V. Cherkassky and A. V. Goldberg. On implementing push-relabel method for the maximum flow problem. In *Proc. 4th Int. Programming and Combinatorial Optimization Conf.*, pages 157-171, 1995.

[6] T. Darrell and A. Pentland. Robust estimation of a multilayered motion representation. In *Proc. IEEE Workshop on Visual Motion*, Princeton, NJ, 1991.

[7] P. A. Ferrari, A. Frigessi, and P. de Sá. Fast approximate maximum a posteriori restoration of multicolour images. *J. R. Statist. Soc.* B, 57, 485-500, 1995.

[8] D. Geiger and F. Girosi. Parallel and deterministic algorithms for MRFs: surface reconstruction, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13:401–412, 1991.

[9] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6:721–741, 1984.

[10] D. M. Greig, B. T. Porteous, and A. H. Seheult. Discussion of: On the statistical analysis of dirty pictures (by J. E. Besag.) *J. R. Statist. Soc.* B, 48, 282-284, 1986.

[11] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *J. R. Statist. Soc.* B, 51, 271-279, 1989.

[12] H. Ishikawa and D. Geiger. Occlusions, discontinuities, and epipolar lines in stereo. In *Fifth European Conference on Computer Vision*, Freiburg, Germany, 1998. Springer-Verlag.

[13] A. K. Jain and R. C. Dubes. *Algorithms for clustering data.* Prentice Hall, 1988.

[14] G. Kanizsa. *Organization in vision*. Praeger, New York, 1979.

[15] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *Inter. Journ. of Comp. Vis.* 3:73–102, 1989.

[16] D. Mumford and J. Shah. Boundary detection by minimizing functionals, I. In *Proc. IEEE Conf. CVPR*, San Francisco, CA, 1985.

[17] L. Parida, D. Geiger and R. Hummel. Kona: a multijunction detector and classifier. *, Int. Conf, on Energy Minim.. in Comp. Vis. and Pat. Recog.*, Venice, 1997.

[18] S. Roy and I. Cox. A maximum-flow formulation of the N-camera stereo correspondence problem. *ICCV98*, Bombai, India 1998.

[19] S. Sarkar and K. L. Boyer. Quantitative measures of change based on feature organization: eigenvalues and eigenvectors. In *Proc. IEEE Conf. CVPR*, San Francisco, 1996.

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proc. IEEE Conf. CVPR*, Puerto Rico, 1997.

[21] Y. Weiss. Smoothness in layers: motion segmentation using nonparametric mixture estimation. In *Proc. IEEE Conf. CVPR*, Puerto Rico, 1997.