MVA2007 IAPR Conference on Machine Vision Applications, May 16-18, 2007, Tokyo, JAPAN

FAST GRAPH SEGMENTATION BASED ON STATISTICAL AGGREGATION PHENOMENA

Frank Nielsen

Sony Computer Science Laboratories, Inc.
Tokyo 141-0022, Japan
Frank.Nielsen@acm.org

Richard Nock

CEREGMIA/Univ. Antilles-Guyane
97275 Schoelcher, Martinique, France
Richard.Nock@martinique.univ-ag.fr

ABSTRACT

In this paper, we first generalize a recent statistical color image segmentation algorithm [5] to arbitrary graphs, and report its performance for 2D images, 3D meshes and volume data. We then describe a fast pre-segmentation to the main graph procedure that allows us to further speed-up the segmentation by a factor of $\times 2$ to $\times 4$, without decreasing significantly the quality of segmentations. As an application, we built a real-time video segmentation that is robust enough to be used in camera-driven user interfaces and robotics applications.

1. INTRODUCTION

Segmentation is the fundamental task of grouping raw data into a small number of perceptual units: the so-called *segments*. Although image segmentation has been studied from the very beginning of computer vision [1] in the 1970s, it is still nowadays an unsolved and yet hot topic. Since its inception, segmentation has been tackled for various types of data: color images, videos, range images (as known as depth images), 3D volume data (such as CTs or MRIs), 3D meshes, etc. In Section 2, we present a generalization of a statistical color image region merging framework [5] to arbitary graphs. Graphs are convenient universal structures to manipulate, that encode both data and their topologies. Section 3 describes how to design concentration inequalities for the generic graph region merging algorithm, and report on segmentation results for various data types: 2D color images (Section 3.1), 3D meshes(Section 3.2), and gray volume data (Section 3.3). Segmenting frame-by-frame video requires a lot of processing power. We present in Section 4 a video-rate segmentation algorithm that proceeds by pre-segmenting efficiently images in order to build a reduced graph on which the generic segmentation is then applied. Finally, Section 5 concludes our paper.

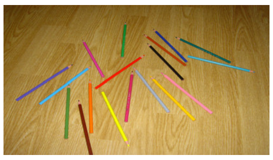
2. GRAPH SEGMENTATION BY REGION GROWING

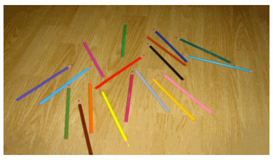
Region growing is one of the oldest *area-based* image segmentation algorithm [1]. A region growing process starts by first initializing for each pixel a corresponding single-pixel region, and then merge stepwise similar adjacent regions until no more fusion is attainable. Thus, at a given step, region growing considers implicitly the underlying region adjacency graph (RAG), and based on a *merging predicate*, decide to merge or not the current pair of adjacent regions. Usually, the RAG is dynamically updated whenever merging occurs. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an arbitrary graph of $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ edges. Let $E.a$ and $E.b$ denote the two nodes defining the edge extremities: $E = (E.a, E.b)$. To each node V of $\mathcal{V}$ (defining a region), we further associate a unique ID denoted by $\mathrm{RegID}(V)$, and a mean feature vector $\bar{\mathbf{f}}(V)$ (say, mean color channels for pictures or mean area/normal for triangle mesh, etc), and its region size $n(V)$. We further weight each edge $E \in \mathcal{E}$ using a weighting function $w(\cdot)$. Let $\mathcal{E}_r = \{E \in \mathcal{E} | w(E) \leq r\}$ denote the set of edges with weights less than or equal to r. Our graph region growing algorithm is a generalization of the color image segmentation of [5] that first proceeds by sorting edges in increasing order of their weights, and then inspects iteratively each edge $E = (E.a, E.b)$ to determine whether the two associated regions (hooked up by their IDs) should be merged or not. The algorithm is fast because it does not maintain dynamically the RAG and yet deliver provably good segmentations under some statistical model [5] (see Section 3). We summarize the graph region growing process as follows:

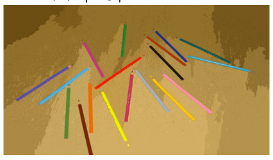
```
GraphSegmentationByRegionGrowing($\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$)
1.  ◁ Sort in increasing order edges according to their weights ▷
2.  $\mathcal{E}' \longleftarrow$ Sort($\mathcal{E}, w$)
3.  for $i \leftarrow 1$ to $|\mathcal{E}'|$
4.      do
5.          if RegID($E'_i.a$) $\neq$ RegID($E'_i.b$)
6.             then if Predicate(RegID($E'_i.a$), RegID($E'_i.a$))
7.                     then ◁ Update $\bar{\mathbf{f}}(.)$ and $n(.)$ accordingly ▷
8.                          Merge(RegID($E'_i.a$), RegID($E'_i.b$))
```

Merging and updating region IDs is done efficiently using the disjoint set union-find data structure of Tarjan [7]. Although one call to Merge can potentially requires logarithmic time to update region IDs, a sophisticated amortized analysis proves that it merely costs almost constant time [7]. Therefore the graph region growing procedure is quasi-linear.[1]

[1] Linear for all practical cases. Slightly supra-linear in theory [5].


(a) $|\mathcal{E}_0| = 14802$


(b) $|\mathcal{E}_5| = 1894775$


(c) $|\mathcal{E}_{10}| = 3186900$

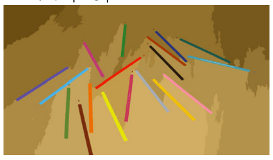

(d) $|\mathcal{E}_{148}| = 4144163$

Fig. 1. Intermediate segmentation results of a high-definition 1920x1080 color toy image `pencils` after processing all edges $\mathcal{E}_r$, for $r \in \{0, 5, 10, 148\}$.

The quality of segmentation clearly depends on both the merging predicate and the edge order. In [5], a predicate based on statistical aggregation phenomena[2] was given for color images and it was further shown that under some particular edge order, the resulting segmentation was close to the "optimal" segmentation with high-probalibity. In the next Section, we show how to build initial graphs, give predicate functions based on concentration inequalities, and report on segmentation results for 2D images, 3D meshes, and volume data, respectively.

3. CONCENTRATION INEQUALITIES AND SEGMENTATION RESULTS

3.1. Segmenting Color Images

Let I be a RGB color image of width w and height h. To each pixel (i, j), we associate a node $V_{i,j}$ with unique ID $\mathrm{RegID}(V_{i,j}) = i \times w + j$. We consider the 4-connexity[3] model of images (C_4) so that besides pixels on the border, all interior pixels have exactly four neighbours that define accordingly edges. Edges are weighted according to the maximum color channel difference:

$$w(E = (V_{i,j}, V_{i',j'})) = \max_{c \in \{R,G,B\}}(|I_c(i,j) - I_c(i',j')|).$$

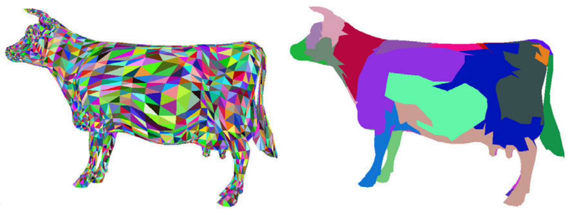
The total number of edges is $m = 2wh - w - h$. The merging predicate based on concentration inequalities is defined as:

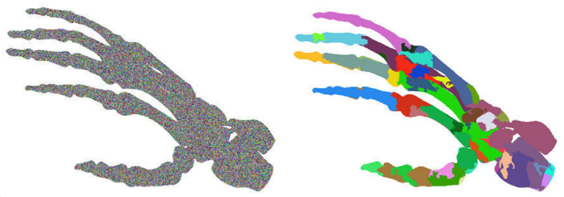
$$\max_{c \in \{R,G,B\}} \left(\bar{I}_c(i,j) - \bar{I}_c(i',j')\right)^2 \leq b(n_{i,j}) + b(n_{i',j'}) \quad (1)$$

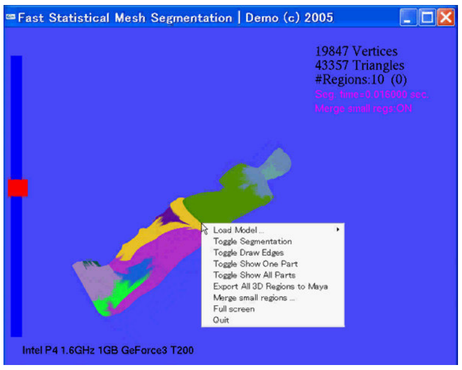
where $b(x) = \frac{256^2}{2Qx}(\min(256, x) \log x + 2 \log 6wh)$ [5], and $n_{i,j}$ ($n_{i',j'}$) denote the number of pixels in the region containing pixel (i, j) ((i', j'), respectively). Parameter Q ($Q = 32$ by default) controls the granularity of segmented regions. Figure 1 depicts the intermediate segmentations as the algorithm proceeds through the sorted sequence of edges. Observe in this case that 75% edges have weights below 10 and that picture of Fig. 1(c) is very close to the final segmentation shown in Fig. 1(d).

[2]For example, the sum of independent uniform random variables yields a Gaussian random variable (well-known central limit theorem). More generally, statistical aggregation phenomena have been recently found for random variables satisfying loose distribution assumptions [4].

[3]Setting the connexity of pixels to C_8 doubles the number of edges without improving much the segmentation.


(a)


(b)


Fast Statistical Mesh Segmentation | Demo (c) 2005

19847 Vertices
43357 Triangles
#Regions:10 (0)
Seg. time=0.016000 sec.
Merge small regs:ON

Intel P4 1.6GHz 1GB GeForce3 T200

(c)

Fig. 2. Segmentation results for 3D meshes: (a) `cow` and (c) `hand` models (5804 and 654666 triangles, respectively). (c) is a snapshot of the mesh segmentation software where parameter Q can be tuned on-the-fly using the slidebar.

3.2. Segmenting 3D Meshes

Segmenting meshes is an important ingredient of geometry processing. Applications of mesh segmentation include parameterization and texture mapping (texture atlas), morphing, multi-resolution modelling, editing, compression, animation and shape matching. Prior work [6] of mesh segmentation algorithms are either based on region growing, or hierarchical/iterative/spectral clusterings, often requiring quadratic time (and thus limiting meshes to a few thousand triangles). Thus, those methods first require to simplify and remesh appropriately original data (introducing some kind of distortion and somehow presegmenting the raw mesh) to then segment new small-size meshes using matrix decomposition techniques. Our segmentation is fast enough to tackle directly the original mesh input. Our graph segmentation algorithm partitions 3D meshes at rate of a million triangles per second on commodity PCs, and hence is suitable for interactive applications. Consider any triangular mesh with t triangles $T_1, ..., T_t$. They are several ways[4] to define input graphs depending the relationships of triange/vertex/edge. We choose to associate for each triangle T a node $V \in \mathcal{G}$. Then, we define a (graph) edge $E_{k,l}$ if and only if the corresponding triangles T_k and T_l share a common edge, and set the weight $w(E_{k,l})$ as the area difference of adjacent triangles: $w(E_{k,l}) = |\mathrm{area}(T_k) - \mathrm{area}(T_l)|$. The merging predicate which exploits the same concentration inequality as [5] return true if and only if:

$$(\bar{V_k} - \bar{V_l})^2 \leq \frac{(n_k \log n_k + n_l \log n_l)}{Q}(\frac{1}{n_k} + \frac{1}{n_l}), \qquad (2)$$

where $\bar{V_i}$ and n_i denote respectively the mean triangle area and the number of triangles in the region containing node V_i, for $i \in \{k, l\}$. Q is a scalar parameter[5] controlling the coarseness of segmentation (useful for multi-scale segmentation or oversegmentation settings). Figure 2 reports the segmentation obtained at interactive rate by adjusting parameter Q. Triangle area statistics are surprisingly good enough[6] to obtain nice segmentations, because most meshes are either scanned regularly or appropriately isotropically remeshed. That is, meshes seem to often encode into triangle areas the implicit surface characteristics. Note that as a preprocessing step, source meshes can be remeshed to meet that condition.

3.3. Volume Images

3D images are dense sets of voxels stored into image stacks (slices). Most of the 3D voxel data are acquired in gray intensities. We use the 6-connexity (C_6) of voxels to define the node/edge graph and choose the merging predicate similarly to Section 3.2. Result of the segmentation is displayed in Figure 3 for two different values of multi-scale parameter Q.

4. VIDEO-RATE IMAGE SEGMENTATION

To further improve the running time of the color image segmentation of Section 3.1 to reach video-rate, one may first consider downsizing the source image. However downsizing images degrades significantly the segmentation quality. A better approach consists in reducing the number of graph nodes and edges by performing a fast block pre-segmentation: We split the image into regular blocks of size $s \times s$ and compute for each block its color extrema for each channel. If the max/min difference is below a given threshold for each color channel, we label the block as "uniform". Otherwise, the

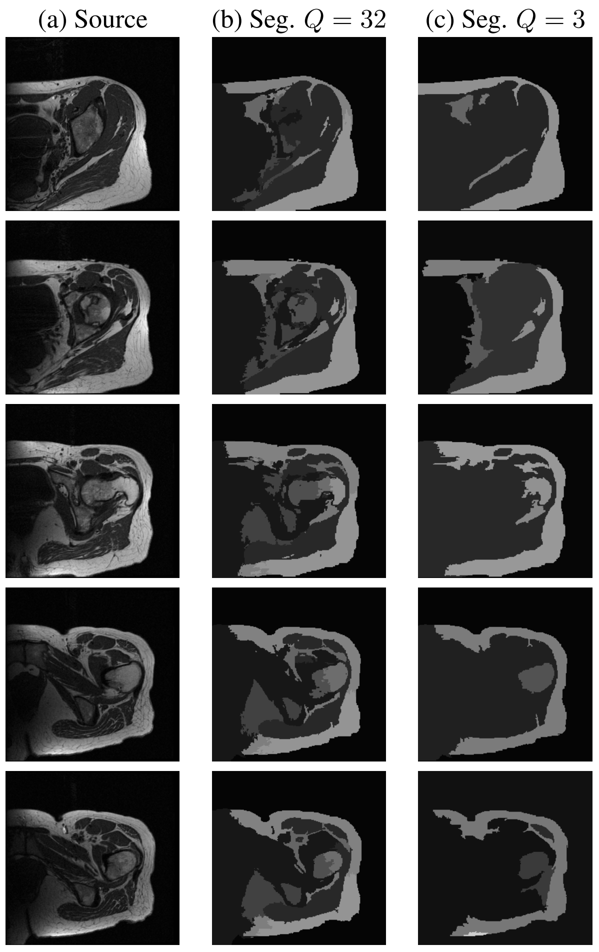


Fig. 3. Result of the segmentation of a grayscale MRI volume data ($256 \times 256 \times 34$ voxels). Column (a) shows source images, (b) result of a typical segmentation, and (c) result of an oversegmentation. Segmentation took less than one second.

block is said "nonuniform." We then consider adjacent pairs of blocks when building the graph edges. For uniform/uniform pairs of blocks, we create only two nodes and a single edge linking the corresponding presegmented regions (squares of s^2 pixels). For uniform/nonuniform blocks, we create interedges as before but there is no intra-edge in the uniform block (modeled by a single node). Although this scheme can be straightforwardly extended in a hierarchical way, we found it ran better on a single level by choosing blocks of $4 \times 4 = 16$ pixels (we usually remove 85% of edges. The table below gives some statistics for image `pencils` (resized to 720×480) of Fig. 1:

Block size	#uniform/#total	#uniform%	seg. time (msec)
24×24	$331/600$	55%	20
16×16	$954/1350$	70%	16
4×4	$20251/21600$	93%	$14,7$
2×2	$85444/86400$	98%	$20,2$

[4]Namely, three ways (graph node/graph edge): triangle/edge, triangle/vertex and edge/vertex.

[5]Although Q has the same meaning as in Eq. 1, its value is different.

[6]We also tried triangle normals but found out that area worked best.

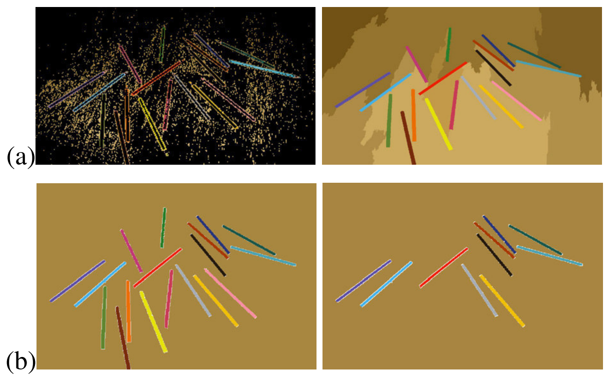


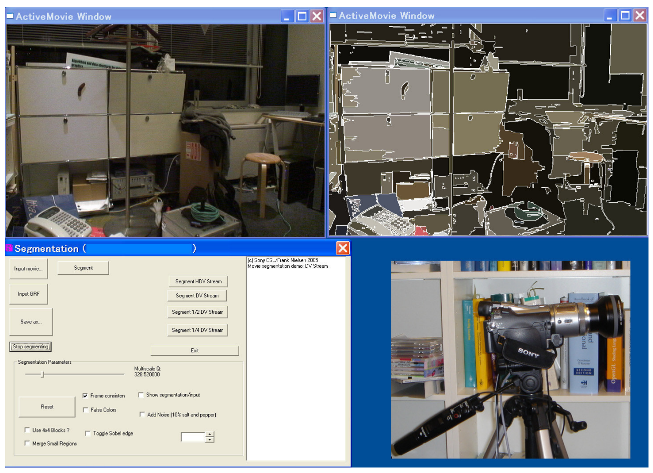
Fig. 4. (a) shows the picture with uniform blocks colored in black and the result of the segmentation. (b) shows the segmentation with all edges with weights less than 10 and 20 automatically merged, respectively. Observe that in the rightmost picture some pencils disappeared (overmerged with the background region).

Observe that although there are more 2×2 blocks than 4×4 blocks, it takes more time to label then as uniform/nonuniform. This explains our choice for 4×4 blocks. In practice, we observe a $\times 2$ to $\times 4$ speed-up factor in segmentation.

To further improve speed, we decide to automatically merge all edges whose weights are below some prescribed threshold. Doing so, we avoid to compute PREDICATE for pairs of regions that are anyway likely to merge. Figure 4 displays the benefit of bypassing the predicate evaluation for those edges. Observe that the background of picture `pencil` is segmented into a single region. To avoid annoying flickering effects when segmenting frame-by-frame videos, we only resort graph edges every p frames ($p = 30$ in our setting). The video segmentation implementation uses the generic graph library with DirectShow® (Figure 5). The system runs at 30 fps for half DV size (320x240) on an Intel Pentium IV 3.6 GHz equipped with 1 GB RAM. No processor-level SIMD optimization has been done. Our algorithm has been successfully used for more elaborate vision tasks in mobile robots. Indeed, real-time video segmentation is a crucial component for RoboCup challengers[7]. See [2, 3] for some up-to-date reports on current embarked segmentations used by soccer robots.

5. CONCLUDING REMARKS

We presented a fast linear-time and versatile graph segmentation algorithm based on the region growing paradigm that does not require to maintain dynamically the region adjacency graph (RAG). The quality of our segmentations is achieved by using tailored merging predicates that rely on concentration inequalities exploiting statistical aggregation phenomena [4].



(c) Sony CSL/Frank Nielsen 2005
Movie segmentation demo: DV Stream

Segmentation Parameters
Multiscale Q:
328.520000
[✓] Frame consisten
☐ Show segmentation/input
☐ False Colors
☐ Add Noise (10% salt and pepper)
☐ Use 4x4 Blocks ?
☐ Toggle Sobel edge
☐ Merge Small Regions

Fig. 5. Screen snapshot of our real-time live camcorder video segmentation (Sony HDR-HC1, MPEG-2 transport stream at 25 Mbps).

The full segmentation library code is a mere few hundred lines of C++ making it ideal for embedded systems. Segmentation is fast enough for handling 1M triangle meshes at interactive rate or segmenting VGA live video feeds. We will perform live VGA video segmentation of the audience during the presentation.

6. REFERENCES

[1] A.Rosenfeld. *Picture processing by computer*. Academic Press, 1969.

[2] A. de Cabrol, P. Bonnin, M. Silly-Chetto, V. Hugel, and P. Blazevic. Clear box evaluation of vision algorithms: Application to the design of a new color region growing segmentation for robotics. In *8th International Symposium on Signal Processing and its Applications (ISSPA)*. IEEE, 2005.

[3] C. Gönner, M. Rous, and K.-F. Kraiss. Real-time adaptive colour segmentation for the RoboCup middle size league. In *RoboCup 2004: Robot Soccer World Cup VIII*, volume 3276 of *Lecture Notes in Computer Science*, pages 402–409. Springer, 2004.

[4] C. McDiarmid. Concentration. In M. Habib, C. McDiarmid, J. ramirez Alfonsen, and B. Reed, editors, *Probabilistic mMethods for Algorithmic Discrete Mathematics*, pages 195–248. 1998.

[5] R. Nock and F. Nielsen. Semi-supervised statistical region refinement for color image segmentation. *Pattern Recognition*, 38(6):835–846, 2005.

[6] A. Shamir. A formulation of boundary mesh segmentation. In *Int. Sympos. 3D Data Processing, Visualization and Transmission (3DPVT)*, pages 82–89, 2004.

[7] R. E. Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *J. Comput. Syst. Sci.*, 18(2):110–127, 1979.

[7] `http://www.robocup.org`