Efficient segmentation trees on the GPU

Yaroslav Ganin, NVIDIA
yganin@nvidia.com

Problem statement

Segmentation tree is a sequence of nested partitions of an image. Such structure provides additional information (comparing to the plain segmentation) about interrelations between different parts of the image. There are numerous computer vision tasks which demand a high performance algorithm for segmentation trees building. Unfortunately, current state-of-the-art methods aimed for the CPU are way too slow.

Algorithm

The proposed algorithm is a GPU-powered version of the algorithm developed by **Haxhimusa et al.** and described in [3]. The original method is based on the **Felzenszwalb's** [2] idea of using *Minimum Spanning Trees* for image partitioning. Among other things Haxhimusa's work contains two key observations:

- **Kruskal's** algorithm can be replaced by **Boruvka's** MST construction approach;
- Felzenszwalb's *"contrast step"* can be omitted [4].

These observations allowed us to speed up the algorithm using the GPU.

Our implementation is heavily based on the MST construction method proposed by **Vineet et al.** [5]. The heart of the algorithm (see Alg. 1 for pseudocode) is **reduceGraph** routine. It is almost entirely composed of basic array operations (like **scan**, **sort**, etc.) which are already implemented efficiently in several dedicated libraries (e.g., **thrust**). This gave us a significant performance boost virtually for free (no need to worry about suboptimality of own kernels).

Alg. 1 Segmentation tree construction

Input: input image I, segmentation tree T

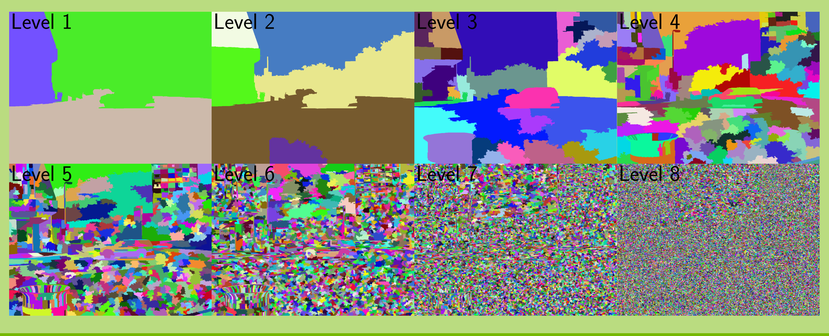
1: $G \leftarrow$ **buildNetGraph**(I)
2: **while** $|G| \neq 1$ **do**
3: $\{G', reductionMap\} \leftarrow$ **reduceGraph**(G)
4: **addNewLevel**(T, $reductionMap$)
5: $G \leftarrow G'$
6: **end while**

Output: T

Full source code will be available in future releases of Cuda Computing SDK.

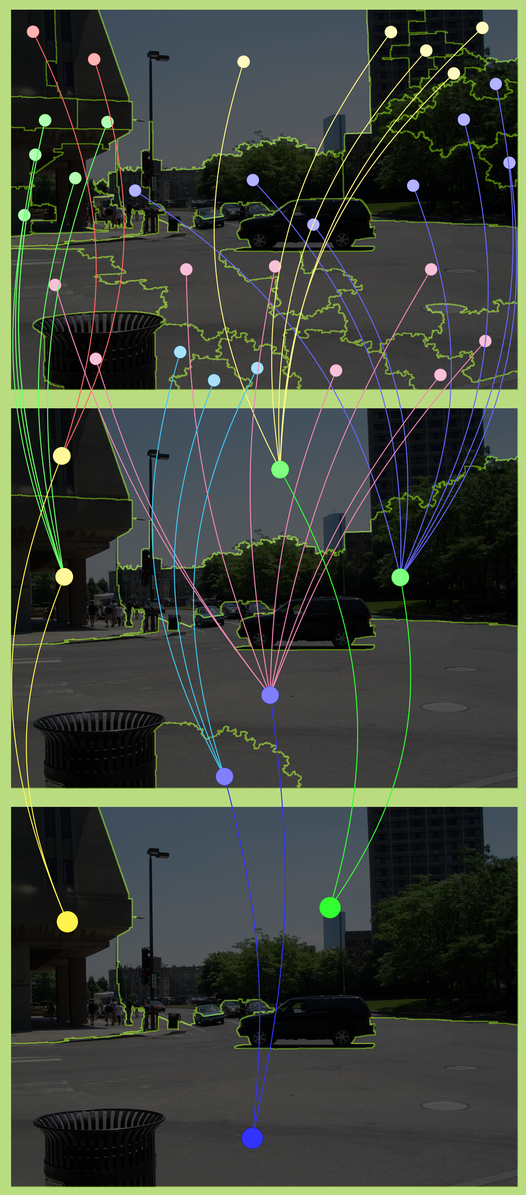
Illustrations

Here you can see several levels of the segmentation tree produced by our software (input image is taken from [1]). Different colors correspond to different superpixels.



More illustrations

Here is how segmentation tree spans its levels (nested segementations).



Future work

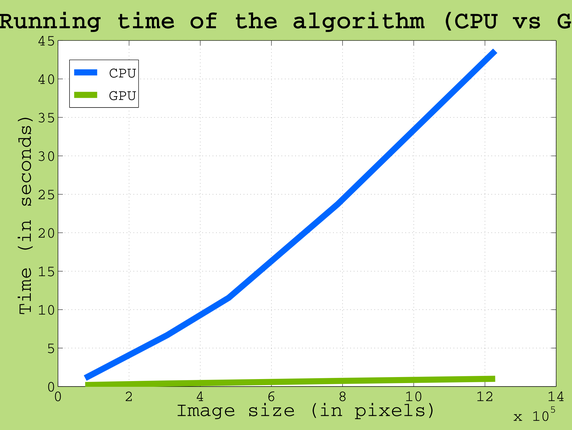
- Try to include omitted "contrast heuristics";
- Try new feature spaces;
- Incorporate machine learning in order to capture semantics of images.

Performance evaluation

Our test platform is:

- 4 GHz Intel Core i7 quad-core CPU;
- Ubuntu Natty 11.10 OS;
- NVIDIA GeForce GTX 260 with 895 MBytes of video memory;
- CUDA 4.1.

We compared our GPU version against the reference implementation [3]. A set of images on different scales ($320 \times 240, 640 \times 480, 800 \times 600, 1024 \times 768$ and 1280×960) was used as the testing sample. The results are shown below on the graph. It is clearly seen that the CPU implementation becomes unusable for online tasks as the size of an input image increases. But it is not the case for the GPU version. The running times remain reasonably low even for quite large images thus making the algorithm suitable for real-time applications. In fact, the proposed implementation is just slightly slower (about 2 times) than Felzenszwalb's software for generating a *single* segmentation.



References

[1] S.M. Bileschi.
StreetScenes: Towards scene understanding in still images.
PhD thesis, Massachusetts Institute of Technology, 2006.

[2] P.F. Felzenszwalb and D.P. Huttenlocher.
Efficient graph-based image segmentation.
International Journal of Computer Vision, 59(2):167–181, 2004.

[3] Y. Haxhimusa and W. Kropatsch.
Segmentation graph hierarchies.
Structural, Syntactic, and Statistical Pattern Recognition, pages 343–351, 2004.

[4] A. Ion, W. Kropatsch, and Y. Haxhimusa.
Considerations regarding the minimum spanning tree pyramid segmentation method.
Structural, Syntactic, and Statistical Pattern Recognition, pages 182–190, 2006.

[5] Vibhav Vineet, Pawan Harish, Suryakant Patidar, and P. J. Narayanan.
Fast minimum spanning tree for large graphs on the gpu.
In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 167–171, New York, NY, USA, 2009. ACM.