Graph Cut

Second Milestone

Bruno Coswig Fiss Max Göttner

[1] Institut für Technische Informatik und Mikroelektronik
Technische Universität Berlin

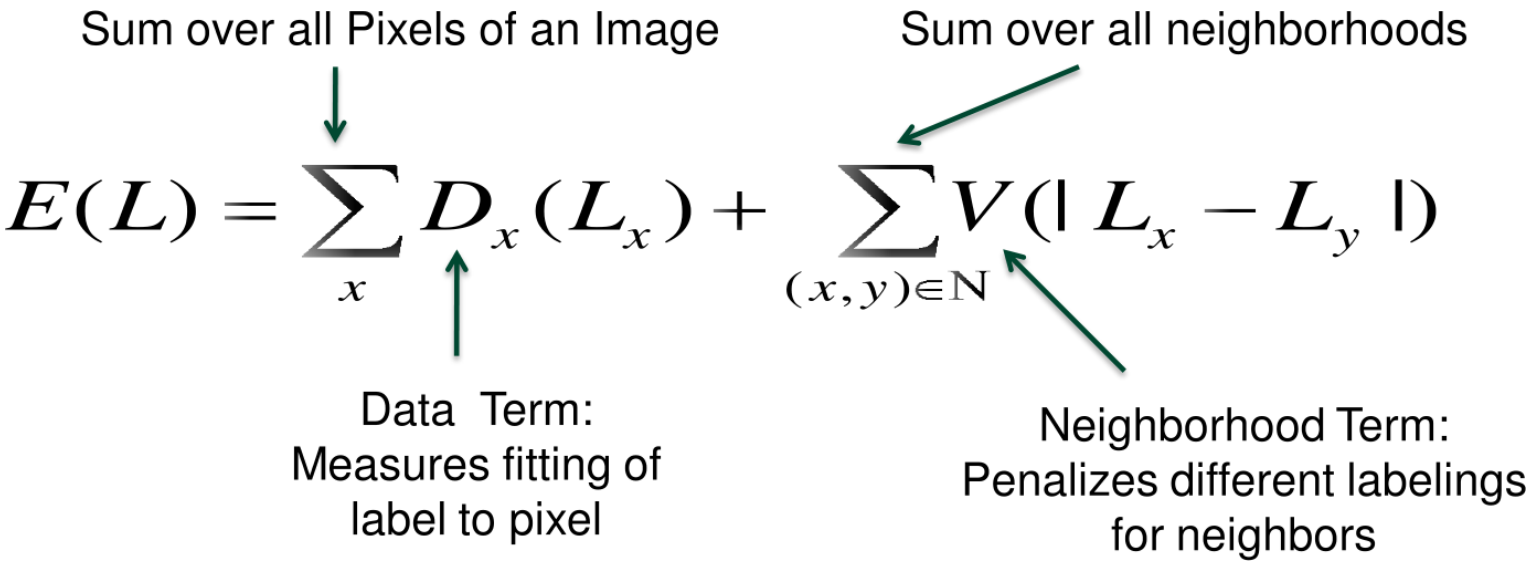
Parallele Algorithmen auf GPUs

Outline

Outline

Using Graph Cut for Energy Minimization

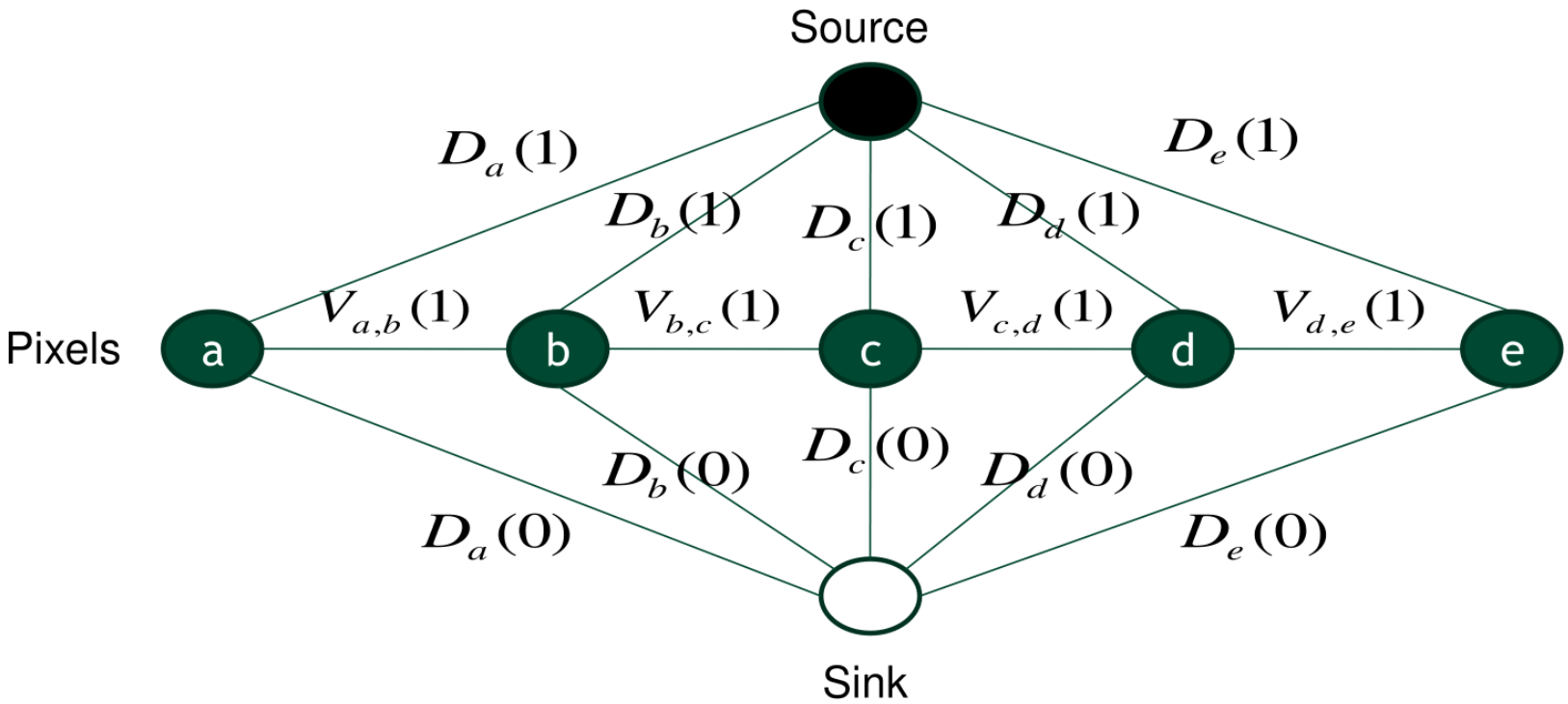
Minimize:



Credit: *Timo Stich, 2009*

Using Graph Cut for Energy Minimization

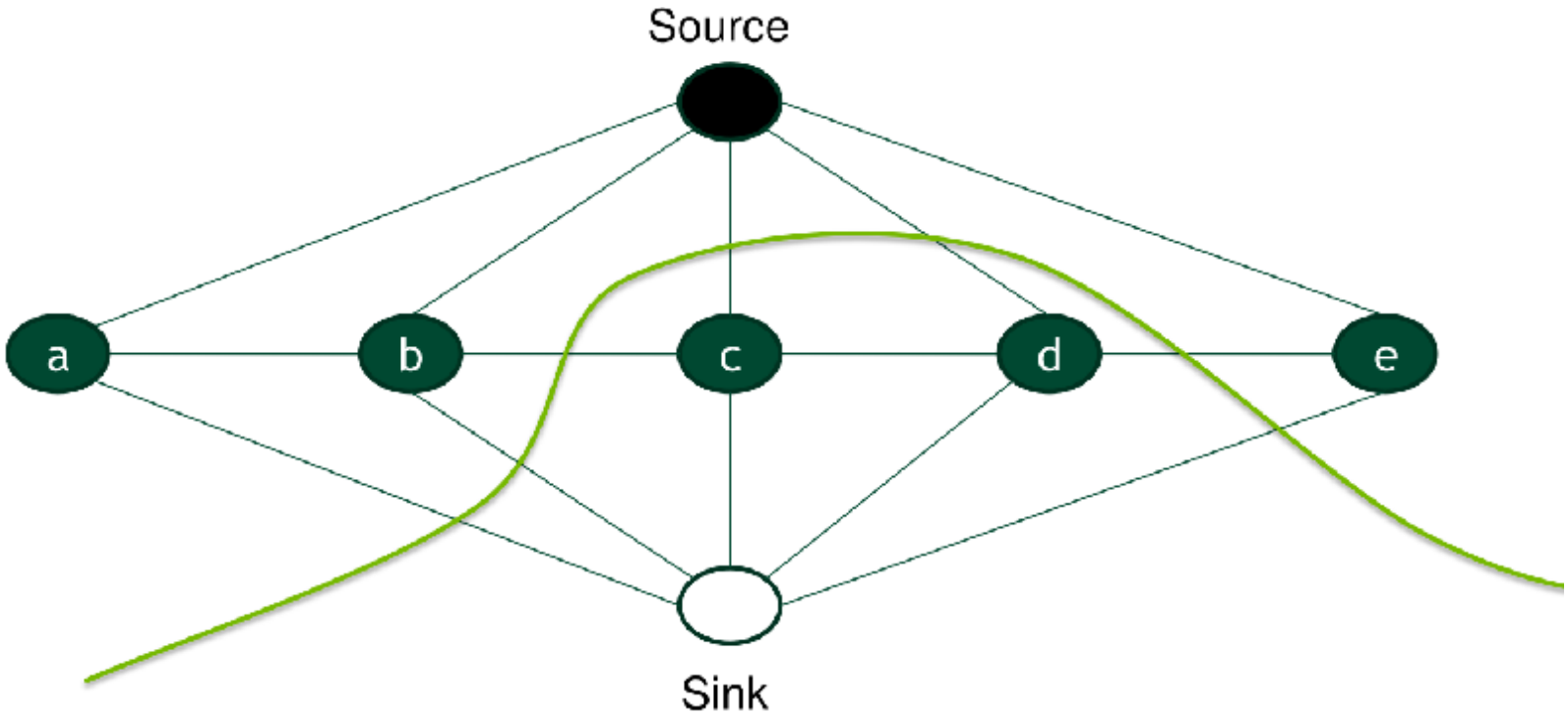
Find minimum cut:



Credit: *Timo Stich, 2009*

Using Graph Cut for Energy Minimization

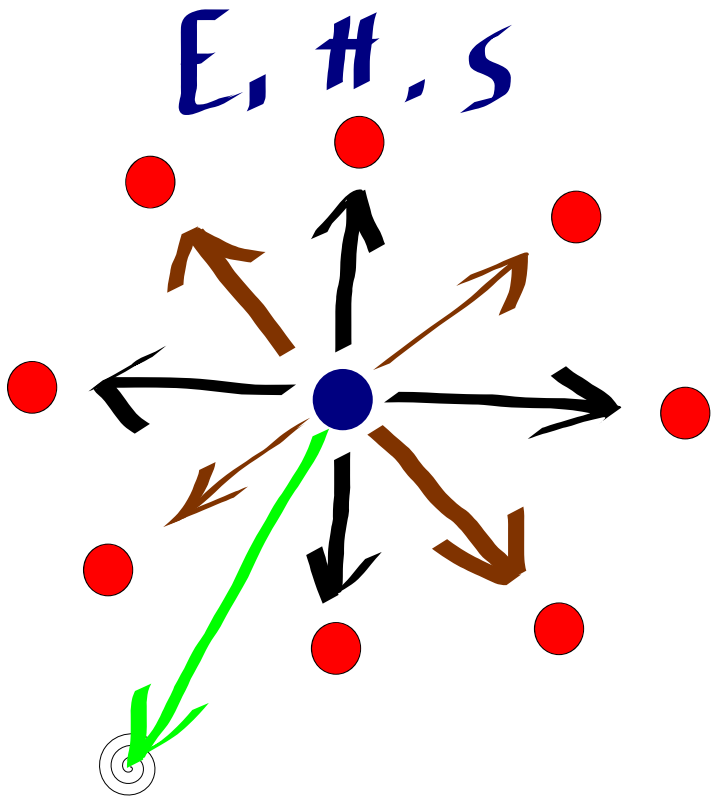
Find minimum cut:



Credit: *Timo Stich, 2009*

Outline

Information on Every Node



Every node has:

- Height.
- Excess flow.
- Status (speed-up).
- 4-8(!) edges, plus sink.

All are integers.

Implementation Structure

- Simple idea: every thread is responsible for one node/pixel.
- Apply Push and Relabel to every node. Repeat it until flow stops changing.
- Size blocks appropriately (32x8 threads, based on previous works). Pad if necessary.
- One kernel to initialize graph.
- One kernel for Push and another for Relabel.
- Two kernels to extract labeling.

Main Operations

Push:

- Read the height of all neighbors (shared memory applicable).
- Read/modify excess flow and edge capacities.
  Avoid hazards by using atomic operations.

Relabel:

- Read excess flow and edge information. Boolean would suffice.
- Read/modify heights (shared memory again applicable).

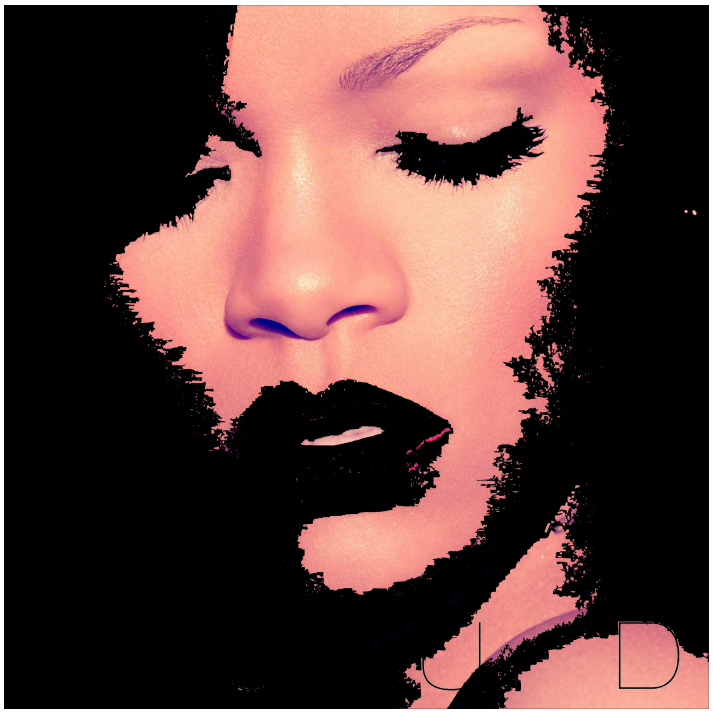
Outline

Neighborhood Comparison

Skin detection based on mixture of Gaussians by Jones, Regh.

Original picture (3600x3600).

Neighborhood Comparison



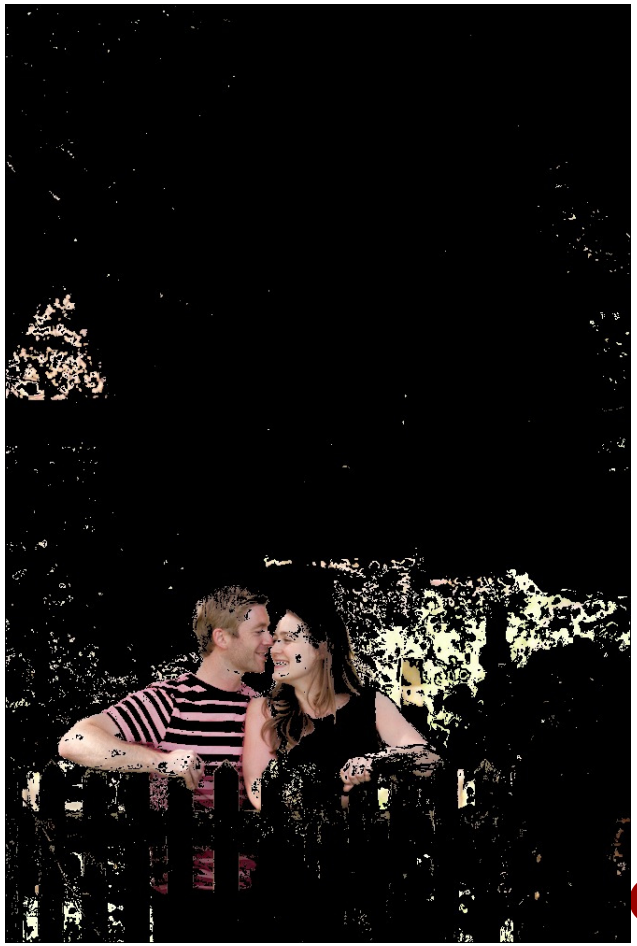
4-Neighborhood, 400 iterations.

8-Neighborhood, 4000 iterations.

More samples

Original and Naive

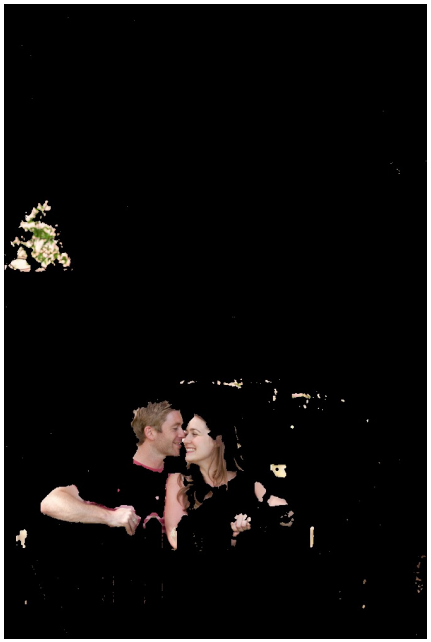
Original picture.



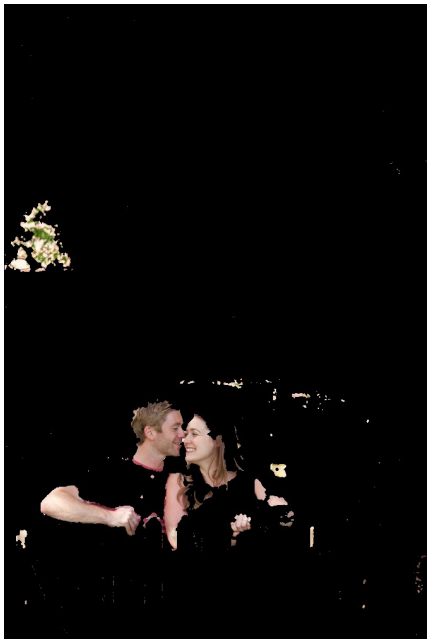
Naive segmentation.

More samples

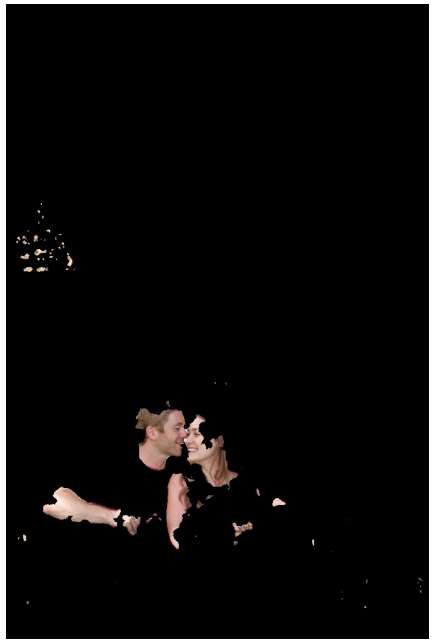
8-Neighborhood



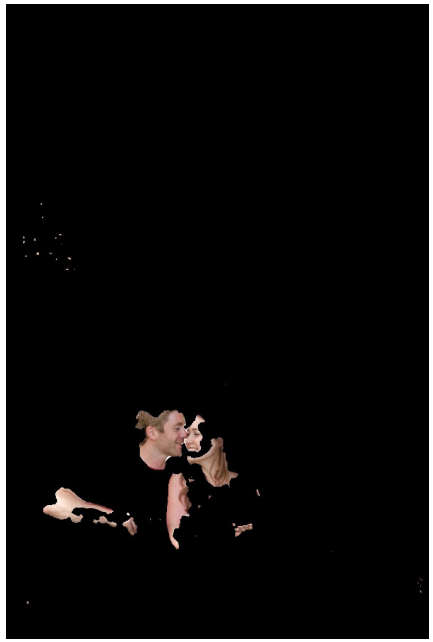
Segmentation with P=1000.



Segmentation with P=1500.



Segmentation with P=2000.



Segmentation with P=2500.

Outline

Comparison with Previous Works

Considered works include:

- CudaCuts by Vibhav Vineet and P. J. Narayanan, 2008.
- Own CPU implementation (for general graphs).
- CPU implementation (for specific application) by Boykov *et al.*.
- Timo Stich, 2009 (Nvidia ©, faster than CudaCuts).

Implementation	μs **per iteration**
CudaCuts (640x480)	122.23
Ours with 8-N (640x480)	101.10
Ours with 4-N (640x480)	71.01
Ours with 4-N (3600x3600)	5319.82

Further Comparison with Previous Works

- Currently only per iteration comparisons done.
- CPU implementations and GPU without source are not yet comparable.
- For a broader comparison, equal test cases must be used.
- Our number of iterations is still high(!): Global Relabeling technique necessary.

Outline

List of Performance Improvements

- Use shared memory for accessing heights (done).
- Keep list of active and inactive blocks (done, can be improved).
- Use Global Relabeling (currently “manually”).
- Height double buffering (because of memory boundness).
- Compress neighbor and edge status.
- Use Wave Push (and Wave Global Relabel).
- Use Dynamic Updating.

Outline

List of Problems

- The implementation depends on atomic functions (sm_11).
- Too many atomic accesses(?) end up generating: (6) the launch timed out and was terminated. [GPU also in use for display]
- Both can be solved by using the Wave technique.

Outline

Finding More Applications

Stereo Depth Estimation

Binary Image Segmentation

Photo Montage (aka Image Stitching)

Credit: MRF Evaluation, Middlebury College

- Skin detection (done - ~40ms per 800x600 frame).
- Segmentation background/foreground.
- Image Stitching.
- Stereo Depth Estimation.

Conclusion

- Suggestions or questions?
- Thank you!

- Some additional material and references:
  - `http://cvit.iiit.ac.in/papers/rtGCuts_2008.pdf`
  - `http://www.nvidia.com/content/GTC/documents/1060_GTC09.pdf`
  - `http://vision.middlebury.edu/`