GPU Programming in Computer Vision

Final Projects

Thomas Möllenhoff, Robert Maier, Caner Hazirbas

Summer Semester 2015

Project Phase (Sept 14 - Oct 2)

- Form groups of 3 people

- Implement a computer vision algorithm in CUDA
  - Select your 3 favorite topics
  - We will assign the projects to the groups

- Regular meetings with your supervisor
- Send source code to your supervisor until Oct 7

- Cheating: all involved groups will get the grade 5.0

Presentations (Oct 5/6)

- 15 minutes per group
- Prepare slides
  - Explain the task
  - Explain how you proceeded to solve the task
  - Show your results
- Live demo
- Q&A session

Final Project Proposals

Implement your own project idea?

1) Matching Deformable 3D Shapes (Emanuele Rodola)
2) Shortest Path Parallel Computation (Frank R. Schmidt, Emanuele Rodola)
3) Nonlinear Shape Registration (Csaba Domokos)
4) Depth-Adaptive Superpixels (Lingni Ma, Thomas Möllenhoff)
5) Joint Motion Estimation and Image Reconstruction (Michael Möller)
6) Variational Super-resolution (Robert Maier, Thomas Möllenhoff)
7) RGB-D Keyframe Fusion (Robert Maier)
8) TSDF Volume with Median Color Fusion (Robert Maier)
9) Octree TSDF Volume (Robert Maier)
10) Dense Visual Odometry (Robert Maier)

Matching Deformable 3D Shapes

Supervisor: Dr. Emanuele Rodolà

State-of-the-art:
Solution that transfers **functions to functions**.

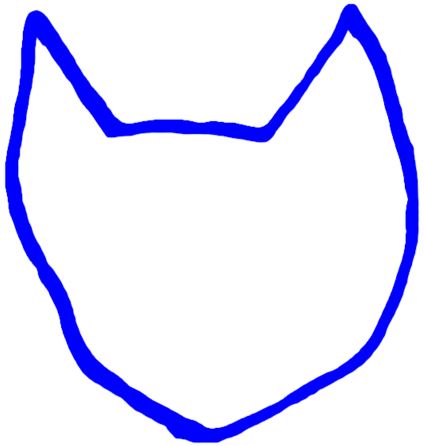
How to convert to a solution that maps **points to points**?

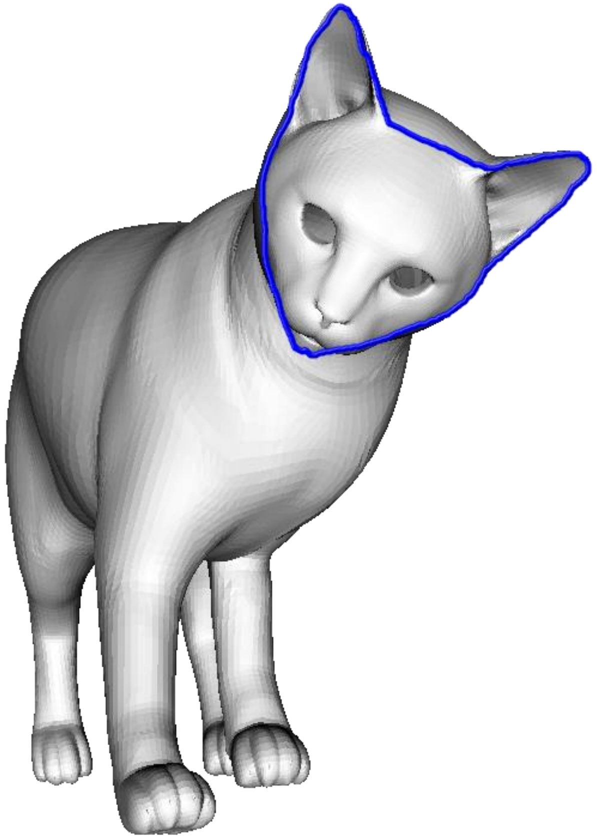
Our solution is an intrinsic variant of **ICP** for deformable shapes.

We have Matlab/C++ code to compare against.

Shortest Path
Parallel Computation

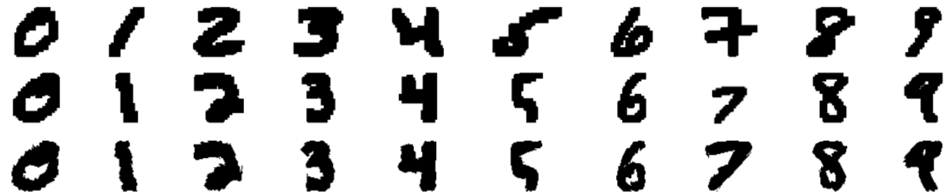
Dr. Frank Schmidt, Dr. Emanuele Rodolà





Nonlinear Shape Registration

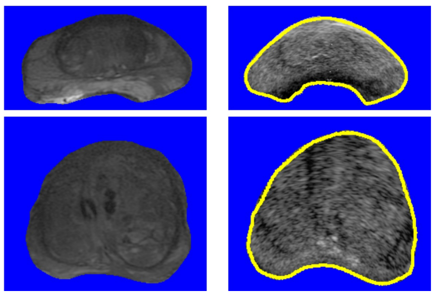
Registration: *find the geometric deformation between the shapes* (*i.e.* binary image)

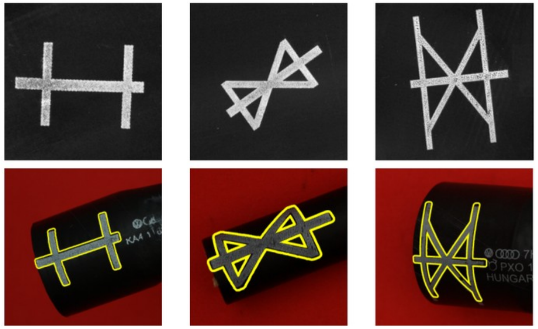


Sub-tasks of the problem

- Calculate image moments
  sum of the power of pixel coordinates
- Calculate efficiently *Thin Plate Spline* (TPS) transformation of a shape
- Solve a system of non-linear equations via *Levenberg-Marquardt algorithm*
  - standard algorithm for nonlinear minimization
  - there already exist some implementation in Cuda

- <u>Optional</u>: interface for Python or Matlab



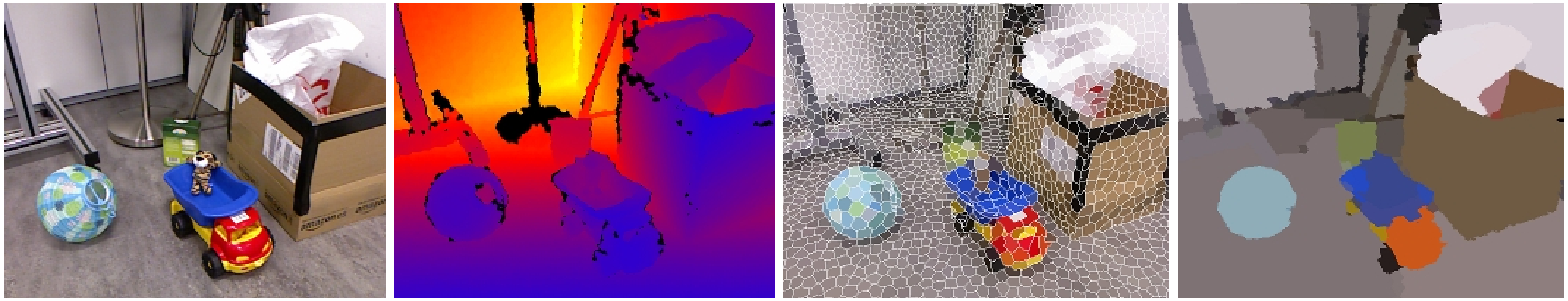


Materials

- The paper of the method is available
  https://docs.google.com/file/d/0B6gqeZujyM56c1k4SGhaZzNjX1U/edit?usp=sharing
- The reference implementation in Matlab is also available
- The first author would be happy to discuss about relating questions
  https://sites.google.com/site/cdomokosres/

An interesting and relevant problem in image processing
You can learn something about nonlinear function minimization

Depth-Adaptive Super Pixels



(a) Color input image (b) Depth input values (c) Depth-adaptive superpixels (d) Segments from sDASP

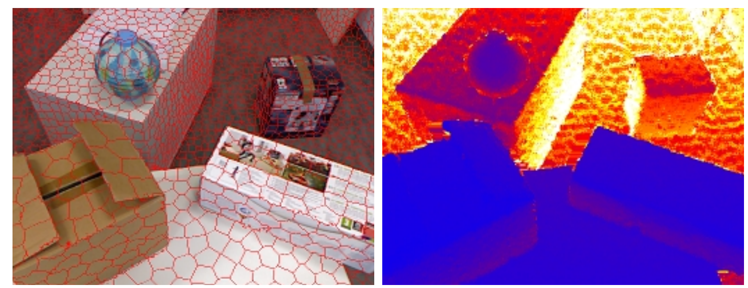
Idea: Group pixels into perceptually meaningful regions, taking depth into consideration.

Task:

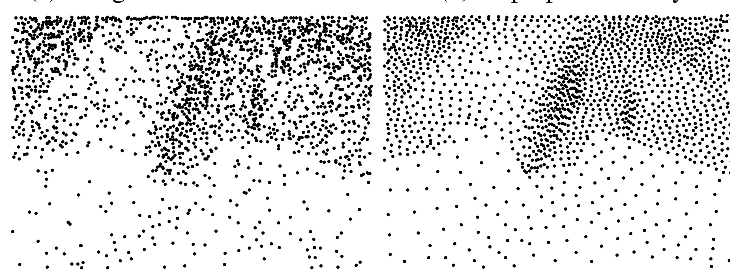
- Compute a superpixel density at each pixel.
- Sample cluster seeds according to this density.
- For each pixel compute a 9-dimensional feature representation.
- Run an iterative k-means algorithm.
- <u>Optional</u>: segmentation (spectral graph theory).

Available material:

- CUDA implementation of [1]
  https://github.com/painnick/gSLIC/tree/master/gSLIC
- CPU implementation of [2]
  https://github.com/Danvil/asp



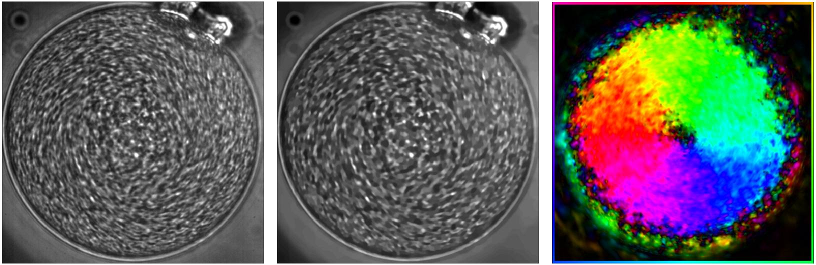
(a) Image with DASP borders (b) Superpixel density



(c) Sampled cluster seeds (d) Centers after 20 iterations

[1] Achanta, Radhakrishna, et al. "SLIC superpixels compared to state-of-the-art superpixel methods.", IEEE Transactions on *Pattern Analysis and Machine Intelligence (2012).*
[2] Weikersdorfer et. al. "Depth-adaptive superpixels.", IEEE International Conference on *Pattern Recognition (2012).*

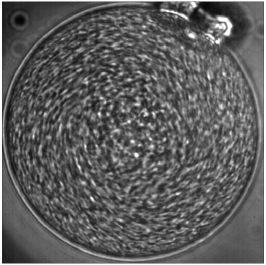
Joint Motion Estimation and Image Reconstruction



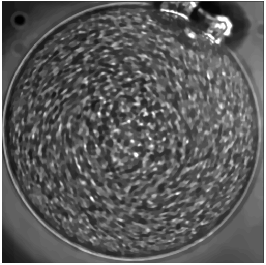
(a) Input frames f (b) Denoised frames (c) Estimated flow

From: Hendrik Dirks, PhD Thesis 2015.

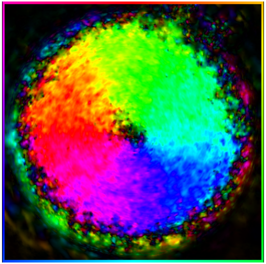
Joint Motion Estimation and Image Reconstruction



(a) Input frames f



(b) Denoised frames



(c) Estimated flow

Energy minimization approach:

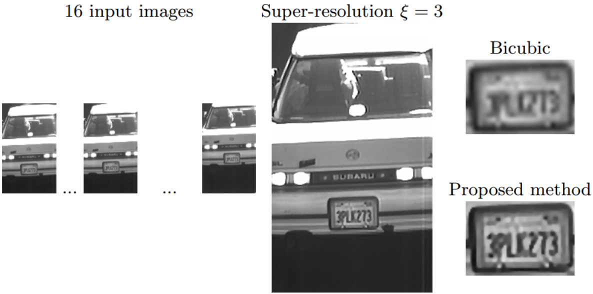
$$\min_{u,\mathbf{v}} \int_0^T \underbrace{\frac{1}{2}\|u-f\|_2^2 + \alpha\|\nabla u\|_1}_{\text{TV denoising}} + \beta \underbrace{\|\nabla \mathbf{v}\|_1}_{\text{Flow regularity}} + \gamma \underbrace{\|u_t + \nabla u \cdot \mathbf{v}\|_1}_{\text{Coupling}} \, dt$$

From: Hendrik Dirks, PhD Thesis 2015.

Joint Motion Estimation and Image Reconstruction

Extended energy minimization approach:

$$\min_{u,\mathbf{v}} \int_0^T \underbrace{\frac{1}{2}\left\|Ku - f\right\|_2^2 + \alpha\left\|\nabla u\right\|_1}_{\text{TV super resolution}} + \beta \underbrace{\left\|\nabla \mathbf{v}\right\|_1}_{\text{Flow regularity}} + \gamma \underbrace{\left\|u_t + \nabla u \cdot \mathbf{v}\right\|_1}_{\text{Coupling}} \, dt$$



First image: Unger, Pock, Werlberger, Bischof 2010.
Second image: Goldlücke, Aubry, Kolev, Cremers 2014.

Variational Super-resolution

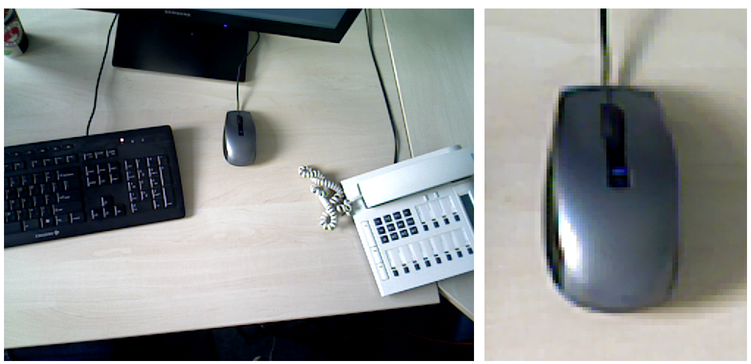
- Input: several low-res. RGB images (RGB-D frames)
- Goal: obtain high-res. image (sharper than input images)

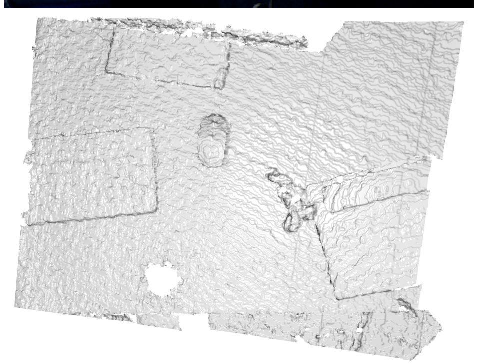
input image close-up   super-resolution texture

- Reason: every surface patch observed in multiple images
  -> invert blurring and downsampling
- Reference: A Convex Approach for Variational Super-Resolution [Unger et al, 2010]

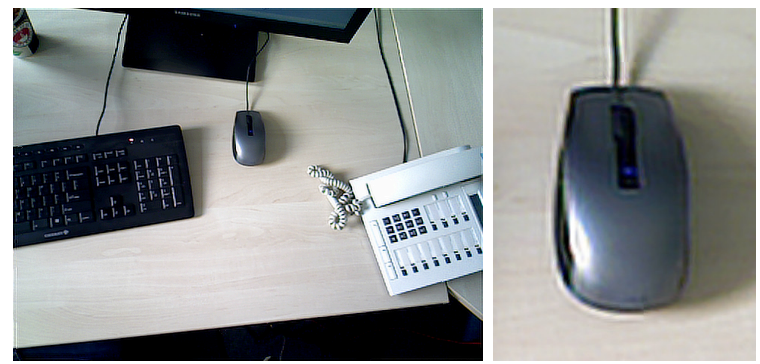
RGB-D Keyframe Fusion

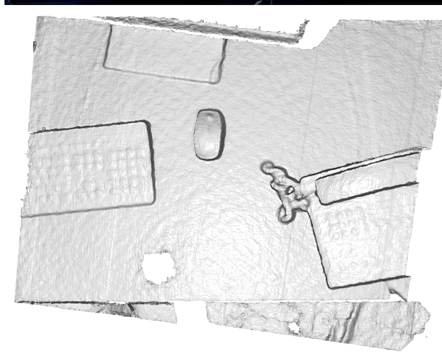
- Idea: fuse low-res. input RGB-D frames into high resolution RGB-D keyframes
  - Depth fusion (warp, upsample, fuse)
  - Color fusion (Deblur, warp, fuse)





LR input frame



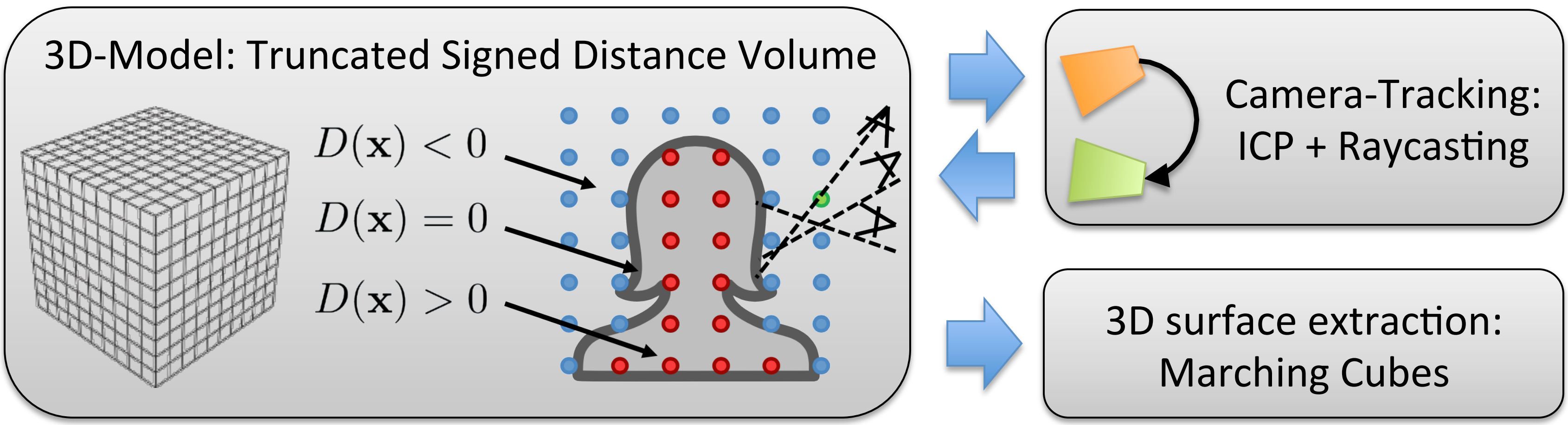


Fused SR keyframe

TSDF Volume

- KinectFusion [Newcombe et al, ISMAR 2011]: Real-time dense 3D reconstruction from RGB-D sensors

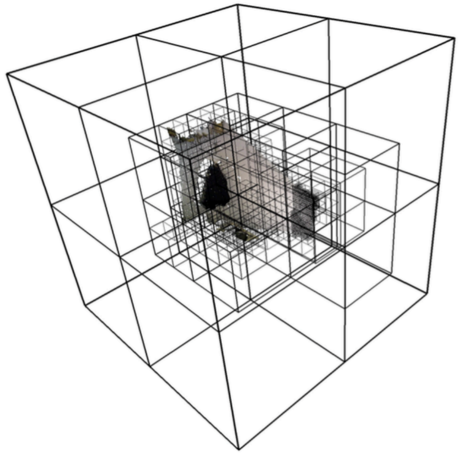
- 3D reconstruction algorithm:



- Idea: Median Color Fusion

Octree TSDF Volume

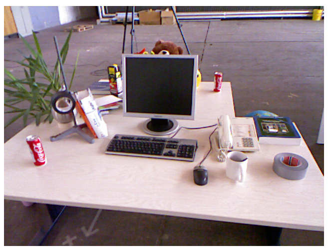
- TSDF voxel grid: limited 3D volume size and/or resolution, high memory consumption
- Idea: Use more memory efficient 3D scene representation based on an Octree

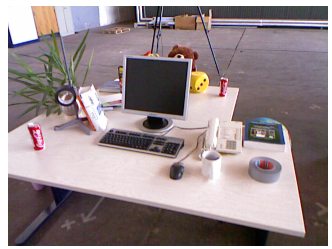


- Reference: Octree-based fusion for realtime 3D reconstruction [Zeng et al., 2013]

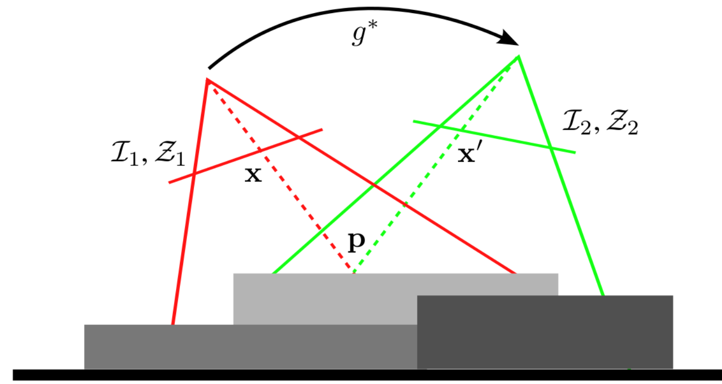
Dense Visual Odometry

- Robust Odometry Estimation for RGB-D Cameras
  - Given: Two RGB-D frames





  - Goal: estimate camera motion g* by minimizing photometric and geometric error



- Real-time CPU implementation (320x240)
- Reference: Robust Odometry Estimation for RGB-D Cameras [Kerl et al, ICRA 2013]

Next steps

- Today: send email to cuda-ss15@in.tum.de
  - Group Members
  - Your 3 favorite topics

- After project assignments: meet with your supervisor

- Any questions?