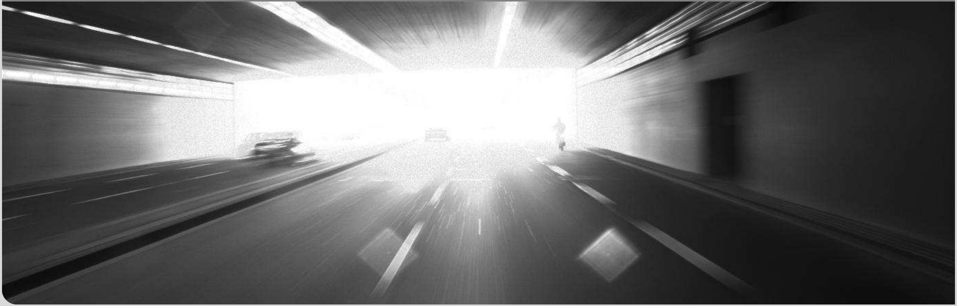
Karlsruhe Institute of Technology

Institut für Mess- und Regelungstechnik

Efficient Large-Scale Stereo Matching

Andreas Geiger*, Martin Roser* and Raquel Urtasun**

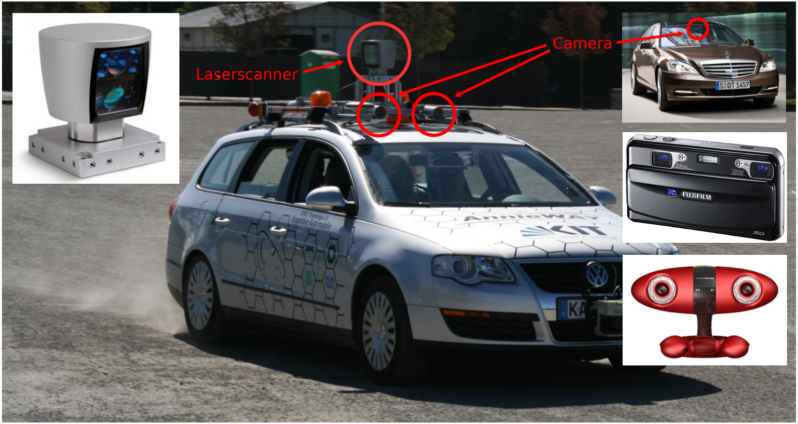
*Karlsruhe Institute of Technology **Toyota Technological Institute at Chicago



Contents

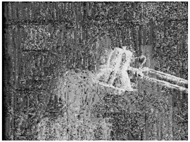
Contents

Motivation



Why is 3D from Stereo hard?

- Ambiguities
- Textureless regions

- Sensor saturation
- Non-Lambertian surfaces

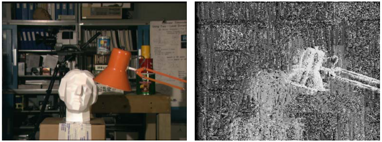
- Δz grows quadratically
- Computational burden

Why is 3D from Stereo hard?

- Ambiguities
- Textureless regions

- Sensor saturation
- Non-Lambertian surfaces

- Δz grows quadratically
- Computational burden

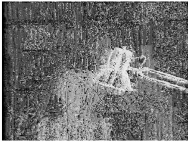


Why is 3D from Stereo hard?

- Ambiguities
- Textureless regions

- Sensor saturation
- Non-Lambertian surfaces

- Δz grows quadratically
- Computational burden



$$\underbrace{|\Delta z|}_{\text{distance error}} \approx \frac{z^2}{\underbrace{f}_{\text{focal length}} \cdot \underbrace{b}_{\text{baseline}}} \underbrace{|\Delta d|}_{\text{disparity error}}$$

Related Work: Local Methods

Local Methods

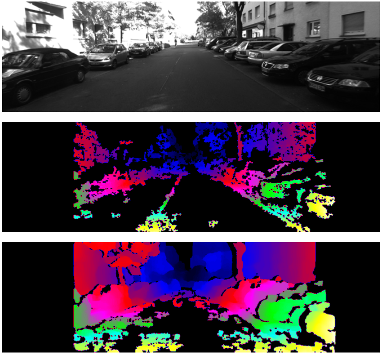
- Winner-takes-All

Examples

- Block matching (Scharstein 02)
- Adaptive windows (Kanade 94, Yoon 06)
- Plane-sweep (Collins 96, Gallup 07)

Problems

- Small matching ratios
- Border bleeding



Related Work: Global Methods

Global Methods

- Minimize 1D/2D energy $E(d) = E_{\text{data}}(d) + \lambda E_{\text{smooth}}(d)$

Examples

- Graph cuts, Belief propagation
  (Kolmogorov 02, Felzenszwalb 06)
- Variational methods
  (Pock 07, Zach 09)
- Fusion moves
  (Woodford 08, Bleyer 10)

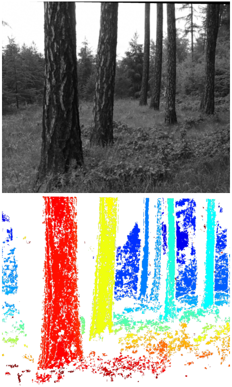
Problems

- Computational and memory
  requirements
- Pairwise potentials
  can not model planarity

Related Work: Seed-and-Grow

Seed-and-Grow Methods

- Grow disparity components from random seeds



Examples

- (Cech 07)
- (Sara 03)

Problems

- Slanted/textureless surfaces
- No dense disparity maps

Contents

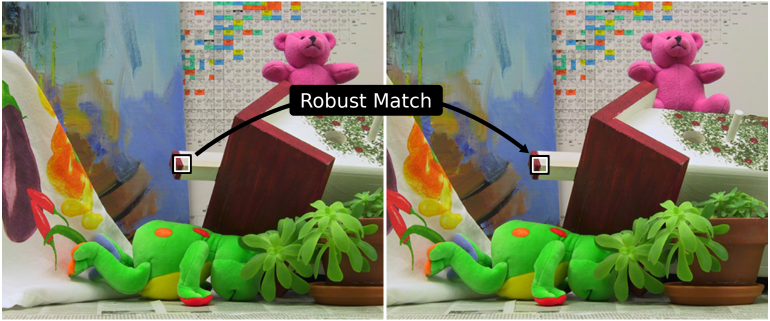
Idea

- Assumption: rectified images
- Image pairs contain 'easy' and 'hard' correspondences
- Robustly match 'easy' correspondences on regular grid
- Build prior on dense search space $\Rightarrow$ dense matching

Idea

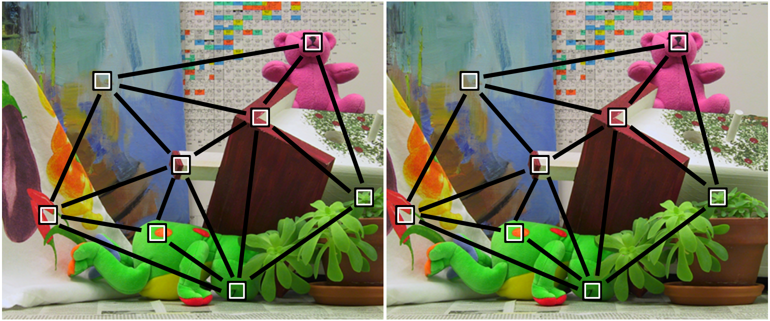
- Assumption: rectified images
- Image pairs contain 'easy' and 'hard' correspondences
- Robustly match 'easy' correspondences on regular grid
- Build prior on dense search space ⇒ dense matching

Idea



- Assumption: rectified images
- Image pairs contain 'easy' and 'hard' correspondences
- Robustly match 'easy' correspondences on regular grid
- Build prior on dense search space ⇒ dense matching

Idea



- Assumption: rectified images
- Image pairs contain 'easy' and 'hard' correspondences
- Robustly match 'easy' correspondences on regular grid
- Build prior on dense search space $\Rightarrow$ dense matching

Efficient Large-Scale Stereo

Notation

- Robust support points $\mathbf{S} = \{\mathbf{s}_1, ..., \mathbf{s}_M\}$ with $\mathbf{s}_m = (u_m \; v_m \; d_m)^T$
- Disparity $d_n \in \mathbb{N}$
- Observations $\mathbf{o}_n = (u_n \; v_n \; \mathbf{f}_n)^T$
- Local image features $\mathbf{f}_n$

Algorithm

- Split image domain into support points $\mathbf{S}$ and dense pixels
- Assume factorization of distribution over disparity, observations and support points into ...

Efficient Large-Scale Stereo

Notation

- Robust support points $\mathbf{S} = \{\mathbf{s}_1, ..., \mathbf{s}_M\}$ with $\mathbf{s}_m = (u_m\ v_m\ d_m)^T$
- Disparity $d_n \in \mathbb{N}$
- Observations $\mathbf{o}_n = (u_n\ v_n\ \mathbf{f}_n)^T$
- Local image features $\mathbf{f}_n$

Algorithm

- Split image domain into support points $\mathbf{S}$ and dense pixels
- Assume factorization of distribution over disparity, observations and support points into ...

Efficient Large-Scale Stereo

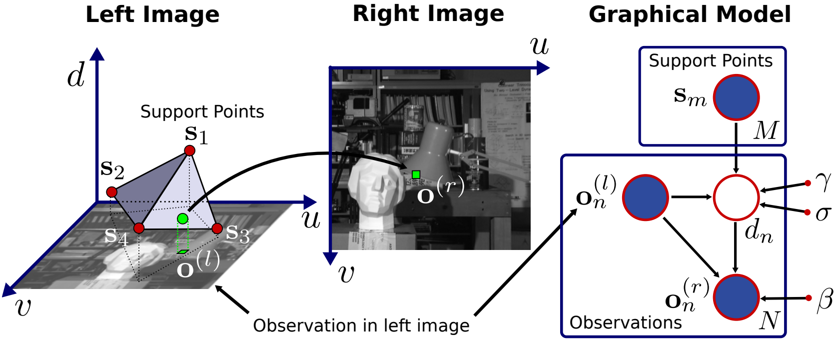
Notation

- Robust support points $\mathbf{S} = \{\mathbf{s}_1, ..., \mathbf{s}_M\}$
  with $\mathbf{s}_m = (u_m \; v_m \; d_m)^T$
- Disparity $d_n \in \mathbb{N}$
- Observations $\mathbf{o}_n = (u_n \; v_n \; \mathbf{f}_n)^T$
- Local image features $\mathbf{f}_n$

Algorithm

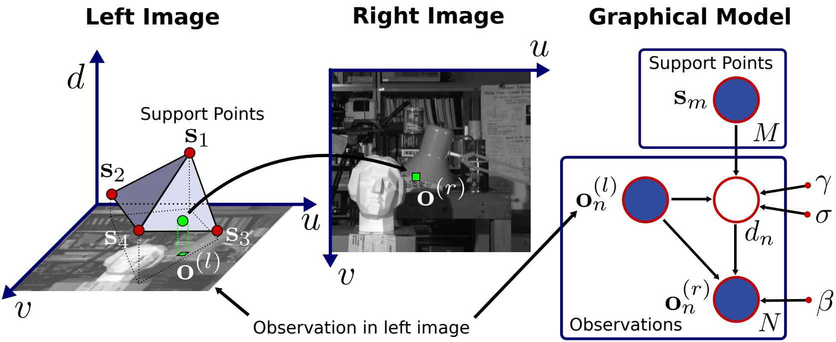
- Split image domain into support points $\mathbf{S}$ and dense pixels
- Assume factorization of distribution over disparity, observations and support points into ...

Model



$$p(d_n, \mathbf{o}_n^{(l)}, \mathbf{o}_n^{(r)}, \mathbf{S}) \propto \underbrace{p(d_n | \mathbf{S}, \mathbf{o}_n^{(l)})}_{\text{Prior}} \underbrace{p(\mathbf{o}_n^{(r)} | \mathbf{o}_n^{(l)}, d_n)}_{\text{Likelihood}}$$

Model

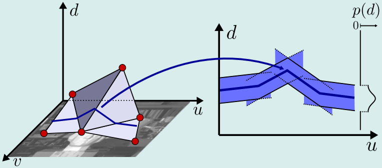


$$p(d_n, \mathbf{o}_n^{(l)}, \mathbf{o}_n^{(r)}, \mathbf{S}) \propto \underbrace{p(d_n|\mathbf{S}, \mathbf{o}_n^{(l)})}_{\text{Prior}} \underbrace{p(\mathbf{o}_n^{(r)}|\mathbf{o}_n^{(l)}, d_n)}_{\text{Likelihood}}$$

Prior and Likelihood

Prior $p(d_n|\mathbf{S}, \mathbf{o}_n^{(l)})$

- Support pt. triangulation
- Piecew. linear manifold
- Local extrapolation



Likelihood $p(\mathbf{o}_n^{(r)}|\mathbf{o}_n^{(l)}, d_n)$

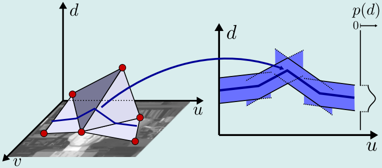
- Laplace distribution
- 5×5 block window
- 3×3 Sobel filter

Prior and Likelihood

Karlsruhe Institute of Technology

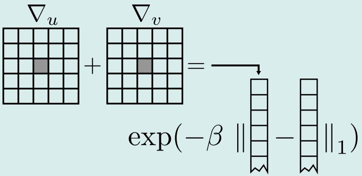
Prior $p(d_n|\mathbf{S}, \mathbf{o}_n^{(l)})$

- Support pt. triangulation
- Piecew. linear manifold
- Local extrapolation



Likelihood $p(\mathbf{o}_n^{(r)}|\mathbf{o}_n^{(l)}, d_n)$

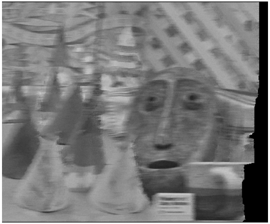
- Laplace distribution
- 5×5 block window
- 3×3 Sobel filter



Sampling from the model

Left image

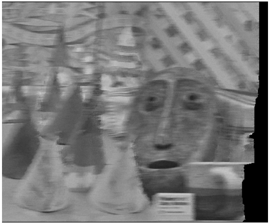
Sample mean



Sampling from the model

Left image

Sample mean



Right image

Contents

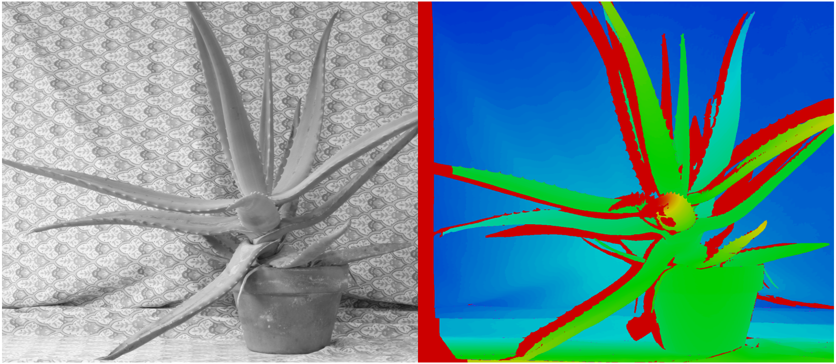
Middlebury Benchmark

900 x 750 pixels, ground truth

Middlebury Benchmark

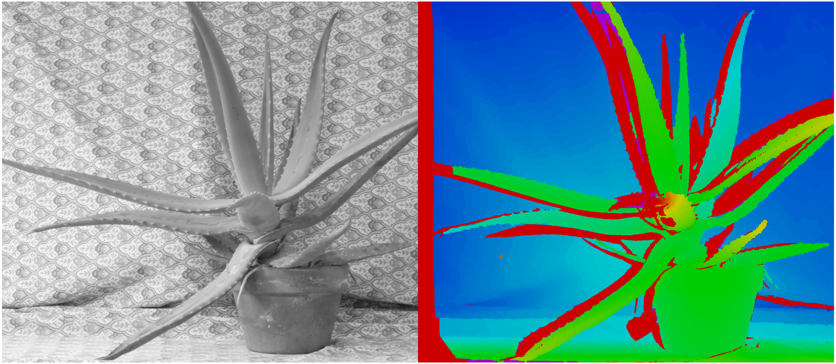
900 x 750 pixels, 0.4 seconds

Middlebury Benchmark



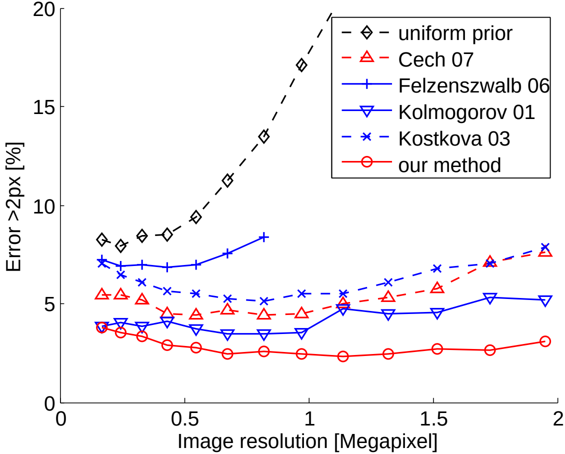
1300 x 1100 pixels, ground truth

Middlebury Benchmark

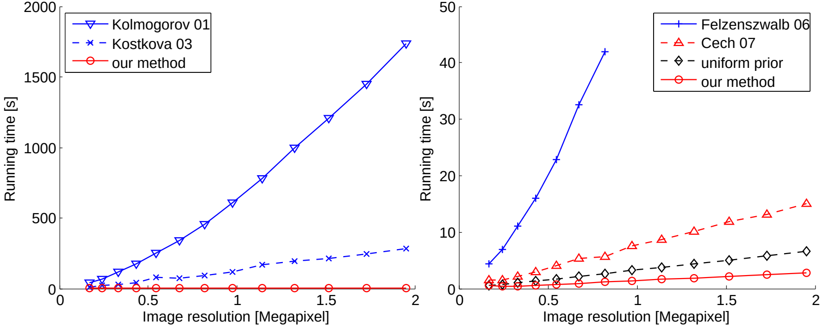


1300 x 1100 pixels, 1 second

Accuracy (on cones image pair)



Running times (on cones image pair)



[For more details see: Geiger et al., ACCV 2010]

3D Reconstruction: Brussels

2 seconds

[http://cvlab.epfl.ch/data/strechamvs/]

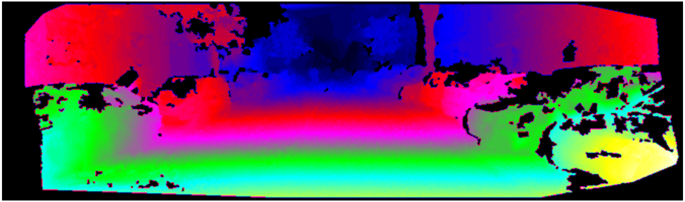
3D Face Reconstruction

[http://www.fujifilm.com/products/3d]

Urban Scene Reconstruction

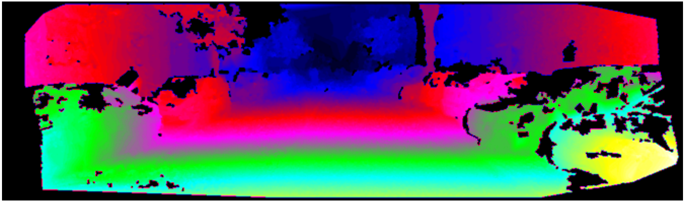
Contents

Summary



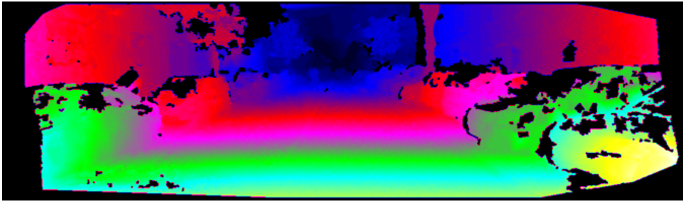
- Simple prior based on sparse feature matches
- Reduced ambiguities and run-time
- Takes into account slanted surfaces
- Real-time 3D reconstruction of static scenes on CPU
- C++ / MATLAB code available at **http://cvlibs.net**

Summary



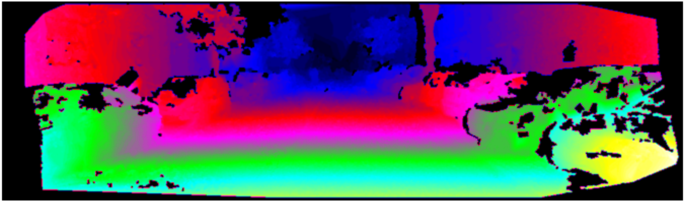
- Simple prior based on sparse feature matches
- Reduced ambiguities and run-time
- Takes into account slanted surfaces
- Real-time 3D reconstruction of static scenes on CPU
- C++ / MATLAB code available at **http://cvlibs.net**

Summary



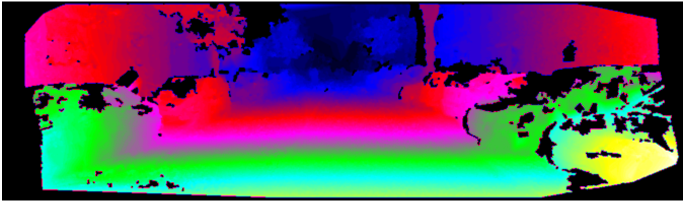
- Simple prior based on sparse feature matches
- Reduced ambiguities and run-time
- Takes into account slanted surfaces
- Real-time 3D reconstruction of static scenes on CPU
- C++ / MATLAB code available at **http://cvlibs.net**

Summary



- Simple prior based on sparse feature matches
- Reduced ambiguities and run-time
- Takes into account slanted surfaces
- Real-time 3D reconstruction of static scenes on CPU
- C++ / MATLAB code available at **http://cvlibs.net**

Summary



- Simple prior based on sparse feature matches
- Reduced ambiguities and run-time
- Takes into account slanted surfaces
- Real-time 3D reconstruction of static scenes on CPU
- C++ / MATLAB code available at **http://cvlibs.net**

Future Work

- Develop better priors
- Incorporate segmentation / global reasoning on lines
- GPU implementation
  (goal: 20 fps at 1-2 megapixels)
- Employ as unitary potentials on global methods
  $\Rightarrow$ smaller label sets

- **Thank you!**

Future Work

- Develop better priors
- Incorporate segmentation / global reasoning on lines
- GPU implementation
  (goal: 20 fps at 1-2 megapixels)
- Employ as unitary potentials on global methods
  $\Rightarrow$ smaller label sets

- **Thank you!**

Future Work

- Develop better priors
- Incorporate segmentation / global reasoning on lines
- GPU implementation
  (goal: 20 fps at 1-2 megapixels)
- Employ as unitary potentials on global methods
  $\Rightarrow$ smaller label sets

- **Thank you!**

Future Work

Karlsruhe Institute of Technology

- Develop better priors
- Incorporate segmentation / global reasoning on lines
- GPU implementation
  (goal: 20 fps at 1-2 megapixels)
- Employ as unitary potentials on global methods
  $\Rightarrow$ smaller label sets

- **Thank you!**