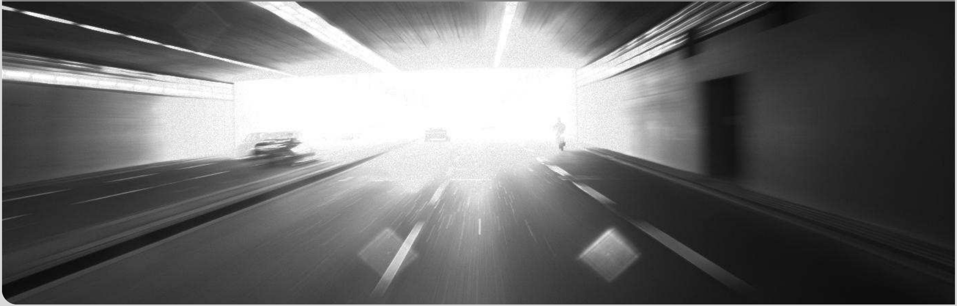
Karlsruhe Institute of Technology

Institut für Mess- und Regelungstechnik

Automatic Camera and Range Sensor Calibration using a single Shot

Andreas Geiger Frank Moosmann Ömer Car Bernhard Schuster

INSTITUTE OF MEASUREMENT AND CONTROL SYSTEMS KARLSRUHE INSTITUTE OF TECHNOLOGY

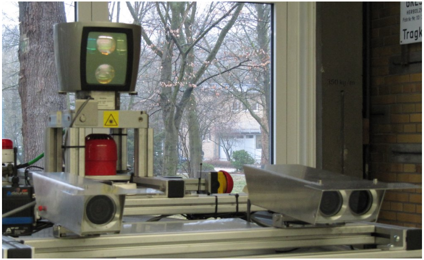


Motivation


KIT
Karlsruhe Institute of Technology
The KITTI Vision Benchmark Suite
KA EV 842
TOYOTA TECHNOLOGICAL INSTITUTE
CHICAGO

Overview

Cameras / Microsoft Kinect



Cameras / Velodyne HDL-64

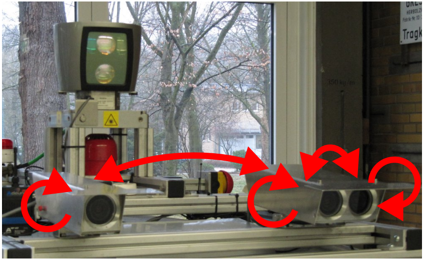
Setup: Video cameras (+ range sensor)

Goals

1. Calibrate cameras intrinsically and extrinsically
2. Register range sensor to cameras

Overview

Cameras / Microsoft Kinect



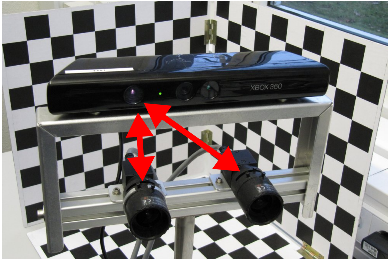
Cameras / Velodyne HDL-64

Setup: Video cameras (+ range sensor)

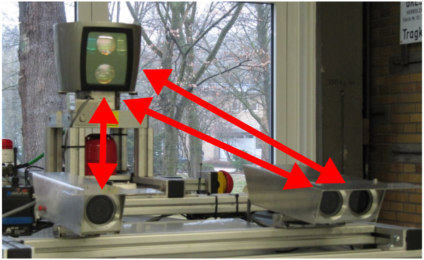
Goals

1. Calibrate cameras intrinsically and extrinsically
2. Register range sensor to cameras

Overview



Cameras / Microsoft Kinect



Cameras / Velodyne HDL-64

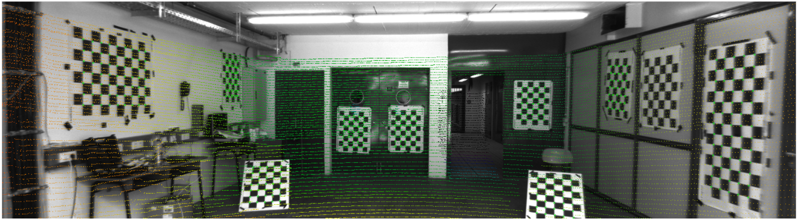
Setup: Video cameras (+ range sensor)

Goals

1. Calibrate cameras intrinsically and extrinsically
2. Register range sensor to cameras

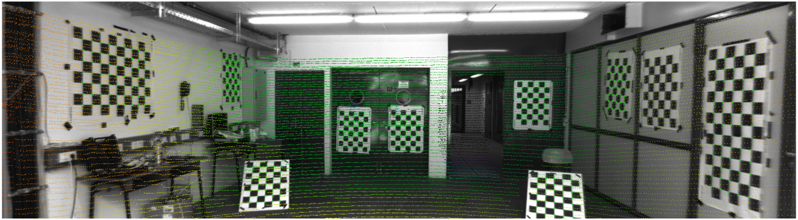
Proposed Method

- **Contributions**
  - Automatic camera and range sensor calibration
  - Our method requires only one image / scan per sensor
  - Processing time < 5 minutes
- **Assumptions**
  - Planar checkerboards (presented at different poses)
  - Overlapping field of view (e.g., stereo)

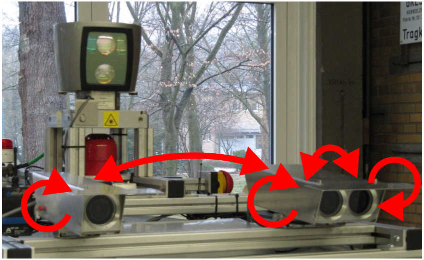


Proposed Method

- **Contributions**
  - Automatic camera and range sensor calibration
  - Our method requires only one image / scan per sensor
  - Processing time < 5 minutes
- **Assumptions**
  - Planar checkerboards (presented at different poses)
  - Overlapping field of view (e.g., stereo)



Camera Calibration

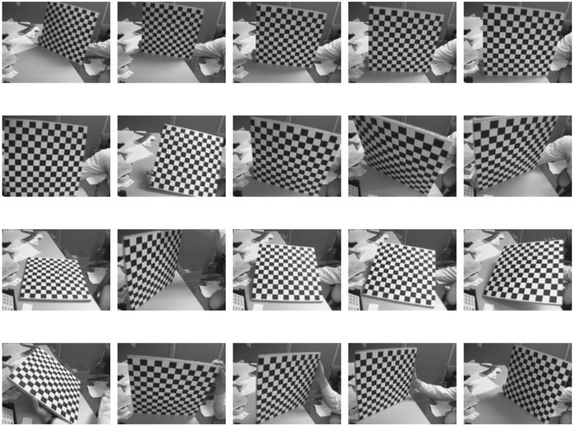


Goals

1. Calibrate cameras intrinsically and extrinsically
2. Register range sensor to cameras

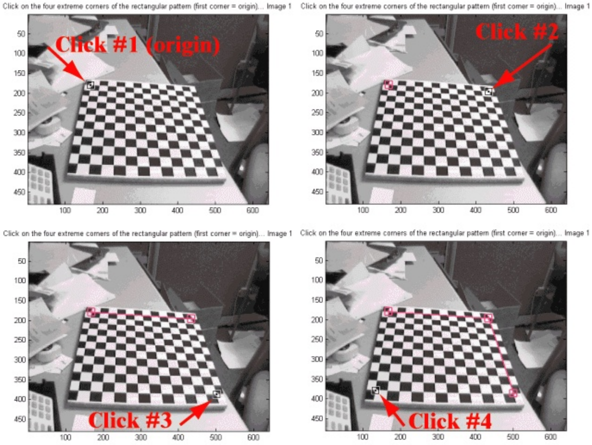
Camera: Related Work

Bouget's Camera Calibration Toolbox for Matlab / OpenCV



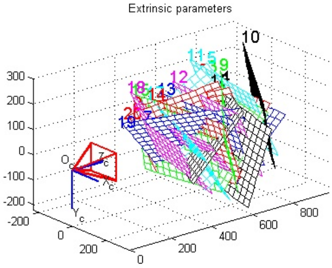
Camera: Related Work

Bouget's Camera Calibration Toolbox for Matlab / OpenCV



Related Work: Camera Calibration

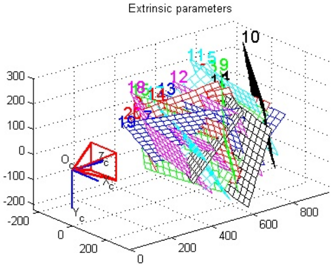
Bouget's Camera Calibration Toolbox for Matlab / OpenCV



- Requires $\geq$ 10 synchronized images
- No automatic corner detection and matching
- Time consuming

Related Work: Camera Calibration

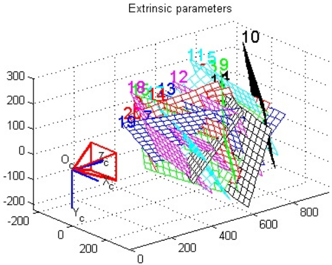
Bouget's Camera Calibration Toolbox for Matlab / OpenCV



- Requires $\geq$ 10 synchronized images
- No automatic corner detection and matching
- Time consuming

Related Work: Camera Calibration

Bouget's Camera Calibration Toolbox for Matlab / OpenCV

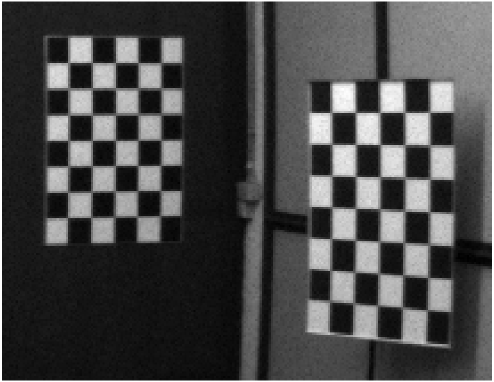


- Requires $\geq$ 10 synchronized images
- No automatic corner detection and matching
- Time consuming

Camera: Corner Detection

Corner detection and subpixel refinement

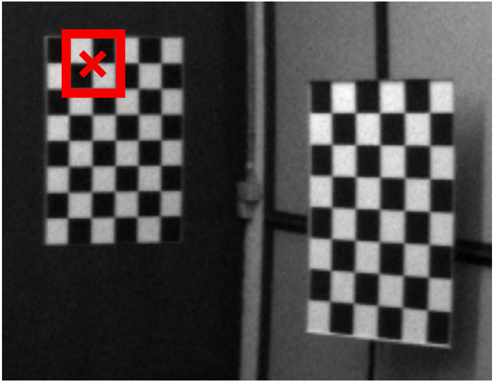
- Compute cornerness score for each pixel
- Non-maximum suppression, sub-pixel refinement



Camera: Corner Detection

Corner detection and subpixel refinement

- Compute cornerness score for each pixel
- Non-maximum suppression, sub-pixel refinement



Camera: Corner Detection

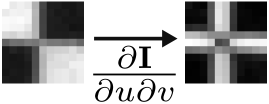
Corner detection and subpixel refinement

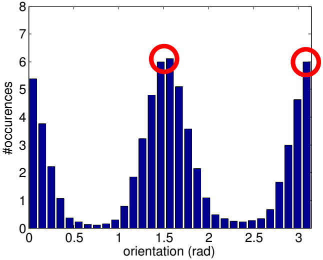
- Compute cornerness score for each pixel
- Non-maximum suppression, sub-pixel refinement

Camera: Corner Detection

Corner detection and subpixel refinement

- Compute cornerness score for each pixel
- Non-maximum suppression, sub-pixel refinement

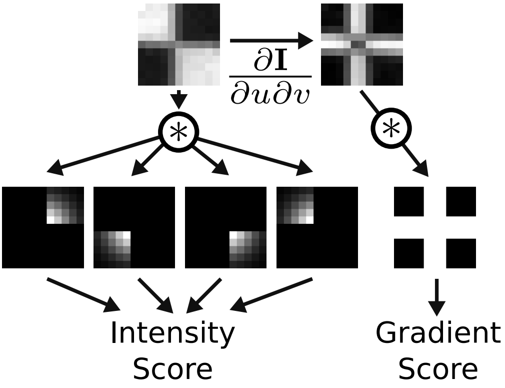




Camera: Corner Detection

Corner detection and subpixel refinement

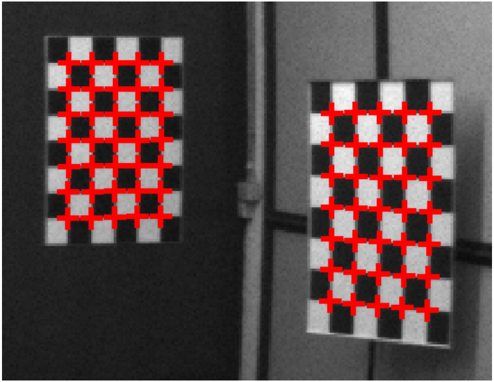
- Compute cornerness score for each pixel
- Non-maximum suppression, sub-pixel refinement



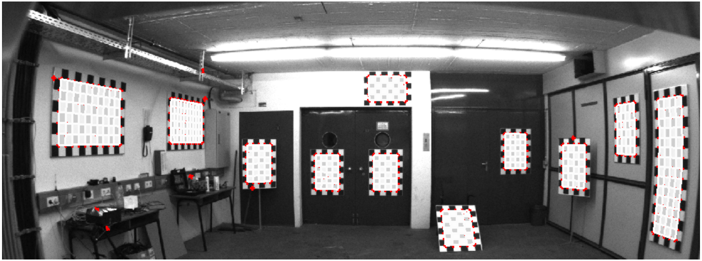
Camera: Corner Detection

Corner detection and subpixel refinement

- Compute cornerness score for each pixel
- Non-maximum suppression, sub-pixel refinement



Camera: Finding Checkerboards



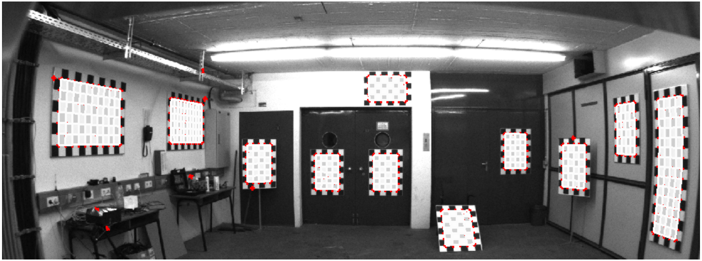
Parameters

- Corner locations: $\mathcal{X} = \{\mathbf{c}_1, .., \mathbf{c}_N\}$, $\mathbf{c}_i \in \mathbb{R}^2$
- Corner labels: $\mathcal{Y} = \{\mathbf{y}_1, .., \mathbf{y}_N\}$, $\mathbf{y}_i \in \{\mathcal{O}\} \cup \mathbb{N}^3$

Energy $E(\mathcal{X}, \mathcal{Y}) = E_{corners}(\mathcal{Y}) + E_{struct}(\mathcal{X}, \mathcal{Y})$

- $E_{corners} = -$ number of explained corners
- $E_{struct} = -$ corner collinearity

Camera: Finding Checkerboards



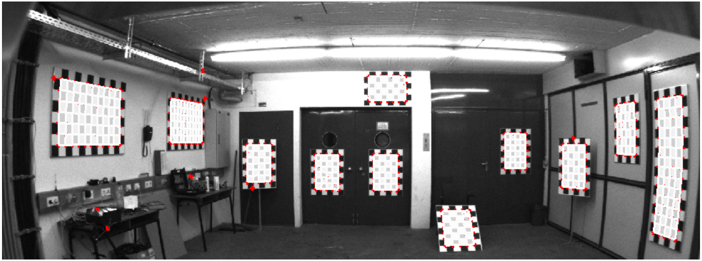
Parameters

- Corner locations: $\mathcal{X} = \{\mathbf{c}_1, .., \mathbf{c}_N\}$, $\mathbf{c}_i \in \mathbb{R}^2$
- Corner labels: $\mathcal{Y} = \{\mathbf{y}_1, .., \mathbf{y}_N\}$, $\mathbf{y}_i \in \{\mathcal{O}\} \cup \mathbb{N}^3$

Energy $E(\mathcal{X}, \mathcal{Y}) = E_{corners}(\mathcal{Y}) + E_{struct}(\mathcal{X}, \mathcal{Y})$

- $E_{corners} = -$ number of explained corners
- $E_{struct} = -$ corner collinearity

Camera: Finding Checkerboards



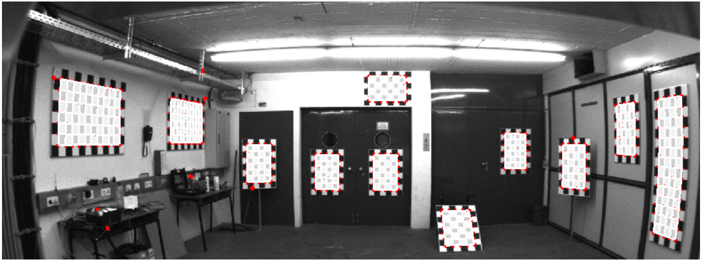
Parameters

- Corner locations: $\mathcal{X} = \{\mathbf{c}_1, .., \mathbf{c}_N\}$, $\mathbf{c}_i \in \mathbb{R}^2$
- Corner labels: $\mathcal{Y} = \{\mathbf{y}_1, .., \mathbf{y}_N\}$, $\mathbf{y}_i \in \{\mathcal{O}\} \cup \mathbb{N}^3$

Energy $E(\mathcal{X}, \mathcal{Y}) = E_{corners}(\mathcal{Y}) + E_{struct}(\mathcal{X}, \mathcal{Y})$

- $E_{corners} = -$ number of explained corners
- $E_{struct} = -$ corner collinearity

Camera: Finding Checkerboards

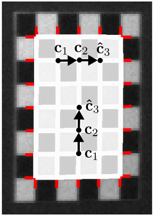


Parameters

- Corner locations: $\mathcal{X} = \{\mathbf{c}_1, .., \mathbf{c}_N\}$, $\mathbf{c}_i \in \mathbb{R}^2$
- Corner labels: $\mathcal{Y} = \{\mathbf{y}_1, .., \mathbf{y}_N\}$, $\mathbf{y}_i \in \{\mathcal{O}\} \cup \mathbb{N}^3$

Energy $E(\mathcal{X}, \mathcal{Y}) = E_{corners}(\mathcal{Y}) + E_{struct}(\mathcal{X}, \mathcal{Y})$

- $E_{corners} = -$ number of explained corners
- $E_{struct} = -$ corner collinearity



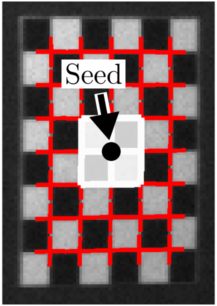
Collinearity

Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

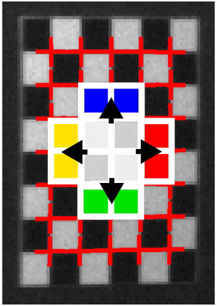


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

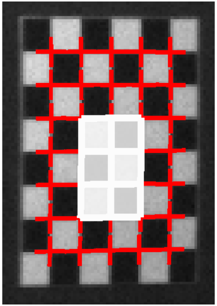


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

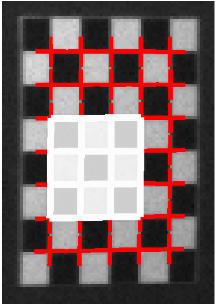


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

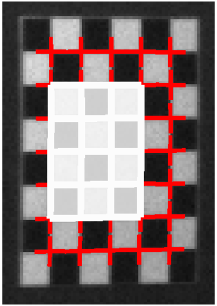


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

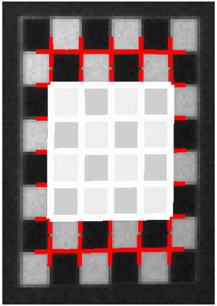


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

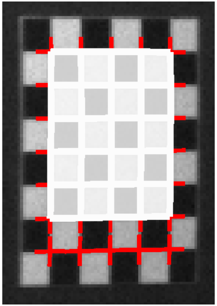


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

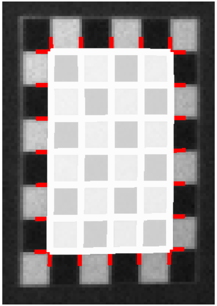


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

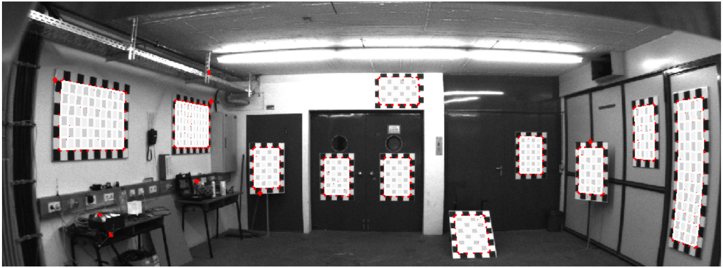


Camera: Finding Checkerboards

Exponential complexity $O(|\mathcal{X}|^{|\mathcal{L}|}) \Rightarrow$ **Search space pruning**

For each corner as seed do:

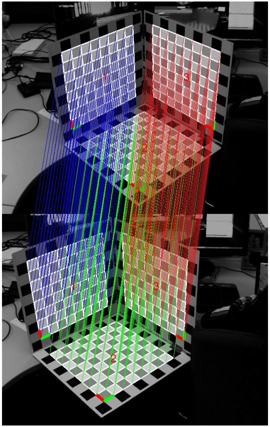
- Incrementally add neighboring corners with lowest energy
- Keep lowest energy solutions in case of overlaps



Camera: Matching and Optimization

Match checkerboards across images

- Sample 2 checkerboards per image
- Compute similarity transformation from center of checkerboards
- Maximize number of inliers



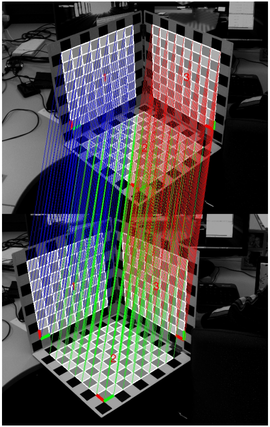
Parameter Optimization

- Parameters: $\{\mathbf{f}, \mathbf{c}, \alpha, k_1, .., k_5\}, \{\mathbf{r}, \mathbf{t}\}$
- Non-linear least squares (Gauss-Newton)

Camera: Matching and Optimization

Match checkerboards across images

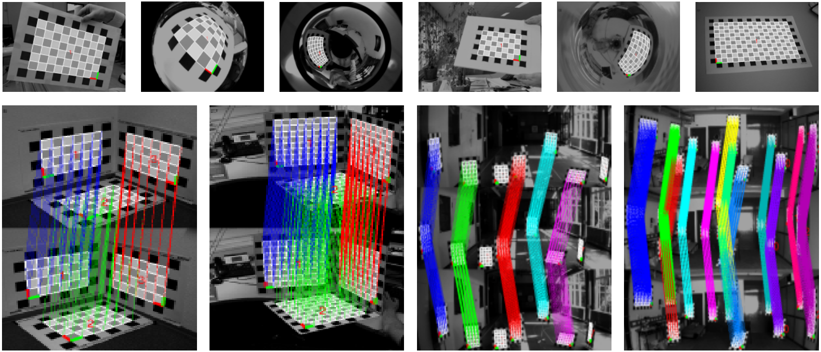
- Sample 2 checkerboards per image
- Compute similarity transformation from center of checkerboards
- Maximize number of inliers



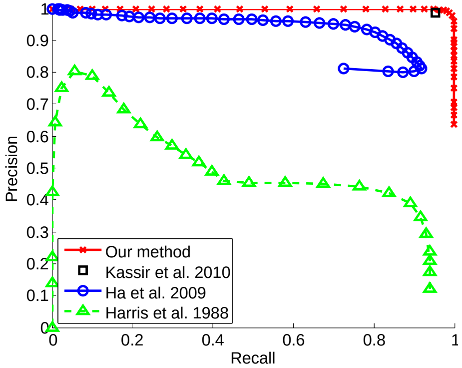
Parameter Optimization

- Parameters: $\{\mathbf{f}, \mathbf{c}, \alpha, k_1, .., k_5\}, \{\mathbf{r}, \mathbf{t}\}$
- Non-linear least squares (Gauss-Newton)

Results: Checkerboard Matching



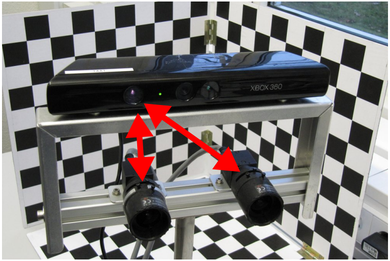
Results: Corner Detection

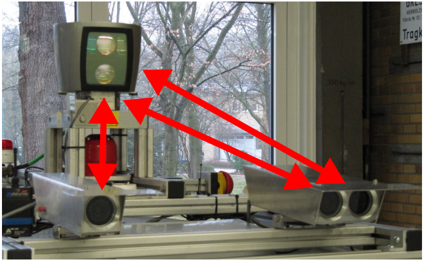


Precision-Recall computed from

- 150 images taken under various conditions
- 80.000 checkerboard corners

3D Range Sensor Registration





Goals

1. Calibrate cameras intrinsically and extrinsically
2. Register range sensor to cameras

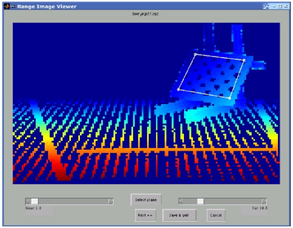
3D Range Sensor: Related Work

[Unnikrishnan and Hebert 2005]

- Interactive GUI for laser-to-camera registration
- User marks calibration object manually

[Scaramuzza, Harati, Siegwart 2007]

- Registration of omnidirectional camera and laserscanner
- Manual correspondence selection required



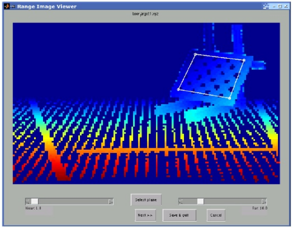
3D Range Sensor: Related Work

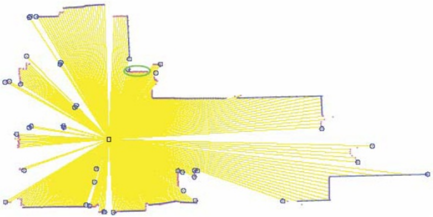
[Unnikrishnan and Hebert 2005]

- Interactive GUI for laser-to-camera registration
- User marks calibration object manually

[Scaramuzza, Harati, Siegwart 2007]

- Registration of omnidirectional camera and laserscanner
- Manual correspondence selection required

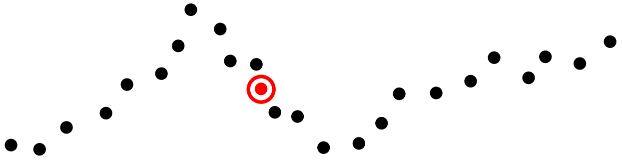




3D Range Sensor: Segmentation

Segments point cloud into planar pieces

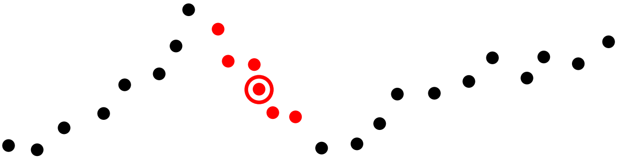
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

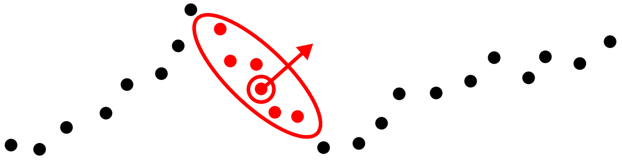
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

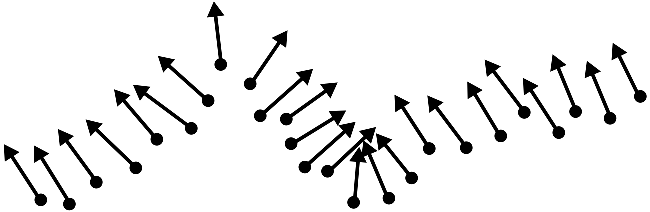
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

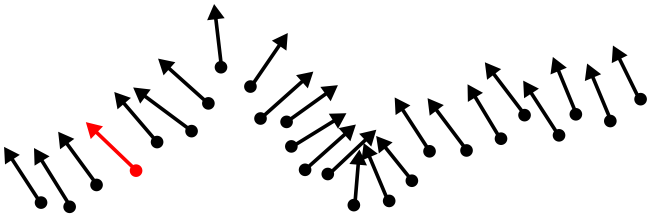
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

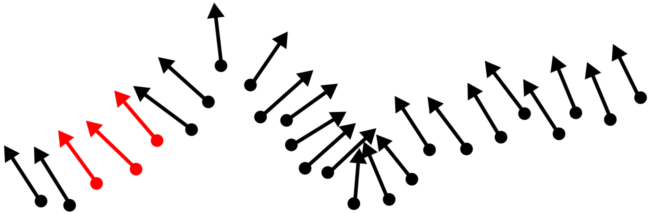
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

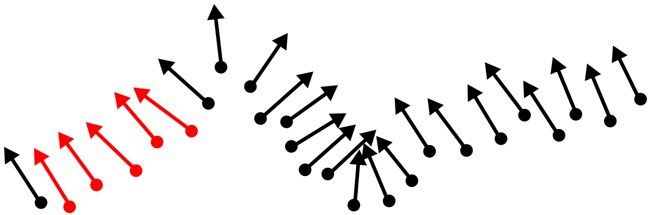
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

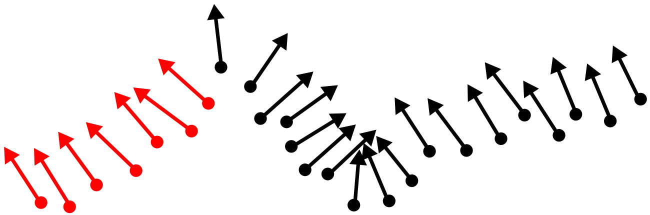
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

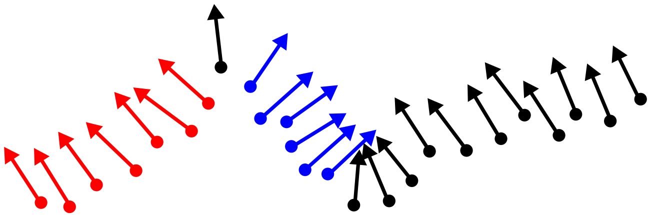
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

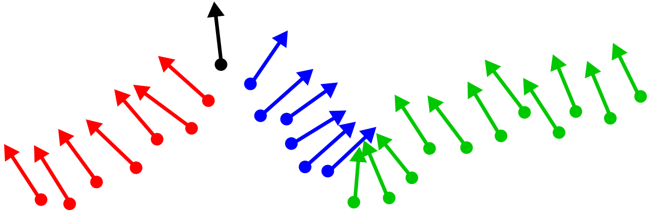
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

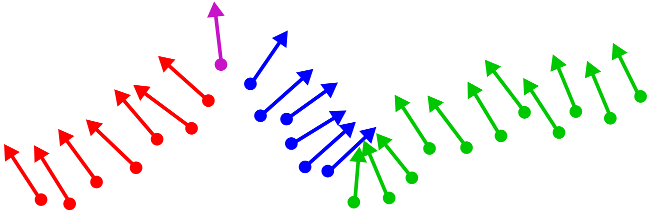
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

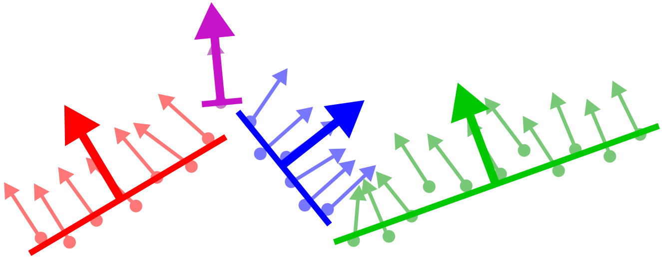
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

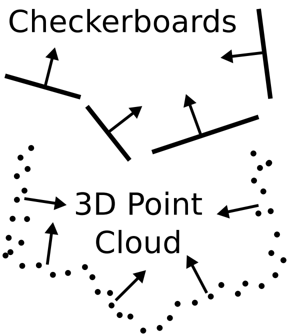
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation $\mathbf{R}$
- Estimate optimal translation $\mathbf{t}$ (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



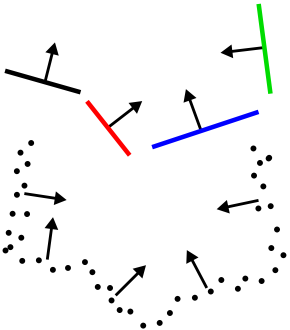
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \| p_i - \mathcal{N}(p_i) \|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



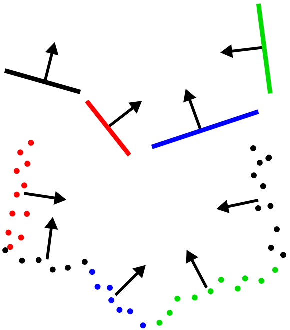
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



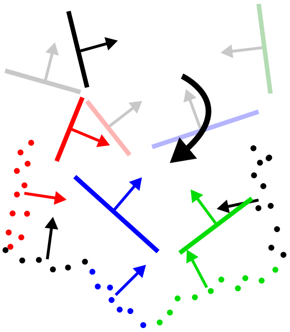
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



Refine best hypotheses using robust Point-to-Point ICP

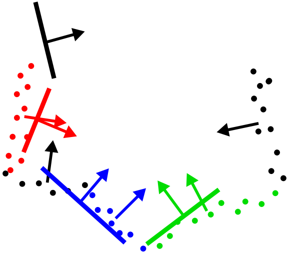
- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

KIT Karlsruhe Institute of Technology

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



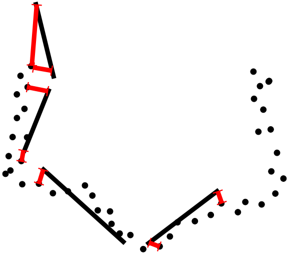
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



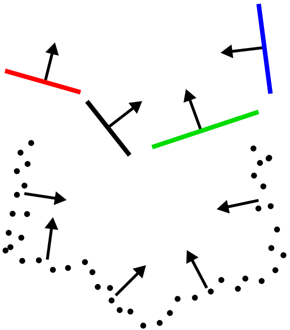
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



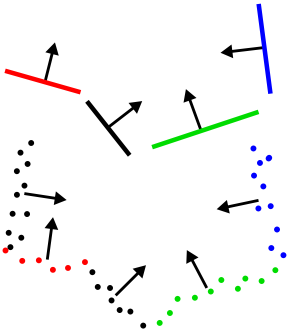
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



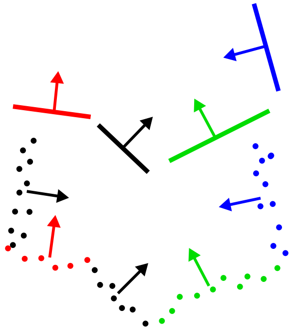
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



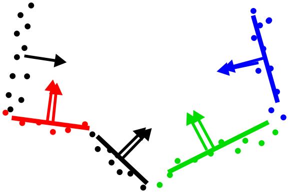
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



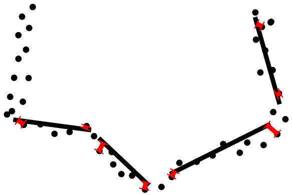
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \| p_i - \mathcal{N}(p_i) \|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



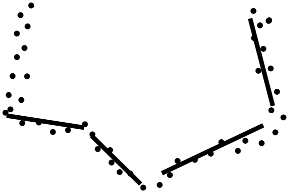
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

3D Range Sensor: Registration

Sample a set of plausible hypotheses

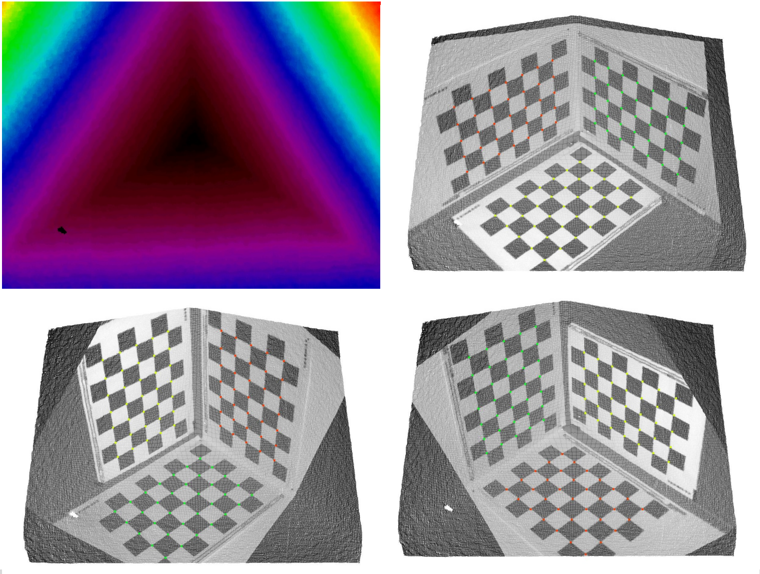
- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t** (minimize center to plane)
- **Score** hypothesis using euclidean distance
- **Repeat** until with probability p the correct solution has been found



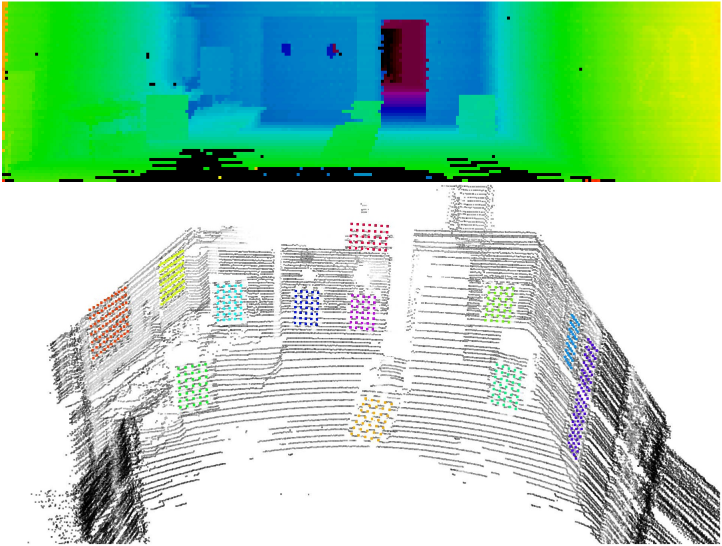
Refine best hypotheses using robust Point-to-Point ICP

- Minimize: $\sum_i w_i \|p_i - \mathcal{N}(p_i)\|_2$ with $w_i \in \{0, 1\}$

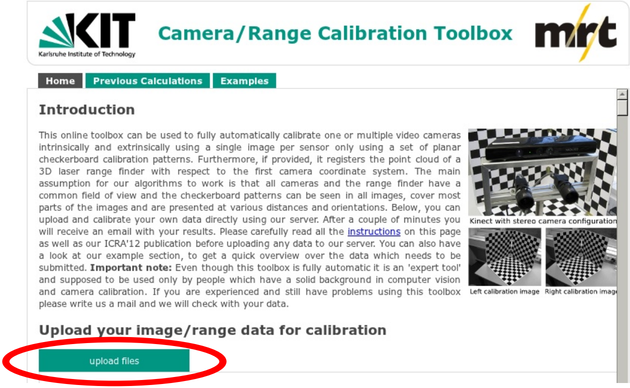
3D Range Sensor: Results

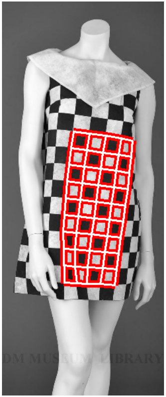


3D Range Sensor: Results



Thank you!





www.cvlibs.net