

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

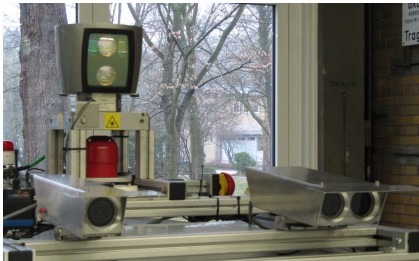


Motivation





Cameras / Microsoft Kinect



Cameras / Velodyne HDL-64

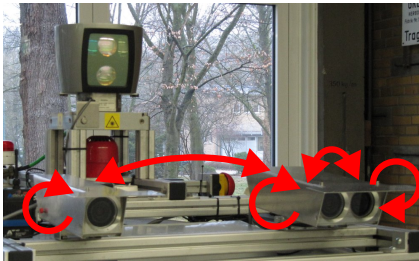
Setup: Video cameras (+ range sensor)

Goals

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



Cameras / Microsoft Kinect

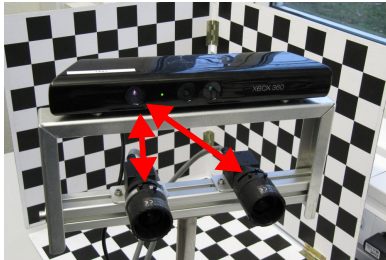


Cameras / Velodyne HDL-64

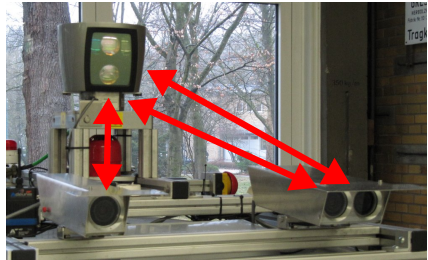
Setup: Video cameras (+ range sensor)

Goals

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



Cameras / Microsoft Kinect



Cameras / Velodyne HDL-64

Setup: Video cameras (+ range sensor)

Goals

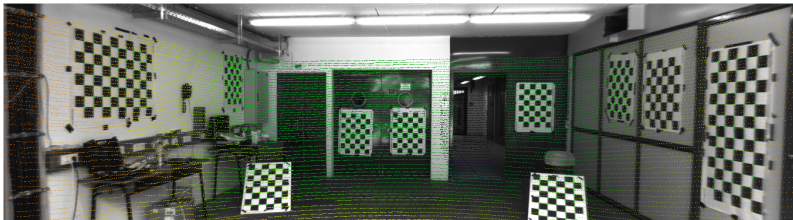
- 1 Calibrate cameras intrinsically and extrinsically
- 2 Register range sensor to cameras

■ 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)

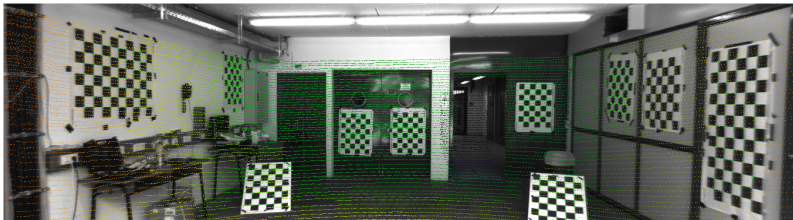


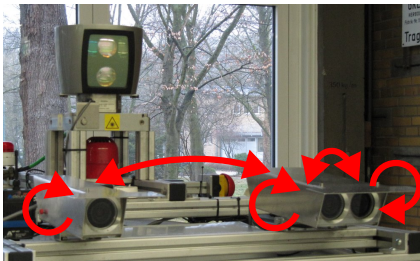
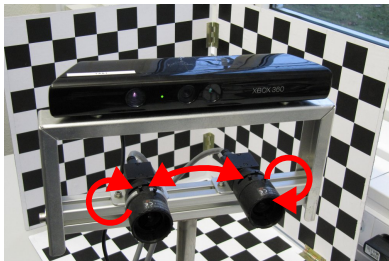
■ Contributions

- 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)

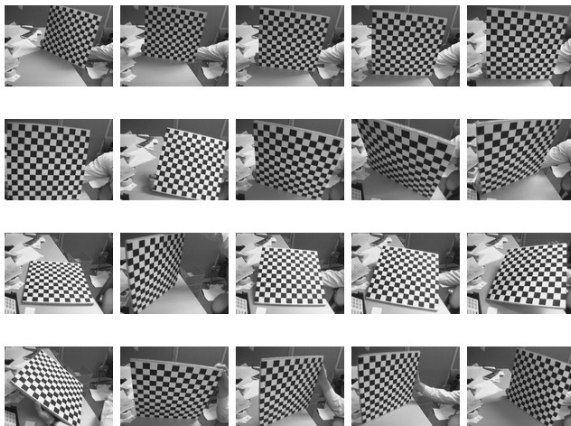




Goals

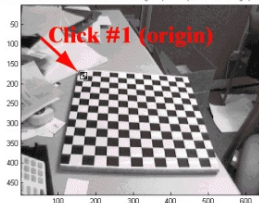
- 1 Calibrate cameras intrinsically and extrinsically
- 2 Register range sensor to cameras

Bouget's Camera Calibration Toolbox for Matlab / OpenCV



Bouget's Camera Calibration Toolbox for Matlab / OpenCV

Click on the four extreme corners of the rectangular pattern (first corner = origin) ... Image 1



Click on the four extreme corners of the rectangular pattern (first corner = origin) ... Image 1



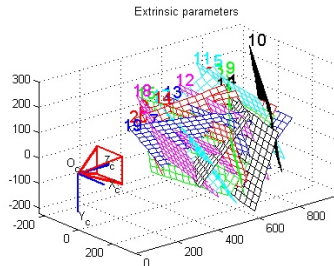
Click on the four extreme corners of the rectangular pattern (first corner = origin) ... Image 1



Click on the four extreme corners of the rectangular pattern (first corner = origin) ... Image 1

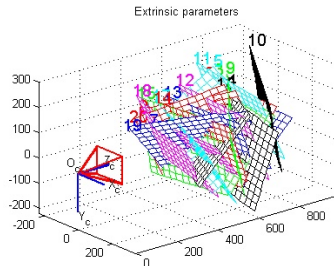


Bouget's Camera Calibration Toolbox for Matlab / OpenCV



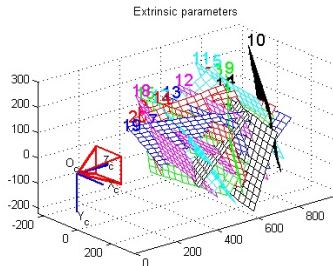
- Requires ≥ 10 synchronized images
- No automatic corner detection and matching
- Time consuming

Bouget's Camera Calibration Toolbox for Matlab / OpenCV



- Requires ≥ 10 synchronized images
- No automatic corner detection and matching
- Time consuming

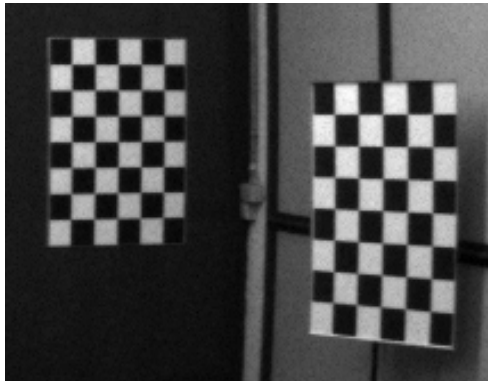
Bouget's Camera Calibration Toolbox for Matlab / OpenCV



- Requires ≥ 10 synchronized images
- No automatic corner detection and matching
- Time consuming

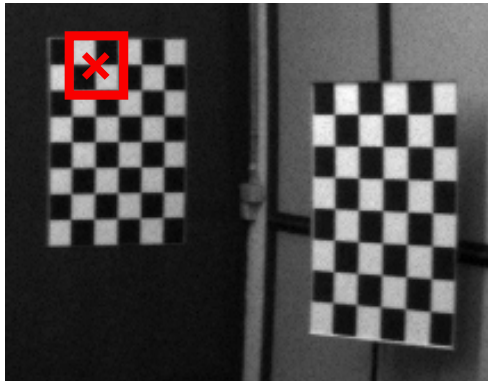
Corner detection and subpixel refinement

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



Corner detection and subpixel refinement

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



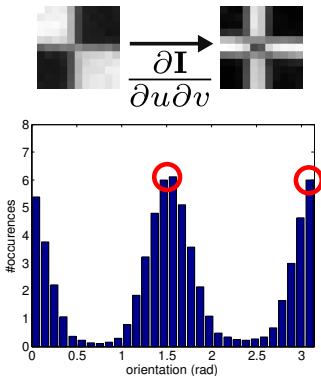
Corner detection and subpixel refinement

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



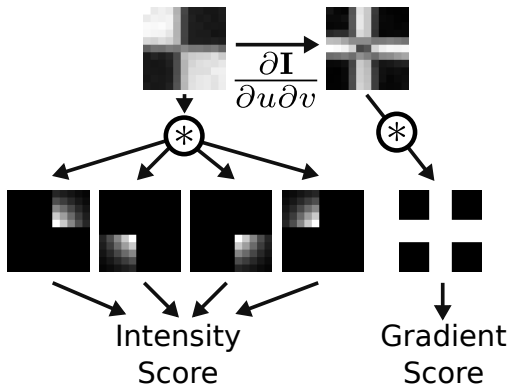
Corner detection and subpixel refinement

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



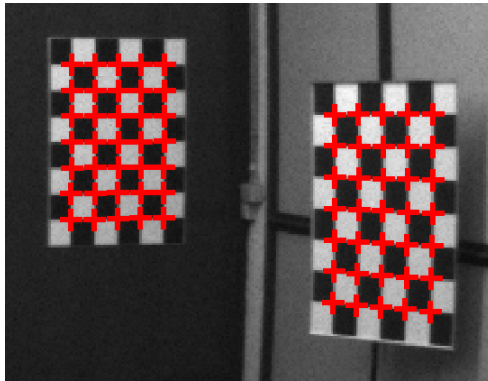
Corner detection and subpixel refinement

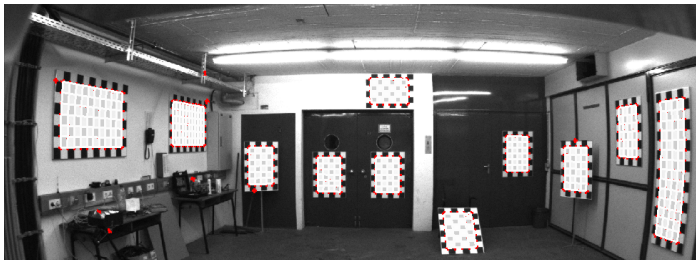
- Compute cornerness score for each pixel
- Non-maximum suppression, sub-pixel refinement



Corner detection and subpixel refinement

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





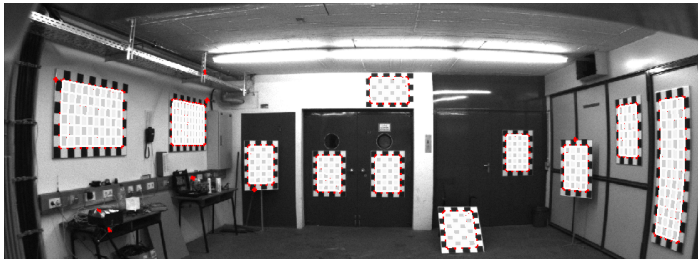
Parameters

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

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

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

Camera: Finding Checkerboards

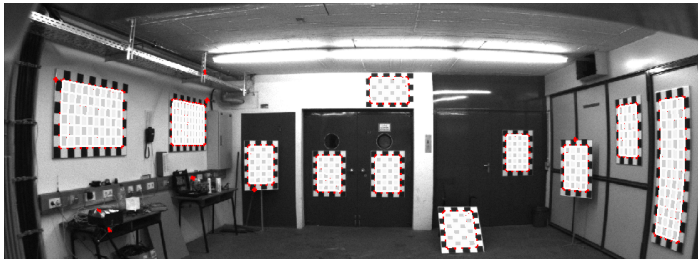


Parameters

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

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

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



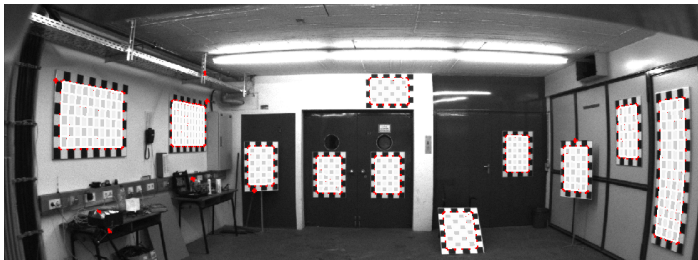
Parameters

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

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

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

Camera: Finding Checkerboards

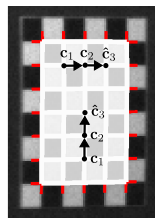


Parameters

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

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

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



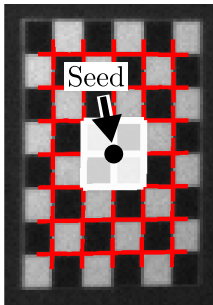
Collinearity

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

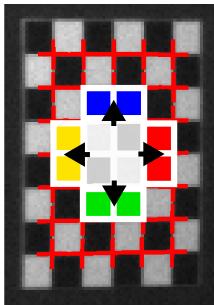


Camera: Finding Checkerboards

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

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

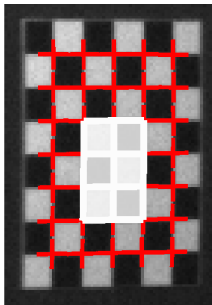


Camera: Finding Checkerboards

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

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

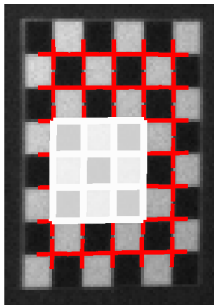


Camera: Finding Checkerboards

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

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

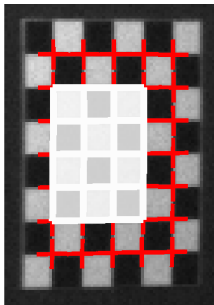


Camera: Finding Checkerboards

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

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

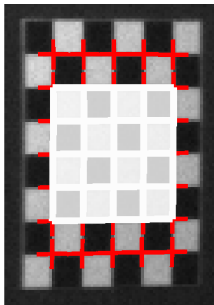


Camera: Finding Checkerboards

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

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

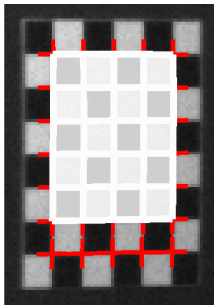


Camera: Finding Checkerboards

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

For each corner as seed do:

- Incrementally add neighboring corners with lowest energy

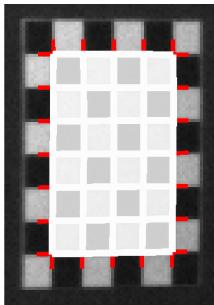


Camera: Finding Checkerboards

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

For each corner as seed do:

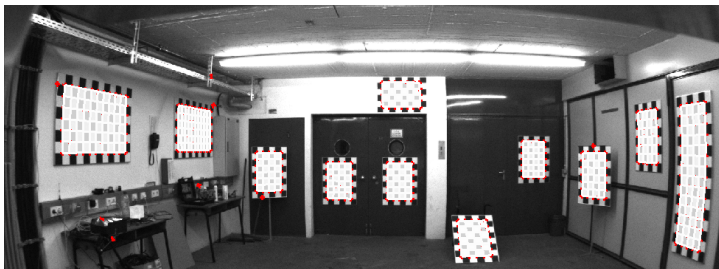
- Incrementally add neighboring corners with lowest energy



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

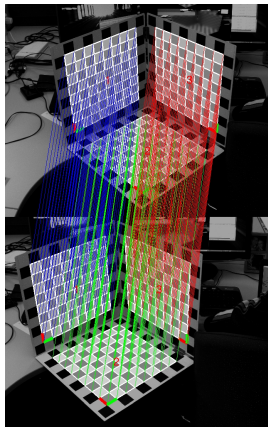


Match checkerboards across images

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

Parameter Optimization

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

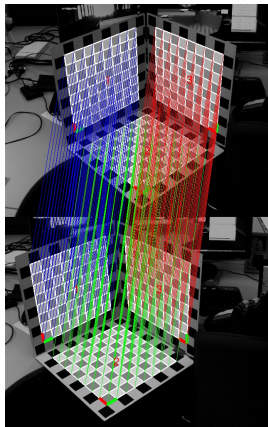


Match checkerboards across images

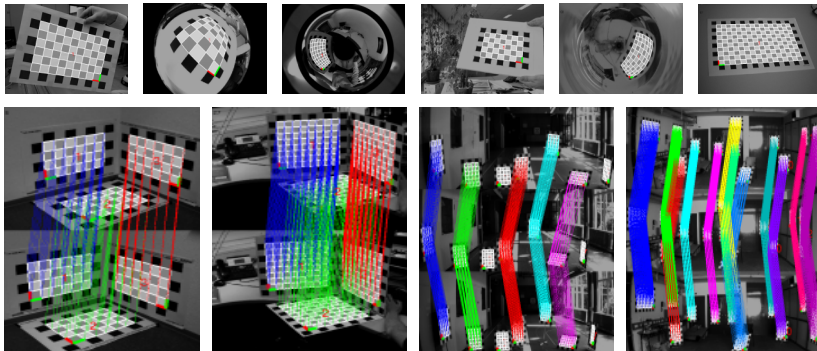
- Sample 2 checkerboards per image
- Compute similarity transformation from center of checkerboards
- Maximize number of inliers

Parameter Optimization

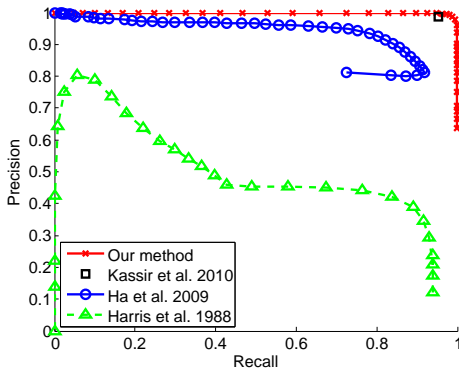
- Parameters: $\{\mathbf{f}, \mathbf{c}, \alpha, k_1, \dots, k_5\}, \{\mathbf{r}, \mathbf{t}\}$
- Non-linear least squares (Gauss-Newton)



Results: Checkerboard Matching



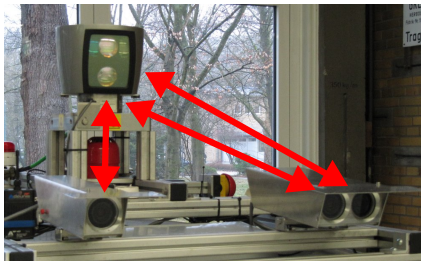
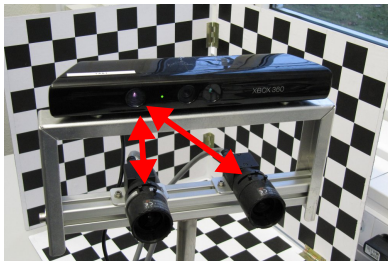
Results: Corner Detection



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

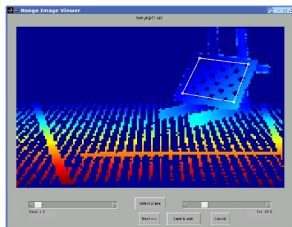
3D Range Sensor: Related Work

[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



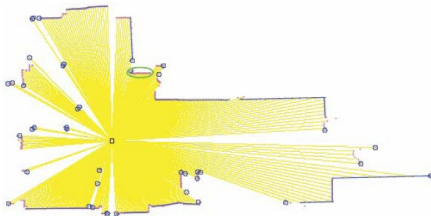
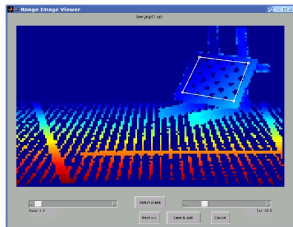
3D Range Sensor: Related Work

[Unnikrishnan and Hebert 2005]

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

[Scaramuzza, Harati, Siegwart 2007]

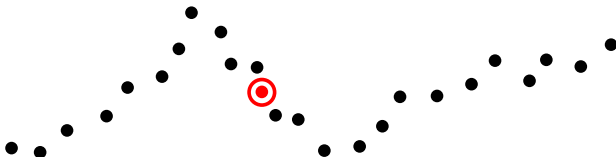
- Registration of omnidirectional camera and laserscanner
- Manual correspondence selection required



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

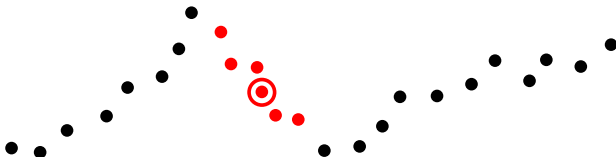
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

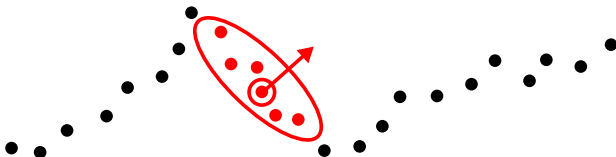
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

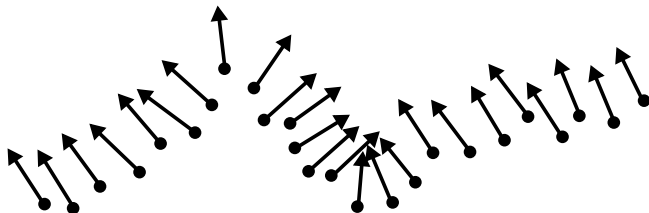
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

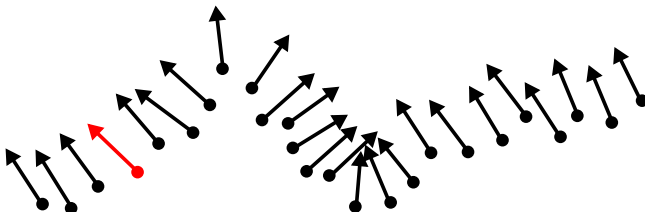
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

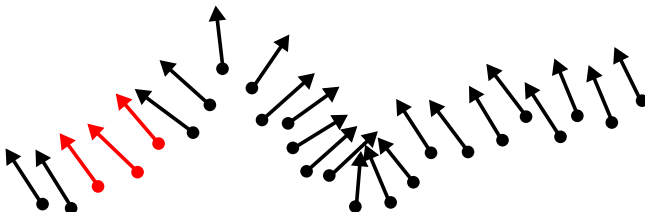
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

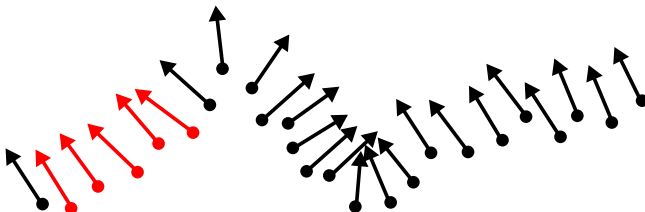
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

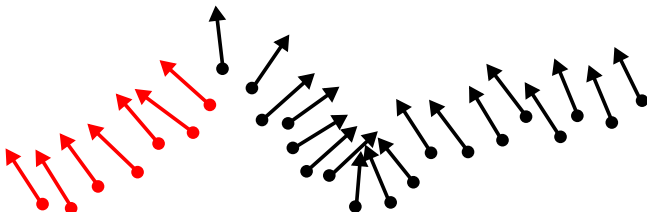
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

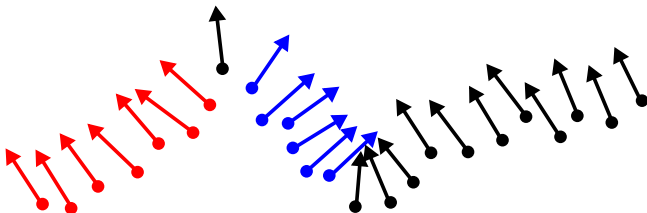
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

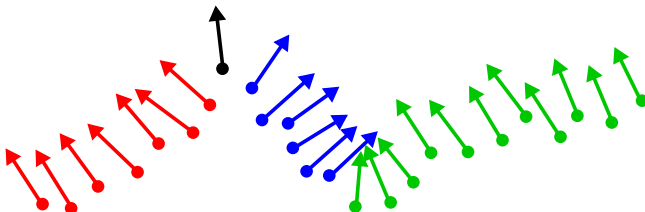
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

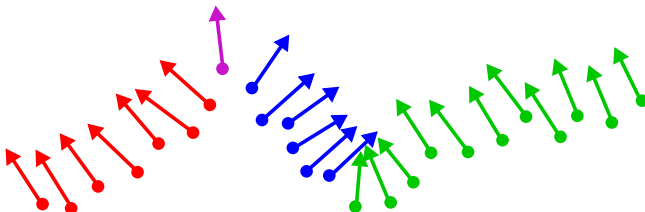
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

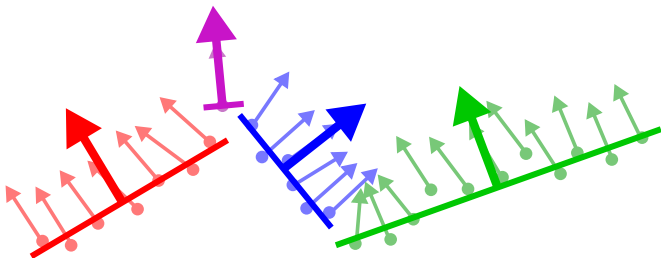
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Segmentation

Segments point cloud into planar pieces

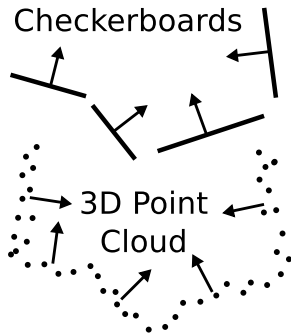
- Compute normal vector $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathbb{R}^3$ by principal component analysis on k-nearest neighbors
- Grow regions from random seeds
- Stop growing when normal gets too dissimilar from seed
- Repeat until segmentation covers all points



3D Range Sensor: Registration

Sample a set of plausible hypotheses

- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **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



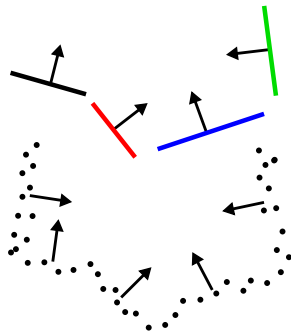
Refine best hypotheses using robust Point-to-Point ICP

- 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



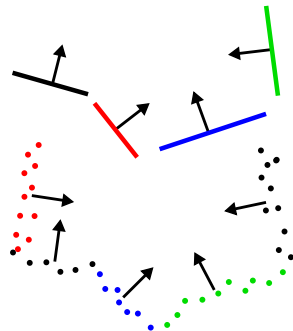
Refine best hypotheses using robust Point-to-Point ICP

- 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



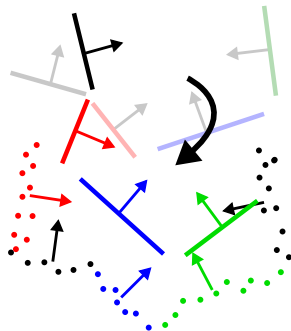
Refine best hypotheses using robust Point-to-Point ICP

- 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



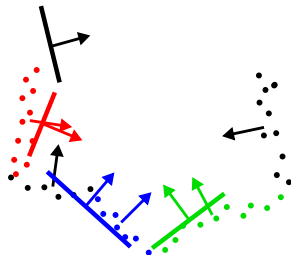
Refine best hypotheses using robust Point-to-Point ICP

- 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



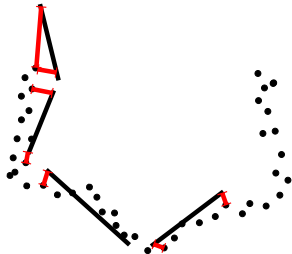
Refine best hypotheses using robust Point-to-Point ICP

- 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



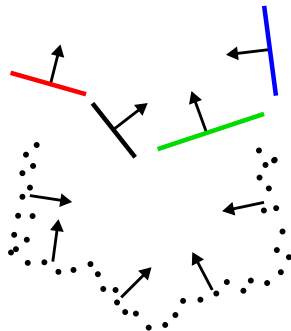
Refine best hypotheses using robust Point-to-Point ICP

- 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



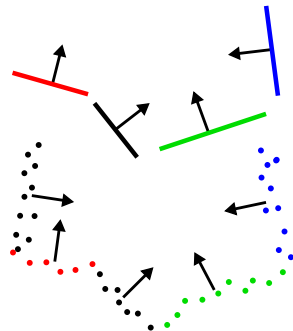
Refine best hypotheses using robust Point-to-Point ICP

- 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



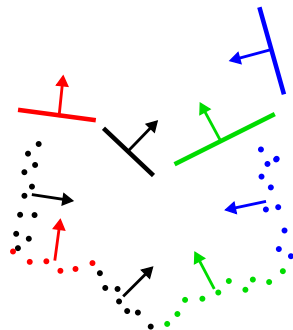
Refine best hypotheses using robust Point-to-Point ICP

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

3D Range Sensor: 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

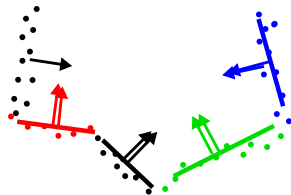


Refine best hypotheses using robust Point-to-Point ICP

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

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

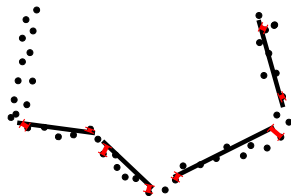


Refine best hypotheses using robust Point-to-Point ICP

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

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

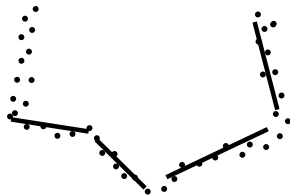


Refine best hypotheses using robust Point-to-Point ICP

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

Sample a set of plausible hypotheses

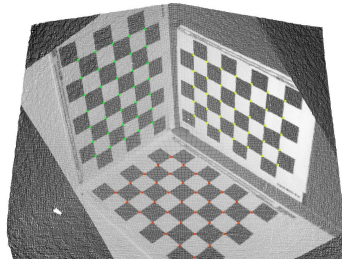
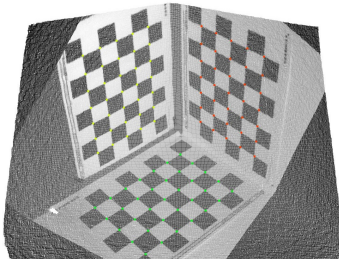
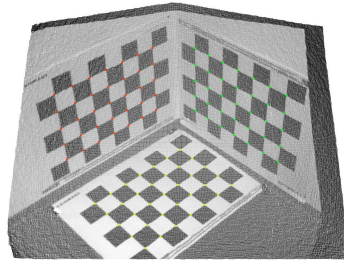
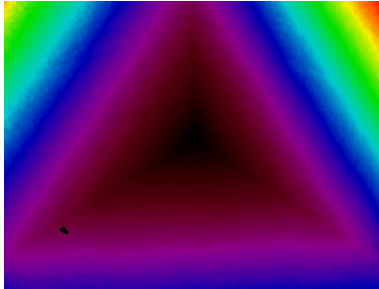
- Draw 3 **checkerboards**
- Draw 3 **segments**
- Estimate optimal rotation **R**
- Estimate optimal translation **t**
(minimize center to plane)
- **Score** hypothesis using
euclidean distance
- **Repeat** until with probability p the
correct solution has been found



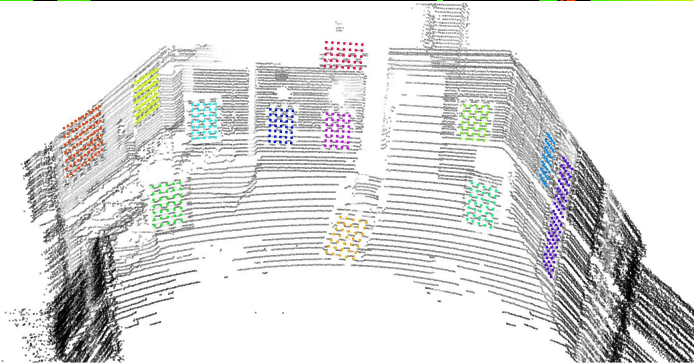
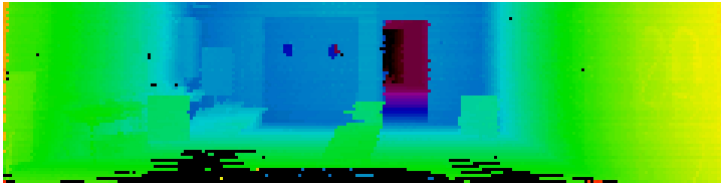
Refine best hypotheses using robust Point-to-Point ICP

- 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!

[Home](#)[Previous Calculations](#)[Examples](#)

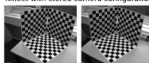
Introduction

This online toolbox can be used to fully automatically calibrate one or multiple video cameras intrinsically and extrinsically using a single image per sensor only using a set of planar checkerboard calibration patterns. Furthermore, if provided, it registers the point cloud of a 3D laser range finder with respect to the first camera coordinate system. The main assumption for our algorithms to work is that all cameras and the range finder have a common field of view and the checkerboard patterns can be seen in all images, cover most parts of the images and are presented at various distances and orientations. Below, you can upload and calibrate your own data directly using our server. After a couple of minutes you will receive an email with your results. Please carefully read all the [Instructions](#) on this page as well as our ICRA'12 publication before uploading any data to our server. You can also have a look at our example section, to get a quick overview over the data which needs to be submitted. **Important note:** Even though this toolbox is fully automatic it is an 'expert tool' and supposed to be used only by people which have a solid background in computer vision and camera calibration. If you are experienced and still have problems using this toolbox please write us a mail and we will check with your data.

Upload your image/range data for calibration

[upload files](#)

Kinect with stereo camera configuration



Left calibration image Right calibration image



www.cvlibs.net