Automatic Camera and Range Sensor Calibration using a single Shot

Andreas Geiger, Frank Moosmann, Ömer Car and Bernhard Schuster

***Abstract*— As a core robotic and vision problem, camera and range sensor calibration have been researched intensely over the last decades. However, robotic research efforts still often get heavily delayed by the requirement of setting up a calibrated system consisting of multiple cameras and range measurement units. With regard to removing this burden, we present a toolbox with web interface for fully automatic camera-to-camera and camera-to-range calibration. Our system is easy to setup and recovers intrinsic and extrinsic camera parameters as well as the transformation between cameras and range sensors within one minute. In contrast to existing calibration approaches, which often require user intervention, the proposed method is robust to varying imaging conditions, fully automatic, and easy to use since a single image and range scan proves sufficient for most calibration scenarios. Experimentally, we demonstrate that the proposed checkerboard corner detector significantly outperforms current state-of-the-art. Furthermore, the proposed camera-to-range registration method is able to discover multiple solutions in the case of ambiguities. Experiments using a variety of sensors such as grayscale and color cameras, the Kinect 3D sensor and the Velodyne HDL-64 laser scanner show the robustness of our method in different indoor and outdoor settings and under various lighting conditions.**

I. INTRODUCTION AND RELATED WORK

Robots are typically equipped with multiple complementary sensors, which require calibration in order to represent sensed information in a common coordinate system. Hereby, each sensor can be characterized by its *intrinsic* (e.g. shape of the camera lens) and *extrinsic* (i.e. pose) parameters. Calibration methods aim at estimating these parameters, often using checkerboard patterns as targets. Although calibration is an ubiquitous problem in robotics and computer vision, current camera calibration tools such as the widely used Matlab Camera Calibration Toolbox [1] or OpenCV [2] are not very robust and often require manual intervention, leading to a cumbersome calibration procedure. Furthermore, only little work on camera-to-range sensor calibration has been done, and to our knowledge [3] and [4] are the only toolboxes online available.

In this work, we try to close this gap by proposing a robust solution to automatically calibrating multiple cameras and 3D range sensors with respect to each other. Our approach relies on a cheap and simple calibration setup: We attach multiple printed checkerboard patterns at the walls and the floor, see Fig. 4 for an illustration. As input, our method requires a single range or camera image per sensor (which we call *shot* in the following), as well as the distance between the inner checkerboard corners for resolving the scale ambiguity.

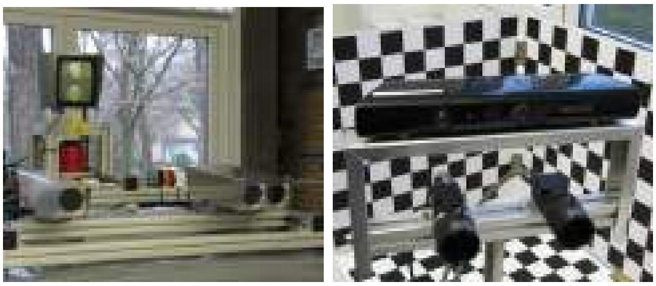


Fig. 1. **Experimental setup.** Trinocular camera with Velodyne HDL-64E laser scanner (left) and binocular camera with Microsoft Kinect (right).

Note that this differs from previous approaches [1], [2], [3], [4] which require multiple (synchronized) images of a single calibration target presented at different orientations, as well as the number of checkerboard rows and columns as input. The only assumption we make is that all sensors return either intensity or depth images and share a common field of view.

Experimentally we show the robustness of our method on calibration images from the internet, a binocular camera setup with the Kinect range sensor and a trinocular camera setup with the Velodyne HDL-64E laser scanner, illustrated in Fig. 1. We will also make an online version of our system[1] available to the community: After uploading images and 3D point clouds, the calibration parameters are computed within one minute.

This work is organized as follows: The next section starts with a discussion of related work. Sec. III and Sec. IV give a detailed description of the proposed method on camera-to-camera and camera-to-range calibration, respectively. After an evaluation in Sec. V the paper is concluded in Sec. VI.

II. RELATED WORK

A. *Camera-to-Camera Calibration*

The most widely used toolbox for camera calibration is probably Bouget's Matlab Camera Calibration Toolbox [1], of which a C++ version has been implemented in the OpenCV library [2]. It uses the distortion model described in [1], [5], [6], covering a large variety of lenses, and allows to calibrate up to two video cameras intrinsically and extrinsically by presenting a checkerboard calibration pattern at multiple orientations. The OpenCV additionally offers an automatic corner detection method based on quadrangles, which has been extended and improved upon by Rufli et al. [7]. Due to the method's sensitivity to clutter in the input image, Kassir et al. [4] propose a more robust corner detector

All authors are with the Department of Measurement and Control, Karlsruhe Institute of Technology. The first two authors contributed equally to this work. Primary contact: {geiger,frank.moosmann}@kit.edu.

[1] **www.cvlibs.net**

based on multi-scale Harris [8] points and an improved filtering mechanism inspired by [9]. From the recovered corner points, they are able to detect a single checkerboard within an image. The problem of multi-camera calibration for the distortion-free scenario has been tackled in [10].

While all of the existing methods concentrate on the problem of detecting only a single (and often known) calibration target, our approach finds multiple unknown checkerboard patterns in an image and automatically matches them between cameras. This facilitates the calibration setup as a single image is sufficient for calibration and unsynchronized sensors such as line sweep laser scanners (e.g. rotating SICK) can be readily employed. Furthermore, our method requires no manual intervention, and the proposed corner detector significantly outperforms current state-of-the art [4], [9], as illustrated by our experiments in Sec. V-A.

B. *Camera-to-Range Calibration*

For camera-to-range sensor calibration, all existing works assume a valid intrinsic calibration of the camera and focus on estimating the six parameters needed to align the two coordinate systems. The approaches can be classified into two groups, based on the type of range sensor used:

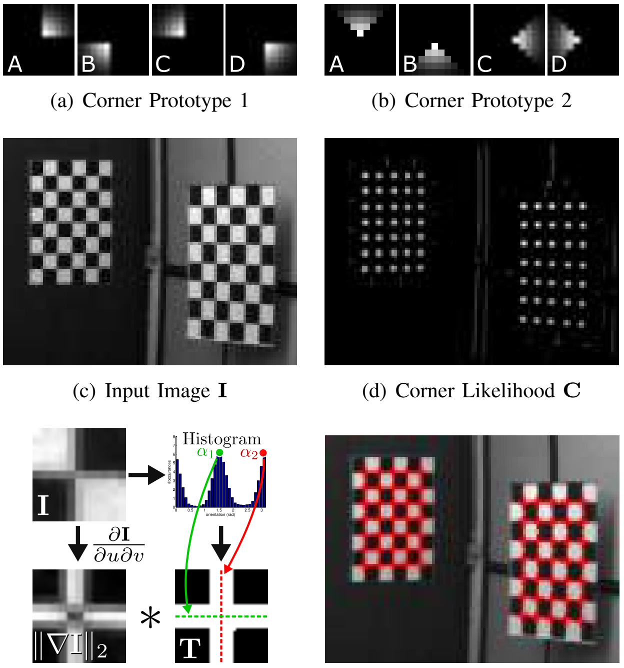
The first group of approaches uses full 3D sensors giving rise to a dense 3D point cloud. Unnikrishnan et al. [3] have developed a toolbox where the user has to mark the calibration pattern in an interactive GUI. The approach of Scaramuzza et al. [11] is comparable but specialized to omni-directional cameras. Likewise, correspondences must be selected manually.

The second and more established group uses one- or four-layer laser scanners. Since these sensors only measure a small part of the scene, most approaches rely on manual selection of correspondences [12], [13], [14]. Only recently this effort was reduced by either classifying line-segments [4] or by exploiting reflectance measurements from the lidar scanner [15]. Unfortunately, these approaches are not directly applicable to 3D range sensors.

Common to all approaches is the use of only one calibration pattern. This requires to take several recordings with the need for manually moving the calibration pattern. In this paper we propose a robust method for 3D range sensors which determines all six extrinsic parameters fully automatically using a single shot only. Hereby, calibration efforts are reduced dramatically.

III. CAMERA-TO-CAMERA CALIBRATION

Following [16], we use planar checkerboard patterns as calibration targets for multiple reasons: They are cheap to employ, corners can be localized with high sub-pixel accuracy and structure recovery profits from strong edges between corners. In contrast to existing methods which capture multiple images of a single checkerboard, we take a single shot of multiple checkerboards placed at different locations in the image. This also facilitates the calibration process, especially with respect to registering the triangulated checkerboards to a 3D point cloud, as detailed in section IV.


A
B
C
D
A
B
C
D
I
Histogram
α_1
α_2
$\frac{\partial \mathbf{I}}{\partial u \partial v}$
$\|\nabla \mathbf{I}\|_2$
$*$
T

(a) Corner Prototype 1 (b) Corner Prototype 2

(c) Input Image **I** (d) Corner Likelihood **C**

(e) Orientation & Score (f) Detected Checkerboard Corners

Fig. 2. **Corner detection.** We filter the input image **I** using corner prototypes, apply non-maxima-suppression on the resulting corner likelihood **C** and verify corners by their gradient distribution. See Sec. III-A for details.

Through not strictly necessary, we assume the prevalent calibration scenario where all sensors have a part of their field of view in common. Our camera calibration algorithm proceeds as follows: First, we robustly locate checkerboard corners in the image (Sec. III-A) and refine them for sub-pixel accuracy (Sec. III-B). Checkerboard structures are recovered by optimizing an energy function subject to structural constraints (Sec. III-C). Correspondences between images are obtained by deterministically sampling affine transformations and maximizing a fitness function (Sec. III-D). The final camera parameters are recovered via non-linear optimization, where we use weak regularizers to avoid degenerate solutions (Sec. III-E). The following subsections give details about each of these steps.

A. *Corner Detection*

Harris points [8] or Shi-Tomasi corners [17] are a common choice for localizing junctions in an image. However, we found that the following procedure gives more robust results with respect to image clutter, blurring artifacts and localization accuracy: In order to locate checkerboard corners in a grayscale image **I** (Fig. 2(c)), we compute a corner likelihood at each pixel in the image using two different $n \times n$ corner prototypes: One for axis-aligned corners (Fig. 2(a)) and one for corners, which are rotated by 45° (Fig. 2(b)). Empirically, we found that these two simple prototypes are sufficient for detecting corners over a wide range of distortions induced by perspective transformations. Each prototype is composed of four filter kernels $\{A, B, C, D\}$, which are convolved with the input image **I**. For an ideal corner, the response of $\{A, B\}$

should be greater than the mean response of $\{A, B, C, D\}$, while the response of $\{C, D\}$ should be smaller, and vice versa for flipped corners. This fact can be expressed formally as follows: Let f_X^i be the filter response of kernel X and prototype i for a particular pixel. The corner likelihood c at this pixel is defined by taking the maximum over all combinations of prototypes and flippings:

$$\begin{aligned} c &= \max(s_1^1, s_2^1, s_1^2, s_2^2) \\ s_1^i &= \min(\min(f_A^i, f_B^i) - \mu, \mu - \min(f_C^i, f_D^i)) \\ s_2^i &= \min(\mu - \min(f_A^i, f_B^i), \min(f_C^i, f_D^i) - \mu) \\ \mu &= 0.25\ (f_A^i + f_B^i + f_C^i + f_D^i) \end{aligned} \tag{1}$$

Here, s_1^i and s_2^i denote the likelihood of the two possible flippings for prototype i. Computing this likelihood for every pixel in the image yields a corner likelihood map $\mathbf{C}$. See Fig. 2(d) for an illustration. Importantly, note that the above definition leads to a low likelihood c, if any of the four filter kernels responds weakly. This is important for removing as many non-checkerboard style corners as possible from the hypotheses space. To produce a list of corner candidates, we apply conservative non-maxima-suppression (with parameters n_{nms} and τ_{nms}) [18] on $\mathbf{C}$, followed by verifying the candidates by their gradient statistics in a local $n \times n$ pixel neighborhood, as illustrated in Fig. 2(e): We compute a weighted orientation histogram (32 bins) from Sobel filter responses and find the two dominant modes α_1 and α_2 using mean shift [19]. Based on the edge orientations, we construct a template $\mathbf{T}$ for the expected gradient strength $\|\nabla \mathbf{I}\|_2$. The product of $\mathbf{T} * \|\nabla \mathbf{I}\|_2$ and the corner likelihood in (1) gives the corner score, which we threshold by τ_{corner} for obtaining a final list of corner candidates, see Fig. 2(f). Here, '$*$' denotes the normalized cross-correlation operator.

B. Sub-pixel Corner and Orientation Refinement

It is well-known that calibration benefits from sub-pixel accurate corner locations [20], [21], [2]. In this work we refine both, the corner location and the edge orientations.

For sub-pixel corner localization, we make use of the fact that at a corner location $\mathbf{c} \in \mathbb{R}^2$ the image gradient $\mathbf{g_p} \in \mathbb{R}^2$ at a neighboring pixel $\mathbf{p} \in \mathbb{R}^2$ should be approximately orthogonal to $\mathbf{p} - \mathbf{c}$, leading to the optimization problem

$$\mathbf{c} = \arg\min_{\mathbf{c}'} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{I}(\mathbf{c}')} \left(\mathbf{g}_\mathbf{p}^T (\mathbf{p} - \mathbf{c}')\right)^2 \tag{2}$$

where $\mathcal{N}_\mathbf{I}$ is a local 11×11 pixel neighborhood around the corner candidate. Note that neighboring pixels are automatically weighted by the gradient magnitude. This problem is straightforward to solve in closed form, yielding the solution:

$$\mathbf{c} = (\sum_{\mathbf{p} \in \mathcal{N}_\mathbf{I}} \mathbf{g_p} \mathbf{g_p}^T)^{-1} \sum_{\mathbf{p} \in \mathcal{N}_\mathbf{I}} (\mathbf{g_p} \mathbf{g_p}^T) \mathbf{p} \tag{3}$$

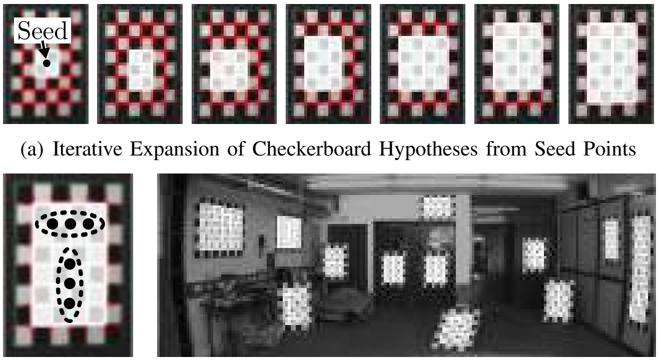
To refine the edge orientation vectors $\mathbf{e}_1 \in \mathbb{R}^2$ and $\mathbf{e}_2 \in \mathbb{R}^2$, we seek to minimize the error in deviation of their normals with respect to the image gradients

$$\mathbf{e}_i = \arg\min_{\mathbf{e}'_i} \sum_{\mathbf{p} \in \mathcal{M}_i} (\mathbf{g}_\mathbf{p}^T \mathbf{e}'_i)^2 \quad \text{s.t.} \quad \mathbf{e}'^T_i \mathbf{e}'_i = 1 \tag{4}$$

where $\mathcal{M}_i = \{\mathbf{p} \mid \mathbf{p} \in \mathcal{N}_\mathbf{I} \wedge |\mathbf{m}_i^T \mathbf{g_p}| < 0.25\}$ is the set of neighboring pixels, which are aligned with the gradient $\mathbf{m}_i = [\cos(\alpha_i) \sin(\alpha_i)]^T$ of mode i. The solution to Eq. 4 is obtained, by setting the derivative of its Lagrangian to zero, leading to an eigenvalue problem, with $\mathbf{e}_i$ being the eigenvector corresponding to the smallest eigenvalue of

$$\sum_{\mathbf{p} \in \mathcal{M}_i} \begin{pmatrix} g_\mathbf{p}^1 \mathbf{g}_\mathbf{p}^T \\ g_\mathbf{p}^2 \mathbf{g}_\mathbf{p}^T \end{pmatrix} \in \mathbb{R}^{2 \times 2} \tag{5}$$

where $g_\mathbf{p}^i$ denotes the i'th entry of $\mathbf{g_p}$.



(a) Iterative Expansion of Checkerboard Hypotheses from Seed Points

(b) Triples

(c) Detected Checkerboards on Large-Scale Example

Fig. 3. **Structure recovery.** We iteratively expand seed points (a) into the direction of the strongest gradient of an energy function evaluating the local structuredness (b). Fig. (c) shows final detection result.

C. Structure Recovery

Let the set of corner candidates be $\mathcal{X} = \{\mathbf{c}_1, .., \mathbf{c}_N\}$ and let $\mathcal{Y} = \{\mathbf{y}_1, .., \mathbf{y}_N\}$ be the corresponding set of labels. Here, $\mathbf{y} \in \{\mathcal{O}\} \cup \mathbb{N}^2$ represents either an outlier detection ($\mathcal{O}$) or the row / column ($\mathbb{N}^2$) within the checkerboard. For all checkerboards present in the image our goal is to recover $\mathcal{Y}$ given $\mathcal{X}$. We do this by minimizing the energy function

$$E(\mathcal{X}, \mathcal{Y}) = E_{corners}(\mathcal{Y}) + E_{struct}(\mathcal{X}, \mathcal{Y}) \tag{6}$$

subject to the constraint, that no two labels can explain the same checkerboard corner. Intuitively, we try to explain as many corners as possible using a regular structured element (the checkerboard): $E_{corners}(\mathcal{Y}) = -|\{\mathbf{y}|\mathbf{y} \neq \mathcal{O}\}|$ is taken as the negative number of explained checkerboard corners and E_{struct} measures how well two neighboring corners i and j are able to predict a third one k, weighted by the number of explained corners:

$$E_{struct}(\mathcal{X}, \mathcal{Y}) = |\{\mathbf{y}|\mathbf{y} \neq \mathcal{O}\}| \max_{(i,j,k) \in \mathcal{T}} \frac{||\mathbf{c}_i + \mathbf{c}_k - 2\mathbf{c}_j||_2}{||\mathbf{c}_i - \mathbf{c}_k||_2} \tag{7}$$

Here, $\mathcal{T}$ denotes the set of all row and column triples of the current checkerboard configuration induced by $\mathcal{Y}$, see Fig. 3(b) for an illustration. In total, we have $|\mathcal{T}| = m(n-2) + n(m-2)$ triples, with m and n denoting the number of rows and columns respectively. Importantly note that due to the local nature of our linearity requirement in Eq. 7, we gain flexibility and also allow for strongly distorted patterns as imaged by fisheye lenses, for instance.

Since the set of possible states $\mathcal{Y}$ can take is exponentially large in N, exhaustive search is intractable. Instead, we employ a simple discrete optimization scheme, which works well in practice as confirmed by our experiments in Sec. V: Given a seed corner, we search for its closest neighbors in the direction of its edges $\mathbf{e}_1$ and $\mathbf{e}_2$, yielding an initial 2×2 checkerboard hypothesis with an associated energy value $E(\mathcal{X}, \mathcal{Y})$. To optimize $E(\mathcal{X}, \mathcal{Y})$, we propose *expansion moves* on $\mathcal{Y}$, which expand any of the checkerboard borders by a single row or column. Amongst all four possibilities, we select the proposal, which reduces $E(\mathcal{X}, \mathcal{Y})$ the most. Fig. 3(a) illustrates the expansion moves exemplarily.

In order to recover multiple checkerboards in a single image, we repeat the aforementioned procedure for every corner in the image as seed, yielding a set of overlapping checkerboards. Duplicates (having at least one corner index in common) are removed greedily by keeping only the top scoring candidates with respect to $E(\mathcal{X}, \mathcal{Y})$, starting with the highest scoring checkerboard first. Fig. 3(c) shows an image with 11 discovered checkerboards exemplarily.

D. *Matching Checkerboards between Images*

Having recovered checkerboards in all camera images, we are left with the problem of finding corner correspondences between cameras. We do this by defining one camera as the reference camera, and independently match all other camera images against this reference image. Due to appearance ambiguities, classical feature descriptors such as SIFT [22] or SURF [23] can not be employed. Instead, we consider all possible combinations of two checkerboards in both images (resulting in a loop over four variables), from which we compute the corresponding unique 2D similarity transformation $\varphi(\mathbf{p}; \mathbf{A}, \mathbf{b}) = \mathbf{A}\mathbf{p}+\mathbf{b}$. Here, φ takes a point $\mathbf{p}$ in the target image and projects it into the reference image by changing translation, rotation and scale. Given φ, we assign all checkerboards of the target image to their closest neighbors in the reference image and resolve the two (rectangular pattern) / four (quadratic pattern) fold checkerboard rotation ambiguity by taking the minimizer of the corner projection errors independently for each matched checkerboard. Here, we only assign checkerboards, which agree in the number of rows *and* columns and for which the relative projection error is smaller than τ_{match}, measured relative to the image width. From all samples, we choose the final solution as the one which maximizes the number of matched checkerboards. Fig. 8 shows the final matching results for different scenarios.

E. *Optimization*

Following [1], [2], we assume a pin-hole camera model with radial and tangential lens distortion as described in [1], [5], [6]. In total we have 10 intrinsic $(f_u, f_v, c_u, c_v, \alpha, k_1, k_2, k_3, k_4, k_5)$ and 6 extrinsic parameters for each camera / reference camera combination. For optimization, we extended the Matlab Camera Calibration Toolbox [1] to handle an arbitrary number of cameras. We initialize the intrinsic parameters of each camera independently by exhaustively searching for f_u and f_v, placing the principal point (c_u, c_v) at the center of the image and setting $\alpha = k_1 = k_2 = k_3 = k_4 = k_5 = 0$. In practice we found this procedure to yield more robust results than closed-form solutions, such as the one proposed by Zhang et al. [24], especially in the presence of only a small number of checkerboards. The extrinsic parameters are initialized by averaging the checkerboard-to-image plane homographies [24]. To carry out the final non-linear refinement, we minimize the sum of squared corner reprojection errors using Gauss-Newton optimization. Since we found the sixth order radial distortion coefficient k_5 hard to observe, we add a quadratic regularization term to prevent k_5 from getting too large. Alternatively, individual distortion parameters can be fixed to 0 in our toolbox.

IV. CAMERA-TO-RANGE CALIBRATION

Goal of this section is to estimate the 6-DOF rigid transformation parameters $\boldsymbol{\theta} = (r_x, r_y, r_z, t_x, t_y, t_z)^T$ specifying the relative pose of the reference camera coordinate system wrt. the coordinate system of a range sensor. A 3D point in camera coordinates $\mathbf{p}_c$ can then be transformed into range sensor coordinates via $\mathbf{p}_r = \mathbf{R}_{\boldsymbol{\theta}} \cdot \mathbf{p}_c + \mathbf{t}_{\boldsymbol{\theta}}$ using the corresponding rotation matrix $\mathbf{R}_{\boldsymbol{\theta}}$ and translation vector $\mathbf{t}_{\boldsymbol{\theta}}$.

Input from the range sensor is an unordered set of 3D points $\mathcal{P}_r = \{(x, y, z)\}$ which keeps the approach as generic as possible. Similarly, we obtain sets of 3D points $\mathcal{P}_c^j = \{(x, y, z)\}$ from the camera-to-camera calibration stage, where each set j corresponds to one checkerboard and comprises the corner locations in 3D. In the following, we describe how both sets of points can be aligned, giving rise to the transformation parameters $\boldsymbol{\theta}$.

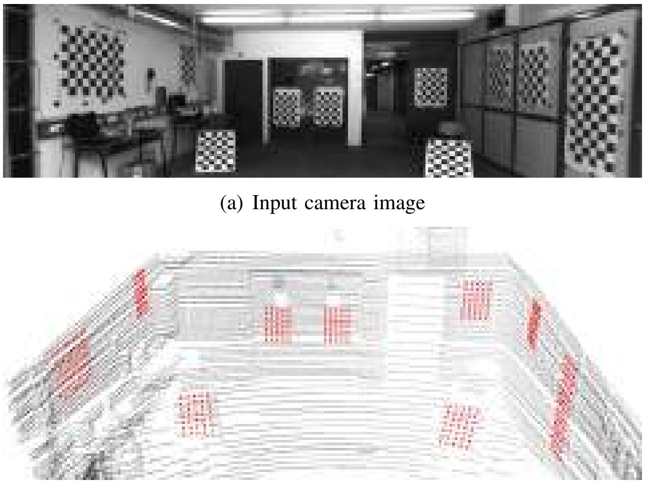
Existing methods [14], [13], [11], [3], [12] simplify camera-to-laser calibration by manual user intervention and the fact that several scans are taken with only one board visible at a time. Our system is designed for easy usage with the consequence of having a more difficult situation: We can neither rely on correspondences between the two point clouds, nor is it possible to use existing 3D point cloud alignment methods due to the small overlap (full world vs. checkerboards only) and the unknown initialization.

The proposed algorithm proceeds as follow: First, segmentation is leveraged to identify planes in the range data (Sec. IV-A). Next, transformation hypotheses are generated by random plane associations (Sec. IV-B). The best ones are refined and verified (Sec. IV-C). A final non-maxima suppression step yields all feasible solutions (Sec. IV-D).

A. *Segmentation*

Segmentation aims at finding contiguous regions (or *segments*) $\mathcal{P}_r^j = \{(x, y, z)\} \subseteq \mathcal{P}_r$ in the range data that represent a plane and could potentially correspond to a checkerboard mounting. Note that there might be several checkerboards sticking to the same wall.

As a preprocessing step, we calculate normal vectors $\mathbf{n}_r^i \in \mathbb{R}^3$ for each point $\mathbf{p}_r^i \in \mathcal{P}_r$ by principal component analysis on the K nearest neighbors $\mathcal{N}_{K,\mathcal{P}_r}(\mathbf{p}_r^i)$. Segmentation is carried out by greedily growing regions from random seed



(a) Input camera image

(b) Input range data and calibration result

Fig. 4. **Calibration example**. Camera with Velodyne HDL-64E.

points $\mathbf{p}_r^j \in \mathcal{P}_r$, each generating the corresponding set $\mathcal{P}_r^j$. A point $\mathbf{p}_r^i$ is added to the region $\mathcal{P}_r^j$ iff it is a neighbor and its normal vector is similar to the seed's normal:

$$\exists\, \mathbf{p}_r^m \in \mathcal{N}_{K,\mathcal{P}_r}(\mathbf{p}_r^i) : \mathbf{p}_r^m \in \mathcal{P}_r^j \wedge {\mathbf{n}_r^i}^T \mathbf{n}_r^j > \tau_{segment} \tag{8}$$

Each region is grown until it converges, then removed from $\mathcal{P}_r$, and a new seed is selected until no more points are left in $\mathcal{P}_r$. A final filtering step removes segments which are either significantly smaller than a checkerboard or not planar enough.

Note that the above algorithm is nondeterministic: Seed points are chosen randomly and each region-expansion is dependent on the seed. Experiments show that the outcome is nevertheless very stable. Compared to using local decisions this can guarantee that resulting segments are planar.

B. Global Registration

Given a set of planar point clouds $\{\mathcal{P}_r^j\}$ and $\{\mathcal{P}_c^j\}$ from the range sensor and camera respectively, this section generates a set of initial transformations $\mathcal{G} = \{\boldsymbol{\theta}_i\}$. Therefore, each point cloud region $\mathcal{P}_{c/l}^j$ is transformed into a disc-shaped surface $\mathsf{s}_{c/l}^j = (\mathbf{p}_{c/l}^j, \mathbf{n}_{c/l}^j)$ with center $\mathbf{p}_{c/l}^j$ and normal $\mathbf{n}_{c/l}^j$ using principal component analysis. Next, we repeatedly select and associate three surfaces from both sets in a random manner, calculate the transformation, and verify it using a distance measure. Algorithm 1 lists these steps in detail.

In line 4 of the algorithm, three surfaces are selected randomly from the checkerboards. A surface triple $(\mathsf{s}_c^a, \mathsf{s}_c^b, \mathsf{s}_c^c)$ thereby has the following probability of being selected:

$$p(\mathsf{s}_c^a, \mathsf{s}_c^b, \mathsf{s}_c^c) = \frac{1}{Z} \exp(-{\mathbf{n}_c^a}^T \mathbf{n}_c^b - {\mathbf{n}_c^a}^T \mathbf{n}_c^c - {\mathbf{n}_c^b}^T \mathbf{n}_c^c) \tag{9}$$

with Z being a normalizing constant. Intuitively, a surface triple is selected with higher probability if normals point into different directions. The probabilites of all possible combinations are computed in advance. Tractability is guaranteed by the limited number of checkerboards present in the scene.

Algorithm 1 Global registration

```
1: GenerateTransformations($\{\mathsf{s}_c^j\}, \{\mathsf{s}_r^j\}, \mathcal{P}_r$):
2: $\mathcal{G} \leftarrow \{\}$; $\mathcal{S} \leftarrow \{\}$;
3: while not enough($\mathcal{G}$) do
4:     $(\mathsf{s}_c^a, \mathsf{s}_c^b, \mathsf{s}_c^c) \leftarrow$ randomselect($\{\mathsf{s}_c^j\}$)
5:     $(\mathsf{s}_r^a, \mathsf{s}_r^b, \mathsf{s}_r^c) \leftarrow$ randomselect($\{\mathsf{s}_r^j\}$)
6:     $\boldsymbol{\theta} \leftarrow$ minimize($\mathsf{s}_c^a, \mathsf{s}_c^b, \mathsf{s}_c^c, \mathsf{s}_r^a, \mathsf{s}_r^b, \mathsf{s}_r^c$)
7:     $s \leftarrow$ score($\mathsf{s}_c^a, \mathsf{s}_c^b, \mathsf{s}_c^c, \boldsymbol{\theta}, \mathcal{P}_r$)
8:     $\mathcal{G}, \mathcal{S} \leftarrow$ selectbest($\mathcal{G} \cup \boldsymbol{\theta}$, $\mathcal{S} \cup s$)
9: end while
10: return $\mathcal{G}$
```

On the contrary, random selection in line 5 is for each $\mathsf{s}_r^{a/b/c}$ independent with a uniform probability distribution over $\{\mathsf{s}_r^j\}$.

The transformation $\boldsymbol{\theta}$ is computed (line 6) by aligning the selected surfaces. We first estimate the rotation matrix $\mathbf{R}$, which maximally aligns all normal vectors

$$\mathbf{R} = \underset{\mathbf{R}'}{\operatorname{argmax}} \sum_{i \in \{a,b,c\}} {\mathbf{n}_r^i}^T \mathbf{R}' \, \mathbf{n}_c^i = \mathbf{V}\mathbf{U}^T \tag{10}$$

using the singular value decomposition (SVD) of the covariance matrix $\sum_i \mathbf{n}_c^i {\mathbf{n}_r^i}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. Next, we estimate the translation $\mathbf{t}$ by minimizing point-to-plane distances

$$\mathbf{t} = \underset{\mathbf{t}'}{\operatorname{argmin}} \sum_{i \in \{a,b,c\}} ((\mathbf{R} \cdot \mathbf{p}_c^i + \mathbf{t}' - \mathbf{p}_r^i)^T \mathbf{n}_r^i)^2 \tag{11}$$

in closed form using least squares. Each transformation is scored (line 7) by

$$s = -\sum_{i \in \{a,b,c\}} ||\tilde{\mathbf{p}}_c^i - \mathcal{N}_{1,\mathcal{P}_r}(\tilde{\mathbf{p}}_c^i)||, \; \tilde{\mathbf{p}}_c^i = \mathbf{R} \cdot \mathbf{p}_c^i + \mathbf{t} \tag{12}$$

which determines the distance from the transformed checkerboard centers to the nearest neighbors in $\mathcal{P}_r$. We return all solutions for which the score exceeds $\tau_{score} \cdot \max(\{s\})$. The algorithm terminates if either all possible combinations have been processed or enough good transformations have been found ($|\mathcal{G}| > \tau_{comb}$).

C. Fine Registration

In the previous section, the initial transformation set $\mathcal{G}$ has been calculated from point clouds $\{\mathcal{P}_r^j\}, \{\mathcal{P}_c^j\}$ based on their centroids and normal vectors solely. Since this is very fast but less accurate, all transformations $\boldsymbol{\theta} \in \mathcal{G}$ are refined by gradient descent using the following error function:

$$E(\boldsymbol{\theta}) = \sum_{\mathbf{p}_c^i \in \mathcal{P}_c} ||\tilde{\mathbf{p}}_c^i - \mathcal{N}_{1,\mathcal{P}_r}(\tilde{\mathbf{p}}_c^i)||^2, \; \tilde{\mathbf{p}}_c^i = \mathbf{R} \cdot \mathbf{p}_c^i + \mathbf{t} \tag{13}$$

This minimizes the sum of point-to-point distances to closest neighbors, which can be solved with the well-known iterative closest points algorithm [25]. This results in a set of finely registered transformations $\mathcal{F}$.

D. Solution Selection

The transformation set $\mathcal{F}$ might contain several transformations that are very similar. Hence, we employ non-maxima suppression in order to suppress similar transformations in a local neighborhood based on their energy (13).

Under typical conditions, we retain exactly one solution after this process. If not, there exist ambiguities that result from the constellation of the checkerboards, i.e. orthogonality as in Fig. 6. In this case, an automatic view is rendered for each transformation and the final selection is left to the user.

V. EXPERIMENTS

To evaluate our approach, we collected a database of 10 different calibration settings with multiple shots each, as listed in Table I. Calibration settings differ by baseline, focal lengths and range sensors employed. For each shot, we varied the position, orientation and number of calibration targets. To make our dataset more representative, we additionally collected 16 random calibration images from the internet, which are only used to evaluate corner detection. In total, we used 126 camera images and 55 range measurements in our experiments. Despite the fact that the proposed framework extends to an arbitrary number of sensors, we restrict our quantitative evaluation to setups involving two cameras and a single range sensor for clarity of illustration, as illustrated in Fig. 1. As parameters, we empirically picked $n_{nms} = 3$, $\tau_{nms} = \tau_{corner} = 0.02$, $\tau_{match} = 0.1$, $\tau_{segment} = 0.9$, $\tau_{score} = 1.5$ and $\tau_{comb} = 25$ and keep them fixed throughout all experiments. To obtain a modest level of scale invariance and robustness with respect to blur, we detect corners using 4×4, 8×8, and 12×12 pixel windows in Sec. III-A and take the maximum of the three scores. Our C++ implementation achieves running times of less than one minute per calibration scenario, where most time is spent in the camera-to-range point cloud optimization procedures.

A. Corner Detection and Checkerboard Matching

We first evaluate the ability of our method to correctly locate checkerboard corners in all 126 test images, for which we annotated all checkerboard corners manually. We compare our approach to several competitive baseline methods: Harris corners [8] using the Shi-Tomasi criterium [17], a reimplementation of the method of Ha et al. [9] and the detector of Kassir et al. [4]. Note that a fair comparison to the more noise sensitive OpenCV detector [2] and its derivates (Rufli et al. [7]) is not possible due to the fact that they directly return a single checkerboard per image.

Fig. 5 (left) shows the precision-recall plot obtained by varying the detection threshold τ for each of the methods. For Ha et al. [9] we varied the relative number of pixels required to classify a region as dark or bright while employing a very conservative Harris threshold. Kassir et al. [4] has been run using the code and parameters proposed by the authors.

Note that our method significantly outperforms all baselines, especially in terms of recall. The recall of Ha et al. [9] and Kassir et al. [4] is bound by the (multi-scale) Harris corner detector, which serves as an input to those methods.

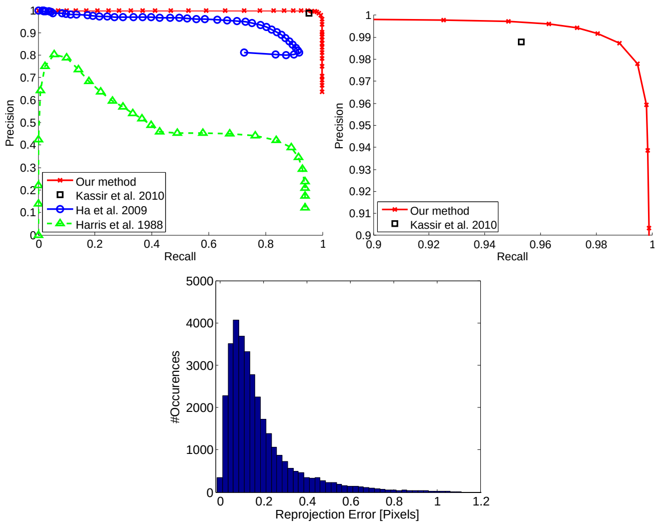


Fig. 5. **Left:** Precision-recall curves for different corner detection methods. **Right:** Close-up for the range $\{precision, recall\} \in [0.9...1.0]$. **Bottom:** Corner reprojection errors of our method after fitting the model parameters.

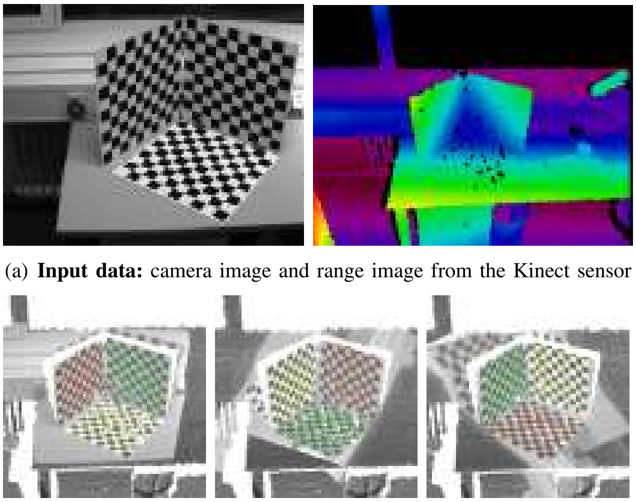
Qualitative results of our detector are illustrated in Fig. 8 and at www.cvlibs.net.

B. Camera-to-Camera Calibration

In this section we evaluate the camera-to-camera calibration accuracy by comparing the estimated baselines and focal lengths to ground truth in all 10 calibration settings. On average, we obtain a corner reprojection error of 0.18 pixels indicating high sub-pixel accuracies and a good model fit. This is also illustrated by the Poisson distribution of the reprojection errors shown in Fig. 5 (right). Further results are depicted in Table I: Here, the fifth and eighth column are the ground truth focal length and baseline, followed by the mean and standard deviation of the estimated values respectively. In general, the errors of our fully automatic calibration system are small. The largest ones occur for setting 7 and 8. This can be explained as those are the outdoor setups, which are more difficult since fewer corners have been matched as a consequence of cast shadows and difficult lighting conditions. Also note that the provided ground truth itself is subject to slight measurement errors, as the values have been measured manually or simply have been read from the objective lens.

C. Camera-to-Range Calibration

Assuming a precise camera-calibration, this section evaluates the robustness of camera-to-range calibration. Since range sensors are independent from lighting conditions and the proposed approach does not assume any initially given parameters, the only two factors of influence are the constellation of the checkerboards and the noise within the range data. The former is covered by the collected dataset (see Table I) that contains a varying number of checkerboards in different constellations. For the latter, we added Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_3)$ to the point cloud $\mathcal{P}_r$ for varying values of σ and carry out calibration. The final result $\mathbf{R}, \mathbf{t}$ is compared against ground truth $\mathbf{R}_g, \mathbf{t}_g$, which was determined with the



(a) **Input data:** camera image and range image from the Kinect sensor

(b) Camera-to-Range calibration result: Three solutions were detected, automatically rendered images help selecting the appropriate solution.

Fig. 6. **Calibration example**. Ambiguities are automatically detected.

fine registration (Sec. IV-C) on the original point cloud in a supervised way. Errors are given independently for rotation and translation

$$e_t = ||\mathbf{t} - \mathbf{t}_g|| \qquad (14)$$

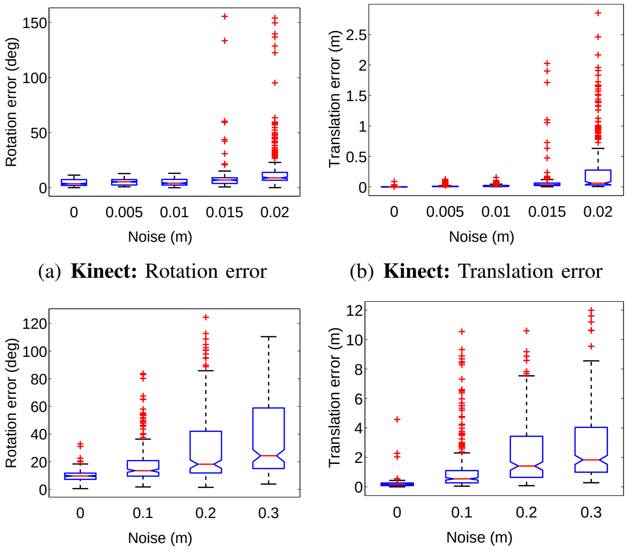
$$e_r = \angle(\mathbf{R}^{-1}\mathbf{R}_g) \qquad (15)$$

where e_r is the smallest rotation angle around a rotation axis that represents the rotation difference. This process is repeated 20 times for each setting, and the statistics over e_t and e_r are gathered for the Velodyne and the Kinect sensor independently. The results, depicted in Fig. 7, indicate that the approach is sufficiently robust to noise, but highly dependent on the calibration setting: Only configurations where the checkerboards constrain the problem sufficiently well lead to low errors. This is the case when the checkerboards cover most parts of the image and they are presented at various distances and orientations. The feasible higher noise level for the Velodyne sensor in comparison with the Kinect sensor is thereby due to sparser range data and roughly five times more distant checkerboards.

VI. CONCLUSIONS

We have proposed a toolbox for automatic camera and range sensor calibration and shown its effectiveness under various conditions. The main limiting assumption of our approach is a common field of view of the camera and range sensors. While this remains a useful scenario for applications such as generating stereo or scene flow ground truth, augmenting images with depth values or colorizing a point cloud, we believe that extending our method to handle partly overlapping fields of view will further increase the range of applications.

VII. ACKNOWLEDGMENTS



(a) **Kinect:** Rotation error

(b) **Kinect:** Translation error

(c) **Velodyne:** Rotation error

(d) **Velodyne:** Translation error

Fig. 7. **Robustness of camera-to-range calibration**. Calibration errors when adding Gaussian noise $\mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I}_3)$ to the range data. On each box, the central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually.

We thank the Karlsruhe School of Optics and Photonics for their support. We also thank the Research Center for Information Technology Karlsruhe (FZI) for providing us with a Kinect range sensor for our experiments.

REFERENCES

[1] J.-Y. Bouguet, *Camera Calibration Toolbox for Matlab*, 2010. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc

[2] G. Bradski, *OpenCV*, 2011. [Online]. Available: http://opencv.willowgarage.com

[3] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-09, July 2005.

[4] A. Kassir and T. Peynot, "Reliable automatic camera-laser calibration," in *Australasian Conference on Robotics and Automation*, 2010.

[5] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Computer Vision and Pattern Recognition*, 1997.

[6] A. S. of Photogrammetry, C. Slama, C. Theurer, and S. Henriksen, *Manual of photogrammetry*. American Society of Photogrammetry, 1980.

[7] M. Rufli, D. Scaramuzza, and R. Siegwart, "Automatic detection of checkerboards on blurred and distorted images," in *Intelligent Robots and Systems*, 2008.

[8] C. Harris and M. Stephens, "A combined corner and edge detector," in *Fourth Alvey Vision Conference*, 1988.

[9] J.-E. Ha, "Automatic detection of chessboard and its applications," *Optical Engineering*, vol. 48, no. 6, June 2009.

[10] J. Pilet, A. Geiger, P. Lagger, V. Lepetit, and P. Fua, "An all-in-one solution to geometric and photometric calibration," in *International Symposium on Mixed and Augmented Reality*, October 2006.

[11] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes," in *Intelligent Robots and Systems*, 2007.

[12] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *Intelligent Robots and Systems*, 2004.

[13] H. Zhao, Y. Chen, and R. Shibasaki, "An efficient extrinsic calibration of a multiple laser scanners and cameras' sensor system on a mobile platform," in *Intelligent Vehicles Symposium*, 2007.

[14] L. Huang and M. Barth, "A novel multi-planar lidar and computer vision calibration procedure using 2d patterns for automated navigation," in *Intelligent Vehicles Symposium*, 2009.

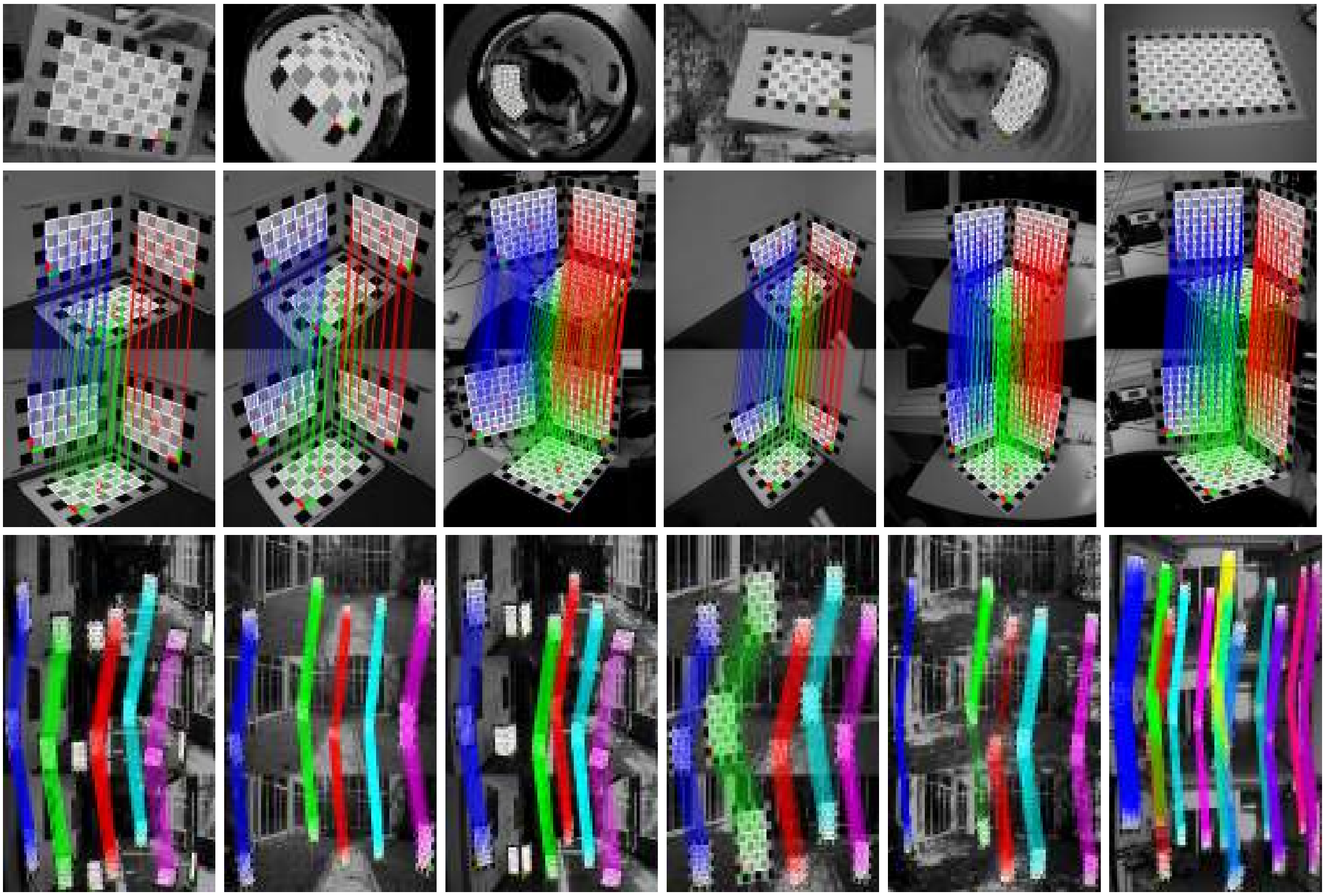


Fig. 8. **Structure Recovery and Checkerboard Matching Results. Top row:** Random monocular wide-angle and fisheye calibration images from [16] and Google Image Search. **Middle row:** Binocular camera setup and Kinect range sensor. **Last row:** Trinocular camera setup with Velodyne HDL 64 laser scanner. Additional results are available at: www.cvlibs.net.

Setting	#Shots	Range sensor	Environment	#Checkerboards	Focal length f	Mean($\hat{f}$)	Std($\hat{f}$)	Baseline b	Mean($\hat{b}$)	Std($\hat{b}$)
0	5	kinect	indoor	3	6.00 mm	6.10 mm	0.01 mm	11.50 cm	11.45 cm	0.01 cm
1	5	kinect	indoor	3	9.50 mm	9.56 mm	0.13 mm	11.50 cm	11.11 cm	0.14 cm
2	3	kinect	indoor	3	9.50 mm	9.68 mm	0.01 mm	20.70 cm	20.98 cm	0.39 cm
3	3	kinect	indoor	3	4.00 mm	4.12 mm	0.01 mm	20.70 cm	20.76 cm	0.05 cm
4	4	kinect	indoor	3	4.00 mm	4.11 mm	0.01 mm	6.20 cm	6.19 cm	0.10 cm
5	9	kinect	indoor	3	11.50 mm	11.61 mm	0.19 mm	6.20 cm	6.04 cm	0.21 cm
6	6	velodyne	hall	7	4.00 mm	4.53 mm	0.11 mm	53.60 cm	52.27 cm	0.93 cm
7	5	velodyne	hall / outdoor	4–7	8.00 mm	8.25 mm	0.29 mm	53.60 cm	56.24 cm	2.60 cm
8	6	velodyne	outdoor	4–5	4.00 mm	4.32 mm	0.11 mm	53.60 cm	54.10 cm	2.76 cm
9	9	velodyne	garage	8–12	4.00 mm	4.23 mm	0.02 mm	53.60 cm	53.17 cm	0.38 cm

TABLE I

EXPERIMENTAL SETTINGS WITH CAMERA-TO-CAMERA CALIBRATION RESULTS

[15] O. Naroditsky, A. Patterson, and K. Daniilidis, "Automatic alignment of a camera with a line scan lidar system," in *International Conference on Robotics and Automation*, 2011.

[16] D. Scaramuzza and A. Martinelli, "A toolbox for easily calibrating omnidirectional cameras," in *International Conference on Intelligent Robots and Systems*, 2006.

[17] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition*, 1994.

[18] A. Neubeck and L. J. V. Gool, "Efficient non-maximum suppression," in *International Conference on Pattern Recognition*, 2006.

[19] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," *Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[20] L. Lucchese and S. Mitra, "Using saddle points for subpixel feature detection in camera calibration targets," in *Asia-Pacific Conference on Circuits and Systems*, 2002.

[21] J. Lavest, M. Viala, and M. Dhome, "Do we really need an accurate calibration pattern to achieve a reliable camera calibration?" in *European Conference on Computer Vision*, 1998.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, 2004.

[23] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *European Conference on Computer Vision*, 2006.

[24] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *International Conference on Computer Vision*, 1999.

[25] P. Besl and H. McKay, "A method for registration of 3-d shapes," *Transactions on Pattern Analysis and Machine Intelligence*, 1992.