2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Infrastructure-Based Calibration of a Multi-Camera Rig

Lionel Heng[1], Mathias Bürki[2], Gim Hee Lee[1], Paul Furgale[2], Roland Siegwart[2], and Marc Pollefeys[1]

Abstract— **The online recalibration of multi-sensor systems is a fundamental problem that must be solved before complex automated systems are deployed in situations such as automated driving. In such situations, accurate knowledge of calibration parameters is critical for the safe operation of automated systems. However, most existing calibration methods for multi-sensor systems are computationally expensive, use installations of known fiducial patterns, and require expert supervision. We propose an alternative approach called *infrastructure-based calibration* that is efficient, requires no modification of the infrastructure, and is completely unsupervised. In a survey phase, a computationally expensive simultaneous localization and mapping (SLAM) method is used to build a highly accurate map of a calibration area. Once the map is built, many other vehicles are able to use it for calibration as if it were a known fiducial pattern.**

We demonstrate the effectiveness of this method to calibrate the extrinsic parameters of a multi-camera system. The method does not assume that the cameras have an overlapping field of view and it does not require an initial guess. As the camera rig moves through the previously mapped area, we match features between each set of synchronized camera images and the map. Subsequently, we find the camera poses and inlier 2D-3D correspondences. From the camera poses, we obtain an initial estimate of the camera extrinsics and rig poses, and optimize these extrinsics and rig poses via non-linear refinement. The calibration code is publicly available as a standalone C++ package.

I. INTRODUCTION

Many companies are devoting substantial parts of their budgets to research and development of robots and automated systems. As a result, in the coming years, we will see a wide-scale deployment of such systems in consumer-oriented markets. This raises a number of fundamental problems related to the *long-term* autonomy of robotic systems—autonomy for months at a time without the constant supervision of experts. Calibration of complex robotic systems is one of these fundamental problems.

In many automated systems such as autonomous cars or driver assistance systems, so-called *calibration parameters*—transformations between sensors, scale factors, camera lens parameters, etc.—must be known with a high degree of precision to ensure safe and robust operation in the presence of pedestrians and other vehicles. Although systems can be calibrated in the factory, some parameters will change due to normal wear and tear during extended operation. Within the research and development community, calibration and recalibration of multi-sensor/multi-actuator systems is a continual burden that requires expertise, specialized equipment, and special vehicle motions. Consequently, it is necessary that we seek robust and accurate online self-calibration algorithms that require no operator input.

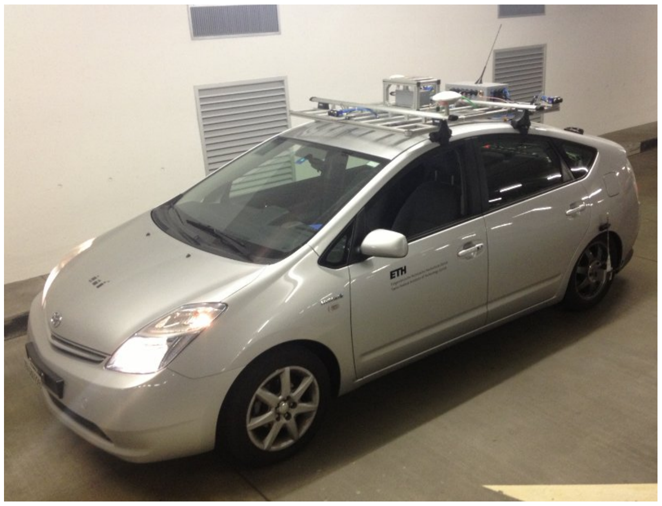


Fig. 1: Our Prius platform equipped with a set of four fish-eye cameras that provides an all-surround view of the environment. This paper demonstrates the use of *Infrastructure-Based Calibration* to estimate the extrinsic transformations of this multi-camera system based solely on a pre-existing map. No specialized fiducial markers such as chessboards are used.

One possibility for calibration of mobile robotic systems is to build "calibration areas" that include special instrumentation for different sensors, such as known fiducial markings for cameras or known structural geometry for lasers. The use of specially designed fiducials can resolve appearance and geometric ambiguities and reduce the computational complexity of system calibration. For example, this was the strategy adopted by Geiger et al. [5] as they were collecting an extensive multi-sensor dataset over a number of months [6]. They installed a number of chessboards covering the full field of view of the cameras and laser scanner and used them to recalibrate the vehicle before every run of data collection. As attractive as this method is, it still requires modification of the infrastructure, which could make deployment on the large scale complicated and expensive, and represent yet another barrier to the deployment of autonomous systems.

In this paper, we introduce a method called *Infrastructure-Based Calibration* that shares positive aspects with the above method but relaxes the requirement to modify the infrastructure. This method leverages on multi-sensor SLAM to build calibration areas using an already calibrated robotic system. Although these SLAM-based methods can be com-

[1]L. Heng, G.H. Lee, and M. Pollefeys are with the Computer Vision and Geometry Lab, ETH Zürich, Switzerland. {hengli@inf.ethz.ch, glee@student.ethz.ch, marc.pollefeys.inf.ethz.ch}

[2]M. Bürki, P. Furgale, and R. Siegwart are with the Autonomous Systems Lab, ETH Zürich, Switzerland. {mbuerki@student.ethz.ch, paul.furgale@mavt.ethz.ch, r.siegwart@ieee.org}

putationally expensive, the resulting data may be used for calibration of any number of other vehicles at a fraction of the computational cost. Furthermore, we show how a state-of-the-art self-calibration method [8] requiring special motions of the vehicle may be used to bootstrap the process, further reducing the cost of deployment by removing the need for a specially calibrated survey vehicle. Infrastructure-based calibration is therefore a both financially and computationally efficient solution to the continuous unsupervised calibration of large numbers of automated vehicles.

We demonstrate the accuracy of the method through the calibration of the vehicle-mounted multi-camera system shown in Figure 1. Multi-camera setups have seen a rapidly increasing number of applications, which however, require an accurate calibration to achieve optimum results. A calibration is accurate only if the computed camera intrinsics and extrinsics allow one to relate 2D image points to 3D scene points with low reprojection error. Environmental factors such as temperature variations and vibration cause the camera extrinsics to deviate much more than the camera intrinsics from their original values over time. Hence, in this paper, we focus on estimating the camera extrinsics while assuming that the camera intrinsics stay constant over time. The camera extrinsics refer to the set of camera poses with respect to a reference frame located on the rig. If odometry data is available for the camera rig, we can compute the transform between the camera rig's reference frame and the odometry frame. Examples of odometry sources are wheel odometry, a GPS/INS system, or a Vicon motion capture system.

A. *Related Work*

We focus on existing work that calibrates a multi-camera rig without assuming overlapping fields of views. A majority of existing work [11, 12, 13] requires a pattern board. The additional use of a mirror in Kumar et al. [11] creates a limitation in which the mirror has to be in the camera's field of view, while at the same time, the entire pattern is visible in the camera. Lebraly et al. [12] uses two pattern boards, and requires the rig to manoeuvre such that each camera sees both pattern boards at different times. Li et al. [13] requires neighboring cameras to see some part of the pattern at the same time. We note that the use of a pattern board comes with a constraint that makes calibration of multi-camera rigs non-straightforward.

We then look at unsupervised methods that do not require specific calibration patterns. Our approach is most similar to the works of Carrera et al. [2] and Heng et al. [8] in the sense that we perform an unsupervised extrinsic calibration based on natural features in the environment. Carrera et al. [2] builds a globally consistent sparse feature map for each camera. Subsequently, feature correspondences are exhaustively found between each pair of maps, and an inlier set is obtained from the 3D similarity transform together with RANSAC. At the end, a global bundle adjustment is run to optimize the camera poses, 3D scene points, and robot poses. Here, the 3D similarity transform step can fail in outdoor environments where the majority of natural features are located far away from the cameras, and their estimated 3D locations can have substantial noise as a result, leading to few inliers. Heng et al. [8] overcomes this difficulty; for each image in each camera, they find feature correspondences between the image and a set of the most recent images from each of all other cameras. To maximize the number of feature correspondences, the heading from odometry data is used to rectify each image pair on a common image plane before feature matching is done between the rectified images.

We note that since these two SLAM-based works do not assume a prior map, they have to perform an exhaustive search of feature correspondences between images from different cameras, and rely on loop closures which may fail sometimes. By relying on a pre-existing map, we remove the need to find inter-camera feature correspondences and loop closures. Furthermore, we do not have to do global bundle adjustment. As a result, our infrastructure-based calibration is simpler, more robust, and requires a much shorter time compared to the two above-described methods. We can think of our method as requiring a one-time fixed cost in terms of map generation but entailing a much lower variable cost per calibration. Hence, only our method is advantageous if many calibrations are needed within a short time span.

This statement succinctly describes the motivation behind our infrastructure-based calibration: "The world is a giant chessboard." We use a map of the environment as a virtual 3D chessboard which is used for quickly inferring camera poses with high accuracy. This can be seen as an analogy to the ubiquitous method of inferring camera poses via the use of a chessboard in intrinsic camera calibration. A pre-verification of the map accuracy and the non-need to find loop closures maximize the robustness of our calibration method. There is no perfect robust loop closure framework as we cannot identify all false loop closures in every possible scenario.

II. Platform

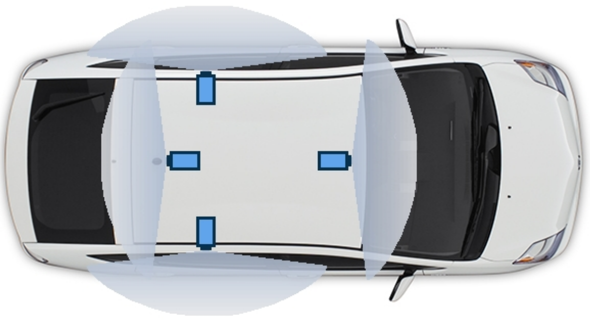


Fig. 2: The sensor coverage of the car platform. The four fish-eye cameras are marked in blue.

The platform is a Toyota Prius car that is equipped with 4 mvBlueCOUGAR cameras from Matrix Vision. Each camera has an image resolution of 1280 $\times$ 960 and uses a fish-eye lens. For synchronous image capture, one camera acts as a master trigger source for all other cameras. We compute the odometry in real-time by using the individual wheel velocities and steering wheel angle to compute the position and heading of the car.

For purposes of clarity, in subsequent figures, we color all camera poses and 3D scene points associated to the front, left, rear, and right cameras as red, green, blue, and yellow respectively.

III. INTRINSIC CAMERA CALIBRATION

We require all cameras to be calibrated before we run our infrastructure-based calibration method. For each camera, we perform an intrinsic calibration in which we find the parameters for a given camera model. Here, we use the reduced Kannala-Brandt camera model [10] to model the camera intrinsics. This camera model consists of 8 parameters: k_1, k_2, k_3, k_4, m_u, m_v, u_0, and v_0. This reduced model does not take radial and tangential distortions into account; from experiments, we find that this model is accurate nevertheless as we are using high-quality lenses with minimal radial and tangential distortions.

Given a scene point $P = [X\ Y\ Z]^T$, we find its image coordinates (u, v):

$$\begin{aligned}
\theta &= \arccos \frac{Z}{\|P\|} \\
\phi &= \arctan \frac{Y}{X} \\
r(\theta) &= \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 \\
\begin{bmatrix} x \\ y \end{bmatrix} &= r(\theta) \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \\
\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \begin{bmatrix} m_u & 0 & u_0 \\ 0 & m_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}
\end{aligned} \tag{1}$$

Here, $r(\theta)$ is the distance between the image point and principal point on the normalized plane. Inversely, given an image point $p = [u\ v]^T$, we find its corresponding ray $[X\ Y\ Z]^T$:

$$\begin{aligned}
\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} m_u & 0 & u_0 \\ 0 & m_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \\
d &= \sqrt{x^2 + y^2} \\
&= \theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9
\end{aligned} \tag{2}$$

We solve for θ using the companion matrix. If there are no real roots, we use $\theta = d$.

$$\begin{aligned}
\phi &= \begin{cases} 0 & \text{if } d = 0 \\ \arctan \frac{y}{x} & \text{otherwise} \end{cases} \\
\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} &= \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix}
\end{aligned} \tag{3}$$

We propose an unique method to obtain the parameters for the reduced Kannala-Brandt model. First, chessboard detection is performed to find the image coordinates of all interior corners on the chessboard in each image. This chessboard corner data is used to compute an initial estimate of the intrinsic parameters and camera poses. In this initial estimate, we set $k_1 = k_2 = k_3 = k_4 = 0$, $m_u = m_v = f$, $u_0 = \frac{w}{2}$ and $v_0 = \frac{h}{2}$ where w and h are the width and height of the image respectively. Here, we can see that computing the initial estimate of the intrinsic parameters only requires us to estimate the value of f which is the focal length. The initial estimate transforms the reduced Kannala-Brandt model into the well-known equidistant fish-eye model, and allows us to use existing calibration methods for equidistant fish-eye models.

From Hughes et al. [9], we see that for a equidistant fish-eye model, if we fit a circle through the corners of each row of the chessboard in one image, all the resulting circles intersect at two vanishing points v_1 and v_2. We can then find $f = \frac{\|v_1 - v_2\|}{\pi}$. For each chessboard image, we obtain a hypothesis for f, and we choose the value of f to be that of the best hypothesis which corresponds to the lowest sum of all reprojection errors. We infer the camera poses by solving the PnP problem using 2D-3D correspondences. We then optimize the intrinsic parameters and camera poses via non-linear refinement in which we minimize the sum of all reprojection errors.

In our calibration pipeline, other camera models such as the pinhole model and the unified projection model can also be used. The Kannala-Brandt model is able to model a fish-eye camera more accurately when compared to the unified projection model due to the higher number of parameters, but at the cost of increased computation due to the higher computational complexity of backprojection. As a result, the Kannala-Brandt model is recommended for applications which do not have real-time requirements.

IV. INFRASTRUCTURE-BASED CALIBRATION

At the beginning, we build a sparse feature map of the environment in which calibration is conducted. We use this map as the basis for multiple calibrations as long as the environment does not change substantially. In addition, we assume that the cameras used in the calibration have been calibrated. Before we start the calibration, we log synchronized images from all cameras as the rig moves through the environment. We also log odometry data if such data is available. We then run a pipeline which processes the logged data, and estimates the camera extrinsics with respect to a designated reference frame located on the camera rig. The pipeline first infers camera poses via visual localization, and subsequently, an initial estimate of the camera-rig transforms and rig poses. In turn, a non-linear refinement step optimizes the camera-rig transforms and rig poses. If odometry data is available, we find the transform between the camera rig's reference frame and the odometry frame by using a hand-eye calibration method, and subsequently obtain the camera-odometry transforms. We show a diagram of the pipeline in Figure 3, and describe each step of the pipeline in detail below.

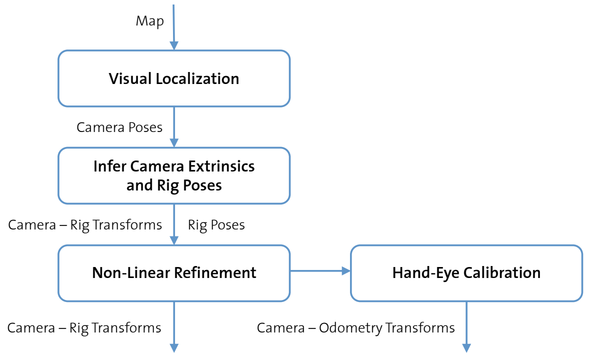


Fig. 3: A sparse feature map and images are input to the infrastructure-based calibration pipeline which generates the camera extrinsics.

A. Building A Sparse Feature Map

A standard structure-from-motion implementation can be used to build a sparse feature map. An example of such an implementation is Wu [16]. Furthermore, we can use a different camera setup such as a stereo camera to build the map. This map is a graph data structure in which a node can be either an image, a 2D feature point, or a 3D scene point. Edges link an image to 2D feature points that are detected in that image. In addition, an edge links a 2D feature point to a corresponding 3D scene point.

After the map is generated, we build a vocabulary tree by converting each image in the map to a bag-of-words vector and adding this vector to the vocabulary tree.

B. Visual Localization

The visual localization step takes images from the multi-camera system and the sparse feature map as input, and outputs camera poses with respect to the map's reference frame. Visual localization allows us to infer the camera poses for a given set of frames captured simultaneously from all cameras. First, for each image, we use the vocabulary tree to find n most similar images. For each candidate, we obtain 2D-2D feature correspondences from matching features between the query image and the candidate using the well-known distance ratio. As the feature points in the candidate image already have corresponding 3D scene points from the sparse feature map, it is trivial to obtain 2D-3D feature correspondences. We use the EPnP method [15] to find the camera pose together with the inlier set of 2D-3D feature correspondences. For the query image, we choose the camera pose associated with the candidate which has the highest number of inlier 2D-3D feature correspondences. The camera pose is defined to be unknown if the highest number of inlier correspondences does not exceed a threshold, which in our case, is 25. We store the set of camera poses if the following two conditions are met:

1) if at least 2 camera poses are found, and
2) the minimum distance between the current camera pose and the previous camera pose over all cameras exceeds a threshold, which in our case, is 0.3 m.

The former condition is necessary for the set of camera poses to be useful for calibration, and the latter condition minimizes bias by avoiding situations where the majority of sets of camera poses is concentrated in a few locations. These situations occur when the car is at a standstill or moving slowly.

C. Inferring Camera Extrinsics and Rig Poses

In this step, we use the sets of camera poses inferred from visual localization to infer the camera-rig transforms and the rig poses. Since the cameras are rigidly fixed to the rig, we express each set of camera poses at any point of time as a function of the rig pose at that time and the camera extrinsics. Here, the camera extrinsics comprise a rigid body transform from each camera's reference frame to the rig's reference frame.

When computing an initial estimate of the rig poses, we choose the reference frame of the rig to be aligned with that of the first camera without loss of generality. However, the rig's reference frame may not necessarily be aligned with the first camera's reference frame at the end of the non-linear refinement step.

We only use complete sets of camera poses to estimate the camera extrinsics. A set of camera poses at any time is considered to be complete if the poses for all cameras can be estimated from the images captured at that time. For each complete set of camera poses, we compute a hypothesis of the camera extrinsics. We use this hypothesis to compute the rig poses. In each set of n camera poses in which the pose for the first camera may not be available, we use the hypothesis of the camera extrinsics to compute n estimates of the rig poses. We obtain the rig pose by using the quaternion averaging method [14] to obtain the average rotation, and simple averaging to obtain the average translation. We choose the best hypothesis of the camera extrinsics that minimizes the average reprojection error over all images. This best hypothesis also gives us the rig poses.

D. Non-Linear Refinement

In this step, we minimize the sum of all reprojection errors by optimizing the camera extrinsics and the rig poses while keeping the coordinates of all 3D scene points fixed.

Formally, we solve the optimization problem:

$$\min_{P_i,T_c} \sum_{c,i,p} \rho\left(||\pi(C_c, P_i, T_c, X_p) - p_{cip}||^2\right) \tag{4}$$

π is a projection function that predicts the image co-ordinates of the scene point X_p seen in camera c given the camera's intrinsic parameters C_c, the rig pose P_i, and the rigid body transformation from the camera to the rig's reference frame T_c. p_{cip} is the observed image coordinates of X_p seen in camera c with the corresponding rig pose P_i. ρ is a robust cost function used for minimizing the influence of outliers. We use the Cauchy cost function in this case.

E. Hand-Eye Calibration

If odometry data is available, we can optionally choose to obtain the rig-odometry transform, and thus, the camera-odometry transforms. Otherwise, we skip this step, and output the resulting camera-rig transforms.

We compute the rig-odometry transform by finding a least-squares solution to the hand-eye calibration problem that relates relative rig motions to relative odometry motions. In the case of 6-DoF motion of the camera rig, we use the dual quaternion approach [3] to find the rig-odometry transform. In the case of planar 3-DoF motion, we use the method described in Guo et al. [7] to obtain the rig-odometry transform.

V. IMPLEMENTATION

In our implementation, we use SURF[1] to detect features and compute their descriptors. The CamOdoCal library [8] is used to build a sparse feature map for the infrastructure-based calibration. We use the DBoW2 implementation [4] of the vocabulary tree. Non-linear refinement is implemented using the Ceres library [1].

VI. EXPERIMENTS AND RESULTS

We verify the accuracy and repeatability of our infrastructure-based calibration via real-world experiments in both an indoor parking garage and outdoor urban environment on the ETH campus. Figure 5 shows images of both areas taken by the front camera. We design our experiments to demonstrate that our infrastructure-based calibration exhibits a high level of accuracy in both indoor and outdoor environments in the presence of moving cars and pedestrians.

Before we conduct the experiments, we use the CamOdoCal pipeline to build a sparse feature map for both areas with initial unknown extrinsics as shown in figures 6a and 7a. To determine the accuracy of our estimated camera extrinsics, we compare the estimated camera extrinsics against those estimated by CamOdoCal. We first compute the pose of each camera with respect to the first camera for both sets of extrinsics. Then, we use these relative poses to compute the rotation error and two types of translation errors: the angle between the two translation vectors, and the norm of the difference between the two translation vectors. These three errors are used to give a qualitative measure of the accuracy of the camera extrinsics estimated by our infrastructure-based calibration method.

In one indoor experiment, figure 6b shows the subset of scene points that are from the sparse feature map and used for the calibration. Furthermore, the figure shows the estimated camera poses which are used to infer the camera extrinsics. Similarly, for one outdoor experiment, the scene points and camera poses are shown in figure 7b.

A. Experiment — Indoor Parking Garage

In this experiment, the Prius is driven along one loop with the same camera configuration that was used to generate the sparse feature map. This loop trajectory differs from that taken by the Prius during the data collection for building the sparse feature map. This experiment aims to show that our estimated extrinsics and those estimated by CamOdoCal are the same. The accompanying video shows the infrastructure-based calibration process.

[1]http://docs.opencv.org/modules/nonfree/doc/feature_detection.html

TABLE I: Indoor experiment: comparison of extrinsics estimated by our method and those estimated by CamOdoCal.

	Left Cam	Rear Cam	Right Cam
Rotation error (deg)	0.0044	0.0032	0.0088
Translation error (deg)	0.0563	0.0468	0.0349
Translation error (m)	0.0016	0.0022	0.0021

We tabulate the errors between our estimated extrinsics and the CamOdoCal extrinsics in table I. The infrastructure-based calibration estimated the extrinsics from 167 sets of camera poses with an average of 3.05 camera poses per set. A total of 37860 2D-3D correspondences were used. The initial estimates of the extrinsics and rig poses had an associated average reprojection error of 0.99 pixels which reduced to 0.69 pixels after non-linear refinement. It is observed from the results in table I that the extrinsics estimated by our method are virtually the same as those estimated by CamOdoCal.

B. Experiments — Outdoor Urban Environment

We run a total of four experiments. In each of the first three experiments, the Prius is driven in one loop in the same scene and with a different camera configuration. As in the indoor experiment, this loop trajectory differs from that taken by the Prius during the data collection for building the sparse feature map. These three experiments aim to show that our calibration can reliably estimate the camera extrinsics for a camera configuration different from that used for building the sparse feature map. For the fourth experiment, we drive the Prius over 25 loops for 1 hour with the same camera configuration that was used to generate the sparse feature map. This experiment shows the impact of a changing environment on the calibration accuracy and repeatability; during this one hour, cars and pedestrians continually move around, and plants and trees sway significantly in moderate wind conditions. From odometry measurements, the average distance of each loop in all experiments is 308 m.

We use a three-way tripod head and a sliding plate in figure 4 to ensure that camera configuration changes can be measured as precisely as possible.

1) Experiment 1: The left and right cameras are moved towards the front of the car by 9.7 and 10.0 cm respectively as measured with a ruler. The infrastructure-based calibration used 427 sets of camera poses with an average of 3.17 camera poses per set. Based on the results from our method, the left and right cameras are deemed to have moved 9.72 cm and 10.03 cm respectively. The rear camera is estimated to have moved 0.005 cm which is negligible. These estimates closely agree with the hand measurements.

2) Experiment 2: The left camera is rotated around its x-axis towards the ground by 30° as measured with a scale

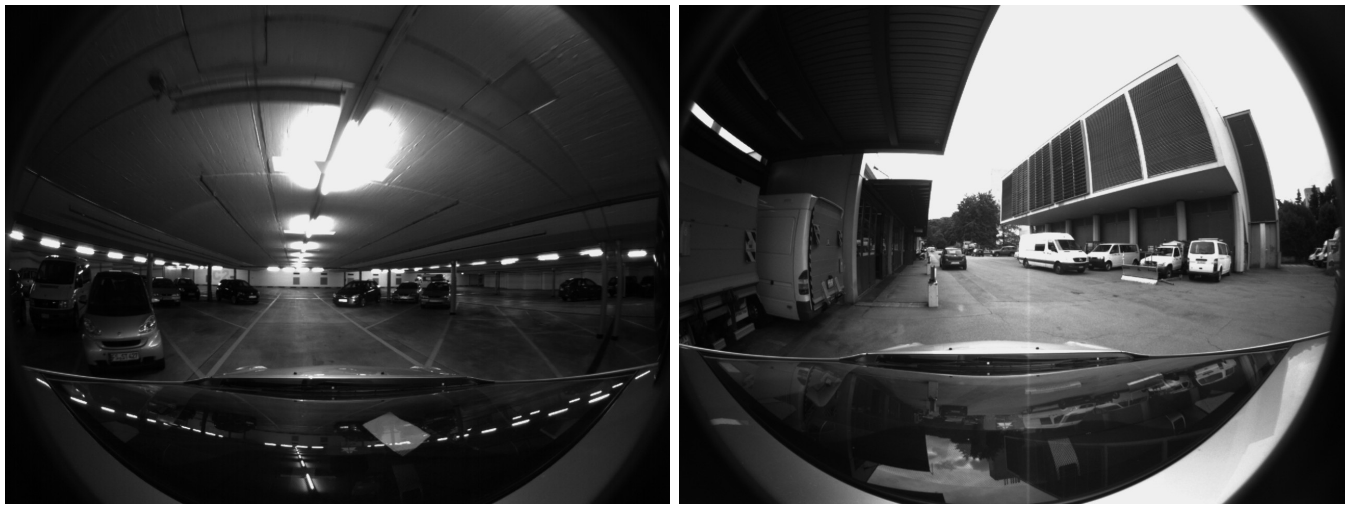


Fig. 5: (a) Indoor parking garage and (b) outdoor urban environment as seen from the front camera

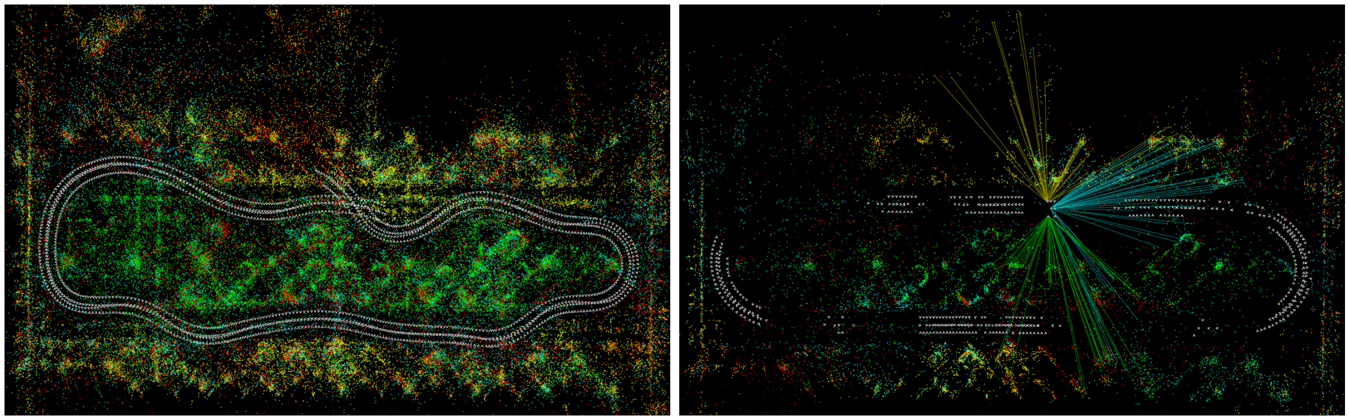


Fig. 6: Experiments in an indoor parking garage as shown in figure 5a. (a) The sparse feature map generated with the CamOdoCal pipeline. The camera poses marked in white illustrate the twisting path taken by the car. (b) The subset of scene points from the sparse feature map and which is used for calibration is shown. The camera poses inferred from PnP are marked as white triangles. Here, the car approaches the end of a loop, and the current set of camera poses are derived from 2D-3D correspondences visualized as lines from the camera poses to the scene points.

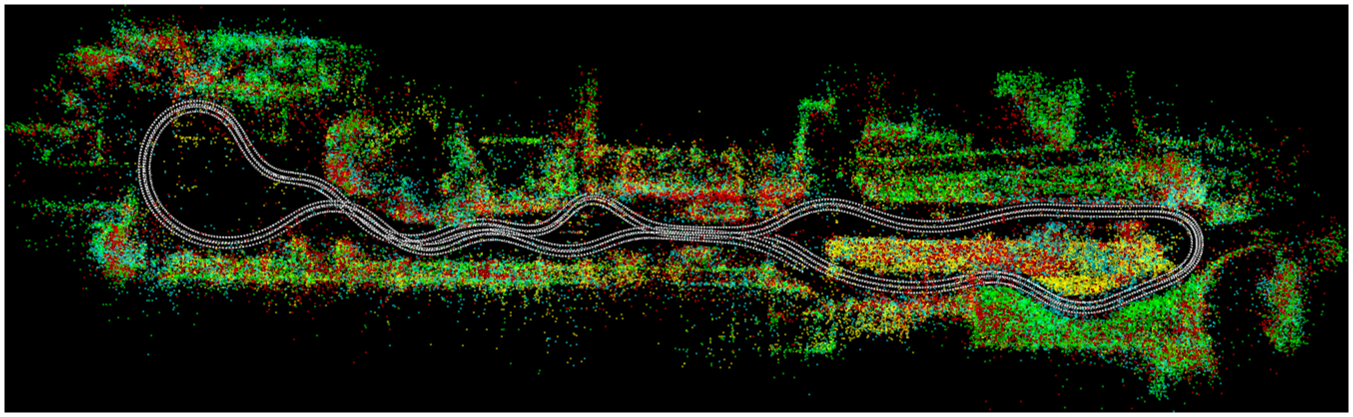
that is available for each axis of movement of the three-way tripod head. The infrastructure-based calibration used 411 sets of camera poses with an average of 3.14 camera poses per set. Our method estimates the left camera to have rotated about its x-axis by 29.9°. This estimate closely agrees with the hand measurement.

3) Experiment 3: The left camera is rotated around its z-axis towards the front of the car by 15°. The infrastructure-based calibration used 381 sets of camera poses with an average of 3.20 camera poses per set. Our method estimates the left camera to have rotated around its x-axis by 14.3°. This estimate closely agrees with the hand measurements.

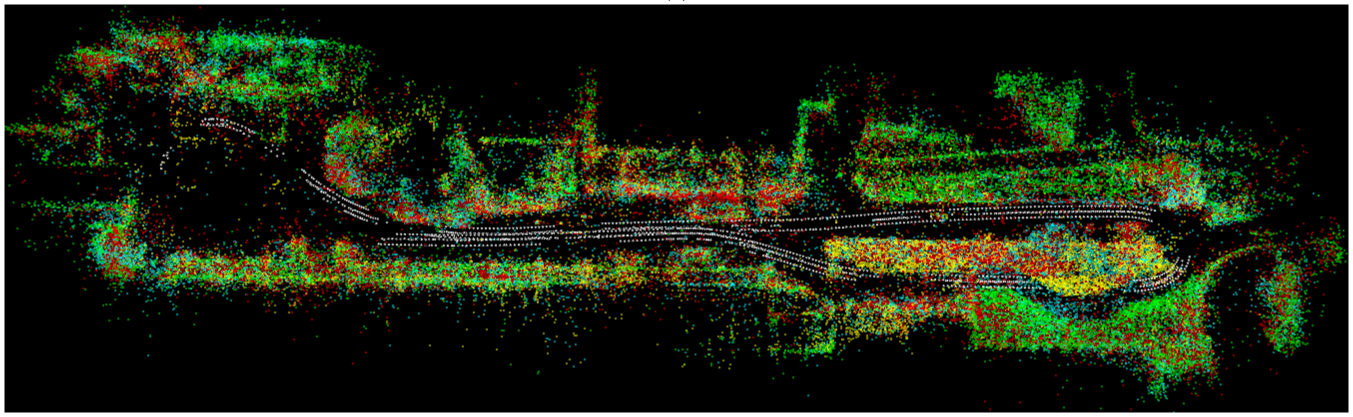
4) Experiment 4: For each of the 25 loops, we estimate the extrinsics, and find the maximum of the three error metrics among all cameras. We then plot these maximum errors in figure 8. This plot shows that our calibration method is still very accurate regardless of changes in the environment, as the maximum rotation and translation errors do not exceed 0.025°, 0.22°, and 0.77 cm respectively.

C. Discussions

To calibrate a 4-camera rig based on 500 frames per camera, our infrastructure-based calibration takes 10 minutes on average while the CamOdoCal pipeline takes 2 hours. Our infrastructure-based calibration requires a much shorter running time, and hence, is extremely useful when multiple calibrations are needed within a short time. Furthermore, our extensive experiments clearly demonstrate the high accuracy of our infrastructure-based calibration. The calibration is shown to work with camera configurations which significantly differ from that used to build the sparse feature map. We also show changes in the environment to have no impact on the calibration accuracy. It is important to note that the accuracy of the infrastructure-based calibration is dependent



(a)



(b)

(c)

Fig. 7: Experiments in an outdoor urban environment as shown in figure 5b. (a) The sparse feature map generated with the CamOdoCal pipeline. (b) The subset of scene points from the sparse feature map and which is used for calibration. The camera poses inferred from PnP are marked as white triangles. (c) Aerial imagery of the environment.

on both the metric accuracy of the map and the accuracy of the intrinsic calibration. Using a substandard map will cause the infrastructure-based calibration to produce inaccurate camera extrinsics.

VII. CONCLUSIONS

Our experimental results demonstrate the feasibility and high accuracy of our method for infrastructure-based calibration of a multi-camera rig. In contrast to SLAM-based calibration methods, we require a prior map, which however, makes our calibration method much more robust and vastly reduces the time required for each calibration. With these two important advantages, only our infrastructure-based calibration method is feasible in scenarios which require multiple calibrations in a short time without expert supervision. In future work, we will look at extending the usable lifespan of the map used for the calibration by exploring time-invariant feature descriptors. The code for the infrastructure-based calibration is available as part of

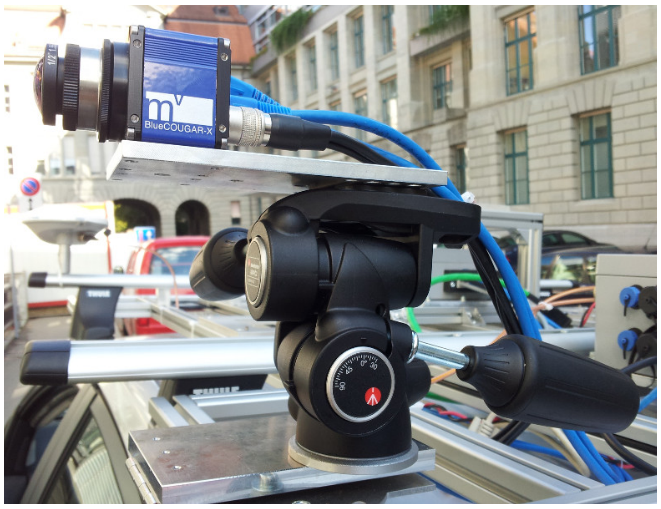


Fig. 4: The left camera is mounted on a three-way tripod head, which in turn, is mounted on a sliding plate. This sliding plate is attached to the car roof via screws.

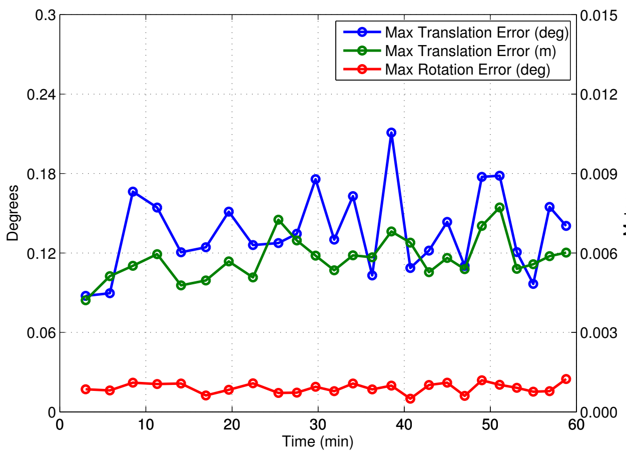


Fig. 8: A plot of the maximum errors against the time at which each loop was completed.

the CamOdoCal library which can be downloaded from `https://github.com/hengli/camodocal`.

VIII. ACKNOWLEDGMENTS

The first author is funded by the DSO National Laboratories Postgraduate Scholarship. In addition, this work is supported in parts by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant #269916 (V-Charge) and 4DVideo ERC Starting Grant Nr. 210806. We thank Jerome Maye for his hard work in setting up our experimental platform.

REFERENCES

[1] S. Agarwal, K. Mierle, and Others. Ceres solver, 2013. `https://code.google.com/p/ceres-solver/`.

[2] G. Carrera, A. Angeli, and A. Davison. Slam-based automatic extrinsic calibration of a multi-camera rig. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[3] K. Daniilidis. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research (IJRR)*, 18(3):286–298, 1999.

[4] D. Galvez-Lopez and J. Tardos. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[5] A. Geiger, F. Moosmann, O. Car, and B. Schuster. Automatic calibration of range and camera sensors using a single shot. In *International Conference on Robotics and Automation (ICRA)*, 2012.

[6] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013.

[7] C. Guo, F. Mirzaei, and S. Roumeliotis. An analytical least-squares solution to the odometer-camera extrinsic calibration problem. In *International Conference on Robotics and Automation (ICRA)*, 2012.

[8] L. Heng, B. Li, and M. Pollefeys. Camodocal: Automatic intrinsic and extrinsic calibration of a rig with multiple generic cameras and odometry. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[9] C. Hughes, P. Denny, M. Glavin, and E. Jones. Equidistant fish-eye calibration and rectification by vanishing point extraction. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(12):2289–2296, 2010.

[10] J. Kannala and S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(8):1335–1340, 2006.

[11] R. Kumar, A. Ilie, J. Frahm, and M. Pollefeys. Simple calibration of non-overlapping cameras with a mirror. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[12] P. Lebraly, E. Royer, O. Ait-Aider, C. Deymier, and M. Dhome. Fast calibration of embedded non-overlapping cameras. In *International Conference on Robotics and Automation (ICRA)*, 2011.

[13] B. Li, L. Heng, K. Köser, and M. Pollefeys. A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[14] F. Markley, Y. Cheng, J. Crassidis, and Y. Oshman. Averaging quaternions. *Journal of Guidance, Control, and Dynamics*, 30(1):12–28, 2007.

[15] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate non-iterative o(n) solution to the pnp problem. In *International Conference on Computer Vision (ICCV)*, 2007.

[16] C. Wu. Visualsfm: A visual structure from motion system, 2011. `http://homes.cs.washington.edu/~ccwu/vsfm`.