Unmanned Systems, Vol. 5, No. 2 (2017) 59–78
© World Scientific Publishing Company
DOI: 10.1142/S2301385017500054

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us

Cross-Calibration of RGB and Thermal Cameras with a LIDAR for RGB-Depth-Thermal Mapping

Aravindhan K Krishnan[*,‡], Srikanth Saripalli[†,§]

[*]School of Earth and Space Exploration, Arizona State University, Tempe, Arizona 85281, USA

[†]Department of Mechanical Engineering, Texas A&M University, College Station, Texas 77843, USA

We present a method for calibrating the extrinsic parameters between a RGB camera, a thermal camera, and a LIDAR. The calibration procedure we use is common to both the RGB and thermal cameras. The extrinsic calibration procedure assumes that the cameras are geometrically calibrated. To aid the geometric calibration of the thermal camera, we use a calibration target made of black-and-white melamine that looks like a checkerboard pattern in the thermal and RGB images. For the extrinsic calibration, we place a circular calibration target in the common field of view of the cameras and the LIDAR and compute the extrinsic parameters by minimizing an objective function that aligns the edges of the circular target in the LIDAR to its corresponding edges in the RGB and thermal images. We illustrate the convexity of the objective function and discuss the convergence of the algorithm. We then identify the various sources of coloring errors (after cross-calibration) as (a) noise in the LIDAR points, (b) error in the intrinsic parameters of the camera, (c) error in the translation parameters between the LIDAR and the camera and (d) error in the rotation parameters between the LIDAR and the camera. We analyze the contribution of these errors with respect to the coloring of a 3D point. We illustrate that these errors are related to the depth of the 3D point considered — with errors (a), (b), and (c) being inversely proportional to the depth, and error (d) being directly proportional to the depth.

Keywords: Cross-calibration; thermal cameras; RGB thermal mapping.

1. Introduction

Calibration refers to the operation that, under specified conditions establishes a relationship between quantity values along with measurement uncertainties [1]. Calibration improves the accuracy and the quality of the data acquired. Hence a good calibration is essential for any sensing system. In the context of Robotics, sensing modalities fall under two categories [26] — proprioceptive sensors and exteroceptive sensors — and the calibration procedures vary accordingly.

Proprioceptive sensors measure values internal to the robot such as motor velocity, robot heading, etc. Examples of proprioceptive sensors include IMU, GPS, Magnetometer, etc., Exteroceptive sensors obtain information about the robot's external environment and these sensors can be 'active' or 'passive'. Active sensors emit signal and measure the properties of the returns (e.g., LIDAR, TOF camera, and SONAR) whereas 'passive' sensors measure the ambient signal in the environment (e.g., color and thermal cameras). IMU calibration [33] involves determining the biases and sensitivities of the orthogonal triad of the accelerometers and gyroscopes. Camera calibration involves correcting for the pixel to pixel sensitivities [25] and estimating the intrinsic parameters [34] such as focal length, center of the image, skew, etc. Radiometric calibration [7] of thermal cameras map the pixel intensity values to the temperature of the source. The intensity calibration of a LIDAR maps the LIDAR intensity returns to the reflectance of the surface [6, 16]. In multi-beam LIDARs, calibration involves

Received 27 August 2016; Revised 26 February 2017; Accepted 26 February 2017; Published 30 May 2017. This paper was recommended for publication in its revised form by editorial board member, Han Wang.
Email Addresses: ‡aravindhan.krishnan@asu.edu, §ssaripalli@tamu.edu

estimating the offsets between the individual beams [16]. These calibration procedures ensure that the data obtained from the sensors is of high fidelity.

Multi-modal sensing presents alternate ways to solve sensing problems in Robotics and in some cases simplifies the problems to be solved. For instance, a radiometrically calibrated thermal camera is capable of detecting pedestrians and other animals purely based on their body temperature, which is much simpler than training machine learning algorithms. More descriptive place signatures can be derived from multi-modal data which can potentially help place recognition and change detection in urban environments. In multi-modal systems, calibration is necessary to register the data obtained from various sensing modalities as well as correcting errors in the individual sensors. For instance, in a LIDAR-camera system [11, 28], the relative pose between the LIDAR and the camera is essential to register the range data from the LIDAR with the color data from the camera. In a IMU-camera system, correcting the IMU biases is essential for accurate pose estimation and the IMU-camera transform is necessary to represent the estimated poses in a global frame [14]. In this paper, we present a multi-modal sensing system comprising of a color camera, a thermal camera, and a LIDAR. The calibration problem that we are trying to solve is the estimation of the relative poses between the RGB and thermal cameras and the LIDAR to produce RGB-Depth-Thermal (RGBDT) data.

[13, 15, 24] have demonstrated methods to combine data from RGB cameras and LIDAR to obtain RGBD data. But there has not been much work on fusing thermal data with LIDAR. It presents new applications like navigation and scene understanding in the dark. And to the best of the authors' knowledge only Borrmann *et al.* [5] present a work that merges data from a LIDAR, thermal, and RGB cameras to create a RGBDT map. Recently, there has been work on obtaining RGBDT data from a thermal camera and a Kinect [17, 31]. These approaches are applicable only for indoor environments. To obtain RGBDT data of outdoor environments, a couple of problems have to be solved: (A) Computing the intrinsic parameters of the thermal camera. (B) computing the extrinsics between the thermal/RGB cameras and the LIDAR. As a solution to problem (A), we describe our calibration target that looks like a checkerboard pattern in the thermal image. And for problem (B), we present our extrinsic calibration method that aligns edges in the thermal/RGB images with the edges in the LIDAR. Since we are calibrating both a color camera and a thermal camera with a LIDAR, we propose a single setup that can be used for cross-calibration of both the thermal and RGB cameras.

Typically, cross-calibration algorithms define an objective function parameterized in terms of the extrinsic parameters which is then minimized or maximized. They fall under two categories: (i) error minimization methods and (ii) information maximization methods.

Error minimization methods either reduce the reprojection error when the LIDAR points are projected onto the image (or) minimize the alignment error of planes defined in the camera frame and the LIDAR frame. Information maximization methods represent the scene content in various sensing modalities as intensity distributions and maximize the correlation between them. Our method falls under the category of error minimization methods. We define a cost function and a calibration setup that can be used for the cross-calibration of *both* RGB and thermal cameras.

A question that is not addressed in the cross-calibration literature is — what factors result in the wrong coloring of a LIDAR point after cross-calibration? In this work, we identify the various sources of coloring errors and analytically derive the conditions under which these factors affect the coloring of a LIDAR point. Additionally, we discuss a setup that allows us to quantify the coloring errors.

To summarize our contributions: (i) we present a calibration target that can be used to find the intrinsic parameters of a thermal camera. (ii) we present a single setup/method to cross-calibrate *both* RGB and thermal cameras with a LIDAR. (iii) we identify various sources of coloring errors after cross-calibration and discuss their contributions to the coloring of a 3D point. (iv) we propose a setup to quantify coloring errors and present comparative results using this setup.

2. Related Work

To merge data from a LIDAR and a camera, the extrinsic parameters (i.e. rotation and translation parameters) between the LIDAR and the camera should be known. Typically, finding the extrinsics is formulated as an optimization problem that computes the transformation (T) between the camera frame and the LIDAR frame. The first approaches to camera-LIDAR calibration used calibration targets similar to camera calibration. The calibration target is placed in the common field-of-view (FOV) of the camera and the LIDAR. The equation of the plane (in the LIDAR frame) containing the planar calibration target is computed after segmenting the LIDAR point cloud [29]. For a 2D LIDAR, a line is used instead of a plane [21]. The equation of the plane in the camera frame is computed from the extrinsics between the camera frame and the world frame (which is defined on the calibration target). The required transformation T is obtained by aligning the plane (or the line in [21]) in the LIDAR frame to the plane in the camera frame. Recently, a similar approach 'single shot calibration' has been presented in [9]. The toolbox that implements 'single shot calibration' also provides an approach to detect checkerboard corners automatically. The toolbox requires that many checkerboards are placed in the scene at once, rather than having to collect multiple scans/images of the checkerboard placed in different orientations. The above approaches rely on the plane equations of the calibration target obtained from the camera calibration to compute the cross-calibration parameters; hence are not suitable for online calibration.

Another approach is to find the pose of the camera in the LIDAR frame using the perspective-from-n-points (PnP) algorithm [24]. Corresponding points between the camera image and the LIDAR point cloud are selected by the user using an intermediate Bearing Angle Image (derived from the point cloud).

Few other recent approaches to camera-LIDAR calibration assume that the camera is calibrated and the intrinsics of the camera are known. The LIDAR point cloud (with intensity values) is projected to the camera frame using the camera intrinsics resulting in an intensity image. The cross-calibration problem is then formulated as an optimization problem that finds the transformation (T) which aligns the LIDAR intensity image with the camera image.

Algorithms that use mutual information as the optimization function are presented in [20, 27]. These methods rely on the intensity values from the LIDAR which may not be accurate. Pandey *et al.* [20] calibrate the intensity returns for their Velodyne LIDAR before doing the cross-calibration. Mutual information methods are targetless and are suitable for online calibration. However, the mutual information metric is not very useful when a high resolution image (e.g., 3 MP image) is used with a sparse point cloud containing a few thousand points. The few thousand points, when projected back to the image, contribute only to a fraction of pixels in the intensity image and the other pixels have to be interpolated [18]. So, an averaging of information occurs before the mutual information is maximized. This can result in poor convergence.

An edge alignment approach is presented by Levinson and Thrun [15] where the edges in the LIDAR point cloud are projected on to the image. The optimal parameter (T_{opt}) is determined from an initial estimate (T) by a brute force search in a discrete set of points $\mathcal{P} \in [T - \Delta, T + \Delta]$ that projects the edges in the LIDAR data to the edges in the image. The assumption here is that the edges in the point cloud which arise out of depth discontinuities have corresponding edges in the image. A very good initial estimate of the parameters is required for this solution to work as one can easily get stuck in a local minima with a local brute force search.

A calibration technique using a circular calibration target is presented in [3]. The plane containing the circle is segmented from the LIDAR point cloud and the pose of the circle is estimated in the LIDAR frame. The pose of the circle in the camera frame is estimated as described in [19]. The circles are then aligned using the ICP point-to-plane metric. Another method that uses circles as a calibration target is presented by Velas *et al.* [30].

Thermal cameras are used in various applications such as surveillance, building inspection, gas detection, human detection, etc., but are rarely used in the context of 3D mapping. A survey of the applications of thermal cameras can be found in [8]. 3D Mapping using stereo thermal cameras is described in [22] and monocular SLAM using a thermal camera is presented in [32]. Borrmann *et al.* present their work on integrating thermal camera and a LIDAR in [5]. RGBDT data provides a rich set of applications to autonomous systems such as pedestrian detection, change detection, localization, etc. And with the cost of thermal cameras going down in recent years, it presents a new mode of sensing for future robotic systems.

A RGBDT data stream of indoor environments is created in [17] by combining data from the Kinect sensor with a thermal camera. Their approach aligns the thermal image with the RGB image from the Kinect, producing RGBDT data. Since the depth information for a pixel is already available from the Kinect, their problem is reduced to an image alignment problem between the two cameras, without having to align the depth image. A similar hand held setup which creates RGBDT maps of indoor environments is presented in [31].

A fusion of thermal and depth cameras is presented in [22]. The thermal and depth images are treated as a stereo pair and circular calibration patterns are used for the stereo calibration. It requires that the cardboard containing the circles is heated in a dryer to produce a thermal contrast for circle detection.

3. Method

Our setup consists of a PointGrey CM3-U3-13S2M-CS color camera, FLIR Tau2 (long-wave-infrared, uncooled core) thermal camera, and a Hokuyo UTM-30LX LIDAR. To get 3D point clouds, the scanner is rotated using a Robotis Dynamixel PRO motor which gives ~12 encoder readings per degree. The cameras are rigidly mounted to the frame containing the LIDAR. An image of the system is shown in Fig. 1.

To obtain a 3D point cloud, we integrate the encoder readings from the motor and the 2D scan from the LIDAR based on the time stamps of corresponding ROS topic messages. 2D scans in one tilt of the LIDAR are accumulated to create a point cloud. We assume that there are no errors

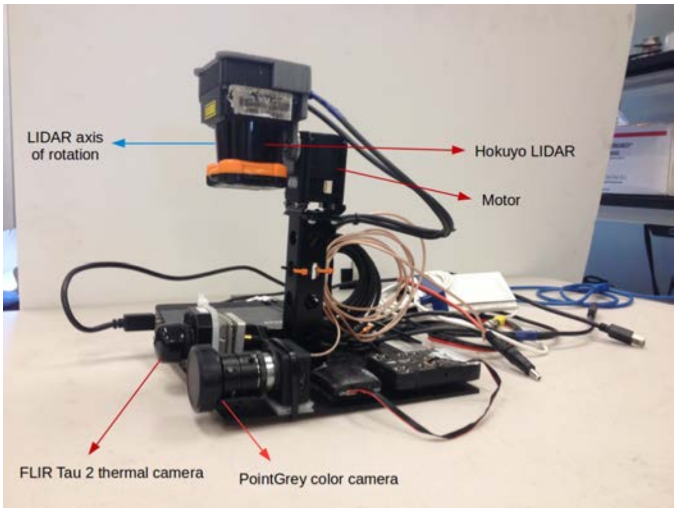
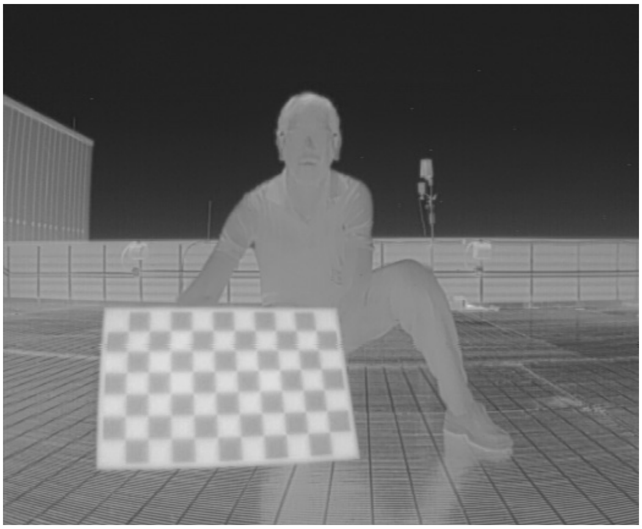


Fig. 1. The setup consisting of a Hokuyo LIDAR, PointGrey color camera, and a FLIR Tau2 thermal camera.

(a) The target made produced by cutting squares of black and white melamine in a laser cutter and gluing them alternatively on a board



(b) Image of the calibration target in the thermal camera. The intensity contrast was good enough to detect the checkerboard corners

Fig. 2. Calibration target used for thermal and RGB camera calibration.

in the encoder readings and the scene remains static during one tilt of the LIDAR (in the up-down tilt motion of the motor). Our cross-calibration method assumes that the intrinsic parameters of the cameras are known. To compute the intrinsic parameters of the thermal camera, we designed a calibration target that looks like a checkerboard pattern in the thermal image. Initially, we describe our calibration target for the thermal camera. Later, we describe the setup for the extrinsic calibration and finally the algorithm for the cross-calibration.

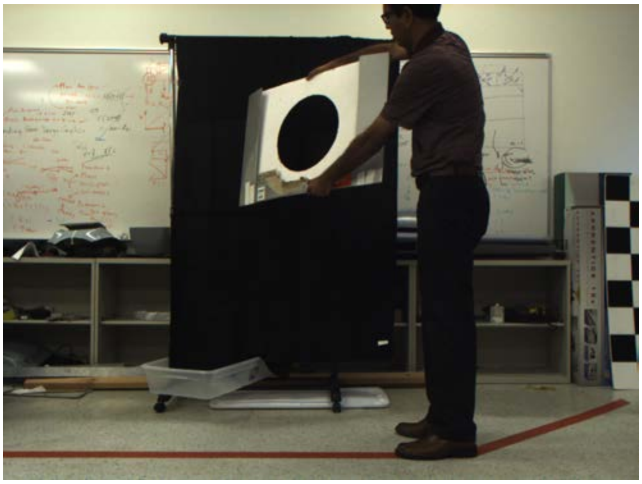
3.1. *Calibration target for the thermal camera*

The target was made by cutting squares of black and white melamine in a laser cutter and gluing them alternatively on a board. First, a black rectangular frame is cut and glued to the board — which acts as a border. The squares (2×2 inches) are then glued inside, starting from one of the corners, ensuring that they are placed tightly without any gaps in between. Because of the color contrast, the same target is used for RGB camera calibration too. When placed in the sun, the different albedo of the white and black squares creates the thermal gradient needed for camera calibration.

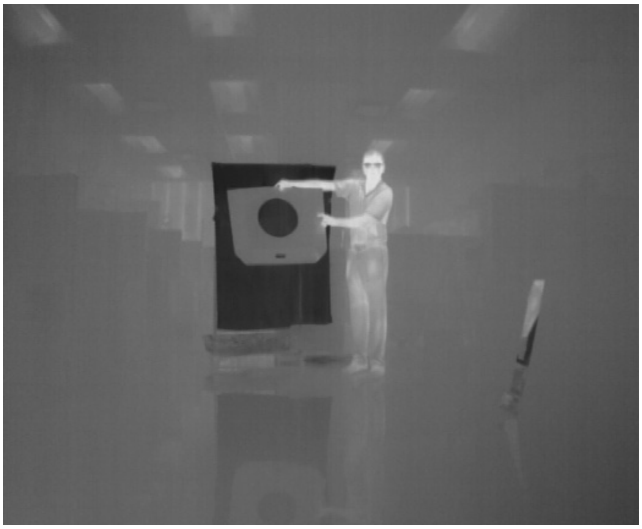
The images of the calibration target in the color camera and the thermal camera is shown in Figs. 2(a) and 2(b), respectively. We use the RADOCC toolbox [12] for camera calibration. As seen in Fig. 2(b), the intensity contrast in the thermal image was good enough for detecting the checkerboard corners. 40 images of the calibration target were obtained from various orientations and we achieved a reprojection error of 0.24 pixels.

3.2. *Cross-calibration setup*

The cross-calibration algorithm aligns the edges of a circular target in the LIDAR to its corresponding edges in the color and thermal images. Hence we designed a setup that facilitates the accurate detection of edges in different sensing modalities. A circle of known radius was cut out from a white cardboard. The cardboard (which can be of any color) is then placed before a background of a contrasting color. We draped a wet black cloth over a garment rack and used it as a background. The black color produces a contrast for the color camera and the wetness of the cloth



(a) Image of the calibration setup in the RGB camera. The black cloth provides the necessary contrast to detect the circle



(b) Image of the calibration setup in the thermal camera. The wet (black) cloth provides the necessary contrast to detect the circle

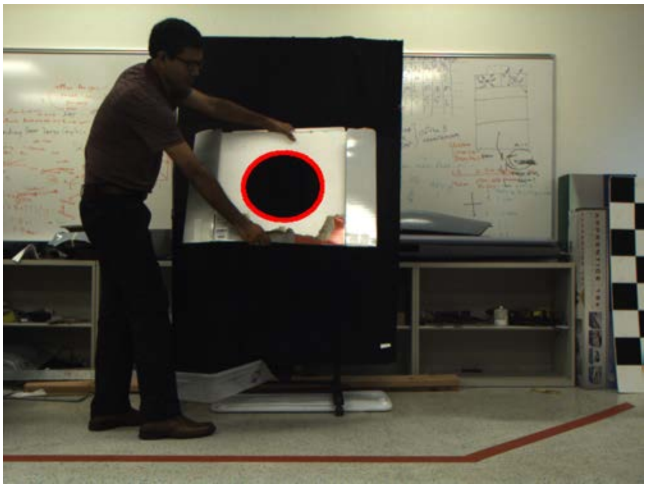
Fig. 3. Calibration target used for thermal and RGB camera calibration.

produces the intensity contrast in the thermal image. The edges of the circle were detected across all the sensing modalities and the cross-calibration algorithm computes the extrinsics that aligns the edges in the images with the edges in the LIDAR. Images of the calibration setup for the color and thermal cameras are shown in Figs. 3(a) and 3(b), respectively.

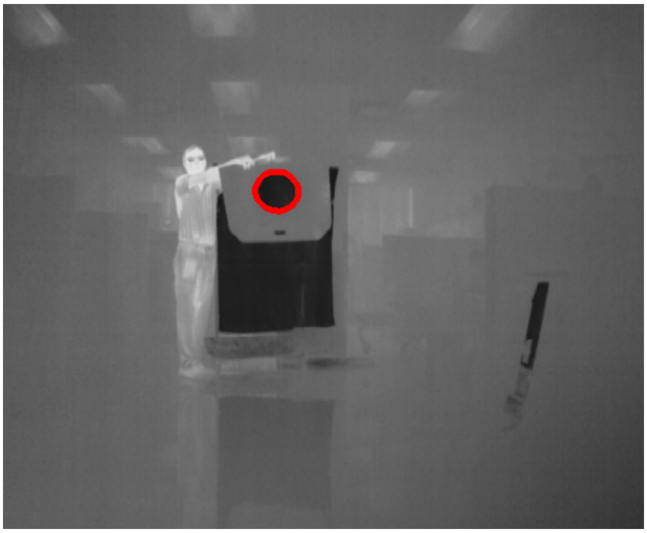
3.3. *Detecting edges in the image*

We use a region growing algorithm to detect the edges of the circle in the color and thermal images. The user clicks a point within the circle and the clicked point is used as the *seed* point for the region growing segmentation. The region is grown by adding the neighboring pixels to the segment when the pixel intensities are similar. The boundary of the grown segment represents the edges of the circle in the image. The detailed work flow is described in Algorithm 1.

To begin with, our algorithm creates a binary image that is of the same size of the input image. All pixels in the binary image are initialized to 1. A queue is used to keep track of the recently added pixels to the region. Initially, the *seed* point is pushed to the queue. The 8 neighbors of the seed point are compared with the seed point and are pushed to the queue if the intensities match within a given threshold. Whenever a point is pushed to the queue the corresponding



(a) Color image - Detected edge shown in red



(b) Thermal image - Detected edge shown in red

Fig. 4. Detecting the edges of the circular target in the image.

pixel in the binary image is set to 0. After the neighbors of a pixel are compared for similar intensity values, the pixel is removed from the queue. This process is repeated until the queue becomes empty. The set of all pixels that were pushed to the queue represents the circular region and the corresponding pixels in the binary image have 0 intensity values.

A list of points $\mathcal{L}$ that were pushed to the queue is maintained. This list is used to detect the boundary of the estimated segment and the boundary corresponds to the edges of the circular target. For every point in $\mathcal{L}$, the corresponding neighbors in the binary image is considered. It is obvious that pixels that have at least one nonzero neighbor constitute the boundary of the circular region and these pixels correspond to the edges of the circular target. Figures 4(a) and 4(b) show the detected edges in color and thermal images, respectively.

3.4. *Detecting edges in the LIDAR point cloud*

The edge detection algorithm requires that the point cloud is represented in cartesian (x, y, z) as well as spherical coordinates (r, θ, ϕ). The range r and scan angle θ are obtained from a single scan in the 2D laser scanner and the tilt angle

Algorithm 1: Detecting edges of the circular target in the image

Input : 1) Gray scale image $\mathcal{I}$ of the cross-calibration setup 2) Seed point p 3) Pixel intensity threshold ϵ
Output: Edge points $\mathcal{E}$ of the circular calibration target

```
/* Binary image                                                      */
BI = CreateImage (I.rows, I.cols);
BI = FillOnes ();
Q = ∅
Q = AddToQueue (p);
L = ∅
while Q.notEmpty () do
    point = Q.top ();
    Q.pop ();
    N = FindNeighbors (point);
    foreach neighbor ∈ N do
        if | neighbor.intensity - point.intensity | ≤ ε then
            /* Point already selected                                */
            If BI (neighbor) = 0  continue
            else
                Q.push (neighbor);
                L ← L ∪ neighbor
                BI (neighbor) = 0;
            end
        end
    end
end
E = ∅
foreach point ∈ L do
    N = FindNeighbors (point);
    foreach neighbor ∈ N do
        /* Atleast one neighbor is not set to 0 in the previous step */
        if BI (neighbor) == 1 then
            E = E ∪ point
            break
        end
    end
end
```

Algorithm 2: Detecting edges of the circular target in the point cloud

Input : 1) Point Cloud $\mathcal{FR}$ represented in cartesian as well as spherical co-ordinates 2) Clicked point p 3) Plane fit threshold δ 4) Edge detection threshold $\delta\theta$

Output: 3D edge points $\mathcal{E}$ of the circular calibration target

$\mathcal{FR}_\mathcal{F}$ = SACPlaneSegmentation ($\mathcal{FR}$)
$\mathcal{FR}_\mathcal{B}$ = $\mathcal{FR} - \mathcal{FR}_\mathcal{F}$
`/* If the plane in the background is chosen as the foreground in the previous step */`
if $p \in \mathcal{FR}_\mathcal{B}$ **then**
    Swap ($\mathcal{FR}_\mathcal{B}$, $\mathcal{FR}_\mathcal{F}$);
end
`/* Removing the noisy edge points */`
$\mathcal{C}$ = RANSAC_PlaneFit ($\mathcal{FR}_\mathcal{F}$, δ);
Φ = ComputeUniquePhiList ($\mathcal{C}$);
`/* Sorting points by scan line */`
$\mathcal{P} = \emptyset$
foreach $\phi \in \Phi$ **do**
    p_s = FindPointsWithPhi (ϕ)
    p_s = SortByTheta (p_s)
    $\mathcal{P} = \mathcal{P} \cup p_s$
end
`/* Detecting the edges of the circle */`
$\mathcal{E} = \emptyset$
foreach $p_s \in \mathcal{P}$ **do**
    foreach $p_s^i \in p_s$ **do**
        if $|p_s^i.theta - p_s^{i+1}.theta| \geq \delta\theta$ **then**
            $\mathcal{E} = \mathcal{E} \cup p_s^i$
            $\mathcal{E} = \mathcal{E} \cup p_s^{i+1}$
        end
    end
end

of the motor ϕ is obtained from the motor. Initially, the user clicks on a point $\mathcal{P}$ on the cardboard containing the (cut-out) circle. Points that are within a radius R of the clicked point are filtered and the circle edges are detected in the filtered segment. (R is chosen sufficiently large that it covers the size of the cardboard containing the circle.) The work flow is described in Algorithm 2.

Since the filtered region ($\mathcal{FR}$) represents points that are within a radius R of the clicked point, it also includes points that are not on the cardboard, i.e., points that are on the black cloth in the background. So the filtered region contains two planes — one representing the cardboard and another representing the cloth in the background. As a first step, we extract the plane containing the cardboard from the filtered region.

Sample Consensus (SAC) segmentation method is used to detect 'a' plane in the filtered region — this can be either the cardboard in the foreground ($\mathcal{FR}_\mathcal{F}$) or the cloth in the background ($\mathcal{FR}_\mathcal{B}$). The implementation from the Point Cloud Library [23] is used for the SAC segmentation. Given $\mathcal{FR}_\mathcal{F}$, one can compute $\mathcal{FR}_\mathcal{B} = \mathcal{FR} - \mathcal{FR}_\mathcal{F}$. Similarly, $\mathcal{FR}_\mathcal{F}$ can be computed given $\mathcal{FR}_\mathcal{B}$ and $\mathcal{FR}$. The region (among $\mathcal{FR}_\mathcal{F}$ and $\mathcal{FR}_\mathcal{B}$) containing the clicked point $\mathcal{P}$ is the plane containing the circular target and is further considered for detecting the edges.

To filter out any noisy edge points in the region obtained in the previous step, we again fit a plane using RANSAC on this region with a plane fitting threshold of 2 cm. This filters out all the noisy edge points and we get a new set of points $\mathcal{C}$. The new region is shown in Fig. 5(a).

Algorithm 3: Generating points on the 3D circle

Input : 1) Detected edge points $\mathcal{E}$. 2) Radius of the circle r. 3) Output circle resolution $\Delta\theta$.
Output: 3D edge points $\hat{\mathcal{E}}$ of the circular target at a resolution $\Delta\theta$

Σ = ComputeCovariance ($\mathcal{E}$);
$\{\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3\}$ = EigenVectors (Σ)
/* The eigen vector corresponding to the smallest eigen value is the normal vector */
$n = \mathcal{V}_3$
$\hat{c}$ = $\text{argmin}_c \sum_i [p_i - c]^T [p_i - c] - r^2 + n \cdot [p_i - c]$ where $p_i \in \mathcal{E}$.
/* Points on the XY plane */
$\mathcal{E}_{2\text{d}} = \emptyset$
for *(θ = 0; $\theta \leq$ 360; $\theta + = \Delta\theta$)* **do**
  $p_{2d} = [r\cos(\theta),\ r\sin(\theta),\ 0]^T$
  $\mathcal{E}_{2\text{d}} = \mathcal{E}_{2\text{d}} \cup p_{2d}$
end
/* Points on the 3D circle */
$\hat{\mathcal{E}} = \emptyset$
foreach $p_{2d} \in \mathcal{E}_{2d}$ **do**
  $p_{3d} = [p_{2d} \cdot (\mathcal{V}_1),\ \ p_{2d} \cdot (\mathcal{V}_2),\ \ p_{2d} \cdot (\mathcal{V}_3)]^T + \hat{c}$
  $\hat{\mathcal{E}} = \hat{\mathcal{E}} \cup p_{3d}$
end

Since we also maintain the spherical co-ordinate representation for the LIDAR points, we consider points in $\mathcal{C}$ scan line by scan line (a scan line represents all points (r, θ, ϕ) for a particular ϕ) and look for any discontinuities in θ for adjacent points in a scan line. These discontinuities correspond to the background points that were filtered in the Sample Consensus step described earlier. And the points at the point of discontinuity represent the edge points ($\mathcal{E}$) of the circle. The detected edge points can be seen in Fig. 5(b). The RANSAC plane fitting procedure is not perfect and does include some noisy edge points (that are within the plane fitting threshold). So not all circle edges are detected and we interpolate the other edge points of the circle as described in the next section.

3.5. *Generating all points on the circle*

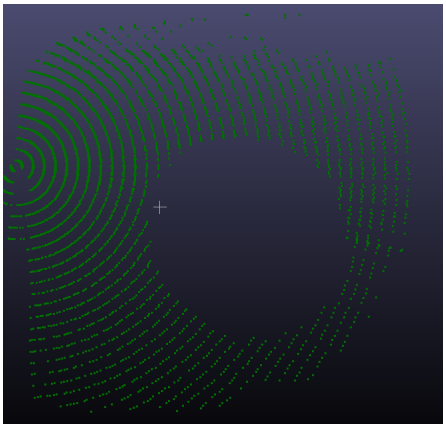
As seen in Fig. 5(b), the edge detection method described in the previous section does not compute all edge points of the circle. To generate the remaining points on the circle, we estimate the parameters of the circle by doing a least squares fit on the detected edge points ($\mathcal{E}$). The parameters of a circle are the radius (r) and the center (c). In our case r is known, so we estimate the center using least squares. The least squares cost function that we minimize is given below, where $p_i \in \mathcal{E}$ are the detected edge points, n is the normal to the plane containing the edge points and c is the parameter to estimate.

$$f(c) = \sum_i [p_i - c]^T [p_i - c] - r^2 + \underbrace{n \cdot [p_i - c]}_{\text{A}}. \qquad (1)$$

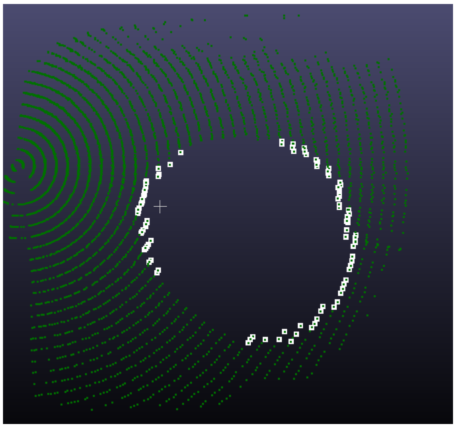
The above function is convex, so minimizing this function gives a unique minima. We use Levenberg–Marquardt algorithm to minimize the cost function and obtain the center of the circle 'c'. Part A ensures that the resulting center lies on the plane containing the edge points. (n is the normal to the plane and $[p_i - c]$ is a vector on the plane containing the circle. So the dot product $n \cdot [p_i - c]$ should be 0.)

To generate the points on the 3D circle, we first generate points ($\mathcal{E}_{2\text{d}}$) on a 2D circle (*XY* plane) from the computed parameters. We then compute the orthogonal basis vectors $(\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3)$ of the detected edge points ($\mathcal{E}$). The basis vectors are given by the eigen vectors of the covariance matrix of $\mathcal{E}$. The points in $\mathcal{E}_{2\text{d}}$ are projected on to the orthogonal basis vectors $\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3$ to obtain the points on the 3D circle ($\hat{\mathcal{E}}$). The procedure is described in Algorithm 3 and the result in shown in Fig. 5(c).

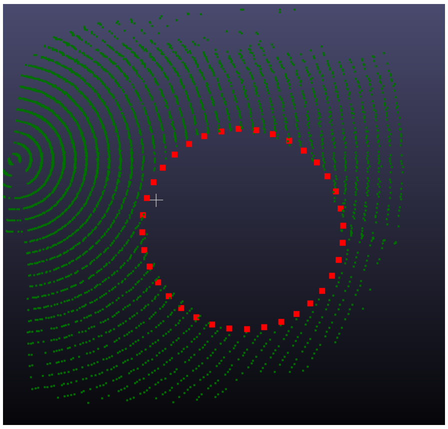
Generating the missing points of the circle using a least squares approach in not mandatory but it provides more data points for minimizing Eq. (2).



(a) Detected plane



(b) Detected circle edges



(c) Circle edges reconstructed

Fig. 5. Detecting the edges of the circular target in the LIDAR point cloud.

3.6. *Computing the extrinsics between the cameras and the LIDAR*

The circular target is placed in different orientations and places in the common field of view of the LIDAR and the cameras. The relative transformation is computed by aligning the edges of the circular target in the LIDAR point cloud to the edges of the target in the images. We solve the edge alignment problem as an optimization problem. The optimization function is shown below.

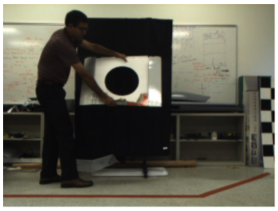
$$T_{\text{optimal}} = \underset{T}{\operatorname{argmin}} \sum_j \sum_i ||I_j^c - \mathcal{N}(K * T * E_j^i)||. \tag{2}$$

Here, the index j refers to the 'image–point cloud' pair containing the same view of the calibration target. E_j^i are the edge points of the circular target in the jth point cloud. T is the parameter we are estimating. K is the camera matrix containing the intrinsic parameters. $\mathcal{N}(\cdot)$ produces the normalized image co-ordinates. I_j^c is the closest edge point in the jth image, when E_j^i is projected on to the camera.

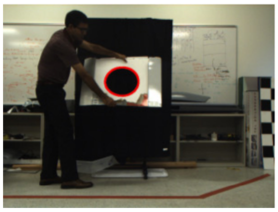
The optimization function in Eq. (2) is implemented as a distance transform of the edges computed on the image. For an edge image, the distance transform of a pixel gives the distance to the closest edge pixel. When the 3D point is projected to a pixel (x, y) in the image, the corresponding distance transform at (x, y) gives the distance to the closest edge pixel in the image. When the distance transform is computed only on the edges of the circular target, it is precisely the optimization function in Eq. (2). An example distance transform is shown in Fig. 6.

One can notice that the distance transform looks convex for most part as the distance to an edge pixel grows gradually towards the boundary of the image. We used the Levenberg–Marquardt (from Ceres solver [2]) algorithm for optimization.

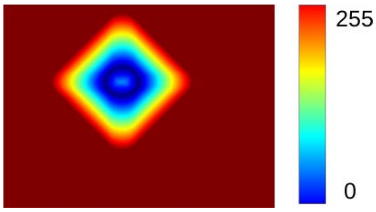
We used numerical differentiation to compute the gradients for minimizing Eq. (2). The *central difference* method for computing the gradient of a function f at a point T is given by $\frac{f(T+\Delta)-f(T-\Delta)}{2\Delta}$, where Δ is usually chosen to be $\sim$1e-6. But the optimization function in Eq. (2) is a distance transform and both $f(T)$ and $f(T+\Delta)$ result in the same pixel x when $\Delta =$1e-6, producing a zero gradient. This causes the optimization algorithm to terminate early.



(a) An image of the calibration setup



(b) Edges of the circular target shown in red



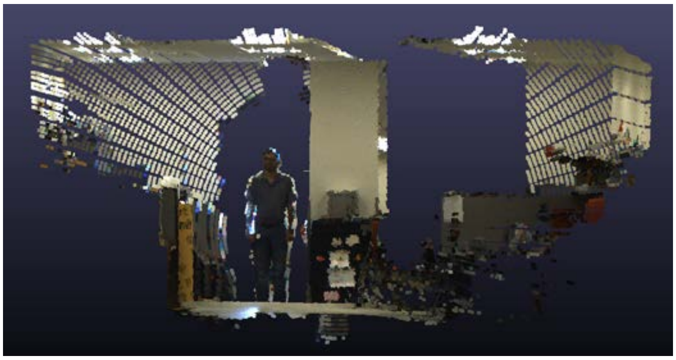
(c) Distance transform of the detected edge points of the circular target colored with 'jet' color map.

Fig. 6. Distance transform.

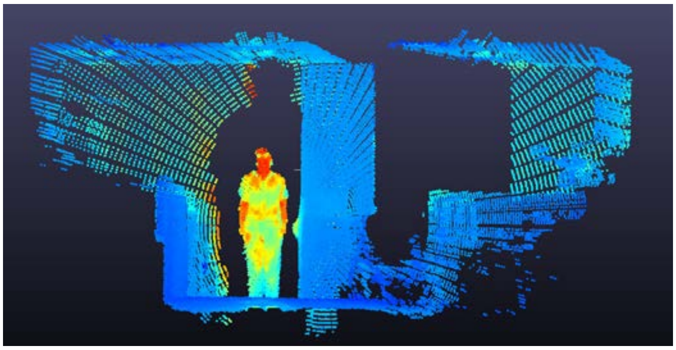
To overcome this, we fit a plane on the distance transform values for a 10×10 window centered around each pixel (T). The plane represents a parameterized continuous surface and we can obtain the objective function values even for small deviations Δ centered around a pixel T. With this surface approximation, the *central difference* method for computing gradients is possible and the optimizer converges. The plane is computed only on pixels onto which the LIDAR points are projected. The plane parameters (computed per pixel) are stored in a look-up table to avoid recomputation on future iterations of the optimization.

4. Results and Discussion

Images of a point cloud textured by our algorithm with color and thermal images are shown in Figs. 7(a) and 7(b), respectively. In this section, we discuss various aspects of the cross-calibration problem. We begin with a discussion on the smoothness/convexity of the objective function in Eq. (2). We then report the timing analysis of our solution. Later, we discuss the various factors that can result in the wrong coloring of a 3D point after cross-calibration.



(a) Point cloud textured with a color image



(b) Point cloud textured with a thermal image

Fig. 7. Textured point clouds.

We discuss a setup that allows us to quantify the coloring errors and compare our solution to the method described in [20] which is based on maximizing mutual information. We hypothesize why mutual information methods do not work for relative pose estimation between a thermal camera and a LIDAR. Finally, we present some results on our RGBDT mapping in outdoor environments during day and night.

4.1. *Convexity of the objective function*

The optimization function in Eq. (2) is a nonlinear function, as the transformation matrix T includes sine and cosine terms. So we investigated the convexity of the function by varying the number of point cloud–image pairs used for optimization. Figures 8(a) and 8(b) show the variation of the objective function as we vary one extrinsic parameter (roll, pitch, yaw, tx, ty, and tz) while keeping others constant. It is noticeable that the function becomes smooth as the number of scans is increased. The smooth function ensures that the optimizer does not encounter discontinuous points on the objective function, thus helping convergence. We observed that $\geq$8 scans were sufficient for the optimization.

4.2. *Convergence analysis*

To test the convergence of the algorithm, we simulated a test dataset consisting of circles placed at different orientations. A camera with known intrinsics and extrinsics was defined and the 3D points were projected on to the image. The distance transform of the projected circles were then computed. The extrinsics were recovered from the optimization and compared to the ground truth transformation.

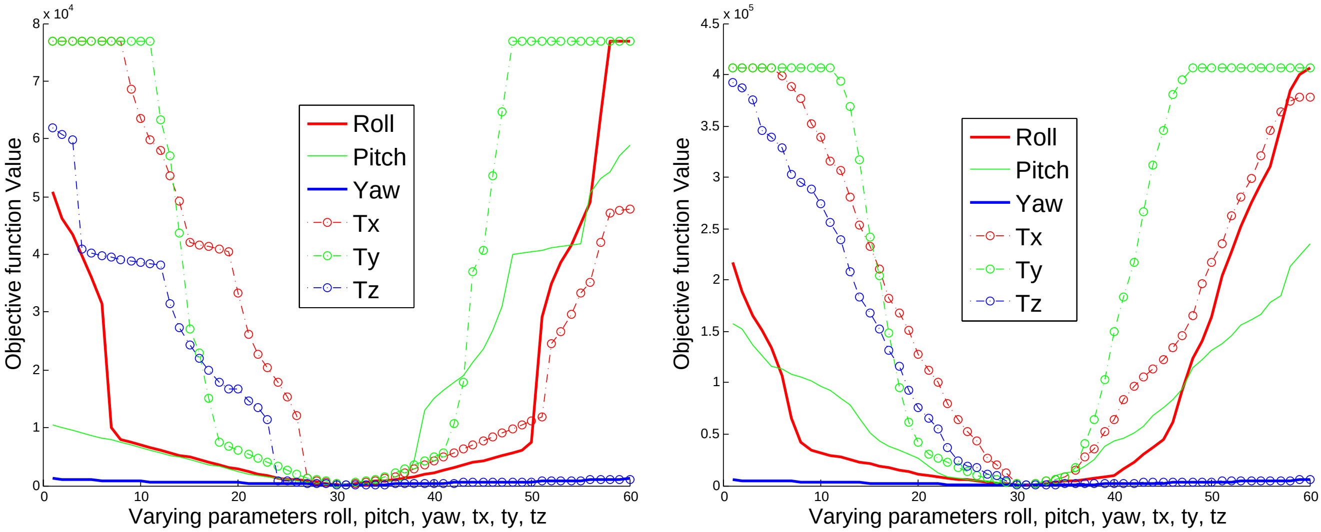
Table 1 presents the errors observed for various values of the initial estimates along with the runtime. Eight circles defined by 360 points each were used in these tests, so a total of 2880 points were used. We did not observe a huge difference in the runtime for various initial configurations, with the maximum time taken being $\sim$36 s.

4.3. *Coloring error after cross-calibration: An analysis*

There are three sources of coloring error after cross--calibration

(1) Noisy 3D points (arising from the LIDAR),
(2) Error in the camera calibration parameters (K matrix),
(3) Error in the extrinsic parameters between the camera and the LIDAR.

These errors manifest in coloring a 3D point after cross-calibration. We will analyze each source of error in this section.



(a) 2 scans were used to create the above plot.
The function is not smooth

(b) 10 scans were used to create the above plot.
The function becomes smooth as the number of scans are increased

Fig. 8. (a) and (b): The variation of parameters roll, pitch, yaw, tx, ty, and tz around the global minimum. Roll, pitch, and Yaw were varied by −30 degrees to +30 degrees around the global minimum. Tx, Ty, and Tz were varied by −3 m to +3 m around the global minimum. The Y-axis shows the objective function values. One can notice that the function appears smooth and there are no steep local minima.

4.3.1. *Noisy 3D points* (*arising from the LIDAR*)

In the context of cross-calibration the reprojection error for a ‘single’ 3D point is only a discrete quantity (as it represents the distance to the closest edge pixel). So the noisy 3D point *does not* contribute to the reprojection error unless the error is ≥ 1. We will base our analysis on ‘When does the noisy 3D point introduce a reprojection error of 1?’. The projection of a 3D point on an image is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{3}$$

Throughout our error analysis, we consider only the x coordinate of the point on the image. x^g refers to the ground truth value of the pixel, and x^p refers to the projected point (with noise).

$$x^g = f_x \frac{X}{Z} + c_x. \tag{4}$$

The projection of a noisy 3D point on the image is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X + \Delta x \\ Y + \Delta y \\ Z + \Delta z \end{bmatrix}, \tag{5}$$

$$x^p = f_x \frac{X + \Delta x}{Z + \Delta z} + c_x. \tag{6}$$

The variation of $|x^g - x^p|$ with respect to Z is shown in Fig. 9(a). It is noticeable that $|x^g - x^p|$ reduces with depth. This can be explained from Eqs. (4) and (6). As Z grows $Z + \Delta z \approx Z$, so

$$|x^g - x^p| \approx \left| f_x \frac{X}{Z} +, c_x -, f_x \frac{X + \Delta x}{Z} -, c_x \right| = f_x \frac{\Delta x}{Z}. \tag{7}$$

In the above equation, f_x and Δx are constant. As Z increases, the ratio approaches 0, which explains the graph in Fig. 9(a). The physical interpretation of this phenomenon is that the foot print of a pixel grows with depth, and for a given noise (Δx), the noisy 3D point lies within the pixel foot print beyond a particular depth. A reprojection error is introduced when $|x^g - x^p| = 1$ and this happens at a particular depth Z beyond which the reprojection error does not contribute to the coloring error. Now let us derive the depth at which the reprojection error is 1.

$$\left| f_x \frac{X}{Z} +, c_x -, f_x \frac{X + \Delta x}{Z + \Delta z} -, c_x \right| = 1,$$

$$\left| \frac{f_x XZ + f_x X \Delta z - f_x XZ - f_x \Delta x Z}{(Z)(Z + \Delta z)} \right| = 1,$$

$$\left| \frac{f_x X \Delta z - f_x \Delta x Z}{Z^2 + Z \Delta z} \right| = 1.$$

Assuming that $\Delta x = \Delta z$

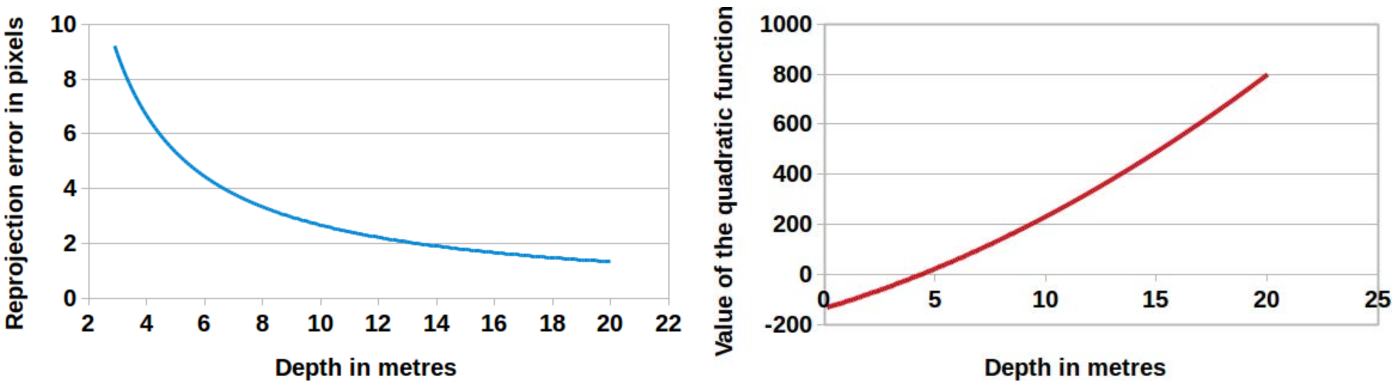
$$\left| \frac{(f_x X - f_x Z) \Delta z}{Z^2 + Z \Delta z} \right| = 1, \tag{8}$$

$$Z^2 + Z \Delta z = f_x X \Delta z - f_x Z \Delta z,$$

$$Z^2 + Z(\Delta z + f_x \Delta z) - (f_x X \Delta z) = 0 \tag{9}$$

Table 1. Convergence analysis: This table reports the errors observed when the optimizer was initialized with different initial estimates. We observe that our method converges when the initial estimates are off by 15° for rotation parameters and by 1.25 m for the translation parameters. The maximum time taken for convergence in these experiments is 36 s.

Parameters varied (angles in degrees, translations in m)						Errors (deviation from ground truth, angles in degrees, and translations in cm)						
roll	pitch	yaw	tx	ty	tz	roll	pitch	yaw	tx	ty	tz	Time taken
15	0	0	0	0	0	0.0523168	0.144605	0.0859966	0.8135	0.210657	0.956593	17.436
10	0	0	0	0	0	0.0940413	0.0845955	0.00993841	0.6244	0.891273	0.0529625	19.455
5	0	0	0	0	0	0.0643341	0.0247152	0.0205116	0.254	0.712511	0.337724	20.077
0	15	0	0	0	0	0.0034421	0.245484	0.0304735	2.2636	0.0275577	0.92469	22.545
0	10	0	0	0	0	0.021747	0.0970168	0.113332	1.0269	0.196635	0.885022	18.75
0	5	0	0	0	0	0.0182029	0.0952324	0.00114047	0.5024	0.0449326	0.547343	15.204
0	0	15	0	0	0	0.0356087	0.0811664	0.00520452	0.9209	0.442558	0.297665	19.946
0	0	10	0	0	0	0.0660356	0.0797771	0.0160321	0.6626	0.379972	0.644084	17.063
0	0	5	0	0	0	0.0373623	0.368059	0.0916428	3.4867	0.387682	0.857359	18.971
15	5	0	0	0	0	0.111903	0.182906	0.00517674	1.1868	0.840184	1.47685	30.003
10	5	0	0	0	0	0.2341	0.229946	0.0213931	1.8346	1.73832	0.481518	20.017
5	5	0	0	0	0	0.00364017	0.00398985	0.00160532	0.021859	0.0156679	0.00669822	36.671
0	15	5	0	0	0	0.228048	0.0640444	0.142484	0.5593	1.58886	1.25486	22.305
0	10	5	0	0	0	0.00148157	0.000827679	0.000937362	0.00112	0.00581074	0.0067958	17.726
0	5	5	0	0	0	0.00388925	0.000493943	0	0.001451	0.0190742	0.00401433	15.951
15	0	5	0	0	0	0.000291771	0.0009166	0.000238562	0.003265	0.000997543	0.00186952	33.825
10	0	5	0	0	0	0.000168009	0.000667008	0.000775507	0.003628	0.000329497	0.00703347	19.692
5	0	5	0	0	0	0.00185809	0.000604628	0.000973266	0.003305	0.00842194	0.0110486	16.2
0	0	0	1.25	0	0	0.0397519	0.03962	0.0141592	0.0001	0.084576	0.366757	14.48
0	0	0	0.5	0	0	0.00104444	0.00194265	0.00117359	0.007521	0.00468133	0.000771878	21.458
0	0	0	0	1.25	0	0.026567	0.0749654	0.0492644	0.6719	0.388302	0.714497	24.18
0	0	0	0	0.5	0	0.00135704	0.00204081	0.000186389	0.010989	0.00522841	0.00486685	21.75
0	0	0	0	0	1.25	0.0770089	0.0608532	0.0204936	0.3585	0.662404	1.395	17.233
0	0	0	0	0	0.5	0.000743211	0.00347903	0.000501473	0.01632	0.00306397	0.00910646	14.13
0	0	0	0	1.25	1.25	0.232977	0.03549	0.0222843	0.2595	1.8212	0.904681	19.449
0	0	0	1.25	0	1.25	0.064379	0.0164796	0.0558783	0.2096	0.602648	0.664117	23.588
0	0	0	1	1.25	1.25	0.000296194	0.000779906	0.000224807	0.006364	0.00402595	0.0159362	13.174
0	0	0	1.25	1	1.25	0.000146178	0.00474163	0.000616265	0.023547	0.00203176	0.000038664	24.898
0	0	0	1.25	1.25	1	0.19629	0.109361	0.0800192	0.9352	1.5647	0.0430677	25.5



(a) Variation of reprojection error with the depth of the 3D point; it decreases with depth

(b) Quadratic curve in z. The x intercept of this curve give the depth at which the reprojection error is 1

Fig. 9. Reprojection error verus depth of the 3D point.

which is a quadratic equation in Z, where the quantities within the braces are known (Δz modeled from LIDAR noise characteristics, f_x known from camera calibration, and X is an arbitrary point chosen such that it is projected on to the image, for e.g., the center of the image). The solution to Eq. (9) gives the depth beyond which the reprojection error is ≤ 1 and hence there is no coloring error. As an example, we considered $f_x = 1335.054$ (focal length of our color camera), and $\Delta z = 0.02$ (2 cm noise for our LIDAR). We obtained $Z = 4.303249$ as the root of Eq. (9). The quadratic function in Eq. (9) for the considered values is plotted in Fig. 9(b). In case of multiple roots to the quadratic equation, the largest root is chosen (negative roots are ignored as depth is a non-negative quantity, and the largest root gives the depth beyond which the reprojection error is always ≤ 1). Similarly, we obtained $Z = 3.512751$ for our thermal camera ($f_x = 417.20215$, $\Delta z = 0.02$).

4.3.2. *Error in the camera calibration parameters (K matrix)*

The reprojection error in camera calibration translates to errors in the intrinsic parameters f_x, f_y, c_x, and c_y as well as the extrinsic parameters (between the camera frame and the world frame) R and T. However, the extrinsic parameters R and T will not be used in the cross-calibration. So we consider only the errors in the intrinsic parameters when analyzing cross-calibration errors. We can assume that we have a perfect R and T and all errors are in the intrinsic parameters. This way, we are considering only the worst case error in the intrinsic parameters for any further error analysis. In this section, we consider the contribution of both the LIDAR noise and the camera calibration error towards the cross-calibration. The projection of a 3D point on an image is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{10}$$

$$x^g = f_x \frac{X}{Z} + c_x. \tag{11}$$

The projection of a noisy 3D point on the image with errors in intrinsic parameters and LIDAR noise is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} \begin{bmatrix} f_x + \Delta f_x & 0 & c_x + \Delta c_x \\ 0 & f_y + \Delta f_y & c_y + \Delta c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X + \Delta x \\ Y + \Delta y \\ Z + \Delta z \end{bmatrix}, \tag{12}$$

$$x^p = (f_x + \Delta f_x)\left(\frac{X + \Delta x}{Z + \Delta z}\right) + (c_x + \Delta c_x). \tag{13}$$

A reprojection error is introduced when $|x^g - x^p| = 1$. We proceed similar to the previous section and arrive at the following equation.

$$Z^2 + Z(f_x \Delta x + \Delta f_x \Delta x + \Delta z) - (\Delta z f_x X + \Delta z \Delta f_x X) = 0. \tag{14}$$

Again, this equation provide a constraint on Z at which the reprojection error for cross-calibration is 1. Things to notice are

(1) All terms within the braces are known. Δf_x is given by the output of the camera calibration toolbox. X is chosen such that the point lies within the image (e.g., the center of the image) and the error analysis is based on the chosen value of X.
(2) Δf_x provides an additional constraint on Z similar to the LIDAR noise.
(3) Intuitively, one would expect that the solution for Z would be greater than the one obtained in the previous section.

As an example, we considered $f_x = 1335.054$ (focal length of our camera), and $\Delta z = 0.02$ (2 cm noise for our LIDAR), and $\Delta f_x = 2$ (uncertainty reported by the camera calibration toolbox). We obtained $Z = 4.304037$ as the root of Eq. (14). It is noticeable that the Z obtained here (4.304037) is greater than the Z obtained in the previous section (4.303249). Similarly, we obtained $Z = 3.524249$ for our thermal camera ($\Delta z = 0.02$ and $\Delta f_x = 6$).

4.3.3. *Error in the extrinsic parameters*

For simplicity, let us analyze the error in rotation components and the translation components separately.

Error in rotation components

When there is a rotation between the LIDAR frame and the camera frame, the projected point on the image is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} [R] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \tag{15}$$

Now, let us consider only the error in yaw. The projected point on the image when there is no error in yaw is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{16}$$

$$x^g = f_x \frac{X\cos(\theta) - Y\sin(\theta)}{Z} + c_x. \tag{17}$$

The point on the image when there is error ($\Delta\theta$) in yaw is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta+\Delta\theta) & -\sin(\theta+\Delta\theta) & 0 \\ \sin(\theta+\Delta\theta) & \cos(\theta+\Delta\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix},$$

$$x^p = f_x \frac{X\cos(\theta+\Delta\theta) - Y\sin(\theta+\Delta\theta)}{Z} + c_x. \tag{18}$$

The reprojection error is given by $|x^p - x^g|$. Making the assumption that $\theta = 0$, we get

$$x^p - x^g = \frac{f_x X\cos(\Delta\theta) - ZY\sin(\Delta\theta) - f_x X}{Z}. \tag{19}$$

In the above equation, Z is both in the numerator and the denominator. So, reprojection error and the depth are not inversely related in this case. Figure 10 explains the situation.

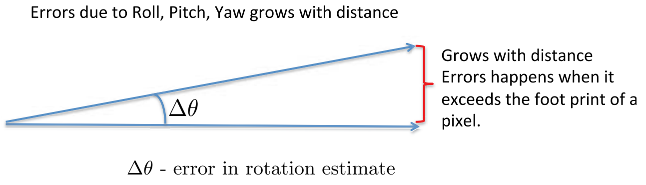


Fig. 10. Coloring error due to error in rotation.

Error in translation components

When there is a translation between the LIDAR frame and the camera frame, the projected point on the image is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}, \tag{20}$$

$$x^g = \frac{f_x(X + t_x) + c_x(Z + t_z)}{Z + t_z}. \tag{21}$$

When there is an error in translation, the projected point on the image is given by

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_x + \Delta t_x \\ t_y + \Delta t_y \\ t_z + \Delta t_z, \end{bmatrix}, \tag{22}$$

$$x^p = \frac{f_y(X + t_x + \Delta t_x) + c_x(Z + t_z + \Delta t_z)}{Z + t_z + \Delta t_z}. \tag{23}$$

As Z grows, $Z + t_z + \Delta t_z \approx Z + t_z$. So

$$x^p \approx \frac{f_x(X + t_x + \Delta t_x) + c_x(Z + t_z)}{Z + t_z}. \tag{24}$$

The reprojection error is given by

$$x^p - x^g = \frac{f_x \Delta t_x}{Z + t_z}. \tag{25}$$

The numerator is constant in the above equation and the denominator grows with depth. So as Z increases, the error in coloring due to translation error reduces.

To summarize our analysis of coloring errors

- Errors in LIDAR points and the estimation of the camera matrix do not affect the coloring of a 3D point beyond particular depth.
- Errors in the estimation of the relative translation between the camera and the LIDAR do not affect the coloring of a 3D point beyond a particular depth.
- Errors in the estimation of the relative rotation between the camera and the LIDAR increases the coloring error with depth.

4.4. *Coloring error analysis setup*

As discussed previously, coloring errors increase with depth if the estimation of the extrinsic (rotation) parameters are inaccurate. To evaluate this for our sensor suite, we created an experimental setup consisting of a white card board placed before a wet black cloth. The setup was placed at various depths and the laser scans of the setup were colored using the corresponding images. The setup was placed beyond 5 m in our tests so that the coloring error due to the error in LIDAR points and the error in camera matrix are completely ruled out. As derived previously, the depth at which these errors disappear are 4.304 m and 3.524 m for the color camera and the thermal camera, respectively. Coloring errors were observed at the edges of the white card board. Any wrong coloring along the edges of the white card board is due to the inaccuracies in the estimation of the extrinsic parameters. Figure 11 shows the coloring of the setup placed at various depths. The coloring errors are noticeable at 5 m and it increases at 7.5 m — suggesting an error in the estimation of the rotation parameters. We measured the distance along the edges in the horizontal direction (x) and the vertical direction (y) that were mis-colored using Cloud Compare [10] (a 3D visualization tool) as shown in Fig. 12. The errors we measured for the color and thermal cameras are reported in Tables 2 and 3, respectively.

4.5. *Comparison*

As mentioned in Sec. 1, cross-calibration methods fall under two categories — error minimization methods and information maximization methods. Error minimization methods reduce the reprojection error when a LIDAR point is projected on to the corresponding image point (or) minimize the alignment error of planes defined in the camera frame and the LIDAR frame. Information maximization methods maximized the correlation between the scene content observed in the different sensing modalities. We compare our method against two algorithms, one from each approach.

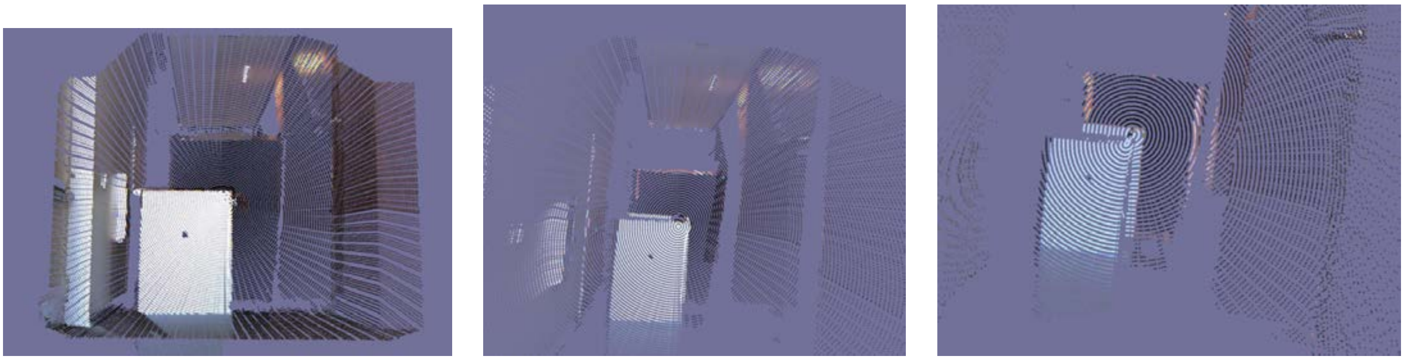
4.5.1. *Error minimization*

We chose the single shot calibration toolbox [9] which minimizes the alignment error of planes defined in the camera frame and the LIDAR frame. As seen in Table 2, the coloring errors are marginally higher than our method, but the difference is not significant. However, the toolbox requires that multiple checkerboard patterns are placed in the scene and creating such targets for a thermal camera is a tedious task. Additionally, extracting all planes (containing the calibration targets) from the LIDAR point cloud reliably is a difficult task as there could be similar planes in the scene. Choosing the wrong set of planes will affect the final calibration results. We manually removed some planes in the scene (using Cloud Compare [10]) to get this toolbox to work and obtain the reported results.

4.5.2. *Information maximization*

We compared the results of our algorithm to the method described in [20] which is based on maximizing mutual information between the intensity distributions of the LIDAR and the RGB images. The source code and a sample dataset are publicly available. Similar to their sample dataset, we collected 10 indoor scans and 10 outdoor scans using our setup. The extrinsic parameters were obtained and errors were compared using our error analysis setup. We observed a 20 cm error at 5 m and the errors increase hence forth.

We investigated the causes for the high errors and we realized that the mutual information method requires the intensity values of the LIDAR to be calibrated. The authors in [20] use the calibration procedure described in [16] to calibrate the intensity values of their Velodyne LIDAR. This calibration method relies on the fact that the same surface patch is hit by different LIDAR beams (out of the 64 beams of Velodyne) and the average intensity of all those LIDAR



(a) At 5 m, coloring errors barely noticeable (b) 7.5 m, coloring errors can be seen (c) 10 m, coloring errors increase

Fig. 11. Coloring error analysis setup. Errors can be seen along the edges of the white card board as the depth increases.

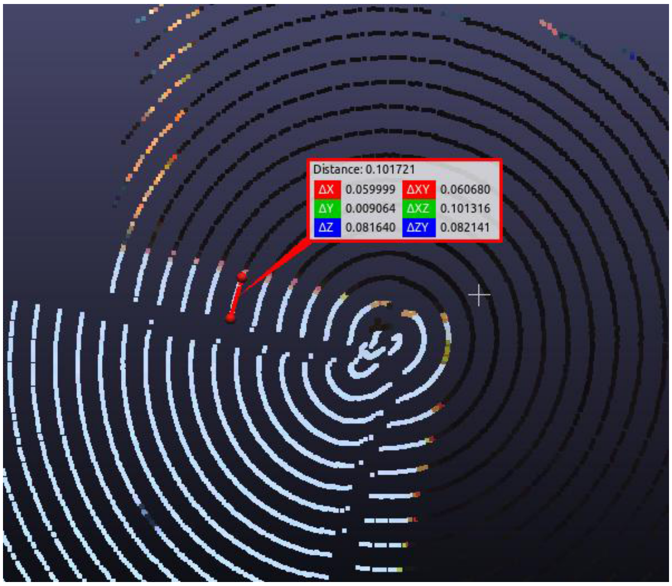


Fig. 12. Measuring the coloring error using Cloud Compare. Here the coloring is off by 10 cm as seen in the call out.

Table 2. Coloring errors at various depths for the RGB camera.

	Errors in cm			
	Our algorithm		Single shot calibration	
Depth (in m)	x	y	x	y
5	0	2	3	6
7.5	1	3	4	7
10	3	6	5	10
12	5	8	5	11
15	6	10	9	12

Table 3. Coloring errors at various depths for the thermal camera.

	Errors in cm	
Depth (in m)	x	y
5	2	4
7.5	3	6
10	3	7
12	4	10
15	5	13

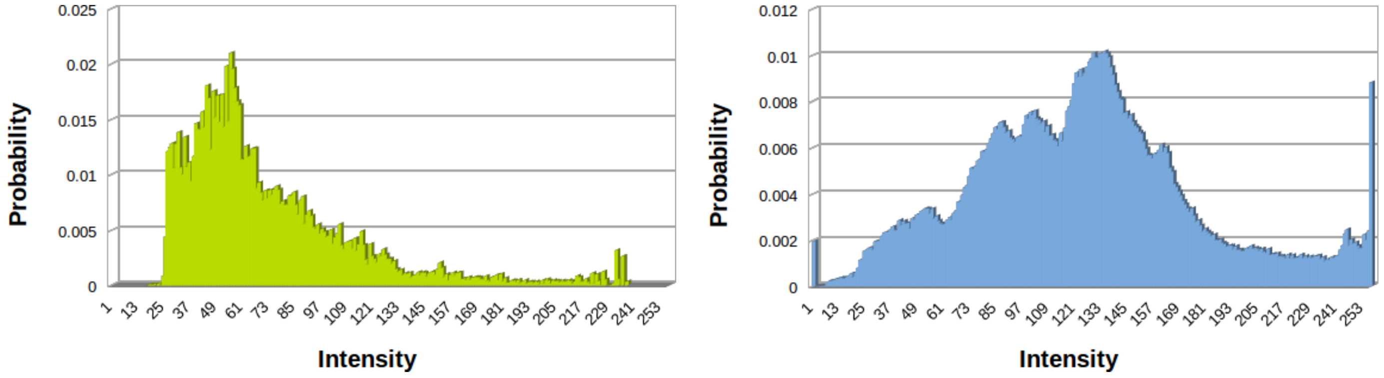
returns represents the true surface albedo of that patch. A 64×256 lookup table is built corresponding to the 256 intensity levels for each beam. The intensity distribution of the LIDAR and camera data in their dataset (henceforth referred as MI dataset) in shown in Fig. 13 and the corresponding distributions of our dataset is shown in Fig. 14. It is hard to evaluate how similar the distributions are visually. However, it is safe to say that (i) the distributions in Fig. 13 have a single peak and the distribution tapers down on either side of the peak and (ii) the peaks are shifted. No such pattern is visible in Fig. 14. To quantitatively compare the similarities of the distributions, we use the information gain metric and we observed that the information gain for distributions in Fig. 13 is 4.82 which was higher (4.31) than the information gain for distributions in Fig. 14.

Also, the intensity calibration method in [16] is specific for multi-beam LIDARs and is yet to be verified for a single beam LIDAR like the Hokuyo. Further, it is unclear on how the inaccuracies in the intensity calibration affects the cross-calibration algorithm.

Maximizing the mutual information between intensity distributions of the LIDAR and the thermal images (as implemented in [20]) also resulted in high errors similar to the RGB images discussed above. We hypothesize that this is because thermal cameras measure emitted radiation in the far-infrared ($7\,\mu$ to $14\,\mu$) whereas LIDAR intensities correspond to the reflectivity in the near infra red ($\sim 1\,\mu$). Figure 15 shows the intensity distributions of a scene in the LIDAR scan and the thermal image. In the LIDAR histogram multiple peaks are observable at 61 and between 200 and 255. However, in the thermal histogram, only two peaks are visible — one at 40 and the other at 255. It is obvious that the distributions look dissimilar and information maximization methods do not work in such cases. (However, in the case of color cameras the measured reflectivity in the visible spectrum ($0.4\,\mu$–$0.7\,\mu$) is closer to the near infra red spectrum ($0.7\,\mu$–$1\,\mu$) and the reflectivity values are assumed to have a strong correlation, making mutual information metric based on intensity distribution useful). Different mutual information metrics such as maximizing edge correlations could yield better results, but it is a case of further research.

5. RGB-Depth-Thermal Mapping

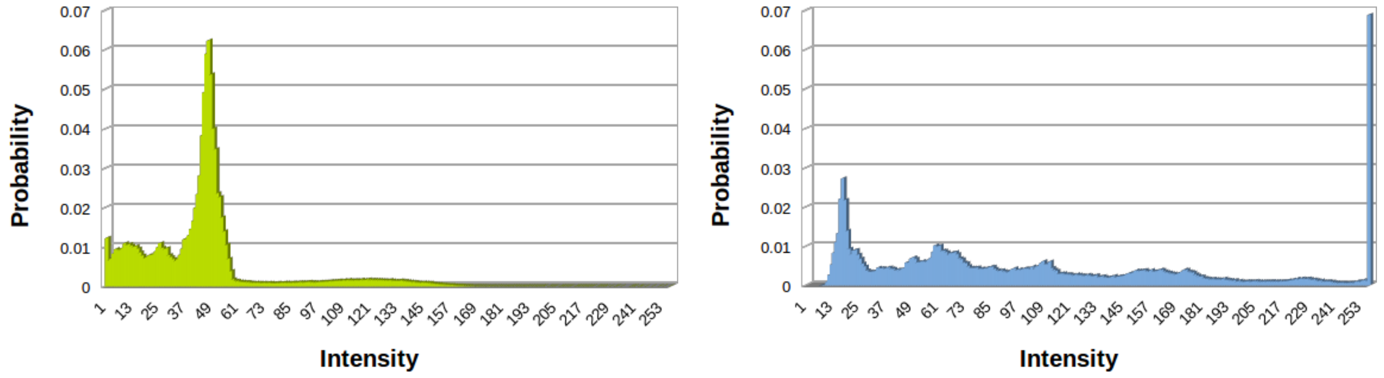
Initially, we computed the extrinsics between the cameras and the LIDAR using the approach described in this paper. Then, we mounted our setup on a cart and collected data for RGBDT mapping of outdoor environments. For each point cloud obtained from the LIDAR, we apply the computed extrinsic parameters and project them on to the RGB and thermal images. We find the corresponding RGB and thermal texture for each point in the point cloud. We then pass this *textured* point cloud to our 3D mapping pipeline (which is implemented using the Iterative Closest Point [4] algorithm) and produce RGBDT maps. One point cloud is



(a) Intensity histogram of the LIDAR data

(b) Intensity histogram of the color images

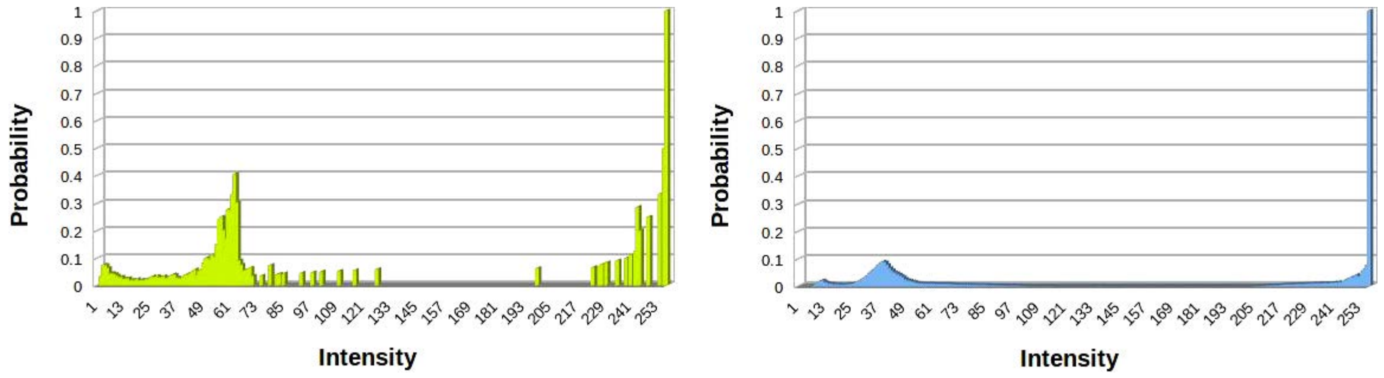
Fig. 13. Intensity distributions of the MI dataset — the LIDAR intensities are calibrated.



(a) Intensity histogram of the LIDAR data

(b) Intensity histogram of the color images

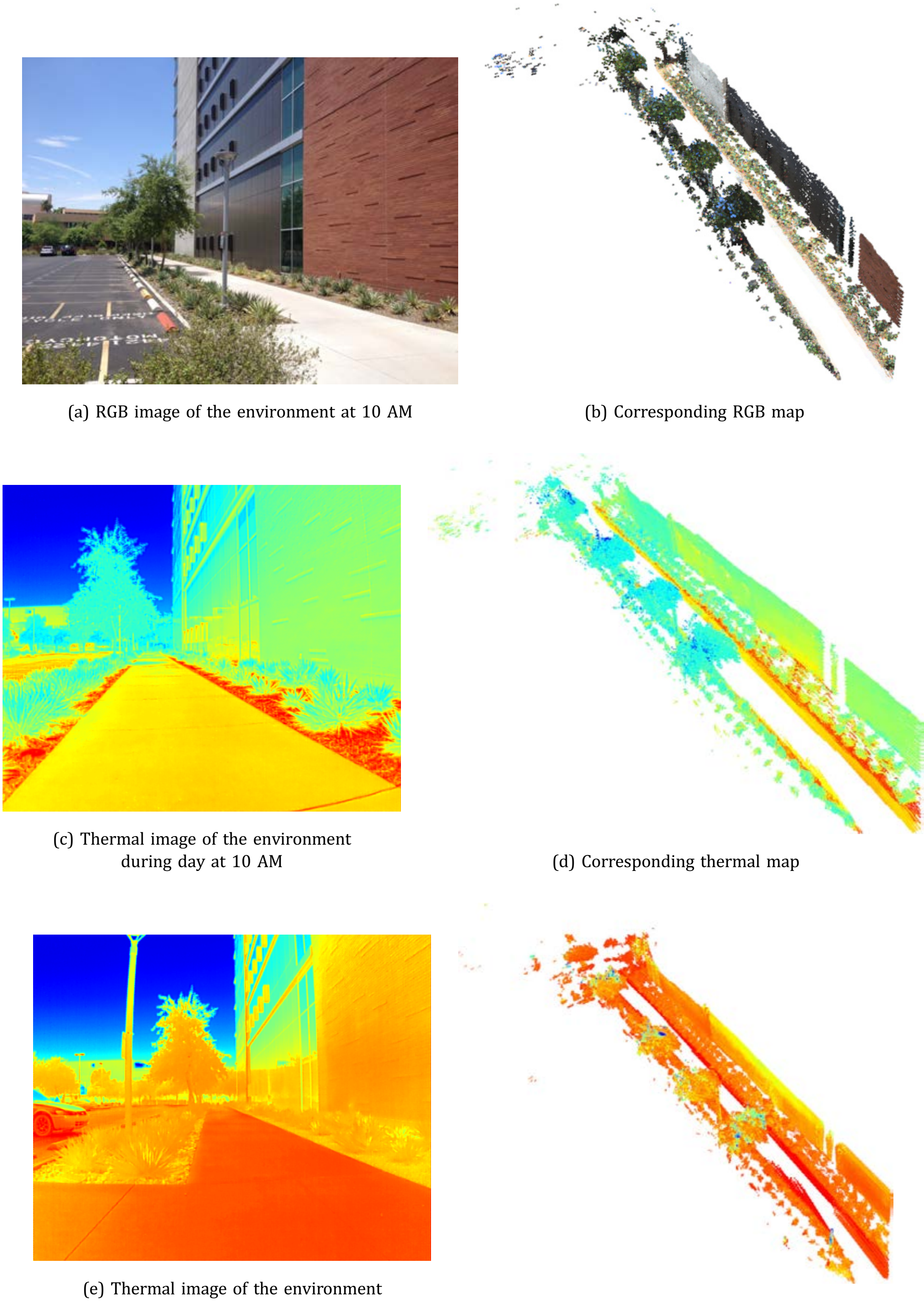
Fig. 14. Intensity distributions of our dataset — the LIDAR intensities are NOT calibrated.



(a) Intensity histogram of the LIDAR data

(b) Intensity histogram of the thermal images

Fig. 15. Intensity distribution.



(a) RGB image of the environment at 10 AM

(b) Corresponding RGB map

(c) Thermal image of the environment during day at 10 AM

(d) Corresponding thermal map

(e) Thermal image of the environment during night at 10 PM

(f) Corresponding thermal map

Fig. 16. RGBDT mapping.

generated for every 600 ms (time taken for 1 tilt of the LIDAR). The frame rates for the RGB and thermal cameras are 5 fps and 25 fps, respectively. Figure 16 shows the RGB and thermal maps (containing ~1.2 million points) of the same environment during various times of the day. It is noticeable in the thermal maps that the pavement can be differentiated from the building and the trees during the day and the night. So thermal maps can potentially be used to segment roads for urban night driving. We are optimistic that there are interesting applications to look at as we gather more data of urban environments at night.

6. Conclusions and Future Work

We presented an approach to cross-calibrate an RGB and a thermal camera with a LIDAR. Initially, we described a calibration target for the intrinsic calibration of the thermal camera. Next, we described our setup for the extrinsic calibration of the thermal and the RGB cameras with a LIDAR. We presented our calibration algorithm that uses the distance transform of the edges in the image as an objective function, which when minimized gives the extrinsics between the cameras and the LIDAR. Later, we analyzed the coloring error due to errors in the 3D points, the camera matrix K, and the extrinsic parameters. We concluded that the coloring errors due to errors in rotation grow with the depth of the 3D point and the errors due to other factors disappear with depth. Finally, we presented our results on the RGBDT mapping of outdoor environments using our setup. In the future, we are interested in building maps at a larger scale by mounting the sensor suite on a car and investigate the applications of RGBDT mapping for autonomous driving.

References

[1] Wikipedia: Calibration, https://en.wikipedia.org/wiki/Calibration.
[2] S. Agarwal, K. Mierle *et al.*, Ceres solver (2012).
[3] H. Alismail, L. D. Baker and B. Browning, Automatic calibration of a range sensor and camera system, *2012 Second Int. Conf. on 3D Imaging, Modeling, Processing, Visualization & Transmission* (IEEE, 2012), pp. 286–292.
[4] P. J. Besl and N. D. McKay, Method for registration of 3-d shapes, in *Robotics-DL Tentative* (International Society for Optics and Photonics, 1992), pp. 586–606.
[5] D. Borrmann, H. Afzal, J. Elseberg and A. Nüchter, Mutual calibration for 3d thermal mapping, in *Proc. 10th Int. IFAC Symp. Robot Control (SYROCO12)*, Vol. 10 (2012).
[6] A. Carballo, A. Ohya *et al.*, People detection using range and intensity data from multi-layered laser range finders, *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ Int. Conf.* (IEEE, 2010), pp. 5849–5854.
[7] P. S. Chavez Jr., Radiometric calibration of landsat thematic mapper multispectral images, *Photogramm. Eng. Remote Sens.* **55**(9) (1989) 1285–1294.
[8] R. Gade and T. B. Moeslund, Thermal cameras and applications: A survey, *Mach. Vis. Appl.* **25**(1) (2014) 245–262.
[9] A. Geiger, F. Moosmann, O. Car and B. Schuster, Automatic camera and range sensor calibration using a single shot, *2012 IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE, 2012), pp. 3936–3943.
[10] D. Girardeau-Montaut, Cloud compare3d point cloud and mesh processing software, Open Source Project (2015).
[11] E. Guizzo, How googles self-driving car works, *IEEE Spectrum*, 18 October 2011.
[12] A. Kassir, Radocc camera calibration toolbox, Available at http://www-personal.acfr.usyd.edu.au/akas9185/-AutoCalib/AutoCamDoc/index.html.
[13] A. Kassir and T. Peynot, Reliable automatic camera-laser calibration, in *Proc. 2010 Australasian Conf. Robotics & Automation* (ARAA, 2010).
[14] J. Kelly and G. S. Sukhatme, Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration, *Int. J. Robot. Res.* **30**(1) (2011) 56–79.
[15] J. Levinson and S. Thrun, Automatic online calibration of cameras and lasers, *Robotics: Science and Systems* (2013).
[16] J. Levinson and S. Thrun, Unsupervised calibration for multi-beam lasers, in *Experimental Robotics* (Springer, 2014), pp. 179–193.
[17] J. T. Lussier and S. Thrun, Automatic calibration of rgbd and thermal cameras, *2014 IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (IEEE, 2014), pp. 451–458.
[18] A. Napier, P. Corke and P. Newman, Cross-calibration of push-broom 2d lidars and cameras in natural scenes, *Robotics and Automation (ICRA), 2013 IEEE Int. Conf.* (IEEE, 2013), pp. 3679–3684.
[19] J.-N. Ouellet and P. Hébert, Precise ellipse estimation without contour point extraction, *Mach. Vis. Appl.* **21**(1) (2009) 59–67.
[20] G. Pandey, J. R. McBride, S. Savarese and R. M. Eustice, Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information, in *Proc. AAAI National Conf. Artifical Intelligence* (2012).
[21] R. Pless and Q. Zhang, Extrinsic calibration of a camera and laser range finder, *IEEE Int. Conf. Intelligent Robots and Systems (IROS)*, Vol. 13 (2004), pp. 150–154.
[22] A. Rankin, A. Huertas, L. Matthies, M. Bajracharya, C. Assad, S. Brennan, P. Bellutta and G. W. Sherwin, Unmanned ground vehicle perception using thermal infrared cameras, in *SPIE Defense, Security, and Sensing* (International Society for Optics and Photonics, 2011), pp. 804503–804503.
[23] R. B. Rusu and S. Cousins, 3d is here: Point cloud library (pcl), *2011 IEEE Int. Conf. Robotics and Automation (ICRA)* (IEEE, 2011), pp. 1–4.
[24] D. Scaramuzza, A. Harati and R. Siegwart, Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes, *IROS 2007. IEEE/RSJ Int. Conf. Intelligent Robots and Systems, 2007* (IEEE, 2007), pp. 4164–4169.
[25] J. A. Seibert, J. M. Boone and K. K. Lindfors, Flat-field correction technique for digital detectors, *Medical Imaging'98* (International Society for Optics and Photonics, 1998), pp. 348–354.
[26] R. Siegwart, I. R. Nourbakhsh and D. Scaramuzza, Autonomous mobile robots, Massachusetts Institute of Technology (2004).
[27] Z. Taylor and J. Nieto, Automatic calibration of lidar and camera images using normalized mutual information, *2013 IEEE Int. Conf. Robotics and Automation (ICRA)* (2013).
[28] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann *et al.*, Stanley: The robot that won the darpa grand challenge, *J. Field Robot.* **23**(9) (2006) 661–692.
[29] R. Unnikrishnan and M. Hebert, Fast extrinsic calibration of a laser rangefinder to a camera (2005).
[30] M. Veĺas, M. Španěl, Z. Materna and A. Herout, Calibration of rgb camera with velodyne lidar, in *WSCG 2014 Communication Papers Proc.*, Vol. 2014 (Union Agency, 2014), pp. 135–144.

[31] S. Vidas, P. Moghadam and S. Sridharan, Real-time mobile 3d temperature mapping, *IEEE Sensors J.* **15**(2) (2015) 1145–1152.
[32] S. Vidas and S. Sridharan, Hand-held monocular slam in thermal-infrared, *2012 12th Int. Conf. Control Automation Robotics & Vision (ICARCV)* (IEEE, 2012), pp. 859–864.
[33] P. Zhang, J. Gu, E. E. Milios and P. Huynh, Navigation with imu/gps/digital compass with unscented kalman filter, *IEEE Int. Conf. Mechatronics and Automation*, Vol. 3 (IEEE, 2005), pp. 1497–1502.
[34] Z. Zhang, A flexible new technique for camera calibration, *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(11) (2000) 1330–1334.

Aravindhan K Krishnan is a Roboticist primarily focussing on 3D Perception, Sensor Calibration, and Mapping algorithms. He received his Ph.D. in Exploration Systems Design from Arizona State University in 2016 and his Masters in Computer Science from International Institute of Information Technology (IIIT), Hyderabad in 2011. He was a Program Committee Member for IEEE International Symposium on Safety, Security, and Rescue Robotics in 2014 and 2015.

Srikanth Saripalli is an Associate Professor in Mechanical Engineering at Texas A&M University. He received Ph.D. and M.S. degrees from the University of Southern California in 2007 and 2002, respectively, and the B.E. (Hons) degree from Birla Institute of Technology and Sciences, Pilani, India, in 1999. He is a Roboticist with research interests in unmanned systems in general and aerial vehicles in particular. His research focuses on robotic exploration: particularly in air and ground vehicles and necessary foundations in perception, planning, and control for this domain. He was the Program Chair for International Conference on Unmanned Aerial Systems 2015, a Senior Member of IEEE and a Member of AIAA.