A Compact Parametric Solution to Depth Sensor Calibration

Andrew Spek
andrew.spek@monash.edu
Tom Drummond
tom.drummond@monash.edu

ARCV Centre for Robotic Vision
Department of Electrical Engineering
Monash University

Abstract

We present a method for calibration of low-cost depth sensors such as the Microsoft Kinect. We show this method is effective at correcting the structured sensor error using a compact parametric solution, that uses only a small fraction of the number of parameters used in many existing approaches. Additionally we provide this calibration as an open-source implementation, with limited required external dependencies. We demonstrate our approach can optimise directly for a geometric depth and radial distortion calibration function in minutes on modern hardware.

Figure 1: **Top**: Several frames from the dataset used to generate the model below. **Bottom-Left**: Model generated using uncalibrated data. **Bottom-Right**: Model generated using our calibrated depth data. The black lines indicate the approximate true position of the room corner and walls, with the scale of the room shown for each model. We include an enlarged section of the corner of each model to demonstrate significant reduction in error using our calibration method.

© 2015. The copyright of this document resides with its authors. It may be distributed unchanged freely in print or electronic forms.
1 Introduction

The introduction of low-cost depth sensors (such as the Microsoft Kinect) has greatly simplified the collection of real-time depth scans. These sensors provide a registered stream of colour and depth images at 30Hz, allowing fast and accurate modelling of complex indoor scenes [7][11]. While some previous methods calibrate for time-of-flight (ToF) sensors [1][10], which have interesting but different noise characteristics, the focus of this paper is on the variety of depth sensors known as *light coding* depth sensors. This type of depth sensor projects a known pseudo random infra-red grid pattern onto the environment, then detects it using an infra-red camera. Using the known displacement of the camera and projector, and finding corresponding patches in the projected pattern allows the device to estimate a dense depth image. A more detailed explanation of this process can be found in [14]. Although these sensors have proven useful for modelling and tracking they have been shown to contain significant calibration errors [9][14] (Figure 1).

Previous approaches have been used to correct the error present in these types of sensors [4][5][12][17] with varying degrees of success. The approaches seem to fall into two main categories, planar and SLAM based calibration. The former approach uses a known planar surface and attempts to correct the reported depth to align more accurately to a fitted plane. While the SLAM/model based approaches attempt to accurately model a complex scene and correct the depth such that the model produced will minimise some global error function across all depth images.

In this paper we present a compact polynomial (tens of parameters) method of depth sensor calibration using an unsupervised SLAM based approach. We provide an open-source implementation that uses few external libraries and provides a simple and reliable method of depth sensor calibration using a relatively small number of keyframes (10-15). The advantages of the presented calibration method are:

- It is highly compact, requiring a greatly reduced number of parameters than existing methods (see Section 2.2).
- It requires minimal supervision from the user (see Section 3.2).
- It provides noticeably improved depth estimates for a range of depth values (see Section 4.1).
- It outperforms previous approaches qualitatively and quantitatively (see Section 4.2)

1.1 Related Work

One of the most successful unsupervised calibration method for the Kinect was [16]. Teichman et al. use an approach which relies on building a globally consistent SLAM model of a scene prior to calibration. They attempt to ensure the model quality and pose accuracy by rejecting depth values greater than 2 meters when optimising the model. This is because they require very accurate initial pose estimates. Using the pre-built SLAM model they then attempt to minimise the error between the projected depth and the model using a maximum likelihood across all corresponding depths patches. The correction is a linear function applied to depth calculated for all 8x6 image patches for discrete depths and interpolated between adjacent patches. The process of modelling-calibration process can be repeated, but reportedly this is largely not required for a good calibration.

Zhou and Koltun took a similar approach to calibration in [17], but instead of requiring the user to manually cycle the modelling and calibration steps, it automatically recomputes a
more accurate SLAM model by using the corrected depth images in their model generation stage after each iteration. Again they use a set of patch centred linear scaling factors for discrete depths, however they use tri-linear interpolation between patches of a particular depth and between specific depths. Both [16] and [17] use a patch based approach to save computational time and memory usage for the resulting calibration model, and avoid over-fitting the data as the devices used produce high frequency noise. Both methods are reportedly overnight computational processes on modern hardware. However, both methods can suffer from over fitting to the noise due to the extremely high model capacity using thousands of parameters.

An early example of planar calibration is shown in [4]. Herrera et al. present a method for improving the depth to colour registration provided by the manufacturer of the Kinect and then attempt to improve the depth estimates. They use a joint checker-board approach to register depth and colour images, then using the checker-board on a planar surface, they optimise for a solution that corrects depth to the plane. In [12] Raposo et al. provide an extension to [4] and improve the performance of the calibration in terms of the number of required frames and speed of calibration. Both methods show improved accuracy but both require the additional step of calibrating the colour to the depth camera. In [8] Jin et al. simplify this supervised calibration method by having the user identify cuboids in the scene and iteratively optimise an error function over the planar regions in the selected cuboids. One of the drawbacks of these methods is an emphasis on supervision, and external ground truth estimates which take additional time and/or require accurate measurements to be made by the user in order to function.

A similar planar approach used in [5] where Di Cicco et al. present a calibration approach that uses machine learning. An Artificial Neural Network (ANN) is used to calibrate the sensor, removing the need for any specific knowledge or sensor modelling. They also describe a method for collecting large planar surfaces for calibration that eliminates non-planar data from collected datasets. This method greatly underestimates the difficulty of capturing a planar surface from more than 6 meters away which limits the effective range of the calibration.

2 Fundamentals

2.1 ICP

Iterative Closest Point (ICP) is a state of the art technique for aligning multiple overlapping point clouds first developed in [2] and [3]. The principle of the algorithm is to iteratively update the alignment of multiple point clouds in order to minimise the distance between corresponding points in all the point clouds. There are several variants of ICP that are defined by their distance metric used for correspondence estimation and point cloud alignment. A popular variant of ICP is known as Point-to-Plane (Pt-Pl) ICP [13]. The error for the kth point is defined by the residual function \( \hat{r}_k \) in Equation 1.

\[
\hat{r}_k = \hat{n}_k \cdot (p_k - E(\gamma)q_k)
\]

where \( E \) is the transformation that aligns point cloud \( Q \) to \( P \), and \( p_k \in P, q_k \in Q \) are corresponding points in those two point clouds. \( \gamma = \{ \theta_x, \theta_y, \theta_z, t_x, t_y, t_z \} \) is the set of motion parameters that defines the transformation matrix \( E \), which can be represented as \( E(\gamma) = e^{\sum_{i=1}^{6}(G_i\gamma)} \) as shown in [6]. This can be used to define a cost function in terms of the motion parameters...
\[ \bar{\gamma} = \sum_{k=0}^{N} (\hat{r}(k))^2 \]  

(2)

and solved for an update to the motion parameters using a standard Gauss-Newton approach. The general form that this update takes is

\[ \Delta \bar{\gamma} = -(J^T J)^{-1} J^T \hat{r} \]  

(3)

where \( J \) is the derivative of the error function \( (\epsilon) \) w.r.t. the motion parameters \( \bar{\gamma} \). For more information about common approaches to solving the ICP algorithm please refer to \([13]\).

2.2 Calibration Model

In order to calibrate low-cost depth sensors we propose a corrective polynomial model, which can be used to approximate the geometric and lens distortion present in projective depth sensors. We express the calibration as both a radial distortion factor \( r(u', v') \), and a multiplicative depth correction factor \( c = C(uu', vv', d) \). The corrected points \( \bar{p} \) are computed from the normalised camera coordinates \( u, v \) and the sensed depth \( d \) as:

\[ \bar{p}(k) = \text{proj}(u', v', c) = [x \ y \ z \ 1]^T = [cdru' \ cdvv' \ cd \ 1]^T \]  

(4)

where \( \text{proj}(u', v', c) \) is the projection of the radially corrected parameters for used in the geometric correction function and \( d = d_i(u', v') \) is the depth at \( u', v' \) in depth scan \( i \). The normalised camera coordinates \( u', v' \) are computed using:

\[ u' = \frac{X - u_0}{f_x}, \quad v' = \frac{Y - v_0}{f_y} \]

where \( u_0 \) and \( v_0 \) are the coordinates of the principal point, \( f_x \) and \( f_y \) are the focal lengths in the horizontal and vertical dimensions respectively, and \( X \) and \( Y \) are pixel coordinates.

The radial distortion correction is given by:

\[ r(u', v') = 1 + \beta_0 (u'^2 + v'^2) + \beta_1 (u'^2 + v'^2)^2 \]  

(5)

as shown this is defined only for the first two radial distortion parameters. The depth correction factor \( c \) is calculated as:

\[ c = C(u, v, d) = 1 + \alpha_0 u + \alpha_1 v + \alpha_2 uv + d(\alpha_n u + \alpha_{n+1} v + \alpha_{n+2} uv + \ldots) \]  

(6)

which enumerates the initial terms of the radial \((\alpha_0, \alpha_1, \ldots \alpha)\) and depth \((\alpha_n, \alpha_{n+1}, \ldots)\) geometric correction coefficients, and \( u, v \) are the radially corrected normalised camera coordinates. We continue this pattern of enumerating all possible polynomial combinations \((u, v, uv, u^2, v^2, \ldots, du, dv, duv, \ldots)\) with a separate coefficient \( \alpha_i \) for each up to the desired order. The number of terms increase approximately quadratically w.r.t. the order of the polynomial model. In this paper we present up seventh order which requires 70 coefficients.
3 Method

3.1 Error Minimisation Function

We define a residual function \( \hat{r}(i, j, k) \) for each depth image pair \((i, j)\) and every correspondence between frame pairs \(k\).

\[
\hat{r}(i, j, k) = R_i \hat{n}(k) \cdot (E_i \hat{p}_i(k) - E_j \hat{p}_j(k))
\]  

where \(i \neq j\) and \(E_i = E(\hat{\gamma}_i)\) is the transformation described by the motion parameters \(\hat{\gamma}_i\) that would take the projection \(\hat{p}_i\) (as described in Equation 4) of the \(i\)th depth image \((d_i)\) to a common world coordinate frame. In this form Equation 7 gives the residual in terms of absolute poses instead of a single relative pose as in Equation 1. Points \(p_i\) and \(p_j\) are calibrated corresponding points from the depth images \(d_i\) and \(d_j\) respectively, which have been corrected according to Equation 4. We estimate these correspondences are estimated based on the current projection of any point given the current calibration estimate, into each frame and taking the closest point in a neighbourhood search. We use this to define a cost function for depth image pairs in terms of the poses (\(\bar{\Gamma}\)), geometric(\(\bar{\alpha}\)) and radial distortion(\(\bar{\beta}\)) parameters:

\[
\epsilon(\bar{\Gamma}, \bar{\alpha}, \bar{\beta}) = \sum_{i=1}^{\hat{N}} \sum_{j=1}^{\hat{N}} \sum_{k=1}^{N} (\hat{r}(i, j, k))^2
\]  

where \(i \neq j\), \(\hat{N}\) is the number of depth images used in the joint optimisation, \(N\) is the number of correspondences in the depth image pair and \(\bar{\Gamma} = \{\tilde{\gamma}_1, \tilde{\gamma}_2, ..., \tilde{\gamma}_N\}\) is the set of pose parameters for each image. We use a standard Gauss-Newton approach to minimise error across this function. Using the product-rule we compute the derivative w.r.t. to all motion and calibration parameters across all images shown in Equation 9 below.

\[
\frac{d \hat{r}}{d \bar{c}} = \frac{\partial E_i}{\partial \bar{c}} \hat{n}_i \cdot (E_i \hat{p}_i - E_j \hat{p}_j) + E_i \hat{n}_i \cdot \left( \frac{\partial E_i}{\partial \bar{c}} \hat{p}_i + E_i \frac{\partial \hat{p}_i}{\partial \bar{c}} - \frac{\partial E_j}{\partial \bar{c}} \hat{p}_j + E_j \frac{\partial \hat{p}_j}{\partial \bar{c}} \right)
\]  

where \(\bar{c} = \{\bar{\Gamma}, \bar{\alpha}, \bar{\beta}\}\) represents the separate motion parameters for each frame and the joint calibration parameters for simplicity. To simplify understanding Equation 9 we can examine the calculation for a single image pair and consider the separate Jacobian calculations. Each range image pair has two sets of motion parameters (\(\tilde{\gamma}_i\) and \(\tilde{\gamma}_j\)) to solve for the absolute pose of those point clouds, and a joint set of calibration parameters (\(\bar{\alpha}, \bar{\beta}\)) that is tied to a single depth sensor.

\[
\frac{d \hat{r}}{d \tilde{\gamma}_i} = J_{\tilde{\gamma}_i} = \frac{\partial E_i}{\partial \tilde{\gamma}_i} \hat{n}_i \cdot (E_i \hat{p}_i - E_j \hat{p}_j) + E_i \hat{n}_i \cdot \left( \frac{\partial E_i}{\partial \tilde{\gamma}_i} \hat{p}_i \right)
\]

\[
\frac{d \hat{r}}{d \tilde{\gamma}_j} = J_{\tilde{\gamma}_j} = -E_i \hat{n}_i \cdot \left( \frac{\partial E_j}{\partial \tilde{\gamma}_j} \hat{p}_j \right)
\]

\[
\frac{d \hat{r}}{d (\bar{\alpha}, \bar{\beta})} = J_{\bar{\alpha} \bar{\beta}} = E_i \hat{n}_i \cdot \left( E_i \frac{\partial \hat{p}_i}{\partial (\bar{\alpha} \bar{\beta})} - E_j \frac{\partial \hat{p}_j}{\partial (\bar{\alpha} \bar{\beta})} \right)
\]  

where \(J_{\bar{\alpha} \bar{\beta}} = \{J_{\bar{\alpha}}, J_{\bar{\beta}}\}\) If we represent the absolute pose transformations as matrix exponentials we can compute \(J_{\hat{\gamma}}\) as the following:

\[
J_{\hat{\gamma}} = [ \hat{r}^T \hat{r}^T \hat{p}_j^T ]^T
\]
where \( \hat{n}' \) represents the surface normal rotated to the \( i \)th frame by

\[
R_i\hat{n}_i(k) = \hat{n}' = \begin{bmatrix} \hat{n}'_x & \hat{n}'_y & \hat{n}'_z \end{bmatrix}^T
\]

\( R_i \) represents just the rotation of matrix \( E_i \) and \( \bar{p}'_j \) is the transformed homogeneous coordinate in the global frame. Somewhat conveniently it can be shown that \( J_{\bar{\gamma}_j} = -J_{\bar{\gamma}_i} \), which simplifies computation.

We can now compute the calibration Jacobians using the chain rule, first differentiating points \( \bar{p}_i \) w.r.t. the calibration parameters \( \bar{\alpha} \):

\[
\frac{\partial \bar{p}_i}{\partial \bar{\alpha}} = \left[ x_i \ y_i \ z_i \ 0 \right]^T \cdot \frac{\partial c}{\partial \bar{\alpha}}
\]

where the matrix \( \frac{\partial c}{\partial \bar{\alpha}} \) will vary in length depending on the size of the parameterisation chosen (see Section 2.2). \( \frac{\partial \bar{p}_j}{\partial \bar{\alpha}} \) is computed in the same way substituting \( i \) for \( j \) in Equation 13. The resulting Jacobian with respect to the calibration parameters \( \bar{\alpha} \) is given by Equation 14.

\[
J_{\bar{\alpha}} = \hat{n}' \cdot \left( E_i \left[ \begin{array}{cccc} x_i & y_i & z_i & 0 \end{array} \right]^T \frac{\partial c}{\partial \bar{\alpha}} - E_j \left[ \begin{array}{cccc} x_j & y_j & z_j & 0 \end{array} \right]^T \frac{\partial c}{\partial \bar{\alpha}} \right)
\]

Finally the derivative w.r.t. the radial distortion parameters (\( \bar{\beta} \)) is given by:

\[
J_{\bar{\beta}} = \hat{n}' \cdot \left( E_i \left[ \begin{array}{cccc} c d_i u_i & c d_i v_i & 0 & 0 \end{array} \right]^T \frac{\partial r}{\partial \bar{\beta}} - E_j \left[ \begin{array}{cccc} c d_j u_j & c d_j v_j & 0 & 0 \end{array} \right]^T \frac{\partial r}{\partial \bar{\beta}} \right)
\]

where

\[
\frac{\partial r}{\partial \bar{\beta}} = \left[ u_i^2 + v_i^2 \ (u_i^2 + v_i^2)^2 \right]
\]

### 3.2 System Overview

![Diagram](Image)

Figure 2: The underlying structure of our calibration system, demonstrating the calculation of independent Jacobians across threads which are combined into a Joint Jacobian that contains all jointly available information.

The diagram in Figure 2 shows a broad overview of the underlying architecture of our calibration system. Our system requires a global pose estimate for each frame as an initial...
input to the system. We provide a method for capturing frames in the repository along with
detailed instructions on how to use our system. A good scene for calibration should have
large visible planes in multiple overlapping frames as these are the most stable source of
depth correction. If the scene contains little or no geometric features ICP will fail to produce
good alignments are calibration will fail. Choosing frames for is up to the user, and as such
this is an important point for the user to consider when calibrating. Another subtle point
that may affect the stability of our system is the quality of the normal estimates which are
effected by sensor noise. To improve stability of our normal estimates we use the method

The calibration process can be broken down into the following steps:

- The captured frames are combined into frame pairs and they are divided evenly amongst
  available threads for a Jacobian ($J$) calculation (Section 3.1). This makes the compu-
tational complexity of our approach is proportional to the number of frame pairs, or
  the square of the number of frames $O(N^2)$
- The resulting Hessian ($J^T J$) and residual matrices from each thread are recombined
  into a Joint Hessian and a Cholesky decomposition is used to calculate an update
  (using standard Gauss-Newton approach Equation 3)
- The computed updates are applied to a global set of camera parameters ($\hat{\beta}$) and geo-
  metric calibration parameters ($\hat{\alpha}$) and each current pose estimate ($\hat{\gamma}$)
- The coordinates, corrected coordinates and surface normals are recomputed using the
  updated model.
- This compute-update cycle is repeated until convergence as shown in Figure 2.

4 Results

4.1 Planar Test

Due to the difficulty of collecting ground truth datasets we use the method of quantitative
evaluation proposed in [16] for our calibration method. We use each device to record a test
dataset of a flat section of a wall and measure how well the resulting corrected depth images
fit to a plane. This provides a measure of how well the uncalibrated data can be corrected at
a range of depths. The results of this experiment for two devices is shown in Figure 3 and
demonstrates the significant improvement our approach provides over the uncalibrated depth
produced by the sensor (particularly for the Kinect V1). We demonstrate a consistent near
50% improvement for our highest order calibration at each depth range.

4.1.1 Choosing the Degree of the Polynomial

In order to choose the degree of polynomial used by our method we calibrated each device
separately with each degree polynomial up to 7th order and examined the improvement on
the planar test described in Section 4.1 above. The results of this are shown in Figure 3
and demonstrate that increasing the degree of the polynomial above 6th order provides little
improvement. Ideally we would like to use the smallest degree polynomial required to give
a good calibration as this will reduce computational time, and that appears to be 6th or 7th.
Additionally using higher order polynomials can lead to unstable calibrations for areas with
less information and larger gradients such as corners.
Figure 3: The RMS error from fitting a planar surface to corrected data, for Kinect V1 (Left) and an Asus Xtion (Right). This demonstrates the gain in accuracy for higher order polynomials reduces as you increase the degree of the polynomial for our calibration method.

Figure 4: Left: Planar error for the uncalibrated data. Middle: Correction function at the same depth. Right: Calibrated depth data. A key is shown to indicate the error at each point in mm.

### 4.1.2 Visualising Calibration Function

In order to examine how well our calibration function fits the data, we inspect it at various depth values and compare against the uncalibrated error at that same approximate depth from the planar test set. We show the result of this for a depth of 2 meters in Figure 4. The correction function fits the data very well, and the calibrated image shows a large reduction in the systematic sensor error, showing this order of the model is appropriate to significantly reduce the structural error. However the data still retains the original high-frequency noise, which varies greatly over time and can be reduced using tradition filtering approaches.

### 4.2 Comparative Performance

We compare the performance of our system against [10] and [16] using the same test described in Section 4.1. We also show a number of qualitative comparisons on real-world data, to demonstrate the comparable performance of our approach. However our approach which is also CPU side only takes minutes, while CLAMS takes many hours. The approach in [10] requires a more specific set of images be collected, including known checker-boards in each image, and each part of the frame must be covered to ensure a successful calibration. Once this dataset is collected this approach does take minutes, given the relatively low amount of data used in the computation. The results are shown in Figure 5, and demonstrate our approaches competitive performance against CLAMS([16]) and our improvement upon
Figure 5: Quantitative and qualitative comparisons of our calibration result against the approach from [16] (CLAMS). **Top-Left** Results for the same planar test, comparing several approaches. **Bottom-Left** Compares a single calibrated frame using our approach and CLAMS for a planar surface (top-down). **Right** Compares a single calibrated frame using our approach and CLAMS for the corner of a room (top-down). Black lines are used to indicate the approximate location of true straight edges in the scene.

MIP([10]). For this test we calibrated MIP([10]) using 80 images of checker boards captured at the range of depths and image locations as described in the calibration manual. For the comparison to CLAMS([16]) we calibrated the kinect using 5 minutes of footage, then retrained with a further 5 minutes of footage (10 minutes in total), this final calibration was used in the comparison.

4.3 Calibrated Dataset

To further demonstrate the requirement for calibration, we evaluate the accuracy of estimating a known trajectory using set of the uncalibrated and calibrated depth images. We collected a straight line on-rails dataset using the same Kinect used for calibration (the set up is shown in Figure 6). We optimise the global poses jointly for this dataset before and after calibration using a joint point-to-plane ICP approach. We show the RMS error in pose rotation and translation in Figure 6. The uncalibrated sensor data causes significant pose drift and is visibly reduced after correction using our approach, approximately a 50% reduction in error.

4.4 Model Generation

Finally we present a qualitative improvement in scene modelling in Figure 1. We observe a clear improvement over the original model, after our calibration system has been used to correct the depth sequence. This is most visible in the curved wall, which demonstrates the affect of structured sensor noise on model drift and modelling accuracy.
Figure 6: **Left-Top:** A frame from the dataset. **Left-Bottom:** The on-rails Kinect set up, with an arrow showing the direction of motion. **Middle:** Shows the uncalibrated and calibrated trajectories and approximate ground truth. **Right:** Rotational error in pose RMS(rad)

5 Conclusion and Further Work

In this paper we present a novel solution for low-cost depth sensor calibration, that requires minimal supervision. A suitable image sequence of roughly twenty frames can be used to calibrate a sensor in approximately 10 minutes on a relatively modest Intel Core i7-2600 at 3.4GHz. A future implementation might use a compact DCT formulation to improve the calibration at the corners of images, at this stage these values are simply removed. One key motivation for the creation of this work was difficulty in getting the previous referenced calibration methods to work or source code was unavailable, many also require large external libraries or have insufficient documentation to operate.

6 Acknowledgements

This work was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016).

References


