Why Would I Want a Gyroscope on my RGB-D Sensor?

Hannes Ovrén, Per-Erik Forssén, and David Törnqvist
Department of Electrical Engineering, Linköping University, Sweden
hannes.ovren@liu.se

Abstract

Many RGB-D sensors, e.g. the Microsoft Kinect, use rolling shutter cameras. Such cameras produce geometrically distorted images when the sensor is moving. To mitigate these rolling shutter distortions we propose a method that uses an attached gyroscope to rectify the depth scans. We also present a simple scheme to calibrate the relative pose and time synchronization between the gyro and a rolling shutter RGB-D sensor.

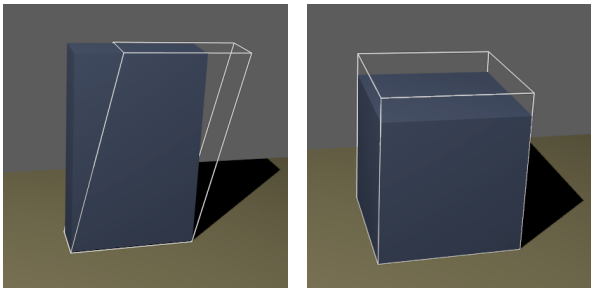
We examine the effectiveness of our rectification scheme by coupling it with the the Kinect Fusion algorithm. By comparing Kinect Fusion models obtained from raw sensor scans and from rectified scans, we demonstrate improvement for three classes of sensor motion: panning motions causes slant distortions, and tilt motions cause vertically elongated or compressed objects. For wobble we also observe a loss of detail, compared to the reconstruction using rectified depth scans.

As our method relies on gyroscope readings, the amount of computations required is negligible compared to the cost of running Kinect Fusion.

1. Introduction

RGB-D sensors, such as the Microsoft Kinect have recently become popular as a means for dense real-time 3D mapping. Dense RGB-D mapping was introduced in the Kinect Fusion algorithm [9], which is aimed at augmented reality. Recently the Kinect Fusion algorithm has also been adapted to *simultaneous localisation and mapping* (SLAM) [18], and to odometry and obstacle avoidance [13].

As pointed out in [11], RGB-D sensors that use the structured-light sensing principle, e.g. the Kinect, are built using CMOS image sensors with rolling shutters (RS). A sensor with a rolling shutter has a line-by-line exposure, and this will cause geometric distortions in both the colour images and the depth maps from an RGB-D sensor, whenever either the sensor or objects in the scene are moving. Illustrations of the rolling shutter effect on meshes reconstructed using an RGB-D sensor can be found in Figure 1.



(a) Pan (left to right) (b) Tilt (upwards)

Figure 1: Synthetic visualization of rolling shutter effects on reconstructed meshes. The solid shape is the correct mesh and the wireframe is the distorted mesh.

In this paper we investigate how the influence of rolling shutter distortions in dense SLAM can be mitigated, by equipping an RGB-D sensor with a 3-axis MEMS gyro sensor. Gyro sensors for consumer electronics are inexpensive and can provide angular velocity measurements at rates well above most camera frame rates. The use of a gyro sensor means that the extra computational burden of optical flow, and non-linear optimisation used for RS rectification in [11] need not be added to the already high cost of SLAM computation. An additional benefit is that the gyro provides angular velocity also in scenes with low texture, where optical flow computation is difficult.

Our experiments are designed around the Kinect Fusion algorithm [9], which has gained popularity due to the open source KinFu implementation in PCL [14]. We characterize the situations where RS distortions occur, and investigate to what extent a rotation based RS rectification can improve the output of Kinect Fusion.

1.1. Related Work

As our system makes use of an external gyroscope sensor, we require the clocks of the camera and the gyroscope to be synchronized, and their relative pose to be known. Several methods for camera-IMU calibration have been proposed in the past. A recent example is [7]. Such methods

typically compute the full relative pose between IMU and camera (both translation and rotation). In this paper we have instead chosen to introduce a new calibration scheme, for two reasons: Firstly, all current algorithms assume global shutter geometry, which means that their application to a rolling shutter camera needs to be done with care. Secondly, when only gyro measurements are to be used, the translation component is not needed, and this allows us to simplify the calibration considerably.

Our time synchronization procedure uses the same cost-function as [5, 8]. Just like [5] we only search for the time shift, but instead of performing gridding on the cost-function, we solve for the time shift in two steps: Firstly we find a coarse alignment using correlation of the device motion function, secondly we refine this estimate using derivative free search. In contrast to [5, 8], our method also deals with finding the unknown time scaling factor.

A dataset for evaluation of RGB-D SLAM accuracy was recently introduced by TU Münich and University of Freiburg, and will be presented at IROS 2012 [17]. Evaluation using this dataset consists of comparing a camera motion trajectory against ground truth from a motion capture system. As we rely on gyro measurements, we cannot use these datasets, and instead we demonstrate the effectiveness of our algorithm by comparing obtained 3D models with, and without applying our depth-map rectification.

The rolling shutter problem has been extensively studied in the past [3, 12, 1, 11]. Most closely related to our work is [11], on which we base our rectification scheme. In [11], the RGB-D device motion is computed using a sparse optical flow that is obtained from Kinect NIR images. Instead of using optical flow, we rectify the depth maps using the angular velocity provided by a 3-axis MEMS gyro sensor. This makes the resultant system more robust, as we can easily deal with two cases that are challenging for optical flow based techniques: 1. Scenes with large untextured regions 2. Scenes where the amount of ambient light present in the scene is too low.

1.2. Structure

This paper is divided into three parts: Section 2 describes the gyro and rolling shutter camera calibration. Section 3 describes our approach to depth map rolling shutter rectification. In Section 4 we perform a number of experiments that show the effect of rolling shutter rectification for RGB-D cameras.

1.3. Notation

We use superscripts to denote the used frame of reference where needed. t^g and t^c denote time in the gyro and RGB-D camera frames of reference respectively. A relation between frames is expressed as combinations, e.g. $\mathbf{R}^{cg}$ is the rotation from the gyro frame to the camera frame of reference. Vectors and matrices are expressed in bold ($\mathbf{x}$, $\mathbf{\Omega}$).

2. Sensor Calibration

The gyro provides angular velocity measurements for each of its three axes as the angular velocity vector $\boldsymbol{\omega}(t^g) = (\omega_x, \omega_y, \omega_z)$. The RGB-D camera provides us with RGB images $\mathbf{I}(t^c)$ and depth images $\mathbf{D}(t^c)$.

To associate the data from the two sensors we must know the relation between their timestamps, t^g and t^c, as well as the relative pose, $\mathbf{R}^{gc}$, between their coordinate frames.

2.1. Synchronizing the Timestamps

Assuming that both timestamp generators provide timestamps that are linear in time, the two timestamps will be related via a linear function

$$t^g = m^{gc} \cdot t^c + d^g \,. \tag{1}$$

The multiplier m^{gc} will be constant when both timestamp generators are stable and do not drift. The time offset d^g depends on when each timestamp generator was initialized. Typically reinitialization can occur at any time due to e.g. a hardware reset, which makes it necessary to recompute d^g for every experiment.

Although the multiplier has to be known in order to calculate the offset, we will start by describing how to calculate the latter.

2.1.1 Finding the Offset

Since the offset, d^g, has to be computed for every experiment it should be fast to compute. To achieve this we divide the task into first finding a rough estimate which is then refined.

The rough offset is found based on the assumption that a rotation of the sensor platform will in some way be observable by both sensors. For the gyro this is trivial. For the RGB-D sensor we assume that the rotation is manifested in the optical flow magnitude between consecutive frames.

We begin by defining the gyro speed as

$$W(t^g) = \|\boldsymbol{\omega}(t^g)\| \,. \tag{2}$$

The optical flow displacement magnitude is calculated as

$$\underbrace{F(t^c_j)}_{j=1\ldots(J-1)} = \frac{1}{N}\sum_{n}^{N} \|\mathbf{x}_n(t^c_j) - \mathbf{x}_n(t^c_{j+1})\| \,, \tag{3}$$

which is the average optical flow at the frame j, where N is the number of points tracked between frames j and $j+1$, and $\mathbf{x}_n(t^c_j)$ are the image coordinates of the tracked point n in frame j.

The assumed proportionality between $F(t^c)$ and $W(t^g)$, after mapping through (1), then becomes

$$W(t^g) \propto F(\frac{t^g - d^g}{m^{gc}}) \,. \tag{4}$$

After applying the multiplier m^{gc} and resampling the signal with the lowest sampling rate to match the other, we can use cross-correlation to find the offset d^g.

It is important to note that the chosen sensor platform movement must be such that it produces a signal form that is not periodic, as this could make the cross-correlation fail. Since data collection from both sensors is initiated at approximately the same time, we can extract slices of the original signals which are known to contain the synchronization movement. An example of the proportionality and the synchronization movement can be seen in Figure 2.

2.1.2 Refining the Time Offset

The correlation based time offset was found to be accurate to about ± 2 frames. Since we need sub-frame accuracy, the time offset must be refined further.

In [5] Hanning et al. describe a method to find an unknown offset between image timestamps and gyro timestamps. Points are tracked through an image sequence, and a grid search is used to find the time offset that best removes the rolling shutter effects.

Since we know that the offset is off by at most a few frames, the function to be optimized will be convex. This allows us to replace grid search with the much more efficient Brent's method [10].

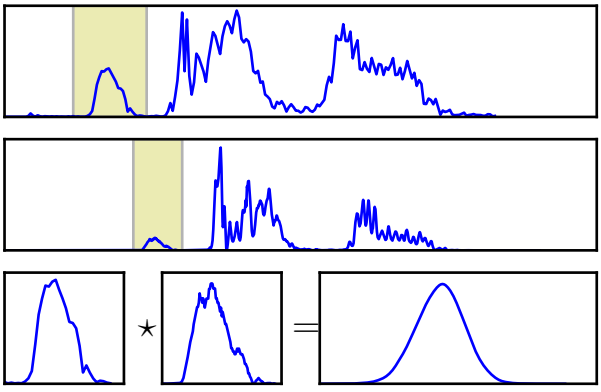


Figure 2: Optical flow and gyro speed comparison. From top to bottom: Optical flow displacement magnitude $F(t^c)$, gyro angular speed $W(t^g)$, and correlation response. Correlation is calculated from slices known to contain the synchronization movement (highlighted).

2.1.3 Finding the multiplier

To find the multiplier, the sensor platform is kept still except for two short and distinct movements, where the second movement is delayed sufficiently long.

We generate two short rotations of the sensor platform which will both be observable in both $W(t^g)$ and $F(t^c_j)$. The time between the two rotations, T, is measured in each sensor's frame of reference and the multiplier is calculated as the average of N such sequences

$$m^{gc} = \frac{1}{N} \sum_{n}^{N} \frac{T^g_n}{T^c_n} \,. \tag{5}$$

The calculated multiplier is then assumed to be valid for sequences of at least length T.

2.2. Relation of Coordinate Frames

No matter how carefully the IMU and camera are joined there will likely be some alignment error which could disturb the results [7].

A relative pose consists of a rotation and a translation from one coordinate frame to another. For our implementation only the gyro is used so the translation is not needed.

The basic idea of the relative pose estimation is that if we have two or more orientation vectors in one coordinate frame, and corresponding orientation vectors in the other coordinate frame we can find uniquely the rotation between them.

By rotating the sensor platform around two orthogonal axes we find the axes of rotation as seen by the camera coordinate frame and the IMU coordinate frame.

2.2.1 Gyro Coordinate Frame

Given a sequence where the gyro is rotating, we want to find the axis of rotation $\hat{\mathbf{r}}$. We do this by defining the following maximization problem:

$$\hat{\mathbf{r}} = \arg\max_{\mathbf{r}} J(\mathbf{r}) \tag{6}$$

$$J(\mathbf{r}) = \sum_{n=1}^{N} \|\mathbf{r}^T \boldsymbol{\omega}_n\|^2 \tag{7}$$

$$J(\mathbf{r}) = \sum_{n=1}^{N} \mathbf{r}^T \boldsymbol{\omega}_n \boldsymbol{\omega}_n^T \mathbf{r} = \mathbf{r}^T \underbrace{\left(\sum_{n=1}^{N} \boldsymbol{\omega}_n \boldsymbol{\omega}_n^T \right)}_{\boldsymbol{\Omega}} \mathbf{r} \tag{8}$$

Here N is the total number of gyro samples in the chosen sequence.

The rationale of this cost function is that the principal axis of rotation should be parallel to $\boldsymbol{\omega}$, and large velocities

should have a larger influence on the result. The solution $\hat{\mathbf{r}}$ is the eigenvector of $\mathbf{\Omega}$ with the largest eigenvalue.

We are however not certain if we have found $\hat{\mathbf{r}}$ or $-\hat{\mathbf{r}}$ as both will give the same cost. The sign can be determined by testing whether the scalar product between the acquired $\hat{\mathbf{r}}$ and all $\boldsymbol{\omega}_n$ is positive or negative such that

$$\hat{\mathbf{r}} \leftarrow \operatorname{sgn}\left(\sum_{n=1}^{N} \hat{\mathbf{r}}^T \boldsymbol{\omega}_n\right) \hat{\mathbf{r}}\,. \tag{9}$$

2.2.2 Camera Coordinate Frame

The problem of finding the rotation axes of the camera can be formulated as an *Orthogonal Procrustes problem* [15, 4]. For a given rotation sequence we track a number of points from the first image frame to the last. To make sure the resulting point correspondences are of high quality the points are retracked from the last image to the first. Points are discarded if the distance from the original point is larger than 0.5 pixels. It is very important that the first and last frame of the sequence are captured when the sensor platform is not moving, otherwise rolling shutter effects would bias the result.

Using the camera calibration matrix $\mathbf{K}$, and a depth map $z(u, v)$ the 2D points are back projected to 3D using the equation

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = z(u,v)\mathbf{K}^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix}\,. \tag{10}$$

If we denote the set of 3D points from the first and last image $\mathbf{X} = (\mathbf{X}_1, \mathbf{X}_2, ...)$ and $\mathbf{Y} = (\mathbf{Y}_1, \mathbf{Y}_2, ...)$ respectively, the problem of aligning them can be formulated as

$$\arg\min_{\mathbf{R},\mathbf{t}} \|\mathbf{X} - (\mathbf{R}\mathbf{Y} + \mathbf{t})\|^2 \quad \text{s.t.} \quad \mathbf{R}\mathbf{R}^T = \mathbf{I} \tag{11}$$

where $\mathbf{R}$ is a rotation matrix and $\mathbf{t}$ is a translation.

Procrustes now gives us an estimate of $\mathbf{R}$ and $\mathbf{t}$ using the SVD

$$\mathbf{U}\mathbf{D}\mathbf{V}^T = \operatorname{SVD}[(\mathbf{X} - \boldsymbol{\mu}_X)(\mathbf{Y} - \boldsymbol{\mu}_Y)^T] \tag{12}$$

$$\mathbf{R} = \mathbf{U} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{U}\mathbf{V}^T) \end{pmatrix} \mathbf{V}^T \tag{13}$$

$$\mathbf{t} = \boldsymbol{\mu}_X - \mathbf{R}\boldsymbol{\mu}_Y\,. \tag{14}$$

Here $\boldsymbol{\mu}_X$ and $\boldsymbol{\mu}_Y$ are the means of the vectors $\mathbf{X}$ and $\mathbf{Y}$ respectively.

Although the Procrustes solution provides us with both rotation and translation, only the rotation is needed in our implementation.

The rotation matrix $\mathbf{R}$ can be written on axis-angle form as $\varphi\hat{\mathbf{n}}$ where $\hat{\mathbf{n}}$ is the principal axis of the rotation that we want to find.

To convert from matrix form to axis-angle form we use the method described by Hartley and Zisserman [6]

$$2\cos\varphi = \operatorname{trace}(\mathbf{R}) - 1 \tag{15}$$

$$2\sin\varphi\hat{\mathbf{n}} = \begin{pmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{pmatrix}\,. \tag{16}$$

It is worth noting that this representation forces the rotation angle φ to only take positive values. Therefore, if we have two rotations about the same axis $\varphi_1\hat{\mathbf{n}}$ and $\varphi_2\hat{\mathbf{n}}$ where $\varphi_2 = -\varphi_1$ we would measure the latter as $\varphi_1(-\hat{\mathbf{n}})$ which is a positive angle and a flipped rotation axis. This makes the result consistent with the behaviour of the gyro rotation axis calculation in (9).

2.2.3 Calculating the Relative Pose

Using the methods in sections 2.2.1 and 2.2.2 we can collect a set of corresponding rotation axes, $\mathbf{X}^g$ and $\mathbf{X}^c$, in the gyro coordinate frame and RGB-D camera coordinate frame respectively.

Once again we can formulate an Orthogonal Procrustes's Problem to find the relative sensor pose

$$\begin{aligned} \mathbf{R}^{cg}, \mathbf{t}^{cg} &= \arg\min_{\mathbf{R},\mathbf{t}} \|\mathbf{X}^c - (\mathbf{R}\mathbf{X}^g + \mathbf{t})\|^2 \\ &\text{s.t.} \quad \mathbf{R}\mathbf{R}^T = \mathbf{I} \end{aligned} \tag{17}$$

We typically use two forward-backward sequences, along two approximately orthogonal axes. This gives us four measurements in total, from which the relative pose may be determined.

3. Depth Map Rectification

3.1. Gyro integration

We obtain the rotation of the sensor platform (relative to some initial orientation) by integrating the gyro angular velocity measurements. This accumulated rotation is later used for depth map rectification.

The accumulated rotation at time t is denoted by the unit quaternion

$$\mathbf{q}(t) = [\cos\frac{\varphi}{2}; \sin\frac{\varphi}{2}\mathbf{n}]\,, \tag{18}$$

where the unit vector $\mathbf{n}$ is the axis of rotation and φ is the magnitude of the rotation.

Using the timestep Δt and angular velocity measurements $\boldsymbol{\omega}$, the integration becomes

$$\mathbf{q}(0) = [1;\, \mathbf{0}] \quad \text{and} \tag{19}$$

$$\mathbf{q}(t + \Delta t) = \mathbf{q}(t) \odot \mathbf{w}(\boldsymbol{\omega}; \Delta t)\,, \quad \text{where} \tag{20}$$

$$\mathbf{w}(\boldsymbol{\omega}; \Delta t) = \left[\cos\left(\frac{\|\boldsymbol{\omega}\| \Delta t}{2} \right) ; \frac{\sin\left(\frac{\|\boldsymbol{\omega}\| \Delta t}{2} \right)}{\|\boldsymbol{\omega}\|} \boldsymbol{\omega} \right] . \tag{21}$$

Here $\odot$ is the quaternion multiplication operator. $\mathbf{q}$ is kept as a unit quaternion by renormalizing after each step.

3.2. Rectification

Once the camera and IMU are synchronized and their relative pose is known we can perform the rectification.

We are only measuring rotations, so the update equation for image coordinates from [11] simplifies to

$$\mathbf{x}' = \mathbf{K}\mathbf{R}(t_{\text{mid}})\mathbf{R}^T(t_{\text{row}})\mathbf{K}^{-1}\mathbf{x}\,, \quad \text{where} \tag{22}$$

$$t_{\text{row}} = t_0 + t_r \frac{x_2}{\#\text{rows}} \quad \text{and} \tag{23}$$

$$t_{\text{mid}} = t_0 + t_r \frac{1}{2}\,. \tag{24}$$

Here t_{row} and t_{mid} are the times when the current row and middle row were captured given the start of frame time, t_0, and readout time, t_r. $\mathbf{R}(t)$ is the rotation of the camera at time t, which is constructed as $\mathbf{R}(t) = \mathbf{R}^{cg}\mathbf{M}(\mathbf{q}(t))$, where

$$\mathbf{M}(\mathbf{q}) = \begin{pmatrix} 1 - 2q_2^2 - 2q_3^2 & 2q_1q_2 + 2q_0q_3 & 2q_1q_3 - 2q_0q_2 \\ 2q_1q_2 - 2q_0q_3 & 1 - 2q_1^2 - 2q_3^2 & 2q_2q_3 + 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 - 2q_0q_1 & 1 - 2q_1^2 - 2q_2^2 \end{pmatrix}, \tag{25}$$

transforms a unit quaternion to a rotation matrix [16]. $\mathbf{x}$ and $\mathbf{x}'$ are homogenous image coordinates before and after rolling shutter rectification.

A way to interpret (22) is that a 2D point is back-projected to a 3D point, which is rotated back to the initial camera position, then rotated back to the time the middle row was captured, and finally projected again to a 2D point.

The inevitable drift accumulated in $\mathbf{q}(t)$ in previous frames is effectively cancelled out, as the combined rotation in (22) is relative to the middle row, and not to the start of the sequence.

Note that the fact that we are neglecting translations also means that the depth $z(\mathbf{x})$ can be ignored in the projections as it is now simply a scale factor in a homogenous equation.

Using (22) we can construct a forward mapping for each pixel coordinate in the original image. We use this to forward interpolate new rectified depth images.

In order to propagate also the the valid/invalid status of pixels each depth image is interpolated twice. First an interpolation using a weighted gaussian 3x3 kernel is used on all pixels that have valid depth values. Second we interpolate using nearest neighbour interpolation all pixels with invalid depth values. This avoids having invalid pixels become valid due to interpolation.

4. Experiments

We will now demonstrate the gain with using rolling shutter rectification when doing 3D reconstruction using an RGB-D camera. Three experiments have been performed on three different types of sensor movement: pan, tilt and wobble. Other types of sensor movement such as roll and translation also cause rolling shutter effects but have not been investigated. Roll motions can be handled by our method, but visualizing and measuring the effect is difficult. As our method neglects translations, rolling shutter effects from translations are not handled. However, these effects appear only when the image plane is moving at high speeds which makes the impact much weaker than rotational movements.

To carry out the reconstruction we have used the Kinect Fusion algorithm implemented in PCL [14] under the name KinFu.

Our sensor platform consists of one Kinect RGB-D camera, to which is attached an ArduIMU gyro and accelerometer. The Kinect captures depth images at 29.97 Hz and RGB images at 30Hz, while the ArduIMU provides gyroscope measurements at approximately 170Hz. The discrepancy between the frame rates of the RGB and depth camera can safely be ignored during 3D reconstruction, since only depth images are used here. To capture the Kinect data we wrote a data logging application that makes sure that no frames are skipped and also uses the raw timestamps from the clock on board the Kinect.

We use the method in Section 2 to synchronize the RGB camera and the gyroscope. Using the raw timestamps is important as RGB and depth timestamps are then in the same time frame, and the synchronization is thus automatically valid also for the depth frames. The Kinect timestamps are 32-bit unsigned integers generated by a 60 MHz clock.

4.1. Pan and Tilt distortions

Items of varying size were placed on a desk that was surrounded by screens to avoid background clutter (see Figure 3). The data logging applications were started and after that a short time synchronization action was performed, consisting of a short panning angular motion while the sensor platform was standing on the desk.

The experiment was performed such that the objects of interest in the scene were in view of the RGB-D sensor only during motion. But to allow the Kinect Fusion some good frames to initialize, the sensor platform was initially held as still as possible while not observing the objects of interest.

Figure 3: Scene used for experiments

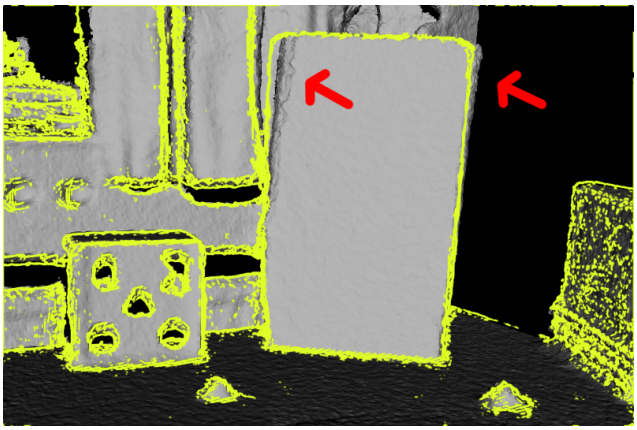


Figure 4: Grey: Mesh created by Kinect Fusion, captured while panning with an angular speed of about 1 rad/sec. Yellow: Outline of mesh from rectified depth frames. Arrows indicate the rolling shutter artifacts.

Skipping this initial step often resulted in failed reconstructions. The sensor platform was then either panned from left to right or tilted down to up. For each motion, three sequences with different angular speed were recorded.

After the data was captured a rectified version was created using the method described in Section 3. Both sequences were then processed by the Kinect Fusion algorithm which produced two reconstructed mesh representations of the scene. Kinect Fusion was run in offline mode to make sure that all frames were handled, and that processing did not depend on the GPU speed.

In Figure 4 we visualize the result of one pan motion experiment. The original and rectified mesh were aligned, and the outline (an edge map) of the rectified mesh was drawn on top of the original mesh. To align the meshes we used the iterative closest point algorithm (ICP) [2]. The meshes are not related through a simple rigid transformation, so the entire mesh can not be used for ICP alignment. Since our scene had a flat ground surface we instead opted to align the meshes such that this ground plane was aligned as well as possible. We selected points on the desk surface and close to distinct objects in both meshes, and applied ICP to this smaller point set.

Pan	$90.59^\circ \pm 0.56$
	$89.57^\circ \pm 0.48$
Tilt	17.3 ± 0.1

(a) Ground truth measurements. See tables b and c for details.

$\|\boldsymbol{\omega}\|$	Original	Error	Rectified	Error
0.5	$89.52^\circ \pm 0.67$	-1.1°	$90.44^\circ \pm 0.27$	-0.1°
	$90.46^\circ \pm 0.19$	0.9°	$89.51^\circ \pm 0.57$	-0.1°
1.1	$87.11^\circ \pm 1.07$	-3.5°	$90.27^\circ \pm 0.35$	-0.3°
	$92.95^\circ \pm 0.57$	3.4°	$89.99^\circ \pm 0.51$	0.4°
2.5	$87.00^\circ \pm 0.17$	-3.6°	$91.51^\circ \pm 0.49$	0.9°
	$97.41^\circ \pm 0.28$	7.8°	$92.74^\circ \pm 0.47$	3.1°

(b) Pan measurements. The two upper angles of the rectangular box, measured in degrees.

$\|\boldsymbol{\omega}\|$	Original	Error	Rectified	Error
0.7	16.9 ± 0.3	0.4	16.8 ± 0.1	0.5
1.1	17.7 ± 0.1	0.3	17.2 ± 0.1	0.1
2	19.2 ± 0.2	1.9	18.3 ± 0.1	1.0

(c) Tilt measurements. Height of cube object, measured in cm.

Table 1: Measurements for different angular speeds (in rad/sec). The raw data is expressed as $\mu \pm \sigma$ where μ is the mean and σ is the standard deviation of the measurement. The error columns of tables b and c are deviations from the ground truths in table a.

To measure the impact of the rectification we compared the obtained meshes with a ground truth mesh. The ground truth mesh was reconstructed from a long sequence where the sensor platform was moving at very low speed. The ground truth measurements are presented in Table 1a. A panning motion should produce a slant in the depth images, and we therefore expect the mesh to become slanted (see Figure 1a). By finding the corners of the front face of the large rectangular box, the angles of its corners can be calculated. Since clicking in the mesh is prone to errors, each angle measurement was done five times and the mean was used as the true angle. Table 1b shows the deviations from the ground truth for the two upper angles of the rectangular box for different angular speeds.

With a tilt motion in the upwards direction, the rolling shutter effect causes objects to become extended and appear longer than they really are (see Figure 1b). Once again

we measured the deviation from the ground truth mesh, by measuring the height of the smaller cube-shaped object. Like before, the height was measured five times in each mesh, and the average height was used as the true height. The results can be found in Table 1c.

Note that at angular speeds of about 2 rad/sec and above, the reconstructed meshes are bad due to large amounts of motion blur.

4.2. Wobble distortions

The wobble experiment was carried out by keeping an object (in our case, a telephone) in view of the sensor while shaking the sensor platform. In contrast to the pan and tilt experiment, with wobble we do not expect the general shape of the objects in the scene to change. However, since the Kinect Fusion algorithm integrates measurements over time, we do expect rolling shutter wobble to blur out smaller details. In Figure 5 we show a zoomed in view of the telephone, and one can see that e.g. the buttons and cables are more pronounced in the rectified version than in the original version.

We examined the frequency of the wobble by applying the FFT to each axis, and calculated a conservative estimate of the combined frequency content as

$$G(f) = \sqrt{|\Omega_x(f)|^2 + |\Omega_y(f)|^2 + |\Omega_z(f)|^2}\,. \qquad (26)$$

The result in Figure 6 shows that the frequency of our handheld wobble was approximately 4 Hz. We can also see that the energy beyond 15 Hz is negligible, which implies that our sampling rate of 170 Hz is sufficient.

5. Concluding Remarks

In this paper we have shown that the rolling shutter effect will create notable errors in 3D reconstructions from the Kinect Fusion algorithm. We also show that these errors can be mitigated by applying rolling shutter rectification on the depth data before 3D reconstruction.

We have also introduced a simple scheme for calibrating the time synhronization and relative orientation between a gyroscope and a rolling shutter RGB-D sensor.

In the future we would like to improve the depth map rectification scheme. Our current approach, while avoiding interpolation of bad depth values, sometimes produces jagged edges due to nearest neighbour interpolation.

A potential benefit of RS rectification which is not studied here is reduction of drift in SLAM. Once the Kintinuous SLAM system [18] is made available for testing by other researchers, it would be of great interest to feed it with rectified Kinect scans, and to investigate the benefits of this.

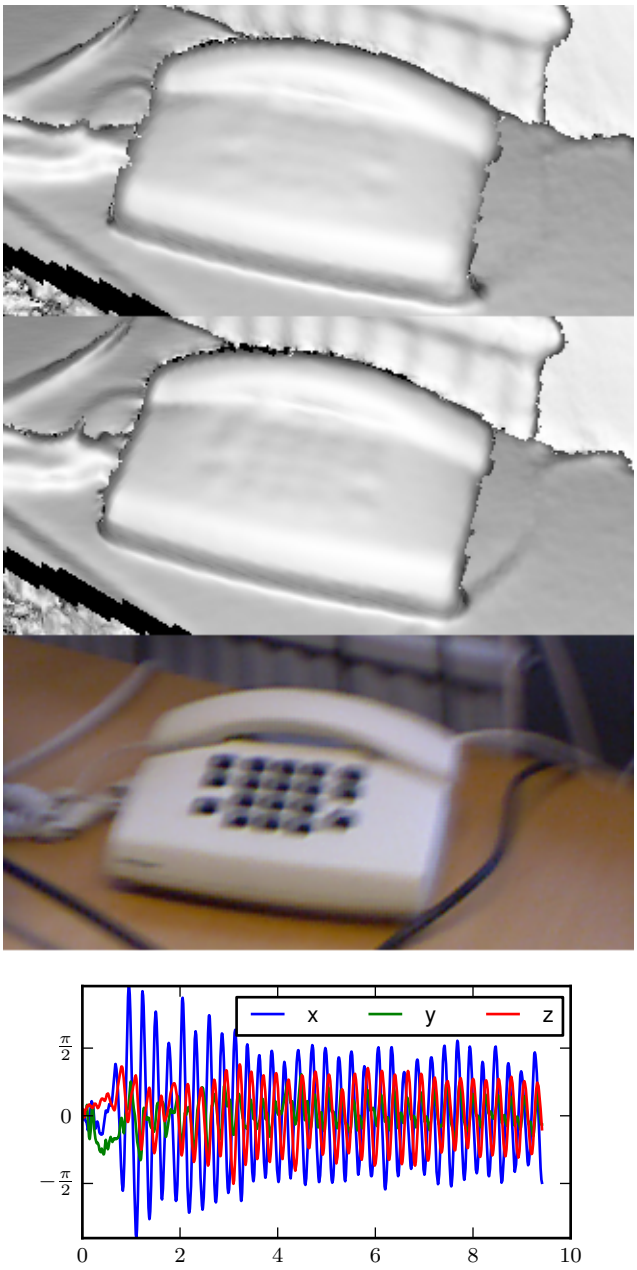


Figure 5: Wobble experiment. Zoom in on raycasted mesh. From top to bottom: Original mesh, rectified mesh, one selected RGB image from the sequence, and gyro measurements.

6. Acknowledgements

This work was supported by the Swedish Research Council through a grant for the project *Embodied Visual Object Recognition*, and by Linköping University.

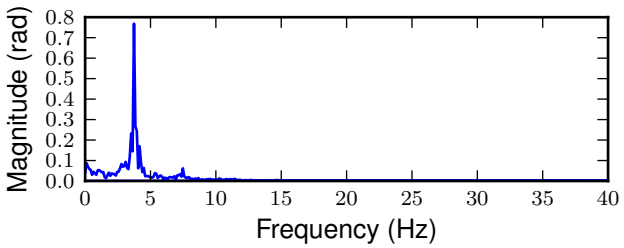


Figure 6: Composite frequency content, $G(f)$, of the wobble experiment.

References

[1] S. Baker, E. Bennett, S. B. Kang, and R. Szeliski. Removing rolling shutter wobble. In *IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, USA, June 2010. IEEE Computer Society.

[2] P. Besl and H. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[3] C. Geyer, M. Meingast, and S. Sastry. Geometric models of rolling-shutter cameras. In *6th OmniVis WS*, 2005.

[4] G. H. Golub and C. F. van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 1983.

[5] G. Hanning, N. Forslöw, P.-E. Forssén, E. Ringaby, D. Törnqvist, and J. Callmer. Stabilizing cell phone video using inertial measurement sensors. In *The Second IEEE International Workshop on Mobile Vision*, Barcelona, Spain, November 2011. IEEE.

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[7] J. D. Hol, T. B. Schön, and F. Gustafsson. Modeling and calibration of inertial and vision sensors. *International Journal of Robotics Research*, 29(2):231–244, February 2010.

[8] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy. Digital video stabilization and rolling shutter correction using gyroscopes. Technical Report CSTR 2011-03, Stanford University Computer Science, September 2011.

[9] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality ISMAR'11*, Basel, Switzerland, October 2011.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing*. Cambridge University Press, New York, NY, USA, 1992.

[11] E. Ringaby and P.-E. Forssén. Scan rectification for structured light range sensors with rolling shutters. In *IEEE International Conference on Computer Vision*, Barcelona, Spain, November 2011. IEEE, IEEE Computer Society.

[12] E. Ringaby and P.-E. Forssén. Efficient video rectification and stabilisation for cell-phones. *International Journal of Computer Vision*, 96(3):335–352, February 2012.

[13] H. Roth and M. Vona. Moving volume kinectfusion. In *British Machine Vision Conference (BMVC12)*, University of Surrey, UK, September 2012. BMVA, BMVA. http://dx.doi.org/10.5244/C.26.112.

[14] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[15] P. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, March 1966.

[16] K. Shoemake. Animating rotation with quaternion curves. In *Int. Conf. on CGIT*, pages 245–254, 1985.

[17] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, October 2012.

[18] T. Whelan, J. McDonald, M. Kaess, M. Fallon, H. Johannsson, and J. J. Leonard. Kintinuous: Spatially extended kinectfusion. In *RSS 2012 Workshop on RGB-D Cameras*, Sydney, July 2012.