Rolling Shutter and Motion Blur Removal for Depth Cameras

Siddharth Tourani[1] Sudhanshu Mittal[2] Akhil Nagariya[1] Visesh Chari[1] Madhava Krishna [1]

***Abstract*— Structured light range sensors (SLRS) like the Microsoft Kinect have electronic rolling shutters (ERS). The output of such a sensor while in motion is subject to significant motion blur (MB) and rolling shutter (RS) distortion. Most robotic literature still does not explicitly model this distortion, resulting in inaccurate camera motion estimation. In RGBD cameras, we show via experimentation that the distortion undergone by depth images is different from that of color images and provide a mathematical model for it. We propose an algorithm that rectifies for these RS and MB distortions. To assess the performance of the algorithm we conduct an extensive set of experiments for each step of the pipeline. We assess the performance of our algorithm by comparing the performance of the rectified images on scene-flow and camera pose estimation, and show that with our proposed rectification, the performance improvement is significant.**

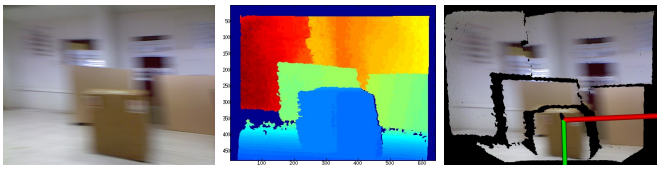
I. INTRODUCTION

There are two types of shutters used in digital cameras. One is the global shutter (GS) where the entire image is captured simultaneously, the other is the rolling shutter (RS) where the image is captured row-by-row at slightly displaced time instances as shown in Fig. 2. Recently cameras equipped with ERS have become popular due to their low cost, low power consumption. This, however, comes at the cost of a more complicated camera projection model. Due to this RS cameras capture distorted images when the camera is moving. Unless these distortions are modelled and rectified the use of SLRS are limited to slow motions, when these effects are negligible.

Another distortion introduced in image data, due to camera motion is motion blur. Much like the RS distortion, MB negatively impacts the camera pipeline during exposure time. Motion blurring causes the smearing of edges and distinct points, which are image primitives on which all subsequent vision algorithms rely. MB is modelled as a convolution of a latent image $\mathtt{l}$ with a kernel $\mathtt{k}$ giving a blurred image $\mathtt{b} = \mathtt{k} * \mathtt{l}$. The job of a de-blurring algorithm is to recover $\mathtt{l}$ from $\mathtt{b}$. This is an ill-posed problem and requires sophisticated priors to be solved.

We discovered while capturing data from the Kinect that the nature of blur affecting the depth sensor is substantially different from the one affecting color images. In color images blur has an averaging effect on the image where the value of a pixel at a point is the weighted average of it?s surroundings. For depth images, we found that blur causes a much greater loss of high-frequency information. More specifically we found that the depth observed by the pixel was the minimum of all the depths it was exposed to during the row exposure period. We call this the **min-filter effect**. This effect has

[1]Robotics Research Center, IIIT Hyderabad, India
[2]KTH, Sweden



(a) (b) (c)

Fig. 1. Rolling shutter effect in consumer depth cameras. (a) shows a color image captured from a Kinect. In addition to the motion blur, the camera motion also causes a slanting of all the vertical lines in the image. (b) shows a color map of the corresponding depth image. RS distortion here causes, in addition to the slanting, fattening of the depth map at the edges of objects, as seen clearly for the box. (c) Shows the point cloud corresponding to (a) and (b). The color and depth images are misaligned, and some of the background from the floor and table project onto the box.

an **edge-fattening** effect which causes the boundaries of the depth map to extend beyond the actual boundaries of the object.

A vast majority of the computer vision literature has algorithms designed for slow-moving cameras. The paradigm of fast motion is as of yet widely unexplored. The algorithms that deal with high-level robotic vision tasks, like object recognition and viewpoint estimation work on images taken from slow-moving cameras. So, to make use of these algorithms in their current form, we would like to be able to completely remove RS+MB distortion, from both the depth and intensity images. Also, we would like to have a perfectly aligned depth map.

An algorithm capable of providing this would be ideal, but is unlikely, given the severely ill-posed nature of blur removal, in both the intensity and depth images. In this paper, we propose an algorithm that is a step in this direction.

We start off by rectifying for RS in depth maps. As depth information is more accurate than image intensity in camera motion estimation, we use it to estimate the camera trajectory. This is done by first estimating the time continuous 3D camera trajectory, and then transforming the 3D points to where they would have been, if the camera had been stationary. We adopt the new $\mathbb{SE}(3)$ spline parametrization proposed in [1] for the camera trajectory. Once the depth map is rectified, using the spline coefficients, we can interpolate the camera trajectory to the intensity image time-stamp and rectify for it as well. This solves the RS distortion, which is the easy portion of our problem. To solve for MB we project the estimated camera motion, from the previous step during camera exposure onto the image plane. This serves as the blur kernel for the intensity image deblurring. The min-filter is far too lossy an operation on the original depth map for recovery. So instead we settle for giving a probabilistic labelling of the uncertain pixels in the edge fattened depth map. This can be thresholded to give a depth map, with reduced edge fattening.

A. Related Work

There have been several works that address rolling shutter rectification. Due to space constraints, we focus only on the ones relevant to us. Most approaches for monocular cameras [2, 3, 4, 5] assume the availability of additional sensors (IMU's) / certain scene structures (planes). We don't need any such assumption due to the availability of depth map information. Also, they focused only on RS correction and did not address the problem of MB.

Recently, in [6], both RS and MB were addressed in a joint formulation, and showed that estimation RS rectification helped in de-blurring. Again, in our case access to the depth map allows for improved camera pose estimation, and subsequently improved deblurring.

For SLRS cameras, within the literature, we found only one work addressed RS removal in depth cameras [7]. In it they used a linear spline parametrization (via Spherical Linear Interpolation, SLERP) [8] for the camera trajectory, and formulated an ICP like algorithm for depth map correction. In it they enforce a uniform camera velocity during exposure. Rather than enforcing a constant velocity, we choose to add it as a regularizer in our formulation, resulting in improved camera trajectory estimation, as shown in section IV-D. Also, Ringaby et al. restrict their discussion to RS rectification only.

B. Contributions

- We present a mathematical model for depth image blurring that explains the edge-fattening and loss of high-frequency information observed in depth maps taken from a fast moving camera. We give a method to correct for the edge-fattening.
- We present a pipeline that rectifies for both RS and MB in both intensity and color images, and show that it is able to reduce RS+MB in both depth and intensity images. To the best of our knowledge this is the first attempt made to rectify footage from a fast moving SLRS camera.
- We will release a new calibrated RS+MB correction dataset taken under a calibrated set-up, for large motions, which is a first of it's kind.
- We show a practical robotic application of our algorithm in a camera trajectory estimation task. The rectified output of our algorithm has a far more accurate camera trajectory, than the other methods compared against.

II. CAMERA MOTION MODEL

For the purposes of this paper, we assume that our scene is stationary. Fig. 2 illustrates the mechanism of capture for both the global and rolling shutter sensors. In the case of the GS, all rows of the sensor are simultaneously exposed to the light. In the presence of camera motion, the pixels in a GS camera integrate over the camera motion trajectory. In a RS camera different scan-lines integrate over slightly displaced segments of the camera trajectory, giving rise to the rolling shutter effect. MB in both types of sensors is caused by light from an object striking different parts of the sensor during

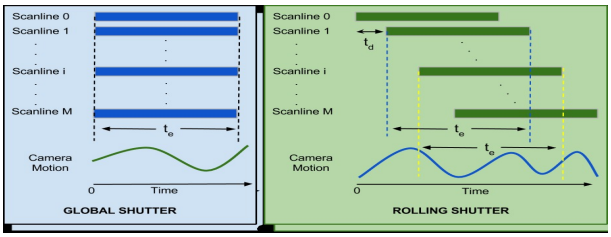


Fig. 2. Illustration of GS and RS sensors. Each horizontal bar represents the exposure time t_e of a scanline. In the presence of camera motion, all pixels in a GS camera integrate over the same motion trajectory, while different scanlines integrate over a different segment of the trajectory in a RS sensor. The difference in exposure start times between successive scanlines is called line delay and is given by t_d

exposure. We give in sections II-B and II-C, mathematical models describing the image capture for global and rolling shutter respectively.

A. Notation and Terminology

For clarity and neatness of mathematical exposition the symbols explained below are summarized in Table I

Symbol	Meaning
$\mathtt{t}$	Time
$\mathtt{I}(\mathtt{p},\mathtt{t}):\Omega \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$	Color Image Brightness Fn. wrt time
$\mathtt{I}(\mathtt{p}):\Omega \times \mathbb{R}^+$	Color Image Brightness Fn.
$\mathtt{D}(\mathtt{p},\mathtt{t}):\Omega \times \mathbb{R}^+ \rightarrow \mathbb{R}^+$	Depth Image Brightness Fn. wrt time
$\mathtt{D}(\mathtt{p}):\Omega \times \mathbb{R}^+$	Depth Image Brightness Fn.
$\mathtt{K}$	Intrinsic Calibration Matrix
$\mathtt{p_i}$	p_i in homogeneous co-ordinates
$\mathtt{o_i} = \mathtt{K}^{-1}\bar{\mathtt{p}}_\mathtt{i}\mathtt{D}(\mathtt{p_i})$	3D pt. corresponding to p_i
$\mathtt{f_v} = (\mathtt{v}, \boldsymbol{\omega}) \in \mathbb{R}^6$	v: Linear velocity, $\boldsymbol{\omega}$: Angular Velocity
$\mathtt{T}$	An $\mathbf{SE}(3)$ transformation
$\mathtt{w}(\mathtt{T}, \mathtt{o_i})$	warping function, transforms o_i by $\mathtt{T} \in \mathbf{SE}(3)$ followed by projection
$\mathtt{t_e}$,$\mathtt{t_d}$	Row exposure time, Line delay

TABLE I
NOTATION

The image domain is represented by Ω. $\mathtt{I}(\mathtt{p},\mathtt{t})$ is the instance of the intensity image at time $\mathtt{t}$ and pixel $\mathtt{p}$ during exposure. It is different from $\mathtt{I}(\mathtt{p})$ which is the intensity of the final formed image at pixel tp after exposure. Likewise, for the depth image $\mathtt{D}(\mathtt{p},\mathtt{t})$ and $\mathtt{D}(\mathtt{p})$. Homogeneous coordinates are represented with a bar on top of its corresponding symbol, like in $\bar{\mathtt{p}}$. To avoid cluttered equations, $\mathtt{K}$ is used to represent the camera calibration matrix in both homogeneous (3 3 matrix) and non-homogeneous form (4 4 matrix). Which use is intended should be clear from context. The 3D point $\mathtt{o_i}$ corresponding to pixel $\mathtt{p_i}$ is obtained by back-projecting $\mathtt{p_i}$, via $\mathtt{o_i} = \mathtt{K}^1\bar{\mathtt{p}}_\mathtt{i}\mathtt{D}(\mathtt{p_i})$. $\mathtt{f_v} = (\mathtt{v}, \boldsymbol{\omega})$ denotes the linear and angular velocity of the camera. $\mathtt{T} \in \mathbb{SE}(3)$ will sometimes be a function of time t. The warping function $\mathtt{w}$ transforms $\mathtt{o_i}$ by transformation $\mathtt{T}$ and then projects it onto the image plane.

$\mathtt{t_e}$ is the time duration during which a row of the camera is active. $\mathtt{t_d}$ is the line-delay between the start of exposure between two successive rows of the sensor (scanlines). Fig. 2 shows these two times diagrammatically. These constants are calibrated for using the technique from [7]. The $\mathtt{t_d}$ and

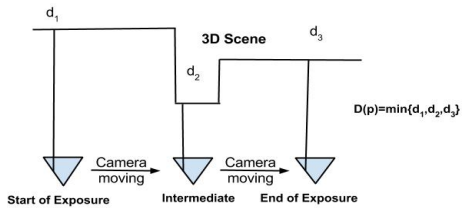


Fig. 3. Diagrammatic Illustration of the Edge-Fattening Effect for a single point during exposure.

$\mathtt{t_e}$ are different for depth and intensity images of the Kinect. $\mathtt{t_d}$ and $\mathtt{t_e}$ were 25.717ms and 26.231ms for intensity images. For depth images, we calculated 30.018ms and 31.528ms for $\mathtt{t_d}$ and $\mathtt{t_e}$ respectively.

B. Global Shutter

In a GS camera, at a particular time instance t, the image at pixel $\mathtt{p_i}$, $\mathtt{I}(\mathtt{p_i},\mathtt{t})$ depends on the light incident on the sensor from $\mathtt{o_i}$. Thus $\mathtt{I}(\mathtt{p_i},\mathtt{t}) = \mathtt{I}(\mathtt{w}(\mathtt{T},\mathtt{o_i}))$, where $\mathtt{T}$ is the transformation from the world frame to camera frame.

Now, the image formed by a moving GS camera results from integrating over all $I(p,t)$ seen by the camera along its motion path during its exposure period $t \in [0,t_e]$, i.e.

$$\mathtt{I}(\mathtt{p_i}) = \frac{1}{\mathtt{t_e}} \int_0^{\mathtt{t_e}} \mathtt{I}(\mathtt{w}(\mathtt{T}(\mathtt{tf_v}),\mathtt{o_{i,t}}))\mathtt{dt} \tag{1}$$

$$\mathtt{T}(\mathtt{tf_v}) = \mathtt{e}^{\mathtt{t}[\mathtt{f_v}]_\wedge}[f_v]_\wedge = \begin{bmatrix} [\omega]_\times & v \\ 0 & 0 \end{bmatrix} \in \mathfrak{se}(3) \tag{2}$$

$[.]_\times$ denotes the matrix exponential operator. $\mathtt{o} = \{\mathtt{o_i},\mathtt{t}|\mathtt{t} \in [0,\mathtt{t_e}]\}$ represents the set of 3D points which project to the pixel $\mathtt{p_i}$ during camera exposure. Eqn. 1 is a many-to-one mapping from $\mathtt{o}$ to $\mathtt{p_i}$ and is a result of the moving camera seeing different 3D points during exposure. Similarly, a 3D point $\mathtt{o_i}$ may project to different pixels of an image, resulting in another many-to-one mapping. This lack of distinctness gives rise to MB. From the equation, we can also see that when the camera is stationary, $\mathtt{f_v}$ is zero, $\mathtt{I}(\mathtt{p},\mathtt{t}) = \mathtt{I}(\mathtt{p})$, i.e., we recover the original image.

C. Rolling Shutter

In RS cameras, the final image formed is a result of the integration over different intervals of the camera trajectory. As shown in Fig. 2, the exposure intervals for two successive scan-lines is offset by the line delay $\mathtt{t_d}$. Let the final rolling shutter image for a particular row be $\mathtt{I_r}$, where $\mathtt{r}$ is the row number. The final image can be written as $\mathtt{I_{RS}} = [\mathtt{I_1},\ldots,\mathtt{I_R}]$.

$$\mathtt{I_r}(\mathtt{p_i}) = \frac{1}{\mathtt{t_e}} \int_{\mathtt{r.t_d}}^{\mathtt{r.t_d}+\mathtt{t_e}} \mathtt{I_r}(\mathtt{w}(\mathtt{T}(-\mathtt{tf_v}),\mathtt{o_{i,t}}))\mathtt{dt} \tag{3}$$

Here, $\mathtt{r}$ is the row number. $\mathtt{I_r}$ is the final image of the $\mathtt{r^th}$ row. The integration is carried out from $\mathtt{r.t_d}$ to $\mathtt{r.t_d}+\mathtt{t_e}$, for this row. The transformations $\mathtt{T}(.)$ and set of 3D points $\mathtt{o}$ are likewise dependent on the row number.

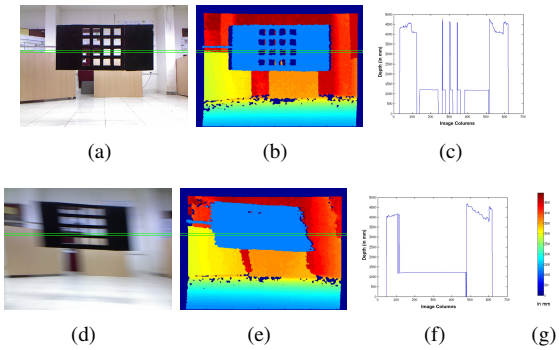


Fig. 4. **Please zoom in.** The Edge-Fattening Effect of Camera Motion on the Depth Image. First row: (a) and (b) are the color and depth images of a meshed plate taken by a stationary camera. In it is marked the row within a green rectangle. In (c) is plotted the depth values at this particular row, as a function of column number. This is for the row marked in green in (a). Second row: (d) and (e) are the color and depth images of the same scene taken by a camera rotating at 40 rpm, with the same row marked (f) is the corresponding depth histogram plot. (g) gives the color coding for the maps (b) and (e). Each pixel in the depth maps (b) and (e) encodes the depth recorded by the camera at that location. Missing information is given zero depth. As can be seen in the (b) and (c) for the stationary scene, the gaps in the viewed grill are present. As they are occluded in the depth maps, the values of the depth map at those locations are in-fact marked zero. The dips in (c) also show this. In comparison, in the second row, edge fattening causes the filling up of those gaps and causes a flat-lining of the histogram (f).

D. Edge-Fattening in the Depth Map

For the depth map, we observed that, rather than having an averaging effect (modelled as a convolution) motion blur has an effect more akin to a min filter, i.e, the final value registered by the depth map of the camera was the minimum over the depth of all points observed during exposure, i.e.

$$\mathtt{D_{final}}(\mathtt{p_i}) = \min_{\mathtt{t}\in[0,\mathtt{t_e}]} \{\mathtt{D}(\mathtt{o_{i,t}})\} \tag{4}$$

Here, $\mathtt{D}(\mathtt{o_k})$ is the depth at point $\mathtt{o_k}$, $\mathtt{o_{i,t}}$ is defined as in eqn. 1. The min-filter effect causes far more information loss than the averaging filter for intensity images, thus making depth image recovery even more ill-posed than intensity image recovery. This phenomenon was referred to as edge fattening in [9]. In our literature survey on this particular topic we have found very little explanation of the phenomenon.

Fig. 3 illustrates how the edge fattening works. During the time a particular sensor row is active, a number of rays reach the IR sensor of the Kinect. The minimum of these depths is taken resulting in the image shown in the figure.

Fig. 4 (a) and (b) show the color image and depth map taken of a high-frequency object (grill) when the camera is stationary. In them is marked a row by a green rectangle. (c) shows the depth plot of this row as a function of column number, of the row marked in green in (a). (d) and (e) show the results obtained the same when obtained by a camera rotating at 40 rpm as measured by our calibrated set-up. (f) is the corresponding plot to (c) for the same row. As can be seen in (e) in addition to the rolling shutter distortion the high-frequency image, gaps in the grill are lost due to edge fattening. The corresponding histogram is also completely flattened.

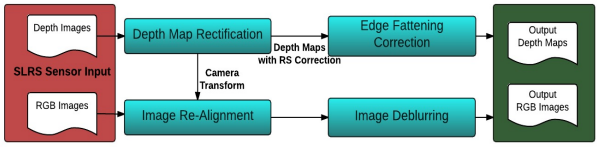


Fig. 5. Overview of the algorithm

III. Method

Fig. 5 gives the overview of our proposed method. Due to the different exposure times and nature of distortions introduced in the color and depth images, we compensate for the RS effect in the depth image and color image separately. Firstly, the depth map rectification is carried out using four depth maps for camera transformation estimation. Four depth maps are used instead of the usual two, to allow for more robust camera transformation estimation. Using the transformation and timestamp information, we rectify the depth map. As the intensity images are captured at a different time instances from the depth images, we interpolate the transformation at the intensity image timestamp from the depth map transformations and correct for RS in the intensity image. Simultaneously, the edge fattening effect in the depth map is corrected for, as described in section III-B. Using the information of the camera motion as an initialization for our deblurring algorithm, we deblur the intensity image.

A. Depth Map Rectification

Following Ringaby et al. [7], we rectify the depth map by estimating a transformation between distorted depth images, by parameterizing the camera trajectory to estimate the transformation at a particular row and correct for it by applying the desired inverse transformation.

We found during experimentation that their SLERP based approach (which linearly interpolates between camera trajectories) works well when the camera frame-rate is nearly constant, but declines in performance when frame-rate varies/non-uniform motion during image capture, as the validity of linear interpolation between transformations is violated. Frame-rate can vary as a result of the way a processor schedules the tasks it is supposed to do. Non-uniform motion during image capture can occur due to a camera being mounted on a vibrating platform. To correct for this instead of using a linear interpolation of trajectories, we use a cubic spline which has the desired flexibility to explicitly handle these situations.

We use the recently proposed continuous time trajectory parametrization [1] to model the camera trajectory. We use a cubic B-spline to parametrize time, allowing for camera pose estimation at queried time instances via interpolation. Their formulation allows for efficient interpolation from available camera poses to queried camera poses. The interpolation property is useful for un-distorting the intensity images. The accurate pose estimates from depth images, give accurate pose interpolation at the intensity image timestamps. These interpolated poses are then used to un-distort the intensity images as explained in III-C.

The continuous trajectory of the Kinect is parametrized by camera control poses $\mathtt{T_{w,i}}$ at times $\mathtt{t_i}, \mathtt{i} \in (\{0,\ldots,n\}, \mathtt{T_{w,i}}$ is the transformation from the Kinect frame at time $\mathtt{t_i}$ to the world frame $\mathtt{w}$. We assume that the control poses are uniformly distributed in time, in intervals of size $\Delta\mathtt{t}$. As per the formulation of cubic splines, the value of the spline curve at time $\mathtt{t}$ only depends on four control poses. The pose in the spline trajectory at time $\mathtt{t} \in [\mathtt{t_i}, \mathtt{t_{i+1}})$ is

$$\mathtt{T_{w,s}}(\mathtt{u}(\mathtt{t})) = \mathtt{T_{w,i}} \prod_{\mathtt{j}=1}^{3} \exp(\tilde{\mathtt{B}}_\mathtt{j}(\mathtt{u}(\mathtt{t}))\Phi_{\mathtt{i}+\mathtt{j}-1} \tag{5}$$

$$\tilde{\mathtt{B}}(\mathtt{u}) = \frac{1}{6}\begin{bmatrix} 6 & 0 & 0 & 0 \\ 5 & 3 & -3 & 0 \\ 1 & 3 & 3 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix} \tag{6}$$

where $\mathtt{u}(\mathtt{t}) = \frac{\mathtt{t}-\mathtt{t_i}}{\Delta\mathtt{t}} \in [0,1)$ is the time increment relative to the time instance $\mathtt{t_i}$. The incremental pose from frame at time $\mathtt{t_{q-1}}$ to the frame at time $\mathtt{t_q}$ is encoded in the twist $\Phi_\mathtt{q} = \log(\mathtt{T}^{-1}_{\mathtt{w,q-1}}\mathtt{T_{w,q}})$. $\tilde{\mathtt{B}}(\mathtt{u})$ are the cumulative basis functions for B-splines, written in the De Boor-Cox notation [10].

Approach: The continuous trajectory parametrization allows us to use more than 2 depth maps for trajectory estimation. Given our choice of use cubic-splines, we use 4 successive depth maps for transformation estimation, represented here as $\mathtt{D_i}, \mathtt{i} = 1 \rightarrow 4$. As the depth image is captured by an RS camera, using the timestamp information and the line delay $\mathtt{t_d}$ value, we can estimate, with a reasonable degree of accuracy, when a particular row of the image was captured.

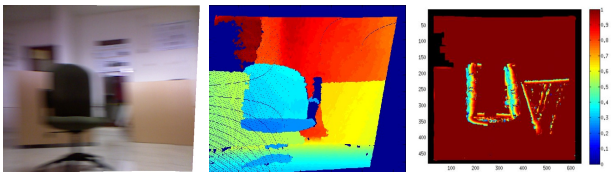
Firstly, a feature matching between the successive depth maps is done. As descriptors the Point Feature Histograms (PFH) [11], which are matched using the FLANN library [12]. The points successfully tracked in all four frames are used for trajectory estimation. Since the camera is moving during frame capture, corresponding points in the depth maps will back-project to different 3D points. Since, we assume that the scene is static, this information can be used to estimate the camera motion. Let a 3D point $\mathtt{o_1}$ be viewed in row $\mathtt{r_1}$ of $\mathtt{D_1}$. Likewise, it's corresponding points are $\mathtt{o_i}$ viewed in row $\mathtt{r_i}$ of $\mathtt{D_i}$ for $\mathtt{i} = 2 \rightarrow 4$. We want to obtain a rectified depth map, where both these points transform to the same 3D point o_0, had the entire image been captured at the same time as the first row of the first image,

$$\bar{\mathtt{o}}^\mathtt{i}_\mathtt{o} = \mathtt{T}(\mathtt{t_i} + \mathtt{r_i}\mathtt{t_d})\bar{\mathtt{o}}_\mathtt{i} \text{ for } i \in 1:4 \tag{7}$$

This leads to the logical cost-function of

$$\mathtt{E}(\theta) = \sum_{\mathtt{j}=1}^{3}\sum_{\mathtt{k}=1}^{\mathtt{K}} ||\mathtt{T}(\mathtt{t_j} + \mathtt{r_{1,k}}\mathtt{t_d})\bar{\mathtt{o}}_{\mathtt{j,k}} - \mathtt{T}(\mathtt{t_{j+1}} + \mathtt{r_{2,k}}\mathtt{t_d})\bar{\mathtt{o}}_{\mathtt{j+1,k}}||^2_2 \tag{8}$$

to be minimized, θ representing the camera trajectory. For one 3D point, between two frames, we thus have to estimate 12 parameters, 6 each for the transformations in eqn. 7, if they are completely un-parametrized. However, thanks to our cubic spline parametrization, we can rewrite these transformations in terms of the control poses, and only solve for the control poses, making our optimization problem well-posed. Additionally, we set the first control pose to the sidentity giving a total of $6 \times (4-1)$, parameters to estimate



(a) (b) (c)

Fig. 6. **Please zoom in.** Fig (a) and (b) show an RS rectified intensity and depth image respectively. (c) shows a probabilistic labelling of the depth map. The red side of the spectrum indicate absence of edge fattening, whereas the blue side indicates edge-fattening.

for each spline. As control poses, we use the transformations taken at the camera timestamps, namely $\mathtt{T}(\mathtt{t_i}), \mathtt{i}:1\rightarrow 4$. The final cost function we optimize is

$$\mathtt{E}(\theta)=\sum_{\mathtt{j}=1}^{3}\sum_{\mathtt{k}=1}^{\mathtt{K}}||\mathtt{o}_{0,\mathtt{k}}^{\mathtt{j}}-\mathtt{o}_{0,\mathtt{k}}^{\mathtt{j}+1}||_2^2+\sum_{\mathtt{j}=1}^{4}||\mathtt{D_y o}_0^{\mathtt{j}}||_2^2 \tag{9}$$

The second term encourages a constant velocity model, for points belonging to the same image. This regularizer shows improvements in camera trajectory estimation in case of varying frame-rates [13], which is true for the Kinect footage.

B. Probabilistic Labelling of the Depth Map

Edge fattening is a combined effect of depth of the object, exposure time and camera motion. To remove the fattened portions of the depth map, we first assign a probability measure as to how likely a pixel was to have been fattened. We model this probability as a mixture of two contributing factors, camera motion and depth. Large camera motion gives rise to increased edge-fattening because a larger portion of the scene is exposed to a particular pixel, increasing the chances of finding a lower depth reading. Likewise, portions of the scene at closer depth have a larger likelihood of being fattened. This is due to larger area covered by the camera trajectory when projected into the image plane for nearby points, as compared to far-off points. The probability function $\mathtt{Pr}(\mathtt{p_i})$ at pixel $\mathtt{p_i}$ given by

$$\mathtt{Pr}(\mathtt{p_i})=\mathtt{Pr_M}(\mathtt{p_i})\mathtt{Pr_D}(\mathtt{p_i}) \tag{10}$$

$$\mathtt{Pr_M}(\mathtt{p_i})=\mathtt{e}^{\frac{-||\mathtt{d_i}-\mu_i||^2}{\sigma_i}} \qquad \mathtt{Pr_D}(\mathtt{p_i})=\frac{\mathtt{d_i}-\mathtt{d_{min}}}{\mathtt{d_{max}}-\mathtt{d_{min}}} \tag{11}$$

$\mathtt{Pr_M}$ and $\mathtt{Pr_D}$ are probabilities due to un-certainty due to motion and depth respectively. μ_i and σ_i are the mean and std. dev. of depth values sampled by the camera trajectory over the interval $[\mathtt{t_{p_i}}-\mathtt{t_e}/2, \mathtt{t_{p_i}}+\mathtt{t_e}/2]$. $\mathtt{d_{min}}$ and $\mathtt{d_{max}}$ are the minimum and maximum depth values in the image row corresponding to pixel $\mathtt{p_i}$. For both probabilities, values close to zero indicate regions more probable to have been fattened. $\mathtt{Pr_M}$ is highest when $\mathtt{d_i}=\mu_i$. Likewise, $\mathtt{Pr_D}$ is highest when $\mathtt{d_i}=\mathtt{d_{min}}$. This term labels pixels with low-depths more likely to have been fattened.

Fig. 6 (a) shows the color image of a scene. Fig. 6 (b) shows the RS rectified depth image captured when the camera was in motion. (c) shows the probabilistic labelling. Probabilities close to 1 represent values that are not influenced by the min-filter effect. The bluer regions in the depth map are eventually thresholded out of the depth map.

C. Rolling Shutter Removal in Intensity Images

Once the RS distortions of the depth map have been rectified, the RS effect in the intensity image can be easily rectified. For a 3D point $\mathtt{o}$, the projected points $\mathtt{p_i}$ and $\mathtt{p_j}$ are given by:

$$\mathtt{p_i}=\pi(\mathtt{KT}(\mathtt{t}(\mathtt{y_i}))\bar{\mathtt{o}}) \text{ and } \mathtt{p_j}=\pi(\mathtt{KT}(\mathtt{t}(\mathtt{y_j}))\bar{\mathtt{o}}) \tag{12}$$

$\pi(.)$ being the perspective projection function. Combining these two equations we get

$$\bar{\mathtt{p}}_\mathtt{j}=\mathtt{C}(\mathtt{t}(\mathtt{y_j}),\mathtt{t}(\mathtt{y_i}))\bar{\mathtt{p}}_\mathtt{i} \qquad \mathtt{C}(\mathtt{t_1},\mathtt{t_2})=\mathtt{KT}(\mathtt{t_1})\mathtt{T}((\mathtt{t_2})^{-1}\mathtt{K}^{-1} \tag{13}$$

Given the warping matrix $\mathtt{C}$, we set the rolling shutter duration t_s to 0. We then compute $\mathtt{C}(\mathtt{t_{desired}},\mathtt{t}(\mathtt{y_i}))$ at each image row $\mathtt{y_i}$ of the current frame $\mathtt{i}$, and apply the warp to that row. Notice that the first term of $\mathtt{C}$ now only depends on the frame time $\mathtt{t_i}$. This operation maps all input frames onto the camera positioned at the start off the frame capture and as a result, removes rolling shutter warping.

D. Image De-blurring

Having corrected the depth and estimated camera pose, we can use this information to get a good initialization for the blurring kernel, by projecting the motion of the camera into the image plane.

$$\min_{\mathtt{k},\mathtt{l}}\frac{1}{2}||\mathtt{k}*\mathtt{l}-\mathtt{b}||_2^2+\lambda_1||\nabla\mathtt{l}||_1+\lambda_2||\mathtt{k}||_2^2 \tag{14}$$

The first term of the equation is a standard least squares term stating that the calculated latent image $\mathtt{l}$ and kernel $\mathtt{k}$ when convolved, give the resulting observed image $\mathtt{b}$. The second term is a sparsity prior on the gradient of the latent image, it enforces sharper edges in the latent image. The third is a penalizer on the kernel, which has been shown to improve the de-blurring performance [6]. The optimization is done using a modified ADMM [14] algorithm.

IV. Results

We first start off by testing the efficacy of each component of our proposed pipeline. Then, we show the benefit our algorithm has on a robotic application, namely camera trajectory estimation.

A. Evaluation of Rolling Shutter Rectification and Motion Blur

Ideally, the performance of our algorithm should be evaluated using RMSE (Root Mean Squared Error) between ground truth and the image we choose to compare. However, in case of deblurring RMSE is not considered an optimal criterion for evaluation [15], due to its excessive sensitivity to lighting conditions and location of artifacts in the image. The paper mentions that the artifacts in a blurred image that are salient can be unrecognisable while having a low RMSE for the image, when compared with the ground-truth. Thus, we evaluate our algorithms performance indirectly via a scene-flow algorithm [16]. We choose scene-flow because it incorporates both visual and depth information in finding dense correspondences between two images, thus making use of both sensing modalities of our sensor. An additional

Input Type	RS	RS Correction	MB	MB Correction	RS+MB	RS+MB Correction
AAE	68.910768	8.535388	13.456721	11.456721	94.437321	23.132389
Std. Dev. AAE	16.507602	0.986454	0.396911	0.596911	24.505380	2.376848
RMS-OF	1.743003	0.155864	0.278669	0.236029	13.174825	7.591635
RMS-SF	2.276523	0.188462	0.400429	0.23294	17.591635	7.678561

TABLE II

RESULT OF ROLLING SHUTTER AND MOTION BLUR ALONG WITH THEIR COMPENSATION BY OUR ALGORITHM ON THE SYNTHETIC DATASET.

benefit of a scene-flow based evaluation is it portable. For our set of experiments, as ground-truth we use two sets of images, horizontally separated by a fixed distance (15cm), taken from a slow moving camera (like in a stereo setup). For synthetic data, we use eqn. 3 to introduce RS+MB distortion in our data. For the real experiments, we make use of the experimental setup shown in Fig. 12 to introduce RS+MB distortion.

For quantitative evaluation, we use the standard scene-flow metrics. Namely, RMS-OF (Root Mean Square Error in optical flow computation), RMS-SF (RMS for Scene Flow) and Average Angular Error (AAE), which is the RMS of the difference in the orientations of optical flow vectors.

(a) (b) (c) (d)

Fig. 7. **Please zoom in.** Results of the proposed algorithm on a synthetic sequence. In each figure the florescent green box indicates the position of the box in the image if the exposure time and line delay were both infinitesimal. (a) and (b) show the RS+MB distorted images. (c) and (d) show the results after correction.

B. Results on Synthetic Data

In our synthetic data experiment, we took a textured 3D object and applied RS+MB distortion by applying a discretized form of eqn. 3. We did two sets of experiments. The 1^{st} just by introducing RS in our data, by increasing the line delay t_d. The 2^{nd} by introducing both, by increasing line delay t_d and row exposure time t_e. The values are set to those mentioned in the last paragraph of II-C. Fig. 7 shows the results of our experiment. (a) and (b) show RS+MB distorted images. (c) and (d) show the result of our rectification algorithm. The florescent box in all four frames shows the ground truth corresponding to the case of a camera having a GS, with infinitesimal exposure time.Figures (a), (b) show the RS+MB distorted input to our algorithm. (c) and (d) show the result of rectification. As can be seen in the figure, our algorithm is able to rectify for RS effectively, while simultaneously reducing blur. Table II shows the quantitative results of the experiment. In the case of only RS distortion, our algorithm is able to reduce the error of nearly 90% on synthetic data, for all 3 metrics. On introducing MB in addition to RS, our algorithm manages to reduce error by 60%. The performance drop is an indicator of the severely detrimental effect MB has.

Fig. 8. Experimental setup for camera trajectory estimation. The Kinect is placed on an oscillating crank-shaft mechanism, which introduces high-frequency image jitter into the Kinect data. The mechanism is placed on an autonomous wheel chair.

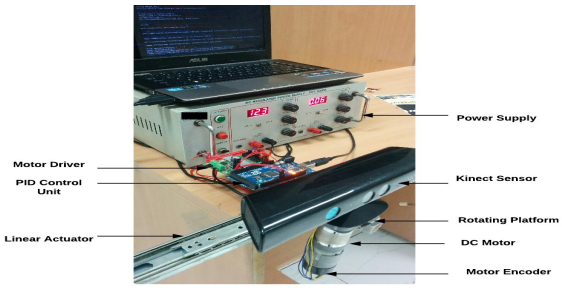
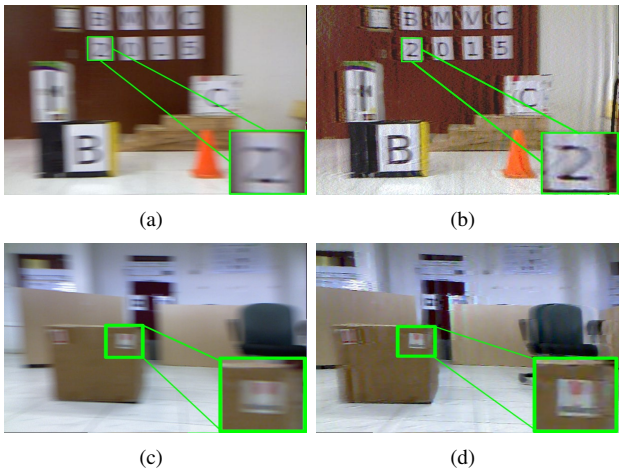


Fig. 9. Labelled experimental set-up capable of precise calibrated linear and rotational motion.

C. Results on Real Data

To evaluate our algorithm on real data, we developed an experimental setup (shown in Fig. 9) capable of executing calibrated motion. The setup consists of a Kinect mounted on a motor-powered rotating platform. This platform is attached to a linear actuator. Both the rotating platform and the linear actuator are controlled by a PID controller. To introduce real RS+MB distortion in our captured data, we take images using this setup at the separation equal to the ground truth while the camera is rotating and horizontally translating. The motors were timed, such that the camera faces roughly the same scene as the two ground-truth images. We use rotation and translation, instead of just rotation, because it is known shown in [17] to be a more pronounced source of MB than translation.

Fig. 11 shows the results of our algorithm on a real dataset taken at various rotational speeds. The dataset consists of three primary objects (a box, a chair and a desk) placed at different depths. The odd rows in the figure, show the input taken by our algorithm. The even rows, show the RS+MB rectified output. In the depth maps black lines are marked roughly parallel to the box edge in the scene, to more easily visualize the RS correction. After rectification they are closer to the vertical than initially. This is more



(a) (b) (c) (d)

Fig. 10. Qualitative Evaluation of Deblurring Algorithm (a) and (c) show images affected by motion blur. (b) and (d) the deblurring algorithm. The green boxes provide a zoomed in view of the same area before and after deblurring for visual comparison.

pronounced in the case of the camera undergoing 40rpm rotation. The deblurring performance of our algorithm is noticeable for 20rpm, albeit with some ringing artifacts. For 40rpm the effect is far less pronounced. This is due to the low signal-to-noise ration in the image, making an already ill-posed problem even more difficult. This however is an extreme case and is unlikely to occur within a robotic setting with SLRS cameras that are primarily indoor cameras. Future work can possibly involve incorporating prior knowledge of the scene and learning the location specific blur kernels for better deblurring.

Fig. 10 shows a zoomed in view of the dataset before and after de-blurring. As can be seen in the zoomed in green boxes, our proposed method is able to restore the high-frequency details of the scene-text, which can be useful for robotic navigation.

Table III summarizes the results of our experiments on the real-dataset. For 20 and 30 rpm, our algorithm is able to reduce RMS-SF and RMS-OF by around 50%. For the extreme case of 40 rpm, the performance of our algorithm degrades gracefully, while still maintaining a 20% error reduction across the different error metrics.

D. Camera Trajectory Estimation

In this section we show the effect RS+MB distortion and it's correction have on the camera trajectory estimation. To introduce RS+MB blur into the Kinect footage we use the setup shown in Fig. 9. It consists of a Kinect mounted on an oscillating mechanism. The mechanism is powered by a motor oscillating at 450rpm, sufficiently high to introduce RS+Mb distortion. The purpose of this experiment was to show, how our pipeline fairs in the realistic case of a Kinect mounting on a vibrating moving platform. For ground-truth we use the odometry taken from the wheel encoders of the wheel-chair. We ran this fRexperiment for two sets of trajectories, one a straight line motion, roughly 10 meters long. The other, a loop of length nearly 32 meters around our lab.

Dataset	Method	RMSE Pose	Med. Pose	RMSE Ori.	Med. Ori.
Straight	ICP	175.939	152.038	3.754	2.891
	SLERP	97.526	79.384	2.911	2.732
	Ours	72.447	61.953	1.761	1.675
Loop	ICP	268.346	202.363	11.622	10.295
	SLERP	227.344	223.402	7.078	6.991
	Ours	155.772	138.593	3.387	2.734

TABLE IV
POSE AND ORIENTATION ERRORS FOR CAMERA TRAJECTORY ESTIMATION IN THE TWO SEQUENCES (IN MM AND DEGREES)

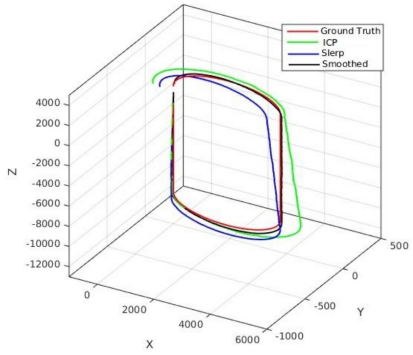


Fig. 12. Trajectory Estimation Result for the loop sequence. The curve in red, corresponding to the ground-truth. The curve in green corresponding to the camera trajectory computed without RS correction. The curve in black, corresponding to the smoothed trajectory, computed after correction. Measurements are in mm.

Fig. 12 shows the camera trajectory plots on the loop sequence, for different algorithms. The units in the figure are in mm. In red, is plotted the ground-truth. In green is shown the trajectory computed using only ICP, it starts to deviate from the ground-truth significantly at the curved section of the trajectory. The trajectory in blue, shows the result of SLERP interpolation, like the kind proposed by [7], which is much better than the ICP based trajectory, but still suffers from drift. Our proposed approach, in black, on the other hand is able to deal with the RS+MB distortion effectively, yielding a trajectory especially close to the ground-truth. The quantitative results of this experiment are given in table IV. The pose error is measured in millimetres and orientation in degrees.

For the straight line sequence, the cubic spline interpolation give is a 28% improvement in terms of pose error on the linear spline (SLERP), while beating conventional ICP by 50%. The orientation is low for all three methods which is expected, given that the sequence is a straight line. For the loop sequence similarly our algorithm, is noticeably better, having substantially lower orientation error than the other two methods. This is due to the flexibility of the cubic spline formulation.

V. CONCLUSIONS

In this paper, we presented an algorithm for RS and MB distortion rectification for SLRS cameras capable of dealing with large distortions. In doing so we introduced a novel mathematical model to describe the blurring experienced by a depth map. This model explains the edge-fattening effect seen in depth maps taken from moving cameras as a result of the min-filter effect. To test our algorithm we created

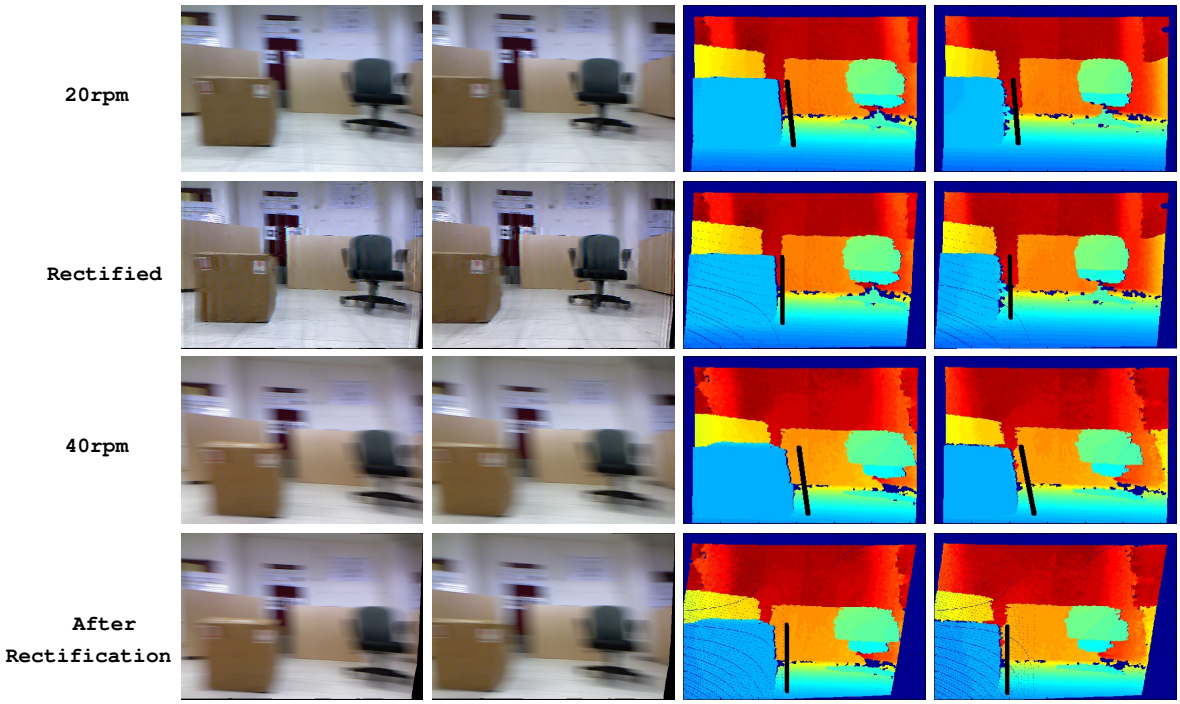


Fig. 11. Results of the proposed algorithm on real data. The first row contains input data. The second row contains rectified output data. The dataset consists a box, a chair and a desk at different depths. As can be seen in row 2, the proposed algorithm is able to rectify the RS and MB effects giving more coherent shapes in the depth map. The edges of the box and desk are nearly vertical.

Input Type	20 rpm	Rectified	30 rpm	Rectified	40rpm	Rectified
AAE	9.515508	3.709758	34.17264	23.515508	129.291094	109.848781
Std. Dev. AAE	4.382542	3.935804	23.81221	14.17264	98.789264	79.194166
RMS-OF	11.910281	8.250955	48.152351	23.81221	89.696	78.107836
RMS-SF	22.298120	11.211356	47.278201	28.152351	181.983099	168.483439

TABLE III

RESULT OF ROLLING SHUTTER AND MOTION BLUR ALONG WITH THEIR COMPENSATION BY OUR ALGORITHM ON THE REAL DATASET

a dataset of RS+MB distorted images of varying severity, and showed that depth rectification achieved more than 60% rectification. In addition, we showed the effect on two robotic applications, namely the evaluation of scene-flow, as well as camera trajectory estimation. For scene-flow, the images after rectification by our algorithm gave significantly less error in all standard error metrics. The camera trajectory outputted by our algorithm is significantly better than the one done using standard ICP or SLERP interpolation. For future work, we intend to improve our overall pipeline by learning betters priors for deblurring. We also intend to incorporate object information to improve the probabilistic labelling of objects.

REFERENCES

[1] Alonso Patron-Perez, Steven Lovegrove, and Gabe Sibley. "A Spline-Based Trajectory Representation for Sensor Fusion and Rolling Shutter Cameras". In: *International Journal of Computer Vision* (2015), pp. 1–12.

[2] Stergios I Roumeliotis and Chao Guo. *EFFICIENT VISION-AIDED INERTIAL NAVIGATION USING A ROLLING-SHUTTER CAMERA WITH INACCURATE TIMESTAMPS*. US Patent 20,150,369,609. 2015.

[3] Alexandre Karpenko et al. "Digital video stabilization and rolling shutter correction using gyroscopes". In: *CSTR* 1 (2011), p. 2.

[4] Simon Baker et al. "Removing rolling shutter wobble". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE. 2010, pp. 2392–2399.

[5] Johan Hedborg et al. "Rolling shutter bundle adjustment". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 1434–1441.

[6] Shuochen Su and Wolfgang Heidrich. "Rolling Shutter Motion Deblurring". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1529–1537.

[7] Erik Ringaby and Per-Erik Forssén. "Scan rectification for structured light range sensors with rolling shutters". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1575–1582.

[8] Ken Shoemake. "Animating rotation with quaternion curves". In: *ACM SIGGRAPH computer graphics*. Vol. 19. 3. ACM. 1985, pp. 245–254.

[9] D Alex Butler et al. "Shake'n'sense: reducing interference for overlapping structured light depth cameras". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2012, pp. 1933–1936.

[10] Carl De Boor. "On calculating with B-splines". In: *Journal of Approximation Theory* 6.1 (1972), pp. 50–62.

[11] Radu Bogdan Rusu et al. "Towards 3D point cloud based object maps for household environments". In: *Robotics and Autonomous Systems* 56.11 (2008), pp. 927–941.

[12] Marius Muja and David G Lowe. "Flann, fast library for approximate nearest neighbors". In: *International Conference on Computer Vision Theory and Applications (VISAPP'09)*. 2009.

[13] Matthias Grundmann et al. "Calibration-free rolling shutter removal". In: *Computational Photography (ICCP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 1–8.

[14] Stephen Boyd et al. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Foundations and Trends® in Machine Learning* 3.1 (2011), pp. 1–122.

[15] Martin Čadík et al. "New measurements reveal weaknesses of image quality metrics in evaluating graphics artifacts". In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), p. 147.

[16] Julian Quiroga et al. "Dense semi-rigid scene flow estimation from rgbd images". In: *Computer Vision–ECCV 2014*. Springer, 2014, pp. 567–582.

[17] Oliver Whyte et al. "Non-uniform deblurring for shaken images". In: *International journal of computer vision* 98.2 (2012), pp. 168–186.