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

Empirical Modelling of Rolling Shutter Effect

Liam O'Sullivan[1] and Peter Corke[1]

***Abstract*— We propose and evaluate a novel methodology to identify the rolling shutter parameters of a real camera. We also present a model for the geometric distortion introduced when a moving camera with a rolling shutter views a scene. Unlike previous work this model allows for arbitrary camera motion, including accelerations, is exact rather than a linearization and allows for arbitrary camera projection models, for example fisheye or panoramic. We show the significance of the errors introduced by a rolling shutter for typical robot vision problems such as structure from motion, visual odometry and pose estimation.**

I. INTRODUCTION

The mass proliferation of consumer cameras has given roboticists access to low cost and powerful imaging technology. These cameras, found in a variety of devices such as mobile phones or Microsoft's Kinect, have enabled powerful computer vision algorithms such as point feature tracking and optical flow estimation techniques to be run on-board a variety of small and cheap robotic platforms. Increasinging capable and low-cost computing power and imaging sensors has allowed the complex problem of Structure from Motion (SfM) to be solved in real time through various Simultaneous Localisation and Mapping (SLAM) and Visual Odometry (VO) approaches e.g. Parallel Tracking and Mapping (PTAM) [1], Dense Tracking and Mapping (DTAM) [2] and MonoSLAM [3].

Rolling shutter is a familiar annoyance when we take pictures with our phone cameras. Artefacts such as wobble, skew, smear, partial exposure and aliasing are introduced into the image when there is relative motion between camera and the scene. What is less well appreciated is that the geometric distortions introduced by this effect are quite significant, easily many pixels, and this has real implications for vision-based robotic algorithms such as SfM, VO and pose estimation. Figure 1 shows what these distortions can look like — the geometric shape of a checkerboard pattern has been distorted due to camera motion.

The problem is caused by the way in which CMOS imaging sensors operate, which is fundamentally different to the older CCD (charge coupled device) sensors. CMOS sensors have overtaken CCD technology because they support a very high pixel readout rate which is essential for high resolution video, and they can be fabricated using a standard silicon production line which reduces costs. The low cost, high resolution and high pixel rate have enabled CMOS sensors to achieve a massive take-up in consumer devices such as phones, tablets and laptop computers.

[1]L. O'Sullivan and P. Corke are with the CyPhy Lab, School of Electrical Engineering and Computer Science, Queensland University of Technology, Australia. {lt.osullivan, peter.corke}@qut.edu.au

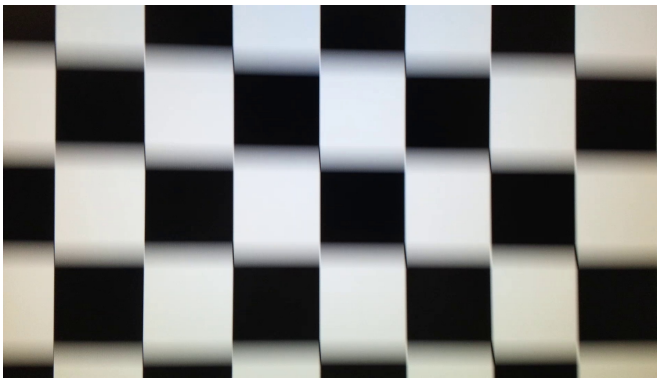

Fig. 1. Rolling shutter geometric distortion of a checkerboard.

In a CCD sensor all pixels are exposed over the same time period and then the integrated intensity, the number of photo-electrons in the charge well at the photo site, of all pixels are transferred instantaneously and in parallel to the vertical transport registers. A snapshot of the intensity is made. A CMOS camera utilises an electronic rolling shutter (ERS) — the pixels are read sequentially, typically row-wise, from the top left corner. The value that is read from the pixel is the integrated charge over the period since it was last read. The frame time is the time required to read all the rows, so the last pixel in the frame is exposed over a period that is almost the same as the first pixel of the next frame. The rolling shutter problem is caused by each pixel having a unique exposure period and if objects viewed by the camera in the world environment or the camera itself move before the entire frame can be exposed then artefacts will be introduced into the image. For CMOS cameras with an electronic shutter, a parallel process address the pixels m lines ahead of where they are being read out and resets the pixel, so the exposure time is proportional to m.

The vision sensors we use in robotics are based on consumer technology, for example Kinect or webcams. Since the effect of an ERS is to introduce significant geometric distortion, see Figure 1, any robot vision algorithm that uses image geometry to infer characteristics of the world will be affected by this problem and this includes pose estimation, VO and SfM.

In this paper we make two contributions. First we present the results of carefully designed experiments to measure the effect of ERS and to estimate the rolling shutter scan rate of a consumer camera. This parameter, commonly assumed not measured, is critical to compensate the effect of rolling shutter. Second, we introduce a new algorithm which predicts

978-1-4799-3684-7/14/$31.00 ©2014 IEEE

exactly the geometric distortion introduced by a rolling shutter for general camera and/or object motion and for any imaging geometry (perspective, fisheye etc). The remainder of the paper is structured as follows, the relevant rolling shutter background and theory will be presented in Section II-A and II-B. Section II-C will explore the the effect of rolling shutter motion estimation in simulation using the presented theory. Section III will detail our iterative based rolling shutter prediction method and finally experimental verification of the rolling shutter effect will be demonstrated in Section IV.

II. BACKGROUND

A. Related Work

Early research into the geometric modelling of a camera with rolling shutter was provided by [4]. Geyer et al. showed the analytical perspective equation for fronto-parallel motion and demonstrated that the equations become quadratic for general camera motion and non-linear without linearised camera motion [4]. Numerous researchers have worked on the removal of the rolling shutter effect in recorded data. Initial rolling shutter removal approaches began by rectifying the image plane to account for rolling shutter. In [5], [6] the deformation of the image was modelled as a globally constant translation motion across the image. This was extended by Liang who used Bezier curves to give each row different camera motion in [7]. Baker [8] used a similar technique as in [7] but used L1 regularisation instead of Bezier interpolation to allow for more generalised camera motions. Forssn and Ringaby [9] used spherical linear interpolation (SLERP) to interpolate the 3D rotational poses of the camera for each scanline and applied this method in [10] to compute the bundle adjustment for rolling shutter cameras.

Other works include Kim et al. [12] and Grundmann et al. [13] who partition the image into blocks and produce a mixture model of homographies between image frames to remove rolling shutter distortions. Hardware based solutions have also been developed where Hanning et al. [14] used inertial measurements from on-board accelerometers and gyroscopes to stabillise mobile phone video capture. Their rectification uses the 3D rotational model developed in [10] but uses inertial measurements instead. Inertial measurements are also used in the Visual Inertial Odometry (VIO) system [11] where mobile phone camera motion is estimated in the presence of rolling shutter. The downside to using hardware based solutions is the need for sychronisation between the inertial measurements and video capture and relies on a static environment to model the motion of the camera. Much of the research in the vision community is concerned particularly with rotational motion which is important for a hand held camera but for robotics translational camera motion is perhaps more important.

Finally almost all of the above methods require some calibration to model the motion of the camera. A calibration routine was devised in [4] and improved in [15] to measure the time taken to capture an image frame for a camera sensor. A list of capture times for some camera models is available [2] which is used in the VirtualDub Deshaker plugin.

B. Rolling Shutter Projection Equation

The notation used in this paper will be described by the following camera motion example. For a global shutter camera, the perspective projection equation is

$$\begin{pmatrix}\tilde{u}\\ \tilde{v}\\ \tilde{w}\end{pmatrix} = \begin{pmatrix}\lambda & 0 & u_0\\ 0 & \lambda & v_0\\ 0 & 0 & 1\end{pmatrix} \xi_t \begin{pmatrix}X\\ Y\\ Z\\ 1\end{pmatrix} = \boldsymbol{K}\xi_t\boldsymbol{P} \tag{1}$$

where $\boldsymbol{p} = [u;v]$, $u = \tilde{u}/\tilde{w}$, $v = \tilde{v}/\tilde{w}$ is the global shutter point coordinate in the image frame in pixels, λ is the normalised camera focal length in pixels, u_0 and v_0 are the principal point coordinates of the camera, $\boldsymbol{K}$ is the intrinsic camera parameter matrix, $\xi_t = [R(t)T(t)]$ is the pose of the camera at time t in $SE(3)$, $R(t)$ is the camera rotation in $SO(3)$, $T(t)$ is the camera translational in $\mathbb{R}^3$ and $\boldsymbol{P} = (X,Y,Z)$ is the world coordinate of a static point in space. A linear approximation of $[R(t)\ T(t)]$ can formed at $t = 0$ to give

$$[R(t)\ T(t)] \approx [(I_3 + t\omega_\times)R(0)T(0) + \nu t] \tag{2}$$

where $R(0)$ and $T(0)$ is the initial rotation and translation of the camera, $\omega_\times$ is the skew symmetric matrix where $\omega = [\omega_x, \omega_y, \omega_z]$ is the instantaneous rotational velocity of the camera and $\nu = [\nu_x, \nu_y, \nu_z]$ is the instantaneous translational velocity of the camera.

Consider the constrained camera motion case where $\omega = [0,0,0]$ and $\nu = [0,\nu_y,0]$ i.e. there is only translational movement in the vertical axis of the image frame. The linearisation in Equation 2 becomes exact and if $R(0) = I_3$ and $T(0) = 0$ then $R(t) = I_3$ and $T(t) = \nu_y t$.

For a rolling shutter camera, Geyer et al. presented a similar derivation and introduced the concept of the *rolling-shutter constraint* [4] which showed that the vertical coordinate v is

$$v = tr - v_s \tag{3}$$

where v_s is the index of the first exposed scanline of image sensor (typically 0) and r is the scan rate of the rolling shutter image sensor in lines/s. Ideally r would be equal to the frames per second (fps) capture rate of the camera multiplied by the number of scan lines in the image frame. In practise this may not be the case since other latencies in the capture process should be included.

Equation 3 can be substituted into Equations 1 and 2 to solve for the rolling shutter point coordinate $\boldsymbol{p}_r = [u_r;v_r]$. The expression for t in this case will be

$$t = \frac{Y\lambda}{Zr - \lambda\nu_y} + \frac{Zv_0}{Zr - \lambda\nu_y}$$

[2] Available at: http://www.guthspot.se/video/deshaker.htm

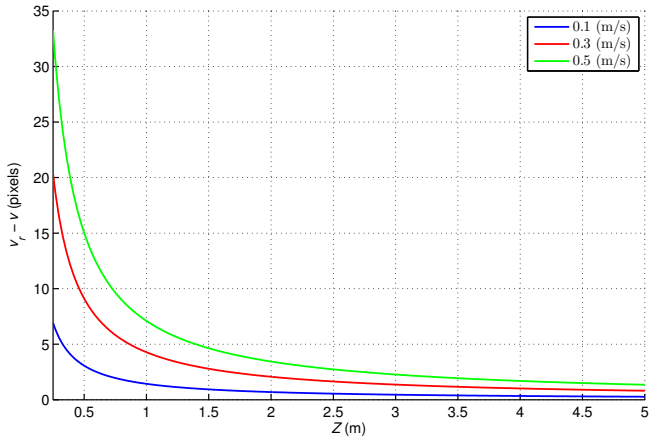

Fig. 2. Vertical pixel error introduced by rolling shutter as world point range and camera velocity vary.

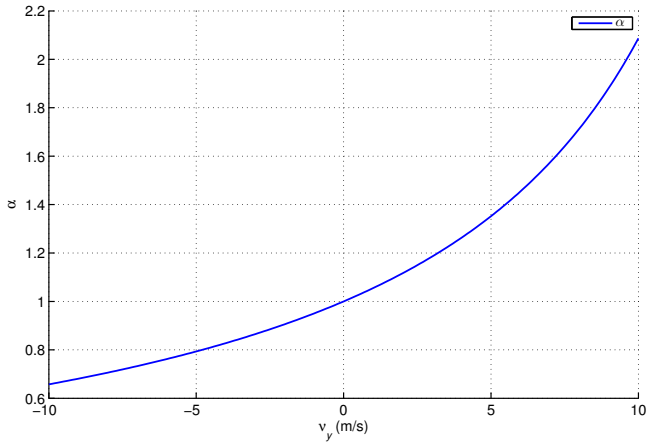

Fig. 3. α values for different v_y velocities ($Z = 0.5$ m).

and the rolling shutter point coordinate $\boldsymbol{p}_r$ will be

$$\boldsymbol{p}_r = \begin{pmatrix} u_r \\ v_r \end{pmatrix} = \begin{pmatrix} \frac{X\lambda}{Z} + u_0 \\ \frac{Y\lambda}{Z} + v_0 + \frac{\lambda v_y (Y\lambda + Z v_0)}{Z(Zr - \lambda v_y)} \end{pmatrix} = \begin{pmatrix} u \\ \frac{Zr}{(Zr - \lambda v_y)} v \end{pmatrix} = \begin{pmatrix} u \\ \alpha v \end{pmatrix} \quad (4)$$

That is $\boldsymbol{p}_r$ is equal to the global shutter point projection $\boldsymbol{p} = [u; v]$ and v_r is multiplied by a scaling factor α due to the effect of the rolling shutter. The magnitude of α will be dependent on Z, r, λ and v_y e.g. as the scan rate $r \to \infty$ then $\alpha \to 1$ and Equation 4 would model a global shutter camera as in Equation 1. This limit relationship can be seen in Figure 2 where the difference between v_r and v for varying depth Z and v_y values is shown. Note the other parameters in Figure 2 are kept constant for each Z and v_y value where $Y = 0.05$ m, $\lambda = 800$ pixels, and $r = 30\,\text{fps} \times 1024\,\text{lines} = 30,720$ lines/s. Thus as the depth of the point increases, the difference decreases (or $\alpha \to 1$) and v_r approaches v. Also the difference increases in magnitude as v_y increases since it begins to dominate the denominator in Equation 4.

The scaling factor α is a function of v_y and this relationship can be seen in Figure 3 where Z is kept constant at 0.5 m. Positive v_y motion corresponds to a world point moving down the image frame and $\alpha > 1$ i.e. the v_r coordinate will lead the v coordinate and appear further down the image frame. Conversely, $\alpha < 1$ when negative v_y motion is applied and the v_r coordinate will appear above v in the image frame. Finally if the world point is static and $v_y = 0$, then $\alpha = 1$ and the model of a global shutter camera is obtained once again. We see that v_r in Equation 4 is singular when

$$Zr = \lambda v_y, \ \therefore v_y = \frac{Zr}{\lambda}$$

and this corresponds to the velocity at which the image is moving down the image plane at exactly the same rate as the lines are being read. In that case every line in the resulting image would be the same. For even higher velocities $\alpha < 0$ which would theoretically mean that the image would be inverted since it is moving faster than the line readout rate.

C. Rolling Shutter Motion Estimation

Even simple camera motion will result in inaccurate pose and motion estimation outputs when a rolling shutter is present. We illustrate this with a simulation to compare the effect of global and rolling shutter pose estimation on the Efficient Perspective-n-Point (EPnP) algorithm [16] where the object pose is represented as $[X, Y, Z, R, P, Y]^T$. The parameters used in the simulation were $\lambda = 800$ pixels, principal point $[u_0; v_0] = [320; 240]$ pixels, $v_y = 2$ m/s, $r = 14,400$ lines/s and $n = 15$ randomly generated world points representing the object whose pose is to be determined. The global and rolling shutter projections are shown in Figure 4.

The EPnP pose solution for the global shutter was $[0\text{ m}; 0\text{ m}; 0\text{ m}; 0^\circ; 0^\circ; 0^\circ]$ whereas for the rolling shutter it was $[-0.018\text{ m}; 0.051\text{ m}; -0.490\text{ m}; 0.34^\circ; 0.37^\circ; 0.08^\circ]$. The rolling shutter has caused the EPnP algorithm to return a significantly inaccurate pose where all degrees of freedom of the camera contain error, and the z-axis error is 0.5 m. The root cause is that the image plane projections cannot be adequately explained by any real camera-object relative pose. Similar effects are observed for fundamental and essential matrix estimation. If incremental motion estimates are integrated over time this problem will be compounded.

III. ROLLING SHUTTER ALGORITHM

In this section we propose a new algorithm to predict the effect of ERS for general camera motion and *any* camera projection model, that is, we lift the constraint on constant velocity in Equation 2 and fronto-parallel motion and perspective projection [4]. Conceptually we consider the acquisition of a single frame as being achieved by a *sequence of cameras* $C_k, k = 1..N$ with pose ξ_k which are situated in space at the pose where the original moving camera would be when line k is being read. Line k is read at time $t = k/r + t_i$

therefore $\xi_k = [R(t)|T(t)]$ which can be determined from the, assumed known, general motion model of the camera.

Consider now that this sequence of cameras is viewing a single static world point. Figure 5 shows the trajectories of the projected point, the *projection line*, as k varies — its projection into each of the N cameras for different velocities. For the case of zero camera velocity, the red line, we see that the projection is constant since all N cameras have the same pose. However for finite camera velocity the cameras have different poses along a short trajectory in $SE(3)$ and we see that the projected point moves as we progress along the camera sequence. The steepness of the line increases with camera velocity. Since the pose of camera k corresponds to the time at which line k is being read, only points projected to line k can be observed by the camera. The essence of the algorithm is to project the world point $\boldsymbol{P}$ to the image plane (u_k, v_k) of every camera and choose k such that $v_k = k$. Graphically this corresponds to the intersection of the projection line with the unit slope diagonal (blue line) in Figure 5. For the parameters used to generate this figure

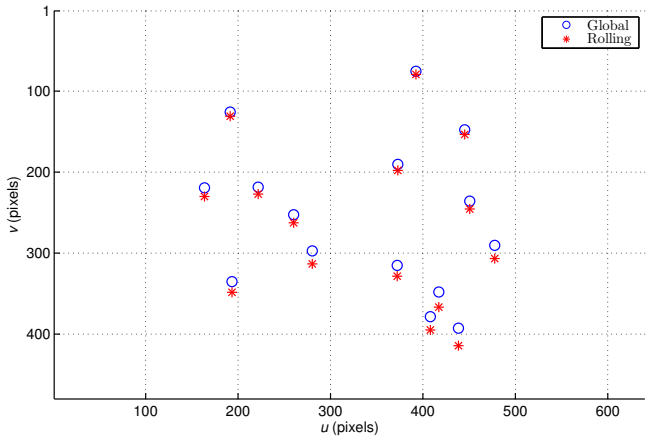

Fig. 4. Global and rolling shutter image plane projections.

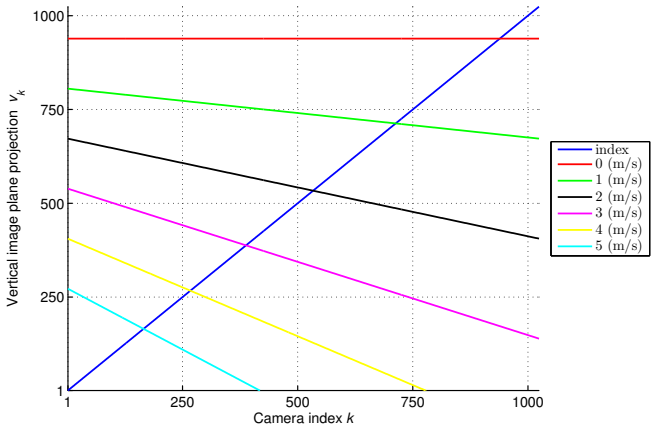

Fig. 5. Camera index k versus world point projected vertical coordinate (projection line) for different v_y velocities. Parameters: $\boldsymbol{P} = (0.2; 0.1; 2)$ m, $\lambda = 800$ pixels, $r = 30,720$ lines/s.

we see that as camera velocity increases k decreases i.e. the rolling shutter causes the point to move higher in the image.

In practice, the condition $v_k = k$ will rarely be exactly met so we look for the transition from $v_{k_1} < k_1$ to $v_{k_2} > k_2$ and consider k as the mean of k_1 and k_2. Graphically these are the discrete points on the projection lines above and below the diagonal line. Our algorithm is summarised graphically for one world point and multiple velocities in Figure 5 and presented in full for n world points in Algorithm 1.

The cost of this algorithm is N times the cost of projecting world points to the image plane. Savings could be achieved by first projecting the points assuming a global shutter and then searching adjacent rows $v_k - \Delta \leq k \leq v_k + \Delta$. Alternatively we could perform a Newton-style search to find the intersection of the projection line and the unit slope diagonal.

Significantly the algorithm requires only the projection function $\pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$ which can be easily written for any type of camera. This allows for exact modelling of rolling shutter effect with non-perspective cameras such as fisheye and panoramic cameras. In a real camera the lines are not read instantaneously, they are read pixel by pixel. The algorithm we have presented could be extended to consider a camera at every pixel position.

IV. MEASURING ROLLING SHUTTER EFFECT

We conducted carefully controlled experiments to measure the effect of rolling shutter in a real camera and compare it

Algorithm 1: Rolling shutter coordinate calculation

Input: N, $\boldsymbol{P}$, n, $\boldsymbol{K}$, t_i, t_f, ξ_t, ν, ω
Output: $\boldsymbol{p}_r^n$

```
// Compute the time step for each pose
δ_t = (t_f − t_i)/N ;
// Pose counter
k = 1 ;
// Compute the projection at each pose
for i = t_i + δ_t : δ_t : t_f − δ_t do
    // Update the camera pose at k
    ξ_k = T(ξ_{k−1}, ν, ω, δ_t) ;
    // Calculate the point projections
    p_k^n(p, n, k) = π(K, ξ_k, P) ;
    // Increment pose counter
    k++ ;
end
// Row index array
rows = 1 : 1 : N ;
// Find rolling shutter coordinates
for j = 1 : 1 : n do
    // Compute the row index diff
    v_sub = rows − p_k^n(v, j, :) ;
    // Find the min value index
    k_min = min(abs(v_sub)) ;
    // Set rolling shutter coordinate
    p_r^n(p, j) = p_k^n(:, j, k_min) ;
end
```

to the simulated model results, providing insight into the true line-scan rate r.

A. Camera Calibration

The camera used for this experiment was the rear camera of Samsung Galaxy S3 mobile phone which contains a Sony IMX145 8.4 MP BSI CMOS image sensor. Video data was recorded at 1920×1080 (1080p) resolution at 30 frames per second. The camera was calibrated using the Automatic Multi-Camera Calibration Toolbox [3] which is a modified and automated version of Bouget's Camera Calibration Toolbox for MATLAB. The following calibration data was obtained at 1080p resolution

- Focal length $\lambda = 1565$ pixels.
- Principal point $[u_0; v_0] = [953; 512]$ pixels.
- Radial distortion $[k_1, k_2, k_3] = [0.109, -0.177, 0]$.
- Tangential distortion $[p_1, p_2] = [-0.0021, -0.001]$.

The scan rate was calculated to be $r = 1080 \times 30 = 32,400$ lines/s and we have assumed that any extra capturing latencies are negligible for the Samsung Galaxy S3.

B. Experimental Setup

The Samsung Galaxy S3 was mounted in a RepRap 3D printer (refer to Figure 6) where it could see four square markers placed 0.14 m from the camera on the RepRap bed. The camera was oriented in a way such that the square markers could move in the vertical direction of the camera view. The 3D printer was used such that accurate and repeatable trajectories could be created at varying speeds through the G-code numerical control programming language. The G-code commands caused the four squares markers to move with a triangular position profile at a specified velocity and this visual movement was recorded by the phone. The motion was constrained to be fronto-parallel and along the vertical axis of the image frame as seen by the phone camera. Video sequences were recorded for vertical velocities v_y of $500, 1000, 2000, 3000, 4000, 5000, 6000, 7000$ mm/min.

The rolling shutter effect was measured in the video recordings by calculating the vertical centroid separation distance Δv between a pair of square markers which can be seen on the RepRap bed in Figure 6. The captured frames were rectified using the lens distortion parameters found by the camera calibration process. For every frame the centroid of each square was estimated by performing binarization and blob detection. We use data only from the left marker pair.

C. Simulation Setup

The simulator mirrored the experimental setup and outputs Δv, the vertical distance between two points which can be derived from Equation 4

$$\begin{aligned}\Delta v &= v_2 \frac{Zr}{(Zr - \lambda v_y)} - v_1 \frac{Zr}{(Zr - \lambda v_y)} \\ &= \frac{Zr}{(Zr - \lambda v_y)}(v_2 - v_1) \\ &= \alpha(v_2 - v_1) \end{aligned} \quad (5)$$

[3] Available at: https://code.google.com/p/amcctoolbox/

TABLE I

Δv ERROR VALUES FOR EACH v_y VELOCITY.

v_y (mm/min)	Δv error (pixels)	
	$r = 32,400$ lines/s	$r = 40,905$ lines/s
500	+0.13	-0.03
1000	+0.3	-0.02
2000	+0.7	+0.07
3000	+1.0	-0.028
4000	+1.3	-0.013
5000	+1.8	+0.11
6000	+2.2	+0.05
7000	+2.7	+0.2

Thus Δv will be equal to $(v_2 - v_1)$ as seen with a global shutter multiplied by a scaling factor α. As discussed earlier, α is a function of v_y, r, λ and Z.

D. Experimental Results

Figures 7-9 show the measured and predicted Δv for v_y values of 1000, 3000 and 7000 mm/min. We show the global shutter (red), ERS simulated (blue) and measured (black). It is clear that the rolling shutter effect increases as v_y increases. The global shutter Δv distance is constant (since $\alpha = 1$) and both the simulated and rolling shutter Δv distances vary around the global shutter Δv distance. The simulated and experimental results for all speeds tested are summarized in Table I.

We see from the middle column in Table I, that in every case, the predicted effect of ERS is greater than measured. We hypothesise that this is due to an incorrect assumption of r which we estimated naively based on the number of lines and the camera frame rate. If we take the measured values of the ERS effect we can estimate the effective value of r from

$$r = \frac{\Delta v \lambda v_y}{Z(\Delta v + v_1 - v_2)} \quad (6)$$

based on our knowledge of the other parameters. Using this Equation 6 we calculated the experimental mean value of r

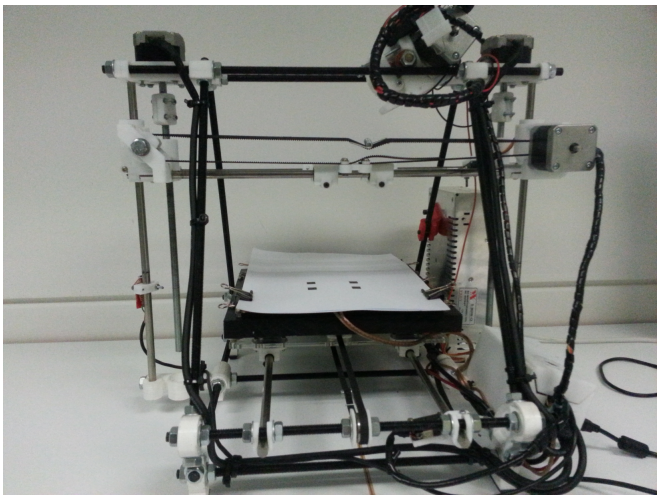

Fig. 6. RepRap 3D printer used in the controlled experiments.

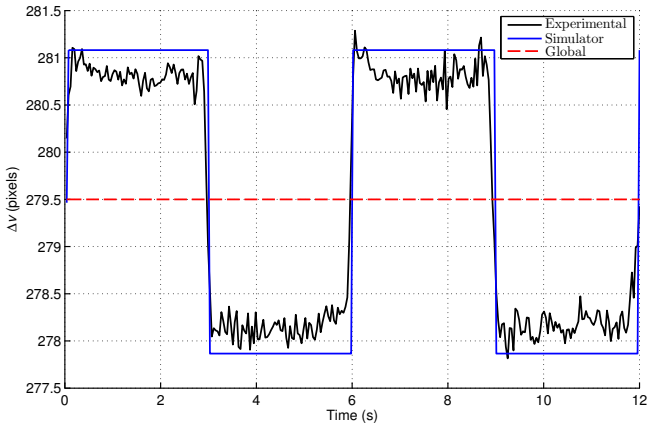

Fig. 7. Δv distances at 1000 mm/min.

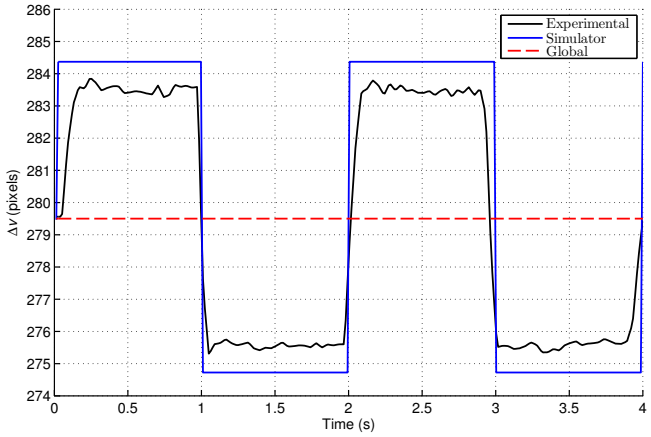

Fig. 8. Δv distances at 3000 mm/min.

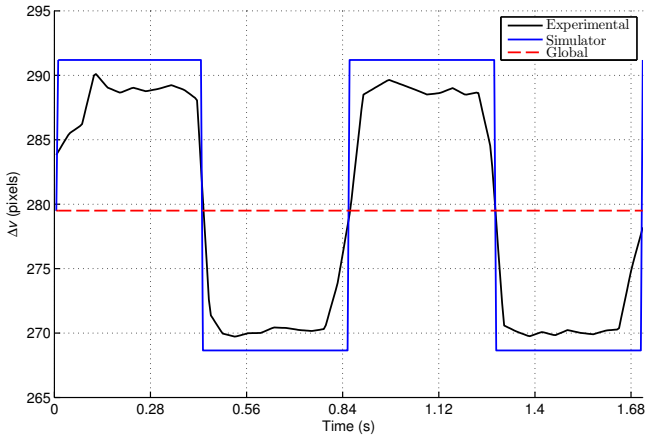

Fig. 9. Δv distances at 7000 mm/min.

to be 40,905 lines/s. The last column in Table I shows the Δv error with this empirical value of r and we see that the error is substantially reduced. The camera has a frame rate of 30 Hz which implies a frame time of 33.3 ms. In reality, 1080 lines at 40,905 lines/s implies a readout time of 26.4 ms with a 6.9 ms latency between frames, perhaps for image compression or writing to flash.

V. CONCLUSIONS AND FUTURE WORK

A rolling shutter, even for simple camera motion, was shown to cause significant errors for common robotic vision tasks such as pose estimation. A critical parameter for modelling image formation with a rolling-shutter camera is the line scan rate. In this paper we presented a novel and effective methodology for estimating this parameter, and our experimental results show a significant difference between the naive assumed value and that measured. We also introduced a new algorithm for modelling rolling shutter effect which has no motion restrictions and can be applied to cameras with arbitrary projection models. Future work includes experimentation and verification for more complex camera motion and how the empirical modelling of the rolling shutter effect could be used in robotic applications.

REFERENCES

[1] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[2] R. A. Newcombe, S. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *ICCV*, 2011, pp. 2320–2327.

[3] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1052–1067, 2007.

[4] C. Geyer, M. Meingast, and S. Sastry, "Geometric models of rolling-shutter cameras," in *Proceedings of Omnidirectional Vision, Camera Networks and Non-classical Cameras*, October 2005.

[5] L.-W. Chang, C.-K. Liang, and H. Chen, "Analysis and correction of rolling shutter distortion for cmos image sensor arrays," in *ISCOM'05*, 2005.

[6] J.-B. Chun, H. Jung, and C.-M. Kyung, "Suppressing rolling-shutter distortion of cmos image sensors by motion vector detection," *Consumer Electronics, IEEE Transactions on*, vol. 54, no. 4, pp. 1479–1487, 2008.

[7] C.-K. Liang, L.-W. Chang, and H. Chen, "Analysis and compensation of rolling shutter effect," *Image Processing, IEEE Transactions on*, vol. 17, no. 8, pp. 1323–1330, 2008.

[8] S. Baker, E. P. Bennett, S. B. Kang, and R. Szeliski, "Removing rolling shutter wobble," in *CVPR*, 2010, pp. 2392–2399.

[9] P.-E. Forssen and E. Ringaby, "Rectifying rolling shutter video from hand-held devices," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 507–514.

[10] J. Hedborg, P.-E. Forssén, M. Felsberg, and E. Ringaby, "Rolling shutter bundle adjustment," in *CVPR*, 2012, pp. 1434–1441.

[11] M. Li, B. Kim, and A. I. Mourikis, "Real-time motion estimation on a cellphone using inertial sensing and a rolling-shutter camera," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013, pp. 4697–4704.

[12] Y.-G. Kim, V. Jayanthi, and I.-S. Kweon, "System-on-chip solution of video stabilization for cmos image sensors in hand-held devices," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 21, no. 10, pp. 1401–1414, 2011.

[13] M. Grundmann, V. Kwatra, D. Castro, and I. Essa, "Effective calibration free rolling shutter removal," *IEEE ICCP*, 2012.

[14] G. Hanning, N. Forslow, P.-E. Forssen, E. Ringaby, D. Tornqvist, and J. Callmer, "Stabilizing cell phone video using inertial measurement sensors," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp. 1–8.

[15] L. Oth, P. Furgale, L. Kneip, and R. Siegwart, "Rolling shutter camera calibration," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, 25–27 June 2013.

[16] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *Int. J. Comput. Vision*, vol. 81, no. 2, pp. 155–166, Feb. 2009.