Towards Kilo-Hertz 6-DoF Visual Tracking Using an Egocentric Cluster of Rolling Shutter Cameras

Akash Bapat, Enrique Dunn, *Member, IEEE*, and Jan-Michael Frahm, *Member, IEEE*

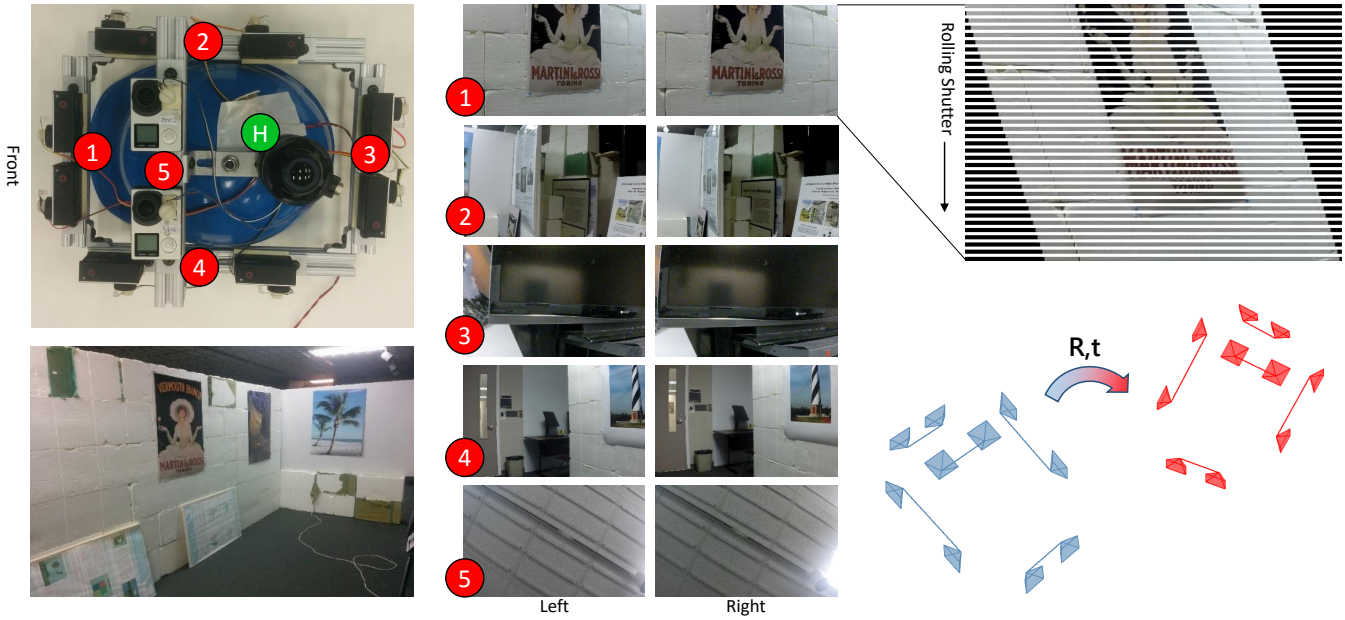


Fig. 1. *Top Left*: Our prototype cluster built by mounting 10 Go-Pro cameras on a helmet. (1) through (5) denote stereo pairs pointing forward, right, backward, left, and up, relative to the head of the user. (H) marks a Hi-ball tracking camera, which was used to obtain ground-truth 6-DOF motion for the device. *Bottom*: View of the room where our real-world experiments were captured. *Middle*: Example stereo pair images for one timepoint of capture from our real-world data. Stereo pair (5) captures a view of the ceiling. *Top right*: Illustration of the rolling shutter effect for a single image. Rows of the image are captured sequentially at high frequency. Motion of the camera during this capture induces distortion in the x-direction of the image plane. *Bottom right*: Tracking the pixel shift from row to row in multiple cameras allows us to estimate 6-DOF device motion from one time point (blue) to the next (red) at a much higher frequency than the camera frame rate.

Abstract—To maintain a reliable registration of the virtual world with the real world, augmented reality (AR) applications require highly accurate, low-latency tracking of the device. In this paper, we propose a novel method for performing this fast 6-DOF head pose tracking using a cluster of rolling shutter cameras. The key idea is that a rolling shutter camera works by capturing the rows of an image in rapid succession, essentially acting as a high-frequency 1D image sensor. By integrating multiple rolling shutter cameras on the AR device, our tracker is able to perform 6-DOF markerless tracking in a static indoor environment with minimal latency. Compared to state-of-the-art tracking systems, this tracking approach performs at significantly higher frequency, and it works in generalized environments. To demonstrate the feasibility of our system, we present thorough evaluations on synthetically generated data with tracking frequencies reaching 56.7 kHz. We further validate the method's accuracy on real-world images collected from a prototype of our tracking system against ground truth data using standard commodity GoPro cameras capturing at 120 Hz frame rate.

Index Terms—High frequency, Visual inside-out tracking, Rolling shutter

1 Introduction

End-to-end system latency has long been the fundamental challenge in Augmented Reality / Virtual Reality (AR/VR) applications. A convincing slight-of-hand for AR/VR requires that the basic tasks of the system – namely tracking, rendering, and display – be performed at faster rate than user perception. While all three of these tasks present their own challenges, obtaining high-quality tracking is a significant roadblock for system realization, since errors and delay in tracking necessarily limit the performance of the subsequent steps. This is especially true for AR scenarios, where users are able to notice even small misalignments of the virtual world with the external environment. Tracking solutions in VR devices, too, are still underdeveloped, and to a large extent there exists no high-quality, low-cost VR tracking system that works in unrestricted environments. With the increasing ubiquity of inexpensive VR displays, and with the continual development of AR displays, obtaining accurate, low-latency 6-DOF tracking is one of the most crucial immediate problems for the AR and VR

- *Akash Bapat is with UNC Chapel Hill. E-mail: akash@cs.unc.edu.*
- *Enrique Dunn is with UNC Chapel Hill. E-mail: dunn@cs.unc.edu.*
- *Jan-Michael Frahm is with UNC Chapel Hill. E-mail: jmf@cs.unc.edu.*

Manuscript received 18 Sept. 2014; accepted 10 Jan. 2015. Date of Publication 20 Jan. 2015; date of current version 23 Mar. 2015.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.

communities to solve.

In commercially available VR head mounted displays (HMDs) like the Oculus Rift DK2 and Samsung GearVR, inertial measurement units (IMUs) are used to estimate the relative 3-DOF rotation of the device. IMUs such as gyroscopes and accelerometers provide fairly reliable rotational measurements with nearly 1000 Hz tracking frequency. In addition, an external camera might be used to get low frequency positional tracking, as in Oculus Rift. The recent research has sought to provide full 6-DOF tracking missing from inertial sensors by relying on computer vision techniques (visual tracking) that allow for either *inward* or *outward* tracking of the device in relation to the surrounding environment. For inward tracking, one or more "inward-looking cameras" (*i.e.,* cameras situated in the environment and pointed at the user) track distinct markers on the HMD. Such markers are typically light sources, *e.g.,* infra-red light emitting diodes (LEDs) on the Oculus Rift, or large ping-pong-size balls, as on the Sony Move. In an inverse configuration, outward tracking techniques use one or more cameras on the AR/VR device to detect markers placed in the surrounding environment. The main drawback to these tracking approaches is they are typically high-latency, high-cost, or both. Moreover, the requirement of cameras or markers placed in the external environment limits the generality of these systems – that is, 6-DOF tracking of the AR/VR device is only possible within the small area containing the cameras/markers. Newer systems based on Simultaneous Localization And Mapping (SLAM), refer Sec. 2 are being developed to perform markerless tracking to mitigate this.

Two basic challenges for high-frequency visual tracking are: 1) the basic assumption of a single pose for each captured image, and 2) the requirement of multiple frames for reliable pose estimation. Accordingly, multi-frame approaches inject high latency and over-smoothing of the estimated 6-DOF motion. Moreover, the required tracking frequency for AR/VR is commonly much higher than the frame rate of the camera. For instance, the human neck can support rotational speeds of up to 700 deg/s at peak rates and up to 70 deg/s at normal rates [4], and normal human walking speed is 1.39 m/s (5 km/hr). Extremely fast tracking is essential to precisely match the motion of a user's head.

In this paper, we propose a new markerless approach for outward visual tracking of AR/VR devices that achieves high accuracy and low latency without requiring any off-device tracking elements. Our key insight is that extremely high-frequency image sampling is attainable using rolling shutter cameras. We treat a rolling shutter camera as a high frequency 1D sensor capturing H 1D row-images sequentially in time, rather than a 2D sensor capturing a single H$\times$W 2D frame at each time-point. This enables us to process the video frame per-row rather than waiting for the complete image formation. Using a rig of multiple rolling-shutter stereo pairs, we obtain small-motion pose estimates at very high sampling rates *entirely* from cameras located on the HMD. We demonstrate this approach can obtain tracking rates as high as 57 kHz on synthetic data simulating actual user head motion. We also provide a proof-of-concept real data experiment using a prototype model of our rolling shutter tracking system with 5 rolling-shutter stereo pairs capturing at 120 frame rate. Using this cluster, we are able to perform tracking at frequencies as large as 80.4kHz on real data. Fig. 2 shows an overview of our approach.

The rest of this paper is laid out as follows: In Section 2, we discuss work related to the problem of high-frequency 6-DOF tracking of AR/VR devices. We proceed in Section 3 to describe our approach to row-wise rolling-shutter tracking, and we detail our camera cluster set-up in Section 4. In Section 5, we provide thorough experimental evaluation of our method on both simulated and real-world data. We address future work and conclude the paper in Section 6.

2 Related Work

High-frequency throughput has long been a main research thrust in the three primary technologies (namely tracking, rendering, and display) of AR/VR [3, 40]. In this section, we provide an overview of related work that has focused on solving the tracking problem and then discuss recent research on rolling shutter effect. On a high level, methods in tracking can be divided into three categories: sensor-based, a hybrid of sensor- and visual-based, and purely visual-based.

Sensor-based tracking. High-frequency sensor-based tracking systems have been deployed with some degree of success for many decades; see [28] for a good review of early methods on this front. More recently, LaValle *et al*. [20] have proposed to use only inertial sensors – such as accelerometers, magnetometers, and gyroscopes – combined with a predictive system for head pose tracking by assuming constant angular velocity or constant acceleration. Intriguingly, they also report limited success in positional tracking, in addition to rotational tracking, using only the inertial sensors. However, the inherent limitations of inertial sensors render them ineffective for full 6-DOF tracking, and as such, most recent work has focused on using sensors and vision systems jointly to perform tracking.

Hybrid tracking. One approach toward hybrid sensor/vision tracking is to use inertial sensors as the primary tracking component and augment them with vision systems to mitigate drift. Klein *et al*. [18] used predictions from IMU sensors to account for motion blur and built a parametric edge detector on video input to prevent drift. In Persa [26], IMUs and Kalman filtering were used for tracking, and to correct for drift, the author used GPS in outdoor environments and fiducial markers in indoor scenes. Both of these systems use the IMU as the workhorse, and computer vision is a secondary tool.

Visual-based tracking. Visual trackers, which consider cameras as primary sensors, can be broadly classified into either feature-based or direct approaches. Whereas the former analyze a set of salient image keypoints, the latter rely on global image registrations. Many visual-based tracking systems fall under the category of simultaneous localization and mapping (SLAM) approaches. Regarding feature-based systems, Ventura *et al*. [35] used a client-server model, where the server has a 3D model of the environment constructed offline. A mobile phone acts as the client and runs a SLAM system in local frame of reference, using the server for global registration and bundle adjustment. Forster *et al*. [12] demonstrated a 'semi-direct' approach running at 300 fps on a consumer laptop. This method uses photometric error between projections of 3D points in consecutive frames for motion estimation and employs FAST [29] features for the mapping stage.

Direct SLAM approaches have also succeeded for both visual tracking and scene reconstruction in recent years. The LSD-SLAM algorithm of Engel *et al*. [9] and its derivatives [5, 10] use direct, semi-dense image alignment to reconstruct 3D models at nearly the rate of frame input. Schöps *et al*. [32] further introduced a direct approach on a mobile phone using semi-dense depth maps for mesh-based geometry representation. Their method is hybrid in that they find the ground plane using data from the built-in accelerometer on the mobile device. Overall, direct SLAM methods have been demonstrated to work on a large scale with relatively low computational requirements, and these systems have been leveraged for pose estimation in AR systems. However, the key limiting factor preventing these visual tracking methods from general AR/VR use is they require full camera video frame inputs, which is a substantial bottleneck for overall system latency. Effort by Dahmouche *et al*. [7] towards this end was to predict and capture only the regions of interest (ROI) around feature points as opposed to capturing complete image and then finding the features. They predicted the ROIs based on previous data under a constant velocity assumption to simultaneously track pose and velocity at 333Hz using a high speed camera. Unlike SLAM systems, our camera tracking system does not build an environmental map. Instead we rely on scan-line stereo depth variations across time to estimate 6 DOF motion.

Other feature-based hybrid tracking systems have used strategically placed fiducial markers for tracking distinct points in the environment [38, 23]. Typically, Light Emitting Diodes (LEDs), beacons, or unique texture markers are used in this framework. The Hi-Ball system [36] is one such construction, in which blinking LEDs are placed on the ceiling of a capture environment. A cluster of infrared cameras (the "Hi-Ball") observes the blinking pattern of LEDs, and the system uses strong triangulation constraints combined with motion prediction

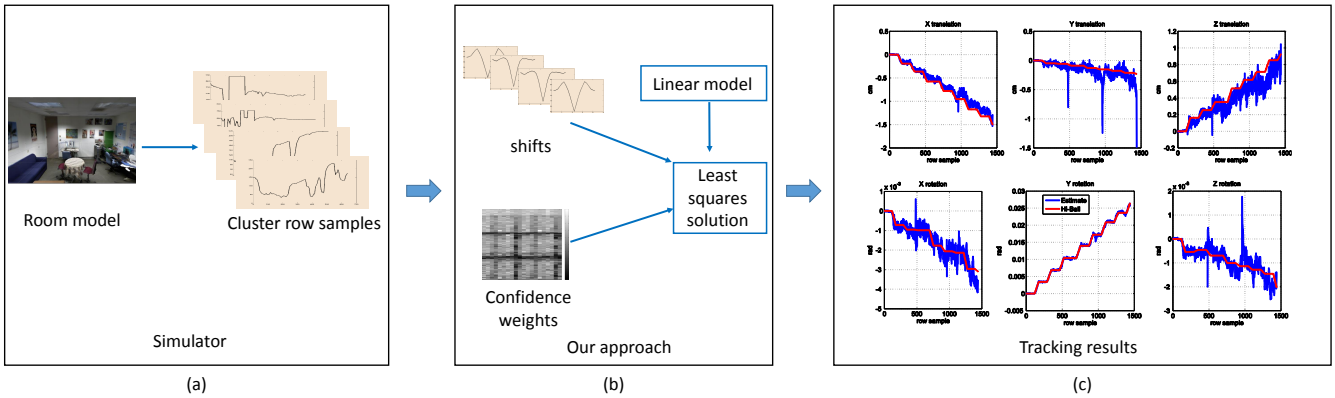


Fig. 2. Overview of our system for simulated experiments: (a) Simulation provides image data for multiple virtual cameras. (b) Our approach estimates shifts and confidence scores for pose estimation. (c) We compare our tracking results with ground-truth Hi-Ball tracker motion data.

to provide highly accurate 6-DOF pose. The major drawback of using such fiducial markers is that it becomes cumbersome as well as costly to place them throughout the environment, and, moreover, the tracking system is limited to the confines of the area in which the markers have been installed. To obtain ground truth for our real-world experiments, we have mounted a Hi-Ball on our prototype camera rig (see Fig. 1, top left) and use such a system only to validate the accuracy of our approach.

Current state-of-the-art visual tracking systems, such as the Oculus Rift, perform rotational and positional tracking of the AR/VR device using an inward-facing camera focused on the user. Because this approach involves tracking the user based on fiducial points in the 2D image plane, the system has a strict lower bound on latency that is determined by the frame rate of the camera (*e.g.,* 60 Hz). Moreover, the capture set-up inherently confines the tracking area to the field of view of the camera. To solve both these problems, we propose to instead perform visual tracking using an *outward-view*, *rolling-shutter* approach, where multiple rolling-shutter camera stereo pairs located on the device itself are used to recover the motion of the user, without any specific design for the surrounding environment. In the following, we provide a brief introduction to rolling-shutter capture.

Rolling Shutter. Traditional film cameras, as well as CCD arrays in early digital cameras, exposed the entire 2D image space during a common exposure period; this type of exposure is called global shutter. With the advent of low-power CMOS camera sensors, rolling-shutter (RS) cameras (see Fig. 1, top right) have quickly become ubiquitous for everyday camera-equipped devices. To save space on the silicon chip, the newer CMOS sensors perform sequential exposure of sensor rows, which only requires a single, simpler set of holdout circuitry to read all rows. In this fashion, each individual row of the rolling-shutter image is actually a snapshot of the scene at a different time instance. For image captures without significant scene or camera motion, rolling shutter is often an effective, low-cost imaging technique. However, when the scene contains a fast-moving object (*e.g.,* the blades of a fan), or when the camera has fast motion relative to the frame rate of the camera, artifacts such as wobble skew and other undesirable effects become apparent [11]. Recent research has focused on modeling RS effects and solving traditional problems like multiple-view stereo for RS cameras [30] and bundle adjustment [15]. Albl *et al.* [2] propose R6P, which is rolling shutter version of the perspective-n-point (PnP) problem of finding camera pose from n 2D-to-3D correspondences. They solve R6P by modeling rolling shutter effect in their linear solver based on Gröbner basis functions [6] which requires at least 6 correspondences. They propose a linearized camera model around identity rotation and use the solution from standard P3P [14] as initialization to provide higher number of inliers in a RANSAC setup. Su and Heidrich [34] proposed a method to remove motion blur using a single rolling shutter image. Their method explicitly models for rolling shutter to estimate the motion during exposure by fitting polynomials to each degree of freedom of camera motion. Another paper dealing with motion blur in a RS camera is [22], where the authors demonstrate a real-time structure-and-motion (SaM) system for RGB-D sensor while accounting for motion blur by assuming uniform velocity over the image.

Geyer *et al.* [13] proposed a method to estimate RS line-delay, *i.e.,* the time interval between start of capture of two consecutive rows, using specialized hardware. Oth *et al.* [24] improved the results while using only video. They model rolling shutter and use weak motion prior with a continuous time trajectory model. They parameterize the pose using fourth order B-spline and iteratively update their model parameters while minimizing reprojection errors to estimate the line-delay.

For correct modeling of rolling shutter, one has to consider a (potentially) different pose for each row in the image. Traditionally, such pose estimation has been employed to correct for motion artifacts, in an attempt to simulate global shutter exposure. For example, Ringaby and Forssén [27] parametrized intra-frame rotation with a linear spline and used this model for image rectification and video stabilization. In their approach, homographies between images rows are used to reconstruct the rectified image. We adopt a similar technique in our method when mapping between image rows. Other works, in particular that of Ait-Aider *et al.* [1], have treated a rolling shutter camera as a velocity sensor to estimate pose and instantaneous velocity of 3D objects. Instead of serving as a source of error, the rolling shutter effect is a source of information in their system.

3 Tracking with Multiple Rolling Shutter Cameras

In the following, we describe how rolling shutter can be leveraged for high-frequency tracking. We then introduce our linear model used for rolling-shutter-based tracking, explain how to search for correspondences between captured rows, and address how to estimate rotational motion via homographic mappings between video frames. At the end of this section, we briefly describe corrective measures and confidence scores that help to maintain stable tracking.

Rolling shutter as a 1D sensor. As in Ait-Aider *et al.* [1], we exploit the rolling shutter effect to our advantage to enable high-frequency tracking. We leverage the fact that each row of a rolling shutter image is shifted in time by a very small offset. We propose to treat a rolling shutter camera as a high-frequency line sensor where each of the rows constitutes an individual sample. With this definition, even an inexpensive smartphone camera with 1000 rows and 30 fps has an equivalent 1D sampling frequency of 30 kHz. The natural barrier of camera frame rate, which is the limiting factor for visual tracking approaches using 2D images, is surpassed by several orders of magnitude using our approach. The trade-off we must mitigate is that this increased sampling comes with a substantial reduction in the vertical field of view.

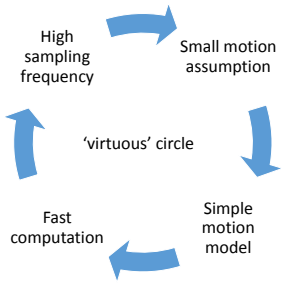


Fig. 3. *Virtuous Circle.* Each step reinforces the subsequent step.

3.1 Our Model

Bishop [4] introduced an inside-out tracker using a cluster of custom-made 1D sensors capturing line-images at 1000 fps. The proposed tracker utilized a computationally efficient linear motion estimation framework under a small-motion assumption. Assuming small motion results in a simplified model, which reduces the computation time. In turn, the lower computational burden allows higher sampling frequencies, reinforcing the original small motion assumption. This 'virtuous' circle (see Fig. 3) is one of the key enablers of the approach.

In our work, we generalize Bishop's [4] model to use rolling shutter cameras, with each image row treated as an individual 1D sensor. This provides us with an array of line sensors per RS camera, each sampling at the frame rate of the camera (*e.g.,* 120 Hz). Because the row sampling occurs sequentially, the effective sampling rate of these line sensors is in the kHz range for a typical HD frame sensor. Moreover, these line sensors are in a fixed configuration with respect to each other, meaning change in relative pose between camera rows is entirely governed by the motion of the entire device.

To achieve this high-rate capture, it is necessary to model the set of line sensors in a manner similar to a single high-frequency sensor, for which the small motion assumption is trivially valid. For the single-sensor case, sequential captures are directly comparable because the same physical sensor array is used for both timepoints. For finding such correspondences in RS imagery, the key challenge we face is to hypothesize the image of a line sensor n in the current row as if it had been captured at the same time – and with its respective 6-DOF pose at that time – as the previous row. To this end, we use an inter-frame homography to predict a line n in the current frame by transforming lines from the previous frame. We follow [4] and measure shifts between the current row-image and the recreated row-image to quantify the small motion. The shift between these rows separated in time is denoted as s_t.

Given that Bishop's model requires estimation of depth, cameras are arranged in stereo pairs in the cluster. A row-wise depth value, s_d, is then estimated by measuring the shift within each stereo pair. We assume a rigid cluster configuration with known intrinsic and extrinsic camera calibrations, which can be achieved in practice using calibration methods such as [19, 21]. Due to the relatively small baseline of our stereo pairs, which leads to poor depth estimates in large, open spaces, we also assume a static indoor scene for the capture environment. In our simulation, we use 10 stereo-pairs of RS cameras looking in all directions, and in our real-world data experiments, we use 5 stereo pairs.

3.2 Linear Model

Next, we introduce the linear model used in our work. Let $V_{cam} \in SE(3)$ represent the fixed relative pose (cluster-to-camera) of a given camera relative to the coordinate frame of the cluster center. Consider a 3D point X^t_{cam} whose coordinates are expressed relative to the reference frame of camera *cam* at time t. If M^t represents the change in camera cluster pose between time-steps t and $t+1$, we can express the change in a 3D point's position relative to a given camera as

$$Y^{t+1}_{cam} = V_{cam} M^t V^{-1}_{cam} X^t_{cam}, \tag{1}$$

where Y^{t+1}_{cam} is the new 3D position of X^t_{cam} relative to the given camera at time $t+1$.

The camera model in our case is a 1D rolling-shutter sensor row, with a sampling rate equal to the frame rate of the imaging device. Let $\hat{X}^t_{cam}$ denote the 3D point that projects onto the center pixel of this row. For a given time point, we can apply Eq. (1) to $\hat{X}^t_{cam}$ for each camera. This gives us one independent linear equation for each camera.

Now, the shift from time-point t to $t+1$ can be expressed in terms of stereo and temporal disparities. As mentioned previously, let s_d represent the distance, in pixels, between $\hat{X}^t_{cam}$ and its corresponding point in the other stereo camera at time t. Let s_t represent the detected pixel shift between the $\hat{X}^t_{cam}$ and $\hat{Y}^t_{cam}$ when the points are projected onto the 1D image plane of the camera. (We explain the method for obtaining this correspondence in the next section.) Then, the homogeneous points $\hat{X}^t_{cam}$ and $\hat{Y}^{t+1}_{cam}$ can be expressed in terms of the measured shifts s_t and s_d as follows:

$$\hat{X}^t_{cam} = \begin{bmatrix} 0 & f_r(r) & -f_d(s_d) & 1 \end{bmatrix}^T \tag{2}$$

$$\hat{Y}^{t+1}_{cam} = \begin{bmatrix} f_s(s_t) & \boxtimes & \boxtimes & W \end{bmatrix}^T, \tag{3}$$

where the $\boxtimes$ denote unknown values that require future row-images for estimation, and W is simply the homogeneous co-ordinate. As the $\boxtimes$ values are only determined by future observations, we do not make use of them in our model. Here, f_d converts the disparity s_d into camera-space distance according to the focal lengths and baseline b of the stereo pair. Similarly, f_r and f_s convert the row index r and the shift due to motion s_t into camera space.

Motion model. Any general motion M, can be expressed in terms of translation and rotation in each direction. At high sampling frequencies, the small motion assumption allows us to model rotation as differential, that is, as a small perturbation from identity. The differential motion dM takes the following form:

$$dM = \begin{bmatrix} 1 & -\theta_z & \theta_y & T_x \\ \theta_z & 1 & -\theta_x & T_y \\ -\theta_y & \theta_x & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

Where the values θ_* are the (differential) Euler angles while T_* are small translations in each direction. If we combine Eq. (1) with Eqs. (2) and (3), then expand to express in terms of the six unknowns, we arrive at the over-determined weighted linear system

$$CAY = CB. \tag{5}$$

Matrix $Y = \begin{bmatrix} T_x & T_y & T_z & \theta_x & \theta_y & \theta_z \end{bmatrix}^T$ represents the unknowns, and C captures our confidence in the shifts s_t; see Sec. 3.5 for a description of C. A is a function of the cluster configuration, row index r and shifts s_d which can be computed form previous data. Since the vector B is the only part of Eq. (5) that depends upon the shifts s_t obtained using current data, the matrix A can be pre-computed.

This is an incremental system which inherently drifts over time. In the system of [4], the use of beacons placed strategically in the scene was suggested as a solution. In the case of 1D sensors, the necessity for this constraint arises because the sensors cannot be used for global drift correction; hence, beacons are needed to serve as fixed external references. As we use rolling shutter cameras, we do not have this limitation, and we can use the 2-D images and the per-row poses as an input for feature-based systems, which can run at a lower frequency for global drift correction. We leave it as a future work to employ a SLAM-type system which would run in parallel to our tracking approach.

3.3 Shift Estimation

Row Descriptor. So far, we have assumed that we are able to accurately measure the shifts s_t and s_d. We now describe the method we use to estimate the shifts described above. Bishop [4] highlighted the importance of a binary descriptor for rows, but the approach was not

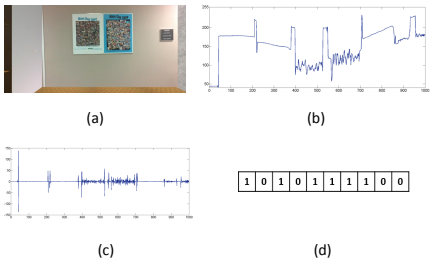


Fig. 4. (a) Original rolling shutter image. (b) Visualization of pixel intensity values for the 200^{th} row of the image. (c) Double derivative of the smoothed row. (d) Representative binary descriptor.

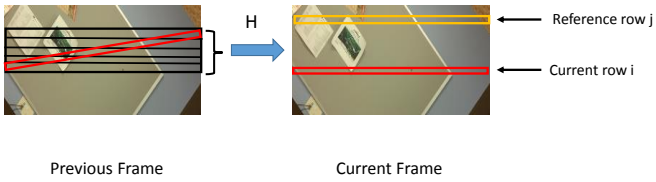


Fig. 5. Rotation compensation: A block of rows from the previous frame are transformed using homography H to predict how the current row i will appear under the rotation of reference row j.

robust to camera noise patterns. In our system, we adapt Bishop's representation and convolve each row sample with a double derivative of the Gaussian to obtain a smoothed version of the edge response (see Fig. 4 as an example). The smoothing provides robustness against noise, and the double derivative helps us to localize shifts to sub-pixel accuracy. After this convolution, the resulting values are thresholded at zero to provide a binary descriptor for each row.

Descriptor Matching. The binary descriptor is used for matching the rows to estimate shifts s_t and s_d. The Hamming distance is used for fast computation of the matching cost. The evaluated pixel shifts s_t are limited to the range of $[-20, 20]$ pixels. We fit a parabola at the minima to recover a subpixel shift estimate [31]. To estimate the depth disparity s_d, we match the descriptor for a larger range of $[10, 70]$ pixel shifts. We use a lower bound because disparity becomes inaccurate at very high depths, and we use an upper bound because a very high disparity indicates that the camera is too close to a surface (and hence the stereo-pair cameras might not observe sufficient structure to produce an accurate shift).

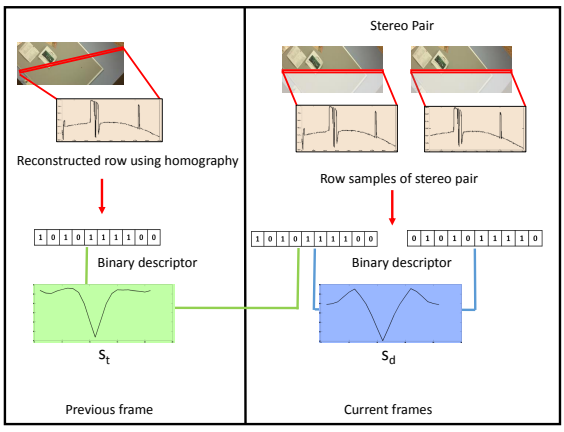


Fig. 6. Measuring s_t and s_d using rotation compensation (best viewed in color).

3.4 Rotation Compensation

To estimate the motion-induced shift of a single camera across time, we compare the pixel content of the currently sensed row-image against the corresponding content observed on the previous frame. In practice, the motion between successive frames is dominated by the rotational component. Hence, in order to justify the use of the small motion assumption, we compensate for this rotation through an homographic warping between consecutive frames. We extract a block of rows from the previous frame and transform them to match the rotation of reference row j occurring some time before the current row i (see Fig. 5). This rotation compensation is performed using an adaptively specified reference row j so that the rotation between row i and row j does not fall in the magnitude of noise and hence can be estimated reliably. Let us denote the rows which are to be transformed using a homography as l_q, where q is the row index in the image. If the current frame index is k, row-image i is in k^{th} frame, while row l_q is from the $(k-1)^{th}$ frame. Let the rotation at row l_q be R_q and the rotation at the j^{th} row be R_j. Our adaptive reference scheme (see Sec. 3.4.1) decides whether row j is from frame k or from frame $k-1$. The homography between row j and row l_q, given the camera intrinsic matrix K, can be computed as

$$H_{j,q} = K R_j R_q^T K^{-1}. \tag{6}$$

R_j and R_q are known quantities which were previously estimated, hence it is trivial to compute $H_{j.q}$. Elimination of the dominant image motion, *i.e.,* rotation, ensures that a large range of motions can be covered by our system. Fig. 12 shows that we can handle motion as large as 500 deg/s using this compensation technique.

3.4.1 Adaptive Reference

As implied above, the reference index j need not necessarily be equal to $i-1$. We adaptively change the reference index according to the rotation velocities. The pixel shift values between two consecutive rows of same frame at peak rotational velocities (*e.g.,* 500 deg/s) fall between the range of -0.5 to 0.5 pixels. For normal motion, they are in the range of 0.1-pixel shifts. Such a small shift is at the magnitude of noise and thus cannot be measured accurately. We instead introduce an adaptive technique that enables us to work at reasonable shifts without violating the small motion assumption. To change the reference row index j adaptively, we set a maximum allowable rotation between rows, denoted as $d\theta$. Whenever this threshold is crossed, the reference index j is updated to a new row where the shifts would not fall below the magnitude of noise.

3.5 Corrective Measures and Confidence Scores

We employ the widely used disparity confidence measure Peak Ratio (PKR) [16] to form the diagonal matrix C to weigh each equation in the linear system according to the confidence in its shift s_t. For robust outlier rejection of shifts, we use single exponential smoothing [17] to predict shift S_{pred} at the current time using previous data. If the difference between measured shift S_{meas} and S_{pred} is high, we use the predicted shifts but give a lower confidence score.

4 Cluster Setup

Synchronization. Previously, we assumed that the cameras in the cluster are capturing in synchronization. In reality, this is difficult to achieve without a hardware genlock [37]. As a proof of concept, we have used GoPro cameras in our experiments which do not expose a hardware genlock. Here, we describe our approach to achieve synchronization using a few post-processing steps. Additional specific details are provided in the experiments section.

The GoPro cameras have a vertical rolling shutter. We attach one red LED per camera so that it occupies the entire vertical field of view. For synchronization, we feed a square wave signal to all the LEDs. The peaks of the mean red channel value in the frames enable frame-level synchronization for each camera. Since the LEDs are blinking at know frequency, the rolling shutter effect becomes apparent (Fig. 7).

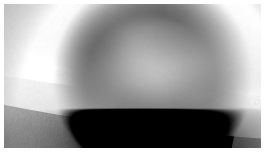


Fig. 7. Red channel image of a blinking red LED showing the rolling shutter effect.

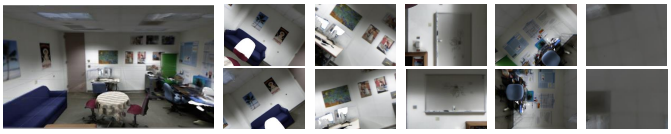


Fig. 8. Scanned room, with small images showing the RS images captured by the left camera in each of the stereo pair from the virtual cluster.

As the turn-off and turn-on time of a LED is on the order of nanoseconds, we can precisely detect which row is at the temporal edge of the drastic change in light. We detect such rows over a number of blinks and regress to get accurate row indices. As we know the period of the square wave, we can match the slopes and extrapolate to any frames captured after the synchronizing signal is stopped. With this, we precisely know across all cameras in the cluster, which rows were exposed at the same instant.

Calibration. While calibration of intrinsic camera parameters and extrinsic stereo pair parameters is nowadays a relatively straightforward task, it is inherently difficult to calibrate cameras with non-overlapping views. The method introduced to calibrate non-overlapping view cameras by Li *et al*.in [21] proved to be insufficiently accurate for our needs. Moreover, to compare the tracker performance against the ground-truth data captured from the Hi-Ball tracker [36], we need to calibrate the cluster with the Hi-Ball, as well. Let T_{c2H} denote the transformation from a camera in the cluster to the Hi-Ball coordinate system. To calibrate the entire cluster, we first calibrated the stereo pairs independently using the traditional checkerboard pattern and Zhang's method [39]. Then, we calibrated the left camera of each stereo pair with the Hi-Ball. The transformation from the cameras to the Hi-Ball T_{c2H} follows the Hand Eye Calibration (HEC) problem. Shah *et al*. [33] provide a good survey of solving HEC, and we follow Park and Martin's method [25] that casts the HEC problem in a least-squares framework. In HEC, there are two known dynamic poses that change over time but can be estimated, and there are two unknown static poses which need to be estimated. To solve for T_{c2H}, we simultaneously capture images of a checkerboard pattern, as well as the Hi-Ball tracker data, at various poses. In our case, the two known poses at each time step are the pose of left camera w.r.t. the fixed checkerboard pattern and the pose of the Hi-Ball w.r.t. to the ceiling. The two unknown static poses are the unknown T_{c2H} and the pose of the checkerboard pattern w.r.t. ceiling.

Number of camera pairs and their layout. As noted in Sec. 3.2, we get one linear equation of the form Eqn. 1 for each camera. To complete the equation, we need depth, hence we need cameras in stereo-pairs. For a fully instep genlocked camera cluster, where each camera captures a row with the same index at the same time, we need atleast 6 stereo-pairs. In our experiments, although we know when each row was captured for each camera using the synchronization method described above, there is a stagger between the rows, *e.g.,* Cam1 captured row 310, Cam2 captured row 331,... at the same instant. Hence we get one equation from each camera due to row staggering, to give us 10 equations in total from 10 GoPros. We have placed the 5 stereo-pairs heuristically to minimize overlap between their fields of view. This ensures that each stereo-pair provides newer information, and reduces the chance of multiple cameras capturing homogeneous scenes.

5 Experimental Results

To evaluate our approach, we first present rigorous experiments on rolling shutter (RS) image data using a simulator, which provides us complete flexibility over camera parameters, their relative positioning, size of the room in which data is captured, and the ground-truth motion. For testing, synthetic ground-truth motion is the most suitable in terms of flexibility, but it may not capture the distinctiveness of real human motion data. Therefore, we utilize captured human motion using Hi-Ball tracker [36] system. According to the authors of [36], this system can record over 2000 pose estimates per second with less than 1ms latency and less than 0.5mm and 0.03 degrees of absolute error. To simulate this captured human motion at more than 2000 Hz, we linearly interpolate the translation and use spherical linear interpolation (SLERP) for rotation. We characterize errors in our tracking system in terms of pixel errors in display of synthetic objects at a distance of 1m away from the user, as done by [4]. We consider an error of 1 pixel display error as permissible. To quantify these errors, we take the display resolution of the most recent HTC Vive VR device, which is 1080×1200 pixels per eye [1]. In all experiments, we assume that the motion for the first frame is zero, and hence all rows in this frame share the same pose. As a result, the first image is essentially a global shutter image, while the rest are captured using rolling shutter. We use this assumption because our system is incremental and requires a reference pose; having a known pose for each row of the initial rolling shutter frame would also be acceptable We set $d\theta = 0.06$ radian as the threshold for rotation compensation.

Simulations. We use a simulator written in OpenGL and QT to simulate synthetic ground truth motion and capture RS images at very high frequencies. To simulate a real indoor scene, we use a room mesh obtained from system similar to Dou *et al*. [8], which we call the *scanned* room. For all simulator experiments, we use 10 RS stereo pairs with a baseline 10cm. Each camera has 640×480 resolution, 60 degree vertical FoV, and captures at a frame rate of 120 Hz. The effective RS sampling frequency is 57600 rows/s.

Experiment 1. In this experiment, we examine the general characteristics shown by our tracker. Fig. 9 shows the tracking results for Hi-Ball motion tracker data simulated inside the *scanned* room, where the X axis is the row-sample index (analogous to time) and the Y-axis is the estimated variable. The red graph is the ground-truth data obtained from the Hi-Ball tracker, while the blue graph is our tracking result. Note that in (a) and (b), the scale of the Y axis for each variable is different. The plots show that the relative error increases when the variable to be estimated is within the order of noise. Fig. 9 (c) depicts the confidence as a heat map, where the X axis depicts the camera number and the Y axis is the row-sample index (time). The color signifies the confidence level of each camera for a particular row. The confidence scores indicate that cameras 9, 10, 19, and 20 have low confidence in measured shifts. Upon further examination, it was found that these cameras were observing a region with homogeneous color. Fig. 9(d) highlights the problem of drift, where an error in estimate is carried over to the next frame and periodically increases. The red marks at the bottom signify image boundaries.

Experiment 2. In this experiment, we simulate synthetic motion data with added Gaussian noise to simulate moderate and extremely large motions. For the moderate motion, we consider a translation velocity (v) of 1.4 m/s and rotational velocity (ω) as 120 deg/s. As we can see from the tracking plots in Fig. 10, jumps occur at image boundaries (480, 960,...) which is also reflected in the confidence score. If all the confidence scores are less than 1, it implies that we did not measure shifts correctly, and hence the estimated motion should be ignored. This occurs because we reconstruct row samples using a homography, but at the boundary of the image, an insufficient number of rows are available for accurate prediction. For extreme motion (Fig. 12), we consider $v = 1.4$ m/s and $\omega = 500$ deg/s. The error plot shows that even with sustained extreme motion, the pixel error incurred hovers around the acceptable limit of 1 pixel but increases at the image boundary. Our rotation compensation technique (Section 3.4), enables us to track at these extreme speeds.

[1] http://www.htcvive.com/us/product/

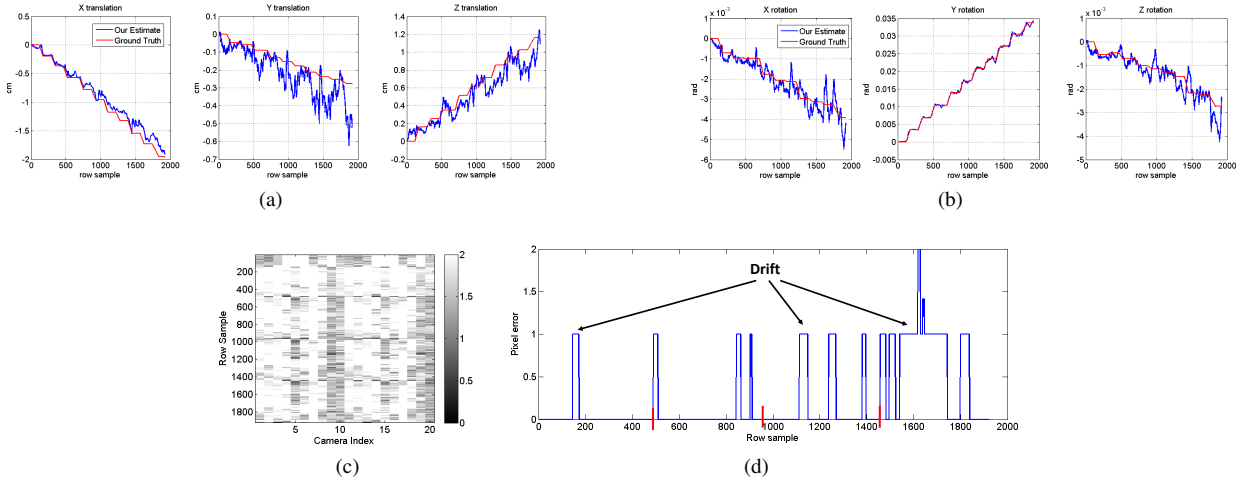


Fig. 9. Comparison of pose estimates versus ground truth for the scanned room simulation, using Hi-Ball motion ground truth. (a) Translation. (b) Rotation. (c) Confidence score. (d) Pixel error in display. Note that y-axis scales are not the same (best viewed in color).

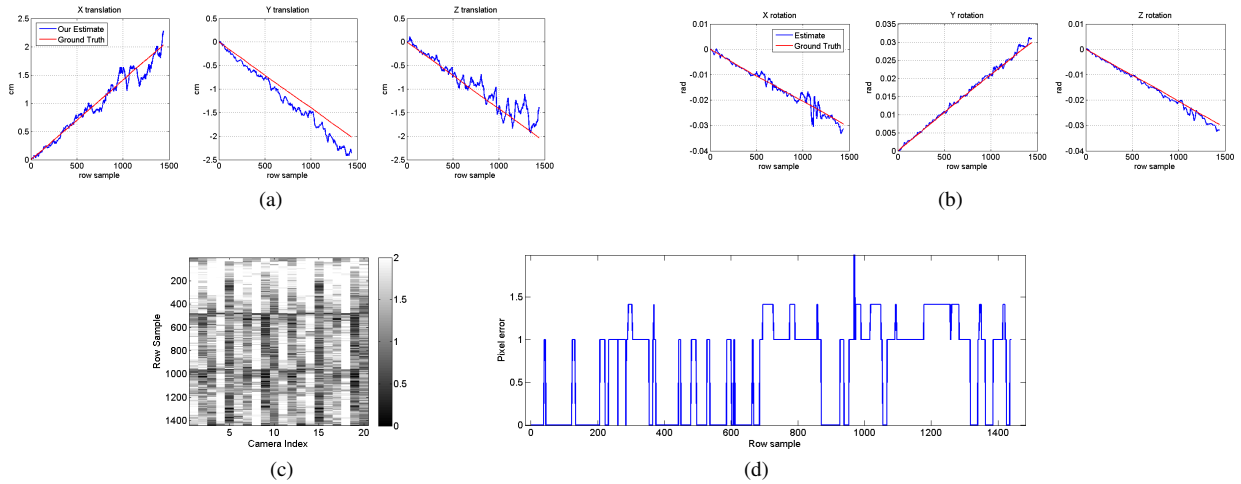
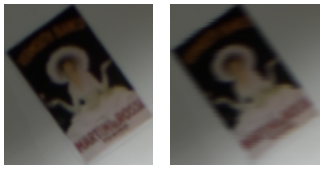


Fig. 10. (a) Translation, (b) Rotation, (c) Confidence score and (d) Pixel error in display, Estimated for Scanned Room, synthetic large motion, note scales not same (best viewed in color)

Exp #	E_{RMS}	T_x (cm)	T_y (cm)	T_z (cm)	θ_x (rad)	θ_y (rad)	θ_z (rad)
Experiment 3	Linear approximation	0.0841	0.079	0.11	0.000332	0.000253	0.000295
	Non-Linear solution	0.07799	0.0898	0.1089	0.000909	0.000499	0.000384
Experiment 4	Motion without blur	0.09573	0.2476	0.2002	0.001269	0.001308	0.001433
	Motion with blur	0.10535	0.2429	0.2243	0.001615	0.0016	0.0023

Table 1. RMS error in the estimation variables over 1500 row-samples for v=1.4m/s and ω = 120 deg/s with added Gaussian noise



(a) (b)

Fig. 11. (a) RS image without motion blur, (b) RS image with motion blur, blur is amplified to test for worst case scenarios

Experiment 3. Now, we study the effect of the linear approximation. The fundamental assumption in our approach is that between two rows of RS image, the head motion is linear. We verify how whether this assumption holds and check how much it differs from the non-linear solution. We use a non-linear solver to solve the Eqn. 1 with the linear approximation solution as a starting point for optimization algorithm. As the Table 1 shows, the difference is in the non-linear vs the linear approximation solution not discernible.

Experiment 4. Here, we study the effect of motion blur. In general, a high-fps camera has shorter exposure and hence lower motion blur. Since we use commodity RS cameras, we maintain long exposure times even under the fast sampling of consecutive lines. We simulate the motion blur in software and use the blurred rows as input to our system. We exaggerate motion blur so that we can test for the worst possible conditions. Our approach is robust to motion blur, as highlighted by Table 1.

Experiments with real data. For real experiments, we use the camera rig shown in Fig. 1 with 10 Go-Pro cameras arranged in 5 stereo-pairs. The rig is moved in a room of size 2.6m×6.4m×2.7m with the Hi-Ball tracking LEDs on the ceiling to record precise ground truth. We build the camera cluster using 8 Go-Pro Hero 3+ silver edition and 2 Go-Pro Hero 4 black edition cameras. The cameras are set at a narrow field of view (FoV) with a resolution of 1280×720 capturing at 120 fps. For synchronization, each camera is attached with one red LED so that it will occupy the entire vertical FoV of the camera. A commodity micro-controller (like Arduino or Raspberry Pi) is used to generate a square wave. A current of about 10mA is supplied to each LED so that it can saturate the camera. After the cameras record enough LED blinks, we turn off the synchronization signal and remove the LEDs without stopping the recording. This part of the video is used to synchronize the cameras. After this, we can capture the real data for carrying out the experiments. For the real data experiments, we process one row-image per RS camera at a time and hence work at the highest sampling frequency, which is h×fps. Given that the cameras need to be rectified, the effective height of the RS image captured by the camera is reduced to a minimum of 670 pixels. Thus, effective tracking frequency comes out to be 80.4 kHz at 120 fps. Fig. 13 shows our tracking results using our camera cluster; note that the results are noisy. To mitigate this, an exponential smoothing can be employed for stable tracking at the cost of increased latency (see Fig. 14). We do not show the error plots for the errors in terms of display pixel errors in both the cases because they are less than 0.5 pixels and quantize to zero error.

Effect of noise and scene homogeneity. Noise and homogeneity of scene adversely affects the tracking accuracy. A combination of both is seen in real data. Fig. 15 shows tracking results for translation in X direction for different Gaussian noise with $\sigma = 3, 5, 10$ added to the simulation images; we exclude other variables as they follow the same trend. While noise can be mitigated by better lighting conditions or using quality sensors, it is difficult to overcome scene homogeneity. We simulate scene homogeneity by blurring the simulation images with a Gaussian kernel, $\sigma = 3, 5, 10$. Fig. 16 shows the degradation in performance due to scene homogeneity, which is characteristically different than noise. While noise is random, scene homogeneity's effect on tracking has more structure, where blocks of rows might not provide correct measurements. At extreme noise levels or homogeneous scenes we cannot track *any* motion. Possible future directions to address these issues will be to process blocks of rows rather than individual rows to average out noise, and to use prediction techniques [17, 36] to mitigate scene homogeneity.

Preconditioning. Now, we examine the stability issues of the linear system. Eqn. 5 describes a weighted linear system where A is $N \times 6$, N is the number of cameras in the cluster. When we simplify Eqn. 1 to construct A, the three elements of each row of A are V_{11},V_{12} and V_{13} from camera pose matrix V, which constitute the first row of rotation matrix of the pose. Hence, the scale of the unknown variables becomes important as V_{11}, V_{12}, and V_{13} are guaranteed to be between -1 and 1. Hence, pre-scaling is important where the translation should be in meters and the rotations should be measured in radians.

6 Conclusion

We have presented a markerless, multi-camera, and egocentric visual tracker that breaks through the frames-per-second (FPS) sampling barrier, by leveraging commodity rolling shutter cameras as dynamic 1D (*e.g.,* line scan) sensors. Towards this end, we model the spatio-temporal relationships of a rigid camera cluster in terms of linear motion approximation and efficient image representations, which are based on the small-motion assumptions enabled by our kHz sampling frequencies. Moreover, we have deployed an online prototype system achieving upwards of 80kHz visual tracking. Also, we have validated and characterized the performance and limits of our tracking components on representative synthetic data. Finally, we have discussed integration and implementation details critical to system deployment, such as temporal synchronization and spatial calibration of our setup.

The importance of the proposed approach lies in bridging the sampling frequency gap between inertial (*i.e.,* IMU) and visual sensors, by enabling visual tracking at frequencies that are orders of magnitude greater than the RS camera native FPS. Along these lines, the long-term potential for a more homogenized sensing frequency landscape opens exciting opportunities for novel sensor integration mechanisms. In the short term, research priorities lie on expanding the applicability of our approach through automated synchronization and self-calibration techniques.

Acknowledgments

The authors wish to thank True Price for reviewing the manuscript and Jim Mahaney for constructing frame of cluster. They also wish to thank Henry Fuchs and Alexander Berg for providing GoPro cameras. This research was supported by NSF grant No. CNS-1405847

References

[1] O. Ait-Aider, N. Andreff, J. M. Lavest, and P. Martinet. Exploiting rolling shutter distortions for simultaneous object pose and velocity computation using a single view. In *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, pages 35–35. IEEE, 2006.

[2] C. Albl, Z. Kukelova, and T. Pajdla. R6p-rolling shutter absolute camera pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2292–2300, 2015.

[3] R. T. Azuma. A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385, 1997.

[4] G. Bishop. *SELF-TRACKER*. PhD thesis, University of North Carolina at Chapel Hill, 1984.

[5] D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *Intl. Conference on Intelligent Robots and Systems*, 2015.

[6] D. A. Cox, J. Little, and D. O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[7] R. Dahmouche, O. Ait-Aider, N. Andreff, and Y. Mezouar. High-speed pose and velocity measurement from vision. In *Robotics and Automation. ICRA. IEEE Intl. Conference on*, pages 107–112, 2008.

[8] M. Dou, L. Guan, J.-M. Frahm, and H. Fuchs. Exploring high-level plane primitives for indoor 3d reconstruction with a hand-held RGB-D camera. In *ACCV 2012 Workshops*, pages 94–108. Springer, 2013.

[9] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, 2014.

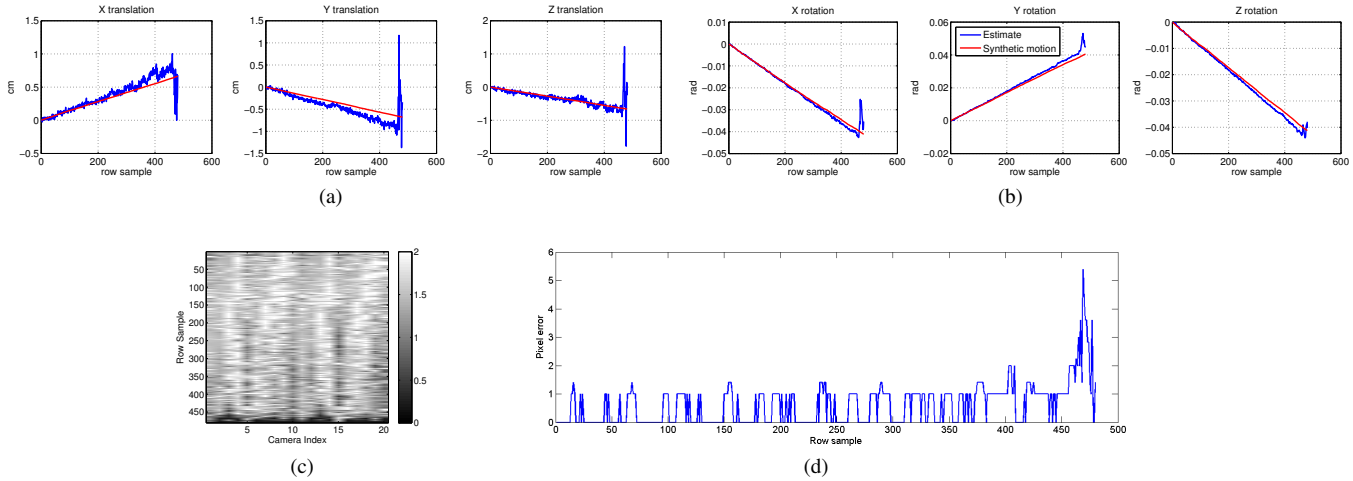


Fig. 12. (a) Translation, (b) Rotation, (c) Confidence score and (d) Pixel error in display, Estimated for Scanned Room, synthetic extreme motion of v=1.4m/s and ω=500 deg/s, note scales not same (best viewed in color)

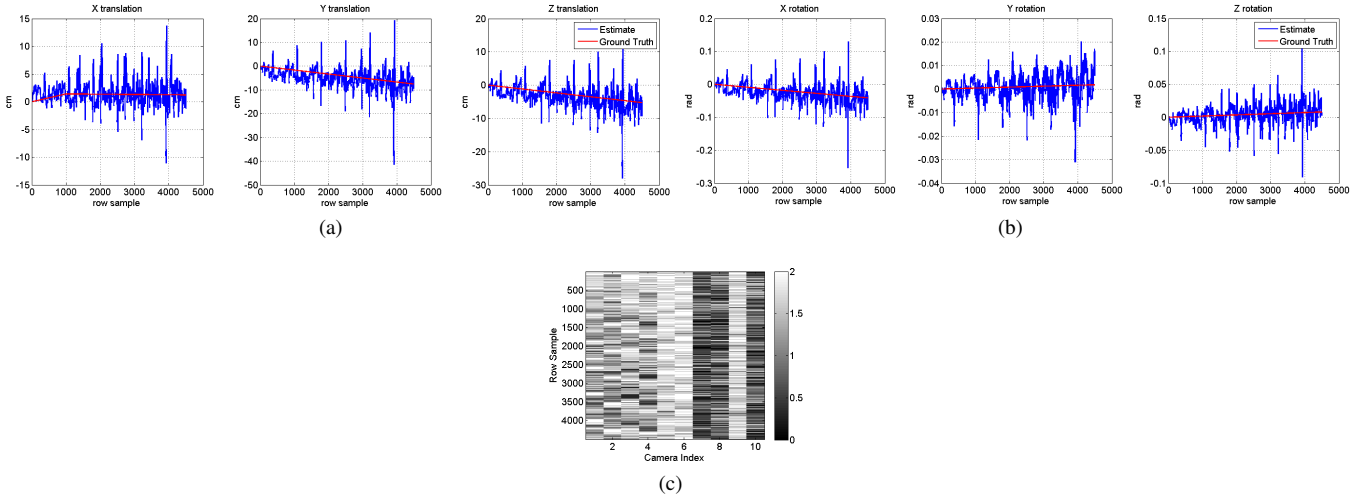


Fig. 13. (a) Translation, (b) Rotation, (c) Confidence score. The results are noisy which when smoothed give the correct results. Note the axis scales are not same. (Best viewed in color.)

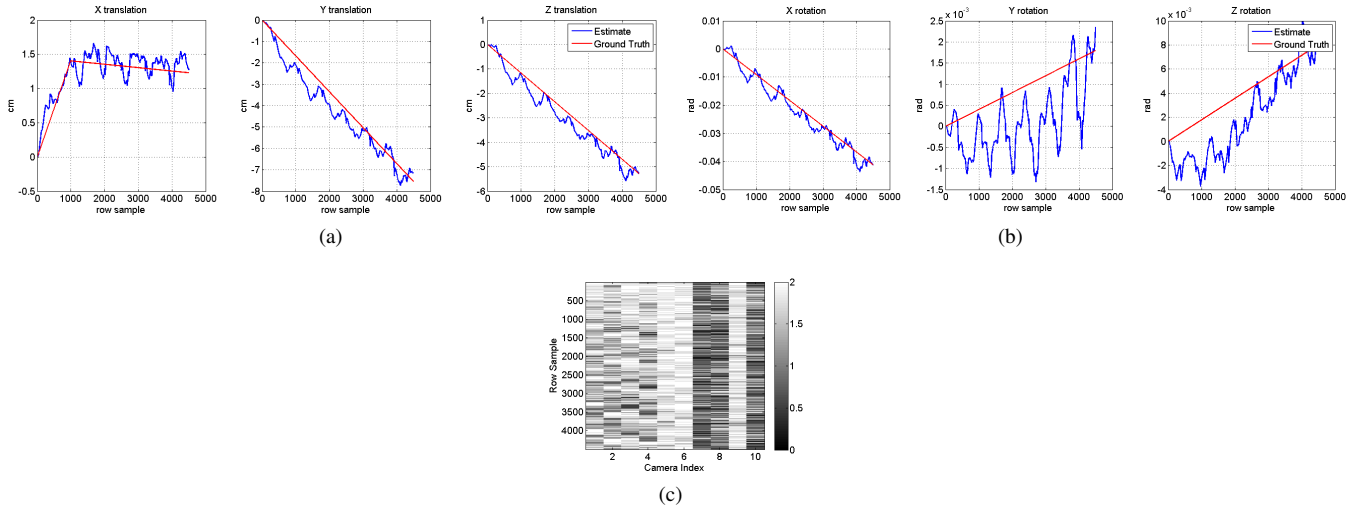
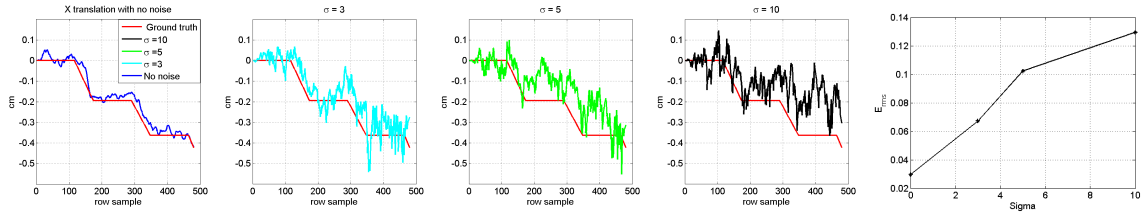
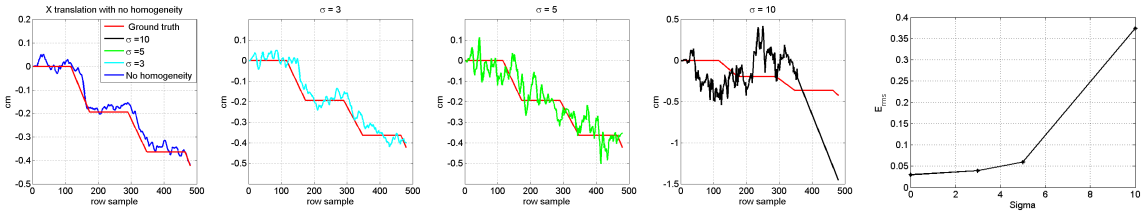


Fig. 14. (a) Translation, (b) Rotation, (c) Confidence score. Exponential smoothing is used to lower noise. Note the axis scales are not same. (Best viewed in color.)



(a) (b)

Fig. 15. Tracking results for translation in x direction with different levels of noise, and (b) corresponding RMS error.



(a) (b)

Fig. 16. Tracking results for translation in x direction with different levels of scene homogeneity, and (b) corresponding RMS error.

[10] J. Engel, J. Stueckler, and D. Cremers. Large-scale direct slam with stereo cameras. In *Intl. Conference on Intelligent Robots and Systems*, 2015.

[11] P.-E. Forssén and E. Ringaby. Rectifying rolling shutter video from hand-held devices. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 507–514. IEEE, 2010.

[12] C. Forster, M. Pizzoli, and D. Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 15–22. IEEE, 2014.

[13] C. Geyer, M. Meingast, and S. Sastry. Geometric models of rolling-shutter cameras.

[14] R. M. Haralick, C.-n. Lee, K. Ottenburg, and M. Nölle. Analysis and solutions of the three point perspective pose estimation problem. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 592–598. IEEE, 1991.

[15] J. Hedborg, P.-E. Forssén, M. Felsberg, and E. Ringaby. Rolling shutter bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1434–1441. IEEE, 2012.

[16] X. Hu and P. Mordohai. A quantitative evaluation of confidence measures for stereo vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2121–2133, 2012.

[17] P. S. Kalekar. Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi School of Information Technology*, 4329008:1–13, 2004.

[18] G. S. Klein and T. W. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776, 2004.

[19] R. K. Kumar, A. Ilie, J.-M. Frahm, and M. Pollefeys. Simple calibration of non-overlapping cameras with a mirror. In *Computer Vision and Pattern Recognition, 2008.*, pages 1–7. IEEE, 2008.

[20] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov. Head tracking for the oculus rift. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 187–194. IEEE, 2014.

[21] B. Li, L. Heng, K. Koser, and M. Pollefeys. A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 1301–1307. IEEE, 2013.

[22] M. Meilland, T. Drummond, and A. Comport. A unified rolling shutter and motion blur model for 3d visual registration. In *Proceedings of the IEEE Intl. Conference on Computer Vision*, pages 2016–2023, 2013.

[23] L. Naimark and E. Foxlin. Encoded led system for optical trackers. In *Proceedings of the 4th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 150–153. IEEE Computer Society, 2005.

[24] L. Oth, P. Furgale, L. Kneip, and R. Siegwart. Rolling shutter camera calibration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1360–1367, 2013.

[25] F. C. Park and B. J. Martin. Robot sensor calibration: solving ax= xb on the euclidean group. *IEEE Transactions on Robotics and Automation (Institute of Electrical and Electronics Engineers)*, 10(5), 1994.

[26] S.-F. Persa. *Sensor fusion in head pose tracking for augmented reality*. TU Delft, Delft University of Technology, 2006.

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

[28] J. P. Rolland, L. Davis, and Y. Baillot. A survey of tracking technology for virtual environments. *Fundamentals of wearable computers and augmented reality*, 1:67–112, 2001.

[29] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(1):105–119, 2010.

[30] O. Saurer, K. Koser, J.-Y. Bouguet, and M. Pollefeys. Rolling shutter stereo. In *Proceedings of the IEEE Intl. Conference on Computer Vision*, pages 465–472, 2013.

[31] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[32] T. Schöps, J. Engel, and D. Cremers. Semi-dense visual odometry for AR on a smartphone. In *International Symposium on Mixed and Augmented Reality*, September 2014.

[33] M. Shah, R. D. Eastman, and T. Hong. An overview of robot-sensor calibration methods for evaluation of perception systems. pages 15–20, 2012.

[34] S. Su and W. Heidrich. Rolling shutter motion deblurring. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1529–1537. IEEE, 2015.

[35] J. Ventura, C. Arth, G. Reitmayr, and D. Schmalstieg. Global localization from monocular slam on a mobile phone. *Visualization and Computer Graphics, IEEE Transactions on*, 20(4):531–539, 2014.

[36] G. Welch, G. Bishop, L. Vicci, S. Brumback, K. Keller, and D. Colucci. High-performance wide-area optical tracking: The hiball tracking system. *presence: teleoperators and virtual environments*, 10(1):1–21, 2001.

[37] B. Wilburn. *High performance imaging using arrays of inexpensive cameras*. PhD thesis, Citeseer, 2004.

[38] X. Zhang, S. Fronz, and N. Navab. Visual marker detection and decoding in ar systems: A comparative study. In *Proceedings of the 1st International Symposium on Mixed and Augmented Reality*, page 97. IEEE Computer Society, 2002.

[39] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Trans. on*, 22(11):1330–1334, 2000.

[40] F. Zhou, H. B.-L. Duh, and M. Billinghurst. Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 193–202. IEEE Computer Society, 2008.