2014 IEEE 11th Intl Conf on Ubiquitous Intelligence & Computing and 2014 IEEE 11th Intl Conf on Autonomic & Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Associated Symposia/Workshops

Mobiscan3D: A low cost framework for real time dense 3D reconstruction on mobile devices

Brojeshwar Bhowmick
Tata Consultancy Services
Email: b.bhowmick@tcs.com

Apurbaa Mallik
Tata Consultancy Services
Email: apurbaa.mallik@tcs.com

Arindam Saha
Tata Consultancy Services
Email: ari.saha@tcs.com

***Abstract*—In this paper we propose a computationally inexpensive framework for dense 3D reconstruction on smart device platforms leveraging the feed from motion sensors. In contrast to other methods, we solely rely on the motion sensors to compute pairwise epipolar relationships [1]. In particular, camera positions are obtained only through noisy mobile sensor data which is further optimized globally using iterative reweighted least squares. Rotations are also obtained using mobile sensors. Our method of obtaining camera motion reduce the processing time of the entire pipeline. We further use pairwise epipolar relationships along with normalized cross correlation and gradient information in a pair of images to obtain dense correspondences. The calibrations and correspondences are accurate enough for triangulation which in turn serve as a good initializer for final global bundle adjustment in near real time. Experimental results show that our method works reliably well for both indoor and outdoor scenes of different size and shapes without even utilizing mobile GPU.**

Keywords*—structure from motion, mobile sensors, iterative reweighted least square, dense 3D reconstruction.

I. Introduction

Structure from motion (SfM) is a well known technique in computer vision. Several approaches have been used to produce accurate 3D models from multiview. These works include approach by Snavely *et al.* in [2], which incrementally builds 3D stucture from a set of images. Wu *et al.* in [3], proposed a methodology which produces 3D structure from multiview in near linear time. Also, various industrial software have emerged in this domain such as Patch Based Multi-View Stereo [4] and Autodesk 123D [5] which are capable of producing accurate dense 3D structure of a scene from images. But, all of these methods require massive computing power.

With advancement in technology, mobile devices such as tablets and smart phone are equipped with more processing power and high end GPU's. Also, they have additional sensors like accelerometer, gyroscope and magnetometer which provide information for estimating the mobile rotation and position. By combining the sensor data with the images, a smart mobile device can be used as a quick scanner which can produce dense 3D model of a scene in real time. In this line of thought, Lee *et al.* propose a 3D model framework in [6] where the images are captured in the mobile and the modeling procedure is carried out on a remote server. Hartl [7] *et al.* propose an approach to generate a coarse 3D model of small objects on mobile CPU. The objects' dimension limit the applicability of their approach in general. Pan *et al.* carried out 3D reconstruction completely on mobile phone in [8]. But, due to sparse model no accurate structural information was obtained. Lin *et al.* in [9], utilize the inertial sensors available in the smart mobile devices for 6DoF pose estimation. But, due to simplistic approach, the sensor data solely could not provide accurate camera pose information. In a very recent work by Tanskanen *et al.* in [10], accelerometer and gyroscope embedded in mobile is used to estimate the motion parameters for dense 3D reconstruction. But, they used visual inference from images alongside with mobile sensors to determine the motion parameters, which unnecessarily increased the complexity of the reconstruction pipeline. To reduce the time they have to use the facility of GPU. They also used computationally expensive methods like 5-Point algorithm by Nistér [11] and RANSAC [12] in their framework for outlier filtering. Also, both [9] and [10] implement extended kalman filter to get noise free sensor output. But, we rely on iterative reweighted least square (IRLS) position averaging to achieve a better and accurate sensor output in this scenario. A detailed description is presented in Section III-C and comparison is provided in Table II.

Hence, we propose a framework for SfM which only utilize real time feed from mobile sensors for estimating camera motion and producing a dense 3D reconstruction of the scene. So, our method is computationally efficient for processing on mobile devices. Initial mobile devices were devoid of GPUs. Our method is capable to work on such devices as well and generate 3D output in real time. In [8], a multi core CPU based platform is used for SfM and it takes around 4-5 seconds per image. Method proposed in [10], takes 2-3 seconds per image in a CPU-GPU based platform. Our proposed method takes 2-3 seconds per image in a multi core CPU based implementation. The novelty of our method lies in the fact that we perform real time dense 3D reconstruction on smart devices using the CPU only. These dense 3D models can provide user mobility in diverse occasions which may include some scenarios like 3D room reconstruction requiring real time augmentation of furniture and real time damage analysis by insurance company during any vehicle accident.

The main contributions of our paper are

- Our method solely utilizes the mobile sensor feed to find an accurate estimate for rotation and initial estimate for position. The estimated positions are further optimized using iterative reweighted least squares position averaging.
- A computationally inexpensive epipolar geometry estimation from sensor data without using image correspondences.
- We use global optimization at the end which

978-1-4799-7646-1/14 $31.00 © 2014 IEEE
DOI 10.1109/UIC-ATC-ScalCom.2014.79

reduces computation thereby increasing the efficiency on smart mobile device.
- A dense 3D reconstruction framework which exploits the processing power of the CPU and run in parallel with other stages of the pipeline, thereby increasing the efficiency of our pipeline.

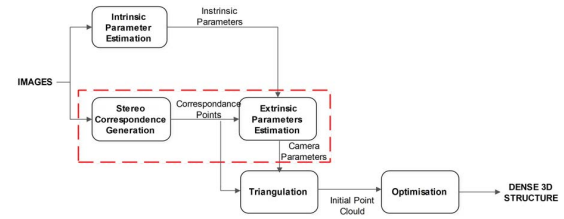
The rest of the paper is organized as follows. Section II describes about the pipeline of 3D reconstruction. Section III and Section IV describe the detail process of calibration and epipolar geometry respectively. Section V discusses about the dense reconstruction process. Section VI describes about our detail experiments and Section VII concludes the paper with discussions and future works.

II. 3D RECONSTRUCTION PIPELINE

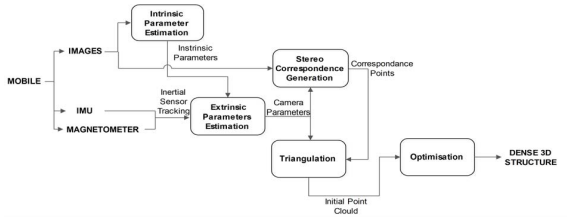
Any reconstruction pipeline follows the following three main stages

- Camera Calibration
- Epipolar Geometry Estimation
- Dense 3D Reconstruction

We aim to develop a light weight robust algorithm for each and every stage which can run in near real time on mobile devices. Figure 1 compares our proposed framework against familiar 3D reconstruction framework.



(a)



(b)

Fig. 1: (a) shows the familiar dense 3D reconstruction framework and (b) represents our proposed mobile based dense 3D reconstruction framework.

III. CAMERA CALIBRATION

For intrinsic calibration, we have collected several images with checkerboard pattern using mobile phone at different distances from an object. We notice that the intrinsic parameters of the mobile camera almost remain the same in all the situations. Therefore for a particular camera internal parameters are calibrated one time using the process described by Zhang *et al.* in [13]. The external parameters consisting of the orientation and position of the camera, are computed using mobile sensors.

A. Inertial Sensor Tracking

For estimation of the extrinsic camera parameters, we utilize mobile motion sensors such as accelerometer, gyroscope and magnetometer. The motion sensing unit contains three orthogonal accelerometers and gyroscopes which measure the linear acceleration and angular velocity of the device respectively. The mobile magnetometer estimates the unit vector towards the true magnetic north direction. The magnetometer output remains almost stable unless it encounters a strong magnetic field which happens rarely. The sampling rate of both mobile sensor and magnetometer are time-synced for temporal consistency. The combination of mobile accelerometer, gyroscope and magnetometer data is adequate to estimate the rotation of the mobile camera with respect to the world coordinate system. The accelerometer in conjunction with the gyroscope, provides the unit gravitational vector $\mathbf{r}_z$ along Z-Axis. Conventionally, the unit vector $\mathbf{r}_y$ towards our estimated north direction, is considered as the Y-Axis. The X-axis unit vector, $\mathbf{r}_x$ is calculated by taking the cross product between $\mathbf{r}_y$ and $\mathbf{r}_z$. The rotation matrix, R is a combination of $\mathbf{r}_x$, $\mathbf{r}_y$ and $\mathbf{r}_z$.

The image capture is initiated when the device is static. The displacement of the device from the previous position is made in such a way that the previous image has sufficient overlap with the current one. We utilize the accelerometer readings to understand whether the mobile is static. We also use a threshold on the displacement of the mobile from previous position to capture next image.

In static condition the acceleration of the device should be ideally zero. But, according to Woodman *et al.* [14], the accelerometer readings also contain a constant bias, which when double integrated can gives an error in position . Woodman *et al.* in [14], estimates the constant bias B of the accelerometer data to a certain extent by taking average of the accelerometer's output when it is in a static position as in Equation 1,

$$B = \frac{1}{N}\sum_{i=0}^{N}(\mathbf{a}_i - \mathbf{g}_i) \tag{1}$$

where, N denotes total number of samples of the sensor data when it is in static condition and $\mathbf{g}_i$, $\mathbf{a}_i$ are gravitational vector and acceleration of the mobile respectively for i^{th} sample.

For every sample sensor data, we get an acceleration vector which we can use to estimate camera positions. If Δt_i is the time interval between two readings of sensor data i.e $\Delta t_i = t_i - t_{i-1}$, then the final velocity $\mathbf{v}_i$ is

$$\mathbf{v}_i = \mathbf{u}_i + R_i(\mathbf{a}_i - \mathbf{g}_i - B)\Delta t_i \tag{2}$$

where $\mathbf{u}_i$ is the initial velocity and R_i is the rotation matrix of the mobile. In static condition, initial velocity of a device is taken as zero. For the next readings, the initial velocity is calculated as

$$\mathbf{u}_{i+1} = \mathbf{v}_i \tag{3}$$

Let, $\mathbf{s}_i$ be the displacement of the mobile in Δt_i which denotes the relative position of the mobile in t_i with respect to its position at t_{i-1}. It can be estimated as

$$\mathbf{s}_i = \mathbf{u}_i\Delta t_i + 0.5R_i(\mathbf{a}_i - \mathbf{g}_i - B)\Delta {t_i}^2 \tag{4}$$

Our method of estimation of position is comparable to dead reckoning where the position of an object is predicted based on its last known position, velocity and acceleration [15]. But, pose estimation with only dead reckoning using mobile sensors is quite erroneous due to noisy sensor data. So, we used IRLS in addition to remove the errors.

B. Camera Position Estimation From Inertial Informations

Due to environmental and sensor effect, accelerometer readings get noisy. Some of these noises are white noise, residual bias and flicker noise [14]. So, camera positions need to be estimated more accurately from these noisy data. In our paper, the term *camera position* indicates the device

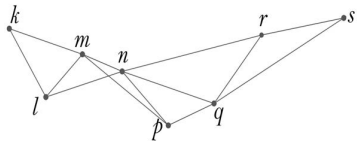


Fig. 2: This is a viewgraph of the initial camera positions. The vertices denote the camera positions, taken in alphabetical order.

position where an image is captured. Using Equation 4, we can determine displacement of the device from time t_{i-1} to t_i. This can be extended for determining the relative camera positions. Let us consider the estimation of relative position between k^{th} and l^{th} camera at time t_k and t_l respectively. For every $\triangle t_{k+w}$, we compute the relative position of the device with respect to its position in t_{k+w-1} by applying Equation 4, where w=1,2,3...l. So, relative position of l^{th} camera from k^{th} camera, $\mathbf{c}^a_{kl}$ is estimated by adding these piecewise position vectors. As its a relative camera position, it treats the first camera as its origin. So, initial velocity $\mathbf{u}_k$ is initialized to zero. Figure 3 shows comparison of pairwise positional information along $Y-axis$ for all edges similar to $\mathbf{c}^a_{kl}$ against ground truth $\mathbf{c}^g_{kl}$. Also, Figure 4 shows the estimation of global camera position $\mathbf{c}^a_k$ using Equation 4. In this case we consider the initial velocity of every vertex as Equation 3. It is evident from both the figures that the estimations only using accelerometer deviates in considerable amount from the actual values.

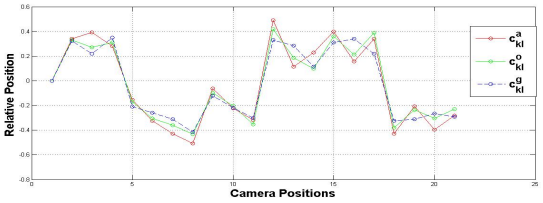


Fig. 3: The points represent relative position of a camera with respect to its previous camera for Y-axis data. The ground truth of relative camera positions, $\mathbf{c}^g_{lk}$ are in blue, initial estimated relative camera positions from inertial sensors, $\mathbf{c}^a_{lk}$ are in red and after position averaging, the relative positions, $\mathbf{c}^o_{lk}$ are in green.

We form a viewgraph with the estimated relative camera positions using $\mathbf{c}^a_{kl}$ as shown in Figure 2. Let us consider a triangle formed using k, l and m in the viewgraph at time t_k, t_l, t_m respectively, where $t_k < t_l < t_m$. We have already

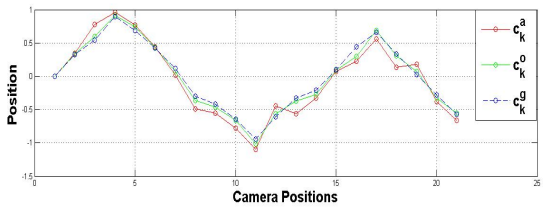


Fig. 4: The points represent global position of the cameras for Y-axis data. The ground truth of absolute camera positions, $\mathbf{c}^g_k$ are in blue, initial estimated global camera positions from inertial sensors, $\mathbf{c}^a_k$ are in red and after position averaging, the global positions, $\mathbf{c}^o_k$ are in green.

computed relative camera positions $\mathbf{c}^a_{kl}$ and $\mathbf{c}^a_{lm}$ using sensor informations taking $\mathbf{u}_k = 0$ and $\mathbf{u}_l = 0$ respectively. It should be noted that the accelerometer provides continuous noisy feed and so, the acceleration at time l is not zero. So, $\mathbf{c}^a_{km}$ is computed in the following two way,

M1: $\mathbf{c}^a_{km} = \mathbf{c}^a_{kl} + \mathbf{c}^a_{lm}$. This is vector addition considering $\mathbf{u}_l = 0$.

M2: $\mathbf{c}^a_{km}$ is calculated using Equation 4 by taking $\mathbf{u}_l \neq 0$ and $\mathbf{u}_k = 0$, i.e here we do not consider the acceleration at the intermediate vertex l to be zero.

TABLE I: Comparison of method $M1$, $M2$ and optimization using IRLS against ground truth for pairwise displacement of cameras in view graph.

Ground Truth (cm)	**M1** (cm)	**M2** (cm)	**Optimized Position** (cm)
15.00	16.69	14.23	15.60
20.00	22.44	17.36	18.25
28.00	25.23	28.86	27.67
35.00	34.92	38.16	36.06
36.00	37.75	33.63	35.86
40.00	41.62	38.03	40.24

In method $M2$, the resultant acceleration after static bias correction at l is very small. But, the estimation of all such edge similar to $\mathbf{c}^a_{km}$ over the entire view graph exhibits that sometimes method $M1$ is more closer to ground truth and sometimes method $M2$ is more closer. This is demonstrated in Table I where we compute the length of all edges similar to $\mathbf{c}^a_{km}$, i.e $||\mathbf{c}^a_m - \mathbf{c}^a_k||$ for all such vertex. Table I, shows the behavior of method $M1$ and $M2$ on different edges which clearly shows both the methods is closer to ground truth for different vertices. So, we can have two different ways of estimation for all vertex similar to m using $M1$ and $M2$.

C. Iterative Reweighted Least Square Position Averaging

For every triangle Δlkm in viewgraph, we compute $\mathbf{c}^a_{lk}$ and $\mathbf{c}^a_{km}$ considering $\mathbf{u}_l = 0$ and $\mathbf{u}_k = 0$ respectively. Position $\mathbf{c}^a_m$ with respect to $\mathbf{c}^a_l$ i.e $\mathbf{c}^a_{lm}$ is computed using method $M2$. If $\mathbf{c}^o_k$ and $\mathbf{c}^o_l$ are the true global positions of the k^{th} camera and l^{th} camera respectively then, we can write,

$$\mathbf{c}^o_l - \mathbf{c}^o_k = \mathbf{c}^a_{kl} \tag{5}$$

Equation 5 holds for every such pair in view graph and can be rearranged as $A\mathbf{x} = \mathbf{b}$ to pose it as linear least

square problem, where $A \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{x} \in \mathbb{R}^n$ and $m, n > 0$ are positive integers. But, in reality an unknown noise component $\boldsymbol{\eta}$ is present in the above system of equations. So, it can be rewritten as $A\mathbf{x} = \mathbf{b}+\boldsymbol{\eta}$. This noise component $\boldsymbol{\eta}$ has an unequal variance. We minimize the error $\boldsymbol{\eta} = A\mathbf{x} - \mathbf{b}$ to get an optimum value of $\mathbf{x}$. As, *l1* minimization is more robust to outliers, an initial estimation of $\mathbf{x}$ is calculated using *l1* minimization and then feed to *l2* averaging.

$$\min_{\mathbf{x}} \|A\mathbf{x} - \mathbf{b}\|_1 \tag{6}$$

We use the solution of $\mathbf{x}$ obtained from Equation 6, to iteratively estimate accurate camera positions using a concept given by Holland *et al.* in [16] called iterative reweighted least square (IRLS). Chatterjee *et al.* in [17], used this concept to find out the global camera rotations when pairwise rotations in a viewgraph are known.

We have modeled the noise in the system, $\boldsymbol{\eta}$ using Pseudo-Huber loss function [1], where the loss value, $L(\|\boldsymbol{\eta}\|) = 2\beta^2(\sqrt{1+(\frac{\boldsymbol{\eta}}{\beta})^2} - 1)$. Here β is a constant that defines an outlier threshold.
IRLS finds the value of $\mathbf{x}$ such that it minimizes the total error of the considered system, $Er = \sum_i L(\|\boldsymbol{\eta}_i\|)$ where $\boldsymbol{\eta}_i$ is the i^{th} element of the error vector. So, optimum value for $\mathbf{x}$ can be calculated as,

$$\begin{aligned} &\tfrac{d}{dx}(Er) = 0 \\ &\Rightarrow \tfrac{d}{dx}(\textstyle\sum_i L(\boldsymbol{\eta}_i)) = 0 \\ &\Rightarrow A^T W(\boldsymbol{\eta})\mathbf{b} = 0 \end{aligned} \tag{7}$$

where $W(\boldsymbol{\eta})$ is a diagonal matrix and every i^{th} diagonal element of the matrix is assigned a weight of $\frac{\boldsymbol{\eta}_i\beta}{(1+(\frac{\boldsymbol{\eta}_i}{\beta})^2)^2}$. It can be shown that the optimum value of $\mathbf{x}$ can be estimated as $(A^TWA)^{-1}AW\mathbf{b}$. If calculated initial position of a camera is erroneous, then its loss value decreases with every iteration. In case the loss value goes below a certain threshold then its corresponding position is excluded from the system of equations.

Figure 4 shows that the positions obtained from optimization are quite nearer to the ground truth in comparison to raw accelerometer data. Also, pairwise camera positions are again calculated using the optimized positions and Figure 3 shows that the newly estimated values are much more closer to the ground truth. Table I also exhibit that the values of the new camera positions are quite good as the pairwise distances between camera positions improved.

In Table II, we provide a comparison between the position estimation using extended kalman filter [18] and IRLS position estimation. In extended kalman filter, the optimum estimate is derived using history and measurement. But, IRLS position averaging is a global optimization technique taking all the data into account and hence provides more robust results in our scenario. Also, kalman filter works well with higher number of samples, which we may not get in all situations. In the triangulation step, the reprojection error obtained using the kalman filter is comparatively higher with respect to our method. So, it introduces more number of iterations in the final bundle adjustment thereby increasing its computation time.

TABLE II: Comparison of Sensor Output using method $M1$, Extended Kalman Fiter and IRLS Position Averaging against Ground Truth for pairwise displacement of cameras.

Ground Truth (cm)	**Sensor Output** (cm)	**Extended Kalman Filter** (cm)	**IRLS Position Averaging** (cm)
26.0	25.3	25.4	26.6
28.0	27.1	26.4	27.6
30.0	36.5	35.2	30.2
32.0	35.9	34.9	32.4
34.0	41.3	39.2	35.1
40.0	44.2	40.8	40.2

IV. Epipolar Geometry Estimation

Let $\mathbf{t}_{kl}$ and R_{kl} be the relative translation and rotation between k^{th} and l^{th} camera respectively then,

$$\begin{aligned} \mathbf{t}_{kl} &= -R_l(\mathbf{c}_l^o - \mathbf{c}_k^o) \\ R_{kl} &= R_l R_k^{-1} \end{aligned} \tag{8}$$

where R_k is the global rotation of the k^{th} camera estimated from mobile sensors and $\mathbf{c}_k^o$ is the optimized global position of the camera. Fundamental matrix algebraically represents the epipolar geometry. It can be estimated from camera parameters as [1]

$$F_{kl} = K^{-T}[\mathbf{t}_{kl}]_\times R_{kl} K^{-1} \tag{9}$$

where F_{kl} represents the fundamental matrix between k^{th} camera and l^{th} camera. K is the internal calibration matrix of camera. The fundamental matrix is used in later sections for identifying outlier free stereo correspondences between image pairs.

V. Dense 3D Reconstruction

A. Feature Extraction And Mapping

An essential step in computing 3D structure is finding out the corresponding points between two or more images. We aim for quick and dirty 3D reconstruction in real time using such correspondences. But, finding stereo correspondences may be troublesome due to presence of homogeneous texture, varying lighting condition etc. We rely on combination of colour and gradient information to obtain the correspondences to a certain extent.

We extract features from every image of the scene using the method described by Shi *et al.* [19]. Given a pair of images say I_A and I_B, a matching is initiated by comparing pixel blocks in I_B with the pixel block in I_A around a feature. In our method, the block size is set to 7×7 pixels. The blocks from image I_B are chosen along the epipolar line corresponding to the point in I_A. For every feature $\mathbf{x}_A$ in I_A we search $\mathbf{x}_B$ in I_B in a region $\{\mathbf{x}_A - \vec{\delta}, \mathbf{x}_A + \vec{\delta}\}$ satisfying Equation 10.

$$d(\mathbf{x}_A, l_A) + d(\mathbf{x}_B, l_B) < Th \tag{10}$$

where l_A is the epipolar line in I_A corresponding to point $\mathbf{x}_B$ in I_B, and $d(\mathbf{x}_A, l_A)$ is the perpendicular distance of $\mathbf{x}_A$ from l_A. More description can be found in [1]. If the perpendicular distance of $\mathbf{x}_A$ from its epipolar line in I_B is

greater than some threshold then we do not consider it to be a correspondence. For a point $\mathbf{x}_A$ in I_A, once we localise a region for correspondence around the epipolar line in I_B, we search for every block in that region for matching using color and gradient. In general, color information in conjunction with gradient can be used as a good cost function for establishing correspondences. We use normalized cross correlation (NCC) and gradient field technique proposed by Scharstein *et al.* [20] to design our cost function. Let us consider the matching between two pixel blocks $I_A(x, y)$ and $I_B(x + \delta x, y + \delta y)$ of I_A and I_B respectively where (x, y) is a feature point in I_A and $(\delta x, \delta y)$ is the displacement in I_B. Then, for n number of pixels in the pixel block, the cost function can be computed as in Equation 11.

$$\begin{aligned} E(x,y) &= \tfrac{1}{n}\sum_{x,y}\tfrac{1}{2}(|G_A(x,y)| + |G_B(x+\delta x, y+\delta y)|) \\ D(x,y) &= \tfrac{1}{n}\sum_{x,y}|G_A(x,y) - G_B(x+\delta x, y+\delta y)| \\ C(x,y) &= NCC(x,y) + \lambda(E(x,y) - D(x,y)) \end{aligned} \tag{11}$$

where $G_A(x, y)$ and $G_B(x + \delta x, y + \delta y)$ denotes the gradient vector field of pixel block $I_A(x, y)$ and $(I_B(x + \delta x, y + \delta y)$ respectively. $E(x, y)$ is a function of gradient field magnitude and $D(x, y)$ contains information about the gradient field directions. The cost function, $C(x, y)$ is a linear combinations of NCC and gradient field. λ is the tuning parameter.
For every point $\mathbf{x}_A$ in I_A, we first find the probable region of finding correspondence in I_B through epipolar constraint using Equation 10. Once we find a region we examine each block in that region which provides maximum value for $C(x, y)$. The point $\mathbf{x}_B$ is correspondent with point $\mathbf{x}_A$ if $C(x, y)$ is maximum for block around $\mathbf{x}_B$ and block around $\mathbf{x}_A$. As the features are independent of the each other, so the process can be made parallel.

These methods of finding correspondences is very simplistic and little noisy. But to have a real-time system in mobile phone we adopt them at the cost of marginal degradation in the quality of dense reconstruction.

B. Global optimized Dense Modeling

Once correspondences are generated we calculate initial 3D points by a well known triangulation method proposed by Hartley *et al.* [21] using the global camera parameters and filtered N-view correspondences. At the final step of our 3D reconstruction method we simultaneously optimize the dense point cloud and cameras parameters using global bundle adjustment. Recent advancement in multicore bundle adjustment [22] allows us to use multiple CPUs efficiently.

VI. Experimental Results

In our experiments, we have used LG Nexus 5 smartphone having Quad-core 2.3 GHz Krait 400 CPU and 2GB RAM. We have evaluated our methods on different indoor and outdoor dataset which do not exhibit any regular geometric shape. In our experimental setup, images with 640×480 resolution are captured at an interval of 1.2-1.5 sec. Sampling rate for accelerometer, gyroscope and magnetometer is 100 Hz. The entire reconstruction pipeline runs in parallel only on the CPU of mobile devices.

TABLE III: Timing details in seconds.

No of Images	**Camera Calibration**	**Triangulation** (No Of Points)	**Optimization** (No Of Points)	**Effective Reconstruction time**
4	0.000598	7.464(112394)	3.6445(90501)	11.678
5	0.0009425	10.8(161254)	5.1225(115321)	16.392
8	0.0021515	13.57(210410)	7.89635(153257)	21.637
10	0.003263	14.25(225472)	10.6014(170050)	24.823
20	0.007176	18.92(281175)	22.904(250142)	42.7

We start the processing of an image by extracting its features immediately after it is captured. The feature extraction of an image and correspondence generation between a image pair typically takes about 0.169 sec and 0.7 sec on average respectively. Thus, the next image capture process do not get delayed due to processing of previous images. We are able to calibrate the camera as described in Section III after obtaining all images. So, there is a marginal delay on getting the output on screen. Among them the key operations that takes major amount of time are triangulation and optimization. Our calibration hardly takes any time (on the order of microsecond) and feature extraction can be performed during data capture process. Table III shows the timing details of the processes that executes for structure estimation. Equivalently, this is the waiting time for user to get displayed the 3D structure on screen. It is clear from the table most of the reconstruction including 360 degree view for which we have used 20 images, can be done within a minute. It should be noted that we only utilize the CPU of a mobile. Utilizing a GPU can bring down the required time considerably.

The time required is mostly dependent on the number of images and number of points we try to reconstruct. Number of images and the size of objects determine the number of points in general. We evaluate of our algorithm with different sizes of object in both indoor and outdoor. Figure 5 shows the dense 3D model of an outdoor object of height 1.87 meters in general illumination condition. We also evaluate our algorithm in some indoor dataset. Figure 6 shows the output of an object having height 0.12 meters in indoor environment. We also attempt to reconstruct 360 degree view. Figure 7 shows an output of a 360 degree human body of height 1.554 meters, with 20 images. It should be noted that the back side of the body is not well reconstructed. This is due to absence of enough images at the back. Our system is simple and can be used for reconstructing any scene in general. However, if the images contain much illumination difference or too much homogeneity then our output contains considerable noises due to the simplistic algorithm for dense reconstruction.

VII. Conclusion and Discussion

We present a new on device system for automatic real time 3D reconstruction. The entire system runs only on CPU of smart phone. So, this is the first on mobile CPU only system for 3D reconstruction. In order to make the process in near real time, an entirely new approach is adopted to estimate the calibration parameters of all cameras. Specifically, only accelerometer, magnetometer and gyroscope sensors are used for camera calibration. Dense point correspondences are obtained from good feature. Triangulated 3D points are optimized using bundle adjustment. The usage of these camera

sensors along with the quad core processor make the entire system on near real time. The algorithm does not have any constrain on shape, size or lighting condition of the structure, so this work on all situation ideally. Results of different outdoor and indoor outputs are the supportive of the claim. However, development of a more accurate and noise free dense reconstruction in realtime could be an immediate future work for this framework.

(a) (b) (c)

Fig. 5: (a) is an outdoor image where object size is 1.87 meters. (b) and (c) are the front and side view of the dense 3D model.

(a) (b) (c)

Fig. 6: (a) is an indoor image where object size is 0.12 meters. (b) and (c) are the front and side view of the dense 3D model.

REFERENCES

[1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[2] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: exploring photo collections in 3d," *ACM transactions on graphics (TOG)*, vol. 25, no. 3, pp. 835–846, 2006.

[3] C. Wu, "Towards linear-time incremental structure from motion," in *3DTV-Conference, 2013 International Conference on*. IEEE, 2013, pp. 127–134.

[4] Y. Furukawa and J. Ponce, "Patch-based multi-view stereo software," http://www.di.ens.fr/pmvs/.

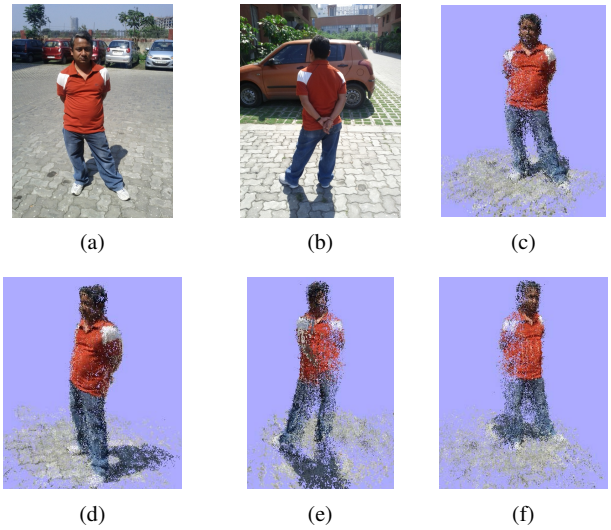
[5] AUTODESK, "Autodesk 123d," http://www.123dapp.com/.

[6] W. Lee, K. Kim, and W. Woo, "Mobile phone-based 3d modeling framework for instant interaction," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1755–1762.

[7] A. Hartl, L. Gruber, C. Arth, S. Hauswiesner, and D. Schmalstieg, "Rapid reconstruction of small objects on mobile phones," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 20–27.

[8] Q. Pan, C. Arth, G. Reitmayr, E. Rosten, and T. Drummond, "Rapid scene reconstruction on mobile phones from panoramic images," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*. IEEE, 2011, pp. 55–64.

[9] L. Chai, W. A. Hoff, and T. Vincent, "3-d motion & structure estimation using inertial sensors and computer vision for augmented reality," Master's thesis, advisors: William A. Hoff and Tyrone Vincent, Div. of Engineering.Colorado School of Mines, 2000.



(a) (b) (c)

(d) (e) (f)

Fig. 7: Dense 3D model providing 360 degree view of a human of height 1.554 meters. (a) and (b) is the front and back side images. (c), (d), (e), (f) are the different views of the 3D model.

[10] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys, "Live metric 3d reconstruction on mobile phones," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 65–72.

[11] D. Nistér, "An efficient solution to the five-point relative pose problem," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 6, pp. 756–770, 2004.

[12] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[13] Z. Zhang, "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 11, pp. 1330–1334, 2000.

[14] O. J. Woodman, "An introduction to inertial navigation," *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, vol. 14, p. 15, 2007.

[15] C. Murphy, "Believable dead reckoning for networked games," in *Game Engine Gems 2*, E. Lengyel, Ed. A K Peters, 2011, pp. 307–328.

[16] P. W. Holland and R. E. Welsch, "Robust regression using iteratively reweighted least-squares," *Communications in Statistics-Theory and Methods*, vol. 6, no. 9, pp. 813–827, 1977.

[17] A. Chatterjee and V. M. Govindu, "Efficient and robust large-scale rotation averaging," in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 521–528.

[18] G. A. Einicke and L. B. White, "Robust extended kalman filtering," *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2596–2599, 1999.

[19] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*. IEEE, 1994, pp. 593–600.

[20] D. Scharstein, "Matching images by comparing their gradient fields," in *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1. IEEE, 1994, pp. 572–575.

[21] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[22] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, "Multicore bundle adjustment," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3057–3064.