Glimpse.3D: A Motion-Triggered Stereo Body Camera for 3D Experience Capture and Preview

Bashima Islam
UNC at Chapel Hill
bashima@cs.unc.edu

Md Tamzeed Islam
UNC at Chapel Hill
tamzeed@cs.unc.edu

Shahriar Nirjon
UNC at Chapel Hill
nirjon@cs.unc.edu

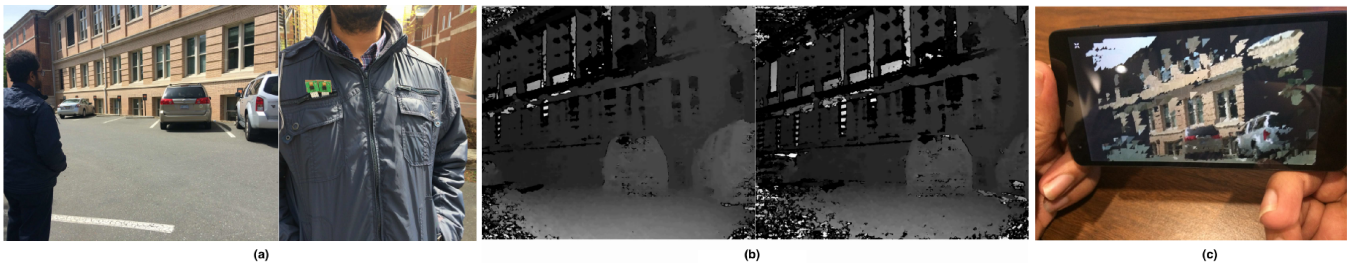


Figure 1: Glimpse.3D in action: (a) A mobile user is wearing Glimpse.3D which is taking stereo images of a scene containing parked cars in front of a building. (b) The bodycam captures and processes stereo images to generate disparity maps, and a 3D point cloud is created inside the camera unit. (c) A smartphone pulls the point cloud and renders it on the screen. We have not applied any post processing step such as meshing or texture mapping on this point cloud–which will improve the rendered image further, but at an additional cost.

ABSTRACT

The Glimpse.3D is a body-worn camera that captures, processes, stores, and transmits 3D visual information of a real-world environment using a low cost camera-based sensor system that is constrained by its limited processing capability, storage, and battery life. The 3D content is viewed on a mobile device such as a smartphone or a virtual reality headset. This system can be used in applications such as capturing and sharing 3D content in the social media, training people in different professions, and post-facto analysis of an event. Glimpse.3D uses off-the-shelf hardware and standard computer vision algorithms. Its novelty lies in the ability to *optimally control camera data acquisition and processing stages to guarantee the desired quality of captured information and battery life.* The design of the controller is based on extensive measurements and modeling of the relationships between the linear and angular motion of a body-worn camera and the quality of generated 3D point clouds as well as the battery life of the system. To achieve this, we 1) devise a new metric to quantify the quality of generated 3D point clouds, 2) formulate an optimization problem to find an optimal trigger point for the camera system that prolongs its battery life while maximizing the quality of captured 3D environment, and 3) make the model adaptive so that the system evolves and its performance improves over time.

CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing**; • **Computer systems organization** → **Embedded systems**; • **Hardware** → **Power and energy**;

KEYWORDS

3D-Reconstruction, Body Camera

1 INTRODUCTION

We are entering an era when 3D experience is going to be an integral part of our digital lives. Capturing real-world experiences in 3D brings us one step closer to bringing the digital world to life. We have already begun to see an increased use of 3D technology in medical imaging, training, crime scene documentation, robotics, city planning, cultural heritage preservation, and industrial quality control and measurements [48]. In coming days, the rich, realistic, and immersive experience offered by 3D technology will enable many applications. Of particular interest to us are the scenarios in which a person wants to captures his experience in 3D, while he is mobile and possibly outdoors, and then share the 3D contents with his friends in the social media.

For example, a person visiting a place like the Washington Monument or the great Grand Canyon may want to capture his/her experience in 3D and share it with others. An archaeologist may want to share his/her expedition experience so that others can feel the thrill when they watch the 3D content on their mobile devices. A home buyer may want to capture the interior of his/her soon-to-be home in 3D and share it with social media friends for comments before signing the deal. In all cases, besides capturing, the user naturally would want to preview the 3D content prior to sharing—just like we preview images before sharing them on platforms like Facebook or Instagram.

To realize a system like the above, we aim at developing a low-cost and energy-efficient wearable body camera system that generates 3D point clouds in near real-time—which are previewed on a player application running on a mobile phone. As a design decision, we choose to use a wearable platform over a smartphone-only solution since it provides a user with hands free experience. Furthermore, not all smartphones come with a built-in stereo camera that could be leveraged to perform a 3D reconstruction. Developing

a wearable system like this, however, is extremely challenging. Before we introduce our solution, let us visit some of these challenges in brief:

• A naive way to capture 3D experiences would be to store and/or stream images taken with stereo cameras or a Kinect-like RGBD camera, and then process the images offline to reconstruct the 3D scene. This is, in fact, how 3D capturing is mostly done today. However, the process is inefficient, unintelligent, power-hungry, costly, and in some cases, ineffective (e.g., Kinect only works indoors). We argue that neither storing nor streaming images is efficient in terms of storage and battery life as images from camera sensors contain up to 8X more redundant information than what is required to create a 3D model of a scene. For example, an ideal point cloud generation and merge process requires about 50% overlap between two images [60]. A greater or less overlap results in a drop in the generated 3D point cloud's quality. Therefore, a key to efficient 3D reconstruction is to be able to predict the right images to capture, which would help us generate the best quality 3D point clouds using the minimum number of captured images.

• Even if we can accurately predict the images to be taken for a high quality 3D reconstruction, the task of reconstruction on an embedded device still remains a significant challenge. Today, a high-quality 3D reconstruction requires several minutes (for low density point clouds) [37] to several days (for high density point clouds) [40] in high-end computing devices. Therefore, a dense 3D reconstruction on a battery-powered embedded device is out of the question. Hence, in order to make 3D point cloud generation possible on a resource-constrained device within a reasonable time, we must fine tune the parameters of the standard 3D reconstruction pipeline, so that the system operates at an optimal point which balances the quality of generated point clouds, the battery life, and the time to generate the cloud.

In this paper, we address these challenges and propose Glimpse.3D, which is a body-worn camera that captures, processes, stores, and transmits 3D visual information of a real-world environment in the form of a sparse 3D point cloud. These point clouds are then pulled by a mobile device such as a smartphone or a virtual reality headset to render a dense 3D point cloud of the whole scene for visualization. The end-to-end process, i.e. from the capturing to previewing, happens in near real-time. Glimpse.3D employs off-the-shelf hardware components and standard computer vision algorithms to capture stereo images and generate 3D point clouds. The novelty of Glimpse.3D, therefore, lies not in the computer vision aspect of it, but in the way it controls the execution of these algorithms.

At the heart of Glimpse.3D is a controller that optimizes the trigger-point of the image capture and processing tasks which guarantees a minimum quality of captured 3D point clouds and battery life of the embedded system. To achieve this, we devise three algorithms that we highlight as the three main contributions of the paper:

• We define a quality metric for 3D point clouds using their *structural similarity index* (SSIM) [64]. Since this metric is computationally expensive, we model it as a function of variables, whose values are obtained as a byproduct of a step in the 3D reconstruction pipeline. This makes it possible to quantify the quality of a point cloud in constant time.

• We formulate an optimization problem that finds an optimal trigger-point for the body camera to prolong its battery life while maximizing the quality of captured 3D environment. The outcome of this optimization is a pair of displacement values (linear and angular) that determines when to trigger the image capture and processing tasks to satisfy the minimum lifetime and quality constraints.

• We devise a self-adaptation mechanism for the camera controller so that it learns and evolves over time as it experiences new scenes and generates more point clouds. The goal of this module is to monitor the performance of the controller, update the model parameters, and readjust the optimal displacement values so that the system becomes precise in terms of maintaining the quality of 3D information and the desired battery life.

Glimpse.3D is different than existing techniques such as simultaneous localization and mapping (SLAM) [47], visual inertial odometery (VIO) [38], and IMU enhanced bundle adjustment [1]. Since we are only interested in the 3D reconstruction/mapping, and not strictly in positioning the user in a topology, SLAM is not going to be as efficient as Glimpse.3D for the task. Additionally, the frame rate required to run SLAM is much higher than what our embedded platform could handle. Although we use an IMU, our usage is different from both VIO and IMU-enhanced bundle adjustment. VIO uses IMU to correct its vision-based odometery (distance). Similarly, IMU-enhanced bundle adjustment helps improve the quality 3D reconstruction. However, in both cases, the use of IMU is after the reconstruction, whereas we use IMU to decide when to take pictures, and thus save a great amount of unnecessary image captures and processing.

2 BACKGROUND

2.1 Stereo 3D Reconstruction Pipeline

Stereo 3D reconstruction is the process of extracting 3D information from two digital images. The basic idea of stereo 3D reconstruction is to find corresponding points in the two input images and to estimate the depth of those points from their relative positions in each image. The 3D reconstruction pipeline is shown in Figure 2. First,

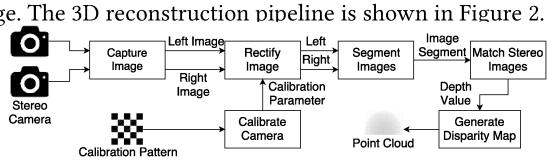


Figure 2: Stereo 3D reconstruction pipeline.

two input images are rectified to reduce distortions [2, 53] using camera calibration [15, 20, 23, 55, 69]. Then they are segmented into groups of similar pixels and corresponding points are obtained with stereo matching algorithms [24, 30] to create a disparity map. This map represents the offset of two corresponding points calculated by the triangulation method [13]. Finally, a 3D point cloud is generated from the disparity map.

A 3D point cloud is a set of data points in a 3D coordinate system defined by their x, y, and z coordinates and RGB color components. Based on the number of points per unit volume, a point cloud is either dense or sparse. A sufficiently dense 3D point cloud represents the external surface of objects. Point clouds often undergo through a triangulation phase to create meshes.

2.2 Merging Point Clouds

A pair of stereo images results in a single 3D point cloud. In order to capture a complete scene and objects within it , multiple point clouds (from different position and orientation of the camera) are merged together. This process is called mapping. For an effective merging, two point clouds need to overlap partially with each other. To merge two point clouds, we use *iterative closest point (ICP)* [34]. It solves the *surface registration* problem, transforming the second point cloud to the coordinate system of the reference (first) point cloud using the point to point correspondences between point clouds. Finally, the world scene is created from registered point clouds.

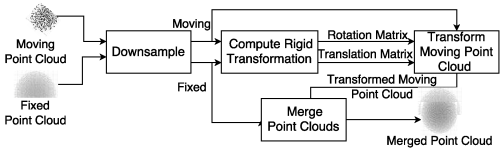


Figure 3: Steps of point cloud merging operation.

Figure 3 shows the steps of merging two point clouds using the ICP algorithm. Down-sampling removes noise from data and increases the accuracy of the algorithm. Rigid transformation matches points between the two point clouds using a kd-tree or the k-nearest neighbor algorithm. Incorrect matches are removed using an outlier detection process and the rotation and translation matrices are calculated from the inlier points. A transformation is applied on the moving (second) point cloud. Finally, the down-sampled fixed (first) point cloud and the transformed moving (second) point cloud are combined to form a large aggregated point cloud.

3 OVERVIEW AND CHALLENGES

This section provides an overview of *Glimpse.3D* and introduces three technical challenges that this paper addresses.

3.1 Overview of Glimpse.3D

Glimpse.3D is a body-worn camera that captures, processes, stores, and transmits 3D visual information of a real-world environment in the form of a sparse 3D point cloud. These point clouds are then pulled by a mobile device such as a smartphone or a virtual reality headset to render a dense 3D point cloud of the whole scene for visualization.

The purpose of Glimpse.3D is to enable 3D experience capture in real-world mobile situations using low-cost camera-based sensor systems that are constrained by the limited processing capability, storage, and battery life. The system is suitable for applications where a user wants to capture a scene in 3D and there is a need for a quick preview of the 3D content. The current implementation of Glimpse.3D assumes a static target scene, but the user wearing the system can be either stationary or moving on feet. The body camera captures and processes stereo images to generate a 3D point cloud inside the camera unit. As the person moves, new points are added to the point cloud and it grows incrementally. The point cloud is pulled by a smartphone and is rendered on the screen.

3.2 Processing Inside a Body Camera Unit

Each body camera unit contains a pair of low cost cameras for capturing stereo images, an inertial measurement unit (IMU) for estimating linear and angular displacements, a micro-controller for data processing and control, a wireless module for communication with a smartphone, and a battery.

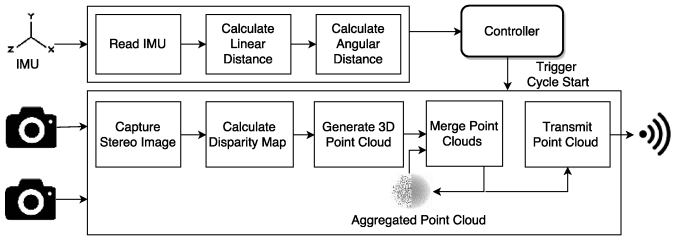


Figure 4: Each body camera unit consists of a pair of cameras, an IMU, a MCU, and a radio.

Figure 4 shows the components inside a single Glimpse.3D camera unit. There are three major subsystems: 1) image processing subsystem, 2) IMU data processing subsystem, and 3) a controller. The first two subsystems implement standard algorithms for image and IMU data processing, which are tuned to fit the constraints of the proposed system. The last component, the controller, which is one of the main contributions of this paper, is introduced in this section and elaborated in the next section.

• *Image Processing Subsystem.* The image processing subsystem is triggered by the controller to perform a sequence of image processing tasks. At first, the stereo camera takes a pair of images. The stereo camera consists of two identical cameras that are calibrated using Tsai's [61] method. These images go through a standard lens distortion [53] removal step. This is further processed to obtain a disparity map [13] using a Sobel filter [54] and semi-global block matching. As disparity is inversely proportional to the depth of an image pixel [60], it gives us the z-coordinates of image pixels. These z-values are used to reconstruct the 3D world coordinates of the points corresponding to each pixel in the disparity map.

Each body camera keeps a timestamped, aggregated point cloud which is updated by merging a newly generated point cloud to it. For merging, we use iterative point cloud (ICP) [34] algorithm. In order to speed up and de-noise the data, we down-sample the data using a box grid filter. To bound the execution time, we limit the iteration count to 20, which is reached only for the degenerate cases.

Upon receiving a pull request from a receiving device, the latest aggregated point cloud is transmitted wirelessly.

• *IMU Data Processing Subsystem.* This step measures the linear and angular displacements from the IMU data, sampled at 100Hz. For linear displacements, we count the number of steps and multiply it with average walking stride length of a human (0.762 meter) to get the distance [25, 66]. First, we remove the effect of gravity using magnitude and variance of acceleration. Then, we detect the *swing* and *stance* phase of a user by applying thresholds. A step is detected when a swing phase ends and a stance phase starts. For angular displacements, we use absolute orientation in the Euler vector form.

• *The Controller Subsystem.* At the heart of Glimpse.3D is a controller that optimally triggers the image processing subsystem, based on the motion of a person wearing the body camera. A naive design of this controller could be to trigger an execution whenever there is a significant change in position or orientation of the body

camera. However, such a reactive algorithm is not capable of jointly optimizing the battery life and the quality of 3D point clouds. Hence, an adaptive controller is designed and implemented in Glimpse.3D, which solves an optimization problem to find optimal values of linear and angular displacements of the body camera, and adapts the model parameters at runtime to accommodate new experiences as the system operates in the real world. The detail of this controller is described in Section 4 along with other technical contributions of this paper.

3.3 Processing Inside an Aggregator Device

Each body camera keeps generating and saving an aggregated point cloud inside it until a viewer sends a pull request. Inside a pull request, the viewer/aggregator device sends a list of device identifiers and timestamps to broadcast the last successfully received point clouds from each body camera it is interacting with. Upon receiving a pull request, a body camera compares the timestamp t_{pull} from the pull request to the timestamp t_{curr} of its most recent aggregated point cloud, and transmits the aggregated point cloud over UDP, only if $t_{curr} > t_{pull}$. Each body camera periodically runs a garbage collector to remove older point clouds from its memory. After receiving a new point cloud, the viewer merges it with an internal point cloud that it maintains throughout a viewing session, i.e. the time between the starting and stopping of the viewer application. In order for a better and denser visualization the sparse point cloud, we triangulate [3] each point with two neighboring ones. This is a linear operation and it is done only at an increment. The amortized rendering time is in the order of 1.5s.

3.4 Technical Challenges

We address three key technical challenges during the design and implementation of Glimpse.3D.

• *Quantifying the Quality of Merged Point Cloud.* Glimpse.3D aims at maximizing the quality of generated 3D point clouds while ensuring a certain battery life. A basic principle in Glimpse.3D is not to spend valuable resources in capturing and processing images unless it is worth doing so. To determine whether generating and merging a new point cloud is worth, we need to quantify the utility of such a merger. Hence, the system needs to quantify the quality of a point cloud merger, even before it actually capture the second image-pair. To address this challenge, Glimpse.3D proposes an algorithm to quantify the expected quality of a point cloud merger based on structural similarity and modeled by internal parameters of ICP-based point cloud merging algorithm. The details of this algorithm are described in Section 4.1.

• *Determining an Optimal Trigger Point to Execute Image Processing Modules.* Despite an extensive tuning of the parameters of all image processing modules in Glimpse.3D, they still remain as the most time and power consuming entities in the system. To increase efficiency, Glimpse.3D implements a controller that measures linear and angular displacements of the person wearing the body camera and triggers the image processing subsystem only when the displacements exceed optimally computed thresholds. The detail of the optimization problem is described in Section 4.2.

• *Adapting the System Model at Runtime.* The optimizer that determines the optimal trigger point uses a set of empirical models pertaining to different aspects of the system. For example, one of these models maps linear displacements to the quality of merged point clouds. Although the parameters of such a model are estimated at design time with extensive measurements, they are likely to require run-time adaptations as the system operates in different scenarios, learns about new environments, and is worn by different persons. To handle this challenge, Glimpse.3D employs model adaptation that monitors the performance of the current system models and updates the parameters at run time, when the current models seem to be less accurate in predicting the system's behavior. The detail of this adaptation process is described in Section 4.3.

4 ALGORITHMS

In this section, we describe the algorithms that handle the challenges mentioned in Section 3.4. For a quick reference, Table 1 lists all symbols and parameters used in this section.

Symbol	Description	Range and Unit
Ω	Quality score.	0-1
ξ	Bounding volume.	m^3
η	Number of inliers.	
$\delta\phi$	Linear displacement	0 - 2.5m
$\delta\omega$	Weighted angular displacement	m/degree
β	Battery capacity.	
ζ_{rem}	Remaining battery life.	1-10 levels
ζ_c	Energy to process an image pair.	Ws
δt	Elapsed time	sec
γ_i	Parameters for quality scores.	
χ	Normalizing parameter.	
Γ_i, α_i	Parameters relating $\{\xi, \eta, \delta\phi, \delta\omega\}$.	

Table 1: A list of symbols that are used in this section.

4.1 Quality of a Merged 3D Point Cloud

In order for us to maximize the quality of generated and merged 3D point clouds, we need to quantify the quality of a point cloud. Unfortunately, there is no established metric for assessing the quality of a point cloud in the computer vision literature. An evaluation of a 3D reconstruction algorithm is often limited to reporting the computation time and algorithmic complexity followed by a series of test images and their 3D reconstructions.

We make an effort to bridge this gap by defining an objective quality score of a merged 3D point cloud C by comparing its structural similarity index (SSIM) [64] against a baseline. SSIM is a quality assessment index based on the computation of luminance (l), contrast (c) and structural (s) terms. These three terms are expressed with the following equations:

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1};\ c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2};\ s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

μ_x, μ_y, σ_x, σ_y, and σ_{xy} are the local means, standard deviations, and cross-covariance for images x, y. SSIM is the multiplicative variant of these three components.

$$SSIM(x,y) = \left[l(x,y)\right]^{p_1}\left[c(x,y)\right]^{p_2}\left[s(x,y)\right]^{p_3} \quad (1)$$

Here, p_1, p_2 and p_3 are multiplicative exponents. The intuition behind using SSIM is to make sure that point clouds that preserve the

shapes and the shades of objects in it gets higher scores than those that do not. To obtain the baseline shapes and shades in the scene, we exploit the two pairs of 2D images that were originally used to generate the point clouds, which have been merged to generate the C. We stich [14] these four images to obtain a panaromic 2D image P. Now, if the merged 3D point cloud C is of sufficiently high quality, one of its 2D projections on a plane $Proj(C)$ should have a strong structural similarity with P. Figure 5 shows the steps of this process using an example. Not that, although an image from each pair of images would suffice, since no two cameras are identical, we take all four images in our calculation for robustness. This maybe redundant and could be optimized further; but since this is an offline step, we lean toward robustness.

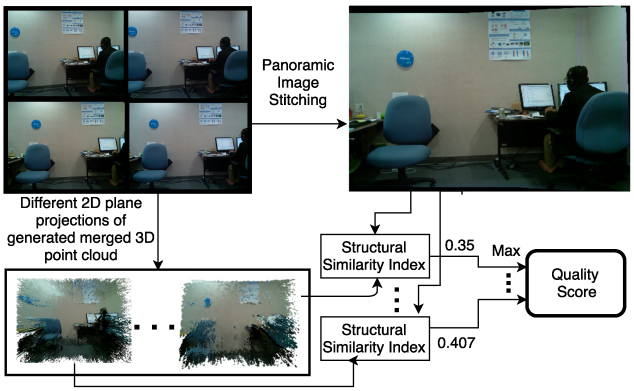


Figure 5: Quality score calculation process.

The proposed scoring method depends on the density of a 3D point cloud. A dense point cloud preserves the structures better. The density of a point cloud, however, depends on the box grid size parameter used during downsampling in the 3D reconstruction pipeline. It divides the point cloud space into cubes and points within each cube are combined into a single output point by averaging their coordinates. In other words, a smaller grid size results in a better 3D point cloud but at the cost of an increased computation time. The trade-off between the quality score and the computation time of 3D point cloud generation is shown in Figure 6. We observe that there is not much of an improvement of quality if we spend more than 2.4s in generating a point cloud. Hence, in our implementation, we set the grid size to 10cm to maintain a runtime of less than 3s for point cloud mergers.

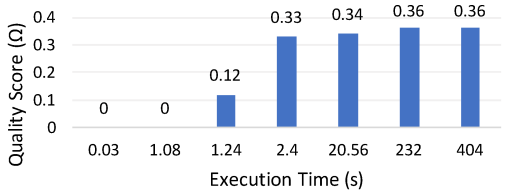


Figure 6: Point Cloud Merging Time VS Quality Score

The proposed method for assessing the quality of a point cloud merger proved to be very effective in Glimpse.3D and Section 6.2 shows that the objective score closely resembles human perception as well. However, directly calculating the score is computationally expensive. Therefore, we model it as a function of a set of variables, whose values are obtained as a byproduct of the ICP step in the 3D reconstruction pipeline. This makes it possible for Glimpse.3D to quantify the quality of a point cloud in constant time at runtime. The steps of this modeling algorithm are as follows:

• *Identifying Correlated Variables.* We identify the parameters of a point cloud merge algorithm, i.e. ICP [34], that are sensitive to the quality score of a merged point cloud. We use over 112 different scenes each representing a merged point cloud where the merger has been done between two point clouds generated inside two body cameras that are δ cm apart and are at ω deg angle, where $0 < \delta < 250$ and $0 < \omega < 60$. In this way, we obtain a large data set of point clouds whose quality scores are computed offline using the direct method as depicted in Figure 6. During the creation of the data set, we track and log the values of over a dozen of variables of the ICP algorithm. As these parameters are already calculated while performing the ICP, there is no extra computational cost for it. For each variable in our log, we perform a correlation analysis [22] to determine their relevance to the quality score. We eliminate variables that show negligible or no correlations, which leaves us seven variables that are shown in Table 2. For each variable, we show their correlation and corresponding p-value to show the significance of the correlation analysis.

Parameter	Correlation	p-value
Root mean squared error	-0.5026	0.0000
Number of inlier points	+0.3478	0.0002
Total points	-0.4557	0.0000
Average distance	-0.4397	0.0000
Bounding volume	+0.3950	0.0000
Iteration count	-0.2139	0.0001
Volume of merged point clouds	-0.2255	0.0168

Table 2: Correlation between different parameters of ICP-based point cloud merge algorithm and the quality scores. Corresponding p-values show significance of correlation.

From Table 2 we observe that *root mean squared error* of ICP, *number of inlier points* (i.e. a pair of matched points whose Euclidean distance falls within a percentage of matching distances), total number of points after merge (*total points*), *average distance* of two point clouds, and *bounding volume* of overlapped region are at least 30% correlated and their p-values are also close to zero, i.e., the correlation is significant. The next two variables i.e. total number of iterations in ICP algorithm (*iteration count*) and *volume of the merged point clouds* are even less correlated.

This step implies two things. First, not all variables are relevant (as expected), and second, because no single variable stands out and shows a very high correlation, we cannot infer quality from a single variable. We need to consider a subset of these weakly correlated variables.

• *Principal Component Analysis.* In this step, we take the variables from Table 2, and perform principal component analysis (PCA) [26] to identify a linear combination of the variables that accounts for high amounts of variability in the data. It turns out that the first two principal components account for up to 83% variability in

data, and the major contributors in those principal components are *{number of inliers, bounding volume}* and *{total points, average distance}*, respectively. This is an interesting result which gives us a choice between these two sets of variables that are statistically similar in explaining the variability in data. We pick the first set of variables, i.e. bounding volume ξ, and number of inliers η in the merged point cloud to model the quality metric.

• *Fitting a Nonlinear Model.* In the final step, we fit a nonlinear model to express the quality score Ω in terms of two identified variables ξ and η. Our process of determining model is iterative, i.e. we start with simple linear models, and increase complexity until the goodness of fit (expressed in root mean squared error, RMSE) is below a threshold. After visualizing the data (Figure 7), we realize that Ω is not a continuous function of ξ. Hence, we condition Ω based on a threshold on ξ to obtain the following model:

$$\Omega = \begin{cases} \gamma_1 - \gamma_2\eta + \gamma_3\xi + \gamma_4\eta^2 + \gamma_5\eta\xi + \gamma_6\eta^3 + \Gamma_7\eta^2\xi, & \text{if } \xi \geq c \\ \gamma_8 + \gamma_9\eta + \gamma_{10}\xi + \gamma_{11}\eta^2 + \gamma_{12}\eta\xi, & \text{otherwise} \end{cases} \quad (2)$$

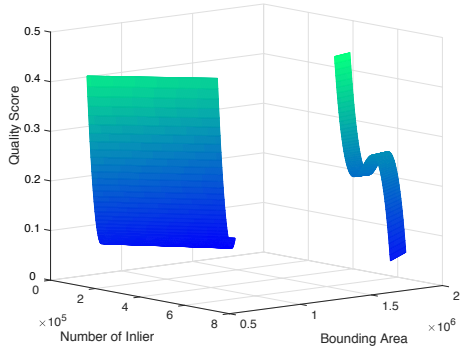


Figure 7: Fitting a nonlinear model to express quality score Ω in terms of number of inliers η and bounding volume ξ.

We empirically determine values of the model parameters $\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5, \gamma_6, \gamma_7, \gamma_8, \gamma_9, \gamma_{10}, \gamma_{11}, \gamma_{12}$, and c within 95% confidence intervals to be 848.4, -0.003918, 3.661E-5, 6.052E-9, - 1.172E-10, - 3.11E-15, 9.368E-17, 29.55, -0.000425, -7.559E-7, 1.651E-9, -1.549E-12, and 1.6E6, respectively.

Note that, the proposed structural similarity index-based metric (illustrated by Figure 5) is generic and is applicable to any 3D reconstruction algorithm to quantify the quality of point clouds. It can be directly used in any system where computational resources are adequate. However, since it is computationally expensive, in Glimpse.3D, we look for an alternative way to find an approximation to the metric that relies only on the byproducts of an ICP-based 3D point cloud estimation algorithm. For a different point cloud estimation algorithm, although the exact relationship/formula would be different, the steps in Section 4.1 (i.e. identifying variables, PCA, and model fitting) remain generic and can be followed to find a relationship similar to Equation 2.

4.2 Determining Optimum Trigger Point

To maximize the battery life of Glimpse.3D, the trigger point for the image processing subsystem needs to be optimum. For example, executing image processing tasks when user is stationary only decreases battery life without contributing new information. On the other hand, when user is moving, the image capturing and processing subsystem must be triggered often enough to maintain a desired quality.

In Section 4.1, we established a model that represents the Quality Score as a function of Number of Inliers and Bounding Volume. Since the overlap between two point clouds affects the quality of their merger, changes in camera views due to user's movement also affects the quality score. Hence, there is a relationship between the parameters representing quality scores (Number of Inliers and Bounding Volume) and a user's displacement. In this section, at first, we model these relationships and formulate an optimization problem that finds an optimum displacements (for both linear and angular movements) which maximizes the quality of the merged point clouds while satisfying battery life requirement. This process is performed offline and the steps of the algorithm are as follows.

• *Defining the Displacement Parameters.* We observe that little angular shift drastically changes the view. Hence, normalizing the angular displacement using the average depth of the most recent point cloud results in a better indicator of angular displacement when we are interested in its effect on point cloud merging. The normalized angular distance ($\delta\omega$) is defined by:

$$\delta\omega = \frac{\text{angular displacement}}{\text{average depth of the most recent point cloud}} \quad (3)$$

The linear displacement ($\delta\phi$) does not change the view as much as the angular displacement. Hence, we directly use the linear displacement value, described in Section 3.

• *Defining the Battery Life Constraint.* A major concern in battery-powered systems like Glimpse.3D is the battery life β. We model the battery life based on the remaining battery level ζ_{rem} and the expected energy for one execution ζ_c of the end-to-end image processing cycle. This relation is represented by Equation 4:

$$\beta(\delta\phi, \delta\omega) = \frac{\zeta_{rem}}{\zeta_c} \times \left(\frac{\delta\phi}{\Delta\phi} + \frac{\delta\omega}{\Delta\omega}\right) \quad (4)$$

where, $\Delta\phi$ and $\Delta\omega$ denote the linear and the angular velocity, respectively. We define ζ_{rem} in 10 levels, where each level corresponds to 10 difference battery percentages. To maintain the uniformity for different battery capacity, we normalize Equation 4 on a scale of 0 to 100.

• *Relationship Between Quality Parameters (η, ξ) and Displacements ($\delta\omega, \delta\phi$).* To estimate the quality from displacement values, we find relationships between the quality parameters (η and ξ) of Equation 2 and two displacement parameters ($\delta\omega$ and $\delta\phi$) by performing correlation analysis on dataset from Section 4.1. The results

	$\delta\phi$	$\delta\omega$
Inlier points, η	+0.55	+0.84
Bounding volume, ξ	-0.19	-0.17

Table 3: Correlation analysis.

of correlation analysis and corresponding scatter plots are shown in Table 3 and Figure 8, respectively. For each of two parameters, η and ξ, we individually fit two linear models having independent variables $\delta\phi$ and $\delta\omega$ and then take weighted summations to get the best estimate over our data set. The relationship between η and displacements($\delta\phi$ and $\delta\omega$), shown in Figure 8(a) and (b), is formulated in Equation 5 and 6.

$$\eta_\phi = \Gamma_1 \delta\phi + \Gamma_2 \quad (5) \qquad \eta_\omega = \Gamma_3 \delta\omega + \Gamma_4 \quad (6)$$

We empirically determine the coefficients Γ_1, Γ_2, Γ_3 and Γ_4 to be 39417, 569365, 14052, and 85439, with 95% confidence.

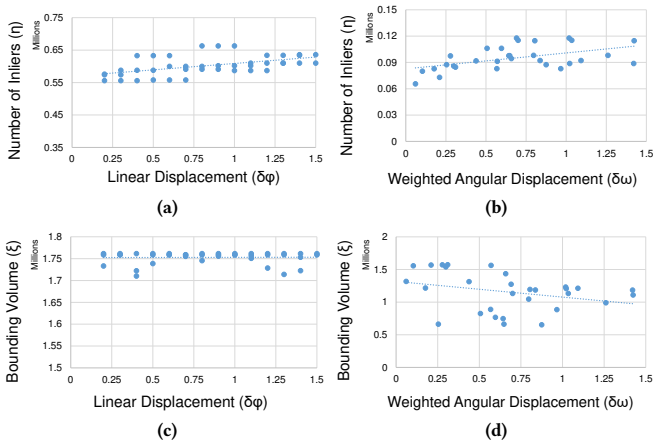


Figure 8: Relation of linear displacement $\delta\phi$ and weighted angular displacement $\delta\omega$ with number of inliers η and bounding volume ξ.

Next, we combine the two estimates η_ϕ and η_ω by taking weighted average to model η in terms of displacements.

$$\eta_{\phi,\omega}(\delta\phi, \delta\omega) = \alpha_\eta \eta_\phi + (1 - \alpha_\eta)\eta_\omega$$
$$\text{here, } \alpha_\eta = \begin{cases} 0, & \text{if } \eta_\phi = 0 \\ 1, & \eta_\omega = 0 \\ 0.40, & \text{otherwise} \end{cases} \quad (7)$$

Here, in absence of linear and angular displacements, α_η is 0 and 1 respectively. When both are present, α_η is 0.40. Similar to Equations 5, 6, and 7, we formulate the following three equations that model the bounding volume ξ in terms of displacements $\delta\phi$ and $\delta\omega$:

$$\xi_\phi = \Gamma_5 \delta\phi + \Gamma_6 \quad (8) \qquad \xi_\omega = \Gamma_7 \delta\omega + \Gamma_8 \quad (9)$$

We empirically determine the coefficients Γ_5, Γ_6, Γ_7, and Γ_8 to be 530.27, 2e+6, -240063, and 1e+6, with 95% confidence. In Equation 10, in the absence of linear and angular displacements, α_ξ is 0 and 1, respectively. When both types of displacements are present, α_ξ is 0.53.

$$\xi_{\phi,\omega}(\delta\phi, \delta\omega) = \alpha_\xi xi_\phi + (1 - \alpha_\xi)\xi_\omega$$
$$\text{here, } \alpha_\xi = \begin{cases} 0, & \text{if } \xi_\phi = 0 \\ 1, & \xi_\omega = 0 \\ 0.53, & \text{otherwise} \end{cases} \quad (10)$$

• *Formulating Optimization Problem.* Finally, we formulate an optimization problem whose objective is to maximize battery life and quality of merged point cloud. The goal is to determine the values of $\delta\phi$ and $\delta\omega$ for which the objective is maximized under given constraints on minimum quality (Ω_{min}) and minimum battery life (β_{min}).

$$\begin{aligned} \underset{\delta\phi, \delta\omega}{\text{maximize}} \quad & \Omega(\delta\phi, \delta\omega) + \chi\beta(\delta\phi, \delta\omega) \\ \text{subject to} \quad & \Omega \geq \Omega_{min} \;\&\; \beta \geq \beta_{min} \end{aligned} \quad (11)$$

We set $\Omega_{min} = 0.12$ based on a user study in Section 6.2 which showed that a quality score below 0.12 is not useful. The value of β_{min} depends on the application scenario. $\chi = 0.01$ is a normalizing factor that balances the two terms with different units. For different values of ζ_{rem}, we solve this optimization problem and save the results in a table. At run-time, we look-up the table as the battery level changes.

In this system, if the value of Ω_{min} is too high, we will achieve a higher quality point cloud but it will decrease the battery life β. If β_{min} is too high then Ω will decrease. Our goal is to find the optimal displacement value to maximize both Ω and β.

4.3 Runtime Model Adaptation

The proposed optimizer outputs certain linear and angular displacements maximizing quality score and battery life. Since the parameters in optimization problem are empirically determined, their generalizability is limited to scenarios similar to the ones in data set. In real-life, especially for mobile systems, the scenarios are likely to vary in terms of lighting and shading, number, types and distance of objects in the scene. To make Glimpse.3D robust to such changes and learn from new scenarios, we monitor and adapt the system at run-time. Figure 9 provides overview of the proposed adaptation mechanism, and the steps of the algorithm are as follows

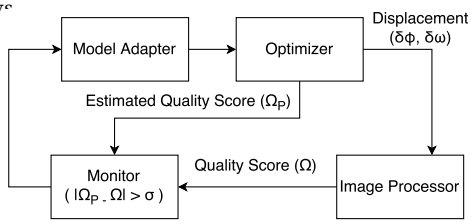


Figure 9: Runtime model adaptation in Glimpse.3D.

• *Monitoring Stability of Optimum Trigger Point.* To find optimum trigger point described in Section 4.2, we solve an optimization problem beforehand using a large empirical dataset. Performance of the controller is expected to be optimal in most scenarios. However, to cope with the stochastic nature of real world, we adapt the system at run-time whenever its performance is different than expected.

After each executing image processing subsystem, *monitor* module compares the *estimated quality* (Ω_p), which is calculated by the optimizer using $\delta\phi$ and $\delta\omega$, with the *actual quality* score (Ω), calculated directly from the ξ and η. If the difference $|\Omega_p - \Omega|$ is below a threshold, the system is assumed to be performing as expected. Otherwise, the models that fit displacements to ξ and η are updated.

• *Updating Models.* This step updates the parameters of the linear models that we developed in the previous section through series of Equations 5, 6, 8, and 9. Since these models are linear, we update them at run-time using basic geometry. Note that, we have validated the linearity of the models by using a k-fold cross validation test, and hence, there is no reason to use a higher order polynomial. A model update is triggered when at least m violations are detected by the monitor module. We fit these m data points to a line, L_{data}. Let, our the existing model L_{prev} be estimated using n data points. We find a new linear model L_{new} which is a weighted average of L_{data} and L_{prev}. We consider two cases.

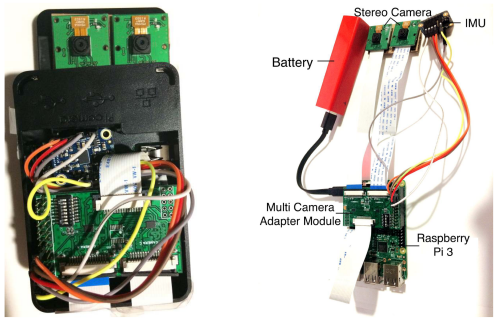
In the first case, L_{prev} and L_{data} are parallel. The new model L_{new} would be a line parallel to the other two, and is at $x = \frac{m \times d}{n+m}$ unit away from L_{prev} toward the L_{data}, where d is the distance between L_{data} and L_{prev}. In the second case, L_{prev} and L_{data} intersects at point O. So, L_{new} will passes through O while dividing the angle θ between the lines L_{data} and L_{prev} into $n : m$ ratio. If, L_{new} makes angle θ_1 with L_{prev}& P is a point on L_{new}, let l_1 and l_2 be distances of L_{prev} and L_{data} from P. We find θ_1 from following relationships:

$$\text{Given,} \quad l_1 = psin\theta_1; \quad l_2 = psin(\theta - \theta_1); \quad \frac{l_1}{l_2} = \frac{m}{n}$$
$$\text{We get,} \quad \theta_1 = tan^{-1}\frac{msin\theta}{n + mcos\theta} \tag{12}$$

• *Running Optimizer.* Each time the system adapts, the optimizer uses the updated models to find solutions to Equation 11. To keep the adaptation process fast and real-time, instead of solving the optimization problem on-the-fly, we use a lookup table. The table is populated offline by solving the optimization problem for combinations of ζ_{rem}, x and θ_1. We run the optimizer for different angular ($0 \leq \theta_1 \leq 360$) and linear ($x_{min} \leq x \leq x_{max}$) displacements. Since populating the table for all possible values of θ and x are not feasible, we select discrete values of θ and x in their ranges. The total number of combinations depends on the available storage. x_{min} and x_{max} are empirically determined and if a new x is out of the range, the table needs to be recomputed again (offline).

5 IMPLEMENTATION NOTES

We develop Glimpse.3D using off-the-shelf hardware and open-source software. Figure 10 shows two views of the hardware. Dataset and software is accessible from here [5].



(a) close up **(b) teared-down**

Figure 10: A closer look at Glimpse.3D camera unit.

• *Computational Unit.* We use Raspberry Pi 3 [8] for on board computation that has 1.2GHz 64-bit quad-core ARMv8 CPU and 1GB RAM. Presence of many GPIO pins & on-board WiFi module make it a perfect choice for our prototype.

• *Camera Unit.* We use 5MP Pi Cameras [7] having a fixed focus (1m to infinity), 3.60mm focal length, 53.50^0 horizontal FOV, 41.41^0 vertical FOV, and 30fps max frame rate. Two camera modules connect to the Pi via an adapter [9].

• *Inertial Measurement Unit.* We use a 100Hz 9-DOF BNO055 absolute orientation sensor [6]. A fusion algorithm that blends IMU data into stable three-axis orientation output is implemented in the sensor.

• *Power.* We use a 2600mAh rechargeable battery. We profile the energy consumption of each process offline using a USB multimeter. At runtime, we estimate the remaining battery by tracking the execution time and process id.

• *Vision Library.* We use OpenCV [12] and OpenGL ES [4] for for image processing and displaying point clouds.

6 EVALUATION

6.1 Microbenchmarks

Table 4 shows execution time, energy consumption, memory and CPU usage of each step of Glimpse.3D.

Process	Thread No.	Runtime (s)	Energy Per Execution (J)	Memory (%)	CPU (%)
IMU Process	1	0.15	0.12	1.7	85.7
Stereo Capture	2	0.20	0.39	3.2	94.6
Disparity Map	2	0.80	0.95	3.2	94.6
Point Cloud Generate	2	0.09	0.11	3.2	94.6
Point Cloud Merge	3	2.44	2.8	5.2	117
Point Cloud Transmit	4	0.01	0.01	0.0	0.6
Controller Adaption	5	0.07	0.08	0.3	2

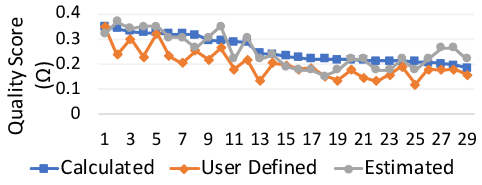
Table 4: Microbenchmarking Glimpse.3D.

Five threads run in Glimpse.3D. Thread 1 performs IMU data processing, which takes 0.15s and uses 0.12J energy, 85.7% CPU and 1.7% memory. Thread 2 calculates the disparity map, captures stereo images, and generated the 3D point in 1.2s, consuming 0.95J, 0.39J, and 0.11J ,respectively and uses similar memory (3.2%) and CPU (94.6%). The costliest step is the point cloud merging (thread 3) that takes 2.44s. It consumes 2.80J energy, 5.2% memory and 117% CPU. These later two threads run only when the image processing is triggered. The fourth and fifth threads are for point cloud transmission and controller adaptation, respectively. The overhead of these threads are negligible. Considering the total energy consumption of an image processing cycle, for a 2600mAh battery, the total number of 3D point clouds Glimpse.3D processes is 10,515.

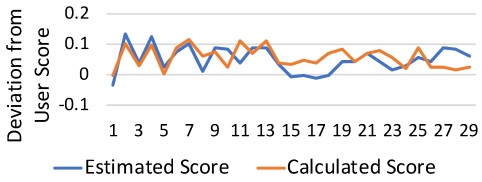
6.2 Evaluation of Quality Scoring

In this section, we compare our proposed SSIM-based quality scores with its estimated version using Equation 2 as well as with human-given scores obtained in a user study. To obtain human-contributed scores, we present the 3D point clouds to the users and let them score the clouds as they like. Since many of the users did not have any prior idea of the quality of 3D point clouds, during a briefing session (prior to the study), each user was shown two reference point clouds with scores- a high-quality and a low-quality cloud to give them a sense of the range. The results are shown in Figure 11.

We use a dataset of 29 point clouds. Five of which are shown in Figure 13 as examples. We take the average of scores from 21 users for each image. The users were of different ages, sexes (11 females), and professions (e.g. students, businessmen, physicians, and engineers). In Figure 11a, we observe that the estimated score has 79% percent correlation with user-defined score. Hence, the estimated score reflects user perception. Moreover, estimated score is also close to the proposed SSIM-based calculated score with



(a) Calculated, user defined, and estimated quality scores.



(b) Deviation of estimated & calculated scores from user scores.

Figure 11: Performance of quality scoring.

82% correlation. Figure 11b shows that the estimated score and the calculated score both are close to user-defined score with an average deviation of 0.051 and 0.058, respectively. Besides, they show similar trend proving that our estimation of the calculated score is correct.

6.3 Evaluation of Optimum Triggering

We compare our proposed optimized solution with two other non-optimized solutions in terms of the quality of captured 3D information and the battery life.

6.3.1 Defining Baselines. We compare Glimpse.3D with a *time-triggered* system and a *simple motion triggered* system. The time-triggered system triggers the image processing subsystem at a regular interval. To choose a suitable period for the time-triggered baseline, we conduct an experiment to obtain the relationship between the time interval and the battery life of the system as shown in Figure 12. As the process of generating point clouds takes around 2s, we pick a time interval of 2s, for which, the system lasts for at least 6 hours. We estimate the battery life from the energy consumption of the system at different states and the fraction of time the system remains in those states. Each state is characterized by the number of threads it runs. Finally, we estimate battery life by dividing the battery capacity by the estimated total energy using the following Equation 13:

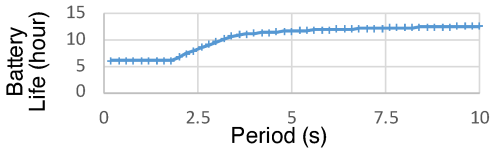


Figure 12: Effect of period on battery life.

$$\text{Battery Life} = \frac{\text{Total Battery}}{(w_0 * p_0) + (w_1 * p_1) + \cdots + (w_n * p_n)} \quad (13)$$

where, p_i are energy consumptions due to zero or more threads and w_i are corresponding weights.

For the simple motion triggered baseline, we trigger image processing whenever the user takes a step or the angular displacement is above 5 degrees.

6.3.2 Defining User Motions. We define three scenarios based on the activity level of a body camera user. An activity means that a user is wearing the system and walking at a somewhat constant speed. A *high* activity level means a user is moving continuously and vigorously, a *low* activity means a user is not moving at all, and a *medium* activity means the user is active only half of the time. We estimate the battery life from the power consumption during each run.

6.3.3 Comparison of Battery Life. Figure 14 shows the battery life of Glimpse.3D and the two baselines at different user activity levels. For low activity level, both simple motion-triggered system and Glimpse.3D show the longest battery life of over 14 hours since no image is processed and the battery consumption is due to the idle state of the system only. But the time-triggered system is periodically executing, which decreases the battery life to 6 hours.

For the medium activity level, the simple motion-triggered system shows the lowest battery life as it is not optimized and executes more image processing tasks than Glimpse.3D. We observe a similar phenomenon in the high activity level where the battery life of the simple motion-triggered system is reduced to 3hr, while Glimpse.3D lasts for about 6hr due to its motion optimized execution triggering. The time-triggered system has no effect on the motion, and hence, its lifetime is always fixed at 6 hours. Although time-triggered seems to be the winner for high activity levels, this is unlikely that a user will be active for 100% of the time. Besides, we also need to consider the quality of the captured 3D information, which we compare next.

6.3.4 Comparison of Captured 3D Information Quality. We compare the quality of the merged point clouds generated by different systems for different battery life at the medium human activity level. Figure 15 shows the results. The quality achieved by the time-triggered system and Glimpse.3D increases with decreased battery life, as these systems capture more images and generates denser point clouds. The quality achieved by the simple motion-triggered system remains the same as the displacement between two sets of point clouds remains static. We also note that the simple motion-triggered system does not last 8 hours as it exhausts the battery sometime after the 6-hour mark. For a 6+ hours of battery life, Glimpse.3D yields better quality 3D point clouds than the time-triggered system. The time-triggered system performs slightly better than Glimpse.3D when the lifetime requirement is less than 2 hours. This slight gap in quality is due to the modeling inaccuracy in Glimpse.3D, which can be eliminated by adapting the system parameters at run-time. We evaluate the effectiveness of adapting model parameters in the next section.

6.4 Adaptive Model Update Evaluation

In this section, we evaluate the effectiveness of adaptive model update algorithm. We take the system to a completely new outdoor environment and run it with and without the adaptation step. For the non-adaptive version of Glimpse.3D, we expect to see that the estimated quality scores (scores prior to processing an image pair)

(a) $\Omega = 0.35; 1.04s$ **(b)** $\Omega = 0.34; 2.3s$ **(c)** $\Omega = 0.33; 2.4s$ **(d)** $\Omega = 0.25; 3.1s$ **(e)** $\Omega = 0.23; 2.2s$

Figure 13: Examples of point clouds along with their quality scores and end-to-end execution times.

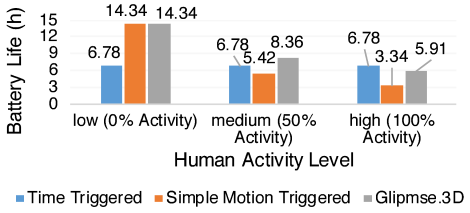


Figure 14: Comparison of battery lifetime.

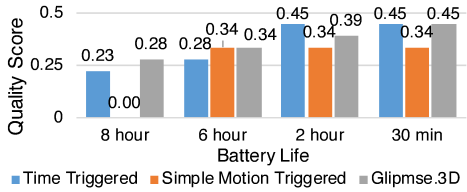


Figure 15: Comparison of captured 3D information quality

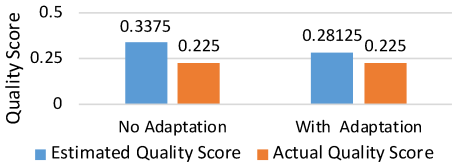


Figure 16: Difference between estimated and actual quality scores for adaptive and non-adaptive strategies.

and the actual scores (scores after processing image-pairs) will have a larger difference. This is evident in Figure 16, where the adaptive version of Glimpse.3D becomes more accurate in predicting the quality scores after it has observed new scene and adjusted its model parameters based on the outcomes from the new scenes. This is a significant result as being able to accurately predict the quality of point clouds at run-time is crucial to make sure that the optimal displacement thresholds calculated by the optimizer are able to guarantee the quality and lifetime requirements of the system in a new environment.

As the optimization process is performed offline, the adaptation process becomes a constant time operation. The adaptation process consumes only 0.08J energy per execution as mentioned in Table 4. Therefore, its effect on battery is negligible. Besides, the number of adaptations needed for a scenario is completely dependant on both the initial dataset (used in Section 4) and the new scenario. If the dataset is large and diverse, the number of needed adaptation will be minimal.

7 DISCUSSION

• *Why not a Depth Sensor?* Although depth cameras provide us the depth of each pixel directly, they have significant limitations [43]. High-resolution RGBD cameras like Bumblebee are very expensive ($3,500) [51], require powerful host machine, and parameter tuning for each new environment. PMD units provide gray-scale images, work only indoors [33]. They are also expensive, and CPU consuming. Kinect is affordable, but they are ineffective in daylight and scenes with large gradients in distances [36], and multiple Kinects in the same environment interfere with each other.

• *Why Point Cloud instead of Image or Key Frames?* We design Glimpse.3D as a closed-loop control system where it makes an educated guess of when to take images. The system corrects itself only after it has computed the quality of the most recent point cloud. If we designed it as an open-loop system, i.e. capturing and sending images or key frames, the system would not perform optimally in different settings. It would capture and transmit either too many images, which will require more bandwidth and energy, or too little images that have enough overlaps and features to generate a decent point cloud. Besides, computing the key frames itself is an expensive task, which we want to avoid.

• *Why On-Board Computing?* There are two reasons for performing the computation on-board. First, the optimization algorithm executes as a subroutine in the overall closed-loop process that is depicted by Figure 9. We lose this feedback loop when only images are streamed based on precomputed optimal displacement parameters. The system would not be able to adapt and optimize itself beyond the training scenarios unless the quality of 3D point clouds are taken into account to tune its parameters. Second, to generate a reasonably good quality point cloud, at least 50% overlap between the input images are expected. Therefore, computing a point cloud on-board is cheaper in terms of required space/bandwidth as the system does not have to store/stream the redundant/common information in its constituent images.

• *Why not SLAM?* The goal of Simultaneous Localization And Mapping (SLAM) [47] is to jointly estimate the pose and map the environment when navigating in an unknown environment. SLAM maps is expressed in many different ways such as landmarks, positions, occupancy grids, and also point clouds. SLAM could be a part

of Glimpse.3D's process chain, but its frame rate requirement (60 fps) is too high for a resource constrained system like Glimpse.3D.

• *Why not VIO?* The Visual Inertial Odometry (VIO) [38] refers to the method of estimating traveled distance using visual features as well as using IMUs and then combining the two results for a better accuracy. For VIO, it is necessary to continuously capture and process images, which we want to avoid in Glimpse.3D due to resource constraints.

• *Why Regression over Classification?* Machine learning tasks (strictly speaking- supervised ML where both inputs and outputs are known and the goal is to find an optimal hypothesis/model that fits the data) are broadly categorized into two types: a classification task and a regression task. A classification problem is formulated when the outputs are discrete-valued and the number of distinct outputs is small- so that each unique output (or a range) can be labeled as a class. The problem at hand, i.e. fitting a model to continuous-valued quality scores within the range of [0, 1] is not well-suited to a classification formulation, as either (1) we have to discretize the range of scores into a small number of sub-ranges- thereby losing the precision of quality scores, or (2) we will have to formulate a 100-1000 class classification problem (based on the number of decimal digits we consider) - which is not practical as that would require a large number of training examples for each of the 100-1000 of classes. Hence, we formulate the problem as a regression analysis task which is a better fit to our need. Furthermore, we iteratively increase the complexity of the model to find the simplest model that best fits the point cloud quality scores, as simpler curves or hypotheses are always preferable to complex ones due to their better generalization ability beyond the empirical data as well as computational efficiency.

• *Why Stereo Vision?* Both stereo vision and structure from motion (SfM) are popular methods for 3D reconstruction. In this paper, we choose stereo vision because of its lower computational requirement than SfM. In SfM, one of the initial steps is to estimate the pose where the extrinsic parameters (angular and linear transformation matrices of two images) are calculated. In stereo vision, since the relative position of the two cameras is fixed, the extrinsic parameters are estimated once during the calibration step and there is no need to perform pose estimates for each pair of images at runtime.

8 RELATED WORK

3D reconstruction is a well studied problem in computer vision. [37] uses efficient feature extraction and dense pixel methods for real-time camera tracking and reconstruction. However, it required multi-core CPUs and powerful GPUs [32, 62]. We process 3D information in a resource constrained embedded system.

Modern smartphones are equipped with multi-core processors and GPU cores. Interactive 3D reconstruction on smartphones has been attempted in [29, 41, 45]. However, their performance is far from that of a high-end computer. [57] demonstrated a mobile app based light-weight 3D reconstruction system which off loads the major 3D reconstruction tasks to a cloud computer. [58, 59] presented dense stereo-based system for real-time interactive 3D reconstruction on a smartphone using the IMU. Few works presented smartphone based outdoor and large-scale 3D reconstruction system utilizing the GPU in Google Tango Development Kit [49, 50]. [45, 46] proposed tracking and reconstruction of objects in mobile phone. IMU have been used in diverse computer vision applications [35, 67]. We use IMU to measure user motion and use it to guarantee 3D information quality and battery life of the system.

Some works have explored resource-constrained 3D reconstructions. [17] proposed a stereo acquisition system developed on DSP for 3D scanner. This system consists of both camera and a projector. [19, 21] proposed an embedded architecture for stereoscopic, plenoptic camera based 3D reconstruction in FPGA based hardware. In our system, we propose a body-worn camera system and exploit human motion for triggering the system.

IMU fusioned body camera systems have been introduced in [11, 31]. SLAM [47] and its variants [10, 18, 27, 65] are used in robot-based 3D reconstruction. For real-time 3D reconstruction and localization visual inertial odometry is also used [28, 39, 63]. In Glimpse.3D, we use the IMU data to decrease the number of captured images by forming a closed-loop control problem.

Raspberry Pis have been used for 3D imaging [44, 56], face recognition [52], drone-based 3D reconstruction [16]. Unlike Glimpse.3D, these systems use Pis for image capturing only, and 3D reconstruction happens offline on powerful machines.

Controlling data acquisition through analysis of sensing data has been applied to few different applications previously. [68] proposed an wrist-pulse retrieval and analysis system using portable wireless sensor system. [42] presented a vibration signal acquisition system for monitoring pedestrians by footstep induced vibration. They proposed a low-cost hardware system with off the shelf vibration sensors. In our system, we use stereo camera sensor for ensuring higher battery life and quality of point cloud.

9 CONCLUSION

This paper describes a low-cost body-worn camera system that captures the 3D experience in real-world, mobile environments. A new metric for quantifying the quality of merged point cloud has been proposed. A new algorithm has been developed to guarantee 3D visual information quality and battery life of this resource-constrained system. The performance of this optimized adaptive body-cam system has been compared with time-triggered and motion-triggered systems, and the effectiveness of model adaptation has been evaluated. It has been shown that the system provides a better guarantee of desired battery life and 3D information quality, and is robust to new environments.

ACKNOWLEDGEMENT

This paper was supported, in part, by NSF grants CNS-1704469. We thank our shepherd, Hae Young Noh, for guidance and the anonymous IPSN reviewers for their comments.

REFERENCES

[1] https://en.wikipedia.org/wiki/Bundle_adjustment.
[2] https://en.wikipedia.org/wiki/Distortion_(optics).
[3] https://en.wikipedia.org/wiki/Triangulation_(geometry).
[4] https://developer.android.com/guide/topics/graphics/opengl.html.
[5] https://github.com/Glimpse3D/Glimpse.3D.git.

[6] https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor.

[7] https://www.raspberrypi.org/products/camera-module/.

[8] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[9] http://www.arducam.com/multi-camera-adapter-module-raspberry-pi/.

[10] Bonin-Font, F., Cosic, A., and Negre, P. L. E. A. Stereo slam for robust dense 3d reconstruction of underwater environments. In *OCEANS* (2015), IEEE.

[11] Bouma, H., Baan, J., and ter Haar, F. B. E. A. Video content analysis on body-worn cameras for retrospective investigation. In *Proc. SPIE* (2015).

[12] Bradski, G., et al. The opencv library. *Doctor Dobbs Journal* (2000).

[13] Bradski, G., and Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.

[14] Brown, M., and Lowe, D. G. Automatic panoramic image stitching using invariant features. *International journal of computer vision* (2007).

[15] Chen, S., Zuo, W., and Zheng, L. Camera calibration via stereo vision using tsai's method. In *Education Technology and Computer Science* (2009), IEEE.

[16] Geipel, J., Link, J., and Claupein, W. Combined spectral and spatial modeling of corn yield based on aerial images and crop surface models acquired with an unmanned aircraft system. *Remote Sensing* (2014).

[17] Giryes, R., Bronstein, A. M., Moshe, Y., and Bronstein, M. M. Embedded system for 3d shape reconstruction. In *Proc. of the 3rd European DSP Education and Research Symposium (EDERS 2008)* (2008), pp. 265–272.

[18] Grisetti, G., Kummerle, R., Stachniss, C., and Burgard, W. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine 2*, 4 (2010).

[19] Hadjitheophanous, S., Ttofis, C., Georghiades, A. S., and Theocharides, T. Towards hardware stereoscopic 3d reconstruction: a real-time fpga computation of the disparity map. In *Proceedings of the Conference on Design, Automation and Test in Europe* (2010), European Design and Automation Association.

[20] Hamzah, R. A., Rahim, R. A., and Noh, Z. M. Sum of absolute differences algorithm in stereo correspondence problem for stereo matching in computer vision application. In *ICCSIT* (2010), IEEE.

[21] Hänsel, M., Rosenberger, M., and Notni, G. Fpga implementation of a multi-view stereo approach for depth estimation and image reconstruction for plenoptic cameras. In *Engineering for a Changing World: Proceedings; 59th IWK, Ilmenau Scientific Colloquium, Technische Universität Ilmenau, September 11-15, 2017* (2017).

[22] Hazewinkel, M. Correlation in statistics. encyclopedia of mathematics, 2001.

[23] Heikkila, J., and Silven, O. A four-step camera calibration procedure with implicit image correction. In *CVPR* (1997), IEEE.

[24] Hirschmuller, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In *CVPR* (2005), vol. 2, IEEE.

[25] Jimenez, A. R., Seco, F., Prieto, C., and Guevara, J. A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu. In *WISP* (2009), IEEE.

[26] Jolliffe, I. *Principal component analysis.* Wiley Online Library, 2002.

[27] Kaess, M., Ranganathan, A., and Dellaert, F. isam: Fast incremental smoothing and mapping with efficient data association. In *Robotics and Automation, 2007 IEEE International Conference on* (2007), IEEE.

[28] Klingensmith, M., Dryanovski, I., Srinivasa, S., and Xiao, J. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics: Science and Systems* (2015), vol. 4.

[29] Kolev, K., Tanskanen, P., Speciale, P., and Pollefeys, M. Turning mobile phones into 3d scanners. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on* (2014), IEEE, pp. 3946–3953.

[30] Konolige, K. Small vision systems: Hardware and implementation. In *Robotics research*. Springer, 1998.

[31] Kourogi, M., and Kurata, T. A method of personal positioning based on sensor data fusion of wearable camera and self-contained sensors. In *Multisensor Fusion and Integration for Intelligent Systems* (2003), IEEE.

[32] Ladikos, A., Benhimane, S., and Navab, N. Efficient visual hull computation for real-time 3d reconstruction using cuda. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW* (2008), IEEE.

[33] Langmann, B., Hartmann, K., and Loffeld, O. Depth camera technology comparison and performance evaluation.

[34] Lara, C., Romero, L., and Calderón, F. A robust iterative closest point algorithm with augmented features. In *Mexican International Conference on Artificial Intelligence* (2008), Springer.

[35] Lobo, J., and Dias, J. Vision and inertial sensor cooperation using gravity as a vertical reference. *TPAMI* (2003).

[36] Mankoff, K. D., and Russo, T. A. The kinect: A low-cost, high-resolution, short-range 3d camera. *Earth Surface Processes and Landforms 38*, 9 (2013).

[37] Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. Dtam: Dense tracking and mapping in real-time. In *IEEE ICCV* (2011).

[38] Nistér, D., Naroditsky, O., and Bergen, J. Visual odometry. In *IEEE, Computer Vision and Pattern Recognition.* (2004).

[39] Omari, S., Bloesch, M., Gohl, P., and Siegwart, R. Dense visual-inertial navigation system for mobile robots. In *ICRA* (2015), IEEE.

[40] Pagani, A., Gava, C. C., and Cui, Y. E. A. Dense 3d point cloud generation from multiple high-resolution spherical images. In *VAST* (2011).

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

[42] Pan, S., Xu, S., Mirshekari, M., Zhang, P., and Noh, H. Y. Collaboratively adaptive vibration sensing system for high-fidelity monitoring of structural responses induced by pedestrians. *Frontiers in Built Environment 3* (2017), 28.

[43] Pece, F., Kautz, J., and Weyrich, T. Three depth-camera technologies compared.

[44] Peyer, K. E., Morris, M., and Sellers, W. I. Subject-specific body segment parameter estimation using 3d photogrammetry with multiple cameras. *PeerJ* (2015).

[45] Prisacariu, V. A., Kahler, O., Murray, D. W., and Reid, I. D. Simultaneous 3d tracking and reconstruction on a mobile phone. In *Mixed and Augmented Reality (ISMAR)* (2013), IEEE.

[46] Prisacariu, V. A., Kähler, O., Murray, D. W., and Reid, I. D. Real-time 3d tracking and reconstruction on mobile phones. *IEEE transactions on visualization and computer graphics 21*, 5 (2015), 557–570.

[47] Riisgaard, S., and Blas, M. R. Slam for dummies. *A Tutorial Approach to Simultaneous Localization and Mapping 22*, 1-127 (2003).

[48] Sansoni, G., Trebeschi, M., and Docchio, F. State-of-the-art and applications of 3d imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors 9*, 1 (2009).

[49] Schöps, T., Sattler, T., Häne, C., and Pollefeys, M. 3d modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. In *3D Vision (3DV), 2015 International Conference on* (2015), IEEE, pp. 291–299.

[50] Schöps, T., Sattler, T., Häne, C., and Pollefeys, M. Large-scale outdoor 3d reconstruction on a mobile device. *Computer Vision and Image Understanding 157* (2017), 151–166.

[51] Šeatović, D., Meiser, V., Brand, M., Scherly, D., et al. Analysis of off-the-shelf stereo camera system bumble-bee xb3 for the fruit volume and leaf area estimation.

[52] Senthilkumar, G., Gopalakrishnan, K., and Kumar, V. S. Embedded image capturing system using raspberry pi system. *International Journal of Emerging Trends & Technology in Computer Science* (2014).

[53] Slama, C. C., Theurer, C., and Henriksen, S. W. *Manual of photogrammetry*. American Society of photogrammetry, 1980.

[54] Sobel, I. History and definition of the sobel operator. *Retrieved from the World Wide Web* (2014).

[55] Steele, J., Debrunner, C., Vincent, T., and Whitehorn, M. Developing stereovision and 3d modelling for lhd automation. In *6th International Symposium on Mine Mechanization and Automation* (2001).

[56] Straub, J., and Kerlin, S. A very low-cost 3d scanning system for whole-body imaging. In *SPIE Sensing Technology+ Applications* (2015), International Society for Optics and Photonics.

[57] Sung, Y.-H., Muchtar, K., Lai, H.-E., Yeh, C.-H., and Chen, C.-Y. Automated reconstruction of 3d object on embedded system for mobile apps. In *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on* (2014), IEEE, pp. 65–66.

[58] Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., and Pollefeys, M. Live metric 3d reconstruction on mobile phones. In *The IEEE International Conference on Computer Vision (ICCV)* (2013).

[59] Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., and Pollefeys, M. Live metric 3d reconstruction on mobile phones. In *Proceedings of the IEEE International Conference on Computer Vision* (2013).

[60] Trucco, E., and Verri, A. *Introductory techniques for 3-D computer vision*, vol. 201. Prentice Hall Englewood Cliffs, 1998.

[61] Tsai, R. Y. An efficient and accurate camera calibration technique for 3d machine vision. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1986* (1986).

[62] Tzevanidis, K., Zabulis, X., Sarmis, T., Koutlemanis, P., Kyriazis, N., and Argyros, A. From multiple views to textured 3d meshes: a gpu-powered approach. In *European Conference on Computer Vision* (2010), Springer.

[63] Usenko, V., Engel, J., Stückler, J., and Cremers, D. Direct visual-inertial odometry with stereo cameras. In *Robotics and Automation (ICRA), IEEE* (2016).

[64] Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* (2004).

[65] Weingarten, J., and Siegwart, R. Ekf-based 3d slam for structured environment reconstruction. In *Intelligent Robots and Systems, 2005.(IROS 2005).* (2005), IEEE.

[66] Writer, L. G. The average walking stride length, Jan 2014.

[67] You, S., Neumann, U., and Azuma, R. Hybrid inertial and vision tracking for augmented reality registration. In *Virtual Reality, Proceedings., IEEE* (1999), IEEE.

[68] Zhang, J., Wang, R., Lu, S., Gong, J., Zhao, Z., Chen, H., Cui, L., Wang, N., and Yu, Y. Easicprs: design and implementation of a portable chinese pulse-wave retrieval system. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems* (2011), ACM, pp. 149–161.

[69] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence 22*, 11 (2000).