Low Power Depth Estimation of Rigid Objects for Time-of-Flight Imaging

James Noraky, *Student Member, IEEE,* Vivienne Sze, *Senior Member, IEEE*

***Abstract*—Depth sensing is useful in a variety of applications that range from augmented reality to robotics. Time-of-flight (TOF) cameras are appealing because they obtain dense depth measurements with minimal latency. However, for many battery-powered devices, the illumination source of a TOF camera is power hungry and can limit the battery life of the device. To address this issue, we present an algorithm that lowers the power for depth sensing by reducing the usage of the TOF camera and estimating depth maps using concurrently collected images. Our technique also adaptively controls the TOF camera and enables it when an accurate depth map cannot be estimated. To ensure that the overall system power for depth sensing is reduced, we design our algorithm to run on a low power embedded platform, where it outputs 640×480 depth maps at 30 frames per second. We evaluate our approach on several RGB-D dataset, where it produces depth maps with an overall mean relative error of 0.96% and reduces the usage of the TOF camera by 85%. When used with commercial TOF cameras, we estimate that our algorithm can lower the total power for depth sensing by up to 73%.**

***Index Terms*—time-of-flight camera, depth estimation, motion estimation, sensor fusion, RGB-D**

I. Introduction

Depth sensing is useful in a variety of applications that range from augmented reality to robotic navigation. One common way to measure depth is to use a time-of-flight (TOF) camera. TOF cameras obtain depth by emitting light and measuring its round-trip time. Compared to other depth sensors, TOF cameras are appealing because they are compact, have no moving parts, and obtain dense depth measurements with minimal computation and latency [1]. The depth measurements obtained by a TOF camera are represented as a depth map, which is an image whose pixel values represent the distance from the sensor to different points in the scene.

However, for applications that run on mobile devices, one drawback of using a TOF camera is that its illumination source is often power hungry, where continuous acquisition of depth can limit the battery life of the mobile device. One way to address this issue is to reduce the usage of the TOF camera, but this is problematic for applications that require depth in real time, or 30 frames per second. Here, we propose an algorithm to address this issue by estimating depth maps without using the TOF camera as shown in Figure 1.

Our technique estimates depth maps by using the pixel wise motion of concurrently collected images, or the optical flow, to estimate changes in the scene and update a previously measured depth map. For many applications, images are routinely collected, and our goal is to reuse them to obtain depth. We focus on estimating the depth of rigid objects and environments, and we show that it is possible to estimate accurate depth maps while significantly reducing the usage of the TOF camera in these scenarios. While the assumption of rigidity may seem restrictive, our approach requires only the local environment that the TOF camera *can sense* to be rigid. For many tasks that include simultaneous localization and mapping (SLAM), obstacle detection and avoidance, and object manipulation, this is a reasonable assumption [2]–[4].

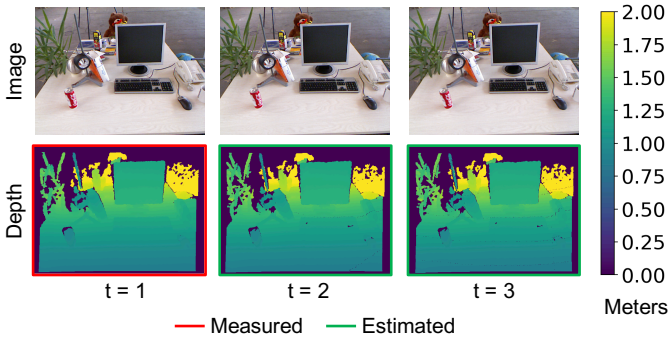


Fig. 1: **Depth Estimation Setup**: We estimate causal depth maps using concurrently collected images and previously measured depth. The TOF camera is used when an accurate depth map cannot be estimated.

To ensure that the overall power for depth sensing is actually reduced, we account for the computation power and require our algorithm to estimate accurate depth maps on a low power embedded platform with minimal latency. This means that we cannot blindly use standard techniques to estimate a new depth map because these platforms have limited compute resources. Our contribution therefore is an optimized algorithm that combines computationally efficient techniques to obtain an accurate and dense depth map with minimal latency. Our approach balances the usage the TOF camera, the computational costs of the algorithm, and the quality of the estimated depth. In particular, we present the following:

- We introduce an algorithm that lowers the usage of the TOF camera and instead obtains depth maps by estimating the 3D motion in the scene, which is used to update a previously measured depth map.
- We reduce the computation required to estimate the 3D motion of every pixel by estimating the pose between frames. We show that it is possible to obtain an accurate depth map by using the pose estimated with the optical

flow determined by a block matching heuristic on a sparse, uniformly-spaced grid. This is essential for our approach to run in real time on an embedded platform, which ensures that the overall power for depth sensing is reduced.
- We develop a mechanism to detect when an accurate depth map cannot be estimated and to adaptively enable the TOF camera in these cases. This is crucial because it is not always possible to estimate accurate optical flow (especially with limited compute resources), which is required for our technique.

To demonstrate the effectiveness of our approach and quantify the reduction in power, we implement our algorithm on the ODROID XU-3 board [5] using only the Cortex-A7 CPUs, which estimates depth maps in real time. In addition to estimating depth maps temporally, we also show how our algorithm can be used to infill depth spatially, making it possible to extend the range of a TOF camera and overcome saturation.

This paper is organized as follows. In Section II, we describe other related approaches that use images to aid in the estimation of depth maps. This is followed by a presentation of our work, where we first describe how we use the optical flow to estimate the 3D motion in a scene (Section III), and then how we use this to robustly estimate depth maps (Section IV). In Section V, we evaluate our algorithm on a variety of RGB-D datasets, where we also analyze the tradeoffs of our approach and compare it to other techniques. To estimate the reduction in the power for depth sensing, we quantify the overall system power of our approach in Section VI. In Section VII, we show how our algorithm can also be used to infill depth spatially. Finally, we conclude this paper in Section VIII.

II. Background

The idea of using images to enhance and estimate depth maps has been explored in many applications. For example, the authors of [6]–[11] use images to upsample low resolution depth maps, and the authors of [12], [13] enhance TOF camera depth maps using stereo images. Given this breadth, we focus only on techniques that have similar problem setups, namely those that estimate new depth maps temporally using concurrently collected images and previously measured depth maps and those that only use consecutive and monocular images to estimate depth.

A. Temporal Depth Map Estimation

Here, we survey techniques that use images to temporally estimate new depth maps from previously measured ones. The authors of [14] address the fact that depth maps obtained from TOF cameras often have lower resolutions and are acquired at lower frame rates than that of digital cameras. To overcome these limitations, they proposed using joint bilateral upsampling techniques to first increase the resolution of the captured depth maps. To estimate the remaining depth maps and equalize the frame rate between the TOF and the digital camera, the authors applied bidirectional block matching algorithms to estimate the optical flow between images without any corresponding depth maps and those with it. These optical flow vectors are used to identify the depth blocks that are averaged to form a new depth map.

Similar to [14], the authors of [15] and [16] also estimate depth maps between frames that have both images and depth maps available using block matching algorithms. However, the authors of [15] selects the depth block from either the preceding or future depth map based on the edges of the corresponding image blocks. The authors of [16] estimate depth by performing a weighted average guided by the underlying texture in the images.

All of these approaches use block matching algorithms to obtain dense optical flow fields, but this process is computationally expensive. To reduce the complexity, the authors of [17] reuse the motion vectors generated in compressed video to accelerate the process of depth map estimation. In this work, the authors assume that depth maps are acquired only at I- or P-frames and estimate the depth map for B-frames using temporal averaging similar to [14].

Unfortunately, we cannot directly use these techniques to obtain new depth maps because computing a dense optical flow field is *prohibitively slow*. For example, OpenCV's implementation of dense optical flow [18] runs at 0.88 frames per second on our embedded device. For applications like 3D video frame upsampling, which can be performed offline, this is not necessarily a problem, but for applications like robotic navigation, these approaches are unsuitable because the underlying applications are sensitive to latency. Furthermore, most of these approaches also estimate depth maps by using the depth from preceding and future frames. This is not possible for real time applications, and we require the estimation of depth maps to be causal. As we show in Sections III and IV, our algorithm uses the assumption of rigidity to significantly reduce the computation required to obtain causal and dense depth maps.

B. Pose Estimation and Structure-from-Motion

As stated in Section I, we use the optical flow to estimate the 3D motion in the scene from frame to frame. For rigid objects, this is represented by the relative pose, which is composed of a rotation and translation. A common way to estimate the pose exploits epipolar geometry and uses the pixel wise correspondences between consecutive images (which can be trivially obtained using the optical flow) to obtain an intermediate quantity known as the essential matrix, which can then factored to obtain the rotation and translation [19]. Depending on the number of correspondences, the essential matrix can be estimated using techniques that range from performing a singular value decomposition (8 correspondences) [20] to finding the roots of a tenth order polynomial (5 correspondences) [21].

One potential benefit of this approach is that it only requires images to obtain pose, although the estimated translation is known only to scale (the magnitude of the translation vector is not known). Furthermore, once the pose is obtained, relative depth can also be estimated by triangulating the corresponding pixels. These techniques are known as structure-from-motion

(SfM) [22]–[25], and we refer the interested reader to a comparison [26] of popular and state-of-the-art pipelines. Unfortunately, one drawback of these approaches is that they typically only estimate depth at a sparse set of keypoints. This is problematic for applications like obstacle avoidance, which require dense depth maps. Furthermore, these techniques also only estimate relative depth.

Unlike the SfM techniques, our approach uses the relative pose to update a previously measured depth map to obtain a new and *absolute* depth map. We show that using previous depth measurements, which is freely available in our problem setup, allows us to estimate the rotation and absolute translation with fewer correspondences, which is important for obtaining accurate depth maps. Previously, we exploited our problem setup to directly estimate the angular and absolute translational velocity, another way of representing rigid motion, using linear least squares [27]. Here, we extend our previous approach to estimate the rotation and translation, which allows us to further reduce the usage of the TOF camera with a negligible increase in computation.

III. Relative Pose Estimation

In this section, we describe how we estimate the relative pose of a rigid object using the 2D motion of its pixels, or the optical flow, by inverting a simple image formation model. We take a different approach from [27] and estimate the rotation and translation. This allows us to further reduce the usage of the TOF camera compared to [27] with a negligible increase in complexity. Once the relative pose is obtained, we can determine the 3D motion in the scene and estimate a new depth map, which we describe in Section IV.

Our approach assumes that images are formed by perspective projection. This means that the i^{th} pixel located at (u_i, v_i) corresponds to the 3D point, X_i, in the camera-centric coordinate system:

$$X_i = \frac{z_i}{f}(u_i, v_i, f)^T \tag{1}$$

where we denote z_i as the depth of the i^{th} pixel and f as the focal length. We simplify notation and assume that all image coordinates are relative to the principal point. Given this, we can obtain the 3D position for each pixel in the depth map.

As the object undergoes rigid motion, its motion can be represented by its relative pose, which is composed of a rotation, R, and a translation, T. This new 3D point corresponds to the pixel located at (u_j, v_j), where:

$$u_j = f\frac{\hat{x}\cdot(RX_i+T)}{\hat{z}\cdot(RX_i+T)} \quad \text{and} \quad v_j = f\frac{\hat{y}\cdot(RX_i+T)}{\hat{z}\cdot(RX_i+T)} \tag{2}$$

Here, we denote $\cdot$ as the dot product and $(\hat{x}, \hat{y}, \hat{z})$ as the unit vectors oriented along the coordinate axes.

Given the pixel-wise correspondences between frames, we can obtain the pose by rearranging Eq. (2) and solve for R and T in a least squares sense. Because rotation matrices are nonlinear, we must solve for the pose iteratively. However, instead of estimating the rotation matrix, we use a more compact representation for rotation, namely Rodrigues' Formula [28], where:

$$R = I + \sin\theta K + (1-\cos\theta)K^2 \tag{3}$$

This describes a rotation of θ radians about an axis, $\hat{k}$. The vector $\hat{k}$ is a unit vector, whose elements form the skew-symmetric matrix, K, such that $KX_i = \hat{k} \times X_i$ (where $\times$ denotes the cross product), and I is the identity matrix.

We substitute Eq. (3) into Eq. (2) and rearrange the terms to obtain the following expression that relates the pixel-wise motion, denoted as $\Delta u_i = u_j - u_i$ and $\Delta v_i = v_j - v_i$, to the pose:

$$\Delta u_i = \frac{f}{z_i}\hat{x}\cdot(WX_i+T) - \frac{u_j}{z_i}\hat{z}\cdot(WX_i+T) \tag{4}$$

$$\Delta v_i = \frac{f}{z_i}\hat{y}\cdot(WX_i+T) - \frac{v_j}{z_i}\hat{z}\cdot(WX_i+T) \tag{5}$$

where $W = \sin\theta K + (1-\cos\theta)K^2$. We can then solve for the pose ($\hat{k}$, θ and T) in a least squares sense by minimizing the mean residual (r_i) error over the N optical flow estimates:

$$\min_{\hat{k},\theta,T} \frac{1}{N}\sum_{i=1}^{N} \underbrace{(\Delta u_i - \Delta\hat{u}_i)^2 + (\Delta v_i - \Delta\hat{v}_i)^2}_{r_i} \tag{6}$$

where we denote $\Delta\hat{u}_i$ and $\Delta\hat{v}_i$ as the right hand side of Eq. (4) and Eq. (5), respectively.

We minimize Eq. (6) using a variant of the Gauss-Newton algorithm, where we linearize the non-linear residual using the Jacobian at $\theta = 0$. This assumes that the rotation between frames is small, which is a reasonable assumption for many indoor applications, where images are acquired at 30 frames per second. Furthermore, each iteration of the Gauss-Newton algorithm is equivalent to the least squares solution presented in [27], which is computationally simple and equivalent to solving a 6×6 linear system. To estimate the pose, we need at least 3 optical flow vectors and its corresponding depth. This is advantageous because it reduces the computation required to obtain a depth map compared to some of the methods in Section II that require dense optical flow.

IV. Proposed Algorithm

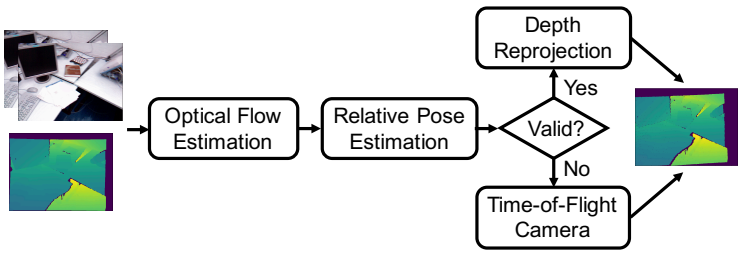


Fig. 2: **Depth Map Estimation Pipeline**: Our algorithm estimates a new depth map using consecutive images and a previously measured depth map. When a reliable depth map cannot be estimated, we use the TOF camera to obtain depth.

In this section, we describe our proposed algorithm, which takes as input consecutive images and a previous depth map and outputs a new one as shown in Figure 2. Our proposed technique is computationally efficient and we highlight our design choices so that our algorithm can run in real time on an embedded platform. We also describe our strategy to adaptively use the TOF camera when an accurate depth map cannot be estimated.

A. Optical Flow Estimation

As shown in Figure 2, we begin by first estimating the optical flow between consecutive images using the three step search (TSS) algorithm [29]. The TSS algorithm obtains the optical flow for a block of pixels in an image by searching for the block in the next image that minimizes a cost function. However, instead of an exhaustive search, the TSS algorithm only considers select locations to reduce computation. In our implementation, we search for the block that minimizes the sum of absolute differences using 15×15 blocks with a step size of 8. We also only compute the optical flow for the pixels on a 12×12 grid that is uniformly spaced across the image. This reduces the computation because our technique does not require dense optical flow estimates, and we do not need to find keypoints or corners.

Our decision to use the TSS algorithm is motivated by its run time on an embedded platform. We compare the run time of the TSS algorithm to the commonly used Lucas Kanade algorithm [30] by profiling both approaches on the ODROID-XU3 board [5], which is an embedded platform that is representative of the compute resources available on mobile devices. We use the Cortex-A7 cores to compute the optical flow for 640×480 images for the pixels on a uniformly spaced 12×12 grid. For the Lucas Kanade algorithm, we used 15×15 blocks and 3 pyramid levels.

On average, we find that the TSS algorithm require 13 ms whereas the Lucas Kanade algorithm requires 51 ms. We also profile the time required to identify corners. We found that the Harris corner detector [31] requires 120 ms, which is intolerable for real time applications, whereas the time to locate the pixels on a uniform grid is negligible. We summarize these run times in Table I.

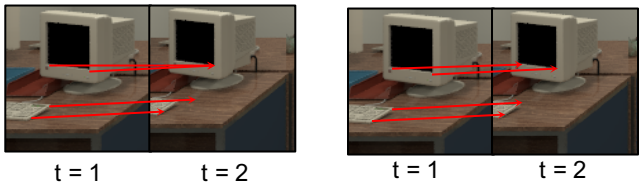
Algorithm	**Runtime (ms)**
Three Step Search	13
Lucas Kanade	51
Harris Corner	120

TABLE I: **Runtime Comparisons**: We profile our design choices on the ODROID-XU3 board [5]. We opt to use the TSS algorithm to ensure our implementation can estimate depth maps in real time.

However, as shown in Figure 3a, one drawback of using the TSS algorithm is that our optical flow estimates can be inaccurate. In the next section, we show how we can mitigate this to robustly estimate the pose. With the pose estimated, in addition to obtaining a new depth map, we can also correct the optical flow field as shown in Figure 3b.

B. Relative Pose Estimation

With the optical flow estimated, we can solve for the pose as described in Section III. However, using the optical flow directly is problematic because it can be different from the underlying motion field [28]. Image sensor noise, occlusions, and the algorithm used to estimate the optical flow affect the accuracy of the estimates. Furthermore, because optical flow is estimated using image intensities, it can be different from the underlying motion field even in the absence of these issues. In regions with uniform intensity, for example, the optical flow would be zero even when the underlying motion field is not. Moreover, the depth values can also be affected by sensor noise in addition irregularities that arise from multipath reflections, specular reflections, and interference [1]. Because our pose estimation directly uses the depth, these errors can also adversely affect the depth map.



(a) TSS (b) Using Pose

Fig. 3: **Select Optical Flow**: We show examples of optical flow vectors estimated using the TSS algorithm and those obtained using the estimated pose.

While these errors are in part mitigated by our least squares formulation, we need a mechanism to distinguish accurate optical flow and depth from erroneous ones because the squared penalty in our formulation is not robust against outliers. This is possible when the pose is known, and we can distinguish the accurate optical flow estimates, or inliers, from the erroneous outliers because the former satisfy Eq. (4) and Eq. (5). This insight suggests that we solve Eq. (6) using RANSAC [32] to iteratively estimate both the pose and the set of inliers.

We proceed by randomly selecting the optical flow estimates and its corresponding depth to obtain an initial pose hypothesis. We use 3 optical flow estimates, which is the minimum required to estimate pose using our technique, to minimize the likelihood of choosing an outlier. To judge the quality of the pose hypotheses, we relax the requirement that the pose must satisfy Eq. (4) and Eq. (5) for all of the inliers and instead compute the residual error for each optical flow estimate. If the number of optical flow estimates with low residual errors, which is determined by a threshold, exceed a fraction of the total number of estimates, we resolve Eq. (6) using only these inliers. We repeat this procedure and select the candidate pose with the lowest mean residual error. When there are no candidates, we enable the TOF camera to acquire a new depth map. This adaptive control of the TOF camera is different from what we presented in [27] and allows for robust depth map estimation. We summarize our approach in Algorithm 1.

To show that Algorithm 1 can mitigate the impact of errors in depth and the optical flow, we first simulate the idealized depth and optical flow for a given pose and corrupt a subset of them. To reflect the fact that our approach uses the TSS algorithm, we also round each optical flow vector to the nearest integer displacement. We then estimate the pose with and without RANSAC and compare it to the pose we used to simulate the data with using the root mean squared error (RMSE) of the translation as defined in [33]. In Table II, we see that RANSAC substantially lowers the RMSE of the

Depth	Optical Flow	Reduction (%)
×		68.0
	×	59.4
×	×	45.1

TABLE II: **Impact of RANSAC**: We present the reduction in the RMSE obtained using RANSAC when there is noise in the depth measurements, the optical flow, and in both.

translation across all scenarios.

In our implementation, we use 30 RANSAC iterations and set the threshold to 4 and accept a pose hypothesis if the size of its inlier set is at least 10% of the number of optical flow estimates. When obtaining the initial pose, we only perform 1 iteration of the Gauss-Newton algorithm, which is equivalent to the method presented in [27]. We resolve Eq. (6) using the inlier set by performing 3 iterations. This is negligibly more computation than [27], and as we describe in Section V-A, the pose estimation accounts for a small fraction of the runtime. This is significant because it allows us to lower of the complexity of the optical flow estimation algorithm and still obtain accurate pose estimates. This is essential to obtaining accurate depth maps in real time.

Algorithm 1 Adaptive Pose Estimation

input: Optical flow $(\Delta u_i, \Delta v_i)$, depth (z_i), and RANSAC parameters (No. of iterations, thresh, and min. size)
output: Pose (R and T) or signal to use TOF camera

repeat ▷ Get the inlier set
  Randomly sample 3 optical flow vectors and its depth
  Solve Eq. (6); Compute residuals, r_i
  Get inlier set, $\mathcal{I} = \{i : r_i < \text{thresh}\}$
  Retain $\mathcal{I}$ with lowest mean residual; $|\mathcal{I}| >$ min. size
until End of RANSAC

if $|\mathcal{I}| = 0$ **then** ▷ Get pose or depth map
  Use the TOF camera
else
  Solve Eq. (6) using $\mathcal{I}$
end if

C. Depth Reprojection

Once the pose is estimated, we obtain a new depth map by applying the pose to each 3D point in the first depth map and projecting its depth, or its $z-$coordinate, to an image. For every pixel in the first depth map, we first compute its 3D point, X_i, using Eq. (1). The reprojected depth map is then obtained as follows:

$$D\left[f\frac{\hat{x}\cdot(RX_i+T)}{\hat{z}\cdot(RX_i+T)}, f\frac{\hat{y}\cdot(RX_i+T)}{\hat{z}\cdot(RX_i+T)}\right] = \hat{z}\cdot(RX_i+T) \quad (7)$$

where D represents the depth map whose entries are indexed by its x and y coordinates. If multiple points are mapped onto the same pixel location, we retain the smallest depth value.

When more than one depth map is predicted consecutively, we obtain a new depth map by reprojecting the last measured depth map. To do so, we update the pose accordingly. Let R_c and T_c represent the current pose that is estimated using the previously estimated depth map. We also assume that the previously estimated depth map was obtained by reprojecting the last measured depth map using R_{t-1} and T_{t-1}. Then, the pose which we now use to reproject the previously measured depth map, denoted as R_t and T_t, is:

$$R_t = R_c R_{t-1} \qquad T_t = T_c + R_c T_{t-1} \quad (8)$$

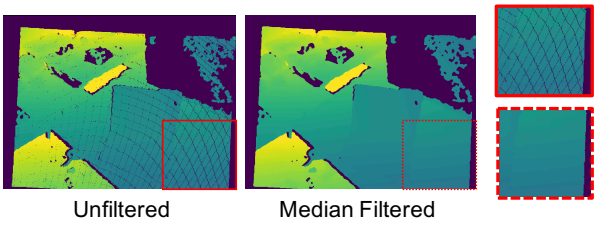


Fig. 4: **Reprojected Depth Maps**: The reprojected depth maps have artifacts where depth is not available. While median filter can infill these regions, we ignore this post-processing step because the holes constitute a small portion of the depth map.

The resulting depth map contains depth estimates for pixels that correspond to the overlapping field of views between the image where the last depth map was measured and the current image. It should be noted that without any additional post-processing, this method also introduces artifacts as shown in Figure 4. These holes arise because the pixels belonging to the same object are treated independently and are not constrained to be contiguous after reprojection and because regions that were previously occluded have been uncovered. While reverse warping would eliminate these holes, it also erroneously infills the previously occluded regions. We want to avoid this, especially as we predict many depth maps consecutively, in the event where the previously occluded region has a different depth from its surroundings. We confirm this by applying our algorithm to sequences from the TU Munich RGB-D dataset [33] and find that reverse warping increases the overall mean relative error (as defined in Section V-C) by 17.4% compared to our approach. Furthermore, if the application needs the depth in the previously occluded regions, this could serve as another signal to use the TOF camera.

One potential way to remove the first type of holes is by applying a median filter with a small kernel size to the resulting depth map as shown in Figure 4. While this may give inconsistent behaviors at depth boundaries, we find in our experiments with the TU Munich RGB-D dataset, the overall mean relative error remains unchanged. However, as our computational resources are limited, we ignore this additional step because these types of holes are minimal, accounting for less than 3% of the estimated pixels while imposing an additional 20 ms overhead. In the next section, we see that this is intolerable for real-time performance.

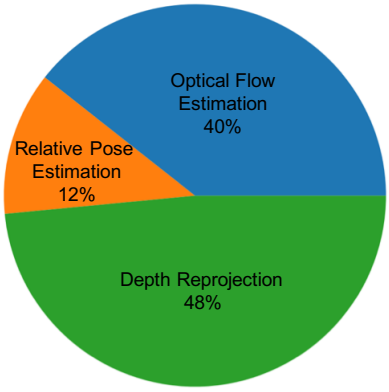


Fig. 5: **Runtime Breakdown**: We profile the implementation of our algorithm on the ODROID-XU3 [5] board, which produces 640×480 depth maps at 30 FPS. Because the time to reproject a new depth map is fixed, we work to reduce the computation time required to estimate the optical flow.

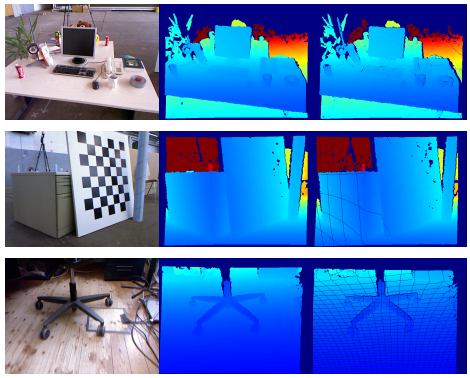


Fig. 6: **Estimated Depth Maps**: We show the estimated depth maps for select sequences in [33]. A video of our algorithm running can be found in [39].

V. Algorithm Evaluation

A. Implementation

We implement our algorithm on the ODROID XU-3 board [5], which is an embedded platform with an Exynos 5422 processor. The Exynos processor is used in the Samsung Galaxy S5 [34] and is representative of the compute power available on mobile devices. Our implementation uses the Cortex-A7 cores of the board and outputs 640×480 depth maps in real time, or 30 frames per second (FPS). To achieve this frame rate, we parallelize our computation across the 4 Cortex-A7 cores. We use the parameter settings described in Section IV and the OpenCV library whenever possible.

As shown in Figure 5, most of the time of our implementation is spent on estimating the optical flow and reprojecting the depth map. This figure further justifies our decision to use the TSS algorithm. Since the time required to reproject a depth map is fixed, we are limited in what we can allocate to obtain the optical flow if we want to estimate depth maps at 30 FPS. We discuss the impact of this decision on the accuracy of the estimated depth maps in Section V-E. This figure also shows that when only the pose is required, which is the case for SLAM, our algorithm can run at nearly 58 FPS.

B. Dataset

We evaluate our algorithm on RGB-D datasets used to benchmark SLAM, visual odometry, 3D reconstruction, and navigation algorithms. These tasks are relevant for many mobile applications, and the images and depth maps are representative of what our approach will encounter. We adapt these datasets to test our approach by using consecutive images and select depth maps to predict new ones, which we then compare to that in the dataset. For our experiments, we use the provided intrinsic parameters and tools to synchronize the images with the depth maps for each dataset.

We use sequences from the following datasets: TU Munich RGB-D [33], NYU Depth V2 [35], Indoor RGB-D [36], CoRBS [37], and ICL-NUIM [38]. These datasets contain 640×480 RGB images and depth maps and most are collected at 30 FPS.

C. Methodology

We apply our algorithm to the first 100 frames of the sequences in each dataset. We quantify the accuracy of the depth maps using the following error metrics:

- **Mean Relative Error (MRE)**: This is defined as $\frac{100}{N}\sum_{j=1}^{N}\frac{|z_j-\hat{z}_j|}{z_j}$, where N is the number of pixels predicted, $\hat{z}_j$ is the predicted depth for the j^{th} pixel, and z_j is the depth measured by the TOF camera. The MRE is presented as a percentage.
- **Mean Absolute Error (MAE)**: This is defined as $\frac{1}{N}\sum_{j=1}^{N}|z_j-\hat{z}_j|$ and presented in centimeters.
- **Root Mean Squared Error (RMSE)**: This is defined as $\sqrt{\frac{1}{N}\sum_{j=1}^{N}(z_j-\hat{z}_j)^2}$ and presented in centimeters.

Because our algorithm uses the TOF camera adaptively, we also quantify the frequency at which it is used using:

- **Duty Cycle (DC)**: This is equal to $\sum_{i=1}^{100}\mathbb{1}(i)$, where $\mathbb{1}(i)$ equals 1 if the i^{th} depth map is obtained using the TOF camera and 0 if the depth map is estimated instead. The DC is presented as a percentage.

To reduce the power required to obtain accurate depth maps, our goal is to lower the TOF camera's duty cycle while maintaining a baseline accuracy. In our analysis, we focus on the MRE because it allows us to compare the performance of our algorithm across datasets that have different ranges of depth. Therefore, for each dataset, we set the threshold parameter in Algorithm 1 to achieve a median MRE of approximately 1% across its sequences in order to measure its duty cycle.

D. Results

We summarize the performance of our algorithm for each dataset in Table III, where we compute the median of each error metric across the depth maps. Examples of the estimated depth maps are shown in Figure 6. Across the datasets, we

Dataset	MRE (%)	MAE (cm)	RMSE (cm)	DC (%)
TU Munich RGB-D [33]	0.96	2.27	7.63	16.0
NYU Depth V2 [35]	0.95	4.04	9.01	10.0
Indoor RGB-D [36]	1.03	2.11	7.54	33.0
CoRBS [37]	1.04	1.79	8.98	15.0
ICL-NUIM [38]	0.67	2.04	5.65	10.0
Mean	0.93	2.45	7.76	16.8
Median	0.96	2.11	7.63	15.0

TABLE III: **Algorithm Evaluation**: We summarize the MRE, MAE, RMSE and DC that our algorithm achieves.

achieve a median MRE of 0.96% and a median duty cycle of 15.0%. In Table III, we see that the duty cycle for Indoor RGB-D [36] is higher than that of the other datasets. This is expected because this dataset contains sequences of a robot moving abruptly in a sparsely textured environment. Furthermore, this shows that our technique can adapt to and still reduce the usage of the TOF camera in these challenging scenarios.

Because different applications have different accuracy requirements for depth maps, we also quantify the tradeoff between the duty cycle and the MRE for our approach. To do so, we vary the threshold in Algorithm 1 that determines if an optical flow estimate is an inlier. In our pipeline, we expect that a lower threshold, which assumes accurate optical flow estimates, will result in depth maps with a lower MRE but also a higher duty cycle because the TSS algorithm cannot consistently obtain accurate optical flow estimates. By the same reasoning, we expect the MRE to be higher but the duty cycle to be lower when the threshold is high. We present this tradeoff in Figure 7, where each point labeled *This Work* in the legend represents the median duty cycle and MRE pair across all of the sequences in each dataset for different thresholds.

E. Impact of Optical Flow Algorithm

To quantify the impact of the TSS algorithm on the overall accuracy of estimated depth maps, we compare our algorithm to a variant that uses the Lucas Kanade algorithm to estimate optical flow. We expect the TSS algorithm to perform worse than the Lucas Kanade algorithm in estimating the optical flow because the TSS algorithm only considers select locations in its search for the best matching block, and to increase the overall MRE of the estimated depth map. In Table IV, we compare this variant (*LK*) to our approach (*This Work*) and present the MRE for the same duty cycle.

From this comparison, we see that our hypothesis is confirmed and that using the Lucas Kanade algorithm in our pipeline reduces the overall median MRE from 0.96% to 0.86%. However, while the Lucas Kanade algorithm reduces the MRE of the estimated depth maps by over 10%, it does not justify the 50% decrease in the estimation frame rate when profiled on the ODROID board. As shown in Table V, its frame rate is 15 FPS, which is intolerable for real time applications.

F. Benefit of Scene Adaptive Estimation

One key feature of our algorithm is that it adaptively uses the TOF camera when an accurate depth map cannot be estimated. This is necessary because it is not always possible to obtain accurate optical flow estimates. We compare our adaptive scheme to our previous work [27], which predicts depth maps at regular intervals. We apply this approach to the datasets in Section V-B and plot the duty cycle and MRE pairs, which are denoted as *Non-Adaptive* in Figure 7.

In this figure, we see that the adaptive scheme of our approach outperforms [27] across all duty cycles with a negligible increase in complexity. For the same duty cycle, we see in Table IV that our adaptive schemes reduces the median MRE from 1.80% to 0.96%. Furthermore, this result make sense upon inspecting the images in the datasets. Images with rapid motion are blurred and contain large displacements, making the estimation of accurate optical flow challenging. Our algorithm is optimized to detect these scenarios and uses the TOF camera while estimating depth maps for frames with slower motion.

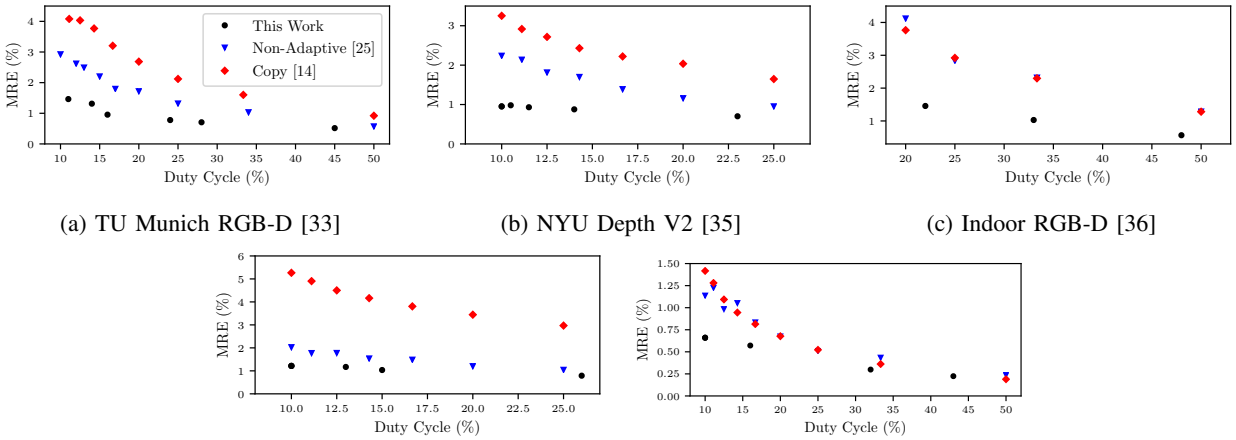
G. Comparison to Previous Work

1) Temporal Depth Map Estimation: We compare our algorithm to a causal variant of [15] as described in Section II-A. This technique estimates depth by copying previous measurements guided by the optical flow. Since our setup requires depth maps to be estimated in real time, we use the optical flow between the current and preceding images to copy the depth from a previous frame. In our experiments, we compute a dense optical flow field using [18]. The estimation of dense optical flow is prohibitively slow on our embedded processor, and this technique runs at 0.83 FPS as shown in Table V. However, we still perform this experiment to quantify the effectiveness of remapping depth. We expect this approach to perform well when the motion between frames is small.

We apply this approach to the datasets and plot the duty cycle and MRE pairs, which we denote as *Copy*, in Figure 7. From this figure, we see that our approach outperforms *Copy* across all duty cycles and datasets. This result shows that our dataset contains non-trivial changes in depth that cannot be captured by simply remapping the pixels of a previous depth map. Furthermore, this experiment suggests that the changes in depth can be estimated by our technique.

2) Structure-from-Motion: We also compare our algorithm to a structure-from-motion (SfM) pipeline that estimates relative depth. Even though SfM estimates relative depth at a sparse set of points, these techniques only use images and can be compelling if it can run in real-time on a low-power embedded platform. We implement an incremental SfM pipeline following standard and state of the art approaches described in [26]. We use SIFT [40] to localize keypoints, match consecutive keypoints using brute force matching, perform geometric validation using the 8 point algorithm, and triangulate using the DLT method [19]. We apply the SfM pipeline to our setup and estimate the depth using two consecutive images. Across the different datasets, our SfM pipeline estimates the depth at approximately 210 keypoints.

As summarized in Table V, our implementation (*SfM-SIFT*) runs at 0.12 FPS on the ODROID XU-3 board, where most of the time is spent on computing and matching the keypoints.



(a) TU Munich RGB-D [33] (b) NYU Depth V2 [35] (c) Indoor RGB-D [36]

(d) CoRBS [37] (e) ICL-NUIM [38]

Fig. 7: **Tradeoff Between Duty Cycle and MRE**: We show the tradeoff between the duty cycle and MRE for our technique (This Work) and the techniques we compare against: Non-Adaptive (Section V-F) and Copy (Section V-G1). Because our technique is adaptive, our duty cycles do not align with the competing techniques, which estimate depth at regular intervals.

Dataset	**This Work**			**LK**			**Non-Adaptive [27]**			**Copy [15]**			**SfM-SIFT [26]**		
	MRE (%)	MAE (cm)	RMSE (cm)	MRE (%)	MAE (cm)	RMSE (cm)	MRE (%)	MAE (cm)	RMSE (cm)	MRE (%)	MAE (cm)	RMSE (cm)	MRE (%)	MAE (cm)	RMSE (cm)
TU Munich RGB-D [33]	0.96	2.27	7.63	0.86	1.68	7.55	1.80	6.26	13.83	3.20	5.80	25.97	36.14	83.77	104.99
NYU Depth V2 [35]	0.95	4.04	9.01	0.82	3.34	8.11	2.24	5.98	14.65	3.25	10.04	40.25	43.03	171.00	212.91
Indoor RGB-D [36]	1.03	2.11	7.54	1.49	3.53	13.20	2.32	5.31	14.96	2.30	5.41	20.04	32.29	119.32	154.14
CoRBS [37]	1.04	1.79	8.98	1.02	1.66	9.82	1.54	2.94	12.28	4.16	9.44	34.87	38.61	80.52	106.66
ICL-NUIM [38]	0.67	2.04	5.65	0.14	0.39	3.02	1.14	3.21	8.26	1.42	3.98	10.62	39.76	126.38	158.74
Mean	0.93	2.45	7.76	0.87	2.12	8.34	1.81	4.74	12.80	2.87	6.93	26.35	37.97	116.20	147.49
Median	0.96	2.11	7.63	0.86	1.68	8.11	1.80	5.31	13.83	3.20	5.80	25.97	38.61	119.32	154.14

TABLE IV: **Algorithm Comparison**: We compare the performance of our algorithm to variants and competing techniques for approximately the same duty cycle to show that our approach estimates accurate depth maps.

Algorithm	**Frame Rate (FPS)**
This Work	30
LK	15
Non-Adaptive [27]	30
Copy [15]	0.83
SfM-SIFT [26]	0.12
SfM-SURF [26]	0.36
SfM-ORB [26]	1.81

TABLE V: **Algorithm Frame Rate Comparison**: We compare the estimation frame rates our approach and other techniques on the ODROID-XU3 board [5].

Due to the low frame rate, we also experimented with using SURF [41] (*SfM-SURF*) and ORB [42] (*SfM-ORB*) features instead of SIFT. These variants estimate sparse depth at 0.36 and 1.8 FPS, respectively. While these variants have a higher frame rate than the standard pipeline, they are still far from real time. To quantify the accuracy of the depth estimates obtained using the standard SfM pipeline, we find the scale factor so that the estimated relative depth best matches the ground truth. We summarize the MRE for each dataset in Table V, where we also compare it (*SfM-SIFT*) to our approach and other competing techniques. Because our pipeline uses only two images, the high MRE is expected. We can lower the MRE by incorporating more frames and performing bundle adjustment [19], but this would increase latency and further decrease the estimation frame rate. Due to the high MRE and the low frame rate, we see that SfM is impractical for the scenario we consider.

VI. System Power Reduction

To lower the power for TOF imaging, our strategy is to lower the duty cycle of the TOF camera and estimate depth maps instead. However, this implies that the power required to estimate a new depth map is less than that of using a TOF camera. Here, we measure the power of an implementation of our algorithm on the ODROID XU-3 board and use it to estimate the overall system power of a system that uses our algorithm alongside the TOF camera to obtain depth.

The ODROID XU-3 board has 4 Cortex-A7 CPUs and 4 Cortex-A15 CPUs. To keep the computation power low, we only use the Cortex-A7 cores to estimate the depth maps. This leaves the Cortex-A15 cores available for other mobile applications that use depth maps and further underscores that our implementation, which outputs 640×480 depth maps in real time, is efficient. The resulting implementation consumes a total of 0.69 W, of which the idle power is 0.19 W. We

Category		Power (W)
Core	Active	0.63
	Idle	0.16
DRAM	Active	0.06
	Idle	0.03
Total	Active	0.69
	Idle	0.19

TABLE VI: **Power Breakdown**: We measure the power of our implementation on the ODROID-XU3 board [5].

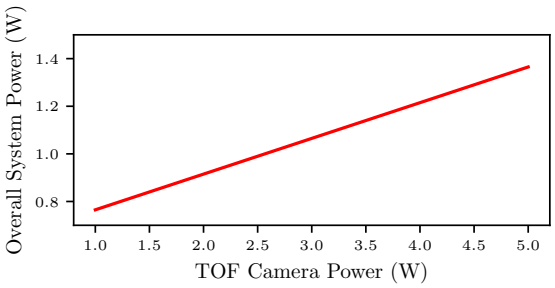


Fig. 8: **Overall System Power**: We estimate the power of a system that uses our algorithm to estimate depth alongside the TOF camera. For commercial TOF cameras, our algorithm can reduces the overall system power by 23%-73%.

summarize the power breakdown of our implementation in Table VI.

Given the power of our implementation, we now estimate the overall system power of a hybrid system that uses the TOF camera and our algorithm to obtain depth. We define the overall system power, denoted as P_S, as follows:

$$P_S = \frac{ON}{100} \cdot (P_{TOF} + P_I) + (1 - \frac{ON}{100}) \cdot (P_C + P_M) \quad (9)$$

where we denote ON as the duty cycle of the TOF camera, P_{TOF} is the power of the TOF camera, P_I is the total idle power, P_C is the active power of the A7 cores, and P_M is the active power of the DRAM. Because we assume that images are routinely collected for other purposes, we ignore its contribution in Eq. (9). Based on a survey of commercial TOF cameras (with ranges up to 4 meters), we also assume that P_{TOF} ranges from 1 to 5 W [43] [44].

Taking the duty cycle to be 15% and using the measurements in Table VI, we plot the power of the hybrid system in Figure 8. For the datasets in Table III, this translates to a median power reduction of 23%-73% compared to just using the TOF camera while producing depth maps with a median MRE of 0.96%.

VII. Infilling Depth Maps

In the previous section, we describe how we use our algorithm to estimate new depth maps *temporally* to lower the power for TOF imaging. Here, we show that our algorithm can also be used to estimate depth *spatially* to infill missing depth values. This means that our algorithm can be used to address two deficiencies of TOF imaging, namely when the sensor goes out of range and when the sensor saturates. We

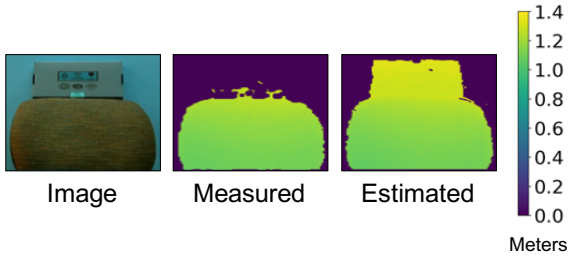


Fig. 9: **Out of Range**: We estimate the depth for objects that exceed the TOF camera's range using our algorithm. The purple regions cannot be sensed by the TOF camera.

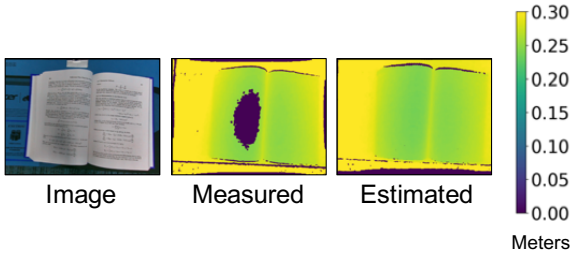


Fig. 10: **Saturation**: We estimate the depth for pixels that are saturated using our algorithm. The purple regions cannot be sensed by the TOF camera.

consider both cases and show how we can, in effect, extend the range of a TOF camera without increasing the power of its illumination source and overcome saturation.

We first consider the scenario where the TOF camera goes out of range by acquiring images and depth maps of a scene shown in the first image of Figure 9. We use the Pico Zense DCAM710 RGB-D sensor [45], which contains a TOF and digital camera that outputs 640×480 depth maps and 1080×1920 RGB images, respectively. We expect that as we move the sensor away from the objects in the scene, we will not be able to measure depth for every object. We show an instance of this in the second image of Figure 9, where the TOF camera goes out of range and the depth for the box is unknown. To infill the depth values for the box, we use a previously measured image and depth map pair, where depth is available for both objects, and the current image to estimate a new depth map, which is shown in the last image of Figure 9. One limitation of our approach is that we can only infill regions where we have previous depth, and in this case, we cannot estimate depth for the wall. To evaluate the accuracy of our depth map, we compute the mean relative error for the overlapping pixels between the measured and estimated depth map. Because the scene is rigid, which means that the relative distance between the box and chair does not change, we expect that this mean relative error is also representative of what we would obtain if the depth for the box is available in the measured depth map. In this example, we achieve a mean relative error of 0.87%.

We also consider the scenario where a TOF camera becomes saturated by acquiring images and depth maps of a scene, shown in the first image of Figure 10, as we move the TOF

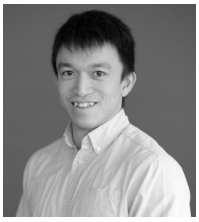
camera closer to the book. As shown in the second image of Figure 10, the sensor saturates and depth is not available in the center of the book. By using a previous image and depth map pair, we are able to overcome this deficiency and estimate depth in this region, achieving a mean relative error of 0.6% for the overlapping pixels.

VIII. Conclusion

In this paper, we present an algorithm to estimate causal depth maps using concurrently collected images and previously measured depth. We use this approach to reduce the power of TOF imaging. Instead of using the TOF camera continuously to acquire depth, we estimate depth maps using our technique and only use the TOF camera when an accurate depth map cannot be estimated. To ensure that the power for depth sensing is reduced, we design our algorithm to run efficiently on a low power embedded platform, carefully balancing the estimation of optical flow with that of pose. The resulting implementation produces 640×480 depth maps in real time, or 30 frames per second. We evaluated our approach on several RGB-D datasets, where our technique produces depth maps with a mean relative error of 0.96% and lowers the usage of the TOF camera by 85%. When used with commercial TOF cameras, our algorithm can reduce the total power for depth sensing by up to 73%.

IX. Acknowledgement

We thank Analog Devices for funding this research. We also thank the research scientists within the company for helpful discussions and feedback.



James Noraky (S'15) received a S.B. degree in Electrical Science and Engineering and a M.Eng. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (MIT) in 2013 and 2014, respectively. He is currently pursuing a Ph.D. degree in the Electrical Engineering and Computer Science (EECS) department at MIT. His research interests include 3D computer vision, image processing, and other areas in signal processing. He received the Siebel Scholarship in 2014 and the MIT EECS Texas Instruments Undergraduate Research and Innovation Scholarship in 2013.

Vivienne Sze (S'04–M'10–SM'16) received the B.A.Sc. (Hons) degree in electrical engineering from the University of Toronto, Toronto, ON, Canada, in 2004, and the S.M. and Ph.D. degree in electrical engineering from the Massachusetts Institute of Technology (MIT), Cambridge, MA, in 2006 and 2010 respectively. In 2011, she received the Jin-Au Kong Outstanding Doctoral Thesis Prize in Electrical Engineering at MIT.

She is an Associate Professor at MIT in the Electrical Engineering and Computer Science Department. Her research interests include energy-aware signal processing algorithms, and low-power circuit and system design for portable multimedia applications including computer vision, deep learning, autonomous navigation, image processing and video coding. Prior to joining MIT, she was a Member of Technical Staff in the Systems and Applications R&D Center at Texas Instruments (TI), Dallas, TX, where she designed low-power algorithms and architectures for video coding. She also represented TI in the JCT-VC committee of ITU-T and ISO/IEC standards body during the development of High Efficiency Video Coding (HEVC), which received a Primetime Engineering Emmy Award. Within the committee, she was the primary coordinator of the core experiment on coefficient scanning and coding, and has chaired/vice-chaired several ad hoc groups on entropy coding. She is a co-editor of "High Efficiency Video Coding (HEVC): Algorithms and Architectures" (Springer, 2014).

Prof. Sze is a recipient of the 2019 Edgerton Faculty Achievement Award at MIT, 2018 Facebook Faculty Award, 2018 & 2017 Qualcomm Faculty Award, 2018 & 2016 Google Faculty Research Award, 2016 AFOSR Young Investigator Research Program (YIP) Award, 2016 3M Non-Tenured Faculty Award, 2014 DARPA Young Faculty Award, 2007 DAC/ISSCC Student Design Contest Award and a co-recipient of the 2018 VLSI Best Student Paper Award, 2017 CICC Outstanding Invited Paper Award, 2016 IEEE Micro Top Picks Award and the 2008 A-SSCC Outstanding Design Award. She is a Distinguished Lecturer of the IEEE Solid-State Circuits Society (SSCS), and currently serves on the technical program committee for the International Solid-State Circuits Conference (ISSCC) and the SSCS Advisory Committee (AdCom). She has also served on the technical program committees for VLSI Circuits Symposium, Micro and the Conference on Systems and Machine Learning (SysML), and as a guest editor for the IEEE Transactions on Circuits and Systems for Video Technology (TCSVT). Prof. Sze will be the Systems Program Chair of SysML in 2020.

References

[1] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time-of-Flight Cameras*, ser. SpringerBriefs in Computer Science. London: Springer London, 2013.

[2] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments," *International Journal of Robotics Research*, vol. 31, no. 5, pp. 647–663, 2010.

[3] J.-J. Hernández-López, A.-L. Quintanilla-Olvera, J.-L. López-Ramírez, F.-J. Rangel-Butanda, M.-A. Ibarra-Manzano, and D.-L. Almanza-Ojeda, "Detecting Objects Using Color and Depth Segmentation with Kinect Sensor," *Procedia Technology*, vol. 3, pp. 196–204, 2012.

[4] J.-A. Fernández-Madrigal and J. L. Blanco Claraco, *Simultaneous Localization and Mapping for Mobile Robots*, ser. Advances in Computational Intelligence and Robotics. IGI Global, 2013.

[5] HardKernel, "ODROID-XU3," www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127, accessed: 2018-07-24.

[6] J. Lu, D. Min, R. S. Pahwa, and M. N. Do, "A Revisit to MRF-Based Depth Map Super-Resolution and Enhancement," in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2011, pp. 985–988.

[7] J. Park, H. Kim, Yu-Wing Tai, M. S. Brown, and I. Kweon, "High Quality Depth Map Upsampling for 3D-TOF Cameras," *International Conference on Computer Vision*, pp. 1623–1630, 2011.

[8] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof, "Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation," in *International Conference on Computer Vision*, 2013, pp. 993–1000.

[9] S. W. Jung and O. Choi, "Learning-Based Filter Selection Scheme for Depth Image Super Resolution," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 10, pp. 1641–1650, 2014.

[10] T. Richter, J. Seiler, W. Schnurrer, and A. Kaup, "Robust Super-Resolution for Mixed-Resolution Multiview Image Plus Depth Data," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 5, pp. 814–828, 2016.

[11] W. Liu, X. Chen, J. Yang, and Q. Wu, "Robust Color Guided Depth Map Restoration," *IEEE Transactions on Image Processing*, vol. 26, no. 1, pp. 315–327, 2017.

[12] J. Zhu, L. Wang, R. Yang, J. E. Davis, and Z. Pan, "Reliability Fusion of Time-of-Flight Depth and Stereo Geometry for High Quality Depth Maps," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 7, pp. 1400–1414, 2011.

[13] S. Schwarz, M. Sjostrom, and R. Olsson, "A Weighted Optimization Approach to Time-of-Flight Sensor Fusion," *IEEE Transactions on Image Processing*, vol. 23, no. 1, pp. 214–225, 2014.

[14] J. Choi, D. Min, and K. Sohn, "2D-Plus-Depth Based Resolution and Frame-Rate Up-Conversion Technique for Depth Video," *IEEE Transactions on Consumer Electronics*, vol. 56, no. 4, pp. 2489–2497, 2010.

[15] H. M. Wang, C. H. Huang, and J. F. Yang, "Depth Maps Interpolation from Existing Pairs of Keyframes and Depth Maps for 3D Video Generation," in *International Symposium on Circuits and Systems*. IEEE, 2010, pp. 3248–3251.

[16] Y. Zhang, J. Zhang, and Q. Dai, "Texture Aided Depth Frame Interpolation," *Signal Processing: Image Communication*, vol. 29, no. 8, pp. 864–874, 2014.

[17] Y. Li, L. Sun, and T. Xue, "Fast Frame-Rate Up-Conversion of Depth Video Via Video Coding," *ACM International Conference on Multimedia*, p. 1317, 2011.

[18] G. Farneback, "Two-Frame Motion Estimation Based on Polynomial Expansion," *Scandinavian Conference on Image Analysis*, vol. 2749, no. 1, pp. 363–370, 2003.

[19] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[20] H. C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981.

[21] D. Nister, "An Efficient Solution to the Five-Point Relative Pose Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004.

[22] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism," in *ACM SIGGRAPH*. ACM Press, 2006, p. 835.

[23] C. Wu, "Towards Linear-Time Incremental Structure from Motion," in *International Conference on 3D Vision*. IEEE, 2013, pp. 127–134.

[24] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *European Conference on Computer Vision*, 2014.

[25] J. L. Schonberger and J.-M. Frahm, "Structure-from-Motion Revisited," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2016, pp. 4104–4113.

[26] S. Bianco, G. Ciocca, and D. Marelli, "Evaluating the Performance of Structure from Motion Pipelines," *Journal of Imaging*, vol. 4, no. 8, p. 98, 2018.

[27] J. Noraky and V. Sze, "Low Power Depth Estimation for Time-of-Flight Imaging," in *International Conference on Image Processing*. IEEE, 2017, pp. 2114–2118.

[28] B. K. P. Horn, *Robot Vision*, ser. MIT Electrical Engineering and Computer Science Series. MIT Press, 1986.

[29] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing," in *NTC81*, 1981.

[30] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[31] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*, 1988, pp. 147–151.

[32] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with," *Communications of the ACM*, vol. 24, pp. 381–395, 1981.

[33] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems," *International Conference on Intelligent Robots and Systems*, pp. 573–580, 2012.

[34] Samsung, "Exynos 5422," www.samsung.com/semiconductor/minisite/exynos/products/mobileprocessor/exynos-5-octa-5422/, accessed: 2018-07-24.

[35] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor Segmentation and Support Inference from RGBD Images," in *European Conference on Computer Vision*, 2012.

[36] A. Schmidt, M. Fularz, M. Kraft, A. Kasiński, and M. Nowicki, "An Indoor RGB-D Dataset for the Evaluation of Robot Navigation Algorithms," in *Lecture Notes in Computer Science*, 2013, vol. 8192 LNCS, pp. 321–329.

[37] O. Wasenmuller, M. Meyer, and D. Stricker, "CoRBS: Comprehensive RGB-D benchmark for SLAM Using Kinect V2," in *Winter Conference on Applications of Computer Vision*. IEEE, 2016, pp. 1–7.

[38] A. Handa, T. Whelan, J. B. McDonald, and A. J. Davison, "A Benchmark for RGB-D Visual Odometry, 3D Reconstruction and SLAM," in *International Conference on Robotics and Automation*. IEEE, 2014.

[39] J. Noraky, "Depth estimation video," www.mit.edu/~jnoraky/lprtof.html, accessed: 2018-10-02.

[40] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[41] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[42] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in *International Conference on Computer Vision*. IEEE, 2011, pp. 2564–2571.

[43] C. S. Bamji, S. Mehta, B. Thompson, T. Elkhatib, S. Wurster, O. Akkaya, A. Payne, J. Godbaz, M. Fenton, V. Rajasekaran, L. Prather, S. Nagaraja, V. Mogallapu, D. Snow, R. McCauley, M. Mukadam, I. Agi, S. McCarthy, Z. Xu, T. Perry, W. Qian, V.-H. Chan, P. Adepu, G. Ali, M. Ahmed, A. Mukherjee, S. Nayak, D. Gampell, S. Acharya, L. Kordus, and P. O'Connor, "IMpixel 65nm BSI 320MHz Demodulated TOF Image Sensor With 3μm Global Shutter Pixels and Analog Binning," in *International Solid-State Circuits Conference*. IEEE, 2018, pp. 94–96.

[44] A. Colaco, A. Kirmani, N.-w. Gong, T. Mcgarry, L. Watkins, and V. K. Goyal, "3dim : Compact and Low Power Time-of-Flight Sensor for 3D Capture Using Parametric Signal Processing," *International Image Sensor Workshop*, pp. 349–352, 2013.

[45] "Pico Zense DCAM710," picozense.picovr.com/, accessed: 2018-07-24.