LOW POWER DEPTH ESTIMATION FOR TIME-OF-FLIGHT IMAGING

James Noraky, Vivienne Sze

Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
{jnoraky, sze}@mit.edu

ABSTRACT

Depth sensing is used in a variety of applications that range from augmented reality to robotics. One way to measure depth is with a time-of-flight (TOF) camera, which obtains depth by emitting light and measuring its round trip time. However, for many battery powered devices, the illumination source of the TOF camera requires a significant amount of power and further limits its battery life. To minimize the power required for depth sensing, we present an algorithm that exploits the apparent motion across images collected alongside the TOF camera to obtain a new depth map without illuminating the scene. Our technique is best suited for estimating the depth of rigid objects and obtains low latency, 640×480 depth maps at 30 frames per second on a low power embedded platform by using block matching at a sparse set of points and least squares minimization. We evaluated our technique on an RGB-D dataset where it produced depth maps with a mean relative error of 0.85% while reducing the total power required for depth sensing by $3\times$.

***Index Terms*—** time-of-flight camera, low-power, depth estimation, RGB-D, depth map

1. INTRODUCTION

Depth sensing is used in a variety of applications. For applications that are sensitive to latency, like augmented reality or robotics, a time-of-flight (TOF) camera is appealing because it obtains depth with minimal computations by emitting and measuring the round trip time of light [1]. However, many of these applications also run on battery powered devices, and one drawback of a TOF camera is its illumination source, which further limits battery life. To minimize the power consumption of TOF cameras, we present an algorithm that produces low latency depth maps by using images, which are routinely collected alongside the TOF camera, to estimate a new depth map without illuminating the scene. One application that benefits from our approach is robotic navigation, where a robot moves in a static environment (Figure 1) and uses the estimated depth maps for tasks that include simultaneous localization and mapping (SLAM) [2] or obstacle avoidance.

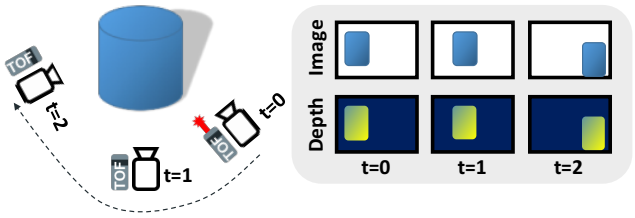

Fig. 1: **Causal Depth Estimation** The TOF camera is used to obtain the first depth map, and all subsequent depth maps are estimated using the concurrently captured images. This minimizes the usage of the TOF camera's illumination source and the total power required for depth sensing.

The idea of using image data to estimate depth has been previously used to increase the frame rate of depth video. In these scenarios, images are captured at higher frame rates than depth maps, and different approaches estimate depth maps for images without any corresponding depth. The authors of [3, 4, 5, 6] use the block-wise correspondences between the images without depth and those with it (from both the preceding and following frames) to identify the depth values to average. This process is repeated for non-overlapping blocks, resulting in a low estimation frame rate. In contrast, our technique is causal and uses only a previously measured depth map along with the image data to obtain a new depth map. We also assume that objects in the scene are rigid, and we exploit this assumption to produce depth maps in real time, or 30 frames per second (FPS), on a low power embedded platform, which has limited compute resources.

Our contribution is an algorithm for depth estimation that

1. Reduces the total power required for depth sensing in TOF cameras
2. Is causal and has low latency
3. Produces dense depth maps in real time (30 FPS)

2. DEPTH MAP ESTIMATION

Our approach takes as inputs a pair of consecutive images and a previous depth map that is synchronized with the first image. The algorithm then outputs a new depth map that corresponds to the second image. While our approach can be

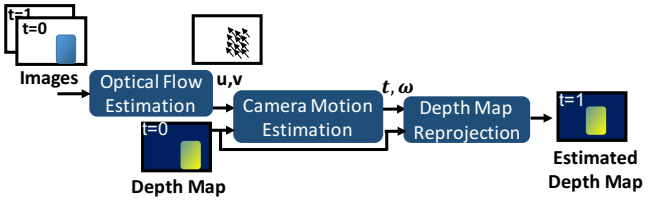

Fig. 2: **Depth Estimation Pipeline** Our algorithm estimates the camera motion using optical flow and uses the camera motion estimate to obtain a new depth map.

used to estimate the depth for any rigid object, we focus on the scenario shown in Figure 1, where the difference between depth maps is caused by the camera motion. Figure 2 depicts the pipeline of our technique. Our approach for depth map estimation is inspired by [7], which estimates camera motion up to a scale factor. However, our work is different because it uses the previously measured depth map to estimate the actual camera motion and then obtains a new depth map.

2.1. Optical Flow Estimation

Optical flow is the apparent motion of the pixels across images. There are several approaches to estimate optical flow. However, many of these algorithms are computationally expensive [8] and are unsuitable for low power embedded platforms, which have limited compute resources.

With complexity in mind, we used block matching to estimate optical flow. Block matching algorithms estimate the optical flow for a pixel by taking a block centered on that pixel and searching the consecutive image to find the best matching block according to some criterion. In our implementation, we performed block matching using 16×16 blocks and searched a 48×48 region centered on each block in the consecutive image to find the matching block that maximized the normalized correlation score. Our algorithm only requires optical flow at a sparse set of points, and consequently, we use block matching for pixels separated by a stride of 80 pixels. For 640×480 images, this means that our algorithm uses at most 48 optical flow estimates to obtain a dense depth map, which enables our implementation to run in real time.

2.2. Camera Motion Estimation

To estimate the camera motion, we invert a model that relates how 3D objects in the scene are captured in a 2D image. We assume that objects in the scene are captured onto the pixels of an image under perspective projection. We orient the camera's coordinate system such that its origin is at the center of projection, its Z-axis is parallel to the optical axis, and the X and Y axes are parallel to that of the image plane. This assumption makes the Z-coordinate of each point equal to its distance from the camera, which is approximately measured by the TOF camera.

Under these assumptions, a point in the scene located at (X_i, Y_i, Z_i) is projected to the i^{th} pixel located at (x_i, y_i) in the image such that

$$\frac{x_i}{f} = \frac{X_i}{Z_i} \qquad \frac{y_i}{f} = \frac{Y_i}{Z_i} \tag{1}$$

where f is the focal length. As the camera moves, the objects are projected onto different pixel locations in a new image that depend on both the camera motion and the objects' distances from the camera. This can be seen by differentiating Eq. (1) with respect to time and rearranging the terms to obtain the following relationship

$$u_i = \dot{x}_i = \frac{f\dot{X}_i - x_i\dot{Z}_i}{Z_i} \qquad v_i = \dot{y}_i = \frac{f\dot{Y}_i - y_i\dot{Z}_i}{Z_i} \tag{2}$$

which relates the optical flow of the i^{th} pixel, which is denoted by u_i and v_i, to the motion of its corresponding 3D point, which is represented by $\dot{X}_i, \dot{Y}_i$, and $\dot{Z}_i$.

Given the inputs to our algorithm, we can compute X_i, Y_i, and Z_i for each pixel in the first image. Using $\dot{X}_i, \dot{Y}_i$, and $\dot{Z}_i$, we can then compute the 3D coordinates of the points that correspond to the pixels in the second image and obtain the new depth map. However, without any further assumptions, the task of estimating $\dot{X}_i, \dot{Y}_i$, and $\dot{Z}_i$ for each pixel is underdetermined. Here, we use the fact that only the camera is moving and treat the scene as a rigid body. Each 3D point then moves relative to the camera with a velocity given by

$$\begin{bmatrix}\dot{X}_i \\ \dot{Y}_i \\ \dot{Z}_i\end{bmatrix} = -\underbrace{\begin{bmatrix}U \\ V \\ W\end{bmatrix}}_{\boldsymbol{t}} - \underbrace{\begin{bmatrix}A \\ B \\ C\end{bmatrix}}_{\boldsymbol{\omega}} \times \begin{bmatrix}X_i \\ Y_i \\ Z_i\end{bmatrix} \tag{3}$$

where $\boldsymbol{t}$ and $\boldsymbol{\omega}$ are the translational and angular velocities of the camera in 3D space, respectively. Combining Eq. (2) and Eq. (3), we have the following constraints between the optical flow and camera motion for each pixel in the image

$$u_i = \underbrace{-\frac{fU}{Z_i} + \frac{x_iW}{Z_i} + A\left(\frac{x_iy_i}{f}\right) - B\left(f + \frac{x_i^2}{f}\right) + Cy_i}_{\hat{u}_i(\boldsymbol{t},\boldsymbol{\omega})} \tag{4}$$

$$v_i = \underbrace{-\frac{fV}{Z_i} + \frac{y_iW}{Z_i} + A\left(f + \frac{y_i^2}{f}\right) - B\left(\frac{x_iy_i}{f}\right) - Cx_i}_{\hat{v}_i(\boldsymbol{t},\boldsymbol{\omega})} \tag{5}$$

To solve for $\boldsymbol{t}$ and $\boldsymbol{\omega}$, we use least-squares minimization to minimize the objective function

$$J(\boldsymbol{t},\boldsymbol{\omega}) = \sum_{i=1}^{N} \left(u_i - \hat{u}_i(\boldsymbol{t},\boldsymbol{\omega})\right)^2 + \left(v_i - \hat{v}_i(\boldsymbol{t},\boldsymbol{\omega})\right)^2 \tag{6}$$

where N is the number of pixels where the optical flow is known.

2.2.1. Outlier Rejection

Given the limited compute resources on low power embedded platforms, our algorithm uses block matching to estimate optical flow. These optical flow vectors are susceptible to errors when there is obstruction between images, when the pixels in a block have different displacements, and when there is ambiguity as to the best match. As such, there is a need to reject these outliers before we estimate $\boldsymbol{t}$ and $\boldsymbol{\omega}$. To do so, we first eliminate the estimates that are obtained from blocks with low normalized correlation scores. We empirically found that 0.5 was a sufficient threshold. We then used RANSAC [9] to robustly estimate $\boldsymbol{t}$ and $\boldsymbol{\omega}$. In our implementation, we found that 100 RANSAC iterations was sufficient to obtain accurate depth maps.

2.3. Depth Map Reprojection

After $\boldsymbol{t}$ and $\boldsymbol{\omega}$ are estimated, we reproject the previous depth map to obtain a new one. To do this, we first obtain the 3D coordinates for each point in the previous depth map using Eq. (1). Then, we find the new coordinate of each point by applying its displacement, which is computed using Eq. (3). Finally, to obtain the new depth map, we reproject the Z-component of each point to a new pixel location obtained using Eq. (1). For points that reproject onto the same pixel position, we choose the smallest value of Z. This approach may introduce holes into the estimated depth map, and approaches like median filtering can be used to fill them in. In our implementation, we skipped this post-processing step because these holes were negligible and the infilling algorithms substantially reduced our estimation frame rate.

3. ALGORITHM EVALUATION

3.1. Test Setup

We evaluated our approach on sequences from the TUM RGB-D dataset [10], which contains images and depth maps captured for various camera trajectories in static scenes. Because the images and depth maps are not synchronized, we associated the images with depth maps by finding the closest match in terms of their timestamps. We used the default intrinsic parameters to obtain the focal length and principal point. To evaluate our approach, we estimated the depth maps sequentially for the first five frames of each sequence. We used the first measured depth map to predict the second depth map. For all subsequent frames, we used the estimated depth maps instead of what was measured.

We quantified the performance of our algorithm by computing the mean relative error (MRE) between our estimated depth map and what was measured in the dataset for overlapping pixels. The MRE weighs depth errors at distances close to the TOF camera more than the same depth errors at further distances and is a common metric used to evaluate depth estimates [11, 12]. We present the MRE in percentage form

$$\text{MRE} = 100 \cdot \frac{1}{N} \sum_{i=1}^{N} \frac{|\hat{Z}_i - Z_i|}{Z_i} \tag{7}$$

where $\hat{Z}_i$ and Z_i are the estimated and ground truth depth for the i^{th} pixel, respectively.

	Depth Frame			
Sequence	2	3	4	5
freiburg1_360	0.51	0.50	0.72	0.81
freiburg1_desk	0.98	1.53	5.61	11.80
freiburg1_floor	0.33	0.50	0.51	0.36
freiburg1_room	0.62	0.84	0.86	1.09
freiburg1_xyz	1.01	1.69	1.22	1.09
Mean	0.69	1.01	1.78	3.03
Median	0.62	0.84	0.86	1.09

Table 1: **Mean MRE** Each cell is obtained by averaging the results of 100 experiments. The shaded cells are instances of when our algorithm fails. However, these cases can be detected by our approach and used to trigger the TOF camera to obtain a new depth map.

3.2. Results

Table 1 summarizes the MRE for each sequence. Since RANSAC is nondeterministic, we averaged the MRE from 100 experiments for each cell. An example of an estimated depth map is shown in Figure 3. In this example, we see the holes, which cluster as curved lines, in the estimated depth map. These areas are insignificant compared to the regions (e.g. monitor screen) where depth was not measured in this dataset and are consequently unavailable in the reprojected depth map.

The shaded cells in Table 1 are examples of when the depth map estimation algorithm fails. For these frames, the search region of the block matching algorithm was too small to capture the underlying camera motion. Because we sequentially estimated the depth maps, this error then propagates to the fifth frame. A natural solution to this problem is to increase the search region size, but this increases the computation for block matching and decreases the estimation frame rate. Fortunately, this scenario can be detected by using the number of inliers identified by RANSAC to trigger the illumination source to obtain a reliable depth map. As a result of this sequence, we provide both the mean and median MRE across all sequences. Overall, the median MRE across all sequences and frames is 0.85%.

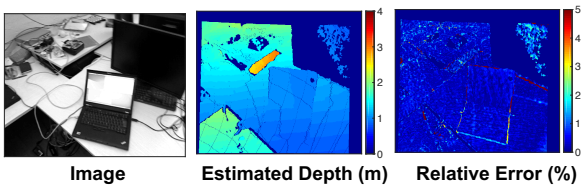

Fig. 3: ***freiburg1_room*** The estimated depth map is shown for the second frame of this sequence.

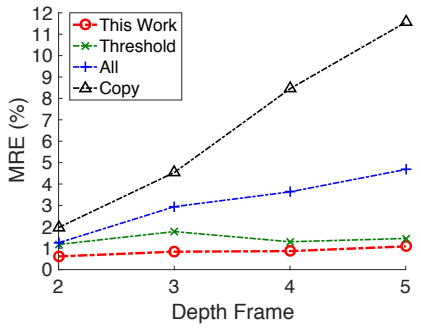

Fig. 4: **Method Comparison** Median MRE across sequences for different approaches described in Section 3.3.

3.3. Discussion

3.3.1. RANSAC Decreases the MRE of Estimated Depth Map

To obtain accurate depth maps, our technique rejects spurious optical flow estimates obtained by block matching using thresholds and RANSAC. To demonstrate that this combination reduces the MRE, we compare it to the cases where all of the optical flow estimates are used (**All**) and where only a threshold is applied to select optical flow estimates from blocks that have high normalized correlation scores (**Threshold**). As one would expect, both our technique and **Threshold** outperforms **All**. Our technique outperforms **Threshold** because a high normalized correlation score does not guarantee that the optical flow estimate is correct. For example, optical flow is difficult to estimate in regions with uniform brightness despite the normalized correlations scores being high. In our approach, RANSAC eliminates these outliers.

3.3.2. Algorithm Compensates For Camera Motion

To compare how our technique compensates for camera motion, we compare it to the case where the first depth map is copied and used to predict all subsequent frames (**Copy**). **Copy** has low latency and can be effective when the camera motion is slow and the difference between consecutive image frames is minimal. Figure 4 shows that this is not the case for these sequences and shows that our algorithm compensates for nontrivial camera motion as the gap between our technique and **Copy** grows with time.

4. POWER CONSUMPTION IN PLATFORM

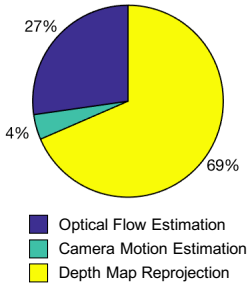

Fig. 5: **Computation Time Breakdown**

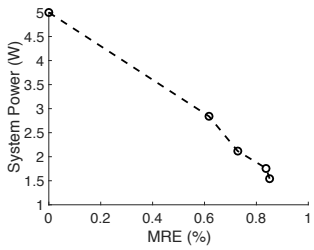

Fig. 6: **MRE vs Estimated System Power**

We implemented our algorithm on the ODROID-XU3 board, which has an Exynos 5422 processor [13]. Our implementation uses the Cortex-A7 cores and outputs 640×480 depth maps at 30 FPS. As shown in Figure 5, **Camera Motion Estimation** is efficient and, even with 100 RANSAC iterations, constitutes only 4% of the computation time. Furthermore, if only odometry is required, our algorithm runs at 96 FPS. Our implementation consumes a total of 678 mW, of which 226 mW is the idle power. In contrast, the illumination source of TOF cameras consumes 2-5 W [14, 15].

Using this measurement, we estimate the power of a hybrid TOF camera system, where we reduce the frequency of the TOF camera usage and obtain depth using our technique. Because digital cameras are often found on depth sensing devices and images are routinely captured for other purposes, we do not factor in their energy costs. As previously shown, reducing the frequency of the TOF camera usage also reduces the accuracy of the estimated depth map. We show this tradeoff in Figure 6 where we plot the median MRE against the estimated system power, where each point represents different frequencies of the TOF camera usage. Here, we see that we can reduce the total power required for depth sensing by $3\times$ while maintaining a median MRE of 0.85%.

5. CONCLUSION

We present an algorithm that reduces the power of TOF imaging by using images to estimate a new depth map without illuminating the scene. Our algorithm is computationally efficient and causal, enabling dense depth map estimation in real time on a low power embedded platform. When evaluated on an RGB-D dataset, our algorithm produced depth maps with a median MRE of 0.85% while reducing the total power required for depth sensing by $3\times$.

6. ACKNOWLEDGEMENTS

We thank Analog Devices for funding this work and the research scientists in the company for helpful discussions.

7. REFERENCES

[1] Stan Zdonik, Peng Ning, Shashi Shekhar, Jonathan Katz, and Xindong Wu, *Time-of-Flight Cameras Principles, Methods and Applications*, 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] Jinwook Choi, Dongbo Min, Bumsub Ham, and Kwanghoon Sohn, "Spatial and temporal up-conversion technique for depth video," *IEEE International Conference on Image Processing*, pp. 3525–3528, 2009.

[4] Hung Ming Wang, Chun Hao Huang, and Jar Ferr Yang, "Depth maps interpolation from existing pairs of keyframes and depth maps for 3D video generation," *IEEE International Symposium on Circuits and Systems*, pp. 3248–3251, 2010.

[5] Yanjie Li, Lifeng Sun, and Tianfan Xue, "Fast frame-rate up-conversion of depth video via video coding," *ACM Multimedia*, p. 1317, 2011.

[6] Yongbing Zhang, Jian Zhang, and Qionghai Dai, "Texture aided depth frame interpolation," *Signal Processing: Image Communication*, vol. 29, no. 8, pp. 864–874, 2014.

[7] Anna R. Bruss and Berthold K. P. Horn, "Passive navigation," *Computer Vision, Graphics, and Image Processing*, vol. 21, no. 1, pp. 3–20, 1983.

[8] Denis Fortun, Patrick Bouthemy, and Charles Kervrann, "Optical flow modeling and computation: A survey," *Computer Vision and Image Understanding*, vol. 134, pp. 1–21, 2015.

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

[10] Jurgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," *IEEE International Conference on Intelligent Robots and Systems*, pp. 573–580, 2012.

[11] Rene Ranftl, Vibhav Vineet, Qifeng Chen, and Vladlen Koltun, "Dense Monocular Depth Estimation in Complex Dynamic Scenes," *Conference on Computer Vision and Pattern Recognition*, pp. 4058–4066, 2016.

[12] Anirban Roy, "Monocular Depth Estimation Using Neural Regression Forest," *Conference on Computer Vision and Pattern Recognition*, pp. 5506–5514, 2016.

[13] ODROID, "ODROID-XU3," `www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127`.

[14] SoftKinetic, "DepthSense Camera — DS525 Datasheet," `www.softkinetic.com/Products/DepthSenseCameras`.

[15] Andrea Colaco, Ahmed Kirmani, Nan-wei Gong, Tim Mcgarry, Laurence Watkins, and Vivek 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.