Multi-Camera DeepTAM

Rohit Suri
ETH Zürich
suriro@ethz.ch

Mayank Mittal
ETH Zürich
mittalma@ethz.ch

Fadhil Ginting
ETH Zürich
gintigfa@ethz.ch

Parker Ewen
ETH Zürich
ewenpa@ethz.ch

Abstract

Visual Odometry methods based on classical 3D geometry have been around for years, using either indirect feature matching or direct visual error minimization. Lately, learning-based methods that combine both matching and geometry estimation in a single network have achieved impressive results. One such method is DeepTAM [23].

Further, it has been shown that classical methods benefit from the extended field of view provided by using multiple cameras. However, these setups have been ignored by current learning-based methods. In this work, we extend the existing DeepTAM pipeline to leverage a multi-camera setup with known geometry. We demonstrate the generalizability of DeepTAM to other monocular setups and highlight the scenarios in which it performs poorly. We show the efficacy of our proposed multi-camera VO pipeline to receive better pose estimates using experiments based on simulation.

1. Introduction

In robotics, accurate pose estimation is an essential prerequisite for a variety of tasks such as obstacle detection, mapping, and motion planning. Although GPS provides a straight-forward solution for outdoor environments, it is highly unreliable in indoor scenarios. Alternatively, the pose of the robot can be estimated incrementally relative to the changes in the robot's surrounding. This is referred to as odometry. Traditional systems used sensors such as rotary encoders to evaluate the position by measuring the angle rotated by the wheels shaft. However, this method is erroneous as wheels tend to slip, and is futile for non-wheeled locomotion systems, such as legged robots, and aerial vehicles. Hence, there has been an increased interest to perform odometry using visual systems [1] such as cameras. These sensors are cheaper and provide a lot more information about the environment compared to inertial sensors such as accelerometers and gyroscopes.

Classical approaches for Visual Odometry (VO) can be broadly categorized into feature-based methods and direct methods. Feature-based methods (such as [12, 8]) typically involve feature extraction and tracking, followed by minimization of a reprojection error between detected landmarks. Although these approaches work well even when there are large frame-to-frame motions, their performances are adversely affected by outliers during feature matching. On the other hand, direct methods (such as [4, 14]) minimize the photometric error between the images from the sensor and thus exploit all the information available. However, they are known to work well only when the frame-to-frame motion is small. Recent approaches (such as [5, 3]) have combined the two classical methods to perform more efficient pose estimation. With the advent of deep learning, learning-based methods (such as [23, 22]) have been also gained attention to perform visual odometry. These algorithms often combine both matching and geometry estimation into a single network and have shown performance comparable to the classical approaches. However, a data-driven approach may overfit on the training data and may generalize poorly to other sensor setups.

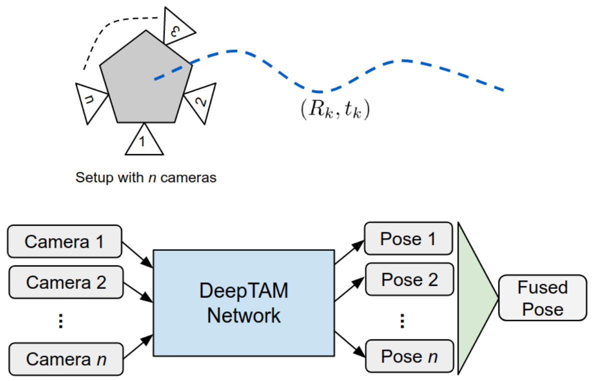
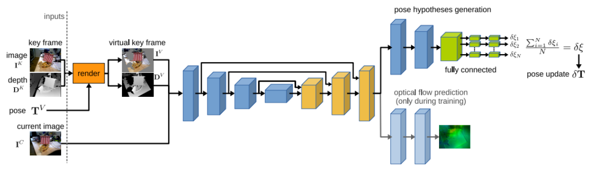
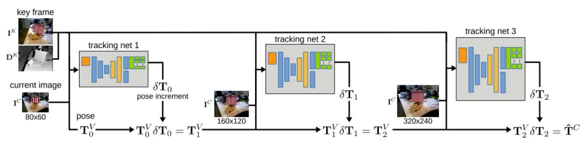


Figure 1: A multi-camera Visual Odometry (VO) pipeline. DeepTAM, a learning-based VO approach, is used for estimating pose of each camera. These poses are then fused to acquire a more accurate estimate.

For most practical purposes, relying on a single camera for VO may result in poor motion estimate due to translational and rotational ambiguities [13]. These ambiguities can be alleviated by using a multi-camera system. In



(a) Tracking model



(b) Coarse-to-fine pose refinement model

Figure 2: DeepTAM architecture for pose estimation. (Images taken from [23])

the past, several multi-camera VO-pipelines have been proposed in conjunction to classical approaches. These systems leverage the extended field-of-view from the multiple cameras to get more robust pose estimates. However, to our knowledge, no such work exists which extends a learning-based approach for VO using a multi-camera system. In this work, we propose a pipeline built on-top of DeepTAM [23] to do so. We make the following contributions:

- We evaluate the generalizability of DeepTAM on other monocular setups, and highlight the scenarios where it performs poorly.
- We show how a multi-camera system with known geometry can be used to estimate better poses using DeepTAM.

The remainder of the paper will discuss the related work on multi-camera VO systems in Sec. 2, followed by a brief background over DeepTAM in Sec. 3. Our technical approach is presented in Sec. 4. We describe our dataset collection and results in Sec. 5. In Sec. 6, we discuss our observations on learning-based VO approaches and future extension of our work. The work distribution and code contribution are mentioned in Sec. 7 and Sec. 8 respectively.

2. Related Work

In the last decade, several visual odometry pipelines have been proposed for monocular and multiple camera systems. In this section, we focus on literature related to our work on multi-camera pose estimation.

Many multi-camera pose estimation frameworks consider a networks of cameras as a single generalized camera. Pless *et al.* [15] propose an approach in which each pixel from an image represents a sampling of a region of space in a scene instead of light rays which interact with the sensor from a particular location. This is done by describing the images in terms of light rays and origin points which facilitates decentralization of the camera model. The benefit of this method is that the generalized model can be computed even if the centers of projection for each individual camera ate at different points. On the other hand, Composeco *et al.* [2] jointly estimate all camera poses using generalized perspective n-point (gPnP). This uses the generalized camera model in place of the individual cameras at a known configuration. The generalized camera model helps in computing minimal solvers which approximate the pose of each camera in the model with the help of RANSAC.

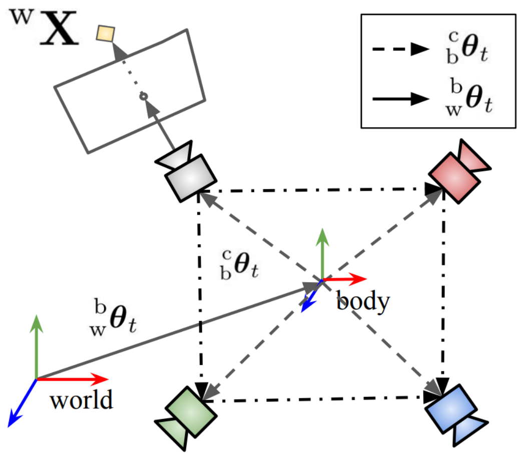
Likewise, the method presented in [9] finds minimal solvers as well. However, it treats each camera individually by using classical camera models instead of the generalized camera model. Pose estimation includes two steps; first, depth information of points is calculated and then rigid transformations are computed. As shown [20], it is also possible to perform a large-scale structure-from-motion (SfM) by using a distributed, multi-camera model. Both these approaches offer more efficient varieties of widely used SfM frameworks. However, they only focus on individual cameras instead of treating them as a combined system.

Visual localization for autonomous driving involves multiple cameras and thus requires fusing the pose estimates between the mounted cameras. By using nodes to represent the estimated poses for each camera and inter-node edge constraints, along with the constraints for the 2D-plane motion, the resulting factor graph can be optimized to obtain better pose estimates for the vehicle [6]. Other optimization methodologies that account for pose drift and aid in pose fusion without graphs are also possible (such as [10]).

3. Background

DeepTAM [23] is a deep learning method for keyframe-based dense tracking and mapping. The algorithm is motivated from an equivalent classical approach DTAM [14]. The tracking architecture performs incremental frame to keyframe tracking. This helps in reducing the dataset bias problem and helps the algorithm to generalize to other setups. The mapping architecture, on the other hand, combines depth measurements with image-based priors and yields accurate depth maps. Both the tracking and mapping portions of DeepTAM are implemented using distinct networks and can be trained separately. For our work, we only focus on the tracking part.

DeepTAM employs several schemes to infer accurate poses from the network. The pose estimation step can be thought of as a two-view relative pose estimation. Each image is compared to the last keyframe in order to determine the pose difference. As the new image goes further away



${}^{\mathrm{w}}\mathbf{X}$
${}^{\mathrm{c}}_{\mathrm{b}}\boldsymbol{\theta}_t$
${}^{\mathrm{b}}_{\mathrm{w}}\boldsymbol{\theta}_t$
${}^{\mathrm{c}}_{\mathrm{b}}\boldsymbol{\theta}_t$
${}^{\mathrm{b}}_{\mathrm{w}}\boldsymbol{\theta}_t$
body
world

Figure 3: The world, rig (base), and camera coordinate systems along with rigid transformations θ_t between them. (Image taken from [17])

from the keyframe, the relative pose estimation becomes less accurate. Thus a new keyframe is set when a certain translation or rotation threshold is crossed.

The encoder-decoder network for tracking takes the last keyframe RGB image and depthmap, along with the current RGB image as inputs. To improve the accuracy, the network generates multiple pose hypothesis and provides the final predicted 6-DOF pose as an average of these hypothesis, as shown in Fig. 2(a). During training, they add optical flow prediction as an auxillary task for the network in order to simulate the training of motion features.

In order to deal with large camera motions, they refine the pose incrementally by using three tracking networks with distinct parameters but similar architectures, as shown in Fig. 2(b).

Ideally, DeepTAM only requires the RGB images as inputs since the mapping part takes care of generating the depthmap for each keyframe. However, in order to mitigate the need of the mapping part, we provide the RGB image as well as the corresponding groundtruth depth image as inputs to the tracking network.

4. Technical Approach

Consider a setup with N-cameras in a known configuration. The translation component of the rigid body transformation θ_t at instant t is parameterized using cartesian coordinates $\mathbf{t}$, while the rotation component is represented in angle-axis coordinates $\mathbf{r}$. As shown in Fig. 3, we use the following coordinate systems: world (w), base frame of sensor rig (b), and the camera frame c_i corresponding to the i^{th} camera in the rig. The rigid-body transformation from the base to camera i is denoted by ${}^{c}_{b}\theta_i$ and is known before-hand.

DeepTAM predicts the pose for each camera in the world frame as denoted ${}^{c}_{w}\hat{\theta}_{t,i}$. We transform all these estimated poses to the base frame before performing the pose fusion, i.e. for each camera $i \in 1, ..., N$:

$$ {}^{b}_{w}\hat{\theta}_{t,i} = {}^{c}_{w}\hat{\theta}_{t,i} \otimes {}^{b}_{c}\theta_i \tag{1}$$

where $\otimes$ denotes the composition of rigid transformation. An outline of our pipeline is shown in Fig. 1.

4.1. Baseline approach

As a baseline approach, we perform simple pose averaging where the pose estimates from each camera are weighed equally [21]. Parallels can be drawn between this method and the single camera DeepTAM step where multiple pose hypotheses are generated. The fused pose estimate ${}^{b}_{w}\hat{\theta}_{\mathbf{t}}$ for the base in the world frame can be expressed as:

$$ {}^{b}_{w}\hat{\mathbf{t}}_t = \frac{1}{N}\sum_{i=1}^{N} {}^{b}_{w}\hat{\mathbf{t}}_{\mathbf{t},\mathbf{i}}, \tag{2}$$

$$ {}^{b}_{w}\hat{\mathbf{r}}_t = \frac{1}{N}\sum_{i=1}^{N} {}^{b}_{w}\hat{\mathbf{r}}_{\mathbf{t},\mathbf{i}}. \tag{3}$$

Prediction accuracy can be distilled into several factors which depend upon the initial predictions. Firstly, if all the pose estimates were centered about a point which was not the ground truth pose, the average would not be the closer to the actual pose (and may in fact be further from the ground truth than one of the camera predictions). Thus, to see improvements in accuracy we need the probability distribution of the estimates to be centered about the ground truth pose.

It may be the case that some cameras have rich scene information while others on the rig see homogeneous regions like a wall. The pose estimate for the camera which sees many features will be more accurate than the one which sees the homogeneous region. Weighing both these estimates equally would not non-optimal.

Ideally we would have information on which pose estimate is closest to the ground truth, but since this information is not known, and is indeed the problem trying to be solved, we must try to find a way of using information given to us to try and discern which cameras are likely to give the best estimates.

4.2. Weighted Averaging using scene richness

Equal weighting may not produce optimal results for the reasons specified above. However, by using the information provided by the cameras we present a weighting scheme for pose fusion in this section. The aspects we consider are:

- richness of information in RGB image
- homogeneity in the depthmaps

4.2.1 Features density for scene richness

During our experiments, we noticed that DeepTAM performed better in scenes with lots of clutter. To capture this information richness, we apply weights to the pose estimates based on the relative number of SIFT features [11] detected in each camera's image. Let f_i denote the number of SIFT features detected in the image from camera i, then the weight provided to current pose estimate for the camera is given by: $w_i^s = \frac{f_i}{\sum_{i=1}^{N} f_i}$

4.2.2 Homogeneity of Depthmaps

In a similar fashion, when homogeneous depth images are encountered (for instance: a camera facing a wall) lesser information is present in the scene and may yield poor tracking results. In order to quantify this, we calculate the standard deviation of the depth information present in the depthmap. Let σ_i denote the standard deviation for the depth map from camera i, then the weight given to the corresponding pose is $w_i^d = \frac{\sigma_i}{\sum_{i=1}^{N} \sigma_i}$.

However, the homogeneous depth images may not solely capture the complete information of the scene (for instance: if a wall contains patterns, then tracking may still work since RGB information is rich). Hence, we take a weighted sum of the weights calculated above. If c_s and c_d denote the weights given to the weights from RGB and depth information respectively, such that $c_s + c_d = 1$, then the fused pose ${}^{b}_{w}\hat{\theta}_{\mathbf{t}}$ is:

$$ {}^{b}_{w}\hat{\mathbf{t}}_t = \frac{1}{N} \sum_{i=1}^{N} (c_s w_i^s + c_d w_i^d) {}^{b}_{w}\hat{\mathbf{t}}_{\mathbf{t,i}}, \tag{4} $$

$$ {}^{b}_{w}\hat{\mathbf{r}}_t = \frac{1}{N} \sum_{i=1}^{N} (c_s w_i^s + c_d w_i^d) {}^{b}_{w}\hat{\mathbf{r}}_{\mathbf{t,i}}. \tag{5} $$

4.3. Averaging using outlier rejection

In this method, we select the estimates of the base frame which agree the most with each other. In order to reject what are considered outlier poses, the average and standard deviations for all the poses are first computed. The acceptability of each pose is evaluated based on the sigma-based rule, i.e. the poses that outside a certain factor of the standard deviation (typically 1.4) from the mean pose discarded. The inlier poses are then fused using the averaging approach used in the baseline.

5. Results

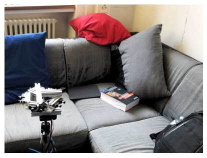
In this section, we first describe the various datasets collected by us. We then discuss the generalizability of DeepTAM in various monocular setups and compare the accuracy of various multi-camera pose fusion methods that were presented in Sec. 4.

(a) SunCG

(b) AirSim

Figure 4: Sythetic Datasets

(a) ZED Mini


(b) CVG Indoor

(c) CVG Outdoor

Figure 5: Real-World Datasets

5.1. Dataset Collection

For a thorough evaluation, we collect datasets from real-world using various hardware setups as well as from high-fidelity simulation environments. Sample images from the our collected simulation and real-world datasets are shown in Fig. 4 and Fig. 5 respectively.

5.1.1 Synthetic Datasets

A variety of simulators have emerged in the last few years which provide realistic RGB images. To this end, we resort to the following well-known simulators for indoor and outdoor environments simulations:

SunCG Dataset: Te SUNCG dataset [19] provides a variety of 3D models of furnished houses. The MINOS simulation framework [16] allows importing of these models and setting up multiple cameras. Using the simulator we setup a planar camera rig comprising of three cameras oriented at -26.5^o, 0^o, and 26.5^o with respect to the base frame.

AirSim Dataset: AirSim [18] is a drone-simulator built on top of Unreal Engine. Using a sub-urban environment in the simulator, we capture poses of the drone and images from a stereo camera mounted on the drone during its flight.

5.1.2 Real-World Datasets

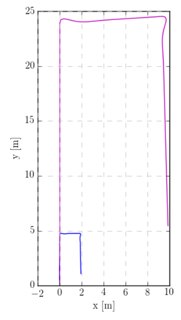
Due to resource limitations, we could not directly acquire accurate groundtruth for a variety of real-world data that we collected. Instead we relied on using recognizable patterns such as checkerboard and fudicial markers for calculating the poses of the cameras.

ZED Mini Dataset: ZED Mini from StereoLabs provides synchronized RGB images and dense depthmaps. We collect dataset from this stereo camera in an indoor environment with a scene structure similar to the TUM RGBD datasets. We use a checkerboard placed on the scene to extract groundtruth pose of the camera.

CVG Stereo Cameras Rig Datasets: For more realistic multi-camera system, we used the camera rig provided by the CVG lab. The rig has five stereo cameras in a pentagonal configuration. Each stereo pair comprises of an RGB camera and high-dynamic-range (HDR) grayscale camera. The rig uses an FPGA to provide synced images from the ten cameras along with the disparity image computed for each stereo pair. However, from our experiments, we found that these disparity images were too noisy to get reasonable results from DeepTAM. Instead we generated our own disparity images. We first perform adaptive equalization of the images received from each stereo pair to deal with the HDR of the grayscale images. We then compute the dispairty by stereo semi-global block-matching [7] and denoise it using a median filter. The scenes we captured using the camera rig includes indoor as well as outdoor scenes such as cluttered desks, inside student offices, and outdoor courtyards. For the dataset used in the results shown in this paper, we get the groundtruth poses using a Vicon system.

5.2. Generalizability of DeepTAM

Learning-based approaches are susceptible to overfitting on the training data. In order to test the generalizability of DeepTAM, we perform a qualitative comparison of the trajectories obtained using the DeepTAM tracking architecture and the groundtruth poses.


(a) Results on AirSim

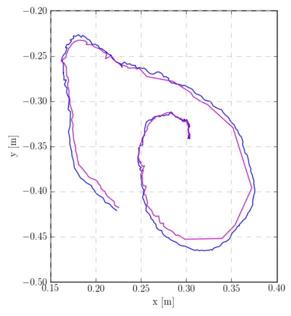

(b) Results on ZED Mini

Figure 6: Qualitative comparisons of the trajectories from DeepTAM (in blue) and the groundtruth (in purple)

AirSim Dataset: In the AirSim dataset, we find that DeepTAM performs poorly. As seen in Fig. 6(a), DeepTAM predicts the trajectory shape correctly. However, it suffers severely from a scaling issue. After investigating the DeepTAM source code further, we arrived at the conclusion that DeepTAM does not perform well in outdoor environments. The reason of this limitation might be because DeepTAM is trained using indoor datasets (SUN3D and SUNCG) which have a lower values of depth images (upto 5 meters). On the other hand, in the AirSim simulation the depth values can go up to 20 meters. Further if we perform depth clipping, the DeepTAM works worse due to missing depth values in the input depthmaps.

ZED Mini Dataset: Experiment with ZED Mini in the indoor dataset shows that DeepTAM manages to generalize well, as can be seen from Fig. 6(b). However, it needs to be noted that the depthmaps provided to the tracker in this case were dense and the scene is indoor.

CVG Stereo Cameras Rig Datasets: During our initial experiments with the CVG camera rig, we found that the tracking prediction of DeepTAM was poor due to the noisy disparity maps that were generated using the FPGA. The scaling of the estimated poses were poor with respect to the groundtruth. Further when using other cameras in the rig, the retrieved poses were worse. However, when we switched to using the depthmaps we genrate, the results were better, as shown in Fig. 7. However, compared to the depthmaps from ZED Mini, our computed depthmap was not at par since it had several missing values and were not accurate. On the basis of this finding along with the results from AirSim, we deduce that rich and accurate depthmaps lie at the core of the tracking part in DeepTAM.

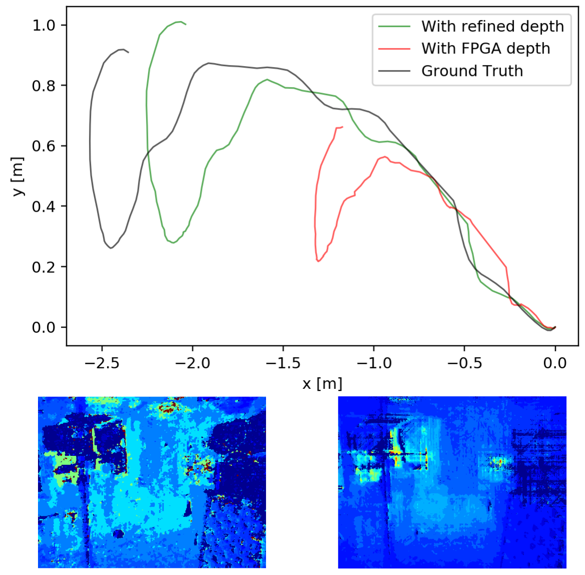
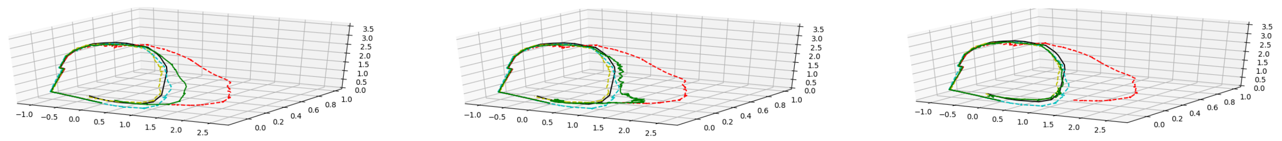


Figure 7: Qualitative comparisons of the trajectories from DeepTAM on the CVG camera rig when using the depthmaps from the FPGA (bottom left) and the refined depthmaps generated by us (bottom right).



(a) Results using simple averaging (as baseline)

(b) Results using scene-richness based averaging

(c) Results using outlier-rejection based averaging

Figure 8: Qualitative comparisons of the trajectories from DeepTAM using the SUNCG dataset with three cameras in the simulated sensor rig. The groundtruth is plotted in black while the fused pose in shown in green. The dotted lines (in red, cyan and yellow) are the estimated pose of the base from individual camera (1, 2, and 3 respectively) pose estimates.

Name	**ATE$_t$ RMSE (in m)**
Camera 1	0.5629
Camera 2	0.3188
Camera 3	0.0975
Baseline	0.2084
Scene Richness	0.2207
Outlier-Rejection	0.05582

Table 1: Quantitative comparisons of the trajectories from DeepTAM using the SUNCG dataset with three cameras in the simulated sensor rig. We use absolute trajectory error (ATE) as a metric to compare the performance.

5.3. Pose Fusion Results

With the SunCG dataset, we observed that DeepTAM was working well in the monocular case. This is rather expected since DeepTAM tracking network is trained from a mix of sequences from Sun3D and SunCG. We thus use the data collected from the three-camera-simulated-rig in MINOS to evaluate our pose fusion approaches. Fig. 8 shows a qualitative comparison of the three pose fusion approaches that we proposed. In Table 1, we show the quantitative evaluation of the approaches as well as the accuracy of the predicted poses of each camera using DeepTAM. We observe that pose fusion by removing the outliers performs better than the other two approaches.

6. Discussion

Although learning-based approaches are prone to overfitting, DeepTAM seems to generalize well to other datasets that we test on. It works well even when the camera motion is large. However, it performs poorly in textureless scenes, when using noisy depthmaps, or for fast camera motions. Leveraging the redundancy from a multi-cameras setup helps improving the tracking accuracy in such cases. Since DeepTAM was not trained on outdoor scenes, it performs poorly on AirSim. However, from our experiments we deduce that the issue in this case is primarily scaling.

7. Work Distribution

Mayank took the responsibility for rewriting portions of the original DeepTAM source code which was required to interface with our algorithms. He, along with Rohit, wrote the pose fusion code based on outlier rejection. He also collected the dataset from the AirSim simulator to test the generalizability of DeepTAM. He, with help from Parker, wrote major portions of the final report.

Parker wrote the code used for pose fusion for the naive method and the method which combines SIFT features with depth image homogeneity for weighted averaging.

Solving the issues associated with the depth images and how they interfaced with DeepTAM fell to Rohit. Pipeline for extracting and pre-processing bag files, depth map refinement, and ground truth from checkerboard patterns were all also essential tasks he was responsible for. He, along with Mayank, streamlined the DeepTAM pipeline to use multiple cameras.

Dataset collection was facilitated by Fadhil. He collected the SunCG dataset. He also wrote the code to compute ground truth from April tags in the datasets, however the poses obtained from this method proved to be unreliable for reasons stemming from the image sequences.

All of us were collectively responsible for collecting dataset from the CVG sensor setup, and making the final poster.

8. Code Contribution

The code for the Multi-Camera DeepTAM method can be found on GitHub (https://github.com/surirohit/multi-camera-deeptam). The original DeepTAM source code was reorganized by Mayank, and wrapped into the class SingleCamTracker to make it easier to interface and run tests with different datasets with. The MultiCamTracker class was created using instances of SingleCamTracker class for each camera and propagating these instances through the DeepTAM algorithm. Pose fusion methods were implemented which are fed the depth and color images from each camera in the setup. Methods for fusing can be selected from within the code.

Acknowledgements

We would like to thank CVG lab for assisting us with this project. Specifically, Marcel Geppert and Viktor Larsson were instrumental in helping us develop our methods and providing feedback.

References

[1] M. O. A. Aqel, M. H. Marhaban, M. I. Saripan, and N. B. Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1897–1897, Oct 2016.

[2] F. Camposeco, T. Sattler, and M. Pollefeys. Minimal solvers for generalized pose and scale estimation from two rays and one point. In *European Conference on Computer Vision*, pages 202–218. Springer, 2016.

[3] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. Mar. 2018.

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

[5] C. Forster, M. Pizzoli, and D. Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[6] M. Geppert, P. Liu, Z. Cui, M. Pollefeys, and T. Sattler. Efficient 2d-3d matching for multi-camera visual localization. *arXiv preprint arXiv:1809.06445*, 2018.

[7] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, Feb 2008.

[8] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[9] G. H. Lee, B. Li, M. Pollefeys, and F. Fraundorfer. Minimal solutions for the multi-camera pose estimation problem. *The International Journal of Robotics Research*, 34(7):837–848, 2015.

[10] P. Liu, M. Geppert, L. Heng, T. Sattler, A. Geiger, and M. Pollefeys. Towards robust visual odometry with a multi-camera system. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1154–1161. IEEE, 2018.

[11] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[12] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[13] C. Netramai, H. Roth, and A. Sachenko. High accuracy visual odometry using multi-camera systems. In *Proceedings of the 6th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, volume 1, pages 263–268, Sep. 2011.

[14] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327, Nov 2011.

[15] R. Pless. Using many cameras as one. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–587. IEEE, 2003.

[16] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017.

[17] H. Seok and J. Lim. Rovo: Robust omnidirectional visual odometry for wide-baseline wide-fov camera systems, 2019.

[18] S. Shah, D. Dey, C. Lovett, and A. Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*, 2017.

[19] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[20] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. Large scale sfm with the distributed camera model. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 230–238. IEEE, 2016.

[21] R. Tron, R. Vidal, and A. Terzis. Distributed pose averaging in camera networks via consensus on se (3). In *2008 Second ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–10. IEEE, 2008.

[22] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] H. Zhou, B. Ummenhofer, and T. Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018.