DPT-SLAM: Dense Point Tracking for Visual SLAM

Tjark Behrens
tbehrens@student.ethz.ch

Damiano Da Col
ddacol@student.ethz.ch

Théo Ducrey
tducrey@student.ethz.ch

Wenqing Wang
wangwen@student.ethz.ch

Abstract

Simultaneous Localization and Mapping (SLAM) aims at creating a 3D map of the environment while continuously estimating the position and orientation of the agent. Traditional solutions often employ complex and expensive arrays of sensors such as cameras, LiDAR, and IMU. We instead focus on Visual SLAM, where the primary source of data comes from monocular, stereo, or multi-camera setups, providing a simpler low-cost alternative. Previous works rely on costly and incoherent pairwise optical flow estimates, which fail to effectively harness the rich temporal information present in input sequences. To address these shortcomings, we present DPT-SLAM which combines dense point tracking into a visual SLAM framework. This novel formulation leverages dense tracks as a meaningful initialization of optical flow estimates, showing improved flow accuracy compared to state-of-the-art visual SLAM pipelines.

1. Introduction

SLAM consists in localizing an agent equipped with onboard sensors, and simultaneously mapping the environment around it. It is particularly useful in robotics applications that act in unknown environments, such as indoor robots, autonomous vehicles, and autonomous drones. Accurate and robust solutions to this problem rely mainly on arrays of sensors, combining cameras, LiDAR, and inertial measurement units (IMU), which can be expensive and difficult to integrate. This paper focuses on Visual SLAM, which primarily utilizes monocular, stereo, or multi-camera setups as the main source of data, offering a more straightforward and cost-effective alternative. Despite the recent advances, existing Visual SLAM solutions often depend on costly and incoherent pairwise optical flow estimates. These estimates fail to fully exploit the rich temporal information embedded within input sequences, leading to sub-optimal performance. To address these limitations, we introduce DPT-SLAM, a novel Visual SLAM framework that incorporates dense point tracking. By leveraging dense tracks as a meaningful initialization for optical flow estimates, DPT-SLAM enhances the accuracy and robustness of flow calculations. DPT-SLAM integrates the strengths of both indirect and direct Visual SLAM methods. Indirect methods, which involve extracting and matching key-points across frames and minimizing the reprojection error, are robust to lighting changes and large motions but are computationally expensive. Direct methods, which operate on pixel intensities and minimize photometric error, can potentially offer more accurate reconstructions but are also more difficult to optimize. By utilizing optical flow estimates from dense point tracks and minimizing the reprojection error, DPT-SLAM takes the best from both approaches, taking the simpler optimization of indirect methods while leveraging all the information of image sequences. Our system is based on DROID-SLAM [14], but unlike DROID-SLAM, we use point tracks to obtain a meaningful initialization of flow. Our approach builds on recent advancements in point tracking and optical flow estimation. Notably, we adapt the CoTracker [8] method, which jointly tracks dense points across video sequences, to operate online within our SLAM framework. This algorithm achieves higher accuracy and robustness in tracking compared to previous methods, which often track points independently. Additionally, inspired by Le Moing *et al*. [10], we perform interpolation from sparse to dense tracks, which allows for a more comprehensive coverage of the scene. Following this interpolation, we extract our meaningful optical flow initialization from the dense tracks. We further refine these flow estimates using RAFT [13], a recurrent neural network-based approach that applies iterative flow updates. Finally, camera poses and 3D points are jointly optimized with DROID-SLAM's Dense Bundle Adjustment layer. We performed the evaluation on the TartanAir dataset [16] and compared our results to DROID-SLAM, which was trained on this dataset. We report improved optical flow accuracy through this approach, which shows that integrating point tracking into Visual SLAM can lead to a better overall system. This integration has the potential to enhance the robustness and accuracy of Visual

SLAM, making it more a viable option for real-time, low-cost applications in diverse environments.

2. Related Work

Visual SLAM. Visual Simultaneous Localization and Mapping (SLAM) focuses on the dual task of mapping an environment and tracking the trajectory of the observer using visual inputs, from one or more RGB cameras. The traditional SLAM formulation is divided into two main components: the *front-end* responsible for processing the raw sensors' data to extract the initial camera poses and 3D Points, and the *back-end* which jointly optimizes the trajectory and map, to improve accuracy and consistency. Visual SLAM approaches can be divided into indirect and direct. Indirect methods [6, 9, 11] first extract distinctive key-points and attach feature descriptors to them. Then Features are matched across pairs of frames to establish stereo correspondences, which enable the estimation of 3D points in space and camera poses, by minimizing the reprojection error (the distance between a projected 3D point and its corresponding observed location in the image). These methods are robust to lighting changes or large frame-to-frame motions but also slow due to feature extraction and matching. Direct methods [4, 5, 12] directly operate on pixel intensities, allowing the use of all the image information, by minimizing the photometric error between frames. This allows for potentially more accurate reconstructions but also more complex optimization and problems with large displacements between images. Our system, similarly to DROID-SLAM [14], takes the best of these two approaches. In particular, we use all the information in the image by extracting optical flow estimates from dense point tracks and we minimize the reprojection error which constitutes a simpler optimization problem. This allows higher modeling capabilities combined with a smoother objective function.

Point Tracking, involves estimating the motion and visibility across consecutive frames of a pixel or feature, corresponding to a 3D point in the environment. Recently, TAP-Net [2] introduced the problem of tracking any point (TAP) and since then many solutions have been proposed in this direction. PIPs [7] tracks points through occlusions with a sliding window approach. TAPIR [3], further improves the global motion trajectory by combining a matching stage inspired by TAPNet [2] with the refinement step from PIPs [7]. Yet, these two methods still track points independently. OmniMotion [15] employs a quasi-3D canonical volume to represent videos, allowing for accurate pixel-wise tracking through occlusions, but the high computational cost makes it unsuitable for real-time tasks such as SLAM. Finally CoTracker [8] proposes to track dense points jointly across a video sequence, leveraging the correlation between points. This formulation achieves higher accuracy and robustness compared to previous tracking algorithms. For this reason, we chose to adapt CoTracker to run online, such that it can be deployed in DPT-SLAM.

Optical Flow, densely estimates the motion of pixels between pairs of frames. As it is a crucial part of visual SLAM, providing dense correspondences between frames, we are interested in retrieving accurate optical flow predictions in our system. Traditional methods relied on brightness constancy and spatial smoothness constraints, while modern solutions tackle this problem using deep learning. In particular, DC Flow [17] introduces a fast way to compute the 4D cost volume which was used in most subsequent works. RAFT [13] iteratively updates flow estimates through a recurrent neural network that performs lookups on 4D correlation volumes. Finally, DOT [10] building up on CoTracker and RAFT, proposes a robust algorithm at the intersection between point tracking and optical flow. It achieves state of the art tracking performance compared to other solutions, such as applying CoTracker densely on all pixels, but also reduces the runtime by two orders of magnitude. Inspired by this work we interpolate sparse tracks from CoTracker into dense tracks and refine flow estimates between pairs of frames with RAFT.

3. Method

3.1. Pipeline

Our method follows the general architecture of DROID-SLAM [14]. We adapted the pipeline such that it extracts optical flow estimates from dense point tracks [8] and iteratively refines them [13]. To achieve this, we accumulate four images at a time and feed them into an online version of CoTracker to generate sparse point tracks. Inspired by the method proposed by Le Moing *et al*. [10] we first interpolate the tracks to get a rough dense point tracking estimate that we use to select the keyframes. These interpolated tracks offer a meaningful initialization that is further refined in RAFT to produce accurate flow estimates. Gaussian weighting on the initial CoTracker tracks acts as a heuristic to estimate the confidence of each per-pixel flow vector. Given this information, a dense bundle adjustment layer optimizes poses and depths minimizing the reprojection error. Consequently, newly collected images lead to an updated frame graph and new optical flow pairs. The whole process is depicted in Figure 1. We will use the following sections to explain our own contributions in greater detail that act upon the standard DROID-SLAM pipeline.

3.2. Own Contributions

3.2.1 Online CoTracker

Our primary goal was to leverage temporal information to improve DROID-SLAM, specifically by enhancing the internal flow computation utilized by DROID. A recent notable work in this field is DOT, which computes a flow esti-

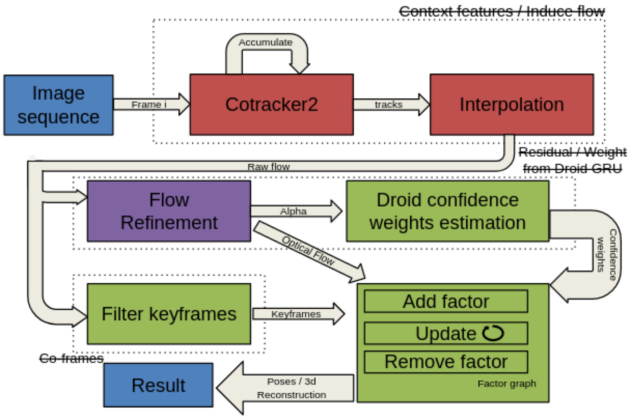

Figure 1. Overview of the DPT-SLAM pipeline

mate using CoTracker and refines it with RAFT. We adopted a similar approach, but implemented it online, calculating the flow not just between the first and last frames of the video, but between every pair of frames exhibiting sufficient motion.

Our approach relies on the newly released CoTracker2 implementation. The underlying method in DOT [10] is inherently offline requiring the total video length in advance. This makes it impractical to use for a Visual SLAM pipeline. Hence, we adapted an online CoTracker version, which was pre-trained by Karaev *et al*. [8] but tracks a fixed grid of points at the start. This poses the problem that the method rapidly reaches the state where all points are out of frame for large camera displacements within a scene.

To continuously track a large number of points throughout the video, we developed a solution that samples points from the first frame of the current video segment, extending tracks with each new frame. We check for a minimum count of tracks visible in at least one of the last five frames. If this threshold is not met, we resample from the start of the current window (- four frames) and restart CoTracker from there using the new points. To ensure continuity across multiple CoTracker instances, we merge the endpoints of old tracks with the starting points of new ones that are sufficiently close to each other.

We used two resampling methods. The first, which CoTracker originally used, employs a grid that is resized based on the number of points to be resampled in the image. The second method, which did not show significant improvement, uses Harris corner detection. In this method, the frame is divided into cells, and Harris corner detection is run within each cell to obtain the required number of tracks. Only enough tracks are resampled in each cell to maintain a visibility threshold, and the ends of the already visible tracks are kept for the future CoTracker instance. This allows long-term key-frame dependencies in the frame graph of DPT-SLAM and the grid cell for a relatively uniform sampling through the frame avoiding a cluster of key-points in only one part of the image, which would force us to interpolate the flow in a big part of the image not covered by any tracks.

Although we lacked the resources to retrain CoTracker, we anticipate improvements when using CoTracker with Harris once retraining is feasible. Additionally, as suggested on the CoTracker forum, making this approach truly online involves reducing its window size or employing an adaptive window size.

3.2.2 Optical Flow Refinement

After getting the dense point tracks from online CoTracker, we can obtain a coarse optical flow by interpolating the changes in coordinates of the tracked points. We refine the flows using RAFT[13] and feed them to our DPT-SLAM system. As shown in Figure 2, the optical flow derived from dense point tracks contains the general motion information of objects in the scene, but it lacks precision and needs improvement, particularly in handling object edges. Therefore, we input the original images along with the coarse flow into the RAFT neural network to obtain a refined flow. This refined flow is more detailed and beneficial for subsequent bundle adjustment. Since the refined flow will be accessed multiple times in further steps, we store it in a dictionary to prevent redundant computations.

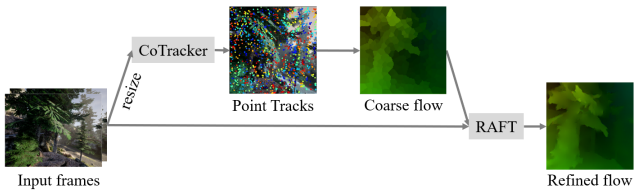

Figure 2. From tracks to flow

3.2.3 Gaussian Weight Approximation

In the original version, Teed *et al*. [14] input learned confidence weights in their dense bundle adjustment. By exchanging the Gated Recurrent Unit (GRU) used previously with RAFT to refine our initial flow estimates, we lose the ability to generate learned weights. Instead, we rely on a heuristic that fits our domain. In particular, we give the initially estimated tracks by CoTracker a higher weighting. We believe that this is a natural assumption as we are more confident around points that have been properly tracked rather than interpolated. The idea is illustrated in Figure 3, where each initial track is weighted by a normal distribution. Occluded points or tracks that move out of frame are assigned zero weight.

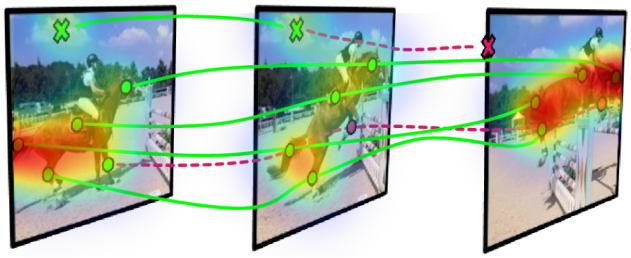

Figure 3. Gaussian weights around the estimated sparse tracks.

3.2.4 DROID-SLAM adaptations

After having obtained refined flow estimates for all pairs of frames in the graph and the corresponding confidence weights, we proceeded to adapt the DROID-SLAM update step such that it integrates these changes. Refined flow for pairs of frames currently contained in the graph is now stored in a dictionary to avoid redundant flow computations. Such that the refinement is only performed for newly added pairs when a new key-frame is selected. The other change was done in regards to the bundle adjustment (BA) call: DROID-SLAM calls the BA only once per iteration but performs multiple iterations for all added key-frames as it is progressively refining the flow each time. DPT-SLAM only performs one update call per added key-frame, thus calling the BA multiple times to progressively improve camera poses and depth maps. The damping factor used in the Dense Bundle Adjustment layer is learned in DROID-SLAM and output by the GRU, in our case we had to choose a fixed value for it, which depends on the iteration number (higher for earlier iterations to aid in the convergence of the Bundle Adjustment (BA) to a solution).

3.3. Team contributions and Code Implementation Overview

DPT-SLAM is a Visual SLAM system based on DROID-SLAM. In particular, DROID-SLAM is based on a factor graph that contains frame pairs and is continuously updated, for each newly added key-frame. From all frame pairs in the graph, flow estimates are extracted from camera poses and depth maps and refined through a GRU unit, which also outputs per-pixel confidence weights. Finally, in the Dense Bundle Adjustment layer, poses and 3D points are jointly optimized to adhere to the refined flow estimates. The system processes the whole sequence following these steps. DPT-SLAM borrows the factor graph and bundle adjustment implementation from DROID-SLAM while completely changing the front-end of the system. We extract tracks with CoTracker and thus had to rewrite parts of this as well such that it can run in an online fashion. The refinement is done through RAFT while the confidence weights were completely implemented by us. Théo Ducrey focused on adapting CoTracker such that it can run online and tracks can be extended to newly input frames. Wenqing Wang was tasked with integrating CoTracker and RAFT into our system, namely computing the refined flow from tracks and feed it to the bundle adjustment. Tjark Behrens modified the key-frame selection to decouple it from the GRU and instead base it on the initial track interpolation while also working on the confidence weights. Damiano Da Col mainly focused on DROID-SLAM, adapting the factor graph and the update step such that, for each new key-frame, we get the refined flow for all pairs of frames in the graph and feed this to the bundle adjustment. Finally, we also tested different implementations of the distance metric to select pairs of frames to add to the graph. In the current version, the system looks at flow magnitude rather than distance between camera poses.

4. Results

4.1. Experiments

The experiments were conducted on the TartanAir dataset [16], which was specifically designed for the visual SLAM use case. The full dataset includes scenes in over 30 environments, with each scene comprising approximately 300 to 3000 frames. TartanAir offers a diverse range of ground truth modalities, making it an attractive choice for analyzing Visual SLAM systems. For us, the 3D trajectory and the optical flow are particularly important. TartanAir is categorized as a very difficult Visual SLAM dataset and therefore offers an interesting reference for evaluation. While DROID is capable of processing almost all scenes and output meaningful poses and depths, most other prior works have several failure cases.

Due to the limitations in terms of GPU memory in the student cluster, we were only able to run our system on three scenes that contained less than 500 frames, even though we experimented with some workarounds by moving objects to CPU memory. However, since this problem can be quickly solved with more resources, and our three tested scenes serve as a good baseline, we have neglected this restriction for the time being.

DROID-SLAM [14] serves as a baseline to our DPT-SLAM method. To investigate whether the introduction of dense point-tracking achieves higher performance, comparing against DROID-SLAM is sufficient to answer this question. This is the case because, in theory, both systems work on the same components with the difference that DPT-SLAM was adapted to support dense point-tracking as described in section 3.

4.2. Metrics

To give a meaningful comparison between DROID- and DPT-SLAM we use two evaluation metrics in the three tested scenes.

Evaluation of the produced poses is quantified by the Absolute Trajectory Error (ATE). It measures the absolute difference between the estimated and the ground truth trajectory. ATE provides an overall measure of the global consistency of the trajectory estimation. Given an estimated trajectory $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and the corresponding ground truth trajectory $\mathbf{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n\}$ where $\mathbf{p}_i$ and $\mathbf{q}_i$ represent the estimated and ground truth positions at frame i respectively, the ATE is computed by:

$$\text{ATE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\mathbf{p}_i - \mathbf{q}_i\|^2}$$

Before evaluation, the predicted trajectory $\mathbf{Q}$ is scale- and pose-aligned with respect to $\mathbf{P}$. A lower ATE value indicates a trajectory that closely follows the ground truth, reflecting better performance of the SLAM algorithm.

One of the main goals of this project was to check whether the meaningful initialization of the flow by the dense point tracking estimates yields better optical flow predictions. We employ endpoint error (EPE) as our evaluation metric in this regard.

Given an estimated optical flow field $\mathbf{F} \in \mathbb{R}^{H \times W \times 2}$ and the corresponding ground truth flow field $\mathbf{G} \in \mathbb{R}^{H \times W \times 2}$, consisting of $n = H \cdot W$ flow vectors each.

$$\text{EPE} = \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{f}_i - \mathbf{g}_i\|$$

Again, a lower value represents a more accurate matching with the ground truth.

To effectively evaluate the EPE in our setting, we need pairwise ground truth for non-consecutive frames due to the selection of keyframes in the pipeline. This is not off-the-shelf available in TartanAir. Therefore, we adapt a published Python program from [16] to integrate this functionality. We use cycle consistency in the process to produce ground truth masks for the optical flow, which are an integral part of the EPE evaluation.

4.3. Evaluation

	Metric	S01	S02	S03
DROID-SLAM	ATE (m)	0.06	0.12	0.05
	EPE (px)	2.80	10.05	4.05
DPT-SLAM (ours)	ATE (m)	3.66	2.85	3.32
	EPE (px)	1.51	7.04	4.23

Table 1. Evaluation: Comparison of our system DPT-SLAM with its main competitor DROID-SLAM. Quantitative results in Absolute Trajectory Error (ATE ↓) and Endpoint Error (EPE ↓).

The most important takeaway is that DPT-SLAM's changed front-end leads to a superior performance in terms of optical flow. As seen in Table 1, our modifications yield competitive results compared to DROID-SLAM, strongly outperforming it in two out of three tested scenes. In those scenes, the error is reduced by more than 30% each.

However, the higher accuracy in this regard did not result in a more accurate reconstruction of the 3D trajectory. DPT-SLAM achieves only ATE values two orders of magnitude higher than those of DROID-SLAM, potential reasons and fixes are discussed in section 5. DPT-SLAM still manages to grasp the general idea of its position and orientation in the scene as depicted qualitatively in Figure 4.

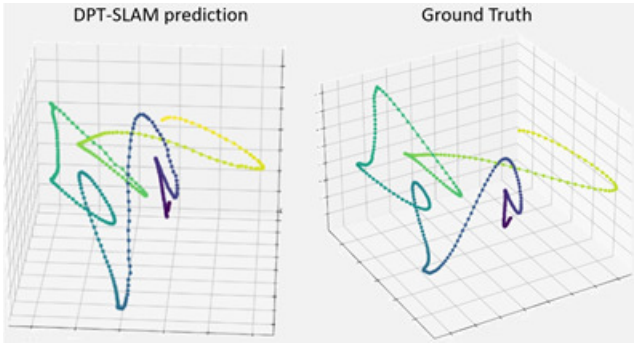

Figure 4. Comparison of camera trajectory of the first scene in our evaluation. The general notion of the ground truth (right) is also apparent in our prediction (left). However, due to drift, the ATE score is not competitive with DROID-SLAM.

4.3.1 Ablation study

		512	1024	2048
Detection Mode	**Harris Detection**	1.42	1.52	1.42
	Regular Grid	1.51	1.48	1.43

Table 2. Ablation Study: Variation in detection style and number of keypoints (512/1024/2048) used in CoTracker2. Results evaluated in Endpoint Error.

We tested DPT-SLAM's underlying point tracking algorithm on how to detect keypoints and variations in the number of tracks to determine the optimal value trading off efficiency and performance. Interestingly, the EPE remained at the same level for most combinations. Reducing the number of keypoints even further than 512, however, sometimes resulted in failure cases due to missing tracks. Hence, we chose this number and a regular grid as the detection method due to its simplicity for our main experiments.

5. Future Work

Despite our work provides a more accurate flow esti-

mation than DROID-SLAM [14], further improvement is needed to achieve a more robust and accurate camera pose and 3D point estimation. The following fields have been identified for future work:

5.1. Training of the whole pipeline

Due to the limitation in computational resources (i.e. GPU), we utilized the pre-trained network for point tracking and optical flow refinement. However, the parameter weights need to be refined to achieve the fully optimized performance. Thus, several weights require fine-tuning, including:

- **Confidence weights**
  The current confidence weights of optical flow are generated using Gaussian weighting rather than neural networks. Though it is simpler, it is not robust to estimation error. Therefore, our next step would be to train a network to compute the confidence weights.

- **Damping factor**
  The damping factor plays a crucial role in controlling the convergence speed and stability of the system. A higher damping factor allows the model to update more conservatively, and vice versa. In DROID-SLAM implementation, the damping factor is learned in the GRU. However, since we removed the recurrent optimization, the learned damping factor is no longer available. Our solution is to set this factor as a hyperparameter with an initial high damping value (i.e. 1×10^{-4}) and set it to a lower value after 2 iterations. Though the current implementation leads to a rather converged result, learning this parameter using a neural network can further optimize our system.

- **Point tracks**
  Currently, we use the CoTracker2 pre-trained weights to track the dense points, but it could be improved by re-training the network to suit our use case.

5.2. Making CoTracker completely online

Our implementation of the point tracks is a window-based online system, meaning it collects a batch of four frames and outputs their corresponding point tracks. This approach works better with a total frame number divisible by 4. However, to make our system more robust, a fully online CoTracker (with window size 1) is required.

5.3. Loop closure and global bundle adjustment

Since the scene from the camera is constantly moving, the tracked points keep disappearing in the frame. If the point disappears for more than several frames, the track is considered invalid. Thus, it is harder for the point track to realize loop closure compared with DROID-SLAM's method. Besides, loop closure is needed for global bundle adjustment. Thus, using only point tracks without long-term frame correlations may lead to inaccurate estimation for long videos. Future work can focus on combining point tracking with other methods to gain both local and global information.

5.4. Trajectory filler

During our experiments, we observed that the camera poses and depth maps from key-frames are usually accurate, but due to the sensitivity to prediction error, the filled trajectory can sometimes include unexpected twists and turns. Therefore, modifying the trajectory filler to make it more robust to the errors in flow would be our next step to focus on.

5.5. Bundle adjustment on dense tracks

An interesting research direction would be, following the suggested approach in LEAP-VO [1], to perform bundle adjustment directly on the predicted tracks instead of the resulting optical flow estimates. This would avoid the computational overhead introduced by the pairwise flow estimates, making the whole approach potentially faster. However, this requires a real time dense point tracking algorithm and hence, relies on further research improvements in this domain.

6. Conclusion

In this paper, we introduced DPT-SLAM, a novel Visual SLAM framework that integrates dense point tracking to enhance optical flow estimation into a Visual SLAM system, to improve its robustness and accuracy. By leveraging dense tracks for optical flow initialization, DPT-SLAM addresses the shortcomings of traditional Visual SLAM approaches that overlook the rich temporal information of input sequences. We adapted the CoTracker algorithm to operate online within our SLAM framework, allowing for continuous point tracking. Additionally, we refined our optical flow estimates using RAFT, a recurrent neural network-based approach and incorporated Gaussian weighting, to introduce per-pixel confidence in our flow estimates for Dense Bundle Adjustment. Experimental results on the TartanAir dataset demonstrate that DPT-SLAM significantly improves optical flow accuracy compared to the state-of-the-art visual SLAM system DROID-SLAM. Specifically, our approach reduces the endpoint error (EPE) by over 30% in two out of three tested scenes. Overall, DPT-SLAM represents a step forward in making Visual SLAM more accurate and viable for real-time, low-cost applications in diverse environments. By effectively utilizing dense point tracking, our approach has the potential to enhance the performance of SLAM systems in various robotics and autonomous vehicle applications.

References

[1] Weirong Chen, Le Chen, Rui Wang, and Marc Pollefeys. Leap-vo: Long-term effective any point tracking for visual odometry. *arXiv preprint arXiv:2401.01887*, 2024.

[2] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.

[3] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023.

[4] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[5] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[6] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

[7] Adam W Harley, Zhaoyuan Fang, and Katerina Fragkiadaki. Particle video revisited: Tracking through occlusions using point trajectories. In *European Conference on Computer Vision*, pages 59–75. Springer, 2022.

[8] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together, 2023.

[9] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[10] Guillaume Le Moing, Jean Ponce, and Cordelia Schmid. Dense optical tracking: Connecting the dots. *arXiv preprint arXiv:2312.00786*, 2023.

[11] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[12] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[13] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 402–419. Springer, 2020.

[14] Zachary Teed and Jia Deng. Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras. *Advances in neural information processing systems*, 34:16558–16569, 2021.

[15] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking everything everywhere all at once. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19795–19806, 2023.

[16] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. 2020.

[17] Jia Xu, René Ranftl, and Vladlen Koltun. Accurate optical flow via direct cost volume processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1289–1297, 2017.