TrimBot2020 Deliverable D3.5

Vehicle and arm localisation

Principal Author: ETH Zürich (ETHZ)
Contributors: Albert-Ludwigs-Universität Freiburg (ALUF), University of Edinburgh (UEDIN)
Dissemination:

Abstract: This report describes and evaluates the algorithms that have been developed to localize the robot (vehicle) in a known garden environment and to localize the cutting head of the arm with respect to the cutting regions.
The vehicle localization algorithm, which is tightly integrated into the SLAM system used to build 3D maps of the garden, has already been described and evaluated in Deliverable D3.2 (*Implementation and evaluation of SLAM, 3D from binocular and motion stereo*). For vehicle localization, the report thus more focuses on integrating semantic information into the localization process for robust long-term localization and multiple sensor fusion.

Deliverable due: Month 32

1 Overview

This report describes localization algorithms, designed for localizing the vehicle and the arm of the robot, respectively. The algorithms serve different purposes and are intended to be executed at different points in time. In addition, they use different camera systems as input. In the following, we briefly describe both algorithms before providing details in Sec. 2 and Sec. 3, respectively. Experimental evaluations of the two systems are provided separately in each section.

The objective of the *vehicle localization* system is to determine the current position and orientation of the robot, i.e., its pose, in a known garden environment. To do so, we first build a 3D map of the garden in an offline processing step. This map is then used to determine the robot pose at any point in time during online processing. This is done using the 10 camera multi-camera rig mounted on the chassis of the robot. The vehicle moves rather slowly, allowing the localization algorithm to use all 10 cameras. The results of the vehicle localization algorithms are used to navigate the robot through the garden, allowing it to safely reach the target positions and objects that need to be cut or trimmed.

The objective of the *arm localization* is to determine the pose of the cutting tool relative to the object that needs to be cut. In contrast to the vehicle, the arm can move faster. As such, the arm localization system needs to run at a higher framerate. Thus, only one stereo pair mounted on the end of the arm is currently used for localization[1].

Notice how both localization algorithms solve different problems: The vehicle localization system needs to provide an absolute pose in the garden to be useful for navigation. In contrast, a global pose in the garden is not required for cutting. Instead, a relative pose with respect to the object to be cut is required. Arm localization also only needs to be performed once the robot has reached the object. Thus, both algorithms are not run at the same time as the computational resources of the TrimBot are limited.

There is another important difference between the settings of the two algorithms. For vehicle localization, it is safe to assume that the largest part of the garden remains static. As such, it can rely on a pre-built map. In contrast, the geometry and appearance of an object change as it is cut, making it necessary that the arm localization algorithm adapts to such changes.

2 Vehicle Localization

We have developed two vehicle localization approaches as part of the TrimBot2020 project. The first approach, which is used on the TrimBot itself, is tightly integrated into the robot's SLAM system. It treats localization as a loop closure problem against an existing map built by the SLAM algorithm. At the same time, the pose of the robot is continuously tracked using SLAM, allowing us to handle parts of the scene that have not yet been mapped or where the existing map differs from the actual geometry and appearance of the scene.

The localization system used on the TrimBot relies on local features (SIFT [1] is currently used) to establish correspondences between images taken at different times or between images and the 3D map. Thus, this approach has problems to robustly and accurately localize the robot if the appearance and geometry of the scene observed at the current point in time differs

[1] In addition, the cameras mounted on the vehicle itself would likely not be able to see all of the object that needs to be cut.

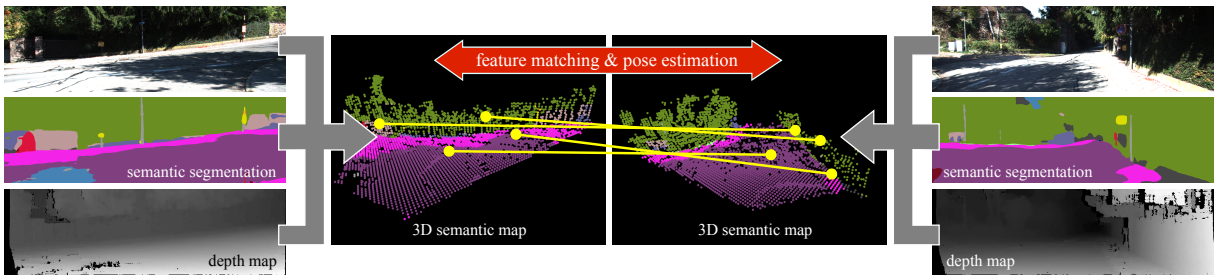

Figure 1: Semantic visual localization: Given an image, its corresponding semantic segmentation, and its corresponding depth map, we build a 3D semantic map. This 3D semantic map is a voxel volume, where each voxel is either unoccupied, occupied by a certain semantic class, or unobserved (i.e., we do not know whether it is empty or occupied as we cannot observe it due to occlusions). The core idea behind our semantic localization approach is to learn a descriptor that allows us to establish correspondences between semantic maps. These 3D-3D correspondences can in turn be used to compute the pose of one map (and its corresponding image) with respect to another map.

too much from the appearance and geometry encoded in the map. This has been shown in the experimental evaluation performed for Deliverable D3.2 (*Implementation and evaluation of SLAM, 3D from binocular and motion stereo*). To overcome this limitation, we developed a second, more experimental localization system [2]. It combines semantic scene understanding, in the form of semantic segmentation, and 3D geometry to learn descriptors that are robust under strong viewpoint changes as well as changes in scene appearance and geometry over time.

The SLAM-based localization approach has been described and evaluated in detail in Deliverable D3.2. In the following, we thus only explain the semantic-based localization system. We outline the main ideas behind our method. For more details on the algorithm and the experimental evaluation, please see our recent publication [2].

2.1 Semantic Visual Localization

The motivation behind our semantic visual localization approach is to develop a more abstract scene representation that is robust against changes in scene appearance and (moderate) changes in scene geometry. This in turn allows us establish correspondences between images taken under conditions widely differing from those under which the 3D map of the scene was built. For example, our approach is able to localize images taken during autumn, when there is no foliage on trees, against maps built during summer or winter, i.e., when there is foliage on trees and snow on the ground, respectively. In contrast, existing approaches from the literature either fail completely or have severe problems in such scenarios. Notice that these challenging scenarios are highly relevant to the TrimBot2020 project.

The main idea behind out semantic localization system is illustrated in Fig. 1. The input to our approach is an image and a depth map for that image. We use the image to compute a semantic segmentation, providing us with a more semantic understanding of the scene. At the same time, the semantic segmentation provides a layer of abstraction away from pixel intensities and thus scene appearance. To obtain a discriminative representation, we combine the semantic segmentation with the 3D depth map of the image. This is done by fusing them into a local semantic map, which is a voxel representation of the scene. Each voxel is either unoccupied

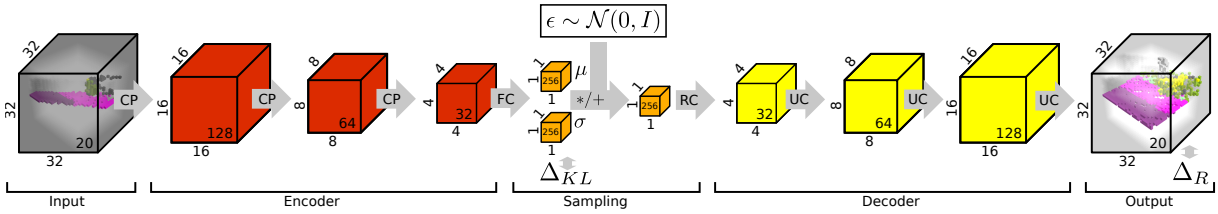

Figure 2: We use a variational encoder-decoder architecture for generative descriptor learning. Figure taken from [2]. *Legend:* CP = Convolution + Pooling, FC = Fully Connected, RC = Reshape + Convolution, UC = Upsampling + Convolution, KL = Kullback-Leibler Divergence, ΔR = Reconstruction Loss. The numbers at the bottom right of each block denote the number of feature channels.

(corresponding to free space in the scene), occupied by a certain semantic class, or unobserved (i.e., it is unclear whether it is occupied or not). Given the local map of the image and a more global map, constructed by fusing the semantic segmentations and depth maps of multiple images, localization reduces to finding correspondences between voxels in the two maps. The resulting 3D-3D matches can then be used for camera pose estimation.

There are two challenges that we face when trying to establish correspondences between voxels: changes in geometry and problems arising due to occlusions. Since we are interested in modelling higher-level structures, which are more stable to changes in geometry, we use rather large voxels with a side length of around 30cm. In addition, we consider a $32\times32\times32$ neighborhood around each voxel when computing descriptors for the voxel, i.e., we consider volumes of size about $10\text{m}\times10\text{m}\times10\text{m}$. At this size, there can be severe problems caused by occlusions, e.g., when a certain part of the scene is observed in the global but not in the local map due to the viewpoint of the query image. We solve this problem through a generative descriptor learning approach.

Fig. 2 illustrates our descriptor learning approach. Given a local voxel volume as input, which represents a partial observation of the scene, we train a variational encoder-decoder to predict a complete observation for the voxel volume. The approach is similar to a variational auto-encoder [3], with the distinction that the input and the output represent differing volumes. During the encoding stage, the network has to generate a compact representation of the volume that contains all the information necessary to reconstruct the completely observed volume. This compact representation, a 256 dimensional vector in our case, thus has to capture the gist of the scene part visible in the voxel volume. By construction, different partial observations of the same complete volume map to points close-by in this latent space of the encoder-decoder network. For localization, we thus only run the encoder part and use the embeddings in the latent space as descriptors. Nearest neighbor search combined with reasoning about co-visibility then yields a set of matches that can be used for pose estimation.

Our descriptor learning approach can be trained without supervision. Volumes extracted from the global map provide us with complete observations. Corresponding incomplete observations can be generated by extracting the corresponding volume from the local maps of the images used to generate the global map.

2.1.1 Experimental Evaluation

Experimental setup We evaluate our approach on the *Michigan North Campus long-term vision and lidar dataset* [4]. The dataset depicts a campus of the University of Michigan at different times of the year, with images and lidar measurements captured from a robot traversing the campus. We use the lidar measurements to generate depth maps as input for our approach. However, we trained our descriptors on the KITTI dataset [5], where depth maps were generated from stereo.

We built 4 global maps from data captured under different conditions: on a partially cloudy midday without foliage on the trees and without snow on the ground, on a sunny morning without foliage on the trees and without snow on the ground, on a sunny evening with foliage on the trees and without snow on the ground, and on a cloudy afternoon without foliage on the trees and with snow on the ground. We use data taken under the same conditions to query the different maps, enabling us to evaluate the impact of differing conditions on the localization performance.

Baselines We compare our approach against different state-of-the-art localization approaches from two different categories: appearance-based methods (SIFT [1], DSP-SIFT [6], MSER [7], DenseVLAD [8]) extract local features from the images, ignoring the available depth maps. In contrast, 3D descriptor-based approaches (FPFH [9], 3DMatch [10], CGF [11]) extract descriptors from 3D scene representations computed from the depth maps, ignoring the image information. We are not aware of a method that combines image and depth information for descriptor extraction. We also compare against PoseNet [12]. PoseNet is a deep learning-based approach that trains a neural network to directly regress a camera pose from an image.

We evaluate four variations of our approach: *Ours (semantic)* is the semantic localization approach outlined above. *Ours (semantic, acc.)* builds the local map that is used for localization from a small set of consecutive images, i.e., aims at localizing a short sequence of images. *Ours (geometric)* and *ours (geometric, acc.)* are variants of these two methods that do not use semantic but only geometric information during the generative descriptor learning and extraction stages. The comparison of the former two with the latter two variants allows us to understand the importance of using semantic scene understanding for our descriptors.

Comparison Fig. 3 shows the results of our experiments. As can be seen, appearance-based approaches perform well when the query images are taken under the same condition under which the map was constructed. This is due to the very similar appearance in the images and a lack of strong viewpoint changes between the query images and the images used to build the maps. Still, our methods using semantics perform competitively in this relatively easy scenario.

Looking at the scenarios where there are appearance and / or geometric changes between the query images and the map (cf. off-diagonal plots in Fig. 3) reveals that these scenarios are considerably harder for appearance-based approaches. 3D descriptor-based methods perform considerably better under these challenging conditions, but still have problems when there are geometric changes (in foliage or snow coverage). In contrast, all variants of our localization system perform considerably better in these scenarios. While there is degradation in performance, it is much less severe compared to the baseline approaches. We also notice a significant increase in localization performance when using semantics and geometry for descriptor learning compared to just using geometric information. This validates our idea of combining 3D geometry and semantic scene understanding to enable long-term localization.

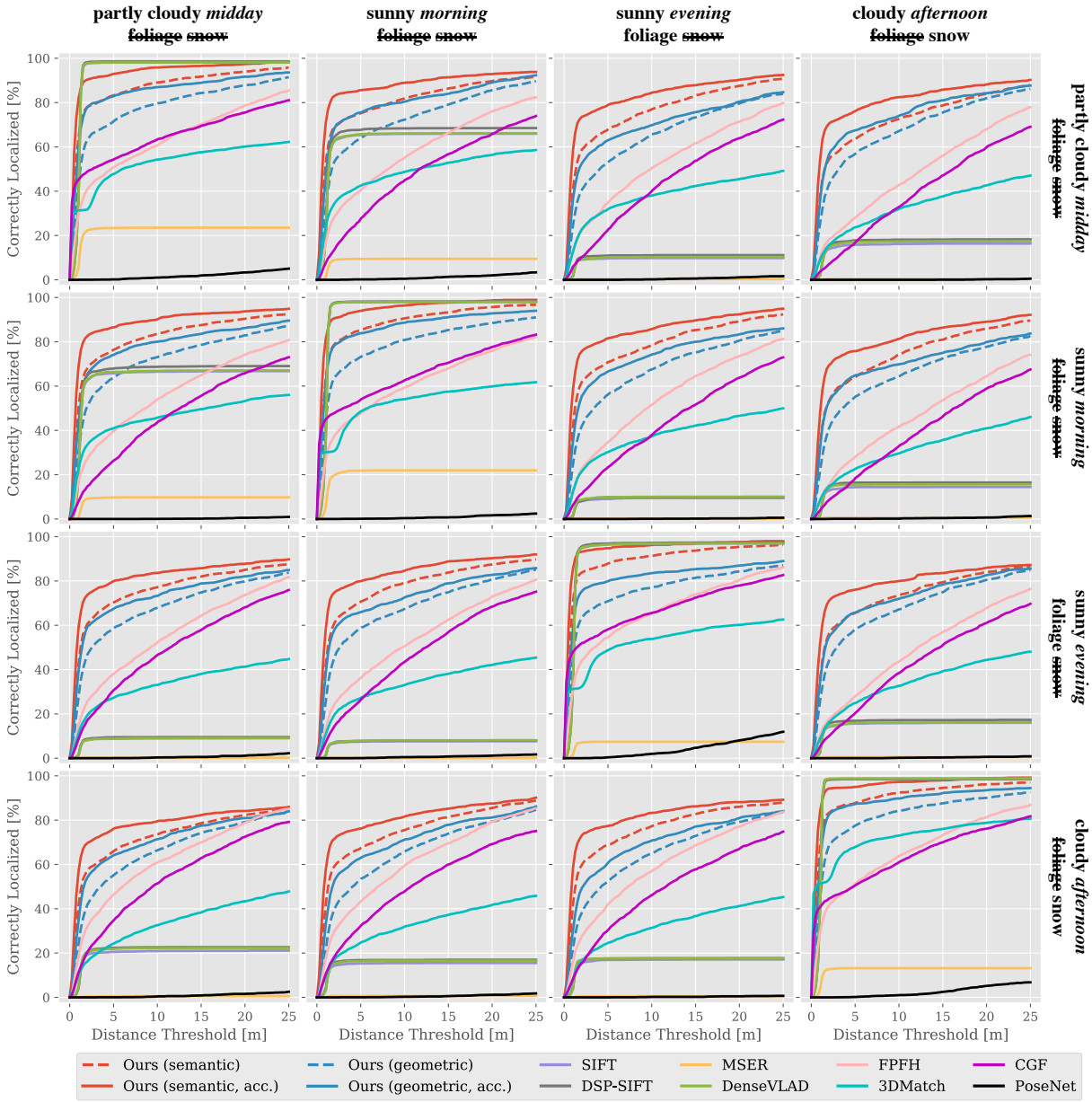

Figure 3: Evaluation of our semantic localization approach on the Michigan North Campus dataset. Each column represents query images taken under a certain condition while each row corresponds to the condition under which the global map was build. For each method, we show a cumulative histogram over the localization accuracy, i.e., we show the percentage of query images localized within Xm of the ground truth position for varying values of X. Figure and legend taken from [2].

Shortcomings The experimental evaluation presented above clearly shows that our semantic localization algorithm is a promising approach to robust and reliable long-term localization. However, it also has its shortcomings: on the practical side, localizing a single image takes around a second (not taking the computation of depth map into account). The bottleneck is the computation of the descriptors, taking around 1ms per volume on a powerful NVidia Titan X GPU. This run-time currently puts our method out of consideration for being run on the TrimBot itself. This is due to the weaker hardware available on the robot and the need to share the resources with other systems, e.g., the SLAM system (in which the approach would need to be integrated). The second shortcoming is in a key assumption made by our approach: We assume that the quality of the semantic segmentation is invariant to the appearance of the scene, allowing us to extract an invariant representation. As shown in Deliverable D3.2, this assumption is not true in practice as state-of-the-art semantic segmentation algorithms often fail when the input images differ too much from their training data. Still, our results demonstrate that there is value in integrating semantics, even if they are not perfect. Developing more robust segmentation algorithms thus seems an important research direction.

2.2 Multiple Sensor Pose Fusion with EKF

While pose estimation with Visual or Semantic SLAM can be highly accurate, it is not guaranteed to provide the smooth pose estimate that the navigation node expects. For this reason it is desirable to include a filter that can use other sensors mounted on the vehicle and motion model to constrain the estimated trajectory of the vehicle to be smooth.

In the Extended Kalman Filter (EKF) model [13], the process (robot motion) can be described as a nonlinear dynamic system with

$$\boldsymbol{s}_k = f(\boldsymbol{s}_{k-1}) + \boldsymbol{w}_k \ , \tag{1}$$

where $\boldsymbol{s}_k$ is the state vector (robot pose) at a discrete time t_k, the non-linear transition function is denoted as f and $\boldsymbol{w}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}_k)$ is the process noise with covariance $\boldsymbol{Q}_k$.

Our goal is to estimate the full 3D (6DOF) pose using the ROS framework [14], where the state $\boldsymbol{s}$ is defined as

$$\boldsymbol{s}_k = [\boldsymbol{x}, \boldsymbol{\varphi}, \dot{\boldsymbol{x}}, \dot{\boldsymbol{\varphi}}, \ddot{\boldsymbol{x}}] \ , \tag{2}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ is the linear translation (XYZ) and $\varphi \in (0, 2\pi)^3$ is the orientation angle (roll, pitch, yaw / RPY) component of the pose, $\dot{\boldsymbol{x}}, \dot{\boldsymbol{\varphi}}$ are the linear and angular velocities and $\ddot{\boldsymbol{x}}$ is the linear acceleration. The transition function f then describes the standard kinematic model:

$$\hat{\boldsymbol{x}}_k = \boldsymbol{x}_{k-1} + \int_{t_{k-1}}^{t_k} \dot{\boldsymbol{x}}_{k-1} \mathrm{d}t + \iint_{t_{k-1}}^{t_k} \ddot{\boldsymbol{x}}_{k-1} \mathrm{d}t^2 \ , \tag{3}$$

$$\hat{\boldsymbol{\varphi}}_k = \boldsymbol{\varphi}_{k-1} + \int_{t_{k-1}}^{t_k} \dot{\boldsymbol{\varphi}}_{k-1} \mathrm{d}t \ . \tag{4}$$

In practice f is discretized and linearized in the filter implementation.

2.2.1 Input Measurements

While the EKF allows a non-linear measurement model, we restrict ourselves to the linear case as with the standard KF. Then multiple sensor measurements $\boldsymbol{z}$ can be modeled as

$$\boldsymbol{z}_k = \boldsymbol{H}\hat{\boldsymbol{s}}_k + \boldsymbol{v}_k \ , \tag{5}$$

Sensor	$\boldsymbol{s}$	x	φ	$\dot{x}$	$\dot{\varphi}$	$\ddot{x}$	*Hz*
Visual SLAM	$\boldsymbol{z}^{slam}$	1	1	(Δ)	(Δ)		5
Wheel Odometry	$\boldsymbol{z}^{odom}$			Δ	Δ		50
IMU	$\boldsymbol{z}^{imu}$				1	1	250

Table 1: Mapping of sensor measurements $\boldsymbol{z}$ to components of the state vector $\boldsymbol{s}$. Observation matrix $\boldsymbol{H}$ for different sensor inputs and their update frequency. Δ indicates differential mode.

where $\boldsymbol{H}$ is the observation matrix and $\boldsymbol{v}_{k-1} \sim \mathcal{N}(0, \boldsymbol{R}_k)$ is the measurement noise with covariance $\boldsymbol{R}_k$. In the case of multi-sensor fusion, there are multiple sensor models and each sensor gives only partial observations of the state, i.e. only certain blocks of $\boldsymbol{H}$ are non-zero, as indicated in Tab. 1. In particular, our system has three inputs:

1) The **Visual SLAM** algorithm provides measurement of the global pose $\boldsymbol{z}_k^{slam} = (\boldsymbol{x}, \boldsymbol{\varphi})$ of the first (front) camera w.r.t. static map coordinates. Each measurement is an independent estimate of the pose given the images captured by the synchronized 10-camera rig. As such, the updates are not guaranteed to be smooth in pose space or time, occasionally resulting in lags (time periods with missing data) and jumps (sudden changes of pose).

2) Wheel Odometry sensors (Bosch Indego) count the rotations of wheels at both sides of the vehicle, which are transformed into linear and angular velocities $\boldsymbol{z}_k^{odom} = (\dot{\boldsymbol{x}}, \dot{\boldsymbol{\varphi}})$ corresponding to the vehicle center (of the rear wheel axis). Internally, those velocities are integrated into a trajectory relative to the initial pose of the vehicle, but for the purpose of the filter updates they are differentiated back to into velocities. The wheel odometry trajectory is typically smooth but drifts away from the actual trajectory due to wheel slippage and accumulated noise.

3) The **Inertial Measurement Unit** (Xsens MTi) includes three-axis gyroscopes and accelerometers that provide angular and linear accelerations, which are typically integrated internally by the sensor to velocities. In practice, we expect the sensor to produce $\boldsymbol{z}_k^{imu} = (\dot{\boldsymbol{\varphi}}, \ddot{\boldsymbol{x}})$ because angular accelerations are not included in the state. Because of the double integration needed to apply accelerations to pose, any measurement bias propagates quadratically, resulting in smooth but rapid drift of the integrated pose. Better results can be achieved when the bias is explicitly estimated and compensated [15], but this gives a significant advantage only in the case of highly dynamic systems such as fast moving and turning UAVs.

The different locations of the individual sensors on the vehicle is taken into account by transforming the measurements into a common reference frame (base link) using a physical robot model.

Measurement Covariance Estimation The model requires the measurement covariance $\boldsymbol{R}$ to be specified for all sensor measurements. We discuss below how the actual values are computed for our three input modalities.

The covariance $\boldsymbol{R}^{imu}$ of the IMU measurements is a constant matrix reported by the sensor driver (based on accuracy specifications). Similarly we set the wheel odometry covariance $\boldsymbol{R}^{odom}$ to a constant value (i.e. 0.1 m/s or 1 deg/s for velocities on the diagonal).

The estimation of the Visual SLAM pose covariance is complicated by the variable accuracy of independent pose estimates, which can include outliers, rendering a constant covariance unusable. In practice the observed behavior is that the pose estimate becomes unreliable in certain regions of map where a sufficient number of visible keypoint matches cannot be established, e.g. due to different lighting conditions. The reliability improves again when the robot leaves

such regions or conditions improve. This observation leads us to the assumption that although Visual SLAM poses are calculated independently, such estimates are still correlated due to visual overlap of consecutive camera views, and we can use the statistics of past poses to predict the covariance of the current measurement.

We assume the vehicle has a physical limit on the velocity $\dot{x}_m$ it can travel. Then any subsequent global pose measurements $\boldsymbol{z}_k^{slam}$ with position difference $\delta_k = ||\boldsymbol{x}_k - \boldsymbol{x}_{k-1}||$ exceeding the distance $\delta_m = \dot{x}_m(t_k - t_{k-1})$ it could travel at the maximum velocity for the corresponding time period should be deemed unreliable or as an outlier.

With this assumption we can estimate the current position variance $\hat{\sigma}_k$ from a buffer of n past position differences as

$$\hat{\sigma}_k^2 = \sigma_s^2 + \frac{1}{n}\sum_{i=k}^{k-n}[\max(\delta_i - \delta_m, 0)]^2 \ , \tag{6}$$

where $\sigma_s = 0.1$ m is a constant (static) position variance. However, a naive implementation using a finite (ring) buffer with fixed capacity n can result in sudden changes in pose when outstanding past values are dropped from the buffer. To lessen this problem, we use an exponential term to give more weight to more recent measurements:

$$\tilde{\sigma}_k^2 = \sigma_s^2 + \frac{1}{n}\sum_{i=k}^{k-n}[\, e^{-a(t_k - t_i)} \max((\delta_i - \delta_m), 0)\,]^2 \ , \tag{7}$$

which is then used to set the diagonal in the covariance matrix $\boldsymbol{R}^{slam}$ individually for all 3 position and 3 orientation terms.

Outlier Rejection Following the common EKF approach, incoming measurements are compared to the current pose estimate using the Mahalanobis distance, which takes into account covariances of both. If such a normalized distance exceeds a given threshold, the measurement is rejected as an outlier. When the uncertainty of the pose estimate increases, this causes the filter to accept even more distant measurements. Additionally, measurements with covariance above a certain threshold are rejected. Also, measurements that fall outside a given height (Z) range (under ground / in the air) are rejected.

In some situations, e.g., after the SLAM pose estimate recovers by switching to a distant location, all subsequent measurements are rejected as outliers. In order to recover from such situations, the filter is re-initialized when the state covariance estimate grows above a threshold.

Handling Lags Lags occur when measurements are not received at expected rate or for an extended period of time. This can occur in the difficult scenarios mentioned above, resulting in global localisation failures. In this time period the filter relies on integration of odometry and IMU measurements to update the current pose, which can result in pose drift.

We observe that when the SLAM pose recovers after lags as it moves out of difficult regions, the first measurements are often unreliable. This is probably a manifestation of a gradual transition into 'easier' regions with more matched features. For smoother transitions when Visual SLAM recovers, we switch the measurement $\boldsymbol{z}_k^{slam} = (\boldsymbol{x}, \boldsymbol{\varphi})$ to differential mode, i.e. the global pose is transformed into the velocity $\boldsymbol{z}_k^{\Delta slam} = (\dot{\boldsymbol{x}}, \dot{\boldsymbol{\varphi}})$. We switch to velocity when the update rate drops below a given threshold (1 Hz), and switch back to position after the rate of consecutive 10 measurements averages above the threshold.

2.2.2 Filter Updates

The EKF iteratively updates the state estimate vector $\boldsymbol{s}_k$ and associated state covariance matrix $\boldsymbol{P}_k$ in alternating predict and correct steps.

Initialization The filter is initialized with the pose

$$\boldsymbol{s}_0 = E[\boldsymbol{s}(t_0)] = [\boldsymbol{z}_0^{slam}, \boldsymbol{z}_0^{odom}, \ddot{\boldsymbol{x}}_0^{imu}] \ , \tag{8}$$

which is in our case the first observed global pose and actual velocity and acceleration as reported by the sensors. The initial state covariance $\boldsymbol{P}_0$ is an empirically set diagonal matrix.

Predict The new value of state vector is computed using the transition function from the previous state

$$\widehat{\boldsymbol{s}}_k = f(\boldsymbol{s}_{k-1}) \ . \tag{9}$$

Correct The state is updated based on measurement residual

$$\boldsymbol{s}_k = \widehat{\boldsymbol{s}}_k + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\widehat{\boldsymbol{s}}_k) \ , \tag{10}$$

where $\boldsymbol{K}_k$ is the near-optimal Kalman gain derived to yield a minimum mean square error (MMSE). Update equations for the state estimate and its covariance $\boldsymbol{P}_k$ are based on the local linearization of the underlying non-linear model and can be found in [14].

We run the predict-correct cycle at constant rate of 10 Hz, when measurements taken during the cycle period (100 ms) are aggregated. With this setting the filter node can run in a single mobile CPU thread.

2.2.3 Evaluation

We evaluate our EKF implementation on the Renningen garden dataset already used in Deliverable D7.5. In this dataset, we observed frequent jumps in the pose estimated by SLAM due to the repetitive structures found on some of the buildings coupled with overexposed images.

Results for 6 trajectories captured in the garden are presented in Figures 4 and 5. A zoomed in plot of one of the trajectories is shown in Fig. 7. As can be seen from the trajectory plots, EKF successfully rejects SLAM outliers and integrates all sensor measurements into locally smooth trajectories. However, it cannot cope smoothly with large offsets in the global position. Most of these cases can only be resolved by an automatic filter reset due to a large covariance estimate and a subsequent 'jump' to the corrected global pose reported by SLAM.

A quantitative comparison in Fig. 6 suggests that the EKF trajectory is more accurate than its SLAM input. Based on the trajectory plots, this improvement seems to be mostly caused by outlier rejection. The presence of larger global offsets in the SLAM poses wrt. GT however does not allow to evaluate the local accuracy of EKF (e.g., smoothness) well in quantitative terms.

Additionally, we run the filter on the more recent data from the Wageningen test garden (which was used in D3.2). We did not observe a noticeable improvement by using EKF. This is due to the fact that the SLAM system already provides smooth and stable results in this garden.

3 Arm Localization

A severe challenge faced during arm localization is that both the appearance and geometry of the scene changes drastically during cutting and trimming. The initial plan for the arm localization algorithm was thus to avoid this hard problem altogether. Rather than developing a camera-based solution, the idea was to use the joint angle measures provided by the Kinova arm together with the known lengths of the joints. According to the fact sheet for the arm, this should have provided us with an accurate localization result. However, we noticed that in practice the measurements provided by the arm do not accurately reflect the actual angles. This is due to the weight of the cutting tool and (partially) the cameras mounted on the arm. As such, we are currently working on computer vision-based approaches to arm localization.

In the following, we describe and evaluate two approaches that can currently be used for arm localization: The first is based on the Generalized Camera SLAM (GCSLAM) approach used for vehicle localization (and described in detail in Deliverable D3.2). GCSLAM is based on sparse keypoints and thus only provides a sparse 3D model of the scene that is unsuitable to plan the cutting motion. To be able to track the motion of the arm while obtaining a dense 3D model, we are also working on adapting DeepTAM [16], a deep learning-based approach recently developed by ALUF.

3.1 Generalized Camera SLAM (GCSLAM)

We perform an evaluation of the same SLAM algorithm that is used for vehicle localization for usage with the arm-mounted cameras. In contrast to the vehicle localization, we do not perform the map creation and localization in two separate steps, but use the SLAM algorithm online for both mapping and camera tracking. In addition, we do not attempt to perform appearance-based image matching for loop closure detection. Instead, we rely solely on the sequential matching, therefore using rather a visual odometry than full SLAM. During actual cutting, we can expect that the appearance of the bush changes significantly, which limits the value of a relocalization against older map parts. Although no cutting is performed during our experiments, we observed that the appearance based matching does not significantly improve the tracking accuracy, but can instead lead to much worse pose estimates due to wrong matches.

Localization accuracy Due to the lack of reliable ground truth poses, we only perform a qualitative analysis of the estimated poses and trajectories. The estimated trajectories for multiple datasets with the cameras moving around a single bush is shown in Fig. 8. We can observe that initialization and consequently the first few SLAM poses can be quite unstable. The first frames set aside, the SLAM trajectories are stable and smooth. As a reference, the trajectories computed from angle measurements of the robot arm are shown as well. As explained above, the angle measurements are not sufficiently accurate to localize the arm and cannot be used as ground truth. Instead, the comparison with these measurements serves the purpose of showing that there is no significant drift in the poses estimated by SLAM. Overall, we suspect that the SLAM poses are more accurate than the poses obtained by the measured angles. We will analyze the SLAM poses in more detail once accurate ground truth for the trajectories of the arm is available.

While the results above are obtained with only 1 camera pair, there are 6 cameras available on the arm, including 2 with wide angle lenses. We could not observe significant improvements when also using the 4 additional cameras in preliminary experiments. A likely explanation

for the similar performances is the large overlap in the cameras' field of view: Using more cameras does not significantly increases the effective field-of-view available to the localization algorithm.

Runtime In order to benefit from the increased accuracy compared to the joint angle measurements, the estimates also need to be available as fast as possible. Here, the SLAM system benefits from the lower number of cameras compared to vehicle localization, and the dropped requirement for relocalization. The processing times per frame for two trajectories (top row in Fig. 8) are given in Fig. 9. Using all 6 available cameras, the required runtime for each frame increases significantly. Fig. 10 shows the runtimes for the same trajectories as in Fig. 9, but using 6 cameras instead of 2. Obviously, these runtimes largely reduce the applicability for live localization.

Field of View In static scenes, we could not observe a benefit from a wider field of view, and even observed disadvantages from increasing the number of cameras. Yet, we expect a more robust performance when using wide angle lenses in scenarios where the scene geometry changes during the cutting operation. Fig. 11 shows the extracted features from a camera with standard lens and one with wide angle lens of the same frame. Clearly, the wide angle lens provides much more information about the background. As such, we expect this setup to yield more stable pose estimates during cutting. We plan further evaluations on sequences with cutting information once they become available.

3.2 DeepTAM

DeepTAM [16] is a keyframe-based dense camera tracking and depth map estimation system which uses neural networks. To achieve higher accuracy, we compute the depth using the stereo pair with DispNet [17]. The camera pose is estimated using the tracking network from DeepTAM. The tracking network contains a stack of encoder-decoder-based networks which implement a coarse-to-fine approach. We render the keyframe image and depth to a virtual frame given a pose guess. The tracking network takes the virtual keyframe and the current frame as inputs and track the relative pose incrementally between them. This incremental formulation simplifies the learning task and reduces the effects of dataset bias. Fig. 12 shows an overview of the tracking network architecture. For detailed description, please refer to our publication [16].

3.2.1 Evaluation

Generalization The most common problem in many learning-based methods is overfitting. The generalization ability is crucial, because there are no available training datasets for dense camera tracking in garden scenes. We took special care in the problem definition and the architecture design to make our model generalize better. Tab. 2 shows the results from a series of cross-validation experiments: Purely training on the synthetic dataset already allows us to outperform a classical baseline. With the help of the real datasets, we can achieve even better performance. The garden scene differs from the scenes that we have trained on and the camera instrinsics are also slightly different. The generalization ability allows our model to provide a reliable pose estimation to this unseen case.

Tracking performance Fig. 13 shows a qualitative reconstruction comparison with the estimated poses of DeepTAM and the poses of the robot arm. The reconstruction is simply

Sequence	RGB-D SLAM Kerl [18]	Ours (SUNCG [19])	Ours (SUNCG [19] + SUN3D [20])
fr1/360	0.119	0.072	**0.061**
fr1/desk	**0.030**	0.042	0.038
fr1/desk2	0.055	0.052	**0.051**
fr1/floor	0.090	0.103	**0.079**
fr1/plant	0.036	0.031	**0.027**
fr1/room	0.048	0.045	**0.044**
fr1/rpy	**0.043**	0.060	0.058
fr1/teddy	0.067	0.074	**0.062**
fr1/xzy	0.024	0.018	**0.016**
average	0.057	0.052	**0.048**

Table 2: Evaluation of our tracking on the RGB-D benchmark [21]. The values describe the translational RMSE in $[m/s]$. We compare the performance of our tracking network with different combination of training data against the RGB-D SLAM method of Kerl et al. [18]. Results of Kerl et al. [18] are taken from their paper. **Ours (SUNCG)** is the model that we train only with the synthetic dataset SUNCG [18], while for **Ours (SUNCG + SUN3D)** we use a combination of the synthetic dataset SUNCG [18] and the real dataset SUN3D [20].

implemented by combining the point clouds of all keyframe using their estimated poses. No extra fusion methods are applied. It can be seen that using the robot arm pose, the reconstructed bush is not a sphere and the ground has several layers. This is due to the inaccuracy of the joint angle measurements provided by the Kinova arm. The noisy reconstruction is too inaccurate for shape fitting and arm control.

Challenges Fig. 14 shows a challenging case where the camera is very close to the bush. Both DispNet [17] and DeepTAM [16] models are not well optimized for this case, which results in a sparse disparity estimation and unstable tracking. One solution is to use a wide angle camera, which has a larger field of view and can provide more information for the tracking. Another solution is to finetune the DispNet [17] and DeepTAM [16] to adapt to the challenging cases.

4 Conclusion & Outlook

In this report, we have described the vehicle and arm localization techniques that are currently used and developed as part of the TrimBot2020 project. We have shown how a generative descriptor learning approach based on fusing semantic image data and 3D can be used to enable long-term localization. In addition, we have described and evaluated two approaches for arm localization, one based on local features and one based on deep learning.

While the semantic localization approach achieves very promising results, it is currently not efficient enough to be integrated into the TrimBot2020 pipeline. Thus, defining more efficient network architectures could be an interesting direction for future work. Still, it is unlikely that this approach will be used on the TrimBot due to its limited computational resources.

For arm localization, natural next steps are to evaluate the localization approaches under trimming operations and to adapt them according to the results. We are currently also working

on obtaining ground truth arm trajectories for quantitative evaluations of the arm localization algorithms.

References

[1] Lowe, D.: Distinctive Image Features from Scale-Invariant Keypoints. IJCV **60** (2004)

[2] Schönberger, J.L., Pollefeys, M., Geiger, A., Sattler, T.: Semantic Visual Localization. In: CVPR. (2018)

[3] Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: Proc. International Conference on Learning Representations (ICLR). (2014)

[4] Carlevaris-Bianco, N., Ushani, A.K., Eustice, R.M.: University of Michigan North Campus long-term vision and lidar dataset. IJRR (2015)

[5] Geiger, A., Lenz, P., Urtasun, R.: Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2012)

[6] Dong, J., Soatto, S.: Domain-size pooling in local descriptors: Dsp-sift. In: CVPR. (2015)

[7] Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. BMVC (2004)

[8] Torii, A., Arandjelović, R., Sivic, J., Okutomi, M., Pajdla, T.: 24/7 place recognition by view synthesis. In: CVPR. (2015)

[9] Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (FPFH) for 3d registration. In: ICRA. (2009)

[10] Zeng, A., Song, S., Nießner, M., Fisher, M., Xiao, J., Funkhouser, T.: 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In: CVPR. (2017)

[11] Khoury, M., Zhou, Q.Y., Koltun, V.: Learning compact geometric features. In: ICCV. (2017)

[12] Kendall, A., Grimes, M., Cipolla, R.: Posenet: A convolutional network for real-time 6-dof camera relocalization. In: ICCV. (2015)

[13] Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of basic Engineering **82** (1960) 35–45

[14] Moore, T., Stouch, D.: A generalized extended kalman filter implementation for the robot operating system. In: Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13), Springer (2014)

[15] Lynen, S., Achtelik, M., Weiss, S., Chli, M., Siegwart, R.: A robust and modular multi-sensor fusion approach applied to mav navigation. In: Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS). (2013)

[16] Zhou, H., Ummenhofer, B., Brox, T.: DeepTAM: Deep Tracking and Mapping. In: European Conference on Computer Vision (ECCV). (2018)

[17] Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR). (2016) arXiv:1512.02134.

[18] Kerl, C., Sturm, J., Cremers, D.: Dense visual SLAM for RGB-D cameras. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. (2013) 2100–2106

[19] Song, S., Yu, F., Zeng, A., Chang, A.X., Savva, M., Funkhouser, T.: Semantic Scene Completion from a Single Depth Image. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017) 190–198

[20] Xiao, J., Owens, A., Torralba, A.: SUN3D: A Database of Big Spaces Reconstructed Using SfM and Object Labels. In: 2013 IEEE International Conference on Computer Vision (ICCV). (2013) 1625–1632

[21] Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. (2012) 573–580

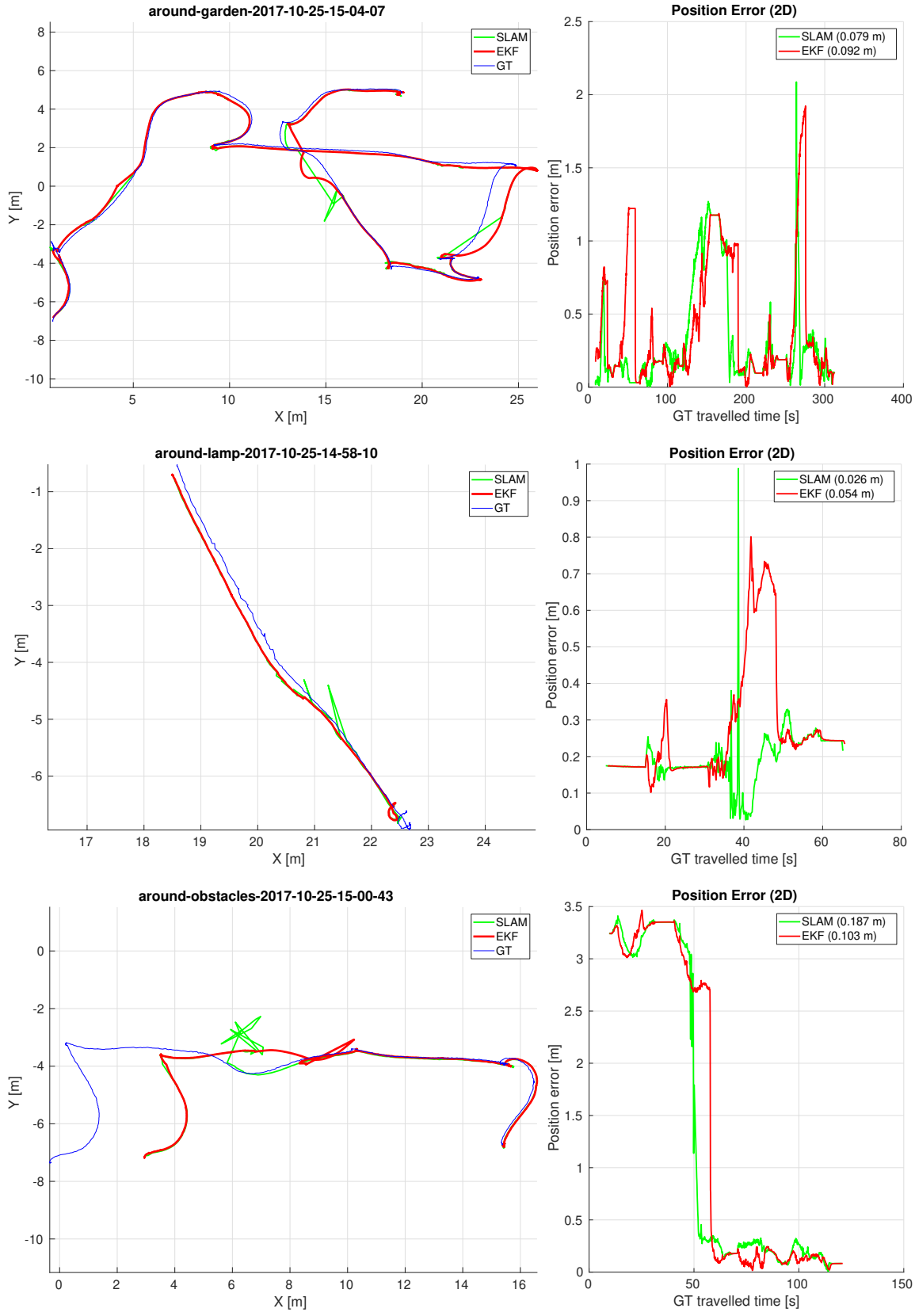

Figure 4: Comparison of the positions estimated by SLAM and EKF on trajectories 1–3 of the Renningen garden dataset. *Left:* trajectory map. *Right:* 2D position error wrt. corresponding GT position (mean absolute distance in brackets).

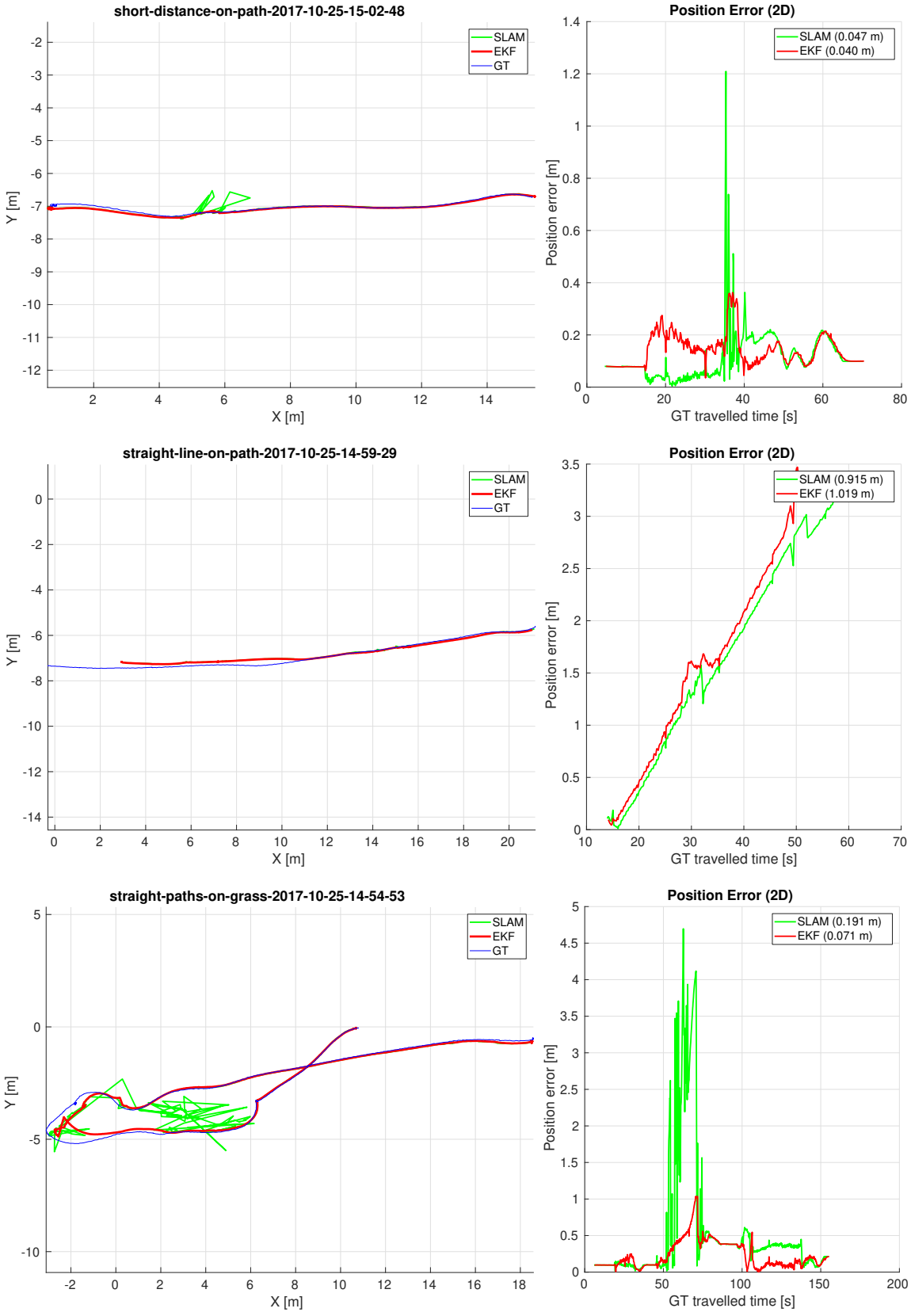

Figure 5: Comparison of the positions estimated by SLAM and EKF on trajectories 4–6 of the Renningen garden dataset. *Left:* trajectory map. *Right:* 2D position error wrt. corresponding GT position (mean absolute distance in brackets).

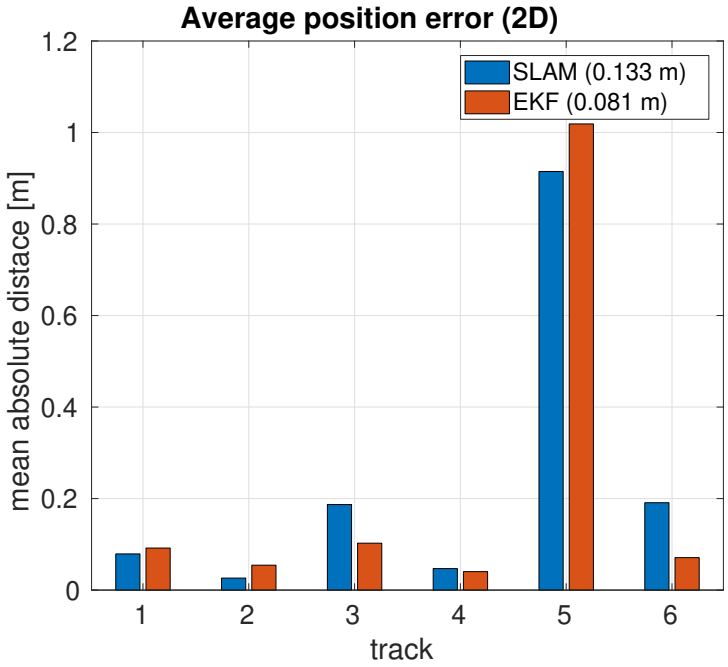

Figure 6: Quantitative comparison of the position accuracy obtained with SLAM and EKF on the different trajectories of the Renningen garden dataset. Mean absolute distances (in brackets) of all compared position estimates show that the accuracy was improved by 40% or 5 cm on average. See D7.5 for detailed discussion of presented SLAM results (the large error in track 5: `straight_line_on_path` comes from the accumulated drift).

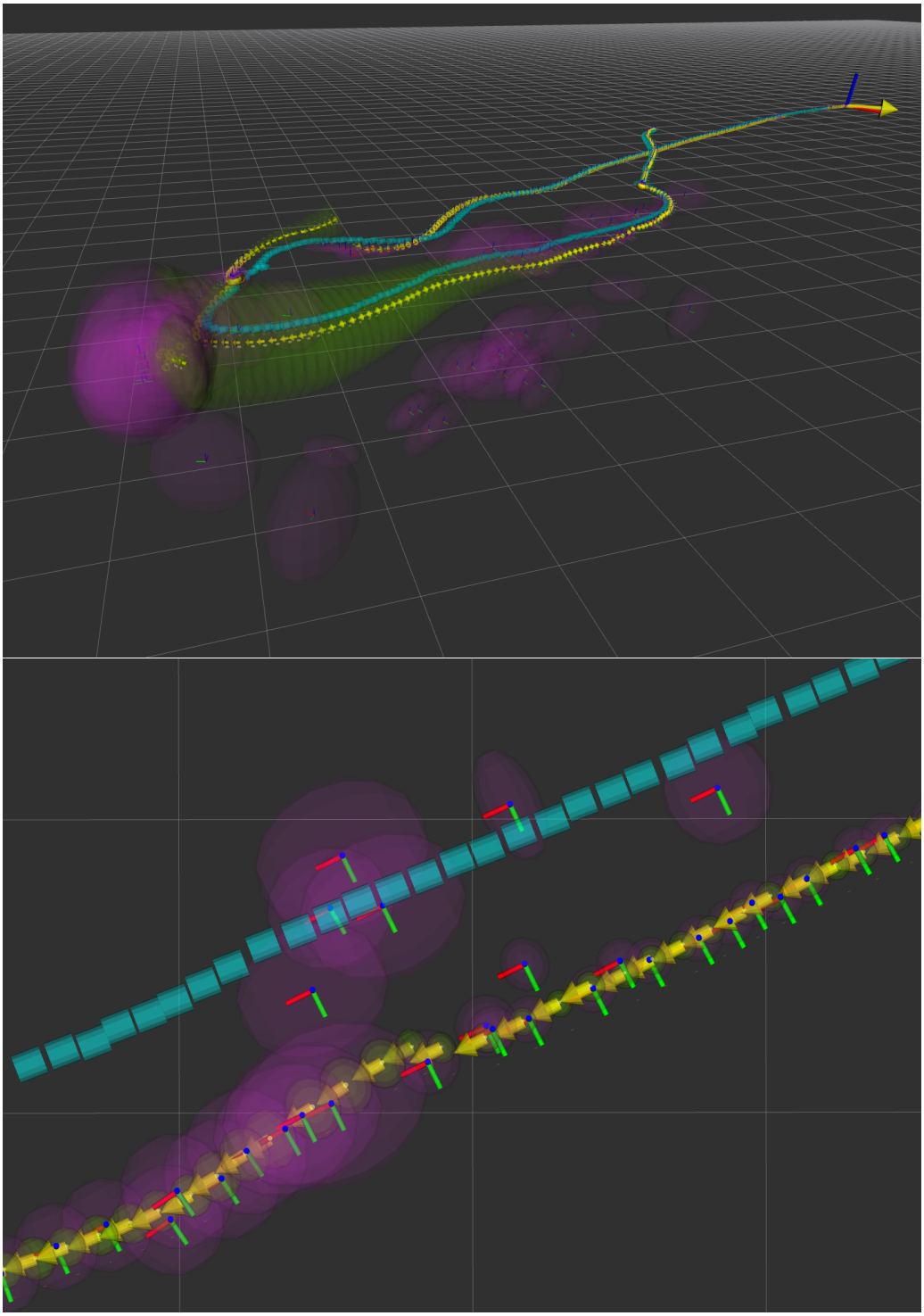

Figure 7: Visualization of trajectories with covariances as ellipsoids for `straight_paths_on_grass` track (top) and its detail (bottom). *Legend:* blue = GT, purple = SLAM, yellow = EKF, big arrow = final pose. *Top:* As the TrimBot moves from right to left (lower part), SLAM measurements below (purple) are rejected as outliers, causing EKF to integrate wheel odometry and IMU measurements, resulting in a gradual increase in covariance (yellow). *Bottom:* EKF rejects outliers and smoothly interpolates inliers at a constant rate.

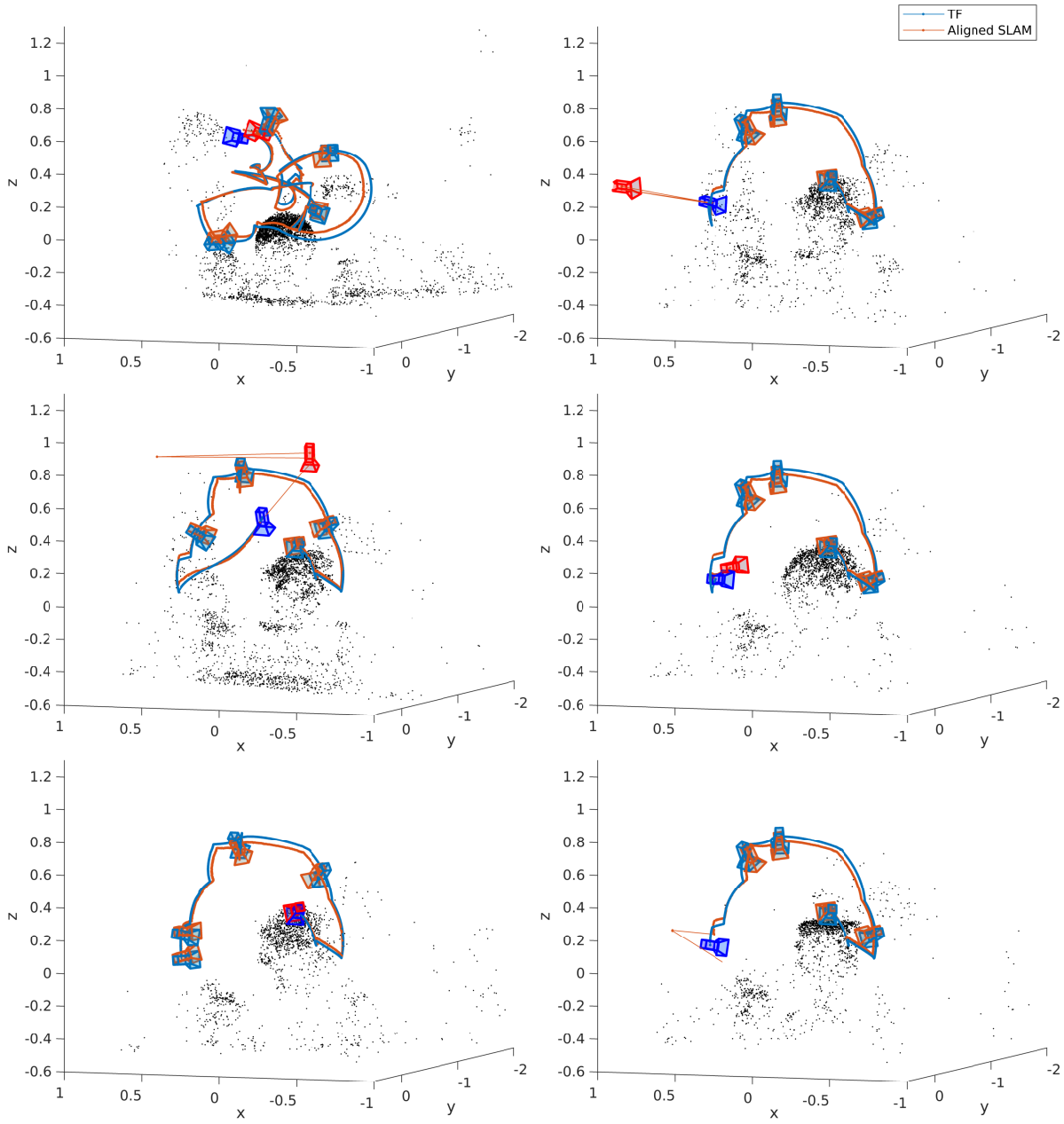

Figure 8: Trajectories estimated by SLAM compared to the estimations from joint angle measurements (TF). The trajectories are aligned by minimizing the distance between the trajectories at each timestamp. Camera orientations are shown for some intermediate timestamps. The first frame is highlighted.

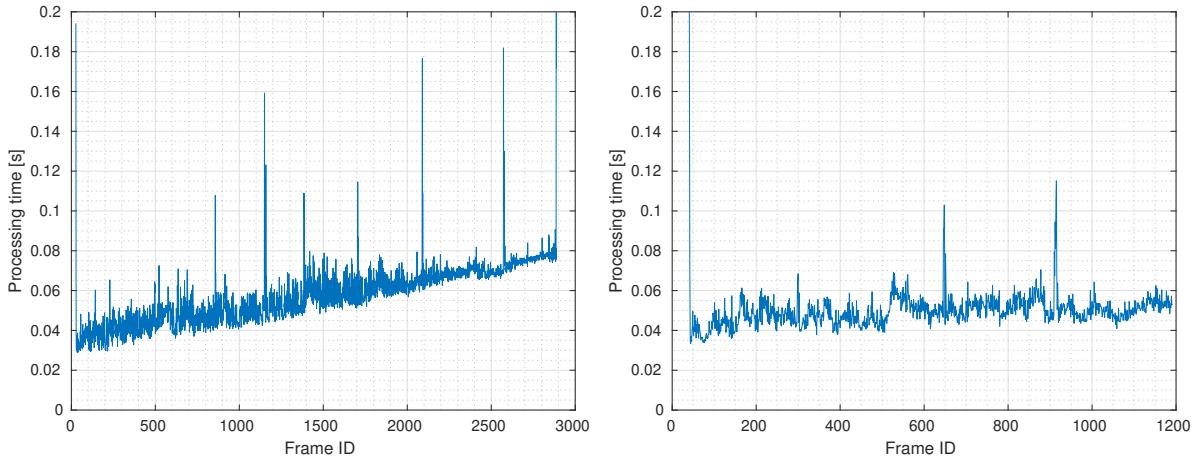

Figure 9: Processing times per frame for the two top row trajectories in Fig. 8. We can observe an increase in runtime for the long trajectory (left), but processing is still faster than the input frame rate of 9Hz. Two cameras were used in this experiment.

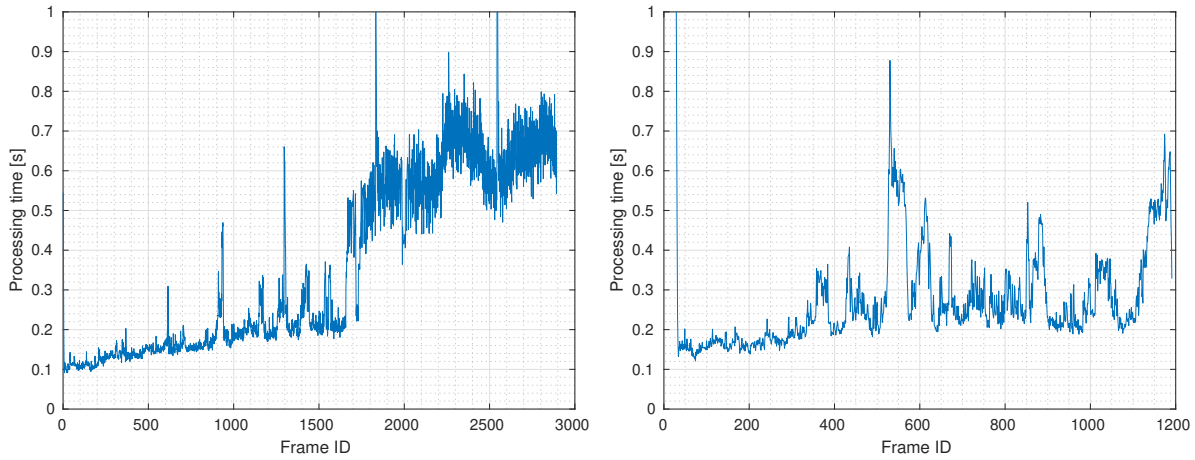

Figure 10: Runtimes with 6 cameras instead of 2. Processing is significantly slower than the input frame rate.

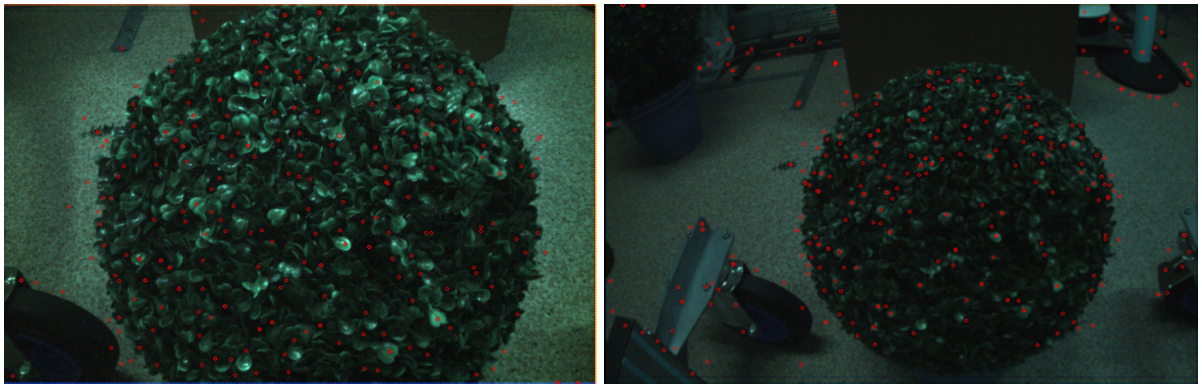

Figure 11: Example images from a camera with standard lens (left) and wide angle lens (right) of the same frame. With the wide angle lens, a lot more background information can be captured. Red dots indicate feature points.

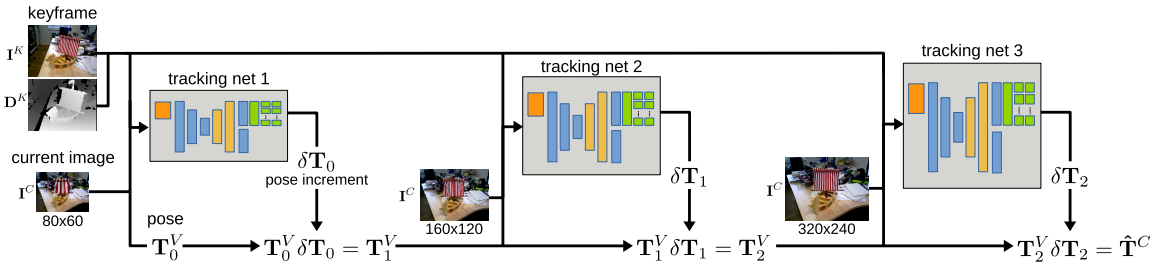

Figure 12: Overview of the tracking networks and the incremental pose estimation. We apply a coarse-to-fine approach to efficiently estimate the current camera pose. We train three tracking networks each specialized for a distinct resolution level. Each network computes a pose estimate $\delta\mathbf{T}_i$ with respect to a guess $\mathbf{T}_i^V$. The guess $\mathbf{T}_0^V$ is the camera pose from the previously tracked frame. Each of the tracking networks uses the latest pose guess to generate a virtual keyframe at the respective resolution level and thereby indirectly tracking the camera with respect to the original keyframe $(\mathbf{I}^K, \mathbf{D}^K)$. The final pose estimate $\hat{\mathbf{T}}^C$ is computed as the product of all incremental pose updates $\delta\mathbf{T}_i$.

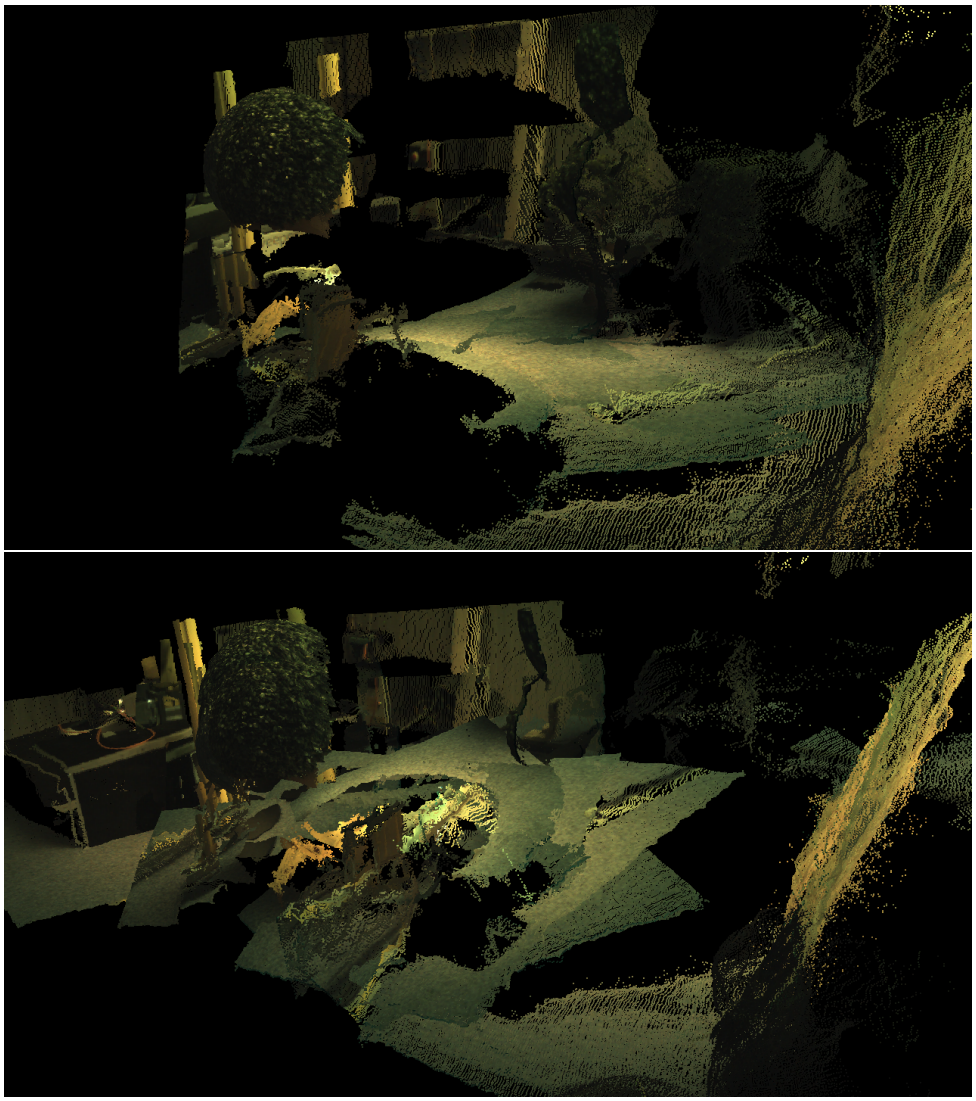

Figure 13: Qualitative reconstruction comparison using the estimated poses from DeepTAM [16] (**Top**) and the measured poses from the arm (**Bottom**).

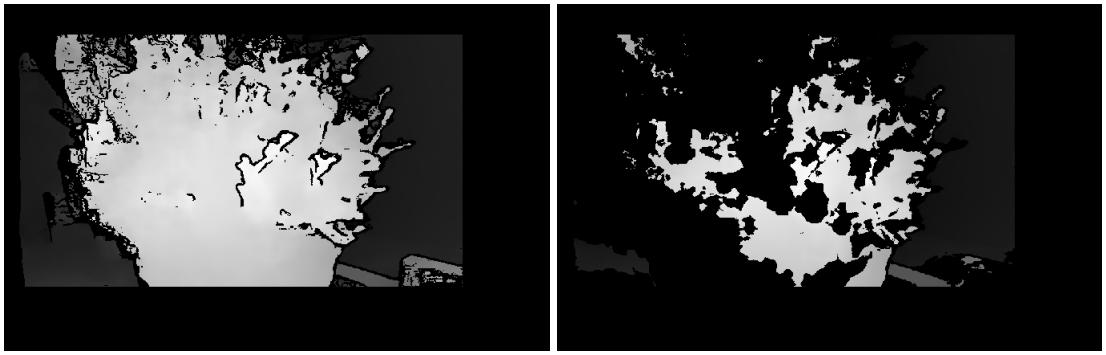

Figure 14: Disparity estimation in a challenging case where the camera is very close to the bush. **Left**: the original disparity estimation from DispNet [17]. **Right**: the disparity estimation from DispNet [17] with left-right-consistency check. We use a threshold of 2.5 pixels.

