Real-time Dense Surface Reconstruction for Aerial Manipulation

Real-time Dense Surface Reconstruction for Aerial Manipulation

Marco Karrer*, Mina Kamel**, Roland Siegwart** and Margarita Chli*

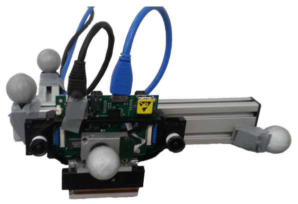
*Vision for Robotics Lab and **Autonomous Systems Lab, ETH Zurich, Switzerland

***Abstract*— With robotic systems reaching considerable maturity in basic self-localization and environment mapping, new research avenues open up pushing for interaction of a robot with its surroundings for added autonomy. However, the transition from traditionally sparse feature-based maps to dense and accurate scene-estimation imperative for realistic manipulation is not straightforward. Moreover, achieving this level of scene perception in real-time from a computationally constrained and highly shaky and agile platform, such as a small an Unmanned Aerial Vehicle (UAV) is perhaps the most challenging scenario for perception for manipulation. Drawing inspiration from otherwise computationally constraining Computer Vision techniques, we present a system combining visual, inertial and depth information to achieve dense, local scene reconstruction of high precision in real-time. Our evaluation testbed is formed using ground-truth not only in the pose of the sensor-suite, but also the scene reconstruction using a highly accurate laser scanner, offering unprecedented comparisons of scene estimation to ground-truth using real sensor data. Given the lack of any real, ground-truth datasets for environment reconstruction, our V4RL Dense Surface Reconstruction dataset is publicly available[1].**

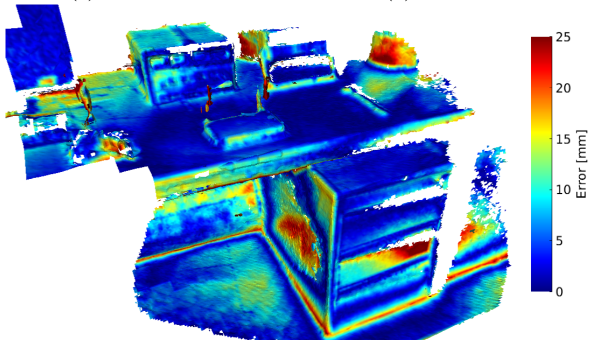
I. INTRODUCTION

For a robot to be able to interact with its environment, awareness of both its ego-motion as well as its workspace are necessary. With Simultaneous Localization And Mapping (SLAM) techniques opening up new horizons in robotic autonomy, we have witnessed a series of impressive breakthroughs to motion and environment estimation all the way from systems using range sensors on ground robots [1], [2] to real-time SLAM from a single camera [3]. It is due to this progress that vision-based flights with all processing and sensing onboard a small Unmanned Aerial Vehicle (UAV) were made possible [4]. Combining visual and inertial cues is now accepted as a powerful setup suitable for UAV navigation offering complementary sensor information at relatively low weight, power and computation resources, which are particularly limited onboard UAVs. Aiming for more robust and scalable solutions, Visual-Inertial (VI) SLAM has come a long way with systems such as [5] and [6] able to perform with unprecedented robustness, albeit still prone to drift and erroneous estimates due to common challenges, such as lighting changes and fast camera motion. With the increasing availability and affordability of UAVs as well as the acquired knowledge on the controls, active interaction of a UAV with its environment is increasingly attracting the interest of the community. With robot's interaction with its environment spanning a great spectrum from guiding physical contact across a set of pre-defined waypoints on a surface [7], [8] to grasping objects during flight [9], [10], the focus in such works is on the control aspect to ensure the integrity and stability of the vehicle during such tasks. However, such works typically rely on knowing the robot's pose and its workspace, for example using an external tracking system to provide the UAV's pose and attempt to interact only with objects of simple, predefined shapes, sometimes compensating for small errors using tactile feedback [7]. In this way, the big challenge of effective and timely robotic perception of its environment are circumvented, albeit limiting the applicability of these works to real scenarios outside the controlled laboratory environment, such as the industrial manipulation tasks envisioned in [11].

Following the demand for more realistic frameworks enabling robot autonomy, recent Robotics research has been turning towards denser scene representations than traditional feature-based SLAM, borrowing ideas from Computer Vision


(a) Sensor-Suite

(b) AscTec Neo[2]



(c) Local scene reconstruction with SLAM in the loop

Fig. 1: (a) The Sensor-Suite used for capturing visual, intertial and depth cues as well as ground truth poses. (b) UAV able to carry these sensors as well as processing all data in real-time. (c) An example local scene reconstruction obtained by the proposed method using a time-of-flight camera and color-coded according to the reconstruction error (mean error of $8mm$ in this frame, processed at $5.2ms$ per frame).

This research was supported by EC's Horizon 2020 Programme under grant agreement n. 644128 (AEROWORKS), the Swiss National Science Foundation (SNSF, Agreement no. PP00P2_157585) and the Swiss National Centre of Competence Research (NCCR) Robotics.

[1]http://www.v4rl.ethz.ch/research/datasets-code.html

[2]http://www.asctec.de

and Photogrammetry. A significant milestone in this direction was the emergence of the Microsoft Kinect camera, which paved the way to a variety affordable depth sensors able to provide dense depth images at high frame rates. Of particular interest are also the new generations of more compact and cheaper Time of Flight (ToF) cameras.

KinectFusion [12] pioneered real-time dense scene reconstruction using a Kinect camera proposing to maintain a discretized Signed Distance Function (SDF) to fit a surface to the scene. Aiming to address the lack of scalability of [12], [13] proposed a movable reconstruction volume, while [14] proposed the use of a discretized Octree scene representation. Putting SLAM in the loop of dense scene estimation, [15] proposed an RGB-D SLAM system built on dense image alignment and an Octree-based mapping of the underlying SDF. However, despite the visually appealing scene reconstructions that they produce, all aforementioned works make use of power-hungry GPUs to compensate for the otherwise unaffordable cost of computation. As the use of such computational power is prohibitive onboard low power and low payload platforms such as UAVs, most recent research focuses on bringing such techniques on the basis of affordable CPU processing. In this direction, [16] demonstrated real-time CPU-only capability based their earlier work [15], while single-camera CPU-only reconstructions have also made their debut [17], albeit with significantly less accurate and less robust frameworks.

In realistic manipulation tasks, for example to clean a surface from oxidation, it is imperative to have a timely, dense and accurate estimation of the robot's workspace. This is especially the case in aerial manipulation, where the base of the manipulator, instead of firmly mounted on a rigid structure, is attached on a highly agile and shaky UAV, highlighting the need for real-time and precise dense scene estimation before any manipulation task can be carried out successfully. With this challenge in mind, in this paper we propose a novel system, which estimates the pose of the sensor-suite in real-time using monocular-inertial SLAM and produces a dense, local scene reconstruction based on [16] using sensing cues from a depth sensor (Figure 1). We evaluate our system on a variety of challenging surfaces and camera motions with respect to scene ground truth obtained by millimetre-accurate laser scans from a Leica MS50[3] station. While several datasets exist in the literature, recording depth values from RGB-D sensors (e.g. [18]), to the best of our knowledge, there are no datasets offering ground truth for evaluating the scene reconstruction at this level of accuracy. The dataset used in this paper, consisting of data from our hand-held multi-sensor setup (Figure 1a), as well as millimetre-precise ground truth for both the pose of the sensor-suite using a Vicon[4] Tracking system and the scene using the Leica laser scanner is available online[1]. Accompanying this paper, a video is available summarizing our approach.

[3]http://leica-geosystems.com
[4]http://www.vicon.com

	VI sensor	RS sensor	ToF sensor
TECHNOLOGY	Monochrome global-shutter CMOS, IMU	RGB imaging, IR depth imaging	Laser (VCSEL) depth imaging
WEIGHT	$130g$	$35g$	$18g$
RANGE	-	$0.5-5m$ indoors, varies outdoors	$0.1-4m$
POWER	$5W$	$1-1.6W$	$300mW$
IMAGING RESOLUTION	480×752	RGB: 1920×1080, IR: 640×480	172×224

TABLE I: Specifications of the sensors used to produce the local scene reconstruction. While the VI is sufficient for pose estimation, we discuss the use of either the RS or the ToF for dense scene estimation. Note that the resolutions correspond to the maximal supported values.

II. METHOD

A. Sensor Setup

Since our framework is intended for the use with an aerial robot, besides accuracy, the weight and power consumption of the sensors are also important specifications. In this paper the Intel RealSense R200[5] (RS), and the novel Time-of-Flight (ToF) camera CamBoard pico flexx[6] from PMD are used for the depth perception. For the onboard pose estimation, the Visual-Inertial (VI) sensor [19] which consist of a stereo camera pair and a time-synchronized Inertial Measurement Unit (IMU) is used. In Table I specifications for the sensors used are shown. We use Vicon, an external visual 6 degree of freedom measurement system consisting of a constellation of multiple infrared cameras, tracking markers such as the ones in Figure 1a at 100 Hz at millimeter-precision. This is used to provide ground truth for the poses of the sensor-suite. The setup with the ToF mounted and the markers for the external tracker is shown in Figure 1a.
For clarity, we refer to the visual image captured by the RS as the "RS image", and equivalently for the ToF we refer to the amplitude image as the "ToF image". Note that both of these sensors also provide a corresponding "depth image".

B. Calibration

In a first step, the camera intrinsics as well as the rigid body transformation between the IMU and the camera are calibrated using the framework of [20], [21] and [22]. The intrinsic parameters of the depth sensors are used as provided by the manufacturer, which were verified using the camera calibration application from MATLAB's Computer Vision Toolbox.
The remaining calibration parameters, namely the rigid body transformations from the left VI-camera (C) to depth sensor (D), which can be either the ToF or the RS, and from the external tracker-body (B) to the camera are depicted in Figure 2. To compute these transformations, a set of images $j = 1, \ldots, J$ of a checkerboard calibration pattern (P) are taken, while simultaneously capturing the pose of the tracker-body ($\boldsymbol{T}_{BW}$) from the Vicon, along with the amplitude image for the ToF or the RGB image of the RS. Instead of just

[5]https://software.intel.com/en-us/realsense/r200camera
[6]http://pmdtec.com/picoflexx/

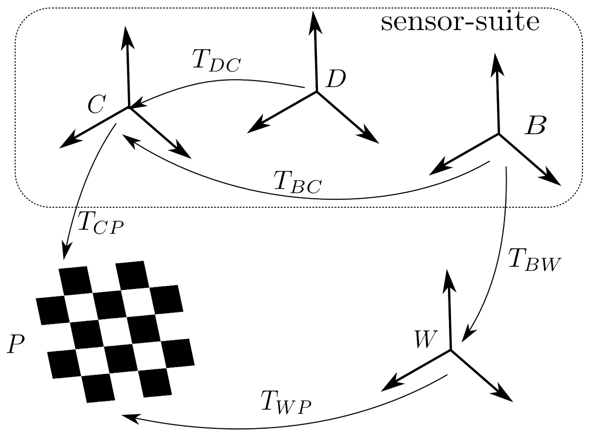


Fig. 2: In the calibration procedure, we aim to estimate the following rigid body transformations: $\boldsymbol{T}_{CP}$ between the VI-camera (C) and the calibration pattern (P), $\boldsymbol{T}_{DC}$ between the depth sensor (D) and camera, and $\boldsymbol{T}_{BC}$ between the tracker body and camera. The pose of the tracker body in the world frame W ($\boldsymbol{T}_{BW}$) is measured given by the Vicon system. Note that during the calibration the transformation between the world frame and the calibration pattern does not need to be estimated.

using the visual images of the calibration pattern, for every frame of the depth sensor (i.e. the amplitude image of the ToF or the RGB image of the RS) we also take into account its corresponding depth image. This enables the calibration procedure to account for systematic depth errors, for example coming from deviations of the actual emitted signal and the correlation function used to capture the depth image with the ToF [23]. For the depth correction of the ToF-camera, the parametric depth correction model presented in [24] was used. The model consists of a third order polynomial in the depth as well as two first order terms to correct for possible tilt of the sensing chip, expressed:

$$\lambda^* = a_0 + (1+a_1)\cdot\lambda + a_2\cdot x + a_3\cdot y + a_4\cdot\lambda^2 + a_5\cdot\lambda^3, \quad (1)$$

where λ corresponds to the measured ray-length, λ^* to the corrected ray length and $a_0, a_1, \ldots, a_5$ are the correction parameters, which have to be calibrated. The variables x and y correspond to the image coordinates. For the RS, we use a simplified model which only takes the constant offset and the linear term of Equation (1) into account, since it does not suffer from the effects caused by the oscillating signal typical in ToF depth imaging.

To estimate the calibration parameters of our system, a joint optimization problem is formulated by parameterizing the rigid body transform with respect to the variables $\boldsymbol{v}$. In order to have a minimal representation of the transformations, these are represented as elements of the Lie-algebra [25], for which

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{pmatrix} = \exp_{\mathfrak{se}(3)}(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \Re^6 \quad (2)$$

holds. Therefore, the optimization vector variables $\boldsymbol{v}$ is composed of

$$\boldsymbol{v} = \left(\boldsymbol{T}^1_{CP}, \ldots, \boldsymbol{T}^j_{CP}, \ldots, \boldsymbol{T}^J_{CP}, \boldsymbol{T}_{CD}, \boldsymbol{T}_{CB}, a_0, \ldots, a_5\right). \quad (3)$$

For clarity we denote the Lie-parametrization as the actual transformation matrices. The notation $\boldsymbol{T}^j_{CP}$ corresponds to the transformation $\boldsymbol{T}_{CP}$ for the j^{th} frame. We pose the optimization problem to include four different error terms, namely the reprojection errors in both the VI-images and the ToF or the RS images, the depth errors, and the pose errors of the tracker-body and the camera. For the first error term we detect the corner points of the calibration patter in each VI-image, as well as the ToF or RS images. The corresponding error term is composed as the error between the detected corner point $\boldsymbol{y}_D$ for the ToF or the RS image, $\boldsymbol{y}_C$ for the VI-image and the projection of the corresponding 3D point $\boldsymbol{y}_p$ expressed in the coordinate frame of P.

$$\boldsymbol{e}^j_{y_C} := \boldsymbol{y}^j_C - h_C\left(\boldsymbol{T}^j_{PC}\boldsymbol{y}_p\right) \quad (4)$$

$$\boldsymbol{e}^j_{y_D} := \boldsymbol{y}^j_D - h_D\left(\boldsymbol{T}_{CD}\boldsymbol{T}^j_{PC}\boldsymbol{y}_p\right) \quad (5)$$

The camera projection models $h_C(\cdot)$ and $h_D(\cdot)$ correspond to the VI-image and the ToF or the RS image respectively. The second error term included is built by the difference of the predicted and the corrected ray-length measured, coming from the depth image, as

$$e^{(u,v),j}_{\lambda} := \lambda^{(u,v),j} - \lambda^{*(u,v),j} \quad (6)$$

where $\lambda^{(u,v),j}$ is the predicted ray-length computed by intersecting the ray formed by the pixel coordinates (u, v) and the plane of the calibration pattern. Furthermore, $\lambda^{*(u,v),j}$ corresponds to the corrected measurement by applying Equation (1). For a more detailed view of the prediction, we refer the reader to [24]. The last error term included accounts for the error between the predicted camera pose ($\boldsymbol{T}_{CP}$) and the measured pose of the tracker-body ($\boldsymbol{T}_{BW}$). In order to eliminate the transformation between the calibration pattern and the Vicon origin ($\boldsymbol{T}_{PW}$), motion is necessary corresponding to recording two poses per error term [26]. Then the error term is expressed as an element of the Lie-group, as

$$\boldsymbol{e}^j_{\xi} := \log_{SE(3)}\left(\boldsymbol{A}\boldsymbol{T}_{BC}\left(\boldsymbol{T}_{BC}\boldsymbol{B}\right)^{-1}\right) \quad (7)$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are defined as

$$\boldsymbol{A} := \left(\boldsymbol{T}^j_{CP}\right)^{-1}\boldsymbol{T}^{j-1}_{CP}, \quad \boldsymbol{B} := \boldsymbol{T}^j_{BW}\left(\boldsymbol{T}^{j-1}_{BW}\right)^{-1} \quad (8)$$

The components of the objective function are defined as

$$G_{y_C} := \sum_{j=1}^{J} \boldsymbol{e}^{j^T}_{y_C}\boldsymbol{W}^{-1}_{y_C}\boldsymbol{e}^j_{y_C} \quad (9)$$

$$G_{y_D} := \sum_{j=1}^{J} \boldsymbol{e}^{j^T}_{y_D}\boldsymbol{W}^{-1}_{y_D}\boldsymbol{e}^j_{y_D} \quad (10)$$

$$G_{\lambda} := \sum_{j=1}^{J}\sum_{(u,v)\in A} \frac{1}{w_\lambda} e^2_\lambda \quad (11)$$

$$G_{\xi} := \sum_{j=2}^{J} \boldsymbol{e}^{j^T}_{\xi}\boldsymbol{W}^{-1}_{\xi}\boldsymbol{e}^j_{\xi} \quad (12)$$

where A denotes the pixels of the depth image corresponding to the area covered by the calibration pattern. The weight-matrices $\boldsymbol{W}_{y_C}, \boldsymbol{W}_{y_D}$ and $\boldsymbol{W}_{\xi}$ were chosen to be diagonal with the corresponding variance expected considering the calibration of the individual sensors or the manufacturer's specification on the uncertainty. The scalar weight w_λ corresponds to the noise variance of the depth sensor. The objective function is composed as:

$$G = G_{y_C} + G_{y_D} + G_\lambda + G_\xi \tag{13}$$

For the initialization of the camera poses the linear solution to the extrinsics problem is used. The transformation between the depth sensor and the camera is initialized using a least squares solution to the reprojection error on a subset of the calibration images. The initial guess for the transformation between the tracker-body and the camera is obtained using the algorithm of [26]. For the optimization of the objective function we use the Levenberg-Marquardt algorithm.

C. Surface Reconstruction

We employ the framework of [16] to estimate the surface and extend this to account for any priors on the uncertainty of the depth measurements as well as with an efficient method for performing a local reconstruction as elaborated in Section II-D. At the core of the approach of [16] lies the Octree structure, which at its leafs, called "bricks" (b), stores the value of the SDF in a 8^3 voxel volume. With the arrival of a new depth image, the corresponding brick for every pixel is looked up and put into a queue. The queue is then iterated over and for every voxel contained inside of the brick, its position $\boldsymbol{p}$ in the world frame transformed into the frame of the depth sensor $\boldsymbol{p}_D$. The measured point is computed using the linear inverse projection model $\tilde{h}_D^{-1}(\cdot)$ on the undistorted depth image and the depth measurement $Z(u,v)$ at pixel coordinates (u,v), as

$$\boldsymbol{p}_{obs} = \tilde{h}_D^{-1}\left(u, v, Z(u,v)\right) \tag{14}$$

Due to the use of a truncated version of the SDF, defined as

$$\Delta_D = \max\left\{\min\left\{\Phi, |\boldsymbol{p}_D - \boldsymbol{p}_{obs}|\right\}, -\Phi\right\} \tag{15}$$

where Φ is the truncation threshold and the multiscale approach, the number of bricks queued per frame are limited. The update of the stored distance values $D(\boldsymbol{p},t)$ at the voxel position $\boldsymbol{p}$ at time t is performed as a weighted running average

$$D(\boldsymbol{p},t) = \frac{D(\boldsymbol{p},t-1)W(\boldsymbol{p},t-1) + \Delta_D w(\Delta_D)}{w(\Delta_D) + W(\boldsymbol{p},t-1)} \tag{16}$$

Where $W(\boldsymbol{p},t)$ corresponds to the accumulated weight at position $\boldsymbol{p}$ and time t. For the weight increment $w(\Delta_D)$ different weighting schemes can be used as presented in [27]. In [16] $w(\Delta_D)$ is constant for areas in front of the surface and decreases linearly until zero behind the surface, i.e.

$$w(\Delta_D) = \begin{cases} 1 & \text{, if} \quad \Delta_D < \delta \\ \frac{\Phi - \Delta_D}{\Phi - \delta} & \text{, if} \quad \Delta_D \geq \delta \quad \text{and} \quad \Delta_D \leq \Phi \\ 0 & \text{, if} \quad \Delta_D > \Phi \end{cases} \tag{17}$$

where δ can be seen as an allowed penetration depth of the measurement. This weighting scheme solely relies on the geometric distance to the measurement, but does not take any additional information into account. A simple noise estimate for example for a ToF camera can be obtained as described by [28]. Since the ToF camera used provides information about the noise level of each pixels, we propose a weighting scheme to incorporate these in the weighting function using it as a scaling factor. Assuming a noise value $\sigma(u,v)$ at pixel coordinates (u,v), we compute a scaling factor according to

$$f_\sigma = \begin{cases} 1 & \text{, if} \quad \sigma(u,v) \leq \sigma_{min} \\ \frac{\sigma_{min}}{\sigma(u,v)} & \text{, if} \quad \sigma_{min} < \sigma(u,v), \end{cases} \tag{18}$$

where σ_{min} is a parameter that can be chosen according to the expected noise of the sensor. In this way, we aim to take the sensor's estimated uncertainty into account when weighting the different incoming votes of the voxels. This results to more informed estimation of the surface, providing robustness to common bottlenecks, for example too oblique angles of incidence. The adapted weighting scheme is defined as:

$$\tilde{w}(\Delta_D) = f_\sigma \cdot w(\Delta_D) \tag{19}$$

This simple scheme allows to incorporate sensor-specific uncertainty measurement, which in case of our ToF camera is available anyway, without degrading the computational efficiency of the estimation. For the RS, we use the uniform weighting scheme according to Equation (17).
The SDF representation has a large memory footprint, which e.g. limits cooperative interactions between multiple agents due to limited bandwidth. Therefore, and also for visualization purposes, the algorithm of [16] keeps track of the updated bricks and performs a re-meshing on those areas using an adapted version of a marching cubes algorithm in order to account for the multiscale approach. For a detailed explanation of the meshing algorithm, we refer to [16].

D. Surface Reconstruction using SLAM Poses

In order to use the reconstruction algorithm with SLAM poses, typically subject to drift, we implemented a scheme which is able to maintain a locally accurate scene estimate. We implement this by introducing a visibility constraint in the sense that we only keep the portion of the reconstruction, which received measurements within the time horizon t_h and discard the rest. This is done by keeping track of the time-stamp of updated bricks and maintaining the local scene reconstruction within t_h with respect to the current camera pose. We denote $\mathcal{B}_s$ as the set of bricks used for the surface estimation at the current time and $\mathcal{B}_o$ as the bricks to be updated. The notation $t(b_i)$ corresponds to the latest time-stamp that brick b_i was updated. The actual voxels grouped inside the brick b_i are indicated by $\boldsymbol{p}_j$, which corresponds to the voxel center coordinates. The update procedure is shown in Algorithm 1. The time horizon t_h influences the behavior of the algorithm. For larger t_h during exploratory motion, drift of the SLAM system result in larger reconstruction errors and higher computational cost as more bricks need to be tracked, i.e. the size of $\mathcal{B}_s$ increases. On the other hand, if the chosen t_h is too small, even quick deviations from the current viewpoint, e.g. by wind disturbances, can result in loss of the previous reconstruction. The reconstruction

Algorithm 1 Time Window for Local Reconstruction

```
$\mathcal{B}_o :=$ all bricks observed from the current pose
$t \leftarrow$ current time-stamp
for all $b_i \in \mathcal{B}_o$ do
    if $b_i \notin \mathcal{B}_s$ then
        add $b_i$ to $\mathcal{B}_s$
for all $b_i \in \mathcal{B}_s$ do
    if $b_i \in \mathcal{B}_o$ then
        $t(b_i) \leftarrow t$
        for all $\boldsymbol{p}_j \in b_i$ do
            update $W(\boldsymbol{p}_j, t)$
            update $D(\boldsymbol{p}_j, t)$
    else if $(t - t(b_i)) > t_h$ then
        remove $b_i$ from $\mathcal{B}_s$
$\mathcal{B}_o \leftarrow \emptyset$
```

algorithm itself is agnostic to the SLAM technique used. For the pose estimation in our system, we use the framework proposed by [29], which uses the sensor readings of both the IMU as well as the camera input streaming from the VI-sensor. The system is based on sparse features on a set of keyframes as well as the incorporation of IMU measurements into a local graph. The window of keyframes is kept bounded by marginalizing out old keyframes. This allows the algorithm to run in real time on a CPU, while maintaining an accurate pose estimate. However, due to the local keyframe approach the system is still prone to global drift. The framework can be run using multiple cameras or as a monocular system. Motivated by the lower computational complexity, while maintaining a comparable performance, here we use the monocular version of the algorithm for state estimation.

III. EXPERIMENTAL RESULTS

A. *V4RL Dense Surface Reconstruction Dataset*

In order to evaluate the performance of the system in terms of accuracy, a dataset consisting of the VI data, as well as the data of the depth sensor (RS or ToF) and the pose of the tracker-body from the Vicon system were recorded. Complementary, the observed scene was scanned using a Leica MS50 laser scanner from multiple viewpoints in order to obtain ground truth for the scene, against which we can compare the estimated reconstruction. This dataset, which includes scene and poses' ground truth for the first time, is publicly available[1].

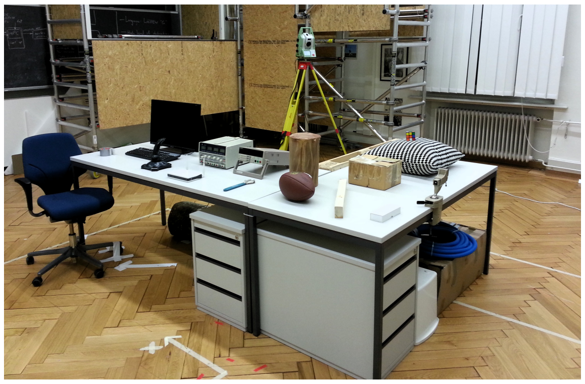


Fig. 3: A view of the Desk scene used for the evaluation together with the Leica MS50 station used to obtain the ground truth scene reconstruction.

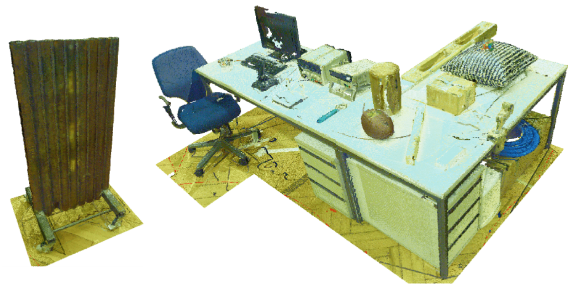


Fig. 4: The registered laser scans of mm-precision used as scene ground truth, left: Pipe structure, right: Desk scene

We choose two different scenes to evaluate the system; one generic desk sequence containing objects of different texture and material (e.g. affecting their reflectance properties) as shown in Figure 3, as well as a more industrial object consisting of a Pipe structure typical in an industrial inspection scenario (ground truth reconstruction in Figure 4). The recordings were made using a hand-held setup in order to ease testing different types of motions (i.e. of different speed, shaky, different viewpoints) as well as to avoid changes of the scene setup due to the airstream of a UAV.

The scene ground truth was captured by 8 separate scans for the Desk scene with a total number of approximately 10M points and 6 scans for the Pipe structure with 2.5M points in total. The scans were pre-aligned by localizing the Leica station with respect to a set of known markers in the room. The final alignment was performed using the Iterative Closest Point (ICP) algorithm on adjacent scans, incrementally building the ground truth point cloud. Outlier filtering using radius search was applied on the obtained point clouds, while points with low neighborhood support were also removed. The final ground truth point clouds for both scenes are shown in Figure 4.

The properties of the sequences used are listed in Table II. Note that the trajectories using the ToF and the RS are similar in each case but cannot be identical due to the hand-held setup.

B. *Reconstruction Accuracy (Global)*

To quantitatively evaluate the reconstruction accuracy isolating it from any pose errors, the scenes are reconstructed

Scene Type	Name		Average linear velocity $[m/s]$	Average rotation velocity $[°/s]$	Trajectory length $[m]$
Desk	d1	ToF	0.2	13.6	25
	d2		0.3	29.6	40
	d3		0.2	12.9	17
	d4		0.3	25.1	24
	d1	RS	0.2	13.2	25
	d2		0.3	29.5	43
	d3		0.2	12.7	14
	d4		0.3	27.5	22
Pipes	p1	ToF	0.3	17.1	22
	p2		0.3	24.2	20
	p3		0.1	6.4	6
	p4		0.2	25.8	11
	p1	RS	0.3	16.2	24
	p2		0.3	26.3	20
	p3		0.1	7.7	5
	p4		0.3	23.9	10

TABLE II: Properties and naming convention of the V4RL Dense Surface Reconstruction dataset.

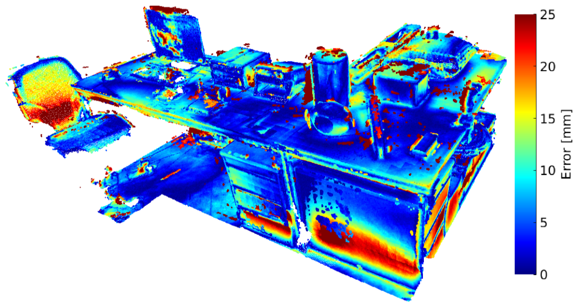
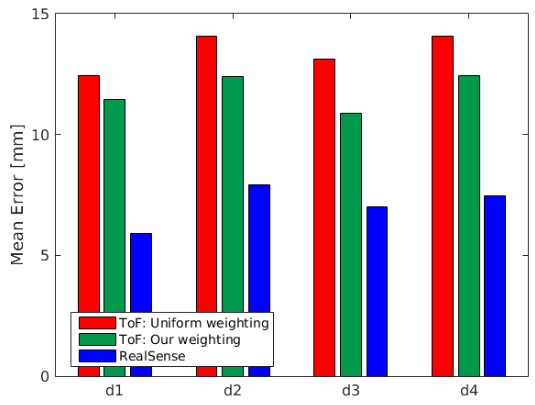
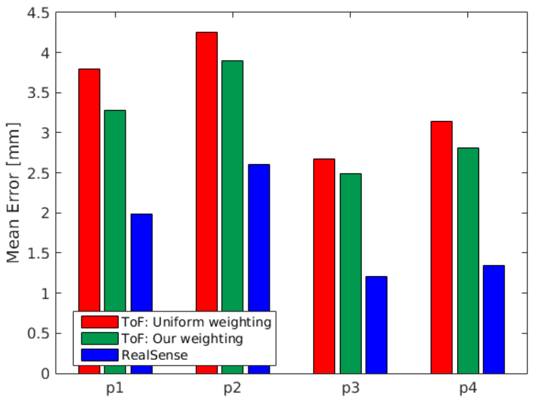


Fig. 5: Resulting reconstruction error using ground truth poses along with the ToF depth measurements on the Desk scene, while applying the proposed weighting scheme.



(a) Desk Scene



(b) Pipe Structure

Fig. 6: Average reconstruction error using ground truth poses. Note: the labels $d1 - d4$ and $p1 - p4$ correspond to the sequences in the dataset defined in Table II

using the ground truth poses from the Vicon system. The reconstruction is performed by fusing the full data sequence in the system for every trajectory. The voxel resolution at the finest level is chosen to have a side length of $6mm$. The transformation between the world frame (of the Vicon system) and the origin of the scene ground truth is only known approximately, since the world frame origin of the Vicon system can only be selected manually. Therefore, for every comparison we perform an ICP alignment of the estimated reconstruction to the ground truth. Although the obtained reconstruction is represented as a mesh, we consider its vertex points for the alignment. For the evaluation process, we compute the distance of every vertex of the reconstruction to its Nearest Neigbhor (NN) in the ground truth point cloud. In order to correct for regions, where no ground truth data is available (due to occlusions, shiny surfaces, etc.), we do not consider vertices whose NN is further away than a certain maximal distance d_{max}. The distance d_{max} was chosen to be 50mm for the Desk scene and 30mm for the Pipe structure, respectively. In Figure 5, an example of a reconstruction color-coded with respect to the NN-distance is shown. The NN-based errors are recorded for every sequence of the two scenes, using both the RS and the ToF. For the ToF camera, both the standard weighting scheme of [16] as well as our proposed weighting system (according to Section II-C) are evaluated. The reconstruction accuracies achieved in each case are summarized in Figure 6. On both the Desk scene as well as the Pipe structure, the RS achieves higher accuracy compared to the ToF. This is caused by the higher resolution $(2\times)$ of and the higher frame rate $(3\times)$ of the RS, which results in a higher information density. For the ToF our proposed weighting scheme improves the accuracy on all sequences on average by 10% up to a maximal improvement of 17%. The error level on the Desk scene is higher which is caused not only by the larger size of the scene, but also due to the different materials and larger variety in the angles of incidence. Furthermore, in the sequences of the Pipe structure all areas of the object are observed from a similar number of views and viewpoints, whereas for the Desk scene some regions are only viewed quickly, while others are observed more thoroughly.

C. Local Reconstruction with SLAM in the loop

To evaluate the estimated reconstruction when using SLAM for the pose estimation, we applied the time window discussed in Section II-D resulting to a local estimation of the scene. The time window t_h for this procedure was set the constant value of $3s$, which gives some tolerance to shaky motions of the camera when observing an area, while still keeping the region of the estimated reconstruction small enough to avoid significant motion drift. Instead of the global

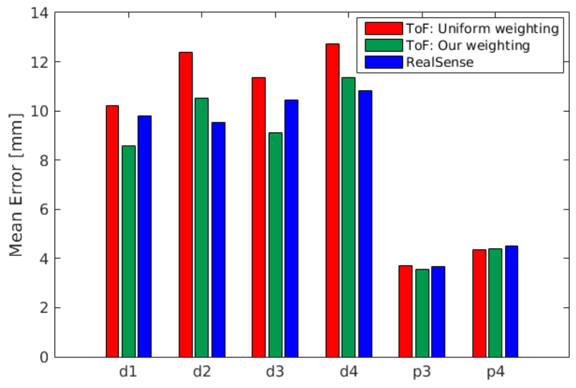


Fig. 7: Average errors of the local reconstructions using poses from VI-SLAM [29]. Note that for the Pipe structure only sequences $p3$ and $p4$ are used, since for the sequences $p1, p2$ there was a significant yaw drift in the SLAM poses when turning around the structure.

Sequence	ToF Uniform Weighting		ToF Our Weighting		RS	
	Fusion Time per Frame [ms]					
	Global	Local	Global	Local	Global	Local
d1	4.2	4.6	4.5	4.9	11.0	12.4
d2	4.6	5.1	5.0	5.5	10.8	12.3
d3	4.3	4.7	4.6	5.0	10.0	11.2
d4	4.4	4.9	4.7	5.2	10.0	11.4
p3	2.2	2.3	2.3	2.4	6.0	6.4
p4	2.1	2.2	2.2	2.3	6.6	7.2
Average	3.6	4.0	3.9	4.2	9.1	10.1

TABLE III: Average time per frame for fusing a new depth image in the model. The columns labeled "Global" correspond to the timings recorded when using ground truth poses with full scene reconstruction, "Local" corresponds to the timings when using SLAM for the pose estimation along with a $3s$ time window for the reconstruction. Despite the slightly increased time necessary for the "Local" reconstruction, our method is always real-time.

model, the most recently estimated local reconstructions were stored at a frequency of about 1 Hz. We aligned each of these local reconstructions to the corresponding scene ground truth using ICP and used the same error metric as for the global reconstruction. The corresponding mean error values are shown in Figure 7.

Surprisingly, the average error for the ToF on the Desk scene is lower than when using the ground truth poses (Section III-B). This is due to the fact that the impact on the overall error coming from poorly reconstructed areas (e.g. due to oblique angles of incidence, few views) is diminished when considering the average error of local reconstructions. Using the time horizon scheme, there is significant overlap in the regions considered at consecutive reconstructions, therefore these areas are bound to have better accuracy and can overall reduce the average reconstruction error.

When reconstructing the scene locally using RS cues, the average reconstruction error is increased and the accuracy disadvantage of ToF-based global reconstruction is diminished. The explanation for this is multi-fold; firstly, because the variability of viewpoints during local reconstruction is limited, any internal calibration errors of the RS become more evident (i.e. resulting to erroneous depth values at contour edges). Moreover, although the maximum range of the ToF and the RS (Table I) appear similar in their specification, in practice we observed that the RS usually perceives more distant measurements increasing the sensitivity of the acquired data (and thus, the reconstruction) to angular pose errors.

D. Evaluation of Computational Performance

A thorough investigation of the computational performance of the reconstruction and meshing algorithm was performed by [16]. Hence, we focused our investigation on the comparison between the ToF and the RS as well as on the difference between our weighting scheme and the standard implementation. Furthermore, we evaluate the computational overhead induced by applying the time window based Local reconstruction with SLAM in the loop, versus the full Global reconstruction with Vicon poses. We conducted the experiments on the sequences used in Section III-C, while setting the voxel size at the finest level to $8mm$. The timings were measured on a Intel Core i7-4710MQ CPU running at 2.5 GHz and are presented in Table III. The computational overhead of our proposed weighting scheme is minimal, on average about $0.4ms$, while reducing the reconstruction error on average by 10%. The difference in the update times between the RS and the ToF stems from the higher resolution of the RS compared to the ToF. Using the local reconstruction over the time horizon t_h increases the computational complexity by $\mathcal{O}(n)$, with n being the number of bricks tracked within the time horizon t_h. Table IV records the average number of bricks per frame tracked during the reconstruction as well as the percentage of these which get updated (per frame). Generally, faster motion or finer the voxel resolution (i.e. more detailed reconstruction) results the higher number of tracked bricks, which influences directly the execution time per update step. The RS requires on average 3 times more bricks tracked compared to the ToF due to its higher resolution as well as its larger measurement range. More execution time is consumed on average to achieve the Local reconstruction, but this is still real-time for the ToF ($< 6ms$) and the RS ($< 13ms$).

In manipulation tasks, we cannot afford to compromise the reconstruction accuracy as this can be catastrophic, especially in the case of UAV manipulation. As a result, in order to ensure sufficient quality of the reconstruction even in the presence of inevitable global drift accumulating in the SLAM pose estimation, a small overhead in execution time is acceptable. In the case of the RS Local reconstruction, this overhead becomes more evident due to a larger number of bricks to track. Overall, the RS outperforms the ToF in terms of accuracy on the global scale (Section III-B), but in terms of local accuracy (Section III-C) both the RS and the ToF perform similarly. As a result, considering that ToF offers a significant advantage in execution time, it poses a better choice for a setup with limited computational resources, such as onboard a UAV.

Sequence	Number of Bricks Tracked		Updated Bricks [%]	
	ToF	RS	ToF	RS
d1	1544	5182	37.4	17.8
d2	2468	6636	28.2	13.1
d3	1658	4867	37.2	15.9
d4	2106	5920	29.3	13.8
p3	543	1843	45.3	16.0
p4	733	2875	30.6	13.8
Average	1509	4554	34.7	15.1

TABLE IV: Average number of bricks tracked using the time window based Local reconstruction, and the percentage of bricks updated related to the number of tracked bricks.

IV. CONCLUSIONS AND FUTURE WORK

In this paper we present a system capable of accurately reconstructing a local scene in real-time, suitable for manipulation tasks even from a highly agile platform, such as a UAV. Fusing depth cues at frame rate, while estimating the pose of the sensor-suite using visual-inertial SLAM, our experimental evaluation on a variety of challenging scenarios reveals the high fidelity of the system achieving reconstruction accuracy of the order of $10mm$ on average.

A thorough evaluation of the proposed approach was presented assessing the accuracy of the obtained 3D reconstruction both on a global scale using ground truth poses and ground truth scene reconstruction, as well as for local reconstructions using poses obtained by a nominal visual-inertial SLAM system. As no such testbed (with real sensing data and scene ground truth) exists in the literature, we release our dataset consisting of visual, inertial and depth data from a time-of-flight and an RGBD camera, as well as pose and scene ground truth of millimeter precision. Future work includes employing this reconstruction framework to perform simple manipulation tasks from a UAV, as well as research into UAV path planning for viewpoints promising more accurate reconstructions.

REFERENCES

[1] J. Leonard and H. Durrant-Whyte, *Directed Sonar Sensing for Mobile Robot Navigation*. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[2] J. Castellanos, J. Tardós, and G. Schmidt, “Buildiong a Global Map of the Environment of a Mobile Robot: The Importance of Correlations,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[3] A. J. Davison, N. D. Molton, I. Reid, and O. Stasse, “MonoSLAM: Real-time single camera SLAM,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 29, no. 6, pp. 1052–1067, 2007.

[4] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, “Monocular Vision for Long-term MAV Navigation: A Compendium,” *Journal of Field Robotics (JFR)*, vol. 30, pp. 803–831, 2013.

[5] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “ROVIO: Robust Visual Inertial Odometry Using a Direct EKF-Based Approach,” in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2015.

[6] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, and R. Siegwart, “Keyframe-based Visual-Inertial SLAM using Nonlinear Optimization,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[7] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart, “Hybrid Predictive Control for Aerial Robotic Physical Interaction towards Inspection Operations,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[8] L. Marconi, R. Naldi, A. Torre, J. Nikolic, C. Huerzeler, G. Caprari, E. Zwicker, B. Siciliano, V. Lippiello, R. Carloni, and S. Stramigioli, “Aerial Service Robots: an Overview of the AIRobots Activity,” in *International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012.

[9] P. Pounds, D. Bersak, and A. Dollar, “Grasping From the Air: Hovering Capture and Load Stability,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[10] S. Kim, S. Choi, and H. J. Kim, “Aerial Manipulation Using a Quadrotor with a Two DOF Robotic Arm,” in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[11] “AEROWORKS: Collaborative Aerial Workers,” url http://www.aeroworks2020.eu, 2015.

[12] R. Newcombe, S. Izardi, O. Hiliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, “Kinect-Fusion: Real-time dense surface mapping and tracking,” in *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011.

[13] H. Roth and M. Vona, “Moving Volume KinectFusion,” in *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

[14] M. Zeng, F. Zhao, J. Zheng, and X. Liu, “A Memory-efficient Kinectfusion Using Octree,” in *Proceedings of the First International Conference on Computational Visual Media*, 2012.

[15] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers, “Large-Scale Multi-Resolution Surface Reconstruction from RGB-D Sequences,” in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.

[16] F. Steinbruecker, J. Sturm, and D. Cremers, “Volumetric 3D Mapping in Real-Time on a CPU,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[17] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct Monocular SLAM,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[18] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A Benchmark for the Evaluation of RGB-D SLAM Systems,” in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2012.

[19] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. Furgale, and R. Siegwart, “A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[20] J. Maye, P. Furgale, and R. Siegwart, “Self-supervised Calibration for Robotic Systems,” in *Inteligent Vehicles Symposium (IVS)*, 2013.

[21] P. Furgale, J. Rehder, and R. Siegwart, “Unified Temporal and Spatial Calibration for Multi-Sensor Systems,” in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[22] P. Furgale, T. D. Barfoot, and G. Sibley, “Continuous-Time Batch Estimation Using Temporal Basis Functions,” in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[23] D. Lefloch, R. Nair, F. Lenzen, H. Schaefer, L. Streeter, M. J. Cree, R. Koch, and A. Kolb, *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer Berlin Heidelberg, September 2013, vol. 8200, ch. Technical Foundation and Calibration Methods for Time-of-Flight Cameras, pp. 3–24.

[24] I. Schiller, C. Beder, and R. Koch, “Calibration of a PMD-Camera Using a Planar Calibration Pattern Together With a Multi-Camera Setup,” in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVII, Part B3a. ISPRS Congress, 2008, pp. 297–302.

[25] E. Eade, “Lie Groups for Computer Vision,” http://ethaneade.com/lie_groups.pdf, accessed: 2016-02-19.

[26] K. Daniilidis, “Hand-Eye Calibration Using Dual Quaternions,” *International Journal of Robotics Research*, vol. 18, pp. 286–298, 1998.

[27] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers, “Real-Time Camera Tracking and 3D Reconstruction using Signed Distance Functions,” in *Proceedings of Robotics: Science and Systems (RSS)*, 2013.

[28] L. Li, “Time-of-Flight Camera - An Introduction,” White Paper, Texas Instruments, May 2014.

[29] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual–inertial odometry using nonlinear optimization,” in *The Internationa Journal of Robotics Research*, 2014.