EVALUATION OF THE QUALITY OF REAL-TIME MAPPING WITH CRANE CAMERAS AND VISUAL SLAM ALGORITHMS

L. Joachim[1,*], W. Zhang[1], N. Haala[1], U. Soergel[1]

[1] Institute for Photogrammetry, University of Stuttgart, Germany -
(lena.joachim, wei.zhang, norbert.haala, uwe.soergel)@ifp.uni-stuttgart.de

Commission II, WG II/5

KEY WORDS: Visual SLAM, Crane Cameras, Mapping Accuracy, Digital Elevation Model

ABSTRACT:

In the context of the development of an autonomous tower crane, the usage of crane cameras to map the respective workspace as basis for autonomous path planning is investigated. The goal is to generate an up-to-date DEM as input for the crane control. As construction sites are highly dynamic scenes, it is crucial to be able to react to any changes. Thus, real-time mapping with a visual SLAM solution is aspired. As the quality of the DEM is important for such a safety critical application, we are evaluating the mapping quality of four state-of-the-art SLAM solutions, namely ORB-SLAM3, LDSO, DSM and DROID-SLAM. The results show that all approaches can handle our specific crane camera setup and thus are generally suited for our application. The DEM accuracies of all tested methods are even competitive with the result from standard offline photogrammetric processing, at least for the major part of the test site. However, there are limitations with regards to the DEM completeness. Consequently, our investigations show that the tested methods deliver a good basis for real-time accurate mapping, but for their application for autonomous path planning further refinements have to be made.

1. INTRODUCTION

The construction industry currently faces the challenge to increase the productivity and efficiency of the building process, as the demand for buildings is increasing. The automation of tower cranes can contribute to this goal, e.g. because their operations can be optimized with respect to time and energy efficiency. In the context of the development of an autonomous tower crane, we are investigating the usage of crane cameras (array of cameras as proposed by (Tuttas et al., 2016), see Fig. 1) to map the workspace of the crane as a basis for autonomous path planning and tracking. The goal is to generate an up-to-date digital elevation model (DEM) as input for the crane control, as this is an easy-to-use data format.

In previous investigations, we found that the quality of DEMs created via standard offline photogrammetric processing of the crane camera images (including steps like Structure-from-Motion and dense Multi-View-Stereo) is sufficient for our requirement of an accuracy of a few cm (Joachim et al., 2021). However, the real-time generation of the DEM is a key requirement to make autonomous path planning and tracking applicable. For dynamic scenes like construction sites, it is crucial to be able to react to any changes. Thus, a visual SLAM solution is aspired. As the quality of the DEM is very important for such a safety-critical application, we are evaluating state-of-the-art SLAM solutions with regards to their mapping quality. To do so, we compare the resulting point clouds and DEMs and the respective results from standard photogrammetric processing with ground truth data from laser scanning. As a first step, a single crane camera is considered.

As most works and benchmark datasets for visual SLAM primarily focus on the trajectory estimation accuracy, comprehensive evaluations of the mapping quality are only rarely available. This was also found by (Wang and Shahbazi, 2019), who investigated the mapping quality of several algorithms for imagery from a low cost rolling shutter camera on a micro aerial vehicle. In contrast to them, we work with higher-quality cameras with a global shutter, because crane cameras do not have restrictions regarding payload or power supply. Thus we expect a better performance, especially for the algorithm LDSO (Gao et al., 2018), which failed for their rolling-shutter camera.

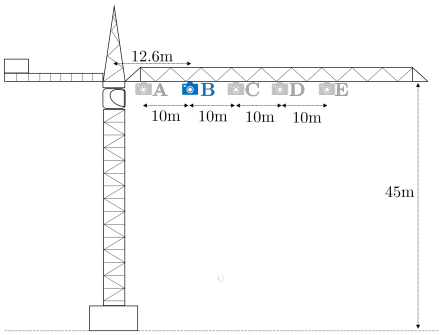


Figure 1. Crane camera setup at the top-slewing tower crane. For the experiments in this work, only camera B is used.

In comparison to typical applications of SLAM, the crane camera scenario is special for several reasons: As the camera is attached to the jib of a top slewing tower crane, it can only move with one degree of freedom, namely on a circle with fixed center point. The pose can be assumed as fixed, if the load-dependent bending of the structure of the crane is neglected. Additionally, the camera always moves orthogonally to the viewing direction, because it is facing downwards. The scene itself has a fixed maximum distance to the camera (distance between ground and jib) and there are no objects close to the camera.

Due to these reasons, we evaluate several state-of-the-art visual SLAM solutions to identify their strengths and weaknesses with regards to our specific use case. For this comparison, promising

* Corresponding author

recent algorithms based on different approaches with publicly available code are chosen: 1) ORB-SLAM3 (feature-based, sparse) (Campos et al., 2021), 2) LDSO (direct, semi-dense) (Gao et al., 2018), 3) DSM (direct, sparse) (Zubizarreta et al., 2020), 4) DROID-SLAM (deep-learning based, dense) (Teed and Deng, 2021).
The following chapters are structured as follows: Chapter 2 introduces the data and methods used for evaluation, as well as the four considered approaches. In chapter 3, the experimental evaluation of the individual SLAM solutions is described. Subsequently, their results are compared in chapter 4 and in chapter 5 follow the conclusion and an outlook to future work.

2. METHODOLOGY

2.1 Dataset and Test Site

Images The image data used within this work is generated by the crane camera system illustrated in Fig. 1. It consists of five downward-facing GigE Vision cameras mounted at the jib of a top slewing tower crane. In this work, we consider the monocular case as a first step, thus only camera B is used. It has a global shutter and a 12 megapixel CMOS sensor (4096×3000 pixels) with a pixel size of 3.45μm. With the focal length of 16mm, a ground sampling distance of approximately 1cm is reached. Thus, the extent of the footprint of one image at the ground is about 41×30m, whereas the long edge of the image is parallel to the jib. Fig. 2 shows two exemplary images of the camera. The images were taken with a framerate of 10fps during a 382° rotation of the crane with a duration of 230s. Thus, the camera moved with a speed of about $0.4\frac{\mathrm{m}}{\mathrm{s}}$. To enable the comparison of different input framerates, three downsampled datasets are derived, such that test sets with 10, 5, 2 and 1 fps are obtained.

Figure 2. Exemplary images of the crane camera showing the test site. RGB only used for visualization, the test dataset consists of grey scale images.

In order to use the imagery as input for the SLAM algorithms, the images, timestamps and the calibration file were stored in the same data structure as the EuRoC dataset (Burri et al., 2016), which all tested SLAM solutions are able to use as input. Because every implementation handles the EuRoC input a bit differently, some minor additional adjustments, e.g. of the format of the calibration file, had to be made. The same radial-tangential camera calibration parameters were used for the whole experiment.

Test site The scene captured by the camera is a research test site, where the tower crane is located approximately in the middle. Figure 3 shows the ground truth geometry of the site. It contains several objects, which are considered as obstacles for path planning. The main ones are an office container (a) and a small hall (b), which are both white and with low texture. Smaller obstacles are a fence surrounding the whole site (d) and several poles of about 1m height and 3cm width distributed at

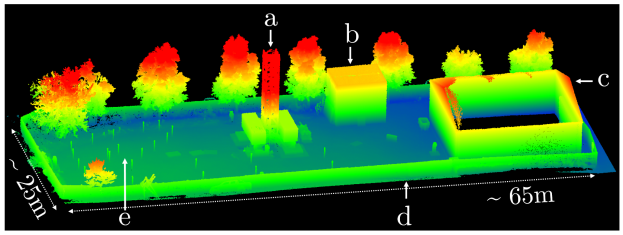


Figure 3. Reference point cloud of the test site with a) crane, b) container, c) hall, d) fence, e) poles.

the left part of the site (e). Additionally, one edge of the site is seamed with trees. For another impression of the site, refer to the images in Fig. 2.

Reference data The ground truth (GT) shown in Fig. 3 was captured with the ZEB Horizon mobile laser scanner (GeoSLAM Ltd., Nottingham, UK) and georeferenced via ground control points. Additionally, the result from standard photogrammetric processing of the 1fps image data with the commercial software Agisoft Metashape (Agisoft LLC, 2021) serves as a baseline result.

2.2 Investigated SLAM Solutions

In the following section, the evaluated SLAM solutions are introduced and the aspects relevant for their application to our dataset are explained.

ORB-SLAM3 ORB-SLAM3 (Campos et al., 2021) is a feature-based sparse visual SLAM method for monocular, stereo and RGB-D cameras. According to the authors, it is the most accurate visual SLAM system. It is an extension of the widely used ORB-SLAM2 (Mur-Artal and Tardos, 2017) and other previous works of the authors and integrates an approach for multimap data association. Although we do not consider this feature in this work, we use ORB-SLAM3 as the newest implementation of the ORB-SLAM approach. As a feature-based method, it applies feature tracking and minimizes the feature reprojection error with geometric bundle adjustment. The number of map points is defined by the parameter *number of features*, which is increased for our experiments to adapt to the high-resolution input images.
Because the published code does not provide the possibility to export the generated point cloud, it had to be extended with this functionality. Therefore, we adapted the solution which (Wang and Shahbazi, 2019) provided for ORB-SLAM2 to work with ORB-SLAM3. The reported processing time is measured for the main processing loop only, such that it doesn't contain the time of initialization and the saving of results.

LDSO LDSO (Gao et al., 2018) is a direct sparse (sometimes also called semi-dense) visual SLAM method, which is an extension of DSO (Direct Sparse Odometry) (Engel et al., 2018) with added loop closure detection and pose-graph optimization. As a direct method, LDSO is based on photometric bundle adjustment (PBA) and thus more robust in featureless areas than feature-based methods. It optimizes the photometric error over a sliding window of recent keyframes and applies a strategy to marginalize old keyframes and map points beyond the tracking window. The loop closure functionality is realized with a common feature-based bag-of-words approach.
For the application of LDSO to our data, the version without input of photometric calibration is used, as it is not available

for the crane camera. According to the authors of LDSO, an increase of the performance can be expected when including photometric calibration. To account for the big image size we evaluate the increase of the parameter *number of active points*, which influences the density of the resulting point cloud. The processing times are measured for the main thread only and thus do not include initialization or saving of the results.

DSM DSM (Direct Sparse Mapping) (Zubizarreta et al., 2020) is a direct monocular visual SLAM method. Like LDSO, it is based on PBA and both of them use the same photometric model. The main difference between them is that DSM is a fully direct method which integrates point reobservations and reuses existing map information directly within the PBA, which leads to a persistent map. Reusing points is especially beneficial for our application, because the crane cameras don't explore unseen areas to the extend of typical SLAM applications, as the crane itself has a fixed location.
The authors of DSM report that to the date of the publication of their work, DSM was the most accurate direct monocular SLAM method. Unlike most other works, they not only evaluated the trajectory but also the mapping results and compared them to LDSO. Their investigations show that the map of DSM is more accurate than the one of LDSO while LDSO yields more map points because it doesn't reuse map points.
Different *numbers of active points* are considered to account for the large image size of our data. This parameter refers to the maximum number of points per keyframe which are part of the PBA and influences the density of the resulting point cloud. To be able to extract the respective number of points, we increased the *number of candidate points* accordingly.
The published code of DSM includes a GUI where the processing has to be manually started and ended. Thus, the reported processing time contains an unknown offset for the manual starting and stopping process and the point cloud export (we estimate this offset to be below 10s), which should be taken into account when comparing it to the other methods.

DROID-SLAM DROID-SLAM (Teed and Deng, 2021) is a deep-learning based dense visual SLAM method, which sets many new state-of-the-art records on multiple SLAM benchmarks. For the image alignment and tracking between consecutive frames, this method is similar to ORB-SLAM3, performing pixel matching explicitly and thus minimizing reprojection errors using bundle adjustments (BA). However, an essential difference is that it does the pixel correspondence matching densely for every pixel by utilizing a dense optical flow prediction network. The network uses the current optical flow computed by current poses and depth estimates to predict a flow update and its confidence. Upon a flow update, the BA is performed with the update as target and the confidences as constraint weights to refine the camera poses and depth maps.
This dense approach comes with a cost that enormous GPU memory is demanded for hosting the input data, as well as for the network feed-forwarding and performing BA. To deal with that, the original implementation of DROID-SLAM resizes by default input images to a resolution with only approx. 20k pixels. This is not significant for usual SLAM applications. However, for the crane camera with a resolution of 3000×4096, the information is reduced by 600 times with many fine details of the scene lost. Thus an extension is performed in this work to enable the processing with higher resolution so that a denser map with finer details can be produced. Especially we observe that the most significant memory bottleneck is at the correlation lookup operator in the frontend, where a 4D correlation volume is precomputed for the lookups afterward. In contrast, the backend uses a memory-efficient alternative implementation, where the correlations are only computed when needed. Therefore, our first trial is to replace this with the memory-efficient operator, though it leads to a longer runtime. Thus to better balance the accuracy and efficiency, we design a hybrid-resolution strategy with low (1/4) and high resolution respective for the frontend and backend processing. To bridge the resolution gap, before the global BA, the low resolution depth maps are up-sampled using nearest-neighbor interpolation, which is taken afterward as initial values for the global BA optimization at the backend.
In this work, the pre-trained model provided by (Teed and Deng, 2021) is used directly without fine-tuning. Even though it is trained only on a synthetic dataset (Wang et al., 2020), we found this model has ability to generalize across domains for the crane images from a top-down perspective. Additionally, to filter out outlier depth estimates, it projects the depth maps of each keyframe into the depth maps of its neighbors to check if the neighbor counterparts are agreed with the estimates. The parameter *threshold* sets the maximum tolerance of depth deviation, while *count* is the minimum number of agreed neighbors. Furthermore, since the crane moves slowly, the parameter *stride* is modified, which reduces the frame rate to 1/*stride*.

2.3 Evaluation

As a first step, some postprocessing steps are performed on the point clouds generated by the SLAM solutions in order to prepare them for the DEM generation and comparison (compare (Wang and Shahbazi, 2019)). The following needed steps are performed in CloudCompare (CloudCompare, 2021):

1) Filtering: Outliers are filtered with a statistical outlier removal filter (SOR), as implemented in CloudCompare. The mild filtering parameters *number of points* $= 6$ and *standard deviation* $= 3$ were chosen in order to only remove large outliers.
2) Determination of scale, georeferencing: As the SLAM point clouds have an arbitrary scale, they are transformed to metric scale via alignment to the ground truth point cloud with manually picked identical points and refinement by ICP (not applied for very sparse clouds).
3) Crop: All point clouds, including the reference, are cropped to the same extent and height range to facilitate their comparison.

Afterwards, a DEM with a raster width of 0.1m is derived from each point cloud. As the crane operates above the scene, the respective maximum height is assigned to each cell. Empty cells are left empty. Finally, the difference Δ_{DEM} between the SLAM DEM and ground truth DEM is calculated for the filled cells as the basis of the evaluations:

$$\Delta_{DEM} = DEM_{SLAM} - DEM_{GT} \tag{1}$$

The DEM generation and differencing is performed with the software OPALS (Pfeifer et al., 2014).
For the evaluation of the results, the completeness and accuracy of the DEMs are considered. Apart from the number of points in the postprocessed point clouds N_p and visual inspection of the result, the completeness is evaluated based on the percentage of filled DEM cells w.r.t. reference N_{filled}. Here, $n_{SLAM,GT}$ are the total numbers of filled cells of the respective DEMs:

$$N_{\text{filled}}[\%] = 100 \cdot \frac{n_{SLAM}}{n_{GT}} \tag{2}$$

As main measure of accuracy, we use Δ_{DEM} and its mean $\overline{\Delta}$ and standard deviation σ_Δ. Additionally, the normalized cumulative histogram of the absolute differences $|\Delta_{DEM}|$ and three of its quantiles ($q_{50\%}$, $q_{75\%}$, $q_{90\%}$) are considered for the evaluations. The normalization is calculated relative to the amount of filled cells of the respective DEM. In general, it should be kept in mind that the accuracy measures are based on varying numbers of observations (filled DEM cells).

All experiments were performed on a PC with an Intel Core i9-10900K CPU and an NVIDIA GeForce RTX 3090 GPU with 24GB memory.

3. EXPERIMENTS

3.1 ORB-SLAM3

In our experiments, the input frame rate had a large influence on the processing time t of ORB-SLAM3. Considering the four tested frame rates, only the 1fps dataset could be processed in real-time, whereas for 2fps the mean processing time increased by approx. 7 seconds. This behavior was independent of the number of features. For 5fps t increases by up to 40 seconds, dependent on the number of features used, and for 10fps t is doubled w.r.t. 1fps. Thus, we do not consider the 10fps results for these evaluations. Regarding the number of features, values between 1000 (default) and 10000 were tested. We observed that for some high numbers of features the system finishes early without throwing an error, such that the resulting point cloud only covers a part of the site. To illustrate the influence of frame rate and number of features on the completeness and accuracy of the result, four exemplary versions of parameters are presented in the following:

a) 1fps, 2000 features
b) 1fps, 5000 features
c) 5fps, 2000 features
d) 5fps, 5000 features

Table 1 shows the measures of completeness and the processing times. As expected, the number of points and filled cells increases according to the number of features. But using more features doesn't only have a positive effect on the completeness, but it also leads to a slight increase of accuracy, as can be seen in Tab. 2. As the processing time doesn't increase significantly (at least for 1fps), it can be stated that for high-resolution images it is beneficial to increase the number of features above the suggested default.

	N_p	N_{filled} [%]	time [s]
a)	4,814	1.6	236
b)	10,490	3.4	237
c)	4,705	1.6	255
d)	9,765	3.3	280

Table 1. Comparison of mapping completeness and processing times of the different parameter versions of ORB-SLAM3.

	$\overline{\Delta}$	σ_Δ	$q_{50\%}$	$q_{75\%}$	$q_{90\%}$
a)	-0.54	1.64	0.05	0.12	1.30
b)	-0.52	1.60	0.05	0.10	1.13
c)	-0.42	1.77	0.07	0.16	1.85
d)	-0.53	1.69	0.05	0.11	1.59

Table 2. Comparison of mapping quality of different parameter versions for ORB-SLAM3.

On the other hand, increasing the frame rate leads to a slight decrease of accuracy (see Tab. 2) and no increase in completeness with the cost of increased processing time. Thus, choosing a low frame rate is sufficient and the amount of data can be kept low without any disadvantages. As a result, version b delivers the best results and is chosen for comparison with the other SLAM methods. Fig. 9a shows the difference to the reference DEM and Fig. 8a shows the corresponding point cloud. Visual inspection shows that ORB-SLAM3 creates evenly distributed points and although the point cloud is sparse, it contains the most solid objects in the scene including the poles.

3.2 LDSO

For our dataset, the performance of LDSO is strongly dependent on the input frame rate. Only with the 5fps dataset, sufficiently correct results could be generated. Higher frame rates lead to a strong increase of noise, artifacts and slow processing times and lower frame rates additionally can even lead to fully degenerated point clouds. Thus on the one hand, LDSO requires a certain minimum image overlap for the tracking to work, but on the other hand, the image overlap should not exceed a certain maximum to limit noise. Regarding the number of active points, we did not observe significant changes of the point cloud for increasing it (2000, 4000, 10000), unlike as in (Engel et al., 2018). Visual inspection only showed a small increase of the total number of points which is mainly caused by noise. That is why we use the proposed default of 2000 active points for the evaluation.
The measures for completeness and accuracy of the result generated with the 5fps dataset and 2000 active points are summarized in Tab. 7 and 6 in chapter 4. Two remarks should be made regarding these results. First, the processing time reported in Tab. 7 refers to LDSO without real-time enforcement, which is accordingly slower. The reason is that with our data, the option *enforcing 1 × real-time execution* leads to entirely failed tracking for all tested frame rates. The authors of DSO (Engel et al., 2018) reported on such failure in rare cases. Secondly, we observed that also for the 5fps dataset the tracking fails in some runs such that the point cloud contains severe artifacts. Although robustness is not in focus of the investigations, these observations indicate that LDSO is not entirely robust for our dataset.
Fig. 8b shows the postprocessed point cloud. It contains all solid objects of the scene and especially the edges of objects are clearly visible, because many points cover them. It is particularly noticeable that the whole point cloud consists of small clusters of points, which most likely belong to the same point in object space (see Fig. 4, left). This is an effect of the point marginalization strategy of LDSO, which leads to the re-creation of points in a repetitive dataset like ours. Furthermore, all raw point clouds contain noise within a cone between the test site

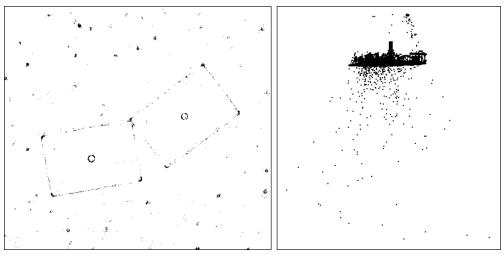


Figure 4. Left: Detail of point cloud generated with LDSO (top view). Right: Side view of raw result with large outliers.

and the cameras and large outliers underneath the ground (see Fig. 4, right). Although for the chosen parameters this noise is smaller than for the other tested versions, filtering is still inevitable in order to receive usable results.

3.3 DSM

Considering the input frame rate, an increase of noise and processing time can be observed for increased frame rates. For 10fps this effect is so strong, that the results are not considered for further evaluation. Additionally, 1fps is discarded as DSM is not stable for such a low frame rate. It either fails or delivers poor results. The behavior for an increased number of active points is similar as for the frame rate: Noise (especially underneath the surface) and processing time increase. Additionally, visual inspection shows no significant increase of detail for the three tested numbers of active points (1500, 2500 and 3500), just a general increase of the point density. Thus, the highest number of active points is discarded. It is remarkable that despite the increased noise, all point clouds look pretty well, there are no artifacts or failures (with the exception of the 1fps dataset). Thus, DSM seems to be robust against changing frame rate or the number of active points. Based on these considerations, four parameter combinations are further evaluated with regards to their influence on the completeness and accuracy of the result:

a) 2fps, 1500 features
b) 2fps, 2500 features
c) 5fps, 1500 features
d) 5fps, 2500 features

Table 3 summarizes the measures of completeness and the processing times for these versions and in Tab. 4 the measures of accuracy are compared. As expected, an increased number of active points increases the number of points in the point cloud as well as N_{filled}. However, the processing time increases as well and there is no gain in accuracy (for the 2fps dataset). Thus, increasing the number of active points does not have any advantages apart from a higher point density. Regarding the frame rate, using 5fps instead of 2fps does not improve the results either. Hence, version a) is chosen for comparison with the other algorithms, because it has the lowest processing time.

	N_p	N_{filled} [%]	time [s]
a)	67,627	6.1	269
b)	116,601	9.9	276
c)	66,173	6.1	293
d)	108,193	9.5	305

Table 3. Comparison of mapping completeness and processing times of the different parameter versions of DSM.

	$\overline{\Delta}$	σ_Δ	$q_{50\%}$	$q_{75\%}$	$q_{90\%}$
a)	-0.47	1.52	0.04	0.07	0.60
b)	-0.48	1.55	0.04	0.07	0.70
c)	-0.52	1.60	0.06	0.09	0.81
d)	-0.48	1.57	0.04	0.07	0.75

Table 4. Comparison of mapping quality of different parameter versions for DSM.

Fig. 9b shows Δ_{DEM} for this result and Fig. 8c the corresponding point cloud. The point cloud only contains the bigger objects of the scene, smaller details like the poles or big parts of the fence are missing or only covered by a single point. Additionally, some parts at the edges of the site are missing (see Fig. 9b). However, it is worth highlighting that the considered results of DSM have very low noise. In fact, the filtering did not remove any points.

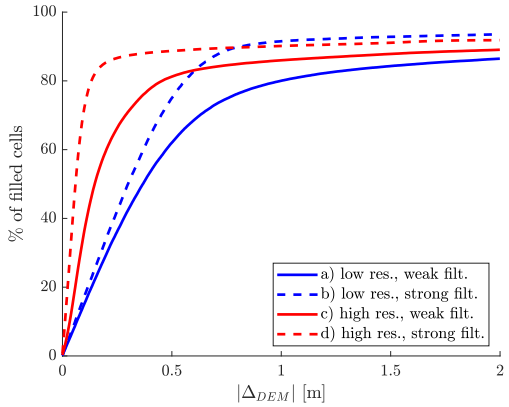


Figure 5. DROID-SLAM: Normalized cumulative histogram of $|\Delta_{DEM}|$. For better visualization, $|\Delta_{DEM}|$ is clipped to 2m.

3.4 DROID-SLAM

For DROID-SLAM, 5fps is the highest frame rate which still delivers real-time results, thus we only consider this dataset for the following evaluations. The parameter *stride* is fixed to 8. We found it does not affect the performance while significantly improving the runtime, as the reduced frame rate is still sufficient for keyframes to be tracked. For the original implementation of DROID-SLAM, the maximum feasible image size is 549×750 pixels. The adapted more efficient implementation introduced in this work is able to increase the image size for the backend to 2272×3104 pixels while the frontend uses 568×776 pixels. In order to compare the results of these two versions, four exemplary cases are evaluated. As the internal filtering step of DROID-SLAM, whose intensity is determined by the parameters *count* and *threshold*, strongly influences the results, we consider the results for weak and strong filtering for both implementations:

a) Original: count 2, threshold 0.02 (weak filt.)
b) Original: count 4, threshold 0.005 (strong filt.)
c) More efficient: count 2, threshold 0.01 (weak filt.)
d) More efficient: count 4, threshold 0.0025 (strong filt.)

Figure 5 compares the normalized cumulative histograms of $|\Delta_{DEM}|$ for these four cases. It clearly shows an increase in accuracy for the efficient implementation. Also the stronger filtering significantly increases the accuracy. Considering the completeness, the efficient implementation increases the number of points by a factor of approx. 20 which leads to a significant increase of N_{filled}, as Tab. 5 summarizes. It also shows that the filtering decreases the completeness of the data. The visual comparison of the point clouds c and d in Fig. 6 shows the reason for that: The strong filtering not only removes

	N_p	N_{filled} [%]	time [s]
a)	159,309	52.5	66
b)	69,639	29.0	66
c)	2,839,362	87.6	187
d)	1,622,690	71.7	187

Table 5. Comparison of mapping completeness and processing times of the different parameter versions of DROID-SLAM.

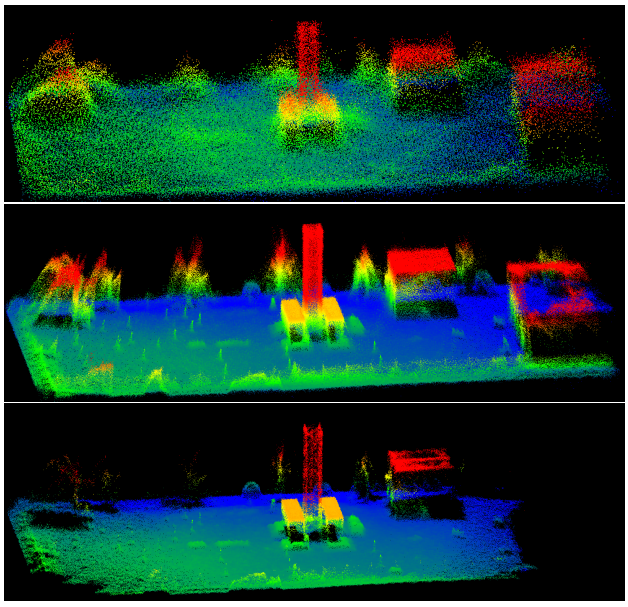


Figure 6. Point clouds generated with DROID-SLAM. From top to bottom: version a, c, d.

noise but the consistency-check also filters out details like the fence or low-textured objects like the hall. Thus, finding a balance between filtering to increase accuracy and keeping important details is crucial and requires fine-tuning w.r.t. the specific scene. Furthermore, Fig. 6 clearly shows the benefit introduced by the more efficient implementation with higher image resolution. The original implementation yields very coarse and noisy point clouds in which many details are missing.
In summary, it can be stated that DROID-SLAM delivers the most accurate and complete results for a high image resolution and appropriately strong filtering. For the comparison with the other SLAM solutions, the version with the highest accuracy (version d) is considered.

3.5 Photogrammetric Baseline

The result of processing the images with Agisoft Metashape (Agisoft LLC, 2021) is shown in Fig. 8e. As expected, it is much denser than the SLAM results and contains almost all objects of the scene. However, the resulting point cloud is quite noisy on textureless surfaces, which can also be seen in the visualization of Δ_{DEM} (Fig. 9e) where higher deviations occur especially at the border of objects with low texture like the hall or the container. In our opinion, this might result from the small image bases in our dataset, which leads to unfavourable angles of intersection during photogrammetric processing. Adding the other cameras of the array would help to eliminate such uncertain matches. Additionally, Fig. 9e shows a small upward bending towards the edges of the point cloud, which can also be caused by the missing image overlap in jib direction.

4. COMPARISON OF RESULTS

Accuracy The normalized cumulative histogram of $|\Delta_{DEM}|$ (Fig. 7) reveals that the mapping accuracies of most tested methods are quite similar. Only LDSO lies a bit behind, which is caused by a bigger portion of larger differences. This is indicated by the large standard deviation σ_Δ and the up to twice as high values for $q_{75\%}$ and $q_{90\%}$ compared to the other results,

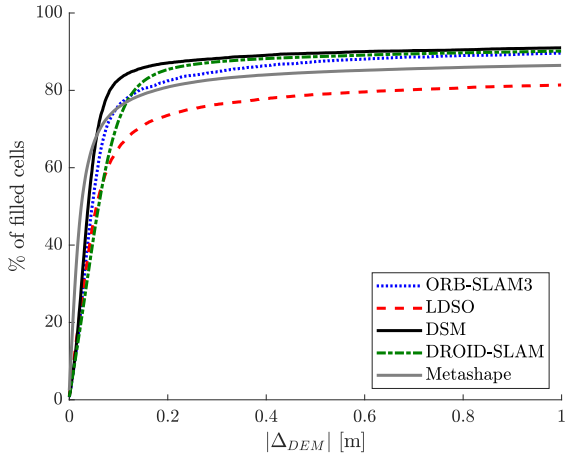


Figure 7. Comparison of normalized cumulative histograms of $|\Delta_{DEM}|$. For better visualization, $|\Delta_{DEM}|$ is clipped to 1 m.

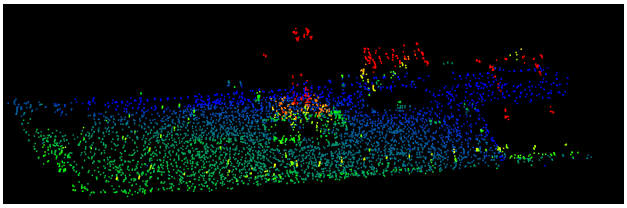
Method	$\overline{\Delta}$	σ_Δ	$q_{50\%}$	$q_{75\%}$	$q_{90\%}$
ORB-SLAM3	-0.52	1.60	0.05	0.10	1.13
LDSO	-0.76	2.03	0.06	0.24	4.21
DSM	-0.47	1.52	0.04	0.07	0.60
DROID-SLAM	-0.46	1.67	0.06	0.11	0.93
Metashape	-0.58	1.88	0.03	0.09	3.11

Table 6. Comparison of mapping quality of different methods.

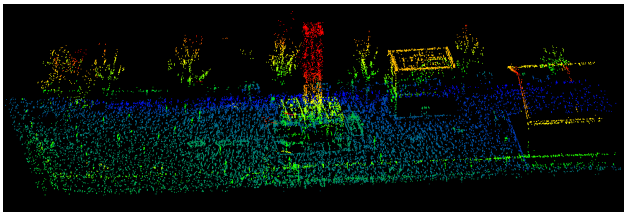
whereas the median is similar for all methods, as Tab. 6 shows. The latter proves the visual impression in Fig. 9c, which indicates that the accuracy at the ground, where the major part of the cells belong to, is similar to the results of ORB-SLAM3 and DSM. This illustration of the spatial distribution of Δ_{DEM} also reveals the main source for the larger differences: The DEM from LDSO covers the largest extent of the test site and thus a bigger portion of its cells lies in areas, where objects which are difficult to reconstruct (e.g. trees, fence, hall) are located.
In general, the three other methods yield similar results for the mean, median and $q_{75\%}$, only for $q_{90\%}$ they slightly differ (Tab. 6). DSM does deliver the most accurate results, both regarding the results in Tab. 6 and when considering the cumulative histogram (Fig. 7). It is remarkable that it even slightly outperforms the results of Metashape. In fact, the results of ORB-SLAM3, DSM and DROID-SLAM all reach a similar level of accuracy as the Metashape result. Although this is promising, it should be remarked that there are bigger differences regarding the completeness, which is discussed in the next paragraph.
Looking at the accuracy in terms of Δ_{DEM} illustrated in Fig. 9, it can be seen that all results including the photogrammetric reference have in common, that higher differences occur at the trees and at the edges of elevated objects. The first is a common problem of image based 3D reconstruction, which doesn't work well for vegetation. Thus the missing trees cause high negative deviations. Apart from the absence of the objects in the data, the latter has two more reasons: Noise causes the objects to be wider than they are in reality, leading to high positive differences (e.g. visible at the poles in the right part of Fig. 9d). The second reason are small displacements of such edges due to the georeferencing, which is often the cause for the differences in the area of the fence (e.g. at top of Fig. 9d).

With regards to the accuracy of the geometry of the test site as a whole, it can be concluded that all methods reach an adequate

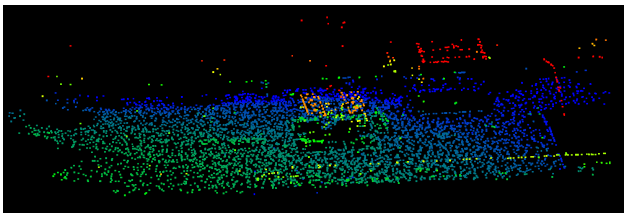
accuracy as no systematic deviations or deformations occur. In fact, the only systematic deformation can be observed for the Metashape result (see chapter 3.5). However, it should be mentioned that the DROID-SLAM result contains noise distributed all over the site (small positive deviations indicated by yellow color in Fig. 9d).



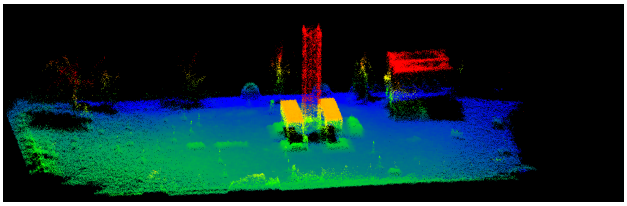
(a) ORB-SLAM3



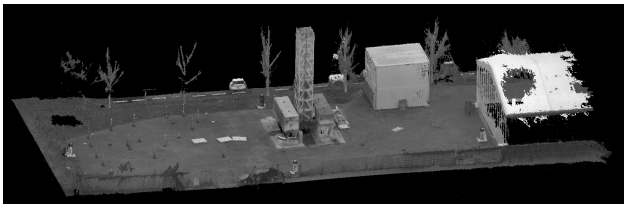
(b) LDSO



(c) DSM



(d) DROID-SLAM



(e) Metashape

Figure 8. Comparison of the point clouds. For a-d color codes height. The point size is enlarged for the sparse results for better visualization.

Completeness The completeness of point clouds generated by sparse, semi-dense and dense methods can hardly be compared fairly as they are designed for different purposes. Nevertheless, this comparison reveals some interesting aspects. On the one hand, DROID-SLAM as the only dense method delivers a very dense result (Fig. 8d) while still having high accuracy. On the other hand, its point cloud lacks important details, like e.g. big parts of the fence. Even in the Metashape result, some parts of the fence are missing. The semi-dense result of LDSO, however, contains most scene objects (at least their contours) including almost the whole fence (Fig. 8b). Thus, according to visual inspection the LDSO point cloud can be considered as the most complete one with regards to the amount of contained obstacles. The sparse point clouds of DSM and ORB-SLAM3 still contain many of the obstacles, even if they are sometimes only represented by a single point (e.g. the poles). The reason for this is that such objects are quite distinct from their surroundings and thus are likely to be chosen for tracking. However, for sparse keypoint-based or direct approaches there is no guarantee that all objects are included. That is why such approaches cannot provide a reliable map, but rather a basis for dense reconstruction (see chap. 5).

Method	N_p	N_{filled} [%]	time [s]
ORB-SLAM3	10,490	3.4	237
LDSO	125,813	11.0	276
DSM	67,627	6.1	269*
DROID-SLAM	1,622,690	71.7	187
Metashape	9,795,377	90.1	-

Table 7. Comparison of mapping completeness and processing times of the different methods.

When considering completeness in terms of the amount of (correctly) filled cells of the DEM as summarized in Tab. 7 and visualized in Fig. 9, it is clear that Metashape delivers the densest and thus most complete result. Additionally, it also covers the largest area of the test site, while for the other results (except LDSO) some parts at the edges of the site are missing. As expected, for the SLAM methods the amount of filled cells varies strongly. For the two sparse methods (ORB-SLAM3 and DSM) the number of filled cells is below 10% of the reference DEM. LDSO as a semi-dense method yields more than ten times the amount of points as ORB-SLAM3 but only about three times more filled cells, which can be explained by the effect of point clusters in the LDSO point cloud. Although DROID-SLAM delivers the most complete DEM, it still has 20% less filled cells than the photogrammetric reference. The comparison of the two DEMs in Fig. 9d/e shows that this is rather caused by some missing parts in the DROID-SLAM DEM (left side, upper right edge and trees) than by a lack of density as for the other methods. The major part of these areas is removed by the consistency-based filtering of DROID-SLAM and is contained in the point clouds with less strict filtering.

Processing time For our time measurements, the processing times of the different methods vary by up to 90s (see Tab. 7). Only ORB-SLAM3 and DROID-SLAM reach real-time ($t \leq 230$s), whereas DROID-SLAM is even significantly faster than the real-time stream of the images. However, it should be noted that the compared results are based on different input frame rates as explained in chapters 3.1-3.4.

5. CONCLUSION AND OUTLOOK

The experiments show that all tested approaches are capable of working with the special geometry of our dataset and are thus in general suited for our application. None of the methods can be clearly identified as the most suitable one, because they have different advantages and disadvantages. With regards to the general accuracy of the DEM, they can even compete with the offline photogrammetric result. However, these measures do not reflect the absence of important parts of the test site in the resulting DEM, which occurs to varying degrees for the different methods. For the sparse and semi-dense methods, the application for real-time workspace mapping as basis for path

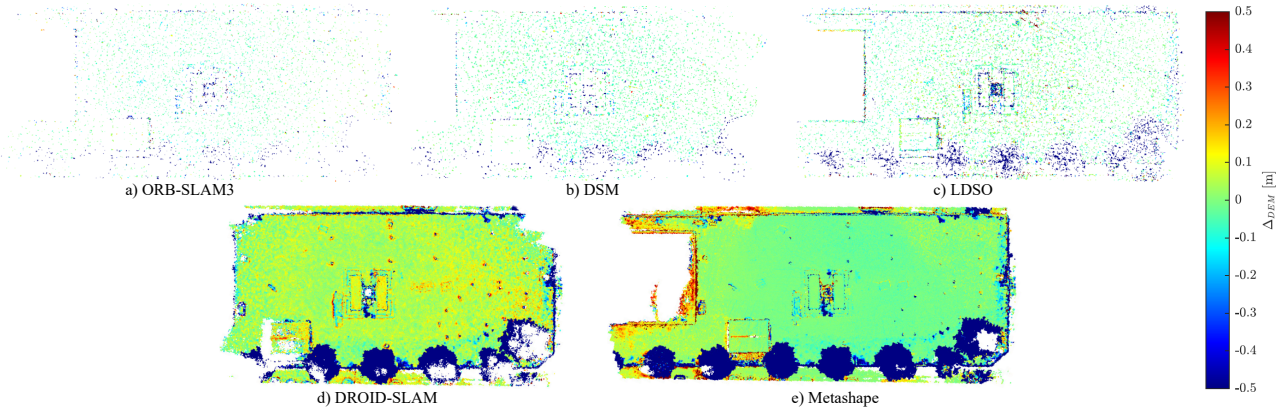


Figure 9. Comparison of Δ_{DEM}. For better visualization, Δ_{DEM} is clipped to $[-0.5, 0.5]$m.

planning would require an additional module to derive a dense and thus more complete point cloud based on the sparse result, as e.g. proposed by (Matsuki et al., 2021). Here, ORB-SLAM3 is a promising base for such an extension due to its fast processing time and DSM due to its high accuracy. The dense result of DROID-SLAM however would have to be refined to be able to capture all obstacles. To do so, further modifications to enable the processing of the full resolution images could be promising. In general, all methods were previously mostly evaluated for images with much lower resolution, which causes challenges in terms of processing time or memory issues. For high-quality mapping, however, the usage of high-quality imagery is an important prerequisite.
Apart from that, for our future work it will be interesting to extend the monocular SLAM approach to a multi-camera approach by treating the whole crane camera array as a partly flexible multi-camera rig. This would not only deliver scale information but could also significantly improve the accuracy and also the reliability of the resulting DEM. Additionally, the integration of a-priori camera pose information derived from crane sensor measurements could be used for direct georeferencing and thus support the tracking and speed up the processing.

6. ACKNOWLEDGEMENTS

Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy – EXC 2120/1 – 390831618.

REFERENCES

Agisoft LLC, 2021. Agisoft Metashape Professional Software, Version 1.7.2. http://www.agisoft.com/downloads/installer/.

Burri, M., Nikolic, J., Gohl, P., Schneider, T., Rehder, J., Omari, S., Achtelik, M. W., Siegwart, R., 2016. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10), 1157–1163.

Campos, C., Elvira, R., Rodriguez, J. J. G., Montiel, J. M. M., Tardos, J. D., 2021. ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM. *IEEE Transactions on Robotics*, 1–17.

CloudCompare, 2021. (Version 2.11.1) [GPL Software]. http://www.cloudcompare.org/.

Engel, J., Koltun, V., Cremers, D., 2018. Direct Sparse Odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Gao, X., Wang, R., Demmel, N., Cremers, D., 2018. LDSO: Direct Sparse Odometry with Loop Closure. *International Conference on Intelligent Robots and Systems (IROS)*.

Joachim, L., Ackermann, S., Haala, N., Soergel, U., 2021. Using crane cameras for workspace mapping and monitoring for an autonomous tower crane. *Proc. of the 7th International Conference on Machine Control & Guidance*, 47–53.

Matsuki, H., Scona, R., Czarnowski, J., Davison, A. J., 2021. CodeMapping: Real-Time Dense Mapping for Sparse SLAM using Compact Scene Representations. *IEEE Robotics and Automation Letters*, 6(4), 7105–7112.

Mur-Artal, R., Tardos, J. D., 2017. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics*, 33(5), 1255–1262.

Pfeifer, N., Mandlburger, G., Otepka, J., Karel, W., 2014. OPALS – A framework for Airborne Laser Scanning data analysis. *Computers, Environment and Urban Systems*, 45, 125-136.

Teed, Z., Deng, J., 2021. DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*.

Tuttas, S., Braun, A., Borrmann, A., Stilla, U., 2016. EVALUATION OF ACQUISITION STRATEGIES FOR IMAGE-BASED CONSTRUCTION SITE MONITORING. *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, XLI-B5, 733–740.

Wang, J., Shahbazi, M., 2019. MAPPING QUALITY EVALUATION OF MONOCULAR SLAM SOLUTIONS FOR MICRO AERIAL VEHICLES. *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, XLII-2/W17, 413–420.

Wang, W., Zhu, D., Wang, X., Hu, Y., Qiu, Y., Wang, C., Hu, Y., Kapoor, A., Scherer, S., 2020. Tartanair: A dataset to push the limits of visual slam. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 4909–4916.

Zubizarreta, J., Aguinaga, I., Montiel, J. M. M., 2020. Direct Sparse Mapping. *IEEE Transactions on Robotics*, 36(4), 1363–1370.