Monocular Localization in Feature-Annotated 3D Polygon Maps

Alexander Mock[1], Thomas Wiemann[1,2], Joachim Hertzberg[1,2]

***Abstract*—Localization in six degrees of freedom is becoming increasingly relevant, especially in indoor environments where GPS is not available. To localize autonomous vehicles like UAVs in such areas, reliable methods for self-localization with low-weight sensors are required. In this paper, we present an approach to precisely localize systems with monocular cameras in polygonal 3D maps annotated with keypoints and feature descriptors computed from LiDAR data and associated reference images. Our contribution consists of offline map computation from high resolution 3D point clouds with corresponding reference images as well as online localization within these maps using low cost sensors. During localization, features extracted from the vehicle's camera image stream are matched against the reference map. The proposed method is capable of real-time localization and suitable for precise global localization. The evaluation shows comparable results to state of the art with high re-localization accuracy.**

***Index Terms*—LiDAR Mapping, Localization, Monocular Localization, Image Features**

I. Introduction

The continuous development of SLAM algorithms allows to generate accurate 3D point clouds using high resolution LiDAR sensor technology. With professional laser scanning systems, it is possible to associate the recorded points with color information from co-calibrated RGB cameras. Such annotated point clouds are geometrically precise, but hard to handle on mobile robots due to the large amount of data and missing topology between the unordered points. Hence, using them for robotic applications other than scan matching is seldom seen. Recently, methods to reconstruct polygonal maps from such point clouds have become available [1]. Such maps have been successfully used for path planning [2] and incremental localization [3] In this paper, we present an approach to fuse geometrically precise 3D polygon maps with keypoints and feature descriptors from the images of the scanning system. The keypoints are projected onto the polygonal maps and serve as landmarks for precise global localization with low-cost cameras.

First, a polygonal reconstruction based on the point cloud data is computed to approximate the 3D point cloud [4]. Re-projection of the computed keypoints and descriptors via raycasting is then used to localize keypoint positions from the 2D camera images precisely in 3D space by computing the intersection of the rays from an inverse camera model with the polygonal mesh. We call this representation *Feature-annotated 3D Polygon Map (FAPM)*.

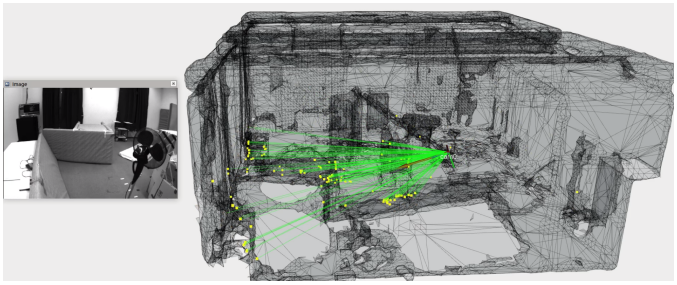

Fig. 1: Feature Annotated Polygon Map Localization (FAPM-L) demonstrated on the EuRoC MAV Vicon room data. The polygonal model was reconstructed from 3D point cloud data. Visual features extracted from the corresponding images are projected onto the mesh and stored as part of the map. This Feature Annotated Polygon Map (FAPM) serves as reference for feature-based 6D localization (FAPM-L) using the UAV's camera, as indicated by the green lines.

In addition to this novel representation, we present a simple global localization approach called FAPM-L for monocular localization using FAPMs. For that, we match feature descriptors and keypoints extracted from the camera stream of a mobile system against the previously recorded FAPM. The main advantage of FAPM-L is that we can use the 3D information encoded in the FAPMs as highly reliable landmarks for feature-based localization. The FAPM-L method itself is independent from the used keypoint extraction algorithm and descriptors, so in principle different methods and representations can be used. Using this idea, we can use well understood keypoint matching methods from the literature to compute 2D-3D correspondences between the extracted keypoints and the corresponding 3D landmarks directly. Since this operation is different from the purpose image features were initially designed for, we evaluate different OpenCV implementations for the use-case at hand. We further demonstrate the benefit of precise geometric information by comparing our FAPM-L results to the re-localization module of ORB-SLAM2 [5] on the well known EuRoC MAV data sets [6] as illustrated in Fig. 1.

[1]The authors are with the Knowledge Based Systems and Autonomous Robotics groups at the Institute of Computer Science, Osnabrück University, Osnabrück, Germany `firstname.lastname@uni-osnabrueck.de`

[2]Thomas Wiemann and Joachim Hertzberg are with the German Research Center for Artificial Intelligence (DFKI), Niedersachsen Lab, Plan-based Robot Control Group, Osnabrück, Germany `firstname.lastname@dfki.de`

The DFKI Niedersachsen Lab (DFKI NI) is sponsored by the Ministry of Science and Culture of Lower Saxony and the VolkswagenStiftung

II. Related Work

For for feature-based visual SLAM various methods exist [7]. In the context of monocular visual SLAM, ORB-SLAM and it's variants are currently state of the art. The first version of ORB-SLAM was introduced in 2015 by Mur-Artal et. al [8]. They designed ORB-SLAM for real-time localization. For that, they use the comparatively fast ORB-features [9] for tracking, mapping, re-localization and loop closing during SLAM. Their approach consists of building graph structures like *Essential Graph*, *Spanning Tree* and *Covisibility Graph* over the inserted keyframes. Furthermore, they use a *Bag of Words* (BoW) [10] approach to measure similarity in keyframes in order to improve loop closing compared to previous approaches like PTAM [11].

ORB-SLAM2 [5] adapted the existing ORB-SLAM implementation to support stereo and RGB-D cameras. Additionally, it introduced a special functionality called *Localization Mode* that disables Local Mapping and Loop Closing. It assumes that an accurate map of the environment has been generated beforehand and that the environment did not change significantly. Similar to our method, this global localization mode is explicitly indented for devices with low computational resources. In this respect, their idea of feature-based re-localization is similar to our approach, hence we chose it as baseline for evaluation. However, in our work we explicitly make use of the high accuracy of laser scanners, state of the art LiDAR SLAM algorithms [12]–[14] and surface reconstruction methods [4] to support the re-localization task.

Our previous work has shown that the automatic generation of textured polygonal maps is feasible with high geometric precision [15]. With known intrinsic and extrinsic parameters, it is also possible to compute color textures for such maps with chosen resolution [16]. To become independent from user-defined parameters, we use the acquired images directly to compute image features. These features are then back-projected via ray-casting onto the reconstructed polygonal surface. This computation is fast on current graphics cards and highly precise, resulting in an accurate localization of keypoints in continuous 3D space. In contrast to approaches such as 2D3D-MatchNet [17], we thus compute accurately located 3D features that can be matched directly with camera images without further post-processing.

During matching, we compare features from high resolution images with references from a cheap camera on a mobile system. Hence, we need to evaluate existing features for the intended application. For that, we created a high resolution reference data set to evaluate SIFT [18], SURF [19], ORB [9], and AKAZE [20] with respect to our requirements. To compare our method with the state of the art, we additionally benchmark our approach with ORB-SLAM2 on the EuRoC MAV data sets.

III. Feature Annotated Polygon Maps

This section presents the data structure and algorithms methods to compute FAPMs. First, we very briefly describe the polygonal map representation based on previous work. The remaining sections concentrate on the required steps for feature annotation.

A. Polygonal Map Creation

For map generation, we assume that we have a LiDAR system with co-calibrated RGB-camera and known intrinsic and extrinsic parameters, to capture point clouds and camera images in a global coordinate system. Using our LVR2 software [1], [15], we compute polygonal meshes from the captured point clouds. The polygonal mesh represents the geometric part of the FAPM and is then annotated with keypoints and feature descriptors re-projected from the system's camera.

B. Feature Projection

From the collected images, we extract SURF, SIFT, ORB, and AKAZE keypoints and descriptors using OpenCV. To make the keypoints more robust under perspective changes, we apply the method proposed by Yu et al. [21]. It simulates a virtual camera that points at the image from different orientations. For each of these simulated perspectives, the respective keypoints are computed from the transformed images. The required additional runtime for this depends on the number of simulated camera poses. Since we are using the extracted features as in a static map, this pre-processing overhead is acceptable.

For *Feature Projection (FP)*, we raycast all detected keypoints using an inverse pin hole camera model to compute the first intersection with the polygonal model as described in [3]. The intersection between the ray originating from the keypoint pixel and the polygonal model results in a precise localization of the respective keypoint in 3D space. As we only need to re-project the extracted keypoints, this is fast using BVH-tree structures [22]–[24]. The re-projected keypoints and descriptors are then combined to a so called *3D Feature* consisting of geometric (3D) and image feature (2D) information. Additionally, the geometric feature descriptor also stores other relevant local information for re-localization like face normals, ray direction and camera pose. The complete list of metrics encoded in a 3D feature is shown in Tab. I.

C. Ambiguity Filtering

To ensure that only stable and unique features are integrated into the global map, we apply a filtering step to reduce redundancies and ambiguities. For that, we compare 3D features with respect to their geometrical distance Δ_p and descriptor difference Δ_d as shown in Tab. II.

TABLE I: Information encoded in a FAPM 3D feature. It consists of 3D information from the polygonal reconstruction, OpenCV keypoint and feature descriptor.

3D Geometry	2D Feature
Point (x,y,z)	Keypoint (OpenCV)
Face (id)	Descriptor (OpenCV Mat)
Face normal (nx,ny,nz)	
Camera pose (T)	
Ray direction (rx, ry, rz)	

TABLE II: Redundancy, ambiguity and uniqueness to compare two 3D features F_i and F_j with 3D position P and descriptor D. Similarity of location and descriptor is characterized by Δ_p and Δ_d.

Location $\lvert P_i - P_j \rvert < \Delta_p$	Descriptor $\lvert D_i - D_j \rvert < \Delta_d$	Classification
true	true	Redundancy
false	true	Ambiguity
true	false	Uniqueness (Perspective)
false	false	Uniqueness

Tab. II also shows our definitions of uniqueness and redundancy. After processing an image, we match every 3D feature against the map and filter out redundant and unstable ones. If a feature has a similar descriptor and is projected to a similar 3D location as an existing feature, it is classified as redundant. It is ambiguous, if the descriptor is similar, but the localization is not unique. If it is completely unique, it is added to the global map. If it was projected to an existing 3D location, but differs in the descriptor, it is added to a list of features associated with that location, assuming the difference is due to affine distortion. Keeping such features supports the re-localization task, as it adds stability under perspective transformations. Filtering is only done in geometric and descriptor space, to make the map invariant against the image processing sequence.

This algorithm can be executed with any feature extraction algorithm and descriptor representation. For evaluation, we computed a global FAPM per feature type and parameterization as separate map layer. A layer in the FAPM is the set of all 3D features derived from a given combination of keypoint detection algorithms and descriptor. After creating all layers, we pre-compute one kd-tree [25] per FAPM layer to support fast k-nearest-neighbor search during re-localization.

D. Storage

To serialize FAPM's to permanent storage, we use the HDF5 file schema presented in [4]. In particular, each FAPM layer is handled as a face attribute layer of the polygonal mesh map that is strictly linked to only one instance of the geometry.

IV. FAPM Localization

In this section, we present FAPM-L, our approach to localize a camera in a previously built FAPM. Besides localization, FAPM-L is also able to refine an existing map by adding additional stable features detected during localization. The general layout of our processing pipeline is shown in Fig. 2 and described in detail in the following sections.

A. Feature Extraction and Parametrization

In the first step of the localization stage, we compute image features on every incoming image. The only restriction to the type of image feature is that its descriptor can be matched to a map layer. In our experiments we analyzed SIFT, SURF, ORB and AKAZE features for their compatibility with our map structure.

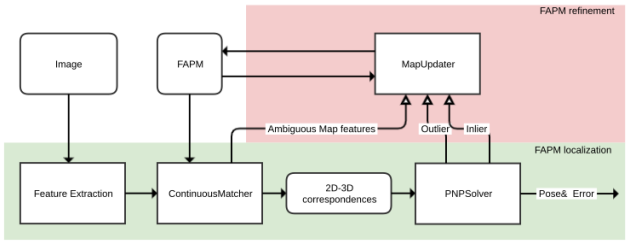

Fig. 2: The proposed FAPM-L pipeline consists of feature extraction from camera images, continuous matching against the reference FAPM and a PnP solver for localization (green). During FAPM-L, ambiguities are detected to refine the map (red).

B. Feature Matching

The main task of the *ContinuousMatcher* is to match the extracted 2D features of a camera image to the 3D features of the FAPM using the pre-computed kd-trees. The resulting set of 2D-3D feature matches is then filtered according to a maximum distance in descriptor space, to consider only matches with high similarity. 2D features that belong to the other set of matches, are features that are not yet registered in the FAPM. Initially, we reject them for matching but put them aside for later map extension. After that, we apply a second filter that compares matches of two consecutive image frames, and the matches of these two images with the FAPM respectively. The first part of matching two sets of 2D features is a commonly used technique to stabilize the matching results in an image stream. For this, we build matches between these two sets by first searching the most similar descriptors of the first image in the second image and then the other way around. If the descriptors of these matches are similar and each of this search directions lead to the same match, we consider it as stable feature match. By matching two features, we inevitably assign them to the exact same point in the world, even if the 3D coordinates of that point stay unknown. With our approach, we take advantage of the already known 3D location and descriptor of that one instance stored in the FAPM. In particular, we can discard 2D and 3D features that belong to certain impossible matching constellations. On the one hand, we are able to reject matches where the responsible pixels point to different instances in the map. On the other hand, two pixels of two images matched to the same instance in the FAPM but were not matched to each other can be rejected as well. In total, the *ContinuousMatcher* only considers matches that build a triangle as depicted in Fig. 3. This allows us to filter out more wrong matches than using only 2D to 2D filtering methods, while also tracking the matched 3D coordinates of the 2D features over time.

Besides the main task of matching, the *ContinuousMatcher* also checks if the map consists of ambiguous features. For each 2D keypoint, we search the FAPM for the k closest descriptors. Next, we iterate over the k corresponding 3D features until two consecutive 3D features meet the ambiguity condition defined in Tab. II. In this case, the *ContinuousMatcher* informs the

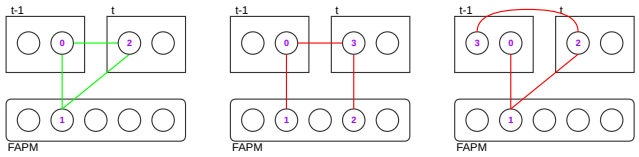

Fig. 3: Three possible combinations of matches between frame t, frame $t-1$ and the FAPM. Purple: Example set of descriptors.

MapUpdater about this ambiguity and discards the match. After that, the remaining set of matches consists of multiple 2D image pixels and 3D map points. These 3D-2D point correspondences are then used to estimate the camera's pose.

C. Pose Estimation

The next task is to estimate the pose of the calibrated camera given a set of n 3D points in the FAPM and corresponding n 2D projections in the image. OpenCV already implements solutions for such *Perspective-n-Point* (PnP) problems. They can be distinguished into analytical solutions and nonlinear optimization solvers. Analytical solutions like P3P [26] or EPNP [27] find an optimal solution directly. The nonlinear optimization solver minimizes the re-projection error iteratively and is based on the Levenberg-Marquardt algorithm [28]. Depending on the initial guess, this method tends to converge to local minima. During continuous pose estimation, we can take advantage of this characteristic. As soon as the filter algorithm fails, ambiguities or less informative features remain in the FAPM. To clarify, lets consider we have a world of two instances of one object. When pointing the camera to one of these instances, we can detect the correct object but hardly the correct instance. If we detect features in the camera image and match them to the world, this could result in matches targeting both of the world's object instances. Consequently, the minimization problem would consist of at least two possible minima. In this case, we preferably choose the local minimum near the last pose estimation over the global minimum far away.

Accordingly, setting the solving parameters of the minimization appropriately, decreases the sensitivity to false matches. Furthermore, we use RANSAC [29] to detect the remaining false correspondences that have been neglected by the previous filter strategies. It returns the set of inliers that were responsible for the resulting pose estimation as well as the outliers. Combining the number of inliers with the re-projection error, we can make assumptions about the solution's accuracy. In particular, we assume a higher pose quality with a higher number of inlier detected by RANSAC and a lower quality with a higher mean re-projection error. To asses this quality, we use the ratio $Q = N_{inlier}/\bar{E}_{repr}$. This ratio should be high for successfully matched images against the FAPM, and low for wrong matches. For our applications, we search for a Q-threshold that determines if the system is well localized. In subsequent experiments, we will empirically determine this threshold by comparing Q to the computed localization error.

In total, we propose two different methods to solve the PnP problem depending on the localization state. If the robot is localized, we use the iterative method from OpenCV with a low number of iterations. This *LocalPnPSolver* finds locally minimal solutions with respect to the previous pose estimation. Otherwise we choose the so called *GlobalPnPSolver*. It first computes the solution using an analytical solver. If the solution indicates that the robot is not localized, it performs an additional search with the iterative minimizing approach. In this case, we invest more time in finding an solution by setting a high number of iterations.

D. Map Update

The *MapUpdate* mechanism is concurrently executed to the localization process and handles all necessary map changes. Map changes can be one of the following three actions: *Add*, *Remove* and *Refine*. Some of these actions are triggered if certain conditions are met by the result of the localization stage. At the beginning of the localization process, features were filtered out, for which no similar features could be found in the map. Accordingly, this features hold information about the environment that has not been recorded in the FAPM yet. Therefore, once the pose estimation is valid, these features will be appended to a list possible additions to the map. This list of 2D features will later be added to the FAPM through Feature Projection. Furthermore, during localization, match filters detect ambiguous 3D features in the map as described in Tab. II. As long as the localization gives valid results, the RANSAC algorithm will generate a list of outlier matches, which probably are false matches. Their 3D features in the FAPM are therefore ambiguous and marked to be removed. We determine the order of removal by descendingly sorting the outlier matches by their re-projection error. Both the RANSAC outliers and the ambiguities are added to a list of possible removals. Each of the RANSAC inliers 2D features are added to the map through Feature Projection. By this, the added 3D feature should refine one existing 3D feature by adding a descriptor recorded from a slightly different perspective to the same 3D location. At this point, the *MapUpdate* stage contains a list of additions and removals. For each addition and removal the kd-tree needs to be restructured. Therefore, we first buffer those changes until we reach a pre-defined number of changes that triggers the map update. This update scheme maintains and optimizes a set of stable features in the FAPM to handle visual changes in the environment. For example, if visual elements, such as portraits or posters, are added to a particular wall, this thread extends the map with the additional features from the new objects. The other way around, our system detects removed objects as well and adjusts the map through removal of the respective features.

V. Evaluation

In this section we evaluate the previously described localization approach. For this, we use two data sets: a set of high resolution laser scans taken with a RieglVZ400i and Nikon 500d camera with 24 mega pixel resolution. Using this high

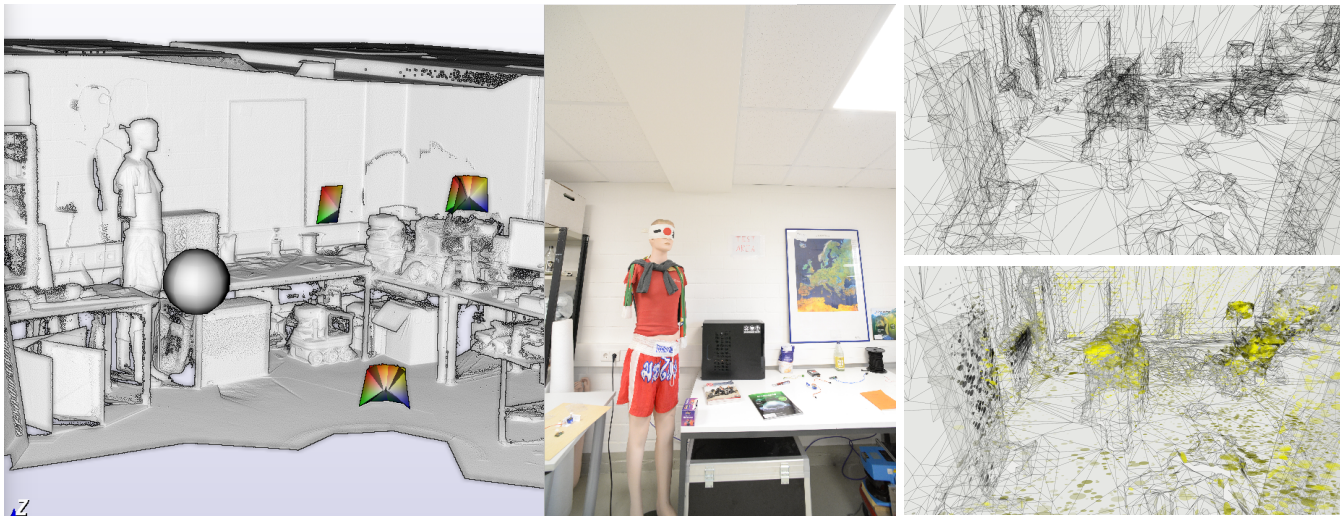

Fig. 4: Evaluated data sets. Left: The high resolution UOS data set with computed ground truth camera poses. Right: Polygonal reconstruction on the lower resolute EuRoC MAV Vicon room data set (top) and projected features (bottom). The ellipsoids of the features correspond to the face formals of the underlying triangles as part of the FAPM 3D features.

resolution reference, we investigate several feature types with respect to their reliability in global matching and analyze the runtime of our system. From that, we determine the optimal feature algorithms and parameters for the localization system. The final section describes an evaluation of the matching approach itself on the well known EuRoC MAV data sets and compares our accuracy to ORB-SLAM2 in localization mode.

A. UOS-Lab data set

For this reference data set, we registered the scans from the Riegl laser scanner manually. The laser scanner provides point clouds with millimeter precision and a precise co-calibration of the system's high resolution camera. The accurately reconstructed polygonal maps for this evaluation have been verified as described in [15] and are shown on Fig. 4. After reconstruction, we raycasted different type of keypoints and descriptors to project them on the surface in order to obtain multiple FAPM layers of the same environment. For filtering ambiguities we set the respective parameters to $\Delta_p = 0.02$ and $\Delta_d = 5$.

For the localization part, we used a camera stream recorded with a Parrot Bebop 2 UAV. We reconstructed the ground-truth camera movement trajectory by manually labeling the pixels of the scan images and the video frames that represent the same real world location carefully with pixel accuracy. Then, the pixels of each scan image label were raycasted to the mesh to determine their 3D position. In each video frame, the labeled pixels of the video stream were then matched to the same named scan image labels to obtain 3D-2D correspondences. To further reduce errors, we assured to have at least 15 correspondences per frame. For each of the reference frames, we then computed the camera's pose by solving the PnP problem. The resulting ground-truth trajectory consists of 810 poses.

B. Parameters

We used the ground-truth trajectory and the drone video infer optimal parameters for FAPM-L localization. Parameterization requires to find the optimal feature type and an intrinsic metric to assess the quality of the resulting pose. For that, we analyzed several types of image features with respect to their compatibility for matching features from cameras of different resolutions at different poses. In more detail, we evaluated, which type of image feature is most invariant against highly differing image resolutions of two completely different camera systems. In this part of the evaluation, we disabled the `MapUpdater` to evaluate the matching of the drone's images to the initial map only. To quantify their usability, we used the number of inliers per filtered matches as metric. The results for AKAZE, ORB, SIFT and SURF are shown in the left graph in Fig. 5. It indicates that AKAZE outperforms ORB, SIFT and SURF in terms of global matching in this scenario.

To analyze the accuracy of our FAPM approach on the ground truth data, we consequently used AKAZE in the following evaluation. First, we compared the reported FAPM-L poses to the ground truth poses from manual labeling. From our implementation, we obtain the of number of initial, filtered, and inlier matches as well as the mean re-projection error of the inliers. The re-projection error is computed by the Euclidean distance of the projected inlier to the corresponding pixel. In addition, we compute the quality metric Q as described in the previous section. The results are plotted in the left graph of Fig. 5.

It shows that the intrinsic quality metric Q is inversely related to the estimated pose's error: high quality relates to a low error and vise-versa. That means, we can infer the pose estimation's accuracy from Q without knowing the explicit error to the ground truth. For further experiments, we decided that global localization is "good enough" if $Q > 0.2$. Note: At

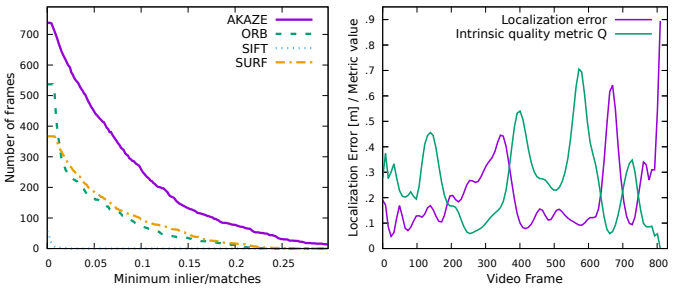

Fig. 5: The parameter evaluation consists of finding the optimal feature type and an intrinsic metric for estimating the quality of the resulting pose.

this stage, we also considered the localization error caused by computing time. Therefore, the accuracy differs to subsequent experiments.

C. Runtime

In this section we analyze the runtime for matching different image features with the FAPM as shown in Fig. 7. It illustrates that the runtime logarithmically increases with the overall size of the FAPM, which allows to represent large environments. Fig. 7 depicts the total runtime of the localization that was used in Fig. 5. Here, the FAPM consists of 73 650 3D AKAZE features. The results show that using AKAZE leads latency of 400 ms between image acquisition and localization.

D. Accuracy

In this section we compare the accuracy of FAPM-L to ORB-SLAM2 (stereo) on the EuRoC MAV data sets [6]. In particular, we use the six Vicon Room data sets, as they include 3D scans of the environment. In contrast to our manually created UOS-Lab data set, these data sets do not consist of two different sensor systems. But we are able to use the UAV's first camera combined with the ground truth estimate to produce the FAPM. Afterwards, we use the second camera to estimate the pose in the FAPM via FAPM-L. Consequently, we demonstrate to what extent the precise LiDAR measurements affects the localization accuracy instead of evaluating the quality of 2D-3D matching here.

Fig. 6 shows that successful matching against the FAPM occurs rarely. However, between these events, it is possible to extrapolate the trajectory. In our experiments, we implemented a simple camera-based pose extrapolation. After global localization, it generates a local FAPM with respect to the global estimated pose. The next frame of the camera is matched against this local FAPM to obtain a relative pose estimation with respect to the last frame and consequently a new guess. This procedure continues until the next successful global localization event. Of course, this relative localization naturally leads to high error propagation. However, the global localization with FAPM-L corrects the high drift regularly and precisely as illustrated in Fig. 6.

TABLE III: RMSE comparison between ORB-SLAM2 (stereo) and our FAPM Localization (FAPM-L). No. FAPM-L reports the number of frames, where global localization has been successful.

EuRoC data set	ORB-SLAM2 RMSE (m)	FAPM-L MSE (m)	No. FAPM-L
V101	0.035	0.0246	65
V102	0.02	0.0066	76
V103	0.048	0.0084	48
V201	0.037	0.0045	20
V202	0.035	0.0008	22
V203	-	0.0008	34

For each of the Vicon Room data sets we analyzed the accuracy of the global pose estimation through root mean squared error (RMSE) computation, as FAPM-L and the ORB-SLAM2 (stereo) are scaled correctly. The results are depicted in Tab. III. In these experiments, the global FAPM-L pose estimations within the FAPM are more accurate then ORB-SLAM2. However, the frequency of correct matches is low compared to ORB-SLAM2, which integrates all subsequent frames.

VI. Conclusion

We introduced an environmental map representation for mobile robotics called FAPM. The novelty of this representation is that we combine LiDAR-based polygonal meshes with different types of image features. The projection of the features in 3D space using raycasting and the calibration parameters of the scanning systems results in a highly accurate localization of the features in 3D space. We used these FAPMs successfully to localize a UAV equipped with a low resolution camera in them. In our experiments, we analyzed various features for compatibility with our approach. We introduced a metric that assesses the quality of the pose estimation by intrinsic information derived from the localization process. From that, we determined the parameterization for the best-evaluated AKAZE algorithm. With these, we were able to positively evaluate our method on EuRoC MAV data sets against ORB-SLAM2. The results showed that our method re-localizes with lower frequency but higher accuracy. The bottleneck in the current implementation is the comparatively high runtime. Future work will focus on improving the runtime and stability of the existing FAPM-L approach. For this, we propose to combine the relative localization of ORB-SLAM2 with our global FAPM-L appropriately. Furthermore, we plan to remove the remaining ambiguities by handling the robots state in a multi-modal probabilistic manner using particle filters.

References

[1] T. Wiemann, I. Mitschke, A. Mock, and J. Hertzberg, "Surface reconstruction from arbitrarily large point clouds," in *2018 IEEE International Conference on Robotic Computing (IRC)*, 2018, pp. 278–281.

[2] S. Pütz, T. Wiemann, M. Kleine Piening, and J. Hertzberg, "Continuous shortest path vector field navigation on 3d triangular meshes for mobile robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

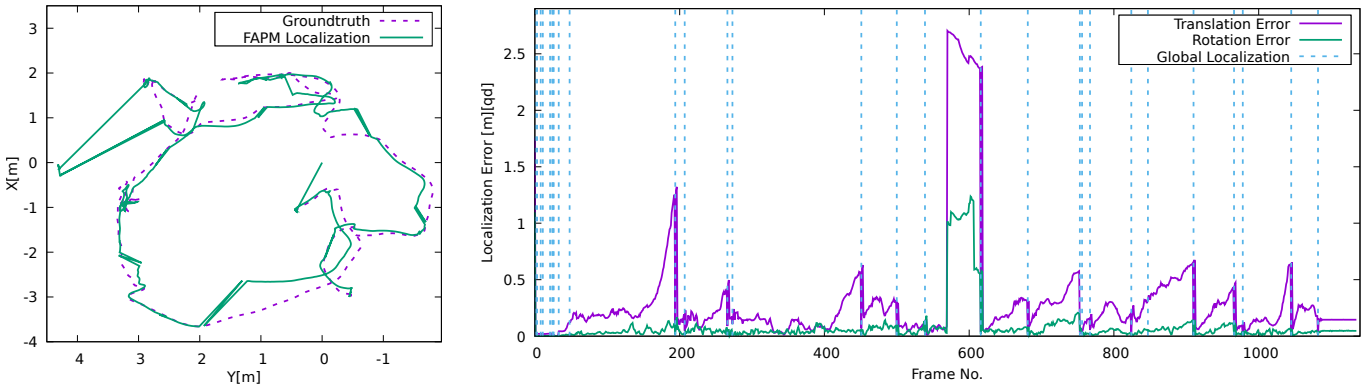

Fig. 6: FAPM localization trajectory and accuracy on the EuRoC MAV data set V201.

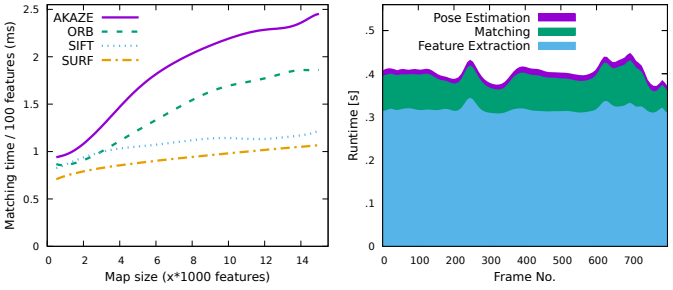

Fig. 7: Empirical analysis of the runtime to match the 5 closest features of the FAPM (left) and of FAPM localization time per frame (right) with AKAZE features using an Intel i7-7700 CPU.

[3] J. Wülfing, J. Hertzberg, K. Lingemann, A. Nüchter, T. Wiemann, and S. Stiene, "Towards real time robot 6d localization in a polygonal indoor map based on 3d tof camera data," *IFAC Proceedings Volumes*, vol. 43, no. 16, pp. 91–96, 2010.

[4] T. Wiemann, F. Igelbrink, S. Pütz, and J. Hertzberg, "A file structure and reference data set for high resolution hyperspectral 3d point clouds," *IFAC-PapersOnLine*, vol. 52, no. 8, pp. 403–408, 2019.

[5] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE transactions on robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[6] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *J. of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[7] R. Azzam, T. Taha, S. Huang, and Y. Zweiri, "Feature-based visual simultaneous localization and mapping: a survey," *SN Applied Sciences*, vol. 2, no. 2, pp. 1–24, 2020.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[9] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, Nov 2011, pp. 2564–2571.

[10] D. Galvez-López and J. D. Tardós, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, Oct 2012.

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

[12] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, no. 9, 2014.

[13] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–9.

[14] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[15] T. Wiemann, H. Annuth, K. Lingemann, and J. Hertzberg, "An extended evaluation of open source surface reconstruction software for robotic applications," *Journal of Intelligent & Robotic Systems*, vol. 77, no. 1, pp. 149–170, 2015.

[16] A. Mock, T. Wiemann, D. Borrmann, T. Igelbrink, and J. Hertzberg, "Real time texture generation in optimized large-scale polygon meshes with kinectfusion," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*. VDE, 2016, pp. 1–7.

[17] M. Feng, S. Hu, M. H. Ang, and G. H. Lee, "2d3d-matchnet: Learning to match keypoints across 2d image and 3d point cloud," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4790–4796.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[19] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[20] P. F. Alcantarilla and T. Solutions, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *IEEE Trans. Patt. Anal. Mach. Intell*, vol. 34, no. 7, pp. 1281–1298, 2011.

[21] G. Yu and J.-M. Morel, "Asift: An algorithm for fully affine invariant comparison," *Image Processing On Line*, vol. 1, pp. 11–38, 2011.

[22] D. Baldwin and M. Weber, "Fast ray-triangle intersections by coordinate transformation," *Journal of Computer Graphics Techniques (JCGT)*, vol. 5, no. 3, pp. 39–49, September 2016.

[23] R. Torres, P. J. Martín, and A. Gavilanes, "Ray casting using a roped bvh with cuda," in *Proc. 25th Spring Conference on Computer Graphics*, 2009, pp. 95–102.

[24] T. Möller and B. Trumbore, "Fast, minimum storage ray-triangle intersection," *J. Graph. Tools*, vol. 2, no. 1, pp. 21–28, Oct. 1997.

[25] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, 2014.

[26] L. Quan and Z. Lan, "Linear n-point camera pose determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 8, pp. 774–780, Aug 1999.

[27] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal of Computer Vision*, vol. 81, no. 2, p. 155, Jul 2008.

[28] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM Journal on Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.