Topomap: Topological Mapping and Navigation Based on Visual SLAM Maps

Fabian Blöchliger, Marius Fehr, Marcin Dymczyk, Thomas Schneider and Roland Siegwart

Abstract— Visual robot navigation within large-scale, semi-structured environments deals with various challenges such as computation intensive path planning algorithms or insufficient knowledge about traversable spaces. Moreover, many state-of-the-art navigation approaches only operate locally instead of gaining a more conceptual understanding of the planning objective. This limits the complexity of tasks a robot can accomplish and makes it harder to deal with uncertainties that are present in the context of real-time robotics applications.

In this work, we present *Topomap*, a framework which simplifies the navigation task by providing a map to the robot which is tailored for path planning use. This novel approach transforms a sparse feature-based map from a visual Simultaneous Localization And Mapping (SLAM) system into a three-dimensional topological map. This is done in two steps. First, we extract occupancy information directly from the noisy sparse point cloud. Then, we create a set of convex free-space clusters, which are the vertices of the topological map. We show that this representation improves the efficiency of global planning, and we provide a complete derivation of our algorithm. Planning experiments on real world datasets demonstrate that we achieve similar performance as RRT* with significantly lower computation times and storage requirements. Finally, we test our algorithm on a mobile robotic platform to prove its advantages.

I. INTRODUCTION

Mobile robots have recently left the research laboratories and are becoming more and more ubiquitous. In many of the resulting applications, including those aimed at the consumer market, navigation is a key capability. It is hard to imagine robotic vacuum cleaners or surveillance robots without at least a minimal suite of navigation and path planning skills. Another rapidly developing market are the Augmented Reality (AR) and Virtual Reality (VR) applications, where it is often expected that a mobile device will be able to guide a user to a certain location. In both of these use cases, reliable navigation within a global coordinate frame is crucial and ideally requires a minimal sensor setup and limited computational resources.

State-of-the-art navigation and path planning approaches are often based on an occupancy map representation [1]. Then, a planning algorithm, such as RRT [2] or variants thereof, can be deployed to obtain a global path within the free space of the given map. Occupancy maps, however, are usually built using either expensive laser sensors [3], RGB-D cameras [4] or computationally demanding stereo cameras [5]. Additionally, an occupancy map representation does not provide a higher-level understanding of the environment, for example division of the free space into
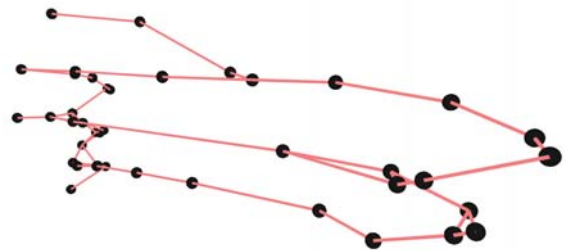
All authors are with the Autonomous Systems Lab, ETH Zurich. Contact: `fabian.bloechliger@mavt.ethz.ch`.
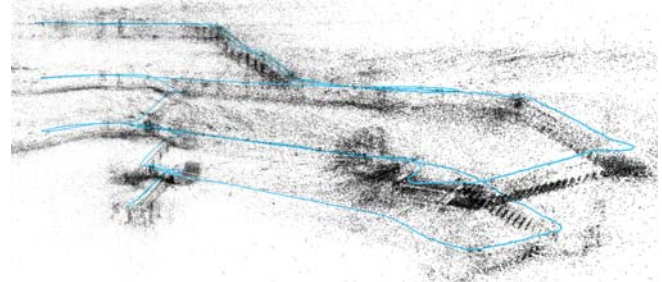
(a) Visual SLAM map.

(b) Topological map (convex voxel clusters).

(c) Navigation graph.

Fig. 1: Our proposed topological mapping approach applied to a multi-floor indoor scene: (a) The input to our framework is a sparse visual SLAM map of the environment with $423'000$ triangulated 3D landmarks. The landmark positions are estimated based on the multi-view geometry and the feature tracking pipeline of a visual-inertial odometry estimator. The length of the trajectory is $360\,\mathrm{m}$. (b) Using free space information extracted from the landmarks, we build a topological map consisting of convex voxel clusters (vertices) and their adjacent areas (edges), which we call *portals*. The vertices denote free, traversable areas within the environment. (c) The derived navigation graph makes global path planning within the explored environment easy and computationally inexpensive.

separable parts (e.g. rooms in a building). Finally, planning and navigation using occupancy maps is a computationally demanding problem and can be challenging in the presence of noisy input data. This limits the capabilities of many mobile platforms.

This work introduces *Topomap*, a lightweight topological mapping and navigation approach which addresses the afore-mentioned disadvantages. It represents a complete and simple solution for autonomous navigation within large-scale visual SLAM maps, where topological maps are derived directly from triangulated 3D positions of the visual landmarks. This implicitly leads to a tight coupling of the localization and navigation map. Our approach solely relies on a standard monocular camera that is lightweight, small and can easily be placed on most robotic platforms. The working principle of our framework is demonstrated in Fig. 1. We divide the environment into a set of free space clusters which correspond to the vertices of the topological map. We enforce the convexity of their shape and as a result, a robot can cross each of those regions without the risk of running into a static obstacle. On the other hand, the portals (adjacency regions) are the places where a safe transition from one to another vertex is possible.

Having divided our free space into a set of convex clusters, we can derive two graphs which are shown in Fig. 2. The topological graph (Fig. 2a) holds the connectivity of the convex voxel clusters (vertices), whereas its dual graph, the navigation graph (Fig. 2b), can be used for path planning by any graph based planning algorithm. To the best of our knowledge, *Topomap* is the first system which is designed to extract free space from sparse visual features in order to create a topological map representation of the environment. By using sparse features, efficient 3D structures and a simple navigation concept, our algorithm can be deployed on mobile platforms with limited computational resources.

The contributions of this work can be summarized as follows:

- We propose to use a sparse visual SLAM map to create a reliable free-space representation and a subsequent topological map of the area.
- We present an entire processing pipeline that takes a visual map as input and creates a topological map that can be used by a global planner.
- We propose an algorithm that employs – based on a volumetric occupancy grid – voxel cluster growing and merging to generate convex free space clusters from noisy and partly incomplete visual SLAM data.
- We present an extensive evaluation of the framework using real life datasets with different topological charac-teristics and compare our navigation concept to a state-of-the-art grid based planner.

II. RELATED WORK

Topological mapping combined with vision sensors has a long tradition in mobile robotics [6]. In the context of SLAM, there are multiple reasons to partition an environment into a

(a) Topological graph (connectivity of clusters).
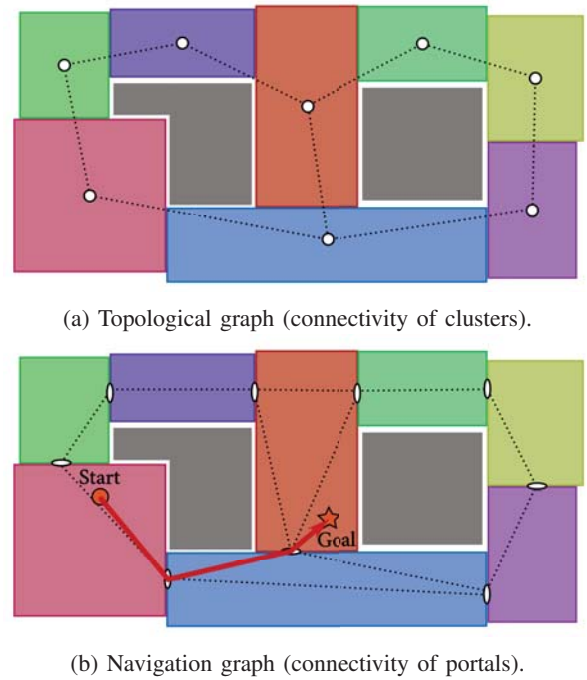

(b) Navigation graph (connectivity of portals).

Fig. 2: The two basic elements of the topological map: (a) The *topological graph* contains the relation between the convex free space clusters (vertices): All adjacent vertices are connected by a topological edge, which indicates that a robot can directly move between the corresponding ver-tices. (b) The *navigation graph* is the dual graph of the topological graph. It is obtained by connecting all portals of each topological vertex. In order to keep the navigation approach simple, we only use the centers of the portals in our experiments. An example of an A-to-B path planning task is shown.

number of discrete places: Past works include topometric lo-calization [7], the use of a hierarchical bundle adjustment [8] or map reduction purposes [9].

One idea related to our proposed algorithm is to construct topological maps on top of 2D grid-based maps by dividing the free space into disjoint regions. The regions are delimited by narrow passages derived from the environment's Voronoi decomposition [10], which can also be done in an incremen-tal fashion [11]. However, we believe that applying Voronoi diagrams in the context of visual SLAM maps is challenging, as we do not have accurate 2D/3D laser maps, but noisy and sparse 3D landmarks.

Another widely spread idea is to divide a map into mean-ingful keyframe or landmark clusters based on a similarity measure, e.g. landmark co-visibility [12]–[15]. This approach might result in meaningful clusters from a human point of view (e.g., entering a new room results in a new group of keyframes), but co-visibility clusters have no concept of free space or convexity, which we believe to be key components for efficient path planning. Enforcing convex free space clusters guarantees that the robot can move freely within each cluster. Furthermore, a typically low viewpoint

invariance of visual features will tend to create two regions of a single place which has been traversed from different directions when using co-visibility based methods.

The third group of approaches to topological mapping is attaching local occupancy grids at different places along the metric SLAM map [16]. The key part of those algorithms is the strategy to select places to store the grid. This can be done, for example, by using fixed size, overlapping cubes [17]. These approaches may help to capture the topology of an environment. However, they only partially simplify the navigation process as a local path needs to be planned through the topological vertices, whereas we have a strong prior assumption about the convex shape of the vertices.

The works which are closest to ours but more targeted towards simplifying polynomial trajectory generation for MAVs are [18] and [19], which generate large overlapping convex regions [20] and compute a path through this regions which can be followed by a MAV. Instead of using many overlapping clusters, we propose to use a compact expansion step in the cluster creation to capture more of the local free space and reduce the total number of clusters. Furthermore, our voxel based cluster creation algorithm allows a seamless integration with discrete occupancy maps from real world data.

III. METHODOLOGY

This section introduces the methodology of the proposed algorithm. In Section III-A, the topological map representation is explained. Then, in Section III-B we describe in detail how we can derive free space information solely from sparse visual SLAM features. In the next step, presented in Section III-C, we grow compact, convex free space clusters using the information computed beforehand. Neighboring clusters are then merged if their combined shape includes a low number of obstacle voxels, as described in Section III-D. This reduces the number of clusters and results in a topological map whose vertices are convex free space regions. In Section III-E, we show how to use the obtained topological map for global path planning.

A. Topological Map Representation

We propose to construct a topological map by segmenting the free space of the entire environment into a a set of convex regions (topological vertices). As a result, each vertex corresponds to a certain partially enclosed area within the environment (e.g. a room) which is connected to neighboring vertices by portals. This topological map representation resembles the way humans perceive the environment when navigating [21], and is likewise convenient from a robot's planning perspective.

The convex regions are represented as clusters of voxels, which means that they can be directly derived from voxel based occupancy maps. The occupancy maps, however, do not need to be stored, which significantly reduces the storage requirements compared to state-of-the-art approaches. Instead, we only serialize the convex hull of the regions

corresponding to the topological vertices. This way, the useful information of our topological map is preserved (vertex volume, topological connections and portals). The serialized data is sufficient to deduce in which region a given position is located.

B. Occupancy from Sparse Features

Topomap extracts discrete approximate free space information from sparse landmarks by using voxel based Truncated Signed Distance Fields [22]. TSDFs are commonly used in combination with depth sensors (e.g. laser rangefinders or densified multi-camera setups) [23]. We will, however, focus on using sparse visual SLAM features for this purpose.

Our first steps will be analogous to fusing depth measurements into a volumetric TSDF grid using the traditional sensor modalities, i.e. by ray tracing the 3D grid from sensor origin to the measured 3D point. Hence each triangulated 3D landmark present in the SLAM map is ray traced from its observer pose. The distance values in all traversed voxels are updated according to the distance to the landmark up to a predefined maximum distance value (truncation distance). Ray tracing observations of all 3D landmarks will result in a voxel map which contains projective distances to obstacle surfaces, which is in fact only an approximation to the real distance. The TSDF construction step is provided by the volumetric mapping library *voxblox* [24].

The TSDF representation based on noisy and sparse visual features requires some additional post-processing to obtain reliable information about the voxel occupancy. First of all, we binarize the information by thresholding the distance value of each voxel (we chose 90% of the truncation distance). Secondly, we propose a subsequent filtering step which removes small occupied voxel groups which are not connected to any other occupied part. These outliers might come from dynamic objects while building the SLAM map or badly triangulated landmarks.

C. Compact Cluster Growing

The next step in the *Topomap* pipeline is to grow a set of compact, convex clusters based on the binarized TSDF reconstruction of the environment. Enforcing a compact growing leads to a sphere-like expansion of the clusters (i.e. a similar spread in all directions) instead of making it dependent on the cubic shape of the voxels or the orientation of the voxel grid.

The cluster growing algorithm initializes a first cluster center voxel at a random position along the explorer trajectory, because there we have the highest certainty about free space. Then we start an iterative growing of the initialized cluster, which consists of three main steps (Fig. 3):

1) Find all directly adjacent non-occupied voxels for the current voxel cluster.
2) Perform Principal Component Analysis and find the principal axes of the current cluster shape that contain most of the currently included voxels (we use the threshold of 98%), assuming an ellipsoidal shape for the clusters. If the smallest half axis of this ellipse has a
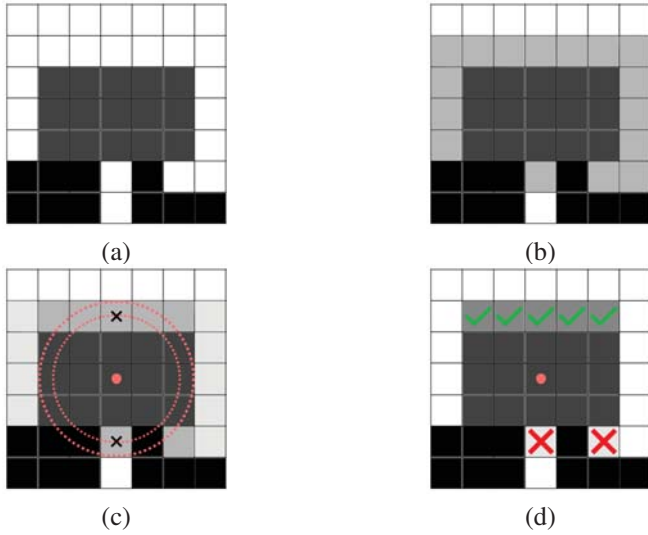
Fig. 3: Outline of one step of the cluster growing algorithm. (a) Cluster shape of the current iteration. (b) Find direct neighbors of the cluster. (c) Choose all neighbors which make the cluster compact. (d) Only keep the voxels that preserve the convexity.

length of $r_{\min}$, we only allow voxels up to a distance of $r_{\min} + \delta$ to the cluster center to be added to the cluster (compact candidates).

3) Only add these compact candidates which ensure that the convex hull of the current cluster does not include any obstacle voxels. This is done by checking if the rays from all candidate voxels to the cluster voxels are obstacle-free.

The growing of a single cluster terminates after no more voxels have been added. This means that there are either no more compact candidates found or that the existing cluster is no longer convex when adding these candidates. After this, we start growing the other clusters in the same way until the whole explorer trajectory is contained within the clustered space.

D. Convex Cluster Merging

The algorithm described in the previous section yields a relatively high number of compact, convex regions which are conservative in the sense that they do not include any obstacle voxels and have a similar expansion in all directions. We now introduce a second step, where regions are merged if the convex hull of the combined region contains a low number of obstacle voxels. Our algorithm is inspired by *dynamic region merging* [25], which first over-segments an image into small and conservative regions (superpixels), and then merges similar segments iteratively. This two-step procedure makes the final segmentation less prone to noisy input data, as the merging step is based on a good but over-conservative pre-segmentation of the input pixels/voxels.

The proposed algorithm is iterative and repeatedly attempts to merge neighboring candidate pairs of clusters. In each iteration, it starts by searching for all merge candidates, that is pairs of clusters which are directly adjacent. Then, it iterates through all tentative cluster merge pairs in a randomized fashion. For each cluster pair, it computes the combined convex hull of this pair using the Quickhull algorithm [26]. The cluster pair is merged if the relative number of contained occupied voxels (i.e. obstacles) within this convex hull is smaller than some set *obstacle ratio threshold* (we set it to 1-5% in our experiments). Increasing this value will lead to a lower number of clusters, but will indeed leave more responsibility to a local planner. We elaborate on the influence of this parameter in Section IV-B. The procedure is terminated when no more merges were performed based on the merging criterion.

E. Topological Navigation

After building a convex cluster representation of the environment, we can proceed with the topological navigation. We interpret the convex clusters as the vertices of our topological graph. Similarly, the topological edges are created whenever two clusters are adjacent (portals). Fig. 2b outlines our approach of path planning on a topological map.

Let us consider a simple A-to-B path planning case: both A and B have to be located within the clusters as we have no information about the space outside of them. We start by building a navigation graph by connecting the portal centers of all topological vertices. Additionally we connect A and B to the portal centers of their corresponding topological vertices. In order to get the shortest path from A to B based on this graph, we can perform an A* search. The path planning algorithm would then plan a path where the agent moves from A to the portal, then starts traversing intermediate clusters until it reaches the cluster that contains B, where it can move directly from the portal to the desired destination.

IV. EXPERIMENTAL EVALUATION

In this section, the evaluation of three main parts of *Topomap* and the results of the entire framework are presented. Section IV-A evaluates the performance of TSDF integration based on sparse visual 3D landmarks and compares it to the approach based on dense reconstruction using a stereo camera. It indicates the potential compromises we are making by avoiding the use of any expensive hardware or significant computational power. Then, we demonstrate the performance of the cluster creation algorithm on multiple real world environments in Section IV-B. Finally, Section IV-C compares our topological planner to RRT*, a state-of-the-art planner that is commonly used within the robotics community.

We acquired our datasets using a synchronized visual-inertial sensor [27] and the experiments were performed using a dual-core Intel i7-7600U (2.8 GHz) processor. The SLAM maps are built by first running a visual-inertial odometry pipeline based on [28] and then running global bundle adjustment and loop closure using *maplab*, an open source mapping framework [29].
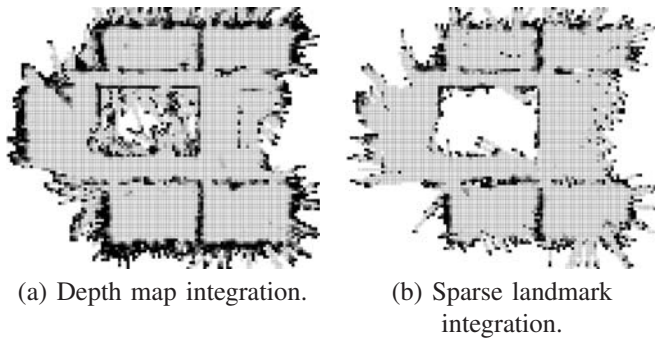
(a) Depth map integration. (b) Sparse landmark integration.

Fig. 4: Slice of a 3D TSDF reconstruction of an office environment. The voxel size is $0.25\,\text{m}$. Integrating the 1322 stereo images takes $43.3\,\text{s}$ and produces a TSDF map of $65'231$ voxels, whereas the integration of the $177'269$ landmarks lasts for $2.1\,\text{s}$ and the corresponding map contains $39'791$ voxels. Observe how using sparse landmarks for TSDF construction preserves most of the relevant environment structure.

A. TSDF Maps from Visual Landmarks

In the previous sections we proposed ray tracing free space from sparse SLAM landmarks that significantly reduces computational requirements and simplifies the sensor setup, but it might affect the quality of the TSDF reconstruction. Capturing both the free space and occupied areas correctly is essential for a reliable creation of the convex clusters and navigation in the subsequent steps of our algorithm. We would therefore like to evaluate the TSDF maps of the proposed approach and compare them with TSDF maps that were created from dense stereo images. By employing the semi-global matching algorithm [30] implemented in OpenCV [31], we get dense depth images from the $10\,\text{cm}$ baseline stereo camera. These are integrated into a TSDF from their corresponding observer poses in the same way as the 3D landmarks (see Section III-B) using the identical integration parameters. The most important parameters are the maximum ray length and the truncation distance, which we set to $4.0\text{-}7.0\,\text{m}$ and $0.1\text{-}0.5\,\text{m}$, respectively.

A qualitative comparison between sparse visual landmarks and depth maps to create a TSDF map is presented in Fig. 4. Even if the TSDF map based on the landmarks is sparser and partially occupied by outlier voxels in some of the free space areas, it still contains the relevant structure which can be inferred from the dense map (walls, corridors, doorways). In fact, the *Topomap* framework handles outliers and missing information in the subsequent stages by filtering out small connected components in the occupancy map and by enforcing convexity for all added voxels during the growing process. Convexity prevents clusters from growing through or around obstacles (e.g. walls) even if some occupancy information is missing.

In the next step, we want to evaluate quantitatively how much free space can be captured using the sparse visual landmarks compared to the dense maps generated from
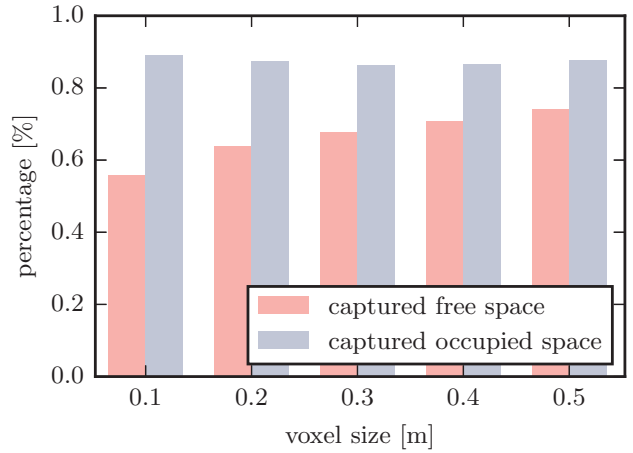

Fig. 5: Influence of the voxel size on the amount of captured free/occupied space in the sparse TSDF map. The dense TSDF map is taken as a reference: For each free voxel in the dense map, we check if the corresponding voxel in the sparse map is free as well. In order to get the captured occupied space, we also take into account unmapped space in the sparse map. This is reasonable as during the voxel growing, unmapped space is treated as occupied space.

a stereo rig. Here, we take the dense TSDF map as a reference. Obviously, this is only an approximation, but should give a good insight about how well the landmark integration performs in real world scenarios. Fig. 5 shows the percentage of captured free and occupied space for five voxel resolutions. More free space can be captured if we increase the voxel size, but at the same time, we will also get larger discretization errors (e.g. small obstacles will not be captured in the occupancy map). Evidently, this latter effect is not captured in the shown plot, as the dense TSDF map suffers from the same effects.

B. Topological Map Creation

In this section, we want to evaluate the core part of *Topomap* – growing and merging of the voxel clusters. Fig. 6 shows the cluster arrangement after the growing step and the subsequent iterations of the merging algorithm in a warehouse environment. While the cluster growing step expands the clusters within the free space, the merging procedure leads to a substantial complexity reduction of the topological map structure. It is worth to emphasize how the algorithm captures the complex topology of this environment by combining multiple small compact clusters into larger ones along corridors, but preserves a more fine-grained clustering in the corners.

To give more insights into the results of the clustering algorithm, we also showcase the output from *Topomap* for three different datasets in Fig. 7: The *office* dataset constitutes a typical example of a structured environment, which is segmented into clusters in a similar fashion to what a human would typically do (separation into *rooms* and *corridors*). The second example, *open space*, is more demanding as

Fig. 6: Iterative cluster merging demonstrated in a *warehouse* setting. Starting with a large number of small and compact clusters, we merge cluster pairs which contain only a low number of obstacle voxels within their combined convex hull. The initial 105 clusters are reduced to 24 clusters within 3 merging steps, and the number of topological edges is reduced from 121 to 29. In this example, we set the obstacle ratio threshold to 5%.

(a) Office dataset, 20 m trajectory

(b) Open space, 50 m trajectory
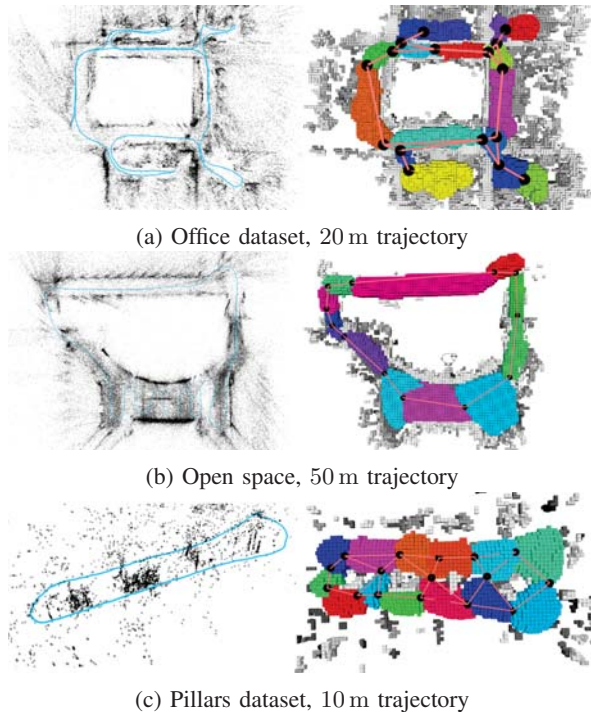

(c) Pillars dataset, 10 m trajectory

Fig. 7: SLAM maps (left) and their corresponding topological maps (right). (a) Office: Typical topological mapping dataset consisting of narrow corridors and clearly separated rooms. (b) Open space: The large open spaces of this dataset are correctly represented by convex clusters. (c) Pillars: The ray traced empty space between the clusters is sufficient to introduce topological edges and enable path planning in these areas where the explorer never traversed.
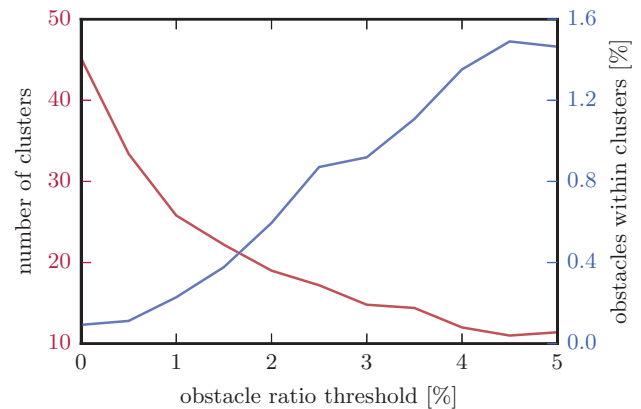

Fig. 8: An evaluation of the obstacle ratio threshold parameter for the *open space* dataset. As expected, a higher value of this parameter leads to a lower number of final clusters (red curve) as many of them are merged. On the other hand, the blue curve expresses how many obstacle voxels are contained within the voxel clusters. It increases if we leverage the condition of contained obstacle voxels.

it consists of large open spaces at the one hand and of some more narrow passages on the other (top part of the map). Note how our topological map representation succeeds to simplify the map given the winding explorer trajectory. Clearly, the merging algorithm did well by creating rather large clusters in the open space part of the dataset. Finally, the *pillars* dataset highlights how *Topomap* can infer traversable areas that cannot be deduced purely from the explorer trajectory. This means that a robot that uses this topological map could use the passages in between the pillars even if they have never been traversed by the explorer.

This would not be possible for a teach-and-repeat navigation strategy.

The *Topomap* algorithm does not require a large set of environment specific parameters or tedious fine-tuning for a specific use case. We would like, however, to highlight a parameter that affects both the robustness towards outliers and the accuracy of the topological map. Fig. 8 demonstrates the influence of the *obstacle ratio threshold* (introduced in Section III-D) on the segmentation result. This parameter gives us control over the following trade-off: Larger values of the *obstacle ratio threshold* will reduce the complexity of the topological map by accepting small obstacles to be present within the merged clusters. These violations of the requirement that clusters are completely empty shifts part of the planning responsibility to the local planner and requires a local collision avoidance algorithm to circumvent these obstacles within a cluster. Keeping the values of this parameter low will have an opposite effect – only few obstacle voxels will be allowed within the clusters which will lead to a large number of small clusters. It will therefore be

TABLE I: Timing and storage requirement statistics of the *Topomap* pipeline for the datasets evaluated in this paper. The volume within the brackets shows the volume of all mapped voxels within the TSDF map. All voxel sizes were set to $0.25\,\text{m}$. The storage requirements of our approach are significantly reduced when compared to the full TSDF.

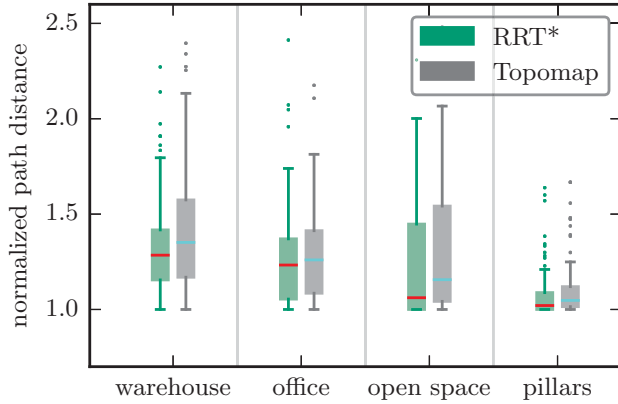|	computation time [s]		storage requirements [kB]		
	TSDF	clusters	full clusters	convex hulls	TSDF
multifloor ($2209\,\text{m}^3$)	4.1	86.1	567	249	4320
warehouse ($895\,\text{m}^3$)	0.4	11.2	147	110	1784
office ($622\,\text{m}^3$)	2.1	9.9	123	73	1162
open space ($1464\,\text{m}^3$)	2.7	42	379	150	2932
pillars ($269\,\text{m}^3$)	0.46	3.1	51.9	62.7	607

Fig. 9: A comparison of the normalized path distances of RRT* operating on a TSDF map from dense stereo images and our proposed topological planner (100 runs per dataset). The path distances are normalized by the direct line distance. Our lightweight topological planner generates paths just marginally longer than RRT*, but requires much simpler computations (A* on a small navigation graph).

harder to tune this parameter within large spaces where we encounter varying obstacle/free space configurations.

We claim that *Topomap* is particularly useful in the applications that put constraints on the computational power or limit the available perception hardware. Special care was taken to guarantee that the framework uses a limited amount of resources, including the storage requirements of the topological map. Table I summarizes some statistics about the topological map creation for the datasets presented throughout this paper. Having the SLAM map as an input to our system, we could create topological maps for most of these datasets within less than a minute. The storage requirements for the topological map are low as only the convex hulls of the voxel clusters are stored. Obviously, this compact representation has also the potential to over-simplify the environment and we may lose important details in some cases.

C. Path Planning

The proposed topological mapping concept is primarily targeting navigation and path planning. Below, we present an evaluation of the entire pipeline that includes both the

Fig. 10: Setup of the Turtlebot experiments. The VI sensor [27] is used for global localization within the SLAM map, and a laser is used for local obstacle avoidance only using the dynamic window approach [33].

assessment of the generated paths as well as a deployment of the system on an real robotic platform.

First, we compare *Topomap* to the RRT* planner from OMPL [32], which is provided a TSDF map from stereo images. We sample 100 trajectories with random start and goal positions for both planners and compare the path distances normalized by the direct line distance (see Fig. 9). The planning time of the RRT* planner is set to $2\,\text{s}$, which led to successful paths given the complexity of our environments and the voxel resolution. In general, the path lengths generated by the topological planner are slightly longer, but at the same time, our A* planning time is drastically lower than for RRT* (typically in the order of some $100\,\mu\text{s}$). This directly corresponds to our goal to replace demanding algorithms with a lightweight counterpart with only a marginal quality loss.

Secondly, we have integrated our proposed navigation system on a Turtlebot robot equipped with a VI sensor, and successfully performed path planning tasks within a semi-structured industrial site. In a first step, the robot performed global localization within the visual SLAM map using the approach described in [34], and was then given a target position. The topological planner then computed the fastest path to the given location, and Turtlebot successfully completed a trajectory of approximately $15\,\text{m}$, using a 2D laser for local obstacle avoidance only. These experiments proved the usefulness of our system on a mobile platform equipped only with a camera and a computationally constrained processing unit. A video footage demonstrating the Turtlebot experiment is provided as a supplementary material to this paper[1].

V. CONCLUSIONS

In this paper, we have presented *Topomap*, a novel framework for creating versatile topological maps and reliable navigation therein. Our approach can handle noisy and sparse visual measurements, which significantly reduces the hardware requirements when compared with state-of-the-art approaches. The core component of the proposed system is a voxel based growing and merging algorithm, which segments the free space into convex clusters. This enables path planning algorithms that are orders of magnitude faster

[1]https://youtu.be/UokjxSLTcd0

than conventional grid based planners. Additionally, the chosen structure of the topological map makes the resulting maps very compact.

The evaluations of *Topomap* demonstrate that it is possible to reliably build TSDF maps from sparse vision-based measurements. We also have proved that the results of the framework do not exhibit any significant quality loss when compared to RRT* operating on a dense TSDF map. Finally, the system was successfully deployed on a mobile robotic platform. We believe the results of this work will be interesting for everyone working on the navigation of mobile platforms within large-scale environments, and where the computational resources, size or weight are limited.

For future work, we plan to evaluate our approach on some larger real world scenarios and we would like to integrate it on a flying platform, which would exploit the full 3D capabilities of this framework. Furthermore, we intend to include semantic information in the topological maps, as this has the potential of pushing boundaries of robotic autonomy even further.

REFERENCES

[1] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[2] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep., 1998.

[3] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[4] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, "3-d mapping with an rgb-d camera," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177–187, 2014.

[5] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1872–1878.

[6] D. Kortenkamp and T. Weymouth, "Topological mapping for mobile robots using a combination of sonar and vision sensing," in *AAAI*, vol. 94, 1994, pp. 979–984.

[7] H. Badino, D. Huber, and T. Kanade, "Visual topometric localization," in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2011, pp. 794–799.

[8] J. Lim, J.-M. Frahm, and M. Pollefeys, "Online Environment Mapping using Metric-topological Maps," *The International Journal of Robotics Research*, pp. 1–15, 2012.

[9] M. Dymczyk, S. Lynen, M. Bosse, and R. Siegwart, "Keep it brief: Scalable creation of compressed localization maps," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 2536–2542.

[10] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, pp. 21–71, 1998.

[11] M. Liu, F. Colas, L. Oth, and R. Siegwart, "Incremental topological segmentation for semi-structured environments using discretized GVG," *Autonomous Robots*, 2014.

[12] Z. Zivkovic, B. Bakker, and B. Krose, "Hierarchical map building using visual landmarks and geometric constraints," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2480–2485, 2005.

[13] J. L. Blanco, J. Gonzalez, and J. A. Fernández-Madrigal, "Consistent observation grouping for generating metric-topological maps that improves robot localization," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2006, pp. 818–823.

[14] F. Fraundorfer, C. Engels, and D. Nister, "Topological mapping, localization and navigation using image collections," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3872–3877, 2007.

[15] R. Vázquez-Martín, P. Nunez, A. Bandera, and F. Sandoval, "Spectral clustering for feature-based metric maps partitioning in a hybrid mapping framework," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009, pp. 4175–4181.

[16] K. Konolige, E. Marder-Eppstein, and B. Marthi, "Navigation in Hybrid Metric Topological Maps," *ICRA*, pp. 3041–3047, 2011.

[17] P. Schmuck, S. A. Scherer, and A. Zell, "Hybrid Metric-Topological 3D Occupancy Grid Maps for Large-scale Mapping," *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 230–235, 2016.

[18] R. Deits and R. Tedrake, "Efficient mixed-integer planning for uavs in cluttered environments," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 42–49.

[19] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, pp. 1688–1695, 2017.

[20] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 109–124.

[21] K. Lynch, *The image of the city*. MIT press, 1960, vol. 11.

[22] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[23] H. Oleynikova, A. Millane, Z. Taylor, E. Galceran, J. Nieto, and R. Siegwart, "Signed Distance Fields: A Natural Representation for Both Mapping and Planning," in *Robotics: Science and Systems*, 2016.

[24] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[25] B. Peng, L. Zhang, and D. Zhang, "Automatic image segmentation by dynamic region merging," *IEEE Transactions on Image Processing*, vol. 20, no. 12, pp. 3592–3605, 2011.

[26] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The quickhull algorithm for convex hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, Dec. 1996.

[27] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 431–437.

[28] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visualinertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[29] T. Schneider, M. T. Dymczyk, M. Fehr, K. Egger, S. Lynen, I. Gilitschenski, and R. Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, 2018.

[30] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[31] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

[32] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[33] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[34] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-intertial localization," *Robotics: Science and Systems (RSS) XI*, pp. 37–46, 2015.