Signed Distance Fields: A Natural Representation for Both Mapping and Planning

Helen Oleynikova, Alex Millane, Zachary Taylor, Enric Galceran, Juan Nieto and Roland Siegwart
Autonomous Systems Lab, ETH Zürich

Abstract—How to represent a map of the environment is a key question of robotics. In this paper, we focus on suggesting a representation well-suited for online map building from vision-based data and online planning in 3D. We propose to combine a commonly-used representation in computer graphics and surface reconstruction, projective Truncated Signed Distance Field (TSDF), with a representation frequently used for collision checking and collision costs in planning, Euclidean Signed Distance Field (ESDF), and validate this combined approach in simulation. We argue that this type of map is better-suited for robotic applications than existing representations.

I. INTRODUCTION

Any discussion of map representations in robotics generally has two sides: the perception side, which often focuses on creating high-quality 3D reconstructions of natural environments from limited sensor data, and the planning side, which uses pre-built maps to navigate through the environment in a safe and collision-free manner.

Mapping and planning often have very different requirements from an environmental representation. For mapping from the perception standpoint, it is often most important to be able to output a high-quality colored surface model, such as a mesh. For navigation and planning, it is often most essential to have fast collision checking and be able to compute clearance and direction toward nearest obstacles.

In this work, we focus on the intersection of these two fields, with special attention given to image-based sensing, such as stereo vision and RGB-D cameras: creating 3D maps, online, from noisy sensor data in order to be used by online planners for obstacle avoidance.

Euclidean Signed Distance Fields (ESDFs) have long been used in planning literature for collision checking (especially of complex shapes), inferring distances and gradients to objects for planning, and finding large free areas [1].

On the other hand, with the advent of RGB-D cameras, KinectFusion has brought projective Truncated Signed Distance Fields (TSDFs) into the forefront as a fast, flexible map representation that implicitly computes the position of the surface using zero crossings [2].

Though both of these representations are signed distance fields (SDFs), the way that the distance of a voxel is computed differs. In the case of the ESDFs, a free voxel’s distance represents the Euclidean distance to the nearest occupied voxel (and vice-versa in the case of occupied voxels). The ESDF is computed for every voxel in the map. On the other hand, the distance of a voxel in a projective TSDF represents the distance to the surface along the ray direction from the center of the sensor, and is truncated to only have values very near the surface, allowing for greater compression and decreasing errors due to this approximate distance metric.

In this paper, we argue that these two approaches can be combined into one – and rather than computing separate representations for map building and planning, find a compromise that allows the same representation to be used for both.

We seek to evaluate the accuracy of ray-distance calculations versus Euclidean distance calculations given different numbers of viewpoints from a realistic vision-based sensor. We first remove the truncation requirement of the TSDF, and then evaluate how the ray distances compare to true Euclidean distances for planning purposes, and propose a hybrid approach that should retain the better qualities of both TSDFs and ESDFs for both mapping and planning.

We also offer some comparison to the most-used map representation for online 3D map building and planning in unstructured environments: Octomap [3].

The advantages of our proposed approach over map representations typically used for online planning are as follows:

- Allows a single map representation that’s both well-suited for online construction from sensor data and usage for online planning.
- Has a more natural representation of object surface (zero-crossing of distance field), which allows easier and more accurate modeling of sensor noise.
- Separates sensor measurement (distance to surface) from measurement uncertainty (voxel weight).
- Allows fast look-up of occupancies of complex shapes, especially when represented as sets of spheres, for online planning.
- For optimization-based planning algorithms, has a natural gradient magnitude and direction both inside and outside obstacles – allowing trajectories to be ‘pushed’ out of collision using optimization.
- Allows extracting accurate meshed surface models for other applications.
- If desired, the same map can also be used for SLAM or state estimation, or can be built up decoupled based on input from other state estimators.

We will cover the points in the following sections. First we will describe existing methods in both the mapping and planning fields, then discuss the advantages of using an SDF-based representation over an occupancy-based representation.
like Octomap, and finally provide validation for the claim that
the distances used in these two representations are comparable
using simulation.

II. RELATED WORK

This section discusses relevant previous work and details of
the approaches in mapping and planning literature.

A. Mapping Literature

SDFs have long been used for representing 3D volumes in
computer graphics [4] [5]. They have also been used to build
offline reconstructions of objects from real sensor data since the
1990s [6].

However, TSDFs have come back into the forefront of com-
puter vision and robotics in 2011 with the new RGB-D Kinect
sensors and the work from Newcombe et al. on KinectFusion
[2]. Their approach focuses on doing high-resolution, accurate
3D reconstructions from RGB-D data on a GPU in real-time,
using a TSDF as the main representation. They provide a
Simultaneous Localization and Mapping (SLAM) approach, which
estimates the pose of the camera at the same time as
creating the reconstruction.

There have been a number of extensions to this approach,
including Kintinuous, which allows scanning much larger
spaces [7], and FastFusion which allows online reconstruction
The main end-result of these is to generate a high-fidelity
mesh, usually using a marching cubes algorithm [9].

Another competing representation for high-resolution online
mapping from RGB-D data is RGB-D SLAM [10], which uses
surfels (small planar units with size, color, and surface normal)
to represent 3D structure, as a volumetric analogue to sparse
pointclouds. Such representations are able to take advantages
of the SLAM techniques developed for sparse keypoint-based
maps and are better suited to distorting geometry, for exam-
ple in ElasticFusion [11] where the map is distorted when
encountering loop closures. These methods generally have
high accuracy for state estimation and high-quality models.
However, Bylow et al. have shown that it is possible to have
the same level of accuracy from TSDF-based maps [12].

Therefore, SDF-based representations are well-studied, fast,
and accurate for online surface reconstruction. In the following
section, we will discuss how SDF-based representations are
used for planning.

B. Planning Literature

Maps are essential to planning collision-free paths, with
the representation of the map defining both the quality of the
resulting path, and also what kind of planning algorithms can
be used.

The minimum amount of information a map must provide
is occupancy of a given point in space. Assuming a fixed-size
grid, this enables the use of many different classes of planning
algorithm: search-based methods like A* and D*-Lite [13],
sampling-based methods like RRTs [14].

Occupancy grids represent the most commonly-used type
of map representation for planning in 2D [15]. The approach
of Elfes et al. is to use a fixed-size grid, probabilistic model
of sensor measurements and model observed (known) and
unknown space explicitly, allowing the incorporation of complex
sensor models and reasoning about the environment. There
are now many options available off-the-shelf that will run a
complete 2D SLAM system online and provide occupancy
grids of previously unknown environments [16], and countless
planning algorithms that take 2D occupancy grids as input
[17].

Naively extending occupancy grids to 3D, however, leads
to huge memory requirements as well as slow ray-casts and
look-ups for any space larger than a room. The solution
most commonly used in 3D contexts while building a map
online is Octomap [3]. This approach uses an octree-based
representation of occupancy probabilities of cells in 3D space.
The octree structure allows large blocks of space with the same
probability to be represented by a single large cell, therefore
vastly decreasing the amount of memory needed to represent
areas of unknown or free space.

However, there are planning approaches which require
additional information from a map. For example, trajectory
optimization-based planners, such as CHOMP [11] and Tra-
jOpt [13] require the distance to obstacles and occupancy
gradient information. Algorithms such as these require an
ESDF that is not truncated, and contains distance values over
the entire voxel space. Usually these are constructed from
another map representation, and often from a map hand-crafted
out of object primitives (spheres, cubes) or high-fidelity mesh
models of objects for manipulation [19].

Having a distance map also speeds up collision checking
of complex shapes – for example, many-jointed robot arms
are commonly represented as a set of overlapping spheres and
check the distance field in the center of each sphere (which is
one look-up in the ESDF per sphere) [20] [11] [21].

For gradient-based trajectory optimization methods, the
collision cost (which is necessary to produce collision-free
trajectories) also needs a gradient. For these, the ESDF gives
a natural cost (a function, such as hinge loss [18] or a smoothed
hinge loss [11] of the distance) and checking the distance values
of the neighbors gives the gradient at a given point. This allows
CHOMP and other such methods to follow the upward gradient
of the distance to push points on the trajectory out of collision.

Wanger et al [22] is the closest work to our proposed
approach, where a complete ESDF is built from the output
of KinectFusion [2] and used for trajectory planning with
CHOMP. However, their approach builds the entire ESDF at
once from TSDF data, while our work focuses on exploring
ways to combine these two representations into one.

In summary, SDFs allow for faster collision checking than
occupancy grids while providing additional data needed for
optimization-based planning methods.

III. SDF ADVANTAGES OVER OCTOMAP

In this section, we make arguments about why using an SDF
is a better map representation for both perception (creating
the map) and planning (using the map for collision avoidance
Fig. 1: Comparison of results from a single ray-cast in the various representations discussed: (a) diagram of a single vision-based sensor ray hitting a surface, (b) probability for range from an inverted stereo sensor model [23], (c) vanilla Octomap [3] probabilities of the raycast, (d) L. Heng Weighing [24], (e) TSDF (truncation radius not shown) [2], and (f) the TSDF weights along this ray [12]. Note that these are illustrations, and not to scale.

and clearance calculations) than the most commonly used 3D representation, the Octomap [3].

Since its advent, Octomap has been very widely used for 3D robotics applications, most notably for UAVs [24], [25]. We believe that this is due to a number of factors: first and not least, the open-source implementation and associated ROS wrappers have made it a very easy off-the-shelf solution for many applications. Second, the ‘probabilistic’ nature of the representation (assigning probabilities to each raycast, merging multiple observations of the same scene together) make it a good representation for noisy sensor data, such as stereo matching or RGB-D sensors where ‘speckles’ are common. This adds a level of low-pass filtering even to sensors exhibiting non-Gaussian error models. The third is due to memory efficiency and speed: the flexible voxel size allows representing large areas, and with some straight-forward optimizations, it is possible to get the insertion time of a dense stereo scan at 320x240 down to approximately 10 ms, and performing collision checks (even for a large bounding box) in this space is also very fast [24].

However, this representation also has downsides. The first is that the probability model used does not accurately represent the error model of vision-based depth sensing. Since Octomap was originally designed to use with laser measurements, the accuracy of which does not degrade with distance to the sensor, the Octomap sensor model has a single probability of occupancy for one voxel at the end of the ray-cast. However, this is not an accurate model for stereo- or other vision-based sensing, where it is possible to have an expected error of over a meter at high distances, depending on the camera setup [23]. Heng et al implement a more realistic sensor model using distance weighing in the Octomap, however this tends to have the effect of inflating obstacles in the map [26].

Fig. 1 shows a representation of the different weighing representations for a single 1D ray within Octomap, compared to a TSDF representation. Fig. 1a shows a diagram of a ray cast from a depth camera hitting a wall, Fig. 1b shows the inverse sensor model for a stereo camera observing that wall, and Fig. 1-c–d show vanilla and L. Heng weighing for the sensor hit. Since Octomap has a discrete cut-off probability for considering space occupied, the figure shows why L. Heng weighting tends to distort or inflate object boundaries.

One advantage of the TSDF is that even when discretized, it models a continuous function, as shown in Fig. 1e. Therefore it is possible to recover the position of the surface at a precision above the minimum voxel size, allowing the use of larger voxels and therefore smaller maps in memory.

The other advantage over Octomap is that TSDF has two values for each voxel: the distance to the surface (along the ray from the camera) and the weight/probability of this measurement. This allows us to more accurately model the actual error of vision-based depth estimates, and when merging multiple measurements, leads to a maximum-likelihood estimate of the surface, since the surface is found as a zero crossing. Bylow et al evaluate different weighing functions for TSDFs [12], and we show the linear weighing used by KinectFusion in Fig. 1f. However, since this value is separate from the actual distance measurement, any model can be incorporated without necessarily inflating the surface.
While the advantages for perception are clear, an SDF-based implementation has advantages in terms of path planning as well. Here we discuss the advantages of an E/ESDF (using Euclidean distances to nearest occupied/unoccupied space) over a binary occupancy-based representation. We will discuss how we can combine the TSDF and E/ESDF in the following section.

As discussed in Section [I-B], an E/ESDF allows fast collision checks for complex shapes, as long as they can be expressed as a set of overlapping spheres. It also permits using gradient-based methods, as it gives a smooth cost of collision, which decreases as an object approaches free space. This also allows computation of collision cost gradients, and therefore choosing directions which lead to decreasing costs.

IV. COMBINING E/ESDF AND TSDF: RESULTS

The main difference between the two representations, TSDF for mapping and E/ESDF for planning, is the way that distances are computed. Both are signed distance fields, as the names suggest, but the distance in each voxel represents a different quantity.

In the E/ESDF, the distance in each voxel is the Euclidean distance to the nearest occupied cell (or if inside an object, distance to the nearest free cell). In the TSDF, on the other hand, the distance is computed along the sensor ray – that is, it represents a distance to the nearest occupied cell not in Euclidean space, but along this one-dimensional ray extending from the sensor center.

Here we present models showing how different ray distances (TSDF) are versus Euclidean distances (ESDF) and how this affects the quality of the map for planning. We focus on evaluating the metrics that are important for planning, especially for local optimization-based planners: error in distance to obstacles and direction of the gradient of the field.

We suggest four strategies to evaluate. The first two are building a projecting TSDF without a minimum truncation distance with two different strategies for merging multiple scans into the map:
1. Average the sensor value with the map value, giving equal weight to both (average weighing)
2. Take the minimum distance outside obstacles, and the maximum distance inside (minimum weighing)

We also propose a hybrid E/TSDF, where outside some minimum truncation distance, we iteratively compute Euclidean distances based on the projective distances inside the truncation radius with every new integrated scan. Finally, we also present the 'standard' approach for comparison, where we compute an occupancy grid and calculate Euclidean distances from that grid after building the complete map.

We use a vastly simplified model of the TSDF in 2D and a simulated sensor with a 70° field of view, 0.5° angular resolution, and a maximum range of 8m. There is no noise on this sensor, but it is discretized to the voxel size. The sample environment attempts to mimic an enclosed office space of $10m \times 10m$ with many occlusions, some geometric and some organic shapes. Fig. 2a shows the occupancy map we use for the simulations and Fig. 2b shows the ground-truth E/ESDF computed from this known map, where white is the object borders, pink is inside the objects, and green and yellow are distances (yellow being the furthest from an obstacle).

We generated random viewpoints within this space, sampling uniformly from unoccupied space and yaw. We then cast rays into this map, and generated distance measurements until 0.5m behind a surface. When raycasting into unknown space, the value was simply updated to the ray measurement value. When raycasting into space that has been observed before, we use the strategies described above.

Fig. 2c shows the TSDF generated with average weighing, and Fig. 2d shows the results of minimum weighing after 500 viewpoints. As can be seen, average weighing always overestimates the distance – as ray-distance is always greater than or equal to true Euclidean distance (it is equal in the

[{2a}](b) ESDF (ground truth)

([{2b}](d) TSDF (minimum weighing)

([{2c}](c) TSDF (average weighing)

([{2d}](d) TSDF (average weighing)

([{2e}](e) Hybrid T/ESDF

([{2f}](f) Occupancy + ESDF

Fig. 2: Results of various weighing schemes in SDF, using 500 viewpoints. White represents the observed surface of the object (zero crossings), green and yellow are areas outside the object, and pink are areas inside the object. Quantitative comparison is presented in Table [I].

[Table I](#)
case where the ray direction is along the surface normal, and greater in every other case). Whereas minimum weighing, over a large number of viewpoints, starts to converge to the true ESDF.

Of course, with fewer viewpoints, the estimate is worse – Table I shows the mean absolute error of distance measurements (taken only outside obstacles) over all observed voxels for different number of viewpoints. \( r \) is the radius around the surface at which errors are evaluated; \( r = 0.5m \) is an estimate of the errors close to the surface, and \( r = \infty \) is evaluated over the whole map. The values are compared to the ground-truth ESDF shown in Fig. [23]. However, this is not a strictly fair comparison, as all the viewpoint generated maps have discretization errors and not all obstacles are observed (for a lower number of viewpoints). We therefore also present the results for “Occupancy + ESDF”, which was generated by creating a standard occupancy map from the sensor measurements and computing an ESDF from that map. It can be thought of as a lower bound on the error (though in some cases it actually has slightly higher error than other representations, which is due to discretization).

The other method we evaluate is a hybrid T/ESDF, which functions by behaving like a TSDF around surface edges, holding this surface section fixed and iteratively updating all other values using Euclidean distances. This retains the desirable quantities of both representations – near the surface for mapping and surface reconstruction, and further away from the surface for planning, while adding only slight computation cost per new viewpoint. Table I shows that this hybrid approach has the lowest errors of the TSDF-based approaches, and is comparable to the ESDF map built from occupancy grids.

This shows that depending on the weighing scheme, with sufficient distinct viewpoints, ray-distance approximates Euclidean distance, and there exist hybrid approaches that can further increase the accuracy of these combined maps.

V. CONCLUSIONS

In this paper, we compare two signed distance field representations: truncated signed distance fields (TSDFs), used for computer graphics and surface reconstruction from depth data, and Euclidean signed distance fields (ESDFs), used in planning for fast collision checking and cost and gradient information for optimization-based path planners.

We show the advantages of SDF-based maps for online map building and online planning in 3D compared to the commonly-used Octomap representation [3] and validate some of our claims by showing that projective ray-distances (used in TSDFs) can approximate Euclidean distances (used in ESDFs) when using sufficient viewpoints and an intelligent merging strategy. We also propose a hybrid approach which has advantages of both ESDFs and TSDFs.

We hope that this work can be a starting point for considering different map representations for online mapping for planning and navigation in 3D.

TABLE I: Error analysis of different TSDF representations compared to ESDF. Results of the simulating a 2D SDF with a realistic sensor, comparing the error in using the TSDF distance (distance along the sensor ray) and comparing the error to the true ESDF (distance from nearest object). \( r \) is the radius around the surface at which errors are evaluated: \( r = 0.5 \) only evaluates distances close to the surface, while \( r = \infty \) evaluates all distances in the map. Since below 500 viewpoints, not all parts of the map have been observed, we also give a Unobserved Ratio for reference.

<table>
<thead>
<tr>
<th>Viewpoints</th>
<th>TSDF Average Weighing</th>
<th>TSDF Min Weighing</th>
<th>Hybrid E/TSDF</th>
<th>Occupancy + ESDF</th>
<th>Unobserved Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>V = 10</td>
<td>0.699686</td>
<td>1.022263</td>
<td>0.501408</td>
<td>0.854449</td>
<td>0.134561</td>
</tr>
<tr>
<td>V = 50</td>
<td>0.730469</td>
<td>1.099407</td>
<td>0.317534</td>
<td>0.374416</td>
<td>0.067819</td>
</tr>
<tr>
<td>V = 100</td>
<td>0.717782</td>
<td>0.972221</td>
<td>0.157599</td>
<td>0.197076</td>
<td>0.042314</td>
</tr>
<tr>
<td>V = 500</td>
<td>0.628532</td>
<td>0.788858</td>
<td>0.031576</td>
<td>0.031494</td>
<td>0.015214</td>
</tr>
<tr>
<td>Gradient Error Mag. [m]</td>
<td>0.198165</td>
<td>0.014947</td>
<td>0.030577</td>
<td>0.004643</td>
<td></td>
</tr>
<tr>
<td>Gradient Error Ang. [°]</td>
<td>6.065860</td>
<td>8.699359</td>
<td>9.735610</td>
<td>8.699359</td>
<td></td>
</tr>
</tbody>
</table>

REFERENCES