Robotics: Science and Systems 2023
Daegu, Republic of Korea, July 10-July 14, 2023

Efficient volumetric mapping of multi-scale environments using wavelet-based compression

Victor Reijgwart, Cesar Cadena, Roland Siegwart and Lionel Ott
Autonomous Systems Lab, ETH Zürich, Switzerland.

***Abstract*—Volumetric maps are widely used in robotics due to their desirable properties in applications such as path planning, exploration, and manipulation. Constant advances in mapping technologies are needed to keep up with the improvements in sensor technology, generating increasingly vast amounts of precise measurements. Handling this data in a computationally and memory-efficient manner is paramount to representing the environment at the desired scales and resolutions. In this work, we express the desirable properties of a volumetric mapping framework through the lens of multi-resolution analysis. This shows that wavelets are a natural foundation for hierarchical and multi-resolution volumetric mapping. Based on this insight we design an efficient mapping system that uses wavelet decomposition. The efficiency of the system enables the use of uncertainty-aware sensor models, improving the quality of the maps. Experiments on both synthetic and real-world data provide mapping accuracy and runtime performance comparisons with state-of-the-art methods on both RGB-D and 3D LiDAR data. The framework is open-sourced to allow the robotics community at large to explore this approach.**

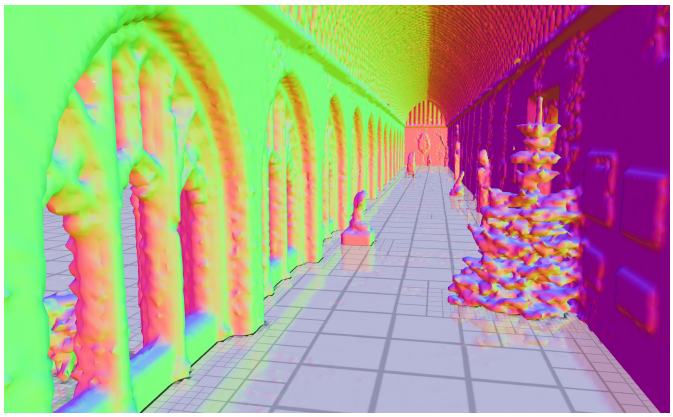

Fig. 1. A reconstruction created by our proposed hierarchical volumetric mapping framework, *wavemap*, highlighting its ability to accurately capture fine objects while also efficiently compressing free space as shown by the adaptive resolution along the transparent slice.

I. Introduction

As robots move from tightly controlled spaces into our everyday lives, there is a growing need for them to autonomously navigate and work in increasingly large, unstructured, and unknown environments. For reliable deployments and robust operation over extended periods of time, robots need to build and maintain their representation of the world using only onboard sensing and computing. Doing this in a timely manner on compute restricted devices using sensors producing large amounts of data is a continual challenge in robotics.

Dense geometric environment representations are widely used to facilitate tasks ranging from navigation to inspection and manipulation, while also serving as building blocks for other representations. Robotics is a particularly challenging field for such representations, due to the demands placed on systems with limited computational resources. For example, building a map of an unknown environment while localizing in it with Simultaneous Localization And Mapping (SLAM) requires the ability to update the map incrementally at interactive rates. To support high-level tasks such as exploration and navigation the representation must also differentiate between unknown space and observed (free or occupied) space. Finally, the map must be able to model arbitrary geometry with sufficient accuracy to guarantee safety when unexpected environmental structures or objects are encountered.

Volumetric map representations can be updated incrementally and explicitly represent unknown space. Furthermore, if a sufficiently high resolution is chosen, they can also represent object surfaces and unknown space boundaries of arbitrary topology. Beyond robotics, volumetric representations are commonly used in 3D reconstruction, reality capture, and augmented reality applications. However, a major drawback of volumetric representations is that their memory usage in naive implementations grows linearly with the observed volume and cubically with the chosen resolution. Several research efforts propose to use multi-resolution representations, often based on trees, and demonstrate significant improvements. In this work, we extend these efforts by approaching the problem from a formal signal processing and data compression perspective. Specifically, we propose to use wavelet compression to obtain a hierarchical volumetric representation. Using Haar wavelets we achieve state of the art lossless compression performance, while also allowing simple yet efficient updates and queries. This is achieved by compressing the occupancy information using a Haar wavelet decomposition and storing the individual decomposition components in a hierarchical data structure. The wavelet transform's linearity makes it possible to perform measurement updates directly in the map's compressed representation. Furthermore, when performing map updates we know that all resolution levels of the map are always up to date and in a valid state due to the Haar basis' orthogonality property. This obviates the need to perform maintenance operations or manual compression passes that are typically seen in other multi-resolution mapping frameworks.

Another trade-off made by many existing volumetric mapping methods is the reliance on simplified measurement models to achieve real-time update rates. A common approach is to use discrete occupancy updates, that systematically inflate obstacles and do not allow for surfaces to be reconstructed with sub-voxel accuracy. Measurement models based on Truncated Signed Distance Functions (TSDFs) overcome the latter limitation but use a projective distance heuristic. Such approaches have a hard time reconstructing thin objects such as branches, cables, or fences. Furthermore, the implied assumption of infinitely thin rays, underlying these observation models, leads to aliasing artifacts in regions where the ray density is low compared to the voxel resolution. In addition to negatively affecting the reconstruction quality, the resulting high entropy regions are hard to compress. Besides alleviating the challenges mentioned above, modeling soft beams provides an opportunity to incorporate angular uncertainties from sensor calibration and pose estimation into the volumetric reconstruction process. Thanks to the computational benefits of the Haar wavelet representation we can adopt a continuous occupancy measurement model, accounting for angular and range uncertainty, inspired by the work of [16].

In order to process data at sensor rate we introduce a specialized measurement integration algorithm that exploits a hierarchical measurement update approach with the information provided by the map itself. The proposed algorithm speeds up measurement integration while guaranteeing that the results are identical to a naive integrator applying the same measurement updates at the highest resolution throughout the field of view.

In summary, the main contribution of this paper is a volumetric mapping system that uses:

- A wavelet-based hierarchical representation, that is guaranteed to keep the hierarchy consistent at all times;
- A continuous occupancy measurement model accounting for range and angular uncertainties;
- A highly-efficient coarse-to-fine measurement integrator that adapts to the observed structure;

The proposed framework is extensively evaluated on synthetic and real-world datasets with comparisons to several state-of-the-art methods. The results demonstrate that our approach is memory efficient yet produces high-quality maps, all while being computationally efficient. The entire framework is open-sourced[1] to enable the robotics community to build on these results.

II. Related work

A. Map model

Two approaches are commonly used to represent maps in robotics [3], sparse feature-based maps and dense maps. The first category uses sparse sets of distinctive features [21, 31] and excels at representing large environments but struggles to model the connectivity of surfaces and distinguish between free and unknown space. This makes it ideal for large scale mapping and localization tasks, but limits its use for manipulation, motion planning, and exploration tasks. The second paradigm uses a large number of geometric elements, such as as points [8, 28], surfels [1, 12, 26, 32, 38], or meshes [37] to model observed obstacles. Voxels, discretizing the space into squares or cubes of fixed size, are another common geometric primitive used to model both occupancy [7, 13] and signed distance information [4, 14, 15, 25].

B. Measurement model

Approximations of the sensor's physical operation have been widely explored. Early approaches modeled uncertainties of the sensors explicitly [7]. Other approaches aim to achieve specific map properties, such as sharp map boundaries [16]. However, when building 3D maps using precise sensors the computational cost incurred by these sensor models motivated the development of simpler ray-based models. These models treat observations as thin rays tracing through the world [13, 25]. Machine-learning based methods exploit more complex relationships, such as inverse rendering [9, 19] or beam-to-beam interactions [24].

C. Map storage

The most common way to store volumetric maps is to discretize the space using a voxelgrid, i.e. a regular grid with fixed size voxels. In the beginning grids with a single fixed resolution [7, 14] were used, but over time spatial data structures, such as hashed voxel blocks [23], trees [13], or hybrids thereof [36] were adopted. These structures fit the observed volume more tightly, can grow dynamically, and improve runtime. To model expansive maps with varying levels of detail, multi-resolution maps [6, 11, 13, 34, 35] are widely used due to being memory efficient and capable of adapting to the needed resolution. Many multi-resolution representations are also hierarchical, allowing users to query the map at varying resolutions [11, 13]. Taking a signal processing perspective on compact map storage leads to the use of wavelet transforms [40], which are inherently multi-resolution and hierarchical, or the discrete cosine transform [29]. Recent learning-based methods, such as NERF [19] or occupancy prediction networks [18, 27], take a different approach and learn the coefficients of a neural network that predicts map information at arbitrary coordinates.

D. Map updates

The manner in which maps are updated with new observations is crucial for the efficiency and map quality. Early volumetric frameworks evaluated the measurement model for all voxels in the observed volume [7, 14]. This was improved by tracing rays from the sensor's center to each measured point and updating the voxels that are intersected by the ray [13, 25]. Advances in sensor technology, enabling high resolution maps spurred further efficiency improvements, such as ray-tracers that bundle [25] or sub-sample [6, 20] similar rays, or rate limit voxel updates [20]. While efficient, these integrators can produce "holes" in the map depending on the sensor's ray density.

[1] https://github.com/ethz-asl/wavemap

This motivates the use of projective integrators which avoid this problem by interpolating the depth image [4, 16]. Other approaches to avoid resolution-related issues include multi-resolution integrators, ray-tracing [6] or projective [35], which reduce the update resolution with distance, as well as methods analyzing the measurement update regularity [10, 11]. While efficient, hierarchical volumetric maps require maintenance to keep the information in the different levels coherent. Octomap [13] employs a fine-to-coarse scheme, integrating measurements at the finest resolution and synchronizing coarser levels in a maintenance pass. Supereight [11, 35] performs multi-resolution updates and synchronizes the remaining levels using an upward and downward propagation scheme.

In contrast to others, our volumetric ray-tracing method uses a wavelet decomposition-based representation which implicitly synchronizes all hierarchy levels at once. Additionally, unlike most ray-tracing methods we use a continuous sensor model, taking angular uncertainty into account, to improve map accuracy.

III. Multi-Resolution Analysis and Wavelets

Multi-resolution representations have been the subject of intensive study by communities ranging from computer vision [2] to physics and mathematics [5, 17]. Mallat and Meyer formalized the expected properties of multi-resolution representations as the Multi-Resolution Analysis (MRA) conditions [17]. The full MRA conditions are summarized in appendix A. In informal terms, they state that increasing the resolution should only add detail and eventually make it possible to represent any signal. Two further requirements are self-similarity in space and in scale. In a mapping context, these imply that the map should behave the same regardless of our frame of reference and choice of units.

A corollary of the fact that increasing the resolution only adds information is that, in areas that are stored at multiple resolutions, the lower resolutions do not carry any unique information and storing them explicitly is redundant. This motivates the use of wavelet decompositions, which allow us to work with maps that form valid MRAs while only storing and processing the differences between the resolution levels. A given wavelet decomposition is characterized by its chosen scaling function and complementary wavelet function. In this work, we focus on the Haar wavelet and scaling function, which form an orthogonal basis. A summary of orthogonal wavelet bases is provided in appendix B. This orthogonality is particularly beneficial because it guarantees that any given volumetric map is characterized by a unique combination of wavelet coefficients. Thus, there are no redundant coefficients that can go out of sync and manually have to be updated after integrating new measurements. Another interesting property of Haar wavelets is that the basis resulting from its scaling functions correspond to box functions arranged to span the cells of a regular grid. Therefore, Haar decompositions can represent anything a regular grid map can, while bringing significant benefits in terms of compression and implicitly maintaining the hierarchy's consistency.

IV. Method

In the following, we describe the components of our approach. We first explain how the map's occupancy posterior can be efficiently updated in its compressed state, thanks to the properties of the wavelet transform. Next, we derive our continuous sensor model, which captures range and angular uncertainties associated with the measurements. After that, we derive an error bound which enables early stopping during the coarse-to-fine observation integration process. Further performance improvements are obtained by skipping updates that do not change the state of the map. Finally, we illustrate how all these pieces fit together with an algorithmic overview.

A. Measurement integration

In the following we will explain how the use of wavelets enables efficient measurement integration. As each new beam endpoint measurement $\mathbf{z}$ arrives, the map's Bayesian occupancy posterior $p(m_\mathbf{x}|\mathbf{z}_{1:t})$, estimated at each point $\mathbf{x}$ in the map m, can incrementally be updated using

$$\mathcal{L}_p(m_\mathbf{x}|\mathbf{z}_{1:t}) = \mathcal{L}_p(m_\mathbf{x}|\mathbf{z}_{1:t-1}) + \mathcal{L}_s(m_\mathbf{x}|\mathbf{z}_t), \tag{1}$$

where $s(m_\mathbf{x}|\mathbf{z}_t)$ is the sensor's inverse measurement model and the log-odds formulation, $\mathcal{L}_p = \log \frac{p}{1-p}$, is used to make the update linear. As the wavelet transform $\mathcal{W}$ is also linear, the update equation for all cells in the map becomes:

$$\mathcal{W}\left(\mathcal{L}_p(m|\mathbf{z}_{1:t})\right) = \mathcal{W}\left(\mathcal{L}_p(m|\mathbf{z}_{1:t-1})\right) + \mathcal{W}\left(\mathcal{L}_s(m|\mathbf{z}_t)\right). \tag{2}$$

Therefore, once the compressed measurement update $\mathcal{W}\left(\mathcal{L}_s(m|\mathbf{z}_t)\right)$ is computed, the map can be updated directly in wavelet space. This is avoids the costly process, employed by other methods, of decompressing the map's observed area, applying the update, and compressing the map again. Computing $\mathcal{W}\left(\mathcal{L}_s(m|\mathbf{z}_t)\right)$ is efficient thanks to the Fast Wavelet Transform (FWT) (Appendix C), which is typically initialized by computing the orthogonal projection of the original signal onto the scaling functions at a pre-determined finest resolution.

Since the wavelet transform itself is lossless, the reconstruction error is fully determined by how well the initial FWT projection approximates the original update. Most applications use a constant finest resolution, but this is not mandatory. Given that inverse sensor models tend to be smooth throughout most of the observed volume, only raising the resolution close to surfaces would improve efficiency and the maximum achievable detail.

B. Measurement models

In order to derive multi-resolution sampling and integration approaches, it is important that the chosen inverse measurement model $s(m_\mathbf{x}|\mathbf{z}_t)$ is well-defined at all points $\mathbf{x}$ in the observed volume. We propose to extend the continuous occupancy model introduced in [16] by modeling the angular uncertainty of each measured beam, in addition to range uncertainty. We model the probability of occupancy $s(m_\mathbf{x}|\mathbf{z})$ at a point $\mathbf{x}$ for a single beam $\mathbf{z}$ by correlating the probability of occupancy given the beam's true endpoint $\bar{s}(m_\mathbf{x}|\bar{\mathbf{z}})$ with

the distribution of the true endpoint position given a noisy observation $o(\bar{\mathbf{z}}|\mathbf{z})$, i.e.:

$$s(m_{\mathbf{x}}|\mathbf{z}) = \int_{\mathbb{S}} \bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}}) o(\bar{\mathbf{z}}|\mathbf{z}) d\bar{\mathbf{z}}, \tag{3}$$

where $\mathbf{x}$, $\bar{\mathbf{z}}$, and $\mathbf{z}$ are expressed in sensor coordinate space $\mathbb{S}$, and the beam start point is at its origin. Extending [16] to include angular uncertainty, we define $\bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}})$ as

$$\begin{aligned}\bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}}) &= \bar{s}(m_{\mathbf{x}}|\bar{z}_r, \bar{z}_\theta) \\ &= \begin{cases} 0 & x_r < \bar{z}_r \ \wedge \ |x_\theta - \bar{z}_\theta| \leq \tau_\theta \\ 1 & \bar{z}_r \leq x_r \leq \bar{z}_r + \tau_r \ \wedge \ |x_\theta - \bar{z}_\theta| \leq \tau_\theta \\ \frac{1}{2} & \text{otherwise} \end{cases}\end{aligned} \tag{4}$$

where τ is an assumed surface thickness parameter in sensor coordinates, see Fig. 2a for a visualization. The subscript r refers to the axis perpendicular to the sensor's image plane, whereas θ refers to the offset along the image plane[2].

Our model assumes that the noise on the measurement beam endpoint position is normally distributed in sensor coordinates, as

$$o(\mathbf{z}|\bar{\mathbf{z}}) \sim \mathcal{N}(\bar{\mathbf{z}}, \Sigma), \tag{5}$$

where Σ is the measurement noise covariance matrix. If Σ is diagonal and $\bar{\mathbf{z}}$ has a uniform prior, the r and θ components are independent and eq. 5 can be simplified as follows:

$$o(\bar{\mathbf{z}}|\mathbf{z}) = o(z_r|\bar{z}_r) o(z_\theta|\bar{z}_\theta) = \mathcal{N}(\bar{z}_r, \sigma_r)\mathcal{N}(\bar{z}_\theta, \sigma_\theta). \tag{6}$$

We approximate the normal distributions with quadratic B-splines, as in [16], such that $o(z|\bar{z}) \simeq q(\frac{\bar{z}-z}{\sigma})$, where

$$q(t) = \begin{cases} \frac{1}{16}(3+t)^2 & -3 \leq t \leq -1 \\ \frac{1}{8}(3-t^2) & -1 < t < 1 \\ \frac{1}{16}(3-t)^2 & 1 \leq t \leq 3 \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

The distribution of the true beam endpoint position given a noisy measurement (Fig. 2b) then becomes:

$$o(\bar{\mathbf{z}}|\mathbf{z}) = q\left(\frac{z_r - \bar{z}_r}{\sigma_r}\right) q\left(\frac{z_\theta - \bar{z}_\theta}{\sigma_\theta}\right). \tag{8}$$

As motivated in [16], we match the surface thicknesses to half the width of their respective B-splines, i.e. $\tau_r = 3\sigma_r$ and $\tau_\theta = 3\sigma_\theta$. This ensures that the measurement model is continuous and that $\mathcal{L}_p(m_{\mathbf{x}}|\mathbf{z}_{1:t})$ converges to 0 if $\mathbf{x}$ lies on an object's surface.

After substituting 4 and 8 into 3, the full inverse measurement model (Fig. 2c) becomes:

$$\begin{aligned} s(m_{\mathbf{x}}|\mathbf{z}) &= \int_0^\infty \int_{-\infty}^\infty \bar{s}(m_{\mathbf{x}}|\bar{z}_r, \bar{z}_\theta) q(v) q(w) d\bar{z}_\theta \, d\bar{z}_r \\ &= \frac{1}{2} + \left(Q(v) - \frac{Q(v-3)}{2} - \frac{1}{2}\right)\Big(Q(w+3) - Q(w-3)\Big), \end{aligned} \tag{9}$$

[2]For pinhole camera projection models, r corresponds to the depth coordinate and θ to the reprojection error. For spherical projection models, e.g. certain LiDARs, r refers to the range coordinate and θ to the relative angle.

where $Q(t)$ refers to the cumulative distribution of $q(t)$, i.e. the cubic B-splines resulting from $Q(t) = \int_{-\infty}^t q(u)du$, and $v = \frac{z_r - \bar{z}_r}{\sigma_r}$, $w = \frac{z_\theta - \bar{z}_\theta}{\sigma_\theta}$.

For depth cameras, the depth uncertainty is often set as $\sigma_r(x) = \kappa_r x_r^2$, where κ_r depends on the sensor setup and post-processing algorithms. For laser-based sensors, the range error is usually assumed to not vary with range, thus $\sigma_r = \kappa_r$ where κ_r is indicated on the sensor's datasheet.

C. Worst-case update error bounds

From the MRA theory (Section III) we know that at some point, integrating information at finer levels of the hierarchy no longer improves the representation. Therefore, to fully exploit the coarse-to-fine measurement integration scheme of our method we need to know at what level of the hierarchy we can stop integrating data. This requires determining, for each point $\mathbf{x}$, the resolution beyond which no further improvements are possible, which we achieve by deriving a conservative approximation error bound. As this work focuses on the use of Haar wavelets, we can exploit a property unique to them, namely that neighbors at the same resolution do not overlap. This results in the leaves of our multi-resolution Haar decomposition perfectly partitioning the original space into non-overlapping cubes of varying sizes. Since Haar scaling functions are constant over their support, the worst-case error ϵ_{max} within each space partition, or voxel, $\mathcal{V}$ is given by:

$$\epsilon_{\text{max}}(\mathcal{L}_s(m, \mathbf{z}), \mathcal{V}) = \max_{\mathbf{x} \in \mathcal{V}} \left|\mathcal{L}_s(m_{\mathbf{x}'}, \mathbf{z}) - \mathcal{L}_s(m_{\mathbf{x}}, \mathbf{z})\right|, \tag{10}$$

where $\mathbf{x}'$ is the chosen sample point, which we set to be the partition's center.

Since ϵ_{max} has to be evaluated millions of times per second in practice, we simplify the computation by only considering three cases based on the state of the space partition, defined as follows:

$$\text{update\_type}(\mathcal{V}, \mathbf{z}_t) = \begin{cases} \text{FullyUnobserved} & \forall \mathbf{x} \in \mathcal{V} : \mathcal{L}_s(m_{\mathbf{x}}|\mathbf{z}_t) = 0 \\ \text{PossiblyOccupied} & \exists \mathbf{x} \in \mathcal{V} : \mathcal{L}_s(m_{\mathbf{x}}|\mathbf{z}_t) > 0 \\ \text{FreeOrUnobserved} & \text{otherwise} \end{cases} \tag{11}$$

Looking at Eq. 9 we can see that the gradient of $s(m_{\mathbf{x}}|\mathbf{z})$ is zero in FullyUnobserved areas and reaches local maxima where $x_\theta = z_\theta \pm 3\sigma_\theta$ or $x_r = z_r$ as illustrated in Fig. 2d. Using the fact that

$$\left.\frac{\partial s(m_{\mathbf{x}}|\mathbf{z})}{\partial x_\theta}\right|_{x_\theta = z_\theta \pm 3\sigma_\theta} = \frac{3}{16\sigma_\theta}, \quad \left.\frac{\partial s(m_{\mathbf{x}}|\mathbf{z})}{\partial x_r}\right|_{x_r = z_r} = \frac{3}{8\sigma_r} \tag{12}$$

and assuming the worst-case orientations for a cube-shaped partition $\mathcal{V}$, i.e. its diagonal projected into sensor coordinates r and θ aligns with either gradient, we obtain the following bounds for the approximation error for the three cases:

$$\epsilon_{\text{max}}(\mathcal{V}) = \begin{cases} 0 & \text{FullyUnobserved} \\ \max\left(\frac{3\mathcal{V}_{h_\theta}}{16\sigma_\theta}, \frac{3\mathcal{V}_{h_r}}{8\sigma_r}\right) & \text{PossiblyOccupied} \\ \frac{3\mathcal{V}_{h_\theta}}{16\sigma_\theta} & \text{FreeOrUnobserved} \end{cases} \tag{13}$$

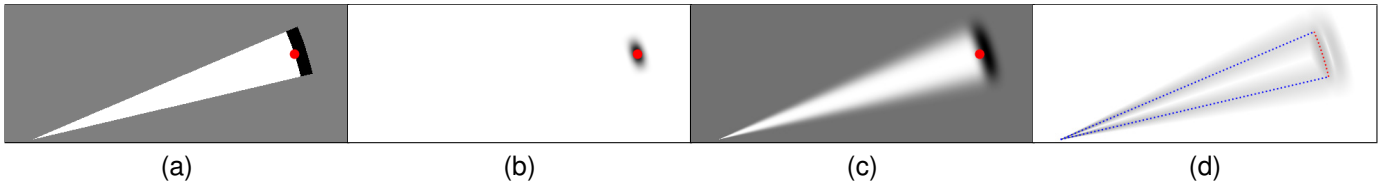

Fig. 2. Figure illustrating our proposed models for a) the occupancy given the true beam endpoint $\bar{s}(m_{\mathbf{x}}|\bar{\mathbf{z}})$ (eq. 4), b) the position of the true endpoint given a noisy measurement $o(\bar{\mathbf{z}}|\mathbf{z})$ (eq. 8), c) the complete inverse measurement model $s(m_{\mathbf{x}}|\mathbf{z})$ (eq. 9), and d) the local maxima used to derive the worst-case error bounds. Values of 0.0, 0.5 and 1.0 are shown in white, grey and black, respectively. The true beam endpoint is indicated in red. Uncertainties are exaggerated for illustration.

where $\mathcal{V}_h$ is the maximum distance a sample can have to $\mathcal{V}$'s center, namely half of $\mathcal{V}$'s diagonal. Note that $\mathcal{V}_{h_\theta}$ decays quickly as the distance to the sensor increases.

D. Saturated region skipping

To preserve the ability to quickly adapt the map when dynamic parts of the environment change, we impose upper and lower bounds on the occupancy posterior $\mathcal{L}_p(m_{\mathbf{x}}|\mathbf{z}_{1:t})$, as proposed by Yguel et al. [41]. As observed by Hornung et al. [13], this clamping policy also significantly improves compression performance by encouraging the majority of the map's posterior to converge to constant values. Namely to the lower bound in areas that are consistently observed as being free, and to the upper bound in areas that are consistently observed as being occupied. We propose to exploit this saturating behavior further to reduce the computational cost of map updates. Applying negative occupancy (free-space) updates in areas where the posterior has already reached the lower bound has no effect, as the updates are canceled out by the clamping operation. Similarly, the posterior is not affected by skipping positive occupancy updates in areas that already converged to the upper bound. Skipping saturated regions leads to a particularly high speedup if it can be done in a coarse-to-fine manner, but doing so is only safe if the map's lower resolutions are always up to date. Both properties are met by our representation and integration scheme. An algorithm that interleaves saturated region skipping, adaptive sampling, and thresholding will be discussed in the next section.

E. Algorithm and data structure

As described previously, Haar scaling functions do not overlap with their neighbors at the same resolution and perfectly partition the space. The support of the scaling functions in a multi-resolution Haar decomposition is, therefore, identical to the hierarchical partitioning scheme of octrees. We can thus store the wavelet coefficients in any optimized octree data structure that allows data to be attached to both inner and leaf nodes, such as supereight [34] or OpenVDB [22].

Leveraging the idea that increasing the resolution in MRAs only adds information, our proposed adaptive multi-resolution update algorithm determines the appropriate update resolution for all points in the observed volume in a coarse-to-fine manner (Section IV-C). The algorithm is initialized at the octree's root and recursively evaluates its children, as illustrated in Algorithm 1. Each recursive call starts by checking which of the three possible update cases, eq. 11, applies to the current node's partition $\mathcal{V}$. If no parts of the partition have been observed by the current measurement $\mathbf{z}_t$, or if saturated region skipping applies, no updates are needed. Otherwise, we continue by checking if the approximation error at the partition's current resolution is acceptable. If this is the case, we evaluate the inverse measurement model $s(m_{\mathbf{x}'}|\mathbf{z}_t)$ at the partition's center and integrate the update into the map. If none of the previous criteria were met, a higher resolution is needed and the recursive function is called for each of the octree node's sub-divisions (octants). In practice, we also compress the measurement update using the wavelet transform and need to traverse the map's data structure. Both of these operations can efficiently be interleaved with the recursive adaptive sampling procedure. Note that although the presented algorithm is recursive, great flexibility exists for its implementation. For example, since each Haar scaling function only overlaps with its parent and children, all partitions at a given resolution and their descendants can be updated in parallel.

V. Experiments

We evaluate our approach on three different datasets, featuring depth cameras and LiDARs, in indoor as well as outdoor environments. Comparisons are presented to three state-of-the-art volumetric mapping frameworks: octomap [13], voxblox [25], and supereight2 [11]. Octomap and supereight2 are both used in multi-resolution occupancy mapping-mode. Voxblox only supports TSDFs mapping and is configured to use its default 'fast' integration method. In terms of implementation details, all approaches are evaluated using their publicly available reference implementations[3][4][5] and wrapped with the same code to process the training data.

For each dataset, we split the original data into training and test sets by reserving every 20th observation for testing and use the remaining frames for mapping. Test points are generated by sampling points along all rays in each test observation, with points along the beam being in free space and the endpoint being occupied. To obtain insights into the behavior of the different methods in various scenarios we compute the distance of each free-space test point to the closest surface point. This allows us to evaluate the performance

[3]https://github.com/OctoMap/octomap_mapping
[4]https://bitbucket.org/smartroboticslab/supereight2
[5]https://github.com/ethz-asl/voxblox

Algorithm 1: Wavemap recursive update

Input: Current measurement $\mathbf{z}_t$,
Previous map posterior $p(m \mid \mathbf{z}_{1:t-1})$,
Lower log-odds threshold $\mathcal{L}_{\text{min}}$,
Approximation error threshold ϵ_{thresh},
Maximum resolution res_{max},
Octree's root partition $\mathcal{V}_{\text{root}}$

Output: Updated map posterior $p(m \mid \mathbf{z}_{1:t})$

Function `RecursiveAdaptiveUpdate`$(\mathcal{V}, \mathbf{z}_t)$ **is**
  `// Use Eq.11 to skip partitions`
  update_type $\leftarrow$ `UpdateType`$(\mathcal{V}, \mathbf{z}_t)$
  if update_type $==$ FullyUnobserved **then**
    return
  end
  if (update_type $==$ FreeOrUnobserved
    and $\mathcal{L}_p(m_\mathcal{V} \mid z_{1:t-1}) \leq \mathcal{L}_{min}$) **then**
    return
  end
  `// Use Eq.13 to terminate early`
  $\epsilon_{\text{max}}(\mathcal{V}) \leftarrow$ `ApproximationError`$(\mathcal{V}, \mathbf{z}_t)$
  if ($\mathcal{V}_{res} == res_{max}$ **or** $\epsilon_{max}(\mathcal{V}) < \epsilon_{thresh}$) **then**
    $\mathcal{L}_p(m_\mathcal{V} \mid \mathbf{z}_{1:t}) \leftarrow \mathcal{L}_p(m_\mathcal{V} \mid \mathbf{z}_{1:t-1})$
    $+\mathcal{L}_s(m_\mathcal{V} \mid \mathbf{z}_t)$
    return
  end
  `// Otherwise, increase resolution`
  for $\mathcal{V}_{child} \in \mathcal{V}$ **do**
    `RecursiveAdaptiveUpdate`$(\mathcal{V}_{\text{child}}, \mathbf{z}_t)$
  end
end
`// Initialize map and start recursion`
$p(m \mid \mathbf{z}_{1:t}) \leftarrow p(m \mid \mathbf{z}_{1:t-1})$
`RecursiveAdaptiveUpdate`$(\mathcal{V}_{\text{root}}, \mathbf{z}_t)$

in different range bands, including: i) small negative values assessing the ability to capture thin objects, ii) distances close to zero to evaluate the surface reconstruction quality, and iii) larger distances to obstacles to detect possible biases or approximation errors. This approach also avoids diluting a small number of challenging situations with a large number of easy-to-classify free space observations.

For each experiment, we report the overall Area under the ROC Curve (AUC) as a general indicator of classification performance. By integrating the Receiver Operating Characteristic (ROC) curve, the AUC quantifies how well each classifier discriminates free and occupied space regardless of the classification threshold. We also report the classification accuracy for the individual range bands. Note that different accuracies can be obtained based on the chosen classification threshold. For this study, we set the thresholds for each framework on each dataset to the value that maximizes the difference between the True Positive Rate (TPR) and the False Positive Rate (FPR), weighed equally.

TABLE I
AREA UNDER THE ROC CURVE RESULTS FOR BOTH DATASETS. HIGHER IS BETTER. THE CORRESPONDING RESOURCE USAGES ARE IN TABLE II.

Dataset	Res	octomap	super-eight2	voxblox	*ours (rays)*	*ours (beams)*
Panoptic	5cm	0.95	0.93	**0.99**	**0.99**	**0.99**
Flat	2cm	0.99	0.95	**1.00**	**1.00**	**1.00**
Newer	20cm	0.82	0.87	**0.92**	0.91	0.91
College	5cm	0.90	0.89	**0.97**	0.94	**0.97**

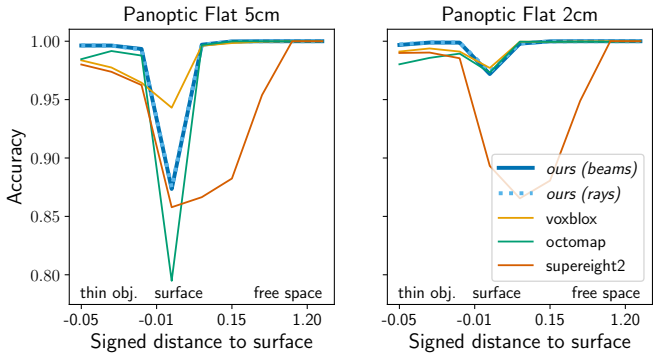

Fig. 3. Accuracy in function of the distance to the surface on the Panoptic mapping dataset at different resolutions. Higher is better.

A. Accuracy evaluations

1) Panoptic mapping dataset: The first set of experiments is conducted using "Run 1" of the panoptic mapping dataset [30], which features depth camera recordings of a simulated studio apartment including realistic household objects. Octomap and voxblox do not support depth images directly, and hence the dataset's images were first converted to pointclouds using the pinhole projection model used by both supereight2 and our method. The camera poses were obtained from the ground truth.

From the AUC values shown in Table I we can see that, when using larger cell sizes, only our proposed method can compete with voxblox. Being TSDF-based, voxblox can more accurately reconstruct smooth surfaces which account for large parts of the environment, giving it a distinct advantage. The remaining two methods have worse overall performance. When moving to a higher resolution the difference shrinks and all methods perform comparably. Looking at the results shown in Figure 3 we can clearly see where octomap and supereight2 accumulate their errors in the $5\,\text{cm}$ resolution case. Octomap struggles to properly localize the surface boundary, while supereight2 is overly pessimistic, labeling cells far from the surface as occupied. Finally, one can see the trade-off between our method, using a beam-based model, and voxblox, using a TSDF model. Voxblox has better at the surface reconstruction performance while our approach is better at reconstructing thin objects. This difference can also be seen in Figure 4 where the chair is missing its legs in the voxblox reconstruction. Looking at the $2\,\text{cm}$ resolution case, all methods but supereight2 perform almost identically. Supereight2 still produces pessimistic results, which likely stem from the approximations used to achieve its impressive speed.

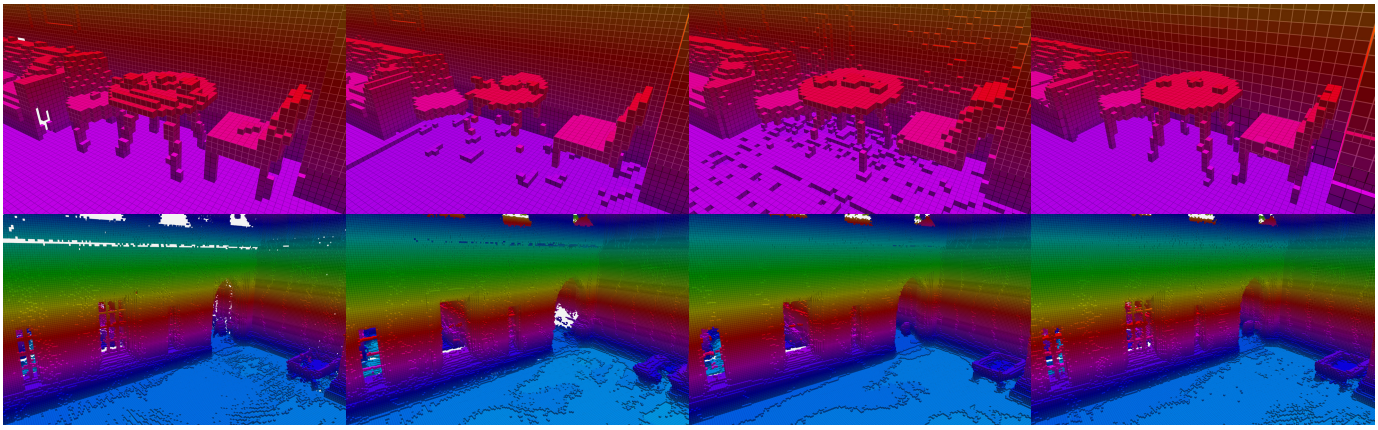

Fig. 4. Qualitative reconstruction comparisons featuring detailed geometry on scenes of the Panoptic mapping (top) and Newer College (bottom) datasets, both at 5cm resolution.

2) Newer College dataset: The second set of experiments uses the Cloister sequence from Collection 2 in the Newer College dataset [42]. This sequence was chosen because it captures geometry with a wide range of scales including wide-open outdoor spaces, indoor spaces with arches and sculptures, and vegetation. Odometry estimates and undistorted point clouds were obtained using FastLIO2 [39] processing the Ouster OS0-128 IMU and point cloud data. The motion-compensated point clouds were used for all frameworks except supereight2, which operates using dense range images and does not yet support motion-undistortion.

Looking at the AUC numbers in Table I we immediately see that this real-world LiDAR dataset is more challenging than the previous synthetic one. When using a coarse $20\,\mathrm{cm}$ resolution octomap performs the worst, with voxblox and our method achieving the best results, and supereight2 landing in the middle. Moving to a higher resolution of $5\,\mathrm{cm}$ octomap and supereight2 end up performing similar while voxblox slightly outperforms our approach. However, the detailed results shown in Figure 5 reveal interesting insights. At $20\,\mathrm{cm}$ resolution octomap struggles to produce accurate surfaces. We also see that our approach and supereight2 have similar performance when it comes to reconstructing the surface but our approach performs slightly better when classifying free space in the vicinity of obstacles. Voxblox again performs the best in surface reconstruction and free space classification, but suffers in the thin object reconstruction domain. Moving to a finer $5\,\mathrm{cm}$ resolution the change is similar to that observed in the Panoptic dataset. The accuracy of every method improves and they move closer together, with supereight2 failing to accurately predict free space close to surfaces. The differences between the other three methods are characterized by octomap not reconstructing thin objects accurately while both voxblox and ours (beams) perform equally well.

3) Sensor model ablation: To verify the benefit of the more costly uncertainty aware sensor model proposed in Section IV-B, we conduct an ablation comparing our proposed sensor model, *ours (beams)*, with one that disregards angular uncertainty, *ours (rays)*. As the Panoptic Flat dataset contains no noise on observation or pose there is, as to be expected, no difference between the two models. On the Newer College dataset, however, there are visible differences. In the coarse setting the proposed uncertainty-aware model improves the ability to reconstruct thin objects. Moving to the higher resolution case both the ability to reconstruct surfaces and thin objects are significantly improved by our proposed model.

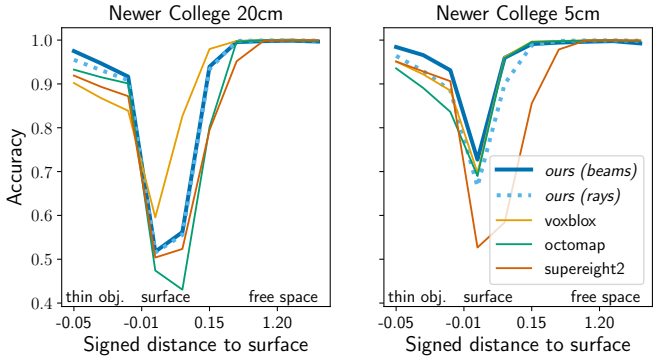

Fig. 5. Accuracy in function of the distance to the surface on the Newer College dataset at different resolutions. Higher is better.

These accuracy evaluations showed several things. The proposed method *ours (beams)* compares favorably to the other three methods. Despite the natural advantage voxblox has in surface reconstruction tasks, being a TSDF-based method, our approach performs on par while having superior performance in thin object reconstruction. The uncertainty-aware sensor model also improves the quality of the map close to surfaces and when dealing with thin surfaces, allowing the reconstruction of objects that other methods can't capture when using the same cell size.

B. Efficiency evaluations

We evaluate the memory usage as well as the runtime of our method in comparison to the three baseline methods. Memory usage is reported as the amount of RAM used by the method as well as the memory used by the map data structure.

While our framework can be implemented using various data structures, we used octomap's octree implementation to keep the comparison as fair as possible. The runtime is reported as the elapsed wall time and the cumulative CPU time across all threads, allowing a fair comparison between single-threaded and multi-threaded methods. All frameworks have their visualizations disabled and all experiments are performed on the same desktop computer with an Intel i9-9900K CPU.

From the numbers shown in Table II we can see that supereight2 ranks first in terms of wall time on the depth camera dataset, and second best for LiDAR. However, the memory usage of its maps is relatively large owing to the fact that it estimates occupancy using weighted averaging instead of log-odds updates (requiring 2 floats per cell instead of 1) and focuses its implementation primarily on speed. Voxblox, as to be expected from a TSDF-based method, has the largest map sizes at higher resolutions but is computationally efficient. Octomap produces large maps, in comparison to our method, and is the slowest of all compared methods by an order of magnitude. Octomap's significant slowdown at high resolutions is caused by the fine-to-coarse model employed by their integrator which needs to touch every single cell. Our proposed method obtains maps that are significantly smaller than those of octomap despite using the same underlying data structure.

The runtime of our method, when looking at the CPU time, is equal or better than that of supereight2. However, as supereight2's implementation uses multiple threads the real-world performance of it is still better. The difference in runtime and memory usage between our method and octomap clearly shows the benefits of using wavelets to represent the map as it enables good compression and allows the use of an efficient coarse-to-fine integrator capable of skipping unnecessary work.

Comparing the memory and runtime of *ours (rays)* and *ours (beams)* we can see that the price for the improved quality is larger maps by about 30% to 70% depending on the resolution and an increase in runtime of around 50%. These increases stem from the fact that the uncertainty-aware model needs to update more voxels and that the map contains more fine details and voxels with partial occupancy values. Overall, our proposed method shows good general performance in both memory usage and runtime, with clear avenues for improvements. The wall time could be reduced significantly using multi-threading, which is easily achievable due to the independence of the voxel updates. Moreover, we believe the memory used to store the map itself could be reduced further by using a more efficient data structure such as the one proposed by OpenVDB [22]. These extensions will be added to the open-source code.

C. Multi-sensor multi-resolution mapping

One key advantage of our framework is its natural ability to handle multiple sensors at different resolutions. In this experiment, we show qualitative results of our mapping framework running in multi-sensor mode on the DARPA SubT Finals dataset [33]. In Figure 6 we show the output of our framework simultaneously integrating two Robosense Bpearl dome-LiDAR sensors and one Velodyne VLP-16 LiDAR. The Bpearls were angled to scan the ground around the robot while the VLP-16 was providing long-range observations. As the sensors provide information for different purposes we integrate them with different resolutions into the map. The Bpearls, responsible for local terrain mapping to enable navigation of a quadruped, are integrated at $2\,\mathrm{cm}$ resolution. At the same time the VLP-16, responsible for long-range mapping and exploration, is integrated with a resolution of at most $16\,\mathrm{cm}$. This results in a unified map, that supports both local trajectory planning as well as global exploration goal placements, without wasting resources on high-accuracy map reconstruction in areas where it is not needed. While shown here for multiple LiDAR sensors the same approach can be used for mobile manipulation setups using a 3D LiDAR for navigation and RGB-D cameras for scene reconstruction, resulting in a map that supports both navigation as well as manipulation.

TABLE II
COMPUTATIONAL RESOURCE USAGE AT DIFFERENT RESOLUTIONS. LOWER IS BETTER.

Dataset	Res	Framework	Memory (MB) RAM	Memory (MB) Map only	Time (s) CPU time	Time (s) Wall time
Panop. Flat	5cm	octomap	162.35	6.50	130.32	129.00
		supereight2	158.23	46.09	27.79	**4.76**
		voxblox	229.96	36.90	58.58	10.68
		ours (rays)	135.69	**4.17**	**5.58**	6.78
		ours (beams)	**130.04**	5.65	6.94	7.20
	2cm	octomap	6202.39	50.94	773.16	763.39
		supereight2	448.38	285.07	50.83	**9.32**
		voxblox	663.53	348.15	244.69	24.61
		ours (rays)	343.26	**39.09**	**33.00**	34.80
		ours (beams)	**294.58**	67.81	57.56	57.51
Newer Coll.	20cm	octomap	203.25	20.78	688.71	709.99
		supereight2	249.03	107.79	411.67	67.14
		voxblox	261.02	66.32	228.12	**48.07**
		ours (rays)	180.86	**6.94**	**87.39**	88.78
		ours (beams)	**138.92**	8.82	107.67	113.26
	5cm	octomap	14404.76	981.02	36252.70	35790.60
		supereight2	2926.42	2333.93	2853.12	404.19
		voxblox	3718.85	2362.58	1788.90	**162.36**
		ours (rays)	1192.95	**241.84**	**1656.26**	1671.58
		ours (beams)	**1065.21**	402.18	2085.05	2083.61

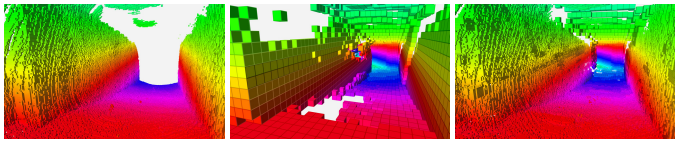

Fig. 6. Example of our framework performing multi-sensor, multi-resolution volumetric mapping on the DARPA SubT Finals dataset, combining data from 2 ground-facing LiDARs at 2cm resolution (left) and 1 horizontal LiDAR at 16cm resolution up to a range of 30m (center) into a single map (right).

VI. CONCLUSION

In this work, we introduced *wavemap*, a hierarchical volumetric mapping framework inspired by multi-resolution analysis. The MRA theory guarantees that using wavelet decomposition, we can safely and very efficiently integrate new

observations in a coarse-to-fine manner. The resulting gains in computational efficiency, together with early stopping criteria for the integrator, allow us to use more complex sensor models such as the proposed angular uncertainty-aware model. In experiments on synthetic RGB-D and real-world 3D LiDAR data, we demonstrate that our proposed method achieves high-quality results while being efficient in terms of memory and compute requirements. We also demonstrate how our method can incorporate observations from multiple sensors into a single map with per-sensor resolution. This allows the use of a single map representation for tasks that would have required several dedicated maps in the past. Finally, we open source the implementation of our approach to facilitate future research.

REFERENCES

[1] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Robotics: Science and Systems*, 2018.

[2] P. Burt and E. Adelson. The Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*, 31(4):532–540, April 1983.

[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 32(6):1309–1332, December 2016.

[4] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1996.

[5] I. Daubechies. *Ten Lectures on Wavelets*. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1992.

[6] D. Duberg and P. Jensfelt. UFOMap: An Efficient Probabilistic 3D Mapping Framework That Embraces the Unknown. *IEEE Robotics and Automation Letters*, 2020.

[7] A. Elfes. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, 1989.

[8] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-Scale Direct Monocular SLAM. In *Europena Conference on Computer Vision*, 2014.

[9] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa. Plenoxels: Radiance Fields without Neural Networks. In *IEEE / CVF Computer Vision and Pattern Recognition Conferenc*, 2022.

[10] S. Fuhrmann and M. Goesele. Fusion of Depth Maps with Multiple Scales. *ACM Transactions on Graphics*, 2011.

[11] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P. Alcantarilla, and S. Leutenegger. Multi-Resolution 3D Mapping With Explicit Free Space Representation for Fast and Accurate Mobile Robot Motion Planning. *IEEE Robotics and Automation Letters*, 2021.

[12] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *International Symposium on Experimental Robotics*, 2014.

[13] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 2013.

[14] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *ACM Symposium on User Interface Software and Technology*, 2011.

[15] O. Kähler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray. Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics*, 2015.

[16] C. Loop, Q. Cai, S. Orts-Escolano, and P. A. Chou. A Closed-Form Bayesian Fusion Equation Using Occupancy Probabilities. In *International Conference on 3D Vision*, 2016.

[17] S. G. Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Elsevier/Academic Press, 3rd ed edition, 2009.

[18] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[19] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *European Conference on Computer Vision*, 2020.

[20] A. Millane, Z. Taylor, H. Oleynikova, J. Nieto, R. Siegwart, and C. Cadena. C-Blox: A Scalable and Consistent TSDF-based Dense Mapping Approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[21] R. Mur-Artal, J. Montiel, and J. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 2015.

[22] K. Museth, J. Lait, J. Johanson, J. Budsberg, R. Henderson, M. Alden, P. Cucka, D. Hill, and A. Pearce. OpenVDB: An Open-Source Data Structure and Toolkit for High-Resolution Volumes. In *ACM SIGGRAPH*, 2013.

[23] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-Time 3D Reconstruction at Scale Using Voxel Hashing. *ACM Transactions on Graphics*, 2013.

[24] S. O'Callaghan and F. Ramos. Continuous occupancy mapping with integral kernels. In *AAAI Conference on Artificial Intelligence*, 2011.

[25] 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*, 2017.

[26] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan. Elastic lidar fusion: Dense map-centric continuous-time slam. In *IEEE International Conference on Robotics and Automation*, 2018.

[27] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, 2020.

[28] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat. Comparing ICP Variants on Real-World Data Sets. *Autonomous Robots*, 2013.

[29] A. Schaefer, L. Luft, and W. Burgard. DCT Maps: Compact Differentiable Lidar Maps Based on the Cosine Transform. *IEEE Robotics and Automation Letters*, 2018.

[30] L. Schmid, J. Delmerico, J. Schönberger, J. Nieto, M. Pollefeys, R. Siegwart, and C. Cadena. Panoptic Multi-TSDFs: A Flexible Representation for Online Multi-resolution Volumetric Mapping and Long-term Dynamic Scene Consistency. In *International Conference on Robotics and Automation*, 2022.

[31] J. Schönberger and J. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition*, 2016.

[32] J. Stückler and S. Behnke. Multi-resolution surfel maps for efficient dense 3D modeling and tracking. *Journal of Visual Communication and Image Representation*, 2014.

[33] M. Tranzatto, M. Dharmadhikari, L. Bernreiter, M. Camurri, S. Khattak, F. Mascarich, P. Pfreundschuh, D. Wisth, S. Zimmermann, M. Kulkarni, V. Reijgwart, B. Casseau, T. Homberger, P. De Petris, L. Ott, W. Tubby, G. Waibel, H. Nguyen, C. Cadena, R. Buchanan, L. Wellhausen, N. Khedekar, O. Andersson, L. Zhang, T. Miki, T. Dang, M. Mattamala, M. Montenegro, K. Meyer, X. Wu, A. Briod, M. Mueller, M. Fallon, R. Siegwart, M. Hutter, and K. Alexis. Team CERBERUS Wins the DARPA Subterranean Challenge: Technical Overview and Lessons Learned, 2022.

[34] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. Kelly, and S. Leutenegger. Efficient Octree-Based Volumetric SLAM Supporting Signed-Distance and Occupancy Mapping. *IEEE Robotics and Automation Letters*, 2018.

[35] E. Vespa, N. Funk, P. H. J. Kelly, and S. Leutenegger. Adaptive-Resolution Octree-Based Volumetric SLAM. In *International Conference on 3D Vision*, 2019.

[36] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss. VDBFusion: Flexible and Efficient TSDF Integration of Range Sensor Data. *Sensors*, 2022.

[37] T. Whelan, M. Kaess, M. Fallon, H. Johannsson, J. Leonard, and J. McDonald. Kintinuous: Spatially Extended KinectFusion. In *AAAI Conference on Artificial Intelligence*, 2012.

[38] T. Whelan, S. Leutenegger, R. Salas Moreno, B. Glocker, and A. Davison. ElasticFusion: Dense SLAM Without A Pose Graph. In *Robotics: Science and Systems*, 2015.

[39] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang. FAST-LIO2: Fast Direct LiDAR-Inertial Odometry. *IEEE Transactions on Robotics*, 2022.

[40] M. Yguel, O. Aycard, and C. Laugier. Wavelet Occupancy Grids: A Method for Compact Map Building. In *Field and Service Robotics*, 2006.

[41] M. Yguel, O. Aycard, and C. Laugier. Update Policy of Dense Maps: Efficient Algorithms and Sparse Representation. In *Field and Service Robotics*, 2008.

[42] L. Zhang, M. Camurri, D. Wisth, and M. Fallon. Multi-camera lidar inertial extension to the newer college dataset. *arXiv preprint arXiv:2112.08854*, 2021.

APPENDIX

This appendix is intended as a primer on wavelet theory, providing additional context for the method section. We start with a short introduction to the MRA conditions, before showing how orthogonal wavelet bases fulfill these requirements. We then discuss how wavelet decompositions can efficiently be computed using the Fast Wavelet Transform. For readers that are interested in learning more about sparse signal processing using wavelets, we warmly recommend [5, 17].

A. Multi-Resolution Analysis

Multi-resolution representations are regularly used in the context of computer vision and robotics. For example, in Laplacian image pyramids introduced by Burt and Adelson [2]. Mallat and Meyer [17], formalized the expected behavior of multi-resolution representations through the MRA conditions:

$$\forall (j,k) \in \mathbb{Z}^2, \qquad f(x) \in V_j \Leftrightarrow f(x - 2^j k) \in V_j \tag{14}$$

$$\forall j \in \mathbb{Z}, \qquad V_{j+1} \subset V_j \tag{15}$$

$$\forall j \in \mathbb{Z}, \qquad f(x) \in V_j \Leftrightarrow f(x/2) \in V_{j+1} \tag{16}$$

$$\lim_{j \to \infty} V_j = \bigcap_{j=-\infty}^{\infty} V_j = \{0\} \tag{17}$$

$$\lim_{j \to -\infty} V_j = \text{closure}\left(\bigcup_{j=-\infty}^{\infty} V_j \right) = L^2(\mathbb{R}) \tag{18}$$

$$V_0 \text{ admits a Riesz basis} \tag{19}$$

where the sequence of subspaces $\{V_j\}_{j \in \mathbb{Z}}$ corresponds to the map's representations at increasing resolution levels 2^j, referred to as scales, and each V_j is a closed subspace of Lebesgue space L^2. Starting with condition 19, the most common Riesz basis used in robotics consists of box functions arranged to span the cells of a regular grid. In this case, the scale 2^j corresponds to the cell width. Condition 14 ensures self-similarity in space. Specifically, if subspace V_j can represent function $f(x)$, it can also represent the same function shifted by integer multiple of the cell size. Condition 15 states that the subspaces are nested. In other words, any function contained in subspace V_{j+1} must also be contained in next finer subspace V_j and by extension in all finer subspaces. Condition 16 ensures self-similarity in scale. If V_j contains $f(x)$, V_{j+1} must be able to contain $f(x)$ dilated by 2. Finally, conditions 17 and 18 ensure completeness. At the coarsest scale ($j \to \infty$), V_j only contains the zero element, whereas refining the scale ($j \to -\infty$) eventually allows us to represent any signal in L^2.

B. Orthogonal wavelet bases

The principal idea behind wavelets is that they represent the difference between the consecutive resolutions of a signal's MRA. Formally, they span a second subspace W_j which is the orthogonal complement to V_j, such that $V_j \oplus W_j = V_{j-1}$ where $\oplus$ is the vector-space direct sum operator. In words, this means that by combining a signal's representation V_j with its wavelet details at the same resolution W_j we obtain the signal's representation at the next higher resolution V_{j-1}.

An orthogonal basis for all V_j can be obtained by translating and dilating a single function ϕ, referred to as the scaling function, as $\phi_{jk}(x) = \frac{1}{2^j}\phi(\frac{x-2^j k}{2^j})$. The scaling function can be found by orthogonalizing the Riesz basis of V_0 as described in [17]. In similar fashion, an orthogonal basis for W_j can be obtained by translating and scaling a single wavelet function ψ as $\psi_{jk}(x) = \frac{1}{2^j}\psi(\frac{x-2^j k}{2^j})$. One condition that any wavelet function has to fulfill in order to be admissible is that its average must be zero $\int_{-\infty}^{\infty} \psi(x)dx = 0$. More generally, the scaling functions and wavelet functions can be seen as complementary low and high-pass filters that, when combined, can perfectly reconstruct the signal from the next finer scale. Since wavelet bases form a valid MRA, this concept can be applied recursively and the entire map can be represented by stacking a single scaling function at the coarsest scale with a hierarchy of wavelet functions at increasing scales.

Note that the Riesz basis consisting of box filters arranged to span the cells of a regular grid, mentioned previously, is already orthogonal. In fact, the unit box filter can be used as a scaling function

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

and doing so directly leads to the Haar basis [17]. The corresponding Haar wavelet function can be derived by finding ϕ's orthogonal complement while enforcing the MRA conditions and is given by

$$\psi(x) = \begin{cases} -1 & 0 \leq x < 1/2 \\ 1 & 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

Orthogonal wavelet bases of $\mathbb{R}$ can be extended to separable orthogonal bases b for $\mathbb{R}^n$ by combining the scaling and wavelet functions along each dimension as

$$b = \left\{ \prod_{k=1}^{n} \phi(x_k)^{o_k} \psi(x_k)^{1-o_k} \right\}_{\forall o \in \{0,1\}^n} \tag{22}$$

C. The Fast Wavelet Transform

The discrete wavelet transform for a function f and wavelet ψ is defined as the projection of f onto the set of all integer scalings and translations of the wavelet function $\{\psi_{jk}\}_{j,k\in\mathbb{Z}}$. Each wavelet coefficient d_{jk} is thus computed as

$$d_{jk} = \sum_{x=-\infty}^{\infty} f(x) \frac{1}{2^j} \psi\left(\frac{x - 2^j k}{2^j}\right) \tag{23}$$

where the summation could be replaced by an integral if the domain of f is real-valued instead of discrete. Note that this transform is linear and, for orthogonal wavelets, orthogonal.

The coefficients d_{jk} can efficiently be computed using the FWT algorithm [17], which exploits the hierarchical MRA structure to remove redundant operations. The FWT is initialized by projecting f onto the scaling functions at the finest scale $a_{0k} = \sum_{-\infty}^{\infty} f(x)\phi(x-k)$ or with a good approximation thereof. At each iteration, these coefficients are then filtered and downsampled to obtain the wavelet and scaling coefficients at the next coarser scale. These iterations are typically repeated until only 1 scaling coefficient is left or a desired number of levels is reached. For wavelets with finite spatial support and functions f sampled at N points, the FWT computes the full wavelet decomposition in $O(N)$ time. Extending the FWT to only (de)compress regions-of-interest or single cells is straightforward and very efficient if the spatial support of the chosen wavelet is small, as is the case for the Haar wavelet.