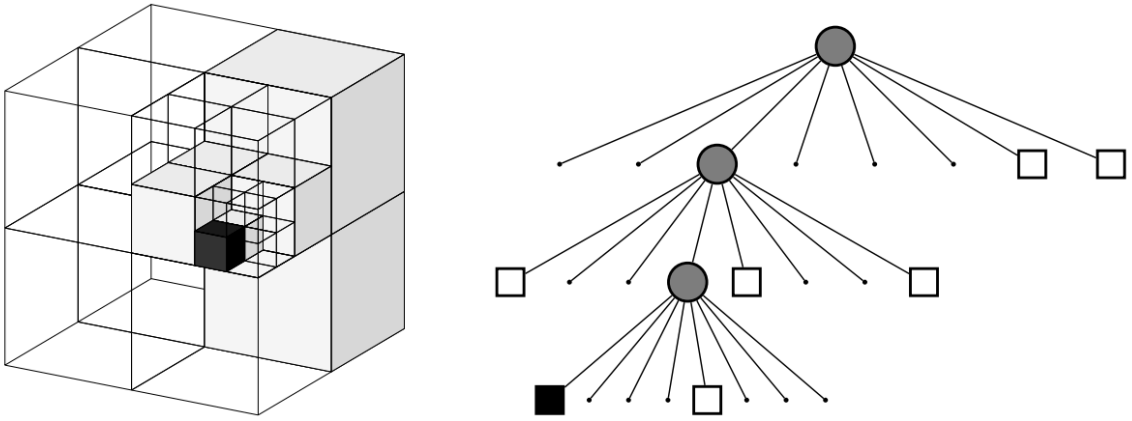
An octree is a tree data structure in which each internal node has exactly eight children

Octrees are most often used to partition a three-dimensional space by recursively subdividing it into eight octants.

Space representation

Each node in an octree represents the space contained in a cubic volume, usually called a voxel. This volume is recursively subdivided into eight sub-volumes until a given minimum voxel size is reached. The minimum voxel size determines the resolution of the octree.

Since an octree is a hierarchical data structure, the tree can be cut at any level to obtain a coarser subdivision if the inner nodes are maintained accordingly.



In its most basic form, octrees can be used to model a Boolean property. In robotic mapping, this is usually the occupancy of a volume.

If a certain volume is measured as occupied, the corresponding node in the octree is initialised. Any uninitialised node could be free or unknown.

Given sensor noise and changing environments, a discrete occupancy label is not sufficient, instead occupancy has to be modelled probabilistically, however this lacks the possibility of lossless compression by pruning.

Probabilistic sensor fusion

The probability $P(n \mid z_{1:t})$ of a leaf node n to be occupied given the sensor measurements $z_{1:t}$ is estimated according to:

$$P(n \mid z_{1:t}) = \left[\frac{1 - P(n \mid z_t)}{P(n \mid z_t)} \frac{1 - P(n \mid z_{t-1})}{P(n \mid z_{t-1})} \frac{P(n)}{1 - P(n)} \right]^{-1}$$

This means you can calculate the probability of a leaf node to be occupied if you keep track of the prior probability and the probability solely based on the latest measurement.

Multiplying many probabilities can become numerically unstable, so we convert them to logarithms such that the multiplications are additions and thus more numerically stable.

Thus, we define the log-odds of the voxel n being occupied:

$$L(n) = \log\left[\frac{P(n)}{1 - P(n)}\right]$$

By using log-odds notation, $P(n \mid z_{1:t})$ of a leaf node n to be occupied given the sensor measurements $z_{1:t}$ can be rewritten as:

$$L(n \mid z_{1:t}) = L(n \mid z_{1:t-1}) + L(n \mid z_t)$$

Log-odds values can be converted into probabilities and vice versa and we therefore store this value for each voxel instead of the occupancy probability.

We can convert back to probabilities using:

$$P(n) = \frac{1}{1 + \exp(-L(n))}$$

When a 3D map is used for navigation, a threshold on the occupancy probability $P(n \mid z_{1:t})$ is often applied. A voxel is considered to be occupied when the threshold is reached and is assumed to be free otherwise, thereby defining two discrete states.

To change the state of a voxel we need to integrate as many observations as have been integrated to define its current state. While this is desirable in static environments, a mobile robot is often faced with changes in the environment. To ensure this adaptability, occupancy estimates are updated according to:

$$L(n \mid z_{1:t}) = max(min(L(n \mid z_{1:t-1}), l_{max}), l_{min})$$

where l_{min} and l_{max} denote the lower and upper bound on the log-odds value

This prevents the robot to adapt when objects move or disappear by not allowing occupancy probability form becoming too certain, allowing it to be changed later, ensure not too many measurements are necessary to change the belief.

Multi-resolution queries

Since an octree is a hierarchical data structure, we can make use of inner nodes in the tree to enable multi-resolution queries. We yield a coarser subdivision of 3D space when the tree is traversed only up to a given depth that is not the depth of the leaf nodes.

To determine the occupancy probability of an inner node, we have to aggregate the probabilities of its children.

We can use the average occupancy:

$$\overline{l}(n) = \frac{1}{8}\sum_{i=1}^{8} L(n_i)$$

or the maximum occupancy:

$$\hat{l}(n) = max_i\ L(n_i)$$

where $L(n)$ returns the current log-odds occupancy value of a node n.

Information rich maps

Octree nodes can be extended to store additional data to enrich the map representation, e.g. voxels could store colour information.

Each additional voxel property requires a method that allows several measurements to be fused.

If we store the average colour of each voxel, we can create visualizations for the user and run a colour-based classification of the environment or appearance-based robot localization from virtual views