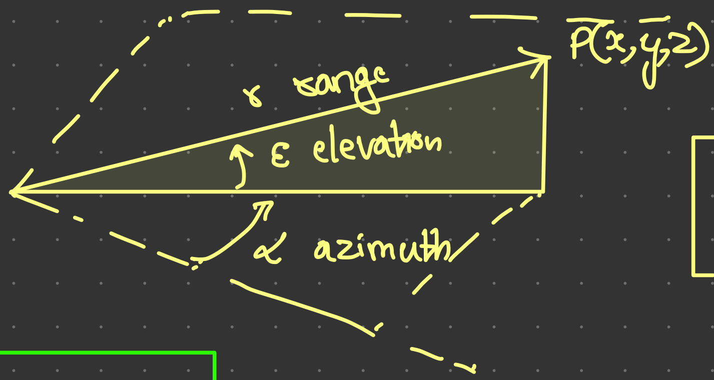
LIDAR Point Clouds



$$\begin{bmatrix}x\\y\\z\end{bmatrix} = h^{-1}(r,\alpha,\varepsilon) = \begin{bmatrix} r\cos\alpha\cos\varepsilon \\ r\sin\alpha\cos\varepsilon \\ r\sin\varepsilon\end{bmatrix}$$

$$P_s = \begin{bmatrix} P_s^{(1)} & P_s^{(2)} & P_s^{(3)} & \dots & P_s^{(n)} \end{bmatrix}$$

$$P_s = \begin{bmatrix} x_s^{(1)} & \dots & x_s^{(n)} \\ y_s^{(1)} & \dots & y_s^{(n)} \\ z_s^{(1)} & \dots & z_s^{(n)} \end{bmatrix}$$

Translation:

$$P_{s'}^{(i)} = P_s^{(j)} - r^{s's}$$

$$P_{s'} = P_s - R_s^{s's}$$

$$R_s^{s's} = \begin{bmatrix} r_s^{s's} & r_s^{s's} & \dots \end{bmatrix}$$

Rotation:

$$r_{s'} = C_{s's}\, r_s$$

$$P_{s'} = C_{s's}\, P_s$$

Scaling:

$$r_{s'} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} r_s$$

$$P_{s'} = S_{s's}\, P_s$$

⟹ All together:

$$P_{s'}^{(j)} = S_{s's}\, C_{s's}\left(P_s^{(j)} - r_s^{s's}\right)$$

$$P_{s'} = S_{s's}\, C_{s's}\left(P_s - R_s^{s's}\right)$$

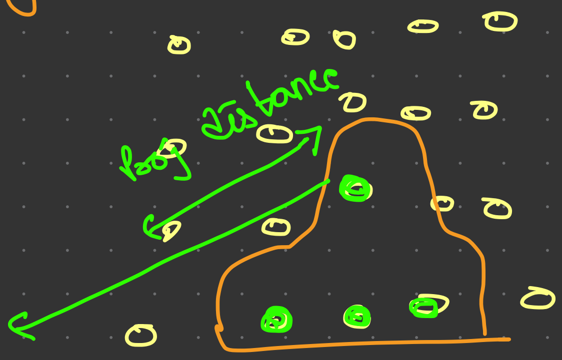
31/08/2021 — LIDAR PDF, OCCUPANCY MAP DERIVATION OCTREE, SDF

0:41:00

Octree has memory advantage.
- ↳ Only divides cell further if needed based on conflict [if occupied]
- ↳ Needs a data structure

Minimum voxel determines resolution.

Signed Distance Function (SDF):



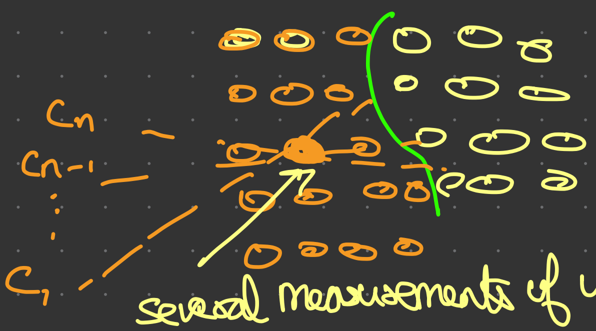
Method:
1) Output Region : $D(x) < 0$
2) On border : $D(x) = 0$
3) On inside region: $D(x)$

→ Measures distance b/w each voxel to observed surface

→ Parallisable
→ Efficient when small interval is considered

$$d_{OBS} = z - I_z(\pi(x, y, z))$$

When multiple camera views are taken,



$$D \leftarrow \frac{WD + wd}{W + w}$$

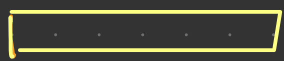
$$W \leftarrow W + w$$

Assume camera poses are known

Occ. map: Explicit representation

one = occupied

SDF: Implicit representation



- ve = outside obj   + ve = inside obj

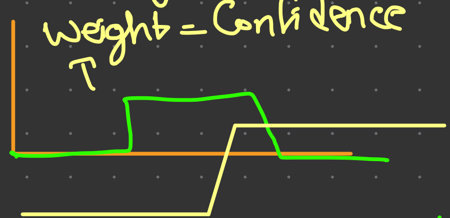
→ Camera for each pose
→ For each grid cell compute projective distance to surface and generalise to 3D
→ Memory usage is cubic in side length

DATA FUSION

→ Compute weighted average
→ For voxel has 2 values
  ↳ Sum of signed distances $D_t(x)$
  ↳ Sum of weights $W_t(x)$
→ When new range image arrives

$$D_{t+1}(x) = D_t(x) + \omega_{t+1}(x) d_{t+1}(x)$$
$$W_{t+1}(x) = W_t(x) + \omega_{t+1}(x)$$

→ Each obs is weighed acc to confidence
→ Can also be influenced by other modalities



Non-zero weight regions need not be stored

→ Hierarchical structure _ voxels grouped in 'bricks'

Requirements for reconstruction band across surfaces:

→ Low storage cost
→ Time-efficient update
→ Online capablity i.e. not all data is present beforehand
    ↳ Grows when added depth info
    ↳ Only bricks in current camera frustum is touched

ESDF:

↳ Euclidean signed distance field
↳ Captures euclidean distance of each voxel to nearest surface (obstacle)
↳ Voxblox: Volumetric mapping library

Explicit Surface Reps
    ↳ Geometry stored explicitly as points, Δ's using point clouds, meshes

Implicit Surf. Rep:
    ↳ Defined as a level set of function over space in which geometry is embedded
        ↳ Parametric: $x^2+y^2=r^e$
        Non-param

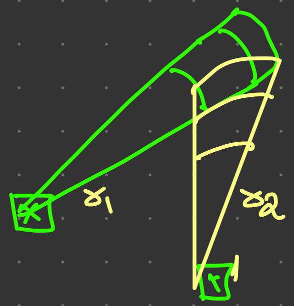
03-09-2021

→ Least Squares
→ MLR
→ P=RQSr t derivation
→ LM

Occupancy mapping

$P(C_i|r_1)$ = Prob that cell C_i is occupied given measurement r_1

$P(\bar{C}_i|r_1)$ = Prob that cell C_i is unoccupied given measurement r_1



$$P(C_i|r_2,r_1) = \frac{P(r_2|C_i,r_1)\cdot P(C_i|r_1)}{P(r_2|r_1)}$$

as $P(A|B,C) = \dfrac{P(B|A,C)\,P(A|C)}{P(B|C)}$

$$P(C_i|r_2,r_1) = \frac{P(C_i|r_2)\,P(C_i|r_1)\,P(r_2)}{P(r_2|r_1)\,P(C_i)}$$

$$\boxed{P(C_i|r_2,r_1) = k\,P(C_i|r_2)\cdot P(C_i|r_1)} \quad 1$$

$$\boxed{P(\bar{C}_i|r_2,r_1) = \bar{k}\,P(\bar{C}_i|r_2)\,P(\bar{C}_i,r_1)} \quad 2$$

$k = \bar{k}$

$$1 \leftarrow \frac{1}{1+2}$$

$$2 \leftarrow \frac{2}{1+2}$$