

# 1 Dense 3D Reconstruction using Robot Hand-Mounted Cameras

This is a general sketch of an idea for reconstructing a dense 3D scene using a hand-mounted depth camera.

## 1.1 Problem

Consider a robot that has a configuration space  $C \subseteq \mathbf{R}^N$ . Mounted on one of the links of the robot is a depth sensor. The task is, given a sequence of noisy configurations at each time step  $\{q_1, q_2, \dots, q_T\} \in C$ , and a sequence of noisy sensor readings (point clouds) captured simultaneously  $\mathbf{Z} = \{Z_1, Z_2, \dots, Z_T\}$ , where  $Z_i = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subseteq \mathbf{R}^3$ , simultaneously reconstruct the true dense 3D geometry of the scene and estimate the true configuration of the robot at each time step.

## 1.2 Configuration Noise Model

The robot's configuration at time step  $t$  is given by joint encoder readings  $q_t$ , which is a random value drawn from the distribution:

$$q_t = q_t^{\text{true}} + \epsilon_t$$

Where  $q_t^{\text{true}}$  are the true joint angles of the robot, and  $\epsilon_t$  is a random offset drawn from some unknown random distribution  $Q$ .

$$\epsilon_t \sim Q(q_t^{\text{true}})$$

In practice,  $Q$  is highly nonlinear, and depends on the dynamics of the system (for example, the direction of gravity, or other factors that depend on the robot's joint angles). Therefore, finding an *a. priori* representation of  $Q$  may be infeasible. The task is to compute  $\epsilon_t^*$ , which is an estimate of the current noisy offset.

## 1.3 General Approach

We want to find the maximum likelihood estimate of  $\epsilon_t$  given the sensor readings at time  $t$ :

$$\epsilon_t^* = \operatorname{argmax}_{\epsilon_t} P(\epsilon_t | \mathbf{Z}_t) \quad (1)$$

$$= \operatorname{argmax}_{\epsilon_t} \frac{P(\mathbf{Z}_t | \epsilon_t) P(\epsilon_t)}{P(\mathbf{Z}_t)} \quad (2)$$

$$= \operatorname{argmax}_{\epsilon_t} \frac{P(\epsilon_t) \prod_{\mathbf{x}_i \in \mathbf{Z}_t} P(\mathbf{x}_i | \epsilon_t)}{P(\mathbf{Z}_t)} \quad (3)$$

$$= \operatorname{argmax}_{\epsilon_t} \log \frac{P(\epsilon_t) \prod_{\mathbf{x}_i \in \mathbf{Z}_t} P(\mathbf{x}_i | \epsilon_t)}{P(\mathbf{Z}_t)} \quad (4)$$

$$= \operatorname{argmax}_{\epsilon_t} \left[ \log P(\epsilon_t) + \sum_{\mathbf{x}_i \in \mathbf{Z}_t} \log P(\mathbf{x}_i | \epsilon_t) - \log P(\mathbf{Z}_t) \right] \quad (5)$$

$$= \operatorname{argmax}_{\epsilon_t} \left[ \log P(\epsilon_t) + \sum_{\mathbf{x}_i \in \mathbf{Z}_t} \log P(\mathbf{x}_i | \epsilon_t) \right] \quad (6)$$

The term  $P(\epsilon_t) = Q(q_t^{\text{true}})$  is the prior on  $\epsilon_t$ . The individual terms  $P(\mathbf{x}_i | \epsilon_t)$  are the posterior probabilities of each sensor reading given a particular configuration offset  $\epsilon_t$ . Calculating this term will require a definition of the world, and the properties of the sensor.

## 1.4 World Representation

We will represent the world as a truncated signed distance function (TSDF). It is defined as  $D(\mathbf{x}) : \mathbf{R}^3 \rightarrow \mathbf{R}$ , and represents the (signed) distance to the nearest obstacle in meters, up to a truncation distance  $\tau$ .

The gradient of the distance field  $\nabla D : \mathbf{R}^3 \rightarrow \mathbf{R}^3$ , is clearly defined and easy to compute.

## 1.5 Forward Kinematics

Assume we have a forward kinematics function of the robot  $F(\mathbf{q}, \mathbf{x}) : \mathbf{R}^N \times \mathbf{R}^3 \rightarrow \mathbf{R}^3$  which maps points in the frame of the sensor to the world.

## 1.6 Sensor Noise Model

Consider the typical ray-cone sensor model for lasers and other depth sensors. In this model, each point  $\mathbf{x}_i$  actually represents a *cone* of rays emanating from the sensor's origin. If the cone hits any surface in the scene, the depth sensor returns a noisy reading of the range from the sensor to that point. If multiple surfaces intersect the cone, the range reading returned randomly flips between the ranges to each surface with a distribution dependant on the amount of cone area each intersected surface sweeps out. This sensor model makes  $P(\mathbf{x}_i | \epsilon_t)$  quite complicated.

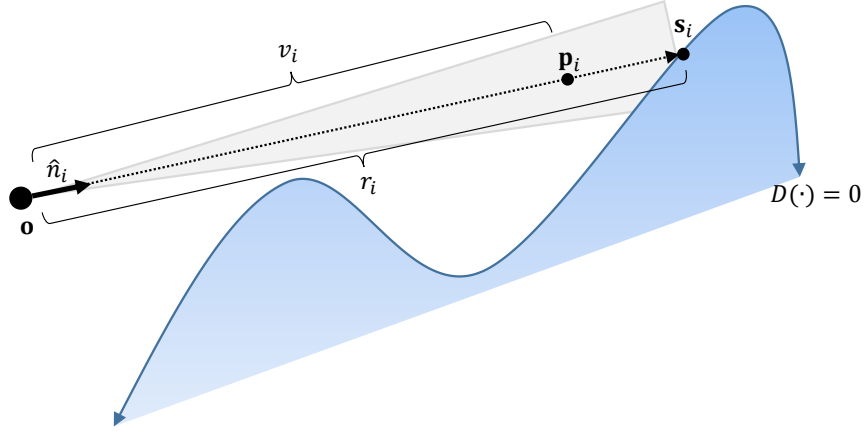


Figure 1: The sensor has origin  $\mathbf{o}$ , a cone is cast out of the sensor with direction  $\hat{n}_i$ . The ray through the center of the cone intersects the world at point  $\mathbf{s}_i$ , and the point implied by the range reading is given by  $\mathbf{p}_i$ . The expected range is  $r_i$ , and the observed range is  $v_i$ .

### 1.6.1 Ray Approximation

We can significantly simplify the problem by modeling the sensor reading as a single *ray* passing through the center of the cone. The origin of the ray is given by  $\mathbf{o} = \mathbf{F}(q_t + \epsilon_t, \mathbf{0})$ , and the endpoint of the ray is given by  $\mathbf{p}_i = \mathbf{F}(q_t + \epsilon_t, \mathbf{x}_i)$ . The ray can be parameterized by arc-length (Fig. 1):

$$R_i(\alpha) = \mathbf{o} + \alpha \hat{n}_i \quad (7)$$

where  $\alpha \in [0, r_i]$  is a range along the ray, and  $\hat{n}_i$  is the unit normal direction of the ray. The likelihood of the ray can then be modeled by considering all points along the ray as independent readings:

$$P(\mathbf{x}_i | \epsilon_t, D) = \prod_{\alpha \in [0, r_i]} P(R_i(\alpha) | D) \quad (8)$$

(note that this assumes each point along the ray is conditionally independent given the distance field  $D$ ). The distribution  $P(R_i(\alpha))$  will differ depending on  $\alpha$ . Points along the ray very near the sensor are much more likely to be unoccupied than points very near the endpoint of the ray. Rays passing through solid objects should have very low probability. Rays with endpoints in empty space should also be very improbable. But rays merely *grazing* surfaces without passing through them should be expected around the silhouettes of objects.

TODO: what is  $P(R_i(\alpha)|D)$ ?

### 1.6.2 Ray Endpoint Approximation

If instead assume that the ray intersects only one surface, we can find the probability that we got a particular sensor reading by considering just the endpoint of the ray:

$$P(\mathbf{x}_i|\epsilon_t) \approx P(\mathbf{p}_i|D) \quad (9)$$

a simple model of  $P(\mathbf{p}_i|D)$  is that the range reading returned by the sensor is corrupted by Gaussian noise. If that's the case, we can determine the expected range reading given the known world model, and compare it to the observed range reading given  $\mathbf{p}_i$ . Assuming the ray actually intersects a surface in the scene, the expected range is given by raycasting to the surface:

$$r_i = \left\| \mathbf{o}_i + \underset{\alpha \in [0, \infty)}{\operatorname{argmin}} [D(\mathbf{o}_i + \alpha \hat{n}_i)^2] \hat{n}_i \right\| \quad (10)$$

Then, if the range reading  $v_i = \|\mathbf{p}_i - \mathbf{o}_i\|$  is drawn from the normal distribution:

$$v_i \sim \mathcal{N}_{r_i, \sigma_i}(x) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp \frac{-(r_i - x)^2}{2\sigma_i^2} \quad (11)$$

where  $r_i$  is the mean of the distribution, and  $\sigma_i$  is its standard deviation, we can compute:

$$P(\mathbf{p}_i|D) = \mathcal{N}_{r_i, \sigma_i}(v_i) \quad (12)$$

Using this approximation ignores the presence of multiple surfaces, and the cone structure of the sensor reading, but it at least takes into account the anisotropic (direction-dependant) range noise of the sensor.

### 1.6.3 Simple Point Approximation

An even simpler approximation of the sensor model ignores both the anisotropy of the noise, and the cone/ray structure of the scene by doing the following: for all surface points in the scene visible to the sensor, a reading is emitted that is the surface point corrupted by linear Gaussian noise.

More precisely, call the set of all surface points visible to the sensor at configuration  $q_t$   $\mathbf{S}_t$ . For a particular point  $\mathbf{s}_i \in \mathbf{S}_t$ , we observe a sensor reading:

$$\mathbf{x}_i = \mathbf{F}^{-1}(q_t, \mathbf{s}_i + \mathbf{d}_i) \quad (13)$$

where  $\mathbf{F}^{-1} : \mathbf{R}^N \times \mathbf{R}^3 \rightarrow \mathbf{R}^3$  takes a point in the world frame and transforms it into the sensor frame given a configuration of the robot, and  $\mathbf{d}_i \sim \mathcal{N}_{\mathbf{0}, \Sigma_D}$  is random linear 3 dimensional Gaussian noise with a covariance of  $\Sigma_D$ .

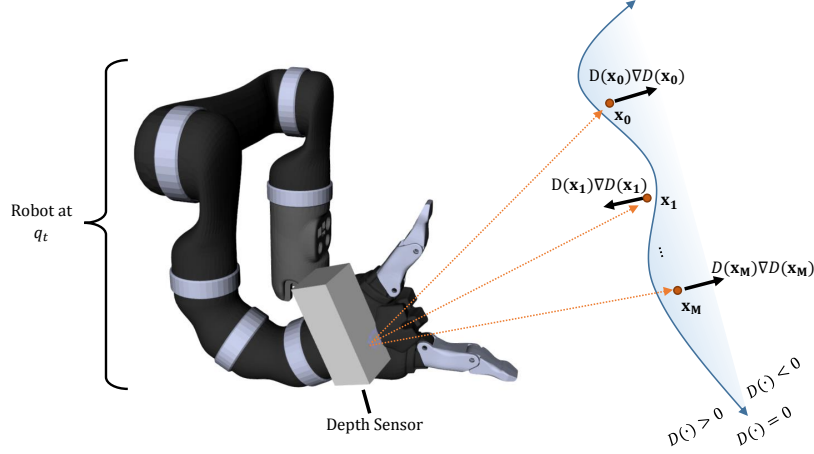


Figure 2: A diagram of the robot sensing a surface with a depth sensor. Rays from the depth sensor are shown as orange dotted lines. The point cloud  $\{\mathbf{x}_1, \dots, \mathbf{x}_M\}$  is used to determine the gradient of the distance field  $\nabla D$  at each point.

This (very inaccurate) model of the sensor essentially just “fuzzes out” all points seen by the robot. Using this model, we can derive the posterior:

$$P(\mathbf{p}_i | D) = \mathcal{N}_{\mathbf{0}, \Sigma_D}(D(p_i)) \quad (14)$$

that is, the probability of a sensor reading is proportional to the observed distance at the end point.

## 1.7 Optimization

We will iteratively build up the TSDF.  $D_t$  is calculated by first estimating a configuration of the robot  $q_t^* = q_t + \epsilon_t^*$  which minimizes the squared error between the sensor measurements  $Z_t$  and the previous TSDF model  $D_{t-1}$ . Then, the sensor measurements are projected back into the world given  $q_t^*$ , and the usual TSDF update is applied.

## 1.8 Computing $\epsilon_t^*$

We can compute  $\epsilon_t^*$  by performing **gradient descent** in the configuration space of the robot. Define the **forward kinematics function**  $F(q_t, Z_t) : \mathbf{R}^3 \rightarrow$

$\mathbf{R}^3$ , which merely transforms the point cloud  $Z_t$  into the world frame, given configuration  $q_t \in C$ . Call

$$F_t = F(q_t + \epsilon_t, z_t)$$

Then, the objective function  $h(\epsilon_t, D_{t-1}) : C \rightarrow \mathbf{R}$  is given by the sum squared distances of all of the projected points in  $F_t$ :

$$h(\epsilon_t, D_{t-1}) = \sum_{x \in F_t} (D_{t-1}(x))^2$$

Minimizing this objective function also maximizes the likelihood of the simple point approximation of the sensor model. (TODO: prove this).

### 1.8.1 Computing the Gradient

Now, we want to find the partial differential of  $h$  with respect to  $\epsilon_t$ :

$$\frac{\partial h}{\partial \epsilon_t} = \frac{\partial}{\partial \epsilon_t} \sum_{\mathbf{x} \in F_t} (D_{t-1}(\mathbf{x}))^2 \quad (15)$$

$$= \sum_{\mathbf{x} \in F_t} \frac{\partial}{\partial \epsilon_t} (D_{t-1}(\mathbf{x}))^2 \quad (16)$$

$$= 2 \sum_{\mathbf{x} \in F_t} D_{t-1}(\mathbf{x}) \frac{\partial \mathbf{x}}{\partial \epsilon_t} \nabla D_{t-1}(\mathbf{x}) \quad (17)$$

And, since  $\frac{\partial \mathbf{x}}{\partial \epsilon_t}$  is the change in  $\mathbf{x}$ , a projected sensor point, with respect to  $\epsilon_t$ , the configuration of the robot at time  $t$ , we have (with some handwaving):

$$\frac{\partial \mathbf{x}}{\partial \epsilon_t} = J_{\mathbf{x}} \in \mathbf{R}^{3 \times N}$$

Where  $J_{\mathbf{x}}$  is the serial manipulator Jacobian computed for the point  $\mathbf{x}$ , as though it were rigidly attached to the manipulator by the ray connecting  $\mathbf{x}$  to the sensor.  $J$  has the form:

$$J_{\mathbf{x}} = \begin{bmatrix} \left| \frac{\partial \mathbf{x}}{\partial \epsilon_t(1)} \right| & \cdots & \left| \frac{\partial \mathbf{x}}{\partial \epsilon_t(N)} \right| \\ \vdots & \ddots & \vdots \end{bmatrix} \quad (18)$$

And so:

$$\frac{\partial h}{\partial \epsilon_t} = 2 \sum_{x \in F_t} D_{t-1}(x) \mathbf{J}_x^T \nabla D_{t-1}(x)$$

We will call this quantity  $\nabla h(\epsilon_t)$ . It has a nice physical interpretation: imagine all the points in the point cloud are attached rigidly to the robot manipulator on rods. At the end of each rod  $x$ , apply a force  $D_{t-1}(x) \nabla D_{t-1}(x)$ . The resulting torque on the robot's joints is proportional to  $\nabla h(\epsilon_t)$  by a factor of 2 (Fig. 2).

### 1.8.2 Gradient Descent

Now, we just follow the update rule, setting  $\epsilon_t^{(0)} = \epsilon_{t-1}$ :

$$\epsilon_t^{(i+1)} = \epsilon_t^{(i)} - \lambda \nabla h(\epsilon_t^{(i)})$$

Where  $\lambda$  is a learning rate. We follow the gradient until convergence, yielding  $\epsilon_t^*$ , treating the joint limits of the robot as a constraint.