A Stability Loss Derivation

Recall that V_o parameterizes a multivariate Bernoulli distribution over binarized voxel grids. For sample $v \sim V_o$, we define $M(v)$ to be the center of mass of v. Furthermore, let $i \in d^3$ index the voxel grid, and $S = \{s : s \perp \vec{g}\}$ be the set of directions perpendicular to $\vec{g}$. Then, for each $s \in S$, let i^s and $M^s(v)$ be the projections of i and $M(v)$ onto the plane defined by s and $\vec{g}$ that passes through the origin. We denote $H_s(i)$ as the set of voxels belonging to other objects that support i in direction s, which can happen when i is directly above or leaning against such voxels. Finally, $V_{\bar{o}}$ is the probabilities of other objects output by the 3D reconstruction network.

Given this notation, we can define our stability loss to be the probability that v is stable. Let $E(v)$ be the event that v is stable. Then

$$P(E(v)) = P(v \texttt{ stable along all directions})$$

Here we introduce our first approximation by discritzing the set of all directions into a finite set of evenly space directions S. In our implementation, we used $|S| = 25$. Then

$$P(E(v)) = \prod_{s_i \in S} P(v \texttt{ stable along } s_i \texttt{ | } v \texttt{ stable along } s_0, s_1, ..., s_{i-1})$$

We introduce a second approximation, that stability along different directions is independent. Then

$$P(E(v)) = \prod_{s_i \in S} P(v \texttt{ stable along } s_i) = \prod_{s_i \in S} 1 - P(v \texttt{ not stable along } s_i) = \prod_{s_i \in S} 1 - u_s$$

$$u_s = P(\texttt{no voxel } i \in d^3 \texttt{ makes } v \texttt{ stable} = \prod_{i \in d^3} 1 - P(\texttt{i makes } v \texttt{ stable})$$

$$= \prod_{i \in d^3} 1 - P(i^s > M^s(v)) P(i \texttt{ is supported by another voxel in direction } s)$$

$$P(E(v)) = \prod_{s_i \in S} \prod_{i \in d^3} \Big[1 - V_o(i) P(i^s > M^s(v)) h_s(i)\Big], \quad h_s(i) = 1 - \prod_{i' \in H_s(i)} \Big(1 - V_{\bar{o}}(i')\Big)$$

$P(i^s > M^s(v))$ is just the cdf of $M^s(v)$, which is a linear combination of Bernoulli variables. This is a generalization of the Poisson Binomial distribution, the cdf of which is inefficient to compute exactly [cite]. Therefore, we use a normal approximation of the cdf over the sum of weighted Bernoulli variables:

$$P(i^s > M^s(v)) \approx \Phi_{\mu,\sigma}(i^s), \quad \mu = \sum_{i \in d^3} i^s v(i), \quad \sigma = \Big[\sum_{i \in d^3} (i^s_v)^2 v(i)(1 - v(i))\Big]^{1/2}$$

Now we compute the derivative of $P(E(v))$ with respect to voxel i. First, we take the log of $[P(E(v))$, noting that maximizing the log also maximizes $[P(E(v))$. Then

$$\frac{d \log P(E(v))}{dV_o(i)} = \sum_{s \in S} \frac{-u'_s}{1 - u_s}$$

$$u'_s = -P\Big(i^s > M^s(v)\Big) h'_s(i) \prod_{i_o \in d^3, i_o \neq i} \Big[1 - P\Big(i^s_o > M^s(v)\Big) V(i_o) h'_s(i_o)\Big]$$

$$\hat{h}_s(i_o) = 1 - \prod_{i_b \in H_s(i)} \Big[1 - V_{\bar{o}}(i_b)\Big]$$

Note that

$$\frac{d \log P\Big(i^s > M^s(v)\Big)}{dV_o(i)} = 0$$

as adding (or increasing the chance that a number appears) in a set of numbers cannot change whether the mean of that set of numbers is greater than, equal to, or less than that number.

The derivative is numerically unstable since the product $\prod_{i_o \in d^3, i_o \neq i} \Big[1 - P\Big(i_o^s > M^s(v)\Big) V(i_o) h_s'(i_o)\Big]$ has millions of terms in practice and will therefore often underflow to zero. To fix this we note that, during inference, if any voxel has $v(i) \geq 0.5$ then we set it as existing, and the notion of continuously existing voxels is only an approximation to aid differentiation. Therefore, we may write

$$\frac{d \log P(E(v))}{dV_o(i)} = \sum_{s \in S} \frac{-u_s'}{1 - u_s}$$

$$u_s' = -P\Big(i^s > M^s(v)\Big) \hat{h}_s(i) \prod_{i_o \in d^3, i_o \neq i} \Big[1 - P\Big(i_o^s > M^s(v)\Big) \mathbb{1}\{V(i_o) \geq 0.5\} \hat{h}_s(i_o)\Big]$$

$$\hat{h}_s(i_o) = 1 - \prod_{i_b \in H_s(i)} \Big[1 - \mathbb{1}\{V_{\bar{o}}(i_b) \geq 0.5\}\Big]$$

completing our derivation of the stability loss.

B Connectivity Loss Derivation

Our connectivity loss imposes a prior on object shape even in occluded regions by allowing the network to infer connections between disjoint parts of observed objects. This complements the stability objective which frequently infers occluded bases of objects. We define v to be connected if for every pair of existent voxels a, b, there exists a path $t = \{i_0, i_1, \dots\}$ between a and b. The probability that a path t exists in v is $P(t) = \prod_{i \in t} V_o(i)$. Let $T(a, b)$ be the set of all possible paths between a and b, $C(v)$ be the event that v is connected, and $C(a, b)$ be the event that there is a path between a and b. Then we define our connectivity objective as the probability that, for every pair of vertices a, b, a exists and b exists and a and b are connected, or not (a exists and b exists), conditioned on the probabilities. We introduce the approximation that voxel connectivity is independent between pairs of voxels. Then

$$P(C(v)) = \prod_{a,b \in d^3, a \neq b} \Big[V_o(a) V_o(b) P(C(a, b)) + 1 - V_o(a) V_o(b)\Big]$$

Taking the log and then derivative with respect to a voxel c we get

$$\frac{d \log P(C(v))}{dV_o(c)} = \sum_{a,b \in d^3, a \neq b \neq c} \frac{V_o(a) V_o(b) \frac{d}{dV_o(c)} P(C(a, b))}{V_o(a) V_o(b) P(C(a, b)) + 1 - V_o(a) V_o(b)}$$

Computing $P(C(a, b))$ requires us to compute combinatorially many paths and is infeasable. We note that many paths are highly unlikely and contribute little to this probability. Therefore, we introduce the approximation of computing $P(C(a, b))$ using the most likely a, b path and the most likely a, b path that includes c; that is

$$P(C(a, b)) = \bigvee_{t \in T(a,b)} P(t) \approx P(t^* \vee t^c)$$

$$P(t^* \vee t^c) = \begin{cases} P(t^*), & t^* = t^c \\ 1 - (1 - P(t^*))(1 - P(t^c)), & t^* \neq t^c \end{cases}$$

and

$$\frac{dP(C(a,b))}{dV(i)} = \begin{cases} \frac{P(t^s)}{V(i)}, & t^s = t^i \\ \frac{P(t^i)}{V(i)}(1 - P(t^s)), & t^s \neq t^i \end{cases}$$

completing our derivation.

C System Implementation Details

Our system is implemented in PyTorch. We train using ADAM for 104,000 iterations with a batch size of 16 and a learning rate of 2e-4. Other training parameters are identical to those used in GenRE. We coarsen the predicted voxel grid by a factor of 8 during connectivity loss computation for efficiency.

D Cluttered Reconstruction Benchmark Details

We introduce a cluttered reconstruction benchmark containing 2318794 training and 484161 test reconstruction instances of cluttered tabletop objects. Objects are drawn from the shapenet xyz categories. Scene generation uses the following randomization parameters:

table	sampled randomly from shapenet
table objects	sampled from selected shapenet categories [1]
number table objects	uniform(3,20)
table and object color	uniform random
object scale	uniform(0.5, 4)
object rotation	75% chance of being none/default (upright object) 25% chance of being uniform random
object drop position center	uniform random point on table
object drop position std dev	0.1m
object drop height	table height+0.2m
camera position x,y	uniform in shell between 0.125 and 1.5m centered on table
camera position z	uniform in half-shell between 0.125 and 1.5m centered on table, on top of table
camera lookat	random point in sphere of radius min(camera distance from table/2, 0.25)

Parameters are selected according to the above table, objects are dropped onto the table, physics is stepped forward for several seconds so object settle, and then 20 random camera views are taken. Each object present in each view is saved as a prediction instance, with RGBD, mask, and model information. Camera views with no objects visible are discarded.

[1] Selected shapenet training categories: bag, traveling bag, travelling bag, grip, suitcase, birdhouse, bottle, bowl, camera, photographic camera, can, tin, tin can, cap, clock, computer keyboard, keypad, dishwasher, dish washer, dishwashing machine, helmet, jar, knife, laptop, laptop computer, microwave, microwave oven, mug, pillow, remote control, remote, telephone, phone, telephone set, cellular telephone, cellular phone, cellphone, cell, mobile phone. Test categories: display, video display, loudspeaker, speaker, speaker unit, loudspeaker system, speaker system, washer, automatic washer, washing machine, printer, printing machine, ashcan, trash can, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin, trash barrel, trash bin.

E Cluttered Manipulation Benchmark

We generate 2574 manipulation tasks: 26 objects $\times$ three tasks (grasping, pushing, and rearrangement) $\times$ eleven levels of target object visibility $\big([0.0 - 0.1), [0.1 - 0.2), ..., [0.9 - 1.0), 1.0)\big) \times 3$. We generated each task by first placing the target manipulation object at a fixed location. We then added distract objects directly in front of the target object, drawing distractor object y positions from

a normal distribution with a center at the target object position and a y standard deviation of 0.1m, until a distractor object was added that caused the desired level of occlusion. We then added more distractor objects using the randomization parameters in the table below.

grasping: In this task the robot needed to grasp and object and lift it at least 5cm off the table.

pushing: In this task the robot needed to push an object 30cm to the right and 5cm forward. The robot succeed if the object ended within 2.5cm of the target position.

rearrangement: In this task the robot needed to grab and object and move it 20cm forward and 20cm to the left. The robot succeed if the object ended within 2.5cm of the target position.

distractor object type	sampled from selected unseen objects [2]
number objects	uniform(1,7)
object color	uniform random
distractor object scale	uniform(0.5, 2)
object rotation	upright, uniform random rotation
distractor object position std dev	0.25m

[2] Target manipulation objects:
YCB Objects: master chef can, cracker box, sugar box, tomato soup can, mustard bottle, apple, orange, pitcher base, bleach cleanser, bowl, mug, wood block, tennis ball, rubiks cube
Objects from internet repository: cup (x2), glass (x3), vase, lamp (x5), trophy

F MPPI Parameters

We used the following parameters for MPPI:

steps	200
paths explored per step	25
path length	5
timestep length	0.02

We used a stateful reward function for MPPI. The reward function had three states: `ungrasped`, `grasping`, and `grasped`. The state starts as `ungrasped`, and changes to `grasping` when

```
ungrasped=number steps<10 or (distance(hand, target object)>2.5mm and
distance(hand, target object) decreased in the last 10 steps)
```

becomes false. When the state becomes `grasping`, the robot will close its gripper for 75 timesteps, and then the state will become `grasped`.

```
ungrasped and grasping: Reward=-distance(robot hand, target object)-5*(hand
height-38cm)-0.5*angle(robot hand, z axis)
```

```
grasped: Reward=-distance(robot hand, target object)-distance(target
manipulation object position, current manipulation object position)
```

G Segmentation Quality Analysis

In Figure 1 we show the ratio of success with imperfect segmentations generated by UOIS to success with ground truth segmentations, broken down by segmentation IoU on the x axis.

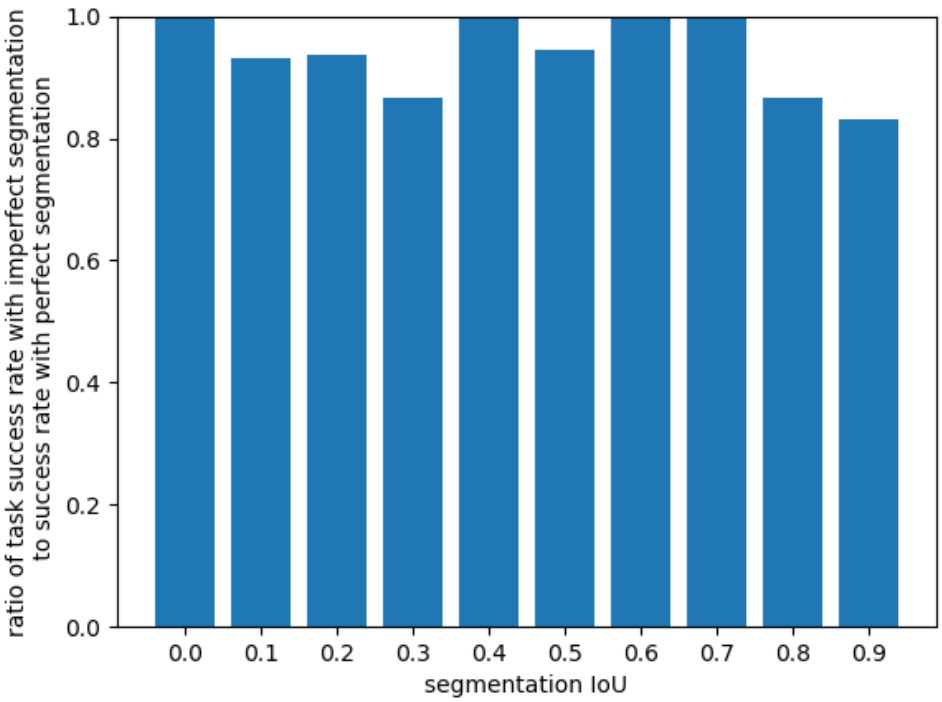

Figure 1: Analysis of impact of segmentation quality on task success rate