3D features maps

1 The projection model

Let

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix},$$

denote the intrinsic matrix of the camera. Let N denote the number of pictures taken by the scanner, and R_i, T_i denote the extrinsics parameters of the camera for the ith shot, i ranging from 1 to N. For any point $x \in \mathbb{R}^3$ in the world coordinates, the projection of x onto the ith shot is given, in homogeneous coordinates, by $Y = K(R_i x + T_i)$. If

$$Y = \begin{pmatrix} x \\ y \\ z \end{pmatrix},$$

then $P_i(X) = (x/z, y/z)$ is the projection in pixel coordinates of point X.

Let $\Omega = [0, W] \times [0, H]$ be the domain of each image, where W and H are the width and height of the image in pixel respectively.

Then, an image is viewed as a (continuous) function

$$I_i : \Omega \to \mathbb{R}^3,$$

So that $I_i(x, y)$ is the RGB value of the ith shot at pixel location x, y. Let $\Omega_i \subset \mathbb{R}^3$ denote the cone visible in image i:

$$\Omega_i = P_i^{-1}(\Omega) = \{X \in \mathbb{R}^3 : P_i(X) \in \Omega\}.$$

2 2D features

2.1 Color or geometry based features

TODO

2.2 Learned semantic segmentation

TODO

3 Backprojections

From the previous section, we obtain, for each

$$M_i : \mathbb{R}^2 \to \mathbb{R}^d$$

is the feature map corresponding to picture i. d is the number of features. For example, in the case of Vesselness features, $d = 1$. If 2D features correspond to multi-class semantic segmentation, d is the number of classes. We renormalize all feature maps so that they belong to the interval $[0, 1]$.

$$M_i(x, y)_k \in [0, 1], \forall (x, y) \in \mathbb{R}^2, \forall i \in \{1, \cdots, N\}, \forall k \in \{1, \cdots, d\};$$

We can imagine that one pixel in an image belongs to several classes, for example if we want to ignore occlusion by *guessing* occlusion of classes by others. Moreover the probabilities of all classes can sum to less than one since the void since the 3D space usually does not correspond to any class in the 3D space.

We can see $M_i(x, y)_k$ as the probability that, given picture I_i, pixel (x, y) is the projection of a point belonging to class k.

3.1 Visual hull

Let's assume in this subsection that $d = 1$. We choose a threshold T and compute a binary labeling function as such:

$$L_i(x) = \begin{cases} 1 & \text{if } M_i(x, y) > T; \\ 0 & \text{otherwise.} \end{cases}$$

Then we obtain a binary volume by intersecting the projection cones of each camera: for $X \in \mathbb{R}^3$.

$$L(X) = \begin{cases} 1 & \text{if } \forall i \text{ s.t. } P_i(X) \in \Omega, L_i(P_i(X)) = 1; \\ 0 & \text{otherwise.} \end{cases}$$

We can also add the condition that at least M cameras see the point X, i.e. that $L(X) = 0$ except if there exist distinct indices $(i_1, \cdots, i_M)$ such that $P_i(X) \in \Omega$.

This is very classical in the litterature.

The main problems with this approach are first that it's very sensitive to the threshold T, which might have to be selected differently for different cameras

due to e.g. lighting conditions. Any false negative (i.e. a point X belonging to the true volume such that $L_i(P_i(X)) = 0$ will lead to a wrong labeling in the 3D space. Another related problem is that it's hard to make it work for more than one class, as soon as there is occlusion between classes.

3.2 Independent backprojections

To overcome these issues, a simple modification of the backprojections is to get rid of the threshold T and compute an average over different views. To get a more contrasted view, a average of logarithms of the values is used instead of a normal average:

$$L(X) = \sum_{i:X\in\Omega_i} \log M_i(P_i(X)).$$

L is the 3D feature map. This can be viewed as a probability that a point belong to the selected class, assuming that all projections are independent random variables. This formula can extend to more than one class.

4 End-to-end learning of 3D features

In this section, we go back to a discrete representation of images and volumes, e.g. for images

$$I_i = (I_i(x, y))_{i\in\{0,\cdots,W-1\},j\in\{1,\cdots,H-1\}}.$$

The idea is to learn an end-to-end neural network from a fixed setup of cameras:

We decompose it into two main sub-functions: first, we compute a 2D feature map from RGB pictures. Let's call n_{2D} the number of 2D features, then we define an operator

$$\varphi : \mathbb{R}^{W\times H\times 3} \to \mathbb{R}^{W\times H\times n_{2D}}.$$

Then the segmentation is defined as a function from an array of 2D feature map, to a 3D multi class classifier:

$$\psi : \mathbb{R}^{W\times H\times n_{2D}\times N} \to \mathbb{R}^{N_x\times N_y\times N_z\times d}.$$

where (N_x, N_y, N_z) is the size of the voxelized 3D volume and d is the number of classes, as in the previous section.

Then, the idea is to train a discriminator minimizing a Loss on the 3D volume on simulated data, for example using the LPY virtual plants or anything else... The loss can be a cross entropy, classical in the litterature:

$$\widehat{\varphi}, \widehat{\psi} = \text{argmin}_{\varphi,\psi} H_V(\psi(\varphi(I_1), \varphi(I_2), \cdots, \varphi(I_N))),$$

where $V \in \mathbb{R}^{N_x \times N_y \times N_z \times d}$ is the ground truth labeled voxel volume and H is the cross-entropy function defined as:

$$H_V(V') = \sum_{j=1}^{d} \sum_{X} V(X)_j \log V'(X)_j + (1 - V(X)_j) \log(1 - V'(X)_j),$$

with the X sum taken over all voxels.