Supplementary material

Vision-as-Inverse-Graphics: Obtaining a Rich 3D Explanation of a Scene from a Single Image

A. Probabilistic Chain Rule within PHNs

In a case of a multidimensional grid-based GMM, it may be undesirable to make predictions directly in the $d$-dimensional Hough space, as this has $O(N^d)$ mixture components, where $N$ is the average number of components per dimension. Assuming the number of hidden units in the layer before the softmax one is on average $U$, the number of weights in the softmax layer is $O(Un^d)$, which would likely lead to over-fitting.

Fortunately PHNs allows us to decompose $p(z|x_i)$ into a number of simpler networks via the probabilistic chain rule. We decompose our network which in a direct 3D case has $O(N^3)$ weights into two networks with a GMM of dimension 2 and 1, so $O(Un^2+UN) = O(Un^2)$ weights. In general we can decompose any grid-based PHN to a single PHN for each dimension, leading to a total of $O(dUn)$ weights in the last layers. Thus using more but smaller networks we can obtain a GMM which if predicted directly would lead to a network that would have too many weights compared to the dataset size we have.

A.1. PHN Decomposition

We use the following decomposition, where $z = z_1 \cup z_2$:

$$p(z|x_i) = p(z_1, z_2|x_i) = p(z_1|z_2, x_i)p(z_2|x_i).$$

We note that we need a decomposition-composition procedure as we do not simply multiply the values, but create the whole density that represents the product in the whole space. We decompose $H = p(z|x_i)$ into two PHNs:

$$H_1(z_1|z_2, x_i) = p(z_1|z_2, x_i),$$

$$H_2(z_2|x_i) = p(z_2|x_i).$$

$H_1$ predicts the probabilistic vote conditioned on the observation and a Hough space variable (or a set of variables) given which there is a simpler solution.

A.2. PHN Composition

Suppose $\text{dim}(H_1) = d_1$ and $\text{dim}(H_2) = d_2$, with $\text{dim}(H) = d = d_1 + d_2$. We want to compose the GMM obtained from $H_1$ whose components are indexed by $(i_{z_1})$ and $H_2$ whose components are indexed by $(i_{z_2})$ into a $d$-dimensional GMM represented by $H$ and indexed by $(i_{z_1}, i_{z_2})$. We can obtain $H = H_1 \otimes_{\text{PHN}} H_2$ representing the density $p(z|x_i)$ from these two models as follows:

we iterate through the values in the $z_2$ dimension at the positions of the components as defined by the grid, denoted by $z_2(i_{z_2})$. Then we evaluate $H_1$ by conditioning on each of these values, obtaining $N_{z_2}$ GMM slices in the $z_1$-space. We also predict GMM in the $z_2$-space using $H_2$. The GMM mixing coefficients are given by:

$$H(i_{z_1}, i_{z_2}|x_i) = H_1(i_{z_1}|x_i, z_2(i_{z_2}))H_2(i_{z_2}|x_i),$$

which directly represents a probability distribution in the desired grid, where each component mean is at the location of respective indices and the covariance matrix is the same as before the decomposition.

B. Integration for Log-Likelihood Computation in the Joint Posterior

$$p(z|X) \propto \prod_{i=1}^n p(z_1|x_i) \prod_{i=1}^{n-1} p(z_2|x_i),$$

To perform the inference, find the MAP value and sample from the posterior $p(z|X)$ as in eq. 5 we do not need to find a normalising constant. However, we need it to compute the log-likelihood for the purposes of evaluating the framework.

We wish to compute the normalising constant $1/Z(X)$ for the RHS of eq. 5. We estimate $Z(X)$ by numerical integration of the predicted density in the $z$-space.

Figure 1. Average error using a uniform sampling (red) and IS (black) in the 2D case.

The function was implemented in C to make the computations faster, and we experimented with integral computation using classical quadrature. We found that in a case of 3 dimensions it was not converging in a desirable time even for
Table 1. The configurations of the main models, which are 18 or 19 layers deep, and learning rates. Layer types are: C – Convolutional, F – Fully connected, Sigm – Sigmoid and Softmax. The layers are described as follows: layer type - number of units - filter size.

<table>
<thead>
<tr>
<th>Detector</th>
<th>PHNs</th>
<th>Predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Scale</td>
<td>$H^1$</td>
</tr>
<tr>
<td>VGG-16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C-50-6</td>
<td>C-50-6</td>
<td>C-50-6</td>
</tr>
<tr>
<td>C-50-6</td>
<td>C-50-6</td>
<td>C-50-6</td>
</tr>
<tr>
<td>C-50-6</td>
<td>C-50-6</td>
<td>C-50-6</td>
</tr>
<tr>
<td>F-200</td>
<td>F-200</td>
<td>F-50</td>
</tr>
<tr>
<td>Softmax-2</td>
<td>Sigm-1</td>
<td>F-30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Softmax-135</td>
</tr>
<tr>
<td>Learning rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class</td>
<td>Scale</td>
<td>$H^1$</td>
</tr>
<tr>
<td>0.001</td>
<td>0.0002</td>
<td>0.001</td>
</tr>
</tbody>
</table>

C. Details of the networks

Table 1 shows the network configurations and learning rates used for training. We use all 13 convolutional layers of VGG-16 as the core but on 128 × 128 pixel input. VGG uses padding, but in our convolutional layers we do not use it. The fully connected layers of the detector networks are implemented as filter 1 × 1 convolutional layers, so they can be efficiently applied in a sliding window manner. We use dropout in the detector networks with $p = 0.5$ and in predictor networks with $p = 0.2$, in the convolutional layers on top of VGG ones. In PHNs a better convergence was obtained when the network was pre-trained using a lower variance scaling factor $\beta$ to encourage the network to activate all Gaussian components. For azimuth prediction recall that we predict the sine and cosine of the angle.

D. Re-projection error

Figure 2 shows the examples of overlaid checkerboards using ground-truth and predicted cameras.

Figure 2. Re-projection error, the green checkerboard is the ground-truth, the magenta one is the prediction, and the pink lines show the errors. The white background is the image frame. Examples of re-projection error (from left): 6.5% and 2.0%.
E. Further PHN results

Figure 3 gives a detailed comparison of Single-PHNs vs Multi-PHNs for both the 2D (top row) and 3D (bottom row) cases. The left column shows log likelihoods, the right re-projection error. Blue dots indicate the mean Single-PHN prediction for a given image, this is to be compared with the Multi-PHN score (horizontal axis). For the log likelihood case higher is better, so when the Multi-PHN wins the blue dots lie below the diagonal line. For the re-projection error lower is better, so when the Multi-PHN wins the blue dots lie above the diagonal line. For each text image we show all Single-PHN predictions as well as the mean (blue dot) so that the variability can be seen.

F. More examples of prediction

Figure 4 presents more examples of prediction for real images.
Figure 4. Results on real scenes. The bottom 4 examples have errors in detection. In each the order is the input image, predicted 3D scene, and the result after iterative refinement.