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

Lukasz Romaszko[1] Christopher K.I. Williams[1,2] Pol Moreno[1] Pushmeet Kohli[3,*]
[1]School of Informatics, University of Edinburgh, UK [2]The Alan Turing Institute, UK [3]DeepMind
lukasz.romaszko@gmail.com, c.k.i.williams@ed.ac.uk
p.moreno-comellas@sms.ed.ac.uk, pushmeet@google.com

Abstract

We develop an inverse graphics approach to the problem of scene understanding, obtaining a rich representation that includes descriptions of the objects in the scene and their spatial layout, as well as global latent variables like the camera parameters and lighting. The framework's stages include object detection, the prediction of the camera and lighting variables, and prediction of object-specific variables (shape, appearance and pose). This acts like the encoder of an autoencoder, with graphics rendering as the decoder. Importantly the scene representation is interpretable *and is of variable dimension to match the detected number of objects plus the global variables. For the prediction of the camera latent variables we introduce a novel architecture termed Probabilistic HoughNets (PHNs), which provides a principled approach to combining information from multiple detections. We demonstrate the quality of the reconstructions obtained quantitatively on synthetic data, and qualitatively on real scenes.*

1. Introduction

Our goal in this paper is the classic computer vision task of *scene understanding*, by which we mean obtaining a representation that includes descriptions of the objects in the scene (shape, appearance and pose) and their spatial layout, as well as global factors like the camera parameters and lighting. This work is carried out in a vision-as-inverse-graphics (VIG) or analysis-by-synthesis framework, where we seek to *reconstruct* the input image by using a graphics engine to render the scene description output by the analysis stage. Analysis-by-synthesis is an old idea (see e.g. [8, 9]) but it can be reinvigorated using recent advances in deep learning for the analysis stages.

Our work is summarised in Fig. 1. Object detectors (stage A) are run over the input image, producing a set of detections. We then predict the scene latent variables, consisting of the camera parameters (stage B), object descriptions (stage C) and global parameters (stage D), and back-project objects into the scene given the predicted camera (stage E). To solve the problem of inferring the camera parameters we introduce a novel Probabilistic HoughNets (PHNs) architecture, which carries out a principled integration of information from multiple object detections. The scene latent variables can then be rendered by a graphics engine to produce a predicted image, and by optimizing them the match to the input image can be refined iteratively.

One highly attractive aspect of the VIG framework is that it produces a *compact* and *interpretable* representation of the scene in terms of an arbitrary number of objects. Such a representation can be useful e.g. if we wish to edit/modify objects in the scene. Much recent work on unsupervised learning of images such as the variational autoencoder [13, 20] or the generative adversarial network (GAN) [7] use a *fixed dimensional* representation of the image in the latent code. This is reasonable for encoding a single object, although it is also helpful to disentangle factors of variation like shape, appearance and pose. However, such fixed dimensional representations are much more problematic for a whole scene. Our target representation is a *scene graph* (see e.g. [2, sec. 9.8]) as used in computer graphics for describing a scene in terms of objects as well as lighting and cameras *etc*. (for instance X3D[1] is a modern format for scene graphs). The stages in Fig. 1 act like the *encoder* of an autoencoder, with the decoder being graphics rendering.

Our contributions are as follows: (i) we develop a VIG-based framework to the problem of scene understanding in 3D from a single image, i.e. predicting the camera and illumination as well as the 3D pose of each object—this is to be contrasted with methods that simply predict 2D image-based bounding boxes or pixel labelling; (ii) we develop accurate recognition models trained on latent variables of realistic synthetic images in a way that they transfer to work with

[*] This work was carried out when PK was at Microsoft Research.

[1]https://en.wikipedia.org/wiki/X3D

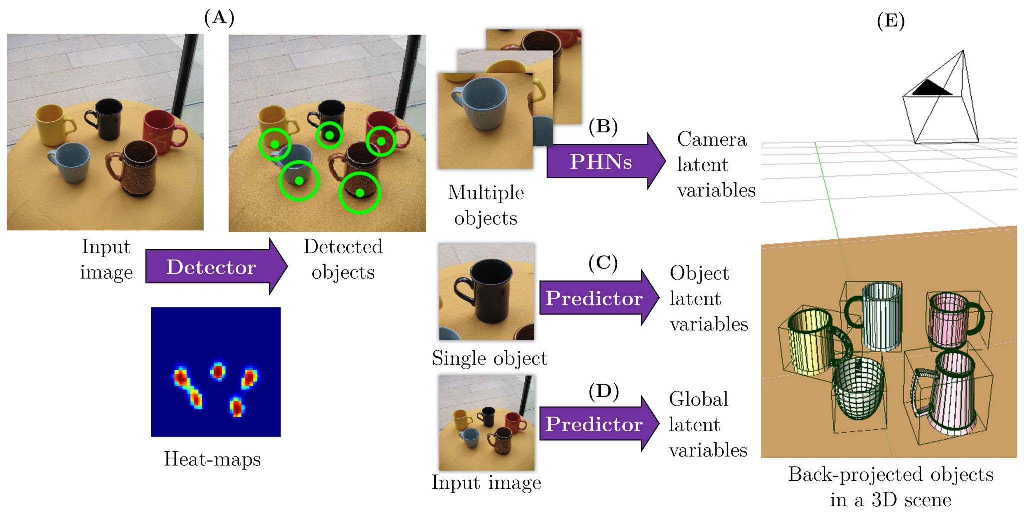


Figure 1. Overview: (A) Objects are detected in the image (green dots: contact points), which jointly predict the camera parameters (B) using PHNs. Then (C) object latent variables and (D) global parameters (e.g. lighting) are predicted. These allow back-projection of the objects into the scene (E), and iterative refinement.

real images; (iii) we consider the camera estimation problem in a novel manner, where given a set of object detections in an image, we seek to estimate the camera parameters by combining information from all of the detections – previous VIG camera-estimation approaches for multiple objects considered only a restricted problem with fixed camera elevation and distance relative to the ground plane; (iv) we introduce PHNs—a coherent and robust probabilistic method for combining predictions from multiple voting elements which represent the predictions using a mixture of Gaussians (not bins), and make these predictions with a deep neural network.

2. Methods

We first introduce Probabilistic HoughNets and their use for estimating the camera parameters in sec. 2.1, as per label B in Fig. 1. In sec. 2.2 we describe the networks used for object detection (label A) and the prediction of object-specific (label C) and global (label D) latent variables. Eventually, we present how we obtain the final 3D scene representation (label E).

2.1. Probabilistic HoughNets

We introduce *Probabilistic HoughNets* (PHNs) in order to combine information from a number of *voting elements*[2]. Our primary example below is the estimation of the camera parameters $\mathbf{z}$ based on detections of multiple objects $\{\mathbf{x}_i\}$ in a scene. Each voting element i has a local descriptor $\mathbf{x}_i$, and provides evidence via a mixture of Gaussians in the *Hough space* for the instantiation parameters $\mathbf{z}$. With Hough transforms, the predictions of a voting element can lie (in the noise-free case) on a low-dimensional manifold in Hough space. When noise is present the manifold is “fuzzed out” in the remaining dimensions. In our case a 1D manifold arises from the trade-off between the distance of an object from the camera and the camera's focal length (or zoom) in creating an object image of a given size. Each detection provides one such manifold and by intersecting them (probabilistically) we reduce the uncertainty on the camera parameters.

[2] We use the terminology from [4].

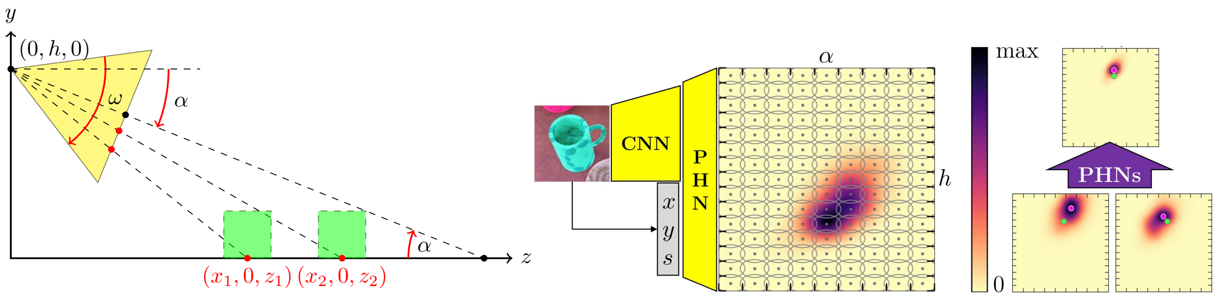
The Hough transform (HT) is a classic computer vision algorithm dating back to 1962 [10]. It was originally proposed for detecting lines, but was then generalized [3] to arbitrary templates. Let the set of voting elements $\{\mathbf{x}_i\}$ be denoted by X. Stephens [23] pointed out how the HT can be made probabilistic by writing

$$p(\mathbf{z}|X) = \frac{p(\mathbf{z})p(X|\mathbf{z})}{p(X)} = \frac{p(\mathbf{z})}{p(X)} \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{z}), \tag{1}$$

assuming that the $\mathbf{x}_i$'s are conditionally independent given $\mathbf{z}$. By taking logs of this equation Stephens shows how terms involving $\log p(\mathbf{x}_i|\mathbf{z})$ can be added up, mirroring the standard Hough space accumulator. If $\mathbf{x}$ is high dimensional (e.g. an image patch) and $\mathbf{z}$ is low dimensional it makes more sense to model $p(\mathbf{z}|\mathbf{x}_i)$ rather than $p(\mathbf{x}_i|\mathbf{z})$. Applying Bayes' theorem again to eq. 1 we obtain

$$p(\mathbf{z}|X) \propto \frac{\prod_{i=1}^{n} p(\mathbf{z}|\mathbf{x}_i)}{p(\mathbf{z})^{n-1}}, \tag{2}$$

ignoring terms involving $p(X)$ or $p(\mathbf{x}_i)$ which are fixed given the image evidence. This argument in eq. 2 was given in [1, §3.5] and [4]. An important aspect of the Hough transform is the ability to deal with outliers; in the framework above this can be handled by robustification, replacing



(A) (B) (C)

Figure 2. Camera parameters and PHNs: (A) camera set-up where $\mathbf{z} = (\alpha, h, \omega)$; (B) Probabilistic HoughNets framework. A CNN is given the detected patch for each object, the input to a PHN is its last dense layer plus position (x, y) of the image patch and size s (projection scale) of the object. Here the density plots represent predictions in (α, h)-space conditioned on ω; the dots are Gaussian means, ellipses show standard deviations; (C) combining multiple PHNs.

$p(\mathbf{z}|\mathbf{x}_i)$ in eq. 2 with $\gamma p(\mathbf{z}|\mathbf{x}_i) + (1 - \gamma)p(\mathbf{z})$ for $\gamma \in [0, 1]$, where $p(\mathbf{z})$ is a broad prior over $\mathbf{z}$-space.

The Probabilistic HoughNet represents $p(\mathbf{z}|\mathbf{x}_i)$ using a mixture of Gaussians with the means $\{\boldsymbol{\mu}_j\}$ arranged in a grid in Hough space:

$$p(\mathbf{z}|\mathbf{x}_i) = \sum_j \beta_j(\mathbf{x}_i)\mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}), \tag{3}$$

where j is an index over the grid, and the $\beta_j(\mathbf{x}_i)$'s are $\mathbf{x}$-dependent mixing coefficients. These are implemented by using a softmax layer at the output of a deep neural network, as in a mixture of experts [11], but where only the mixing proportions but not the $\boldsymbol{\mu}_j$'s or $\boldsymbol{\Sigma}$ depend on $\mathbf{x}$. We train the neural network by maximising the log-likelihood of the ground-truth instantiation parameters given a voting element.

If the Hough space dimension d is high and there are N components per dimension in the grid, then the softmax layer will have N^d outputs parameterized by a large number of weights, which could lead to overfitting. In this case one can make use of the chain rule, e.g. splitting $\mathbf{z}$ into $\mathbf{z}_1 \cup \mathbf{z}_2$ and then writing $p(\mathbf{z}|\mathbf{x}_i) = p(\mathbf{z}_1|\mathbf{x}_i)p(\mathbf{z}_2|\mathbf{z}_1, \mathbf{x}_i)$. In this fashion the exponential scaling of the softmax outputs with d can be mitigated.

The outputs from eq. 3 are combined as per eq. 2, and inference is carried out by seeking the mode of $\log p(\mathbf{z}|X)$ by using BFGS hill-climbing search from each Gaussian centre that has a mixing coefficient larger than a threshold. Instead of using a grid in Hough space as in eq. 2 it would be possible to use a more general mixture of experts framework where the $\boldsymbol{\mu}_j$'s and respective covariance matrices depend on the input; this would likely require fewer experts but would make the PHN more complex and difficult to train.

Our setup is as shown in Fig. 2. Panel (A) shows the camera with unknown parameters $\mathbf{z}$. Panel (B) illustrates how each of the detections is fed to a PHN in order to make predictions in the Hough space, represented as a mixture of Gaussians; these predictions are then combined (panel C) to give a final predictive distribution for $\mathbf{z}$.

Computation of the Joint Posterior: The density $p(\mathbf{z}|X)$ in eq. 2 is a product of several densities that include a Gaussian Mixture Model (GMM). Although a product of two GMMs is still a GMM, the resulting product cannot be computed directly (even for only a few observations) due to the exponential scaling of the number of components. In the PHNs framework the whole computation given the terms obtained from single PHNs is exact. We maintain the functions $p(\mathbf{z}|\mathbf{x}_i)$ and $p(\mathbf{z})$, and do not create the mixture with $\mathcal{O}(N^{dn})$ components explicitly. For each observation i we obtain $p(\mathbf{z}|\mathbf{x}_i)$ from a PHN and we store associated GMM coefficients. We can then evaluate the function at any point and obtain the gradient using Automatic-Differentiation (AD). The derivative is with respect to only d variables, so it is quick to compute by AD. The gradient may be also used for efficient sampling, such as Hamiltonian Monte Carlo [18].

Related work for Hough transforms: We have already described several works related to Hough transforms above. In addition, [6] used random forests to predict (in our notation) $p(\mathbf{z}|\mathbf{x}_i)$, and were able to obtain good results for problems of object detection, tracking and action detection. Their method makes predictions in Hough space for each $\mathbf{x}_i$ as a set of Gaussians at $\mathbf{x}_i$-dependent locations, and then uses a probabilistically incorrect method of summing the predictive densities (rather than their logs); as explained in [1, §3.5] this can be seen as an approximation due to robustification.

Camera parameterization: We use PHNs as per eq. 2 to find the most likely camera configuration. The camera model has both intrinsic and extrinsic parameters. The extrinsics are the translation and rotation of the camera; we assume that the objects lie on the (x, z) plane, and that the camera is at height $y = h$ above the origin. This is valid as we wish to estimate object poses *relative* to the camera. The camera rotation is as shown in Fig. 2A, with the camera looking at the ground plane at angle of elevation α. For the intrinsics, we assume that the camera coordinate frame is centred on the principal point, and the scaling factors m_1 and m_2 in pixels/m on the detector are known, so the parameter to be determined is the focal length f, or equivalently the angle of view (AoV) ω, which are related by $f = a_0/(2\tan\omega/2)$, where a_0 is the known sensor size. Thus $\mathbf{z} = (\alpha, h, \omega)$.

2.2. Scene Explanation

Detector: We use a sliding window approach to produce candidate detections which are sparsified using non-maximum suppression (NMS). The detector is trained to predict whether a particular object class is present at a given location. A positive patch is centered on the projection of the contact point of the object on the plane. The detector is also trained to predict the object projection scale (how large an object is in the image frame). It was crucial to train the detector so it does not activate for other object classes or noisy background in real images, therefore a half of the training dataset consists of random negative patches from real images. The training dataset also contains negative patches where an object is actually present but its contact point is above some distance from the centre. In this way we force the detector to have a local maximum where the object is centered. The output probability map of the detector is thresholded, and finally, NMS is carried out to produce a set of object detections.

Object and global predictor networks: These models predict the scene latent variables (LVs). The global variables are illumination parameters and base plane colour. The plane colour predictor takes the whole image as input. The object LVs are the shape (1-of-K), azimuthal rotation and colour (albedo). The object predictor networks are applied individually to each detected object patch.

We use *softmax* output for classification and *sigmoid* for regression as all our latent variables are bounded. For rotations we predict the sine and cosine of the angle. Illumination is predicted by making lighting predictions for each detected object, and then combining these by taking medians across the detections. This worked better than providing whole images as input; we believe that providing the detections allows cues from shadows and shading to be used more effectively.

Scene graph, back-projection and iterative refinement: The outputs of the above stages are assembled into a scene graph. As the scenes we study contain only certain types of object and lighting *etc*, the output of our analysis can be expressed in terms of a domain specific scene graph language.

The detected objects are back-projected into a 3D scene given the predicted camera to obtain the 3D positions of objects. Using the object appearances, the plane appearance and the illumination LVs we can then render the scene, and refine the fit iteratively. To do so we use a renderer[3] based on OpenDR: Differentiable Renderer [16], extended to simplify rendering multiple textured objects and to use modern OpenGL functionality (i.e. shaders) when rendering. We compute the match between the actual and rendered images using a robustified Gaussian likelihood model, as in [17, eq. 3]. The derivatives of the likelihood computed by OpenDR are fed to a nonlinear conjugate gradient optimizer[4].

3. Related Work for Scene Understanding

As discussed above, VIG is an old idea which can be reinvigorated using recent advances in deep learning in the analysis stages. Below we describe some recent work and how it relates to our paper. Perhaps the most closely related work to ours is that on Neural Scene De-rendering (NSD [24]). The authors consider scenes comprised of either 2D sprite type objects, or simplified 3D objects (from the Minecraft game) with 12 object types with fixed shapes and appearances. However, note that their background scene (green grass and blue sky) is fixed, as are the camera parameters and the lighting. Also their predictions are made only for the cartoon scenes, not real images.

Another related work is the Attend, Infer, Repeat (AIR) network of Eslami *et al* [5], which describes the scene in terms of a number of attentional fixations. One issue with the AIR network is that it uses a LSTM-based recurrent network to direct the fixations, but this means that a *sequential* framework is used to select an unordered *set* of object detections. Most of the AIR work is on 2D scenes, but the authors do provide a demonstration of the AIR network on a simple “tabletop” scene, where different object types and the background have fixed and unique colours, the width of input image is only 32 pixels, where the camera is always at fixed distance and elevation relative the ground plane. Recent work by Ren and Zemel [19] also makes use of a LSTM-based recurrent network to direct fixations, and additionally includes a segmentation network for determining segmentation masks of the detected objects. However, they are operating in a “2.1D” setting (layered occlusions of flat leaves), and so do not face the issues of inferring the camera or dealing with 3D geometry. Other related work in the VIG space includes Picture [14], the deep convolutional inverse graphics network (DC-IGN) [15], and [25, 17]. These methods incorporate “recognition network” type components for predicting latent variables given an image, but these papers only study inference for a single object rather than a scene containing multiple objects.

[3]https://github.com/polmorenoc/inversegraphics
[4]http://learning.eng.cam.ac.uk/carl/code/minimize/

4. Experimental setup

Stochastic Scene Generator: We create realistic synthetic scenes to train our recognition models. The training dataset consists of 35k objects in 7k images. For each image we sample the global parameters and a number of objects from the mug object class which lie on a table-top type plane. These are rendered using Blender at 256×256 resolution. Background images are taken from the NYU Depth V2 dataset [21]. We sample the camera AoV and then height and elevation uniformly in the appropriate ranges: $\omega \in [20°, 60°]$, $\alpha \in [0°, 90°]$, $h \in [0, 150]$ cm. Illumination is represented as uniform lighting plus a directional source, with the strength of the uniform light $\in [0, 1]$, the strength of the directional light $\in [0, 3]$, and the azimuth $\in [0°, 360°]$ and elevation $\in [0°, 90°]$ of the rotation of the directional light. Object colours are sampled uniformly in RGB space.

To sample a scene we first select a target number of objects (up to 7). We then sample the camera parameters and the plane colour. Objects are added sequentially to the scene, and a new object is accepted if at least a half of it is present in the image, it does not intersect other objects, and is not occluded by more than 50%. If is is not possible to place the target number of objects in the scene (e.g. when a camera is pointing downwards from a low height) we reject the scene. For each object we sample its shape (one of 15 mug shapes from ShapeNet[5]), size (diameter chosen randomly in $[8.0, 10.4]$cm), colour and rotation, and also a random texture in such a way that it creates a pattern but the colour is maintained. Some examples are given in Figure 5 and Figure 7.

Once the object detector network is trained, we obtain a derived dataset of the true positive detected patches, which incorporates translation errors made during the detection step. For each detection the closest object within a fixed range is assigned, along with the corresponding object LVs. This range is set to 24 pixels; the average size of an object's bounding box is 42×45 pixels, so this is about half the spatial size of the object. We then use this dataset for training the PHNs and predictor models.

PHNs: The size of the grid where GMM means are located that represent (α, h, ω)-space is $N_\alpha N_h N_\omega = 9 \times 15 \times 10$, a total of 1350 components. The space between components are $(\Delta_\alpha, \Delta_h, \Delta_\omega) = (10°, 10\text{cm}, 4°)$. The parameter γ of the robust model was set to 0.50 and the prior is uniform in the Hough space.

We represent the camera LVs as mixture of Gaussians in 3D space, which allows us to illustrate how PHNs can handle complex or multimodal distributions. In our case we decompose the PHN using the chain rule (see sec. 2.1) into two PHNs: H_1 predicts $p(\alpha, h|\mathbf{x}_i, \omega)$, as a GMM with means located in the grid of centres of size $N_\alpha N_h$, and H_2 predicts $p(\omega|\mathbf{x}_i)$ with a grid of size N_ω. The covariance matrix of the Gaussians for H_1 is $\mathbf{\Sigma}_1 = \beta^2 \text{diag}(\Delta_\alpha^2, \Delta_h^2)$, and $\mathbf{\Sigma}_2 = \beta^2 \Delta_\omega^2$ for H_2, with $\beta = 0.6$.

[5]https://www.shapenet.org/

Evaluation using Re-Projection Error: In addition to average log-likelihood, we evaluate our camera calibration using the re-projection error at the MAP prediction. This task is carried out by placing a known object (often a checkerboard) with a set of K 3D points in the scene at a known location, and comparing the actual locations of these points in the image to those predicted by the estimated projection matrix. Since we know the projection matrix of both ground-truth camera, $\mathbf{P}^{gt}$, and of the predicted camera, $\mathbf{P}$, we can place a checkerboard (virtually) in the scene, namely in the front of the view of a maximum size that fits the ground-truth image. Thus we know the exact positions of the grid-points in the world coordinates, $\mathbf{W}$. We also know the projection of the grid-points in the image frame using the ground-truth camera, $\mathbf{w}^{gt} = \mathbf{P}^{gt}\mathbf{W}$ and the ones obtained using the camera predicted by PHNs, $\mathbf{w} = \mathbf{P}\mathbf{W}$. The re-projection error is simply the RMSE of a deviation of the checkerboard grid-points in both images where $\mathbf{w}^{gt}$ and $\mathbf{W}$ are fixed, i.e.:

$$E(\mathbf{P}) = \sqrt{\frac{1}{K}\sum_{k=1}^{K} |\mathbf{w}_k - \mathbf{w}_k^{gt}|^2}. \quad (4)$$

Core details of the networks: All CNNs are based on the VGG-16 network [22] except for those that predict colour, which are a standard 3-layer CNN. Our main recognition networks are based on VGG-16 network and were optimised on a validation dataset. The networks use all 13 convolutional layers of VGG for 128×128 input, but without the last two max-pooling layers in order to be more spatially accurate, resulting in an output of size $512 \times 16 \times 16$. We then train three convolutional layers with 50 filters each of a size $512/50/50 \times 6 \times 6$. We found this configuration to work the best amongst different CNN architectures. This leads to a CNN output with a feature map with a single entry per feature map. Then we use this representation as an input to fully connected layers. Since the predictor and PHNs networks take as input the detections, we also concatenate this representation with (x, y, s) of a patch (see Figure 2B). We train all the layers on top of the VGG; this decreases significantly the number of trainable weights to approximately 1 million for each of the main networks (detector and PHNs networks, object shape predictor), and 0.4 million for the rest of VGG-based predictor networks. Networks are trained by SGD with Adam optimisation algorithm [12]. We found the *tanh* activation in all the layers on top of VGG to be superior to other activations for all the recognition models. The supplement presents more details.

Case	Evaluation metric	Baseline	CNN	Single-PHNs	Multi-PHNs	Multi better
2D	Log-likelihood	-9.51	-8.36	-7.65	$\mathbf{-6.18}$	**94%**
	Re-projection error	9.62	4.95	3.85	**2.41**	**91%**
3D	Log-likelihood	-13.20	-12.79	-11.33	$\mathbf{-10.18}$	**77%**
	Re-projection error	9.62	5.64	4.91	**3.30**	**86%**

Table 1. Results: average log-likelihood and re-projection error for Baseline, CNN, single-PHNs and Multi-PHNs. Re-projection error is given in % of the image width. The last column shows for each evaluation metric the percentage of images where PHNs predictions after observing multiple detections are better than the average of single PHN predictions.

5. Results

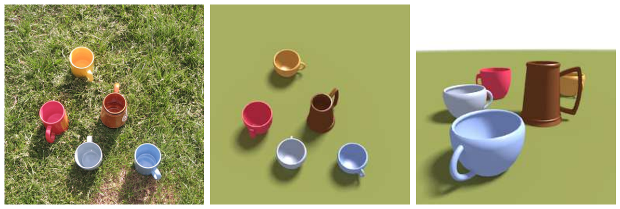


Figure 3. Input image, reconstructed 3D scene and a different view of it. Due to the interpretable representation, one could easily edit the scene, e.g. change object positions or their colors.

We perform a quantitative evaluation of all components on a synthetic test set of two hundred images containing 1k objects for which we know all latent variables. Note we evaluate camera pose, accuracy in object detection and each object latent variable separately, so each aspect of the scene reconstruction is assessed, as we are interested in the correct underlying scene interpretation. We first perform an in-depth evaluation of PHNs for two cases, when either the AoV is known ('2D case') or unknown ('3D case'). Next we evaluate the predictions of the global and object-specific latent variables. Finally, we evaluate the prediction qualitatively for real images, showing that all modules are able to transfer to real images. Figure 3 shows an example of an inferred scene representation.

Camera: A single PHN predicts $\mathbf{z}$ as a mixture of Gaussians: to make a point prediction we find the *maximum a posteriori* (MAP) value. The MAP $\mathbf{z}$-value obtained from combining multiple detections (Multi-PHNs) is found similarly (see sec. 2.1).

We evaluate the quality via the average predictive log-likelihood, and through the average re-projection error. Table 1 shows the results; for log likelihood a higher value is better, while for re-projection error lower is better. For the Single-PHNs column the result is averaged over all detections in a scene, as well as over scenes. As a simple baseline ('Baseline') we use the prior density $p(\mathbf{z})$, and the mean of $\mathbf{z}$ on the training set as a point estimate.

Standard approaches for camera pose estimation use either known objects (e.g. checkerboards) or exploit structure

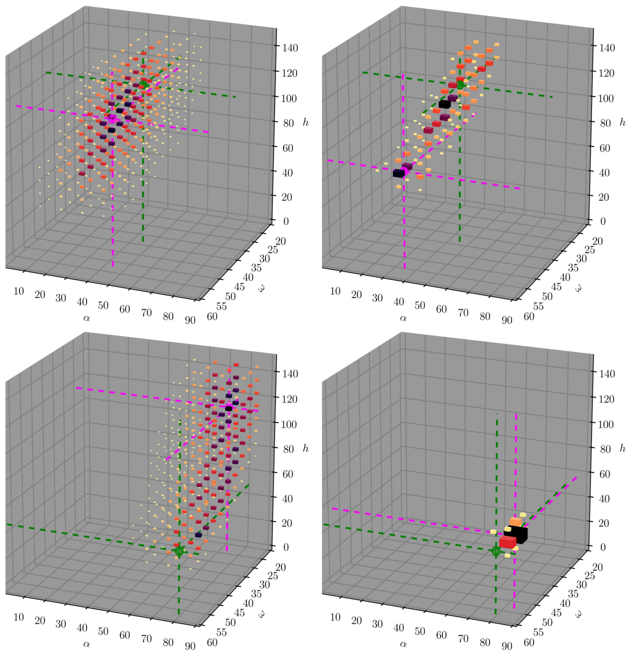


Figure 4. Examples of prediction for the 3D case (α, h, ω). The left plot in each pair shows a randomly selected example of a prediction for a single observation, the right plot shows the joint density with multiple observations. The ground-truth is denoted by a green ball located at the intersection of green guiding-lines. The magenta ball is the MAP. The density is represented as a 3D Hinton plot: the space is divided into voxels and each cube lies in the centre of a given voxel. The cube volume and color-coding represents the amount of the density mass within a single voxel.

like the vanishing points of lines in the scene, but these are not available in our scenes. To prove that object-based PHNs are superior to standard CNNs, as a non-trivial comparison we use a CNN predictor which takes the whole image as input and predicts $\mathbf{z}$. This is based on the VGG-16 architecture using the same configuration of all the hidden layers as the PHN network, where there are sigmoid outputs (scaled to match the Hough space size) for each camera LV. We use the predicted values to evaluate the Re-projection error. To evaluate the log-likelihood, for this CNN $p(\mathbf{z}|X)$ is modelled as a full covariance Gaussian in 2D/3D, robustified by including a term $(1-\gamma)p(\mathbf{z})$ (as in sec. 2.1) to avoid paying

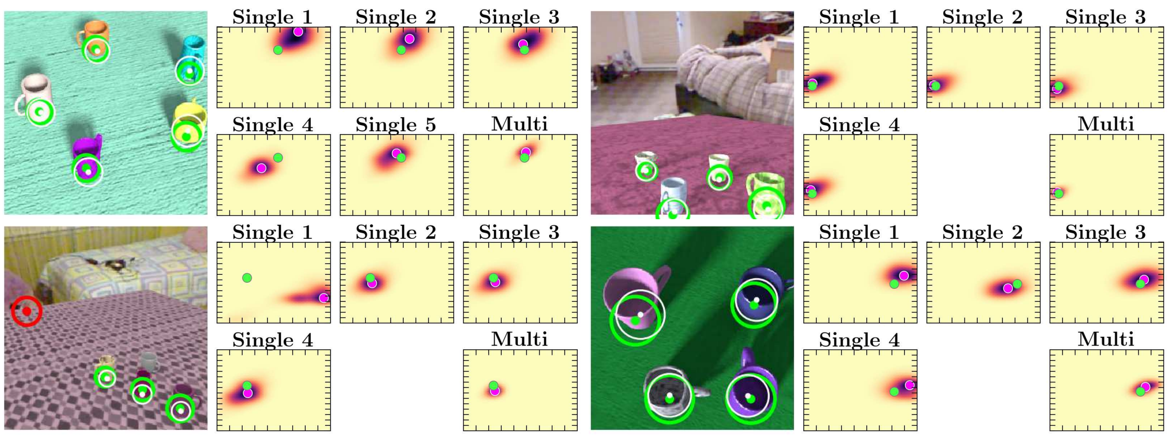


Figure 5. Four examples of PHNs prediction; the density sub-plots are in the (camera elevation, camera height)-space. Each density sub-plot is the prediction for a single observation, apart from the bottom-right sub-plot, which shows the joint density of the multiple observations. The ground-truth is denoted by a green circle, the MAP by a magenta circle. In the bottom-left example note that the outlying prediction 'Single 1' (due to a false positive detection of an object) does not corrupt the final result, due to the outlier model.

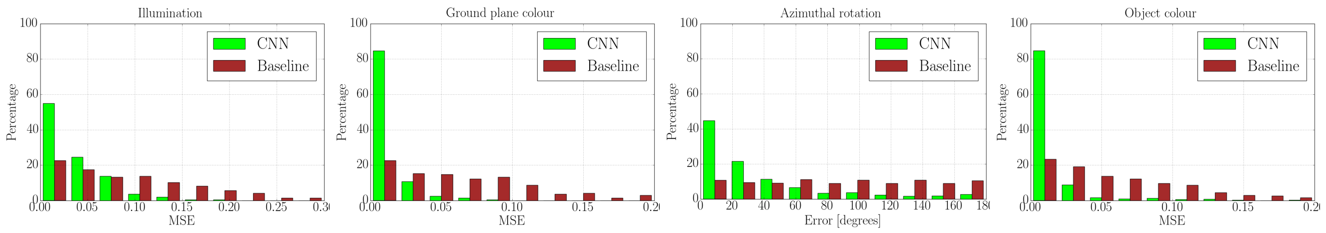


Figure 6. Histograms of the errors in the latent variables.

a high penalty for outliers.

For both the 2D and 3D cases PHNs clearly make better predictions after observing multiple detections. Even a Single-PHN that processes a single detection outperforms the CNN method that takes as input the whole image. The re-projection error (3D) is 4.91 (Single-PHN) vs 5.64 (CNN), and is significantly lower given all the detections (3.30).

Figure 5 shows four example scenes, single object predictions in the 2D Hough space, and the Multi-PHNs prediction (bottom right in each panel). Notice that in the Multi-PHNs plots the uncertainty has significantly decreased compared to single observation plots, so the model works as desired. The same happens for the 3D case as density plots in Figure 4 show. Even in the cases of lower likelihood, the MAP is very close to the ground-truth, which is confirmed by the Multi-PHNs re-projection error that is on average a third lower than Single-PHNs.

Detector: Our detector has 98% precision at 93% recall, using a radius of 24 pixels for detections (as described above). The inclusion of real image patches in the training dataset significantly decreased false positive detections in the background by around one order of magnitude. Our detector is very precise, giving an average translation error of the objects of only 1.2% of the image width in x and 1.5% in y direction, this is several times less than the usual size of a mug (see Figures 1A and 5).

Global LVs: The error metric of the ground plane colour (albedo) is the mean square error (MSE) of colour (a, b) components in the Lab space.[6] The baseline is the mean intensity of each colour channel in the training set. The lighting is projected onto a sphere, scaled so that the difference between maximal and no illumination is unity, and errors are computed by MSE. For illumination we use a baseline that minimizes the error containing both uniform illumination and directional illumination from the top, at optimal strengths.

Object LVs: For azimuthal rotation we measure the absolute angular difference between the prediction and ground truth, but with wrap-around, so the maximum error is $180°$. The baseline is a fixed rotation angle chosen to mimimize the error. We evaluate the object colour in the same way as for global LVs above. For object shape prediction we make a 1-of-K classification ($K = 15$).

Results are given for the global LVs and object LVs in Table 2 and Table 3. Figure 6 shows histograms of the errors

[6]https://en.wikipedia.org/wiki/Lab_color_space

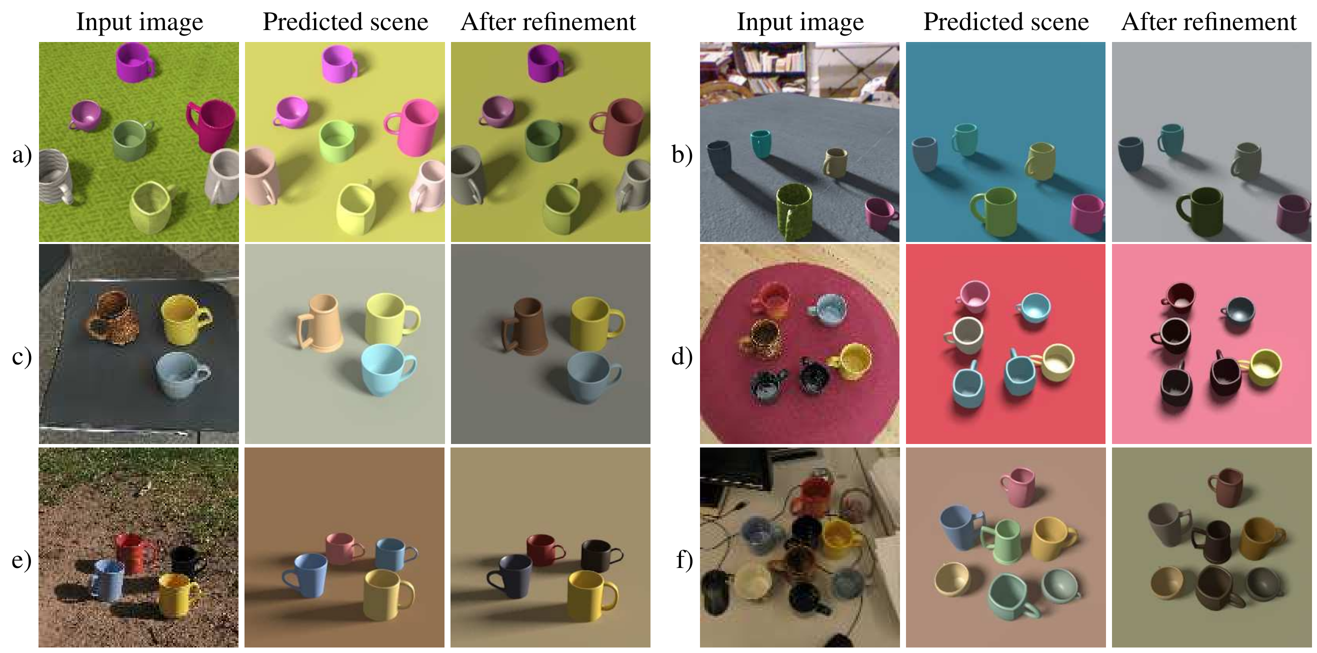


Figure 7. Results on synthetic (top row) and real scenes (middle, bottom). For each example the input image, predicted 3D scene, and result after iterative refinement are shown (left to right).

Global LVs	Baseline	CNN
Illumination	0.084	0.025
Colour	0.054	0.005

Table 2. Global latent variables: median errors.

Object LVs	Baseline	CNN
Azimuthal rotation	91°	22°
Colour	0.049	0.005

Table 3. Object latent variables: median errors.

in the latent variables for both the baseline (red) and CNN (green predictors). For illumination and object azimuth the median results are more than 3 times better than the baseline, and for the plane and object colours almost $10\times$ better. For object shape classification the accuracy is 31.6%, compared to 6.7% of a random choice. This is a good result as often it is difficult to distinguish particular shapes, e.g. when a mug is viewed from the top, or is far away.

Scene understanding – qualitative results: In Fig. 7 we show results on both synthetic (top row) and real scenes (middle and bottom rows). We note that our methods work well on real images, despite not having been trained on them. The mugs are generally predicted well in location, azimuth and colour, and the camera parameters and lighting are in good agreement with the input image. The iterative refinement (rightmost panels of each example) mainly improves the colours of the objects. In (e) the directional light source is detected properly. In the cluttered scene (f) all mugs are detected properly except one. More examples are in the supplement.

6. Discussion

Above we have shown how to successfully put all of the components together to create an interpretable scene-graph representation of a 3D scene from a single image, by accurately predicting the camera parameters using PHNs, and global and object-based latent variables using CNNs. Our results show the advantages of the PHNs formulation using object detections over a deep VGG-16 based CNN that takes the whole image as input. The framework has been shown to work on both real and synthetic images.

There are a wide variety of extensions that we are currently exploring, including the use of more object classes, and richer models of shape and appearance for each object class. In this case PHNs can easily be extended to take the predicted class variable as input.

Acknowledgements

LR was supported by Microsoft Research through its PhD Scholarship Programme. The work of CW was supported in part by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

References

[1] M. Allan and C. K. I. Williams. Object Localization using the Generative Template of Features. *Computer Vision and Image Understanding*, 113:824–838, 2009. 2, 3

[2] E. Angel. *Interactive Computer Graphics*. Addison Wesley, third edition, 2003. 1

[3] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981. 2

[4] O. Barinova, V. Lempitsky, and P. Kohli. On detection of multiple object instances using Hough transforms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 34(9):1773–1784, 2012. 2

[5] S. M. A. Esalmi, N. Heess, T. Weber, Y. Tassa, K. Kavukcuoglu, and G. E. Hinton. Attend, Infer, Repeat: Fast Scene Understanding with Generative Models. In *Advances in Neural Information Processing Systems 29*, pages 3225–3233, 2016. 4

[6] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Trans. PAMI*, 33(11):2188–2202, 2011. 3

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, pages 2672–2680. 2014. 1

[8] U. Grenander. *Lectures in Pattern Theory: Vol. 1 Pattern Synthesis*. Springer-Verlag, 1976. 1

[9] U. Grenander. *Lectures in Pattern Theory: Vol. 2 Pattern Analysis*. Springer-Verlag, 1978. 1

[10] P. V. C. Hough. Method and means for recognizing complex patterns. *U.S. Patent 3069654*, 1962. 2

[11] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, 3:79–87, 1991. 3

[12] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *CoRR*, 2014. 5

[13] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014. 1

[14] T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. Mansinghka. Picture: A probabilistic programming language for scene perception. In *Proc CVPR*, pages 4390–4399, 2015. 4

[15] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. B. Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems 28*, 2015. 4

[16] M. M. Loper and M. J. Black. OpenDR: An Approximate Differentiable Renderer. In *Computer Vision–ECCV 2014*, pages 154–169. Springer, 2014. 4

[17] P. Moreno, C. K. I. Williams, C. Nash, and P. Kohli. Overcoming Occlusion with Inverse Graphics. In *Computer Vision-ECCV 2016 Workshops Proceedings Part III*, pages 170–185. Springer, 2016. LNCS 9915. 4

[18] R. M. Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 54:113–162, 2010. 3

[19] M. Ren and R. S. Zemel. End-to-End Instance Segmentation and Counting with Recurrent Attention. *arXiv preprint arXiv:1605.09410*, 2016. 4

[20] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *ICML*, 2014. 1

[21] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *ECCV*, 2012. 5

[22] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 5

[23] R. S. Stephens. A probabilistic approach to the Hough Transform. In *Proceedings of the British Machine Vision Conference, (BMVC)*, pages 1–6, 1990. 2

[24] J. Wu, J. B. Tenenbaum, and P. Kohli. Neural Scene Derendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 4

[25] I. Yildirim, T. D. Kulkarni, W. A. Freiwald, and J. B. Tenenbaum. Efficient analysis-by-synthesis in vision: A computational framework, behavioral tests, and comparison with neural representations. In *Thirty-Seventh Annual Conference of the Cognitive Science Society*, 2015. 4