

3D + Graphics

QI ZHU & JUHO KIM

Outline

- Pose estimation (3D recovery from 2D images)
- Novel Image / View synthesis
- Reconstruction and generation of 3D

Part 1

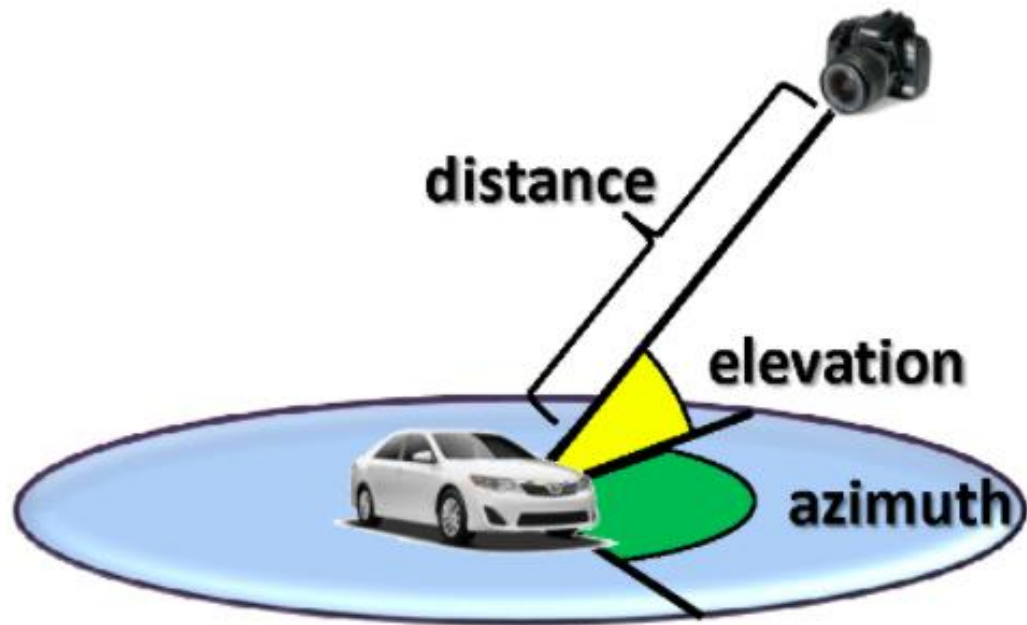
POSE ESTIMATION (3D RECOVERY FROM 2D IMAGES)

Viewpoint Estimation



INPUT: RGB image

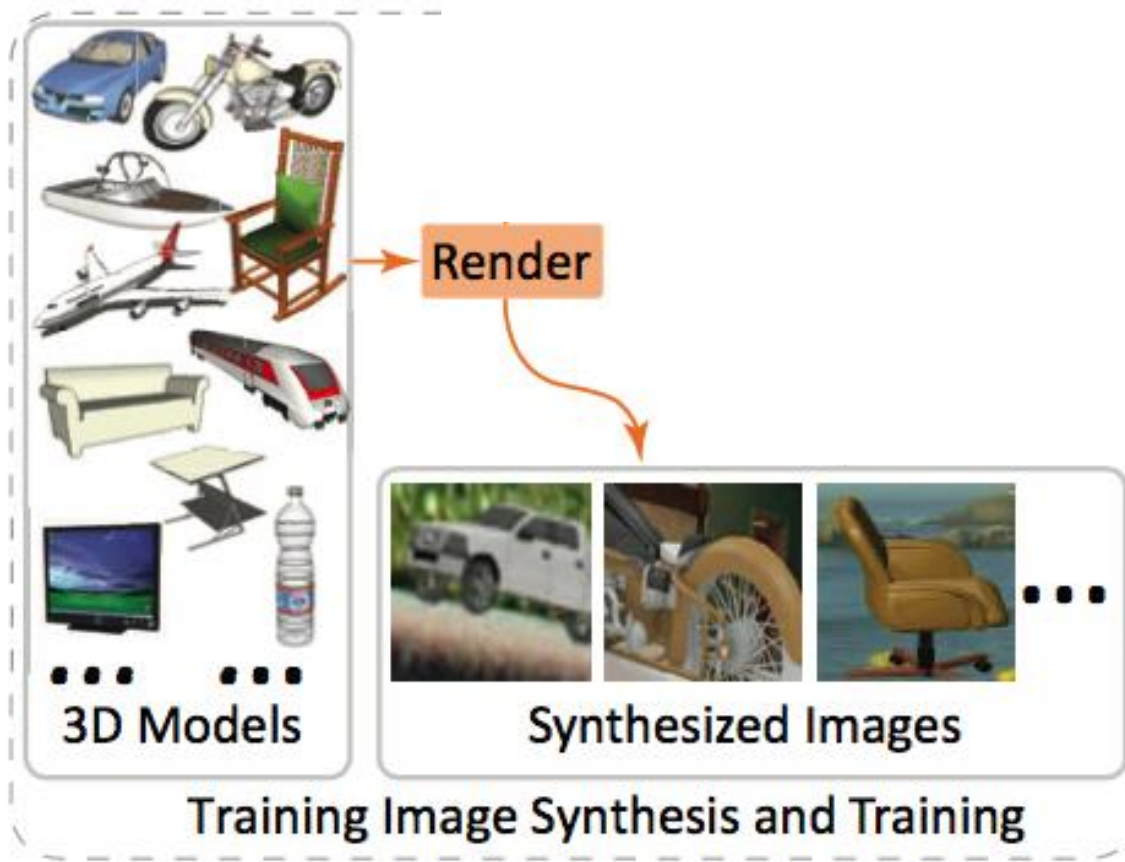
OUTPUT: Camera pose = Rotation (yaw, pitch, roll) and Translation Matrix



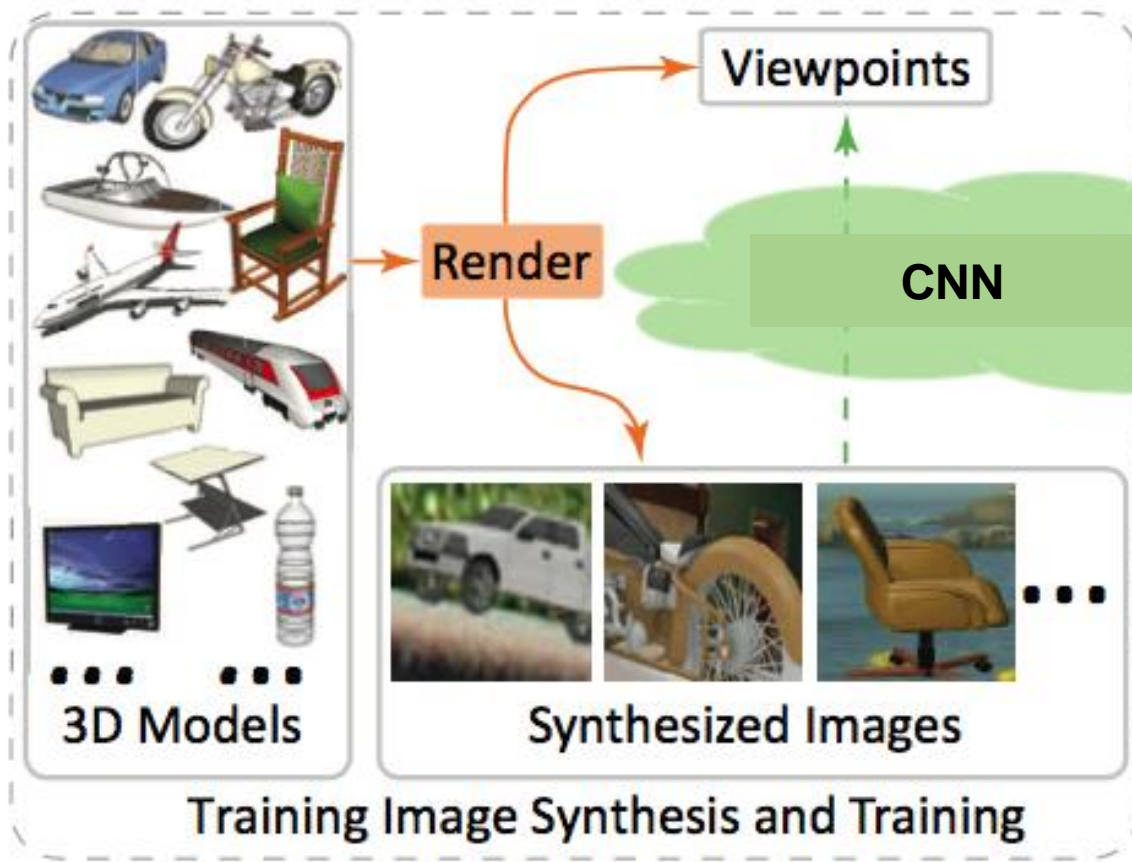
insufficient data!

Render for CNN [ICCV15]

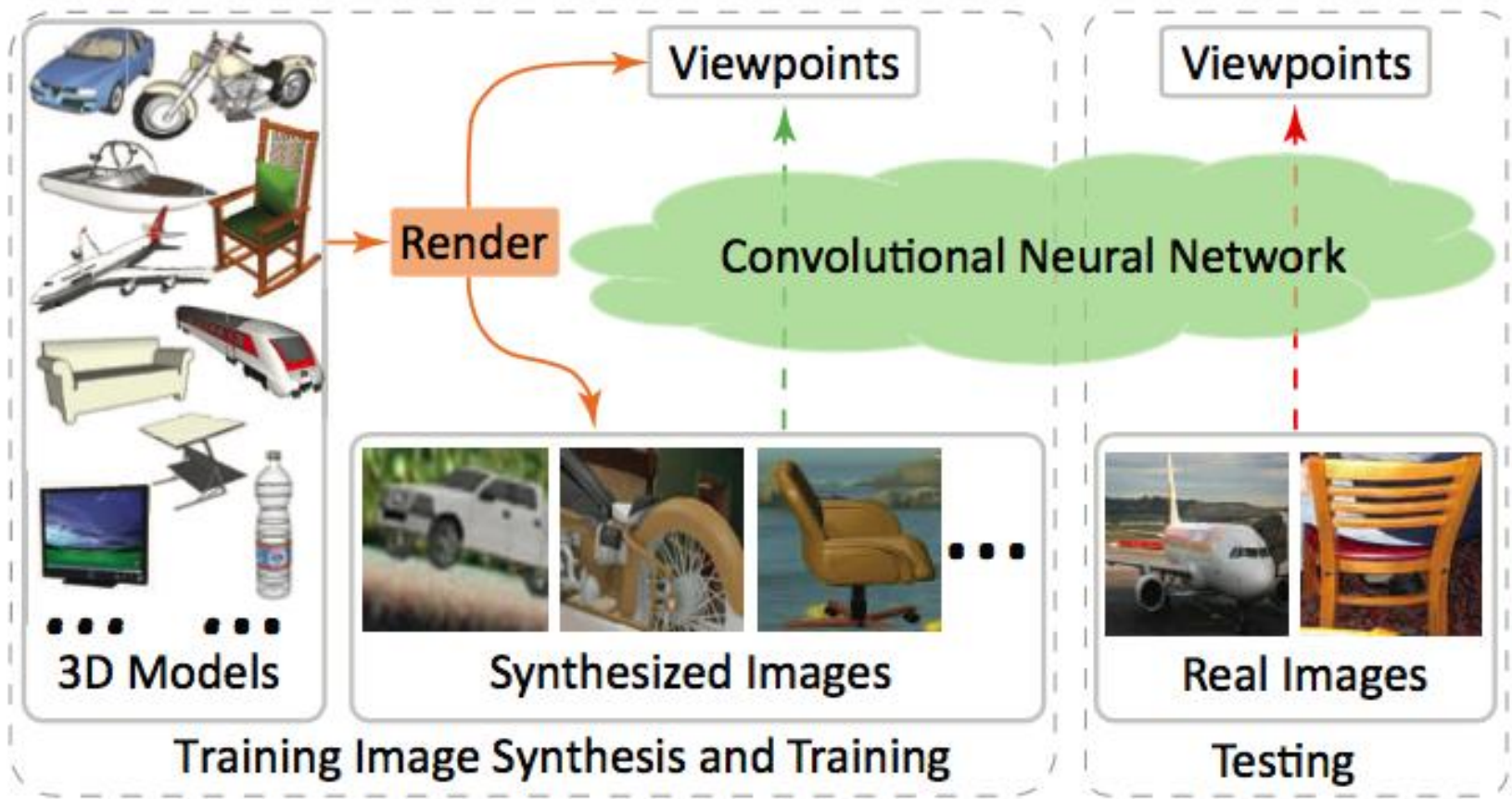
Render for CNN [ICCV15]



Render for CNN [ICCV15]



Render for CNN [ICCV15]



Generating synthesized images



Cont'd

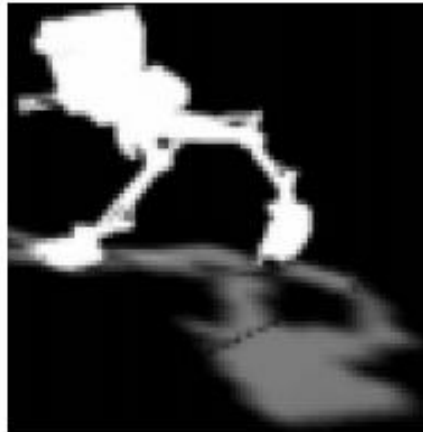
Over

- Vα
- V
- Bα
- α
- C
- A

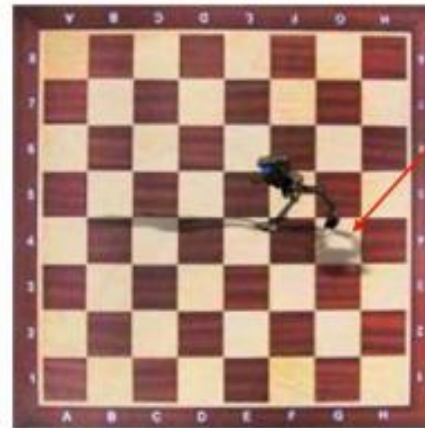


$$I_{\text{comp}} = \alpha I_{\text{fg}} + (1-\alpha)I_{\text{bg}}$$

alpha
mask



3D M



shadow



ameters

Final Training Dataset



Problem formulation

Input: single RGB image

Viewpoint as a tuple (θ, ϕ, ψ) of camera rotation parameters

- Discretized and divided into 360, 180, 360 bins

Rotation reference: predefined initial pose face camera

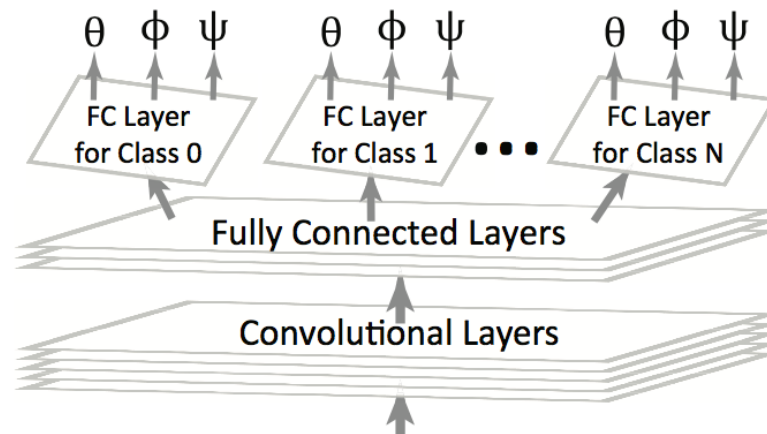
Output: probabilities of each viewpoint

Loss function:

$$L_{vp}(\{s\}) = - \sum_{\{s\}} \sum_{v \in \mathcal{V}} e^{-d(v, v_s)/\sigma} \log P_v(s; c_s),$$

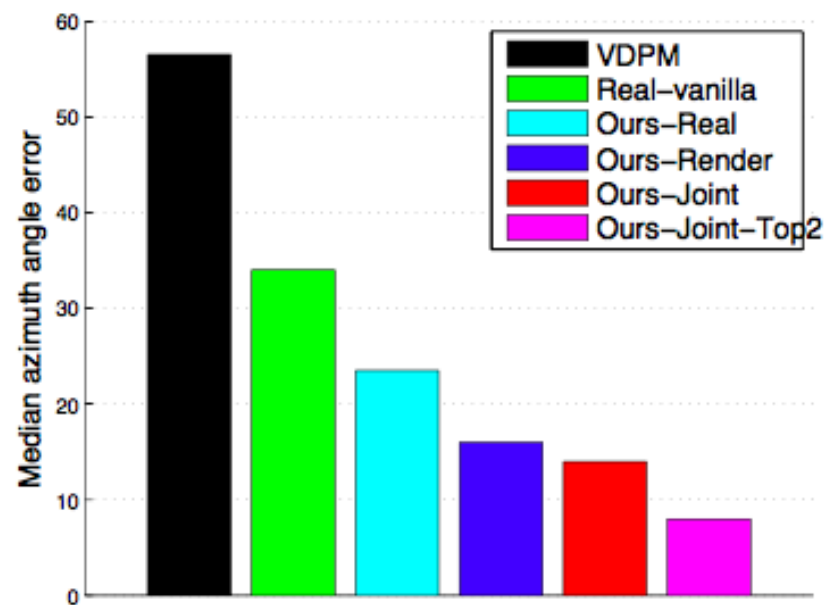
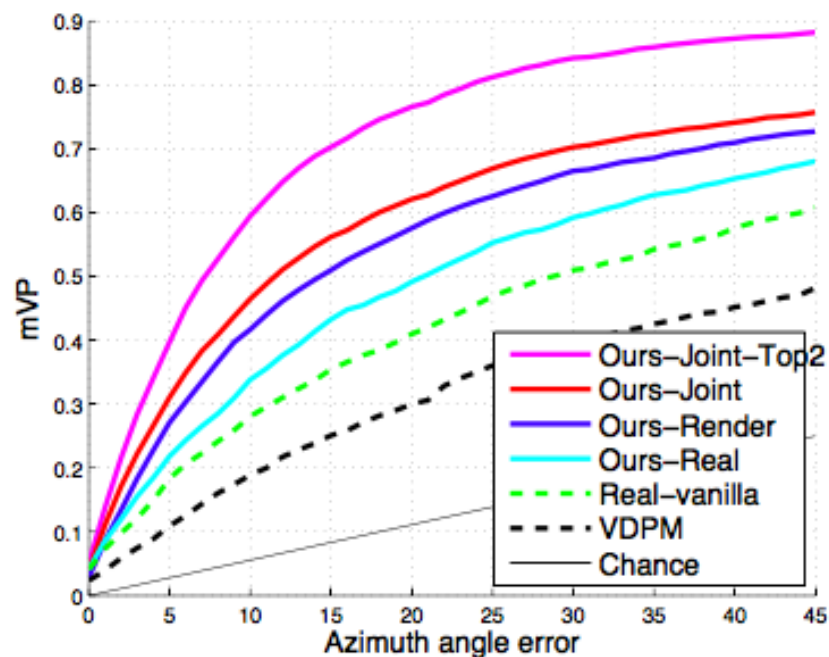
where $P_v(s; c_s)$ is the probability of view v for sample s from the soft-max viewpoint classifier of class c_s

Class-Dependent Network Architecture

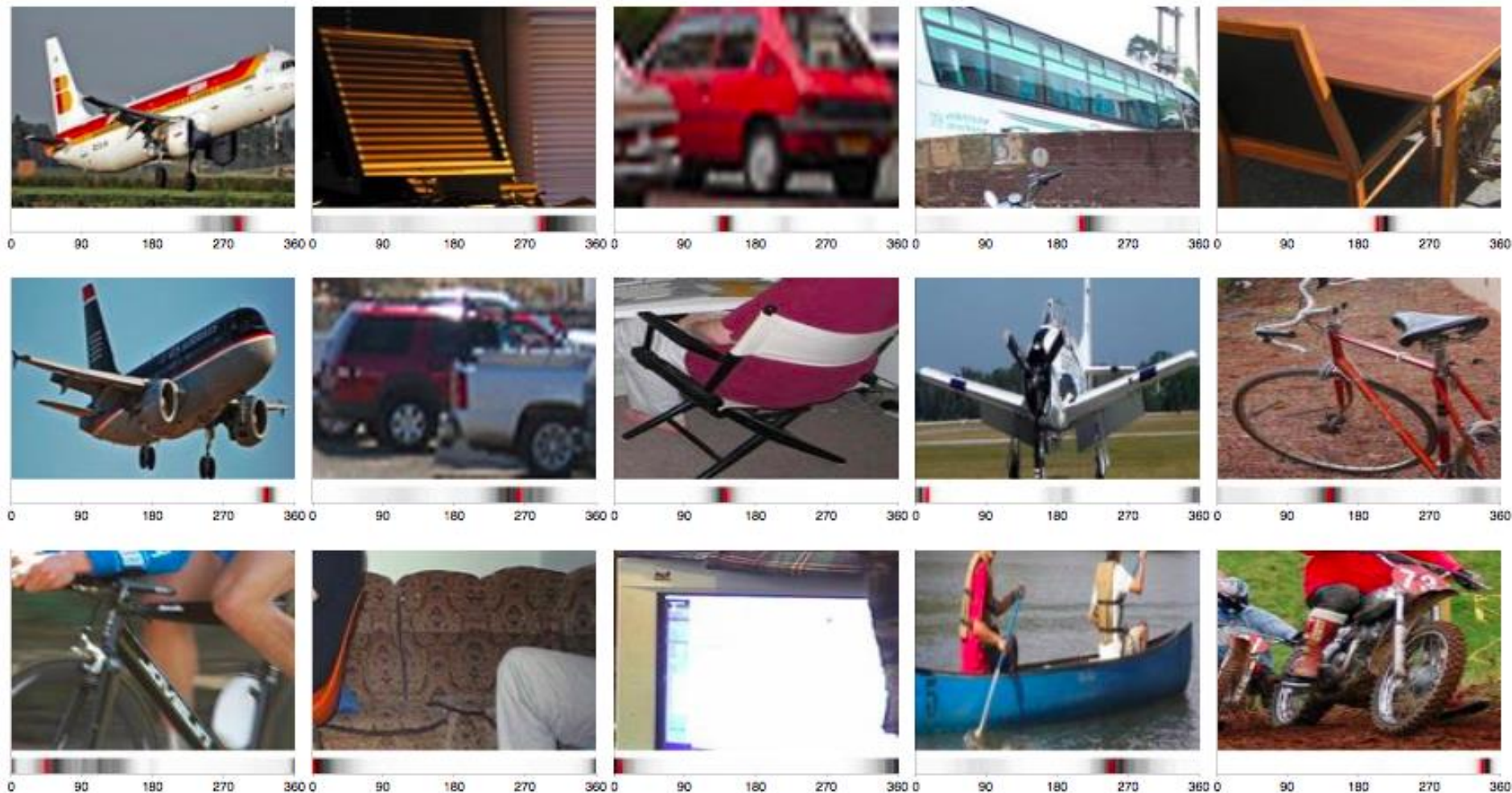


- Based on Alex Net [NIPS12]
- Shared weights but different class FC Layer
- Large number of outputs! $(380+180+360) \times N$
- Claim is that different output layers handle the large variance among different object categories

Does synthetic training data help?



Several results



Keypoint Estimation: 2D to 3D

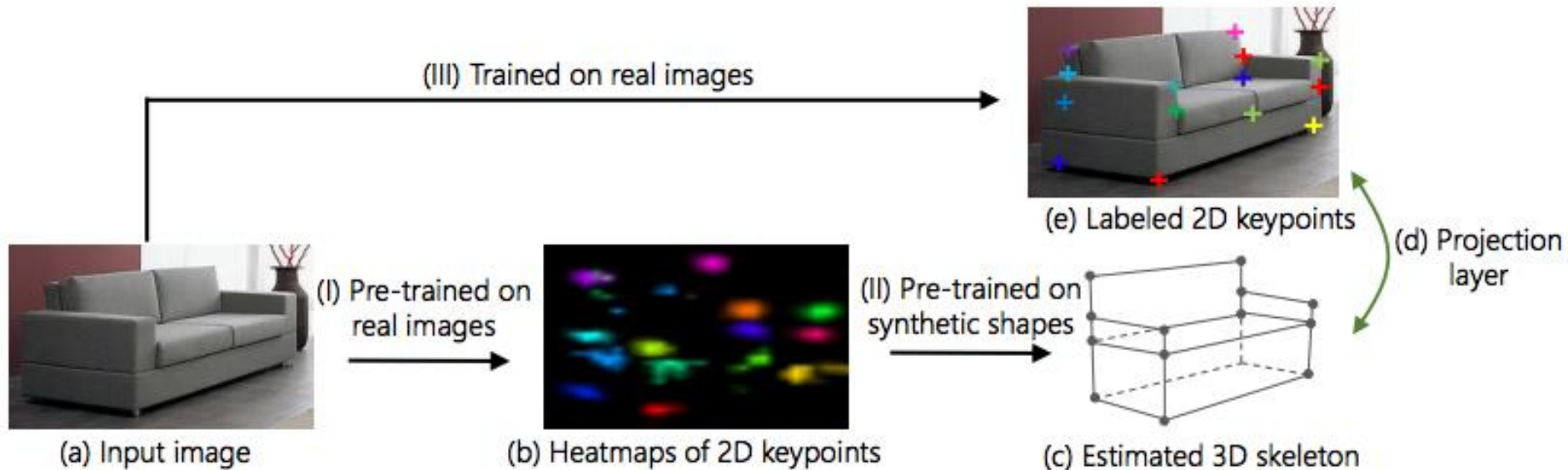


Easy



Hard!

Single Image 3D Interpreter Network [ECCV16]



Simultaneously infer 2D keypoints heatmap and 3D structures from single image!

Challenges

Annotations?

- 2D keypoints labels – easy to get, e.g. crowdsourcing
- 3D object annotations in real 2D images – hard to acquire

Synthetic training data?

- Statistics of real and synthesized images is different



Using both real and synthetic image

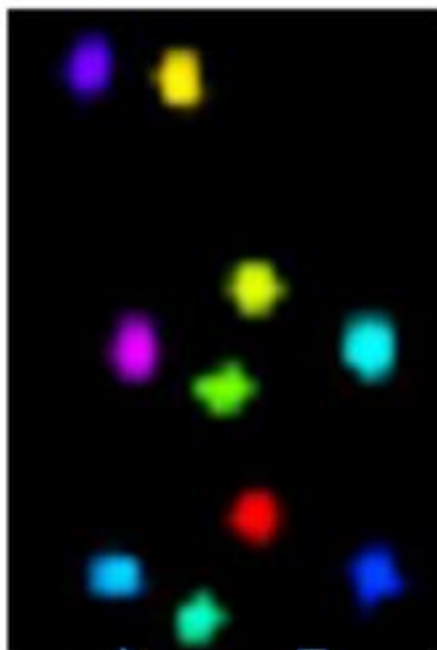


Real images with
2D keypoint labels

Using both real and synthetic image

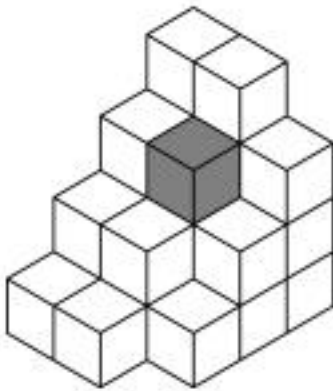


Real images with
2D keypoint labels



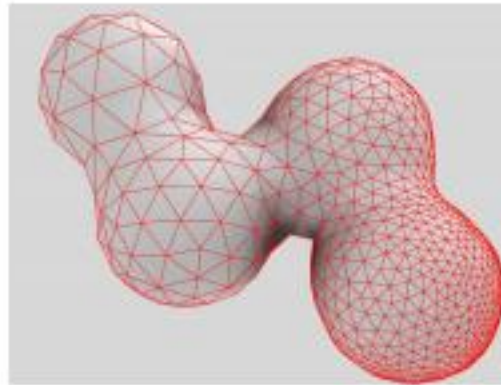
Synthetic
3D models

3D Object Representation



Voxel

Girdhar et al. '16
Choy et al. '16
Xiao et al. '12



Mesh

Goesele et al. '10
Furukawa and Ponce, '07
Lensch et al. '03



Skeleton

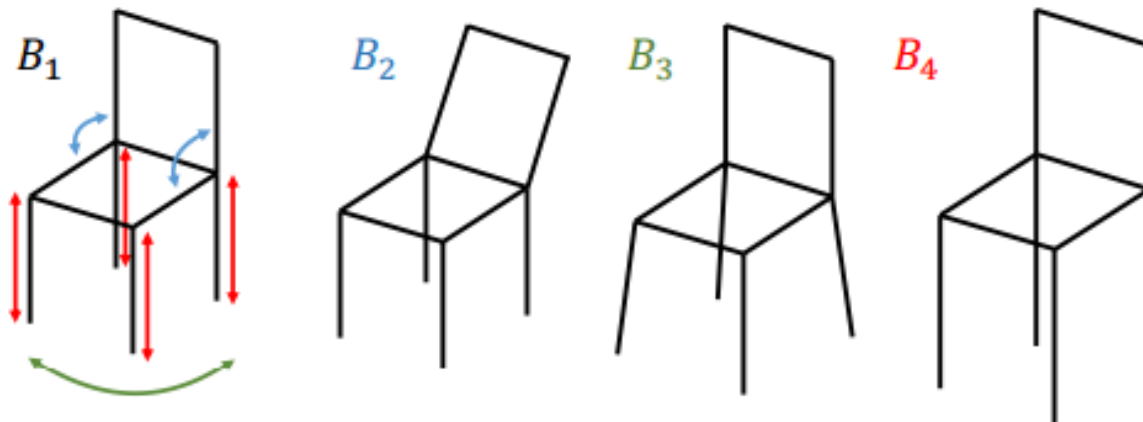
Zhou et al. '16
Biederman et al. '93
Fan et al. '89

3D-Skeleton Representation

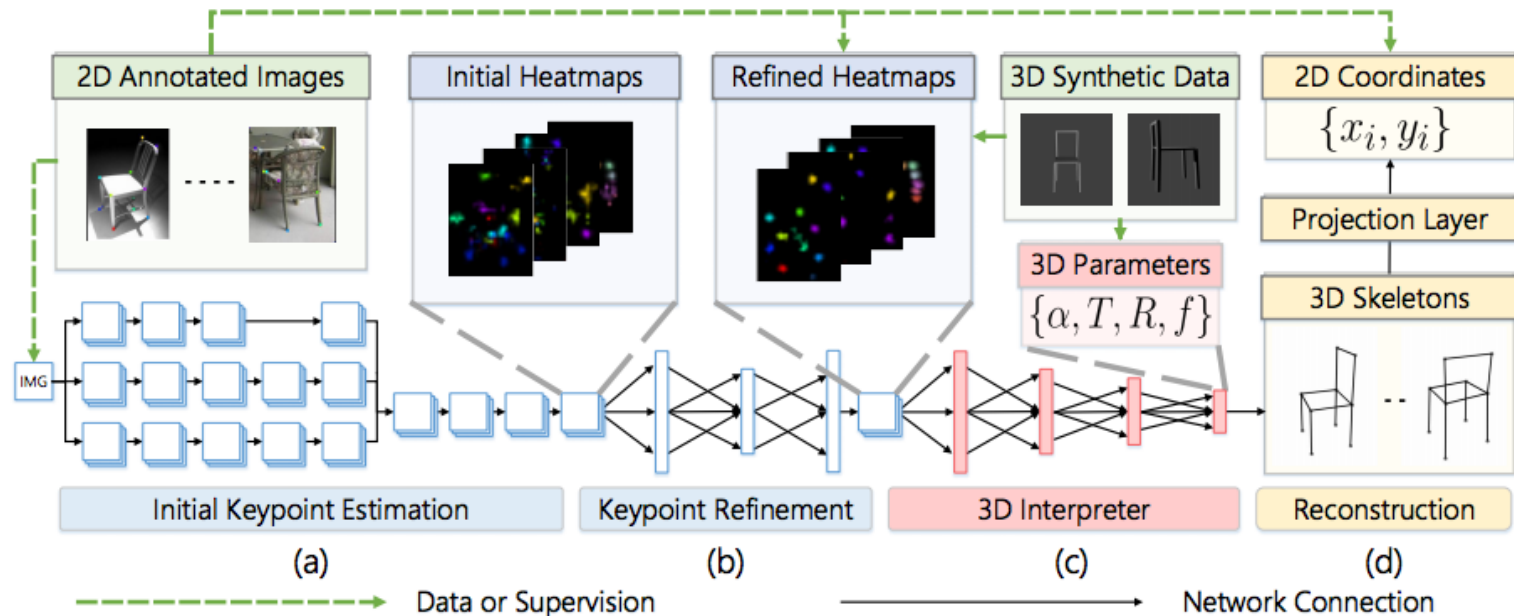
Assumption:

- objects can only have constrained deformations
- The first base shape is the mean shape of all objects within the category
- 3D keypoint locations are a weighted sum of base shapes

$$\mathcal{X} = P(R\mathcal{Y} + T) = P\left(R \sum_{k=1}^K \alpha_k B_k + T\right).$$



Network Overview

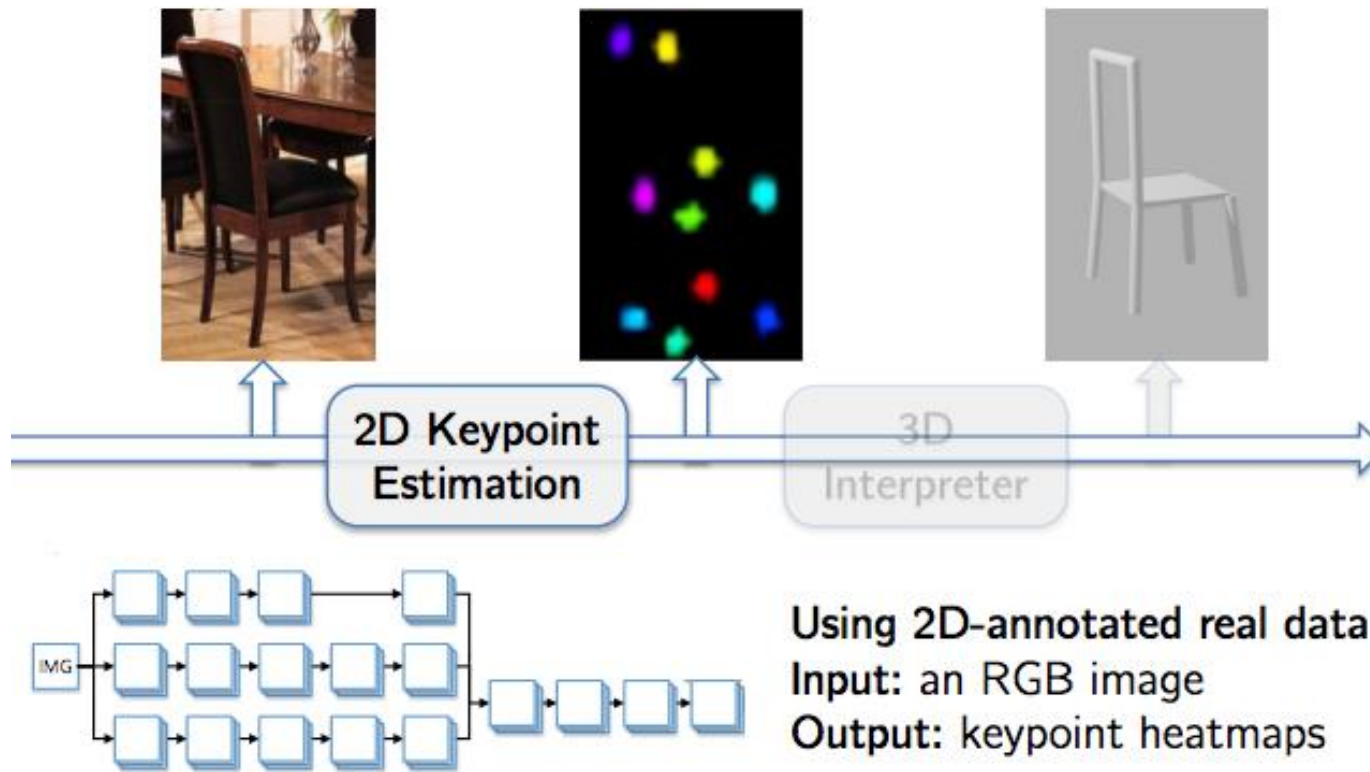


Step 1: Estimates 2D keypoint heatmaps (a and b)

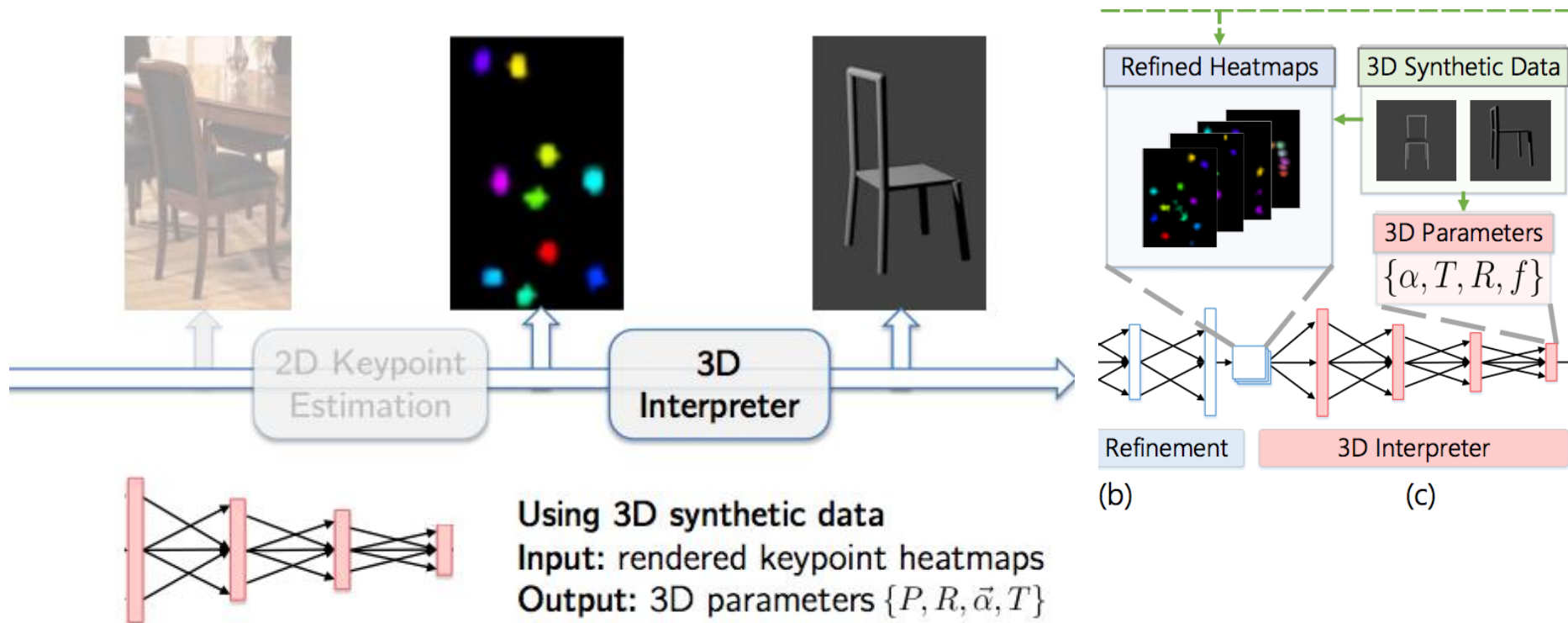
Step 2: Train 3D interpreter on 3D synthetic data (c)

Step 3: Jointly train projection layer from 2D annotations (d)

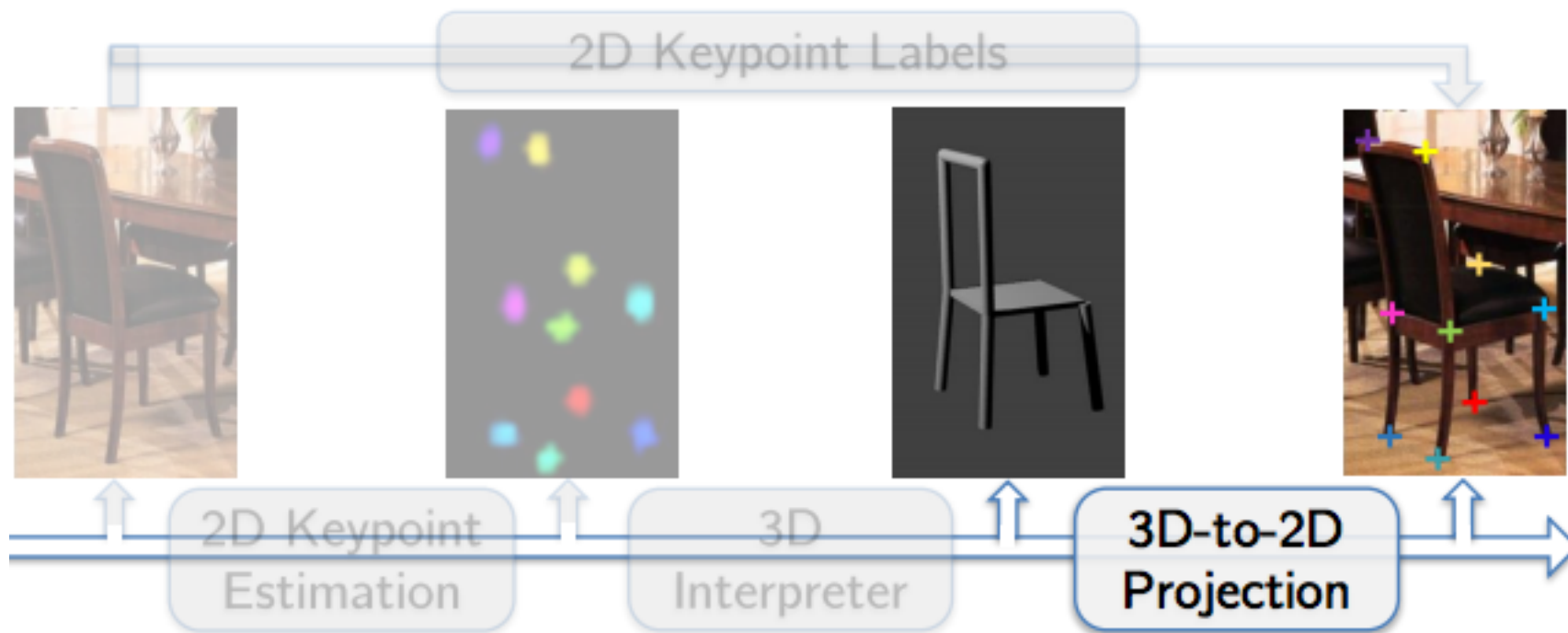
Zoom in: Keypoint Estimator



Zoom in: 3D Interpreter



Zoom In: End to End



Results

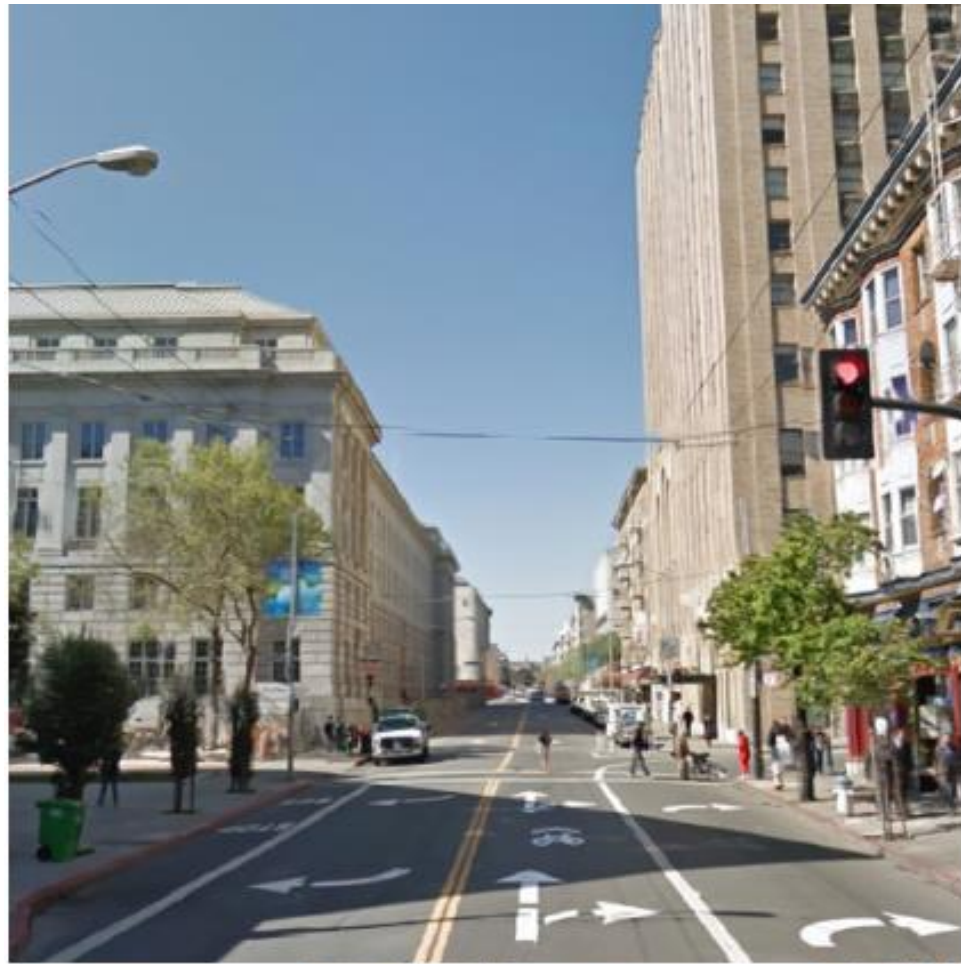


Part 2

NOVEL IMAGE / VIEW SYNTHESIS

New scene synthesis?

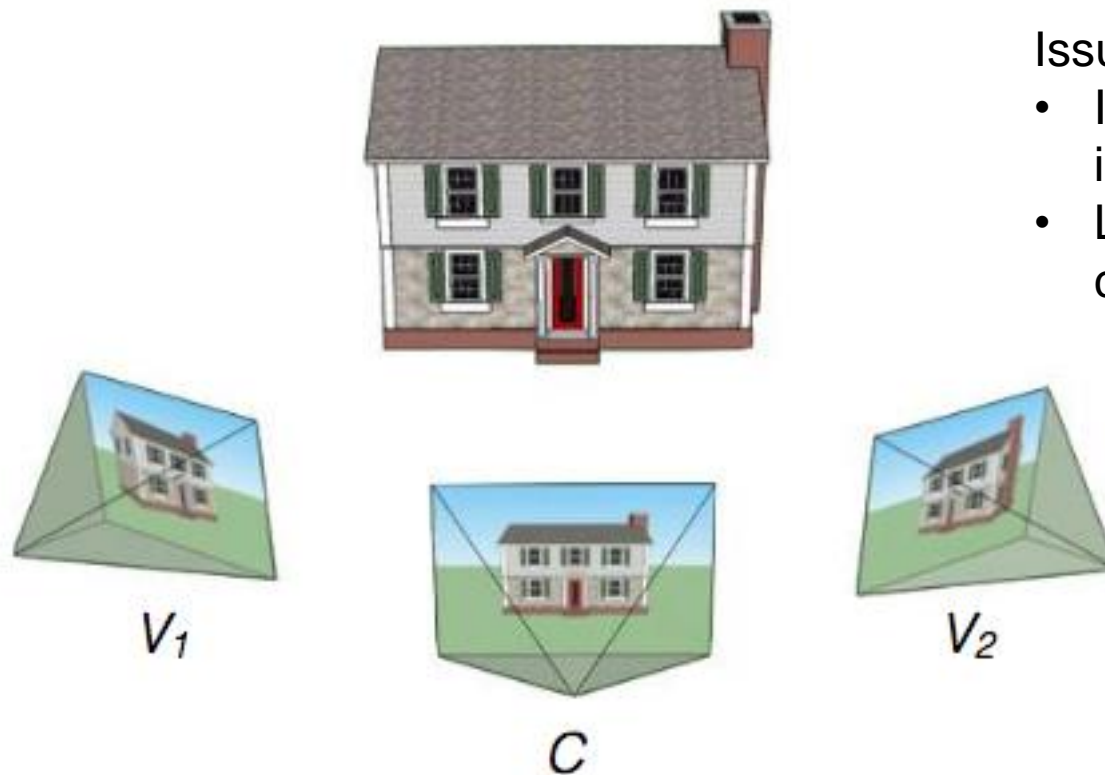
Newly Generated Image
Note the change in viewpoint



Given Images



DeepStereo: Learning to Predict New Views from the World's Imagery

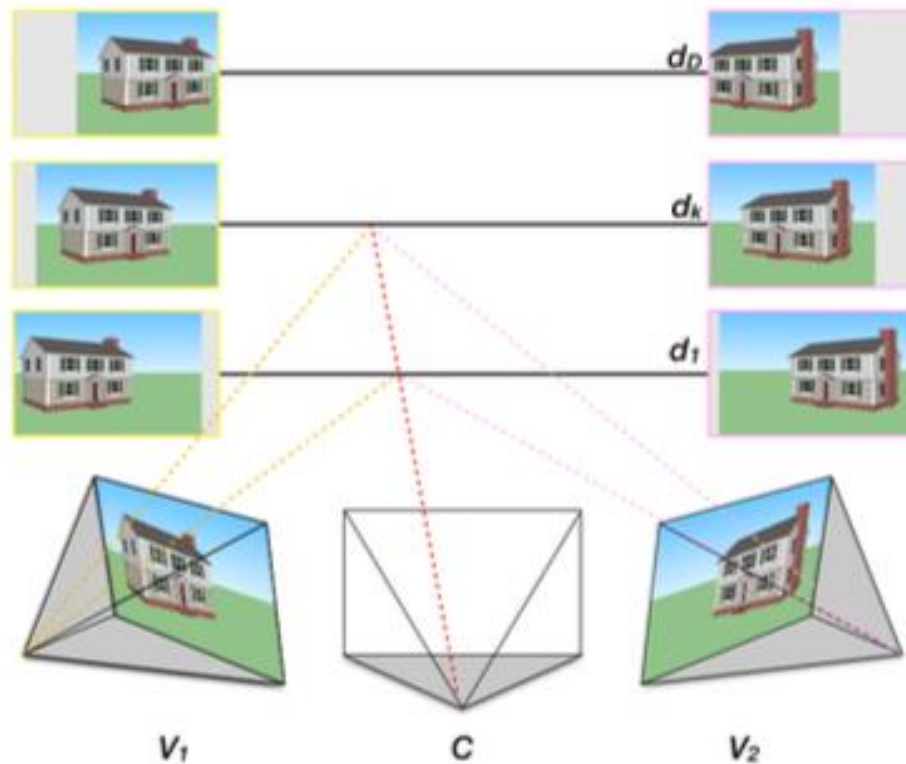


Issues:

- Interpret rotation and image reprojection
- Long-distance pixel correlation

Given V_1 , V_2 , generate C

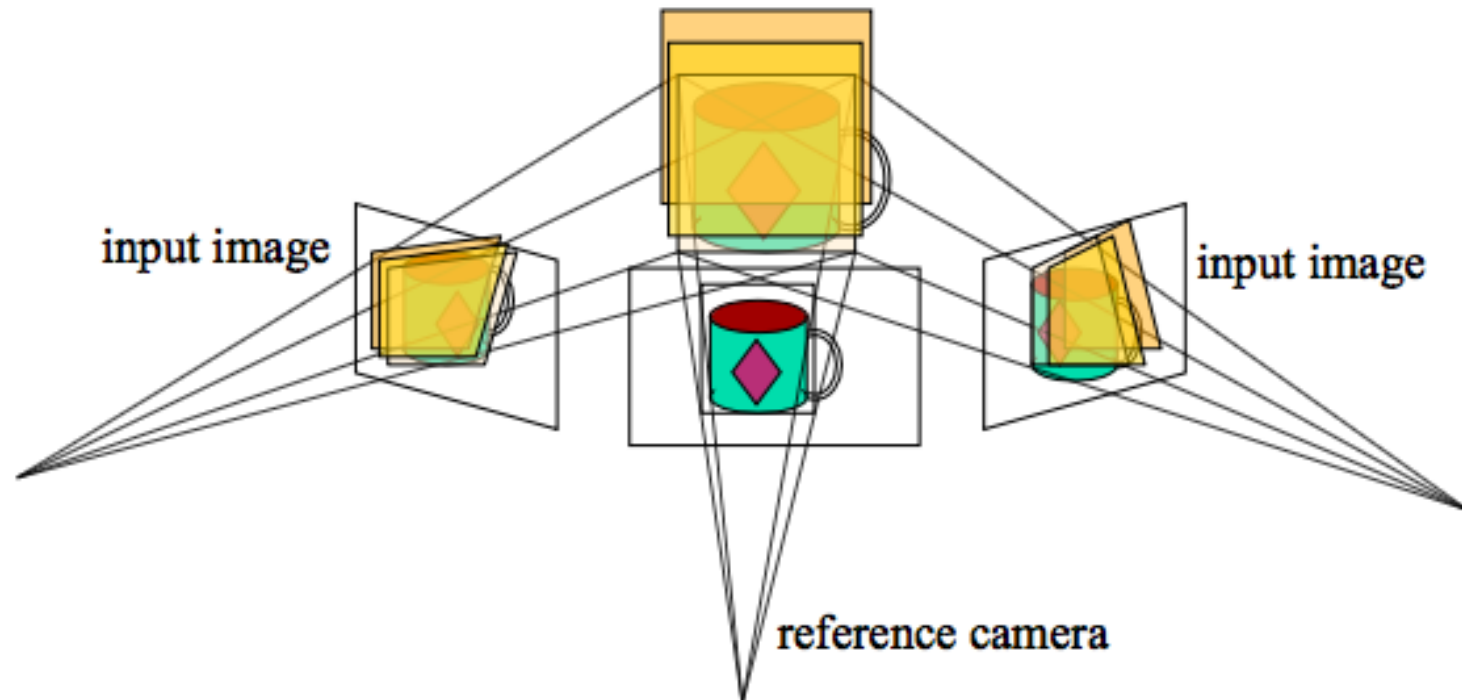
DeepStereo: plane-sweep volumes



Solve two other problems before generating a new view!

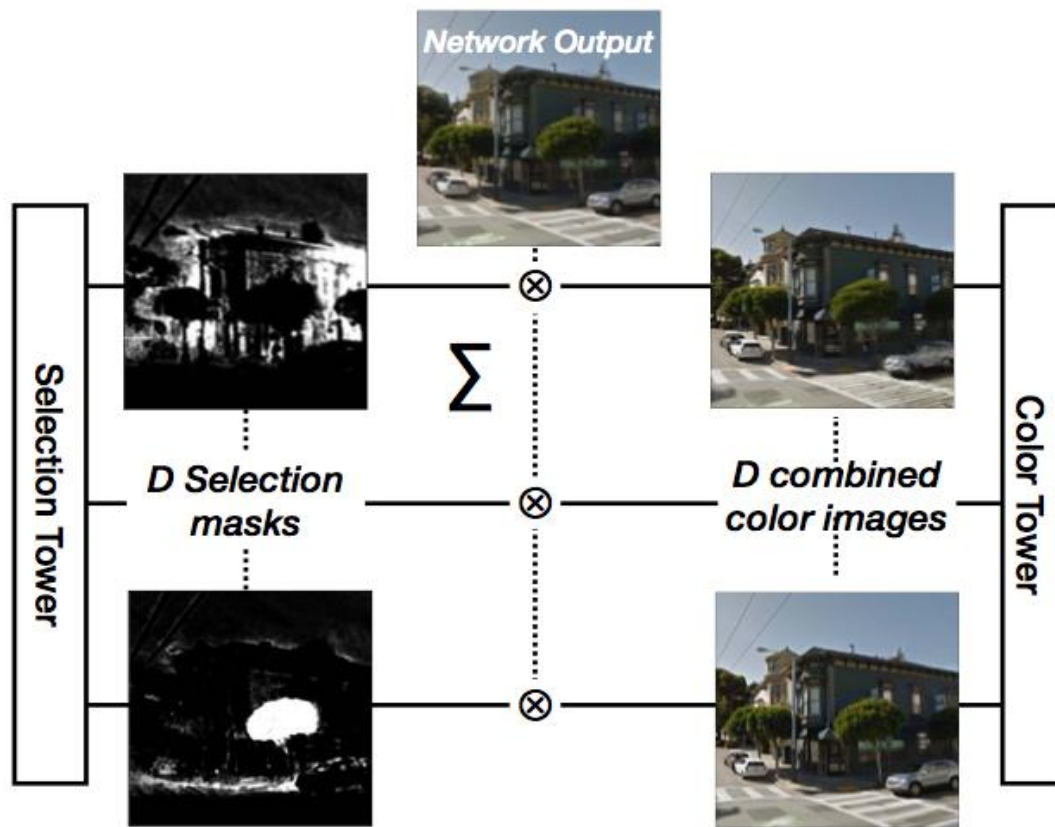
But why?

Plane-sweep volumes (Cont'd)



- Sweep family of planes at different depths w.r.t. a reference camera
- For each depth, project each input image onto that plane
- This is equivalent to a homography warping each input image into the reference view

DeepStereo: overview



$\{\mathbf{S}\}$ Depth Selection tower output

$\{\mathbf{C}\}$ Color tower output

$$c_{i,j}^f = \sum s_{i,j,z} c_{i,j,z}$$

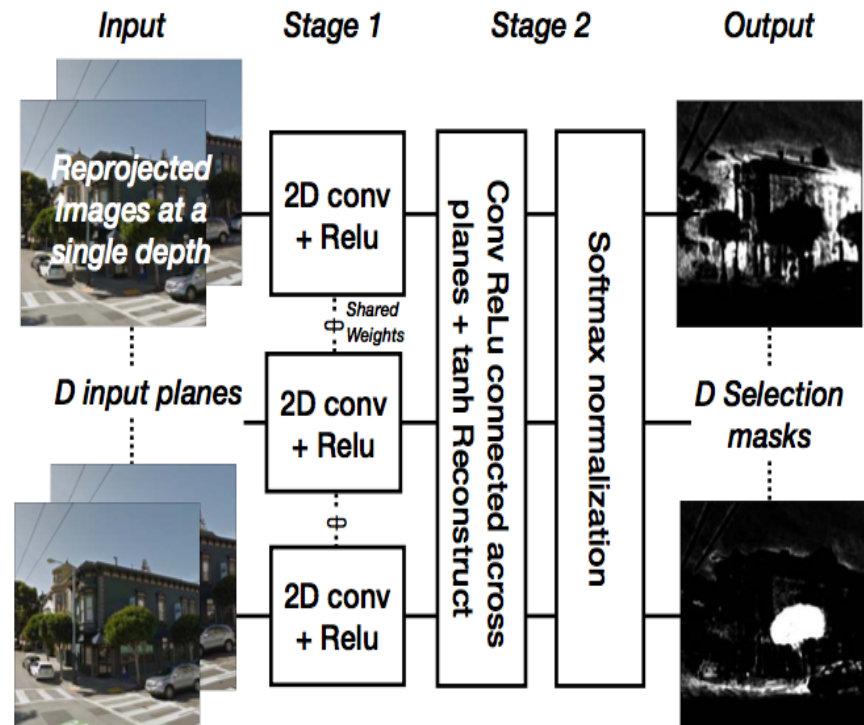
Output is a masked sum over images selected at different depths

DeepStereo: Selection Tower

INPUT: Plane-sweep volumes

OUTPUT: A probability map (or *selection map*) for each depth indicating the likelihood of each pixel having that depth.

Weight-sum image synthesis: Can be interpreted as expectation among depth

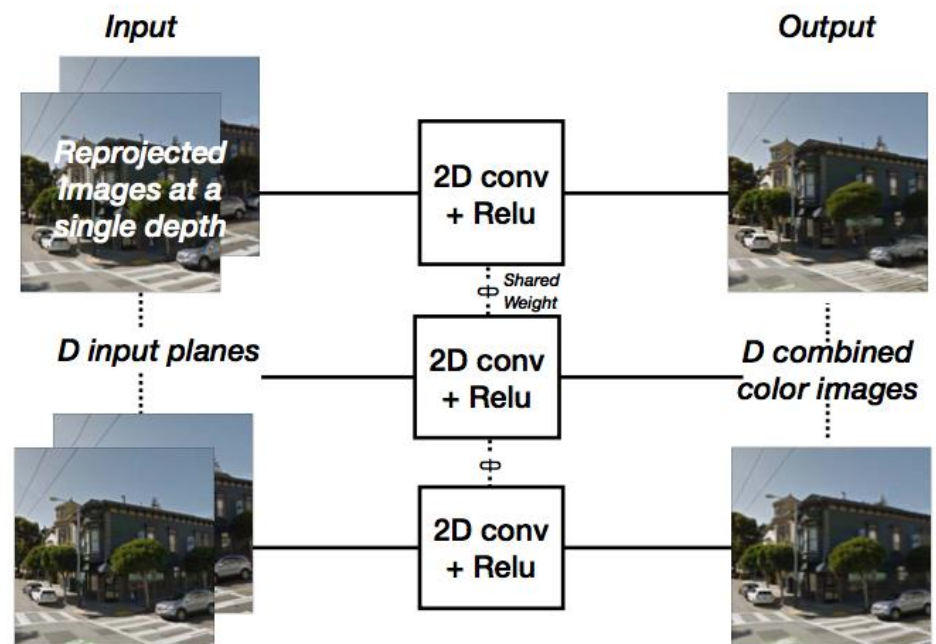


DeepStereo: Color Tower

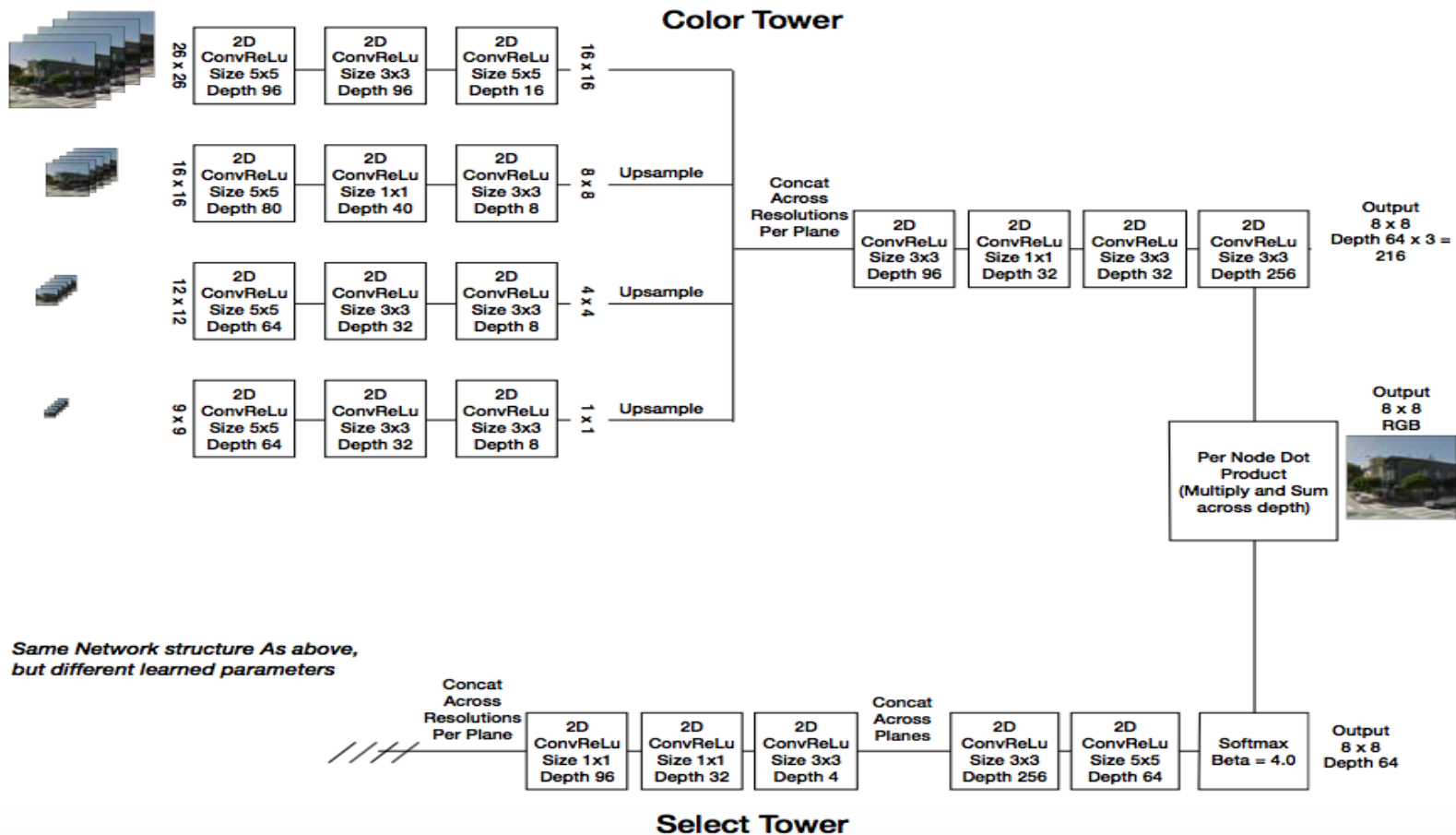
Bonus: D input planes reduce the effect of occlusion

OUTPUT: 3D volume of nodes -> R,G,B channels each pixel

$$c_{i,j}^f = \sum s_{i,j,z} c_{i,j,z}.$$



DeepStereo: Training



DeepStereo: Results



(a) Our result.



(b) Reference image.



(c) Crops of the five input panoramas.

Figure 8: San Francisco park.



(a) Our result.



(b) Reference image.



(c) Crops of the five input panoramas.

Figure 9: Acropolis Museum.

Part 3

RECONSTRUCTION AND GENERATION OF 3D

Part 3

3D structure reconstruction and generation

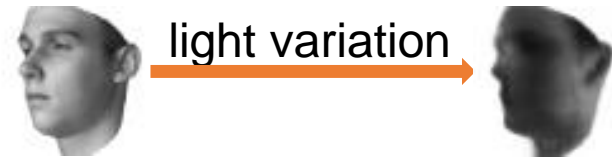
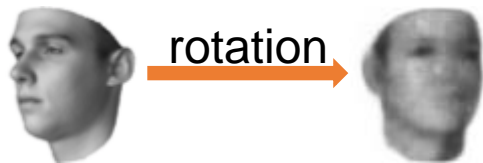
- DC-IGN model (Kulkarni et al. NIPS 2015)
- Perspective transformer networks (Yan et al. NIPS 2016)
- 3D-GAN (Wu et al. NIPS 2016)

Deep Convolutional Inverse Graphics Network (DC-IGN)

- **Motivation:** Can a deep network learn to disentangle factors of image generation such as lighting, rotation, etc.?
- Can we learn a renderer?
- **Recall:** Conditional VAEs and constraints on latent space

DC-IGN

- Generate transformed images w.r.t rotations, light variations, etc.
- Input: 2D image (150 x 150 pixels)
- Output: 2D image that has one different 3D property



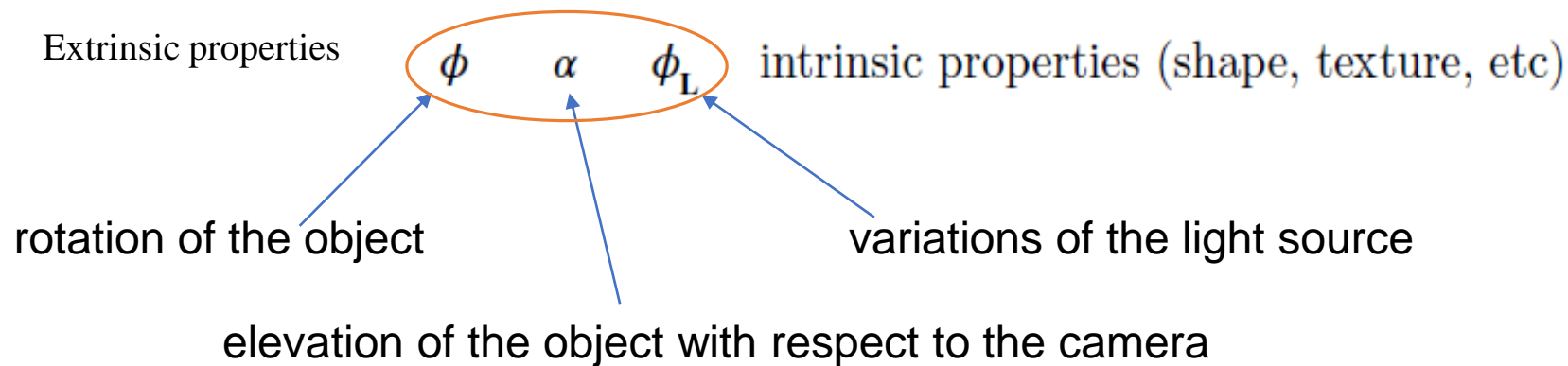
- Learn latent variables that represent complex transformations
- Use an encoder-decoder structure based on VAE

Link: <http://willwhitney.github.io/dc-ign/www/>

DC-IGN

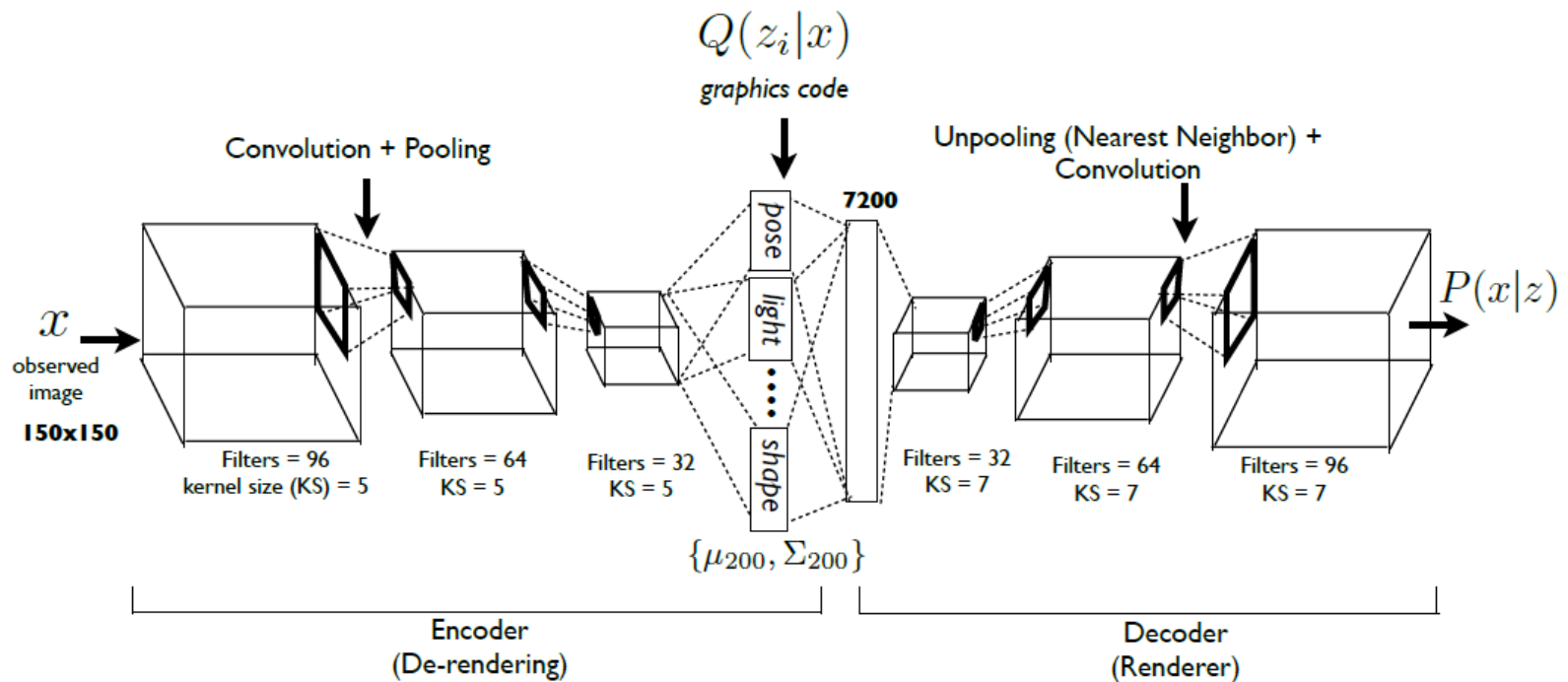
- Graphics codes z

$$Z = \begin{bmatrix} z_1 & z_2 & z_3 & z_{[4,n]} \end{bmatrix}$$



DC-IGN

- Model architecture
 - Deep convolution and de-convolution within a VAE formulation



DC-IGN

- Encoder output: $y_e = \text{encoder}(x)$
- Model parameter: W_e
- Distribution parameters:

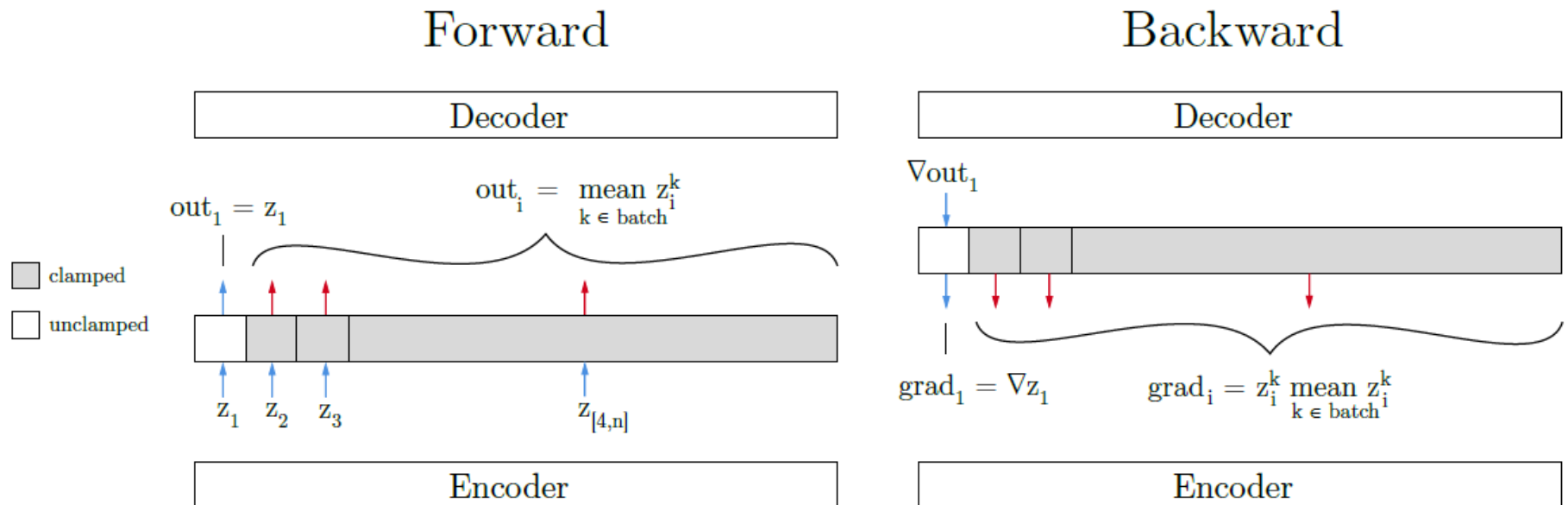
$$\begin{aligned}\theta = (\mu_{z_i}, \Sigma_{z_i}) \quad & \mu_{z_i} = W_e * y_e \\ & \Sigma_{z_i} = \text{diag}(\exp(W_e * y_e)) \\ & \forall i, z_i \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})\end{aligned}$$

- Variational objective function:

$$-\log(P(x|z_i)) + KL(Q(z_i|x)||P(z_i))$$

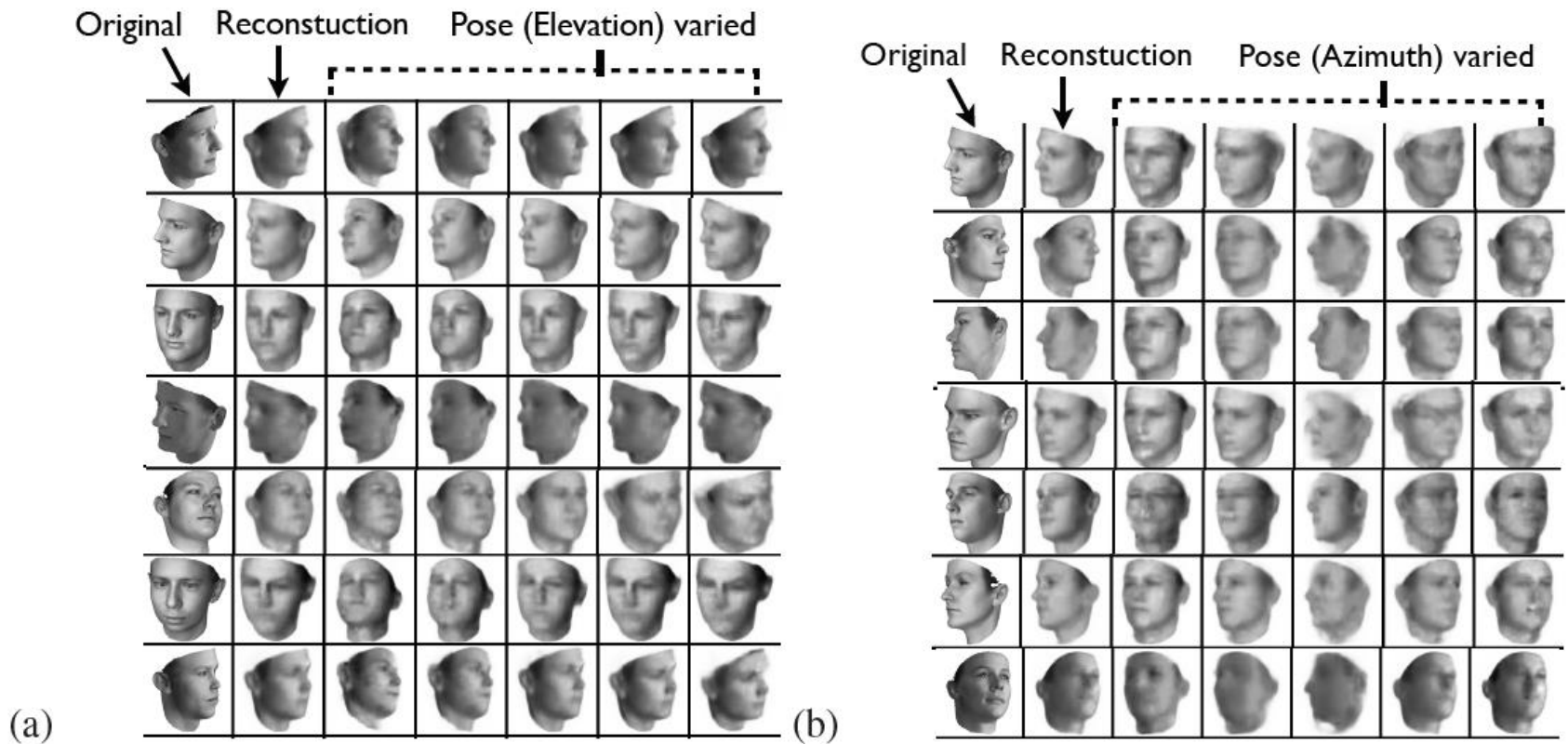
DC-IGN

- Training on a minibatch in which only one extrinsic property changes i.e. Only a specific latent variable changes
- **Key Idea:** Force all other latent variables to be same across examples – Force all of them to be close to the minibatch mean



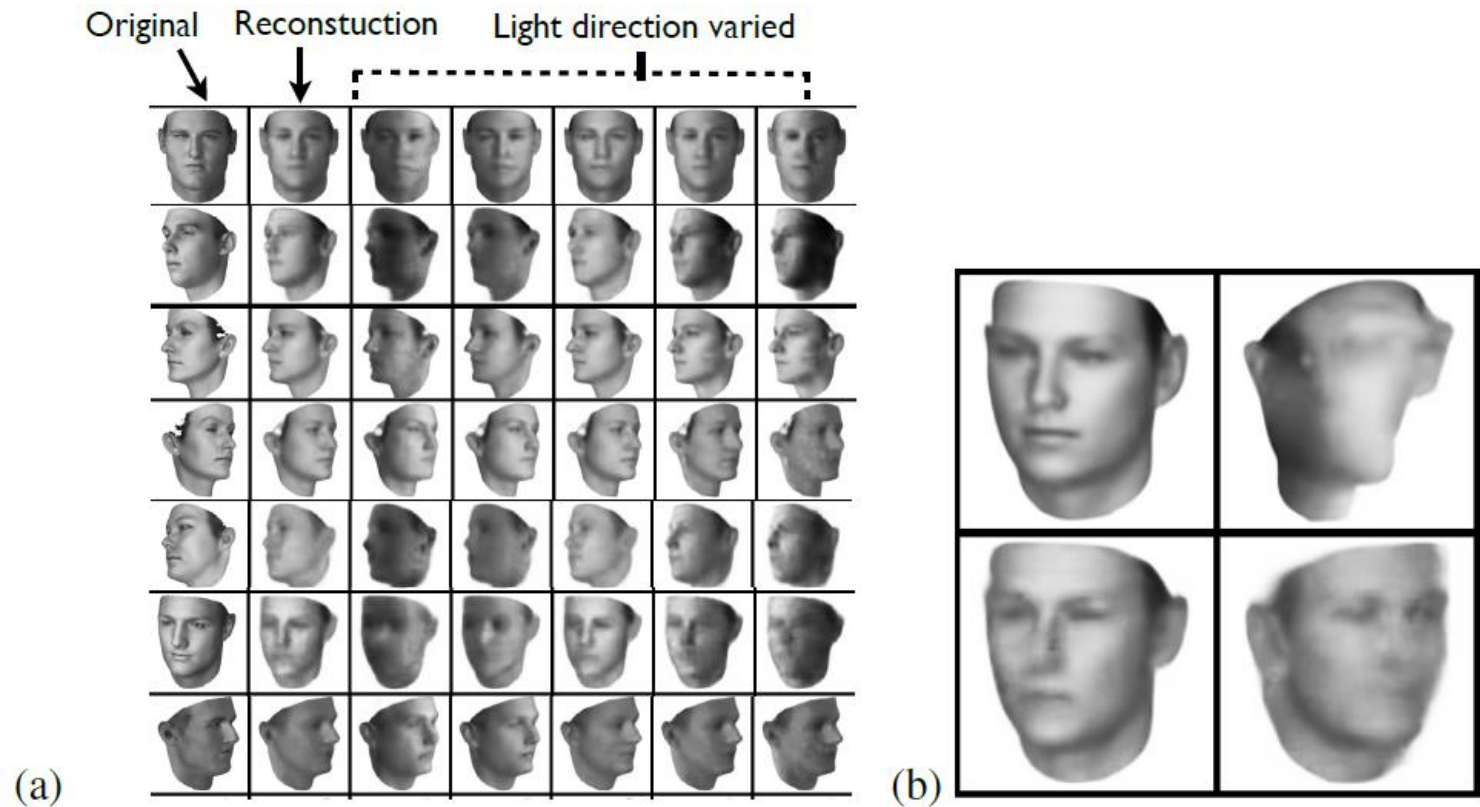
DC-IGN

- Generation w.r.t. manipulation of pose variables



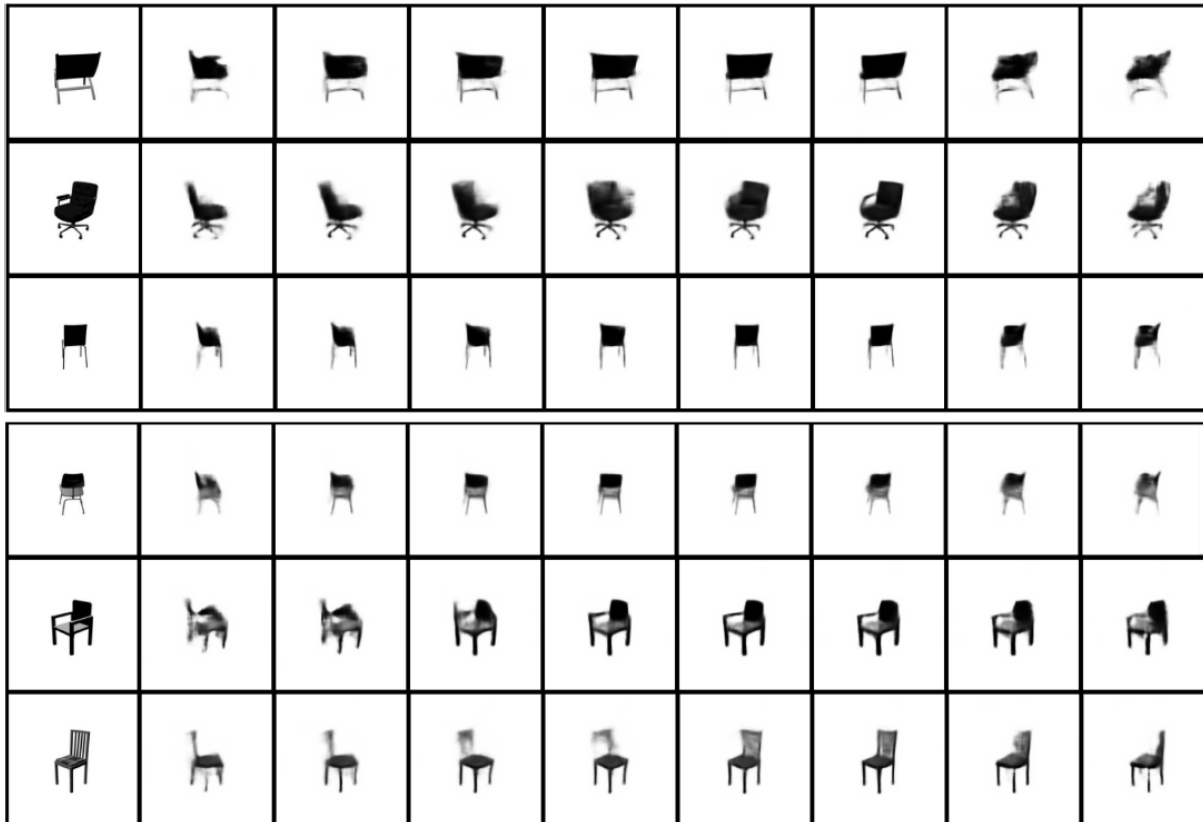
DC-IGN

- Generation w.r.t. light directions



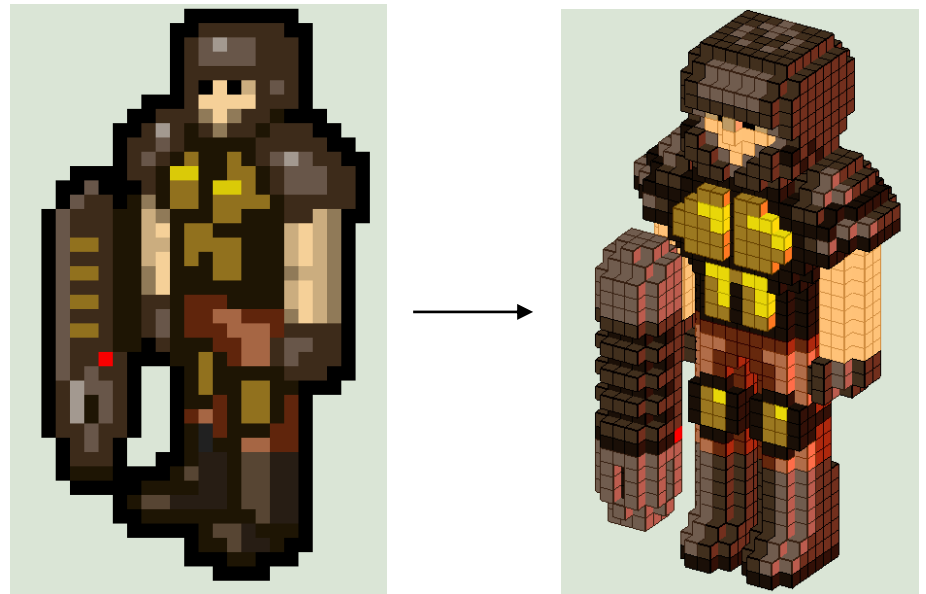
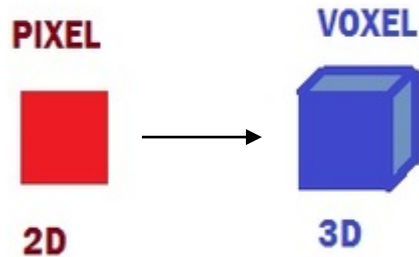
DC-IGN

- Manipulate rotation for a different dataset (chair dataset)



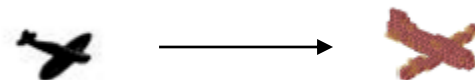
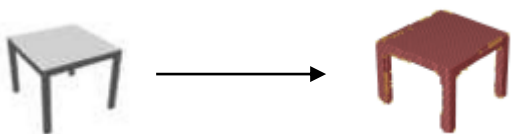
Pixel to Voxel

- DC-IGN: 2D image (pixel) \rightarrow transformed 2D image (pixel)
- Next models: 2D image (pixel) \rightarrow 3D image (voxel)



Perspective Transformer Nets

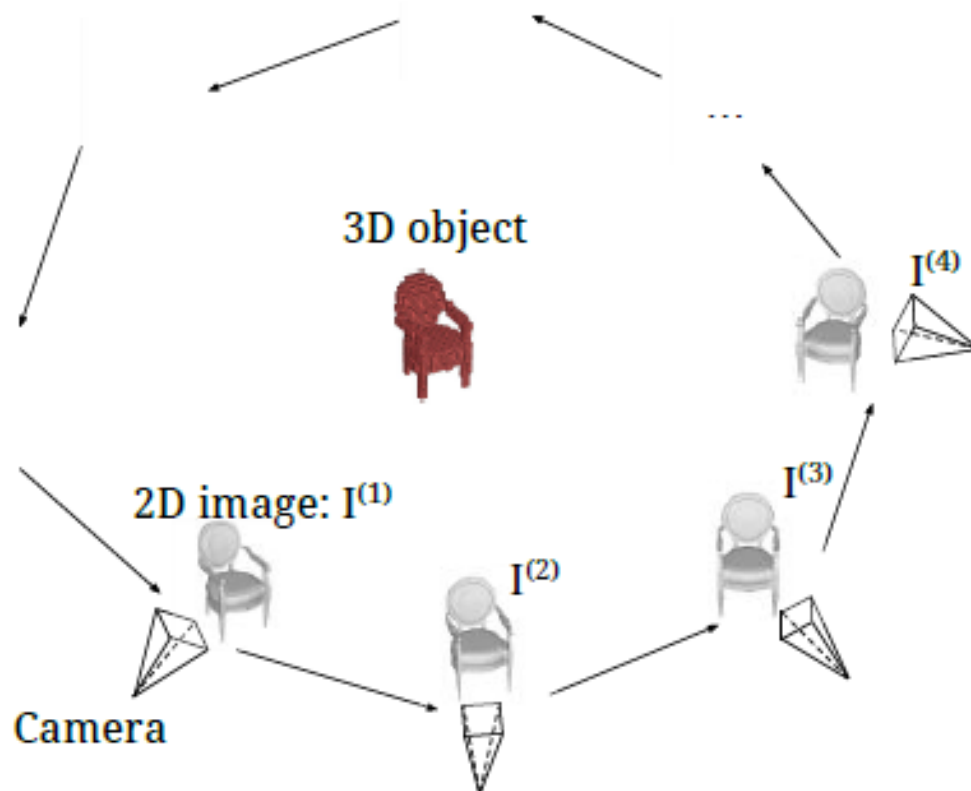
- Predict the underlying true 3D shape of an object given a 2D single image



- Learn 3D object reconstruction without 3D ground-truth data
- Use different 2D images from multiple viewpoints
- Define two loss functions to generate 3D structures

Perspective Transformer Nets

- $I^{(k)}$: 2D image from k -th viewpoint $\alpha^{(k)}$ by projection
 $\rightarrow I^{(k)} = P(X ; \alpha^{(k)})$



Perspective Transformer Nets

Case 1 we know the ground truth 3D volume V

- Generate 3D volume $\hat{V} = f(I^{(k)}) = g(h(I^{(k)}))$

where $h(\cdot)$ learns a viewpoint-invariant latent representation

$g(\cdot)$ is a volume generator $h(I^{(1)}) = h(I^{(2)}) = \dots = h(I^{(k)})$

- Loss function

$$\mathcal{L}_{vol}(I^{(k)}) = \|f(I^{(k)}) - \mathbf{V}\|_2^2$$

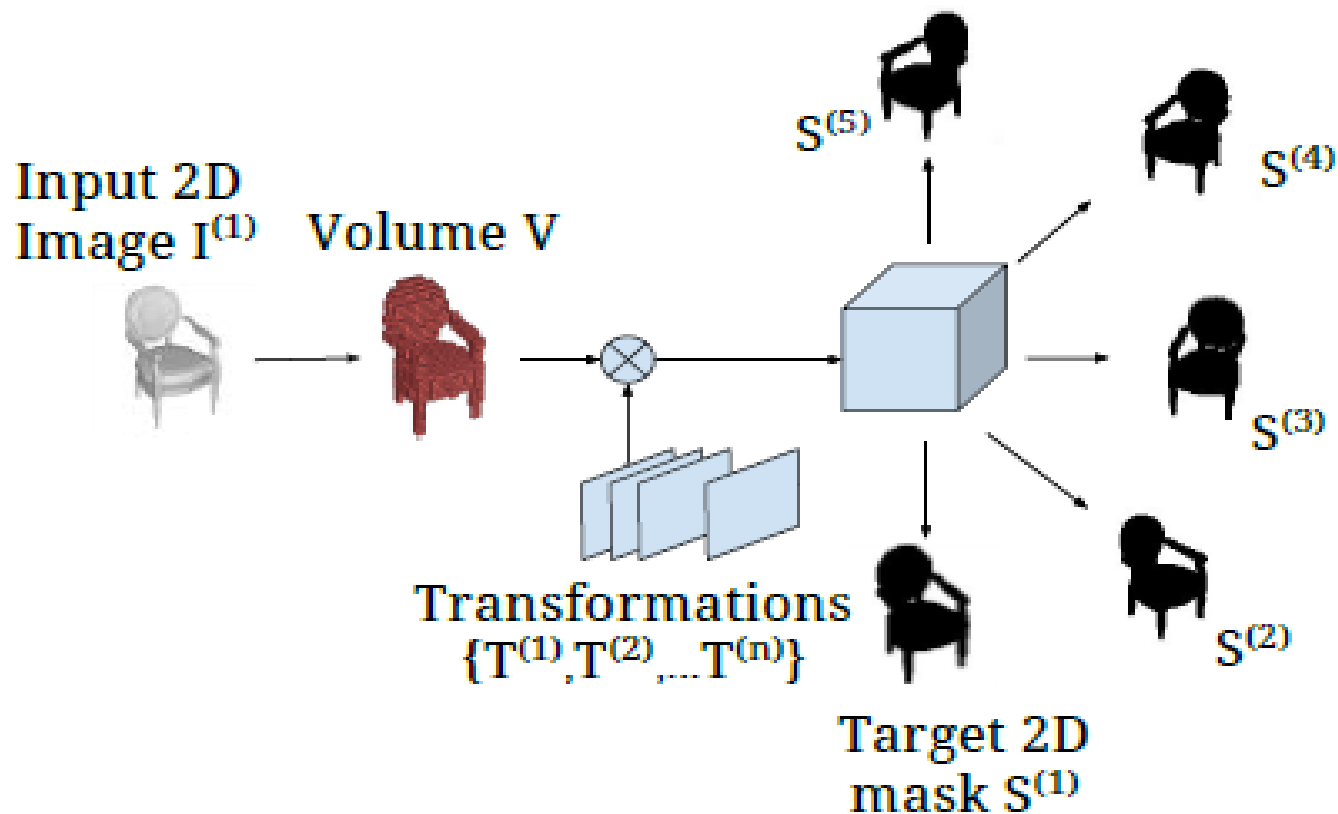
- However, ground truth volume may not be available in practice

Perspective Transformer Nets

Case 2 we do NOT know the ground truth 3D volume V

- Use 2D silhouette images
- $S^{(j)}$: ground truth 2D silhouette image for the j -th viewpoint $\alpha^{(j)}$
- $\hat{S}^{(j)}$: generated silhouettes $P(f(I^{(k)}); \alpha^{(j)})$
- Loss function:
$$\mathcal{L}_{proj}(I^{(k)}) = \sum_{j=1}^n \mathcal{L}_{proj}^{(j)}(I^{(k)}; S^{(j)}, \alpha^{(j)}) = \frac{1}{n} \sum_{j=1}^n \|P(f(I^{(k)}); \alpha^{(j)}) - S^{(j)}\|_2^2$$

Perspective Transformer Nets

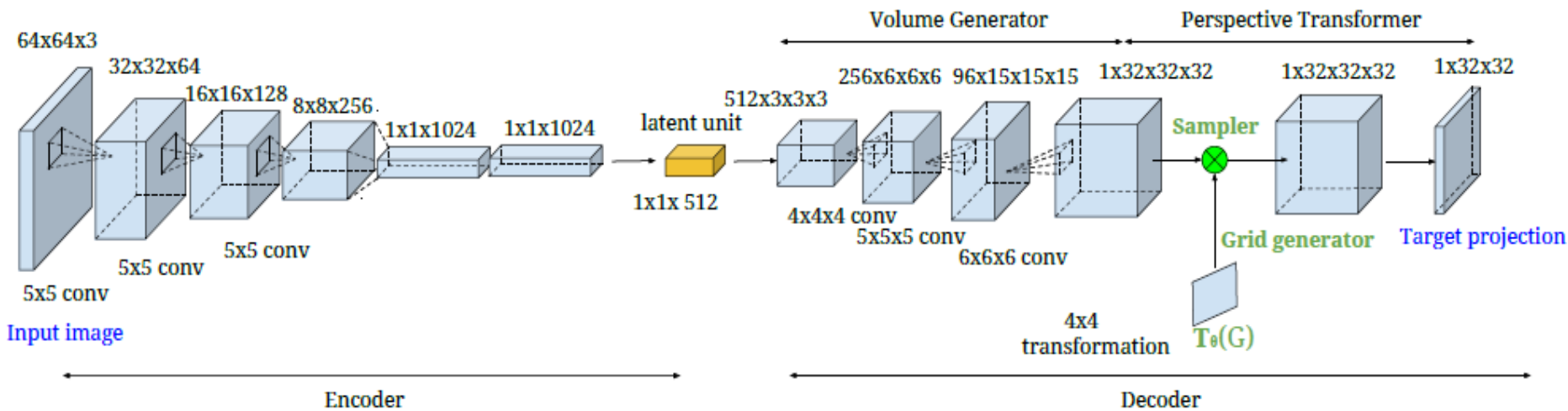


Perspective Transformer Nets

- Consider a combination of \mathcal{L}_{vol} and \mathcal{L}_{proj}

$$\begin{aligned}\mathcal{L}_{comb}(I^{(k)}) &= \lambda_{proj}\mathcal{L}_{proj}(I^{(k)}) + \lambda_{vol}\mathcal{L}_{vol}(I^{(k)}) \\ &= \lambda_{proj}\frac{1}{n}\sum_{j=1}^n ||P(f(I^{(k)}); \alpha^{(j)}) - S^{(j)}||_2^2 + \lambda_{vol} ||f(I^{(k)}) - \mathbf{V}||_2^2\end{aligned}$$

$$f(I^{(k)}) = g(h(I^{(k)}))$$



Perspective Transformer Nets

- How to obtain 2D silhouette $\hat{S}^{(j)}$ - perspective projection
- Transformation matrix

$$\Theta_{4 \times 4} = \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$

where K: camera calibration matrix & (R, t): extrinsic parameters

- Perspective transformation: $\mathbf{p}_i^s \sim \Theta_{4 \times 4} \mathbf{p}_i^t$

where 3D coordinates: $\mathbf{p}_i^s = (x_i^s, y_i^s, z_i^s, 1)$

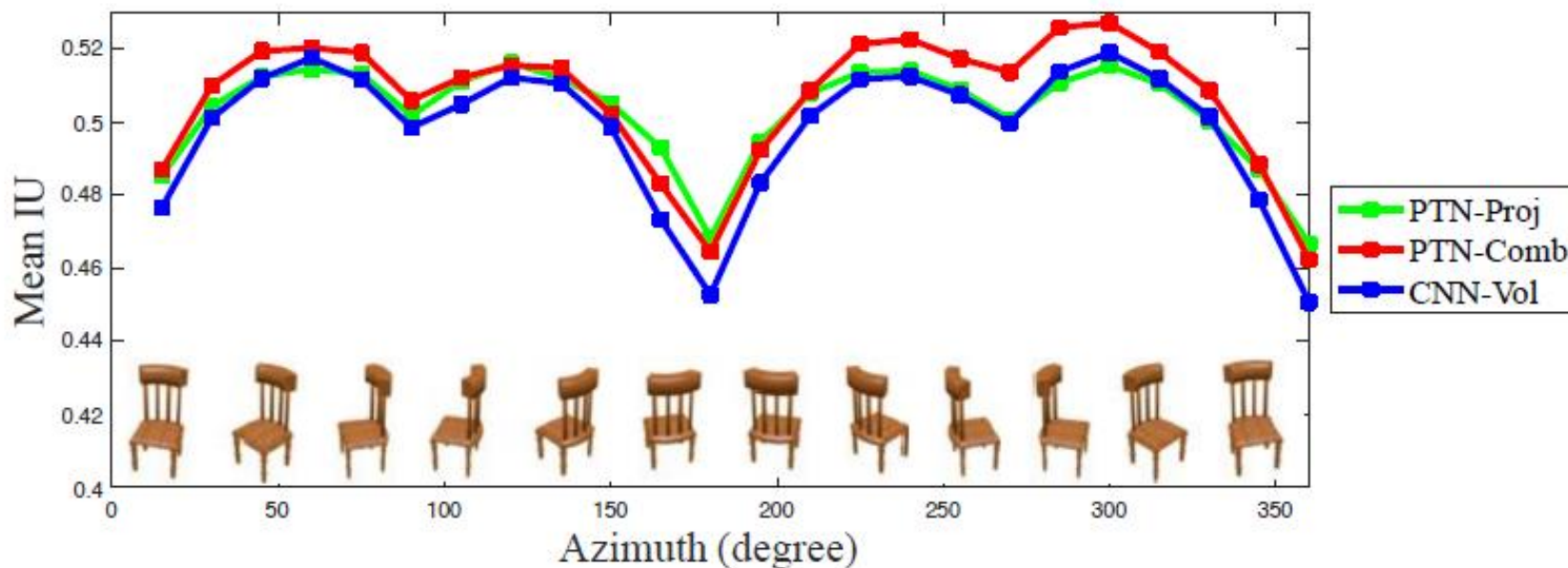
screen coordinates: $\mathbf{p}_i^t = (x_i^t, y_i^t, 1, d_i^t)$

Perspective Transformer Nets









































































- Use spatial transformer network (Jaderberg et al. NIPS 2015)
 - (1) Perform dense sampling from input volume in 3D coordinates to output volume in screen coordinates
 - (2) Flatten the 3D spatial output across disparity dimension.

Perspective Transformer Nets

- Training on single category

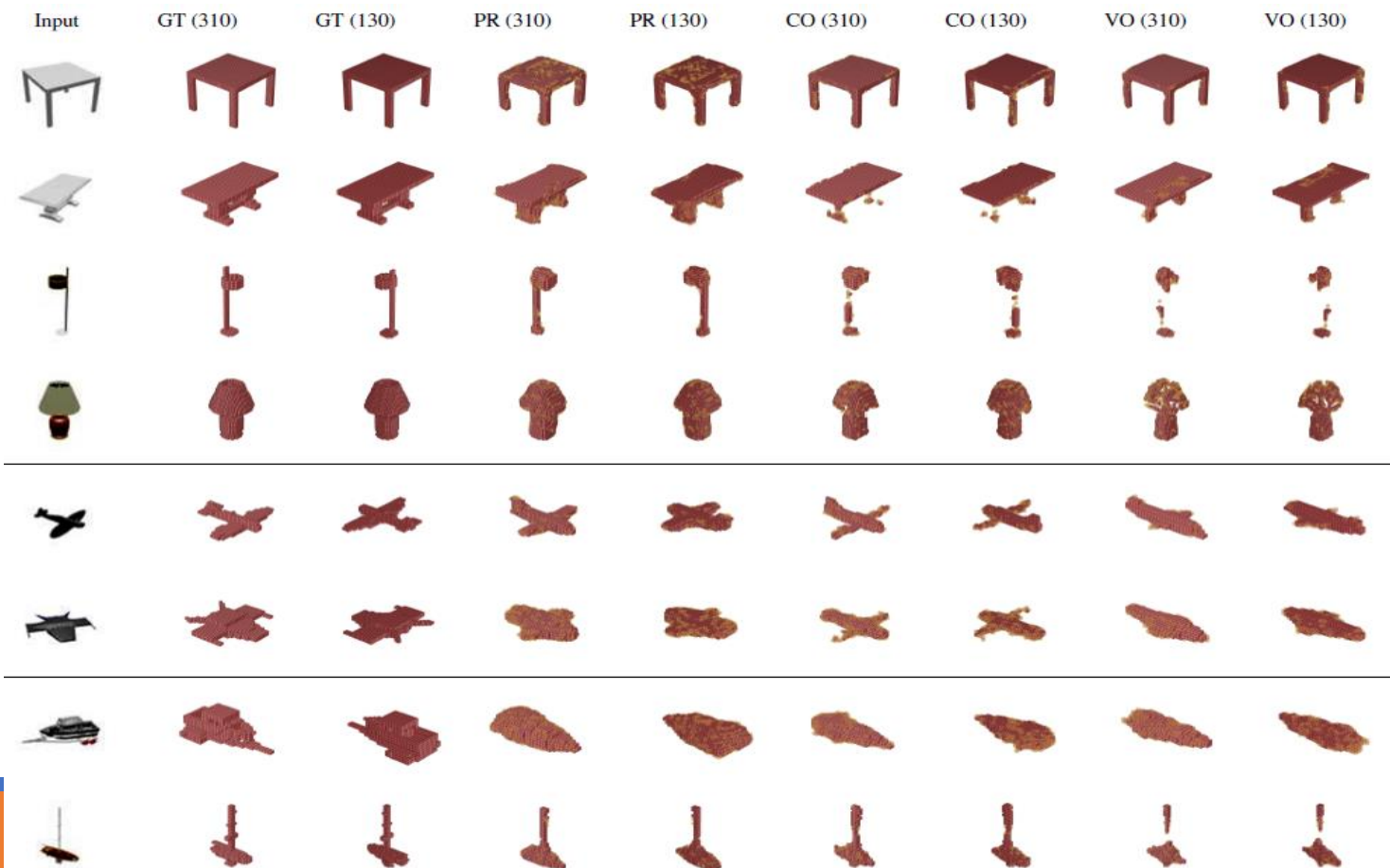


Perspective Transformer Nets

Input	GT (310)	GT (130)	PR (310)	PR (130)	CO (310)	CO (130)	VO (310)	VO (130)
								
								
								
								
								
								
								
								

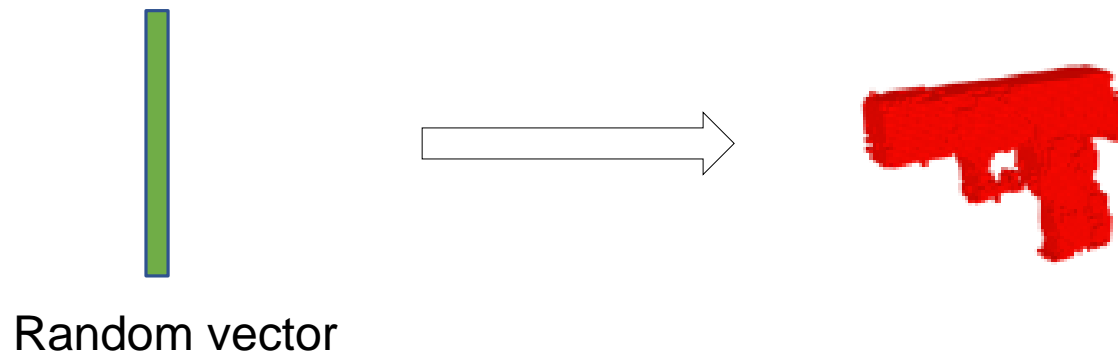
Perspective Transformer Nets

- Training on multiple category



3D-GAN

- Generate an object in 3D voxel space from a randomly sampled vector



- Use the Generative Adversarial Network (GAN)
- Map randomly sampled vector in a latent space to an object in 3D voxel space

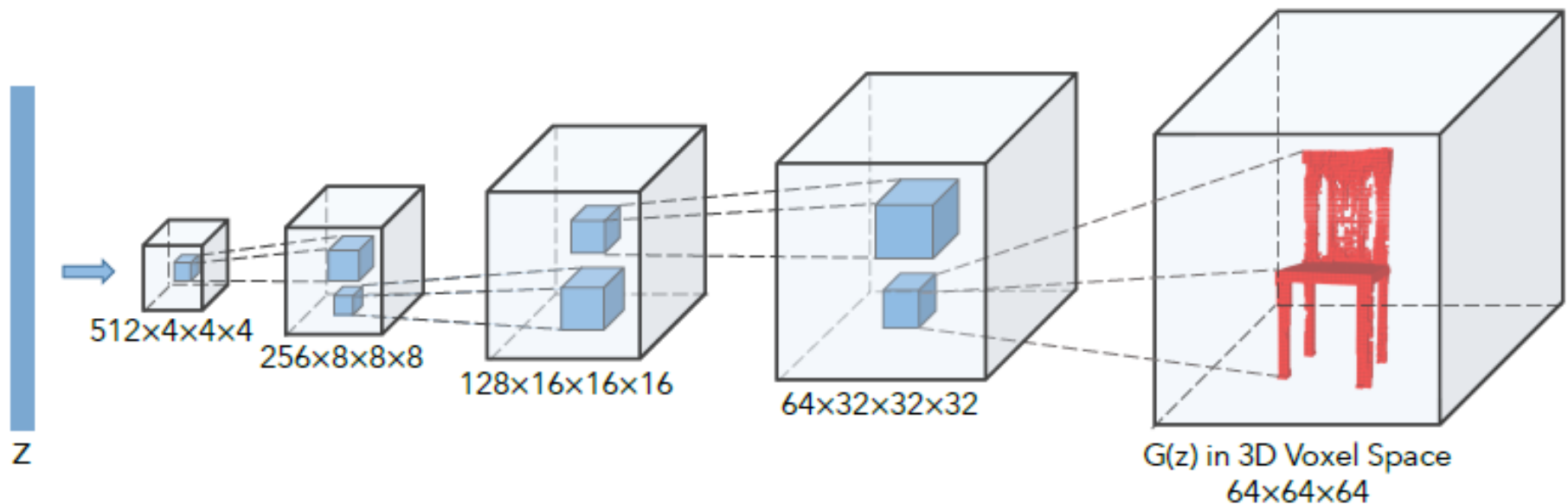
Link: <http://3dgan.csail.mit.edu/>

3D-GAN

- GAN – Generator + Discriminator
- Generator $G: z \rightarrow G(z)$
where z : latent vector (200 dimension),
 $G(z)$: 3D object in 3D voxel space (64 x 64 x 64 cube)
- Discriminator D : output a confidence value of whether an input is real or synthetic
- Overall adversarial loss function: $L_{3D-GAN} = \log D(x) + \log(1 - D(G(z)))$
where x : a real object in a 64 x 64 x 64 space,
 z : randomly sampled noise from $p(z)$

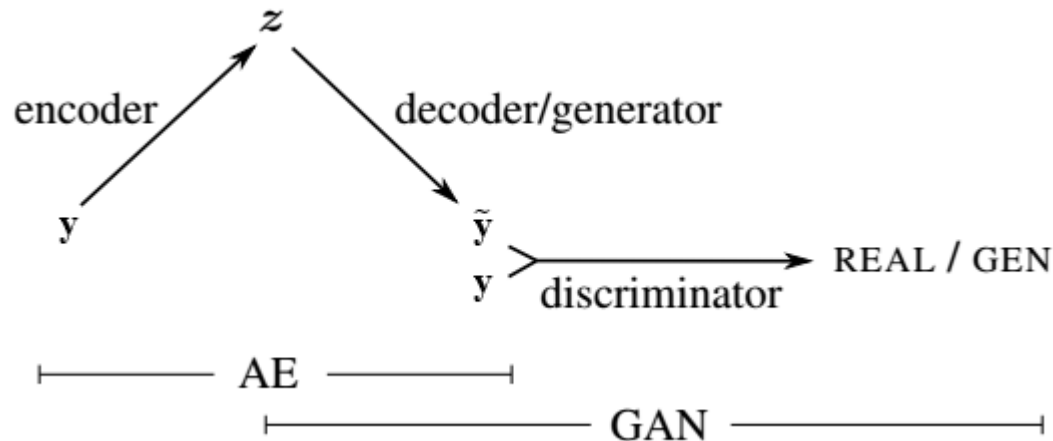
3D-GAN

- Network structure of generator in 3D-GAN



3D-VAE-GAN

- Extension to 3D-GAN
- Inspired by VAE-GAN of Larsen et al. (ICML 2016)



- Take a 2D image as input to generate a 3D object

3D-VAE-GAN

- New loss function

$$L = L_{\text{3D-GAN}} + \alpha_1 L_{\text{KL}} + \alpha_2 L_{\text{recon}}$$

$$L_{\text{3D-GAN}} = \log D(x) + \log(1 - D(G(z)))$$

$$L_{\text{KL}} = D_{\text{KL}}(q(z|y) \parallel p(z))$$

$$L_{\text{recon}} = \|G(E(y)) - x\|_2$$

where x : a 3D shape from the training set,

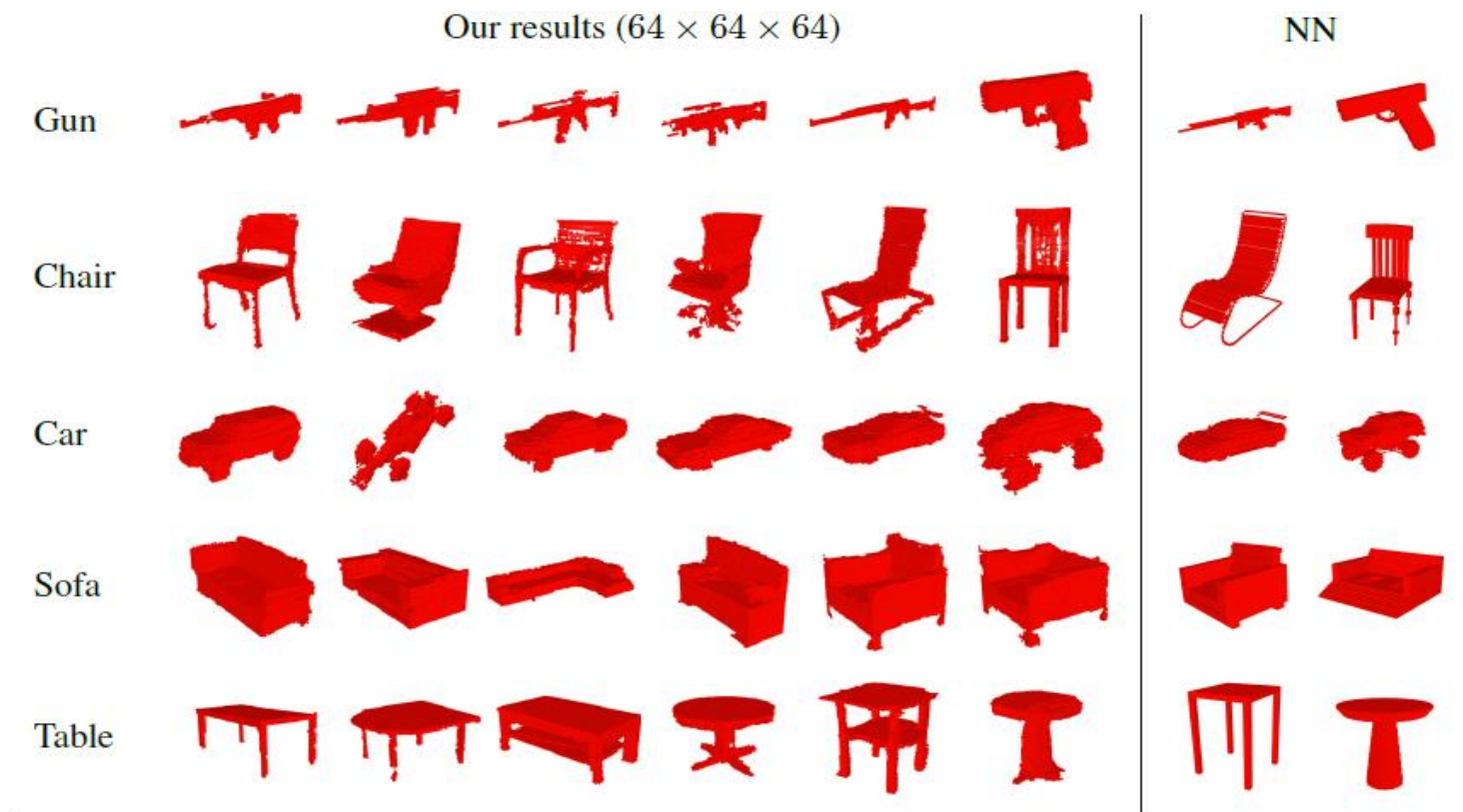
y : x 's corresponding 2D image,

$q(z|y)$: variational distribution of z ,

$p(z)$: multivariate Gaussian prior

3D-GAN

- 3D object generation



3D-GAN

- 3D object classification

Supervision	Pretraining	Method	Classification (Accuracy)	
			ModelNet40	ModelNet10
Category labels	ImageNet	MVCNN [Su et al., 2015a]	90.1%	-
		MVCNN-MultiRes [Qi et al., 2016]	91.4%	-
	None	3D ShapeNets [Wu et al., 2015]	77.3%	83.5%
		DeepPano [Shi et al., 2015]	77.6%	85.5%
		VoxNet [Maturana and Scherer, 2015]	83.0%	92.0%
		ORION [Sedaghat et al., 2016]	-	93.8%
	Unsupervised	SPH [Kazhdan et al., 2003]	68.2%	79.8%
		LFD [Chen et al., 2003]	75.5%	79.9%
		T-L Network [Girdhar et al., 2016]	74.4%	-
		VConv-DAE [Sharma et al., 2016]	75.5%	80.5%
		3D-GAN (ours)	83.3%	91.0%

3D-VAE-GAN

- Single image 3D reconstruction

Method	Bed	Bookcase	Chair	Desk	Sofa	Table	Mean
AlexNet-fc8 [Girdhar et al., 2016]	29.5	17.3	20.4	19.7	38.8	16.0	23.6
AlexNet-conv4 [Girdhar et al., 2016]	38.2	26.6	31.4	26.6	69.3	19.1	35.2
T-L Network [Girdhar et al., 2016]	56.3	30.2	32.9	25.8	71.7	23.3	40.0
3D-VAE-GAN (jointly trained)	49.1	31.9	42.6	34.8	79.8	33.1	45.2
3D-VAE-GAN (separately trained)	63.2	46.3	47.2	40.7	78.8	42.3	53.1

Average precision for voxel prediction on the IKEA dataset.

3D-VAE-GAN

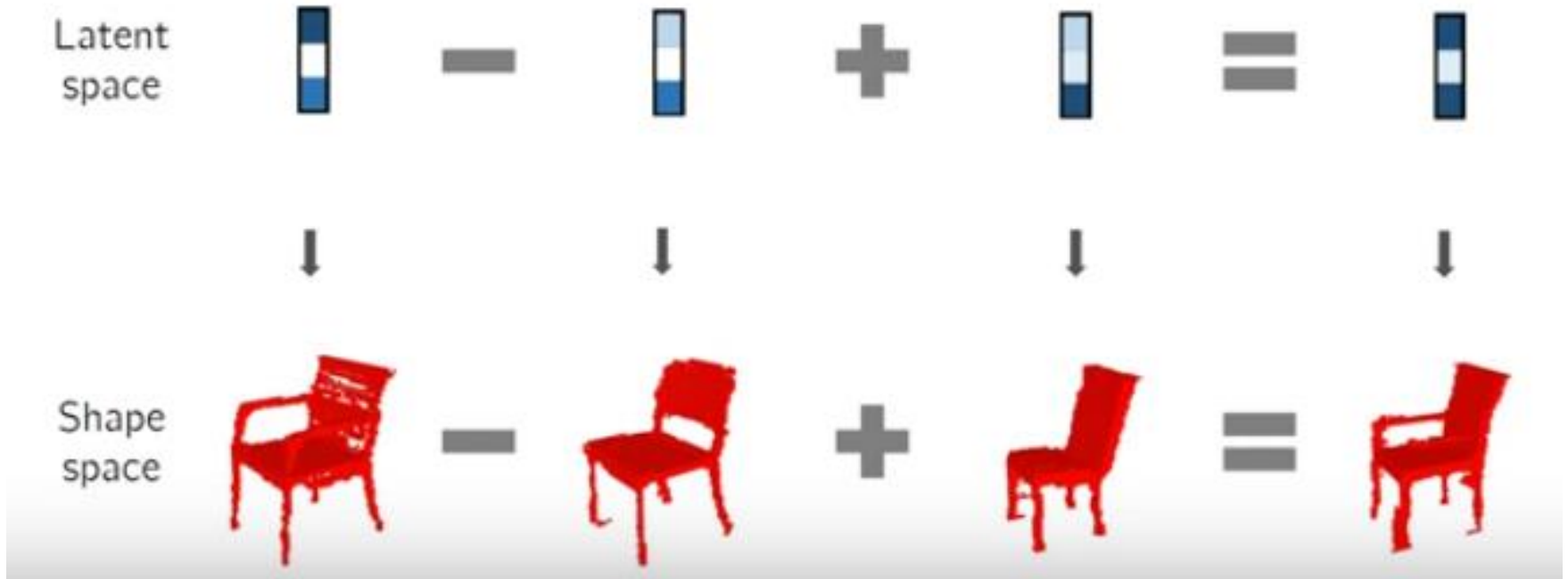
- Single image 3D reconstruction



Qualitative results of single image 3D reconstruction on the IKEA dataset

3D-GAN

- Shape arithmetic for chairs



Summary

- 3D vision and graphics based on deep learning
 - Pose estimation (3D recovery from 2D images)
 - Novel Image / View synthesis
 - Reconstruction and generation of 3D

References

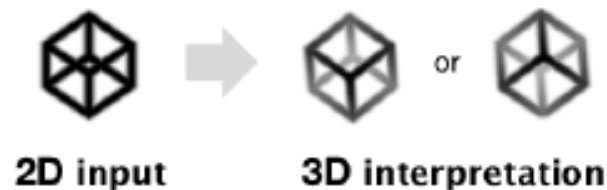
- John Flynn, Ivan Neulander, James Philbin, Noah Snavely, DeepStereo: Learning to Predict New Views from the World's Imagery, CVPR 2016.
- Max Jaderberg, Karen Simonyan, Andrew Zisserman, Koray Kavukcuoglu, Spatial Transformer Networks, NIPS 2015
- Alex Kendall, Matthew Grimes, Roberto Cipolla, PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization, ICCV 2015.
- Tejas D. Kulkarni, William F. Whitney, Pushmeet Kohli, Josh Tenenbaum, Deep Convolutional Inverse Graphics Network, NIPS 2015.
- Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, Nicolas Heess, Unsupervised Learning of 3D Structure from Images, NIPS 2016.
- Hao Su, Charles R. Qi, Yangyan Li, Leonidas J. Guibas, Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. ICCV 2015.
- Jiajun Wu, Tianfan Xue, Joseph J. Lim, Yuandong Tian, Joshua B. Tenenbaum, Antonio Torralba, and William T. Freeman, Single Image 3D Interpreter Network, ECCV 2016.
- Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, Joshua B. Tenenbaum, Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling, NIPS 2016.
- Xinchun Yan, Jimei Yang, Ersin Yumer, Yijie Guo, Honglak Lee, Perspective Transformer Nets: Learning Single-View 3D Object Reconstruction without 3D Supervision, NIPS 2016.
- Jimei Yang, Scott Reed, Ming-Hsuan Yang, Honglak Lee, Weakly-supervised Disentangling with Recurrent Transformations for 3D View Synthesis, NIPS 2015.

Backup slides

REZENDE ET AL., NIPS 2016.

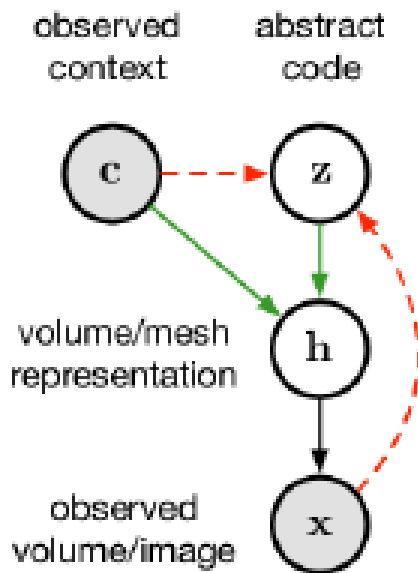
Rezende et al. (NIPS 2016)

- Construct the underlying 3D structures from 2D observations
- Learn a generative model of 3D structures
- Recover the structure from 2D images via inference



Rezende et al. (NIPS 2016)

- Consider a conditional latent variable model



x : observed image or volume

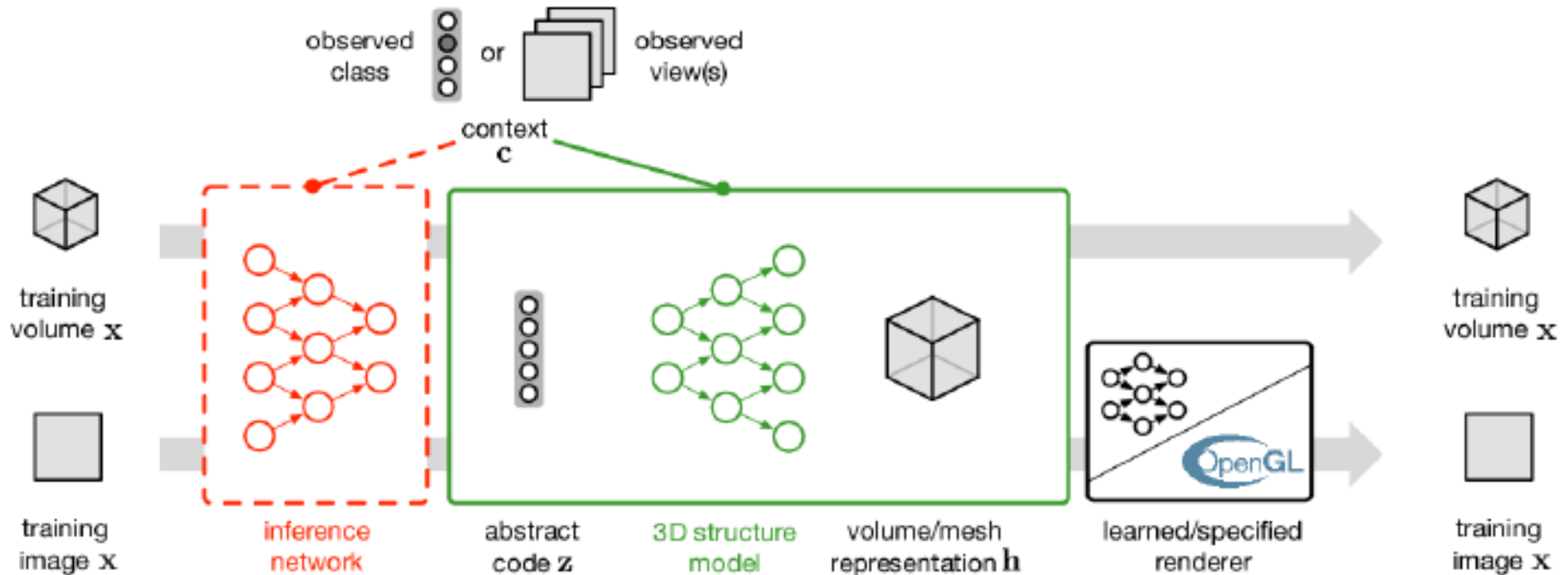
c : observed contextual information (nothing, object class label, or one or more views of the scene from different cameras)

z : low-dimensional codes of latent manifold of object shapes

h : 3D representations (volume or mesh)

Rezende et al. (NIPS 2016)

- Proposed framework



Rezende et al. (NIPS 2016)

- Sequential generative process: refinement of hidden representations

Latents $\mathbf{z}_t \sim \mathcal{N}(\cdot \mathbf{0}, \mathbf{1})$	3D representation $\mathbf{h}_t = f_{\text{write}}(\mathbf{s}_t, \mathbf{h}_{t-1}; \theta_w)$
Encoding $\mathbf{e}_t = f_{\text{read}}(\mathbf{c}, \mathbf{s}_{t-1}; \theta_r)$	2D projection $\hat{\mathbf{x}} = \text{Proj}(\mathbf{h}_T, \mathbf{s}_T; \theta_p)$
Hidden state $\mathbf{s}_t = f_{\text{state}}(\mathbf{s}_{t-1}, \mathbf{z}_t, \mathbf{e}_t; \theta_s)$	Observation $\mathbf{x} \sim p(\mathbf{x} \hat{\mathbf{x}})$

- Each step generates an independent set of \mathbf{z}_t
- f_{read} : task dependent context encoder
- f_{state} : transition function (fully connected LSTM)
- f_{write} : volumetric spatial transformer (Jaderberg et al. NIPS 2015)
- Proj: projection operator from latent 3D representation \mathbf{h}_T to the training data's domain

Rezende et al. (NIPS 2016)

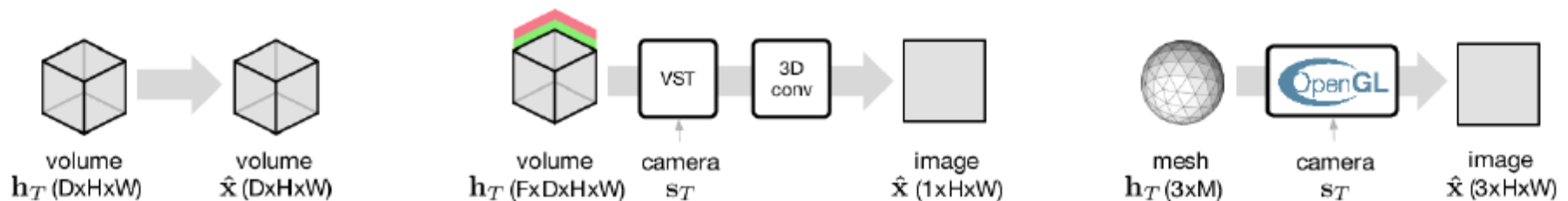
- Volumetric spatial transformer (VST)

$$\text{VST}(\mathbf{x}, \mathbf{h}) = [\kappa_d(\mathbf{h}) \otimes \kappa_h(\mathbf{h}) \otimes \kappa_w(\mathbf{h})] * \mathbf{x}$$

where κ_d , κ_h and κ_w : simple affine transformation of a 3D grid of points that uniformly covers the input image

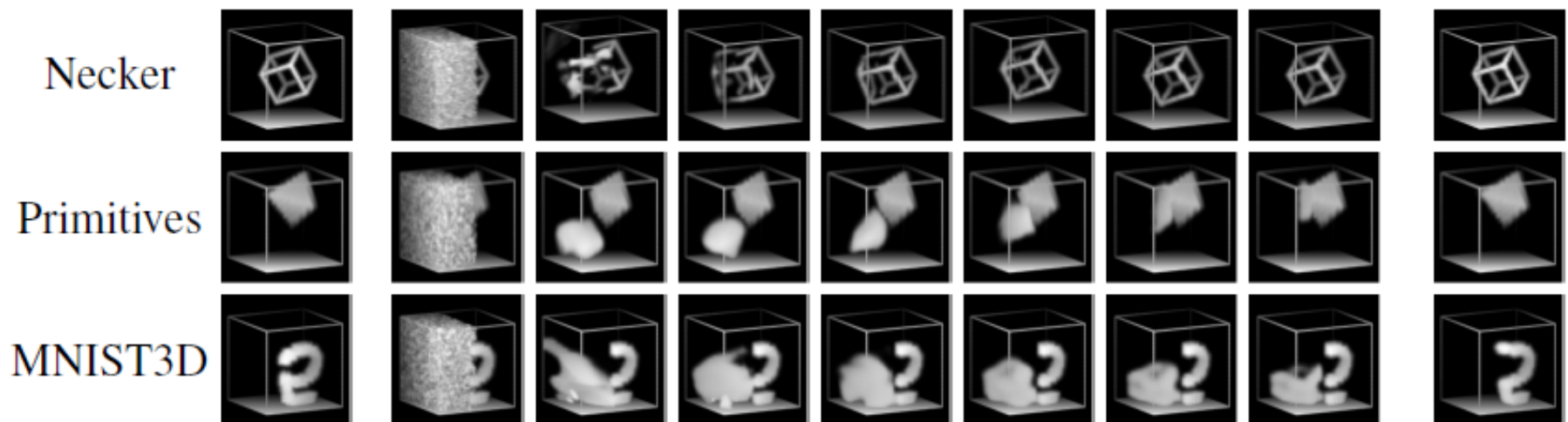
Rezende et al. (NIPS 2016)

- Projection operator (3 types)
 - 3D \rightarrow 3D (identity)
 - 3D \rightarrow 2D neural network projection (learned)
 - 3D \rightarrow 2D OpenGL projection (fixed)



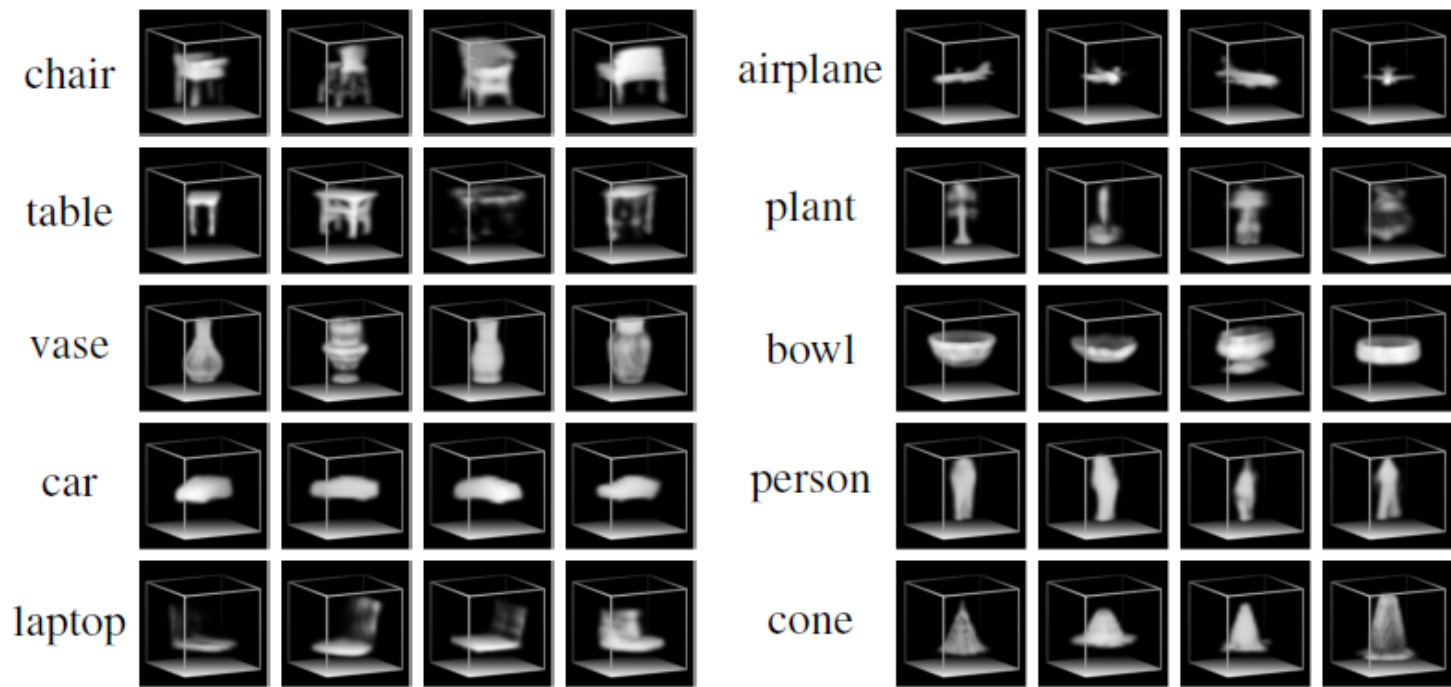
Rezende et al. (NIPS 2016)

- Probabilistic volume completion (Necker cube, Primitives and MNIST3D)



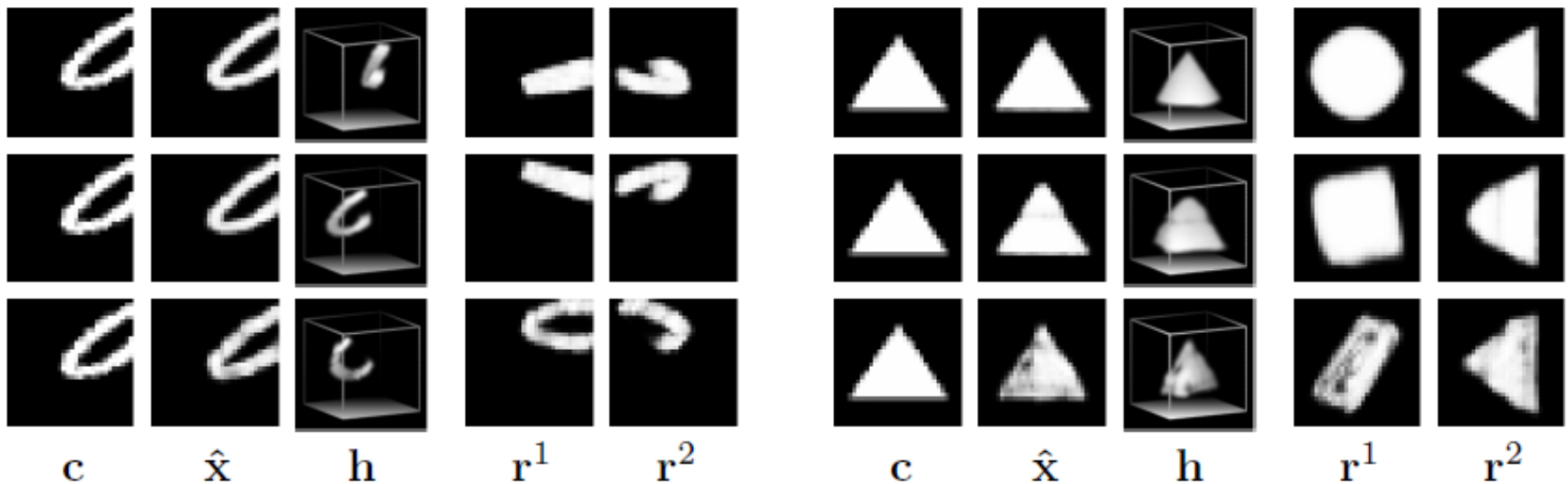
Rezende et al. (NIPS 2016)

- Class-conditional samples (one-hot encoding of class as context)



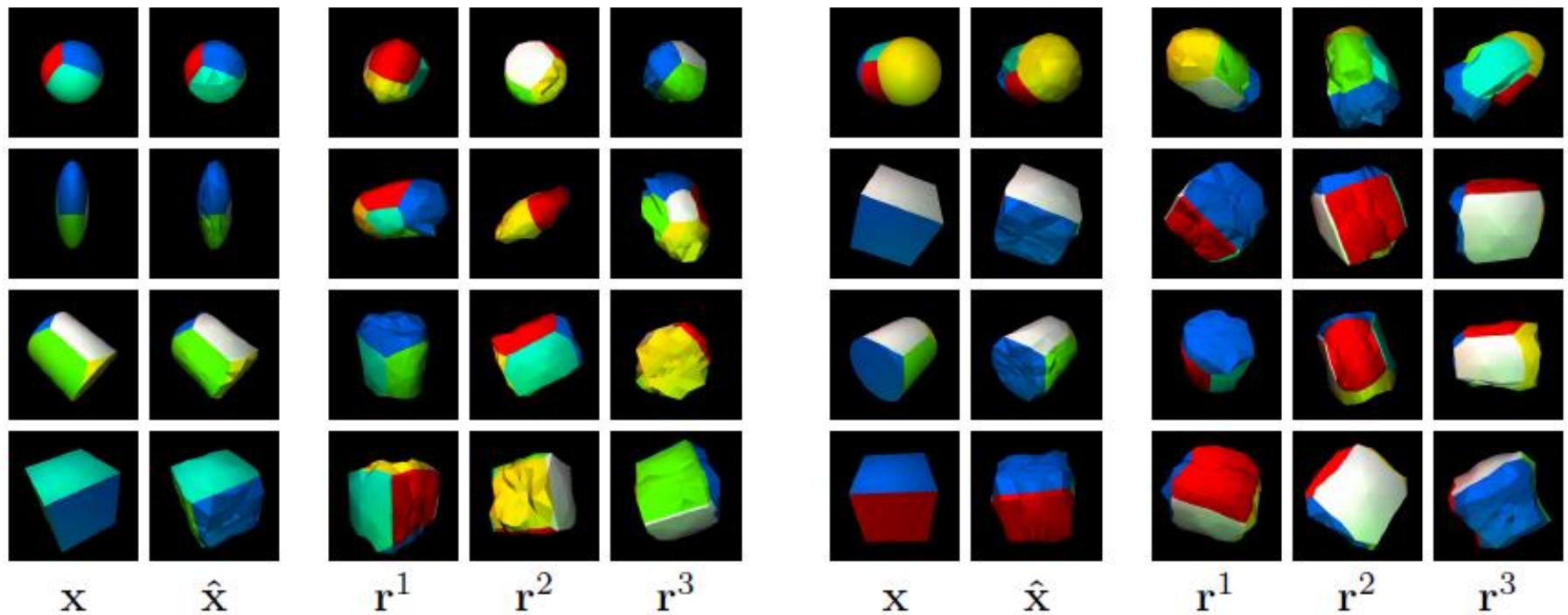
Rezende et al. (NIPS 2016)

- Recover 3D structure from 2D images



Rezende et al. (NIPS 2016)

- Unsupervised learning of 3D structure (mesh representations)



Link: <https://www.youtube.com/watch?v=stvDAGQwL5c>