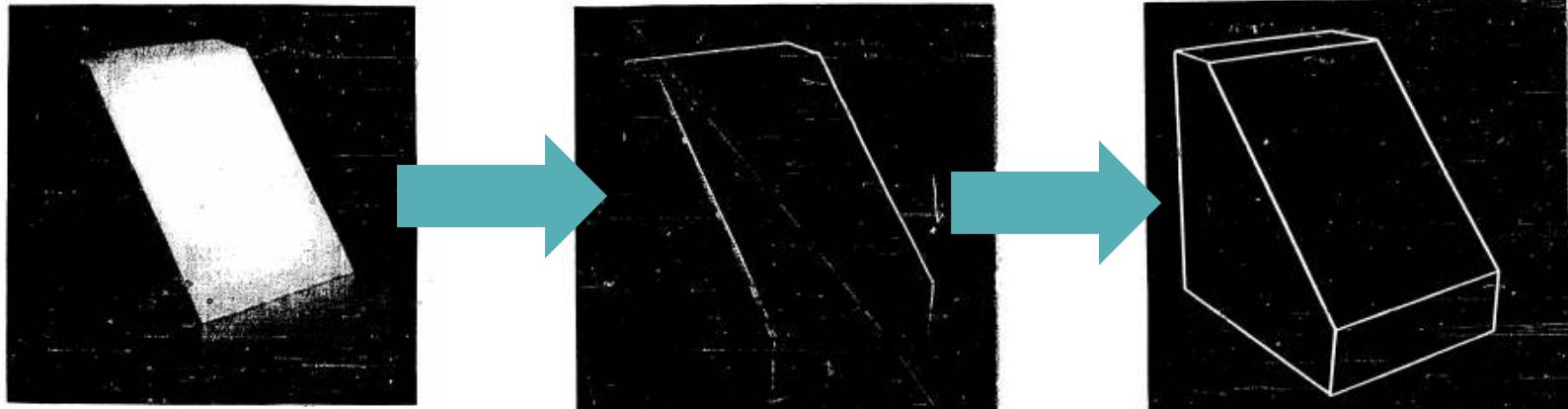


AI and 3D Geometry for [Self-Supervised] 3D Scene Understanding

Vincent Lepetit

at the beginning of computer vision



Machine perception of three-dimensional solids. Roberts, 1963.

3D Scene Understanding from Images



3D Scene Understanding from Images

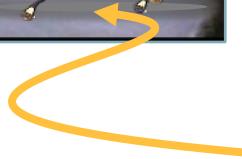


3D Scene Understanding from Images



Applications

- Robotics;



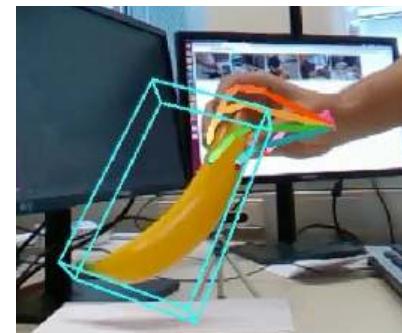
Applications

Digital Twins of factories or construction sites;



Applications

- Augmented Reality and Telepresence;

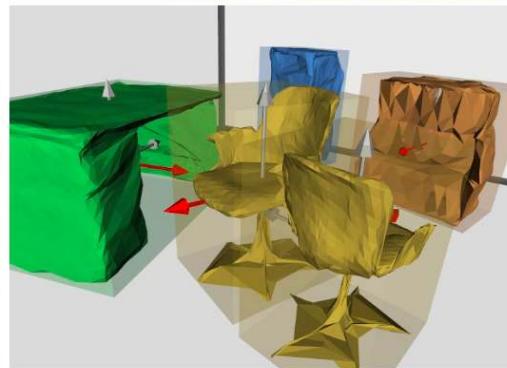
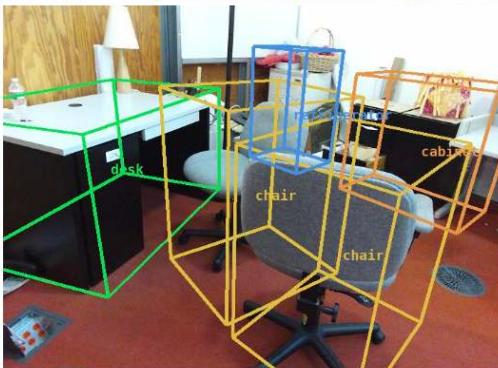
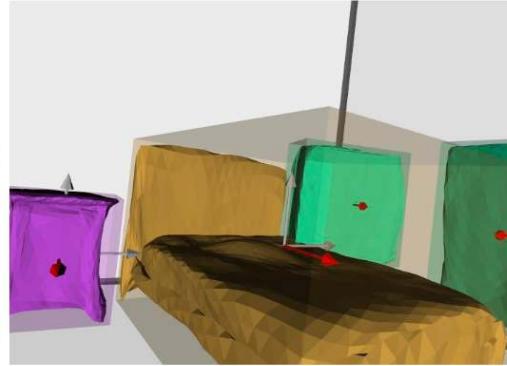
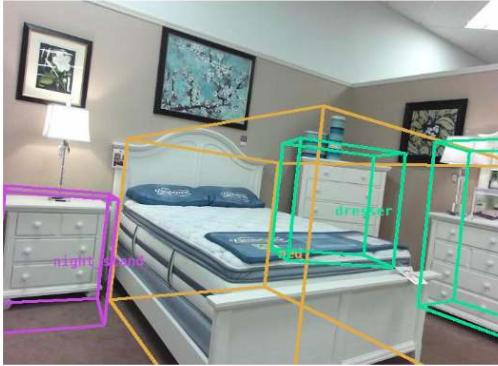


Some results



[Alexander Grabner, Peter Roth, and Vincent Lepetit, Geometric Correspondence Fields: Learned Differentiable Rendering for 3D Pose Refinement in the Wild, ECCV 2020]

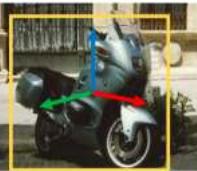
Some results



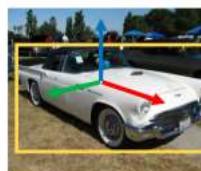
[Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes from a Single Image. Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, Jian Jun Zhang. CVPR 2020]

some results (few-shot learning)

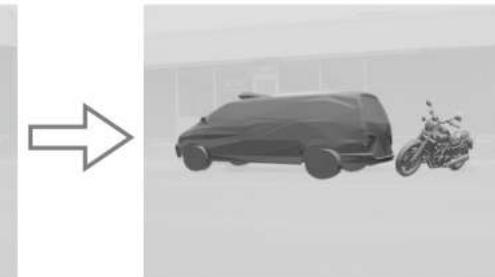
Training Examples (Novel Classes)



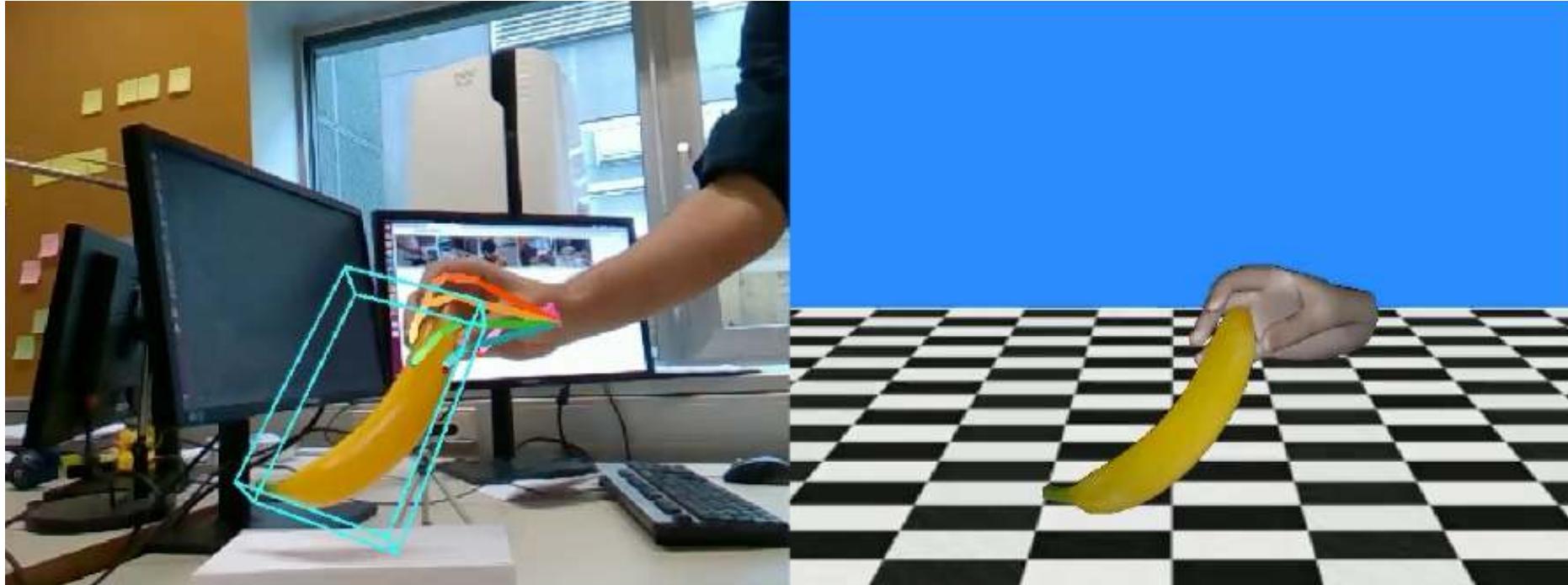
Optional



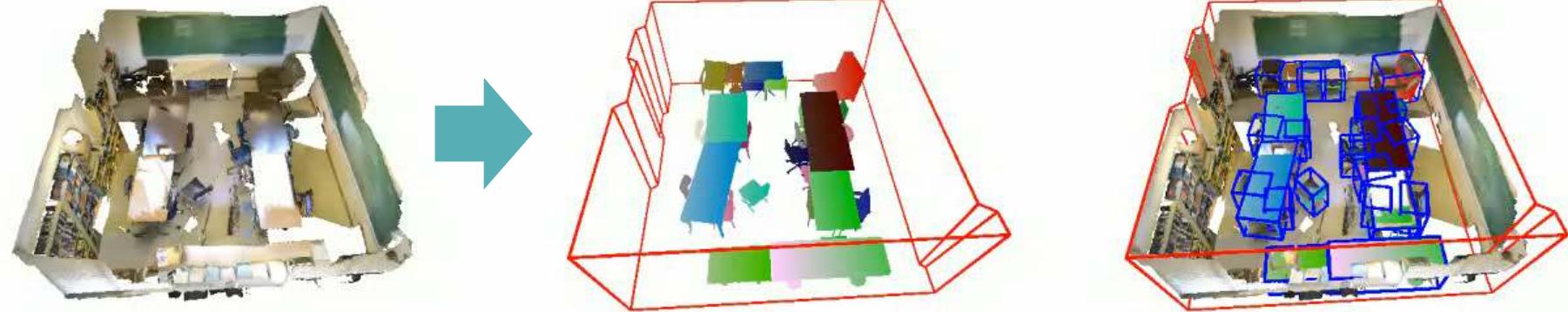
Testing



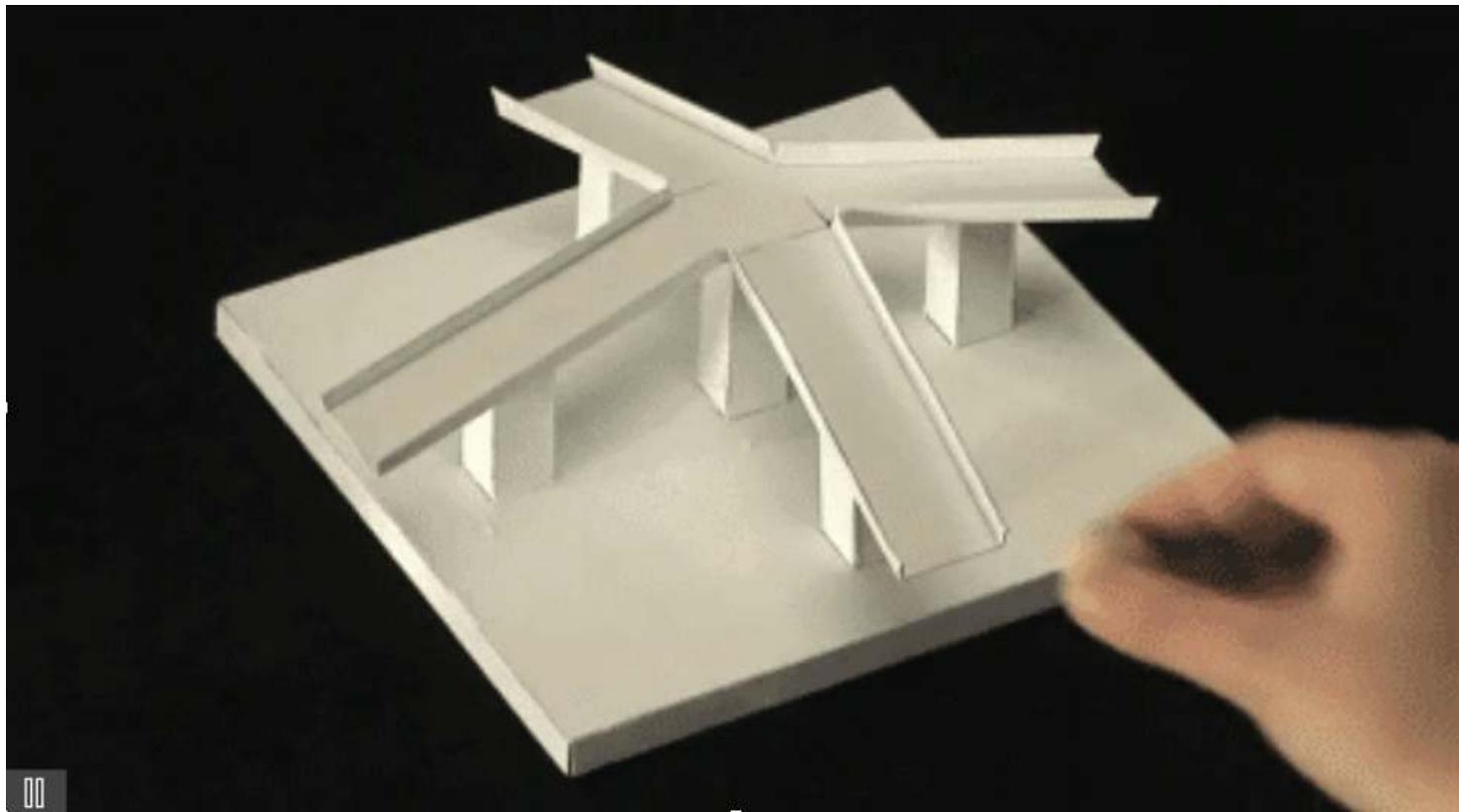
Applications



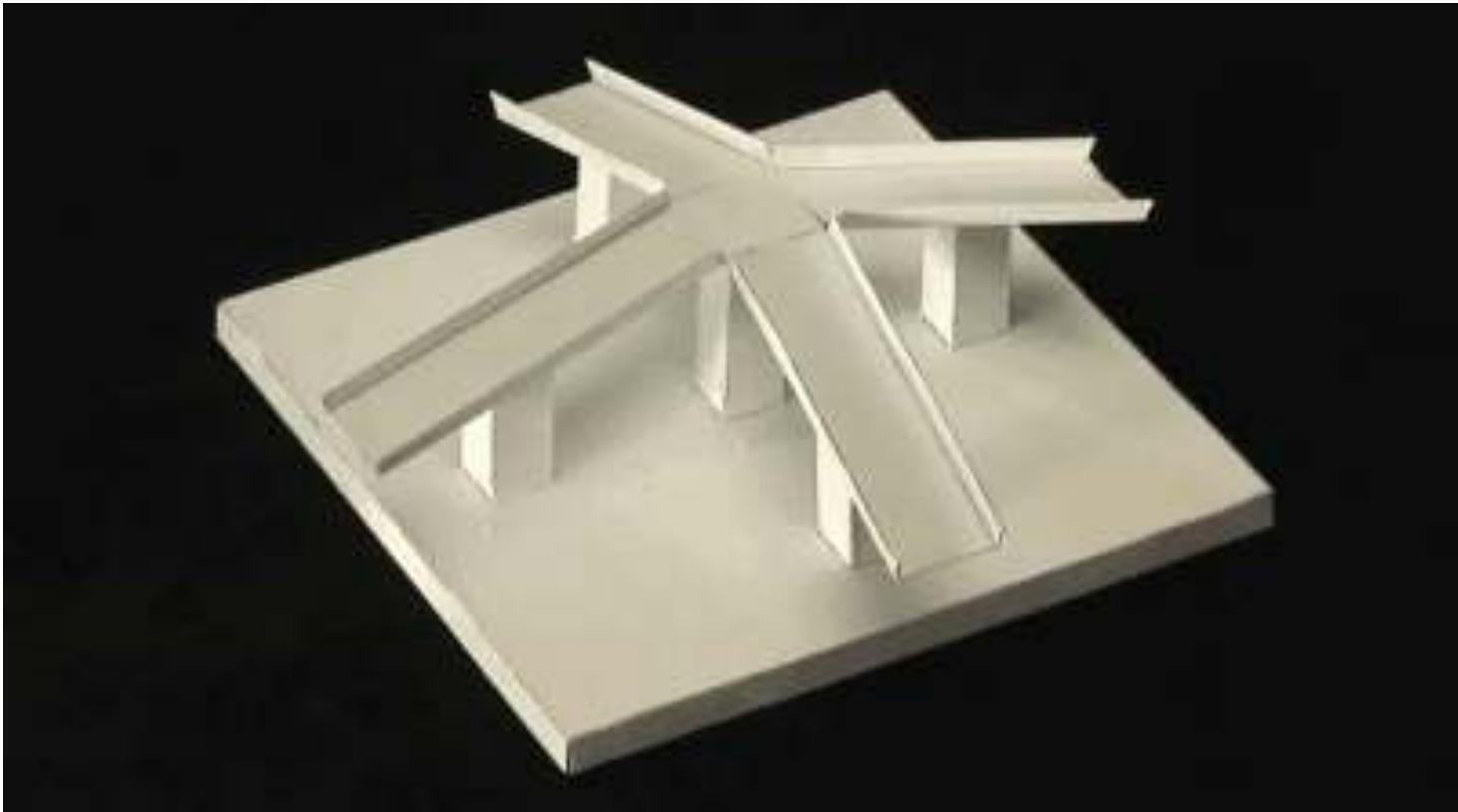
Some results



ambiguities in scene understanding



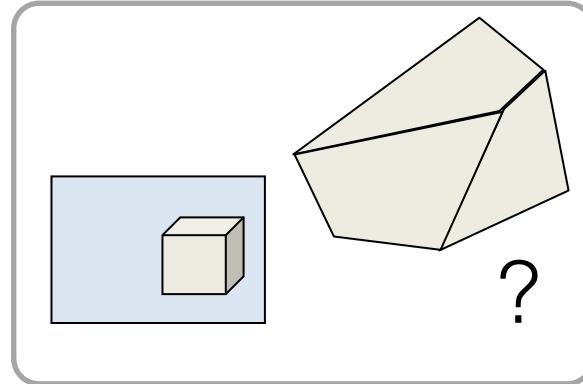
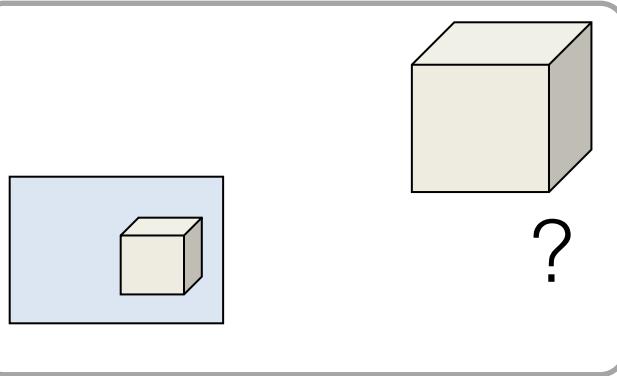
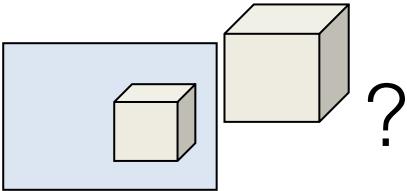
ambiguities in scene understanding



Why Is Vision So Hard?

Not enough information

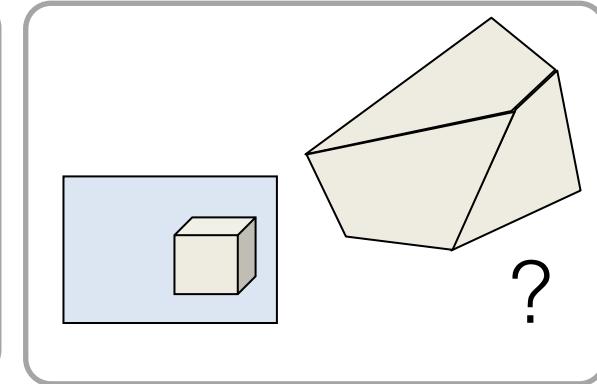
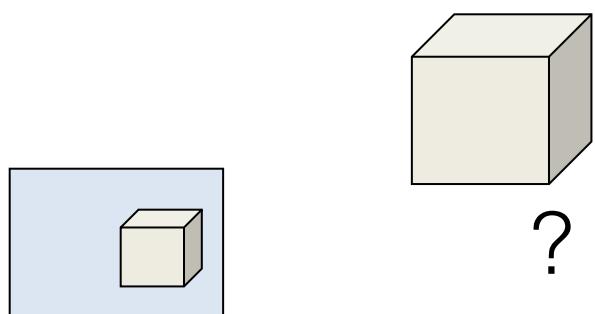
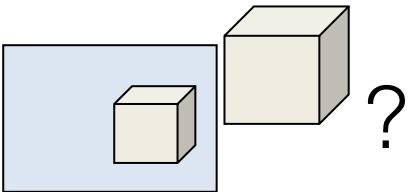
3D information is lost



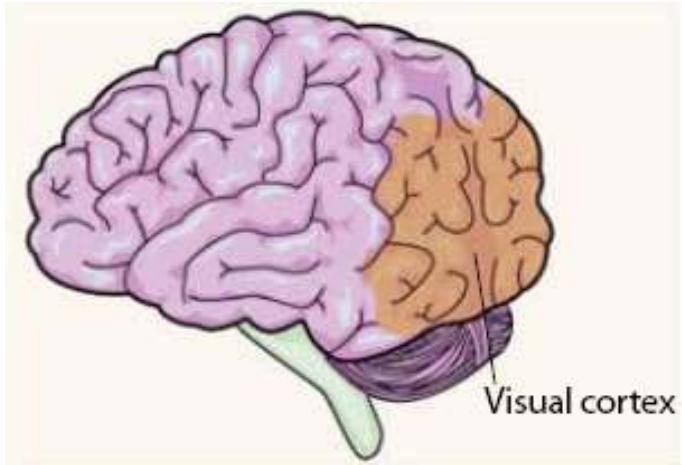
Given an image, there is an infinite family of 3D scenes that could create this image;
How do we choose which one is correct?

- Context
- Prior information (e.g. lines tend to be orthogonal or parallel);
- etc.

So how do we choose?



Human Vision



The visual cortex represents about 20-50% of our brain.

Human vision is unconscious (most of the time).

Intuitions about how human vision works are often wrong...

Overview

Part #1:

- 3D object instance understanding;
- 3D pose prediction and training;

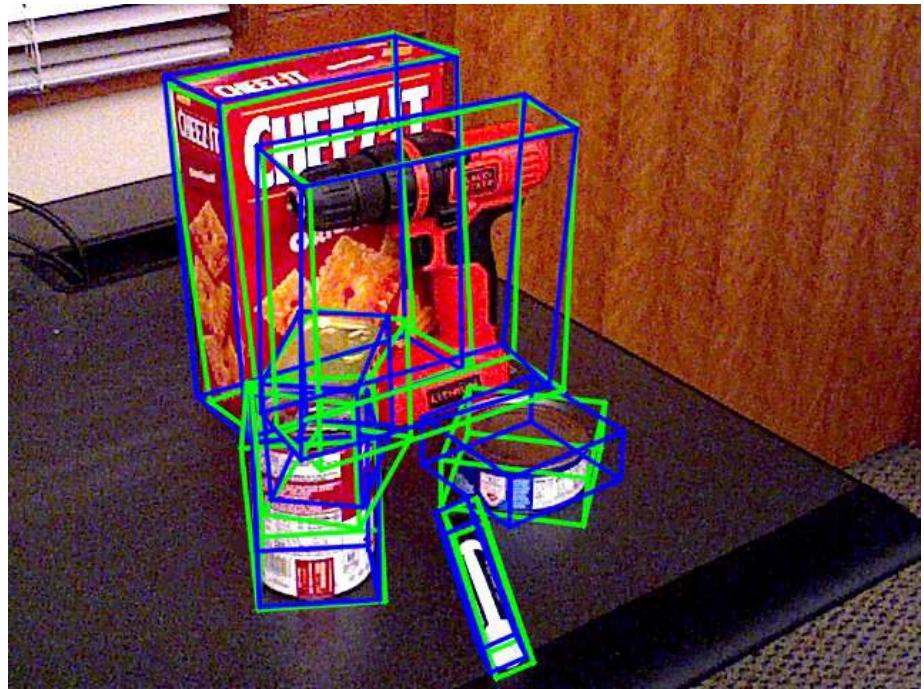
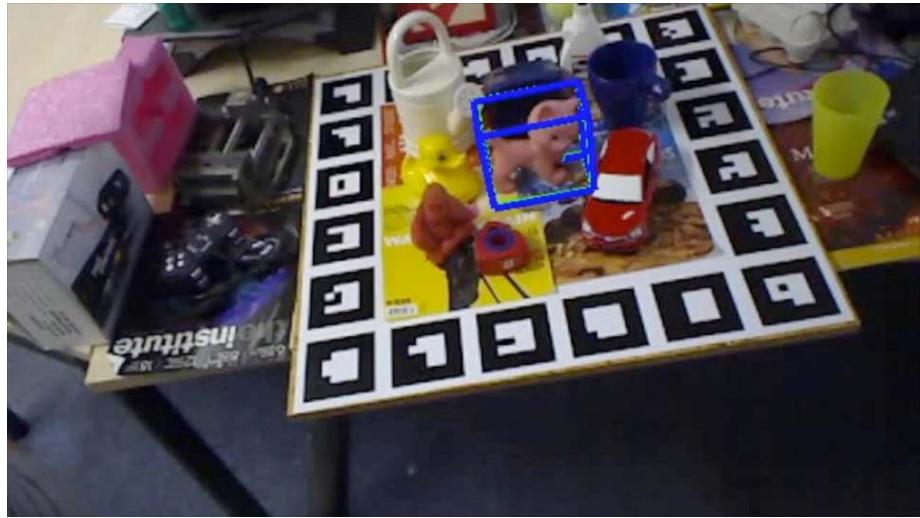
Part #2

- 3D object category understanding and 3D model prediction;
- 3D model prediction from images;

Part #3:

- Few-shot learning for 3D scene understanding;
- Point-cloud analysis;
- Self-learning for 3D scene understanding;
- Object discovery.

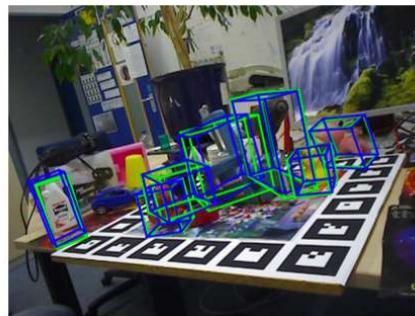
3D object instance understanding



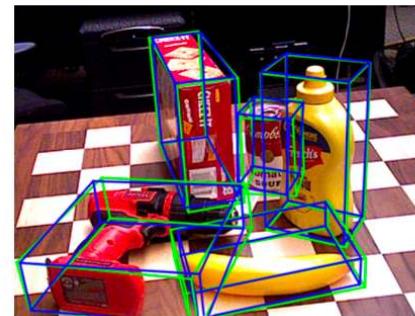
typical datasets



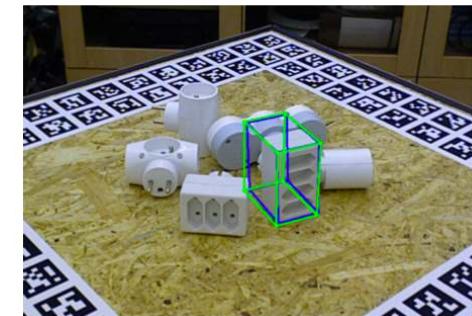
LineMOD



Occluded-LineMOD



YCB-Video



T-LESS

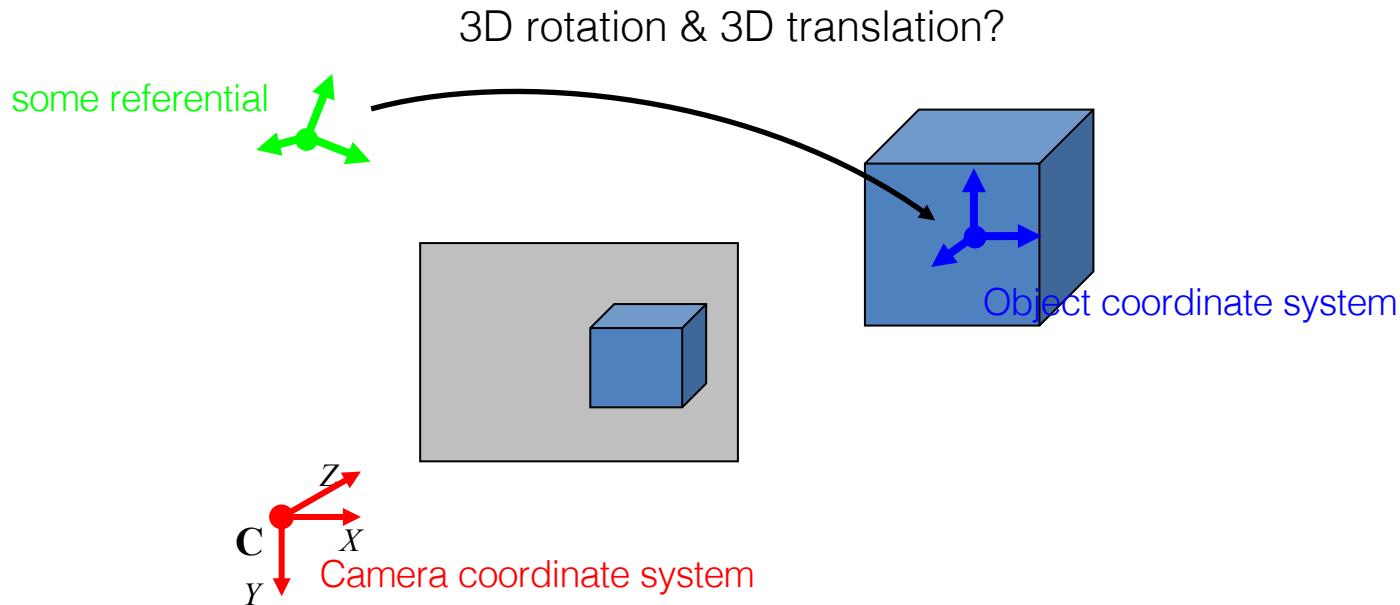
training set: About 200 annotated real Images



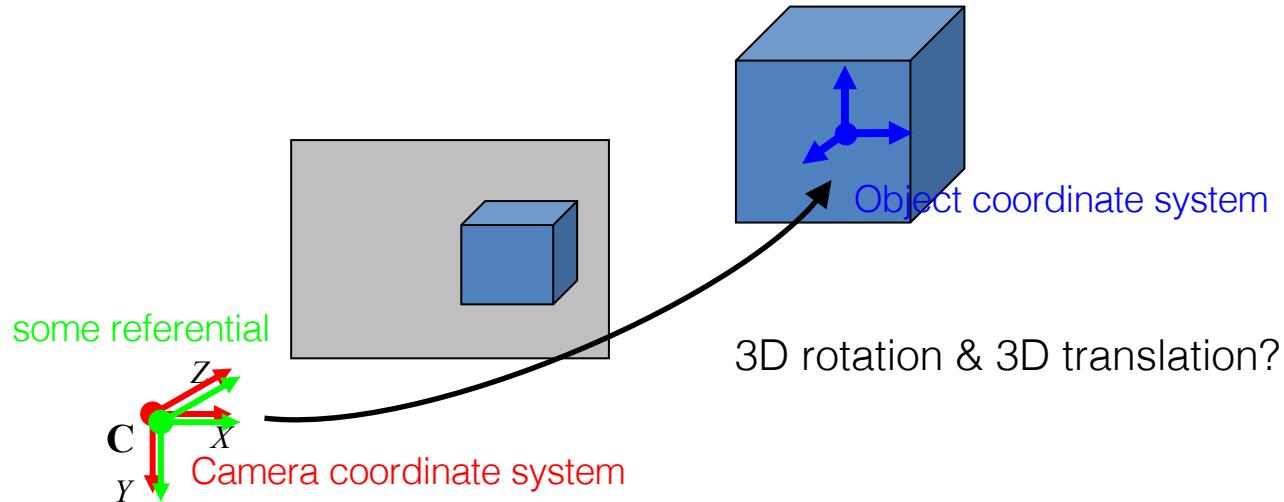
Overview

- 2D detection: Why and how;
- Pose prediction: How to represent an object's pose;
- Training data: real or synthetic?

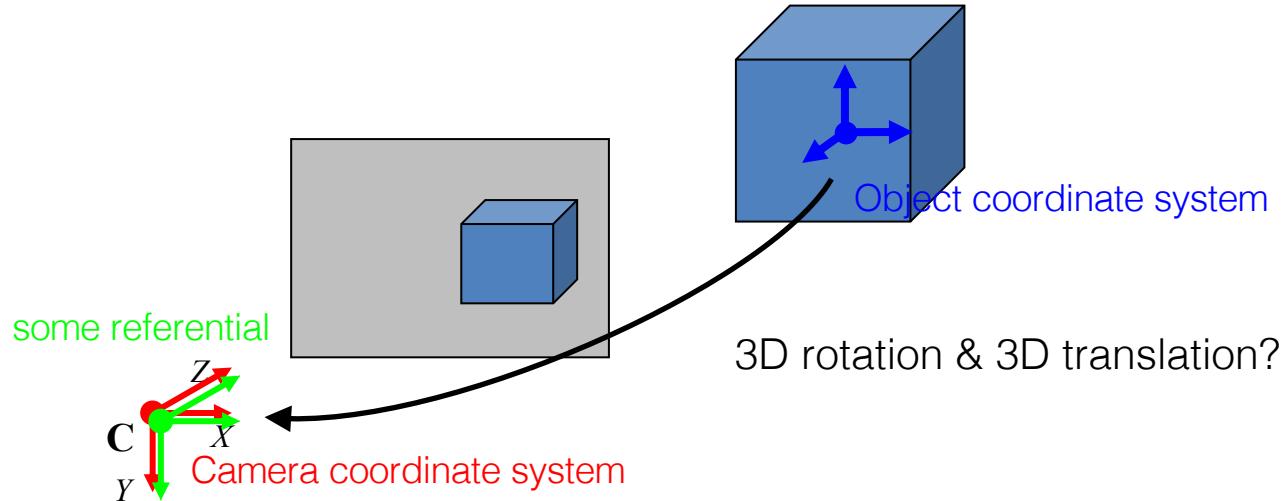
3D Pose / 6D Pose



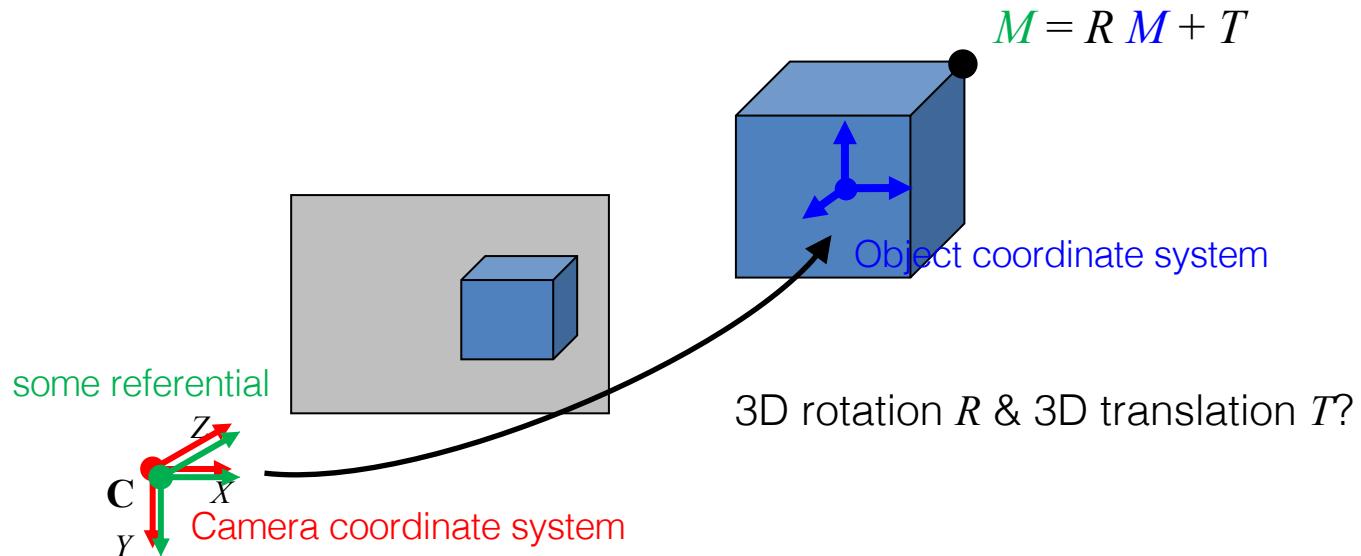
3D Pose / 6D Pose



3D Pose / 6D Pose



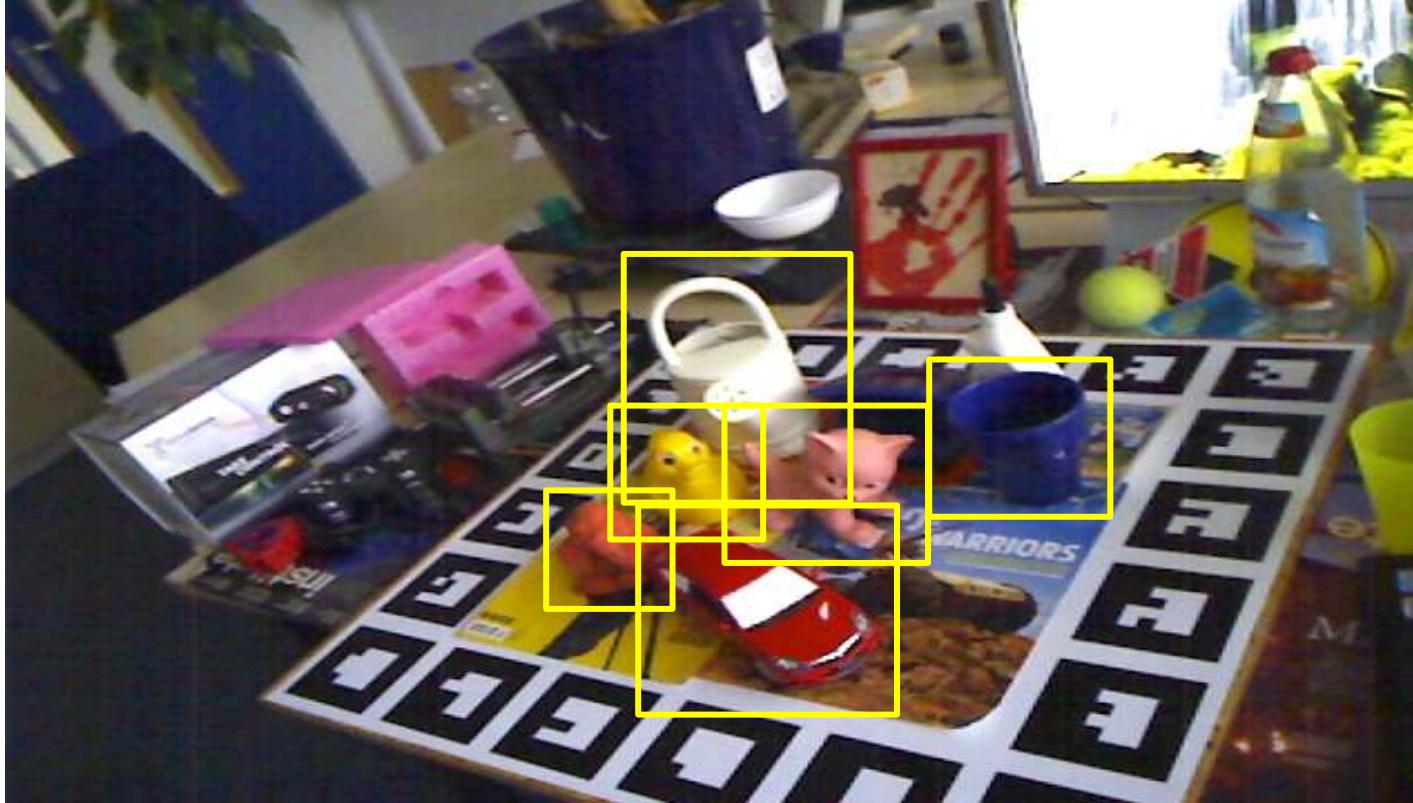
3D Pose / 6D Pose



2D Detection



2D Detection

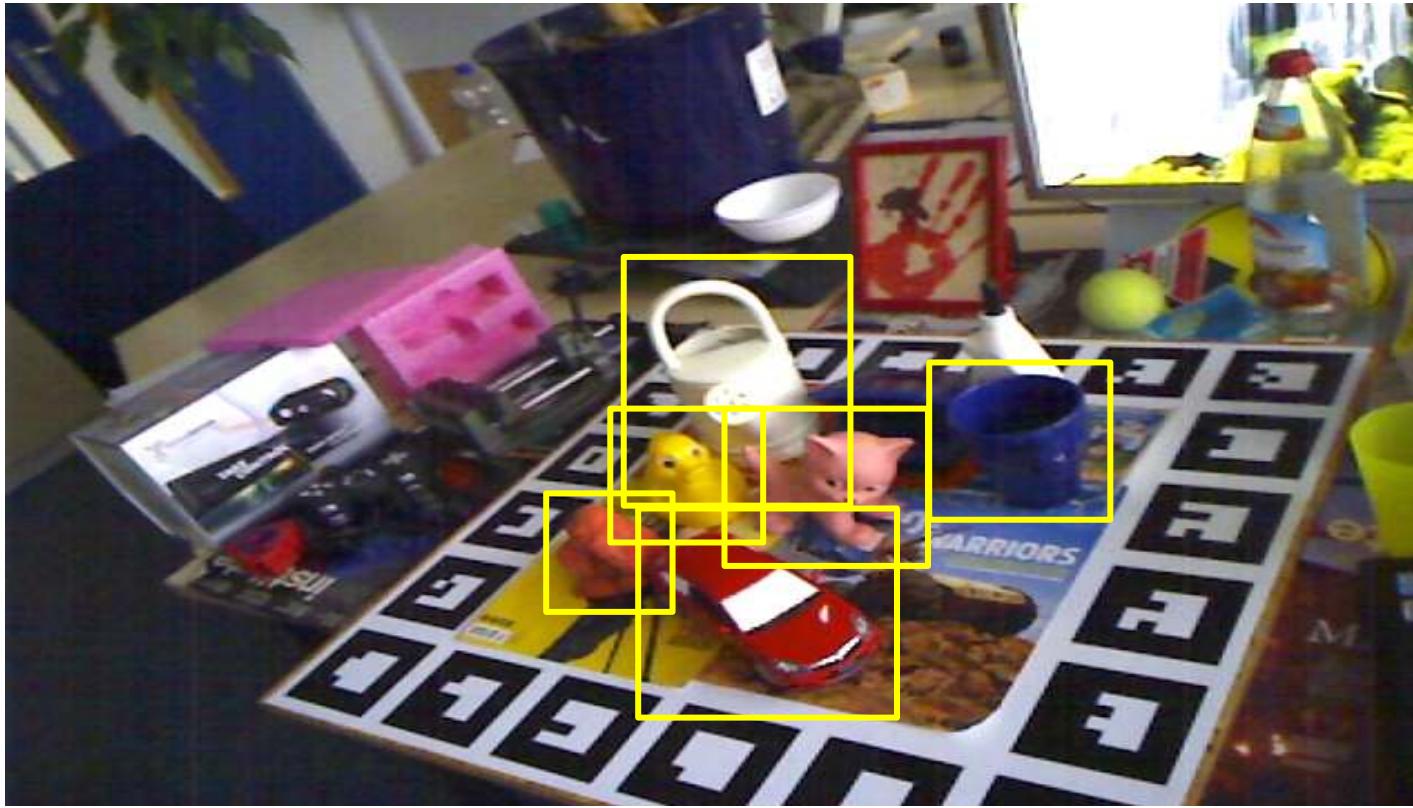


2D Detection



→ deep network → 3D pose

2D detection of multiple objects



→ common solution: Mask-RCNN / Detectron2

R-CNN (1)



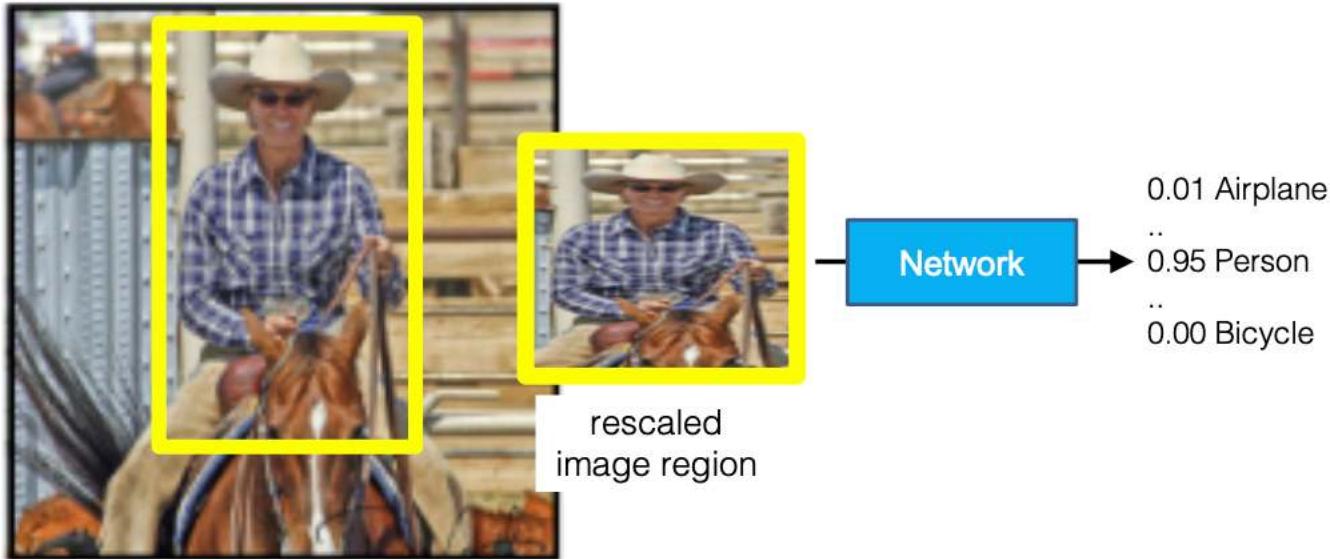
Input Image



Region Proposals
No learning (yet)

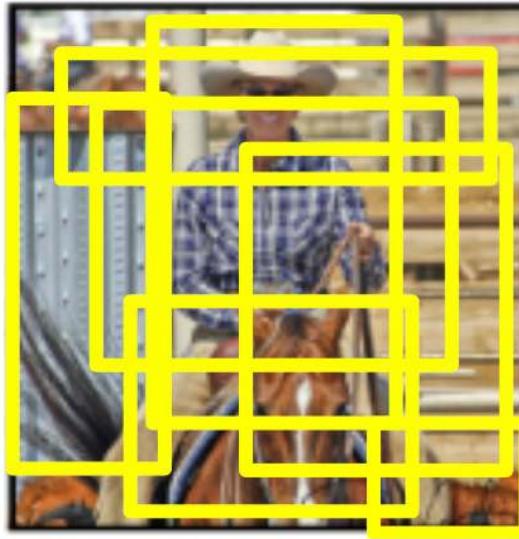
R.B. Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587.

R-CNN (2)



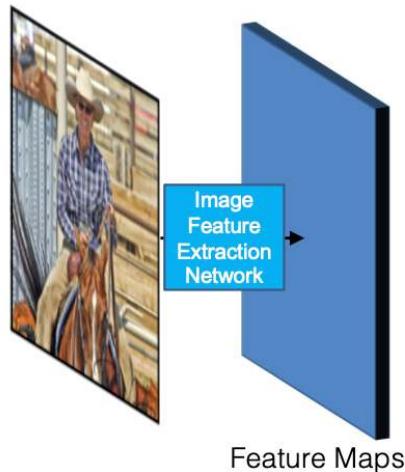
R-CNN (3)

Problem: In practice, many region proposals. R-CNN inefficient since image locations are processed many times.



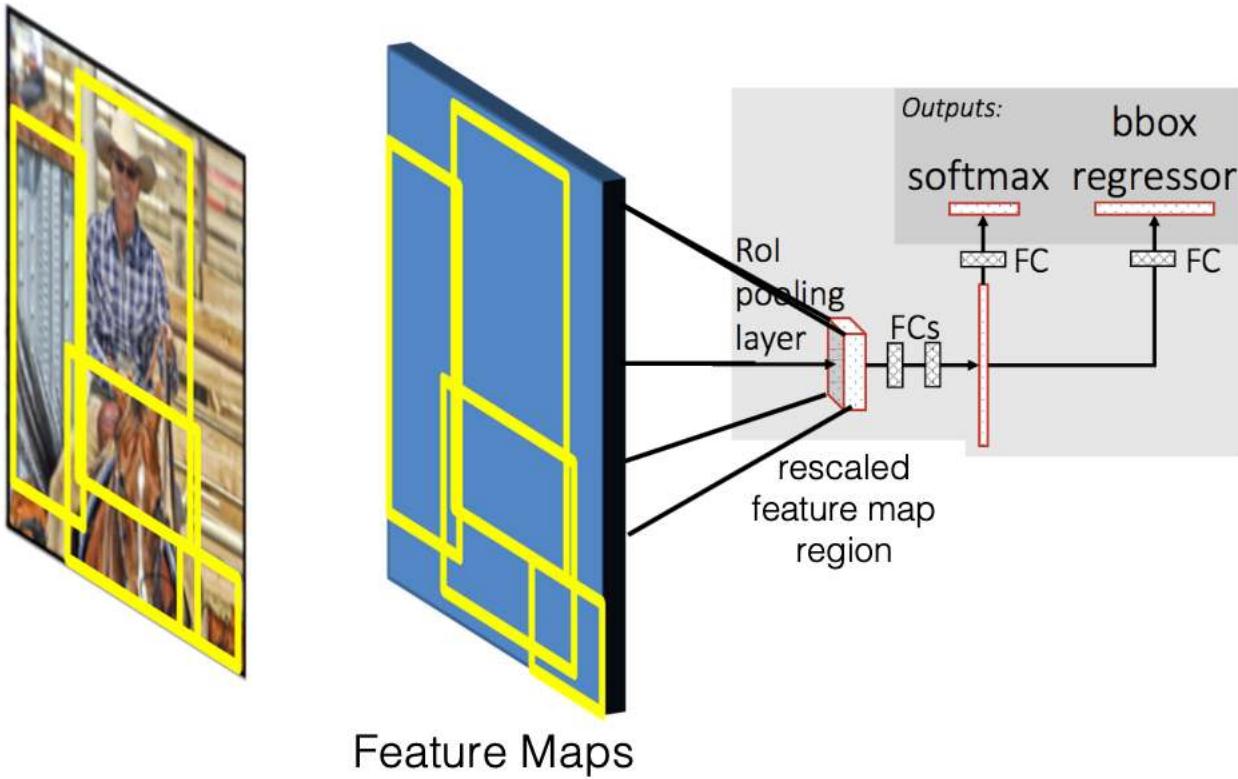
Fast-RCNN (1)

Convolutions are applied only once to the image to extract image features: Much faster.

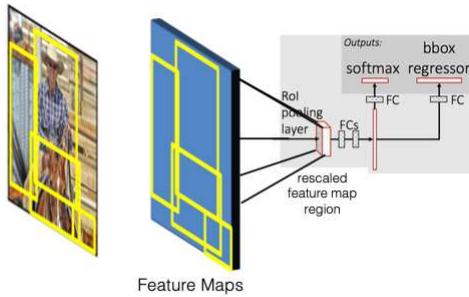


Ross B. Girshick. “Fast R-CNN”. In: *International Conference on Computer Vision*. 2015.

Fast-RCNN (2)



Fast R-CNN: Loss Function (1)



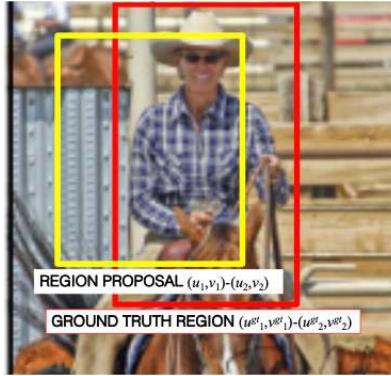
Loss function for 1 region:

$$\mathcal{L}(\Theta) = -\log p_c(f_{\text{cl}}(\mathbf{x}; \Theta)) + \lambda \mathbf{1}_{[c \geq 1]} \mathcal{L}_{\text{bbox}}, \quad (15)$$

where:

- ▶ \mathbf{x} is the rescaled region in the feature maps;
- ▶ c is the true class for \mathbf{x} . $c = 0$ corresponds to the background.
- ▶ $\mathcal{L}_{\text{bbox}}$ is a loss term to refine the region bounding box (see next slide);
- ▶ λ is a weight.

Fast R-CNN: Loss Function (2)



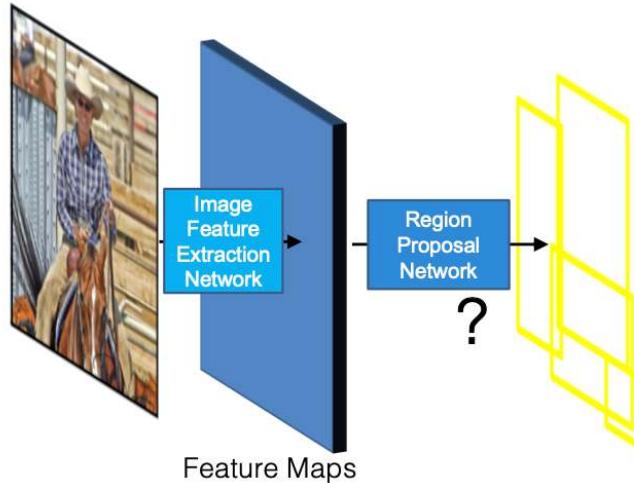
\mathcal{L}_{bbox} is a loss term to refine the region bounding box:

$$\mathcal{L}_{bbox} = (u_1 + f_{bbox}(\mathbf{x}; \Theta)[0] - u_1^{gt})^2 + (v_1 + f_{bbox}(\mathbf{x}; \Theta)[1] - v_1^{gt})^2 + ..$$

where $(u_1, v_1) \times (u_2, v_2)$ are the coordinates of the region bounding box, and $(u_1^{gt}, v_1^{gt}) \times (u_2^{gt}, v_2^{gt})$ are the coordinates of the region bounding box.

Faster R-CNN

Learns to predict the region proposals:



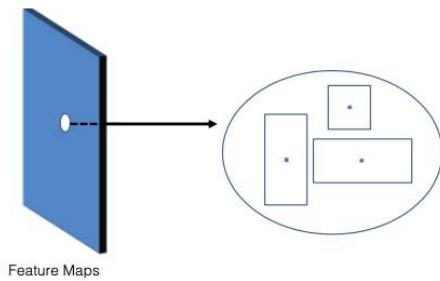
Challenges: the number of regions varies with the image, each region has a different size.

S. Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems*. 2015.

Faster R-CNN: Region Proposal Network

For each 2D location in the feature maps: consider 3 “anchor boxes” and predict for each anchor box:

- ▶ If the anchor box overlaps with an object;
- ▶ An offset to adapt the anchor box.

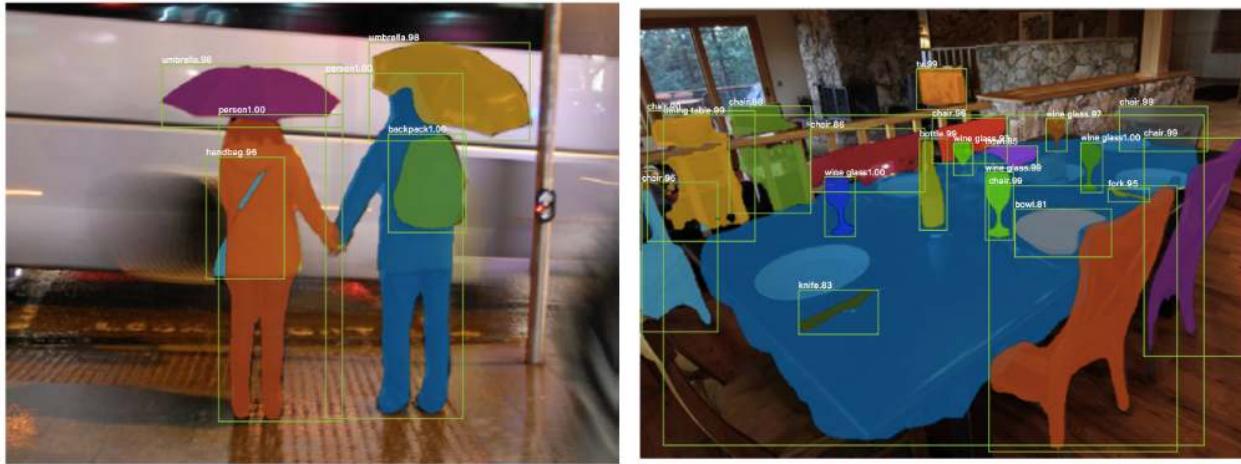


Loss function for 1 image \mathbf{x} with $\mathcal{B} = \{B_i\}_i$ ground truth bounding boxes:

$$\mathcal{L}(\Theta_2) = - \sum_{A \in \mathcal{A}} \log p_{c(A, \mathcal{B})}(g(\mathbf{x}; \Theta_2)[A]) + \lambda c(A, \mathcal{B}) \mathcal{L}_{bbox},$$

with $c(A, \mathcal{B}) = 1$ if Anchor box A overlaps with at least one bounding box B_i in \mathcal{B} , 0 otherwise.

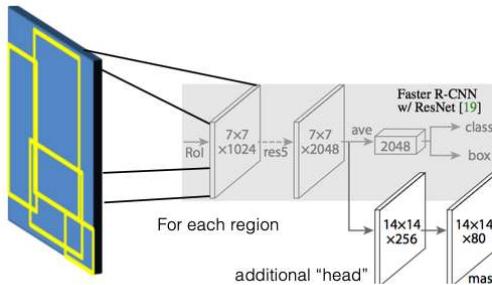
Mask-RCNN



Kaiming He et al. “Mask R-CNN”. In: *International Conference on Computer Vision*. 2017.

Mask-RCNN (2)

In addition to predicting the class and the “delta bounding box”, predict a binary mask for each possible class.



Loss function for one region:

$$\mathcal{L}(\Theta) = -\log p_c(f_{\text{cl}}(\mathbf{x}; \Theta)) + \lambda \mathbf{1}_{[c \geq 1]} (\mathcal{L}_{\text{bbox}} + \lambda_2 \mathcal{L}_{\text{mask}}),$$

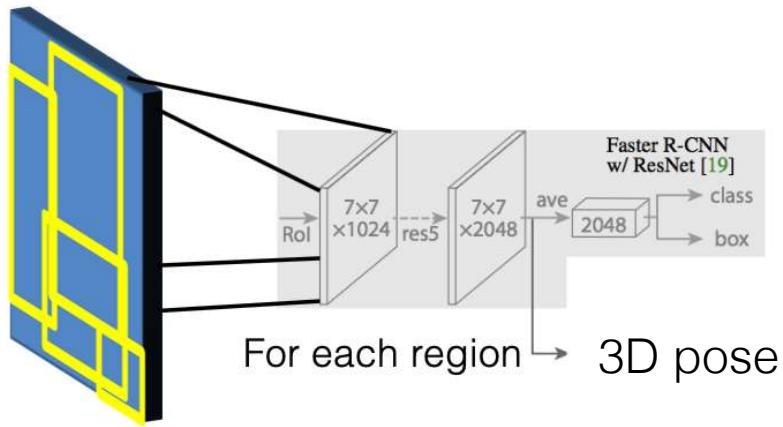
where:

- ▶ c is the true class for \mathbf{x} . $c = 0$ corresponds to the background.
- ▶ $\mathcal{L}_{\text{mask}}$ is a loss term to refine the region bounding box:

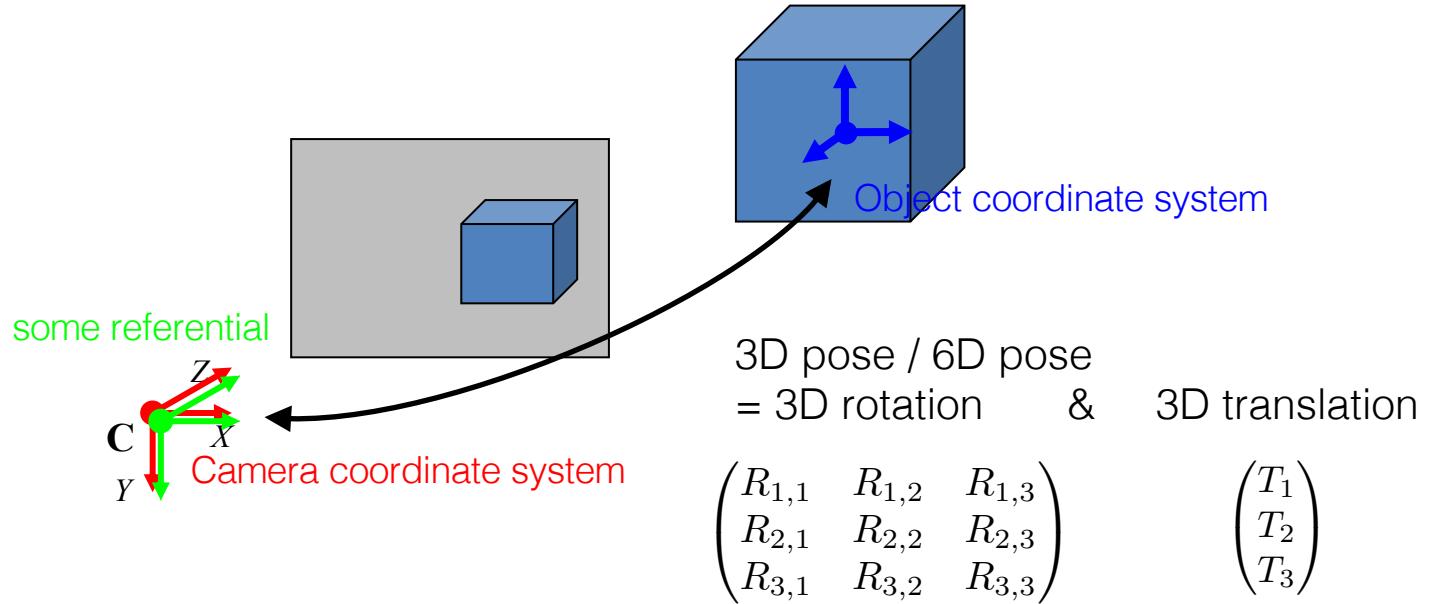
$$\mathcal{L}_{\text{mask}} = \|f_{\text{mask}}(\mathbf{x}; \Theta)[c] - m\|^2$$

- ▶ m is the ground truth mask for the region.

Mask-RCNN: Extension to 3D



3D Pose / 6D Pose



pose prediction from 2D Detection



→ deep network →

$$\begin{pmatrix} R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{pmatrix}$$

$$\begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix}$$

or another representation for the rotation matrix (see next slides)

training set: About 200 annotated real Images



$$I_0 \hat{R}_0 \hat{T}_0$$



$$I_1 \hat{R}_1 \hat{T}_1$$



$$I_2 \hat{R}_2 \hat{T}_2$$



$$I_3 \hat{R}_3 \hat{T}_3$$

loss for pose prediction

For the 3D translation, simply the Euclidean distance between prediction and ground truth:

$$\mathcal{L}_T = \|T - \hat{T}\|^2$$

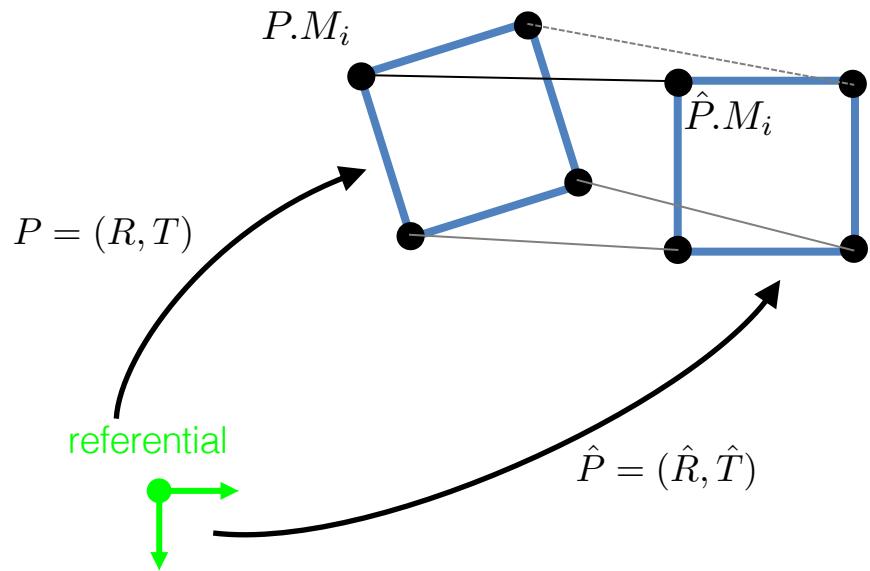
For the 3D rotation, geodesic distance:

$$\begin{aligned}\mathcal{L}_R &= \|\log(R\hat{R}^\top)\|_F \\ &= \cos^{-1}(tr(R\hat{R}^\top) - 1)/2\end{aligned}$$

For the full 3D pose:

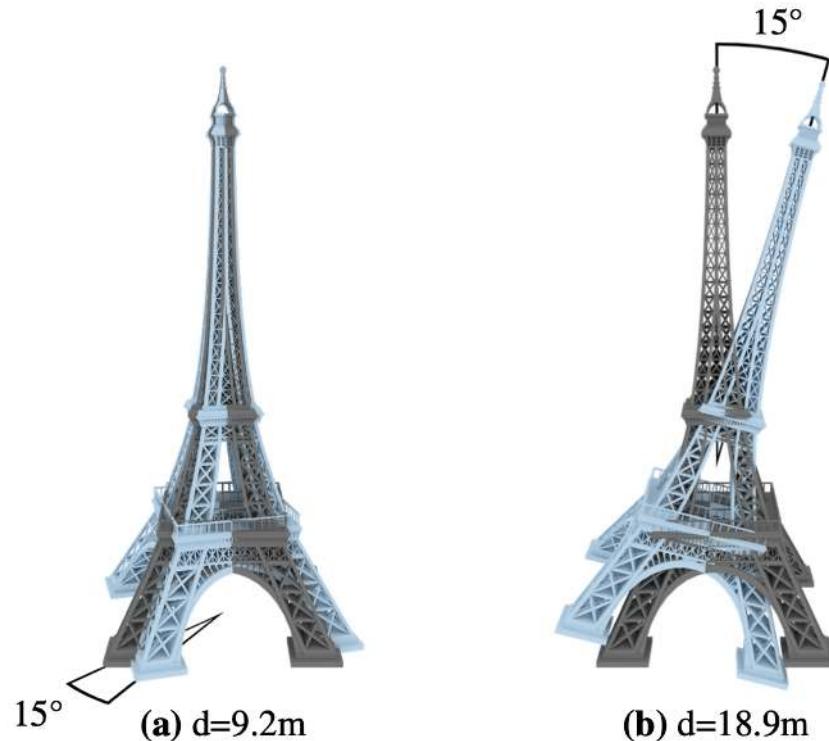
$$\mathcal{L}_{\text{pose}} = \mathcal{L}_T + \gamma \mathcal{L}_R$$

alternative loss for pose prediction



$$\left\{ \begin{array}{l} \mathcal{L} = \sum_i \|P.M_i - \hat{P}.M_i\|^2 \\ P.M_i = RM_i + T \\ \hat{P}.M_i = \hat{R}M_i + \hat{T} \end{array} \right.$$

alternative loss for pose prediction



R. Brégier et al. Defining the Pose of Any 3D Rigid Object and an Associated Distance. IJCV, June 2018.

parameterizing the rotation matrix

properties of the rotation matrix

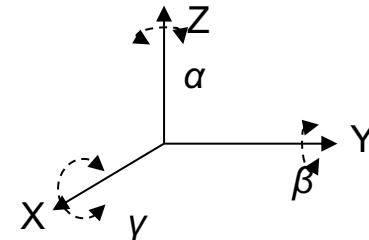
$$\left\{ \begin{array}{l} RR^\top = I \\ \det R = 1 \end{array} \right.$$

possible parameterizations of the rotation matrix

- Directly the rotation matrix (ie 9 values);

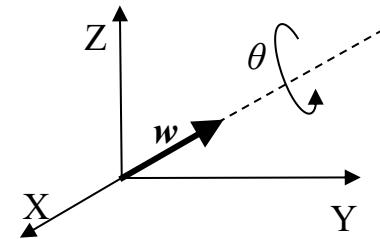
- Euler angles (3 values):

$$\mathbf{R} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$$

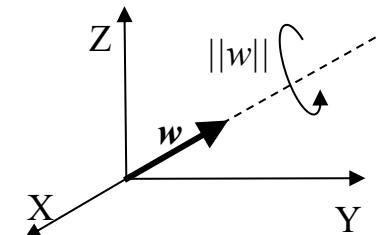


- A unit quaternion (4 values):

$$q = \left(\cos \frac{\theta}{2}, \mathbf{w} \sin \frac{\theta}{2} \right)$$



- An Exponential Map (a unit 3-vector, 3 values):



...

A Unit Quaternion

Quaternions are hyper-complex numbers that can be written as the linear combination $a+bi+cj+dk$, with $i^2=j^2=k^2=ijk=-1$.

Can also be interpreted as a scalar plus a 3- vector: (a, \mathbf{v}) .

A rotation about the unit vector \mathbf{w} by an angle θ can be represented by the *unit quaternion*:

$$q = \left(\cos \frac{\theta}{2}, \mathbf{w} \sin \frac{\theta}{2} \right)$$

To rotate a 3D point \mathbf{M} : write it as a quaternion $p = (0, \mathbf{M})$, and take the rotated point p' to be

$$p' = q \cdot p \bar{q} \text{ with } \bar{q} = \left(\cos \frac{\theta}{2}, -\mathbf{w} \sin \frac{\theta}{2} \right)$$

No gimbal lock.

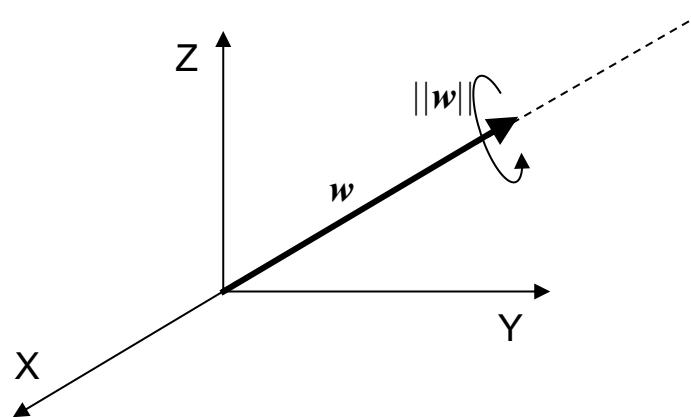
Parameterization of the rotation using the 4 coordinates of a quaternion q :

1. No constraint and rotation performed using $\frac{q}{\|q\|} \rightarrow$ singularity: kq yields the same rotation whatever the value of $k > 0$;
2. Additional constraint: norm of q must be constrained to be equal to 1, for example by adding the quadratic term $K(1 - \|q\|^2)^2$.

Exponential Maps

Parameterization by a 3D vector $\mathbf{w} = [w_1, w_2, w_3]^T$:

Rotation around the axis of direction \mathbf{w} of an amount of $||\mathbf{w}||$



Rodrigues' Formula

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}$$

The rotation matrix is given by:

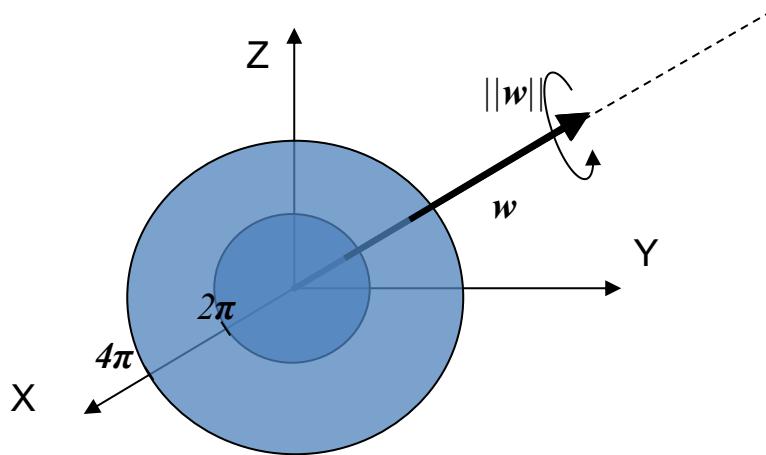
$$\begin{aligned}\mathbf{R}(\boldsymbol{\Omega}) &= \exp(\boldsymbol{\Omega}) = \mathbf{I} + \boldsymbol{\Omega} + \frac{1}{2!} \boldsymbol{\Omega}^2 + \frac{1}{3!} \boldsymbol{\Omega}^3 + \dots \\ &= \mathbf{I} + \frac{\sin \theta}{\theta} \boldsymbol{\Omega} + \frac{(1 - \cos \theta)}{\theta^2} \boldsymbol{\Omega}^2 \quad (\text{Rodrigues' formula})\end{aligned}$$

Not singular for small values of θ even if we divide by θ (see Taylor expansions).

The Singularities of Exponential Maps

Rotation around the axis of direction w of an amount of $\|w\|$

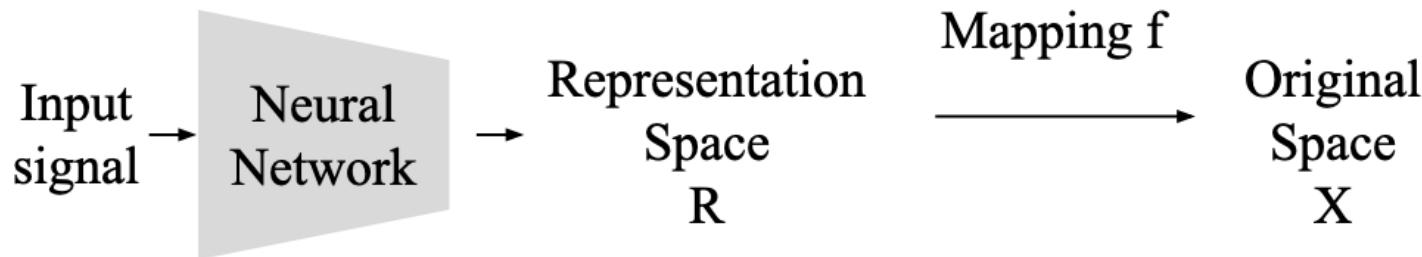
→ Singularities for w such that $\|w\| = 2n\pi$: No rotation, whatever the direction of w .



Avoided during optimization as follows:

when $\|w\|$ becomes close to $2n\pi$, say higher than π , w can be replaced by $\left(1 - \frac{2\pi}{\|w\|}\right)w$
[From Grassia JGT98]

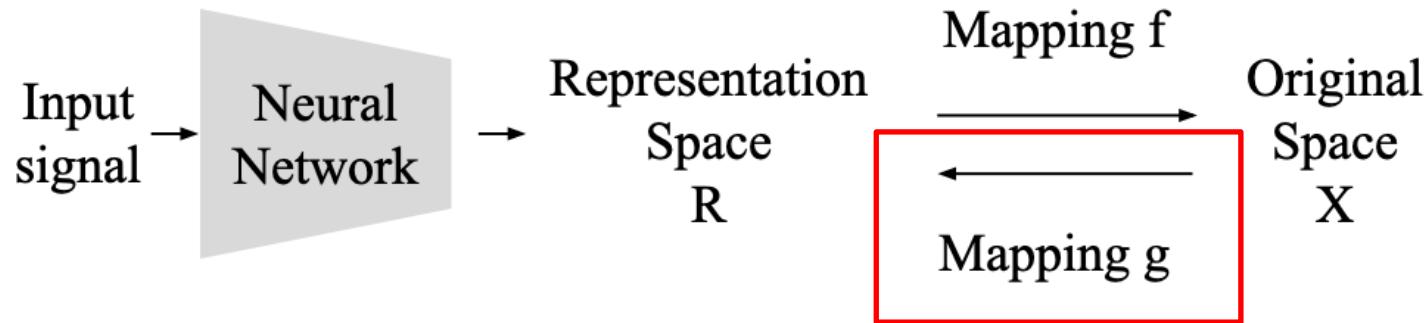
the problem with these representations



for example,
Euler angles

the 3x3 rotation
matrix itself

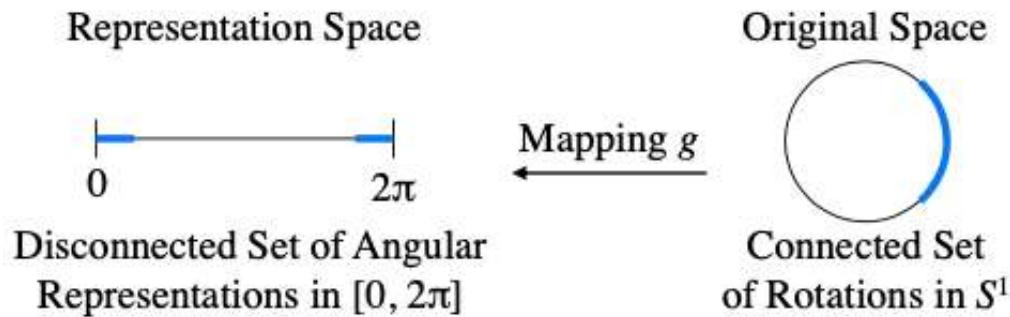
the problem with these representations



Needed for training the network
(in back-propagation).

*Not continuous for these
rotation representations*

discontinuities of g



proposed solution

- 2 3-vectors (6 values): e_1, e_2

$$e'_1 = \frac{e_1}{\|e_1\|_2}$$

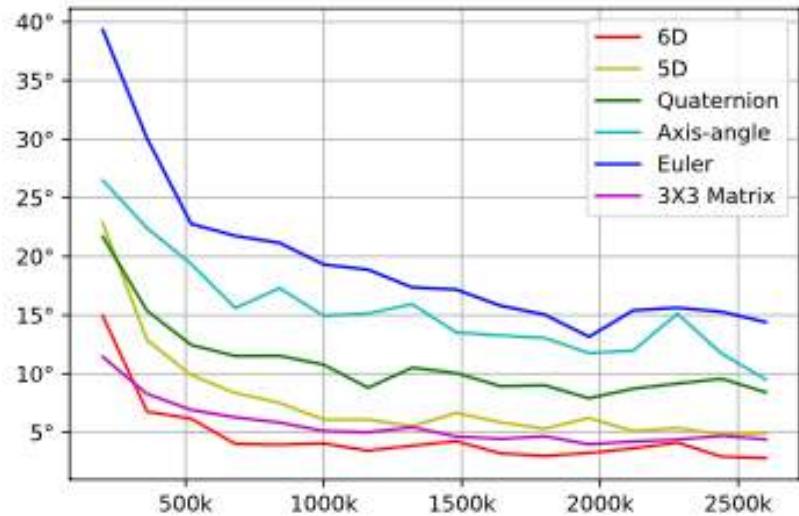
$$e'_3 = \frac{e'_1 \wedge e_2}{\|e_2\|_2} \quad R = (e'_1 \quad e'_2 \quad e'_3)$$

$$e'_2 = e'_3 \wedge e'_1,$$

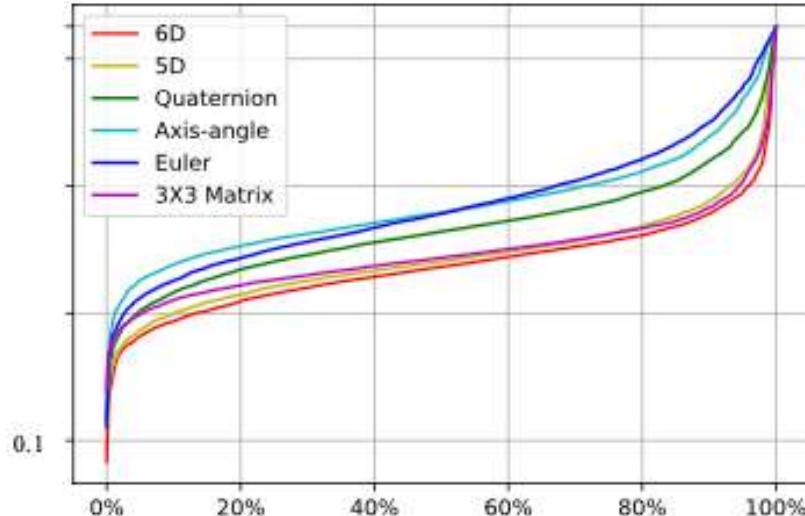
It is then possible to define a $g(R) = (e_1, e_2)$ function that is continuous.

results

3D Point Cloud Pose Estimation Test



d. Mean errors during iterations.



e. Percentile of errors at 2600k iteration.

alternative predictions (1)

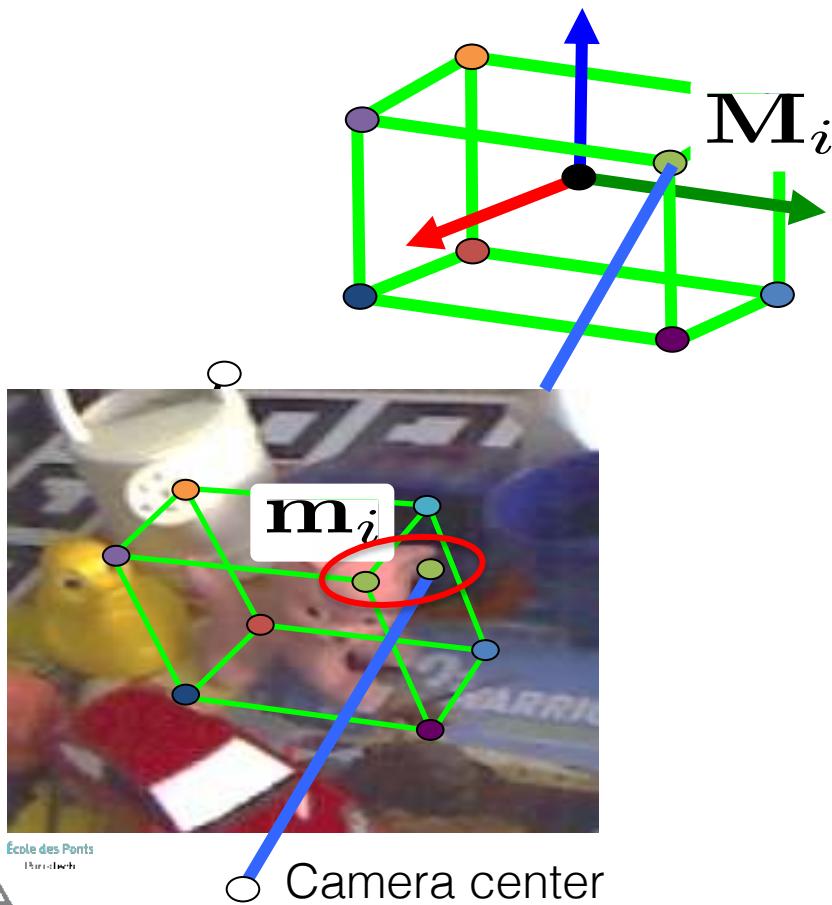


the 2D projections of the 8 corners of the 3D bounding box

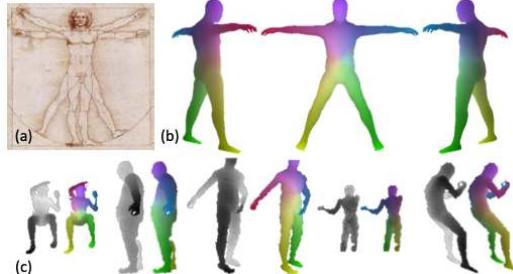


BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth. Mahdi Rad and Vincent Lepetit. ICCV 2017.

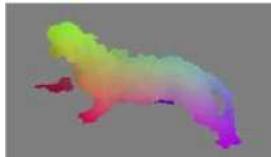
3D pose estimation from correspondences



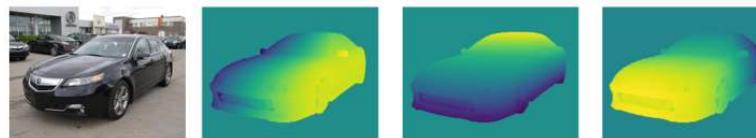
Alternative predictions (2)



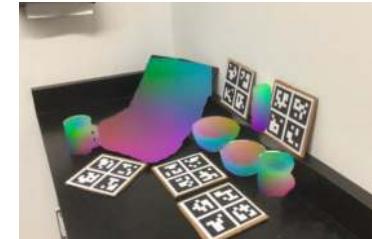
Taylor et al. The Vitruvian Manifold: Inferring Dense Correspondences for One-Shot Human Pose Estimation. CVPR 2012.



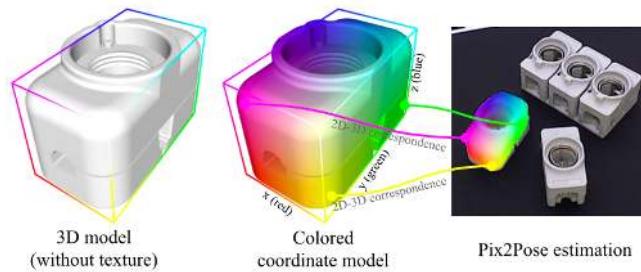
E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother. Learning 6D Object Pose Estimation using 3D Object Coordinates. ECCV 2014.



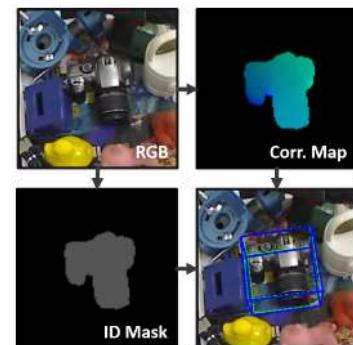
location fields. Wang et al., ECCV 2018



Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. Wang et al., CVPR 2019.

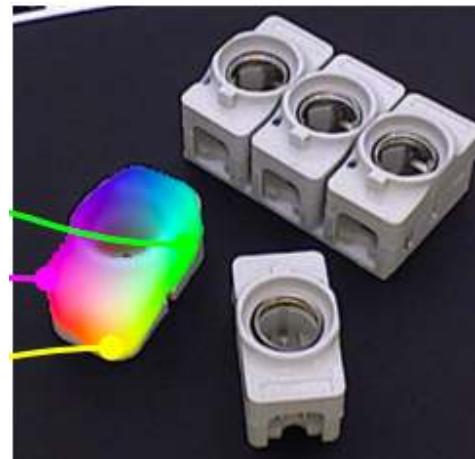
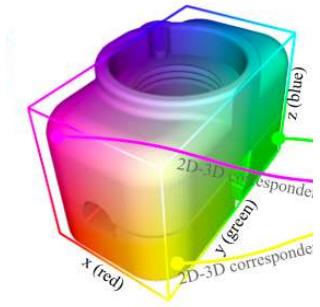


Pix2Pose: Pixel-Wise Coordinate Regression of Objects for 6D Pose Estimation. Park et al., CVPR 2019.

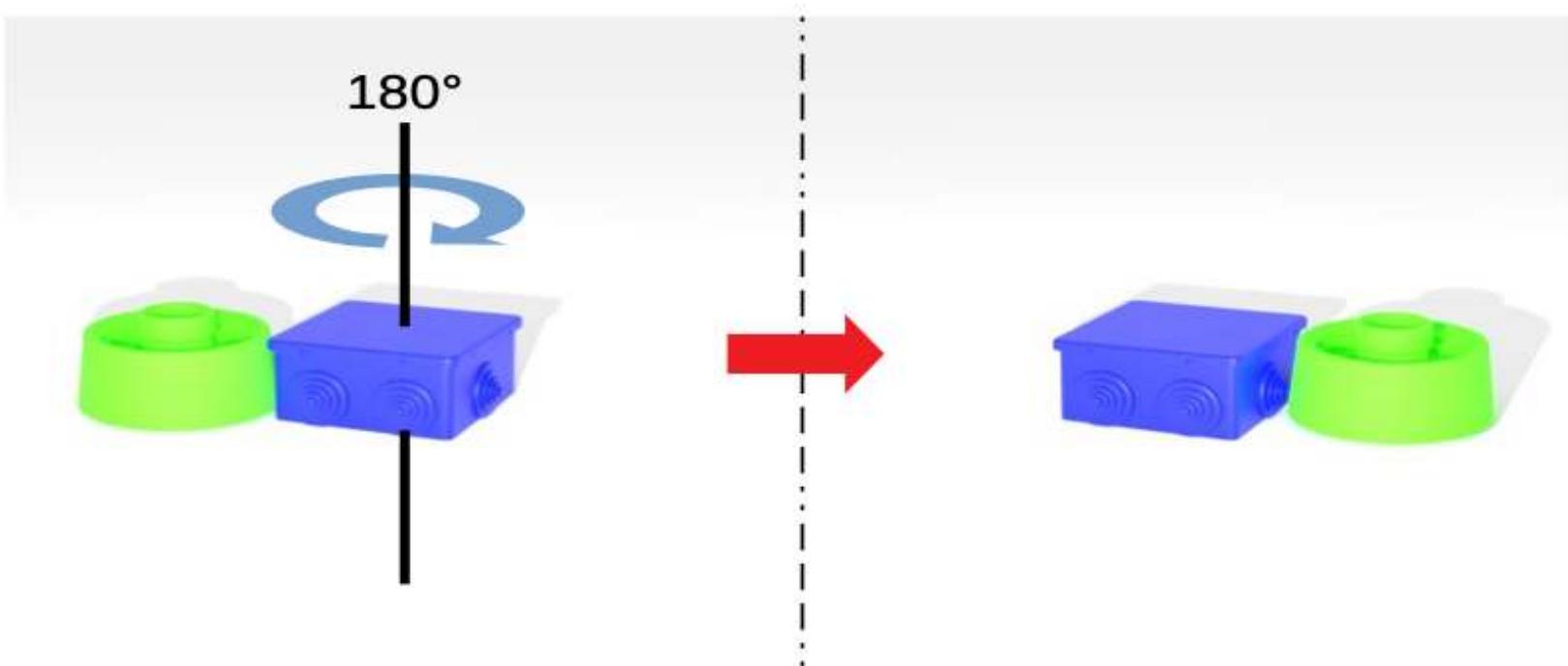


DPOD: 6D Pose Object Detector and Refiner. Zakharov et al. ICCV 2019.

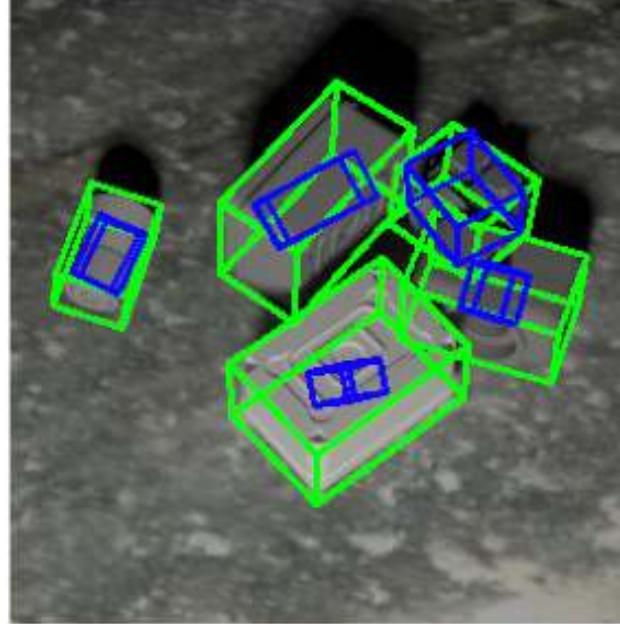
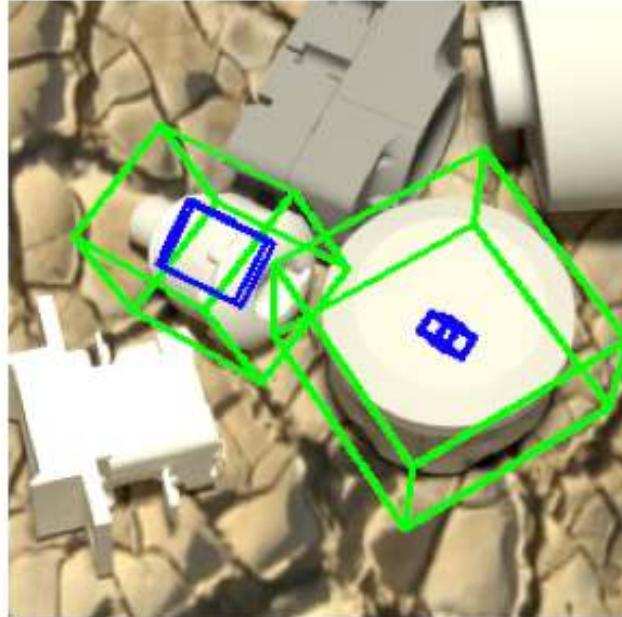
how to use 3D coordinate maps



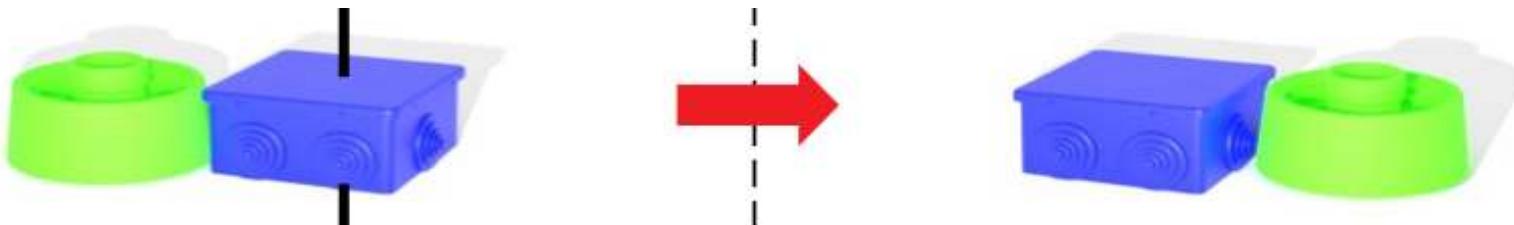
symmetrical objects



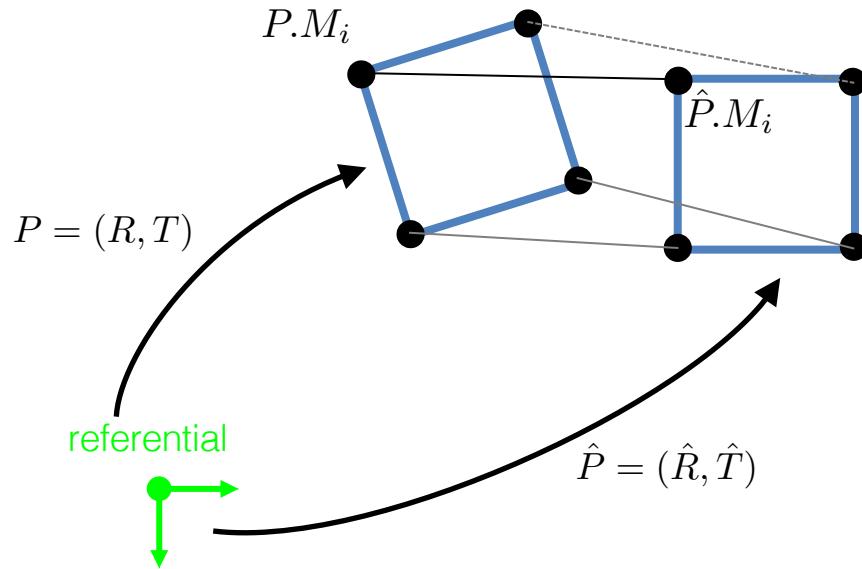
problem with symmetrical objects



why symmetrical objects can be problematic

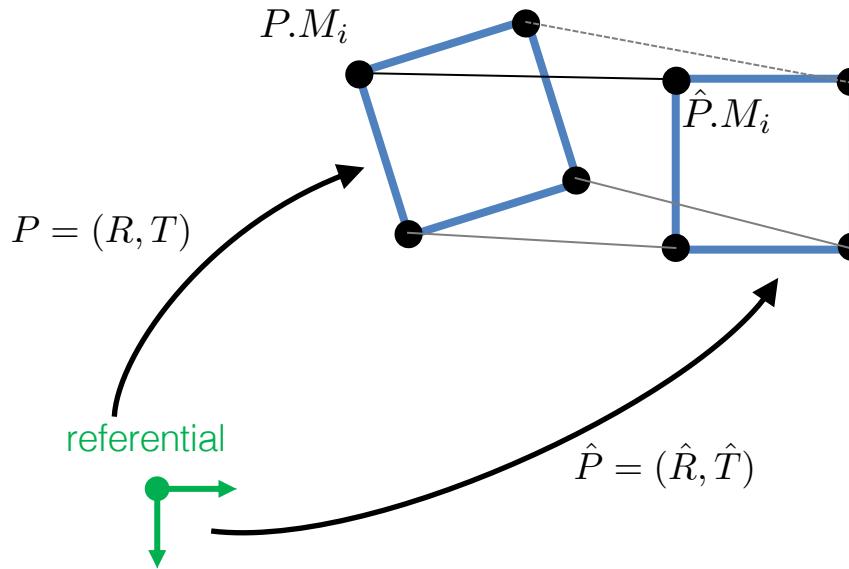


reminder: alternative loss



$$\left\{ \begin{array}{l} \mathcal{L} = \sum_i \|P.M_i - \hat{P}.M_i\|^2 \\ P.M_i = RM_i + T \\ \hat{P}.M_i = \hat{R}M_i + \hat{T} \end{array} \right.$$

for symmetrical objects



replace:

$$\mathcal{L} = \sum_i \|P.M_i - \hat{P}.M_i\|^2$$

with:

$$\mathcal{L} = \min_{S \in \mathcal{S}} \sum_i \|P.S.M_i - \hat{P}.M_i\|^2$$

\mathcal{S} set of symmetries

Training Set: About 200 Real Images + ...

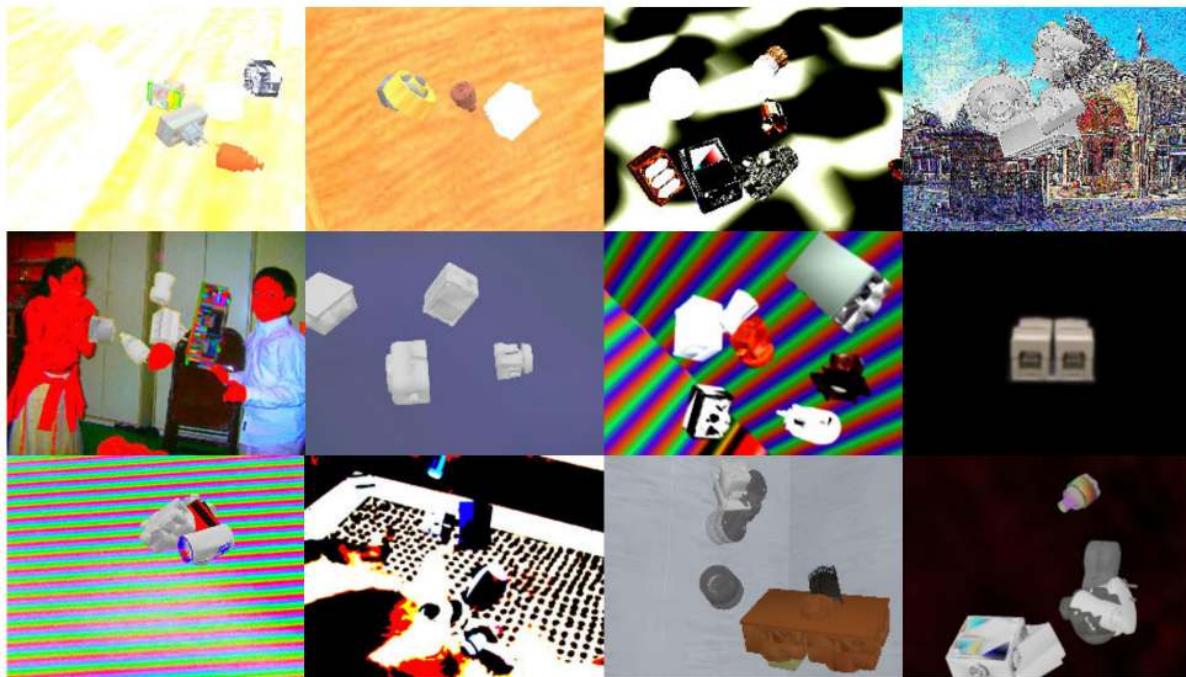


... Data Augmentation (1)



Dwibedi et al. Cut, paste and learn: Surprisingly easy synthesis for instance detection.
ICCV 2017.

Data Augmentation and Domain Randomization



Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. Tobin et al. IROS 2017.

CosyPose: Consistent multi-view multi-object 6D pose estimation. Labb   et al. ECCV 2020.

How Domain Randomization Works

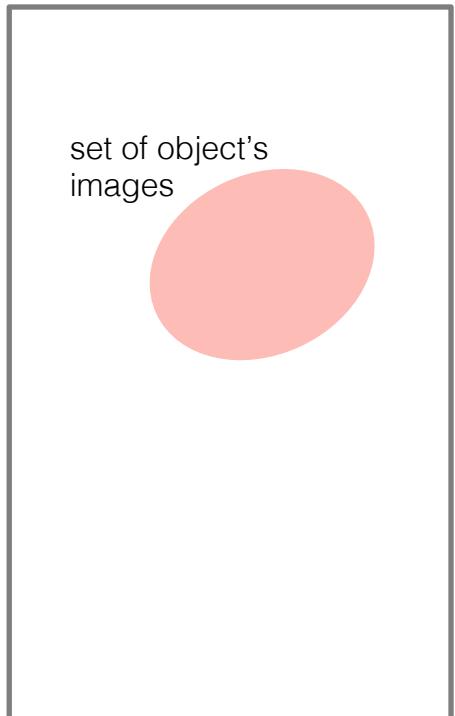


image space

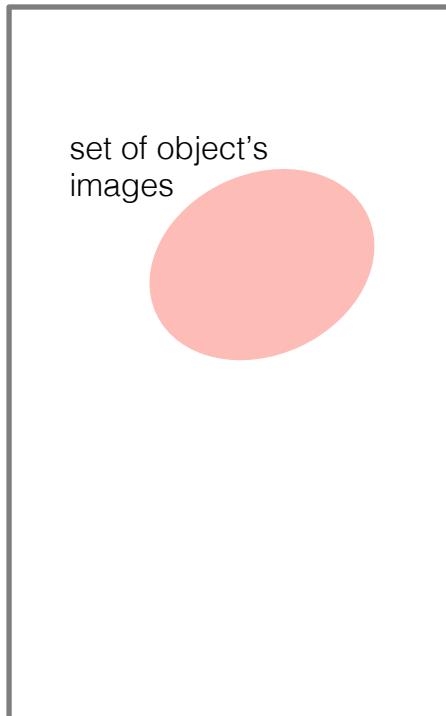


image space

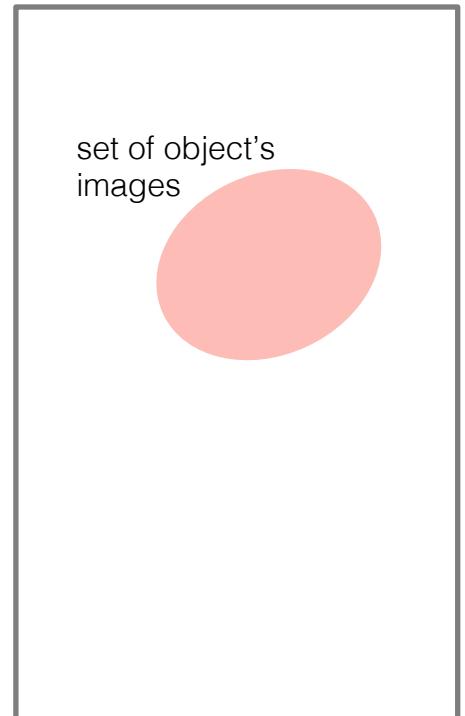
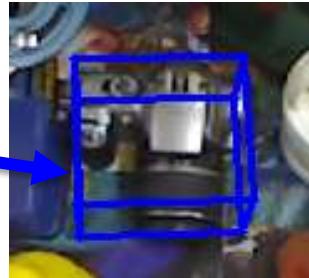


image space

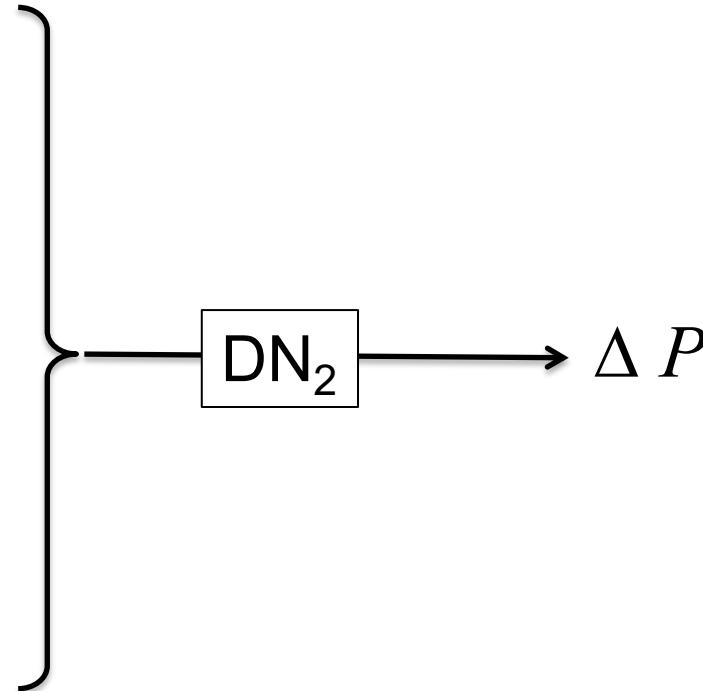
Refining the Pose

bounding
box for the
current pose
estimate

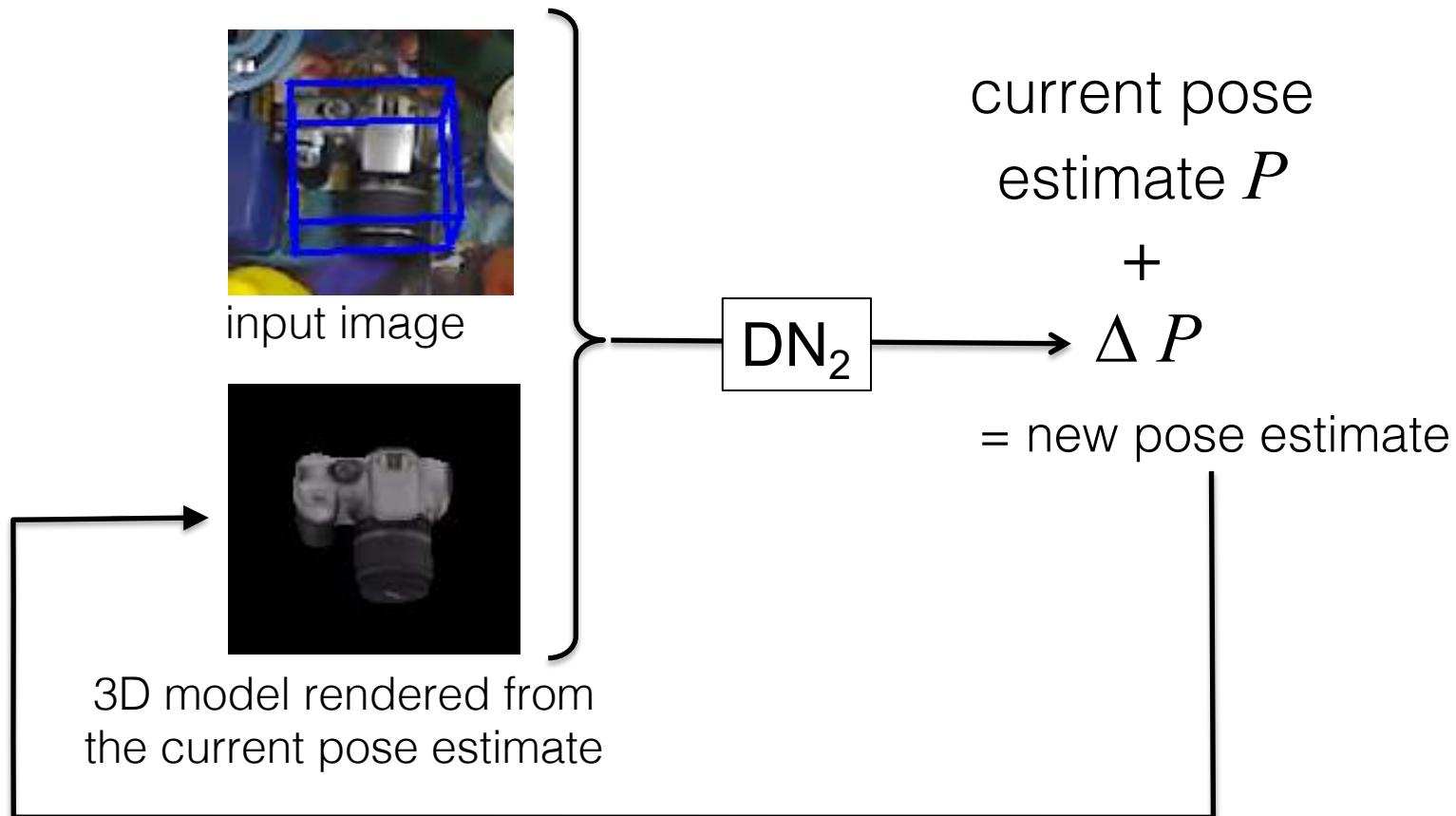


input image

3D model
rendered
from the
current pose
estimate

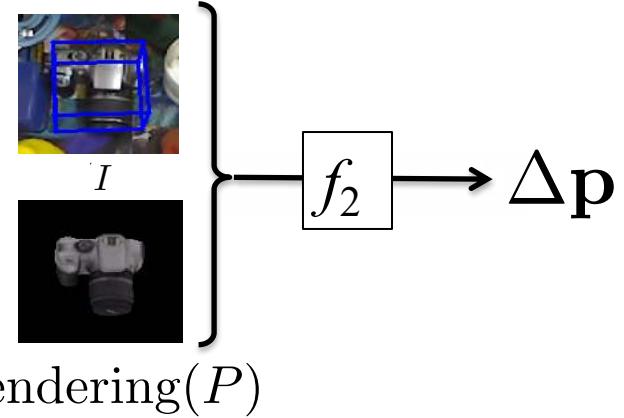


Refining the Pose

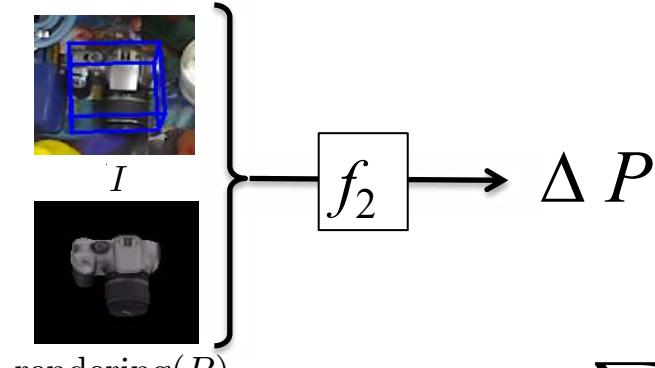


Refining the Pose: Loss

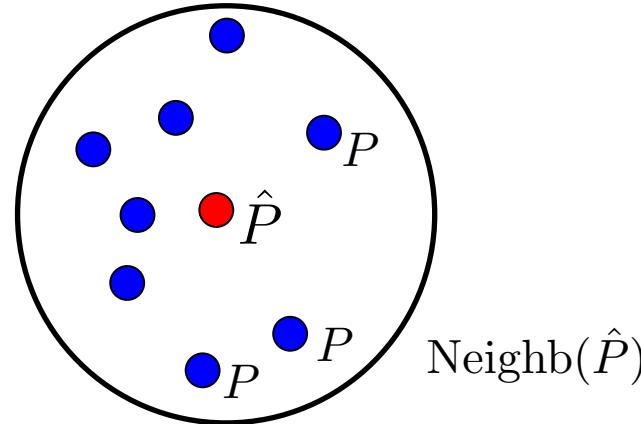
$$\Delta \mathbf{p} = f_2(I, \text{rendering}(\mathbf{p}); \Omega)$$



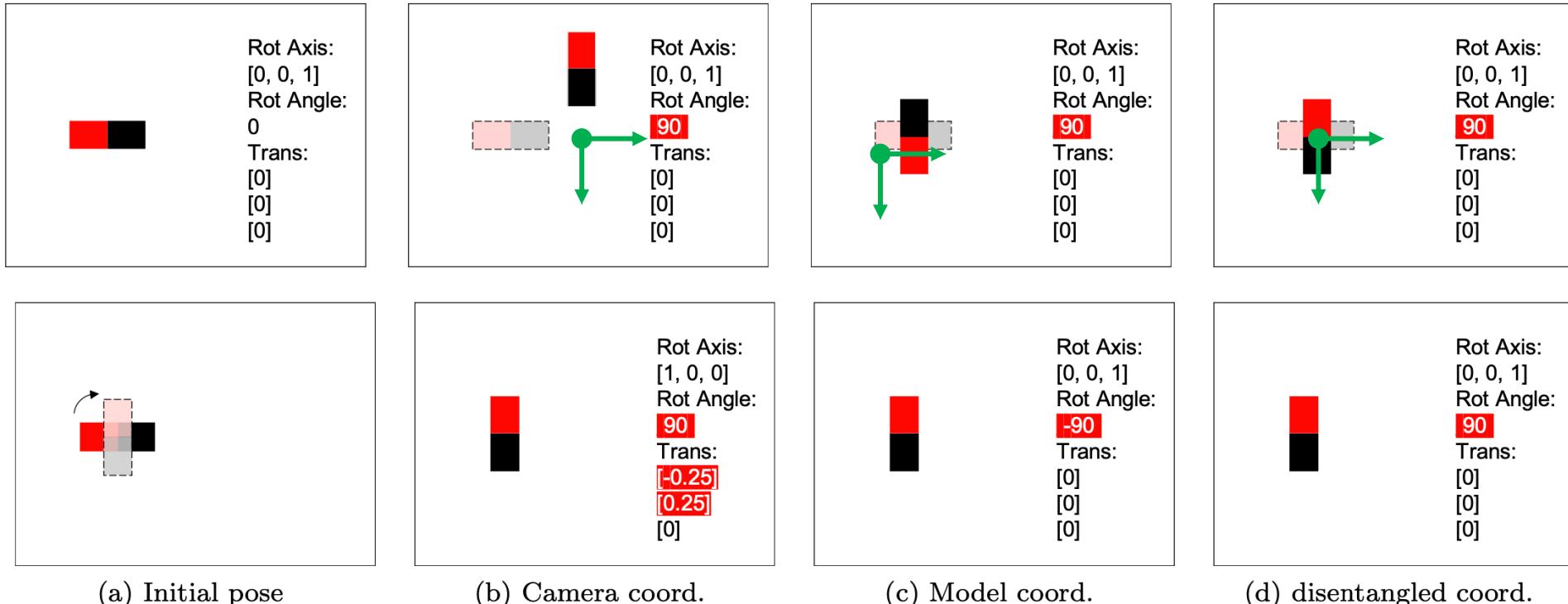
Refining the Pose: Loss



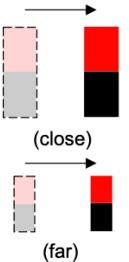
$$\mathcal{L}_2 = \sum_{P \in \text{Neigh}(\hat{P})} \mathcal{L}(\overbrace{f_2(I, \text{rendering}(P))}^{\Delta P}.P, \hat{P})$$



DeepIM: Decoupled Coordinates

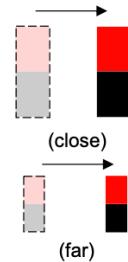


DeepIM: Decoupled Coordinates (T)



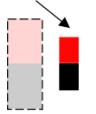
Trans:
[0.3]
[0]
[0]

Trans:
[0.6]
[0]
[0]



Trans:
[0.2]
[0]
[0]

Trans:
[0.2]
[0]
[0]



Trans:
[0]
[0]
[0.2]



Trans:
[0]
[0]
[-0.3]

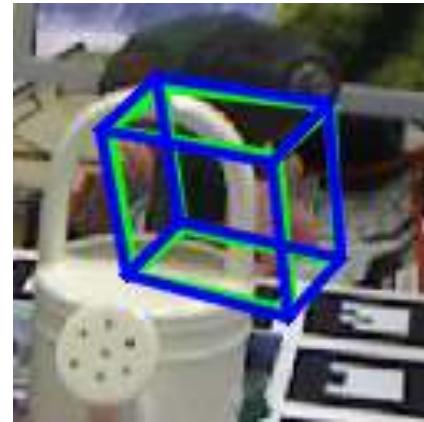
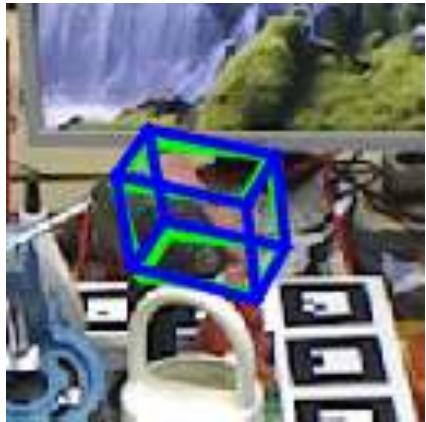
(a) Camera coord. xy-plane translation

(b) Disentangled coord. xy-plane translation

(c) Camera coord. z-axis translation

(d) Disentangled coord. z-axis translation

Dealing with Partial Occlusion



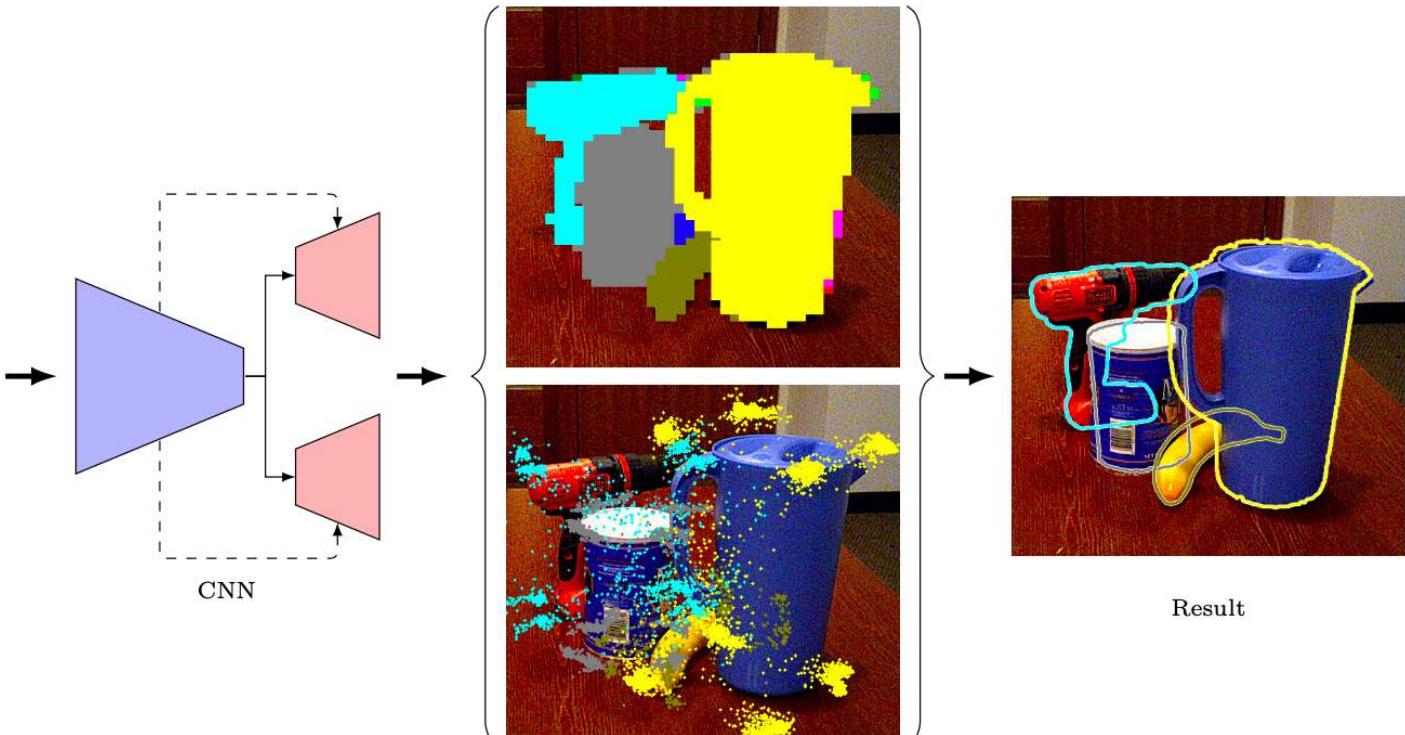
Avoid Occlusions in the Input



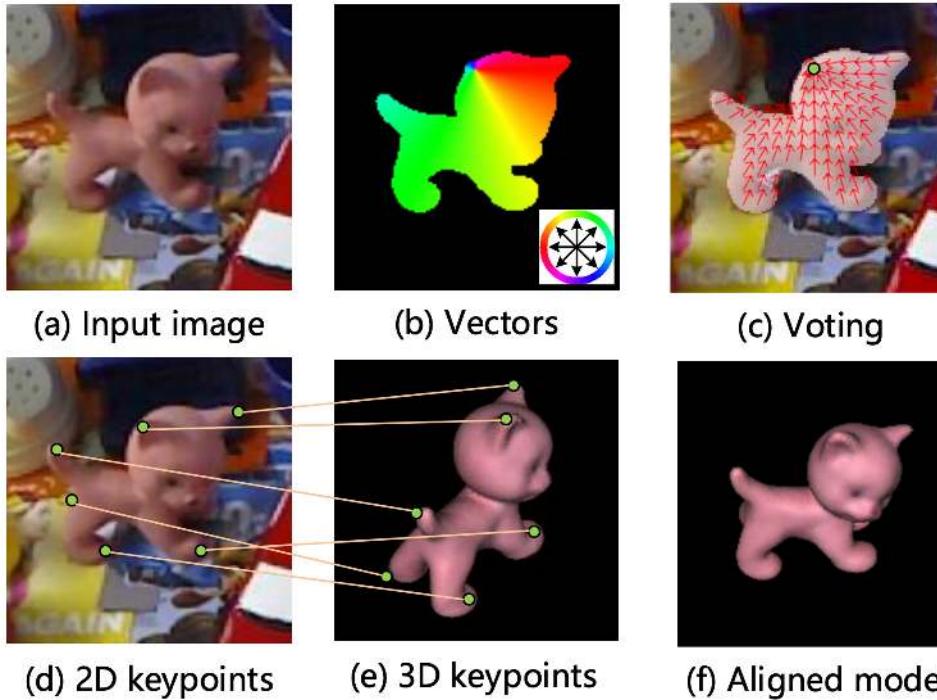
Voting for the pose



Input



Voting for the pose



PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation. Peng et al. CVPR 2019.

Conclusion

- Training set;
- Pose representation;
- Symmetrical objects;
- Partial occlusions.

Object Categories



3D Pose Prediction for Object Categories



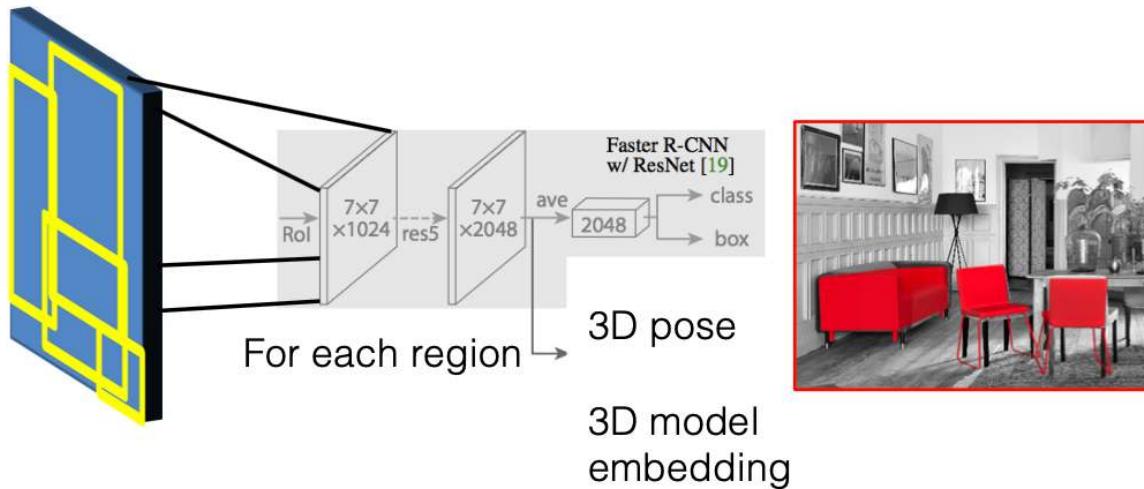
3D Pose Estimation and 3D Model Retrieval for Objects in the Wild. Alexander Grabner, Peter M. Roth, and Vincent Lepetit. CVPR 2018.

3D Pose Prediction for Object Categories



2D bounding boxes from Mask-RCNN

Mask-RCNN: Extension to 3D



Alexander Grabner, Peter M. Roth, and Vincent Lepetit. "3D Pose Estimation and 3D Model Retrieval for Objects in the Wild". In: CVPR. 2018.

Qualitative Results



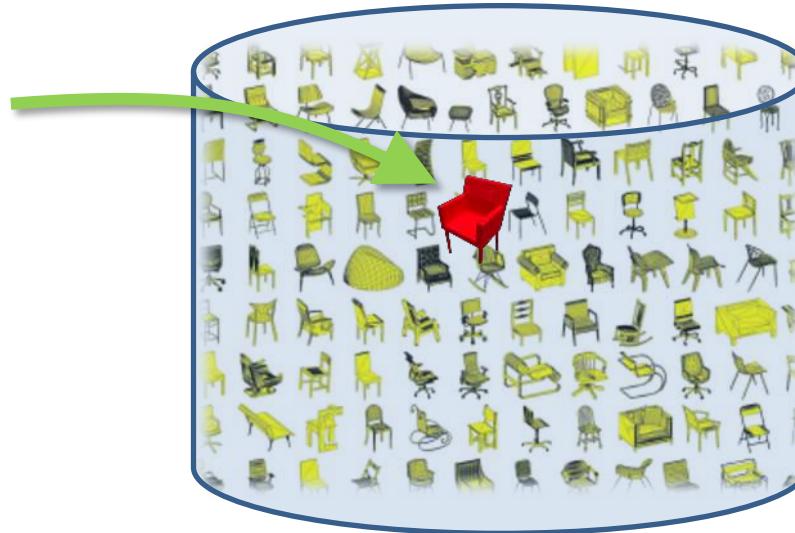
3D Geometry Retrieval for Object Categories



3D Model Retrieval for Object Categories

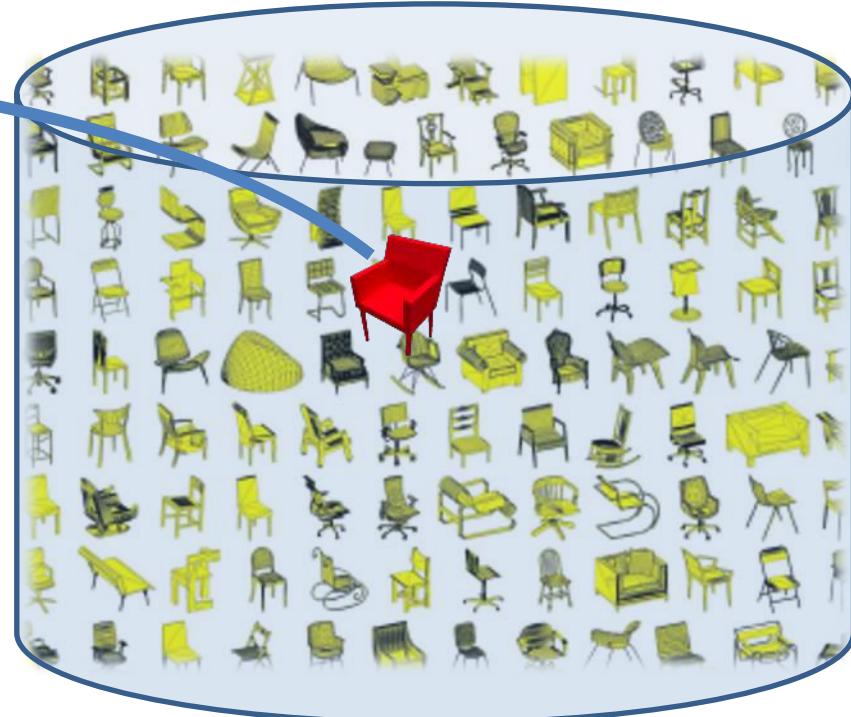
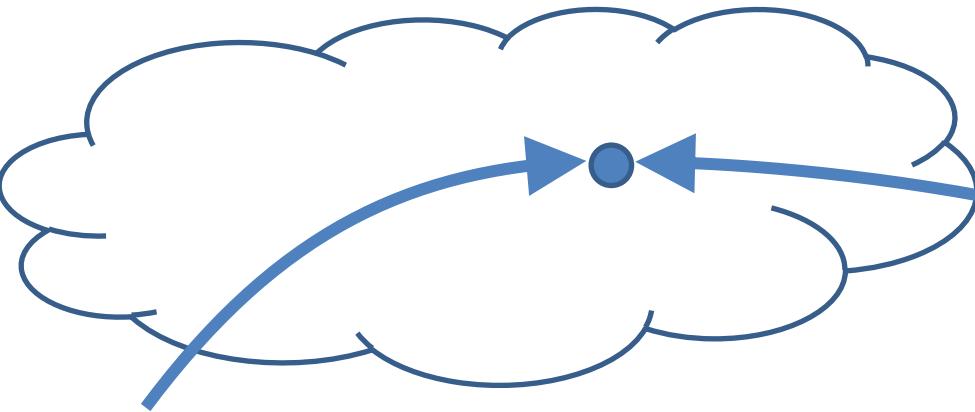
Possible options: Predicting a point cloud, voxels, 3D planes, ...

We look for a man-made 3D model similar to the object.



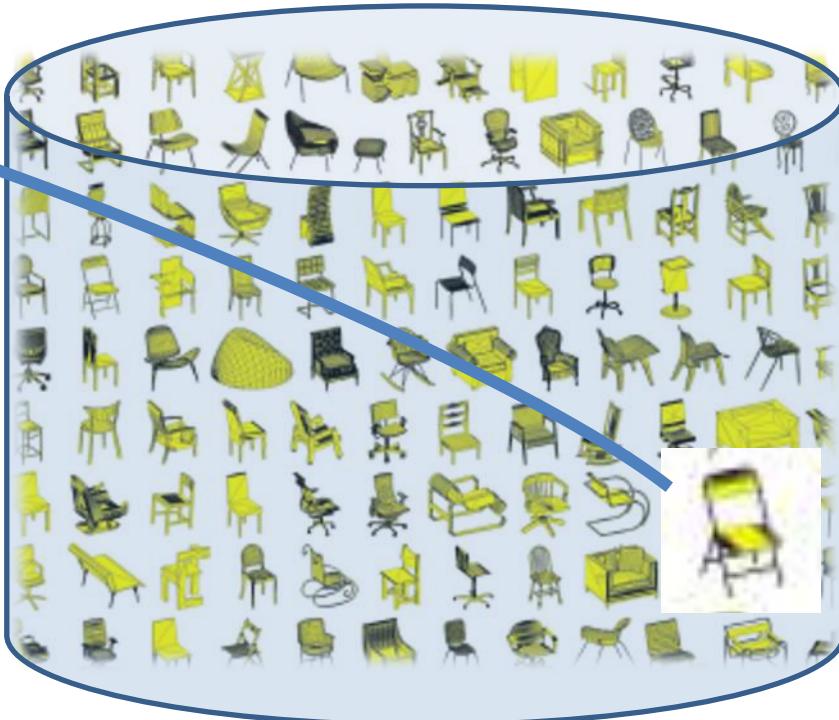
ShapeNet [Chang et al, 2015]

shared embedding space

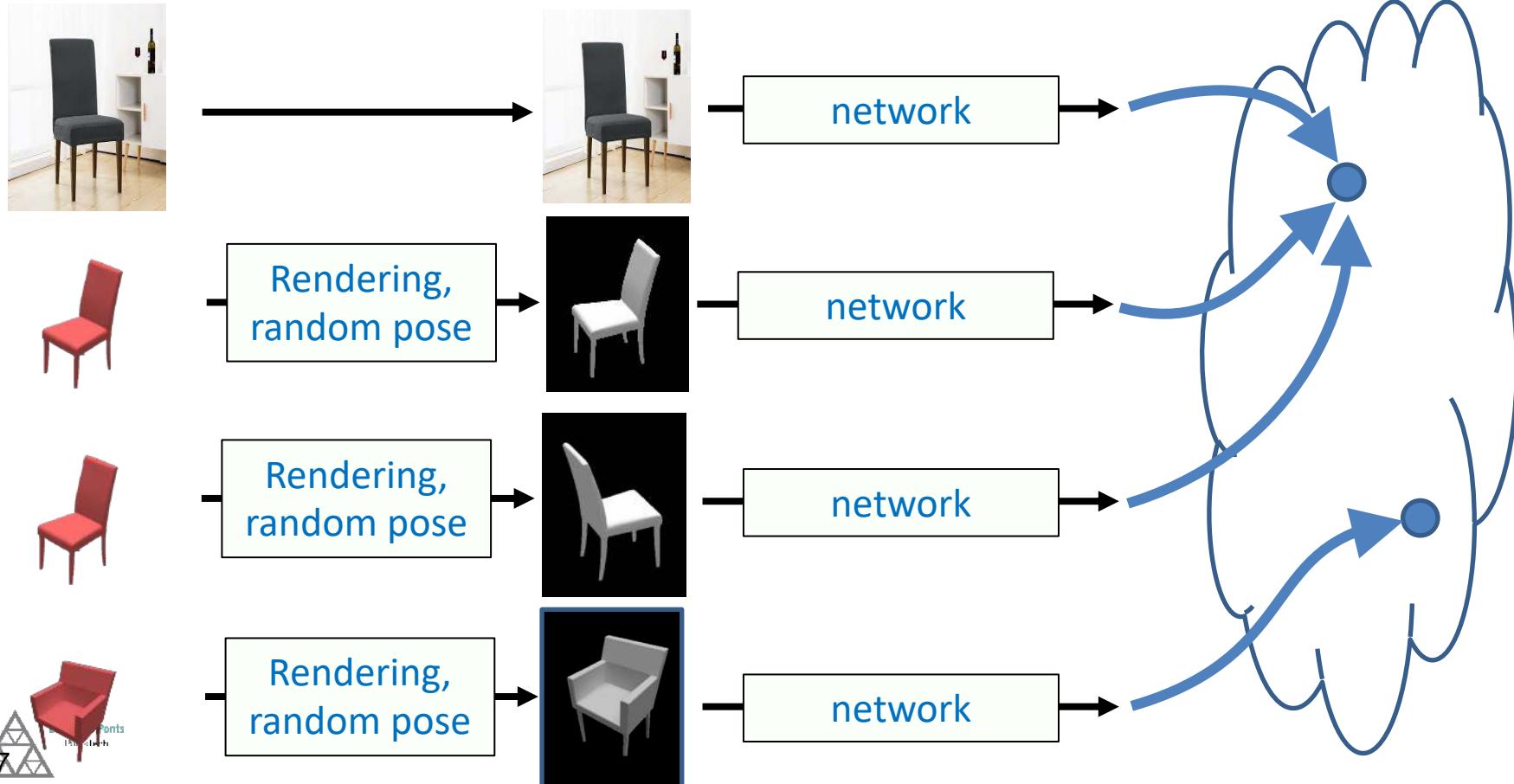


ShapeNet [Chang et al, 2015]

pose-invariant embedding



Pose Invariant Embeddings & Metric Learning



Location Fields/Object Coordinates/..

For each pixel: the 3D coordinates on the object's surface, in the object's coordinate system:



RGB



X

Y

Z

Pose Invariant Embeddings & Metric Learning



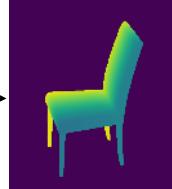
Location Field
prediction
network



network



Rendering,
random pose



network

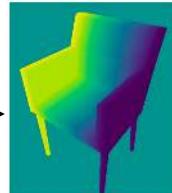


Rendering,
random pose

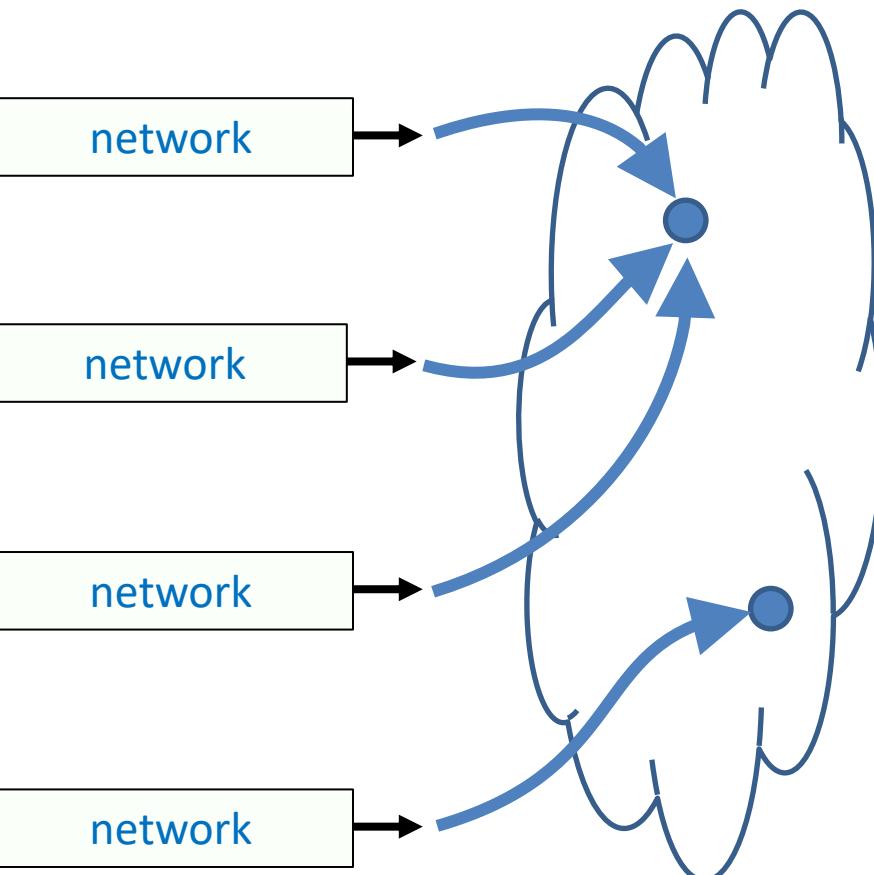


network

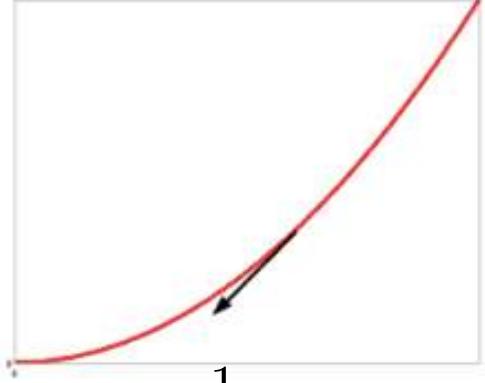
Rendering,
random pose



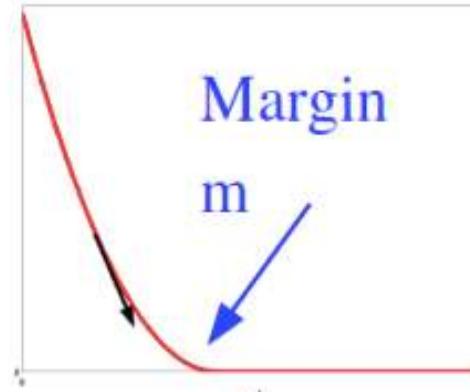
network



Pose Invariant Embeddings & Metric Learning: Loss (called *contrastive loss*)



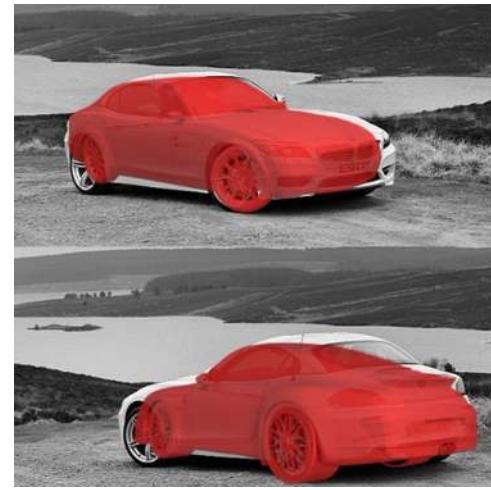
$$L_{\text{Similar}}(x_1, x_2) = \frac{1}{2} D_w^2(x_1, x_2)$$



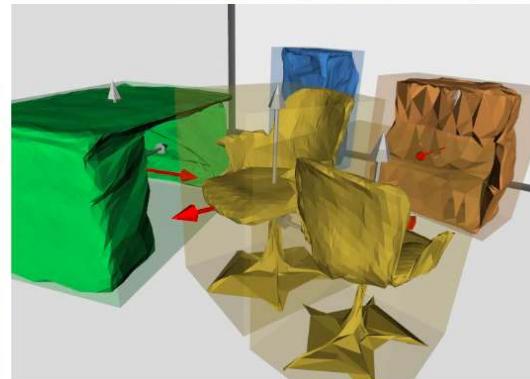
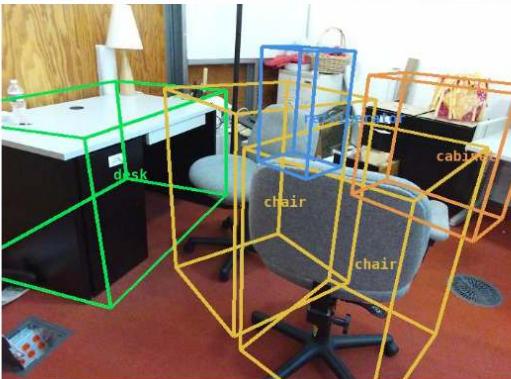
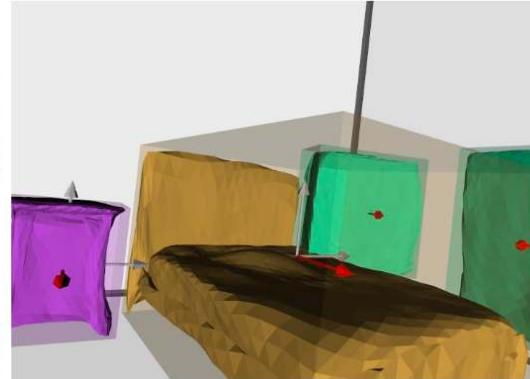
$$L_{\text{Dissimilar}}(x_1, x_2) = \frac{1}{2} [\max(0, m - D_w(x_1, x_2))]^2$$

$$D_w(x_1, x_2) = \|G_w(x_1) - G_w(x_2)\|$$

Qualitative Results

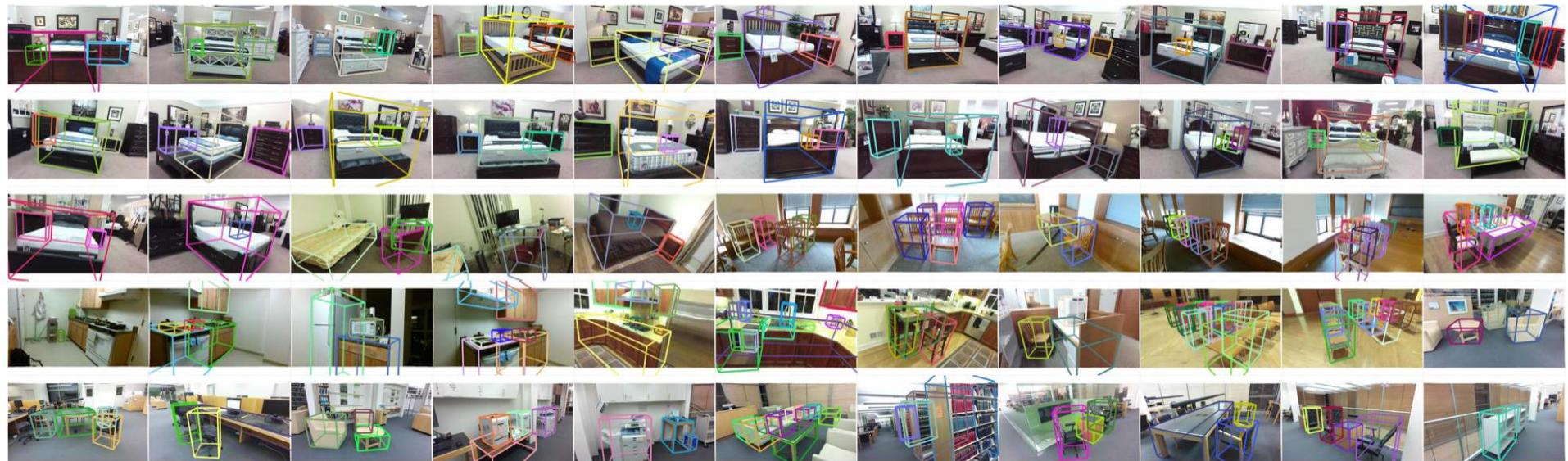


Total3DUnderstanding



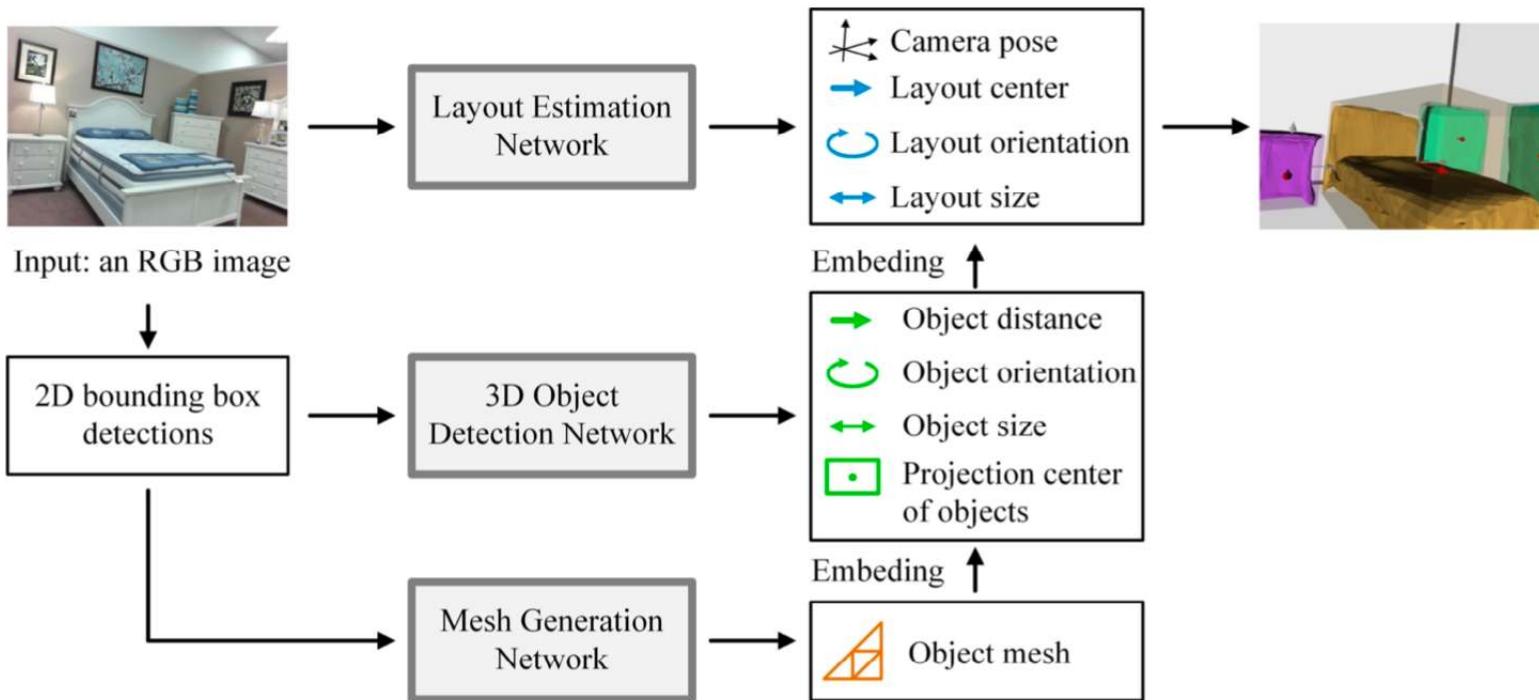
[Total3DUnderstanding: Joint Layout, Object Pose and Mesh Reconstruction for Indoor Scenes from a Single Image. Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, Jian Jun Zhang. CVPR 2020]

Training data

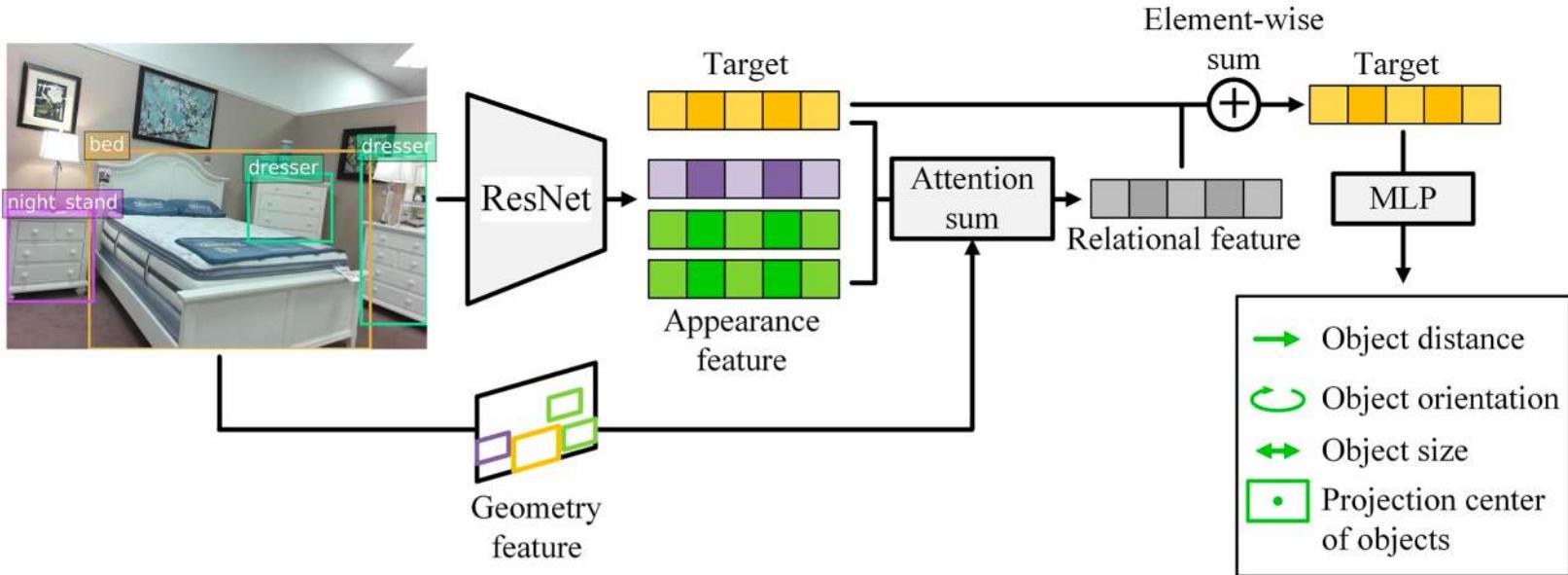


SUN-RGBD dataset, ~10'000 images annotated manually

Total3DUnderstanding



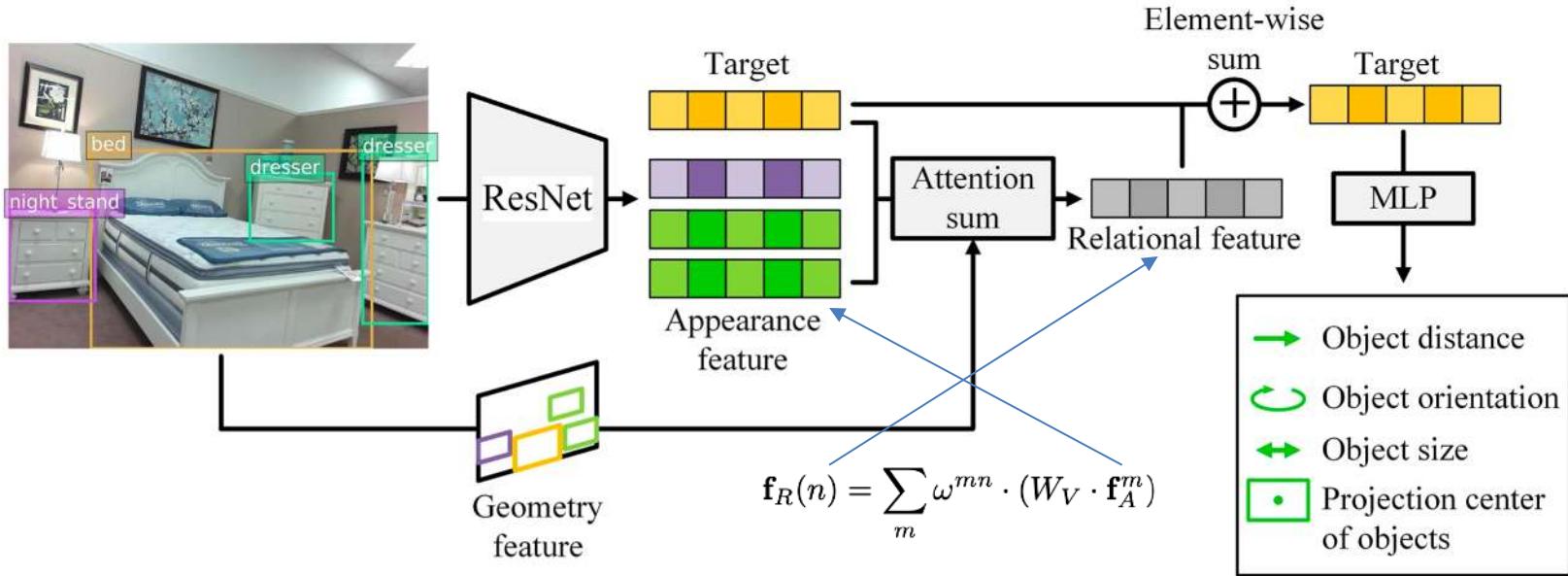
Total3DUnderstanding: 3D object pose prediction



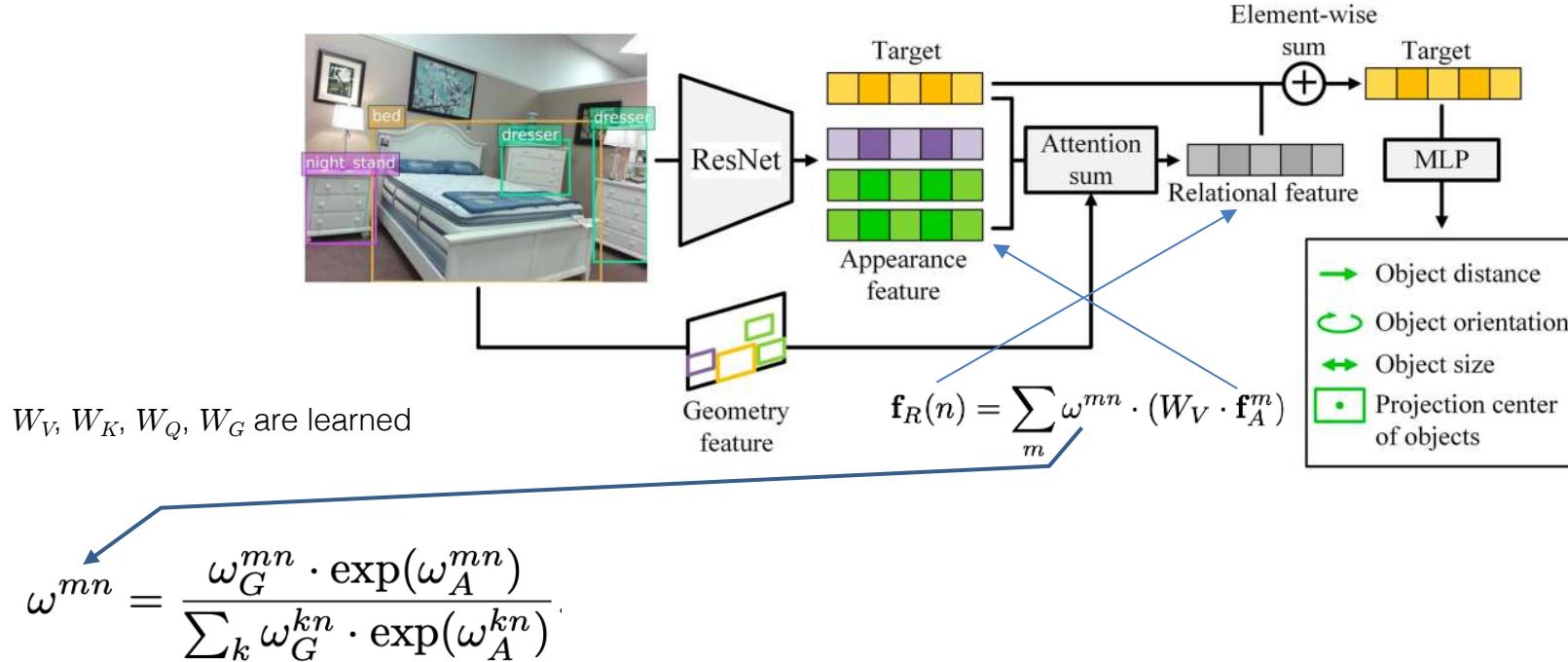
Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation Networks for Object Detection. CVPR 2018.

Analogy with NLP: the appearance features correspond to the embeddings of the words of a sentence, the geometry features correspond to the positions of the words in the sentence. → Use attention!

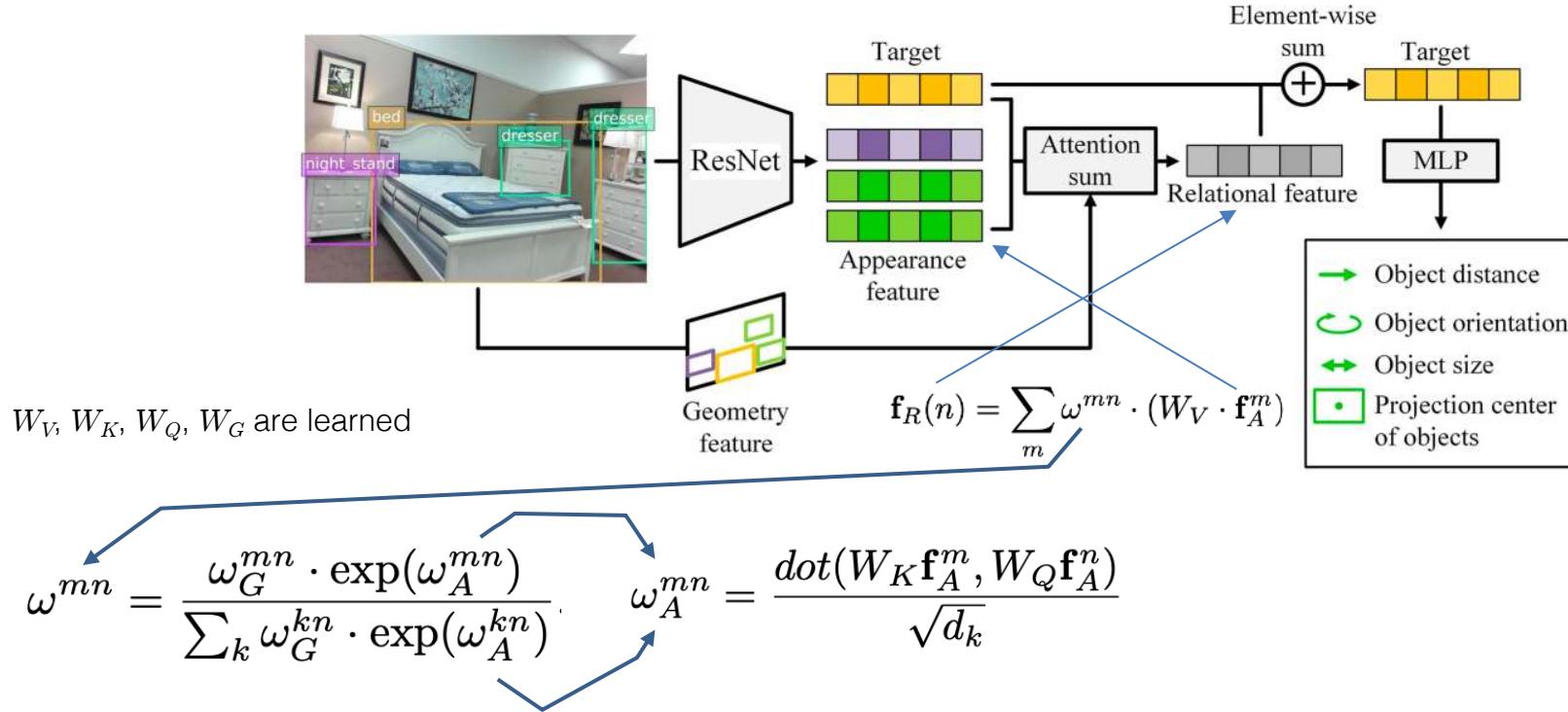
Total3DUnderstanding: 3D object pose prediction



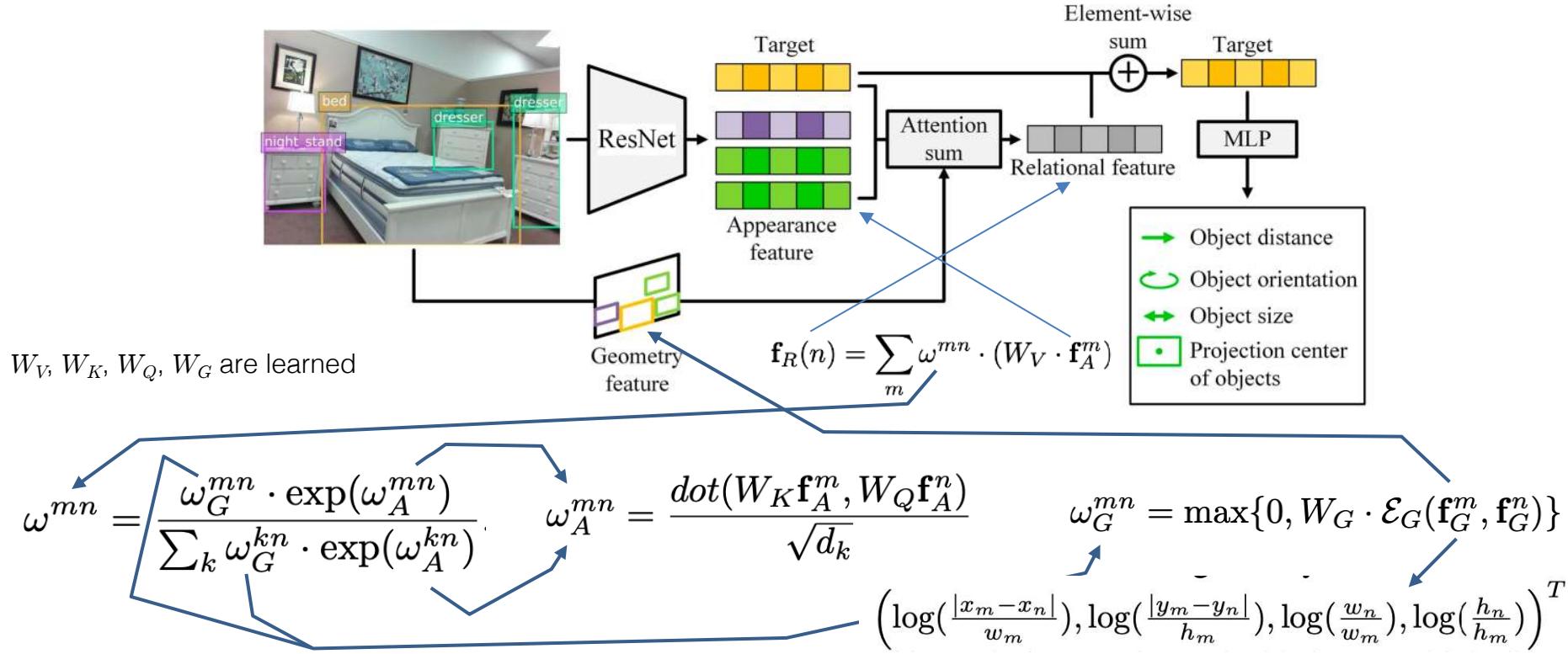
Total3DUnderstanding: 3D object pose prediction



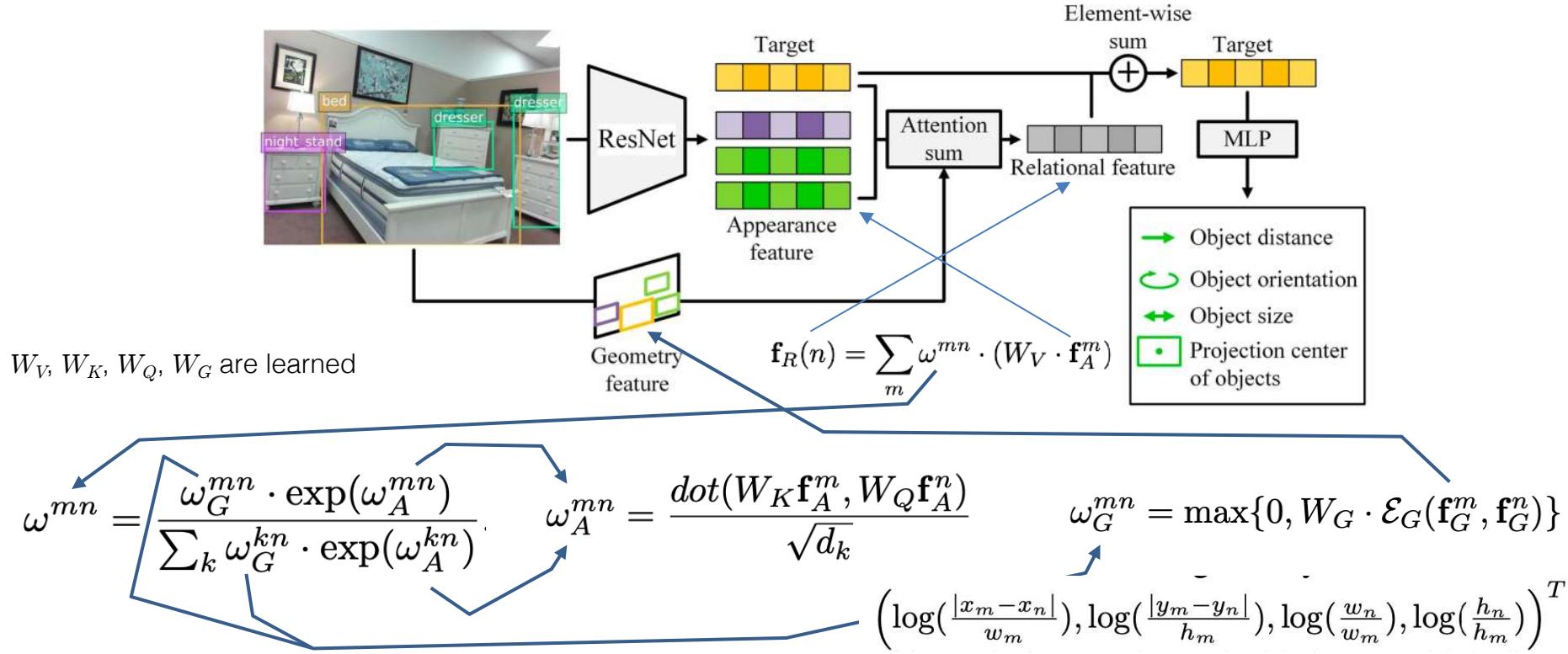
Total3DUnderstanding: 3D object pose prediction



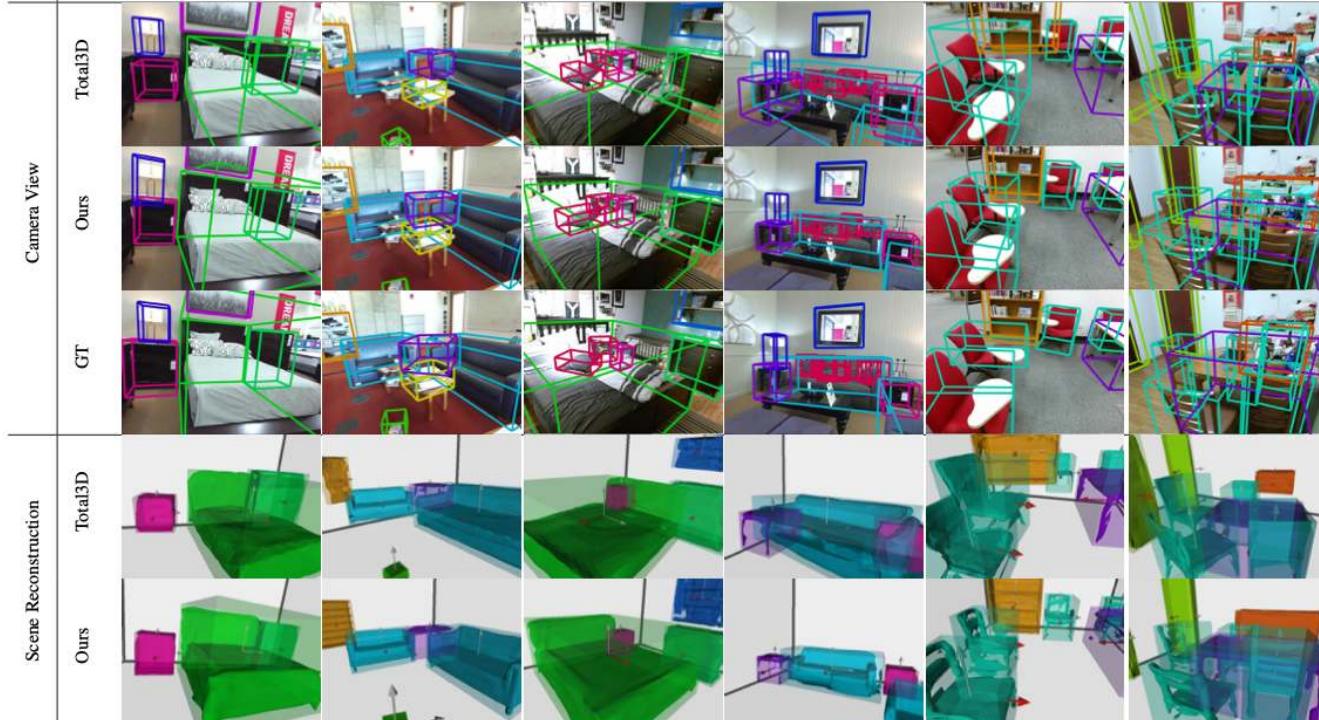
Total3DUnderstanding: 3D object pose prediction



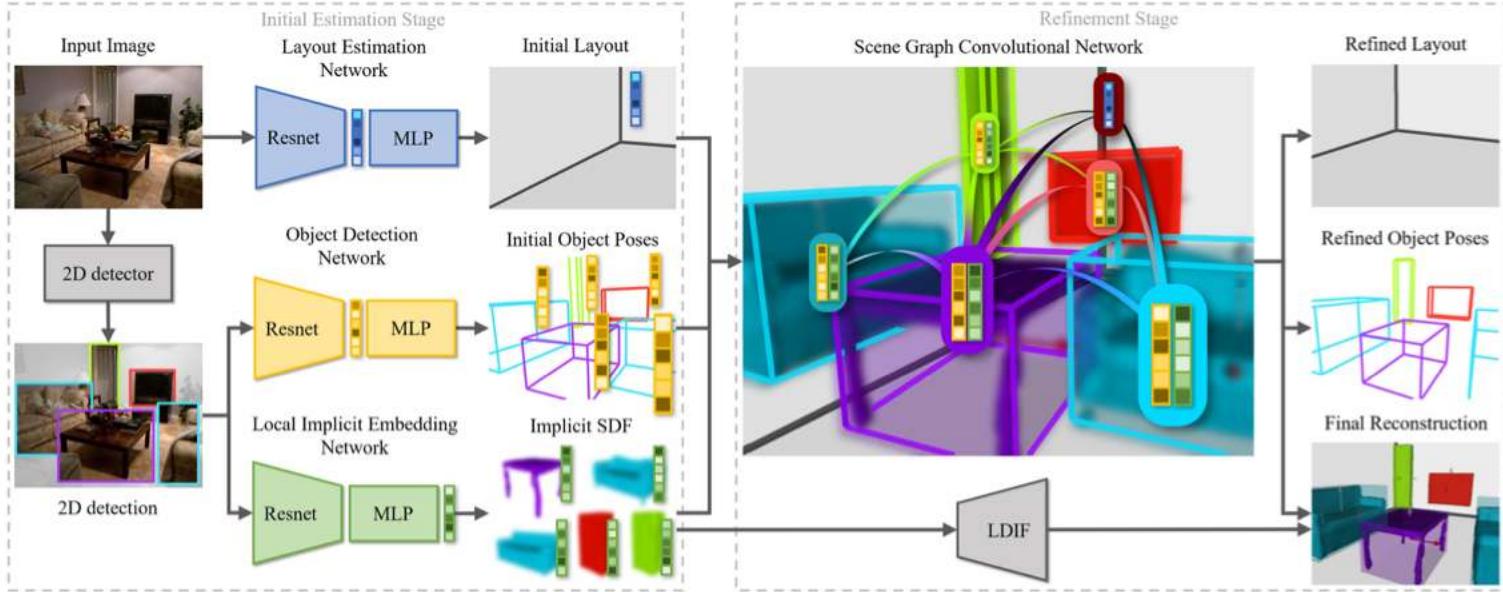
Total3DUnderstanding: 3D object pose prediction



“Holistic 3D Scene Understanding”

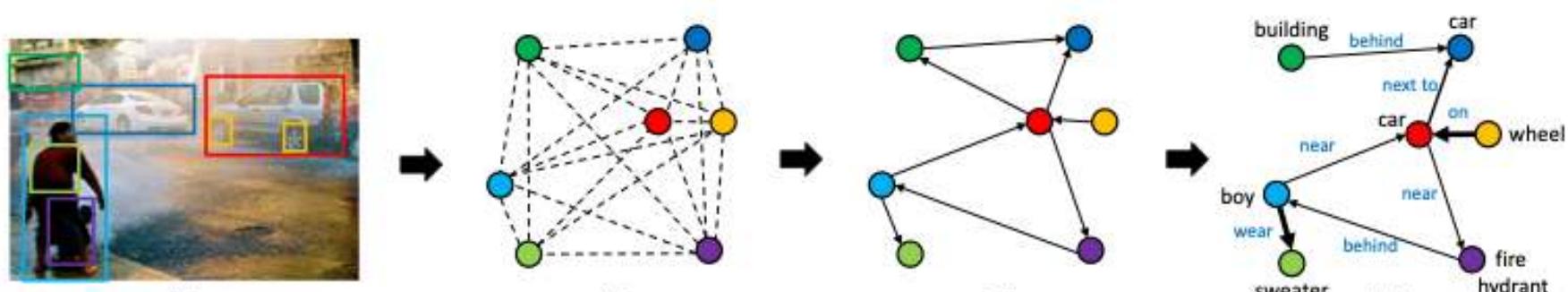


Holistic 3D Scene Understanding from a Single Image with Implicit Representation. Zhang et al. CVPR 2021.



Holistic 3D Scene Understanding from a Single Image with Implicit Representation. Zhang et al. CVPR 2021.

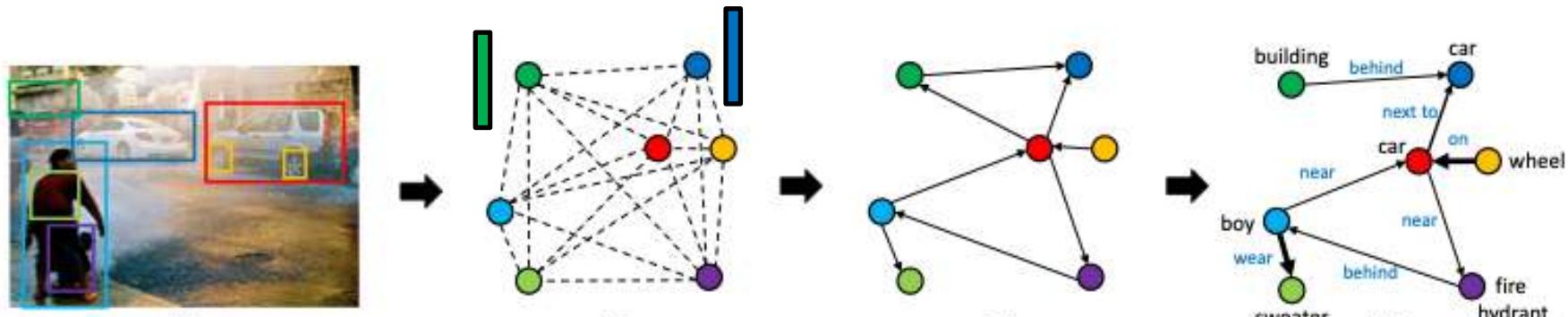
Graph R-CNN for Scene Graph Generation



Graph R-CNN for Scene Graph Generation. Yang et al. ECCV 2018.

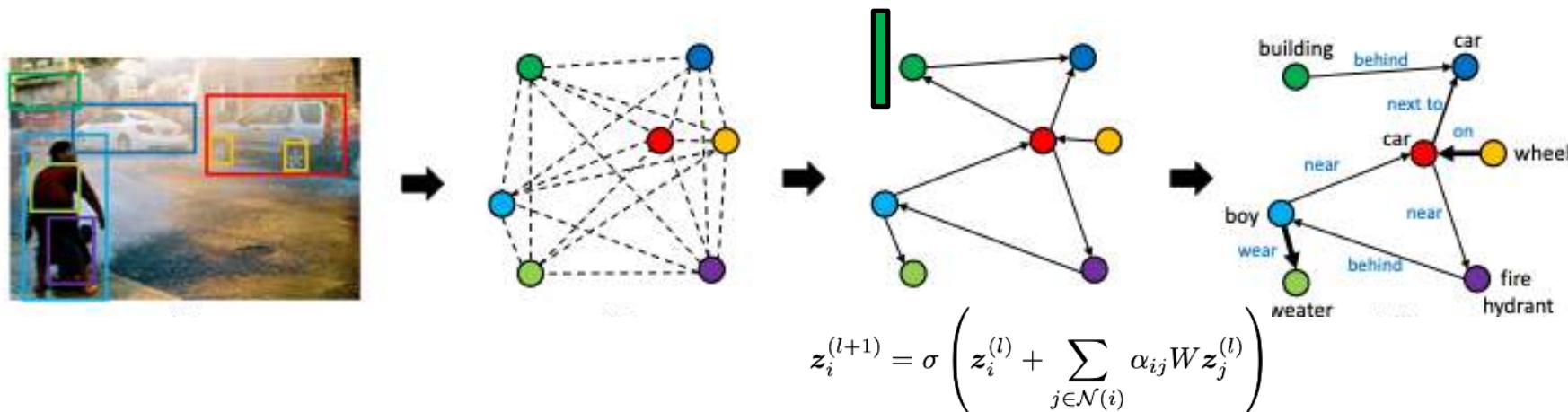
Graph R-CNN for Scene Graph Generation

$$f(\mathbf{p}_i^o, \mathbf{p}_j^o) = \langle \Phi(\mathbf{p}_i^o), \Psi(\mathbf{p}_j^o) \rangle$$



Graph R-CNN for Scene Graph Generation. Yang et al. ECCV 2018.

Graph R-CNN for Scene Graph Generation



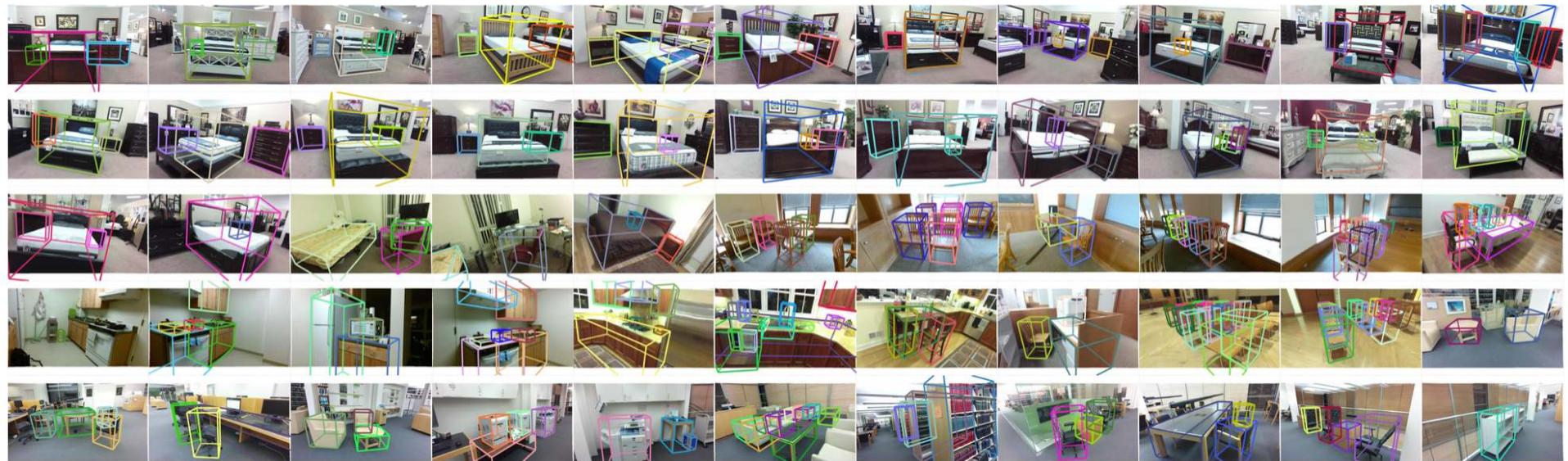
$$\mathbf{z}_i^{(l+1)} = \sigma \left(\mathbf{z}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} W \mathbf{z}_j^{(l)} \right)$$

$$\mathbf{z}_i^{(l+1)} = \sigma \left(W Z^{(l)} \boldsymbol{\alpha}_i \right)$$

$$u_{ij} = w_h^T \sigma(W_a[\mathbf{z}_i^{(l)}, \mathbf{z}_j^{(l)}])$$

$$\boldsymbol{\alpha}_i = \text{softmax}(\mathbf{u}_i),$$

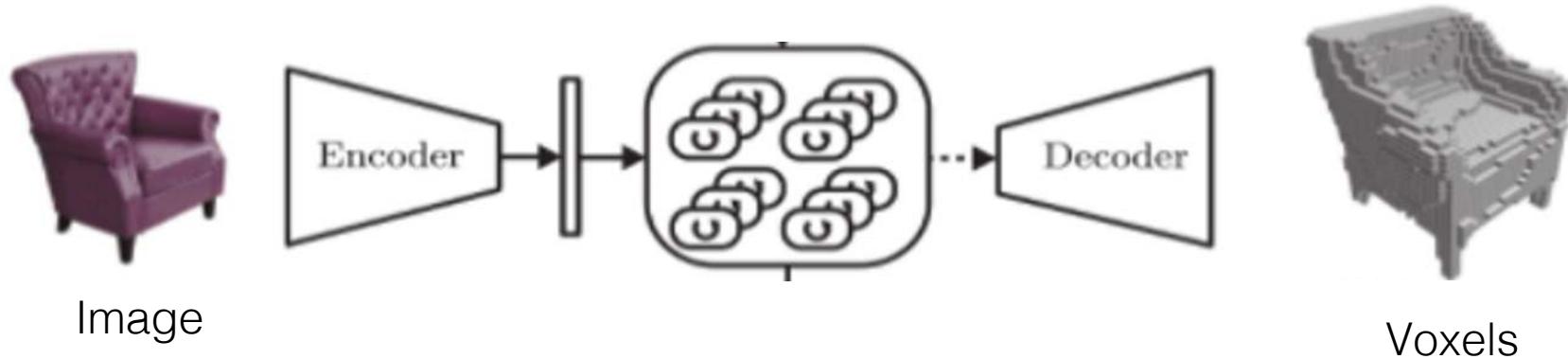
Training data



SUN-RGBD dataset, ~10'000 images annotated manually,
2000+ hours for initial annotations + time for error correction

3D model prediction from an image

Voxels

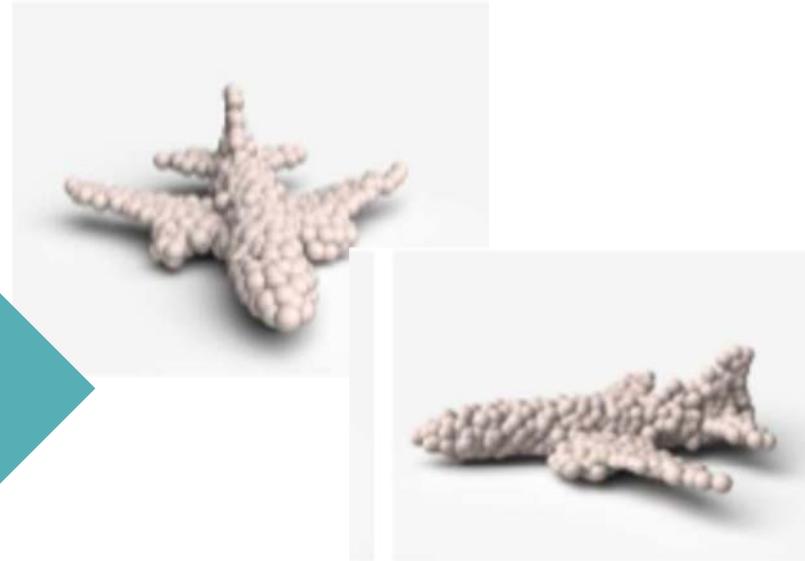


Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction, ECCV 2016

Unstructured Points



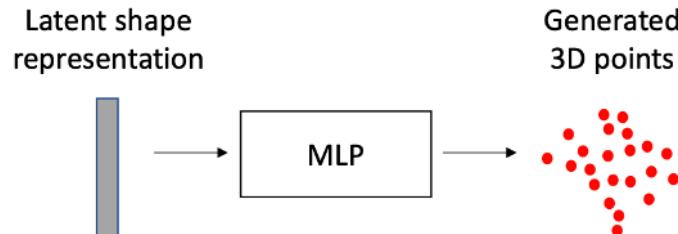
Image



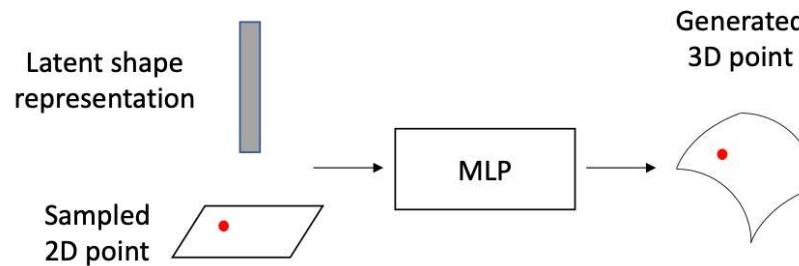
Points

Choy, C. B., Xu, D., Gwak, J., Chen, K., & Savarese, S. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction, ECCV 2016

AtlasNet

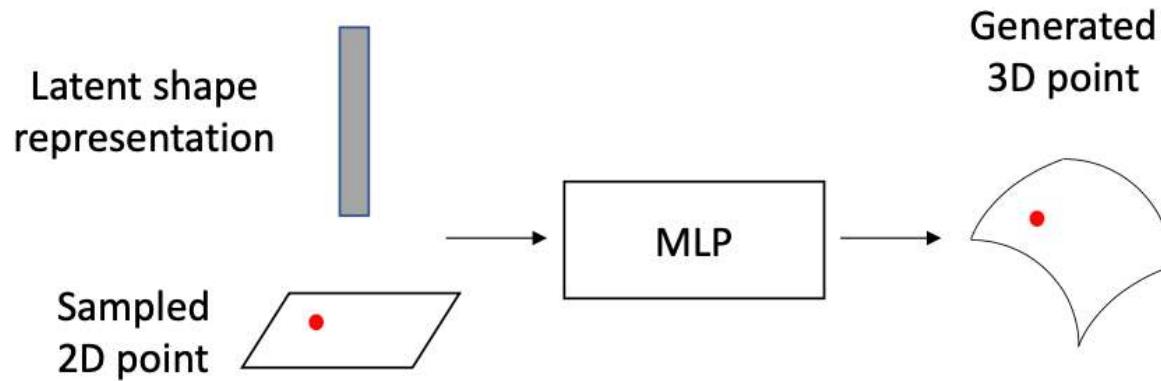


(a) Points baseline.



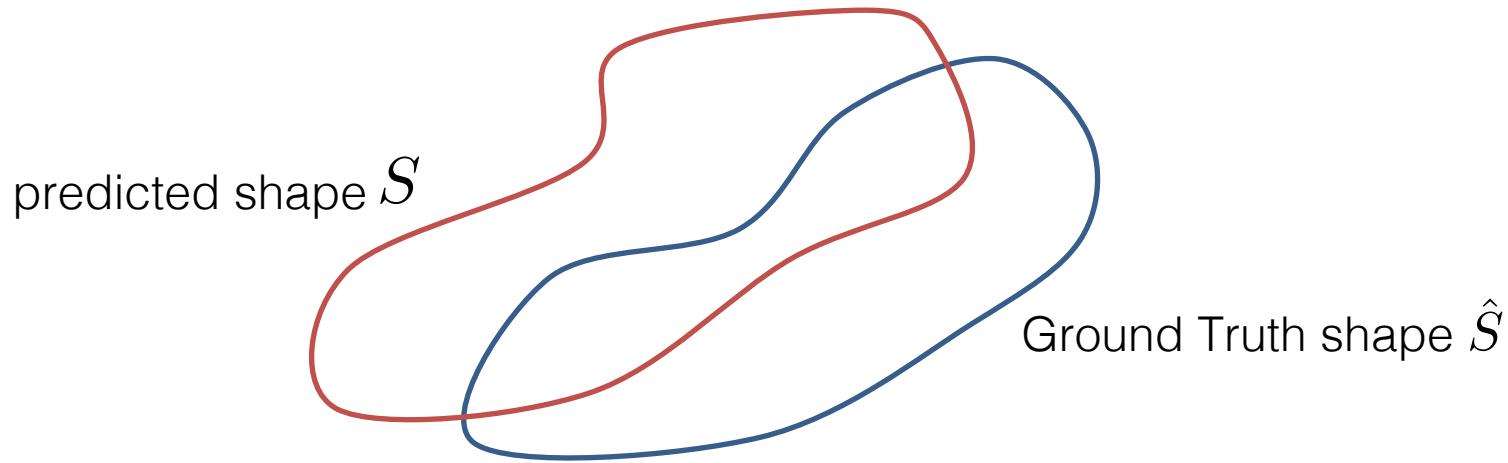
AtlasNet

AtlasNet



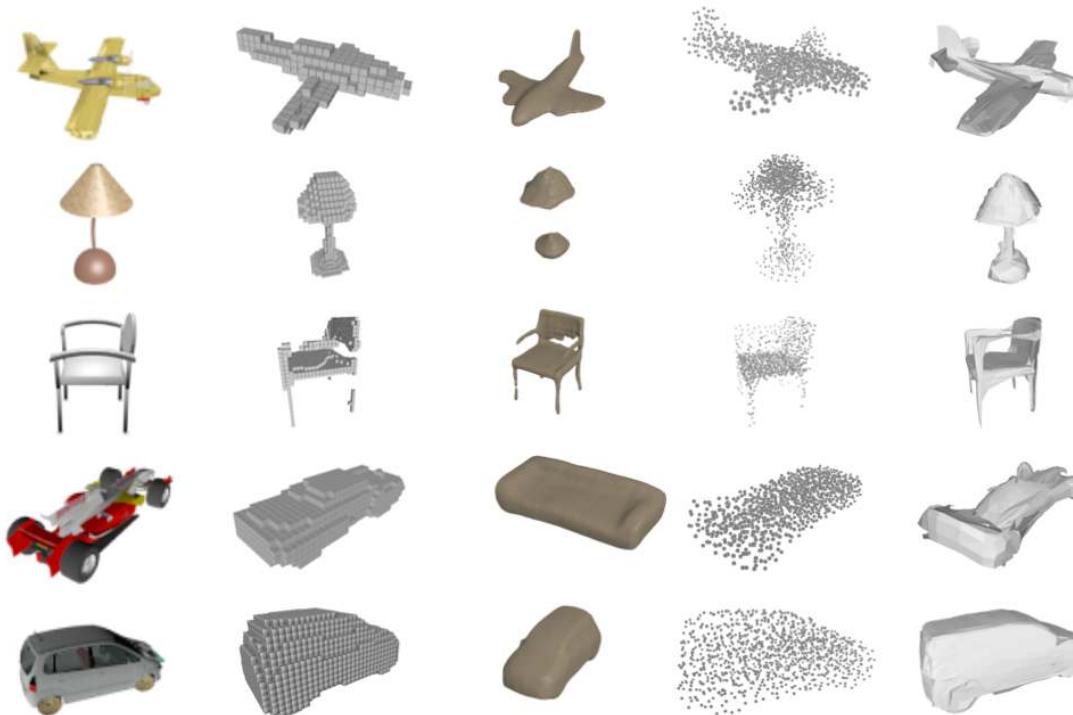
AtlasNet: Loss Function

Loss is the Chamfer distance between the predicted shape and the ground truth shape.



$$\mathcal{L} = \sum_{M \in S} \min_{M' \in \hat{S}} \|M - M'\|^2 + \sum_{M' \in \hat{S}} \min_{M \in S} \|M - M'\|^2$$

AtlasNet: Qualitative comparisons



(a) Input

(b) 3D-R2N2

(c) HSP

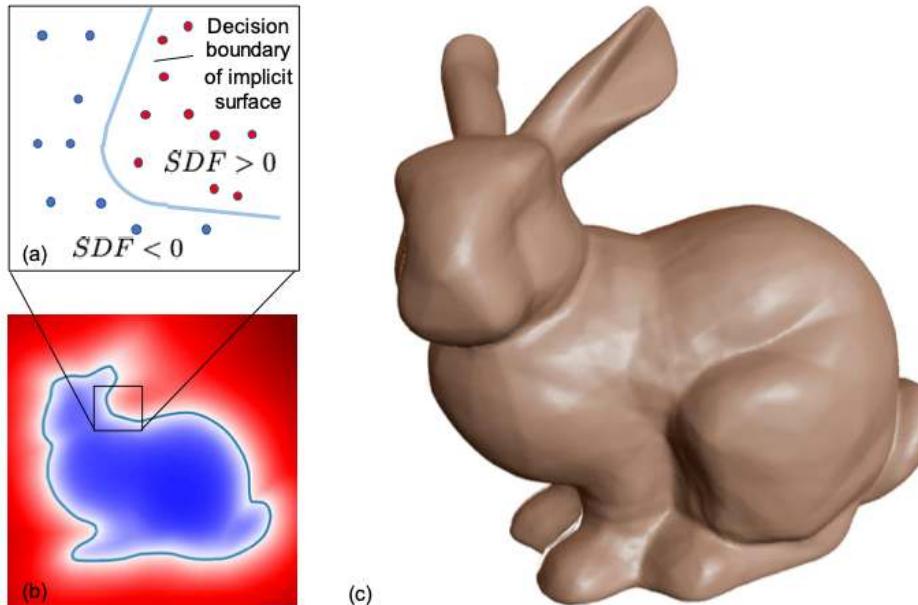
(d) PSG

(e) Ours

Signed Distance Functions

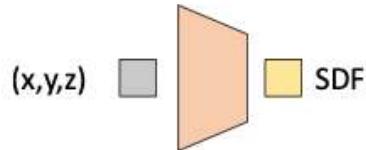


DeepSDF

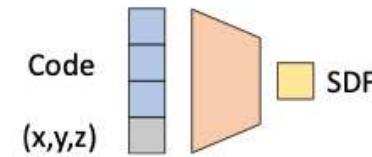


DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation.
Park et al. CVPR 2019.

DeepSDF



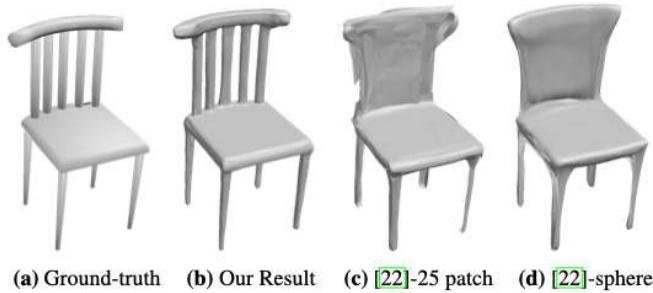
(a) Single Shape DeepSDF



(b) Coded Shape DeepSDF

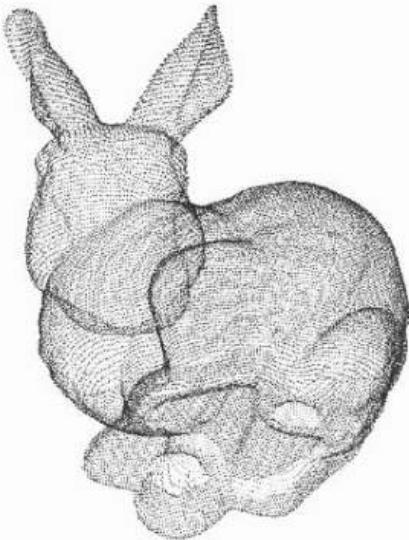
$$\mathcal{L}(f_\theta(\mathbf{x}), s) = | \text{clamp}(f_\theta(\mathbf{x}), \delta) - \text{clamp}(s, \delta) |$$

DeepSDF: some results



Point Cloud Analysis

Processing Point Clouds

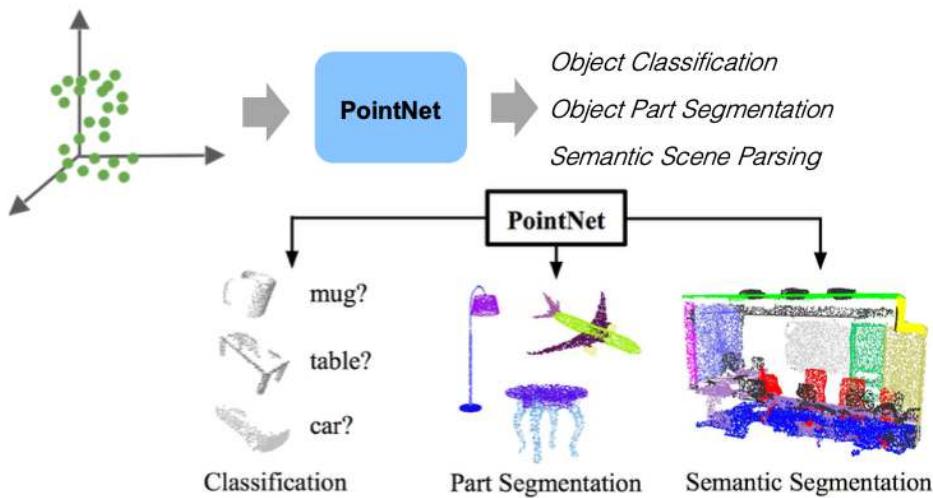


The problem with point clouds

PointNet

End-to-end learning for **scattered, unordered point data**

Unified framework for various tasks



C.R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Conference on Computer Vision and Pattern Recognition*. 2017.

PointNet

- ▶ Unordered point set as input: Model needs to be invariant to $N!$ permutations;
- ▶ Also, model needs to be invariant under geometric transformations.

Permutation invariance:

$$f(x_1, x_2, \dots, x_n) = f(x_{\sigma(1)}, x_{\sigma(2)}, \dots, x_{\sigma(n)}). \quad (19)$$

Examples:

$$f(x_1, x_2, \dots, x_n) = \max\{x_1, x_2, \dots, x_n\} \quad (20)$$

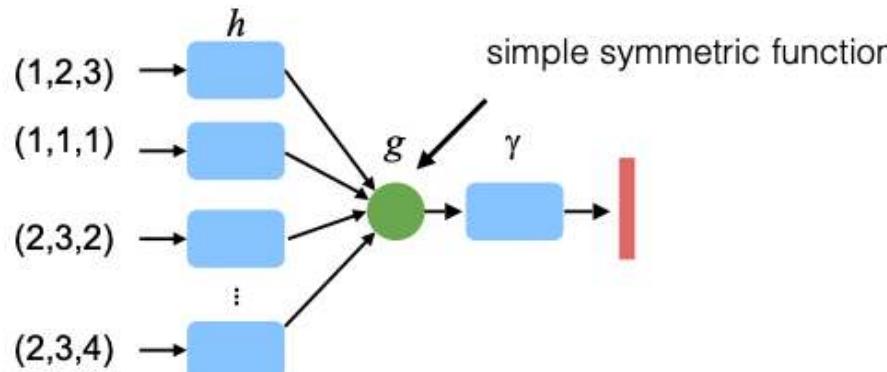
$$f(x_1, x_2, \dots, x_n) = x_1 + x_2 + \dots + x_n \quad (21)$$

How can we construct a family of symmetric functions with neural networks?

PointNet: Invariance to ordering

$$f(x_1, x_2, \dots, x_n) = \gamma(g(h(x_1), h(x_2), \dots, h(x_n)))$$

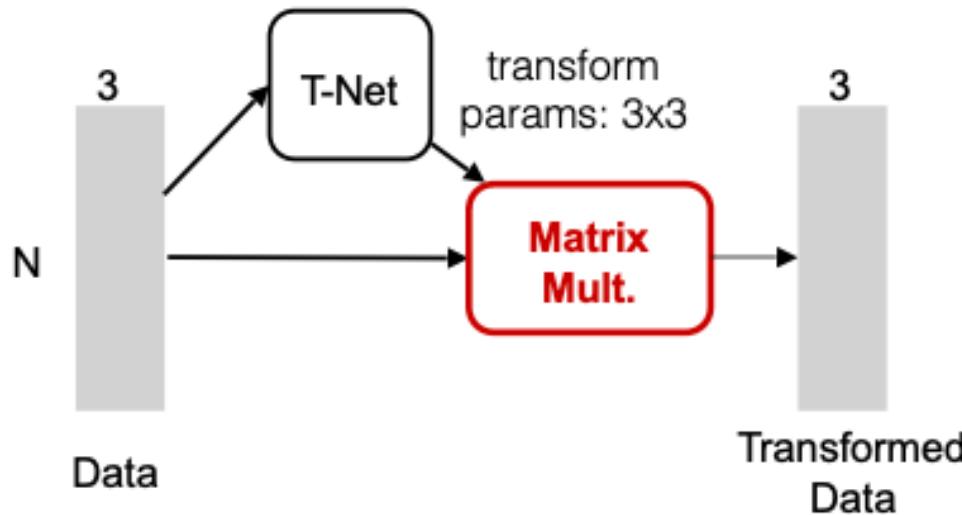
is symmetric if g is symmetric.



Use neural networks for h , γ , and max-pooling for g .

PointNet: Intuition

PointNet: Invariance to geometric transformation



PointNet: Full network



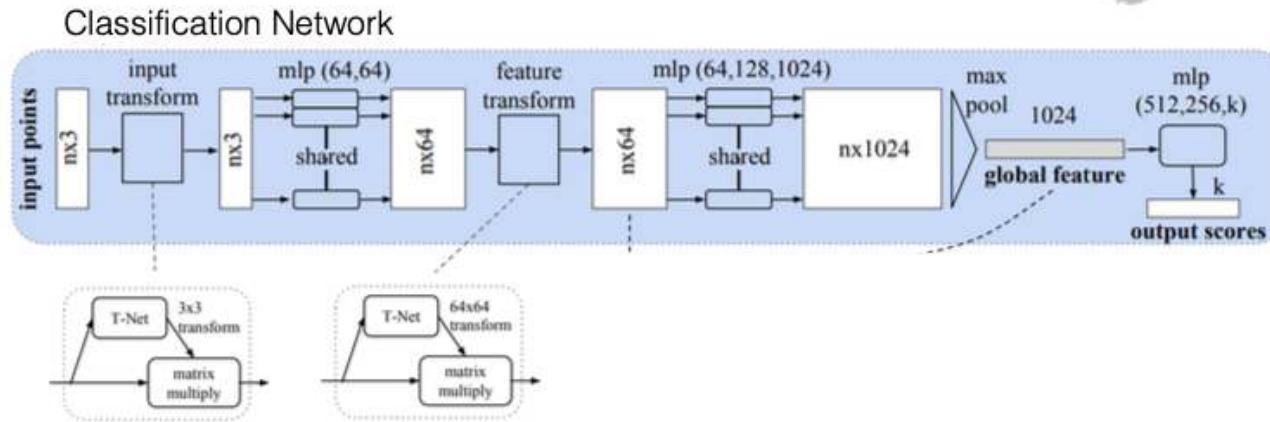
mug?



table?



car?



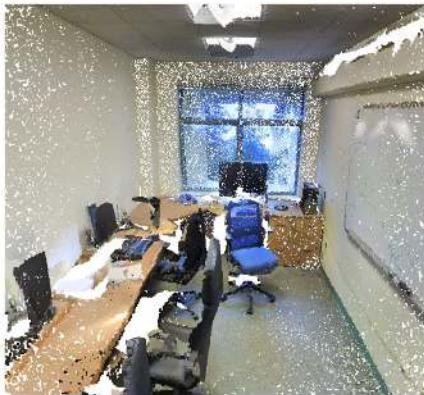
n: number of points;

k: number of possible objects;

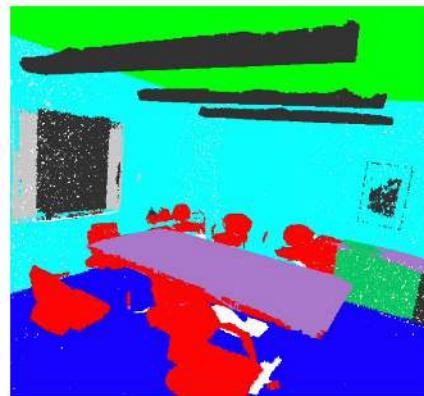
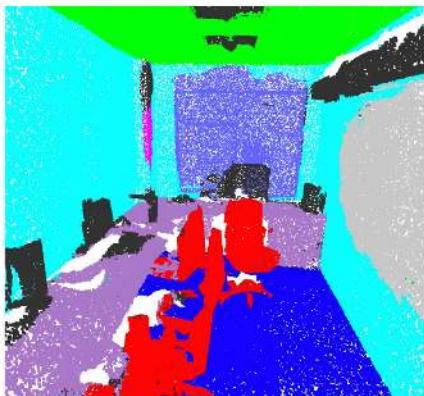
m: number of possible segments.

PointNet: some results

Input

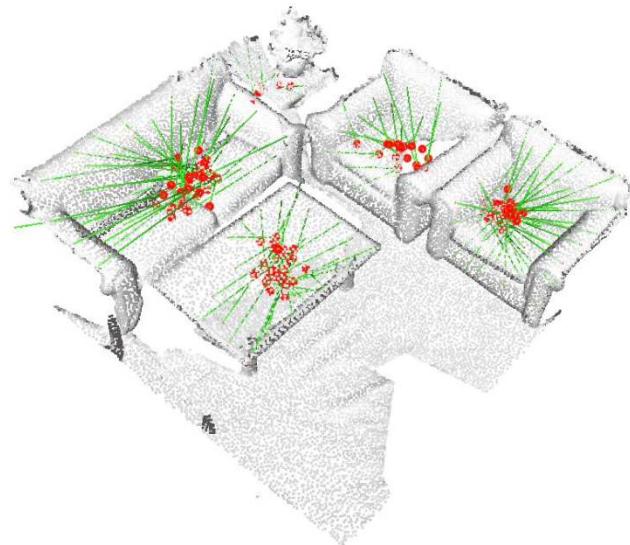


Output

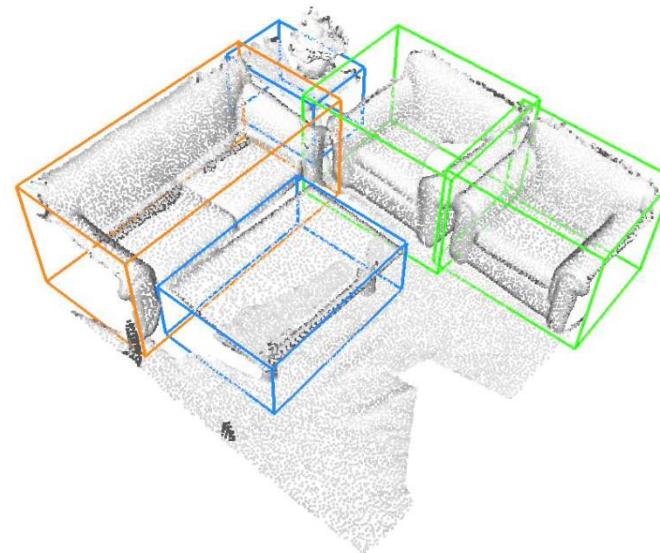


VoteNet

Voting from input point cloud

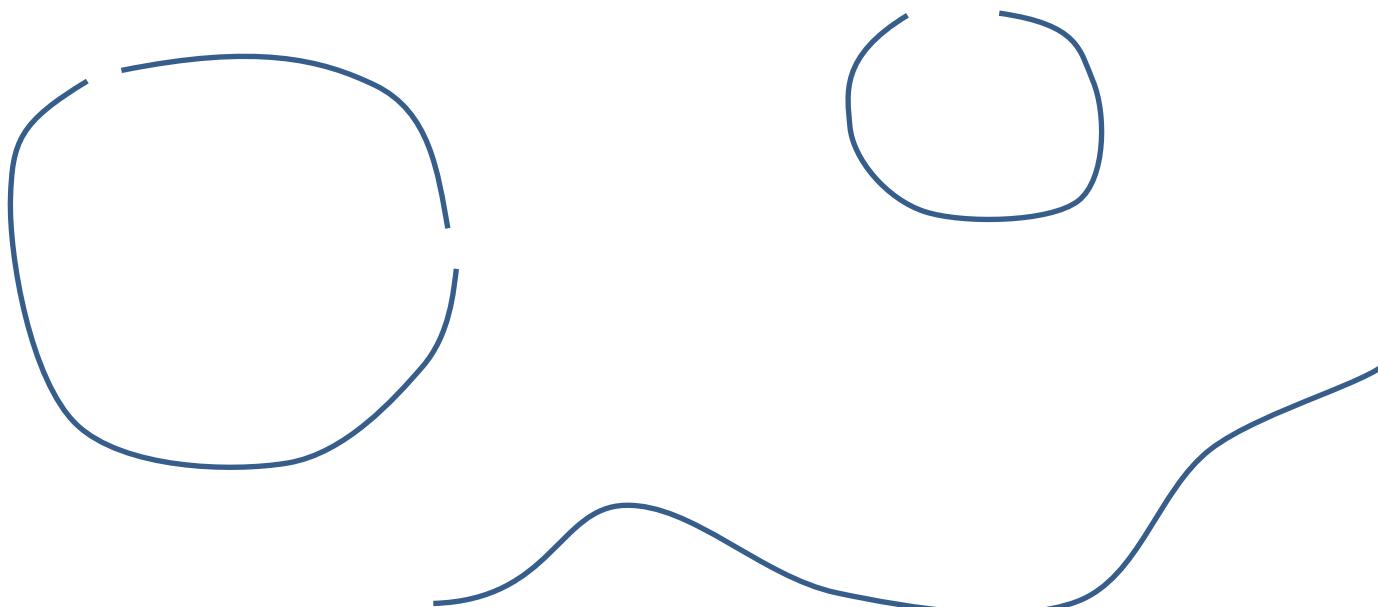


3D detection output

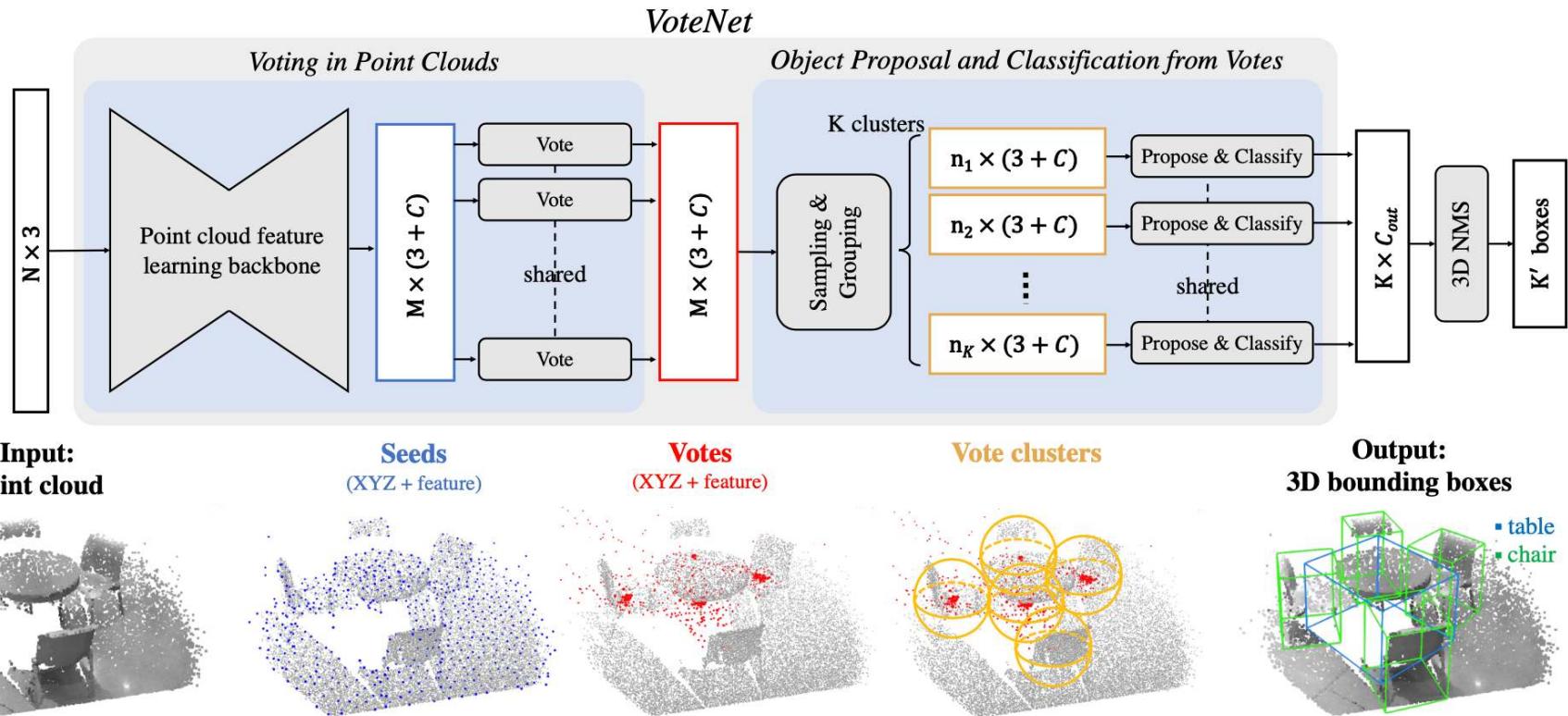


Qi, Charles R and Litany, Or and He, Kaiming and Guibas, Leonidas J}, title = {Deep Hough Voting for 3D Object Detection in Point Clouds}, booktitle = {Proceedings of the IEEE International Conference on Computer Vision}, year = {2019}

Hough Transform (not Deep Learning)

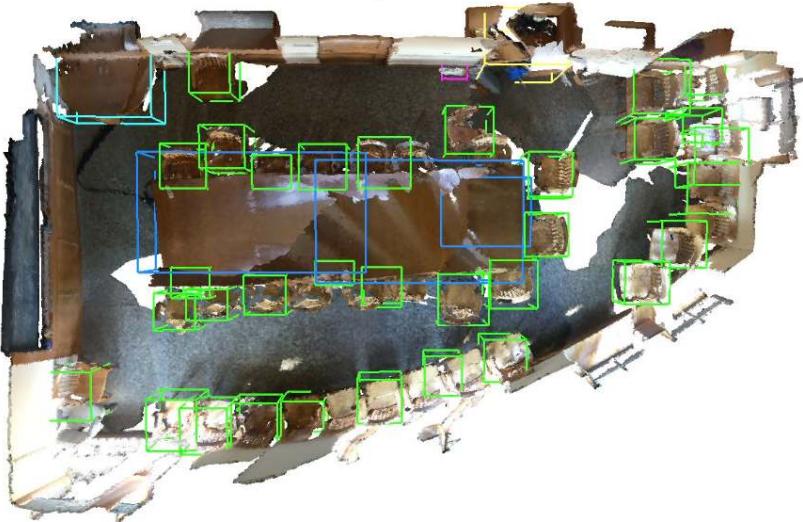


VoteNet

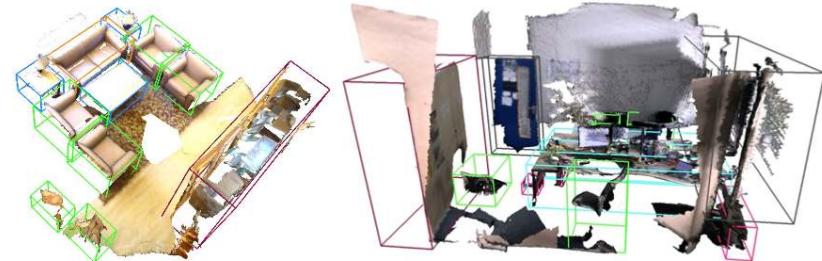
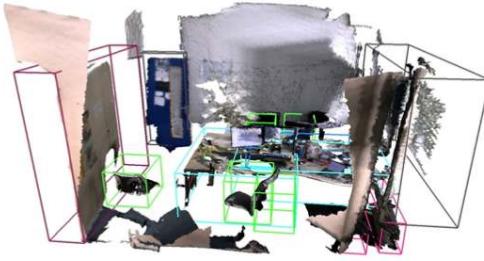
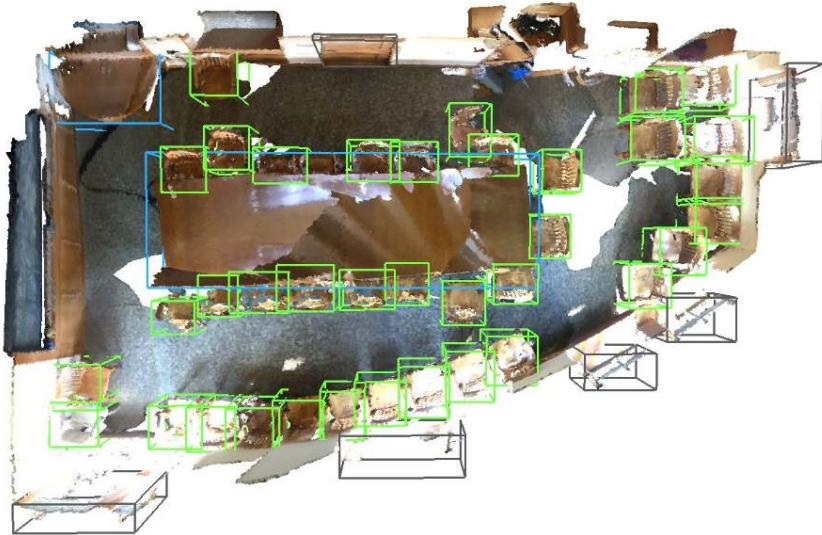


VoteNet: Some results

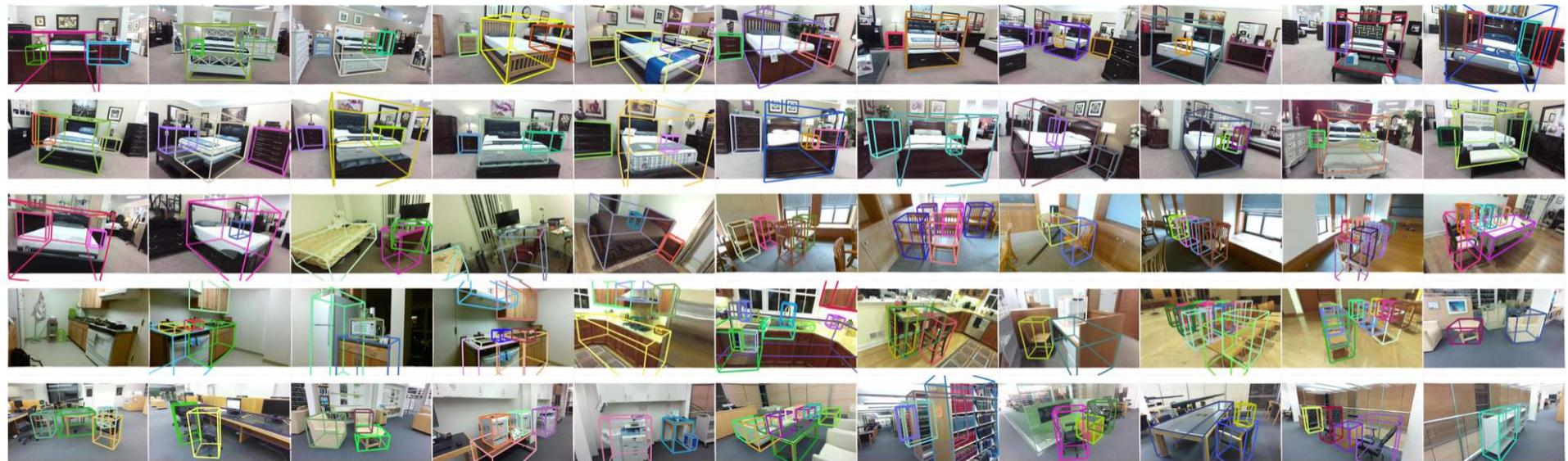
VoteNet prediction



Ground truth

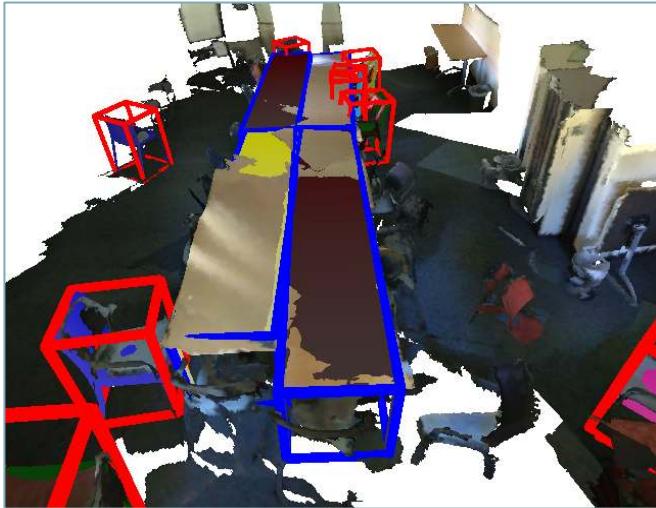


But where do we get training data?

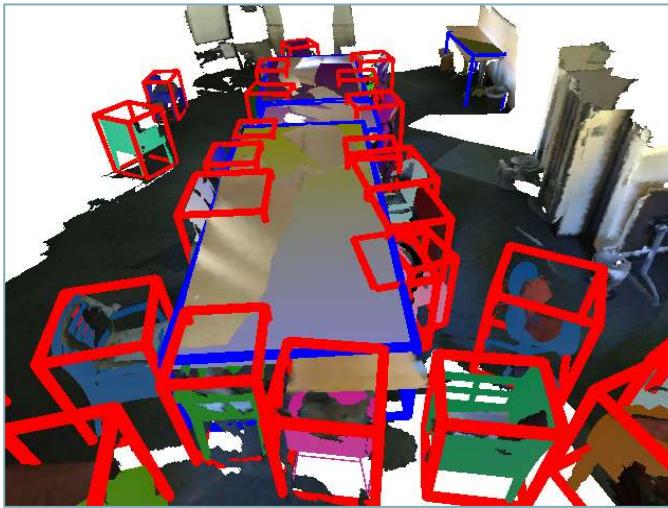


SUN-RGBD dataset, ~10'000 images annotated manually,
2000+ hours for initial annotations + time for error correction

Manual annotations are prone to errors



Manual annotations for ScanNet



Our automated annotations

Synthetic images?



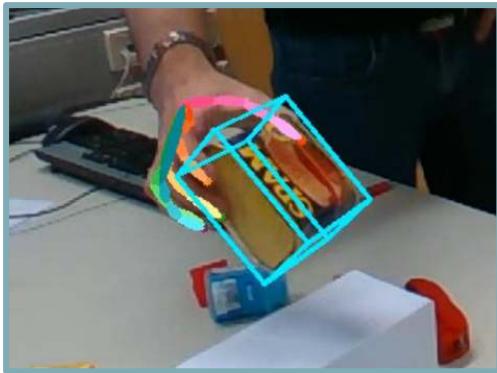
About 71'000 synthetic images. **Costs \$57K to create** (scene creation + image rendering), and took 231 vCPU years (**2.4 years of wall-clock time** on a large compute node).

[Mike Roberts and Nathan Paczan. Hypersim: A Photorealistic Synthetic Dataset for Holistic Indoor Scene Understanding. In *arXiv*, 2020]

What about new scenarios?

Do we want to go to the process of creating training data over and over?
Different countries, cultures have different indoor styles.

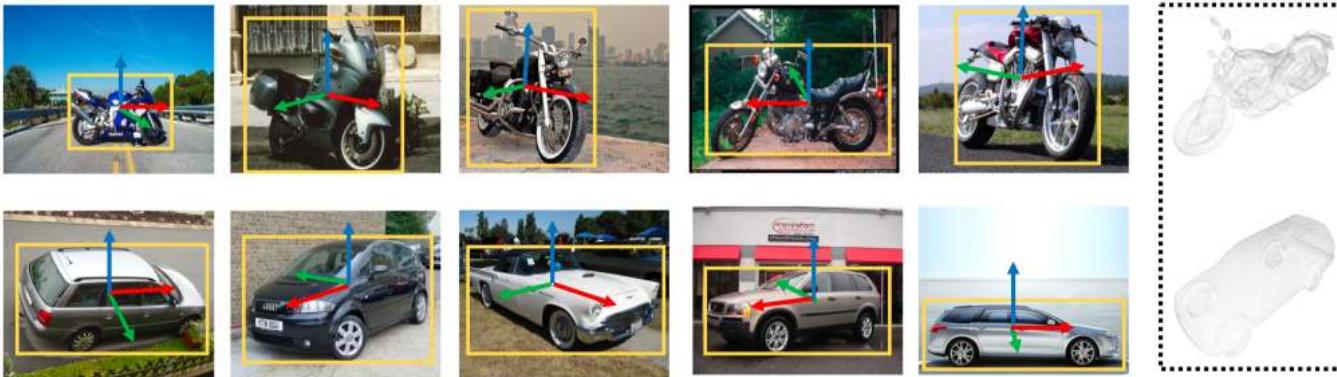
What about new types of scenes?



few-shot learning for 3D scene understanding

few-shot learning for 3D scene understanding

Training Examples (Novel Classes)

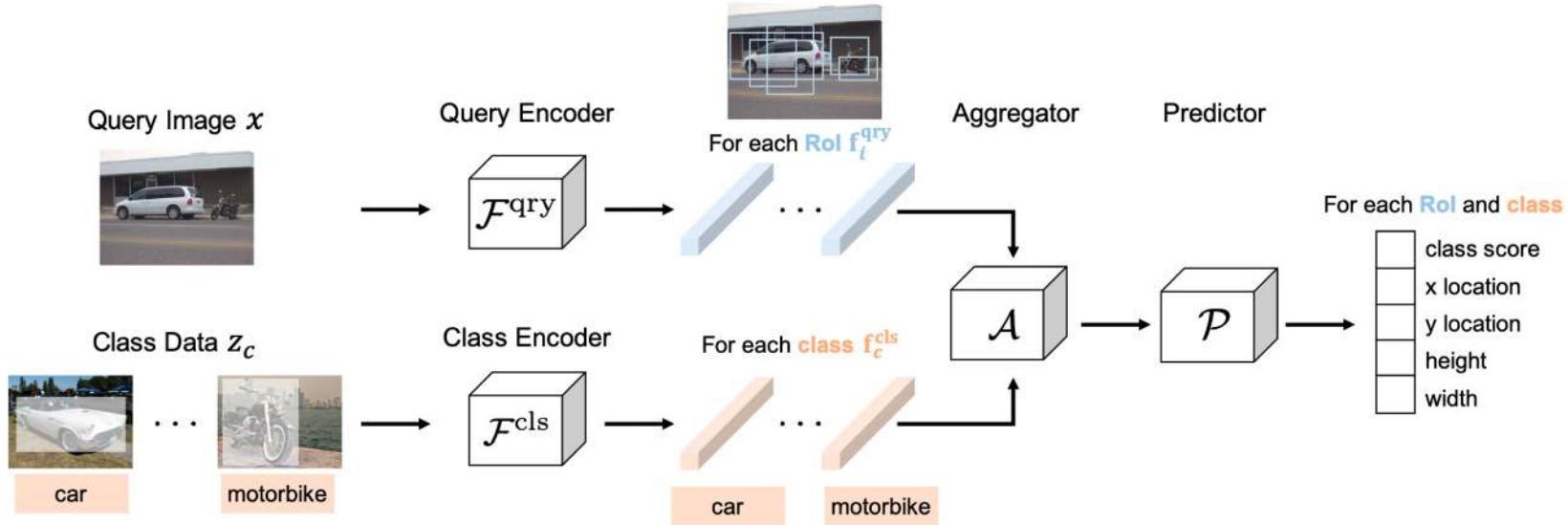


Testing



Few-shot Object Detection and Viewpoint Estimation for Objects in the Wild. Yang Xiao, Vincent Lepetit, Renaud Marlet. arXiv 2020.

few-shot learning for 3D scene understanding

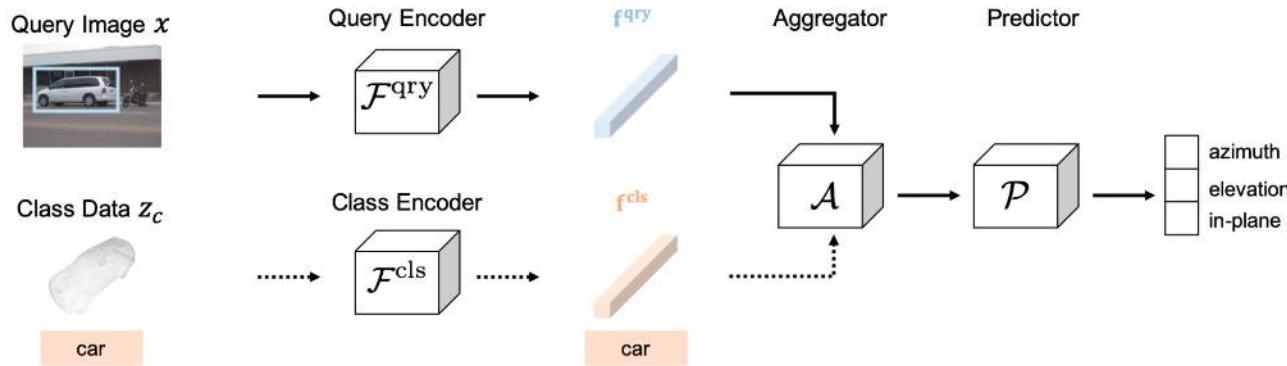


$$\text{cls}_{i,c} = \frac{\alpha \mathcal{A}(f_i^{\text{qry}}, f_c^{\text{cls}})^T w_c}{\|\mathcal{A}(f_i^{\text{qry}}, f_c^{\text{cls}})\| \|w_c\|}$$

$$\mathcal{A}(f^{\text{qry}}, f^{\text{cls}}) = f^{\text{qry}} \otimes f^{\text{cls}}$$

Few-shot Object Detection and Viewpoint Estimation for Objects in the Wild. Yang Xiao, Vincent Lepetit, Renaud Marlet. arXiv 2020.

few-shot learning for 3D scene understanding



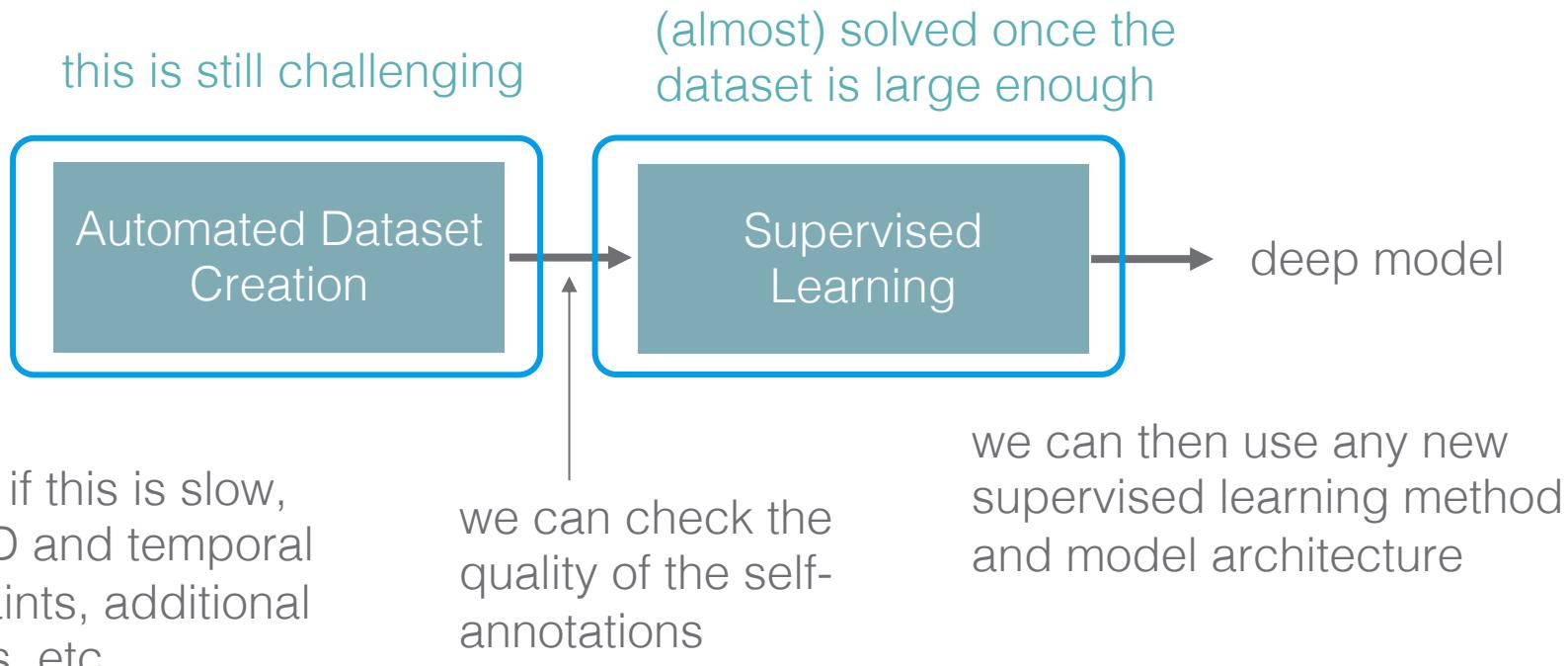
$$(\text{azi}, \text{ele}, \text{inp}) = \mathcal{P}(\mathcal{A}(f^{\text{qry}}, f^{\text{cls}}))$$

with $f^{\text{qry}} = \mathcal{F}^{\text{qry}}(\text{crop}(\text{img}(x), \text{box}(x)))$, and

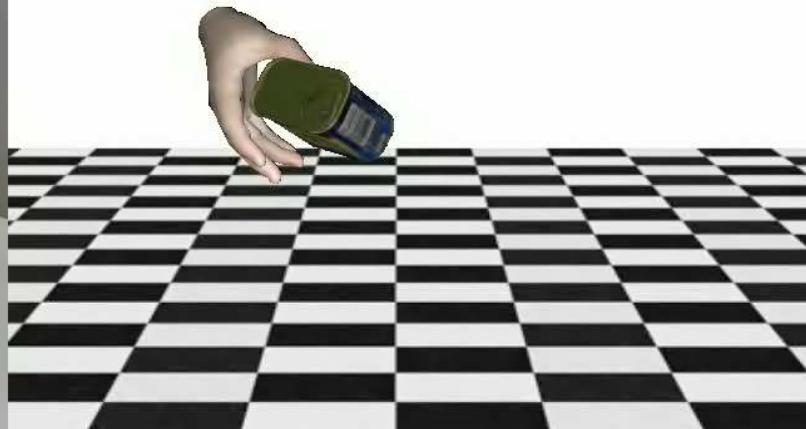
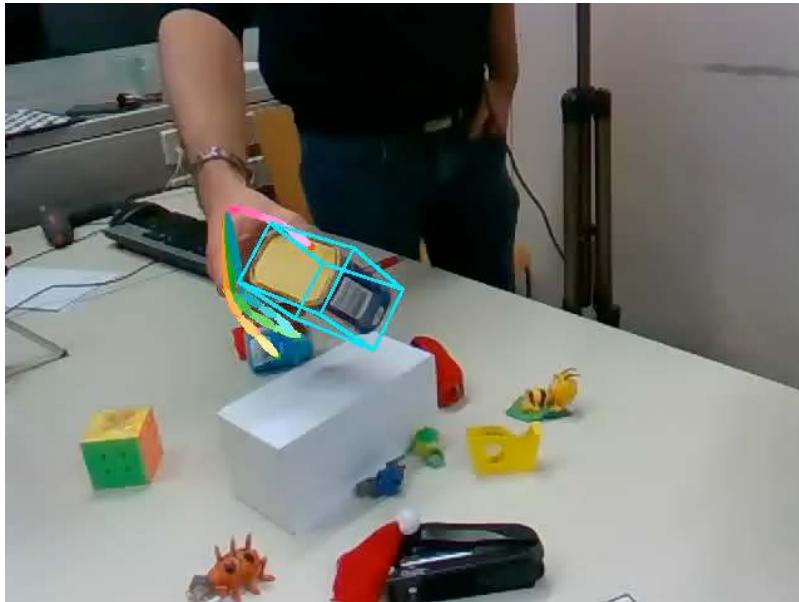
$$f^{\text{cls}} = \mathcal{F}^{\text{cls}}(z_c), c = \text{cls}(x)$$

generating 3D labels automatically

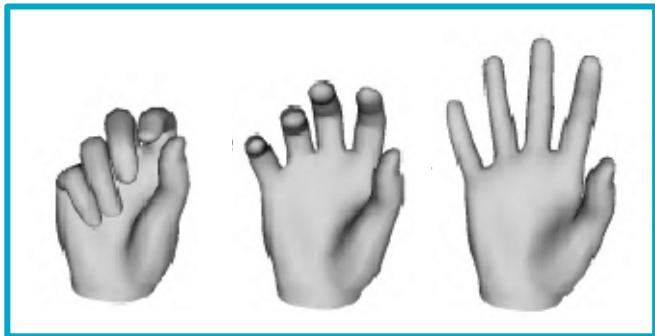
generating 3D labels automatically



Example 1: Automated 3D annotations of a hand manipulating an object



automated 3D annotations of a hand manipulating an object



MANO model
[Romero et al, 2017]



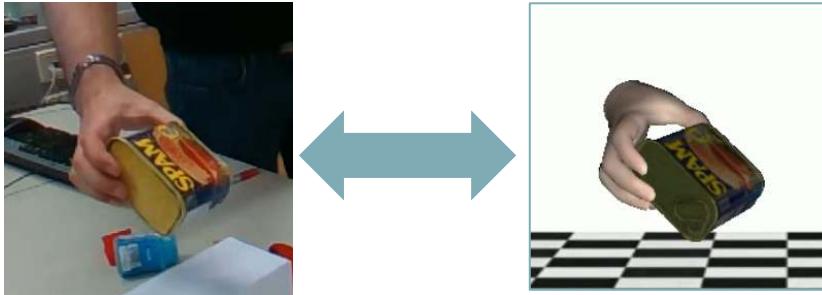
Object 3D model
(YCB dataset)



how can we handle the large hand/object occlusions?

Analysis-by-synthesis and Bayesian framework

$$\max_{\{(\mathbf{p}_t^H, \mathbf{p}_t^O)\}_t} \prod_t \prod_c p((I_t^c, D_t^c) \mid \mathbf{p}_t^H, \mathbf{p}_t^O) p(\mathbf{p}_{t+1}^H, \mathbf{p}_{t+1}^O \mid \mathbf{p}_t^H, \mathbf{p}_t^O) p(\mathbf{p}_t^H, \mathbf{p}_t^O)$$



RGB-D likelihoods

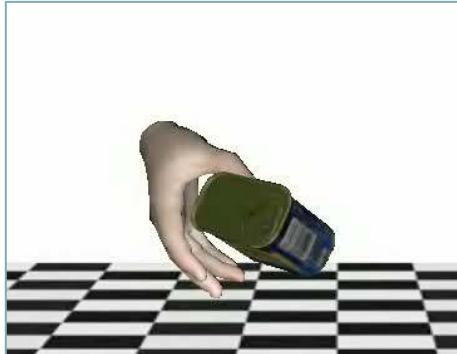
$$\max_{\{(\mathbf{p}_t^H, \mathbf{p}_t^O)\}_t} \prod_t \prod_c \overline{p((I_t^c, D_t^c) \mid \mathbf{p}_t^H, \mathbf{p}_t^O)} p(\mathbf{p}_{t+1}^H, \mathbf{p}_{t+1}^O \mid \mathbf{p}_t^H, \mathbf{p}_t^O) p(\mathbf{p}_t^H, \mathbf{p}_t^O)$$



RGB-D likelihoods

$$\max_{\{(\mathbf{p}_t^H, \mathbf{p}_t^O)\}_t} \prod_t \prod_c \frac{\overline{p((I_t^c, D_t^c) \mid \mathbf{p}_t^H, \mathbf{p}_t^O)}}{p(\mathbf{p}_{t+1}^H, \mathbf{p}_{t+1}^O \mid \mathbf{p}_t^H, \mathbf{p}_t^O)} p(\mathbf{p}_t^H, \mathbf{p}_t^O)$$

temporal constraints

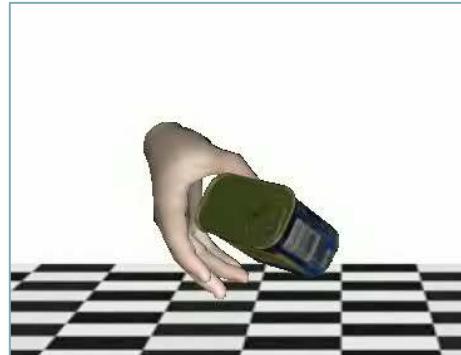




there should not be any intersection between the hand and the object

RGB-D likelihoods

$$\max_{\{(\mathbf{p}_t^H, \mathbf{p}_t^O)\}_t} \prod_t \prod_c \overline{p((I_t^c, D_t^c) \mid \mathbf{p}_t^H, \mathbf{p}_t^O)} \frac{p(\mathbf{p}_{t+1}^H, \mathbf{p}_{t+1}^O \mid \mathbf{p}_t^H, \mathbf{p}_t^O)}{\text{temporal constraints}} \overline{p(\mathbf{p}_t^H, \mathbf{p}_t^O)}$$



Full sequence optimization



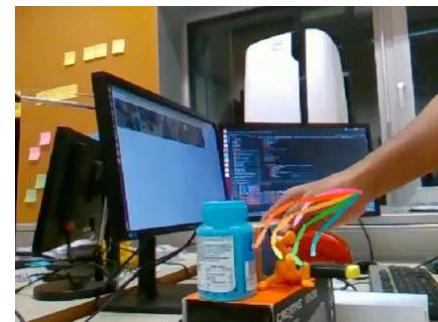
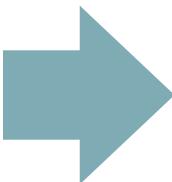
$$\max_{\{(\mathbf{p}_t^H, \mathbf{p}_t^O)\}_t} \prod_t \prod_c p((I_t^c, D_t^c) \mid \mathbf{p}_t^H, \mathbf{p}_t^O) p(\mathbf{p}_{t+1}^H, \mathbf{p}_{t+1}^O \mid \mathbf{p}_t^H, \mathbf{p}_t^O) p(\mathbf{p}_t^H, \mathbf{p}_t^O)$$







dataset of annotated sequences



single image hand/object pose estimation [5ms]



going back to indoor scenes

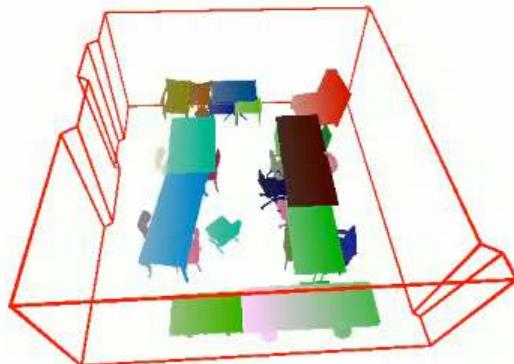


automated indoor annotations

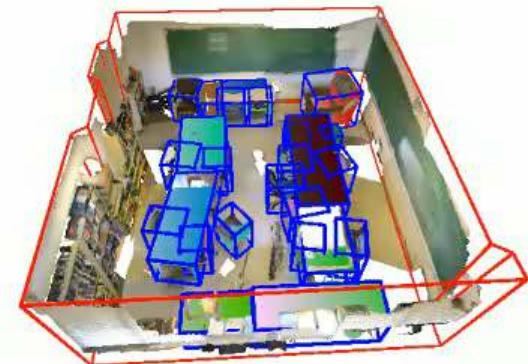
RGBD scan



automated labels



automated labels
with scan



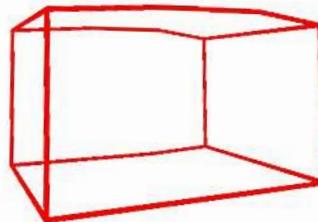
[Monte Carlo Scene Search for 3D Scene Understanding. Shreyas Hampali, Sinisa Stekovic, Sayan Deb Sarkar, Chetan Srinivasa Kumar, Friedrich Fraundorfer, and Vincent Lepetit. CVPR 2021]

Comparisons with Manual Annotations - Walls

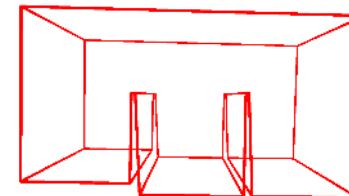
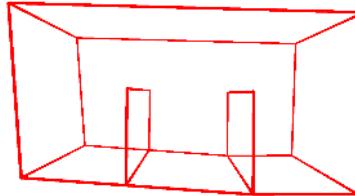
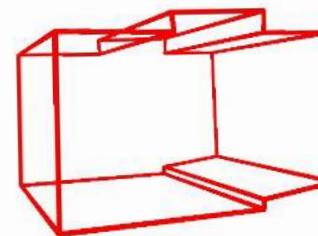
RGB-D scan



manual annotations [2]



our output



[2] Armen Avetisyan, Tatiana Khanova, Christopher Choy, Denver Dash, Angela Dai, and Matthias Nießner. SceneCAD: Predicting Object Alignments and Layouts in RGB-D Scans. In ECCV 2020.

Comparisons with Manual Annotations - Objects

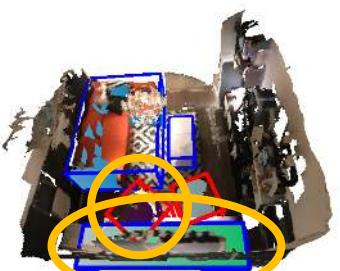
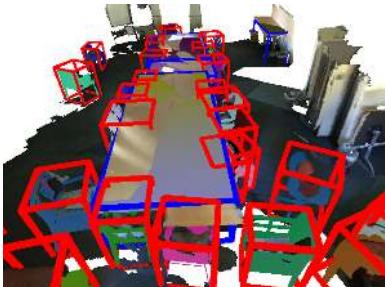
RGBD scan



manual
annotations [1]

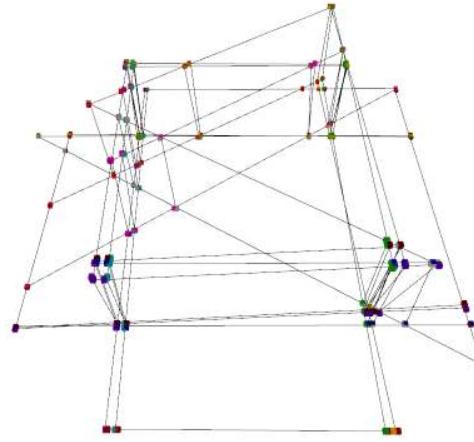


our results

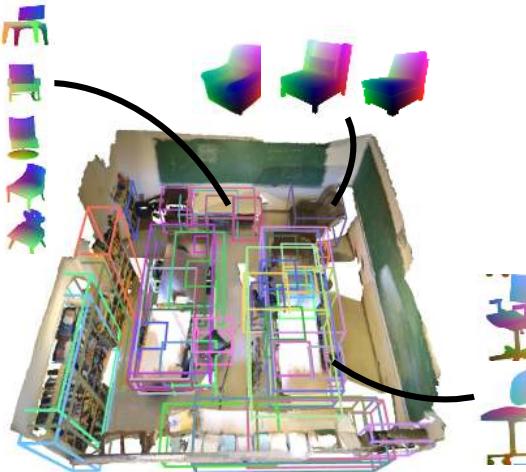


General Idea

Step #1: Make proposals



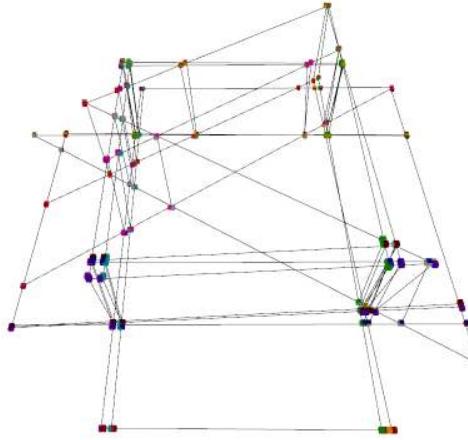
for the walls and..



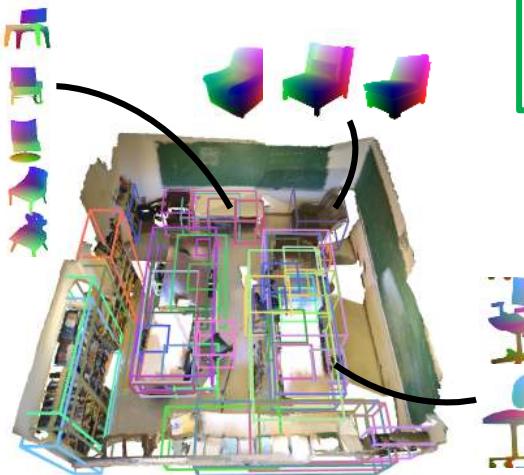
..for the objects

General Idea

Step #1: Make proposals



for the walls and..

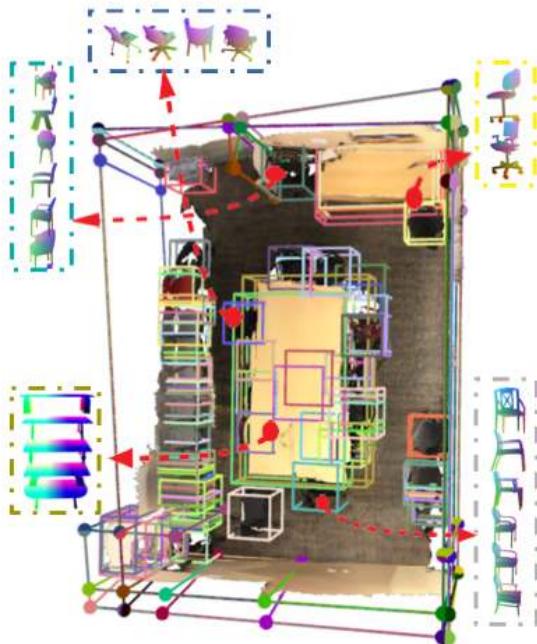


this step does not have to be perfect,
this is actually easy to do because the next
step will filter the false positives!

..for the objects

General Idea

Step #2: Select the correct proposals

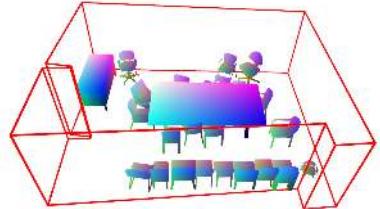


How can we select the
correct proposals?



analysis-by-synthesis and Bayesian framework

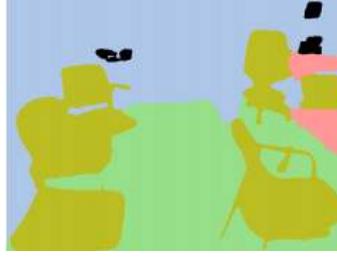
Selected proposals \mathcal{O}



rendered segmentation



image segmentation



rendered depth

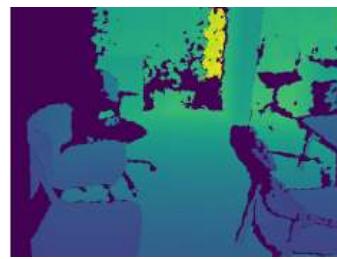
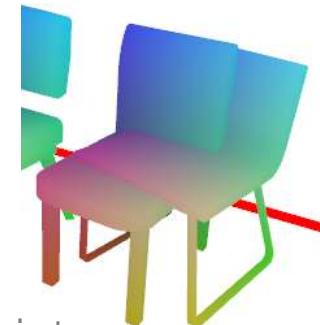


image depth



$$\max_{\mathcal{O}} \prod_i \frac{p((I_i, D_i) | \mathcal{O})}{p(\mathcal{O})}$$

physical constraints



optimization

$$\max_{\mathcal{O}} \prod_i p((I_i, D_i) \mid \mathcal{O}) p(\mathcal{O})$$

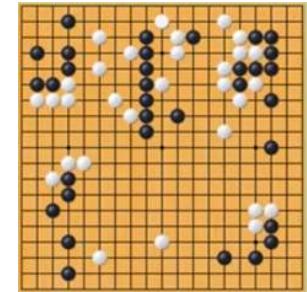
How do we perform this optimization? If we have 100 proposals, an exhaustive search would require 2^{100} ($\sim 10^{12}$) evaluations!

Standard tree search algorithms do not scale to such large trees.

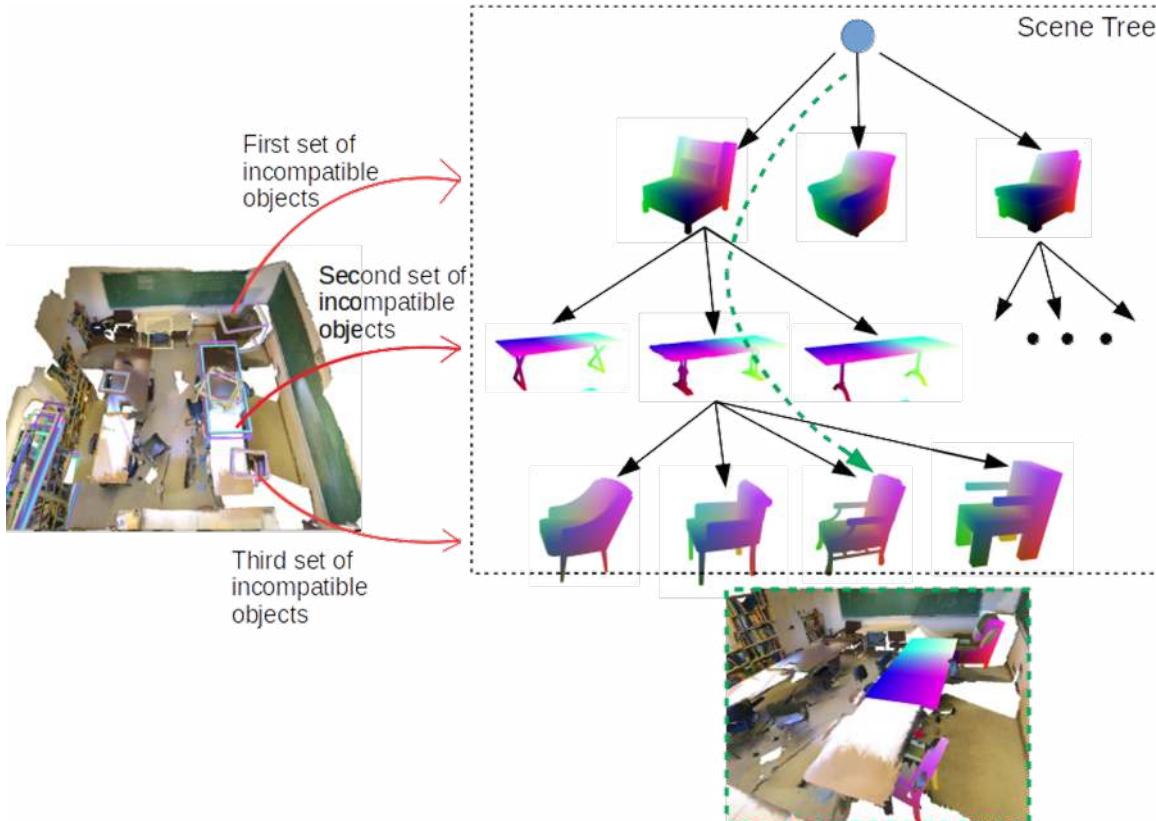
use Monte Carlo Tree Search (MCTS) to find the solution efficiently.

Monte Carlo Tree Search

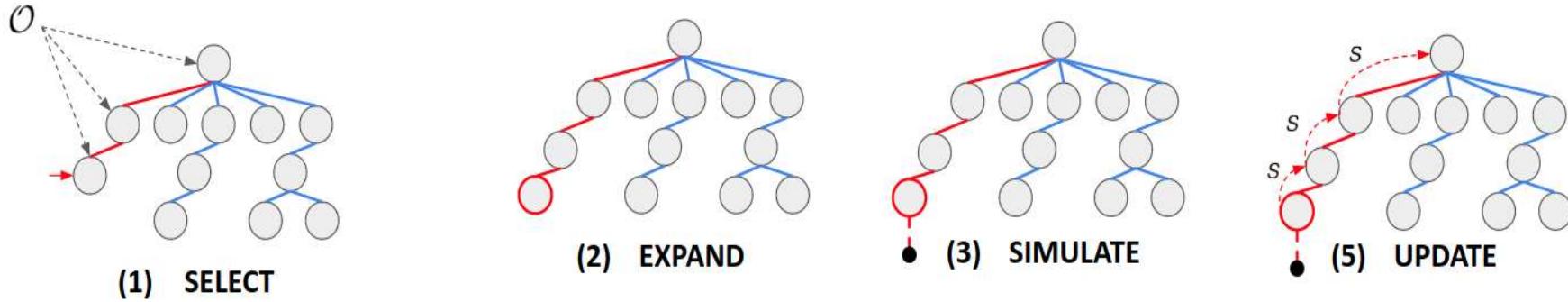
- Originates from the work of Bruce Abramson in the 1980's;
- Combined with Deep Learning by DeepMind in 2016 to play Go.
- No heuristics, exploration based on the objective.



Tree Structure



MCTS



selection is usually based on the UCB score:

$$Q_t(n) + c \sqrt{\frac{\log t}{N_t(n)}}$$

principle of “optimism in the face of uncertainty”

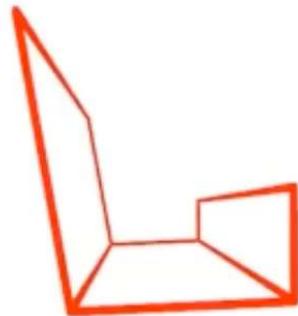
Optimisation

Input Scan

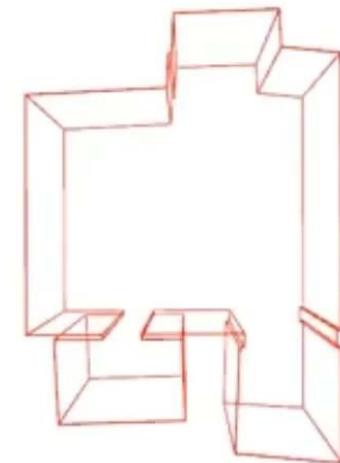


MCSS Output

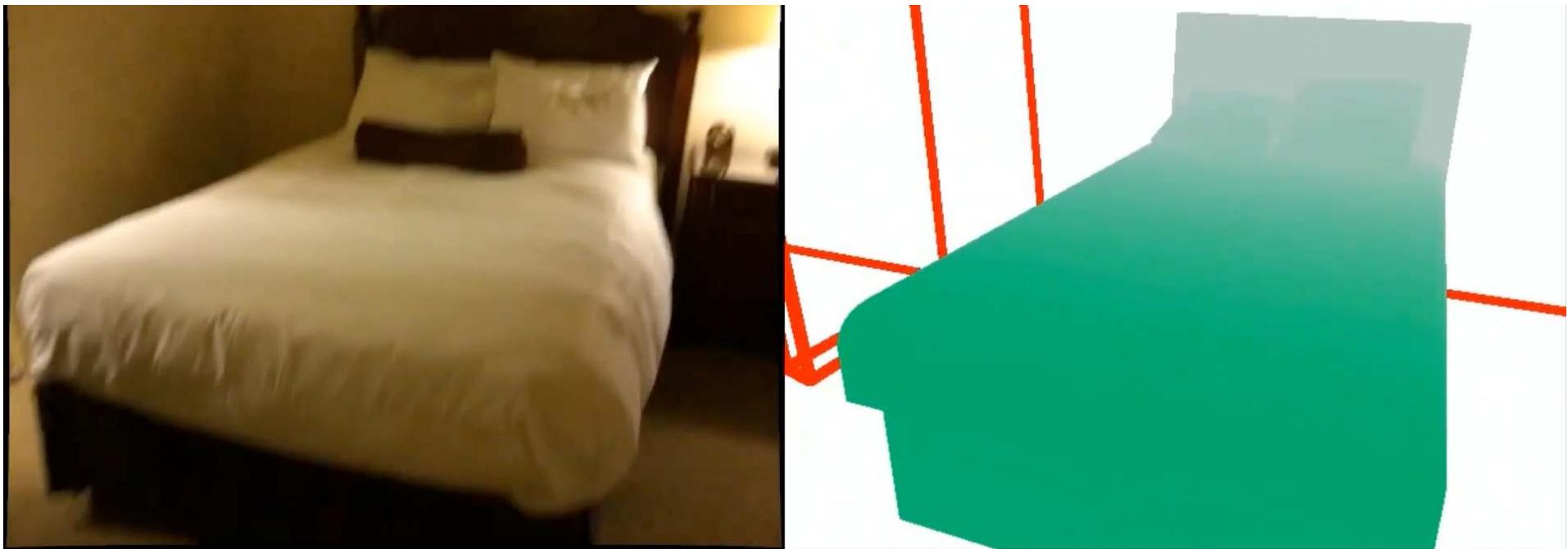
Iteration - 0



Our Manual Annotation



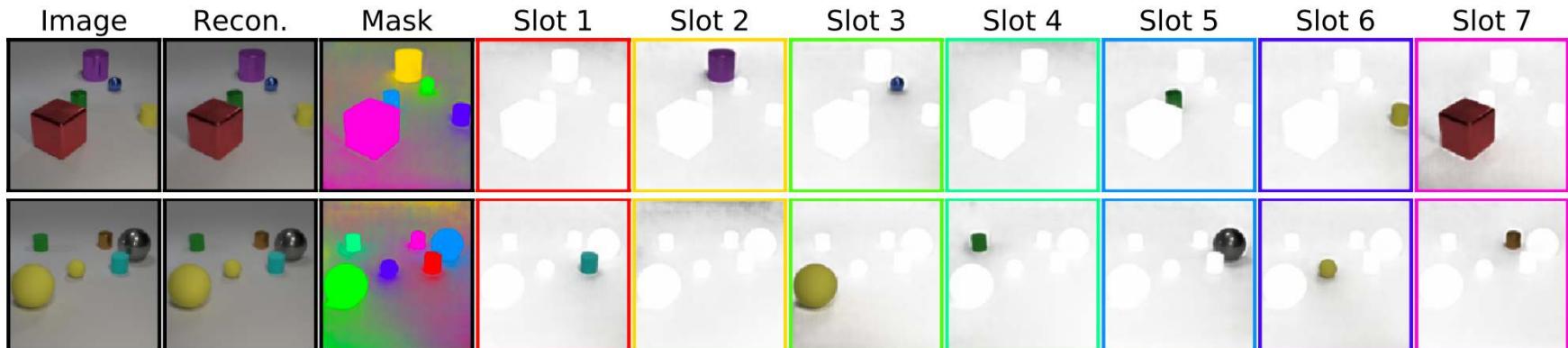
Results [ScanNet]



object discovery

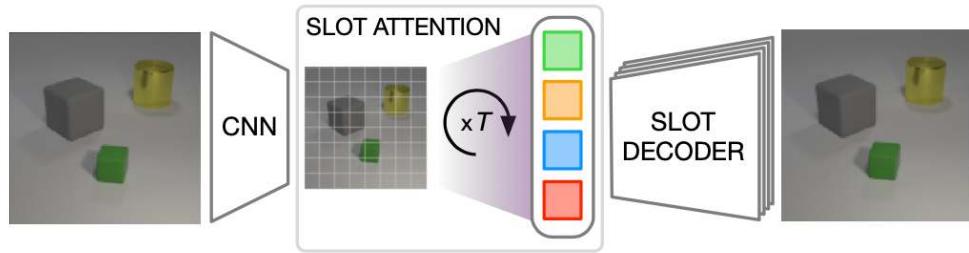
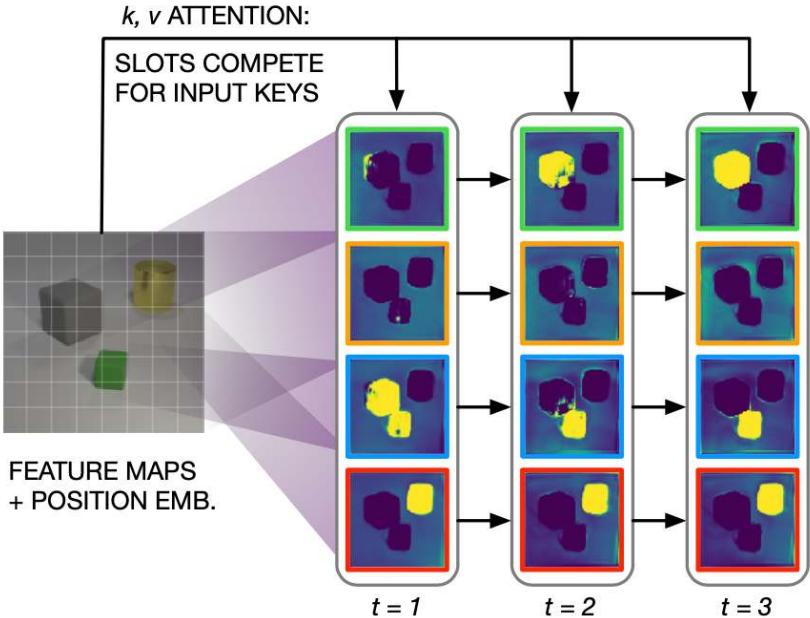
(the future of 3D scene
understanding?)

Slot Attention



Object-Centric Learning with Slot Attention. Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. NeurIPS 2020.

Slot Attention



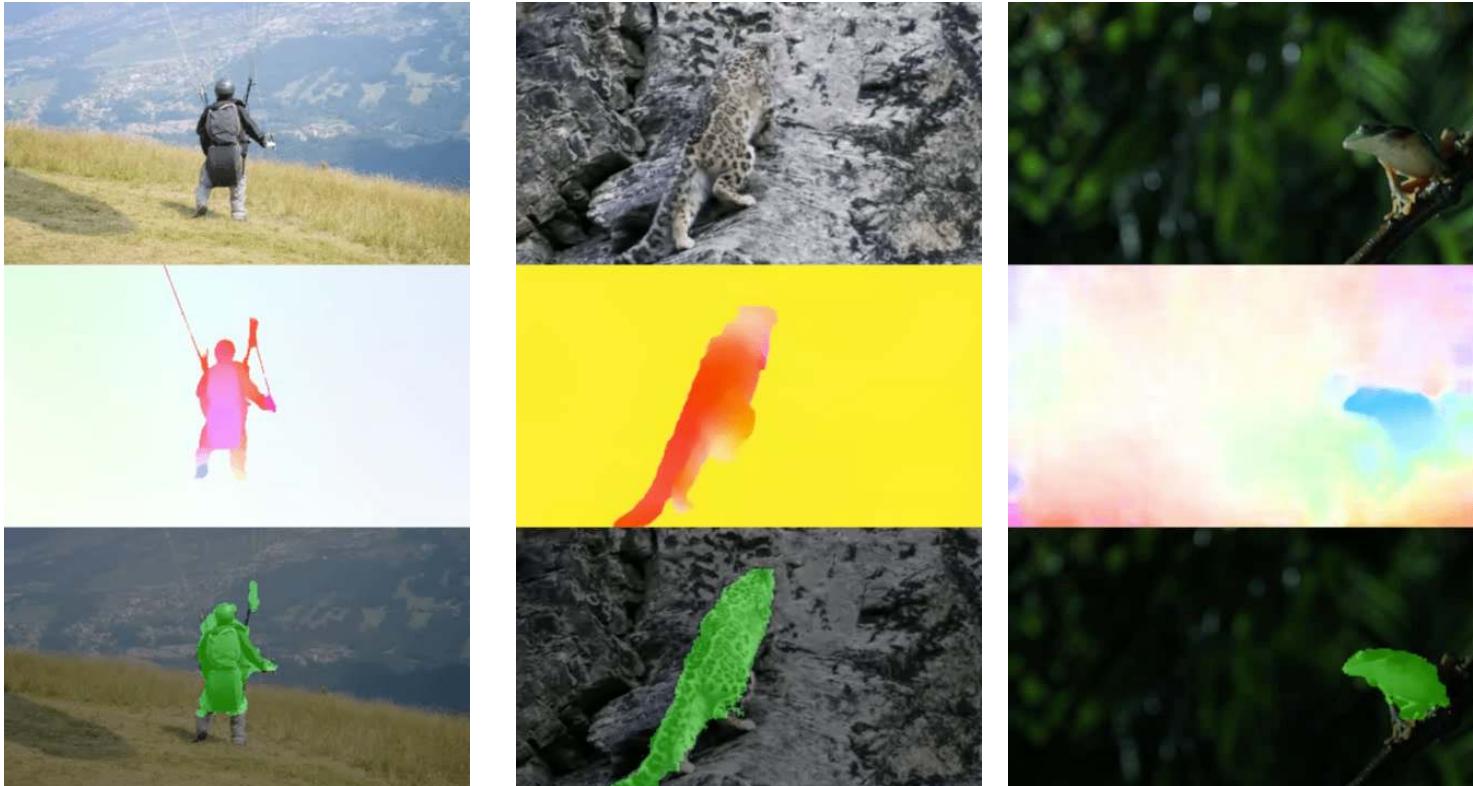
Algorithm 1 Slot Attention module. The input is a set of N vectors of dimension D_{inputs} which is mapped to a set of K slots of dimension D_{slots} . We initialize the slots by sampling their initial values as independent samples from a Gaussian distribution with shared, learnable parameters $\mu \in \mathbb{R}^{D_{\text{slots}}}$ and $\sigma \in \mathbb{R}^{D_{\text{slots}}}$. In our experiments we set the number of iterations to $T = 3$.

```

1: Input:  $\text{inputs} \in \mathbb{R}^{N \times D_{\text{inputs}}}$ ,  $\text{slots} \sim \mathcal{N}(\mu, \text{diag}(\sigma)) \in \mathbb{R}^{K \times D_{\text{slots}}}$ 
2: Layer params:  $k, q, v$ : linear projections for attention; GRU; MLP; LayerNorm(x3)
3:    $\text{inputs} = \text{LayerNorm}(\text{inputs})$ 
4:   for  $t = 0 \dots T$ 
5:      $\text{slots\_prev} = \text{slots}$ 
6:      $\text{slots} = \text{LayerNorm}(\text{slots})$ 
7:      $\text{attn} = \text{Softmax}(\frac{1}{\sqrt{D}} k(\text{inputs}) \cdot q(\text{slots})^T, \text{axis}=\text{'slots'})$       # norm. over slots
8:      $\text{updates} = \text{WeightedMean}(\text{weights}=\text{attn} + \epsilon, \text{values}=v(\text{inputs}))$       # aggregate
9:      $\text{slots} = \text{GRU}(\text{state}=\text{slots\_prev}, \text{inputs}=\text{updates})$           # GRU update (per slot)
10:     $\text{slots} += \text{MLP}(\text{LayerNorm}(\text{slots}))$           # optional residual MLP (per slot)
11:   return  $\text{slots}$ 

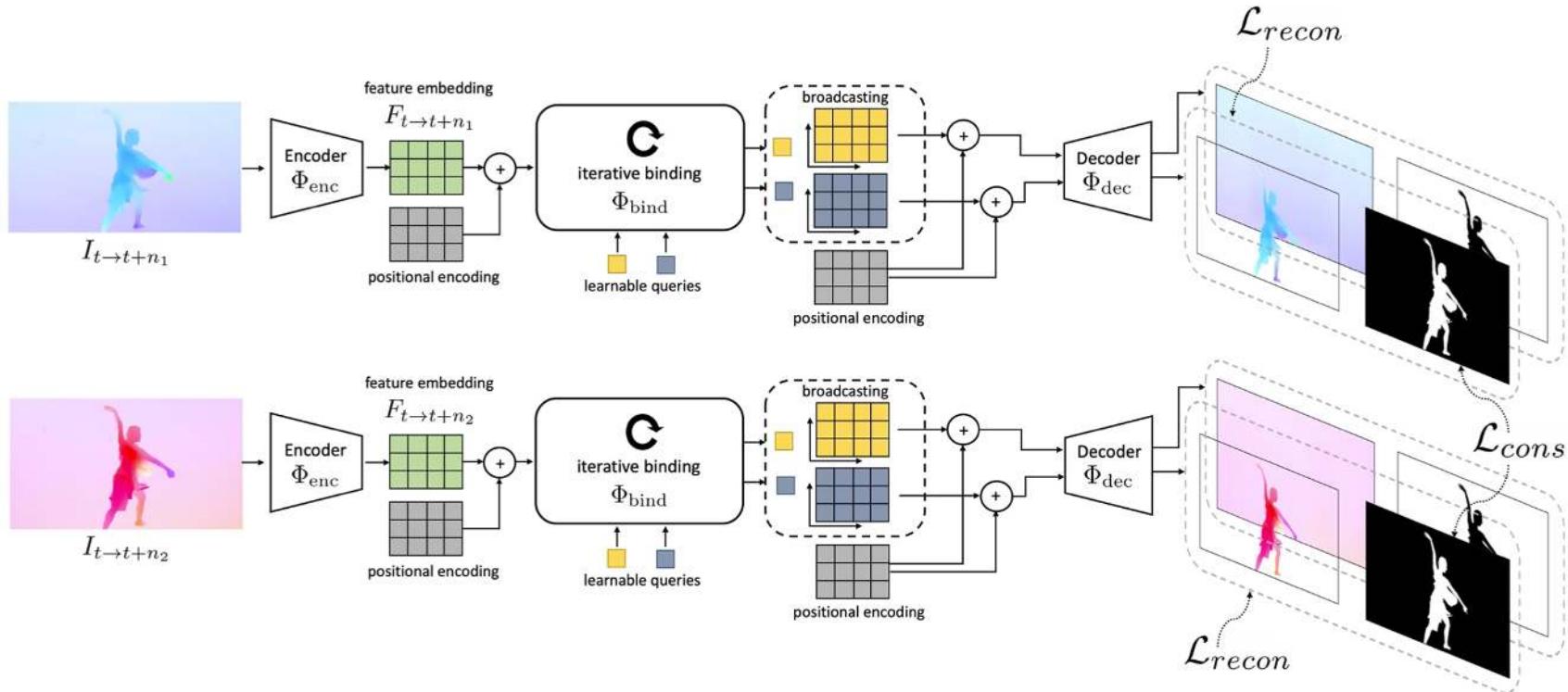
```

Motion Grouping



Self-supervised Video Object Segmentation by Motion Grouping. Charig Yang, Hala Lamdouar, Erika Lu, Andrew Zisserman, Weidi Xie. ICCV 2021.

Motion grouping



Conclusion

Part #1:

- 3D object instance understanding;
- 3D pose prediction and training;

Part #2

- 3D object category understanding and 3D model prediction;
- 3D model prediction from images;

Part #3:

- Few-shot learning for 3D scene understanding;
- Point-cloud analysis;
- Self-learning for 3D scene understanding;
- Object discovery.