Object Pose Estimation by Render and Compare

Siyu ZHANG

Senior UG, SUSTech

Research Intern, 3DV-Research@SenseTime & ZJU CAD

Background & Motivation

- Problem to solve: Estimate the 6DoF pose of objects
- Typical approach
  - Directly regress translation & rotation
  - Estimate key point coordinates and then solve PnP
  - Pose estimation (refinement) by render and compare

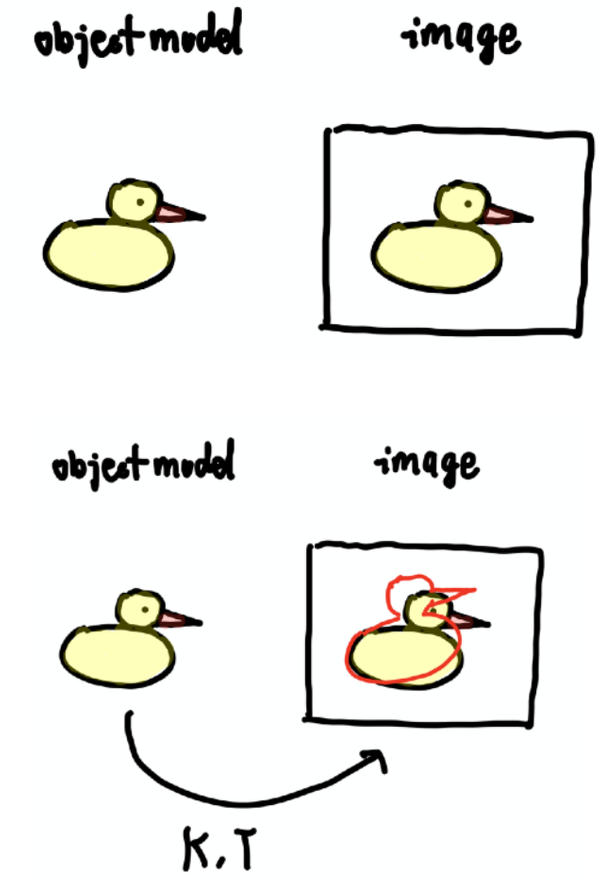
Background & Motivation

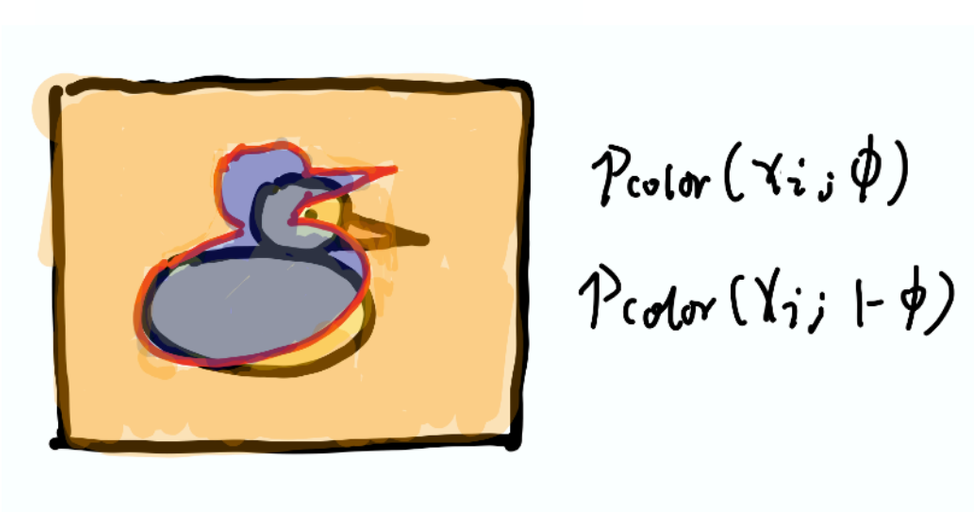
- Input:
  - Image
  - Initial pose (previous frame observation or coarse regressed pose)
  - object model (depends)

- Typical paradigm:
  - Reconstruct Object (depends)
  - Optimize for a better pose:
    - ‘Render’ the object onto the image.
    - Compare rendered result with image cue, compute the residual accordingly for pose optimization.

Background & Motivation

- Example of render & compare pipeline: **Silhouette tracking**
  - Render the silhouette with known model and pose.
  - Convert the silhouette to level set function, evaluate it according to foreground-background similarity
  - Update pose by residual of silhouette.
- Reference: PWP3D

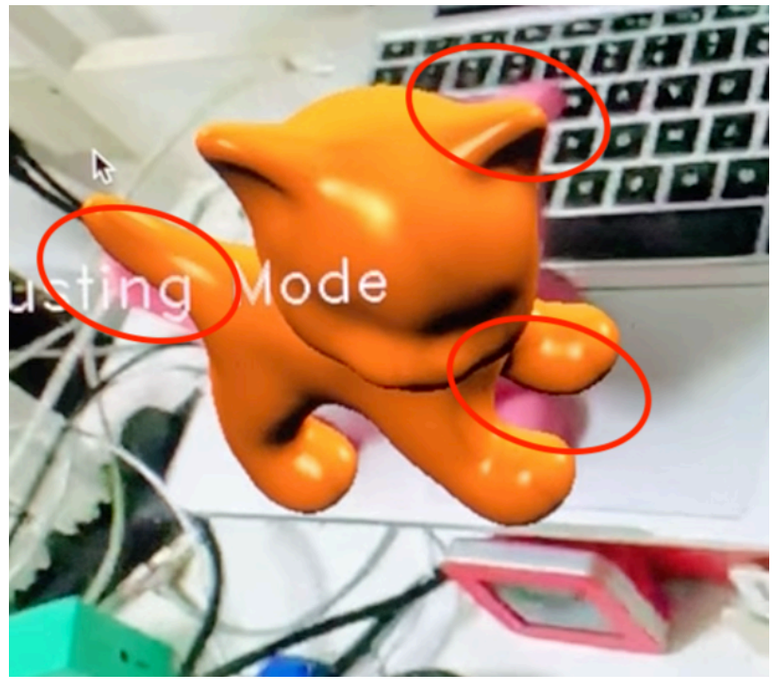




Background & Motivation

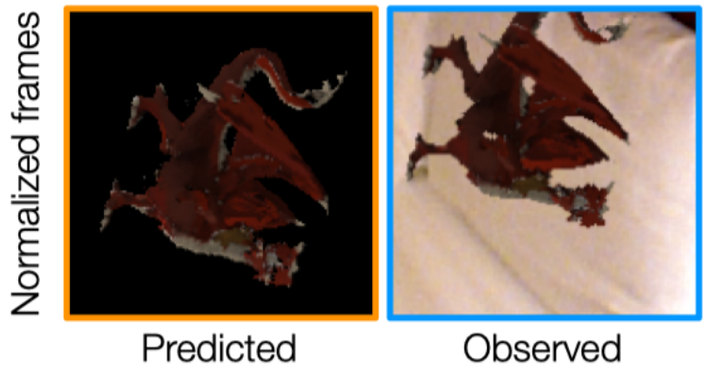
- Can we make it **DEEP**?
  - Color cues are unreliable and suffers from shape ambiguity.

- Can we make it **MODEL FREE**?
  - Accurate mesh model is hard to obtained in real world application.
  - Inaccurate model produces disastrous results for silhouette tracking.



Background & Motivation

- Simple (and naive) approach:
  - Render object with estimated pose
  - Feed rendered image and real image into Neural Network, directly regress relative pose.



Input: $\mathbf{x}_{\text{pred}}$	Input: $\mathbf{x}_{\text{obs}}$
conv3-96	conv3-96
fire-48-96	fire-48-96
concatenation	
fire-96-384	
fire-192-768	
fire-384-768	
FC-500	
FC-6	
Output: $\mathbf{y}$	

- Reference:
  - Deep 6DoF Tracking (*TVCG 2017*)
  - A Framework for Evaluating 6-DOF Object Trackers （*ECCV 2018*)

Background & Motivation

- Simple (and naive) approach:
  - Render object with estimated pose
  - Feed rendered image and real image into Neural Network, directly regress relative pose.

- Reference:
  - Deep 6DoF Tracking (*TVCG 2017*)
  - A Framework for Evaluating 6-DOF Object Trackers （*ECCV 2018*)

Background & Motivation

- Going deeper
  - Reconstruct Object : Construct *feature-metric* model
  - Compared with feature extracted from current frame.
- Paper to share today:
  - LatentFusion:
    - Reconstruct feature voxel, render and compare with depth.
  - Refining 6D Object Pose Predictions using Abstract Render-and-Compare
    - Descriptor learning with contrastive loss.
    - Generate views synthetically and fuse feature to mesh model.

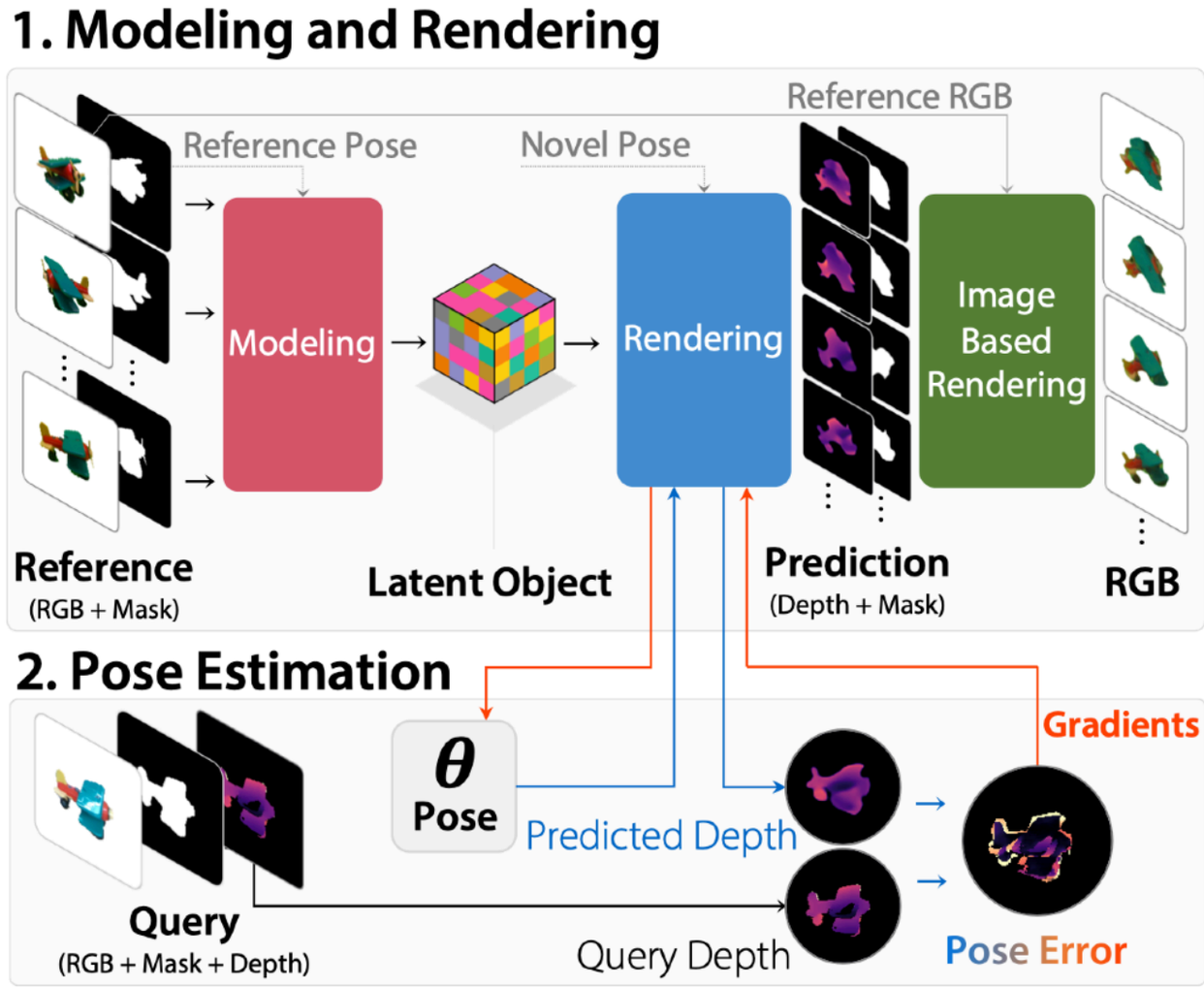
Background & Motivation

- LatentFusion
  - ‘Deep’ render & compare for object pose estimation.
  - Capability of handling unseen object without pre-training.
  - Model-free object pose estimation dataset.
- Refining 6D Object Pose Predictions using Abstract Render-and-Compare
  - Fusion surface feature (descriptors trained by contrastive loss) onto CAD model
  - Differentiable render and compare with feature

LatentFusion

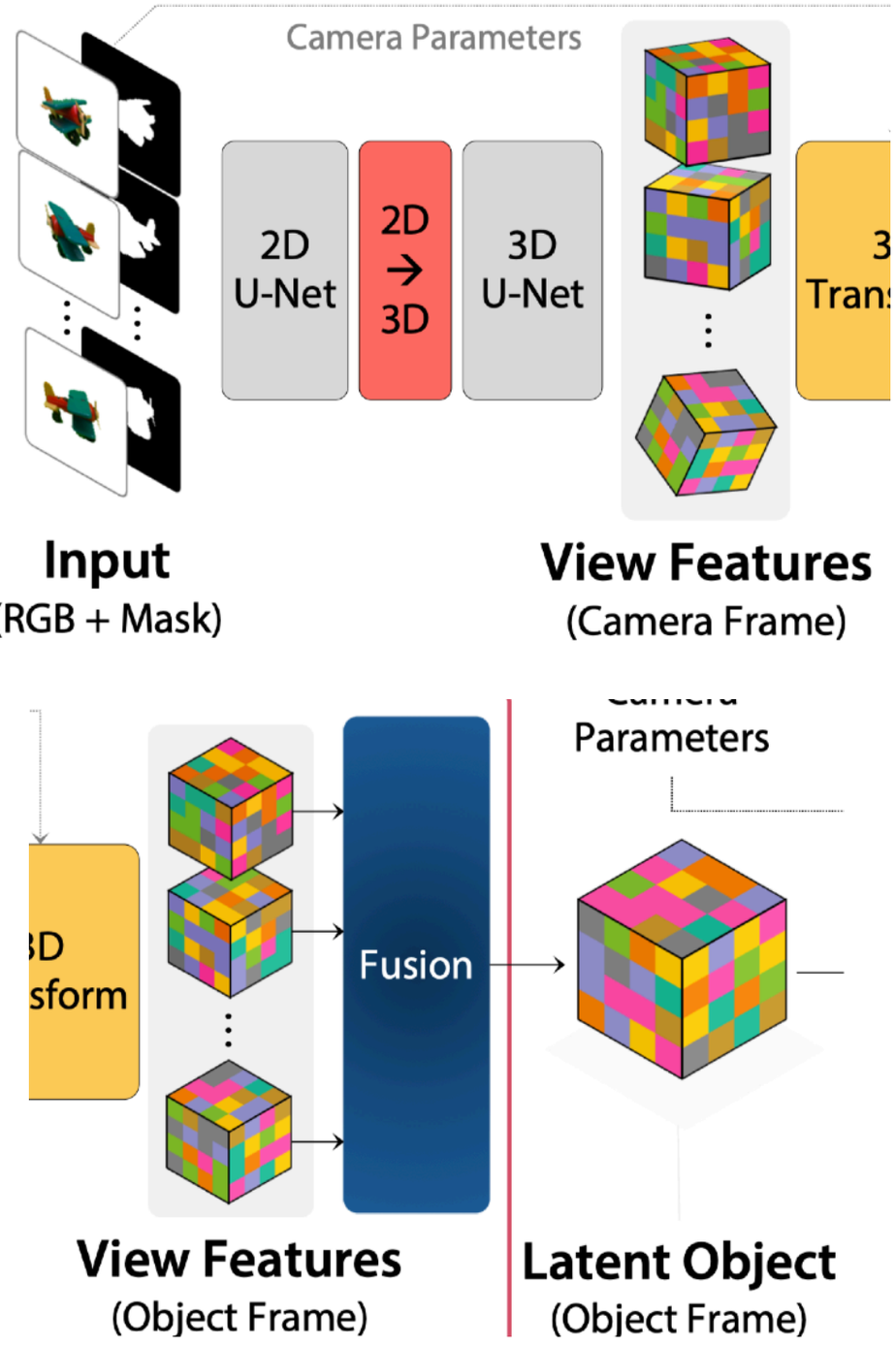
Methodology

- Overall pipeline
  - Reconstruct 3D feature voxel given reference frame (RGB + Mask + Camera pose)
  - Render & compare with depth error and latent feature error.



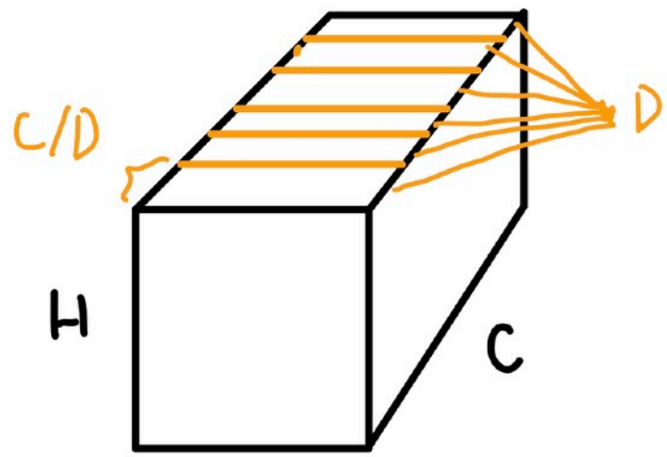
Methodology - Reconstruction

- Feature generation
  - RGB + U-Net = 2D feature map
  - Lift 2D feature map to 3D with de-projection
  - 3D U-Net to generate final 3D feature
- View Fusion
  - Transform from camera frame to object frame
  - View aggregation by computing channel mean or with ConvGRU



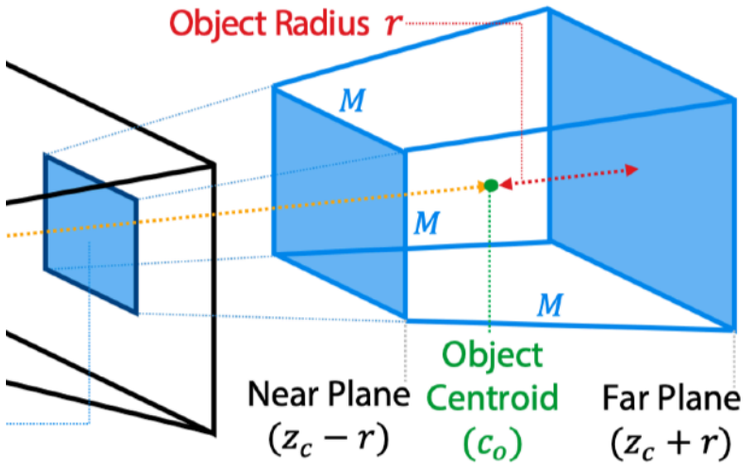
Methodology - Reconstruction

- A Little bit details…
- The Miracle 3D de-projection
  - 2D feature map: C x (H x W)
  - Lifted 3D feature map: C/D x (D x H x W)



- View Fusion - 3D Transformation
  - Generated voxel size: C' x M x M, not defined for 3D rigid body transformation

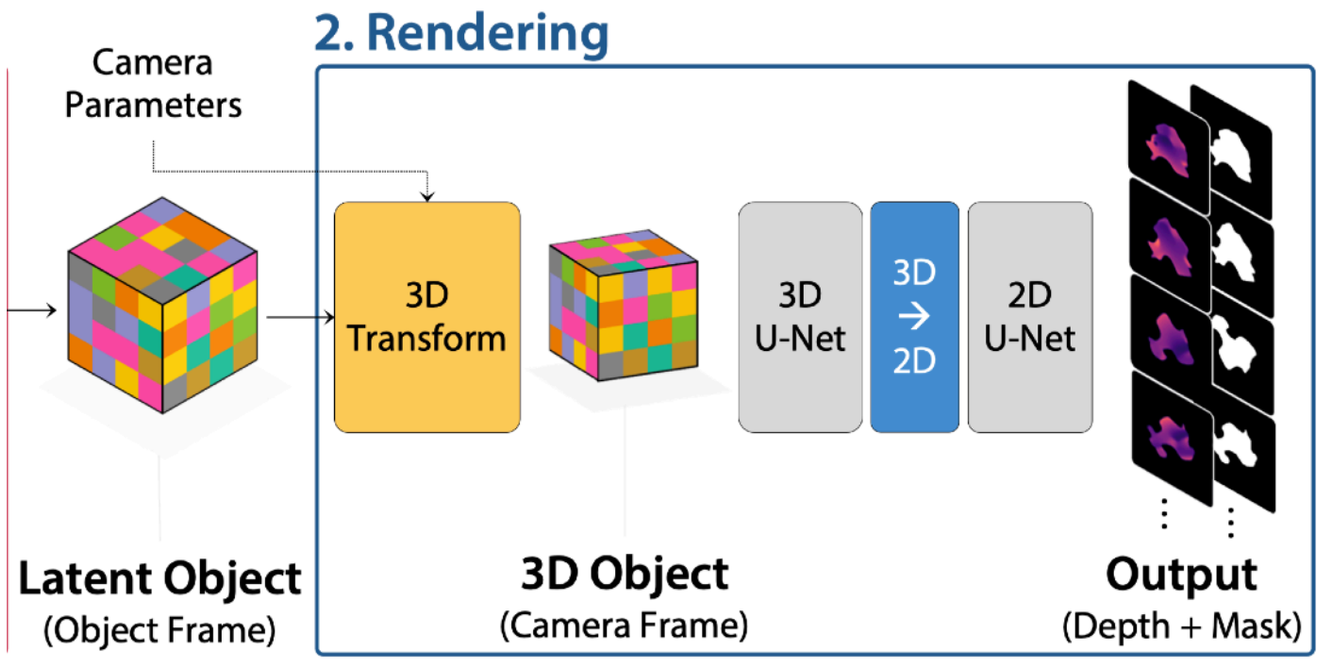
trix. We compute the object-space volume $\hat{\mathbf{\Psi}}$ by sampling $\phi(\boldsymbol{W}^{-1}\boldsymbol{x}'_{ijk})$ for each object-space voxel coordinate $\boldsymbol{x}'_{ijk}$.



Methodology - Rendering

- Rendering here is defined as a inverse process of reconstruction.
  - Given: a coarse estimation of object pose
  - Generate camera frame latent object by transformation.
  - 3D to 2D U-Net to generate depth and mask.

- Image-based Rendering: warping + weight prediction network (not used in the pipeline)



Methodology - Optimization

- Input:
  - Sensory data: RGB, Mask, Depth
  - Feature from previous stages: 3D Latent Object
- Bootstrap:
  - translation 2D-3D object center
  - rotation: uniformly sample
  - select top k with residual value
- Optimization
  - Acquire depth by rendering
  - Evaluate depth error and latent error.

$$\mathcal{L}_{\text{depth}}(\boldsymbol{\mathcal{D}}^*, \boldsymbol{\mathcal{D}}) = \|\boldsymbol{\mathcal{D}}^* - \boldsymbol{\mathcal{D}}\|_1$$

$$\mathcal{L}_{\text{latent}}(\boldsymbol{x}, \boldsymbol{\theta}; \boldsymbol{\Psi}) = \|H_{\boldsymbol{\theta}}(G_{\boldsymbol{\theta}}(\boldsymbol{x})) - H_{\boldsymbol{\theta}}(\boldsymbol{\Psi})\|_1$$

$$\underset{\boldsymbol{\theta}}{\operatorname{argmin}}\, \mathcal{L}_{\text{depth}}\,(\boldsymbol{x}, D(\boldsymbol{\theta})) + \lambda \mathcal{L}_{\text{latent}}(\boldsymbol{x}, \boldsymbol{\theta})$$

Experiment & Results

- Experiment on ModelNet (pose refinement)

Table 1. ModelNet pose refinement experiments compared to DeepIM (DI) [15] and Multi-Path Learning (MP) [32].

	(5°, 5cm)			ADD (0.1d)			Proj2D (5px)		
	DI	MP	Ours	DI	MP	Ours	DI	MP	Ours
bathtub	71.6	**85.5**	85.0	88.6	91.5	**92.7**	73.4	80.6	**94.9**
bookshelf	39.2	**81.9**	80.2	76.4	85.1	**91.5**	51.3	76.3	**91.8**
guitar	50.4	69.2	**73.5**	69.6	80.5	**83.9**	77.1	80.1	**96.9**
range_hood	69.8	**91.0**	82.9	89.6	95.0	**97.9**	70.6	83.9	**91.7**
sofa	82.7	**91.3**	89.9	89.5	95.8	**99.7**	94.2	86.5	**97.6**
tv_stand	73.6	85.9	**88.6**	92.1	90.9	**97.4**	76.6	82.5	**96.0**
wardrobe	62.7	88.7	**91.7**	79.4	92.1	**97.0**	70.0	81.1	**94.2**
Mean	64.3	84.8	**85.5**	83.6	90.1	**94.3**	73.3	81.6	**94.7**

MOPED Dataset

- 11 objects
- RGB-D video sequences with camera pose (from KinectFusion)
- Approximately 300 test images per object

Experiment & Results

- Experiment on MOPED (pose refinement)

Table 2. Quantitative Results on MOPED Dataset. We report the Area Under Curve (AUC) for each metric.

	PoseRBPF [6]			**IBR-LD**			**IBR-D**			**IBR-L**		
Input	*Textured 3D Mesh*			*Images + Camera Pose*								
Training	*Yes*			*No*								
# Networks	*Per-Object*			*Single Universal*								
Pose Loss	-			$\mathcal{L}_{\text{latent}} + \mathcal{L}_{\text{depth}}$			$\mathcal{L}_{\text{depth}}$			$\mathcal{L}_{\text{latent}}$		
	ADD	**ADD-S**	**Proj.2D**	**ADD**	**ADD-S**	**Proj.2D**	**ADD**	**ADD-S**	**Proj.2D**	**ADD**	**ADD-S**	**Proj.2D**
black_drill	59.78	**82.94**	49.80	56.67	79.06	53.77	**62.15**	82.36	**59.36**	51.61	80.81	48.05
cheezit	57.78	82.45	48.47	**61.31**	**91.63**	**55.24**	44.56	90.24	35.10	23.98	88.20	15.92
duplo_dude	56.91	82.14	47.11	74.02	89.55	52.49	**76.81**	**90.50**	**59.83**	53.26	89.51	38.54
duster	**58.91**	82.78	**46.66**	49.13	91.56	19.33	51.13	**91.68**	24.78	39.05	81.57	20.82
graphics_card	59.13	83.20	49.85	**80.71**	**91.25**	**67.71**	79.33	90.90	60.35	60.11	87.91	41.92
orange_drill	**58.23**	82.68	**49.08**	51.84	70.95	46.12	55.52	73.68	45.46	44.20	68.39	41.68
pouch	57.74	82.16	49.01	**60.43**	**89.60**	**49.80**	58.51	89.15	44.40	22.03	82.94	20.19
remote	56.87	82.04	**48.06**	55.38	94.80	37.73	**63.18**	**94.96**	45.27	62.39	91.58	41.96
rinse_aid	57.74	82.53	**48.13**	65.63	92.58	28.61	**67.09**	**93.66**	27.62	57.54	87.44	19.00
toy_plane	**62.41**	85.10	49.81	60.18	**90.24**	**51.70**	56.80	88.54	40.16	34.29	87.22	35.07
vim_mug	**58.09**	**82.38**	**48.08**	30.11	80.76	14.38	49.89	77.79	32.85	27.49	78.59	10.51
mean	58.51	82.76	**48.55**	58.67	87.45	43.35	**60.45**	**87.59**	43.20	43.27	84.01	30.33

Experiment & Results

- Ablation:
  - number of reference view
    - having more than 8 view yields marginal improvement.
  - different view fusion strategy
    - CovGRU is better

Table 3. AUC metrics by number of reference views.

# Views	1	2	4	8	16	32
ADD	15.91	25.00	40.38	55.35	**58.67**	55.81
ADD-S	63.14	75.91	85.62	87.72	87.45	**88.70**
Proj.2D	8.68	15.43	28.41	38.87	**43.35**	38.45

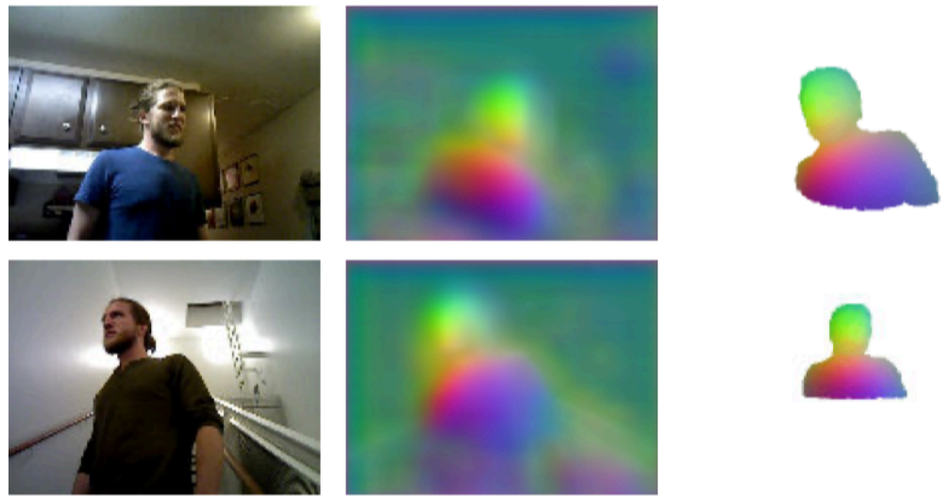
Table 4. Effect of different View Fusion strategies

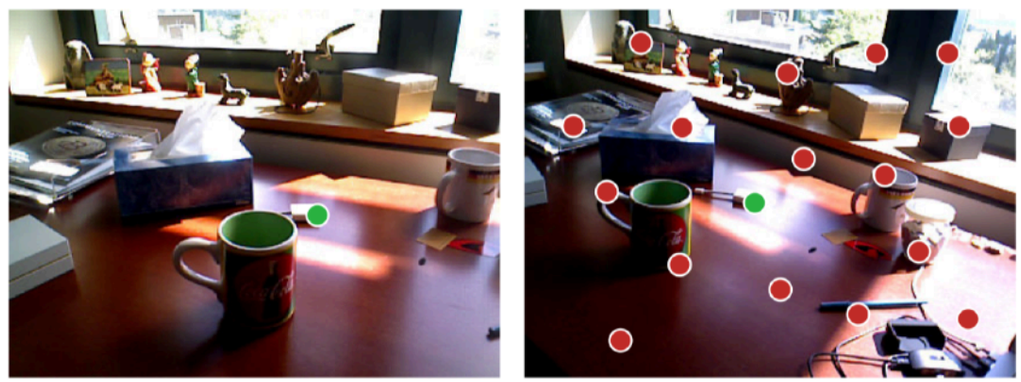
	ADD	ADD-S	Proj.2D
Avg Pool	**56.78**	88.04	39.82
ConvGRU	56.36	**88.28**	**40.43**

Refining 6D Object Pose Predictions using Abstract Render-and-Compare

Preliminary - Descriptor Learning

- Descriptor Learning
  - Learn feature vectors that can uniquely represent a point
  - Hope to have similar feature of the same point across frames
    - Reference: UCN, Super Point, GIFT
- Contrastive Loss
  - Encourage network to produce similar feature for positive samples and distinct feature for negative samples





$$\mathcal{L}_{\text{matches}}(I_a,I_b)=\frac{1}{N_{\text{matches}}}\sum_{N_{\text{matches}}} D(I_a,u_a,I_b,u_b)^2$$

$$\mathcal{L}_{\text{non-matches}}(I_a,I_b)=\frac{1}{N_{\text{non-matches}}}\sum_{N_{\text{non-matches}}} max(0,M-D(I_a,u_a,I_b,u_b))^2$$

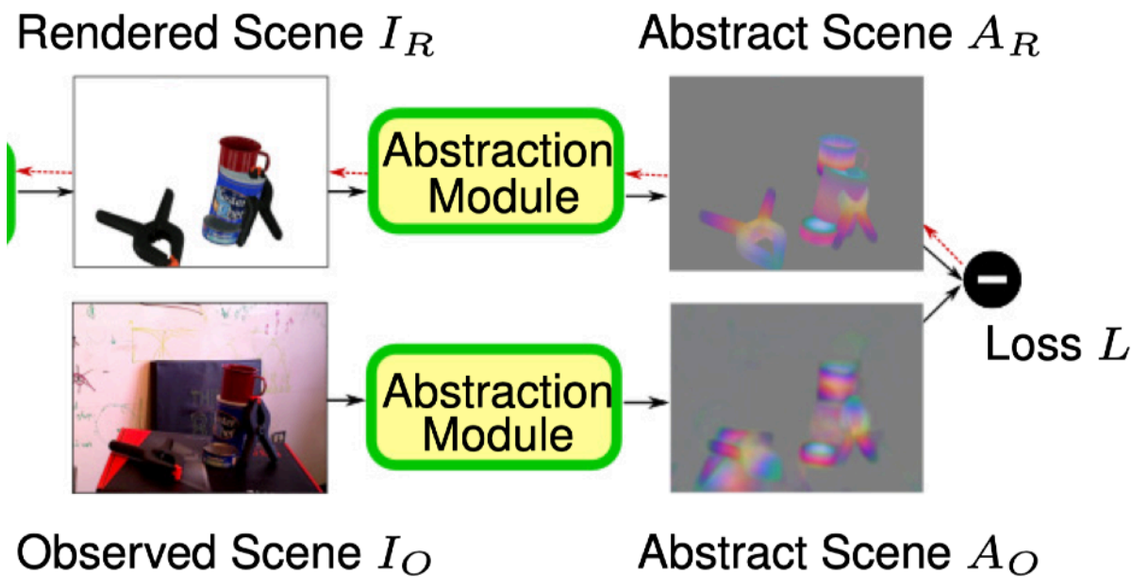
$$\mathcal{L}(I_a,I_b)=\mathcal{L}_{\text{matches}}(I_a,I_b)+\mathcal{L}_{\text{non-matches}}(I_a,I_b)$$

Methodology

- Reconstruction
  - Train descriptor network by **matching synthetic image to real image**
  - Generate meta model: Synthetically generate reference view + Feature fusion

Fig. 3. Learned surface features, projected and fused onto the mesh. The 3D feature vectors are visualized directly as RGB colors.

- Optimization
  - Render meta model onto the image with dense descriptor generated with real image and network.
  - Compare and optimize the pose.



Experiment & Results

TABLE I

POSE REFINEMENT RESULTS ON THE YCB VIDEO DATASET

	PoseCNN [5]		PoseCNN refined (ours)		HeatMaps [26]		HeatMaps refined (ours)		DeepIM [16]	
Object	ADD	ADD-S	ADD(Δ)	ADD-S(Δ)	ADD	ADD-S	ADD(Δ)	ADD-S(Δ)	ADD	ADD-S
master_chef_can	50.2	83.9	63.3(+13.1)	91.7(+7.8)	81.9	91.4	76.7(−5.1)	90.2(−1.2)	65.2	87.8
cracker_box	53.1	76.9	65.3(+12.2)	81.7(+4.9)	83.6	90.0	82.9(−0.7)	89.4(−0.6)	82.6	89.8
sugar_box	68.4	84.2	85.3(+16.9)	92.0(+7.8)	82.1	89.8	86.4(+4.3)	92.2(+2.4)	89.7	93.8
tomato_soup_can	66.2	81.0	59.4(−6.8)	79.9(−1.1)	79.8	89.5	57.4(−22.4)	78.2(−11.3)	81.4	90.1
mustard_bottle	81.0	90.4	86.5(+5.5)	92.3(+1.9)	91.5	95.0	86.7(−4.7)	92.6(−2.4)	90.3	94.4
tuna_fish_can	70.7	88.0	81.1(+10.4)	94.3(+6.3)	48.7	71.7	69.7(+21.0)	85.7(+14.0)	85.4	94.5
pudding_box	62.7	79.1	71.1(+8.4)	83.1(+4.1)	90.2	94.1	68.8(−21.4)	80.7(−13.4)	84.9	91.8
gelatin_box	75.2	87.2	81.5(+6.3)	89.1(+1.9)	93.7	95.9	73.0(−20.7)	82.8(−13.1)	87.7	91.6
potted_meat_can	59.5	78.5	63.7(+4.2)	80.3(+1.8)	79.1	90.0	74.6(−4.5)	87.6(−2.4)	70.0	78.2
banana	72.3	86.0	82.1(+9.8)	91.8(+5.8)	51.7	67.8	68.8(+17.1)	81.0(+13.2)	83.3	92.0
pitcher_base	53.3	77.0	85.1(+31.8)	92.7(+15.7)	69.4	85.0	83.8(+14.4)	92.1(+7.1)	88.7	93.7
bleach_cleanser	50.3	71.6	65.0(+14.7)	80.4(+8.9)	76.2	85.5	78.3(+2.0)	87.6(+2.2)	75.9	86.8
bowl	3.3	69.6	6.5(+3.1)	75.5(+5.9)	3.6	78.1	1.5(−2.1)	66.4(−11.6)	41.5	77.8
mug	58.5	78.2	65.9(+7.4)	84.0(+5.9)	53.9	75.8	57.9(+4.0)	78.9(+3.1)	70.3	86.1
power_drill	55.3	72.7	73.7(+18.4)	85.9(+13.2)	82.9	90.8	81.5(−1.3)	90.4(−0.4)	90.7	94.6
wood_block	26.6	64.3	45.5(+18.9)	73.3(+9.0)	0.0	57.0	0.0(+0.0)	60.3(+3.3)	26.2	60.1
scissors	35.8	56.9	40.0(+4.1)	58.6(+1.7)	65.3	79.6	75.4(+10.1)	85.4(+5.8)	45.5	61.8
large_marker	58.3	71.7	63.9(+5.6)	77.3(+5.6)	56.5	70.2	59.8(+3.3)	70.2(+0.0)	68.1	77.5
large_clamp	24.6	50.2	37.0(+12.4)	65.1(+15.0)	57.2	73.1	75.3(+18.1)	85.6(+12.5)	45.5	72.1
extra_large_clamp	16.1	44.1	25.4(+9.3)	63.7(+19.6)	23.6	54.6	20.4(−3.1)	58.3(+3.7)	29.1	70.0
foam_brick	40.2	88.0	43.3(+3.1)	90.8(+2.8)	32.1	88.9	37.0(+5.0)	92.1(+3.2)	70.5	83.0
ALL	53.7	75.8	62.8(+9.1)	82.4(+6.6)	66.2	82.4	67.0(+0.7)	83.5(+1.1)	70.1	84.2

We report the area under the accuracy curve (AUC) for varying error thresholds on the ADD and ADD-S metrics.