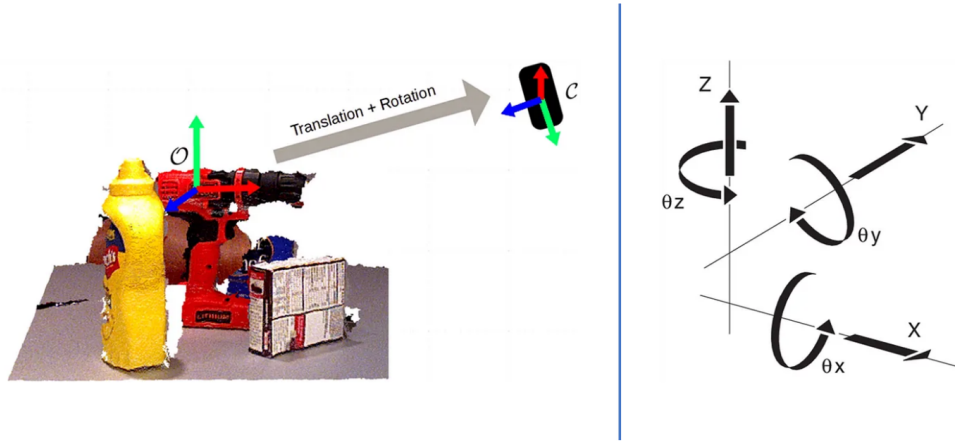
DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion

Presenter: Saina Rezvani

September 17, 2023

What is 6D Pose?

3D translation (x, y, x) and rotation (row, pitch, yaw) which refers to the location and orientation of the object in space.

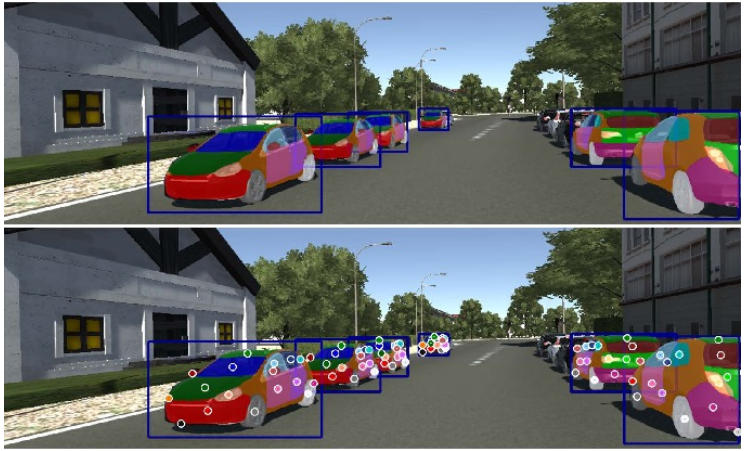


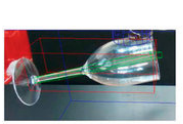
6D object pose estimation (L) The goal of 6D pose estimation is to find the translation and rotation from the object coordinate frame O to the camera coordinate frame C. (R) The translation and rotation both have three degrees of freedom,

Significance of 6D Pose Estimation

Important task because:

- robot manipulation (e.g. bin picking)
- self-driving cars
- augmented reality
- human robot interaction (e.g. learning from demonstration).





Why is 6D Pose Estimation Challenging

- Complexity of the environment: clutter, occlusion.
- Variety of objects with different textures and shapes.
- Change in object apparencies in images due to lighting and other conditions.
- Sensor noise

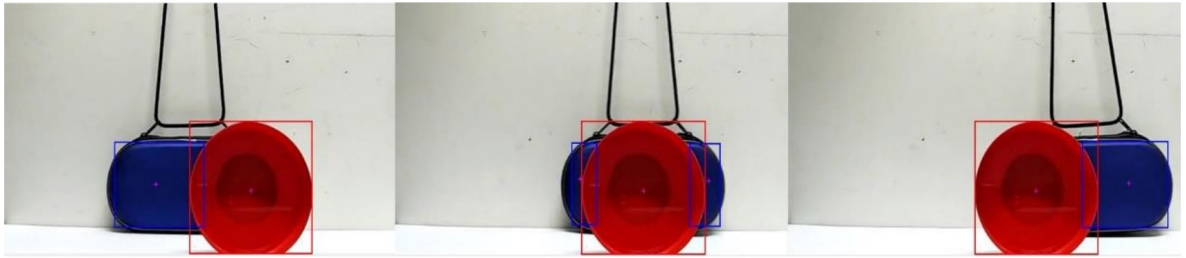


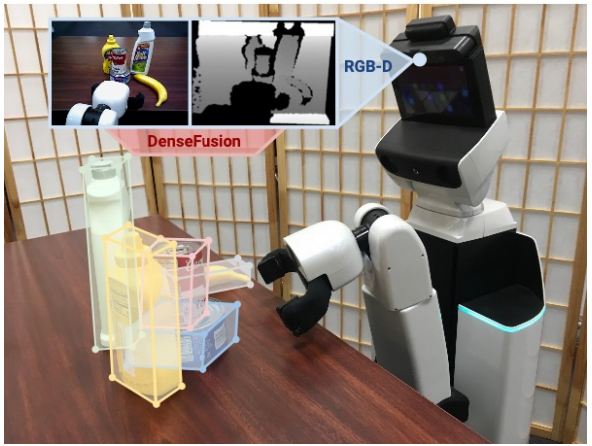
Fig. 1. Occlusion issues in multiple object tracking.

Previous Work

- Handcrafted feature extraction such as Brachmann et al [1]
    - **Cons:** Poor performance with heavy occlusion and lighting variation
- Deep network-based pose estimation such as PoseCNN and MCN
    - **Cons**: require extensive post processing refinement
- Frustrum PointNet and PointFusion improve on deep network methods that perform well in real-time.
    - **Cons**: Perform well in driving scenes, but limited under heavy occlusion

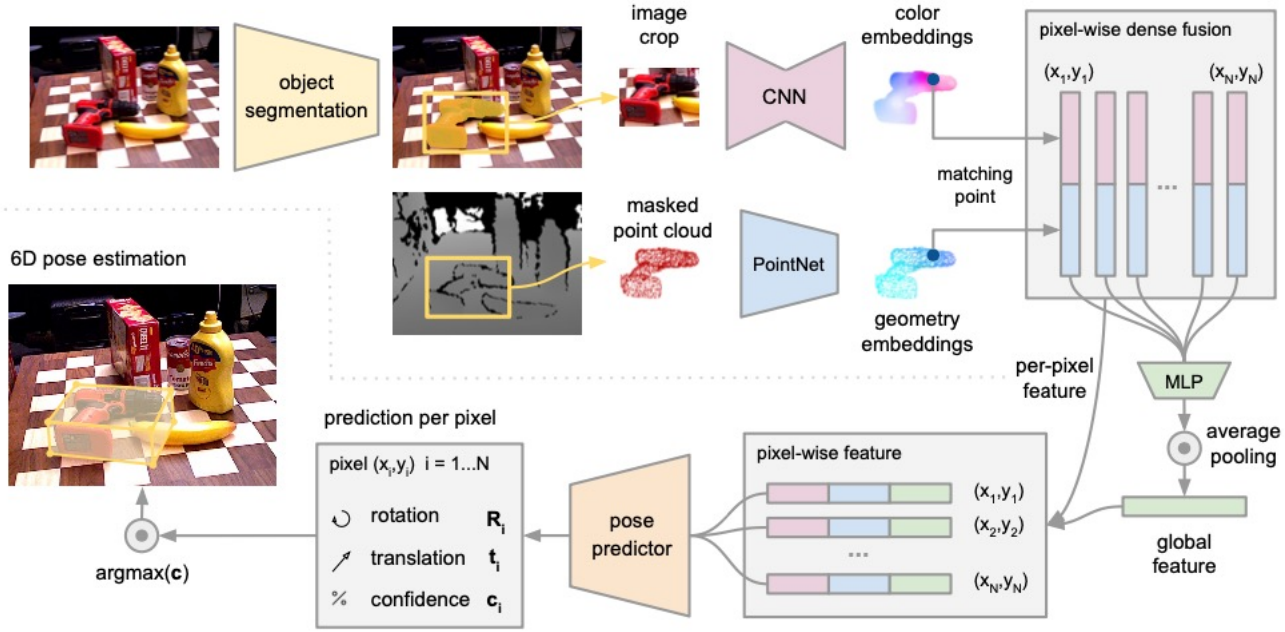
Motivation

- Create an end-to-end deep network to predict 6D pose estimation of known objects in space using RGB-D images.
- Achieve robustness for complex tasks including occlusion.
- Achieve optimal speed for real time applications.
- Utilize both color and depth information in fusion to extract pixel-wise features to estimate the pose.
- Addition of iterative pose refinement method that can run in real time



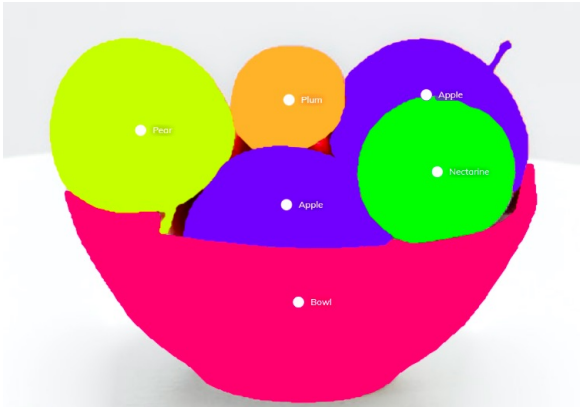
Architecture Overview

1. Semantic segmentation
2. Dense feature extraction
3. Pixel-wise dense fusion
4. 6D object pose estimation
5. Iterative refinement



Semantic Segmentation

- Encoder-decoder architecture
- Takes an image as input
- Creates an N+1 channeled semantic segmentation map
- In each channel, active pixels refer to one objects in the set of N possible objects



Dense Feature Extraction

- Color and depth information are processed separately to create feature embeddings that include the intrinsic structure of each data.
- Dense 3D point cloud feature embedding
  - Converts segmented depth pixels into a 3D point cloud using camera intrinsics
  - PointNet-like model to extract geometric features
- Dense color image feature embedding
  - CNN-based encoder-decoder architecture mapping an image of size H x W x 3 into H x W x d_rgb embedding space
  - Each pixel of the embedding is a d_rgb dimensional vector representing appearance information of the image at the specific location

6D Object Pose Estimation

Minimize the pose estimation loss for each dense-pixel prediction

Loss for asymmetric objects:

$$L_i^p = \frac{1}{M} \sum_j ||(Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i)||$$

x_i Is the j^{ith} point in the M randomly selected 3D points from the object.

$p = [R|t] \rightarrow$ ground truth pose

$\widehat{p_i} = [\widehat{R_i}|\widehat{t_i}] \rightarrow$ predicted pose corresponding to the i^{th} dense-pixel.

6D Object Pose Estimation

Minimize the pose estimation loss for each dense-pixel prediction

Loss for symmetric objects:

$$L_i^p = \frac{1}{M} \sum_j \min_{0<k<M} ||(Rx_j + t) - (\hat{R}_i x_k + \hat{t}_i)||$$

x_i Is the j^{ith} point in the M randomly selected 3D points from the object.

$p = [R|t] \rightarrow$ ground truth pose

$\widehat{p_i} = [\widehat{R_i}|\widehat{t_i}] \rightarrow$ predicted pose corresponding to the i^{th} dense-pixel.

6D Object Pose Estimation

Minimizing the sum of all dense-pixel losses:

$$L = \frac{1}{N} \sum_i L_i^p.$$

The network balances the confidence scores with each pose prediction. Hence, weighing the confidence score with the loss and adding a confidence regularization formula.

$$L = \frac{1}{N} \sum_i (L_i^p c_i - w \log(c_i)),$$

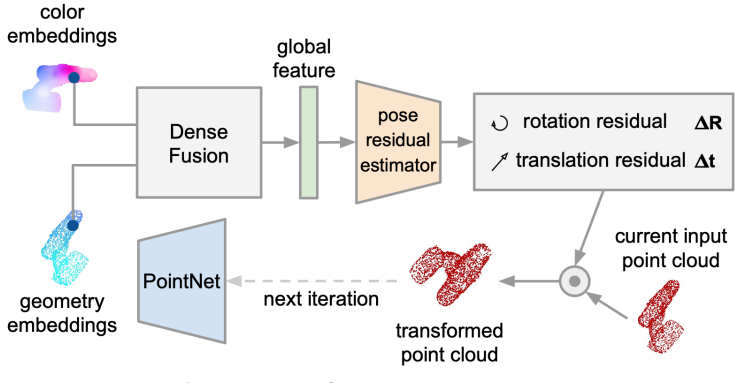
N = the number of randomly selected dense-pixels features from the P elements of the segment

ω = balancing hyperparameter

Iterative Refinement

Neural network based iterative refinement to improve pose estimation

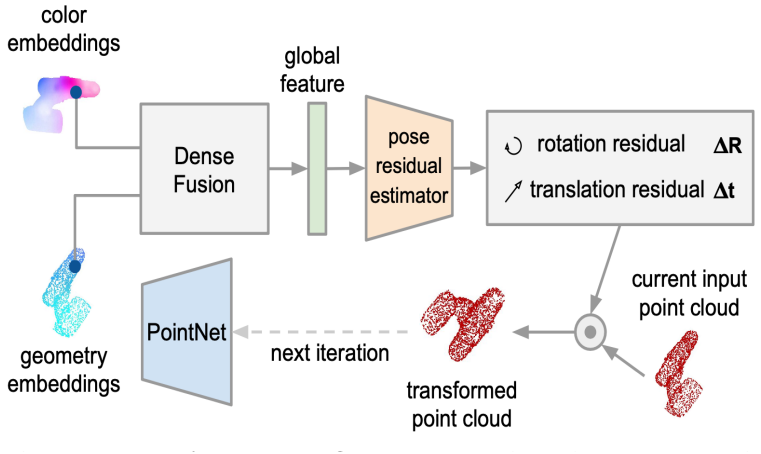
- Step 1: Use the previous predicted pose to transform the input point cloud into its canonical frame.
- Step 2: Feed the transformed point cloud back into the network and predict a residual pose
- Step 3: Iterate to obtain finer pose estimations



Iterative Refinement

- After K iterations the final pose estimation is obtained as a concatenation of per iteration estimates:

$$\hat{p} = [R_K|t_K] \cdot [R_{K-1}|t_{K-1}] \cdot \cdots \cdot [R_0|t_0]$$



YCB-Video Dataset

- 21 objects of varying shapes and texture levels under different occlusion conditions
- 92 RGB-D videos where each video shows a subset of the 21 objects in different indoor environments
- The videos are labeled with 6d poses and segmentation masks
- Divided into 80 videos for training and 12 for testing

LineMOD Dataset

- 13 low-textured objects of the existing 15 objects were used

- Training dataset includes around 200 instances for each object and the test dataset includes around 1000 instances

Experiments

Metrics used:

- **Average Closest Point Distance (ADD-S):** The mean distance from each 3d point transformed by $[\widehat{R}_i|\widehat{t}_i]$ to its closest neighbor on the target model transformed by $[R|t]$.

- **Average Distance of Model Points (ADD):** The mean distance from each 3d point transformed by $[\widehat{R}_i|\widehat{t}_i]$ to the same point on the target model.

- **AUC:** The area under the ADD-S curve

Image embedding network: Resnet-18 encoder + 4 up-sampling layers (decoder)
PointNet Architecture: Multilayer perceptron (MLP) + average-pooling
Iterative pose refinement: 4 fully connected layers outputting pose residual.

- All experiments use 2 refinement iterations.

Experiments using YCB-Video

Table 1. Quantitative evaluation of 6D pose (ADD-S[40]) on YCB-Video Dataset. Objects with bold name are symmetric.

	PointFusion [41]		PoseCNN+ICP [40]		Ours (single)		Ours (per-pixel)		Ours (iterative)	
	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm	AUC	<2cm
002_master_chef_can	90.9	99.8	95.8	100.0	93.9	100.0	95.2	100.0	**96.4**	100.0
003_cracker_box	80.5	62.6	92.7	91.6	90.8	98.4	92.5	99.3	**95.5**	**99.5**
004_sugar_box	90.4	95.4	**98.2**	100.0	94.4	99.2	95.1	100.0	97.5	100.0
005_tomato_soup_can	91.9	96.9	94.5	96.9	92.9	96.7	93.7	96.9	**94.6**	96.9
006_mustard_bottle	88.5	84.0	**98.6**	100.0	91.2	97.8	95.9	100.0	97.2	100.0
007_tuna_fish_can	93.8	99.8	**97.1**	100.0	94.9	100.0	94.9	100.0	96.6	100.0
008_pudding_box	87.5	96.7	**97.9**	100.0	88.3	97.2	94.7	100.0	96.5	100.0
009_gelatin_box	95.0	100.0	**98.8**	100.0	95.4	100.0	95.8	100.0	98.1	100.0
010_potted_meat_can	86.4	88.5	**92.7**	**93.6**	87.3	91.4	90.1	93.1	91.3	93.1
011_banana	84.7	70.5	**97.1**	99.7	84.6	62.0	91.5	93.9	96.6	**100.0**
019_pitcher_base	85.5	79.8	**97.8**	100.0	86.9	80.9	94.6	100.0	97.1	100.0
021_bleach_cleanser	81.0	65.0	**96.9**	99.4	91.6	98.2	94.3	99.8	95.8	**100.0**
024_bowl	75.7	24.1	81.0	54.9	83.4	55.4	86.6	69.5	**88.2**	**98.8**
025_mug	94.2	99.8	95.0	99.8	90.3	94.7	95.5	**100.0**	**97.1**	**100.0**
035_power_drill	71.5	22.8	**98.2**	**99.6**	83.1	64.2	92.4	97.1	96.0	98.7
036_wood_block	68.1	18.2	87.6	80.2	81.7	76.0	85.5	93.4	**89.7**	**94.6**
037_scissors	76.7	35.9	91.7	95.6	83.6	75.1	96.4	**100.0**	**95.2**	**100.0**
040_large_marker	87.9	80.4	97.2	99.7	91.2	88.6	94.7	99.2	**97.5**	**100.0**
051_large_clamp	65.9	50.0	**75.2**	74.9	70.5	77.1	71.6	78.5	72.9	**79.2**
052_extra_large_clamp	60.4	20.1	64.4	48.8	66.4	50.2	69.0	69.5	**69.8**	**76.3**
061_foam_brick	91.8	100.0	**97.2**	100.0	92.1	100.0	92.4	100.0	92.5	100.0
MEAN	83.9	74.1	93.0	93.2	88.2	87.9	91.2	95.3	**93.1**	**96.8**

Experiments using YCB-Video

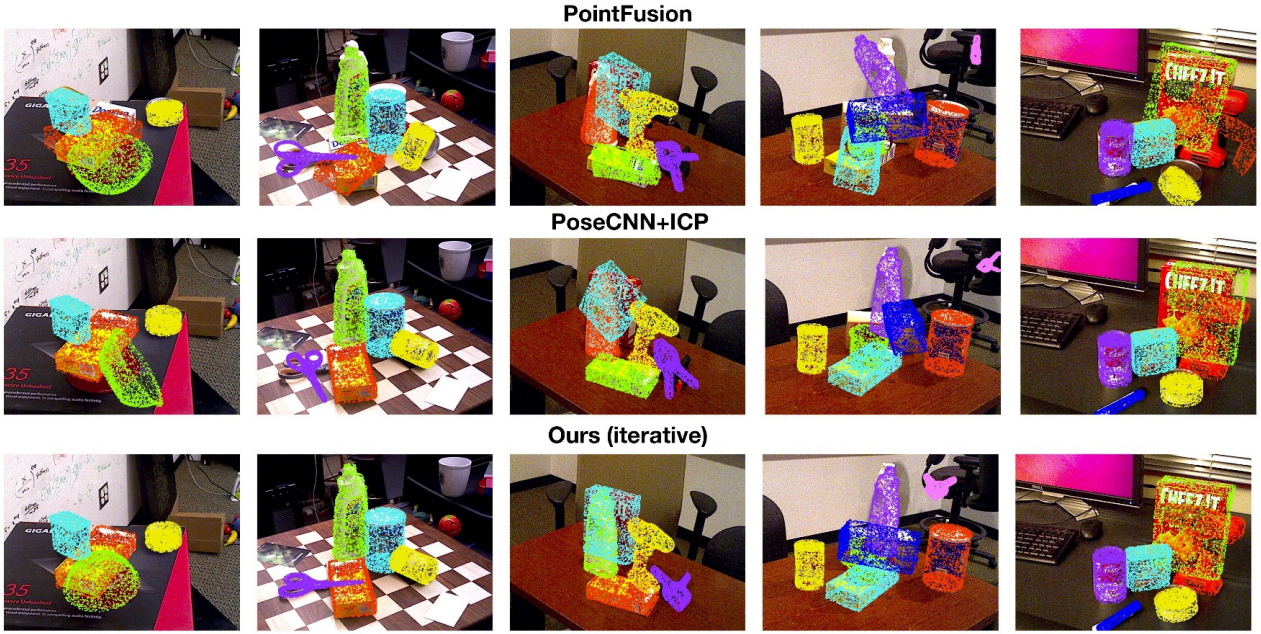


Figure 4. **Qualitative results on the YCB-Video Dataset.** All three methods shown here are tested with the same segmentation masks as in PoseCNN. Each object point cloud in different color are transformed with the predicted pose and then projected to the 2D image frame. The first two rows are former RGB-D methods and the last row is our approach with dense fusion and iterative refinement (2 iterations).

Experiments using LineMOD

Table 2. Quantitative evaluation of 6D pose (ADD[13]) on the LineMOD dataset. Objects with bold name are symmetric.

	RGB		RGB-D				
	BB8 [24] w ref.	PoseCNN +DeepIM [17, 40]	Implicit [30]+ICP	SSD-6D [14]+ICP	PointFusion [41]	Ours (per-pixel)	Ours (iterative)
ape	40.4	77.0	20.6	65	70.4	79.5	92.3
bench vi.	91.8	97.5	64.3	80	80.7	84.2	93.2
camera	55.7	93.5	63.2	78	60.8	76.5	94.4
can	64.1	96.5	76.1	86	61.1	86.6	93.1
cat	62.6	82.1	72.0	70	79.1	88.8	96.5
driller	74.4	95.0	41.6	73	47.3	77.7	87.0
duck	44.3	77.7	32.4	66	63.0	76.3	92.3
eggbox	57.8	97.1	98.6	100	99.9	99.9	99.8
glue	41.2	99.4	96.4	100	99.3	99.4	100.0
hole p.	67.2	52.8	49.9	49	71.8	79.0	92.1
iron	84.7	98.3	63.1	78	83.2	92.1	97.0
lamp	76.5	97.5	91.7	73	62.3	92.3	95.3
phone	54.0	87.7	71.0	79	78.8	88.0	92.8
MEAN	62.7	88.6	64.7	79	73.7	86.2	94.3

Experiments using LineMOD

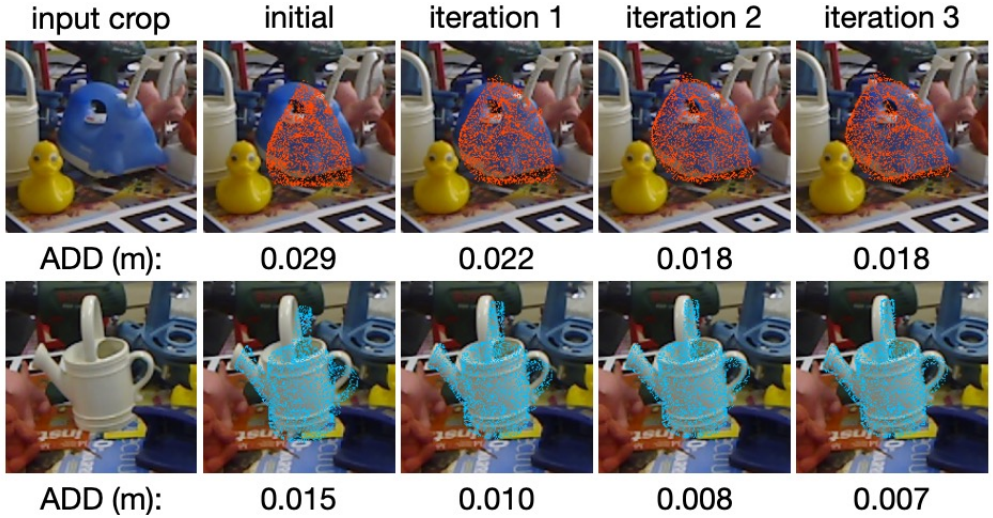


Figure 6. **Iterative refinement performance on LineMOD dataset** We visualize how our iterative refinement procedure corrects initially sub-optimal pose estimation.

Model Performance

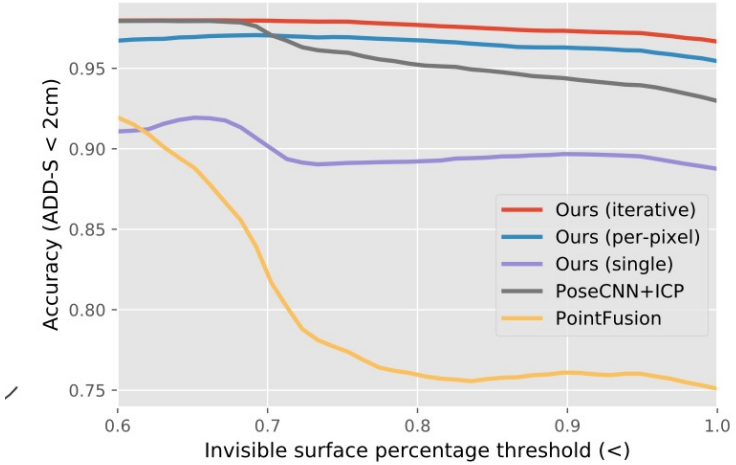


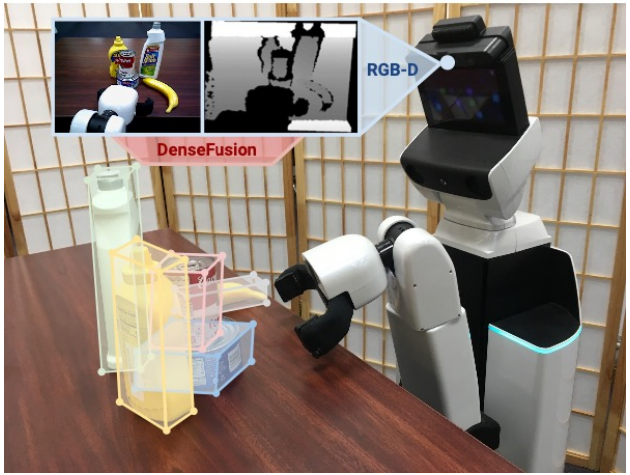
Figure 5. **Model performance under increasing levels of occlusion.** Here the levels of occlusion is estimated by calculating the invisible surface percentage of each object in the image frame. Our methods work more robustly under heavy occlusion compared to baseline methods.

Table 3. Runtime breakdown (second per frame on YCB-Video Dataset). Our method is approximately 200x faster than PoseCNN+ICP. Seg means Segmentation, and PE means Pose Estimation.

PoseCNN+ICP [40]				Ours			
Seg	PE	ICP	ALL	Seg	PE	Refine	ALL
0.03	0.17	10.4	10.6	0.03	0.02	0.01	0.06

Robotics Grasping Experiment

- 5 YCB objects were placed on the table
  - 4 different locations
  - 3 random orientations with partial occlusion
- Robot was commanded to grasp them using the estimated pose
- Robot made 12 attempts for each object (60 total attempts)
- 73% success rate with banana being the most difficult case
  - Banana looked different form dataset?
- Architecture is robust for manipulation with occlusion



Extended Readings

PoseCNN:

Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estima- tion in cluttered scenes,” *ArXiv preprint arXiv:1711.00199*, 2017.

MCN:

C. Li, J. Bai, and G. D. Hager, “A unified framework for multi-view multi-class object pose estimation,” *ArXiv preprint arXiv:1803.08103*, 2018.

PointFusion:

D. Xu, D. Anguelov, and A. Jain, “Pointfusion: Deep sen- sor fusion for 3d bounding box estimation,” *ArXiv preprint arXiv:1711.10871*, 2017.

ICP:

P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, pp. 239–256, 1992.

Thank you for listening!