Take a Deeper Look at Object 6D Pose Estimation

Siyu ZHANG

Research Engineer
ZJU-SenseTime Joint Lab of 3D Vision
Overview

• Some terminologies …

  • Pose Estimation
    • Given: Image_t
    • Target: Pose_t

  • Pose Refinement
    • Given: Image_t, Pose_t_init
    • Target: Pose_t_refined

• Pose Tracking
  • Given: Image_t-1, Pose_t-1, Image_t
  • Target: Pose_t
Overview

• As a regression problem
  • Pose Estimation: direct regression
    • Extract feature from image
    • Directly regress the translation and orientation of target object

*PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. RSS, 2018*
Overview

- As a regression problem
  - Pose Tracking: render and regression
    - Render the image of target object with initial pose and object 3D model
    - Feed the rendered image and input image into Neural Network to extract feature
    - Regress the additive pose for target object

*DeepIM: Deep Iterative Matching for 6D Pose Estimation, ECCV, 2018*
Overview

• As a **regression** problem
  • Pose Estimation: direct regression from image feature
  • Pose Tracking: render, compare with observed image, then regress relative (additive) pose

• **As a matching** problem
Overview

- As a **matching** problem
  - Pose Estimation by matching sparse correspondences
    - Extract semantic 2D keypoint from the image
    - Solve for PnP with corresponding 3D keypoints in object frame
Overview

• As a **matching** problem
  • Pose Estimation by matching dense correspondences
    • Estimate pixel-wise object coordinates for all foreground pixels
  • Apply RANSAC + PnP to solve for object pose
Overview

• As a **matching** problem

• Pose Tracking by keypoint tracking

  • Extract keypoint and descriptor from input images of different frames

  • Using matched keypoint to estimate relative pose between different frames

*Multiple 3D Object Tracking, TVCG, 2011; https://www.youtube.com/watch?v=eqlEzWmuijs*
Overview

- As a **matching** problem
  - Pose Tracking by silhouette tracking:
    - Project object mesh with initial object pose and extract silhouette
    - Compute residual of current silhouette according to local foreground-background similarity
    - Optimize object pose to minimize the residual b gradient-based approach
Overview

- As a **regression** problem
  - Pose Estimation: direct regression
  - Pose Tracking: render and regression

- As a **matching** problem
  - Pose Estimation: matching from image pixels to points in object frame
  - Pose Tracking: matching between frames

- Msic.
  - Tracking by Detection: Single frame estimator + filtering (smoothing)
  - Coarse-to-fine estimation: Coarse pose initialization (or sampling) + Iterative refinement
Overview

• Pose estimation is a solved problem when:
  • Using **sparse keypoint** as representation
  • Given **adequate well-annotated** data and **precise** object model

• Some recent trends
  • sparse representation to denser representation
  • instance-level to category-level
  • model-based to model-free
### Overview

- **Paper to cover**

<table>
<thead>
<tr>
<th>Paper Name</th>
<th>Conference</th>
<th>Model-Free</th>
<th>Type</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOCS</td>
<td>CVPR19</td>
<td>at inference</td>
<td>category-level</td>
<td>NOCS</td>
</tr>
<tr>
<td>DPOD</td>
<td>ICCV19</td>
<td>No</td>
<td>instance-level</td>
<td>UV Map</td>
</tr>
<tr>
<td>Hybrid Pose</td>
<td>CVPR20</td>
<td>No</td>
<td>instance-level</td>
<td>hybrid of geometric primitives</td>
</tr>
<tr>
<td>6-Pack</td>
<td>ICRA20</td>
<td>at training and inference</td>
<td>category-level</td>
<td>keypoints</td>
</tr>
<tr>
<td>Category Level Object Pose Estimation via Neural Analysis-by-Synthesis</td>
<td>ECCV20</td>
<td>at training and inference</td>
<td>category-level</td>
<td>latent vector</td>
</tr>
<tr>
<td>Shape Prior Deformation for Categorical 6D Object Pose and Size Estimation</td>
<td>ECCV20</td>
<td>at inference</td>
<td>category-level</td>
<td>NOCS</td>
</tr>
<tr>
<td>LatentFusion</td>
<td>CVPR20</td>
<td>at training and inference</td>
<td>unrestricted</td>
<td>latent volume</td>
</tr>
<tr>
<td>Reconstruct Locally, Locally Globally</td>
<td>CVPR20</td>
<td>at training and inference</td>
<td>instance-level</td>
<td>object coordinates</td>
</tr>
</tbody>
</table>
Pose Estimation with Various Representations

- **NOCS**
  - Representation: Normalize Object Coordinate Space
  - **Approach:**
    - Estimate NOCS Map, lift to original scale with depth
    - Obtain object point cloud with depth and instance mask
    - Estimate pose with transformation from object coordinate to object point cloud
Pose Estimation with Various Representations

- **DPOD**
  - **Representation:** UV Map
  - **Approach:**
    - Estimate UV Map and instance mask
    - Estimate pose using RANSAC + PnP
    - Refine pose with render and compare
Pose Estimation with Various Representations

- **DPOD**
  - Experiment result on LineMOD
  - Discussion: UV Map VS. NOCS

<table>
<thead>
<tr>
<th>w/o Refinement</th>
<th>+ Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PVNet [25]</strong></td>
<td><strong>DeepIM [18]</strong></td>
</tr>
<tr>
<td>Ours</td>
<td>Ours</td>
</tr>
<tr>
<td>43.62</td>
<td>77.0</td>
</tr>
<tr>
<td>99.90</td>
<td>97.5</td>
</tr>
<tr>
<td>86.86</td>
<td>93.5</td>
</tr>
<tr>
<td>95.47</td>
<td>96.5</td>
</tr>
<tr>
<td>79.34</td>
<td>82.1</td>
</tr>
<tr>
<td>96.43</td>
<td>95.0</td>
</tr>
<tr>
<td>52.58</td>
<td>77.7</td>
</tr>
<tr>
<td>99.15</td>
<td>97.1</td>
</tr>
<tr>
<td>95.66</td>
<td>99.4</td>
</tr>
<tr>
<td>81.92</td>
<td>52.8</td>
</tr>
<tr>
<td>98.88</td>
<td>100</td>
</tr>
<tr>
<td>99.33</td>
<td>96.84</td>
</tr>
<tr>
<td>92.41</td>
<td>94.69</td>
</tr>
<tr>
<td><strong>86.27</strong></td>
<td><strong>88.6</strong></td>
</tr>
<tr>
<td><strong>82.98</strong></td>
<td><strong>95.15</strong></td>
</tr>
</tbody>
</table>
Pose Estimation with Various Representations

• **Hybrid-Pose**

  ![Diagram](image)

  ![Diagram](image)

  ![Diagram](image)

• In a nut shell: estimate multiple geometric primitives with network, then optimize over them

  • Geometric primitives:
    - Pixel coordinate Keypoint
    - 2D edges between pairwise key points
    - Pairs of symmetric points

\[
\begin{align*}
    r^{K}_{R,t}(p_k) & := \hat{p}_k \times (R\hat{p}_k + t), \\
    r^{E}_{R,t}(v_e, p_e) & := \hat{v}_e \times (R\hat{p}_e + t) + \hat{p}_e \times (R\hat{v}_e) \quad \text{(2)} \\
    r^{S}_{R,t}(q_{s,1}, q_{s,2}) & := (\hat{q}_{s,1} \times \hat{q}_{s,2})^T R \hat{n}_e. \quad \text{(3)}
\end{align*}
\]

Residual for initial pose estimation

\[
\begin{align*}
    \min_{R, t} & \sum_{k=1}^{\mid \mathcal{K} \mid} \rho(\|r^{K}_{R,t}(p_k)\|, \beta_K) \|r^{K}_{R,t}(p_k)\|_{\Sigma_k}^2 \\
    & + \frac{\mid \mathcal{E} \mid}{\mid \mathcal{E} \mid} \sum_{e=1}^{\mid \mathcal{E} \mid} \rho(\|r^{E}_{R,t}(v_e)\|, \beta_E) \|r^{E}_{R,t}(v_e)\|_{\Sigma_e}^2 \\
    & + \frac{\mid \mathcal{S} \mid}{\mid \mathcal{S} \mid} \sum_{s=1}^{\mid \mathcal{S} \mid} \rho(r^{S}_{R,t}(q_{s,1}, q_{s,2}), \beta_S). \quad \text{(9)}
\end{align*}
\]

Residual for pose refinement

• Different formulations are used for pose estimation and refinement
  - For pose estimation, the target is to form an \(Ax=b\) linear system
  - For pose refinement, the gradient-based optimization approach is used
Category-Level Pose Estimation

• 6-Pack

• Approach:
  • 3D anchor generation (3D grid)
  • Point-wise feature aggregated to anchor feature
  • Anchor scoring and 3D keypoint regression
  • Intuition: coarse anchor selection + fine-grained keypoint selection to enlarge search space

- Anchor scoring loss
  \[ L_{anc} = \frac{1}{N} \sum_i c_i(||a_i - o_{gt}||_2 - \beta) \]

- Multi-view consistency loss
  \[ L_{mvc} = \frac{1}{K} \sum_i ||k_i^t - [\Delta R_i^t|\Delta t_i^t]| \cdot k_i^{t-1}|| \]

- Pose estimation loss
  \[ L_{rot} = 2 \arcsin \left( \frac{1}{2\sqrt{2}} ||\Delta \hat{R}_t - \Delta \hat{R}_i^t|| \right) \]
Category-Level Pose Estimation

- 6-Pack
- Results on NOCS-REAL275
  - Accurate and stable
  - Run at 10 HZ with GTX1070

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>bottle</td>
<td>5°5cm</td>
<td>10.1</td>
<td>5.9</td>
<td>23.7</td>
<td>24.5</td>
</tr>
<tr>
<td></td>
<td>IoU25</td>
<td>29.9</td>
<td>23.1</td>
<td>92.0</td>
<td>91.1</td>
</tr>
<tr>
<td></td>
<td>R_{err}</td>
<td>48.0</td>
<td>28.5</td>
<td>15.7</td>
<td>15.6</td>
</tr>
<tr>
<td></td>
<td>T_{err}</td>
<td>15.7</td>
<td>9.5</td>
<td>4.2</td>
<td>4.0</td>
</tr>
<tr>
<td>bowl</td>
<td>5°5cm</td>
<td>62.2</td>
<td>16.8</td>
<td>53.0</td>
<td>55.0</td>
</tr>
<tr>
<td></td>
<td>IoU25</td>
<td>40.3</td>
<td>74.7</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>R_{err}</td>
<td>79.7</td>
<td>9.8</td>
<td>5.3</td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td>T_{err}</td>
<td>19.0</td>
<td>8.2</td>
<td>1.6</td>
<td>1.7</td>
</tr>
<tr>
<td>camera</td>
<td>5°5cm</td>
<td>0.6</td>
<td>1.8</td>
<td>8.4</td>
<td>10.1</td>
</tr>
<tr>
<td></td>
<td>IoU25</td>
<td>12.6</td>
<td>30.9</td>
<td>91.0</td>
<td>87.6</td>
</tr>
<tr>
<td></td>
<td>R_{err}</td>
<td>90.6</td>
<td>45.2</td>
<td>43.9</td>
<td>35.7</td>
</tr>
<tr>
<td></td>
<td>T_{err}</td>
<td>3.1</td>
<td>8.5</td>
<td>5.5</td>
<td>5.6</td>
</tr>
<tr>
<td>can</td>
<td>5°5cm</td>
<td>7.1</td>
<td>4.3</td>
<td>25.0</td>
<td>22.6</td>
</tr>
<tr>
<td></td>
<td>IoU25</td>
<td>17.2</td>
<td>42.6</td>
<td>89.9</td>
<td>92.6</td>
</tr>
<tr>
<td></td>
<td>R_{err}</td>
<td>77.0</td>
<td>28.8</td>
<td>12.5</td>
<td>13.9</td>
</tr>
<tr>
<td></td>
<td>T_{err}</td>
<td>4.0</td>
<td>13.1</td>
<td>5.0</td>
<td>4.8</td>
</tr>
<tr>
<td>laptop</td>
<td>5°5cm</td>
<td>25.5</td>
<td>49.2</td>
<td>12.5</td>
<td>22.4</td>
</tr>
<tr>
<td></td>
<td>IoU25</td>
<td>14.8</td>
<td>49.2</td>
<td>91.0</td>
<td>98.1</td>
</tr>
<tr>
<td></td>
<td>R_{err}</td>
<td>94.7</td>
<td>6.5</td>
<td>4.9</td>
<td>4.7</td>
</tr>
<tr>
<td></td>
<td>T_{err}</td>
<td>2.4</td>
<td>4.4</td>
<td>2.5</td>
<td>2.5</td>
</tr>
<tr>
<td>mug</td>
<td>5°5cm</td>
<td>0.9</td>
<td>3.1</td>
<td>22.4</td>
<td>24.1</td>
</tr>
<tr>
<td></td>
<td>IoU25</td>
<td>6.2</td>
<td>3.1</td>
<td>95.2</td>
<td>95.2</td>
</tr>
<tr>
<td></td>
<td>R_{err}</td>
<td>82.8</td>
<td>52.0</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>T_{err}</td>
<td>4.0</td>
<td>6.7</td>
<td>20.3</td>
<td>21.3</td>
</tr>
<tr>
<td>Overall</td>
<td>5°5cm</td>
<td>17.0</td>
<td>13.5</td>
<td>32.5</td>
<td>33.3</td>
</tr>
<tr>
<td></td>
<td>IoU25</td>
<td>16.9</td>
<td>53.0</td>
<td>95.1</td>
<td>94.2</td>
</tr>
<tr>
<td></td>
<td>R_{err}</td>
<td>82.2</td>
<td>30.0</td>
<td>17.1</td>
<td>16.0</td>
</tr>
<tr>
<td></td>
<td>T_{err}</td>
<td>20.2</td>
<td>8.4</td>
<td>3.4</td>
<td>3.5</td>
</tr>
</tbody>
</table>

<5.5cm percentage vs. Starting frame
• Shape Prior Deformation for Categorical 6D Object Pose and Size Estimation

In a nut shell: estimate NOCS with explicit category-level shape prior

• Shape prior: mean (normalized) model point cloud by decoding mean latent vector for each category

• Fuse feature from model point cloud, generated deformation field and correspondence.
Category-Level Pose Estimation

- Shape Prior Deformation for Categorical 6D Object Pose and Size Estimation

- Loss

  - For autoencoder & deformation field: reconstruction loss (Chamfer distance)

  - For $A$: correspondence loss (smooth L1 with gt NOCS)

\[
\begin{align*}
  d_{CD}(M^i_c, \hat{M}^i_c) &= \sum_{x \in M^i_c} \min_{y \in \hat{M}^i_c} \|x - y\|^2_2 + \sum_{y \in \hat{M}^i_c} \min_{x \in M^i_c} \|x - y\|^2_2. \\
  L_{corr}(P, P_{gt}) &= \frac{1}{N_v} \sum_{x \in P} \sum_{i=1,2,3} \begin{cases} 
  5(x_i - y_i)^2, & \text{if } |x_i - y_i| \leq 0.1 \\
  |x_i - y_i| - 0.05, & \text{otherwise}
\end{cases}
\end{align*}
\]
Category-Level Pose Estimation

- Shape Prior Deformation for Categorical 6D Object Pose and Size Estimation

- Experiment on NOCS dataset

<table>
<thead>
<tr>
<th>Data</th>
<th>Method</th>
<th>mAP 3D50</th>
<th>mAP 3D75</th>
<th>mAP 5° 2cm</th>
<th>mAP 5° 5cm</th>
<th>mAP 10° 2cm</th>
<th>mAP 10° 5cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMERA25</td>
<td>Baseline [33]</td>
<td>83.9</td>
<td>69.5</td>
<td>32.3</td>
<td>40.9</td>
<td>48.2</td>
<td>64.6</td>
</tr>
<tr>
<td></td>
<td>Ours (RGB)</td>
<td>93.1</td>
<td>84.6</td>
<td>50.2</td>
<td>54.5</td>
<td>70.4</td>
<td>78.6</td>
</tr>
<tr>
<td></td>
<td>Ours (RGB-D)</td>
<td><strong>93.2</strong></td>
<td><strong>83.1</strong></td>
<td><strong>54.3</strong></td>
<td><strong>59.0</strong></td>
<td><strong>73.3</strong></td>
<td><strong>81.5</strong></td>
</tr>
<tr>
<td>REAL275</td>
<td>Baseline [33]</td>
<td>78.0</td>
<td>30.1</td>
<td>7.2</td>
<td>10.0</td>
<td>13.8</td>
<td>25.2</td>
</tr>
<tr>
<td></td>
<td>Ours (RGB)</td>
<td>75.2</td>
<td>46.5</td>
<td>15.7</td>
<td>18.8</td>
<td>33.7</td>
<td>47.4</td>
</tr>
<tr>
<td></td>
<td>Ours (RGB-D)</td>
<td>77.3</td>
<td>53.2</td>
<td>19.3</td>
<td>21.4</td>
<td><strong>43.2</strong></td>
<td><strong>54.1</strong></td>
</tr>
</tbody>
</table>
Model-Free Pose Estimation

- Latent Fusion

- Reconstruct with reference frames
  - Estimate 2D feature, lift to 3D feature voxels

- Feature aggregation across views

- Inference with optimization
  - Estimate initial translation, sample rotation

- Refine pose by optimization over depth error and latent error
Model-Free Pose Estimation

- Latent Fusion
- Experiment on MOPED dataset
- Comparable results with model-based approach

<table>
<thead>
<tr>
<th></th>
<th>PoseRBPF [6]</th>
<th>IBR-LD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td>Textured 3D Mesh</td>
<td></td>
</tr>
<tr>
<td><strong>Training</strong></td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td><strong># Networks</strong></td>
<td>Per-Object</td>
<td></td>
</tr>
<tr>
<td><strong>Pose Loss</strong></td>
<td>-</td>
<td>(\mathcal{L}<em>{\text{latent}} + \mathcal{L}</em>{\text{depth}})</td>
</tr>
<tr>
<td>black_drill</td>
<td>59.78 82.94 49.80</td>
<td>56.67 79.06 53.77</td>
</tr>
<tr>
<td>cheezit</td>
<td>57.78 82.45 48.47</td>
<td>61.31 91.63 55.24</td>
</tr>
<tr>
<td>duplo_dude</td>
<td>56.91 82.14 47.11</td>
<td>74.02 89.55 52.49</td>
</tr>
<tr>
<td>duster</td>
<td>58.91 82.78 46.66</td>
<td>49.13 91.56 19.33</td>
</tr>
<tr>
<td>graphics_card</td>
<td>59.13 83.20 49.85</td>
<td>80.71 91.25 67.71</td>
</tr>
<tr>
<td>orange_drill</td>
<td>58.23 82.68 49.08</td>
<td>51.84 70.95 46.12</td>
</tr>
<tr>
<td>pouch</td>
<td>57.74 82.16 49.01</td>
<td>60.43 89.60 49.80</td>
</tr>
<tr>
<td>remote</td>
<td>56.87 82.04 48.06</td>
<td>55.38 94.80 37.73</td>
</tr>
<tr>
<td>rinse_aid</td>
<td>57.74 82.53 48.13</td>
<td>65.63 92.58 28.61</td>
</tr>
<tr>
<td>toy_plane</td>
<td>62.41 85.10 49.81</td>
<td>60.18 90.24 51.70</td>
</tr>
<tr>
<td>vim_mug</td>
<td>58.09 82.38 48.08</td>
<td>30.11 80.76 14.38</td>
</tr>
<tr>
<td><strong>mean</strong></td>
<td>58.51 82.76 48.55</td>
<td>58.67 87.45 43.35</td>
</tr>
</tbody>
</table>
Model-Free Pose Estimation

- Category Level Object Pose Estimation via Neural Analysis-by-Synthesis

- Approach:
  - Generate synthetic image given object pose and latent code
  - Compare feature metric error between synthesized and observed images to optimize object pose and object shape (latent code)
Model-Free Pose Estimation

- Category Level Object Pose Estimation via Neural Analysis-by-Synthesis

\[ W(T, R_z) : \begin{bmatrix} u \\ v \end{bmatrix} \mapsto \frac{f}{t_z} \cdot \begin{bmatrix} R_z \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \end{bmatrix} \]

\[ \hat{I} = G(R, T, z) = W(T, R_z) \circ G_{3D}(R_x, R_y, z) \]

- Details:
  - Train generator using **synthetic data only**, use some tricks to reduce the need of network capacity
  - Initial pose are randomly sampled...
  - Optimize with deep feature (pre-trained VGG) + regularization on latent code

\[ E(I, R, T, z) = \| F_{vgg}(I) - F_{vgg}(\hat{I}) \|_2 + \| z \|_2, \]
Model-Free Pose Estimation

- Category Level Object Pose Estimation via Neural Analysis-by-Synthesis
- Experiment on NOCS dataset
  - Comparable to NOCS when using RGB-D input
  - Does not require pose annotation
- Compare with LatentFusion
  - Pros: No reference frames needed
  - Cons: can it extend to arbitrary unseen object?
Model-Free Pose Estimation

- Reconstruct Locally, Localize Globally

- Overall target: learn object coordinate estimation without CAD model

- Approach:
  - predict mask, object coordinates and landmark
  - Single-frame + cross frame consistency supervision
Model-Free Pose Estimation

- Reconstruct Locally, Locally Globally
  - What is a landmark?
    - Descriptor with fewer channels.
    - Descriptors that gain ‘robustness’ to intra-class variations
  - In this work: make no difference with using descriptor…
  - May be used to extend the work to category-level (?)

marks [45, 9, 42]. A dense descriptor associates to each image pixel a $C$-dimensional vector, whereas a dense landmark detector associates to each pixel a 2D vector, which is the index of the landmark in a $(u, v)$ parameterisation of the object surface. Thus we can interpret a landmark as a tiny 2D descriptor. Due to its small dimensionality, a landmark loses the ability to encode instance-specific details of the appearance, but gains robustness to intra-class variations.
Model-Free Pose Estimation

• Reconstruct Locally, Locally Globally

• Results: Comparable with SoTA approaches

<table>
<thead>
<tr>
<th>method</th>
<th>BB8 w/r</th>
<th>BB8 w/r</th>
<th>SSD-6D w/r</th>
<th>Tekin w/r</th>
<th>DeepIM w/r</th>
<th>Dense-Fusion w/r</th>
<th>Pix2-Pose w/r</th>
<th>PVNet w/r</th>
<th>SSD-6D w/r</th>
<th>LieNet w/r</th>
<th>Ours w/o CAD model</th>
</tr>
</thead>
<tbody>
<tr>
<td>ape</td>
<td>27.9</td>
<td>40.4</td>
<td>65</td>
<td>21.62</td>
<td>77.0</td>
<td>92</td>
<td>58.1</td>
<td>43.62</td>
<td>0.00</td>
<td>38.8</td>
<td>52.91</td>
</tr>
<tr>
<td>benchwise</td>
<td>62.0</td>
<td>91.8</td>
<td>80</td>
<td>81.80</td>
<td>97.5</td>
<td>93</td>
<td>91.0</td>
<td>99.90</td>
<td>0.18</td>
<td>71.2</td>
<td>96.51</td>
</tr>
<tr>
<td>cam</td>
<td>40.1</td>
<td>55.7</td>
<td>78</td>
<td>36.57</td>
<td>93.5</td>
<td>94</td>
<td>60.0</td>
<td>86.86</td>
<td>0.41</td>
<td>52.5</td>
<td>87.84</td>
</tr>
<tr>
<td>can</td>
<td>48.1</td>
<td>64.1</td>
<td>86</td>
<td>68.80</td>
<td>96.5</td>
<td>93</td>
<td>84.4</td>
<td>95.47</td>
<td>1.35</td>
<td>86.1</td>
<td>86.81</td>
</tr>
<tr>
<td>cat</td>
<td>45.2</td>
<td>62.6</td>
<td>70</td>
<td>41.82</td>
<td>82.1</td>
<td>97</td>
<td>65.0</td>
<td>79.34</td>
<td>0.51</td>
<td>66.2</td>
<td>67.30</td>
</tr>
<tr>
<td>driller</td>
<td>58.6</td>
<td>74.4</td>
<td>73</td>
<td>63.51</td>
<td>95.0</td>
<td>87</td>
<td>76.3</td>
<td>96.43</td>
<td>2.58</td>
<td>82.3</td>
<td>88.70</td>
</tr>
<tr>
<td>duck</td>
<td>32.8</td>
<td>44.3</td>
<td>66</td>
<td>27.23</td>
<td>77.7</td>
<td>92</td>
<td>43.8</td>
<td>52.58</td>
<td>0.00</td>
<td>32.5</td>
<td>54.74</td>
</tr>
<tr>
<td>eggbox*</td>
<td>40.0</td>
<td>57.8</td>
<td>100</td>
<td>69.58</td>
<td>97.1</td>
<td>100</td>
<td>96.8</td>
<td>99.15</td>
<td>8.90</td>
<td>79.4</td>
<td>94.74</td>
</tr>
<tr>
<td>glue*</td>
<td>27.0</td>
<td>41.2</td>
<td>100</td>
<td>80.02</td>
<td>99.4</td>
<td>100</td>
<td>79.4</td>
<td>95.66</td>
<td>0.00</td>
<td>63.7</td>
<td>91.98</td>
</tr>
<tr>
<td>holepuncher</td>
<td>42.4</td>
<td>67.2</td>
<td>49</td>
<td>42.63</td>
<td>52.8</td>
<td>92</td>
<td>74.8</td>
<td>81.92</td>
<td>0.30</td>
<td>56.4</td>
<td>75.41</td>
</tr>
<tr>
<td>iron</td>
<td>67.0</td>
<td>84.7</td>
<td>78</td>
<td>74.97</td>
<td>98.3</td>
<td>97</td>
<td>83.4</td>
<td>98.88</td>
<td>8.86</td>
<td>65.1</td>
<td>94.59</td>
</tr>
<tr>
<td>lamp</td>
<td>39.9</td>
<td>76.5</td>
<td>73</td>
<td>71.11</td>
<td>97.5</td>
<td>95</td>
<td>82.0</td>
<td>99.33</td>
<td>8.20</td>
<td>89.4</td>
<td>96.64</td>
</tr>
<tr>
<td>phone</td>
<td>35.2</td>
<td>54.0</td>
<td>79</td>
<td>47.74</td>
<td>87.7</td>
<td>93</td>
<td>45.0</td>
<td>92.41</td>
<td>0.18</td>
<td>65.0</td>
<td>89.24</td>
</tr>
</tbody>
</table>

| average  | 43.6    | 62.7    | 79         | 55.95     | 88.6       | 94               | 72.4          | 86.27     | 2.42       | 65.2       | 82.88            |

Table 2. LineMOD: Percentages of correct pose estimates in ADD-10. * denotes that the object is symmetric and is evaluated in ADD-S. w/r means the pose is refined with 3D model.
Discussion

- Solved problem
  - Pose estimation with sparse keypoint set
  - Instance-level pose estimation
  - Given accurate CAD model and pose annotation

- Some recent trends
  - sparse representation to denser representation
  - instance-level to category level
  - model-based to model-free
Discussion

• Optimal representation of an object?
  • Preferred properties:
    • Available on weakly-textured object
    • Generalizable beyond instance-level
    • Available without accurate geometric models
    • Trackable across time
  • Potential solution:
    • Sparse keypoint
    • Dense coordinate map
    • Silhouette + appearance cue
    • Latent representation
    • Hybrid of geometric/appearance primitives
Discussion

- Optimal representation of an object?
- **How to achieve model-free pose estimation?**
  - Learn to reconstruct geometry without accurate model
  - Neural synthesis to generate RGB image (and depth) for later optimization
Discussion

• Optimal representation of an object?

• How to achieve model-free pose estimation?

• How to achieve category-level pose estimation?
  • With intra-category shape prior, either implicit (encoded in network) or explicit (mean shape)
  • Generalizable neural reconstruction
From academia to industry

• Needs for 6DoF Pose Estimation in real application
  • Accuracy VS. Stability
  • Data VS. Algorithm
  • Scalability intra (or even inter) categories
From academia to industry

• Hierarchy of problem to solve
  • Instance-level object 6DoF pose estimation in varied scenes and on varied sensor without fine-tune and adaptation
    • more of an engineering problem of scalable data collection
  • Category-level (with unseen instance) object 6DoF pose estimation on some common categories
    • worth working as academic problem
  • Estimate the pose of arbitrary unseen object
    • zero-shot learning, final target.. but hard if not impossible
The Solution by Google Media Pipe

- MobilePose
  - Approach
    - Joint 2D detection and regression (for 3D box corners and center)
    - Estimate mask and object coordinate if available to augment feature
  - Data pipeline
    - Hand-annotated for the first frame
    - Propagate along the sequence using camera pose from ARCore
- Instant motion tracking
  - Track the 9 keypoints by motion analysis
Thanks for your Attention

Siyu ZHANG
Research Engineer
ZJU-SenseTime Joint Lab of 3D Vision
zhangsiyu1@sensetime.com