Deep 3D Pose Regression of Real Objects Trained With Synthetic Data

Author: Pau Bramon Mora

Supervisor: Sergio Escalera Guerrero
Outlined

1. Overview
2. Datasets
   2.1. Synthetic Generation
   2.2. Real Dataset
3. Network
   3.1. Network architecture
   3.2. Projection Loss
4. Results
5. Conclusion
1. Overview

Regress the location and rotation of objects from RGB images:

- Training Deep Models exclusively with synthetic images
- Using a multi-task Neural Network architecture with a novel loss called *Projection Loss*
2.1. Synthetic Generation

Main problems when using synthetic images for training:

- Reality Gap
- Designing realistic scenes is tedious work
- Rendering is computationally expensive
2.1. Synthetic Generation

Domain Randomization:
- Background randomization
- Object randomization
- Light simplification and randomization
2.2. Real dataset

A real dataset with three objects for test:

**Toy**
Object with a complex shape. Strange views with many different shades.

**Box**
Box with serigraph. Difficult to generate in Blender.

**Symmetric**
Symmetric object. Problematic for the optimization process.
3.1. Network architecture
### 3.1. Network architecture

<table>
<thead>
<tr>
<th>Loss Type</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bounding Box Loss</td>
<td>$\mathcal{L}_b = |\tilde{b} - b|$</td>
</tr>
<tr>
<td>Location Loss</td>
<td>$\mathcal{L}_t = |\tilde{t} - t|$</td>
</tr>
<tr>
<td>Rotation Loss</td>
<td>$\mathcal{L}<em>r = \frac{1}{P} \sum</em>{x \in \mathcal{M}} |Rx - \tilde{R}x|$</td>
</tr>
</tbody>
</table>

Total Loss

$$\mathcal{L}_{total} = \lambda_b \mathcal{L}_b + \lambda_q \mathcal{L}_q + \lambda_t \mathcal{L}_t + \lambda_{PL} \mathcal{L}_{PL} + \lambda_{reg} \mathcal{L}_{reg}$$
3.2. Projection Loss

Projection Loss to deal with view ambiguities and symmetries
- In these cases, the network should predict a valid solution
- The silhouettes of the two objects should match
- IoU as a measure to compare silhouettes
3.2. Projection Loss

In order to compute the loss using the 3D model point cloud:

- We compute the projected image of the ground truth and predicted pose with the pinhole camera model.
- We compare the two projected images with the IoU metric.

In practice:

- This operation cannot be implemented entirely in TensorFlow.
- We use CUDA C to compute the projected images and we need to define the forward and the backward pass of the operation.
3.2. Projection Loss - Forward pass

Part 1 (TensorFlow operations)

Predicted Location [B,3]
Predicted quaternion [B,4]

3D model [P,3]
Ground Truth Location [B,3]
Ground Truth quaternion [B,4]

Projection of points

\[
\begin{bmatrix}
  u \\
  v \\
  1
\end{bmatrix} =
\begin{bmatrix}
  f_x & 0 & c_x \\
  0 & f_y & c_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  r_{11} & r_{12} & r_{13} & t_1 \\
  r_{21} & r_{22} & r_{23} & t_2 \\
  r_{31} & r_{32} & r_{33} & t_3
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

Part 2 (directly in CUDA C)

Predicted 2D coords (u,v) [B,P,3]

L-\text{IoU} computation

Compute loss with IoU metric:

\[
\text{IoU} = \frac{|A \cap B|}{|A \cup B|}
\]

L-\text{IoU} = 1 - \text{IoU}
3.2. Projection Loss - Backward pass

We need to find:

$$\frac{\partial L_{IoU}}{\partial u_i} = \frac{\partial L_{IoU}}{\partial X_v} \cdot \frac{\partial X_v}{\partial u_i}$$

The first part of the $\frac{\partial L_{IoU}}{\partial X_v}$ is simple to derive [1]:

$$\frac{\partial L_{IoU}}{\partial X_v} = \begin{cases} -\frac{1}{U(X,Y)} & \text{if } Y_v = 1 \\ \frac{1}{U(X,Y)} & \text{otherwise} \end{cases}$$

But the sampling operation is a discrete operation, so it is not differentiable:

$$\frac{\partial X_v}{\partial u_i} = ?? \quad \frac{\partial X_v}{\partial v_i} = ??$$

3.2. Projection Loss - Backward pass

The value of each pixel is approximated to a function that can backpropagate the gradients [2-3].

\[
\frac{\partial X_v}{\partial u_i} = \begin{cases} 
D \cdot \max(0, 1 - |n - D \cdot v_i|) & \text{if } D \cdot u_i \leq m \\
-D \cdot \max(0, 1 - |n - D \cdot v_i|) & \text{if } D \cdot u_i > m \\
0 & \text{if } |n - D \cdot u_i| \geq 1
\end{cases}
\]

Only propagate gradients:
- One 2D point per pixel in the projected image.
- If the change in the coordinate actually causes a change in the IoU.

4. Results - Domain Randomization

<table>
<thead>
<tr>
<th></th>
<th>Synthetic Images</th>
<th>Real Images</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADD</td>
<td>ADD-S</td>
</tr>
<tr>
<td>Complete</td>
<td>67.8</td>
<td>71.9</td>
</tr>
<tr>
<td>Fixed Light</td>
<td>78.4</td>
<td>81.3</td>
</tr>
<tr>
<td>Real Background</td>
<td>81.3</td>
<td>84.7</td>
</tr>
<tr>
<td>Fixed Object</td>
<td>72.5</td>
<td>76.9</td>
</tr>
<tr>
<td>Without Randomization</td>
<td>81.3</td>
<td>84.7</td>
</tr>
</tbody>
</table>
4. Results - Projection Loss

<table>
<thead>
<tr>
<th></th>
<th>ADD</th>
<th>ADD-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Projection Loss</td>
<td>34.5</td>
<td>51</td>
</tr>
<tr>
<td>With Projection Loss</td>
<td>36.2</td>
<td>60.7</td>
</tr>
</tbody>
</table>
4. Results - Projection Loss

<table>
<thead>
<tr>
<th>Original image</th>
<th>Without PL</th>
<th>With PL</th>
<th>Original image</th>
<th>Without PL</th>
<th>With PL</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Original image without PL" /></td>
<td><img src="image2" alt="Without PL" /></td>
<td><img src="image3" alt="With PL" /></td>
<td><img src="image4" alt="Original image without PL" /></td>
<td><img src="image5" alt="Without PL" /></td>
<td><img src="image6" alt="With PL" /></td>
</tr>
</tbody>
</table>
4. Results - Final

<table>
<thead>
<tr>
<th></th>
<th>ADD</th>
<th>ADD-S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toy</td>
<td>43.9</td>
<td>48.7</td>
</tr>
<tr>
<td>Box</td>
<td>69.0</td>
<td>74.3</td>
</tr>
<tr>
<td>Symmetric</td>
<td>36.2</td>
<td>60.7</td>
</tr>
</tbody>
</table>
4. Results - Final

But still...
5. Conclusions

- The 6D pose regression task was addressed with Deep Models trained exclusively with synthetic data.
- The reality gap problem can be tackled with domain randomization.
- The proposed multi-task Neural Network was effective for the rotation and localization regression.
- The new Projection Loss function was able to deal with ambiguities and symmetric views.

Directions of future work

- Improve the domain randomization
- Use Generative Adversarial Networks to add realism to synthetic images.
- Use a Render Loss instead of a Projection Loss
- Multi-view 6D pose regression
Thank you for your attention

Deep 3D Pose Regression of Real Objects Trained With Synthetic Data

Author: Pau Bramon Mora

Supervisor: Sergio Escalera Guerrero