InSeGAN: A Generative Approach to Segmenting Identical Instances in Depth Images

Anoop Cherian¹  Gonçalo Dias Pais²  Siddarth Jain¹  Tim K. Marks¹  Alan Sullivan¹

¹Mitsubishi Electric Research Labs (MERL), Cambridge, MA
²Instituto Superior Tecnico, University of Lisbon, Portugal

ICCV Virtual, 2021
Pick an Instance?
Pick an Instance?
Pick an Instance?
Identical Instance Segmentation Problem

Problem setup:
• A collection of depth images
  - Each with multiple instances of the same rigid object
• Unsupervised: No ground truth instance labels for learning
InSeGAN (Instance Segmentation GAN) Approach: Analysis by Synthesis

**Synthesis:** GAN learns to produce outputs that look like real depth images.

**Analysis:** Encoder learns to input a realistic depth image and output instance pose parameters.
InSeGAN Generator: Instance Poses to Depth Image

Instead of one noise vector, input is $n$ noise vectors. System learns to associate each vector with one instance.
InSeGAN Analysis: Depth Image to Instance Poses

- Instance pose encoder $E$
  - takes in a synthesized multiple-instance depth image $\hat{x}$
  - produces pose vectors $\hat{Z}$ that could have produced $\hat{x}$.
Optimal Transport (OT) Alignment Loss

\[ \mathcal{L}_E^a = \| Z - \pi^*(\hat{Z}) \|^2, \text{ where } \pi^* = \arg \min_{\pi \in \Pi(Z, \hat{Z})} \text{OT}(\pi, D(Z, \hat{Z})) \]

\[ Z = \{ z_1, ..., z_n \} \]

\[ \hat{Z} = \{ \hat{z}_1, ..., \hat{z}_n \} \]
Pose Decoding Loss

\[ \hat{Z} = \{\hat{z}_1, ..., \hat{z}_n\} \]

\[ Z = \{z_1, ..., z_n\} \]

\[ \mathcal{L}_E^p = \|G(Z) - G(E(\hat{x}))\|_1 \]
Inference: Instance Segmentation

At test time:
- Input depth image is passed through the pose encoder to get the latent vectors
- Latent vectors are then decoded and rendered one at a time, each rendering a single instance
- Rendered instances are thresholded and transformed into segment masks
Experiments and Results
New Insta-10 Synthetic Dataset

Insta-10 dataset has **10 object classes**, each defined by a 3D object CAD model
Each class has **10,000 depth images**
Each image has **5 object instances in varying poses, occlusions, etc.**
We programmed a Fetch robot to shake a box containing 4 wooden blocks. The depth images were captured using a RealSense RGB-D camera.

- The evaluation test images are hand annotated using the LabelMe tool.
- Collected 3,000 depth images. Annotated 62 images for evaluation.
Comparisons to the State of the Art

<table>
<thead>
<tr>
<th>Method</th>
<th>Nut</th>
<th>Stop.</th>
<th>Cyl.</th>
<th>Bolt</th>
<th>Cone</th>
<th>Conn.</th>
<th>5-pin</th>
<th>Obj01</th>
<th>Obj14</th>
<th>Obj05</th>
<th>Avg mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-Deep Learning Methods</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-Means</td>
<td>0.64</td>
<td>0.297</td>
<td>0.7</td>
<td>0.18</td>
<td>0.35</td>
<td>0.554</td>
<td>0.628</td>
<td>0.208</td>
<td>0.496</td>
<td>0.59</td>
<td>0.464</td>
</tr>
<tr>
<td>Spectral Clustering [31]</td>
<td>0.56</td>
<td>0.36</td>
<td>0.54</td>
<td>0.22</td>
<td>0.41</td>
<td>0.56</td>
<td>0.58</td>
<td>0.25</td>
<td>0.47</td>
<td>0.57</td>
<td>0.452</td>
</tr>
<tr>
<td>GrabCut [36]+KMeans</td>
<td>0.572</td>
<td>0.232</td>
<td>0.572</td>
<td>0.472</td>
<td>0.231</td>
<td>0.519</td>
<td>0.497</td>
<td>0.597</td>
<td>0.557</td>
<td><strong>0.605</strong></td>
<td>0.486</td>
</tr>
<tr>
<td>GraphCut [3]</td>
<td>0.569</td>
<td>0.1</td>
<td>0.589</td>
<td>0.447</td>
<td>0.12</td>
<td>0.476</td>
<td>0.12</td>
<td>0.597</td>
<td>0.540</td>
<td>0.511</td>
<td>0.373</td>
</tr>
<tr>
<td><strong>Deep Learning Methods</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wu et al. [41]</td>
<td>0.45</td>
<td>0.28</td>
<td>0.57</td>
<td>0.27</td>
<td>0.33</td>
<td>0.38</td>
<td>0.43</td>
<td>0.23</td>
<td>0.44</td>
<td>0.57</td>
<td>0.385</td>
</tr>
<tr>
<td>IODINE [9]</td>
<td>0.026</td>
<td>0.059</td>
<td>0.019</td>
<td>0.040</td>
<td>0.089</td>
<td>0.032</td>
<td>0.034</td>
<td>0.058</td>
<td>0.053</td>
<td>0.118</td>
<td>0.053</td>
</tr>
<tr>
<td>Slot Attn. [29]</td>
<td>0.375</td>
<td>0.276</td>
<td>0.535</td>
<td>0.43</td>
<td><strong>0.68</strong></td>
<td><strong>0.662</strong></td>
<td>0.628</td>
<td>0.655</td>
<td><strong>0.622</strong></td>
<td>0.481</td>
<td>0.535</td>
</tr>
<tr>
<td>InSeGAN (2D)*(ours)</td>
<td>0.215</td>
<td><strong>0.365</strong></td>
<td>0.258</td>
<td>0.524</td>
<td>0.435</td>
<td>0.585</td>
<td>0.628</td>
<td>0.365</td>
<td>0.286</td>
<td>0.532</td>
<td>0.419</td>
</tr>
<tr>
<td>InSeGAN (3D) (ours)</td>
<td><strong>0.773</strong></td>
<td>0.301</td>
<td><strong>0.760</strong></td>
<td><strong>0.539</strong></td>
<td>0.47</td>
<td>0.655</td>
<td><strong>0.642</strong></td>
<td><strong>0.686</strong></td>
<td>0.591</td>
<td>0.483</td>
<td><strong>0.590</strong></td>
</tr>
</tbody>
</table>

*InSeGAN2D baseline did not use the 3D template or the pose decoder modules, instead using the noise vector to directly produce a single instance feature vector.*
More Results

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>KMeans</td>
<td>0.797</td>
</tr>
<tr>
<td>Spectral Clustering</td>
<td>0.668</td>
</tr>
<tr>
<td>Graph Segmentation [7]</td>
<td>0.436</td>
</tr>
<tr>
<td>InSeGAN</td>
<td>0.857</td>
</tr>
</tbody>
</table>

Results on Real Data
Qualitative Segmentation Results

Top row: Input depth image. Middle row: Image rendered by InSeGAN. Last row: Segmentations produced by InSeGAN.

See paper for more results
Instance Pose Disentanglement

Depth image input
Instance Pose Disentanglement

Depth image input

Rotating a single instance
Instance Pose Disentanglement

Depth image

Rotating all instances together

Rotating a single instance

Translating a single instance
Thank you!

For questions, please contact us at cherian@merl.com

PyTorch implementation of InSeGAN and the Insta-10 dataset are publicly available at https://www.merl.com/research/