Synthetic data generation for camera-based perception

Jaewon Jung
Chief Architect / Director
Baidu USA
Overview

1. Why synthetic dataset?
2. High fidelity virtual world creation
3. Precise ground truth generation
4. Future work
1. Why synthetic dataset?
1.1. Why synthetic dataset?

• Can easily & quickly generate precise ground truth
  • No manual labeling which is error-prone and time-consuming
  • Some 3D ground truth is impossible to get without the help of an additional sensor like lidar.

• Dynamic variations
  • Time of day
  • Weather
  • Road degradation
  • Vehicle color
  • Random spawning of traffic barriers
### 1.2. Existing synthetic datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Year</th>
<th>#</th>
<th>Res</th>
<th>Diversity</th>
<th>Ground truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>VKITTI</td>
<td>2016</td>
<td>21k</td>
<td>1242x375</td>
<td>5 urban scenes under different imaging and weather conditions</td>
<td>2D/3D box, semantic/instance-level segmentation, optical flow, depth</td>
</tr>
<tr>
<td>Playing for Data</td>
<td>2016</td>
<td>25k</td>
<td>1914x1052</td>
<td>Diverse scenes from GTA 5 under different times of day / weather conditions</td>
<td>Semantic segmentation</td>
</tr>
<tr>
<td>Synthia</td>
<td>2016</td>
<td>213k</td>
<td>1280x760</td>
<td>Urban / highway / green area scenes under different times of day / weather conditions / seasons</td>
<td>Semantic segmentation, depth</td>
</tr>
<tr>
<td>FCAV</td>
<td>2017</td>
<td>200k</td>
<td>1914x1052</td>
<td>Diverse scenes from GTA 5 under different times of day / weather conditions</td>
<td>2D box, segmentation</td>
</tr>
<tr>
<td>Playing for Benchmarks</td>
<td>2017</td>
<td>250k</td>
<td>1920x1080</td>
<td>Diverse scenes from GTA 5 under different times of day / weather conditions</td>
<td>2D/3D box, semantic/instance-level segmentation, optical flow</td>
</tr>
</tbody>
</table>
1.3. Challenges

• How to efficiently create a new scene / world

<table>
<thead>
<tr>
<th></th>
<th>Digital double</th>
<th>Imaginary creation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>Replication by artists</td>
<td>Creation by artists</td>
</tr>
<tr>
<td>Automated</td>
<td>Data driven generation</td>
<td>Procedural generation</td>
</tr>
</tbody>
</table>

• Domain gap (between real and synthetic)
  • Visual quality improvements
  • Domain adaptation
  • Domain randomization
  • Image-to-image translation
2. High fidelity virtual world creation
2.1. Photogrammetry
2.1. Photogrammetry

High Rez Scan  ——  Retopology  ——  Baked Maps
2.2. Reference
### 2.3. Unity

<table>
<thead>
<tr>
<th>Code</th>
<th>C++ based</th>
<th>C# based (core in C++)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Licensing model</td>
<td>Royalty based, free version, full source code access</td>
<td>Subscription based, free version</td>
</tr>
<tr>
<td>Games</td>
<td>Fortnite, PUBG, Gears of War 4, Tekken 7, Mass Effect, BioShock, ...</td>
<td>NASCAR Heat 2, Pokémon Go, Monument Valley, RollerCoaster Tycoon World, ...</td>
</tr>
<tr>
<td>AV sim</td>
<td>CARLA, NVIDIA Drive Sim, monoDrive, AirSim</td>
<td>Metamoto, Cognata, SynCity, AirSim</td>
</tr>
</tbody>
</table>
2.4. Traffic simulation

- Scripted actors (vehicles & pedestrians)
  - Longitudinal collision avoidance using IDM (intelligent driver model)
  - Lane change
  - Pedestrians
  - Signal awareness

- Data-driven multi-agent traffic simulation
3. Precise ground truth generation
3.1. Ground truth data

- Semantic / instance-level segmentation
- Depth image
- Object ground truth with 2D / 3D bounding boxes
  - For vehicles, pedestrians, and traffic barriers
- 3D lane line ground truth
3.2. 2D bounding box

- Projecting 3D bounding box
- 2D convex hull from 3D convex hull
- “Amodal perception”
3.3. Truncation

- Also relies on 2D convex hull
- Calculates what portion of the object is cropped (out of the image) both horizontally and vertically.
3.4. Occlusion

• How much of an object is occluded in the shot?
• Usually discrete (approximate) levels in human labeling
• You can compute an exact occlusion ratio for synthetic data:
  • Render each object separately to get its unoccluded pixel count (using pixel & compute shaders on GPU) (i.e. amodal perception)
  • Count the visible pixels with a regular rendering (again on GPU)
  • Get the ratio of these two -> Naïve (slow), but works!
  • Looking into an approach using MRT (multiple render targets) and bitwise logical blending operation to improve the generation speed
3.5. Depth image

- Z coordinate of each pixel in camera coordinate space, not the distance to the camera optical center
- Compute 16-bit depth values assuming a fixed far plane of 655.35m to get a precision of 1cm
- A 16-bit value is encoded into R/G channels of a pixel in a regular 24-bit 8-bit-per-channel png image:
  - Encoding: $v = \frac{\text{depth}}{655.36}$; $R = v - \frac{255.0 \times v}{255.0}$; $G = \frac{255.0 \times v}{255.0}$
  - Decoding: $(R + G / 255.0) \times 655.36$
  ($R$, $G$ are normalized float pixel values (0~1) in red / green channel.)
3.6. Segmentation image

• Semantic / instance-level segmentation combined into one image
• Instance-level segmentation is provided only for vehicles and pedestrians.
• Require tagging all relevant objects in each 3D scene by categories
  • Sedan, Coupe, SUV, Hatchback, Van, PickupTruck, Truck, Bus, Cyclist, Motorcyclist, Pedestrian, TrafficCone, Barricade, Road, LaneMarking, TrafficSign, TrafficLight, Sidewalk, GuardRail, Sky, Terrain, Pole, StreetLight, Building, Vegetation
• For human-friendliness, we keep the hue constant for a single category while varying saturation / value for instance variation.
  • (an idea from VKITTI dataset)
3.7. 3D lane line ground truth

- Visible portion of each lane line is sampled regularly along its 3D length and outputted as a sequence of points with additional annotations.
- 2D / 3D positions of the sample
- Ego-centric lane index (-4, -3, -2, -1, 1, 2, 3 4)
- Lane marker type (single solid, single dash, double solid, ...), color (yellow, white), topology info (fork / merge)
3.8. Camera sensor simulation

• Lens distortion
• Dynamic range mapping
• Tone mapping
• Low-light noise
• ...

3.8. Camera sensor simulation

Lens distortion

- Radial (barrel / pincushion) / tangential distortion
- Two possible approaches to get a distorted image
  - Backward: Use inverse distortion mapping to sample proper
    texels in undistorted image
    - “An Exact Formula for Calculating Inverse Radial Lens Distortions”
  - Forward: Use a 2D grid mesh (e.g. 20x20) with undistorted
    image applied as a texture and distort the grid vertices
  - Start with a bigger resolution (than the sensor resolution)
    because a barrel distortion shrinks the image
4. Future work
4. What’s next?

• Online grading of perception algorithm
• Whole-stack closed loop simulation
• Support more sensors (lidar, radar, ultrasonic)
• Synthetic data generation as a service
• Simulation as a service
• User-friendly tools for ego vehicle configuration and scenario editing
Game Engine Based Simulation
Shout-out

• Kathy Fu, Guodong Rong, Peitao Zhao
Questions?

- A synthetic data can provide diverse & varied scenes with precise ground truth.
- Our synthetic dataset coming soon!
- jaewonjung@baidu.com
- https://www.linkedin.com/in/all2one/
- @all2one