Motivation and Main Problem

Rendering is the process of generating images of 3D scenes defined by geometry, materials, scene lights and camera properties.
Motivation and Main Problem

• Optimizing 3D scene parameters:
Motivation and Main Problem

- 3D self-supervision pipeline:
  - Image-based training of 3D object reconstruction.
  - Human/hand pose estimation.
  - Face reconstruction, etc.
Contributions

Review the following aspects of differentiable rendering (DR):

- **DR’s mechanisms**: understanding which methods are suitable for addressing certain types of problems.

- **Evaluation methodology**: in order to choose or develop novel DR method.

- **Applications and usage**: in order to use DR in novel task.

- **DR libraries**: facilitate real-time applications or embedded devices.
General Background

- Object representation
General Background

• Rendering step

Frustum Projection

✓ differentiable

Rasterization

✗ non-differentiable
General Background

• **Rasterization**
  • assigning a triangle to the pixel
  • computing the pixel color based on the assigned triangle’s vertices’ colors.

Rasterization is non-differentiable
Problem Setting

**Input:** Shape, Camera, Material, Lighting parameters $\Phi_s, \Phi_c, \Phi_m, \Phi_l$

**Output:** Rendered RGB image $I_c$ or depth image $I_d$

**Rendering function:** $R(\Phi_s, \Phi_c, \Phi_m, \Phi_l) \rightarrow I_c, I_d$

**Problem:** compute gradients of output image w.r.t. input parameters $\frac{\partial I}{\partial \Phi}$

The computation of the gradients can be approximate, but should be accurate enough to minimize the objective function.
Approach (Mesh)

- Approximated Gradients: Neural 3D Mesh Render, OpenDR, etc.
Approach (Mesh)

- Approximated Rendering: Soft Rasterizer, DIB-R, etc.
Approach (Voxel)

Ray Marching Pipeline:
1. Collecting the voxels that are located along a ray
2. Aggregating voxels along a ray

- Perspective transformer nets
- Differentiable Ray Consistency
- Neural volumes
- SDFDiff
- ...
Approach (Point Cloud)

Rendering Pipeline:
1. 3D point world coordinate -> screen space coordinate
2. Compute influence of 3D point on target pixel’s color
3. Aggregate based on influence and z-values
Approach (Implicit surface)

Similar to voxel-based methods, include:

- Sample points along ray (this is challenging because of infinite resolution)
- Check intersection points.

Sphere tracing is efficient
Evaluation Metrics

• Direct gradient evaluation:
  • Lack a common dataset.
  • Some papers focus on approximated gradients.

• Visualization gradients and analyze convergence efficiency.

• Evaluate optimized scene parameters:
  • Lack a common dataset.

• Evaluate 3D reconstruction accuracy.

• Computation time, especially for ray tracing based renderer.
Applications

Object Reconstruction

Inference

Input image → Neural network → 3D model

Camera parameters

Rendered images

Differentiable rendering

Reconstruction loss

Multi-view images
Applications

Human Reconstruction

(a) Training on real images
(b) Training on human shape instances
(c) End-to-end finetuning on real images
## Libraries

### DR library:
- **TensorFlow Graphics**
- **Kaolin**
- **PyTorch3D**
- **Mitsuba 2**

### Non-DR library:
- **Direct3D**
- **OpenGL**
- **Vulkan**
- **Unity**
- **Unreal Engine**

<table>
<thead>
<tr>
<th></th>
<th>PyTorch 3D</th>
<th>Kaolin</th>
<th>TensorFlow Graphics</th>
<th>Mitsuba 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core implementation</td>
<td>PyTorch/CUDA</td>
<td>PyTorch/CUDA</td>
<td>TensorFlow/C++/GPU</td>
<td>C++/CUDA Python bindings</td>
</tr>
<tr>
<td>Supported primitives</td>
<td>Triangle mesh</td>
<td>Triangle/quad mesh</td>
<td>Triangle mesh</td>
<td>Triangle mesh</td>
</tr>
<tr>
<td></td>
<td>Point cloud</td>
<td>Point cloud</td>
<td>Point cloud</td>
<td>Custom</td>
</tr>
<tr>
<td></td>
<td>Voxel grid</td>
<td>SDF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Differentiable</td>
<td>Soft Rasterizer</td>
<td>NMR</td>
<td>Analytical</td>
<td>Loubet et al.</td>
</tr>
<tr>
<td>rendering</td>
<td>(extendable)</td>
<td>Soft Rasterizer</td>
<td>derivatives</td>
<td></td>
</tr>
<tr>
<td>algorithms</td>
<td></td>
<td>DIB-Renderer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rendering Method</td>
<td>Rasterization</td>
<td>Rasterization</td>
<td>Rasterization</td>
<td>Ray tracing</td>
</tr>
<tr>
<td>3D operations</td>
<td>Graph convolutions</td>
<td>Primitive conversions</td>
<td>Graph convolutions</td>
<td>3D transformations</td>
</tr>
<tr>
<td></td>
<td>3D transformations</td>
<td>3D transformations</td>
<td>3D transformations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Point Cloud Operations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Uneyama, ICP, KNN)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shader support</td>
<td>Hard/soft Phong</td>
<td>Phong</td>
<td>Phong</td>
<td>Wide range BSDF</td>
</tr>
<tr>
<td></td>
<td>Hard/soft Gouraud</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hard flat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Soft silhouette</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lighting support</td>
<td>Point</td>
<td>Ambient</td>
<td>Point</td>
<td>Area</td>
</tr>
<tr>
<td></td>
<td>Directional</td>
<td>Directional</td>
<td>Spherical harmonic</td>
<td>Point</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Spot</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Constant environment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Environment map</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Directional</td>
</tr>
<tr>
<td>Loss functions</td>
<td>Chamfer Distance</td>
<td>Chamfer distance</td>
<td>Chamfer distance</td>
<td>Not supported</td>
</tr>
<tr>
<td></td>
<td>Mesh edge</td>
<td>Directed distance</td>
<td>Mesh Laplacian</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Laplacian smoothing</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Normal consistency</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Camera support</td>
<td>Perspective</td>
<td>Perspective</td>
<td>Perspective</td>
<td>Perspective</td>
</tr>
<tr>
<td></td>
<td>Orthographic</td>
<td>Orthographic</td>
<td>Orthographic</td>
<td>(pinhole, thin lens)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Quadratic distortion</td>
<td>Irradiance meter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Radiance meter</td>
</tr>
</tbody>
</table>
Open Issues

• The graphic model is naïve compared to natural image generation, cannot produce photorealistic images.

• Differentiable rendering of videos. The integration of a physics simulator is preferred.

• Incorporating learning-based methods into differentiable rendering is also worth considering.
Contributions (Recap)

Review the following aspects of differentiable rendering (DR):

- **DR’s mechanisms**: understanding which methods are suitable for addressing certain types of problems.

- **Evaluation methodology**: in order to choose or develop novel DR method.

- **Applications and usage**: in order to use DR in novel task.

- **DR libraries**: facilitate real-time applications or embedded devices.