Use PBRT and ISET3d to create spectral radiance data for ML in automotive applications

Introduction:

In the automotive field, machine learning algorithm or particularly deep neural network handles tasks such as image segmentation and reconstruction, object detection and classification, navigation and decision, etc. Just like cortex in our brain reads in spectral information through our eyes and afterwards processes by magic into vivid images, self-driving vehicles and other autonomous systems dramatically relied on machine learning and high-speed computing to reach an advanced level of autopilot. 2

Since the data-driven nature of machine learning, it is super critical to create a collection of feasible data to train a model and evaluation its accuracy in a large amount of real scenarios before launching the solution directly. However, in the automotive industry, because of safety, regulation and cost consideration, samples of dataset including collisions or accidences, extreme weather or traffic conditions might not be included in training dataset. Machine learning with computational graphics serves as a promising medium to automatically generate and simulate these but not limited in normal conditions as a tutorial for a new model.

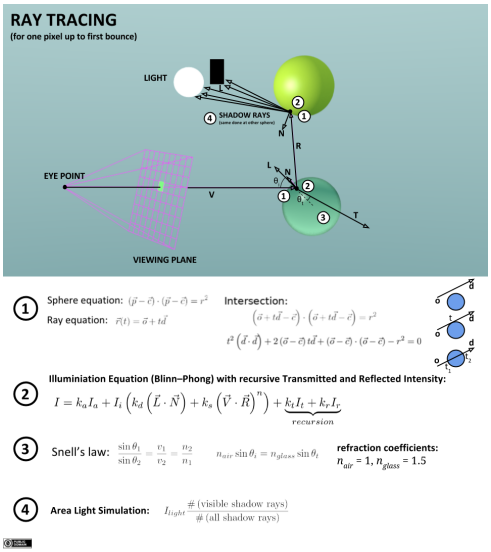
Background:

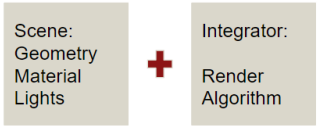
1. PBRT

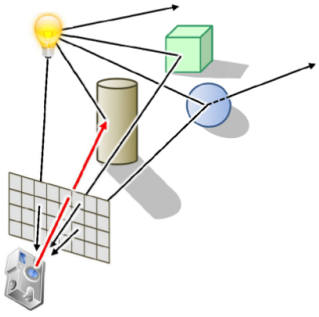
PBRT stands for physically based rendering. It uses principles of physics to simulate reality in the interaction of light and objects based on ray-tracing algorithm. The pixels from the perspective of camera trace a path from the scene, once the closest object is encountered by this ray, based on the material and color information of the pixel, the integrator would render the scene and would also embed reflective and translucent nature of materials in the final calculation of illumination.

A basic illumination equation(Blinn-Phong) with recursive transmitted and reflected intensity describes how the virtual camera views the scene: It breaks up lights into three parts: diffuse, specular and ambient reflection.

For diffuse reflection, it refers to almost uniform scattering of light like in non-shiny material. $\mathbf{I_d} = \mathbf{I_i K_d} \vec{\mathbf{L}} \cdot \vec{\mathbf{N}}$ It is a combination of lights diffuse color, material diffuse color and dot product of normalized direction to the light and surface normal. For specular reflection, it represents the shiny thing on the surface and would take account into angle between eye and light. $\mathbf{I_s} = \mathbf{I_i K_s} \vec{\mathbf{V}} \cdot \vec{\mathbf{R}}^{\mathbf{n}}$ It includes specular reflection, material specular color and dot product of view vector and refection vector with a special shininess N parameter to control the tight of scattering. For ambient reflection, It approximate the interreflections from light through other scenes. $\mathbf{I_a} = \mathbf{m_a L_a}$







2. ISET3D

ISET3D work with pbrt-v3-spectral through Docker and Matlab. Depended on ISETCam, this tool enables to simulate how an image would look when a 3D scene is captured by a camera and optics system.

Results:

A. Data label categories

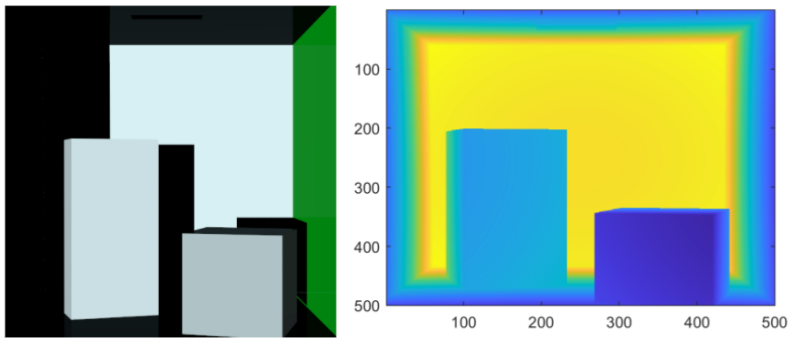
Four kinds of data type could be rendered from ISET3D together from PRBT. These data is stored as 2D array values and later could be compared with training results.

mesh = piRender(pbrtFile,'render type','mesh');

radiance = piRender(pbrtFile,'render type','radiance');

depth = piRender(pbrtFile,'render type','depth');

mat = piRender(pbrtFile,'render type','material');

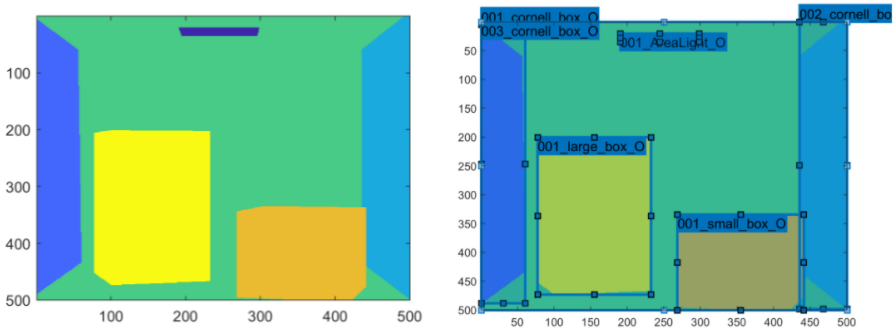


radiance image from cornell box depth image from cornell box

B. Data visualization

From mesh data or other metadata in Iset3D, we could generate image labels to show up our interested label area.

```
mesh = piRender(pbrtFile,'render type','mesh');
bbox = regionprops(mesh,'BoundingBox')
I = openfig(strcat(data,'_mesh.fig'))
table = readtable(strcat(data,'_mesh_mesh.txt'))
for k =1:numel(bbox)
    h = images.roi.Rectangle(gca,'Position',bbox(k).BoundingBox)
    h.Label = string(table.Var2(k))
end
```

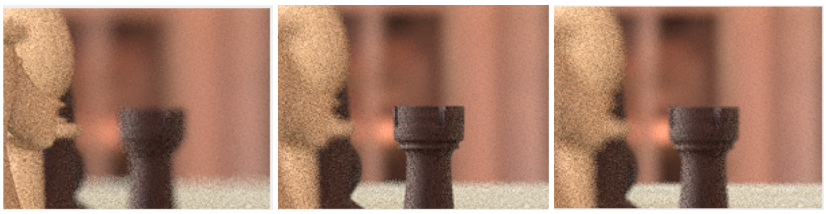


C. Dataset augmentation with lens focus

Together with Isetlens and json based lens files. And by using "thisR.set('focus distance',0.45); % meters" in Iset3D and adjust film diagonal based on filed of view.

```
filmDistance = thisR.get('film distance');
filmDiagonal = filmDistance*tand(fov/2)*2;    % mm
thisR.set('film diagonal',filmDiagonal*1e3);
```

It enables dataset lens focus change in the scene.



distance 0.4m distance 0.8m distance 1m

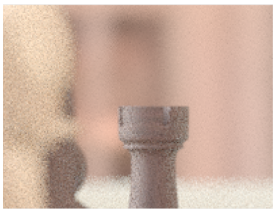
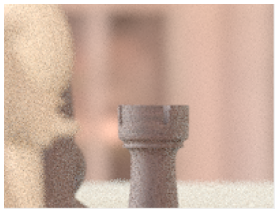
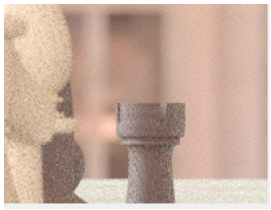
D. Dataset augmentation with aperture

The direct increase of exposure only increase illumination levels which affects noise but its SNR improvement might not always very obvious in the view.

In this case, besides change exposure time straight forward, the aperture diameter related to camera speed is modified according to the time needed during the photography. The field of view(or less blurred) increases with smaller aperture size.

thisR.set('aperture diameter',ap);

thisR.set('camera exposure', 0.5);

6mm aperture 3mm aperture 1mm aperture

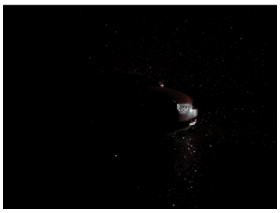
An interesting point here, is that if the image object is translated or rotated during the camera exposure time, the HDR image could also reflect a new dataset with blurred view. However, wheter this is similar to normal rolling shutter image sensor behavior is hard to say.

E. Data Augmentation with light condition change.

By applying a script into light.prdf, we could automatically rendering a scene with different light conditions.



night condition environment reflection skylight-morn



distant rgbL spectrumL D65_1.spd spectrumL D65_1.spd with HDR

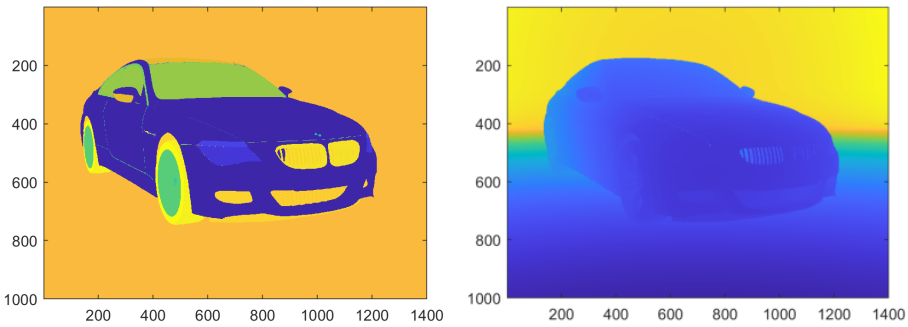
Obviously, how we setup up light conditions, use camera in burst and HDR mode and even the basic PRBT algorithm for ray tracing

```
thisR.set('film resolution',[200 150]);
thisR.set('rays per pixel',32);
%thisR.set('fov',45);
thisR.set('nbounces',5);
```

is associated with 3D scene image qualities.

F. Datasource toward PBRT

-https://pbrt.org/scenes-v3



sample BMW-m6

-Blender model to pbrt

https://github.com/ISET/iset3d/wiki/Blender

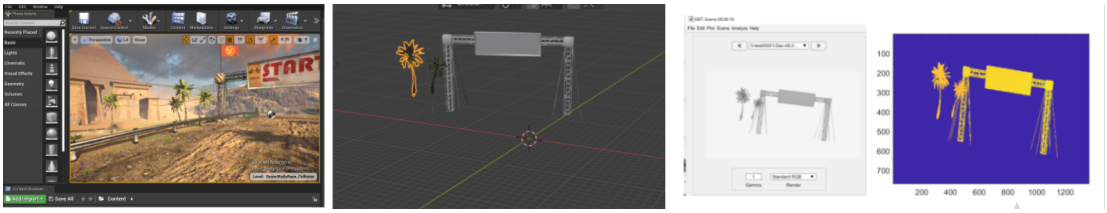
the generated file might not fully compatible with rendering method in Iset3D, some extra parse work for PBRT is necessary.

-Unreal Engine AIRSIM to blender

1.generate .fbx output to blender

2.bake in blender

3.use plugin to PBRT



Unfortunately, it does not generate material and lights pbrt files directly.

Conclusions:

PBRT and Iset3D is very useful to produce a set of new dataset with cameras and lens information from baseline dataset. However, the source of prbt seems very hard to be obtained from internet, to facilitates all designers not only computer graphics designers, a PBRT scene drawer and PBRT model with physically-based dynamically world construction is really welcomed in the near.

References:

1. Physically Based Rendering: From theory to implementation 3rd: https://www.pbr-book.org/3ed-2018
2. Tesla AI day https://www.youtube.com/watch?v=j0z4FweCy4M
3. https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_reflection_model https://en.wikipedia.org/wiki/Ray_tracing_(graphics)
4. https://github.com/ISET/iset3d/wiki
5. https://www.nvidia.com/en-us/on-demand/session/omniverse2020-om1358/?playlistId=playList-84a0a8ef-d253-4221-8a3c-abf506de3923
6. https://github.com/mmp/pbrt-v4
7. https://airsim-fork.readthedocs.io/en/docs/build_windows.html

Appendix:

Thanks so much for lectures, instructions and knowledge from Professors and TAs during the course!