Illumination Prediction
Midterm Report

Team Member: Christopher Chu, Yan Xiao, Xiaoxi Sun

Review of Our Project

Inspired by this paper, we will investigate a lightweight parametric illumination predictor based on Convolutional Neural Network (CNN). Given an input HDR image, we want to estimate the parameters for N light sources in the image. For now, N = 3, and we use point light, ambient light, spotlight and directional light to model the light sources. The parameters include the position, size, rotation, and RGB values. The parameters are represented as an 8 by 1 vector. The output would be a 8N by 1 vector where N is the number of light sources.

We would train the CNN with parts of the environment map to estimate light sources beyond images. The cropped images may not contain the light source, but they are illuminated by the source external to the image.

Our source code is here.

The development of our Proposal:

In our original proposal, we stated that we want to improve on performance and the consistency on estimating lighting information on AR devices. Our goal hasn't change from our original proposal and we will continue to pursue it. Our approach is more concrete, and the path forward is clearer.

- We break the performance into two parts, the time need to run through the network, the time need to render the object. For the first part, we are trying to use simpler pretrained network. Our main focus is the second part. We are using more render engine friendly type of lights and make the prediction in real-time possible. Area lights are not support as real time lights in many render engines. This approach gives us another benefit that we extract less information from the input and thus we may reduce the complexity of the network.
- We want to first solve the performance issue, then focus on the consistency. Some ideas we have right now is to discard unconfident output, interpolate between frames, compensating user's movement.

Current Progress:

Dataset:

The dataset for illumination prediction is from here. This dataset contains over 2100 HDR indoor environment maps. The resolution of each image is 3844x7768, and the size of the entire dataset is 154 GB. Also, this dataset does not contain any label information. Therefore, there are two important considerations: One is to reduce the input size to make the training of CNNs viable on our machines; the other is to design algorithms to label images with illumination source parameters.

Pre-processing:

1. Reduce the workload

In order to address the large size of the dataset, we decided two general approaches. The first approach is to reduce the resolution, i.e., resizing the images to smaller resolution. We resize each HDR image to a resolution of 400x900 and keep its HDR property, i.e., light intensities are still represented by floating points.

For testing purposes, we convert HDR images to jpeg files to test our network architecture. But in the end, we would not use LDR images to train our network.

2. Create Labels

We assumed that light sources in an environment map are in the brightest local areas. The number of light sources is fixed to be 3 (We would probably consider other choices in the future). For now, we model the light source as an ellipse. The process is listed below.

- Determine a threshold for each image. For jpeg, the threshold is extreme value/3. The extreme is defined as the maximum pixel value after summing up the values along the channel axis. For .exr files, the threshold is the extreme value/100. Then, return the threshold image, whose values are only 0s and 1s. 1 denotes the existence of a light pixel.
- Taking the threshold image as the input, the algorithm in this step

would find connected components of light pixels. When the connected components are large enough, the algorithm would regard this area as a light source. For different components that are regarded as light sources, there would be a mask for each one. Each mask is used to denote one light source.

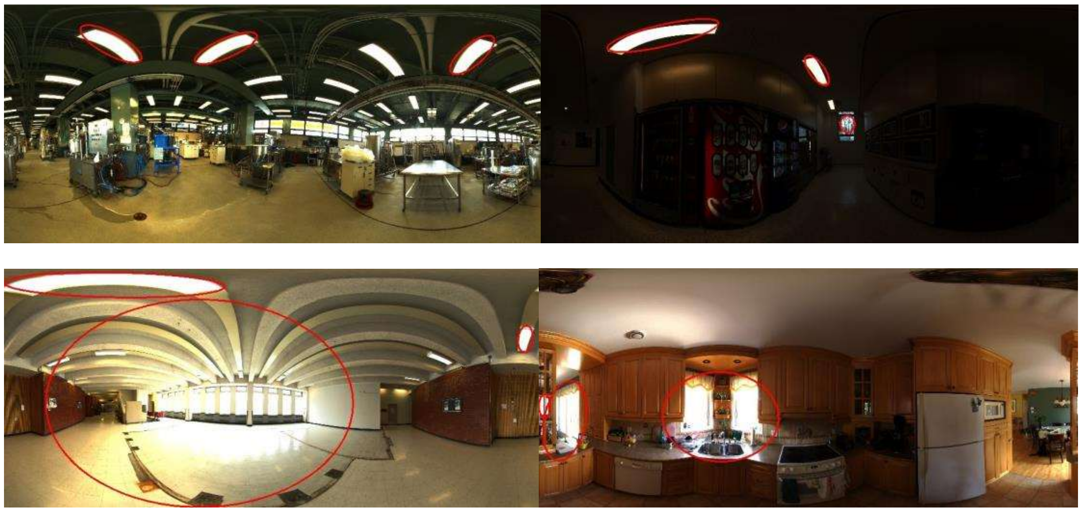
- Based on such masks, our algorithm would draw a minimum area rectangle that surrounds each light source. The parameters defining the minimum area rectangle would also define the parameters for an ellipse. We would use the parameters as the label for each light source. After finishing labeling for each light source, create a pair of the input image and its corresponding label.

- Repeat from 1 until all images are processed.

Therefore, this algorithm could not detect light source beyond images, which justifies our reason of using CNN.

Examples

Here are some annotated images generated by our Jpeg converter and labeler:



CNN network:

The CNN network will use DenseNet 121 as the feature extractor whose weights are fixed during training. The output layer is changed to a single linear layer to produce the labels we defined above. The loss function for now is the simply squared loss between three 8 by 1 vectors.

Moving Forward:

- Pre-processing:

  - In the preprocessing step, we found that extracting locations, color and brightness of the light source from HDR images is more reliable. We would combine the location information from HDR images with the information of the light property from LDR images.
  - Also, we would like to classify types of light sources during labeling. The area light representation in the paper is not suitable for real time render engine. We hope to identify them as primitive light types, such as directional lights, point lights. Our labeler needs to make the classification among these types are generate correct parameter associated with them.
  - **All location data is labelled via Cartesian coordinates, but a spherical coordinate is more reasonable for an environment map. For example, the lights at the ceiling are heavily distorted in Cartesian coordinate system. Therefore, we are looking into transferring our dataset by applying a transformation matrix.**
  - **We need to develop a way to crop the environment map into an unwrapped image that can be fed and trained using the network.**
- CNN network:
  - The size of input images can be modified to have better performance.
  - We are still using the pre-trained network (Densenet 121), but this decision is subject to change, as we hope our system could run on mobile devices. Some pre-trained networks we are considering are ResNet, MobileNet.
  - **The loss function can be improved. The Cosine Loss function would yield better results, since we are calculating the difference between two points on the surface of a sphere. Also, we would design an accuracy measure that would count how many light sources the network predicts successfully.**
- Post-processing:
  - We are exploring interpolating between results in different frames, after compensating the user’s movement.
  - We want the results to be useful for rendering engines, in our case, Blender. We plan to rebuild the labeled lights in Blender and compare the result against the environment map and parametric map lightings, then fine-tune the translations.

Note: Our focus points are in **BOLD TEXT**.