CSCI 5563: Assignment #3

Single View Depth Prediction

1 Submission

- Assignment due: Mar 12 (11:55pm)
- Skeletal code and data can be downloaded from: `https://www-users.cs.umn.edu/~hspark/csci5563_S2021/hw3.zip`.
- Individual assignment
- Up to 3 page summary write-up with resulting visualization (more than 3 page assignment will be automatically returned.).
- Submission through Canvas.
- Use Python 3
- You may use Google Colab for GPU usage (we provide a Pro account that allows extensive use.).
- You will complete the following functions and classes:
  - `TinyScanNetDataset` in `dataset.py`
  - `ExtendedDepthNet` in `network.py`
  - `ComputeDepthError`
  - `ComputeNormalError`
- You will submit the your trained model file `trained_model`
- DO NOT SUBMIT THE PROVIDED IMAGE DATA
- The function that does not comply with its specification will NOT BE GRADED.
- You are not allowed to use computer vision related package functions unless explicitly mentioned here. Please consult with TAs if you are not sure about the list of allowed functions.

CSCI 5563: Assignment #3

Single View Depth Prediction

2 Overview

In this assignment, you will implement a single view depth prediction by learning from ScanNet dataset. The pseudo code can be found in Algorithm 1.

You will train your model by running the following script:

```
!python hw3.py
    --batch_size 64
    --dataset_pickle_file path/to/tiny_scannet.pkl
    --learning_rate 1e-4
```

Algorithm 1 Training Single View Depth Prediction

```
Construct a training data loader                          ▷ TinyScanNetDataset
Construct a convolutional neural network                  ▷ ExtendedDepthNet
for Each epoch less than N do
    for Each batch in a epoch do
        Retrieve a batch of data from data loader
        Predict depth and measure prediction error, L_d   ▷ ComputeDepthError
        Measure consistency error, L_c                    ▷ ComputeNormalError
        Back-propagate error
        Update the network weights
    end for
    Save the trained model using torch.save
end for
```

CSCI 5563: Assignment #3

Single View Depth Prediction

3 Data and Setting Up

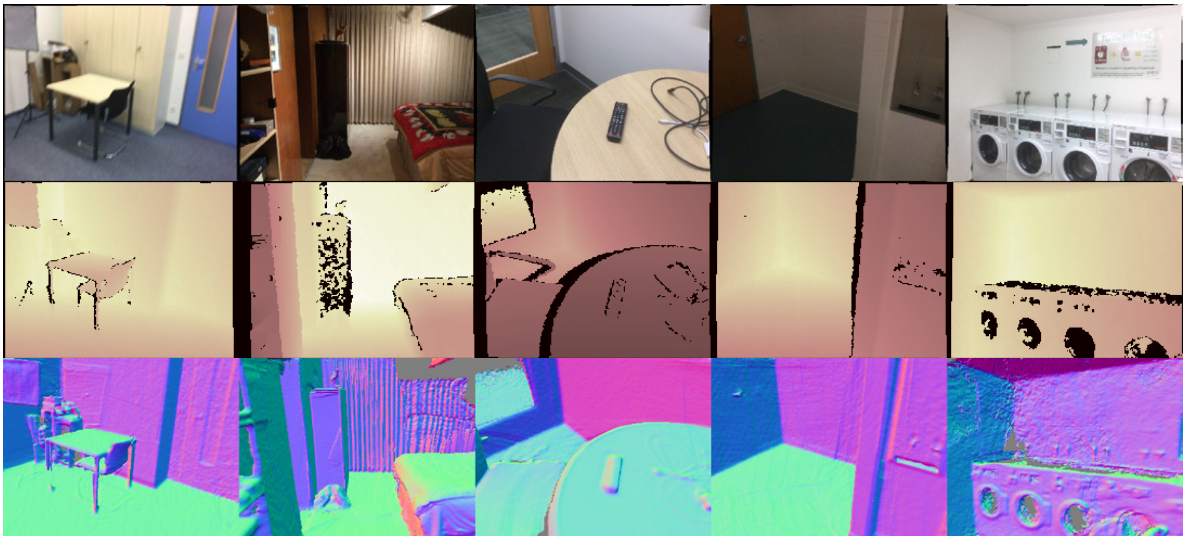

Figure 1: You will design a new convolutional neural network to predict depths from a single view image using both depth and surface normal supervision. Top: images; Middle: ground truth depth (the darker, the closer); Bottom: ground truth surface normal.

The Tiny ScanNet dataset The dataset can be accessed through the Shared drive in your Google Drive, under `Shareddrives/CSCI5563_data/tiny_scannet_data/` or through the following link:
`https://drive.google.com/drive/folders/1wWz74N7CKc6rNWJ6xZ6xrAlLBn00zfQB?usp=sharing`.

The dataset contains 3 folders: train, test, and val. Please consult with TAs if you do not have access to the dataset.

Google Colab To use Google Colab with PyTorch, follow the instructions:
`https://colab.research.google.com/drive/1gCgtlnMPVWYOl1ra8ssS0EzJiWaVR4Pk`

Access to data in Google Colab You can get an access to the data in the Google Colab environment by running the following script to obtain the pickle file `tiny_scannet.pkl` that contains all data paths for the RGB images, depths, and surface normals.

```
!python dataset_creator.py
     --data_path path/to/your/local/dataset
     --output_path path/to/output/folder/tiny_scannet.pkl
```

CSCI 5563: Assignment #3

Single View Depth Prediction

4 Loading Batch Data

You will define a data loader called `TinyScanNetDataset` class in `dataset.py` to load a batch of images, depths, and surface normals. It inherits `torch.utils.data.Dataset` class. The data loader will takes as input `*.pkl` file that lists training, validation, and testing data.

A loaded image is int8 format ranging $[0, 255]$ where you need to convert to float ranging $[0, 1]$. See `torchvision.transforms.to_tensor`. A depth image is int32 format of which values are in millimeter scale. You must rescale the depth to the meter scale for depth prediction by dividing by 1,000. A surface normal image, $\widehat{\mathbf{N}}$ is a regular RGB image ranging $[0, 255]^3$. You can convert it to 3D surface normal by

$$\mathbf{N} = \frac{\widehat{\mathbf{N}}}{127.5} - 1 \tag{1}$$

where $\mathbf{N}$ is the 3D unit surface normal vector.

`class TinyScanNetDataset(Dataset):`
Input: `usage` $= \{$`'train'`, `'val'`, `'test'`$\}$ is a string that specifies the training, validation, and testing data, respectively. `'*.pkl'` is a pickle file that lists the data. See `dataset_creator.py`.

Description: The dataloader will return tensors of image ($[0, 1]^{3\times H\times W}$), depth ($\mathbb{R}^{1\times H\times W}$), and normal ($[-1, 1]^{3\times H\times W}$) through its class method `__getitem__` where H and W are the height and width of an image. These tensors must be `torch.Tensor` format.

Note: We do not share `'test'` dataset that will be used for performance evaluation of your depth prediction.

CSCI 5563: Assignment #3

Single View Depth Prediction

5 Constructing CNN

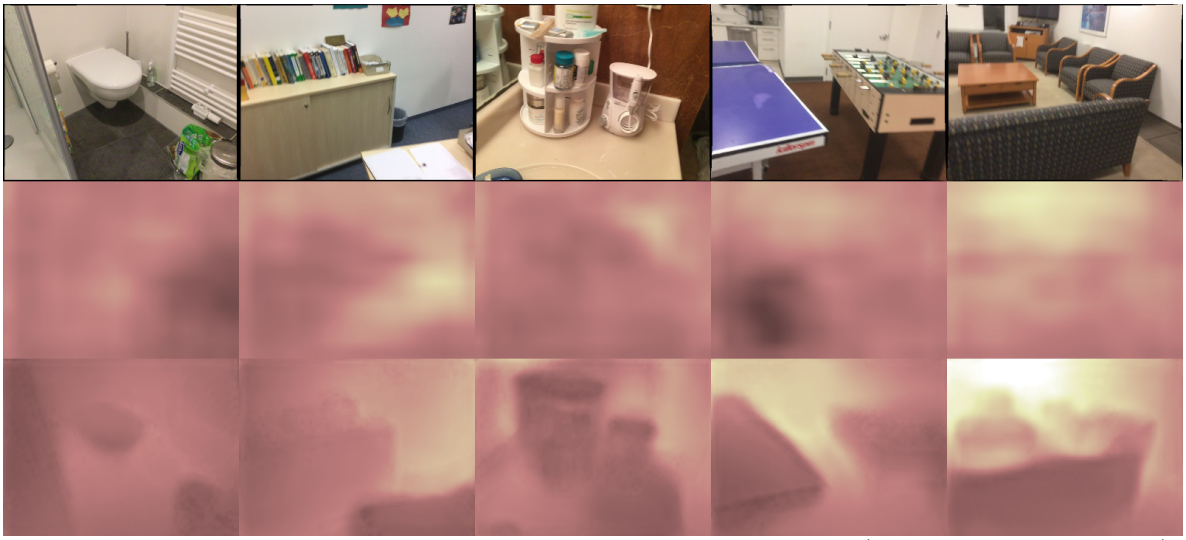

Figure 2: Qualitative results of the depth prediction networks (the darker, the closer). Top: images; Middle: predicted depth maps from the `SimpleDepthNet`; Bottom: predicted depth maps from an `ExtendedDepthNet`.

You will design a convolutional neural network (CNN) using `ExtendedDepthNet` class object in `network.py` to predict depths from an image, respectively. You can combine a series of convolution, batch normalization, activation, pooling layers to encode features and upsampling or transposed convolution with activation to decode features. The last layer operation must be `relu` activation to ensure the predicted depth to be positive.

We provide an example network called `SimpleDepthNet` that has three encoding layers and three decoding layers to predict the depth output. The class object needs `forward(x)` that takes as input a batch of images $[0, 1]^{B\times 3\times H\times W}$, and returns prediction $\mathbb{R}^{B\times 1\times H\times W}$ where B is the number of instances in a batch, i.e.,

$$f(\mathbf{I}) = g_n \circ \cdots \circ g_1(\mathbf{I}), \tag{2}$$

where f is a depth prediction function that takes as input a batch of images $\mathbf{I} \in [0, 1]^{B\times 3\times H\times W}$ and outputs the depth prediction. The prediction function is a composition of modular functions $\{g_i\}$ (e.g., convolution, batch normalization, activation, pooling, upsampling, and transposed convolution).

You are expected to design a new network that can significantly outperform `SimpleDepthNet`. You can explore many existing complex models that produce dense structured outputs such as Eigen [1], FCN [2], FPN [3], and DORN [4].

Note: You are not allowed to use any pretrained model, and any `'val'` data as a training data.

CSCI 5563: Assignment #3

Single View Depth Prediction

6 Measuring Depth Prediction Error

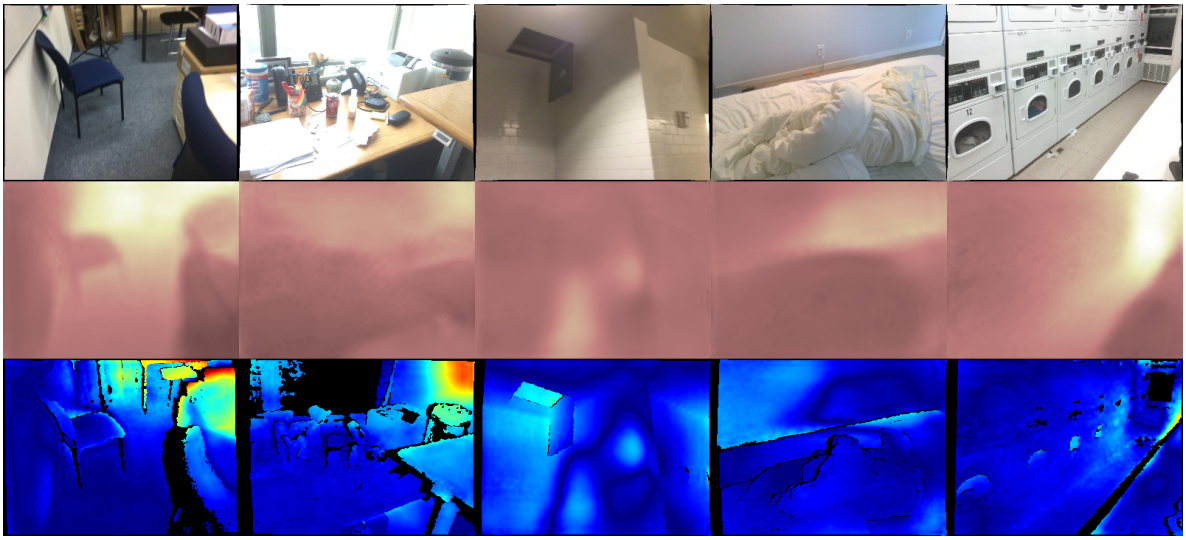

Figure 3: Depth maps' error visualization. Top: images; Middle: predicted depth maps from an `ExtendedDepthNet`; Bottom: predicted depth maps' error (the colder color indicates the smaller error).

Given the prediction, you can measure the depth and normal errors (or loss) for every pixel $\mathbf{u} = (u, v)$ in each batch:

$$L_{\mathrm{d}} = \frac{1}{|\mathbf{M}|} \sum_{i}^{B} \sum_{\mathbf{u} \in [0,W) \times [0,H)} \|\mathbf{M}_{\mathbf{u}}^{i} (\mathbf{D}_{\mathbf{u}}^{i} - f_{\mathbf{u}}(\mathbf{I}^{i}))\|_1, \tag{3}$$

where L_{d} is depth loss that measures difference between the ground truth depth $\mathbf{D} \in \mathbb{R}^{B \times 1 \times H \times W}$ and the prediction f. The subscript $\mathbf{u}$ indicates the pixel location and the superscript i represents the i^{th} instance in a batch. $\mathbf{M}$ is the binary mask that indicates the validity of the ground truth, i.e.,

$$\mathbf{M}_{\mathbf{u}}^{i} = \begin{cases} 1 & \mathbf{D}_{\mathbf{u}}^{i} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and $|\mathbf{M}|$ denotes its cardinality. Note that L_1 distance is used.

CSCI 5563: Assignment #3

Single View Depth Prediction

```
def ComputeDepthError(depth_pred, depth_gt):
  ...
  return L_d
```

Input: `depth_pred`, `depth_gt` $\in \mathbb{R}^{B\times 1\times H\times W}$ are the depth prediction and ground truth.

Output: $L_d \in \mathbb{R}$ is the depth loss measured by Equation (3).

Note: The L_d measures the mean absolute error (MAE) of the predicted depth maps. The expected value of L_d on the `'val'` data is less than 0.53m for the `SimpleDepthNet` and less than 0.45m for an `ExtendedDepthNet` as shown in Figure 3.

CSCI 5563: Assignment #3

Single View Depth Prediction

7 Enforcing Surface Normal Consistency

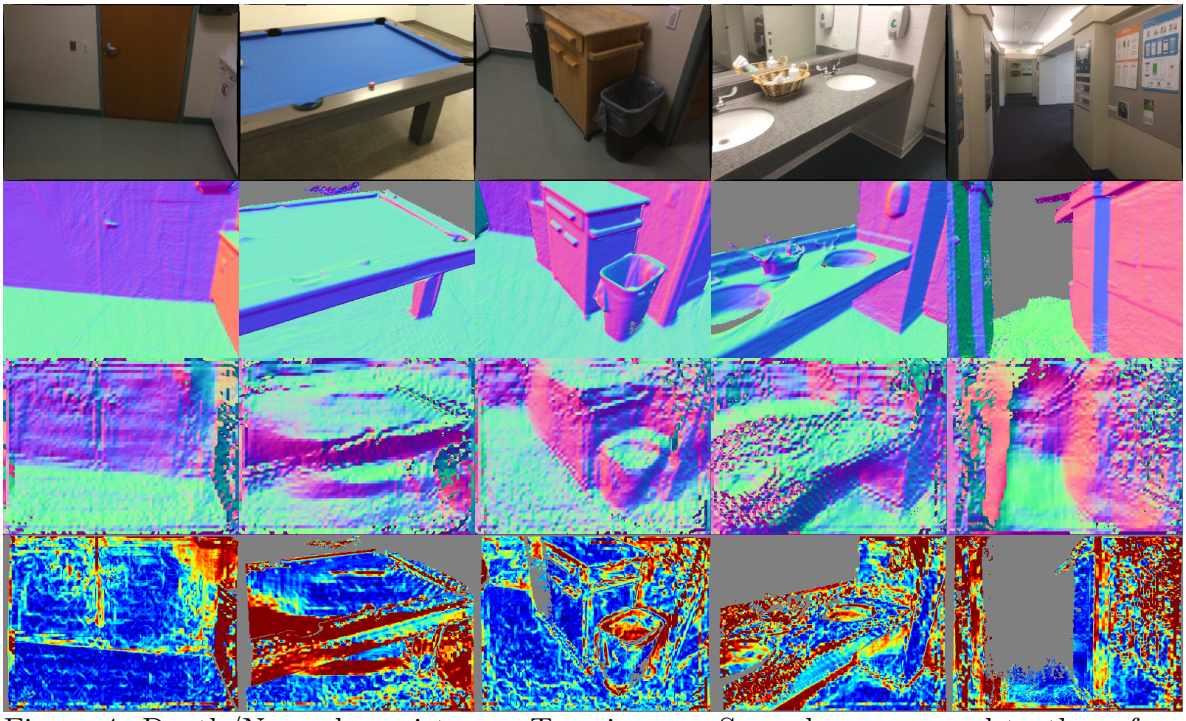

Figure 4: Depth/Normal consistency. Top: images; Second row: ground truth surface normals; Third row: surface normals computed from the predicted depths using an `ExtendedDepthNet`; Bottom: surface normal error map (the cooler color indicates the smaller error).

The prediction of depth must be geometrically consistent with the ground truth surface normal, i.e., the spatial derivative of the depth must align with the surface normal. Given the predicted depth $d = f_{\mathbf{u}}(\mathbf{I})$ at a pixel location $\mathbf{u} = (u, v)$, the 3D point $\mathbf{X}_{\mathbf{u}} \in \mathbb{R}^3$ can be computed using inverse projection:

$$\mathbf{X}_{\mathbf{u}} = d\mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{5}$$

where $\mathbf{K}$ is the intrinsic parameter.

The surface normal of $\mathbf{X}$ can be derived by taking spatial derivative of the 3D point:

$$\widehat{\mathbf{n}}_{\mathbf{u}} = \frac{(\mathbf{X}(u+1,v) - \mathbf{X}(u,v)) \times (\mathbf{X}(u,v+1) - \mathbf{X}(u,v))}{\|(\mathbf{X}(u+1,v) - \mathbf{X}(u,v)) \times (\mathbf{X}(u,v+1) - \mathbf{X}(u,v))\|}, \tag{6}$$

where $\widehat{\mathbf{n}}$ is the estimated surface normal from the predicted depth. Note that the surface normal is an unit vector where it needs to be normalized by its magnitude.

CSCI 5563: Assignment #3

Single View Depth Prediction

The estimate surface normal $\widehat{\mathbf{n}}$ must be consistent (aligned) with the ground truth surface normal $\mathbf{N}$:

$$L_{\mathrm{c}} = \frac{1}{|\mathbf{M}|} \sum_{i}^{B} \sum_{\mathbf{u} \in [0,W) \times [0,H)} \mathbf{M}_{\mathbf{u}}^{i} (1 - |\widehat{\mathbf{n}}_{\mathbf{u}}^{\mathrm{T}} \mathbf{N}_{\mathbf{u}}^{i}|). \tag{7}$$

```
def ComputeNormalError(depth_pred, K, normal_gt, depth_gt):
  ...
  return L_c
```

Input: `depth_pred`, `depth_gt` $\in \mathbb{R}^{B \times 1 \times H \times W}$ are the predicted and ground truth depths, respectively, `K` $\in \mathbb{R}^{3 \times 3}$ is the camera intrinsic parameter, and `normal_gt` $\in [-1, 1]^{B \times 3 \times H \times W}$ is the surface normal ground truth.

Output: `L_c` $\in \mathbb{R}$ is the error of depth and surface normal consistency measured by Equation (7).

CSCI 5563: Assignment #3

Single View Depth Prediction

8 Putting Things Together

The CNN will be trained by minimizing both the depth and consistency errors, i.e., the total error will be:

$$L = L_{\mathrm{d}} + \lambda L_{\mathrm{c}}, \tag{8}$$

where L is the total error that is minimized through error back-propagation using `L_c.backward()`. You can choose λ such that the surface normal supervision can provide a reasonable improvement. Figure 5 shows the training loss L_d, L_c, and L and Figure 6 illustrates the predicted depths, over training iterations.

With the normal consistency loss, the network can predict depth maps that can induce smooth surface normal (see Figure 7).

Note: The expected surface normal error is at most 50 degree, or $L_c < 0.36$.

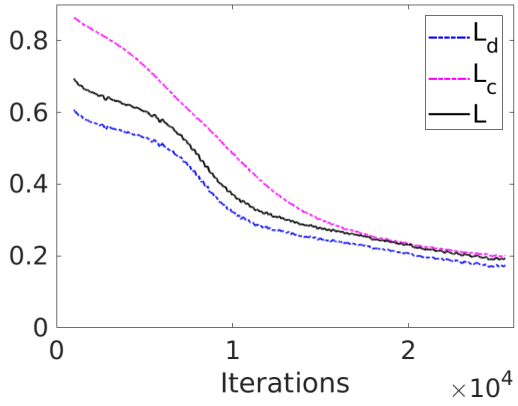

Figure 5: Training loss over iterations.

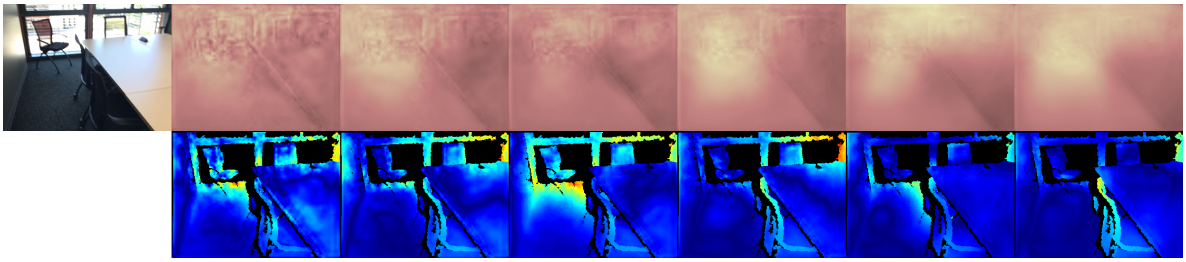

Figure 6: Depth prediction over training iterations. Top: image and its predicted depth over iterations. Bottom: the corresponding depth error.

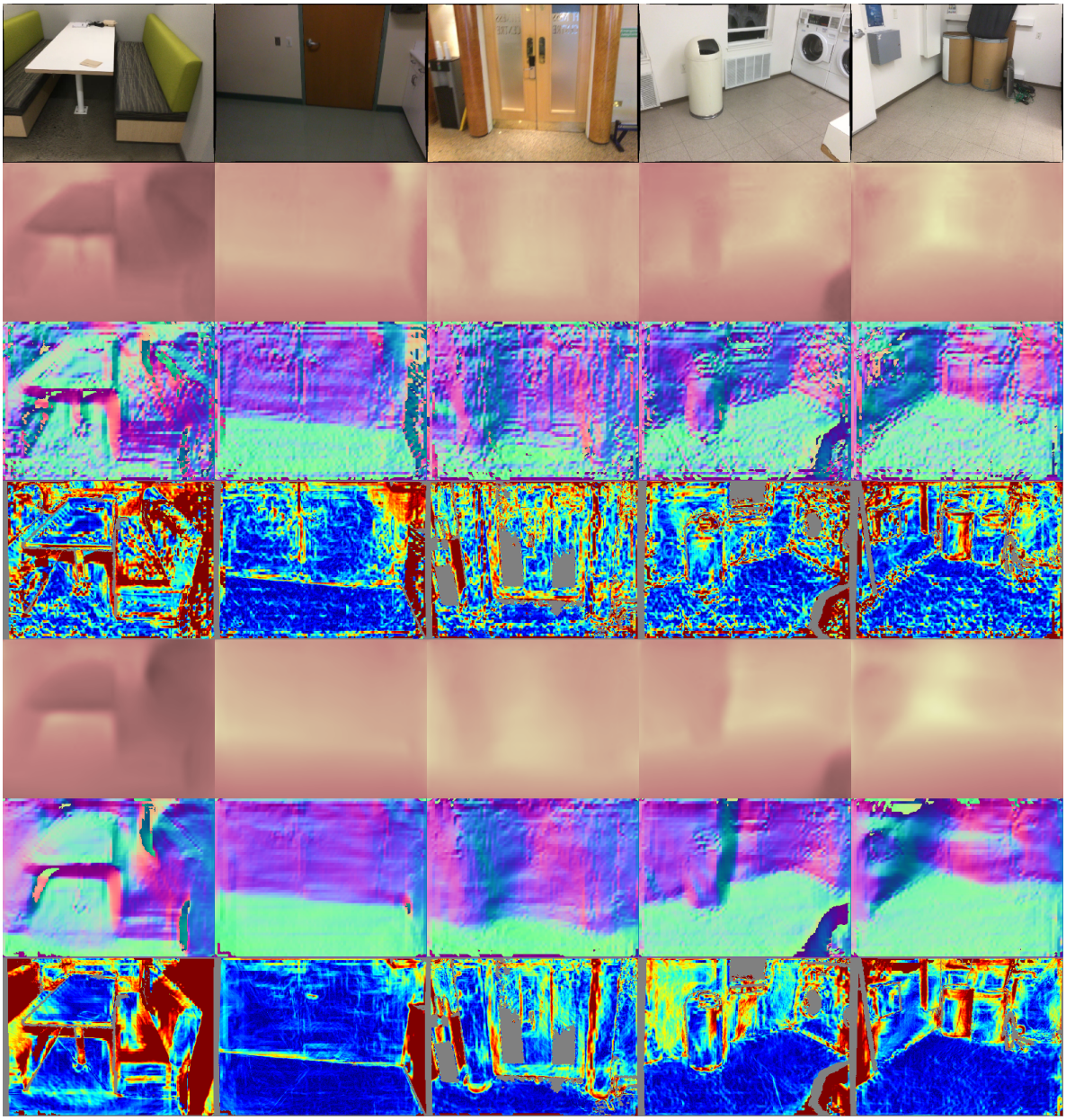

Figure 7: Final depths/normals prediction. Top: images; The next 3 rows and the last 3 rows illustrate: (i) predicted depths, (ii) surface normals induced from depths, and (iii) surface normals' error; w/o and w/ L_c, respectively.

References

[1] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," *NeurIPS*, 2014.

[2] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *TPAMI*, 2017.

[3] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," *CVPR*, 2017.

[4] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao, "Deep Ordinal Regression Network for Monocular Depth Estimation," in *CVPR*, 2018.