Lehrstuhl Computergrafik
Prof. Dr.-Ing. Hendrik P.A. Lensch
Prof. Dr.-Ing. Andreas Geiger

Eberhard Karls
Universität
Tübingen

Machine Learning in Graphics & Vision

Exercise 4

Release date: Wed, 10. June 2020 - **Deadline for Homework: Wed, 24. June 2020 - 21:00**

Exercises

You need to submit a **.zip** archive containing your report as a **.pdf** file and your code (namely the files `handcrafted_stereo.py` for Exercise 4.1 and `learned_stereo.py` for Exercise 4.2). Please do **not** include the dataset or the helper file `stereo_batch_provider.py` in your submission archive.
If you are interested in learning more about how we can use disparity maps to get 3D point clouds, you can additionally solve Exercise 4.3 and include you code in `visualize_pcl.py`. This Exercise 4.3 is **voluntary** and does not provide points.
Do **not** use jupyter-notebooks for these exercises.

4.1 Disparity Estimation with Block Matching (4 + 2 + 2 + 2 Points)

Stereo matching algorithms try to find corresponding pixels in two stereo images which are projections of the same 3D point. In this first task, you estimate the disparity for the 5 stereo pairs in the provided test set by using the **block matching algorithm**. The key idea of block matching is to compare a window $\mathbf{w_L}$ around each pixel (x, y) in the left image against windows $\mathbf{w_R}$ in the right image that are horizontally shifted to the left by values within a fixed disparity range $[0, d_{max}]$. For comparing two windows, we choose the *Sum of Absolute Differences (SAD)* metric defined as

$$\mathrm{SAD}(x, y, d) = \|\mathbf{w_L}(x, y) - \mathbf{w_R}(x - d, y)\|_1$$

where $d \in [0, d_{max}]$ is the candidate disparity value. For the final disparity for each pixel, we choose the value d which minimizes the SAD. This approach is called *Winner-Takes-All (WTA)*.

a) In file `handcrafted_stereo.py` fill the missing body of the function

```
def sad(image_left, image_right, window_size=3, max_disparity=50)
```

(**Hint:** In the code you can see that we apply zero padding to the images so that you can use two nested for loops ranging from $[0, \text{height} - \text{window size} + 1]$ and $[0, \text{width} - \text{window size} + 1]$ to calculate the disparity for each pixel. Here height and width are the image's height and width.)

Bonus: Instead, you can also try to implement a more elegant solution where for each candidate disparity value you first compute the absolute difference image and next convolve this image with a mean filter of size `window_size`.

b) Using your implementation from a, compute the corresponding disparities from the stereo pairs in the test set and visualize them using a nice colormap. Please fill the missing body of the function

```
def visualize_disparity(disparity, title = "Disparity Map",
                        out_file = "disparity.png", max_disparity = 50)
```

In case you end up using matplotlib for your visualizations, keep in mind that you might have to properly adjust the figure size and the `v_min` and `v_max` arguments of the `imshow()` function in order to get nice results.

c) Experiment with the different window sizes $\{3, 7, 15\}$ and report which one leads to better visual results and why? (In case you were not able to solve the previous exercises, you can use the provided disparity maps in the `examples/` folder.)

d) Why do you think the block matching approach fails to lead to good estimations around homogeneous regions such as the road?

4.2 Stereo Matching with Siamese Neural Networks (3 + 2 + 3 + 2 Points)

For this task you have to estimate again the disparity for all the stereo pairs in the test set but this time using a Siamese Neural Network which will learn the task from data. The key idea is to train the network to estimate the *similarity* between two patches. We do this by first extracting features in a convolutional manner and then normalizing the per-pixel feature vectors and calculating the dot product.

a) In file `learned_stereo.py` please use Pytorch's `nn` library to fill the missing bodies of the methods

```
def __init__(self)
def forward(self, X)
```

for the Pytorch module

```
class StereoMatchingNetwork(torch.nn.Module).
```

X is the input tensor of size `batch_size`×`height`×`width`×`channels` holding the image or the patch. You should implement the architecture described below as instance attribute(s) in `__init__(self)`.

```
Conv2d(..., out_channels=64, kernel_size=3)
ReLU()
Conv2d(..., out_channels=64, kernel_size=3)
ReLU()
Conv2d(..., out_channels=64, kernel_size=3)
ReLU()
Conv2d(..., out_channels=64, kernel_size=3)
functional.normalize(..., dim=1, p=2)
```

`forward(self, X)` should then perform one forward pass through those layers. Be aware that the layers form only one of the two identical branches of the siamese network. The forward pass will be called twice, that ensures weight sharing between both branches of the siamese network.

(**Hint**: Note that the convolutional layers expect the data to have shape `batch_size` × `channels` × `height` × `width`. Permute the input dimensions accordingly for the convolutions and remember to revert it before returning the features.)

b) In file `learned_stereo.py` fill the missing body of the function

```
calculate_similarity_score(infer_similarity_metric, Xl, Xr)
```

which estimates the similarity between the patches X_l and X_r. First, run the forward pass on each of the two tensors to estimated the feature tensors F_l and F_r of each branch of the Siamese network. Then calculate the similarity between the features F_l and F_r.

(**Hint:** As F_l and F_r are already normalized, you obtain the similarity by calculating the dot product.)

c) Start training the network by running `python learned_stereo.py train` and report the best training loss and accuracy you get after 1000 training iterations. Create a plot by filling the missing body of the function

```
def save_loss_plot(loss, out_file='./output/learned_stereo/loss.png')
```

and add it to your report. Try to improve the network in terms of training loss and accuracy by training it longer. What happens if you increase or decrease the filter numbers of the convolutional layers? Explain your findings.

d) Run `python learned_stereo.py evaluate` and compare the visualization of the disparity maps from the Siamese Neural Network to the ones obtained by the block matching algorithm. Which predictions are better and why? Can you find regions in the scenes where the differences in predictions are most dominant? (If you were not able to solve the previous exercises, you can use the provided disparity maps in the `examples/` folder.)

4.3 Point Clouds from Disparity Maps (0 Points)

Reminder: This exercise is **voluntary** and does not provide points.
In this task we use the estimated disparity for the first test stereo pair to convert it to a depth map and then to a point cloud. (In case you were not able to solve the previous exercises, you can use the provided disparity file `disparity.npy` in the `examples/` folder.)

a) In file `visualize_pcl.py` fill the missing body of the function

```
def disparity_to_depth(disparity, baseline, focal_length)
```

that transforms the disparity to depth by using the relevant formula discussed in the lecture.

b) As we have the depth now, we can calculate the 3D coordinates for the pixels in the image plane. In file `visualize_pcl.py` fill the missing body of the function

```
def img_to_3D_coordinates(img_x, img_y, focal_length, Z, cx, cy)
```

that transforms the x and y coordinates from the image coordinate system to the X and Y coordinates in 3D space. **Hint:** You can rearrange the pinhole camera projection formula

$$\pi : \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} fX/Z + c_x \\ fY/Z + c_y \end{pmatrix}.$$

c) Run the script `visualize_pcl.py` and analyze the point cloud. You can either use the `--interactive` argument for an interactive plot or adjust the initial view to save images from different viewing angles and add them in your report. Can you find areas in the scene where the point cloud is particularly good / bad? Can you find a correlation to the visualization of of the disparity maps? Explain your findings.