Exercise 1

1 Cost Volume Filtering for Stereo

In this exercise, we will follow some of the ideas in A. Hosni et al.: Fast Cost-Volume Filtering for Visual Correspondence and Beyond, PAMI 2013.

A stereo camera setup takes two images of the same scene – one from the viewpoint of a human’s left eye, the other from the right eye. From these two images, the aim is to reconstruct a depth map, where every pixel is assigned its corresponding distance from the camera setup. You can do the following experiment: hold a finger in front of your eyes, and switch back and forth between your left and right eye. You see the finger “jumping” left and right. The farther from your face the finger is, the smaller the “jump”. Consequently, we need to find – for every pixel in the left image – the corresponding pixel in the right image. From their distance (the disparity), the depth map can be reconstructed.

You can find corresponding PNG images on the exercise’s page.

Let \([L]_{r,c}\) be the pixel in row \(r\) and column \(c\) in the left image. Then a corresponding pixel in the right image can be found in \([R]_{r,d}\) with \(d \in [c + s - l \ldots c + s + l]\), that is in the same row. Here, \(s\) is the average shift we expect in the image, and \(l\) gives a search range around that average value. From all candidate pixels \([R]_{r,d}\), we need to select the best matching one. We will do this here using the sum of absolute differences between the pixels in the two patches.

Implement *sum of absolute differences* as the following function of two image patches \(p_1\) and \(p_2\):

```python
def sad(p1, p2):
    assert p1.shape == p2.shape
    return None
```

Next, implement the nested loop that, for every pixel in the left image:

- constructs a patch \(P\) of size \(W \times H\) around \([L]_{r,c}\).
- constructs \(2 \cdot l + 1\) patches \(Q_i\) around \([L]_{r,c-s}\).
- computes the sum of absolute differences between \(P\) and each \(Q_i\).
- saves the resulting correlations in a cost volume image at an index specifying the \(r\), \(c\) and the considered disparity.

Choose small patch sizes such as \(7 \times 7\), but large values for \(l\) (around 20) and a reasonable shift \(s\) (around \(-15\)). To avoid handling boundary problems, discard patches that would intersect the image boundary.

*Hint:* This procedure is really slow (runtime are minutes). It’s a good idea to save your resulting cost volume image like this (and, for later parts, just load the precomputed data as above):

```python
f = h5py.File("costvol.h5", 'w')
f.create_dataset("cost_volume", data=vol)
f.close()
```

Furthermore, develop your method on a small subset of the image only, or downsize the images.

You can obtain a disparity map by finding – for each pixel – the disparity which exhibited the best sum of absolute deviations. Plot the result. How does it look?

To improve the disparity map, the *Cost volume filtering* paper proposes to apply a smoothing filter to each disparity channel separately, and only afterwards produces the disparity map from the cost volume image.
Use Gaussian smoothing (Hint: vigra.filters.gaussianSmoothing, don’t forget to cast to numpy.float32) and implement this. How does the result look?