Exercise 3

1 Dynamic programming: viterbi path

We will use the cost volume from exercise 1 to solve the following optimization problem:

\[ E(D) = \sum_p C(D_p) + \sum_{q \in N_p} P_1 T[|D_p - D_q| == 1] + \sum_{q \in N_p} P_2 T[|D_p - D_q| > 1] \]

In this optimization problem \( C(D_p) \) is the cost of disparity \( D_p \) at pixel \( p \), i.e. the entry from the cost volume. \( N_p \) is the neighborhood of pixel \( p \). \( P_1 \) is a cost for a disparity jump of 1 between neighboring pixels \( T[|D_p - D_q| == 1] \). \( P_2 \) is a cost for a disparity jump larger than 1 between neighboring pixels.

Implement the dynamic programming algorithm described in the lecture that solves this optimization problem along the rows of the image. Plot the viterbi disparity path image.

2 Dynamic Programming: segmented least squares (optional)

In this exercise, we will implement the segmented least squares regularization for our stereo image pair cost volume from exercise 1.

Information on the segmented least squares method can be found in the slides on Dynamic programming I available from http://www.cs.princeton.edu/wayne/kleinberg-tardos/.

First, read these slides and make sure you understand how the method works.

Then implement a function that computes the least squares fit for a subset of points from index \( i \) to index \( j \) of an array. The function should return the slope and intercept as well as the sum of squares error of the fitted line.

```python
def lsf(line, i, j):
    # do a least squares fit and determine a*x+b and the error
    return a, b, error
```

In the next step implement the actual method that fits a set of lines to a set of points. The method should return the predictions for each point, i.e. the the value of the fitted line at each position.

```python
def segls(line):
    # implement the dynamic programming algorithm and return the least squares prediction of all line segments
    return prediction
```

Choose a sensible regularization parameter such as \( C = 5 \) and make sure you only work on a subset of the volume as the runtime of this algorithm is very high.

```python
import vigra
import numpy
import h5py
import math

def lsf(line, i, j):
    # do a least squares fit and determine a*x+b and the error
    return a, b, error

def segls(line):
    # implement the dynamic programming algorithm and return the least squares prediction of all line segments
```
return prediction

if __name__ == "__main__":
    f = h5py.File("costvol.h5", "r")
    vol = f["costvol"].value[:, :, :]

    im = numpy.argmin(vol, axis = 2)
    vigra.impex.writeImage(im, "disparity_raw.png")

    impred = im.copy()
    for y in range(vol.shape[1]):
        line = im[:, y]
        pred = segls(line)
        impred[:, y] = pred

    vigra.impex.writeImage(impred, "disparity_prediction.png")