Exercise 2

1 OpenGM

In this exercise, we will implement disparity smoothing using a graphical model.

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

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

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.

Please adapt the following code snippet that smooths an image using a potts model to our use case.

```python
import numpy
import opengm

img = numpy.random.rand(4, 4)
dimx = img.shape[0]
dimy = img.shape[1]
numVar = dimx * dimy
numLabels = 2
beta = 0.3

numOfStates = numpy.ones(numVar, dtype=opengm.index_type) * numLabels
gm = opengm.graphicalModel(numOfStates, operator='adder')

# Adding unary function and factors
for y in range(dimy):
    for x in range(dimx):
        f = numpy.ones(2, dtype=numpy.float32)
        f[0] = img[x, y]
        f[1] = 1.0 - img[x, y]
        fid = gm.addFunction(f)
        gm.addFactor(fid, (x * dimy + y,))

# Adding binary function and factors
vis = numpy.ones(5, dtype=opengm.index_type)

f = numpy.ones(pow(numLabels, 2), dtype=numpy.float32).reshape(numLabels, numLabels) * beta
for l in range(numLabels):
    f[1, l] = 0
    fid = gm.addFunction(f)

# Add binary factors
for y in range(dimy):
    for x in range(dimx):
        if (x+1 < dimx):
            # as tuple (list and numpy array can also be used as vi’s)
            gm.addFactor(fid, numpy.array([(x+1) * dimy + y, (x+1) * dimy + y], dtype=opengm.index_type))
        if (y+1 < dimy):
            # as tuple (list and numpy array can also be used as vi’s)
            gm.addFactor(fid, numpy.array([(x+1) * dimy + y, (x+1) * dimy + y], dtype=opengm.index_type))
```
### Exercise 2

```python
#vi as list (tuple and numpy array can also be used as vi's)
gm.addFactor((fid, [x*dimy+y, x*dimy+(y+1)]))

icm = spengm.inference.Icm(gm)
icm.infer()
argmin = icm.arg()

res = argmin.reshape(dimx, dimy)
```

You have to:

- load our cost volume using h5py
- adapt the dimension of the image
- adapt the number of states
- use the unary factors from our cost volume
- adapt the binary factors
- write out the disparity prediction