Image Analysis Excercise Sheet 7

Markus Doering, 3153320

June 12, 2013

1 Natural Image Statistics

All python code for this excercise is found in file `ia_07_01.py` and in the appendix.

In figure 1 we can see the histograms for the gradients of 4 natural images. The histograms arer more heavy-tailed compared to a Gaussian because of the areas of no intensity change and the sharp edges that are present in natural images. There is no significant difference of $\delta/\delta x$ and $\delta/\delta y$ visible. This could be due to the fact that no edge orientation is really dominant, like in the first image, or that most edges are diagonal and counted in both histograms, like in the last image.

Suppose we have an image where each pixel x_i is drawn independently from a normal distribution of mean μ and variance σ^2 ($x_i \sim \mathcal{N}(\mu, \sigma^2),\ \forall 1 \leq i \leq n$). We assume that the image has only one row, but the result clearly extends to normal images as well. The horizontal gradient image has entries $g_i = x_{i+1} - x_i$ (with $x_{n+1} = 0$), and since the x_i are independent we can express the distribution of the gradient pixels as $g_i \sim \mathcal{N}(0, 2\sigma^2)$. Therefore, the gradient of the random image is, again, a random image with zero mean and twice the variance.

3 Integer Linear Programming

All python code for this excercise is found in file `ia_07_03.py` and in the appendix.

In figure 3 we can see the solution to this excercise. The optimum of the ILP is the point $v_1 = (3, 2)$ with gain $g_1 = 18$, whereas the optimum of the LP is at the intersection of the red and the magenta line, $v_2 \approx (2.4, 3.4)$ with gain $g_2 \approx 19.8$. Rounding the LP solution to the closest integer point yields $v_3 = (2, 3)$ with gain $g_3 = 17$, which is so far the worst solution.

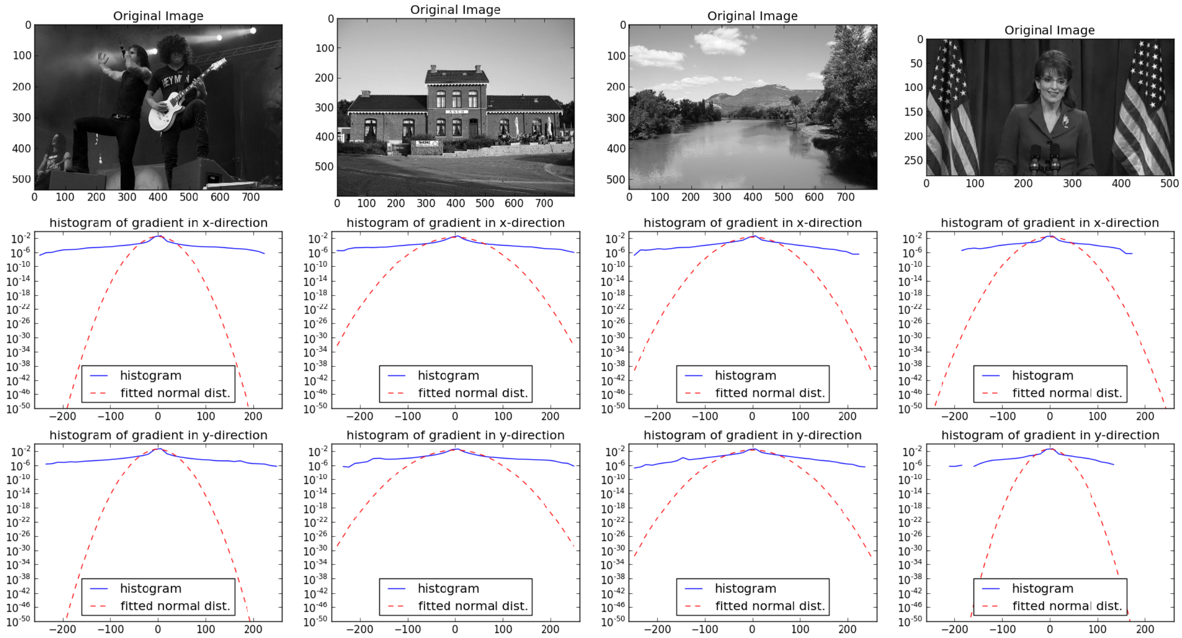

Figure 1: Natural images and the histograms of their gradients in x and y direction. The histograms are heavy-tailed compared to a Gaussian with equal mean and variance.

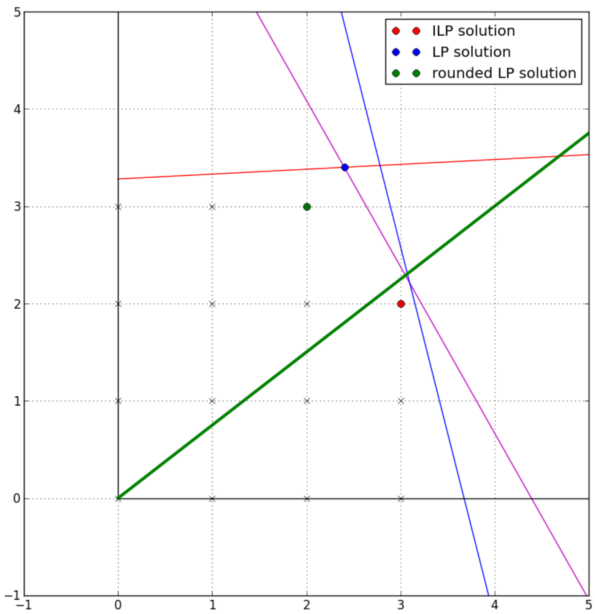

Figure 2: ILP diagram. The thin lines represent the constraints, the thick green line is the cost vector.

```python
#!/usr/bin/python2
# coding: utf-8
#
# Author: Markus Doering
# File: ia_07_01.py
#

import vigra
import numpy as np
from scipy.signal import convolve2d
from scipy.stats import norm

import matplotlib
matplotlib.use('Qt4Agg')
from matplotlib import pyplot as plot
from matplotlib.image import imread

nImg = 4

def rgb2gray(rgb):
    '''
    convert from RGB to grayscale
    http://en.wikipedia.org/wiki/Grayscale#Converting_color_to_grayscale
    '''
    return .299*rgb[:,:,0] + .587*rgb[:,:,1] + .114*rgb[:,:,2]

def getRealWorldImages():
    '''
    read real world images
    '''
    return [rgb2gray(imread("real%d.jpg" % (i,))) for i in range(1,nImg+1)]

def gradient(im, direction='x'):
    '''
    compute the image gradient with filter [-1,1] in the specified direction
    '''

    filt = np.ones((2,1))
    filt[0,0] = -1

    if direction == 'y':
        # first axis is vertical, i.e. y, so the filter is fine
        pass
    elif direction == 'x':
        # transpose the filter to horizontal direction
        filt = filt.transpose()
    else:
        raise ValueError("unknown axis {}".format(direction))

    return convolve2d(im, filt, mode='same')


def myHist(im):
    '''
    compute histogram with bin centers rather than bin edges
    and fit a gaussian to the data
    '''
    bins, bounds = np.histogram(im, bins=40, range=(-255,255), density=True)
```

```
    bincenters = [(bounds[i]+bounds[i+1])/2.0 for i in range(len(bounds)-1)]

    mu = im.mean()
    s = im.var()

    gaussianfit = norm.pdf(bincenters, loc=mu, scale=np.sqrt(s))

    return (bincenters, bins, gaussianfit)


def ex1():
    '''
    solve excercise 1
    '''

    plot.hold(True)

    imgs = getRealWorldImages()

    # compute the gradients in x and y direction separately
    xgrads = [(gradient(img, direction='x')) for img in imgs]
    ygrads = [(gradient(img, direction='y')) for img in imgs]


    for img, xgrad, ygrad, k in zip(imgs, xgrads, ygrads, range(len(imgs))):
        # show image
        plot.subplot(3, nImg, k+1)
        plot.imshow(img)
        plot.gray()
        plot.title('Original Image')

        # show histogram for x gradient
        plot.subplot(3, nImg, k+nImg+1)
        xbincenters, xbins, xgauss = myHist(xgrad)
        plot.semilogy(xbincenters,xbins, 'b')
        plot.semilogy(xbincenters,xgauss, 'r--')
        plot.legend(['histogram', 'fitted normal dist.'], loc='lower center')
        plot.title('histogram of gradient in x-direction')
        plot.axis([-260,260,1e-50,1])

        # show histogram for y gradient
        plot.subplot(3, nImg, k+2*nImg+1)
        ybincenters, ybins, ygauss = myHist(ygrad)
        plot.semilogy(ybincenters,ybins, 'b')
        plot.semilogy(ybincenters,ygauss, 'r--')
        plot.legend(['histogram', 'fitted normal dist.'], loc='lower center')
        plot.title('histogram of gradient in y-direction')
        plot.axis([-260,260,1e-50,1])

    plot.show()

if __name__ == "__main__":
    ex1()
```

```python
#!/usr/bin/python2
# coding: utf-8
#
# Author: Markus Doering
# File: ia_07_03.py
#

import vigra
import numpy as np

import matplotlib
matplotlib.use('Qt4Agg')
from matplotlib import pyplot as plot


def ex3():
    '''
    solve excercise 3
    '''

    xs = np.linspace(0,5)

    fig = plot.figure(1, figsize=(10,10))


    plot.hold(True)
    ys1 = .05*xs+3.28
    ys2 = -1.71*xs+7.51
    ys3 = -3.83*xs+14.08
    cost = .75*xs

    #plot line constraints
    plot.plot(xs,ys1,'r')
    plot.plot(xs,ys2,'m')
    plot.plot(xs,ys3,'b')

    # plot integer constraints
    plot.plot(xs,0*xs,'k')
    plot.plot(0*xs,np.linspace(0,5),'k')

    valid_x = [0,0,0,0,1,1,1,1,2,2,2,2,3,3,3]
    valid_y = [0,1,2,3,0,1,2,3,0,1,2,3,0,1,2]
    plot.plot(valid_x,valid_y,'kx')

    # plot target vector
    plot.plot(xs,cost, 'g', linewidth=3)

    # mark solutions
    ilp_sol = (3,2)
    ilp, = plot.plot(ilp_sol[0], ilp_sol[1],'ro', markersize=7)
    print("ILP: solution={}, gain = {}".format(ilp_sol, 4*ilp_sol[0] + 3*
        ilp_sol[1]))

    xlp = (7.51-3.28)/(0.05+1.71)
    lp_sol = (xlp, -1.71*xlp+7.51)
    lp, = plot.plot(lp_sol[0],lp_sol[1], 'bo', markersize=7)
    print("LP: solution={}, gain = {}".format(lp_sol, 4*lp_sol[0] + 3*lp_sol
        [1]))
```

```python
    rlp_sol = (2,3)
    rlp, = plot.plot(rlp_sol[0], rlp_sol[1],'go', markersize=7)
    print("rounded LP: solution={} gain = {}".format(rlp_sol, 4*rlp_sol[0] +
        3*rlp_sol[1]))

    # adjust plot
    plot.axis([-1, 5, -1, 5])
    plot.legend([ilp,lp,rlp],['ILP solution', 'LP solution', 'rounded LP
        solution'])
    plot.grid()
    plot.show()

if __name__ == "__main__":
    ex3()
```