# 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.

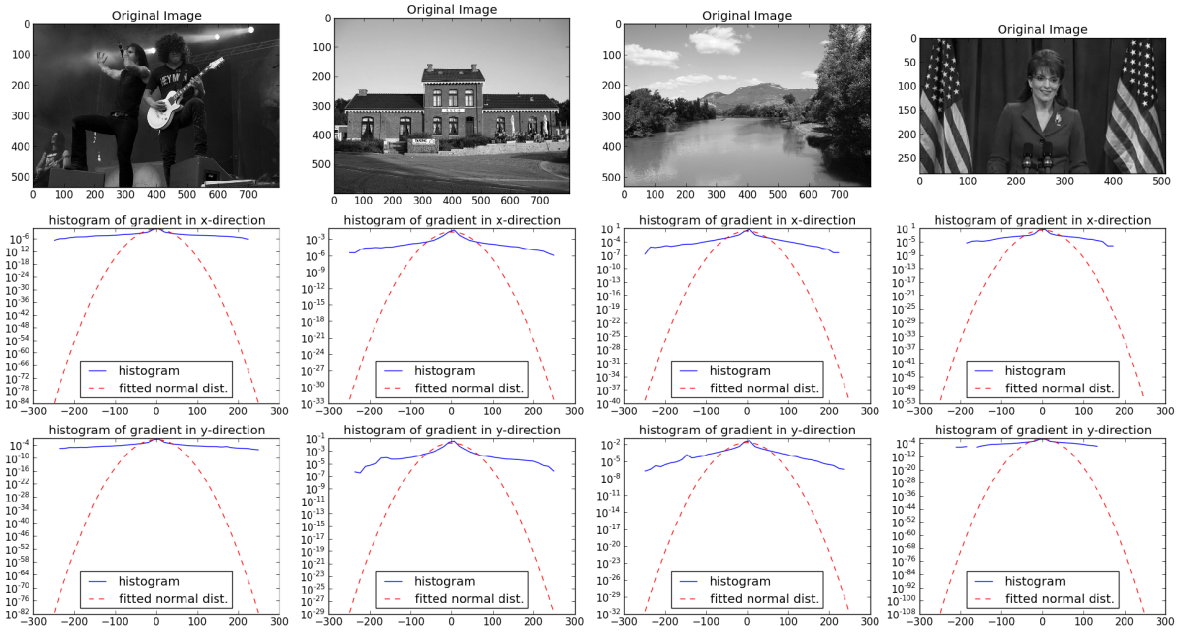


Figure 1: Natural images and the histograms of their gradients in x and y direction. The histograms are heavy-tailed compared to a Gaussian with equal mean and variance.

## 3 Integer Linear Programming

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

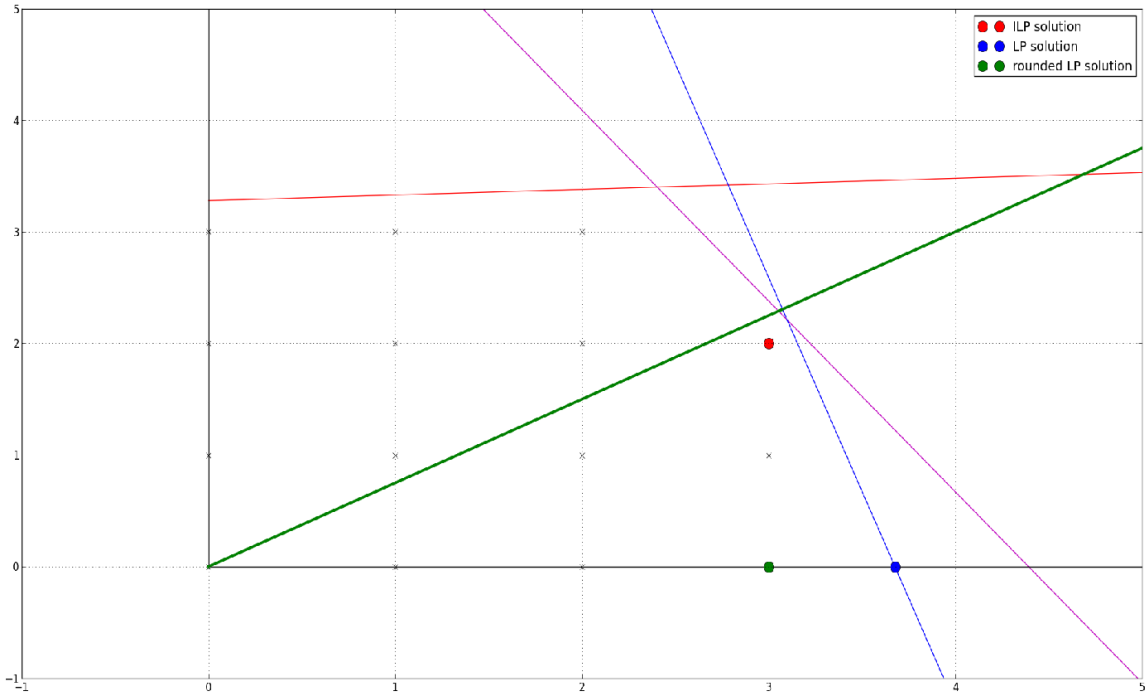


Figure 2: BLALABERBLUB

```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')

        # 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.show()

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

```
#!/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)


    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 solution
    ilp, = plot.plot([3,],[2,],'ro', markersize=12)
    lp, = plot.plot([14.08/3.83,], [0,], 'bo', markersize=12)
    rlp, = plot.plot([3,],[0,],'go', markersize=12)



    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()
```