Problem_3

April 15, 2018

In [1]:
```
import cv2 # tested with 3.1.0 version
import numpy as np
import matplotlib.pyplot as plt
import os
%matplotlib inline
```

1 3. Edge Detection

In [4]:
```
from skimage.measure import compare_ssim
from evaluate import evaluate
# The code returns the accuracy of the edge detector when compared against the ground
# OUTPUT_FILE_PATH: Path of the image containing the edges obtained using edge detecto
# GROUND_TRUTH_PATH: Path of the corresponding ground truth image (present in the fold
# An example of how to use the evaluate function is shown as follows:
OUTPUT_FILE_PATH = './data/test.png'
GROUND_TRUTH_PATH = './data/ground_truth/3096.bmp'
print('Accuracy: ' + str(evaluate(OUTPUT_FILE_PATH, GROUND_TRUTH_PATH)))
```

```
Accuracy: 0.621913070511
```

In [5]:
```
def show_histRGB(img, imagepath):
    color = ('b', 'g', 'r')
    plt.figure(figsize=(20, 6)) # Figure size can be adjusted.
    plt.subplot(121), plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)) # show the corr
    plt.xticks([])
    plt.yticks([])
    plt.title(imagepath)
    for i, col in enumerate(color):
        # a) Calculate the histograms for this image.
        histr = cv2.calcHist([img],[i],None,[256],[0,256])
        plt.subplot(122), plt.plot(histr, color = col) # plot histogram with the b g r
        plt.xlim([0, 256])
        plt.title("histRGB")
    plt.show()
```

1.1 3.1 Edge Detector 1

Canny Edge Detector: The MinVal and MaxVal parameters have been selected for better performance.

```
In [8]: ### Fill your code here
        ### Report the accuracy obtained
        ### Report any improvements you have tried

        # Canny Edge Detector
        IMG_NAMES = ['3096.jpg', '12084.jpg', '33039.jpg', '37073.jpg']
        Val = [(200,300),(700,1100),(600,1100),(200,300)]

        for imagename, Val in zip(IMG_NAMES,Val):
            imagepath = os.path.join('data/img/', imagename)
            groundtruthpath = os.path.join('data/ground_truth/',imagename.split('.')[0]+'.bmp'
            img = cv2.imread(imagepath,1) # read the image from image path using opencv.
            edge_dectected = cv2.Canny(img,Val[0],Val[1],True)
            cv2.imwrite(imagename.split('.')[0]+'_Canny.bmp',edge_dectected)
            print('Accuracy: ' + str(evaluate(imagename.split('.')[0]+'_Canny.bmp', groundtrut
```

```
Accuracy: 0.851458215944
Accuracy: 0.678700267485
Accuracy: 0.769930246566
Accuracy: 0.73535793162
```

1.2 3.2 Edge Detector 2

Sobel Edge Detector: K_size is selected for better performance

```
In [7]: ### Fill your code here
        ### Report the accuracy obtained
        ### Report any improvements you have tried

        # Sobel Operator
        IMG_NAMES = ['3096.jpg', '8023.jpg', '33039.jpg', '37073.jpg']
        K_size = [3,1,1,3]

        for imagename, k in zip(IMG_NAMES,K_size):
            imagepath = os.path.join('data/img/', imagename)
            groundtruthpath = os.path.join('data/ground_truth/',imagename.split('.')[0]+'.bmp'
            img = cv2.imread(imagepath,1) # read the image from image path using opencv.
            sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=k)
            sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=k)
            result = np.uint8(np.sqrt(np.square(sobelx.astype(int))+np.square(sobely.astype(in
            cv2.imwrite(imagename.split('.')[0]+'_Sobel.bmp',result)
            print('Accuracy: ' + str(evaluate(imagename.split('.')[0]+'_Sobel.bmp', groundtrut
```

```
Accuracy: 0.849696569323
Accuracy: 0.82824593105
Accuracy: 0.203288838803
Accuracy: 0.743881192479
```

1.3 3.3 Edge Detector 3

StructuredForests Edge Detector:
Run by Linux
Use parameters and models provided by the author
Edge detected image saved in /data/edges/

```
In [6]: import os
        import sys
        os.chdir('./Reference/StructuredForests')
        sys.path.append(os.getcwd())
```

```
In [12]: ### Fill your code here
         ### Report the accuracy obtained
         ### Report any improvements you have tried
         from StructuredForests import *

         IMG_NAMES = ['3096.jpg', '8023.jpg', '33039.jpg', '37073.jpg']

         rand = N.random.RandomState(1)
         options = {
                 "rgbd": 0,
                 "shrink": 2,
                 "n_orient": 4,
                 "grd_smooth_rad": 0,
                 "grd_norm_rad": 4,
                 "reg_smooth_rad": 2,
                 "ss_smooth_rad": 8,
                 "p_size": 32,
                 "g_size": 16,
                 "n_cell": 5,
                 "n_pos": 10000,
                 "n_neg": 10000,
                 "fraction": 0.25,
                 "n_tree": 8,
                 "n_class": 2,
                 "min_count": 1,
                 "min_child": 8,
                 "max_depth": 64,
                 "split": "gini",
                 "discretize": lambda lbls, n_class:
                     discretize(lbls, n_class, n_sample=256, rand=rand),
```

```
            "stride": 2,
            "sharpen": 2,
            "n_tree_eval": 4,
            "nms": True,
        }
        model = StructuredForests(options,rand=rand,model_dir='./model/')
```

In [13]:
```
def test(model, input_root, output_root):
    from skimage import img_as_float, img_as_ubyte
    from skimage.io import imread, imsave

    if not os.path.exists(output_root):
        os.makedirs(output_root)

    image_dir = os.path.join(input_root, 'img')
    file_names = filter(lambda name: name[-3:] == "jpg", os.listdir(image_dir))
    n_image = len(file_names)

    for i, file_name in enumerate(file_names):
        img = img_as_float(imread(os.path.join(image_dir, file_name)))

        edge = img_as_ubyte(model.predict(img))

        imsave(os.path.join(output_root, file_name[:-3] + "png"), edge)

        print("Processing Image %d/%d\r" % (i + 1, n_image))

test(model, "../../data", "../../data/edges")
```

```
Processing Image 1/10
Processing Image 2/10
Processing Image 3/10
Processing Image 4/10
Processing Image 5/10
Processing Image 6/10
Processing Image 7/10
Processing Image 8/10
Processing Image 9/10
Processing Image 10/10
```

In [17]:
```
for imagename in IMG_NAMES:
    imagepath = os.path.join('../../data/edges/', imagename.split('.')[0]+'.png')
    groundtruthpath = os.path.join('../../data/ground_truth/',imagename.split('.')[0]+
    img = cv2.imread(imagepath,1) # read the image from image path using opencv.
    cv2.imwrite(imagename.split('.')[0]+'_StructureForests.bmp',img)
    print('Accuracy: ' + str(evaluate(imagename.split('.')[0]+'_StructureForests.bmp'
```

```
Accuracy: 0.846056696524
Accuracy: 0.856056631758
```

```
Accuracy: 0.813647580003
Accuracy: 0.694930732314
```