Object Recognition and Image Understanding
Exercise Sheet 11

Sascha Stelling

July 9, 2018

1

To detect an arbitrary shaped object, Generalized Hough Transform (GHT) can be used. First, a reference point (x_c, y_c) inside the shape is picked from which a line is drawn towards the boundary of the object. Now the R-table is build by computing the gradient orientation θ and using this θ to compute the vector r from the boundary to the reference point. This reference point is now stored as a function of θ inside the R-table together with the gradient angle.
To detect an object the gradient orientation θ is now used to index into the R-table and the r vectors are used to vote for their reference point. If all the reference points satisfy the condition, that $H(x_c, y_c) > T$ with T being a set threshold, the desired object was detected.

2

Code (see generalized_hough.py):

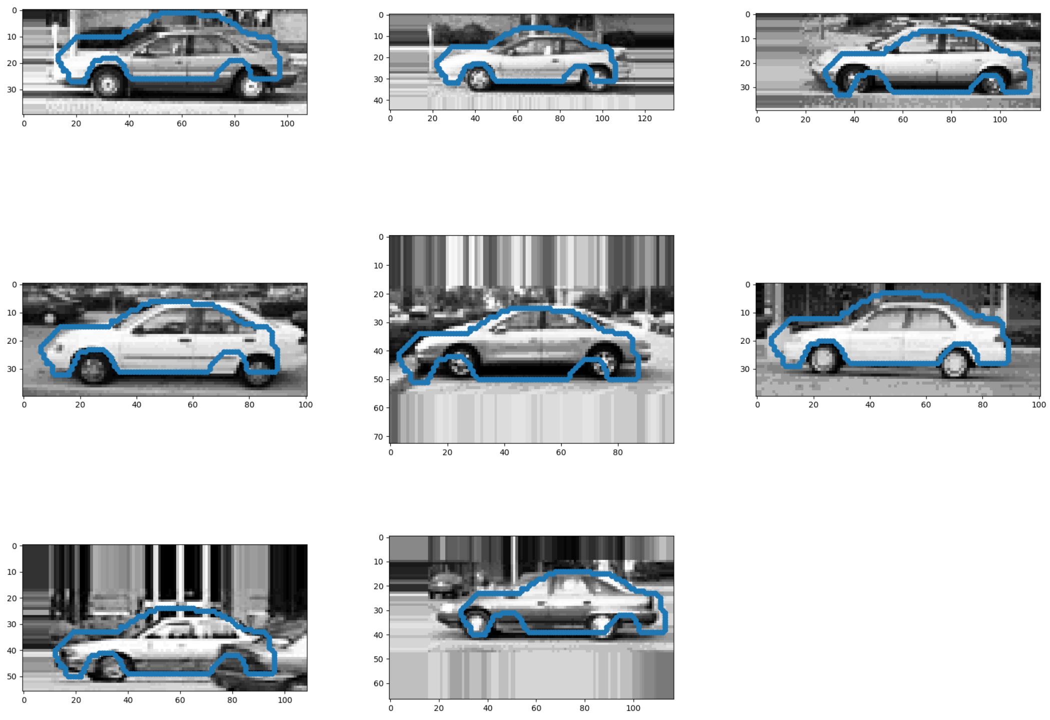
```
#!/usr/bin/env python2
import numpy as np

import os

import matplotlib.pyplot as plt

from collections import defaultdict

from skimage import feature, io
from scipy.ndimage.filters import sobel
from util import *

def gradientOrientation(img):
dx = sobel(img, axis=0, mode='constant')
dy = sobel(img, axis=1, mode='constant')
```

```
gradient = np.arctan2(dy, dx) * 180 / np.pi
return gradient

def buildRTable(templ, nrBins, p_ref):
gradient = gradientOrientation(templ)

r_table = defaultdict(list)
for (i,j),value in np.ndenumerate(templ):
if value:
r_table[gradient[i,j]].append((p_ref[0]-i, p_ref[1]-j))

return r_table

def accumulateGradients(rTable, img):
# Generalized Hough Transform with given R-table and image
edges = feature.canny(img)
gradient = gradientOrientation(edges)

accumulator = np.zeros(img.shape)
for (i,j),value in np.ndenumerate(edges):
if value:
for r in rTable[gradient[i,j]]:
accum_i, accum_j = i+r[0], j+r[1]
if accum_i < accumulator.shape[0] and accum_j < accumulator.shape[1]:
accumulator[accum_i, accum_j] += 1

return accumulator

def n_max(a, n):
# Return the n max elements and indices in a
indices = a.ravel().argsort()[-n:]
indices = (np.unravel_index(i, a.shape) for i in indices)
return [(a[i], i) for i in indices]

def generalizedHough(img, rTable):
accumulator = accumulateGradients(rTable, img)
m = n_max(accumulator, 1)
p_res = m[0][1]
return p_res

def showImage(img):
plt.imshow(img, cmap="gray")
plt.show()

def readImages(path):
# Read Images from path with each (sub)directory containing n files
```

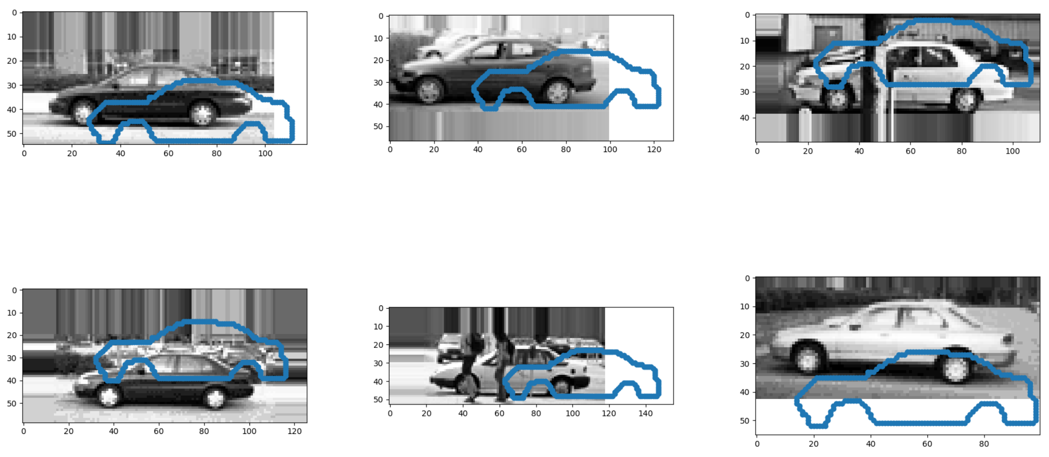
```
images = []
print "Reading images from " , path , "..."

# Go through all subdirs to find image files
for dirpath, dirs, files in os.walk(path, topdown=True):
i = 0
for filename in files:
i += 1
imagePath = os.path.join(dirpath, filename)
image = io.imread(imagePath)
if image is None:
print "Image " , format(imagePath) , " not read properly!"
else:
# Convert image to floating point
image = np.float32(image)/255.0
images.append(image)

numImages = len(images)/2

if numImages == 0:
print "No Images found! Aborting..."
sys.exit(0)

size = images[0].shape

print "Done! " , numImages , " images found."

return images, size

if __name__ == "__main__":
images, size = readImages("exercise_11/data/images/")
templ = np.load("exercise_11/data/template_car.npy")
for img in images:
img = padImgRandomly(img)
p_ref = (img.shape[0]/2, img.shape[1]/2)
rTable = buildRTable(templ, 3,  p_ref)
p_res = generalizedHough(img, rTable)

plotHoughResult(img, templ, p_ref, p_res)
```

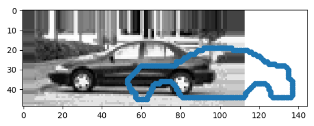
3

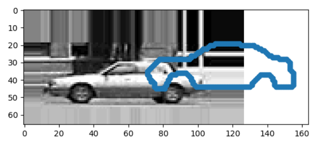
Good results:



Bad results:







- The reason why GHT could not locate the object properly in the bad results could be due to noise in the image resulting in bad corner detection. Noise in the image also leads to bad gradients which results in the points voting for the wrong resulting point.
- Since sometimes the second or third highest vote will be the best result, so it could be good to keep those candidates as well.