CPSC 425

Assignment 3: Face Detection

Robert MacEachern

42415091

m9c7

Script: a3.py

from PIL import Image, ImageDraw

import numpy as np

import math

from scipy import signal

import ncc

def MakePyramid(image, minsize):

'''

Creates a pyramid for an image.

Returns a list including the original PIL image followed

by all the PIL images of reduced size, using a scale factor

of 0.75 from one level to the next.

The pyramid will stop when any further reduction in size will

make the dimension of the image smaller than minsize.

'''

pyramid = []

im = image

while (im.size[0] >= minsize) and (im.size[1]) >= minsize:

pyramid.append(im);

im = im.resize((int(im.size[0] \* 0.75), int(im.size[1] \* 0.75)), Image.BICUBIC)

return pyramid

def ShowPyramid(pyramid):

'''

Joins the images in the list into a single horizontal image

and displays them with image.show()

'''

width = 0

height = 0

for img in pyramid:

width = width + img.size[0]

height = max(height, img.size[1])

image = Image.new("RGB", (width, height), 0xFFFFFF)

offsetX = 0

offsetY = 0

for img in pyramid:

image.paste(img, (offsetX, offsetY))

offsetX = offsetX + img.size[0]

offsetY = 0

image.show()

def FindTemplate(pyramid, template, threshold):

'''

Finds and marks all locations in pyramid at which the normalized

cross correlation of the template with the image is above the threshold.

Returns a PIL image of the largest image in the pyramid marked with red

rectangle's corresponding to the locations of template matches

'''

goalWidth = 15

# resize template

ratio = template.size[0] / goalWidth

template = template.resize((goalWidth, template.size[1] // ratio), Image.BICUBIC)

pointLists = []

for image in pyramid:

nccResult = ncc.normxcorr2D(image, template)

aboveThreshold = np.where(nccResult > threshold)

pointLists.append(zip(aboveThreshold[1], aboveThreshold[0]))

convert = pyramid[0].convert('RGB')

for i in range(len(pointLists)):

pointList = pointLists[i]

scaleFactor = 0.75 \*\* i

for pt in pointList:

ptx = pt[0] // scaleFactor

pty = pt[1] // scaleFactor

adjustx = template.size[0] // (2 \* scaleFactor)

adjusty = template.size[1] // (2 \* scaleFactor)

x1 = ptx - adjustx

y1 = pty - adjusty

x2 = ptx + adjustx

y2 = pty + adjusty

draw = ImageDraw.Draw(convert)

draw.rectangle([x1,y1,x2,y2], outline="red")

del draw

return convert

def runMe():

'''

For each of our three images, it constructs a pyramid with MakePyramid,

shows the pyramid with ShowPyramid, and finds the template matches with

FindTemplate. The template match image is saved and shown.

The threshold used is 0.532, giving an error rate of 0 for our three images.

'''

imgLocs = ['faces/judybats.jpg', 'faces/students.jpg', 'faces/tree.jpg']

for imgLoc in imgLocs:

img = Image.open(imgLoc)

img = img.convert('L')

pyramid = MakePyramid(img, 20)

ShowPyramid(pyramid)

templateLoc = 'faces/template.jpg'

template = Image.open(templateLoc)

template = template.convert('L')

# This threshold gives me the following results:

# judybats.jpg

# non-faces seen as faces: 1 (counting the hit on the front guy's lips)

# missed faces: 1

# students.jpg

# non-faces seen as faces: 3

# missed faces: 5

# tree.jpg

# non-faces seen as faces: 2

# missed faces: 0

# Error rate = (1 + 3 + 2) - (1 + 5 + 0) = 0

threshold = 0.532

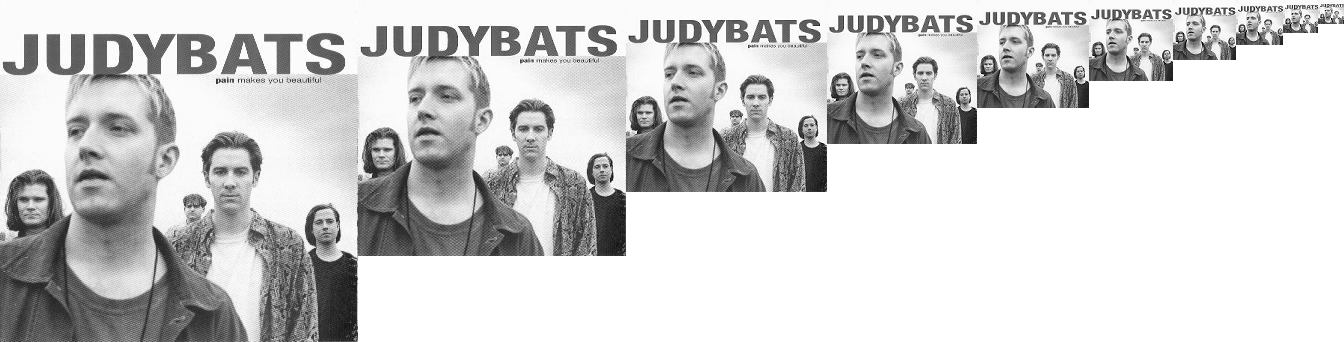
found = FindTemplate(pyramid, template, threshold)

found.save('found/' + imgLoc)

found.show()

runMe()

Pyramid



Final threshold selected: 0.532

