Федеральное государственное бюджетное образовательное учреждение   
высшего профессионального образования  
«Московский государственный технический университет имени Н.Э. Баумана»

(МГТУ им. Н.Э. Баумана)

Факультет ИУ – «Информатика и управление»

Кафедра ИУ-3 – «Информационные системы и телекоммуникации»

Отчёт по лабораторной работе 3 по дисциплине

«Цифровая обработка изображений»

Выполнил: Цветков А.А.

Группа: ИУ3-48М

Проверил: Алфимцев А.Н.

Москва, 2017

**Разработка программы распознавания объектов на спутниковых изображениях с использованием характерных признаков Хаара**

По заданию лабораторной работы 3 необходимо запрограммировать алгоритм определения характерных признаков Хаара для поиска объекта на изображении в соответствии с вариантом (в данном случае осуществляется поиск человека на изображениях).

*Характерным признаком Хаара* *fj* называется некоторая величина, вычисляемая для прямоугольной области, состоящей из пикселей и задаваемой параметрами: *x, y, w, h,*, где *x, y*- координата левого верхнего угла прямоугольной области в окне поиска; *w, h* – соответственно ширина и высота прямоугольной области в пикселях; ** - угол наклона одной из сторон прямоугольной области, причем *0*≤*x, x+w*≤X*, 0*≤*y, y+h*≤*, x, y>=0, w, h>0* и **{**} (рисунок 1). Кроме того, каждый характерный признак Хаара разбит на два или три “белых” и “черных” прямоугольника. Расширенный набор характерных признаков Хаара включает: граничные (1), линейные(2) и центровые(3) характерные признаки в соответствии с рисунком 2. Данные признаки заимствовали свою структуру и название из вейвлетного базиса Хаара.

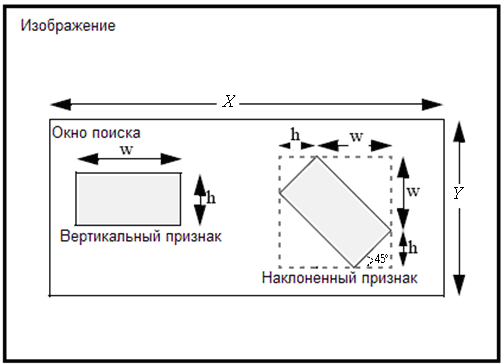


Рисунок 1 – Пример вертикальных и наклоненных признаков Хаара

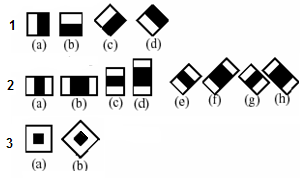


Рисунок 2 – Расширенный набор базовых характерных признаков Хаара

*Окном поиска* в кадре *It*(*V,W*)называется прямоугольная область *Okkt*(*X,Y*), содержащая множество пикселей *X* по горизонтали и *Y* по вертикали, в которой применяется основная функциональность рассматриваемого алгоритма для входного кадра *It*(*V,W*), где *k-* значение коэффициента масштабирования окна поиска, обычно *k=1, 1.2, 1.4, 1.8, 2* (рис. 2.15). Выбор масштаба признака может быть сделан экспертно на этапе обучения характерного признака на объекте.

Значение каждого признака *fj* вычисляется путем применения его к некоторой части кадра *It*(*V,W*)в окне поиска *Okkt*(*X,Y*). Для признака *fj*, состоящего из двух прямоугольников, значение вычисляется как разница между суммой значений полутоновых пикселей, накрытых черным и белым прямоугольным регионом. Для признака *fj*, состоящего из трех прямоугольников – из суммы значений полутоновых пикселей прямоугольника находящегося по центру вычитается сумма значений пикселей прямоугольников расположенных по краям.

Кроме задания набора характерных признаков, еще одной задачей при автоматическом выделении кистей в кадре *It*(*V,W*) с помощью характерных признаков Хаара является задача классификации (распознавания). Классификация выделенных характеристик необходима для определения того, к какому классу относится объект, содержащий эти характерные признаки (в данном случае рассматривается два класса: класс объекта и класс фоновых изображений).

Для того чтобы построить классификатор для каждого характерного признака Хаара *fj*, определяется простой классификатор *Khj*:

, (2.11)

где *fj* – значение характерного признака Хаара найденного в окне поиска *Okkt*(*X,Y*) кадра *It*(*V,W*), *θj*-значение порога, выбираемого экспериментально.

*Таким образом, для того чтобы распознавать объект на изображении необходимо:*

1. *Вычислить значения каждого характерного признака Хаара для данного объекта (этап обучения).*
2. *На основании этих значений выбрать масштаб, положение каждого характерного признака Хаара на объекте и значение порога (этап обучения).*
3. *Задать простые классификаторы (этап обучения).*
4. *Начать обработку изображения окном поиска слева направо, сверху вниз. Применить простые классификаторы к входному изображению в окне (этап распознавания).*
5. *Если большая часть простых классификаторов равна 1, то признать объект распознанным на изображении в данном окне поиска (этап распознавания).*

Для поиска человека на изображениях были созданы специальные примитивы Хаара (рисунок 3).

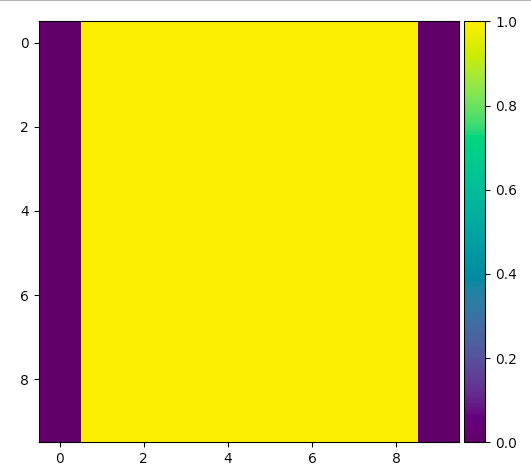
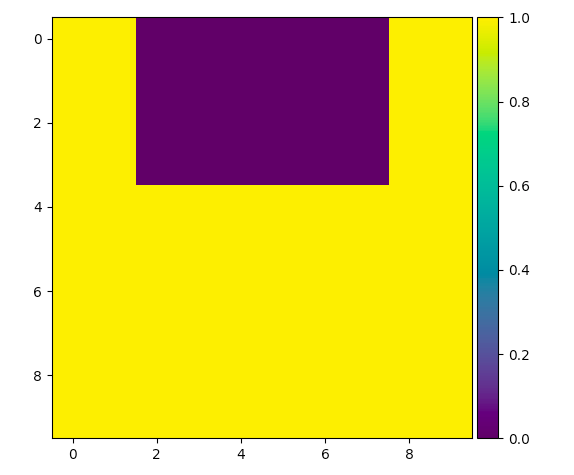
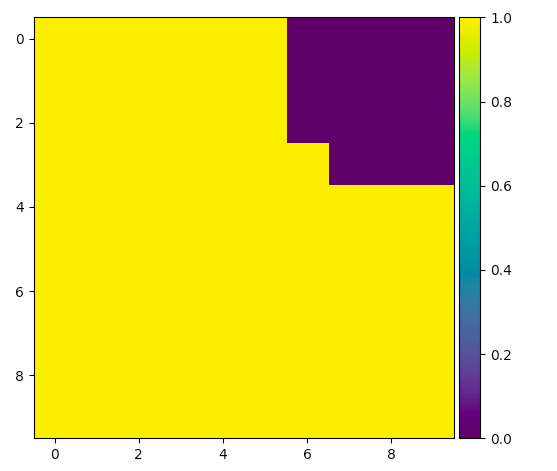


Рисунок 3 – Созданные примитивы Хаара

Результат поиска одного из примитивов с отображением формы примитива отображен на рисунке 4, всех созданных примитивов – на рисунке 5.



Рисунок 4 – Результат поиска одного из примитивов и его отображение



Рисунок 5 – Результат поиска всех созданных примитивов

Для каждого примитива было вычислено пороговое значение для дальнейшего поиска грузовых автомобилей на ряде тестовых изображений. При нахождении примитива на изображении и удовлетворении пороговому значению на исходном изображении помечается найденная интересующая область рисунок 6).



Для отображения найденного объекта были использованы морфологические фильтры дилатации и замыкания (рисунки 7, 8)



Рисунок 7 – Результат применения фильтра дилатации



Рисунок 8 – Результат применения фильтра замыкания

Найденные области помечаются, если они соответствуют пороговому значению по площади области (рисунок 9).



Рисунок 9 – Результат поиска человека на изображений

Данный алгоритм был протестирован на ряде исходных изображений (рисунки представлены в Приложении 1). Результаты тестирования отображены в таблице 1. Листинг программы представлен в Приложении 2.

Таблица 1 – Результаты распознавания блюд

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер изображения | Объекты для возможного распознавания | Распознано правильно | Распознано неправильно (ошибка 2 рода) | Не распознано (ошибка 1 рода) |
| 009 | 1 | 1 | 1 | 1 |
| 015 | 0 | 0 | 3 | 0 |
| 018 | 1 | 0 | 3 | 1 |
| 007 | 0 | 0 | 3 | 1 |

По результатам тестирования можно заключить, что поиск человека на изображениях с помощью примитивов Хаара является низкоэффективным. Это связано с малым размером искомого объекта, низкой контрастностью с окружающими объектами. Малые пороговые значения приводят к большому количеству затрачиваемого времени на обработку каждого изображения (1-5 мин.). Данный алгоритм допускает ошибки при наличии на изображениях предметов яркого цвета (сугробы, листы железа, крышы и др).

**Приложение 1 – Примеры тестирования**



****

**Приложение 2 - Листинг программы**

**import** os  
  
**import** matplotlib.patches **as** mpatches  
**import** numpy **as** np  
**from** matplotlib **import** pyplot **as** plt  
**from** skimage **import** io, color, draw  
**from** skimage.measure **import** label, regionprops  
**from** skimage.morphology **import** disk, dilation, closing, square  
  
basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))  
OUTPUT\_DIR = os.path.join(basedir, 'output')  
  
# Haar primitives  
haar = {  
 'haar1':  
 np.array([[1, 1, 1, 1, 1, 1, 0, 0, 0, 0],  
 [1, 1, 1, 1, 1, 1, 0, 0, 0, 0],  
 [1, 1, 1, 1, 1, 1, 0, 0, 0, 0],  
 [1, 1, 1, 1, 1, 1, 1, 0, 0, 0],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]),  
  
 'haar2':  
 np.array([[1, 1, 0, 0, 0, 0, 0, 0, 1, 1],  
 [1, 1, 0, 0, 0, 0, 0, 0, 1, 1],  
 [1, 1, 0, 0, 0, 0, 0, 0, 1, 1],  
 [1, 1, 0, 0, 0, 0, 0, 0, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1],  
 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]),  
  
 'haar3':  
 np.array([[0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0],  
 [0, 1, 1, 1, 1, 1, 1, 1, 1, 0]]),  
  
 'haar4':  
 np.array([[0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0],  
 [0, 0, 1, 1, 1, 1, 1, 1, 0, 0]]),  
  
}  
  
  
# show primitive  
# io.imshow(haar['haar12'])  
# io.show()  
  
# primitives searching  
**def primitive\_search**(img, haar\_type, th1, th2, point, marker\_size, draw\_haar):  
 # input image rows and cols calculating  
 rows = img.shape[0]  
 cols = img.shape[1]  
  
 prim\_amount = 0  
 cur\_haar = haar[haar\_type] # init haar primitive  
  
 print("%s - %s" % (haar\_type, point))  
  
 res = []  
 arr = []  
  
 # for each pixel in image with step of size window  
 **for** row **in** range(0, rows, win\_size):  
 **for** col **in** range(0, cols - 9, 2):  
 b = 0  
 w = 0  
  
 # for each pixel in window  
 **for** px **in** range(0, win\_size):  
 **for** py **in** range(0, win\_size):  
  
 # if pixel is under '1' in haar primitive  
 **if** cur\_haar[py][px] == 1:  
 b += img[row + py][col + px]  
 # if pixel is under '0' in haar primitive  
 **else**:  
 w += img[row + py][col + px]  
  
 **if** th1 < abs(b - w) < th2:  
 prim\_amount += 1  
  
 # add marker in window  
 **if not** draw\_haar:  
 ax.plot(col + win\_size / 2, row + win\_size / 2, point, markersize=marker\_size)  
 arr.append([col + win\_size / 2, row + win\_size / 2])  
  
  
 # draw haar primitive  
 **else**:  
 **for** px **in** range(0, win\_size):  
 **for** py **in** range(0, win\_size):  
 **if** cur\_haar[py][px] == 1:  
 ax.plot(col + px, row + py, 'bs', markersize=1)  
  
 **else**:  
 ax.plot(col + px, row + py, 'ws', markersize=1)  
  
 ax.text(col + win\_size / 2, row + win\_size / 2, str(abs(b - w))[:5], fontsize=10, color="red")  
  
 res.append(b - w)  
  
 print("max: %s" % max(res))  
 print("min: %s" % min(res))  
 print("amount: %s" % prim\_amount)  
  
 **return** arr  
  
 # fig.savefig(haar\_type + '.png')  
  
  
**def draw\_pix**(img, coord, amount, r, g, b):  
 rows = img.shape[0]  
 cols = img.shape[1]  
  
 **for** el **in** bus:  
 col, row = el  
 **if** bus.count(el) == amount:  
 rr, cc = draw.circle(row, col, 7, img.shape)  
 img[rr, cc, :] = (r, g, b)  
  
 **for** row **in** range(0, rows):  
 **for** col **in** range(0, cols):  
 **if** [col, row] **in** coord:  
 **if** img[row, col, 0] != r **and** img[row, col, 1] != g **and** img[row, col, 2] != b:  
 # draw circle in interesting point  
 img[row, col, :] = (r, g, b)  
 rr, cc = draw.circle(row, col, 7, img.shape)  
 img[rr, cc, :] = (r, g, b)  
  
 **return** img  
  
# open the input image  
image1 = io.imread(os.getcwd() + '/pool/009\_half.jpg')  
  
# convert image from rgb to gray  
image2 = color.rgb2gray(image1)  
  
win\_size = 10  
  
fig, ax = plt.subplots(figsize=(10, 10))  
  
ax.imshow(image1)  
  
# array for interesting points  
bus = []  
result = []  
  
bus += primitive\_search(image2, 'haar1', 34.2, 34.5, 'ws', 7, **False**)  
bus += primitive\_search(image2, 'haar2', 24, 25, 'y^', 6, **False**)  
bus += primitive\_search(image2, 'haar3', 31, 33, 'ro', 6, **False**)  
bus += primitive\_search(image2, 'haar4', 10, 15, 'ys', 6, **False**)  
  
# draw interesting points  
image3 = draw\_pix(image1, bus, 2, 255, 255, 255)  
  
# help image for result  
image4 = color.rgb2gray(image3)  
io.imsave(os.path.join(OUTPUT\_DIR, 'rgb2gray.jpg'), image4)  
  
selem = square(15)  
  
# dilation morph  
image5 = dilation(image4, selem)  
io.imsave(os.path.join(OUTPUT\_DIR, 'dilation.jpg'), image5)  
  
# closing morph  
image6 = closing(image5, selem)  
io.imsave(os.path.join(OUTPUT\_DIR, 'closing.jpg'), image6)  
  
# segment an image with image labelling  
label\_image = label(image6)  
  
# for each interesting region  
**for** region **in** regionprops(label\_image):  
  
 # take regions with large enough areas  
 **if** 500 < region.area < 800:  
 # draw rectangle around segmented objects  
 minr, minc, maxr, maxc = region.bbox  
 rect = mpatches.Rectangle((minc, minr), maxc - minc, maxr - minr,  
 fill=**False**, edgecolor='blue', linewidth=1)  
  
 # add rect to interesting area  
 ax.add\_patch(rect)  
 y0, x0 = region.centroid  
 ax.text(x0, y0, str(region.area), fontsize=15, color='blue')  
  
# set axis off  
ax.set\_axis\_off()  
fig.savefig(os.path.join(OUTPUT\_DIR, 'result.png'))  
plt.show()