Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана (Национальный исследовательский университет)»   
(МГТУ им. Н.Э. Баумана)

Факультет Информатики и систем управления (ИУ)

Кафедра Информационные системы и телекоммуникации (ИУ-3)

**Отчёт**

**по лабораторной работе №1.2**

**«Распознавание объектов по цвету»**

**по курсу**

**«Цифровая обработка изображений»,**

**1 курс, 2-й семестр.**

Выполнил: студентка группы ИУ3-21М

Смирнова К.Д.,

Проверил: Алфимцев А.Н.

Москва, 2020

**Задание**

1. Подготовить выборку 10 цветных цифровых изображений блюд (по

аналогии с примером внизу).

1. В среде Spyder (сборка Anaconda) на языке Python (3.х) создать проект

и подключить библиотеку scikit-image.

1. Из л.р. 1 взять модуль загрузки цветного цифрового изображения и

модуль обработки пикселей.

1. Запрограммировать формулу перевода цветного цифрового

изображения в цветовое пространство в соответствии с вариантом (В1:

RGB, В2: HSV, В3: CMYK, В4: YUV, В5: Hough Circle+HSV, В6:

Hough Circle +HSL)

1. Создать классификацию блюд из цифровых изображений.
2. Для каждого блюда определить цветовые характеристики. Задать

распределение значений каждого цвета.

1. Провести эксперимент по распознаванию блюд с визуализацией

результатов.

1. Дать комментарий каждой строчке кода!
2. Прислать программу преподавателю.
3. Построить график ошибок первого и второго рода по результатам

распознавания.

1. Подготовить и прислать отчет (титульный лист, задание, теоретическая часть, диаграмма структуры программы, принтскрины интерфейса и основных шагов работы программы, заключение и выводы, листинг программы с комментариями, список использованной литературы).

Вариант:

|  |  |
| --- | --- |
| **№** | **Цветовое пространство** |
| 2 | HSV |

**Цветовое пространство HSV**

Цветовая модель — математическая модель описания представления цветов в виде кортежей чисел.

HSV — цветовая модель, в которой координатами цвета являются:

* Hue — цветовой тон, (например, красный, зелёный или сине-голубой). Варьируется в пределах 0—360°, однако иногда приводится к диапазону 0—100 или 0—1.
* Saturation — насыщенность. Варьируется в пределах 0—100% или 0—1. Чем больше этот параметр, тем «чище» цвет, поэтому этот параметр иногда называют чистотой цвета. А чем ближе этот параметр к нулю, тем ближе цвет к нейтральному серому.
* Value — яркость. Также задаётся в пределах 0—100% или 0—1.

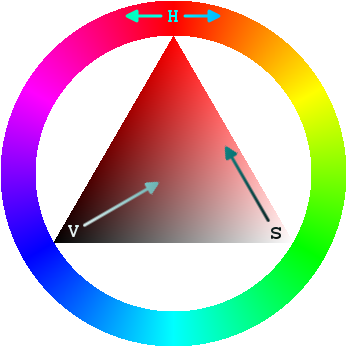
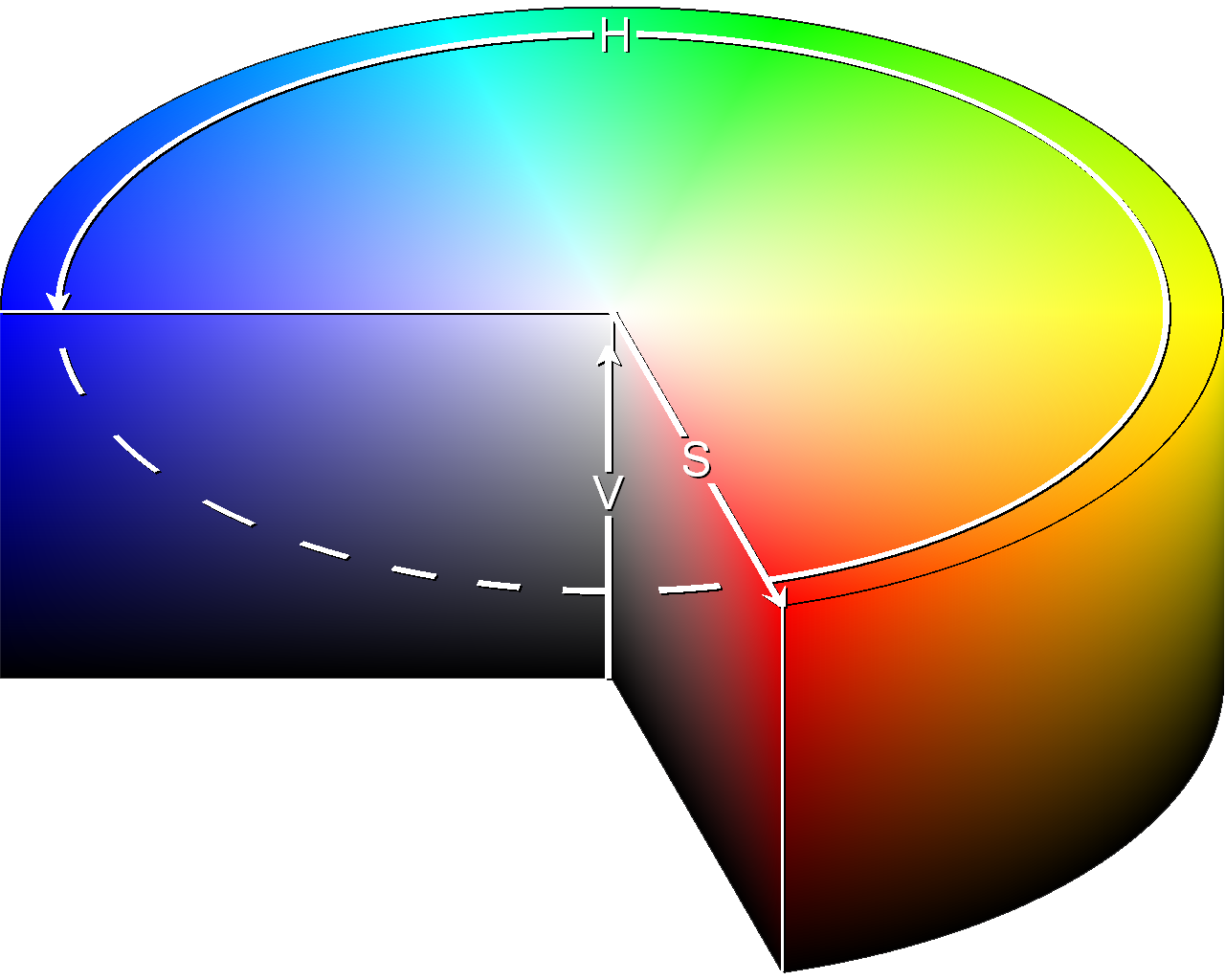
 

Рисунок 1 – Графические представления цветовой модели HSV

Считается, что особенность модели HSV в том, что её устройство ближе к человеческому восприятию цветов. Она представляет информацию о цвете в понятной форме: Что это за цвет? Насколько он насыщенный? Насколько он светлый или тёмный? Это позволило модели HSV закрепиться в применении в художественной деятельности, и процессах распознавания цветов.

**Из RGB в HSV**

Перевод значения пикселя изображения из цветовой модели RGB в HSV в данной лабораторной работе осуществлён в соответствии со следующей совокупностью формул:

После вычисления H, S, и V нормируются для диапазона [0…255].

**Подготовка данных**

Создаётся база изображений-шаблонов известных блюд, которые будет требоваться распознать на общих обрабатываемых фотографиях.

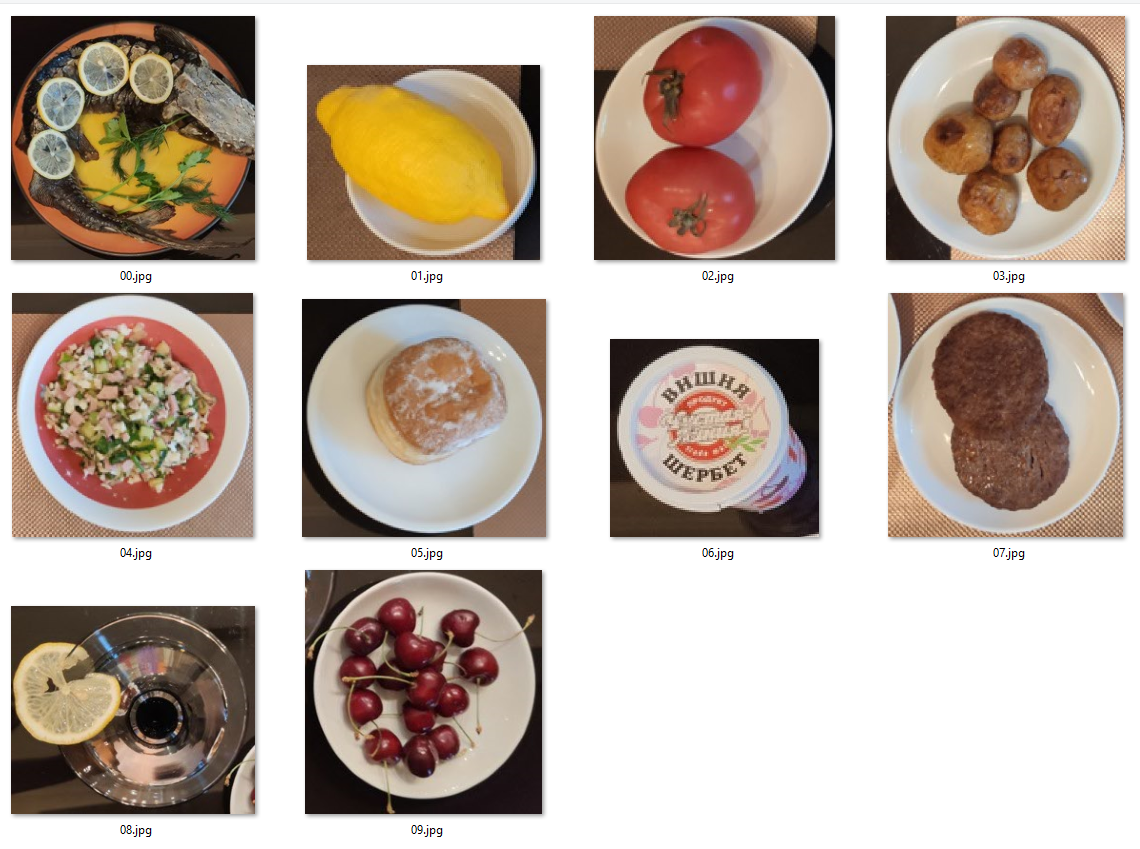


Рисунок 2 – Шаблоны блюд

Здесь по порядку: рыба, лимон, помидоры, картошка, салат, пончик, мороженое, котлеты, мартини, вишня.

Датасет и код размещены на <https://github.com/karinoizerr/Digital-image-processing/tree/master/lab1.2>

Алгоритм распознавания применяется к следующему набору фотографий:



Рисунок 3 – Набор обрабатываемых фотографий еды

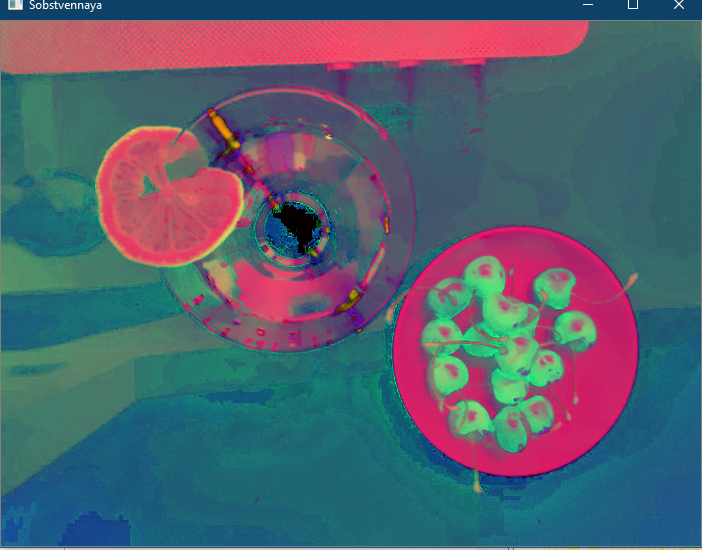
**Обработка**

Все используемые изображения (в том числе шаблоны) согласно варианту задания переводятся в цветовое пространство HSV функцией rgb\_hsv из общего листинга кода.

На рисунке 4 показано результирующее изображение в HSV, демонстрируемое инструментами OpenCV, трактующими его всё равно как BGR.

Для обнаружения фрагментов с блюдами используется функция fragment\_detection с помощью вычисления окружностей на изображении методом OpenCv HoughCircle, подразумевая, что еда находится на круглых тарелках. В качестве параметров функции отмечается минимальное расстояние между центрами окружностей, минимальный и максимальный радиусы окружностей, а также параметры порогов распознавания [1]. Результат действия представлен на рисунках 5 и 6.





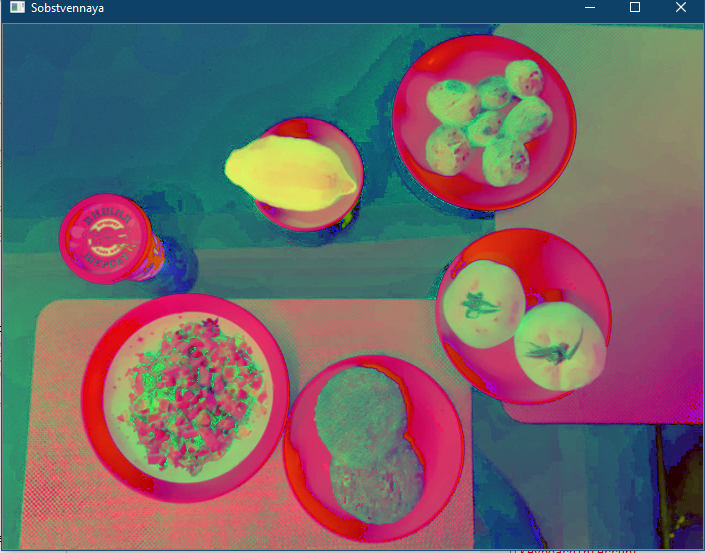
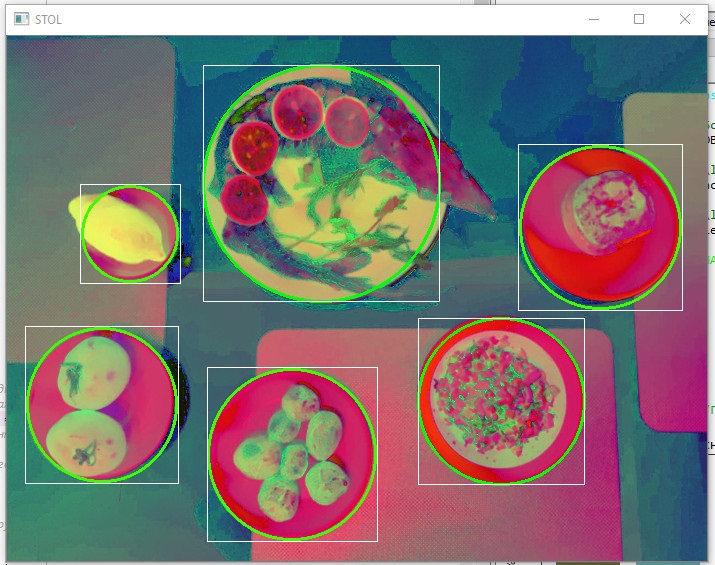


Рисунок 4 – Изображения в HSV



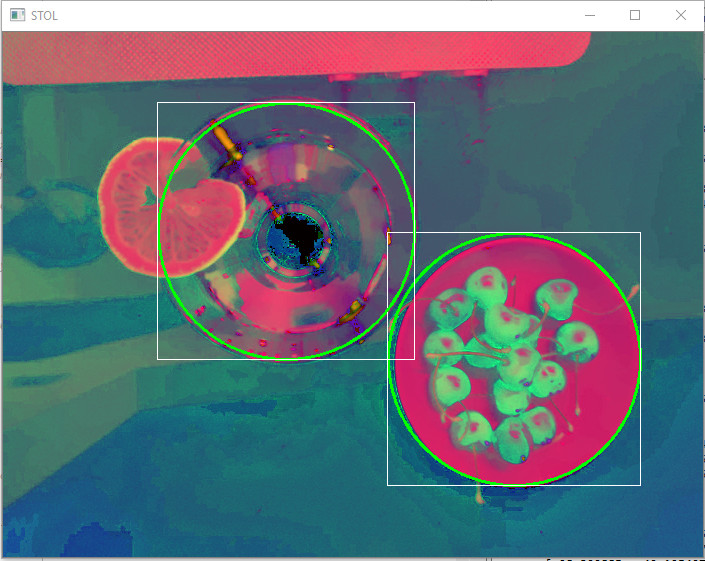




Рисунок 5 – Обнаружение окружностей на изображении и фрагментов с едой

И к найденным фрагментам и к шаблонам применятся анализ цветовых свойств путём кластеризации данных методом k-средних. Такой подход позволяет выявить несколько основных цветов изображения, средний цвет и доминантный цвет. На основе этих данных строится диаграмма цветов изображения. Реализовано это в функции find\_color методом OpenCv kMeans [2].

На рисунок 6 вынесены диаграмма цветов фрагмента (слева), и диаграмма цветов соответствующего фрагменту известного блюда (справа). Как видно, имеются определённые сходства.

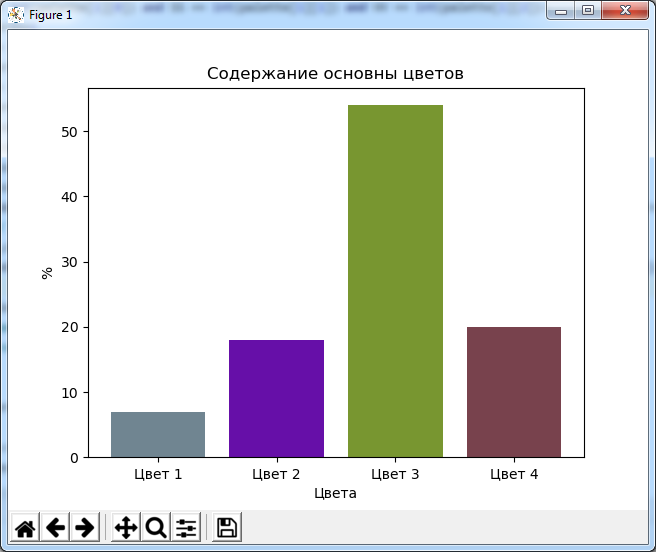
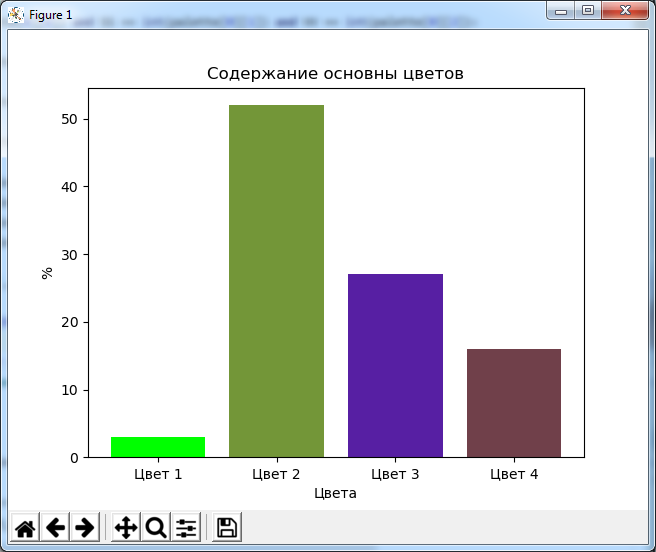


Рисунок 6 – Диаграммы цветов

Сравнивая найденные параметры фрагмента и шаблона, можно с заданной точностью определить принадлежность изображаемого блюда к одному из известных. Происходит это в функции detect

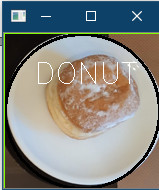
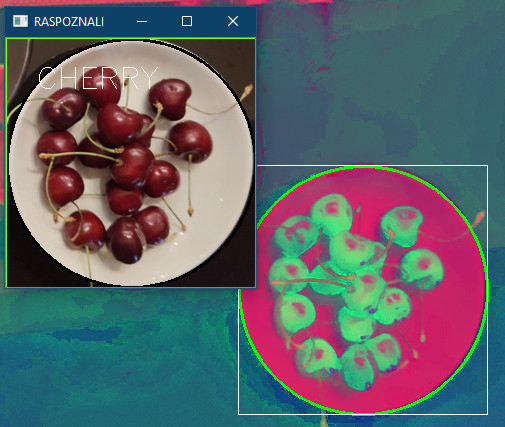
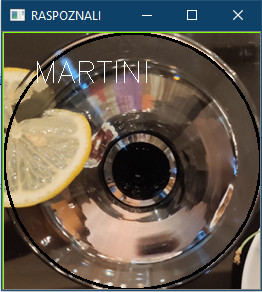
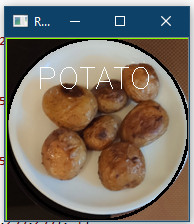
Для такого определения используется поиск минимального значения формулы:

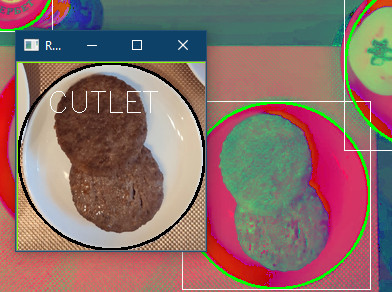
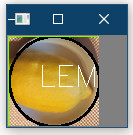
где – цветовая координата изображения определяемого фрагмента;

где – цветовая координата изображения искомого шаблона;

i – проходит по цветовым координатам HSV.

Таким образом, R, для которого формула будет давать минимальный результат, и будет блюдом, которое изображено на фрагменте. Пример результата распознавания показан на рисунке 7.

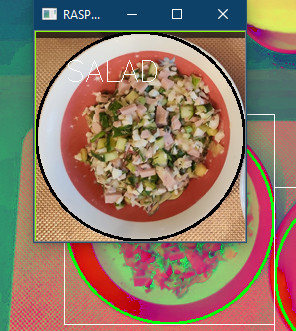
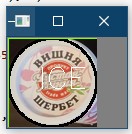
 

Рисунок 7 – Результаты распознавания

**Тестирование и ошибки**

Ошибки, возникающие при распознавании можно разделить на две группы: изначально неправильное выделение фрагмента с блюдом (1 и 2 на рисунке 9) и неверная классификация верно определённого фрагмента (3 на рисунке 9).

При гипотезе, что круг выделяет блюдо, вышеприведённое ошибочное выделение является неверным принятием неверной гипотезы, то есть ошибкой второго рода.

Если существующее блюдо не выделилось, то это непринятие верной гипотезы – ошибка первого рода.

Неточное определение точно выделенного фрагмента – ошибка распознавания.

На графике представлена информация об ошибках 1-го и 2-го рода, а также ошибках распознавания приобработке каждого изображения. Порядок данных соответствует приведённому порядку на рисунке 3 (слева навправо, сверху вниз).

Всплеск ошибок 1-го и 2-го рода наблюдается на изображениях с большим количеством блюд. Зависимость ошибок распознавания от этого фактора не установлена.

**Результаты и выводы**

Результатом выполнения лабораторной работы является алгоритм распознавания еды по цветовым характеристикам в цветовом пространстве HSV.

Тестирования показали, что точность работы зависит от количества блюд с едой на фотографии. Одним из главных выводов работы является понимание недостаточности цветового фактора для однозначного распознавания еды. Некоторые продукты имеют очень схожие цветовые характеристики, приводящие к ошибкам, что и видно по салатовой линии графика ошибок.

**Список источников**

[1] OpenCV 2.4.13.7 documentation [Электронный ресурс]. – Режим доступа: <https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=houghcircles#houghcircles>

[2] OpenCV 3.0.0-dev documentation [Электронный ресурс]. – Режим доступа:<https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_ml/py_kmeans/py_kmeans_opencv/py_kmeans_opencv.html>

**Приложение**

from skimage import data, io, color

import numpy as np

import os

import cv2

import matplotlib.pyplot as plt

# Функция перевода изображения из RGB в HSV

def rgb\_hsv(image):

col, row = image[:,:,0].shape

hsv\_image = [[[0 for ch in range(3)]for r in range(row)]for c in range(col)]

# Создание массива под переведённое изображение

for i in range(len(image)):

for j in range(len(image[i])):

R = image[i][j][0]/255

G = image[i][j][1]/255

B = image[i][j][2]/255

MAX = max(R,G,B)

MIN = min(R,G,B)

if (R == 0) and (G == 0) and (B == 0):

H = 0

S = 0

V = 0

else:

if MAX == MIN:

H = 0

else:

if MAX == R and G >= B:

H = (60 \* (G - B)/(MAX - MIN) + 0)/2

if MAX == R and G < B:

H = (60 \* (G - B)/(MAX - MIN) + 360)/2

if MAX == G:

H = (60 \* (B - R)/(MAX - MIN) + 120)/2

if MAX == B:

H = (60 \* (R - G)/(MAX - MIN) + 240)/2

if MAX == 0:

S = 0

else:

S = (1 - (MIN/MAX))\*255

V = MAX\*255

hsv\_image[i][j][0] = H

hsv\_image[i][j][1] = S

hsv\_image[i][j][2] = V

return hsv\_image

# Функция выделения фрагментов с блюдами

def fragment\_detection(image):

output = image.copy() # Выходное изображение - копия входного

image\_gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) # Перевод изображения в полутона

circles = cv2.HoughCircles(image\_gray, cv2.HOUGH\_GRADIENT, 2, param1 = 100, param2=60, minDist=200, minRadius=120, maxRadius=150) # Нахождение кругов на изображении

fragments = [] # Список изображений-фрагментов

for i in range(len(circles[0][:][:])):

output\_for\_crop = image.copy() # Изображение для конкретного фрагмента

x = circles[0][i][0] # X-координата центра круга

y = circles[0][i][1] # Y-координата центра круга

r = circles[0][i][2] # Радиус круга

cv2.circle(output, (x, y), r, (0, 255, 0), 2) # Рисование обнаруженного круга и квадрата обрезки фрагмента

cv2.rectangle(output,(x-r,y+r),(x+r,y-r),(255,255,255),1)

cv2.circle(output\_for\_crop, (x, y), r, (0, 255, 0), 2)

cv2.rectangle(output\_for\_crop,(x-r,y+r),(x+r,y-r),(255,255,255),1)

crop = output\_for\_crop[int(y-r):int(y+r),int(x-r):int(x+r)] # Выделение фрагмента

fragments.append(crop) # Добавление фрагмента в список

cv2.imshow('STOL',output)

cv2.waitKey(1000)

return fragments

# Функция нахождения цветовых свойств изображения

def find\_color(image):

average = image.mean(axis=0).mean(axis=0) # Средний цвет

pixels = np.float32(image.reshape(-1, 3))

n\_colors = 4 # Количество выделяемых основных цветов

criteria = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, 200, .1) # Критерии для вычисления кластеров цветов

flags = cv2.KMEANS\_RANDOM\_CENTERS

ret, label, palette = cv2.kmeans(pixels, n\_colors, None, criteria, 10, flags) # Нахождение кластеров цветов методом k-средних

\_, counts = np.unique(label, return\_counts=True)

dominant = palette[np.argmax(counts)] # Доминантный цвет

center = np.uint8(palette)

res = center[label.flatten()]

res2 = res.reshape((image.shape)) # Изображение с 4-мя найденными цветами

color1=0

color2=0

color3=0

color4=0

for i in range(len(res2)): # Вычисление процентного содержания 4-х кластерных цветов

for j in range(len(res2[i])):

HH = res2[i][j][0]

SS = res2[i][j][1]

VV = res2[i][j][2]

if HH == int(palette[0][0]) and SS == int(palette[0][1]) and VV == int(palette[0][2]):

color1+=1

if HH == int(palette[1][0]) and SS == int(palette[1][1]) and VV == int(palette[1][2]):

color2+=1

if HH == int(palette[2][0]) and SS == int(palette[2][1]) and VV == int(palette[2][2]):

color3+=1

if HH == int(palette[3][0]) and SS == int(palette[3][1]) and VV == int(palette[3][2]):

color4+=1

c1 = int(100 \* color1/(color1+color2+color3+color4))

c2 = int(100 \* color2/(color1+color2+color3+color4))

c3 = int(100 \* color3/(color1+color2+color3+color4))

c4 = int(100 \* color4/(color1+color2+color3+color4))

fig, ax = plt.subplots() # Построение столбиковой диаграммы кластерных цветов

langs = ['Цвет 1','Цвет 2','Цвет 3','Цвет 4']

colors = [c1,c2,c3,c4]

color = [np.round(palette[0]/255,2),np.round(palette[1]/255,2),np.round(palette[2]/255,2),np.round(palette[3]/255,2)]

ax.bar(langs,colors, color = color)

plt.title('Содержание основных цветов')

plt.xlabel('Цвета')

plt.ylabel('%')

plt.show()

return res2, average, dominant, palette

# Функция обнаружения во фрагменте признаков блюда

def detect(ave,dom,ave\_ref,dom\_ref):

MNK = []

H\_av = ave[0]

S\_av = ave[1]

V\_av = ave[2]

H\_dm = dom[0]

S\_dm = dom[1]

V\_dm = dom[2]

for i in range(len(averages\_ref)):

H\_av\_ref = ave\_ref[i][0]

S\_av\_ref = ave\_ref[i][1]

V\_av\_ref = ave\_ref[i][2]

H\_dm\_ref = dom\_ref[i][0]

S\_dm\_ref = dom\_ref[i][1]

V\_dm\_ref = dom\_ref[i][2]

F = ((H\_av - H\_av\_ref)\*\*2 + (S\_av - S\_av\_ref)\*\*2 + (V\_av - V\_av\_ref)\*\*2)/3

MNK.append(F) # Формула МНК

print('МНК: ',MNK)

print('Минимальное из МНК: ', min(MNK))

print('Индекс: ',MNK.index(min(MNK)))

return MNK.index(min(MNK))

food = cv2.imread('2.JPG') # Считывание общей фотографии

food\_hsv\_2 = np.array(rgb\_hsv(food),dtype=np.uint8) # Перевод изображения в HSV

cv2.imshow('Sobstvennaya',food\_hsv\_2)

cv2.waitKey(0)

dishes = fragment\_detection(food\_hsv\_2) # Нахождение фрагментов с блюдами

klasters=[] # Список кластерных изображений фрагментов

averages=[] # Список средних цветов фрагментов

dominants=[] # Список доминантных цветов фрагментов

palettes=[] # Список кластерных цветов фрагментов

for i in range(len(dishes)):

to\_show = dishes[i]

cv2.imshow('BLUDO', dishes[i])

cv2.waitKey(1000)

klaster, average, dominant, palette = find\_color(dishes[i]) # Вычисление цветовых характеристик фрагментов

klasters.append(klaster) # Заполнение списка кластерных изображений фрагментов

averages.append(average) # Заполнение списка средних цветов фрагментов

dominants.append(dominant) # Заполнение списка доминантных цветов фрагментов

palettes.append(palette) # Заполнение списка кластерных цветов фрагментов cv2.imshow('BLUDO\_KLASTER', klaster)

cv2.waitKey(1000)

print('Выделенные фрагменты')

print('Средний цвет: ', averages)

print('Доминантный цвет: ', dominants)

print('Палитра цветов: ', palettes)

#--------------------------------------------------------------------

files = os.listdir('Ref2') # Список файлов из папки с шаблонами блюд

dishes\_ref = []

for j in range(len(files)):

dish\_ref = cv2.imread('Ref2/' + files[j]) # Считывание изображений шаблонов

dish\_ref\_hsv = np.array(rgb\_hsv(dish\_ref),dtype=np.uint8) # Перевод шаблонов в полутона

dishes\_ref.append(dish\_ref\_hsv) # Заполнение списка

klasters\_ref=[] # Список кластерных изображений шаблонов

averages\_ref=[] # Список средних цветов шаблонов

dominants\_ref=[] # Список доминантных цветов шаблонов

palettes\_ref=[] # Список кластерных цветов шаблонов

for i in range(len(dishes\_ref)):

to\_show = dishes\_ref[i]

cv2.imshow('BLUDO\_REF', to\_show)

cv2.waitKey(1000)

klaster\_ref, average\_ref, dominant\_ref, palette\_ref = find\_color(to\_show) # Вычисление цветовых характеристик шаблонов

klasters\_ref.append(klaster\_ref) # Заполнение списка кластерных изображений шаблонов

averages\_ref.append(average\_ref) # Заполнение списка средних цветов шаблонов

dominants\_ref.append(dominant\_ref) # Заполнение списка доминантных цветов шаблонов

palettes\_ref.append(palette\_ref) # Заполнение списка кластерных цветов шаблонов

# cv2.imshow('BLUDO\_REF\_KLASTER', klaster\_ref) #Выделенные 4 основных цвета

# cv2.waitKey(1000)

print('Шаблоны')

print('Средний цвет: ', averages\_ref)

print('Доминантный цвет: ', dominants\_ref)

print('Палитра цветов: ', palettes\_ref)

# Присвоение фрагмента к блюду шаблону и вывод изображения с надписью

for i in range(len(klasters)):

Number = detect(averages[i],dominants[i],averages\_ref,dominants\_ref)

show = cv2.cvtColor(dishes[i], cv2.COLOR\_HSV2RGB) #dishes[i]

if Number != 100:

if Number == 0:

cv2.putText(show, "FISH", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 1:

cv2.putText(show, "LEMON", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 2:

cv2.putText(show, "TOMATO", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 3:

cv2.putText(show, "ICE", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 4:

cv2.putText(show, "SALAD", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 5:

cv2.putText(show, "DONUT", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 6:

cv2.putText(show, "ICE", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 7:

cv2.putText(show, "CUTLET", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 8:

cv2.putText(show, "MARTINI", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

if Number == 9:

cv2.putText(show, "CHERRY", (30,50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255,255,255), 1)

cv2.imshow('RASPOZNALI', show)

cv2.waitKey(0)

print(Number)

#####################################################################################