CodeListingEx4.png

Санкт-Петербургский Национально Исследовательский
Университет
информационных технологий, механики и оптики
Кафедра систем управления и информатики

Основы технического зрения

Отчет по лабораторной работе №2
Сегментация изображений

Работу выполнили:
Суздалев Олег
Караваев А.А
Группа: R3335
Преподаватель:
Шаветов С.В

Санкт-Петербург
2019

1 Цель работы.

Освоение основных способов сегментации изображений на се- мантические области.

2 Теоретическое обоснование применяемых методов и функций фильтрации изображений

Одной из важных задач технического зрения является сегментация изображений - разбиение исходной картинки на разные семантические области или классы. Например, мы должны понять, к какому классу будет принадлежать тот или иной пиксель на изобржаении, человек это или дорога, дорожный знак или велосипедист.

2.1 Бинаризация.

При использовании простой бинаризации мы просто выбираем порог интенсивности по-которому отсекаем или оставляем пиксели.

Этот порог можно выбирать как вручную, так и в автоматическом режиме статистическим методом Отсу

2.2 Поиск лица на фотографии.

Для поиска лица на фотографии самым простым, но не робастным методом является выбор пикселей, которые лежат в примерно следующем диапазоне $R > 95, G > 40, B < 20$. В лабораторной работе мы будем использовать данный метод, хотя, конечно, ясно, что при изменении освещения или расы испытумого, он не будет работать. В последнее время, первое место на пъедистале занимают нейросетевые подходы.

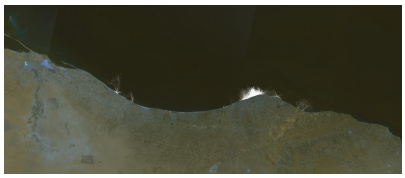
2.3 Сегментация объектов.

Сегментацию объектов на сцене можно делать с помощью метода кластеризации k-средних. Идея метода заключается в определении центров k-кластеров и отнесении к каждому кластеру пикселей, наиболее близко относя- щихся к этим центрам. Все пиксели рассматриваются как векторы x i , i = 1,p. Для того, чтобы освещенность играла меньшую роль стоит перевести картинку из пространства цветов RGB в CIE-LAB.

2.4 Сегментация текстур.

При текстурной сегментации для описания текстуры применя- ются три основных метода: статистический, структурный и спек- тральный. В лабораторной

работе будем рассматривать статисти- ческий подход, который описывает текстуру сегмента как гладкую, грубую или зернистую. Порядок такое: сначала бинаризуем, потом применяем морфологическое преобразование.



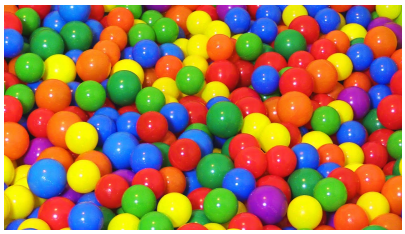


Рис. 1: Исходные изображения.

2.5 Листинг.

Код написанной библиотеки.

```
import cv2
import numpy as np

def detect_face(img:np.ndarray, bounds: tuple=None) -> np.ndarray:
    """ Detect face based on skin color

    Arguments:
        img {np.ndarray} -- Source image

    Keyword Arguments:
        bounds {tuple} -- Lower and upper bounds for skin color (default: {None
    })

    Returns:
        np.ndarray -- Mask with only face
    """
    if not bounds:
        bounds = []
        bounds.append(np.array([0, 60, 80], dtype = "uint8"))
        bounds.append(np.array([20, 255, 255], dtype = "uint8"))

    converted = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    skinMask = cv2.inRange(converted, bounds[0], bounds[1])

    # apply a series of erosions and dilations to the mask
    # using an elliptical kernel
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (11, 11))
    skinMask = cv2.erode(skinMask, kernel, iterations = 2)
    skinMask = cv2.dilate(skinMask, kernel, iterations = 2)

    # blur the mask to help remove noise, then apply the
    # mask to the frame
    skinMask = cv2.GaussianBlur(skinMask, (3, 3), 0)
    skin = cv2.bitwise_and(img, img, mask = skinMask)

    return skin
```

Код исполняемого файла лабораторной работы.

```
import cv2
from closedcv.segmentation import *
from skimage.segmentation import slic
from skimage.color import xyz2lab
from skimage import img_as_ubyte
from skimage.segmentation import (morphological_chan_vese,
                                  morphological_geodesic_active_contour,
                                  checkerboard_level_set)

if __name__=="__main__":
    img = cv2.imread('./input/face.jpg', 1)

    dir_ = './res/'

    ret, thresh = cv2.threshold(img,127,255,cv2.THRESH_BINARY)

    cv2.imwrite(dir_ + 'binarized.jpg', thresh)

    cv2.imwrite(dir_ + 'face.jpg', detect_face(img))

    # Convert to CIE-LAB+
    img = xyz2lab(cv2.imread('./input/balls.jpg', 1))
    vectorized = img.reshape((-1, 3))
    vectorized = np.float32(vectorized)
    K = 3
    attempts=10
    criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
    ret,label,center=cv2.kmeans(vectorized,K,None,criteria,attempts,cv2.
    KMEANS_PP_CENTERS)
    center = np.uint8(center)
    res = center[label.flatten()]
    result_image = res.reshape((img.shape))
    cv2.imwrite(dir_ + 'k_means.jpg', result_image)

    img = cv2.imread('./input/texture.jpeg', 0)

    ret, thresh = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
    cv2.imwrite(dir_ + 'texture_thresh.jpg', thresh)

    kernels = [np.ones((i*5, i*5), np.uint8) for i in range(6)]
    for kernel in kernels:
        closing = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, kernel)
        cv2.imwrite(dir_ + 'texture_k' + str(kernel.shape) + '.jpg', closing)
```

2.6 Результирующие изображения.

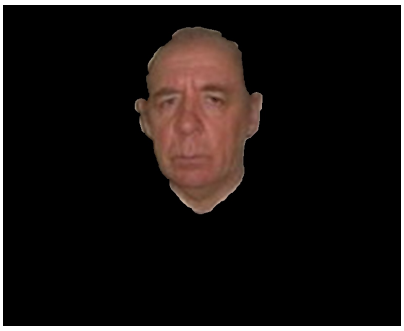


Рис. 2: Сегментированное лицо.

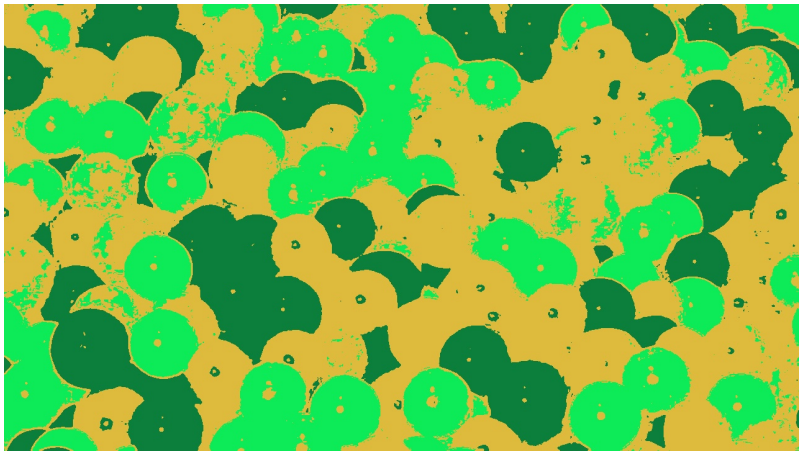


Рис. 3: Кластеризованные шарики.

Рис. 4: Сегментация текстуры при различном размере ядра(5x5, 15x15, 25x25).

3 Вывод.

В данной работе были изучены классиеческие методы сегментирования цифровой картинки. Так же, было получено, что работа в пространстве RGB далеко не всегда предпочтительна из-за освещения на картинке.