ЛАБОРАТОРНАЯ РАБОТА 2
ОБЩИЕ ПРИНЦИПЫ РАБОТЫ С ИЗОБРАЖЕНИЯМИ В SKIMAGE

Знакомство с skimage

skimage –это библиотека для обработки изображений, которая работает с массивами numpy. Для того чтобы импортировать библиотеку необходимо выполнить команду:

```
import skimage
```

Большинство функций библиотеки находится в подмодулях. Для того чтобы их просмотреть используется функция help.

Модуль skimage.data содержит в себе наборы изображений для анализа и тестирования различных алгоритмов сегментации.

Для того, чтобы загрузить изображение используется функция imread модуля io.

Изображения в skimage представляется как массивы библиотеки numpy.

Изображение:	np.ndarray
Пиксели	Значения массива: a[2, 3]
Каналы	Размерность массива
Кодировка изображения	dtype (np.uint8, np.uint16, np.float)
Библиотеки	numpy, skimage, scipy

```
import numpy as np
check = np.zeros((9, 9))
check[::2, 1::2] = 1
check[1::2, ::2] = 1
import matplotlib.pyplot as plt
plt.imshow(check, cmap='gray', interpolation='nearest')
```

Другие библиотеки Python доступные для обработки изображений так же работает с массивами NumPy. scipy.ndimage содержит базовые функции для фильтрации, математической морфологии, свойств объектов. Так же существуют другие мощные библиотеки для работы с изображениями, такие как OpenCV (компьютерное зрение), Insight Segmentation and Registration Toolkit (3D изображения) и другие. Однако, они не столь питоноподобные и дружественные синтаксису NumPy.

В библиотеке skimage есть различные функции:

1. Шаблонные и современные высокоуровневыми алгоритмы.
2. Фильтры, которые преобразовывают изображения
   2.1.На основе NumPy вычислений
   2.2.Общие алгоритмы фильтрации
3. Функции уменьшения размерности данных: построение гистограммы, нахождение минимума и максимума, обнаружение углов
4. Другие функции: ввод-вывод данных, визуализация и т.д.

Кроме этого есть развитая система помощи:

1) Веб-сайт: http://scikit-image.org/
2) Галерея примеров: http://scikit-image.org/docs/stable/auto_examples/

Задание 1. Загрузите библиотеку skimage, отобразите список модулей, составляющих ее. Загрузите изображение camera из модуля data, узнайте его размеры. Загрузите изображение *I* согласно Вашему варианту. Найдите минимальное и максимальное значение по каждому каналу в изображении *I*. Установите на изображении *I* в ноль интенсивности тех пикселей, значение яркости в которых по синему каналу меньше медианы.

Задание 2. Реализуйте код, приведенный в примерах.

Ввод-вывод, типы данных, цветовые пространства

Для импорта изображений используется модуль io библиотеки skimage. Для того, чтобы прочитать изображение используется функция imread(). Считывать изображения можно так же по URL-адресу. Для сохранения изображения используется функция imsave.

```
from skimage import io
import os
filename = os.path.join(skimage.data_dir, 'camera.png')
camera = io.imread(filename)
logo = io.imread('http://scikit-image.org/_static/img/logo.png')
io.imsave('local_logo.png', logo)
```

Функция imread работает со всеми форматами, которые поддерживает Python Imaging Library или любой другой плагин ввода-вывода, который можно передать как параметр функции imread.

Массивы ndarray, содержащие изображения, могут быть рассмотрены как целые (знаковые или беззнаковые) или вещественные. Возможны различные целочисленные типы: 8-, 16 или 32-ух разрядные, знаковые или беззнаковые. Согласно skimage соглашению: изображения с вещественными значениями находятся на интервале [-1, 1].

```
from skimage import img_as_float
camera_float = img_as_float(camera)
print(camera.max(), camera_float.max())
```

Некоторые методы обработки изображений могут работать только с вещественными типами данных, поэтому тип выходного изображения может отличатся от входного.

```
from skimage import filters
camera_sobel = filters.sobel(camera)
print(camera_sobel.max())
```

Утилитные функции используются в skimage для преобразования как типа данных, так и диапазона данных. Согласно соглашению skimage это функции: util.img_as_float, util.img_as_ubyte и другие.

Цветные изображения имеют форму (N, M, 3) или (N, M, 4) (когда альфа-канал определяет прозрачность).

```
import scipy
face = scipy.misc.face()
print(face.shape)
```

Функции преобразования цветовых пространств (RGB, HSV, LAB) находятся в модуле skimage.color : color.rgb2hsv, color.lab2rgb. Большинство функций в skimage могут работать с трехмерными изображениями, как входными аргументами.

З**адание 3.** Реализуйте код, приведенный в примерах.

Обработка изображений.

Локальные фильтры заменяют значение интенсивности в одном пикселе результатом вычисления функции, зависящей от соседних пикселей. Функция может быть, как линейная, так и нелинейная. Возможные варианты соседних пикселей: квадрат, круг или другая фигура более сложной формы.

Пример использование фильтра Собеля, в основе которого для расчета градиентов используется следующий фильтр.

1	2	1
0	0	0
-1	-2	-1

В скрипте показан пример использования фильтра Собеля для детектирования горизонтальных линий.

```
from matplotlib import pyplot as plt
from skimage import data
text = data.text()
hsobel_text = filters.sobel_h(text)
plt.figure()
plt.imshow(text, cmap = 'gray')
plt.figure()
plt.imshow(hsobel_text, cmap = 'gray')
```

Нелокальные фильтры используют все изображение или его часть для изменения интенсивности в одном пикселе. В примере показано изменение контрастности в областях.

```
from skimage import exposure
camera = data.camera()
camera_equalized = exposure.equalize_hist(camera)
plt.figure()
plt.imshow(camera, cmap = 'gray')
plt.figure()
plt.imshow(camera_equalized, cmap = 'gray')
```

Модификация изображения структурным элементом. Эрозия – значение пикселя заменяется на минимальное значение, которое охватывается в его окрестности структурным элементом.

```
from skimage import morphology
a = np.zeros((7,7), dtype=np.uint8)
a[1:6, 2:5] = 1
print(a)
b = morphology.binary_erosion(a, morphology.diamond(1)).astype(np.uint8)
print(b)
```

Дилатация – значение пикселя заменяется на максимальное значение, которое охватывается в его окрестности структурным элементом.

```
a = np.zeros((5, 5), dtype = np.uint8)
a[2, 2] = 1
print(a)
b = morphology.binary_dilation(a, morphology.diamond(1)).astype(np.uint8)
print(b)
```

Операция открытия – последовательное выполнение сначала эрозии, а затем дилатации. Эта операция позволяет удалить мелкие объекты и сгладить углы.

Операции математической морфологии можно так же применять для полутоновых изображений. В пакете так же реализованы другие методы морфологической обработки: выделение остова, границ и т.д. Базовые операции морфолгической обработки реализованы не только в skimage, но и в scipy.ndimage.

```
from skimage.morphology import disk
from skimage import data
from skimage import filters
import matplotlib.pyplot as plt
coins = data.coins()
coins_zoom = coins[10:80, 300:370]
median_coins = filters.median(coins_zoom, disk(1))
from skimage import restoration
tv_coins = restoration.denoise_tv_chambolle(coins_zoom, weight=0.1)
gaussian_coins = filters.gaussian(coins_zoom, sigma=2)
```

Задание 4. Реализуйте код, приведенный в примере. Выполните визуализацию полученных результатов.

Сегментация изображений

Сегментация изображений – это выделение различных регионов на изображении. В случае бинарной сегментации в изображении выделяются два типа пикселей: фоновые и пиксели переднего плана. Метод Оцу – один из самых распространенных методов для пороговой сегментации изображений.

```
from skimage import data
from skimage import filters
camera = data.camera()
val = filters.threshold_otsu(camera)
mask = camera < val
plt.figure()
plt.subplot(121), plt.axis('off')
plt.imshow(camera, cmap = 'gray')
plt.subplot(122), plt.axis('off')
plt.imshow(mask, cmap = 'gray')
```

После того, как объекты были сегментированы, необходимо их отделить (выделить) друг от друга. Для этих целей можно использовать маркирование.

```
import numpy as np
n = 20
l = 256
im = np.zeros((l, l))
points = l * np.random.random((2, n ** 2))
im[(points[0]).astype(np.int), (points[1]).astype(np.int)] = 1
im = filters.gaussian(im, sigma=l / (4. * n))
blobs = im > im.mean()
from skimage import measure
all_labels = measure.label(blobs)
blobs_labels = measure.label(blobs, background=0)
```

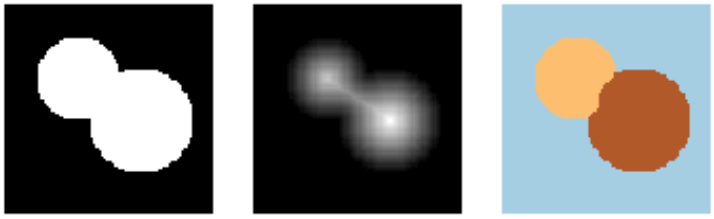
Задание 5. Реализуйте код и выполните визуализацию с различными цветовыми картами.

Водораздельный метод сегментации.

Если внутри области есть маркированные области, это можно использовать для сегментации всего изображения. В основе водораздельного метода лежит наращивание областей, которая заполняет «бассейны» на изображении.

```
from skimage import morphology
from skimage.feature import peak_local_max
# Построение изображения из двух пересекающихся областей
x, y = np.indices((80, 80))
x1, y1, x2, y2 = 28, 28, 44, 52
r1, r2 = 16, 20
mask_circle1 = (x - x1) ** 2 + (y - y1) ** 2 < r1 ** 2
mask_circle2 = (x - x2) ** 2 + (y - y2) ** 2 < r2 ** 2
image = np.logical_or(mask_circle1, mask_circle2)
# Теперь их необходимо разделить
# Построим маркеры как локальные максимумы расстояния
from scipy import ndimage
distance = ndimage.distance_transform_edt(image)
local_maxi = peak_local_max(distance, indices=False, footprint=np.ones((3, 3)), labels=image)
markers = morphology.label(local_maxi)
labels_ws = morphology.watershed(-distance, markers, mask=image)
```

Задание 6. Реализуйте код и выполните визуализацию с различными цветовыми картами. Пример полученного результата



Алгоритм случайного блуждания

Алгоритм случайного блуждания аналогичен водораздельному методу, однако в его основе лежит вероятностный подход. Он основан на «диффузии» маркеров по изображению.

```
from skimage import segmentation
# Нулевые пиксели отмечаются значением -1
markers[~image] = -1
labels_rw = segmentation.random_walker(image, markers)
plt.figure()
plt.subplot(121), plt.axis('off')
plt.imshow(image, cmap = 'gray')
plt.subplot(122), plt.axis('off')
plt.imshow(labels_rw, cmap = 'Paired')
```

Задание 7. Реализуйте код приведенный в примере.

Задание 8. Используя функцию coins из модуля data, сегментируйте его используя различные методы: Оцу, адаптивной пороговой сегментации, водораздельный метод и алгоритм случайного блуждания.