Лабораторная работа №4

Преобразование изображений с использованием библиотеки PIL (Python Imaging Library).

Создание миниатюр

Создавать миниатюры с помощью PIL очень просто. Метод `thumbnail()` принимает кортеж, в котором задается новый размер, и преобразует изображение в миниатюру указанного размера. Вот как создается миниатюра, для которой длина наибольшей стороны равна 128 пикселей:

```
pil_im.thumbnail((128,128))
```

Копирование и вставка областей

Обрезка (кадрирование) изображения производится методом `crop()`:

```
box = (100,100,400,400)
region = pil_im.crop(box)
```

Прямоугольная область определяется 4-кортежем, в котором задаются координаты сторон в порядке (левая, верхняя, правая, нижняя). В PIL используется система координат с началом (0, 0) в левом верхнем углу. С вырезанной областью можно затем производить различные операции, например повернуть и вставить в то же место методом `paste()`:

```
region = region.transpose(Image.ROTATE_180)
pil_im.paste(region,box)
```

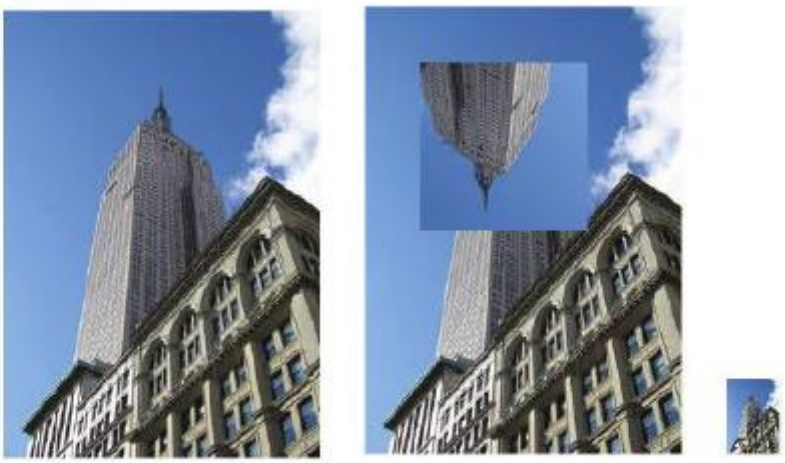

Рис. 1.1. Результаты обработки изображений с помощью PIL

Изменение размера и поворот

Для изменения размера изображения служит метод `resize()`, которому передается кортеж, определяющий новый размер:

```
out = pil_im.resize((128,128))
```

Для поворота изображения вызывается метод `rotate()` и задается угол в направлении против часовой стрелки:

```
out = pil_im.rotate(45)
```

Результаты работы некоторых примеров показаны на рис. 1.1. Слева показано исходное изображение, затем
результат вставки вырезанной и повернутой области и, наконец, миниатюра.

Изменение размера изображения

Массивы `NumPy` будут нашим основным средством при работе с изображениями и данными. Но для массивов нет простого способа изменить размер, хотя для изображений эта операция очень полезна. Чтобы написать функцию изменения размера, мы можем воспользоваться показанным выше преобразованием объектов изображений. Добавьте следующую функцию в файл *imtools.py*:

```
def imresize(im,sz):
  """ Изменить размер массива с помощью PIL. """
  pil_im = Image.fromarray(uint8(im))
  return array(pil_im.resize(sz))
```

Эта функция нам еще не раз пригодится.

Изолинии и гистограммы изображений

Рассмотрим два примера специальных графиков: изолинии и гистограммы изображений. Визуализация изолиний изображения (или изолиний других двумерных функций) может оказаться очень полезной. Для этого нужны полутоновые изображения, потому что изолинии строятся по значению какой-нибудь одной величины. Делается это так:

```
from PIL import Image
from pylab import *

# прочитать изображение в массив
im = array(Image.open('empire.jpg').convert('L'))

# создать новый рисунок
figure()

# не использовать цвета
gray()

# показать изолинии относительно левого верхнего угла
contour(im, origin='image')
axis('equal')
axis('off')
```

Как и раньше, метод `convert()` преобразует исходное изображение в полутоновое.

Гистограмма изображения – это график распределения значений пикселей. Область возможных значений разбивается на интервалы, и для каждого интервала определяется количество пикселей, значения которых попадают в этот интервал. Для построения гистограммы полутонового изображения применяется функция `hist()`:

```
figure()
hist(im.flatten(),128)
show()
```

Ее второй аргумент задает количество интервалов. Отметим, что изображение сначала необходимо линеаризовать, потому что `hist()` ожидает получить одномерный массив. Метод `flatten()` преобразует любой массив в одномерный, располагая значения по строкам. На рис. 1.3 показаны изолинии и гистограмма.

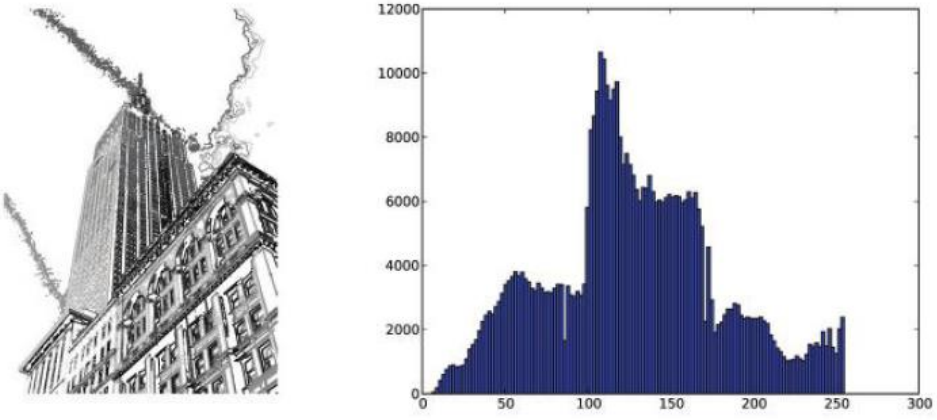

Рис. 1.3. Примеры визуализации изолиний изображения и построения гистограмм с помощью `Matplotlib`

Выравнивание гистограммы

Весьма полезным примером преобразования яркости является *выравнивание гистограммы*. Эта операция изменяет гистограмму яркости, так чтобы результирующая гистограмма содержала все возможные значения яркости и при этом примерно в одинаковом количестве. Она часто применяется для нормировки яркости перед последующей обработкой, а также для повышения контрастности.

В данном случае для преобразования используется *функция распределения* (cumulative distribution function, cdf) значений пикселей в изображении (нормированная так, чтобы привести значения к требуемому диапазону).

Добавьте следующую функцию в файл *imtools.py*:

```
def histeq(im,nbr_bins=256):
  """ Выравнивание гистограммы полутонового изображения. """

  # получить гистограмму изображения
  imhist,bins = histogram(im.flatten(),nbr_bins,normed=True)
  cdf = imhist.cumsum() # функция распределения
  cdf = 255 * cdf / cdf[-1] # нормировать

  # использовать линейную интерполяцию cdf для нахождения
  # значений новых пикселей
  im2 = interp(im.flatten(),bins[:-1],cdf)
  return im2.reshape(im.shape), cdf
```

Эта функция принимает полутоновое изображение и количество интервалов в гистограмме, а возвращает изображение, для которого

гистограмма выровнена с помощью функций распределения. Обратите внимание на использование последнего элемента (с индексом –1) для нормировки cdf на диапазон 0 ... 1. Попробуйте применить ее к какому-нибудь изображению:

```
from PIL import Image
from numpy import *

im=array(Image.open('AquaTermi_lowcontrast.jpg').convert('L'))
im2,cdf = imtools.histeq(im)
```

На рис. 1.6 и 1.7 приведены примеры выравнивания гистограммы. Сверху показаны гистограммы до и после выравнивания, а также сама функция распределения. Как видите, контрастность увеличилась, и детали в темных участках теперь видны более отчетливо.

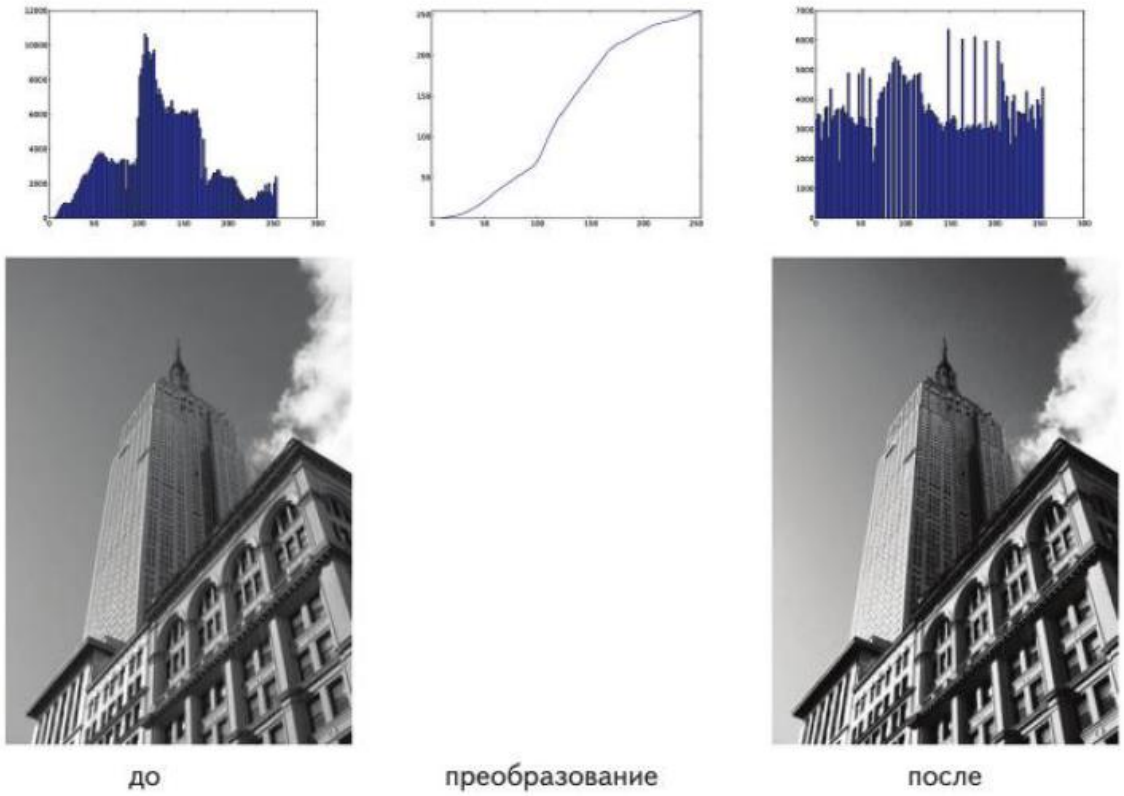

Рис. 1.6. Пример выравнивания гистограммы.
Слева – исходное изображение и его гистограмма.
В центре – функция преобразования уровня яркости.
Справа – изображение и гистограмма после выравнивания

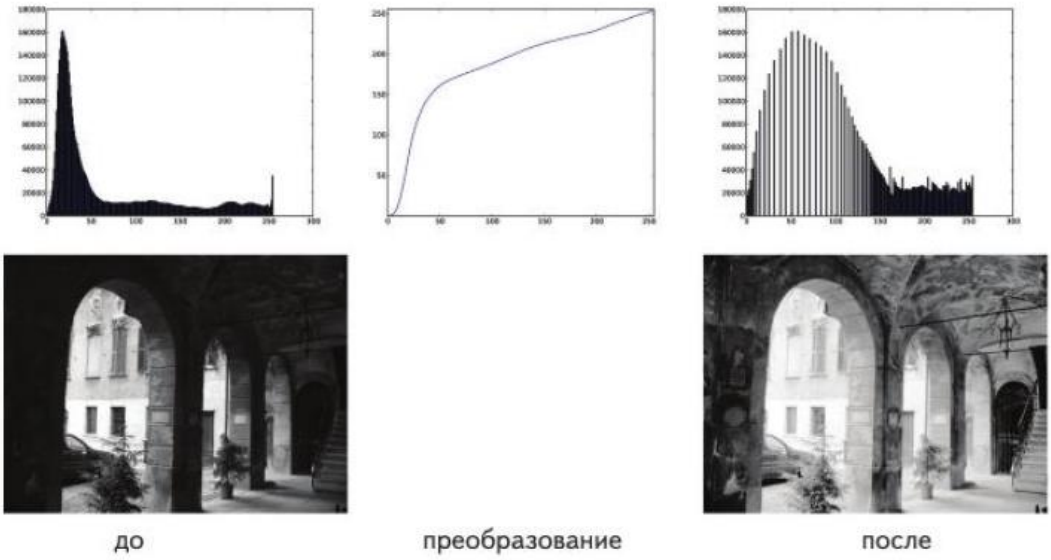

Рис. 1.7. Пример выравнивания гистограммы. Слева – исходное изображение и его гистограмма. В центре – функция преобразования уровня яркости. Справа – изображение и гистограмма после выравнивания

Задание 1. Создайте миниатюру выбранного изображения. Скопируйте и вставьте область изображения, измените размер и выполните поворот изображения.

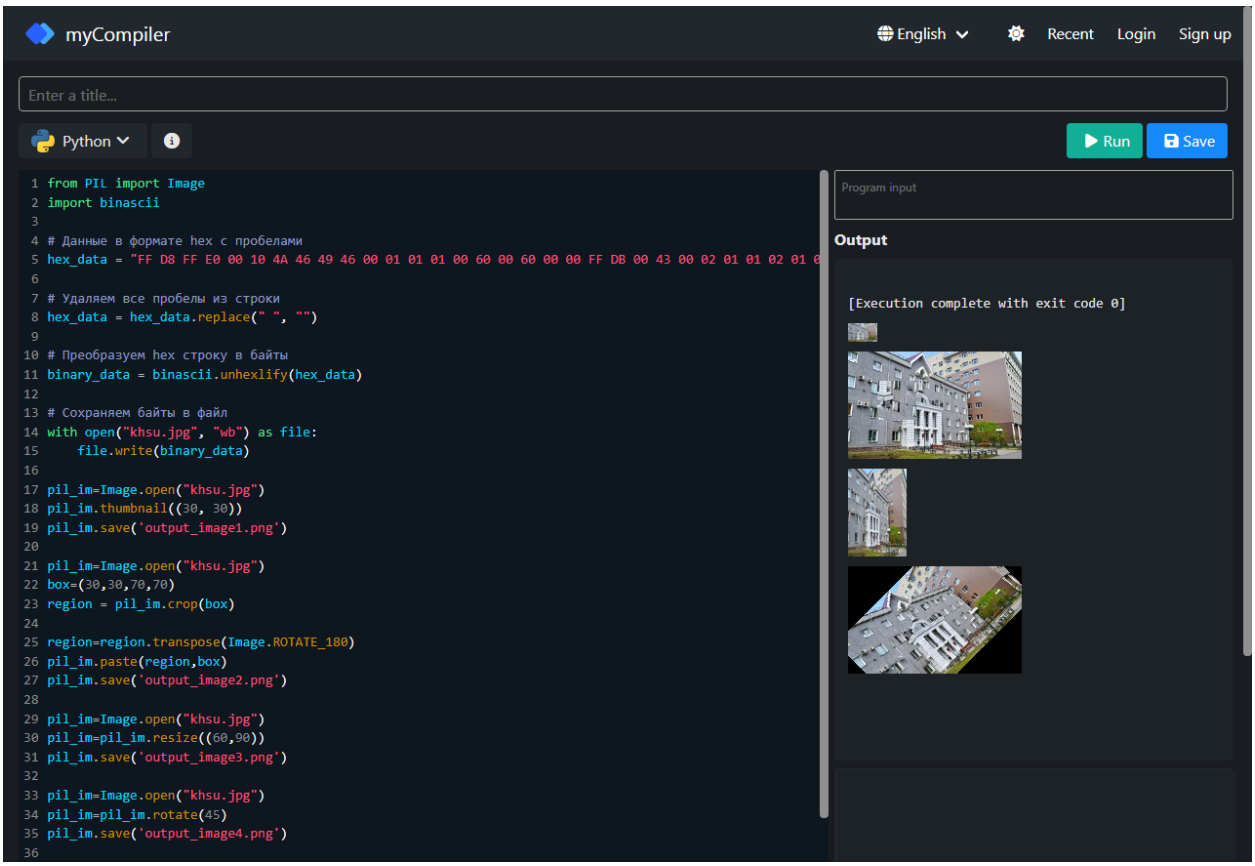

Рисунок 1. Пример преобразования изображения

Задание 2. Постройте изолинии изображения.

myCompiler English Recent Login Sign up

Enter a title...

Python Run Save

```python
from PIL import Image
import numpy as np
import binascii
import matplotlib.pyplot as plt

# Данные в формате hex с пробелами
hex_data = "FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60 00 60 00 00 FF DB 00 43 00 02 01 01 02 01 01
# Удаляем все пробелы из строки
hex_data = hex_data.replace(" ", "")

# Преобразуем hex строку в байты
binary_data = binascii.unhexlify(hex_data)

# Сохраняем байты в файл
with open("khsu.jpg", "wb") as file:
    file.write(binary_data)

# Загружаем изображение
image = Image.open("khsu.jpg").convert("L")  # Конвертируем в градации серого

# Преобразуем изображение в массив данных
data = np.array(image)

# Создаем график изолиний
plt.figure(figsize=(10, 10))
plt.contour(data, levels=20, cmap='jet')  # Устанавливаем количество уровней изолиний
plt.colorbar()  # Добавляем цветовую шкалу
plt.title("Изолинии изображения")
plt.xlabel("X")
plt.ylabel("Y")
plt.gca().invert_yaxis()
plt.show()
```

Program input

Output

[Execution complete with exit code 0]

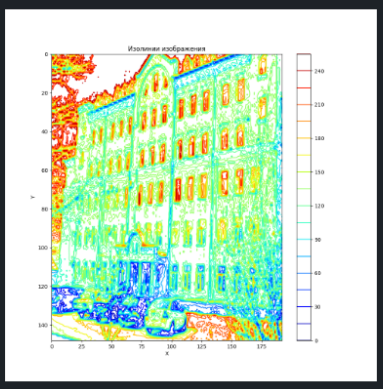

Рисунок 2. Пример построения изолиний изображения

Задание 3. Постройте гистограмму изображения.

myCompiler English Recent Login Sign up

Enter a title...

Python Run Save

```python
from PIL import Image
from pylab import *
import binascii
import matplotlib.pyplot as plt

# Данные в формате hex с пробелами
hex_data = "FF D8 FF E0 00 10 4A 46 49 46 00 01 01 01 00 60 00 60 00 00 FF DB 00 43 00 02 01 01 02 01 0
# Удаляем все пробелы из строки
hex_data = hex_data.replace(" ", "")

# Преобразуем hex строку в байты
binary_data = binascii.unhexlify(hex_data)

# Сохраняем байты в файл
with open("khsu.jpg", "wb") as file:
    file.write(binary_data)

# Загружаем изображение и преобразуем в градации серого
im = array(Image.open("khsu.jpg").convert("L"))

# Отображаем изображение
figure()
imshow(im, cmap='gray')
axis('off')  # Отключаем оси
show()

# Вычисляем и отображаем гистограмму
plt.figure()
plt.hist(im.ravel(), bins=256, range=[0, 256], color='black', alpha=0.7)
plt.title('Histogram')
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.xlim([0, 256])
plt.grid()
plt.show()
```

Program input

Output

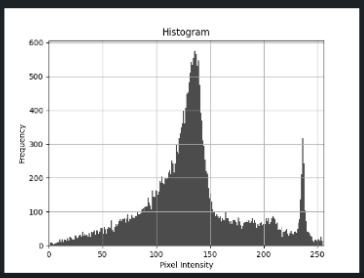

Рисунок 3. Пример построения гистограммы изображения

Задание 4*. Осуществите выравнивание гистограммы. Поясните для чего выполняется данная операция.