Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им Н.Э. Баумана)
Факультет ИУ – «Информатика и системы управления»
Кафедра ИУ3 – «Информационные системы и телекоммуникации»

Отчёт по семинару №6
по дисциплине «Цифровая обработка изображений»

Группа ИУ3-22М
Вариант 18

Студент группы ИУ3-22М _____ М.А. Шевченко

Преподаватель _____ А.Н. Алфимцев

Москва, 2021

1 Аффинные преобразования

1.1 Постановка задачи

В соответствии с вариантом индивидуального задания (таблица 1.1) для данной фигуры в декартовой системе координат рассчитать её координаты после всех преобразований.

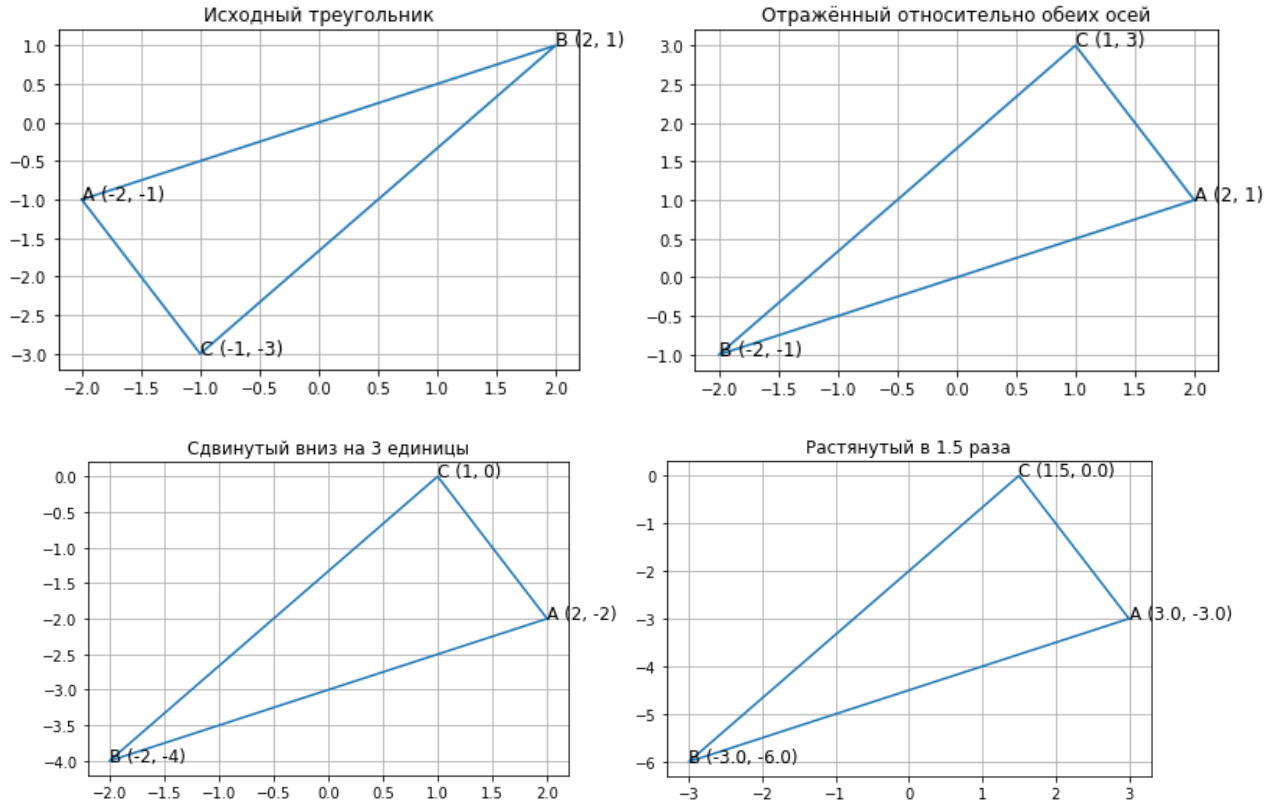
Таблица 1.1 – Вариант задания для аффинных преобразований.

№ варианта	Треугольник	Преобразования
18	5. A(-2, -1); B(2, 1); C(-1, -3)	Преобразования Б: • отражение относительно обеих осей; • сдвиг вниз на 3 единицы; • растяжение в 1,5 раза по обеим осям; • поворот на 90 радиан против часовой стрелки относительно точки B.

1.2 Преобразования

В процессе реализации задания была написана программа на языке Python(3.x). Её листинг приведён в приложении А.

Процесс преобразования треугольника представлен на рисунке 1.



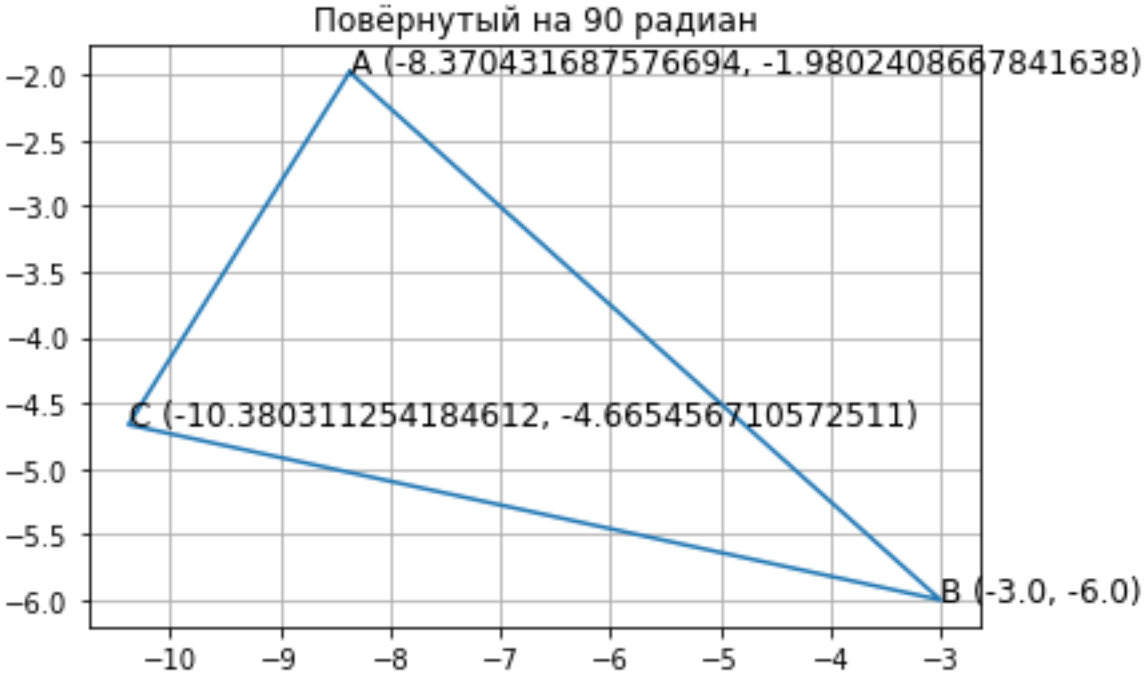


Рисунок 1 – Процесс преобразования треугольника.

2 LookUp Table

2.1 Постановка задачи

В соответствии с вариантом индивидуального задания (таблица 2.1) окрасить изображение в оттенках серого в цвет.

Таблица 2.1 – Вариант задания для LookUp Table.

№ варианта	Таблица	Изображение
18	5. Синяя [0, 0, 1]	Б. [[[0, 0, 0], [12, 170, 30], [170, 30, 12]], [[255, 255, 255], [30, 12, 170], [50, 50, 50]], [[17, 56, 14], [190, 0, 240], [84, 16, 250]]]

2.2 Реализация

В процессе реализации задания была написана программа на языке Python(3.x). Её листинг приведён в приложении Б.

Процесс преобразования изображения представлен на рисунке 2.

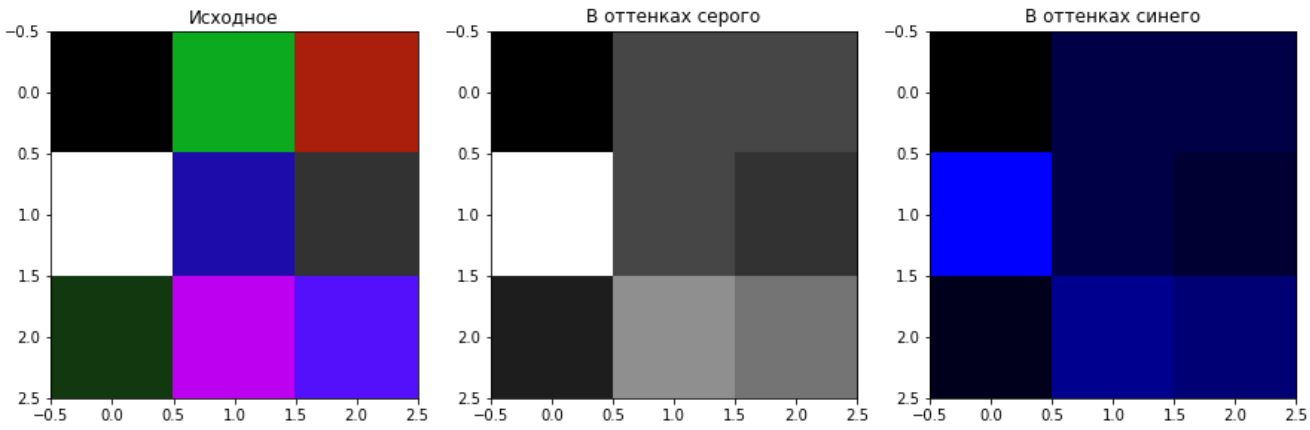


Рисунок 2 – Процесс преобразования цветного цифрового изображения в изображение в оттенках синего.

3 Перевод в цветовую схему

3.1 Постановка задачи

В соответствии с вариантом индивидуального задания (таблица 3.1) перевести заданное изображение из RGB формата в YCbCr формат.

Таблица 3.1 – Вариант индивидуального задания по переводу в цветовую схему.

№ варианта	Цветовая схема	Изображение
18	5. YCbCr – $\begin{bmatrix} 65{,}481 & 128{,}553 & 24{,}966 \\ -37{,}797 & -74{,}203 & 112{,}0 \\ 112{,}0 & -93{,}786 & -18{,}214 \end{bmatrix}$	Б. [[[0, 0, 0], [12, 170, 30], [170, 30, 12]], [[255, 255, 255], [30, 12, 170], [50, 50, 50]], [[17, 56, 14], [190, 0, 240], [84, 16, 250]]]

3.2 Реализация

В процессе реализации задания была написана программа на языке Python(3.x). Её листинг приведён в приложении В.

Результатом перевода получилось следующее изображение:

[[[0.0, 0.0, 0.0], [23388.762, -9708.074, -15146.04], [15287.952, -7307.58, 16007.852]],

[[55845.0, 0.0, 0.0],	[7751.286, 17015.654, -861.8119999999994],	[10949.999999999998, 0.0, -1.1368683772161603e-13]],
[[8661.669, -3229.9170000000004, -3603.0119999999997],	[18433.23, 19698.57, 16908.64],	[13798.752, 23637.804, 3353.924]]]

4 Матричные фильтры обработки изображений

4.1 Постановка задачи

В соответствии с вариантом индивидуального задания (таблица 4.1) произвести свёртку для данного изображения, используя заданный матричный фильтр.

Таблица 4.1 – Вариант индивидуального задания по матричному фильтру обработки изображения.

№ варианта	Фильтр	Изображение
18	5. Наращивание	Б. [[[0, 0, 0], [12, 170, 30], [170, 30, 12]], [[255, 255, 255], [30, 12, 170], [50, 50, 50]], [[17, 56, 14], [190, 0, 240], [84, 16, 250]]]

Перед свёрткой изображение должно пройти предварительную обработку. Она заключается в дополнении изображения пикселями по краю. На обработанном изображении краевые пиксели должны быть равны пикселям исходного изображения.

4.2 Реализация

В процессе реализации задания была написана программа на языке Python(3.x). Её листинг приведён в приложении Г.

Процесс преобразования изображения представлен на рисунке 4.

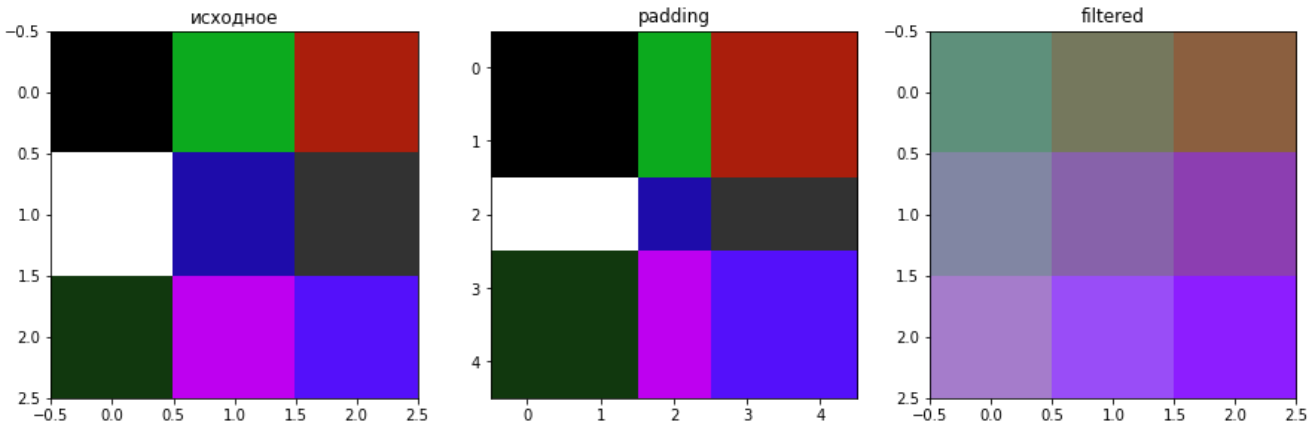


Рисунок 4 – Процесс применения матричного фильтра.

5 Кодирование кривых при помощи трёхразрядного кода

5.1 Постановка задачи

В соответствии с вариантом индивидуального задания (таблица 5.1) закодировать данную функцию при помощи трёхразрядного кода.

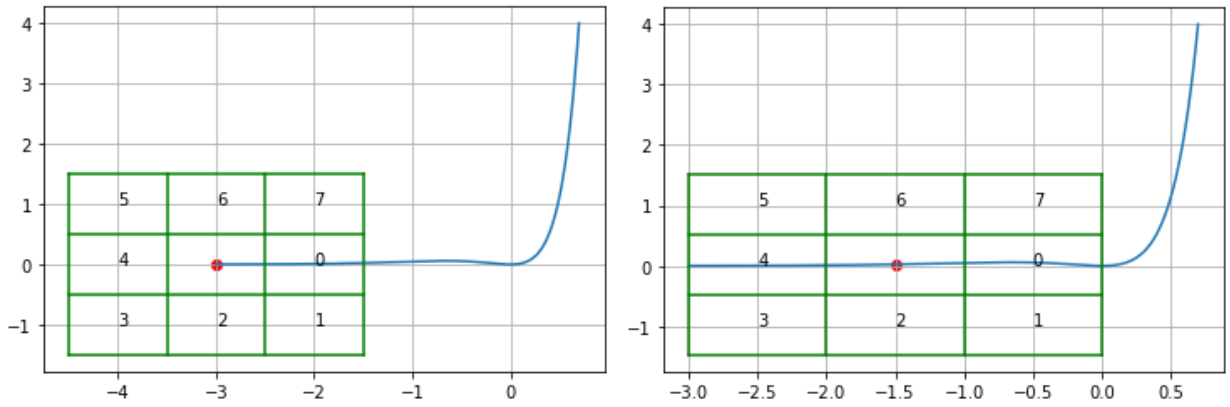
Таблица 5.1 – Вариант индивидуального задания по кодированию кривой при помощи трёхразрядного кода.

№ варианта	Функция	Отрезок
18	$f(x) = x^2 * \exp(3x)$	[-3; 0,7]

5.2 Реализация

В процессе реализации задания была написана программа на языке Python(3.x). Её листинг приведён в приложении Д.

Процесс преобразования изображения представлен на рисунке 5.



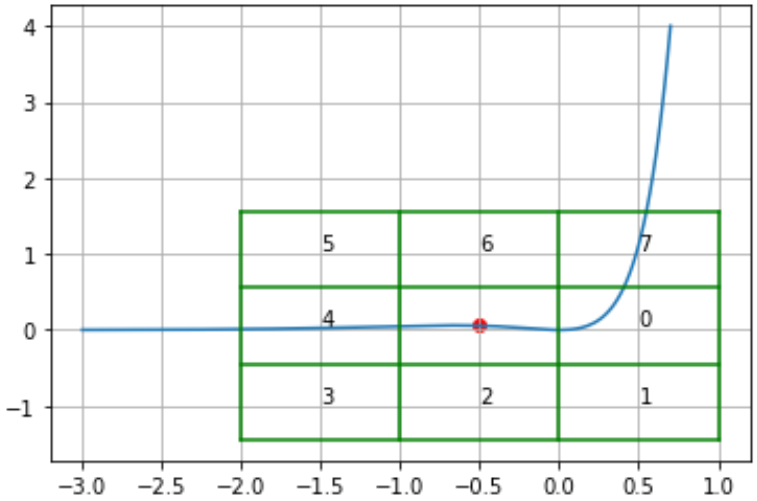


Рисунок 5 – Процесс кодирования кривых при помощи трёхразрядного кода.

Результат кодирования кривой – (0, 0, 7).

Приложение А

Листинг программы аффинных преобразований

```
import matplotlib.pyplot as plt
from numpy import cos, sin


class Triangle:
    """Класс треугольник."""
    def __init__(self, xA, yA, xB, yB, xC, yC):
        self.xA = xA
        self.yA = yA
        self.xB = xB
        self.yB = yB
        self.xC = xC
        self.yC = yC


    def draw(self, title):
        """Рисует треугольник."""
        ax = plt.subplot()
        ax.plot([self.xA, self.xB, self.xC, self.xA], [self.yA, self.yB, self.yC,
self.yA])
        ax.text(self.xA, self.yA, 'A ({}, {})'.format(self.xA, self.yA),
fontsize='large')
        ax.text(self.xB, self.yB, 'B ({}, {})'.format(self.xB, self.yB),
fontsize='large')
        ax.text(self.xC, self.yC, 'C ({}, {})'.format(self.xC, self.yC),
fontsize='large')
        ax.set_title(title)
        ax.grid()
        plt.show()


    def flip(self, ax):
        """
        Отражает треугольник.

        mod == 'x': относительно X;
        mod == 'y': относительно Y;
        mod == 'xy': относительно обеих осей;
        mod == 'yx': относительно обеих осей.
        """
        if ax == 'x':
            self.xA = -self.xA
            self.xB = -self.xB
            self.xC = -self.xC
        elif ax == 'y':
            self.yA = -self.yA
            self.yB = -self.yB
            self.yC = -self.yC
        elif ax in ('xy', 'yx'):
            self.xA = -self.xA
            self.xB = -self.xB
            self.xC = -self.xC
            self.yA = -self.yA
            self.yB = -self.yB
            self.yC = -self.yC


    def shift(self, direction, size):
        """
```

```
        Сдвигает треугольник.

        direction - направление.
        size - размер сдвига.
        """
        if direction == 'наверх':
            self.yA += size
            self.yB += size
            self.yC += size
        elif direction == 'вниз':
            self.yA -= size
            self.yB -= size
            self.yC -= size
        elif direction == 'вправо':
            self.xA += size
            self.xB += size
            self.xC += size
        elif direction == 'влево':
            self.xA -= size
            self.xB -= size
            self.xC -= size


    def transform(self, a=1, b=1):
        """
        Трансформирует треугольник.

        a - коэффициент растяжения/сжатия по оси X.
        b - коэффициент растяжения/сжатия по оси Y.
        """
        self.xA *= a
        self.xB *= a
        self.xC *= a

        self.yA *= b
        self.yB *= b
        self.yC *= b


    def rotation(self, x0, y0, alpha):
        """Поворачивает треугольник относительно точки (x0, y0) на угол alpha."""
        tmp_xA = self.xA
        tmp_xB = self.xB
        tmp_xC = self.xC
        tmp_yA = self.yA
        tmp_yB = self.yB
        tmp_yC = self.yC

        self.xA = (tmp_xA - x0) * cos(alpha) - (tmp_yA - y0) * sin(alpha) + x0
        self.xB = (tmp_xB - x0) * cos(alpha) - (tmp_yB - y0) * sin(alpha) + x0
        self.xC = (tmp_xC - x0) * cos(alpha) - (tmp_yC - y0) * sin(alpha) + x0

        self.yA = (tmp_xA - x0) * sin(alpha) + (tmp_yA - y0) * cos(alpha) + y0
        self.yB = (tmp_xB - x0) * sin(alpha) + (tmp_yB - y0) * cos(alpha) + y0
        self.yC = (tmp_xC - x0) * sin(alpha) + (tmp_yC - y0) * cos(alpha) + y0

    def rotation(self, apex, alpha):
        """Поворачивает треугольник относительно вершины на угол alpha."""
        tmp_xA = self.xA
        tmp_xB = self.xB
        tmp_xC = self.xC
        tmp_yA = self.yA
        tmp_yB = self.yB
        tmp_yC = self.yC

        if apex == 'A':
```

```
            x0 = self.xA
            y0 = self.yA
        elif apex == 'B':
            x0 = self.xB
            y0 = self.yB
        elif apex == 'C':
            x0 = self.xC
            y0 = self.yC

        self.xA = (tmp_xA - x0) * cos(alpha) - (tmp_yA - y0) * sin(alpha) + x0
        self.xB = (tmp_xB - x0) * cos(alpha) - (tmp_yB - y0) * sin(alpha) + x0
        self.xC = (tmp_xC - x0) * cos(alpha) - (tmp_yC - y0) * sin(alpha) + x0

        self.yA = (tmp_xA - x0) * sin(alpha) + (tmp_yA - y0) * cos(alpha) + y0
        self.yB = (tmp_xB - x0) * sin(alpha) + (tmp_yB - y0) * cos(alpha) + y0
        self.yC = (tmp_xC - x0) * sin(alpha) + (tmp_yC - y0) * cos(alpha) + y0

triangle = Triangle(-2, -1, 2, 1, -1, -3)
triangle.draw('Исходный треугольник')

triangle.flip('xy')
triangle.draw('Отражённый относительно обеих осей')

triangle.shift('вниз', 3)
triangle.draw('Сдвинутый вниз на 3 единицы')

triangle.transform(a=1.5, b=1.5)
triangle.draw('Растянутый в 1.5 раза')

triangle.rotation('B', 90)
triangle.draw('Повёрнутый на 90 радиан')
```

Приложение Б

Листинг программы LookUp Table

```
import matplotlib.pyplot as plt
import numpy as np

BLUE_TABLE = [0, 0, 1]


def rgb2gray(image):
    """."""
    gray_image = []
    for line in image:
        gray_line = []
        for pixel in line:
            gray = int((pixel[0] + pixel[1] + pixel[2]) / 3)
            gray_pixel = [gray, gray, gray]
            gray_line.append(gray_pixel)
        gray_image.append(gray_line)
    return gray_image


def gray2lut(gray_image, table):
    """."""
    lut_image = []
    for line in gray_image:
        lut_line = []
        for pixel in line:
            lut_pixel = [a * b for a, b in zip(pixel, table)]
            lut_line.append(lut_pixel)
        lut_image.append(lut_line)
    return lut_image


image = [[[  0,   0,   0], [12, 170,  30], [170, 30,  12]],
         [[255, 255, 255], [30,  12, 170], [ 50, 50,  50]],
         [[ 17,  56,  14], [190,  0, 240], [ 84, 16, 250]]]

grayscale_image = rgb2gray(image)
lut_image = gray2lut(grayscale_image, BLUE_TABLE)

fig, axes = plt.subplots(1, 3, figsize=(15, 5))
axes[0].imshow(image)
axes[0].set_title('Исходное')
axes[1].imshow(grayscale_image)
axes[1].set_title('В оттенках серого')
axes[2].imshow(lut_image)
axes[2].set_title('В оттенках синего')
plt.show()
```

Приложение В

Листинг программы перевода в цветовую схему

```
import matplotlib.pyplot as plt
import numpy as np


image = np.array([[[  0,   0,   0], [12, 170,  30], [170, 30,  12]],
                  [[255, 255, 255], [30,  12, 170], [ 50, 50,  50]],
                  [[ 17,  56,  14], [190,  0, 240], [ 84, 16, 250]]])

YCbCr = np.array([[ 65.481, 128.553,   24.966],
                  [-37.797, -74.203,  112.0],
                  [112.0,   -93.786, -18.214]])


def rgb2ycbcr(image):
    ycbcr_image = []
    for line in image:
        ycbcr_line = []
        for pixel in line:
            ycbcr_pixel = list(YCbCr.dot(pixel))
            ycbcr_line.append(ycbcr_pixel)
        ycbcr_image.append(ycbcr_line)
    return ycbcr_image


ycbcr_image = rgb2ycbcr(image)
print(ycbcr_image)
```

Приложение Г

Листинг программы матричных фильтров обработки изображений

```
import matplotlib.pyplot as plt
import cv2

IMAGE = np.array([[[  0,   0,   0], [12, 170,  30], [170, 30,  12]],
                  [[255, 255, 255], [30,  12, 170], [ 50, 50,  50]],
                  [[ 17,  56,  14], [190,  0, 240], [ 84, 16, 250]]],
dtype='int16')
KERNEL = np.ones((3, 3), np.float32) / 6


def padding(image):
    new_image = []
    for line in image:
        new_line = [line[0]] + list(line) + [line[-1]]
        new_image.append(new_line)
    new_image = [new_image[0]] + list(new_image) + [new_image[-1]]
    return np.array(new_image, dtype='int16')


pad_image = padding(IMAGE)
filtered = cv2.filter2D(IMAGE, -1, KERNEL, borderType=cv2.BORDER_REPLICATE)

fig, axes = plt.subplots(1, 3, figsize=(15,5))
axes[0].imshow(image)
axes[0].set_title('исходное')
axes[1].imshow(pad_image)
axes[1].set_title('padding')
axes[2].imshow(filtered)
axes[2].set_title('filtered')
plt.show()
```

Приложение Д

Листинг программы кодирования кривой при помощи трёхразрядного кода

```
import matplotlib.pyplot as plt
import numpy as np

START_X = -3
STOP_X = 0.7
N = 201

xlist = np.linspace(START_X, STOP_X, N)
ylist = pow(xlist, 2) * np.exp(3 * xlist)

f = lambda x: pow(x, 2) * np.exp(3 * x)


def draw_square(x, y):
    ax = plt.subplot()
    ax.grid()
    ax.plot(xlist, ylist)
    # горизонтальные линии
    ax.plot([x - 1.5, x + 1.5], [y + 1.5, y + 1.5], color='green')
    ax.plot([x - 1.5, x + 1.5], [y + 0.5, y + 0.5], color='green')
    ax.plot([x - 1.5, x + 1.5], [y - 0.5, y - 0.5], color='green')
    ax.plot([x - 1.5, x + 1.5], [y - 1.5, y - 1.5], color='green')
    # вертикальные линии
    ax.plot([x - 1.5, x - 1.5], [y + 1.5, y - 1.5], color='green')
    ax.plot([x - 0.5, x - 0.5], [y + 1.5, y - 1.5], color='green')
    ax.plot([x + 0.5, x + 0.5], [y + 1.5, y - 1.5], color='green')
    ax.plot([x + 1.5, x + 1.5], [y + 1.5, y - 1.5], color='green')
    # точка
    ax.scatter(x, y, color='red')
    # цифры
    ax.text(x + 1, y, '0')
    ax.text(x + 1, y - 1, '1')
    ax.text(x, y - 1, '2')
    ax.text(x - 1, y - 1, '3')
    ax.text(x - 1, y, '4')
    ax.text(x - 1, y + 1, '5')
    ax.text(x, y + 1, '6')
    ax.text(x + 1, y + 1, '7')
    plt.show()

start_x = START_X
start_y = f(start_x)
draw_square(start_x, start_y)
draw_square(start_x + 1.5, f(start_x + 1.5))
draw_square(start_x + 2.5, f(start_x + 2.5))
```