Основы компьютерной графики

Гайсин Эдуард, 20.Б07-мм

Ссылка на исходные файлы

Задачи

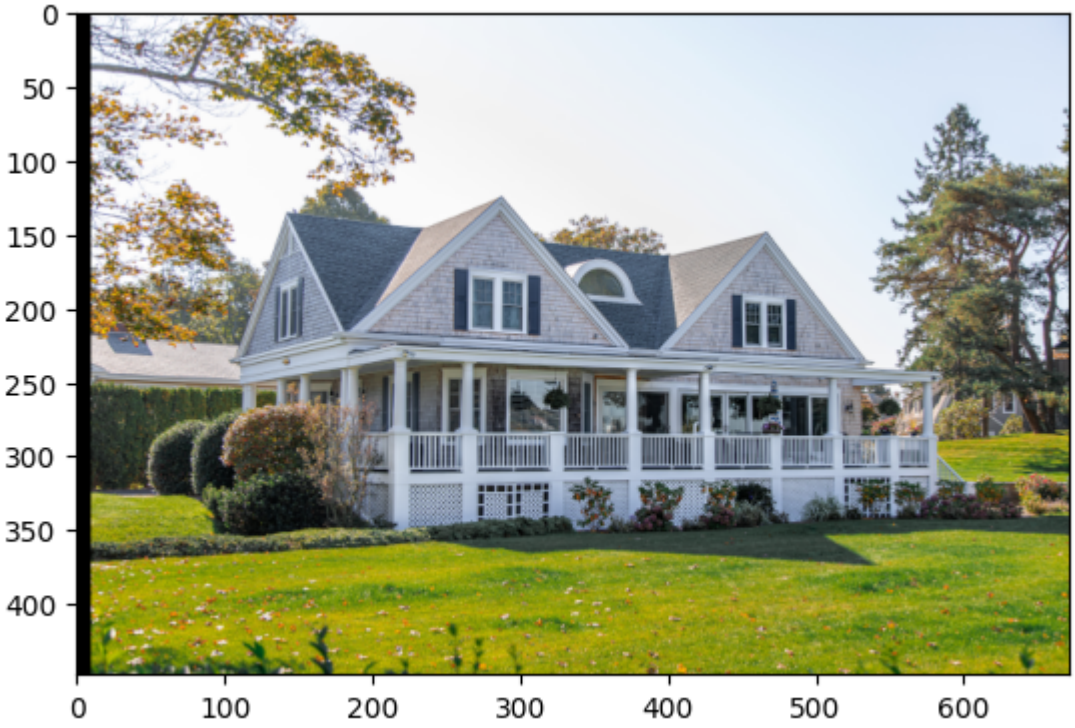
1. Сместить изображение на 10 пикселей вправо
2. Повернуть изображение на 45 градусов
3. Повернуть изображение на 30 градусов вокруг заданной точки
4. Изменить яркость изображения
5. Изменить контрастность изображения
6. Сделать бинаризацию изображения
7. Найти контур изображения используя Лапласиан
8. Перевести изображение в HSV
9. Перевести изображение в grayscale
10. Отразить изображение по правой границе
11. Отразить изображение по нижней границе

In [6]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

img = cv2.cvtColor(cv2.imread("img/img.png", cv2.IMREAD_COLOR), cv2.COLOR_BGR2RGB)
# img = cv2.resize(img, (img.shape[1] // 10, img.shape[0] // 10), interpolation=cv2.I
plt.imshow(img)
plt.show()
```

Сместить изображение на 10 пикселей вправо

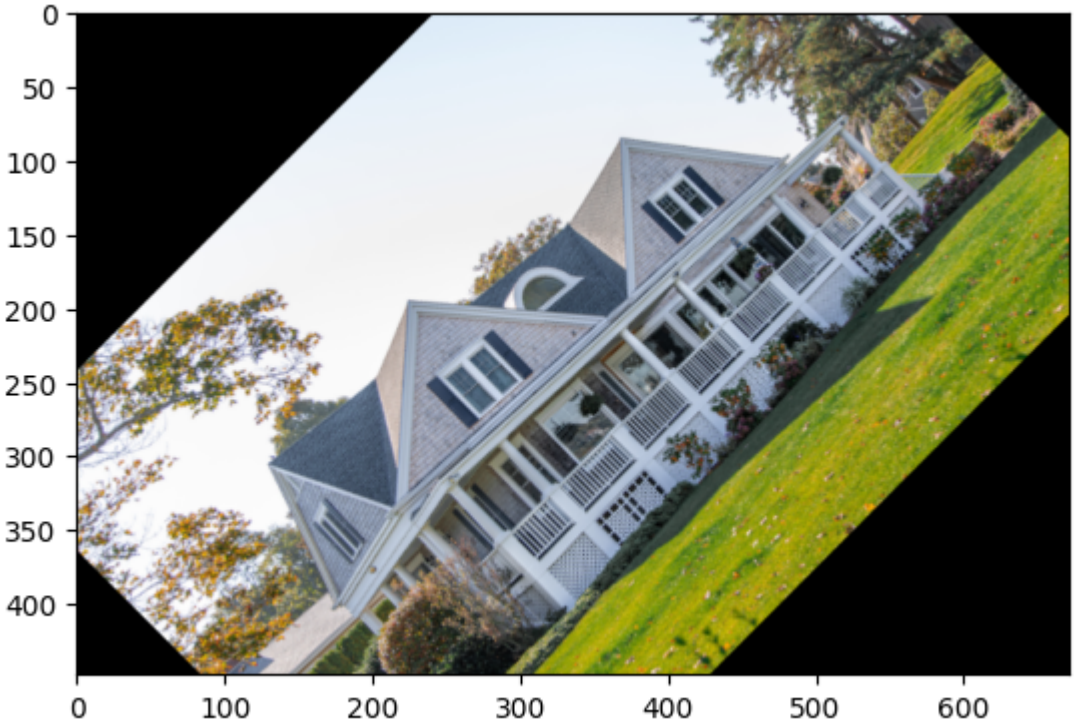
In [2]:
```
def shift(img, shift_value):
    translation = np.asarray([shift_value, 0]).reshape(2, 1)
    matrix = np.concatenate((np.identity(2), translation), axis=1)
```

```
    result = cv2.warpAffine(img, matrix, dsize=(img.shape[1], img.shape[0]))
    plt.imshow(result)

shift(img, 10)
```



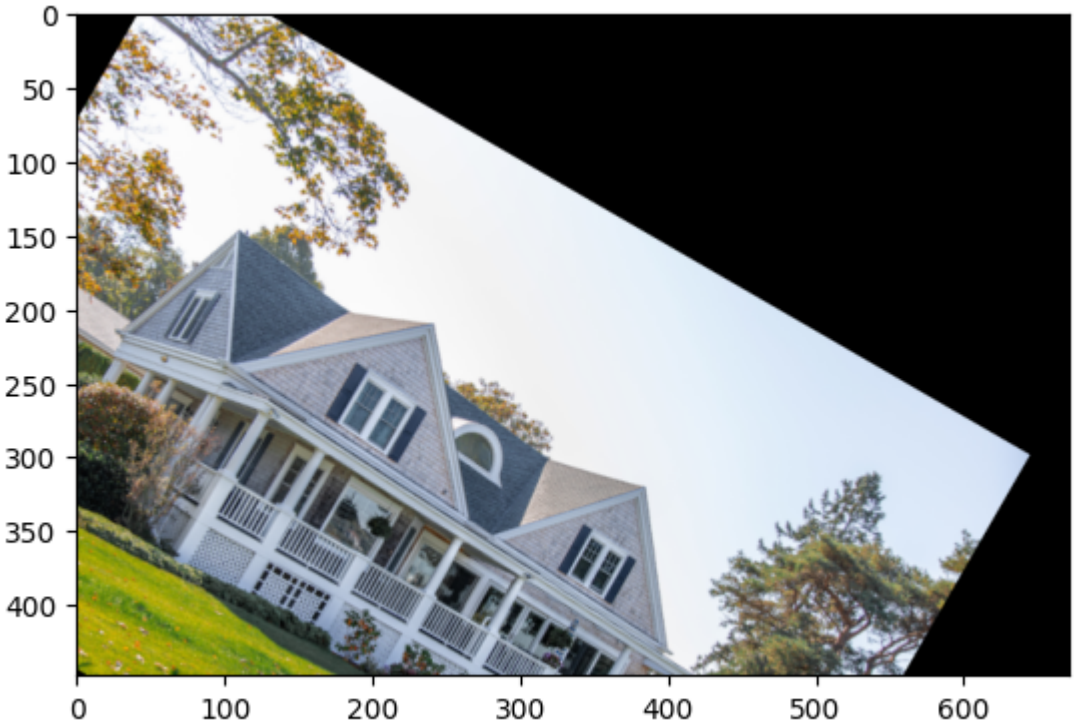
Повернуть изображение на 45 градусов

In [3]:
```
def rotate45(img):
    angle = 45
    point = (img.shape[1] // 2, img.shape[0] // 2)
    matrix = cv2.getRotationMatrix2D(point, angle, scale=1)
    result = cv2.warpAffine(img, matrix, dsize=(img.shape[1], img.shape[0]))
    plt.imshow(result)
rotate45(img=img)
```



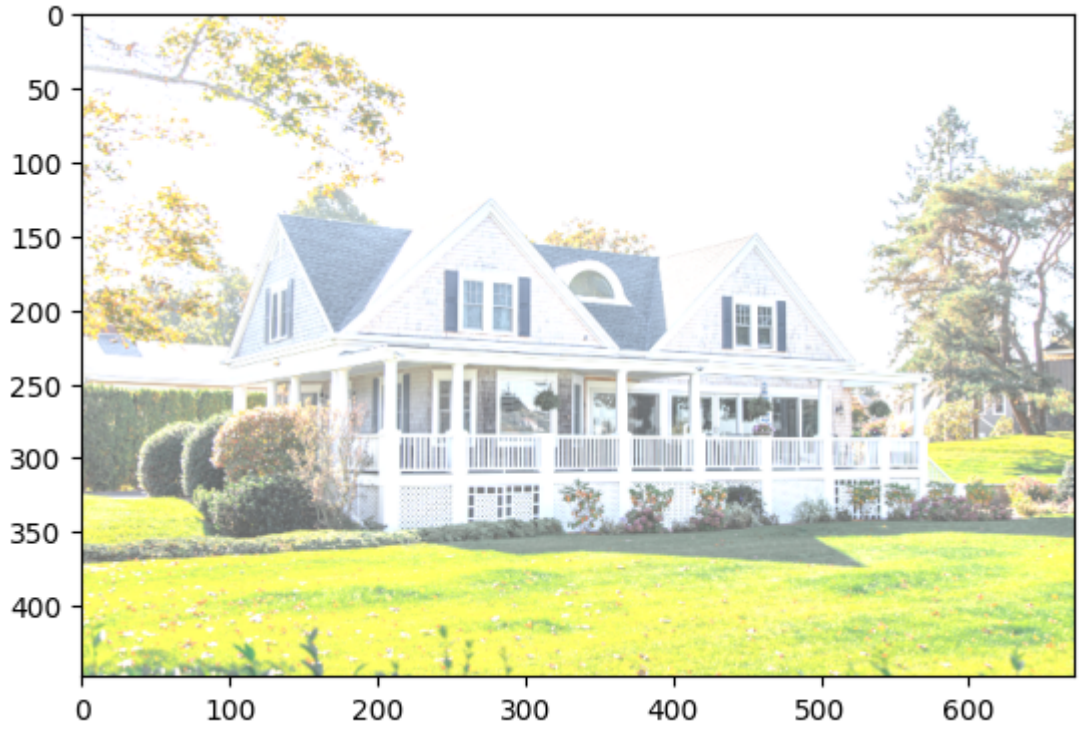
Повернуть изображение на 30 градусов вокруг заданной точки

In [15]:
```
def rotate30(img, point):
    angle = -30
    matrix = cv2.getRotationMatrix2D(point, angle, scale=1)
    result = cv2.warpAffine(img, matrix, dsize=(img.shape[1], img.shape[0]))
    plt.imshow(result)
rotate30(img, (100, 100))
```



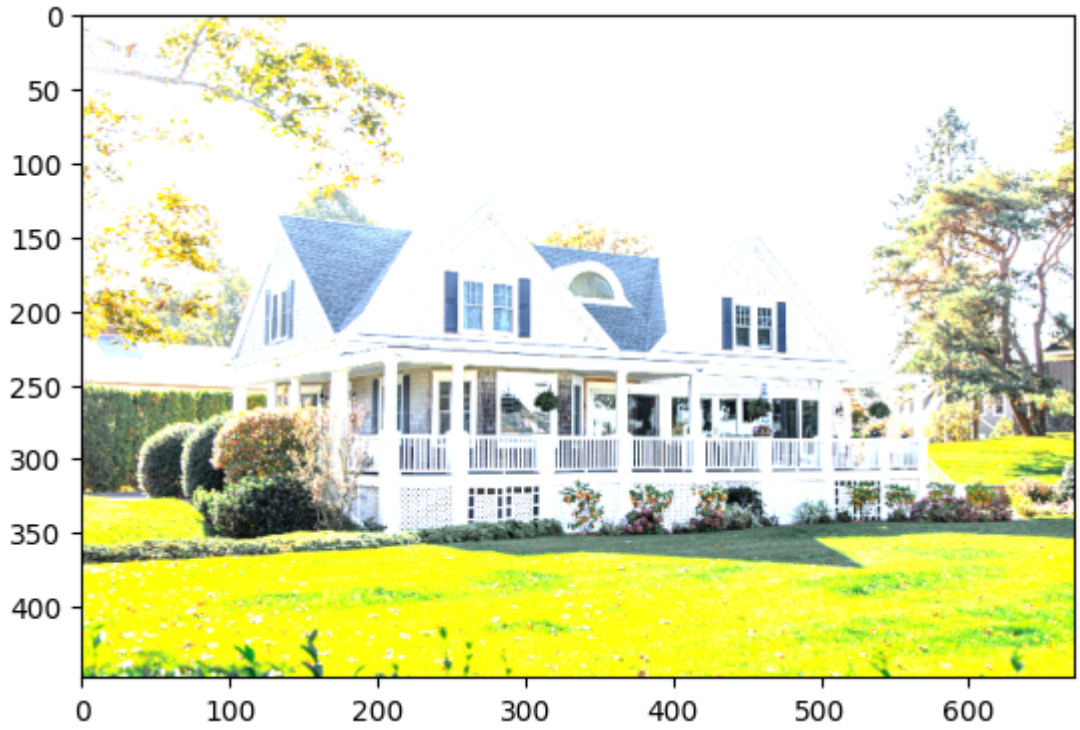
Изменить яркость изображения

In [5]:
```
def brightness(img):
    result = cv2.convertScaleAbs(img, alpha=1, beta=100)
    plt.imshow(result)
brightness(img)
```



Изменить контрастность изображения

In [6]:
```
def contrast(img):
    result = cv2.convertScaleAbs(img, alpha=2, beta=1)
    plt.imshow(result)
contrast(img)
```

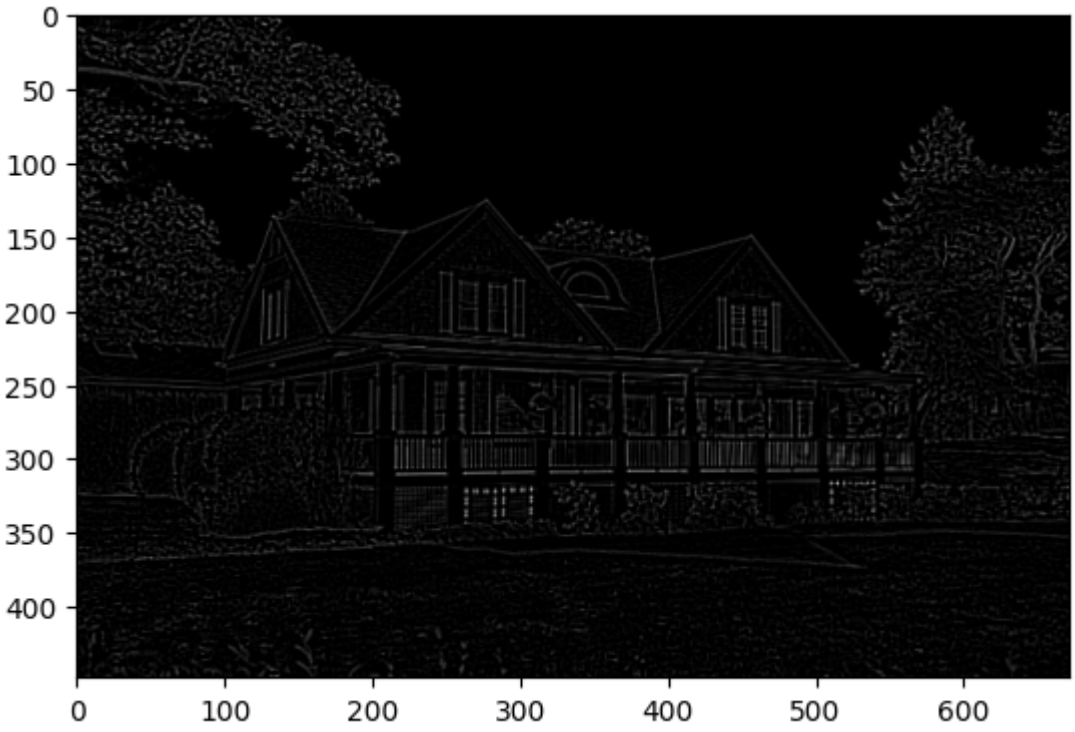


Сделать бинаризацию изображения

In [7]:
```
def binar(img):
    # для создания черно-белого изображения преобразуем img в grayscale
    img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    _, binary = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
    plt.imshow(binary, cmap="gray")
binar(img)
```

Найти контур изображения используя Лапласиан

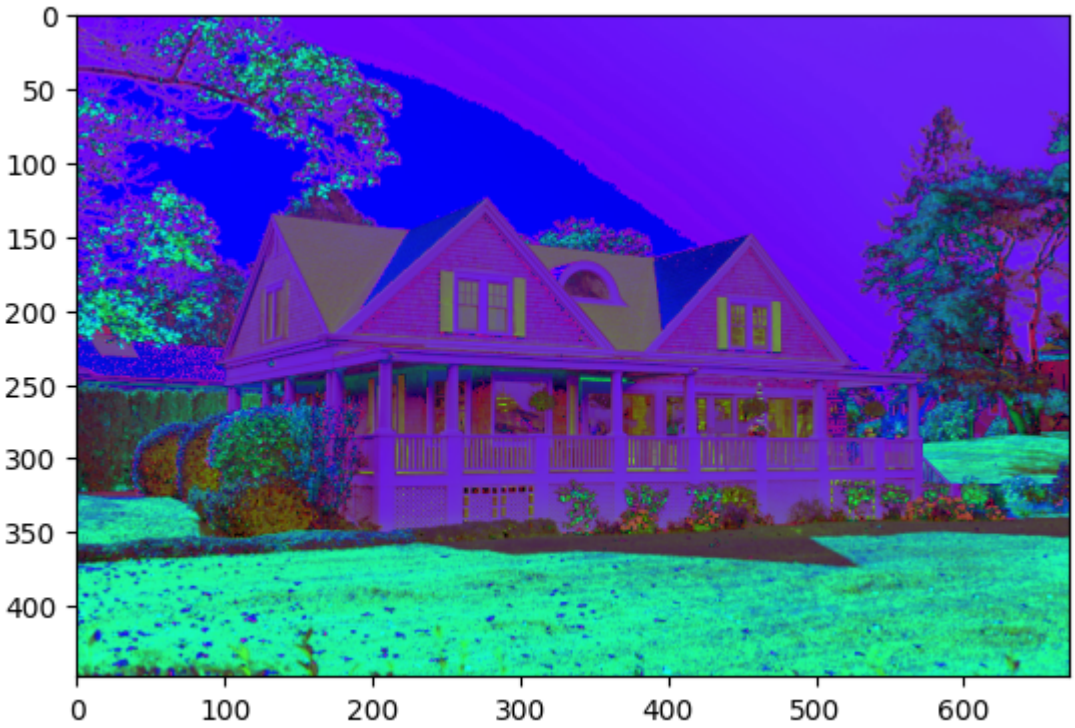
In [8]:
```
def sobel(img):
    img = cv2.blur(img, ksize=(3, 3))
    img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    result = cv2.Laplacian(img, ddepth=10, ksize=3)
    plt.imshow(result, cmap="gray")
sobel(img)
```



Перевести изображение в HSV

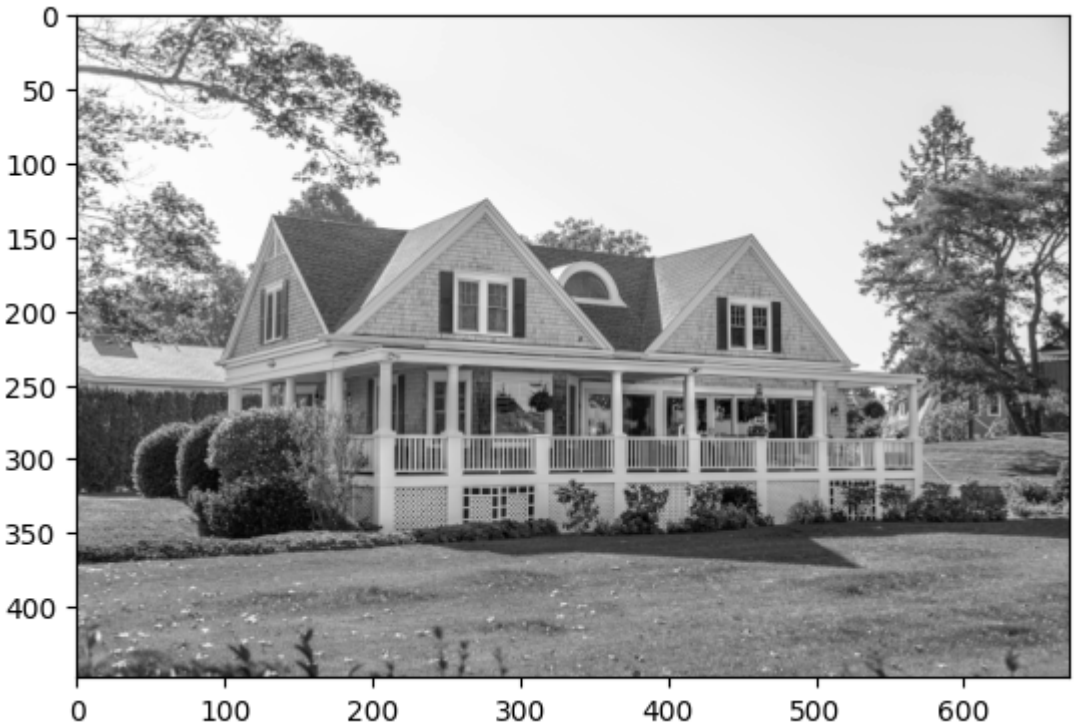
In [9]:
```
def hsv(img):
    result = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
```

```
    plt.imshow(result)
hsv(img)
```



Перевести изображение в grayscale

In [10]:
```
def grayscale(img):
    result = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    plt.imshow(result, cmap="gray")
grayscale(img)
```



Отразить изображение по правой границе

In [11]:
```
def reflect_right(img):
    result = cv2.flip(img, 1)
```

```
    plt.imshow(result)
reflect_right(img)
```

Отразить изображение по нижней границе

In [16]:
```
def reflect_down(img):
    result = cv2.flip(img, 0)
    plt.imshow(result)
reflect_down(img)
```