Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Отчет по лабораторной работе №1

«Гистограммы, профили и проекции»

По дисциплине «Техническое зрение»

Выполнил: Курчавый В. В.,

студент группы R3338

Преподаватель: Шаветов С. В.,

канд. техн. Наук, доцент ФСУ и Р

Санкт-Петербург, 2022

**Цель работы**

Освоение основных яркостных и геометрических характеристик изображений и их использование для анализа изображений.

**Теоретическое обоснование применимых методов**

**Ход выполнения работы**

1. **Гистограммы**
   1. **Исходные данные**

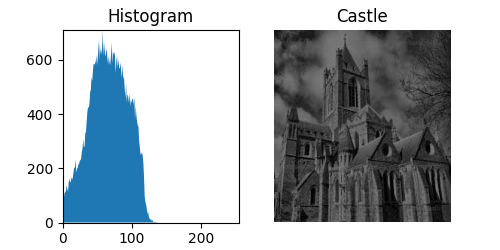


Рисунок 1. Исходное изображение.

Контрастность изображения:

Изображение темное с небольшим контрастом, из-за этого гистограмма смещена влево.

Листинг 1. Считывание изображения и построение гистограммы.

img = cv.imread(PATH\_TO\_PHOTO, cv.IMREAD\_GRAYSCALE)

hist = cv.calcHist([image], [**0**], None, [**256**], [**0**, **256**])

Листинг 2 Совместное изображение гистограммы и изображения

fig, (h, i) = plt.subplots(**1**, **2**, figsize=(**5**,**2.5**))

h.set\_title('Histogram')

h.set\_xlim([**0**, **255**])

h.set\_ylim([**0**, int(np.max(hist))])

h.fill\_between(list(range(**256**)), **0**, list(map(int, hist)))

i.set\_title('Castle')

i.imshow(img, cmap='gray', vmin=**0**, vmax=**255**, aspect='auto')

i.axis('off')

plt.show()

* 1. **Линейное преобразование**

Увеличим каждое значение интенсивности на 60, чтобы изображение стало более светлым.

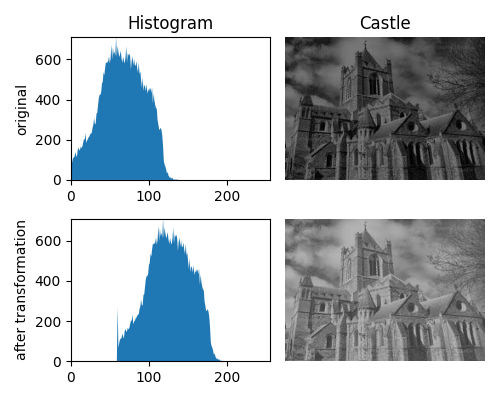


Рисунок 2. Влияние линейного сдвига.

Как и ожидалось изображение стало более светлым, но при этом не осталось очень темных тонов.

Листинг 3. Функция линейного сдвига.

**def** **linear\_transform**(img, shift=**0**):

**return** img + shift

* 1. **Растяжение динамического диапазона**

Применим растяжение динамического диапазона с параметром с различными .

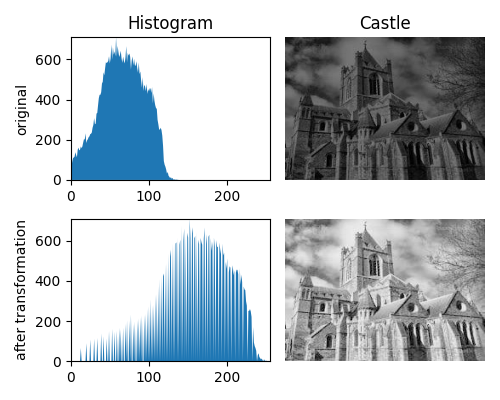
**

Рисунок 3. Влияние динамического растяжение с =0.6.

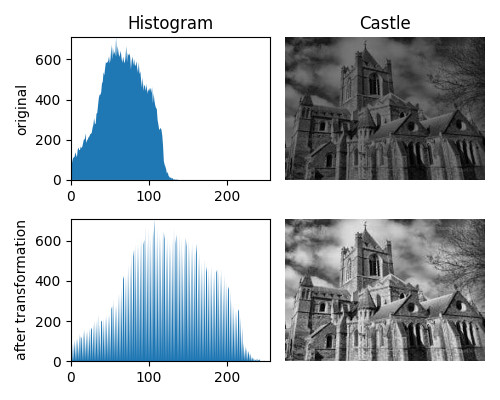
**

Рисунок 4. Влияние динамического растяжение с =1.

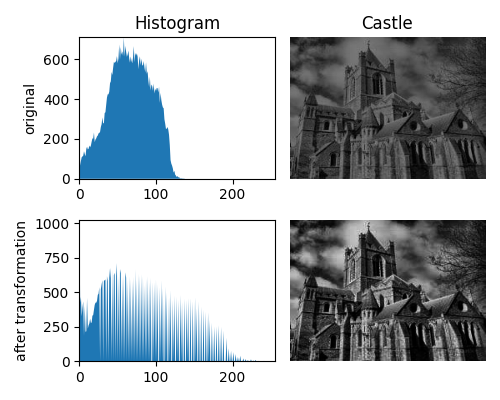


Рисунок 5. Влияние динамического растяжение с =1.9.

Можно заметить, что при увеличении картинка становится темнее, так как значения интенсивностей становятся меньше. При этом изображения становятся контрастнее.

Листинг 4. Функция динамического растяжения.

**def** **stretching\_tranform**(img: ndarray, a: float = **1**) -> ndarray:

i\_max = np.max(img)

i\_min = np.min(img)

**return** (**255**\*(np.power((img - i\_min)/(i\_max - i\_min), a))).astype(np.uint8)

* 1. **Равномерное преобразование**

Применим равномерное преобразование для исходного изображения:

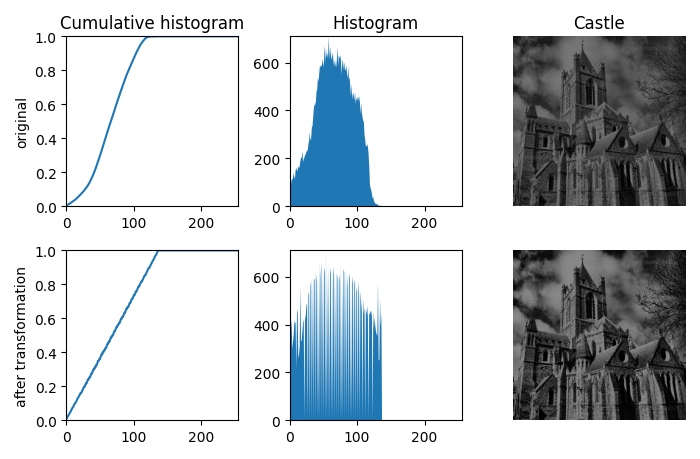


Рисунок 6. Влияние равномерного преобразования.

Можно заметить, что кумулятивная гистограмма стала линейно расти, а количество различных значений интенсивностей стало менее разбросанным. Контраст изображения не поменялся.

Листинг 5. Функция расчета нормированной кумулятивной гистограммы.

**def** **cum\_histogram**(hist, num\_rows, num\_column):

**return** np.cumsum(hist) / (num\_rows \* num\_column)

Листинг 6 Функция расчета равномерного проебразования.

**def** **uniform\_transform**(img: ndarray, cum\_hist: ndarray) -> ndarray:

i\_max, i\_min = np.max(img), np.min(img)

new\_img = ndarray(img.shape)

**for** x **in** range(img.shape[**0**]):

**for** y **in** range(img.shape[**1**]):

new\_img[x][y] = (i\_max - i\_min) \* cum\_hist[img[x][y]] + i\_min

**return** new\_img.astype(np.uint8)

* 1. **Экспоненциальное преобразование**

Вычислим новое значение интенсивностей по формуле:

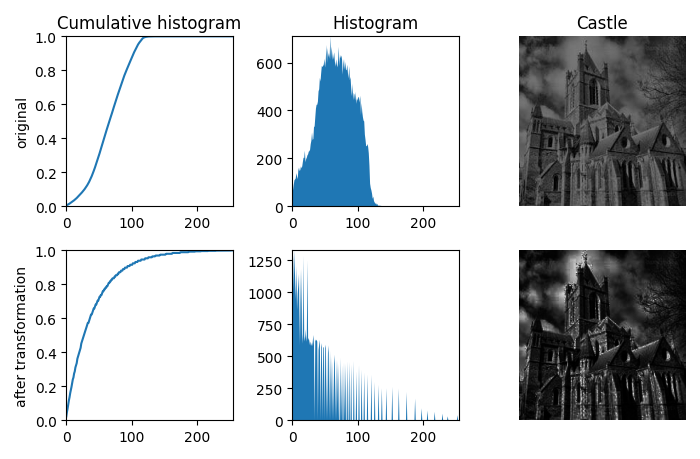


Рисунок 7. Влияние экспоненциального преобразования при .

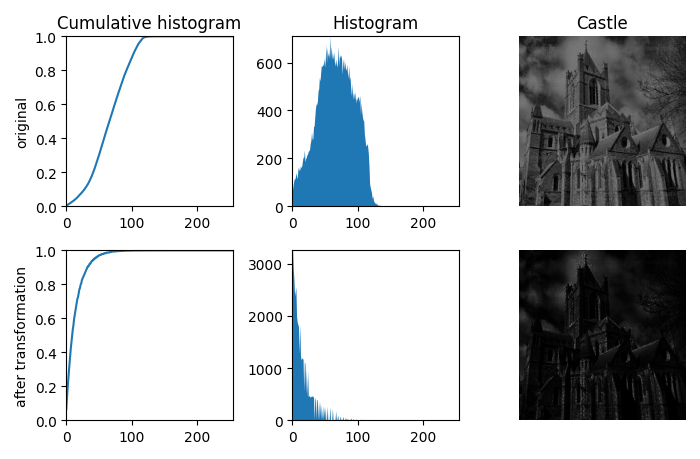


Рисунок 8. Влияние экспоненциального преобразования при .

Можно заметить, что, чем меньше α, тем с более меньшей скоростью растет кумулятивная гистограмма. Кумулятивная гистограмма растет по экспоненте. Значения на обычной гистограмме сосредоточены близко к нулю, поэтому фотографии получаются в темных тонах.

Листинг 7. Функция расчета экспоненциального преобразования.

**def** **exponential\_transform**(img: ndarray, cum\_hist: ndarray, a) -> ndarray:

i\_min = np.min(img)

new\_img = ndarray(img.shape)

**for** x **in** range(img.shape[**0**]):

**for** y **in** range(img.shape[**1**]):

new\_img[x][y] = i\_min - **1**/a \* np.log(**1** - cum\_hist[img[x][y]])

**return** new\_img.astype(np.uint8)

* 1. **Преобразование по закону Рэлея**

Вычислим новое значение интенсивностей по формуле:

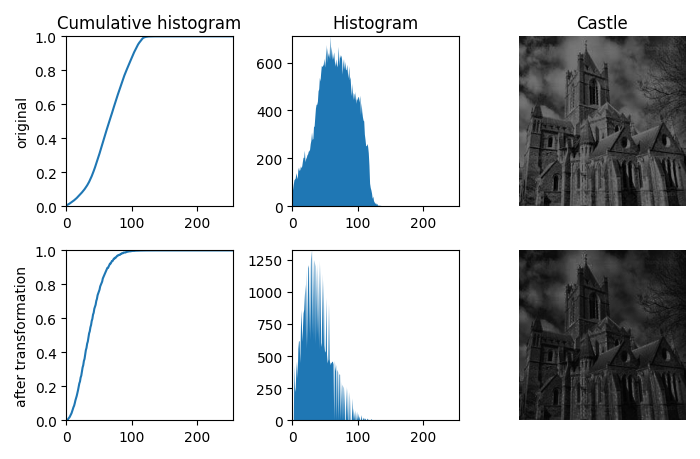


Рисунок 9. Влияние преобразования по закону Рэлея при .

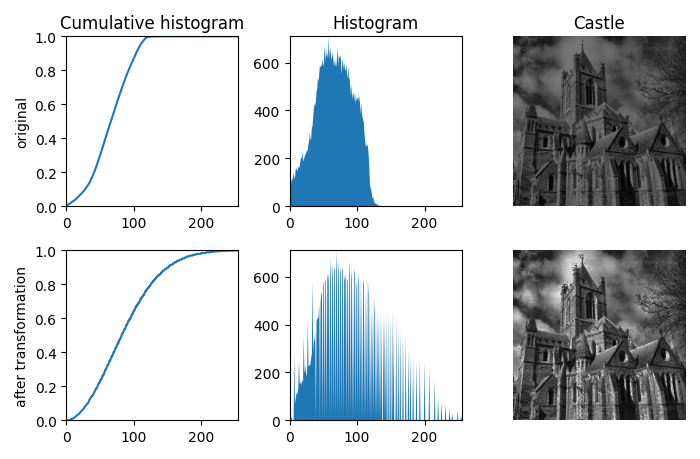


Рисунок 10. Влияние преобразования по закону Рэлея при .

Можно заметить, что при увеличении увеличивается контраст изображения, но только до определенного значения , после которого, начинает теряться информация об изображении.

Листинг 8. Функция расчета преобразования по закону Рэлея.

**def** **rayleigh\_low\_transform**(img: ndarray, cum\_hist, a: float = **100**):

i\_min = np.min(img)

new\_img = ndarray(img.shape)

**for** x **in** range(img.shape[**0**]):

**for** y **in** range(img.shape[**1**]):

new\_img[x][y] = i\_min + np.power(**2**\*np.power(a, **2**) \* np.log(**1** / (**1** - cum\_hist[img[x][y]])), **0.5**)

**return** new\_img.astype(np.uint8)

* 1. **Преобразование по закону**

Вычислим новое значение интенсивностей по формуле:

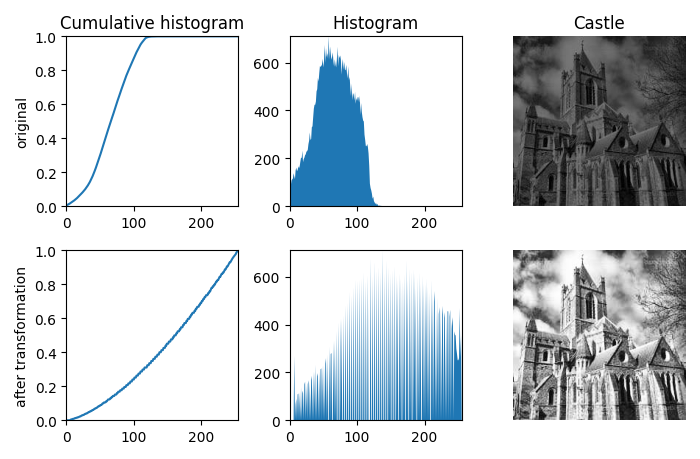


Рисунок 11. Влияние преобразования по закону .

Можно заметить, что после преобразования изображение стало сильно контрастным с преобладанием светлых тонов.

Листинг 9 Функция преобразования по закону .

**def** **two\_thirds\_low\_transform**(img: ndarray, cum\_hist: ndarray):

new\_img = ndarray(img.shape)

**for** x **in** range(img.shape[**0**]):

**for** y **in** range(img.shape[**1**]):

**print**(cum\_hist[img[x][y]])

new\_img[x][y] = **255**\*np.power((cum\_hist[img[x][y]]), **2**/**3**)

**return** new\_img.astype(np.uint8)

* 1. **Преобразование по гиперболическому закону**

Вычислим новое значение интенсивностей по формуле:

часто выбирают равным *.*

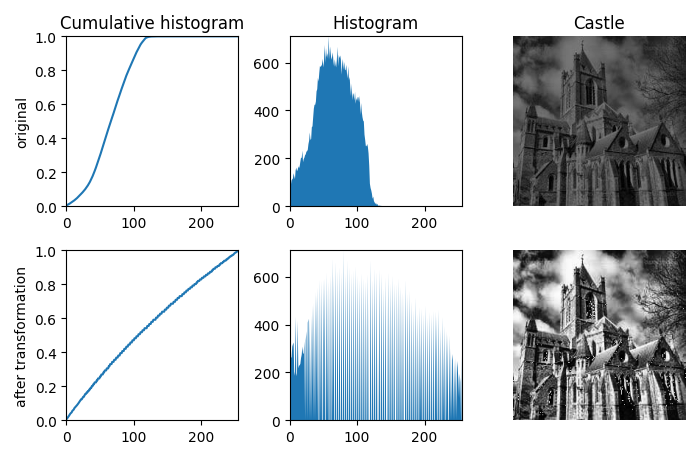
**

Рисунок 12. Влияние преобразования по гиперболическому закону.

Можно заметить, что после преобразования изображение стало сильно контрастным.

Листинг 10 Функция преобразования по гиперболическому закону.

**def** **hyperbolic\_transform**(img: ndarray, cum\_hist: ndarray, a=None):

**if** a **is** None:

a = np.min(img)

**if** a == **0** **or** a == **1**:

a = **2**

new\_img = ndarray(img.shape)

**for** x **in** range(img.shape[**0**]):

**for** y **in** range(img.shape[**1**]):

**print**(cum\_hist[img[x][y]])

new\_img[x][y] = **255**\*np.power(a, cum\_hist[img[x][y]])

**return** new\_img.astype(np.uint8)

* 1. **Преобразование с помощью встроенных в OpenCV функций**

Воспользуемся функцией [*equalizeHist*](https://docs.opencv.org/3.4/d6/dc7/group__imgproc__hist.html#ga7e54091f0c937d49bf84152a16f76d6e)*()* из OpenCV

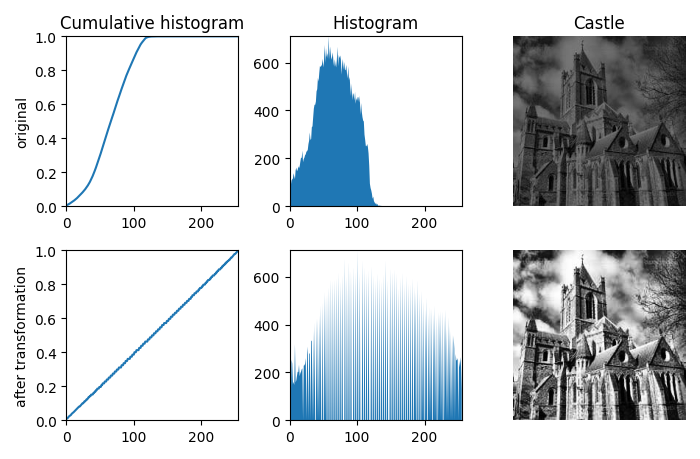


Рисунок 13. Влияние преобразования с помощью функции [equalizeHist](https://docs.opencv.org/3.4/d6/dc7/group__imgproc__hist.html#ga7e54091f0c937d49bf84152a16f76d6e)().

Преобразованные изображение контрастное. При этом кумулятивная гистограмма линейная. Реализация функции аналогична алгоритму равномерного преобразования, за исключением того, что , а , чтобы максимально увеличить контраст изображения.

Воспользуемся функцией *createCLAHE()* из OpenCV с параметрами *clipLimit = 6* и *tileGridSize = (10, 11),* и преобразуем изображение.

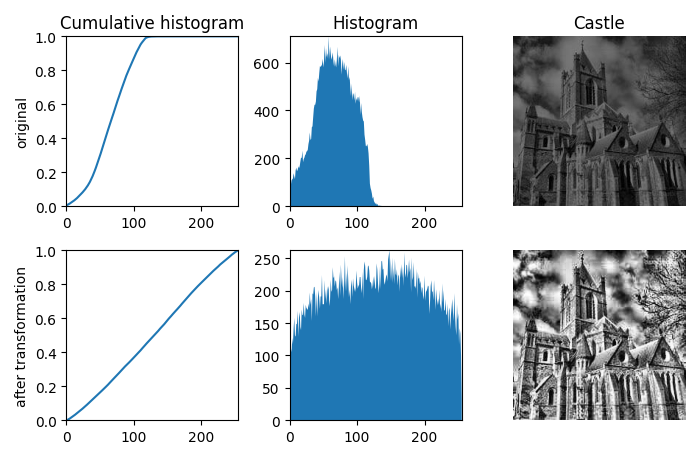


Рисунок 14. Влияние алгоритма CLAHE.

Можно заметить, что изображение стало контрастным и при этом распределение интенсивностей почти равномерно.

Листинг 11 Использование встроенных функций.

# CLAHE

algorithm\_CLAHE = cv.createCLAHE(**10**, (**10**, **11**))

image\_2 = algorithm\_CLAHE.apply(image)

# equalize Hist

image\_3 = cv.equalizeHist(imagE)

* 1. **Lookup-Table**

Воспользуемся методом соляризации. Рассчитаем lookup-table по формуле:

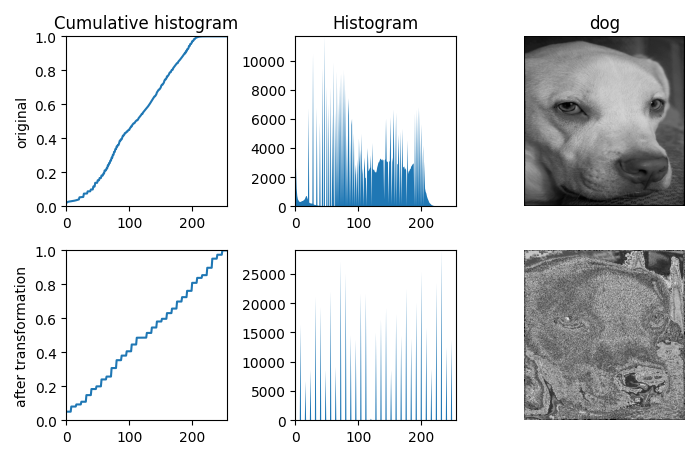


Рисунок 15. Влияние соляризации.

С помощью LUT можно задавать произвольное преобразование интенсивностей.

Листинг 12. Функция создания LUT для соляризации.

**def** **create\_sabattier\_lut**():

lut = np.arange(**256**, dtype=np.uint8)

lut = **4**\*lut\*(**255**-lut)

lut = np.where(lut > **0**, lut, **0**)

lut = np.clip(lut, **0**, **255**)

**return** lut

Листинг 13. Применение соляризации.

image\_2 = cv.LUT(image, create\_sabattier\_lut())

1. **Проекции**

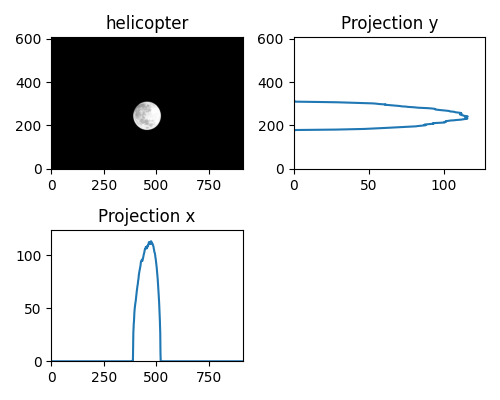


Рисунок 16. Проекции изображения.

По проекциям можно определить границы объекта на монотонном фоне. К примеру, на данном изображении, границы по оси x будут примерно 380 и 525 а по оси y и 185 и 315.

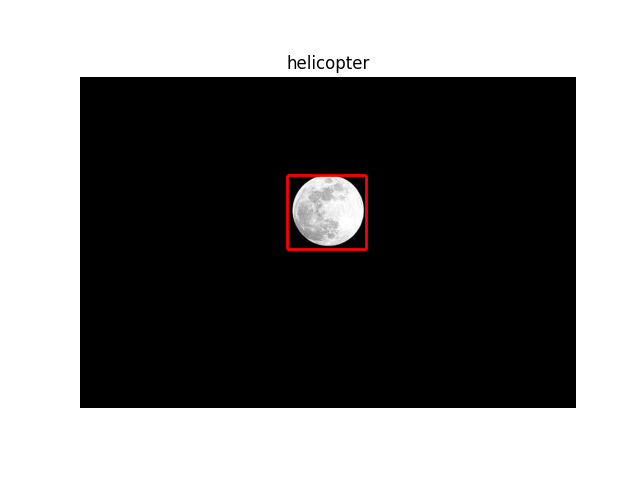


Рисунок 17. Выделение границ.