Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
Национальный исследовательский университет ИТМО

Факультет Систем Управления и Робототехники
Дисциплина: Техническое зрение

Отчет
По лабораторной работе № 1
По теме: «Гистограммы, профили и проекции»

Выполнил: Новичков Д.Е.
Группа: R3235
Преподаватель: Шаветов С.В.

Санкт-Петербург
2024 г.

Цель работы

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

Используемые термины

Гистограмма — это распределение частоты встречаемости пикселей одинаковой яркости на изображении.

Яркость — это среднее значение интенсивности сигнала.

Контраст — это интервал значений между минимальной и максимальной яркостями изображения.

Профиль вдоль линии — это функция интенсивности изображения, распределенного вдоль данной линии (прорезки).

Проекция на ось — это сумма интенсивностей пикселей изображения, взятая в направлении перпендикулярном данной оси.

Гистограмма изображения

Для 8-битного полутонового изображения гистограмма яркости представляет собой одномерный целочисленный массив Hist из 256 элементов [0 . . . 255]. Элементом гистограммы Hist[i] является сумма пикселей изображения с яркостью i. По визуальному отображению гистограммы можно оценить необходимость изменения яркости и контрастности изображения, оценить площадь, занимаемую светлыми и темными элементами, определить местоположение на плоскости изображения отдельных объектов, соответствующих некоторым диапазонам яркости. Для цветного RGB-изображения необходимо построить три гистограммы по каждому цвету.

Листинг 1.1. Построение гистограмм изображения.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

```
hist_size = 256
hist_range = (0, 256)
color = ('b', 'g', 'r')

image = cv2.imread("MIT_mini_Cheetah.jpg")

print(f'Размер изображения: {image.shape[:2]}, имеющего {image.shape[2]} канала')
print(f'MAX: {max(image.reshape(-1))}\nMIN: {min(image.reshape(-1))}')
```

```
Размер изображения: (1536, 2048), имеющего 3 канала

MAX: 97

MIN: 0
```

Выведем исходное изображение рассчитанную базовую гистограмму:

```
def show_image_hist(image):
    fig, axis = plt.subplots(1, 2, figsize=(15, 5))

    axis[0].imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    axis[0].axis('off')
    axis[1].set_xlim([0,255])

    image_BGR = cv2.split(image)
    for i, col in enumerate(color):
        hist = cv2.calcHist(image_BGR, [i], None, [hist_size], hist_range)
        axis[1].plot(hist, color = col)

    plt.show()
```

```
show_image_hist(image)
```

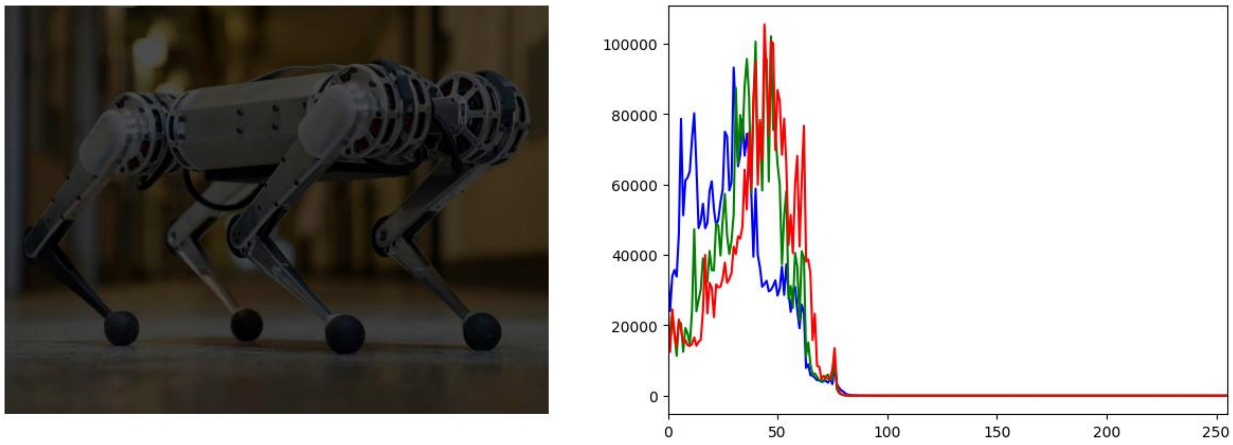


Рис.1. Исходное изображение и его гистограмма.

Как видно из гистограммы, присутствует много тёмных пиков.

Листинг 1.2. Арифметические операции.

Простейшими способами выравнивания гистограммы являются арифметические операции с изображениями. Например, в случае если большинство значений гистограммы находятся слева, то изображение является темным. Для увеличения детализации темных областей можно сдвинуть гистограмму правее, в более светлую область.

```
bias = 50

show_image_hist((image + bias) % 255)
```

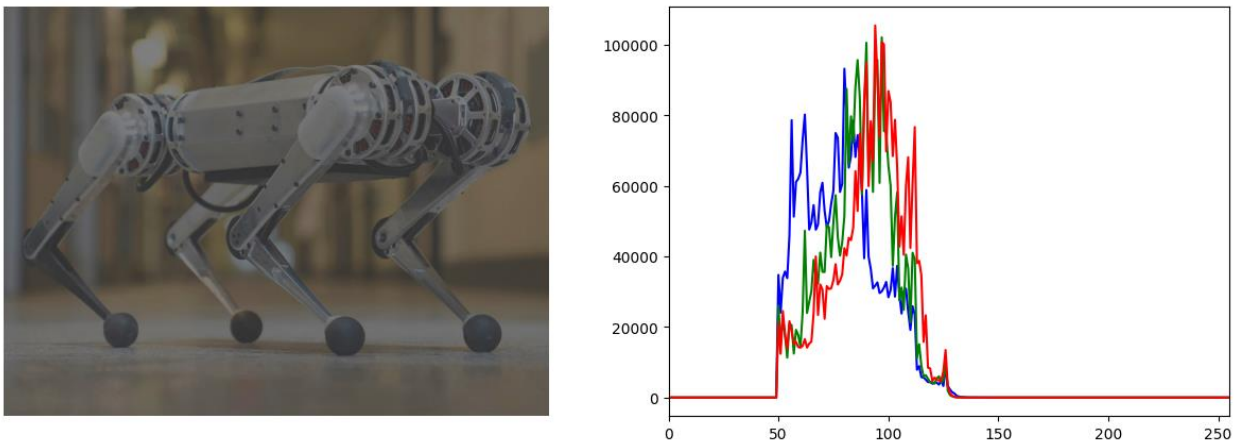


Рис.2. Изображение и гистограмма после линейного сдвига.

После сдвига гистограммы вправо на 50 градаций, стали более различимы темные объекты робота.

Листинг 1.3. Растяжение динамического диапазона

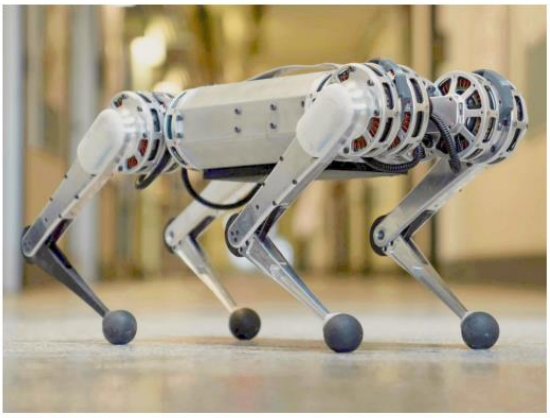
Если интенсивности пикселей областей интереса находятся в узком динамическом диапазоне, то можно растянуть этот диапазон. Подобные преобразования выполняются согласно следующему выражению:

$Inew$ = 256 * (($I - Imin$)/($Imax - Imin$))^α (1.1)

где I и $Inew$ — массивы значений интенсивностей исходного и нового изображений соответственно; $Imin$ и $Imax$ — минимальное и максимальное значения интенсивностей исходного изображения соответственно; α — коэффициент нелинейности. Данное выражение является нелинейным из-за коэффициента α. В случае, если α = 1, применение формулы (1.1) к исходному изображению не даст желаемого эффекта, поскольку гистограммы цветовых компонент изображения занимают весь возможный диапазон. Нелинейные преобразования проводятся для каждой цветовой составляющей.

```
image_new = image.astype(np.float32) / 255

alfa = 0.5


image_BGR = cv2.split(image_new)

image_new_BGR = list()


for layer in image_BGR:

    Imin = layer.min()
```

```
    Imax = layer.max()
    image_new = np.clip((((layer - Imin) / (Imax - Imin))**alfa), 0, 1)

    image_new_BGR.append(image_new)

image_new = cv2.merge(image_new_BGR)

image_new = (255 * image_new).clip(0, 255).astype(np.uint8)

show_image_hist(image_new)
```



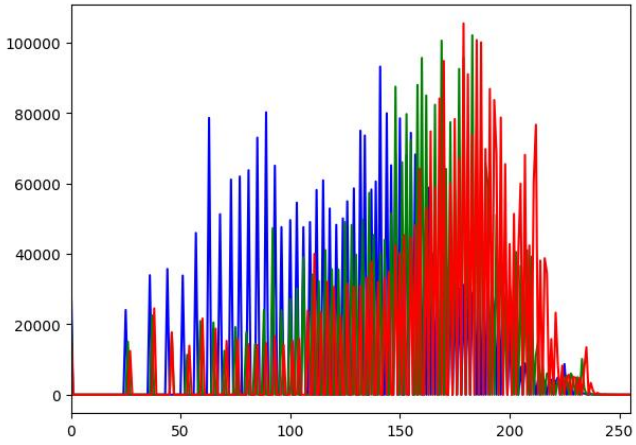


Рис.3. Изображение и гистограмма изображения после нелинейного преобразования интенсивностей.

Изображение стало различимо и практически реалистичным.

Листинг 1.4. Равномерное преобразование

Осуществляется по следующей формуле:

$Inew = (Imax - Imin) \cdot P(I) + Imin$ (1.2)

где $Imin$, $Imax$ — минимальное и максимальное значения интенсивностей исходного изоб ражения I; $P(I)$ — функция распределения вероятностей исходного изображения, которая аппроксимируется кумулятивной гистограммой:

$P(I) \approx$ sum(Hist(m) for m in range(0, I)) / (numRows * numCols) (1.3)

```
base_hist = []
CH = []

image_BGR = cv2.split(image)

for i, col in enumerate(color):
    base_hist.append(cv2.calcHist(image_BGR, [i], None, [hist_size], hist_range))
    CH.append(np.cumsum(base_hist[i]) / (image.shape[0] * image.shape[1]))

image_new = []
image_new_BGR = []

for i, layer in enumerate(image_BGR):
    Imin = layer.min()
    Imax = layer.max()

    image_new = np.clip(((Imax - Imin) * CH[i][image[:, :, i]] + Imin), 0, 255)
    image_new_BGR.append(image_new)

image_new = cv2.merge(image_new_BGR)
image_new = image_new.astype(np.uint8)

show_image_hist(image_new)
```

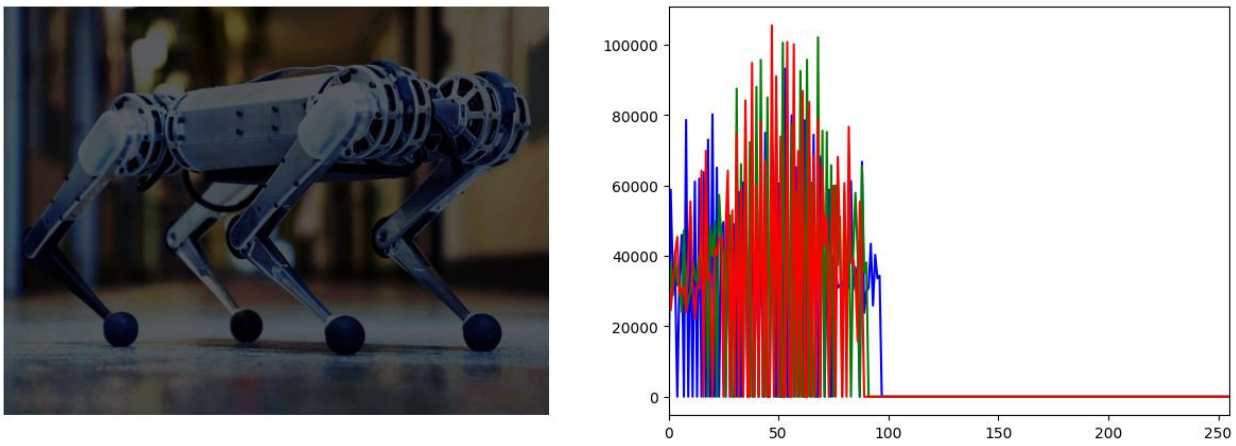


Рис.4. Изображение и гистограмма после равномерного преобразования интенсивностей.

Изображение получилось более насыщенным. Гистограмма получилась неровная, много цветов пропадает.

Стоит проверить алгоритм *равномерного преобразования* на черно-белом изображении:

```
image_test = cv2.imread("test.jpg")

base_hist = []
CH = []
```

```
image_BGR = cv2.split(image_test)

for i, col in enumerate(color):
    base_hist.append(cv2.calcHist(image_BGR, [i], None, [hist_size], hist_range))
    CH.append(np.cumsum(base_hist[i]) / (image_test.shape[0] *
image_test.shape[1]))

image_new = []
image_new_BGR = []

for i, layer in enumerate(image_BGR):
    Imin = layer.min()
    Imax = layer.max()

    image_new = np.clip(((Imax - Imin) * CH[i][image_test[:, :, i]] + Imin), 0,
255)
    image_new_BGR.append(image_new)

image_new = cv2.merge(image_new_BGR)
image_new = image_new.astype(np.uint8)

show_image_hist(image_new)
```

Рис.5. Черно-белое изображение и гистограмма после равномерного преобразования интенсивностей.

Гистограмма после равномерного преобразования интенсивностей на черно-белом изображении получилась более «ровная», чем на цветном.

Как видно из результатов графика, преобразование асимптотически линейно:

```
plt.plot(CH[0])
plt.show()
```

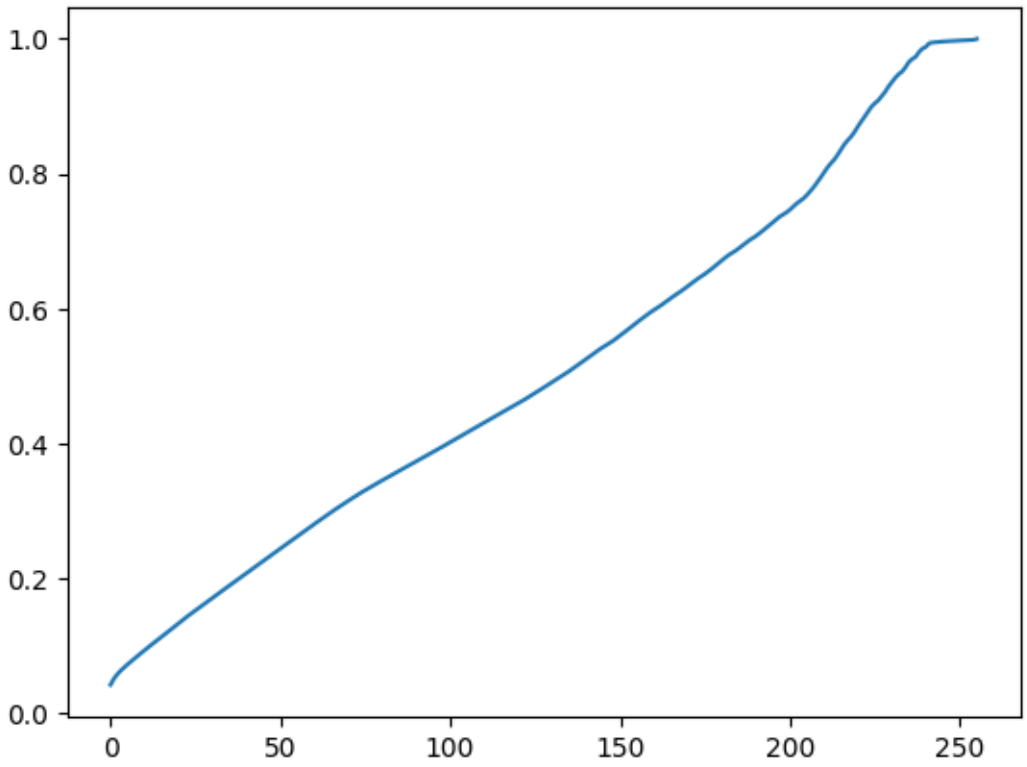


Рис.6. График кумулятивной гистограммы черно-белого изображения.

Листинг 1.5. Экспоненциальное преобразование

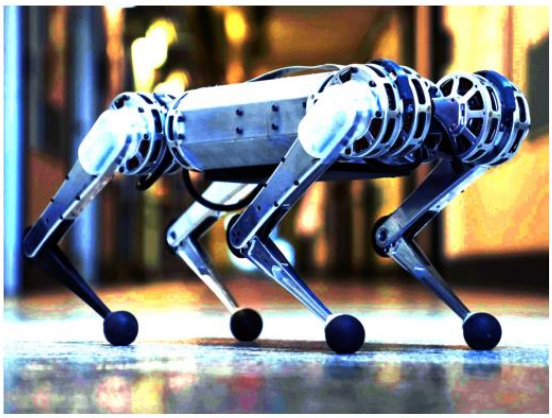
Осуществляется по следующей формуле:

$$Inew = Imin + 255 \cdot (1 / \alpha) \cdot \ln(1 - P(I)) \qquad (1.4)$$

где α — постоянная, характеризующая крутизну преобразования. Согласно формуле (1.4) можно вычислить значения интенсивностей пикселей результирующего изображения.

```
alpha = 1.7

base_hist = []
CH = []

image_BGR = cv2.split(image)

for i, col in enumerate(color):
    base_hist.append(cv2.calcHist(image_BGR, [i], None, [hist_size], hist_range))
    CH.append(np.cumsum(base_hist[i]) / (image.shape[0] * image.shape[1]))

Imin = [np.min(image[:,:,i]) for i in range(image.shape[2])]
image_new = np.zeros(image.shape, dtype="uint8")

for i in range(image.shape[2]):
    for y in range(image.shape[0]):
```

```
        for x in range(image.shape[1]):
            Inew = np.clip(Imin[i] - 255.0 * (1 / alpha) * np.log(1 -
CH[i][image[y][x][i]]), 0, 255)

            image_new[y][x][i] = int(round(Inew))

show_image_hist(image_new)
```



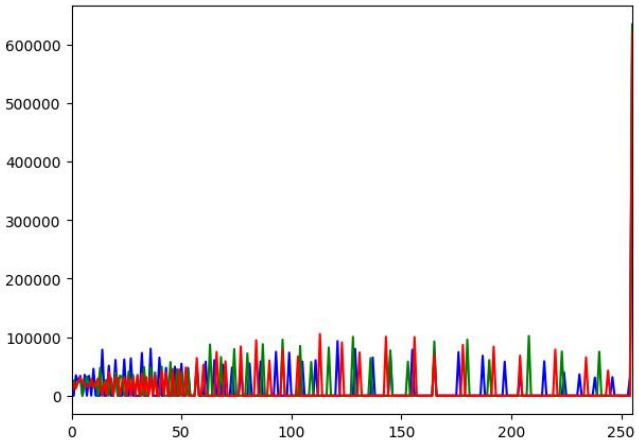


Рис.7. Изображение и гистограмма после экспоненциального преобразования интенсивностей.

При параметре α = 1.7 повысилась контрастность. Гистограмма стала похожей на нормальное распределение, но на имеет пик белого цвета.

Листинг 1.6. Преобразование по закону Рэлея

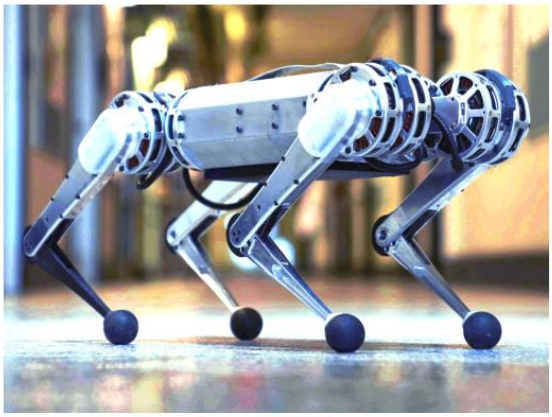
Осуществляется по следующей формуле:

$Inew = Imin + 255 \cdot (2\alpha\text{^}2 \log(1 / (1 - P(I)))) \text{ ^ } (1/2)$

де α — постоянная, характеризующая гистограмму распределения интенсивностей элемен тов результирующего изображения.

```
alpha = 0.5

base_hist = []
CH = []

image_BGR = cv2.split(image)

for i, col in enumerate(color):
    base_hist.append(cv2.calcHist(image_BGR, [i], None, [hist_size], hist_range))
    CH.append(np.cumsum(base_hist[i]) / (image.shape[0] * image.shape[1]))

Imin = [np.min(image[:,:,i]) for i in range(image.shape[2])]
image_new = np.zeros(image.shape, dtype="uint8")

for i in range(image.shape[2]):
```

```
    for y in range(image.shape[0]):
        for x in range(image.shape[1]):
            Inew = np.clip((Imin[i] + 255.0 * pow((2 * pow(alpha, 2) * np.log(1 /
(1 + 1e-9 - CH[i][image[y][x][i]]))), 0.5)), 0, 255)

            image_new[y][x][i] = int(round(Inew))

show_image_hist(image_new)
```



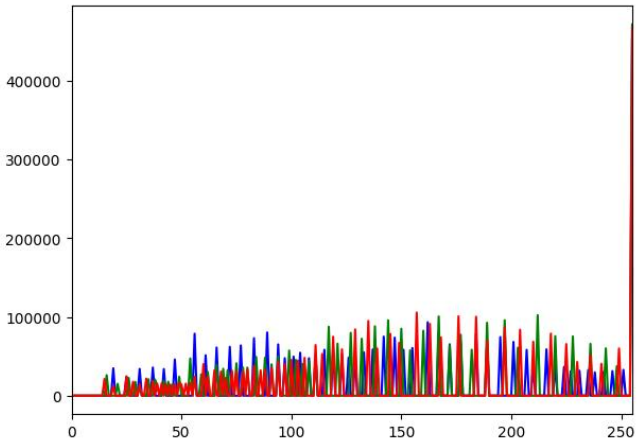


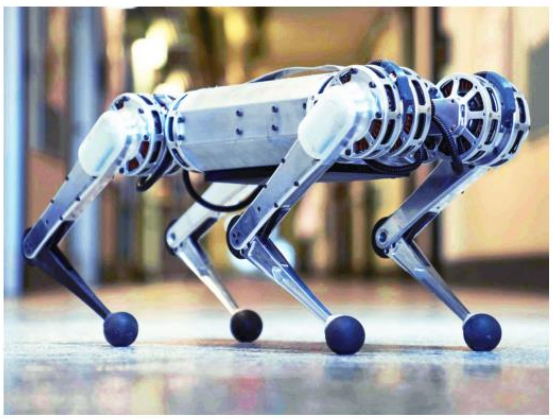
Рис.8. Изображение и гистограмма после преобразования Рэлея.

Изображение получается ярким и насыщенным, но еще присутствует пик белого на гистограмме

Листинг 1.7. Преобразование по закону степени 2/3

Осуществляется по следующей формуле: $Inew = 255 \cdot (P(I))^{(2/3)}$

```
base_hist = []
CH = []

image_BGR = cv2.split(image)

for i, col in enumerate(color):
    base_hist.append(cv2.calcHist(image_BGR, [i], None, [hist_size], hist_range))
    CH.append(np.cumsum(base_hist[i]) / (image.shape[0] * image.shape[1]))

Imin = [np.min(image[:,:,i]) for i in range(image.shape[2])]
image_new = np.zeros(image.shape, dtype="uint8")

for i in range(image.shape[2]):
    for y in range(image.shape[0]):
        for x in range(image.shape[1]):
           Inew = 255.0 * pow(CH[i][image[y][x][i]], 2 / 3)

           image_new[y][x][i] = int(round(Inew))

show_image_hist(image_new)
```



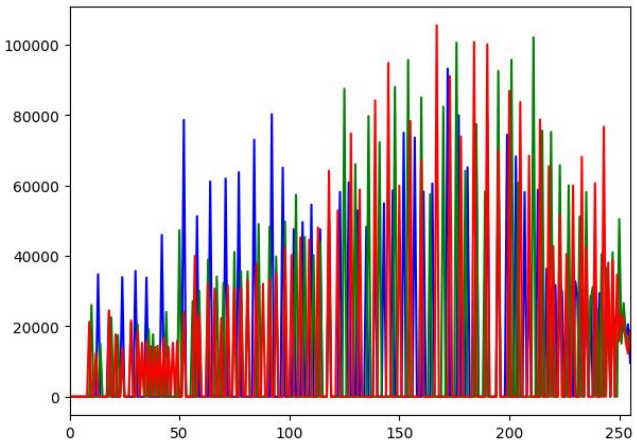


Рис.9. Изображение и гистограмма после преобразования по закону степени 2/3.

Изображение стало более «светлым» и отчетливым.

Листинг 1.8. Гиперболическое преобразование

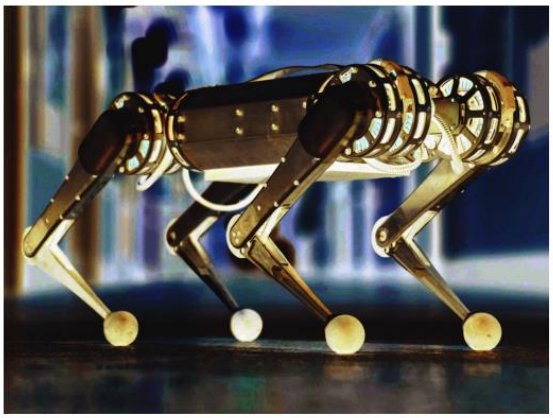
Осуществляется по следующей формуле:

$Inew = 255 * \alpha \wedge (P\ (I))$

где α — постоянная, относительно которой осуществляется преобразование и, как правил о, равная минимальному значению интенсивности элементов исходного изображения $\alpha = Imin$.

```
base_hist = []
CH = []

image_BGR = cv2.split(image)

for i, col in enumerate(color):
    base_hist.append(cv2.calcHist(image_BGR, [i], None, [hist_size], hist_range))
    CH.append(np.cumsum(base_hist[i]) / (image.shape[0] * image.shape[1]))

alpha = [np.min(image[:,:,i]) for i in range(image.shape[2])]
image_new = np.zeros(image.shape, dtype="uint8")

for i in range(image.shape[2]):
    for y in range(image.shape[0]):
        for x in range(image.shape[1]):
```

```
            Inew = 255.0 * pow(alpha[i] + 0.05, CH[i][image[y][x][i]])


            image_new[y][x][i] = int(round(Inew))


show_image_hist(image_new)
```



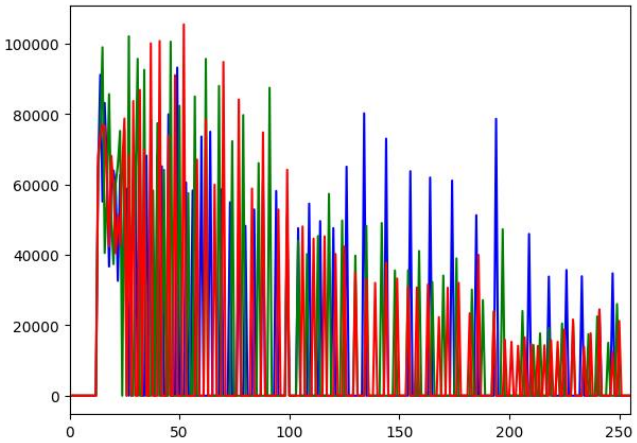


Рис.10. Изображение и гистограмма после гиперболического преобразования.

Это похоже на эффект "Негатив". Гистограмма интенсивности ведёт себя не постоянно и сильно меняется от изменения коэффициента смещения alpha_bias = 0.05, который предотвращает возведение «0» в степень.

Листинг 1.9. Эквализация и контрастно-ограниченное адаптивное выравнивание гистограммы

Для обработки цветного изображения данные функции можно применять поочередно к каждому цвету.

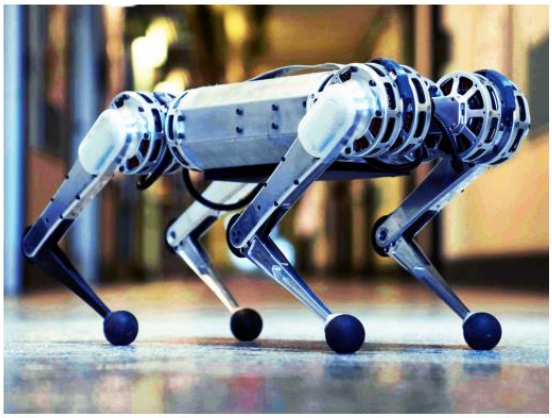
В библиотеке OpenCV также реализовано несколько функций для автоматического выравнивания гистограммы полутонового изображения :

1. equalizeHist() — эквализирует (выравнивает) гистограмму методом распределения значений интенсивностей элементов исходного изображения;

2. createCLAHE() и CLAHE.apply() — выполняет контрастно-ограниченное адаптивное выравнивание гистограммы методом анализа и эквализации гистограмм локальных окрестностей изображения
https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html)

3. createCLAHE (Contrast Limited Adaptive Histogram Equalization); (example https://docs.opencv.org/master/d5/daf/tutorial_py_histogram_equalization.html)

```
channels = np.array(cv2.split(image))

for i in range(len(channels)):
    channels[i] = cv2.equalizeHist(channels[i])

image_equalizeHist = cv2.merge(channels)
show_image_hist(image_equalizeHist)
```



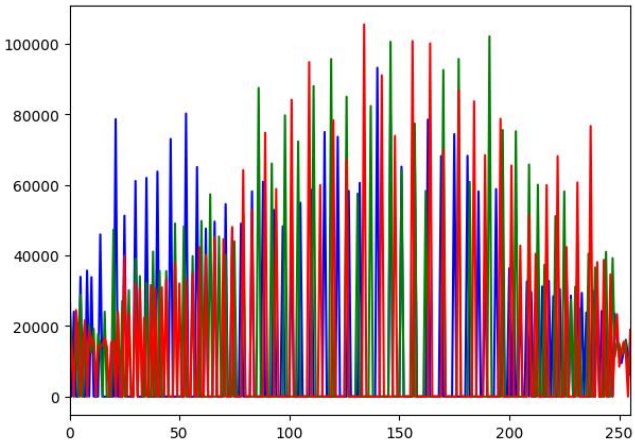


Рис.11. Изображение и гистограмма после equalizeHist.

```
clahe = cv2.createCLAHE(clipLimit=4.0, tileGridSize=(16,16))
channels = np.array(cv2.split(image))

for i in range(len(channels)):
    channels[i] = clahe.apply(channels[i])

image_CLAHE = cv2.merge(channels)
show_image_hist(image_CLAHE)
```

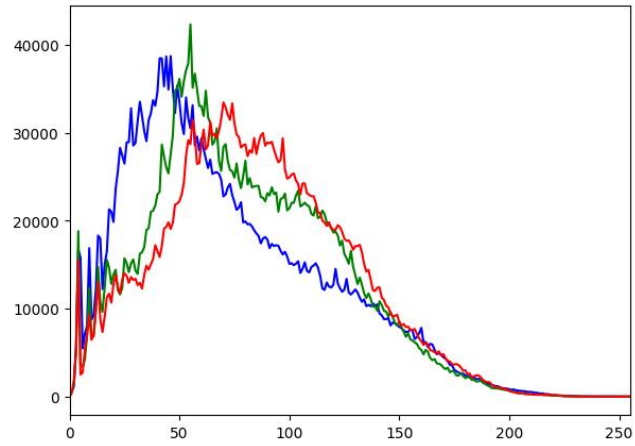


Рис.12. Изображение и гистограмма после CLAHE.

Изображения стали более детальными, следовательно данные функции можно применять для извлечения «глубоких» признаков из изображения.

Листинг 1.10. Поиск профиля штрих-кода вдоль оси Ox.

Профиль изображения

Профилем изображения вдоль некоторой линии называется функция интенсивности изобр ажения, распределенного вдоль данной линии (прорезки). Простейшим случаем профиля изображения является профиль строки:

Profile $i(x) = I(x,i)$, (1.8)

где i — номер строки изображения I. Профиль столбца изображения:

Profile $j(y) = I(j,y)$, (1.9)

где j — номер столбца изображения I.

В общем случае профиль можно рассматривать вдоль любой прямой, ломаной или кривой линии, пересекающей изображение. После формирования массива профиля изображения п роводится его анализ стандартными средствами. Анализ позволяет автоматически выделят ь особые точки функции профиля, соответствующие контурам изображения, пересекаемы м данной линией.

Рис.13. Исходный штрихкод.

```
image = cv2.cvtColor(cv2.imread("barcode.png"), cv2.COLOR_BGR2GRAY)


fig, axis = plt.subplots(1, 2, figsize=(15,5))


image_line = cv2.imread("barcode.png")


cv2.line(image_line, (0, image.shape[0] // 2), (image.shape[1], image.shape[0] //
2), color=(0, 0, 255), thickness=3)

axis[0].imshow(cv2.cvtColor(image_line, cv2.COLOR_BGR2RGB))

axis[0].axis('off')


profile = image[image.shape[0] // 2]

axis[1].plot(profile, color='black')


plt.show()
```

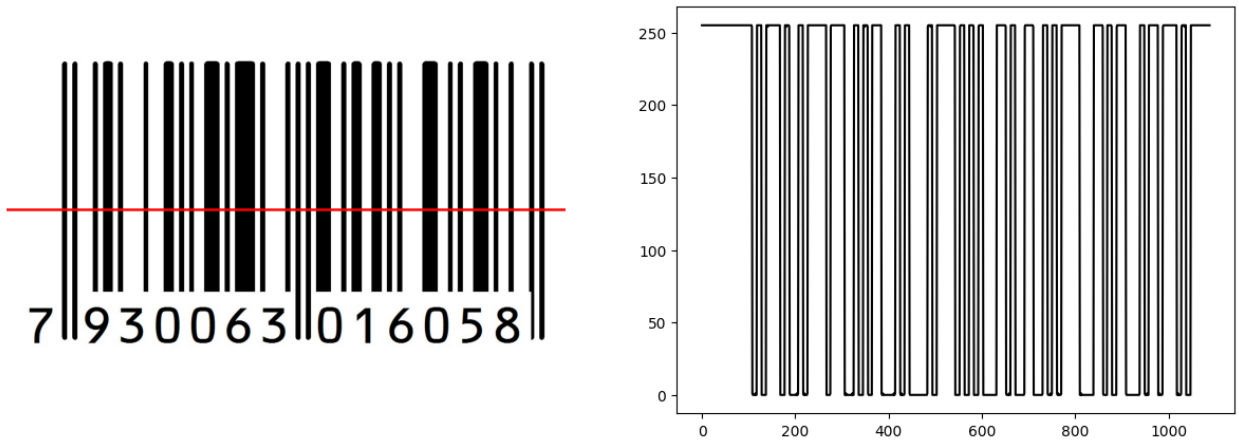


Рис.14. Штрихкод с центральной линии и профиль вдоль нее.

Полученные данные можно использовать для считывания самого штрих кода.

Листинг 1.11. Определение положения текста.

Проекция изображения Проекцией изображения на некоторую ось называется суммаинтенсивностей пикселей изображения в направлении, перпендикулярном данной оси. Простейшим случаем проекции двумерного изображения являются вертикальная

проекция на ось Ox, представляющая собой сумму интенсивностей пикселей по столбцам изображения:

Proj X(x) = sum($I(x,y)$ for y in range (0, len(Y)))

и горизонтальная проекция на ось Oy, представляющая собой сумму интенсивностей пикселей по строкам изображения:

Proj Y(y) = sum($I(x,y)$ for x in range (0, len(X)))

Запишем выражение для проекции на произвольную ось. Допустим, что направление оси задано единичным вектором с координатами (ex,ey). Тогда проекция изображения на ось Oe определяется следующим выражением:

Proj E(t) = sum($I(x,y)$ for all i where x = ex * t + ey * i + and y = ey * t + ex * i) Анализ массива проекции позволяет выделять характерные точки функции проекции, которые соответствуют контурам объектов на изображении. Например, если на изображении имеются контрастные объекты, то в проекции будут видны перепады или экстремумы функции, соответствующие положению каждого из объектов.

Подобные проекции могут быть использованы в алгоритмах обнаружения и сегментации текстовых строк в системах распознавания текста.

```
image = cv2.cvtColor(cv2.imread("text.jpg"), cv2.COLOR_BGR2GRAY)

Ox_projection = np.sum(255 - image, axis=0)
Oy_projection = np.sum(255 - image, axis=1)

plt.imshow(image, cmap='gray')
plt.axis('off')
plt.show()
```

Рис.15. Исходное изображение с текстом.

```
fig, axis = plt.subplots(1, 2, figsize=(15,5))

axis[0].set_title('Ox projection')
axis[0].plot(Ox_projection)

axis[1].set_title('Oy projection')
axis[1].plot(Oy_projection)

plt.show()
```

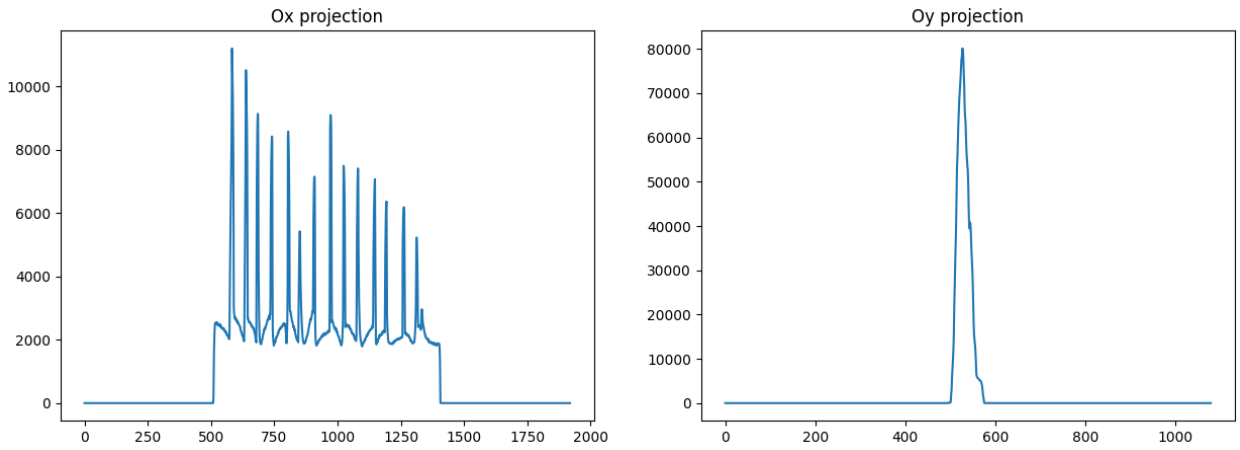


Рис.16. Проекция изображения с текстом относительно Ox и Oy.

```
def find_corners(projection):
    res1 = 50
    res2 = 0

    for i, val in enumerate(projection):
        if val > 0:
            res1 = i
            break

    for i, val in enumerate(projection, start=-len(projection)):
        if val > 0:
            res2 = abs(i)
            break

    return res1, res2

x1, x2 = find_corners(Ox_projection)
y1, y2 = find_corners(Oy_projection)

image = cv2.rectangle(cv2.imread("text.jpg"), (x1, y1 - 10), (x2, y2 + 10),
color=(255,0,0), thickness=3)

plt.axis('off')
plt.imshow(image)
plt.show()
```

Рис.32. Определение границ областей текста через bounding box.

По пикам относительно осей можно сделать вывод о местоположении текста.

Вопросы к защите лабораторной работы:

1. Что такое контрастность изображения и как её можно изменить?

Ответ:

Контрастность - разница между оттенками цвета предмета наблюдения и окружающего его фона. Если сформулировать проще, это разница между различными расположенными рядом цветами. Чем выше контрастность, тем более резко мы наблюдаем переход от одного цвета к другому.

Чтобы увеличить контраст изображения, нужно увеличить расстояние между максимальной и минимальной интенсивностью пикселей, т.е. выполнить растяжение гистограммы

2. Чем эффективно использование профилей и проекций изображения?

Ответ: Методы профилей и проекций помогают сжать/преобразовать информацию об изображении. Например, метод профиля для штрихкода помогает представить изображение в виде одномерной функции, состоящей только из горизонтальных и вертикальных линий. Метод проекции также сжимает данные ортогонально выбранной оси, тем самым получая пики функции (рис. 16), с помощью которых можно детектировать и сегментировать объекты, при условии однотонного фона.

3. Каким образом можно найти объект на равномерном фоне?

Ответ: При условии равномерного фона, объекты, зачастую текстовые, можно найти с помощью профилей, относительно двух осей, или через проекции на две оси, впоследствии через которые, с помощью пиков (значений, превышающих порог фона), определяют границы объектов.

Вывод

Были проведены преобразования гистограммы изображения, и обнаружен самый красивый способ выравнивания гистограммы – CLAHE.

Вычислены профили и проекции изображения на заданные оси.

Все вычисления проводились в python с использованием пакетов opencv, numpy.