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: студент гр. R3338,

Кирбаба Д.Д.

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 применяемых методов**

Основной единицей изображения является пиксель, который описывается следующими составляющими: положение (*x, y)*, интенсивность *I*. Для представления информации об интенсивности изображения строится гистограмма – график, показывающий распределение пикселей по интенсивности *I*. Анализируя гистограмму, можно узнать сведения о том, какие тона преобладают в изображении, определить диапазон интенсивностей (контраст), детектировать объекты по определенным диапазонам яркости.

Арифметические операции с изображениями – простейшие методы для увеличения или уменьшения интенсивности объекта, однако следует помнить, что при данном преобразовании может теряться информация об изображении.

Также необходимо уметь строить кумулятивную гистограмму – это такая гистограмма, элементом которой является сумма элементов гистограммы с индексами

Растяжение динамического диапазона, равномерное преобразование, экспоненциальное преобразование, преобразование по закону Рэлэя, преобразование по закону степени 2/3, гиперболическое преобразование – все эти способы анализа изображения по сути являются функциями, имеющими следующий общий вид:

где – массивы значений интенсивностей исходного и нового изображений соответственно. В зависимости от вида преобразования, в функциях используются также параметры (минимальное или максимальное значение интенсивности изображения, коэффициенты нелинейности, функция распределения вероятностей исходного изображения).

Различные преобразования используют для изменения свойств изображения (в данном случае яркостных свойств) как на отдельных частях, так и на целом изображении.

Для более эффективного преобразования можно не применять функции преобразования к каждому пикселю, а заранее вычислить все значения для массива . Таким образом останется только заменить исходные значения интенсивности на новые значения. Полученная в итоге таблица соответствия называется *LUT (Lookup table)*. Данный метод имеет широкое практическое применение.

Также применяются следующие способы анализа изображения: профиль изображения (функция интенсивности по некоторому направлению) и проекция изображения на направление (сумма интенсивностей на некоторую прямую, перпендикулярную заданному направлению). С помощью данных методов можно выделять особые точки профиля или проекции, и таким образом, получив информацию о контурах объекта, локализовать объект на монотонном фоне (проекция) или исследовать штрихкод (профиль).

**Ход выполнения работы**

1. **Гистограммы**



Picture 1: исходное слабоконтрастное изображения (слева – RGB, справа – Grayscale).

Контрастность изображения равна:

* 1. **Смещение гистограммы**

Выполним смещение гистограммы на 100 градаций влево, так как исходное изображение слишком светлое.

Listing 1. Сдвиг гистограммы.

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
data\_folder = "C:/denFiles/technical-vision/1lab/images/"  
  
img = cv.imread(data\_folder + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
  
i\_min = np.min(img)  
i\_max = np.max(img)  
contrast = i\_max - i\_min  
  
offset = -100  
  
new\_img = np.clip(img.astype(np.int16) + offset, 0, 255).astype(np.uint8)  
  
hist\_size = 256  
hist\_range = [0, 256]  
  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
  
plt.plot(hist, 'b', label="Initial")  
plt.plot(new\_hist, 'r', label="Shifted")  
plt.legend(loc="upper left")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(data\_folder + "offset.jpg", imgs)  
cv.imshow("Offset", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

Chart

Description automatically generated

Figure 2: Гистограммы изображений.

A picture containing outdoor, snow, tree, skiing

Description automatically generated

Figure 3: Исходное (слева) и смещенное (справа) изображения.

Так как в исходном изображении преобладают светлые тона, то и на построенной гистограмме мы видим, что почти все пиксели имеют интенсивность в диапазоне .

Сдвигом на 100 тонов влево мы добились уменьшения яркости (картинка стала темнее) и, соответственно, смещения гистограммы.

* 1. **Линейное выравнивание**

Применим метод линейного выравнивания изображения для того, чтобы увеличить контраст и распределить более равномерно количество пикселей по интенсивностям.

Listing 2. Линейное выравнивание.

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
data\_folder = "C:/denFiles/technical-vision/1lab/images/"  
  
img = cv.imread(data\_folder + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
  
l = 255  
n\_pix = img.size  
  
hist\_size = 256  
hist\_range = (0, 256)  
  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
norm\_hist = (l / n\_pix) \* hist  
cumul\_hist = np.clip(np.cumsum(norm\_hist), 0, 255)  
  
new\_img = cumul\_hist[img].astype(np.uint8)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
new\_norm\_hist = (l / n\_pix) \* new\_hist  
new\_cumul\_hist = np.clip(np.cumsum(new\_norm\_hist), 0, 255)  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 2)  
axis[0].plot(cumul\_hist, 'b', label="Initial")  
axis[0].plot(new\_cumul\_hist, 'r', label="Linear alignment")  
axis[1].plot(hist, 'b')  
axis[1].plot(new\_hist, 'r')  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(data\_folder + "lin\_al.jpg", imgs)  
cv.imshow("Linear alignment", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

Chart, histogram

Description automatically generated

Figure 4: Кумулятивные гистограммы (слева) и обычные гистограммы (справа) двух изображений (до и после применения выравнивания).

Как мы видим, кумулятивная гистограмма после применения метода выравнивания стала похожа на прямую линию, что говорит о том, что пиксели с одинаковой вероятностью имеют каждое значение интенсивности (так как кумулятивная гистограмма аппроксимирует функцию распределения пикселей по интенсивности). Однако из-за большого количества пикселей с интенсивностью лежащих в узком диапазоне линия переходит в ломаную, из-за этого на новом изображении будут резкие переходы во светлых тонах.

A picture containing tree, outdoor, snow, sky

Description automatically generated

Figure 5: Исходное (слева) и новое (справа) изображения.

Применим данный метод к другой картинке с более равномерным распределением пикселей.

A black and white photo of a building with a flag on top

Description automatically generated with low confidence

Figure 6: Исходное (слева) и новое (справа) изображения.

Детали изображения стали более различимы. Если в исходном изображении серые цвета превалировали, то после применения метода, картинка стала более контрастной.

Chart, line chart, histogram

Description automatically generated

Figure 7: Кумулятивные гистограммы (слева) и обычные гистограммы (справа) двух изображений (до и после применения выравнивания).

Убедились, что проблема разрыва в светлых тонах была из-за наличия большого количества пикселей в узком диапазоне, так как при втором изображении метод работает хорошо.

* 1. **Растяжение динамического диапазона**

Как видно из формулы:

Данный метод позволяет расширить динамический диапазон за счет сдвига (), а также влияет на новые интенсивности путем возведения в степень .

*Листинг 3. Растяжение динамического диапазона*

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
img\_folder = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(img\_folder + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
i\_max = np.max(img)  
i\_min = np.min(img)  
  
hist\_size = 256  
hist\_range = [0, 256]  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
cumul\_hist = np.cumsum(hist)  
  
non\_linearity\_factor = 0.7  
new\_img = np.power((img.astype(np.float32) - i\_min) / (i\_max - i\_min), non\_linearity\_factor)  
new\_img = (255 \* new\_img).astype(np.uint8)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
new\_cumul\_hist = np.cumsum(new\_hist)  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 2)  
axis[0].plot(cumul\_hist, 'b', label="Initial")  
axis[0].plot(new\_cumul\_hist, 'r', label="Linear alignment")  
axis[1].plot(hist, 'b')  
axis[1].plot(new\_hist, 'r')  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(img\_folder + "stretch\_dyn\_r.jpg", imgs)  
cv.imshow("Stretching dynamic range", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

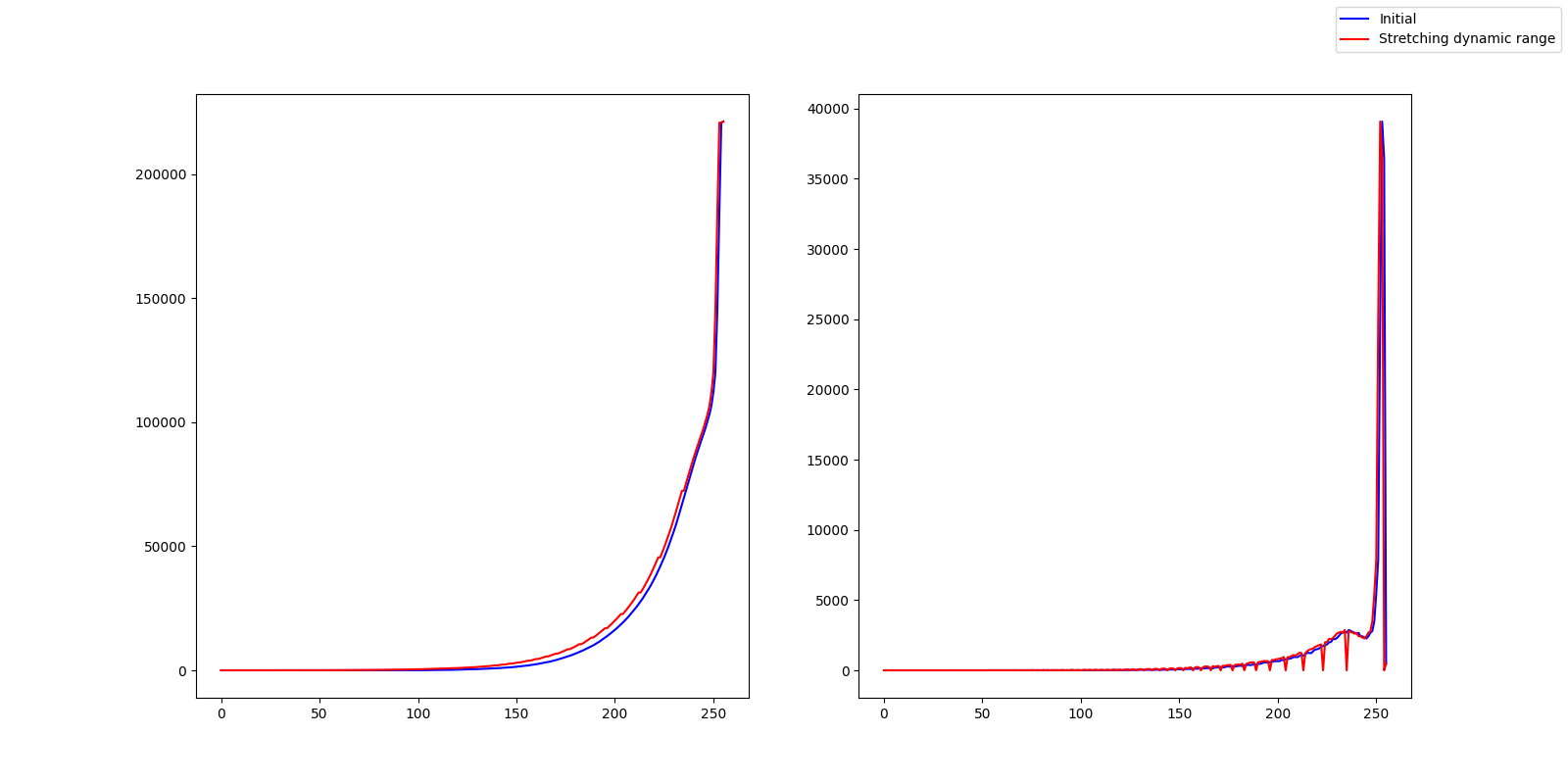


Figure 8: Кумулятивные и обычные гистограммы изображений .

A snowy landscape with trees

Description automatically generated with low confidence

Figure 9: Исходное (слева) и новое (справа) изображения .

Применение метода не дало существенных результатов, так как коэффициент нелинейности мал () для нужного нам увеличения исходных интенсивностей.

Попробуем

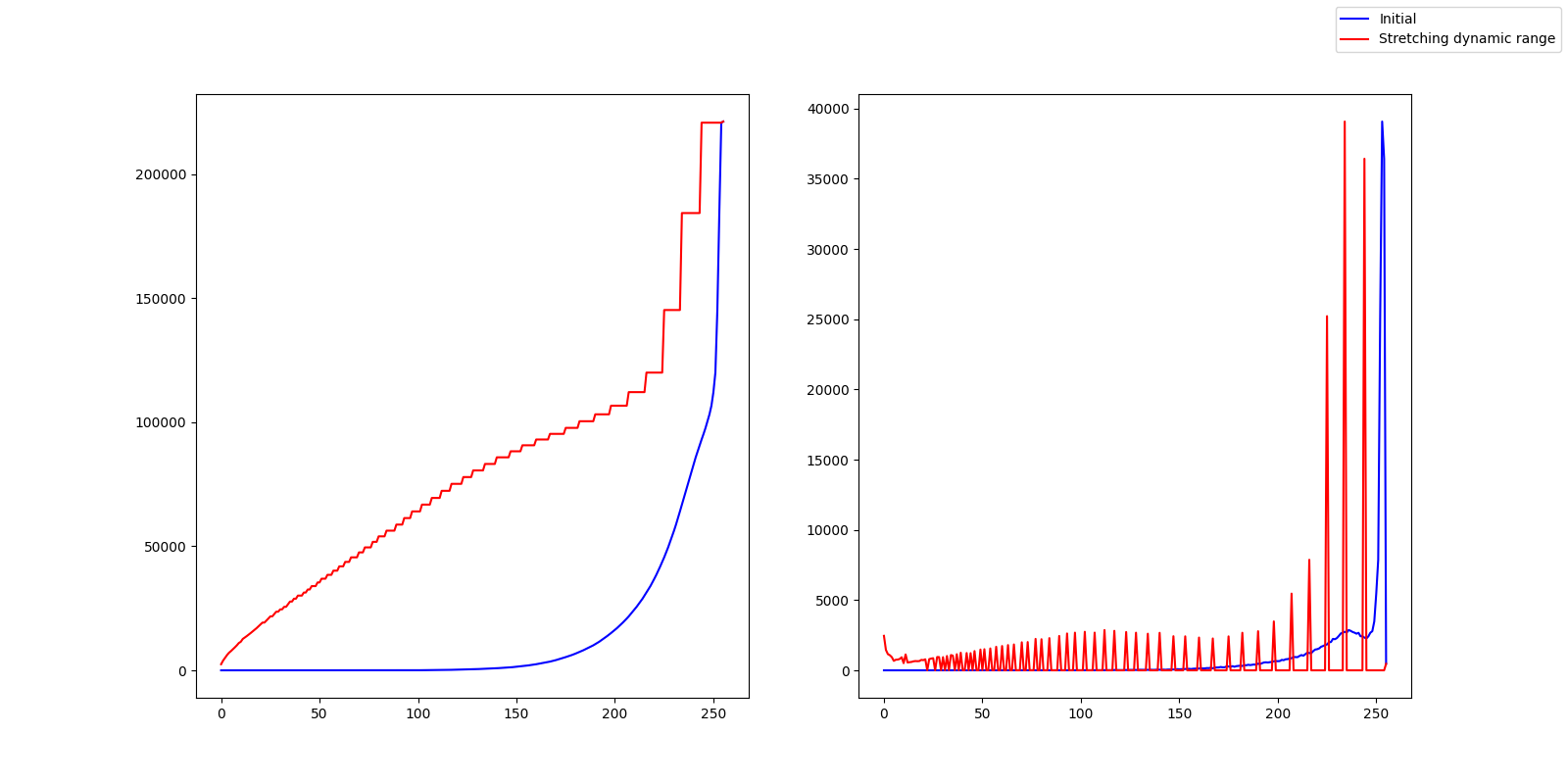


Figure 10: Кумулятивные и обычные гистограммы изображений .

A picture containing tree, outdoor, snow, sky

Description automatically generated

Figure 11: Исходное (слева) и новое (справа) изображения .

В итоге, данный метод позволяет усилить или наоборот уменьшить исходную конфигурацию интенсивностей изображения в зависимости от выбора степени. Тогда же как рассмотренный ранее метод линейного выравнивания старается сравнять вероятности появления каждого значения из всего диапазона интенсивностей.

* 1. **Равномерное преобразование**

Данное преобразование практически аналогичен линейному выравниванию, за исключением того, что равномерное преобразование делает отступ на значение , а значит, что диапазон интенсивностей будет меньше.

*Listing 4. Равномерное преобразование*

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
src\_path = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(src\_path + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
i\_max = np.max(img)  
i\_min = np.min(img)  
  
hist\_range = [0, 256]  
hist\_size = 256  
  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
cumul\_hist = np.cumsum(hist) / img.size  
  
new\_img = np.clip((i\_max - i\_min) \* cumul\_hist[img] + i\_min, 0, 255).astype(np.uint8)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
new\_cumul\_hist = np.cumsum(new\_hist) / new\_img.size  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 2)  
axis[0].plot(cumul\_hist, 'b', label="Initial")  
axis[0].plot(new\_cumul\_hist, 'r', label="Uniform transformation")  
axis[1].plot(hist, 'b')  
axis[1].plot(new\_hist, 'r')  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(src\_path + "uniform\_transform.jpg", imgs)  
cv.imshow("Uniform transformation", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

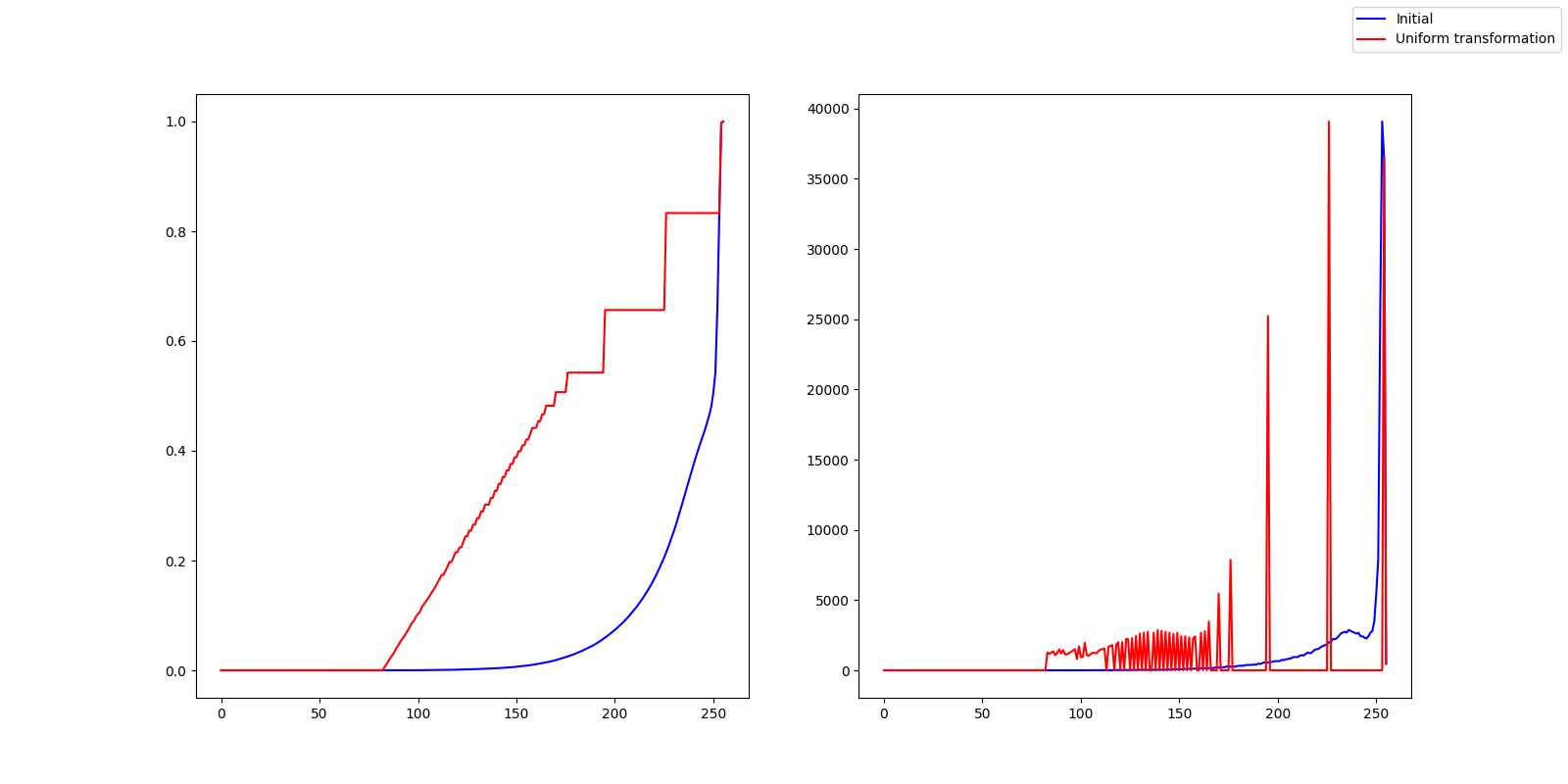


Figure 12: Кумулятивные и обычные гистограммы изображений.

A picture containing tree, outdoor, snow, covered

Description automatically generated

Figure 13: Исходное (слева) и новое (справа) изображения.

Сходство с методом равномерного выравнивания большое, однако из-за отступа тут не присутствуют темные тона.

* 1. **Экспоненциальное преобразование**

*Listing 5. Экспоненциальное преобразование*

import cv2 as cv  
import matplotlib.pyplot as plt  
import numpy as np  
  
img\_path = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(img\_path + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
i\_min = np.min(img)  
  
hist\_size = 256  
hist\_range = (0, 256)  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
cumul\_hist = np.cumsum(hist) / img.size  
cumul\_hist = np.where(cumul\_hist < 0.99999, cumul\_hist, 0.99999)  
  
slope\_factor = 2  
new\_img = np.clip(i\_min - (1 / slope\_factor) \* 255 \* np.log(1 - cumul\_hist[img]), 0, 255).astype(np.uint8)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
new\_cumul\_hist = np.cumsum(new\_hist) / new\_img.size  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 2)  
axis[0].plot(cumul\_hist, 'b', label="Initial")  
axis[0].plot(new\_cumul\_hist, 'r', label="Exp transformation")  
axis[1].plot(hist, 'b')  
axis[1].plot(new\_hist, 'r')  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(img\_path + "exp\_transform.jpg", imgs)  
cv.imshow("Exp transformation", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

Chart, histogram

Description automatically generated

Figure 14: Кумулятивные и обычные гистограммы изображений.

A picture containing tree, outdoor, snow, slope

Description automatically generated

Figure 15: Исходное (слева) и новое (справа) изображения.

Данное преобразование не увеличивает диапазон интенсивностей, так как в формуле присутствует слагаемое . Также кумулятивная гистограмма будет выглядеть как сдвинутый логарифм с определенным коэффициентом, который является по сути гипер-параметром .

* 1. **Преобразование по закону Рэлея**

*Listing 6. Преобразование по закону Рэлея*

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
src\_path = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(src\_path + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
i\_min = np.min(img)  
  
hist\_range = [0, 256]  
hist\_size = 256  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
cumul\_hist = np.cumsum(hist) / img.size  
cumul\_hist = np.where(cumul\_hist < 0.99999, cumul\_hist, 0.99999)  
  
alpha = 0.3  
new\_img = i\_min + 255 \* (2 \* (alpha \*\* 2) \* np.log(1/(1 - cumul\_hist[img]))) \*\* (1 / 2)  
new\_img = np.clip(new\_img, 0, 255).astype(np.uint8)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
new\_cumul\_hist = np.cumsum(new\_hist) / new\_img.size  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 2)  
axis[0].plot(cumul\_hist, 'b', label="Initial")  
axis[0].plot(new\_cumul\_hist, 'r', label="Rayleigh transformation")  
axis[1].plot(hist, 'b')  
axis[1].plot(new\_hist, 'r')  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(src\_path + "rayleigh\_transform.jpg", imgs)  
cv.imshow("Rayleigh transformation", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

*Chart, line chart, histogram

Description automatically generated*

Figure 16: Кумулятивные и обычные гистограммы изображений.

A picture containing tree, outdoor, snow, covered

Description automatically generated

Figure 17: Исходное (слева) и новое (справа) изображения.

Данное преобразование схоже с экспоненциальным, однако кумулятивная гистограмма напоминает логистическую функцию, таким образом, наибольшее количество пикселей будет иметь интенсивность – среднюю из всего диапазона (так как в середине логистической функции – производная максимальна (это не так на моем примере – из-за обилия светлых тонов в узком диапазоне)). В то время как при экспоненциальном преобразовании наибольшее количество писклей будет стремиться иметь интенсивность в начале диапазона (рост логарифма наибольший в начале).

* 1. **Преобразование по закону степени 2/3**

Данное преобразование будет очень схоже с экспоненциальным, так как также больше всего увеличивает интенсивность пикселей, находящихся в начале диапазона.

Также стоит отметить, что данное преобразование не имеет смещения и из-за этого будет увеличен диапазон интенсивностей.

*Listing 7. Преобразование по закону степени 2/3*

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
src\_path = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(src\_path + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
  
hist\_range = [0, 256]  
hist\_size = 256  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
cumul\_hist = np.cumsum(hist) / img.size  
  
new\_img = cumul\_hist[img] \*\* (2 / 3)  
new\_img = np.clip(new\_img \* 255, 0, 255).astype(np.uint8)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
new\_cumul\_hist = np.cumsum(new\_hist) / new\_img.size  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 2)  
axis[0].plot(cumul\_hist, 'b', label="Initial")  
axis[0].plot(new\_cumul\_hist, 'r', label="2/3 degree")  
axis[1].plot(hist, 'b')  
axis[1].plot(new\_hist, 'r')  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(src\_path + "two\_thirds\_transform.jpg", imgs)  
cv.imshow("2/3 degree transformation", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

*Chart, histogram

Description automatically generated*

Figure 18: Кумулятивные и обычные гистограммы изображений.

A picture containing tree, outdoor, snow, sky

Description automatically generated

Figure 19: Исходное (слева) и новое (справа) изображения.

Преобразование по закону 2/3 степени будет стремится увести наибольшее количество пикселей в области с большой интенсивностью, так как наибольший рост кумулятивная гистограмма будет иметь ближе к концу диапазона интенсивности.

* 1. **Гиперболическое преобразование**

*Listing 8. Гиперболическое преобразование*

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
src\_path = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(src\_path + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
  
hist\_range = [0, 256]  
hist\_size = 256  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
cumul\_hist = np.cumsum(hist) / img.size  
  
alpha = np.min(img) / 255  
new\_img = alpha \*\* cumul\_hist[img]  
new\_img = np.clip(new\_img \* 255, 0, 255).astype(np.uint8)  
new\_hist = cv.calcHist([new\_img], [0], None, [hist\_size], hist\_range)  
new\_cumul\_hist = np.cumsum(new\_hist) / new\_img.size  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 2)  
axis[0].plot(cumul\_hist, 'b', label="Initial")  
axis[0].plot(new\_cumul\_hist, 'r', label="Hyperbolic transformation")  
axis[1].plot(hist, 'b')  
axis[1].plot(new\_hist, 'r')  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, new\_img), axis=1)  
cv.imwrite(src\_path + "hyperbolic\_transform.jpg", imgs)  
cv.imshow("Hyperbolic transformation", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

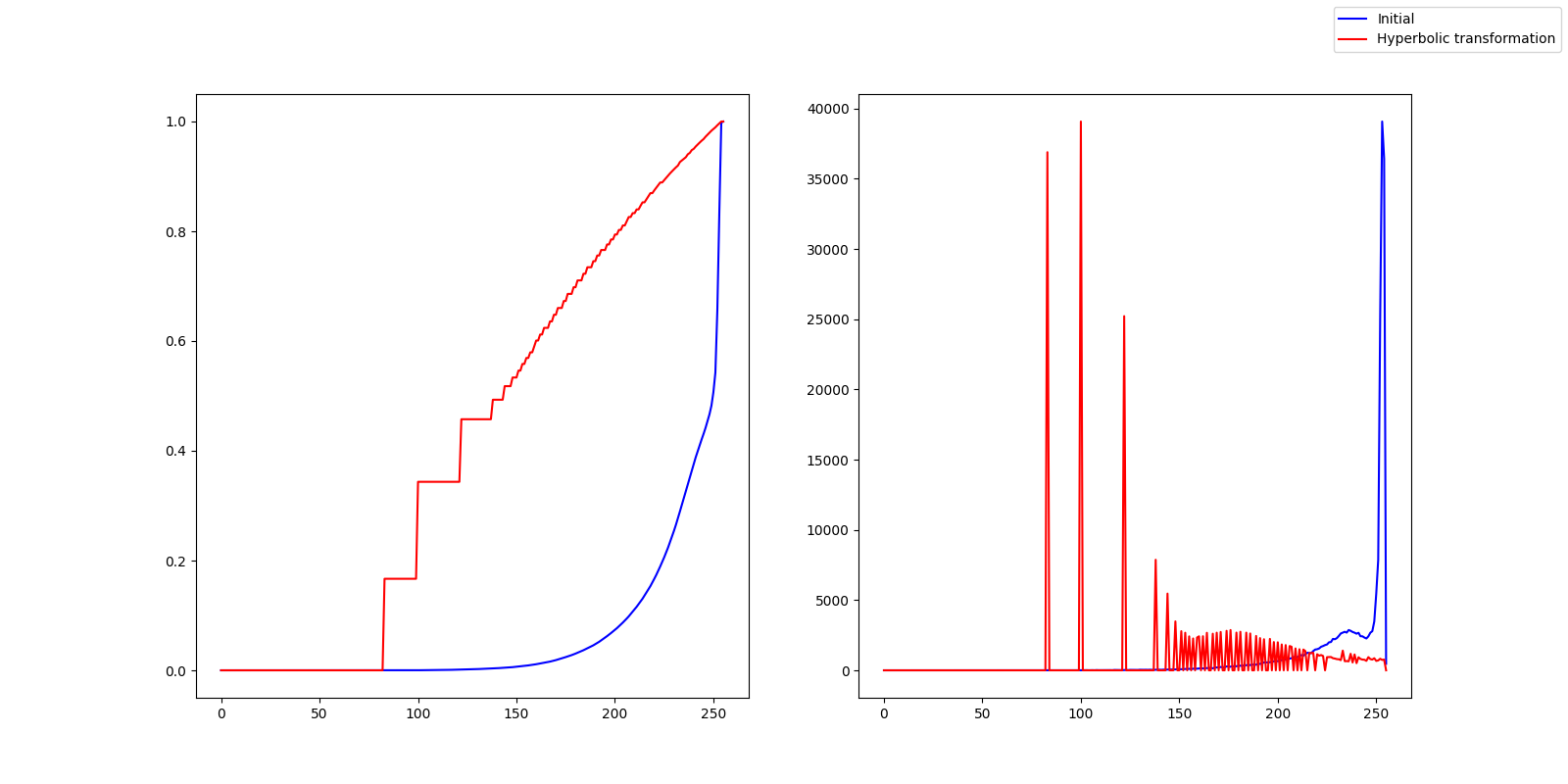


Figure 20: Кумулятивные и обычные гистограммы изображений.

A picture containing tree, outdoor, snow

Description automatically generated

Figure 21: Исходное (слева) и результирующее (справа) изображения.

* 1. **Встроенные функции преобразования**

Рассмотрим две функции для автоматического выравнивания гистограммы из библиотеки OpenCV: equalizeHist() и CLAHE.

*Listing 9. Встроенные функции преобразования.*

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
src\_path = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(src\_path + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
hist\_size = 256  
hist\_range = [0, 256]  
  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
eq\_img = cv.equalizeHist(img)  
eq\_hist = cv.calcHist([eq\_img], [0], None, [hist\_size], hist\_range)  
  
clahe = cv.createCLAHE()  
clahe\_img = cv.CLAHE.apply(clahe, img)  
clahe\_hist = cv.calcHist([clahe\_img], [0], None, [hist\_size], hist\_range)  
  
plt.rcParams["figure.figsize"] = (16,8)  
plt.plot(hist, 'r', label="initial")  
plt.plot(eq\_hist, 'b', label="equalised")  
plt.plot(clahe\_hist, 'g', label="CLAHE")  
plt.legend()  
plt.show()  
  
imgs = np.concatenate((img, eq\_img, clahe\_img), axis=1)  
cv.imshow("built-in funcs", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

Chart, histogram

Description automatically generated

Figure 22: Гистограммы исходного изображения и двух изображений после преобразований

A picture containing text, tree, outdoor, plant

Description automatically generated

Figure 23: Исходное изображение (слева), изображение после equalizeHist() (в центре) и изображение после CLAHE (справа)

Функция выравнивания гистограммы equalizeHist() дает очень схожий результат работы с преобразованием по закону степени 2/3 и обычным линейным выравниванием.

Метод CLAHE сработал лучше всего из всех предоставленных выше методов.

* 1. **Таблица поиска**

Применим метод LUT с реализацией динамического растяжения диапазона к изображению и сравним с таким же преобразованием. Но без использования LUT.

*Listing 10. LUT*

import cv2 as cv  
import numpy as np  
from matplotlib import pyplot as plt  
  
src\_path = "C:/denFiles/technical-vision/1lab/images/"  
img = cv.imread(src\_path + "snowforest.jpg", cv.IMREAD\_GRAYSCALE)  
i\_max = np.max(img)  
i\_min = np.min(img)  
  
hist\_size = 256  
hist\_range = [0, 256]  
hist = cv.calcHist([img], [0], None, [hist\_size], hist\_range)  
  
alpha = 0.7  
lut = np.arange(256, dtype=np.uint8)  
lut = (lut - i\_min) / (i\_max - i\_min)  
lut = np.where(lut > 0, lut, 0)  
lut = np.clip(255 \* np.power(lut, alpha), 0, 255).astype(np.uint8)  
img\_new = cv.LUT(img, lut)  
hist\_new = cv.calcHist([img\_new], [0], None, [hist\_size], hist\_range)  
  
plt.rcParams["figure.figsize"] = (16,8)  
figure, axis = plt.subplots(1, 1)  
axis.plot(hist, 'b', label="initial")  
axis.plot(hist\_new, 'r', label="dynamic range stretching lut")  
figure.legend(loc="upper right")  
plt.show()  
  
imgs = np.concatenate((img, img\_new), axis=1)  
cv.imwrite(src\_path + "dyn\_r\_lut.jpg", imgs)  
cv.imshow("Dynamic range stretching lut", imgs)  
cv.waitKey(0)  
cv.destroyAllWindows()

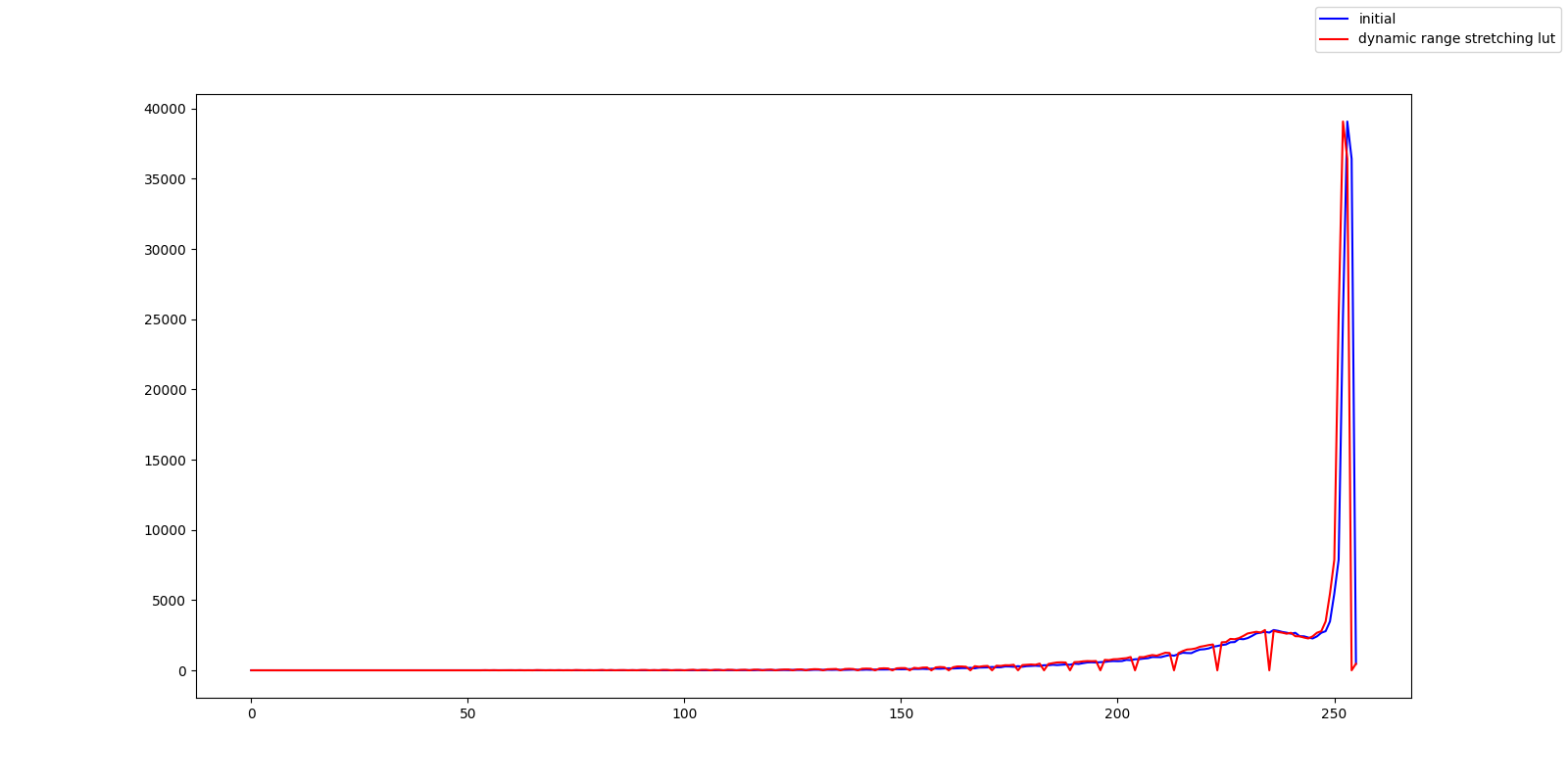
**

Figure 24: Гистограммы исходного и результирующего изображений.



Figure 25: Исходное (слева) и результирующее (справа) изображения.

Результаты полностью совпадают при применении как LUT, так и при прямом расчете интенсивностей при растяжении динамического диапазона, что и предполагалось.

Однако LUT работает намного быстрее (если речь идет о применении заранее вычисленной таблицы), так как нам необходимо запомнить всего лишь массив из 256 значений, а далее просто заменить исходные интенсивности на соответствующие в LUT. Таким образом, можно составить таблицы для любого преобразования яркостных характеристик изображения.

1. **Профили**
2. **Проекции**

**Выводы**