**ATILIM UNIVERSITY**

**CMPE464**

**Digital Image Processing**

**HOMEWORK #02**

Müslüm Barış KORKMAZER

19244710045

Ankara,

Spring, 2024

# Introduction

In this image processing report, we will analyze two given images by performing various operations including histogram plotting, histogram equalization, and local histogram equalization with different window sizes. These operations aim to enhance the visual quality of the images and provide insights into their pixel intensity distributions.

# Implementation Overview

We utilized Python along with the OpenCV and NumPy libraries for implementing the image processing tasks. The following operations were carried out for each image:

1. Plotting Histogram: The histogram of the input image was plotted to visualize the distribution of pixel intensities.
2. Histogram Equalization: Histogram equalization was applied to enhance the contrast of the image by redistributing pixel intensities.
3. Local Histogram Equalization: Local histogram equalization was performed with varying window sizes to observe its effect on different spatial scales of the image.

# Results and Analysis

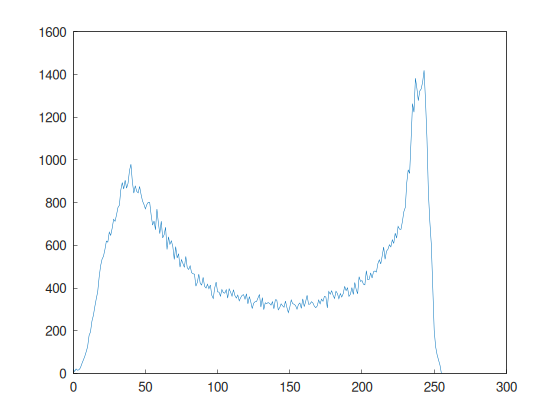
## 1st Image: Soldier

dış mekan, ağaç, giyim, kişi, şahıs içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 1: Soldier image

### Histogram Plotting

The histogram of Image 1 revealed, we can see peaks at beginning and end of plot.

Picture 2: Original image’s histogram

### Histogram Equalization

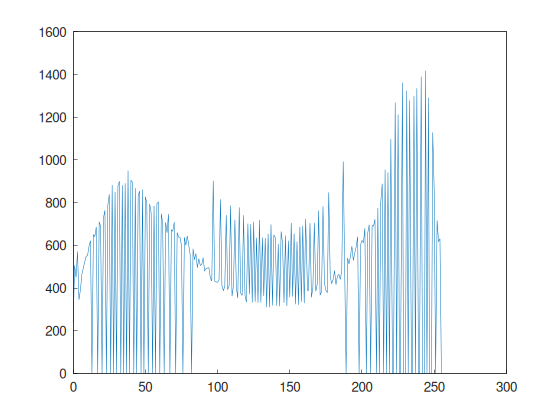
After applying histogram equalization, the contrast of Image 1 was significantly improved. Dark regions became brighter and details were enhanced.

dış mekan, ağaç, giyim, kişi, şahıs içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 3: Equalized soldier image

And here you can see in the histogram with these new peaks we can see more brighter and detailed image.



Picture 4: Equalized soldier image’s histogram

### Local Histogram Equalization

With a small window size, local histogram equalization preserved fine details but may have resulted in over-enhancement of noise.

Increasing the window size led to smoother transitions between regions but might have caused loss of sharpness in edges.

dış mekan, giyim, yürüyüş yapma, kişi, şahıs içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 5: Local equalized soldier image

metin, çizgi, ekran görüntüsü, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 6: Equalized soldier image

## 2nd Image: City

metin, harita içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 7: City image

### Histogram Plotting

The histogram of Image 2 exhibited, we can see peaks at middle of plot. I think its refers to river at photo.

çizgi, ekran görüntüsü, metin, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 8: Original image’s histogram

### Histogram Equalization

siyah beyaz, dış mekan, su, nehir içeren bir resim

Açıklama otomatik olarak oluşturulduSimilar to Image 1, histogram equalization significantly enhanced the contrast of Image 2, making details more distinguishable.

Picture 9: Equalized soldier image

metin, ekran görüntüsü, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturulduAnd here you can see in the histogram with these new peaks we can see brighter and more detailed image.

Picture 10: Equalized soldier image’s histogram

### Local Histogram Equalization

Analysis of Image 2 with local histogram equalization revealed similar trends as observed in Image 1, with the impact varying based on the window size.

harita, siyah beyaz, Hava fotoğrafçılığı, su içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 11: Local equalized soldier image

öykü gelişim çizgisi; kumpas; grafiğini çıkarma, diyagram, çizgi, metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Picture 12: Equalized soldier image

# Code

## Histogram/Histogram.hpp

#include <string>

#include <vector>

#include <iostream>

#include <opencv2/opencv.hpp>

class Histogram

{

private:

    cv::Mat image = cv::Mat(256, 256, CV\_8UC3);

    std::string image\_path;

    std::string image\_name;

    std::vector<cv::Mat> bgr\_channels;

    int HISTOGRAM\_SIZE;

    std::array<float, 2> RANGE[2] = {0, 255};

    cv::Mat gray\_image;

    cv::Mat normal\_histogram;

    cv::Mat equalized\_image;

    cv::Mat equalized\_histogram;

    cv::Mat equalized\_local\_image;

    cv::Mat equalized\_local\_histogram;

public:

*// Default constructor with random image*

    Histogram();

*// Constructor with parameters*

    Histogram(std::string const &\_image\_path, std::string const &image\_name, int HISTOGRAM\_SIZE, std::array<float, 2> const &\_RANGE);

*// Convert image to grayscale*

    void convertToGrayscale();

*// Calculate histogram*

    void calculateHistogram();

*// Equalize*

    void equalize();

*// Calculate histogram of equalized image*

    void calculateEqualizedHistogram();

*// Local Equalize*

    void localEqualize();

*// Calculate histogram of local equalized image*

    void calculateLocalEqualizedHistogram();

*// Get Image Name*

    std::string getImageName() const;

*// Get Image*

    cv::Mat getImage() const;

*// Get histogram*

    cv::Mat getHistogram() const;

*// Get Equalized Image*

    cv::Mat getEqualizedImage() const;

*// Get Equalized Histogram*

    cv::Mat getEqualizedHistogram() const;

*// Get Equalized Local Image*

    cv::Mat getEqualizedLocalImage() const;

*// Get Equalized Local Histogram*

    cv::Mat getEqualizedLocalHistogram() const;

*// Get X coordinates for histogram*

    std::vector<int> getPlotXCoordinates() const;

*// Get Y coordinates for histogram*

    std::vector<double> getPlotYCoordinates(cv::Mat) const;

};

## Histogram/Histogram.cpp

#include "Histogram.hpp"

#include <string>

#include <vector>

#include <iostream>

#include <opencv2/opencv.hpp>

*// Default constructor with random image*

Histogram::Histogram() : image\_path{"../images/"}, image\_name("random\_image.jpg"), HISTOGRAM\_SIZE(256)

{

*// Check if image path exists*

    if (*this*->image\_path.empty())

    {

*// Create folder for images*

        std::string command = "mkdir -p " + *this*->image\_path;

        system(command.c\_str());

    }

*// Fill image with random values*

    cv::randu(*this*->image, cv::Scalar::all(0), cv::Scalar::all(255));

*// Save image*

    cv::imwrite(*this*->image\_path + *this*->image\_name, *this*->image);

*// Split image into bgr channels*

    cv::split(*this*->image, *this*->bgr\_channels);

}

*// Constructor with parameters*

Histogram::Histogram(std::string const &\_image\_path, std::string const &image\_name, int HISTOGRAM\_SIZE, std::array<float, 2> const &\_RANGE)

    : image\_path{\_image\_path}, image\_name{image\_name}, HISTOGRAM\_SIZE{HISTOGRAM\_SIZE}, RANGE{\_RANGE}

{

*// Read image*

*this*->image = cv::imread(*this*->image\_path + *this*->image\_name, cv::IMREAD\_COLOR);

*// Split image into bgr channels*

    cv::split(*this*->image, *this*->bgr\_channels);

}

*// Convert image to grayscale*

void Histogram::convertToGrayscale()

{

*// Convert image to grayscale*

    cv::cvtColor(*this*->image, *this*->gray\_image, cv::COLOR\_BGR2GRAY);

}

*// Calculate histogram*

void Histogram::calculateHistogram()

{

*// Calculate histogram of image*

    std::array<const float \*, 1> histRange = {*this*->RANGE->data()};

    cv::calcHist(&*this*->bgr\_channels[0], 1, 0, cv::Mat(), *this*->normal\_histogram, 1, &*this*->HISTOGRAM\_SIZE, histRange.data());

}

*// Equalize image*

void Histogram::equalize()

{

*// Convert image to grayscale*

*this*->convertToGrayscale();

*// Equalize histogram of image*

    cv::equalizeHist(*this*->gray\_image, equalized\_image);

}

*// Calculate histogram of equalized image*

void Histogram::calculateEqualizedHistogram()

{

*// Calculate histogram of equalized image*

    std::array<const float \*, 1> histRange = {*this*->RANGE->data()};

    cv::calcHist(&*this*->equalized\_image, 1, 0, cv::Mat(), *this*->equalized\_histogram, 1, &*this*->HISTOGRAM\_SIZE, histRange.data());

}

*// Local Equalize histogram*

void Histogram::localEqualize()

{

*// Convert image to grayscale*

*this*->convertToGrayscale();

*// Apply local histogram equalization to the image*

*// Create a CLAHE object with 4.0 contrast limiting factor and 8x8 grid size*

    cv::Ptr<cv::CLAHE> clahe = cv::createCLAHE(4.0, cv::Size(8, 8));

    clahe->setClipLimit(4); *// Set the clip limit for contrast limiting*

*// Apply local histogram equalization to the image*

    clahe->apply(*this*->gray\_image, *this*->equalized\_local\_image);

}

*// Calculate histogram of local equalized image*

void Histogram::calculateLocalEqualizedHistogram()

{

*// Calculate histogram of local equalized image*

    std::array<const float \*, 1> histRange = {*this*->RANGE->data()};

    cv::calcHist(&*this*->equalized\_local\_image, 1, 0, cv::Mat(), *this*->equalized\_local\_histogram, 1, &*this*->HISTOGRAM\_SIZE, histRange.data());

}

*// Get Image Name*

std::string Histogram::getImageName() const

{

    return *this*->image\_name;

}

*// Get Image*

cv::Mat Histogram::getImage() const

{

    return *this*->image;

}

*// Get histogram*

cv::Mat Histogram::getHistogram() const

{

    return *this*->normal\_histogram;

}

*// Get Equalized Image*

cv::Mat Histogram::getEqualizedImage() const

{

    return *this*->equalized\_image;

}

*// Get Equalized Histogram*

cv::Mat Histogram::getEqualizedHistogram() const

{

    return *this*->equalized\_histogram;

}

*// Get Equalized Local Image*

cv::Mat Histogram::getEqualizedLocalImage() const

{

    return *this*->equalized\_local\_image;

}

*// Get Equalized Local Histogram*

cv::Mat Histogram::getEqualizedLocalHistogram() const

{

    return *this*->equalized\_local\_histogram;

}

*// Get X coordinates for histogram*

std::vector<int> Histogram::getPlotXCoordinates() const

{

*// Create x coordinates*

    std::vector<int> x(*this*->HISTOGRAM\_SIZE);

    for (int i = 0; i < *this*->HISTOGRAM\_SIZE; ++i)

    {

        x[i] = i;

    }

    return x;

}

*// Get Y coordinates for histogram*

std::vector<double> Histogram::getPlotYCoordinates(cv::Mat hist) const

{

*// Create plot data*

    std::vector<double> y;

    for (int i = 0; i < *this*->HISTOGRAM\_SIZE; ++i)

    {

        y.push\_back(hist.at<float>(i));

    }

    return y;

}

## main.cpp

#include <iostream>

#include "Histogram/Histogram.hpp"

#include <matplot/matplot.h>

*// Necessary for this project*

const std::string IMAGES\_PATH = "/home/babico/Projects/CMPE464-Github/HW2/images/";

*// int main(int argc, char const \*argv[])*

int main()

{

    using namespace matplot;

*// Array of images*

    std::vector<Histogram> images = {

        Histogram(),

        Histogram(IMAGES\_PATH, "city.jpg", 256, {0, 255}),

        Histogram(IMAGES\_PATH, "soldier.bmp", 256, {0, 255}),

    };

*// Calculate histograms of all images*

    for (auto& image : images) {

*// Calculate histogram of image*

        image.calculateHistogram();

*// Equalize the image*

        image.equalize();

*// Calculate histogram of equalized image*

        image.calculateEqualizedHistogram();

*// Local Equalize the image*

        image.localEqualize();

*// Calculate histogram of local equalized image*

        image.calculateLocalEqualizedHistogram();

    }

*// Save images*

    for (auto const& image : images) {

*// Create folder for images*

        std::string image\_name = image.getImageName().substr(0, image.getImageName().find("."));

        std::string IMAGE\_PATH = IMAGES\_PATH + image\_name + "/";

        std::string command = "mkdir -p " + IMAGE\_PATH;

        system(command.c\_str());

*// Matplot++ instance*

        auto plotinstance = matplot::figure(true);

*// Create plot for histogram*

        plot(image.getPlotXCoordinates(), image.getPlotYCoordinates(image.getHistogram()));

        plotinstance->save(IMAGE\_PATH + "histogram\_" + image\_name + ".png");

*// Save equalized image*

        cv::imwrite(IMAGE\_PATH + "equalized\_" + image\_name + ".png", image.getEqualizedImage());

*// Save equalized histogram of image*

        plot(image.getPlotXCoordinates(), image.getPlotYCoordinates(image.getEqualizedHistogram()));

        plotinstance->save(IMAGE\_PATH + "equalized\_histogram\_" + image\_name + ".png");

*// Save local equalized image*

        cv::imwrite(IMAGE\_PATH + "equalized\_local\_" + image\_name + ".png", image.getEqualizedLocalImage());

*// Save local equalized histogram of image*

        plot(image.getPlotXCoordinates(), image.getPlotYCoordinates(image.getEqualizedLocalHistogram()));

        plotinstance->save(IMAGE\_PATH + "equalized\_local\_histogram\_" + image\_name + ".png");

    }

    return 0;

}

# Git Repository

[babico/CMPE464: Atılım University CMPE464 Digital Image Processing (github.com)](https://github.com/babico/CMPE464)