Computer vision- Assignment 2

Image segmentation and image enhancement

In this assignment report we look at two fundamental problems in computer vision, **image segmentation** and **image enhancement**. The aim of image segmentation is to partition an image into multiple regions or segments, with each segment corresponding to a distinct object or part of the image. Detection, recognition and tracking are some examples of the different use-cases for segmentation. The segmentation methods I've used for this are: thresholding, k-means clustering and region-growing. The aim of image enhancement is to improve the quality of an image, this can be done through adjusting for example the brightness, or the contrasts in the picture.

I used MATLAB for implementing our image segmentation algorithms, and the code can be accessed through GitHub: https://github.com/marhjoh/Image-segmentation-and-enhancement

Thresholding is a simple and fast method of image segmentation that separates pixels into two classes based on their intensity values. Firstly I converted the input image to grayscale, then I performed thresholding. The pixels with intensity values above the threshold were considered part of one class, while all pixels with intensity values below the threshold were considered part of the other class.

```
%% Threshhold Segmentation
% Thresholding is a simple image segmentation method that separates pixels into two classes based on their intensity values.
% All pixels with intensity values above a certain threshold value are considered part of one class, while all pixels with
% intensity values below the threshold are considered part of the other class.

function binary_img = my_thresholding(img, threshold)

% Convert the image to grayscale
img_gray = rgb2gray(img);

% Perform thresholding
binary_img = img_gray > threshold;
end
```

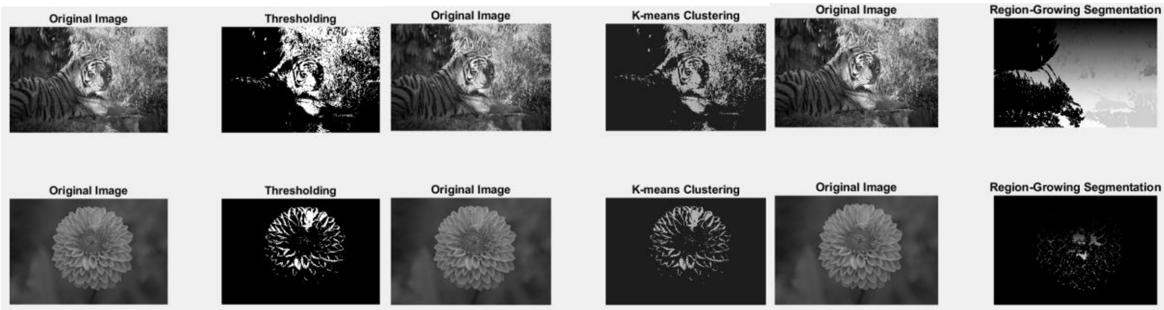
K-means clustering is a method that segments an image into K clusters. Firstly I converted the input image to grayscale and double, initialized the segmentation matrix, and defined the step size for thresholding. I then initialized the centers array and computed the initial centers. After that the code loops over each pixel in the image, computes the index of the closest center, and lastly assigns each pixel to the corresponding segment.

```
%% K-means clustering Segmentation
% The k-means algorithm is used to segment an image into K regions or
% clusters, much alike thresholding but with the oppurtunity to allow more
% than two "classes".

function [seg, centers] = my_kmeans(img, K)

    % Convert the input image to grayscale and double
    img_gray = double(rgb2gray(img));

    % Initialize the segmentation matrix
    seg = zeros(size(img_gray));

    % Define the step size for thresholding
    step = 256/K;

    % Initialize the centers array
    centers = zeros(1,K);

    % Compute the initial centers
    for i = 1:K
        centers(i) = (i-1/2)*step;
    end

    % Loop over each pixel in the image
    for i = 1:size(img_gray, 1)
        for j = 1:size(img_gray, 2)
            % Compute the index of the closest center
            [~, index] = min(abs(img_gray(i,j) - centers));
            % Assign the pixel to the corresponding segment
            seg(i,j) = index;
        end
    end
end
```

Region-growing is a method that segments an image based on the similarity of neighboring pixels. Firstly I converted the input image to grayscale and double, initialized the segmentation matrix, and defined the 4-connected neighborhood. Then the code loops over each pixel in the image and adds neighboring pixels that were within the threshold.

```
%% Region-Growing Segmentation
% The threshold in the region growing segmentation algorithm determines the maximum allowed difference in pixel intensity
% between neighboring pixels in the same segment. Low threshold result in neighboring pixels that are very different in
% intensity are allowed to belong to the same segment making the execution time high. High threshold result in only neighboring
% pixels that are very similar in intensity are allowed to belong to the same segment. Choosing an appropriate threshold is critical
% for the success of the region growing segmentation algorithm.

function seg = my_region_growing(img, threshold)
    % Convert the input image to grayscale and double
    img = double(rgb2gray(img));

    % Initialize the segmentation matrix
    seg = zeros(size(img));

    % Define the 4-connected neighborhood
    neighbor = [-1 0; 1 0; 0 -1; 0 1];

    % Loop over each pixel in the image
    for i = 1:size(img, 1)
        for j = 1:size(img, 2)
            % If the current pixel has not been assigned to a segment
            if seg(i, j) == 0
                % Create a new segment and add the current pixel to it
                seg_id = max(seg(:)) + 1;
                seg(i, j) = seg_id;

                % Initialize the region growing queue with the current pixel
                queue = [i, j];

                % Grow the segment by adding neighboring pixels that are within the threshold
                while ~isempty(queue)
                    % Get the next pixel from the queue
                    curr_pix = queue(1, :);
                    queue(1, :) = [];

                    % Check the 4-connected neighbors of the current pixel
                    for k = 1:size(neighbor, 1)
                        neighbor_pix = curr_pix + neighbor(k, :);

                        % If the neighbor pixel is within the image bounds
                        if all(neighbor_pix > 0) && neighbor_pix(1) <= size(img, 1) && neighbor_pix(2) <= size(img, 2)
                            % If the neighbor pixel has not been assigned to a segment
                            if seg(neighbor_pix(1), neighbor_pix(2)) == 0
                                % If the neighbor pixel intensity is within the threshold
                                if abs(img(curr_pix(1), curr_pix(2)) - img(neighbor_pix(1), neighbor_pix(2))) <= threshold
                                    % Add the neighbor pixel to the segment and the queue
                                    seg(neighbor_pix(1), neighbor_pix(2)) = seg_id;
                                    queue(end+1, :) = neighbor_pix;
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
```

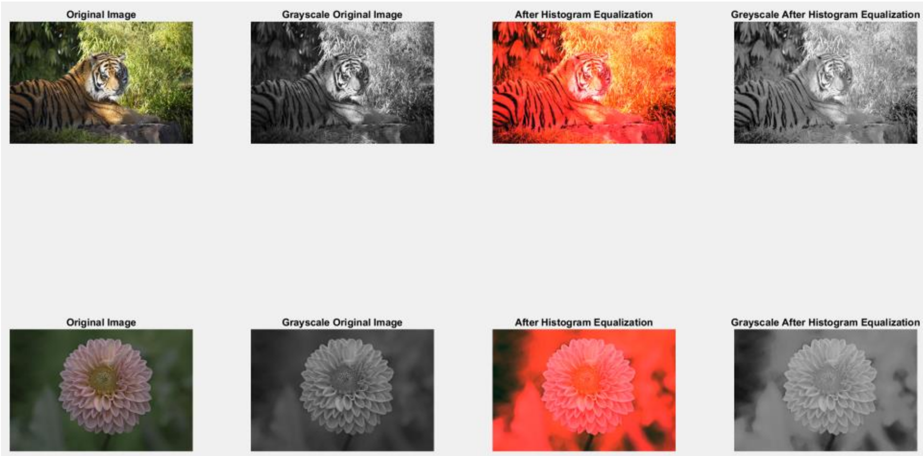
I tested the segmentation methods on two different pictures that were provided in the assignment ‘flower.jpg’ and ‘tiger.jpg’ and achieved decent results for thresholding and k-means clustering. I didn’t quite manage to achieve a satisfiable result for the region-growing segmentation method as it became a bit too complex for me too create a more optimal method. Observations that were made during the image segmentations were that thresholding was the fastest method, k-means clustering was not far behind, while region-growing was much slower. I also noticed that the appropriate selection of parameters such as threshold and the number of clusters significantly affected the segmentation results and the execution time.



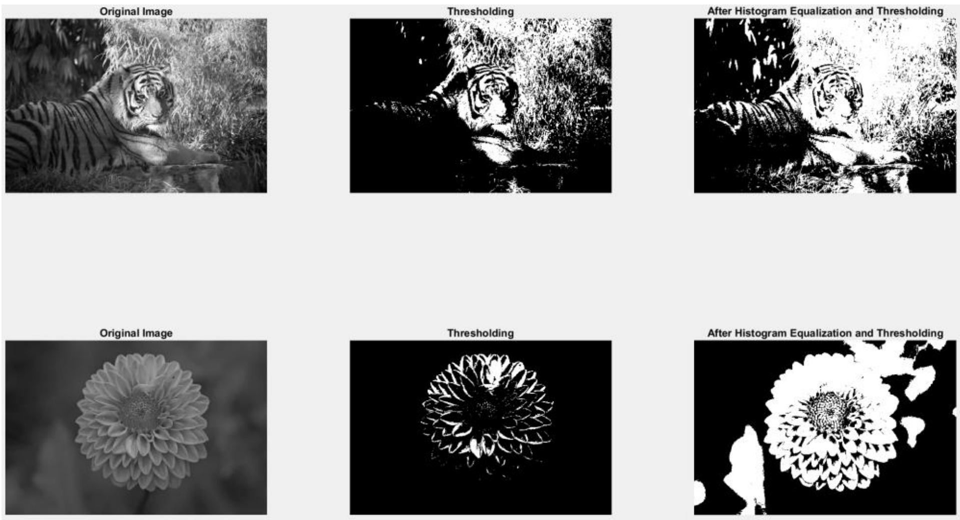
The **image enhancement** technique I decided to use was **histogram equalization**, which is used to improve the contrast of an image by redistributing the pixel values in the image histogram. In other words, it spreads out the intensity values in the image to cover the entire intensity range, which results in a more evenly distributed histogram.

```
%% Histogram Equalization
% Histogram equalization is a technique used to improve the contrast of an
% image by redistributing the pixel values in the image histogram. In other
% words, it spreads out the intensity values in the image to cover the
% entire intensity range, which results in a more evenly distributed histogram.

function [img_new, cdfnorm] = my_histeq(img)

    % calculate cumulative distribution function of original image
    h = imhist(img);
    cdf = zeros(1,256);
    for i=1:256
        cdf(i) = sum(h(1:i));
    end

    cdfnorm = round( ((cdf - min(cdf))/(size(img, 1)*size(img, 2) - min(cdf)))*255 );

    % apply the mapping to image
    img_new = img;
    for i=1:size(img,1)
        for j=1:size(img,2)
            img_new(i,j) = cdfnorm(img(i,j)+1);
        end
    end

    % calculate cumulative distribution function of enhanced image
    h_new = imhist(img_new);
    cdf_new = zeros(1,256);
    for i=1:256
        cdf_new(i) = sum(h_new(1:i));
    end
end
```

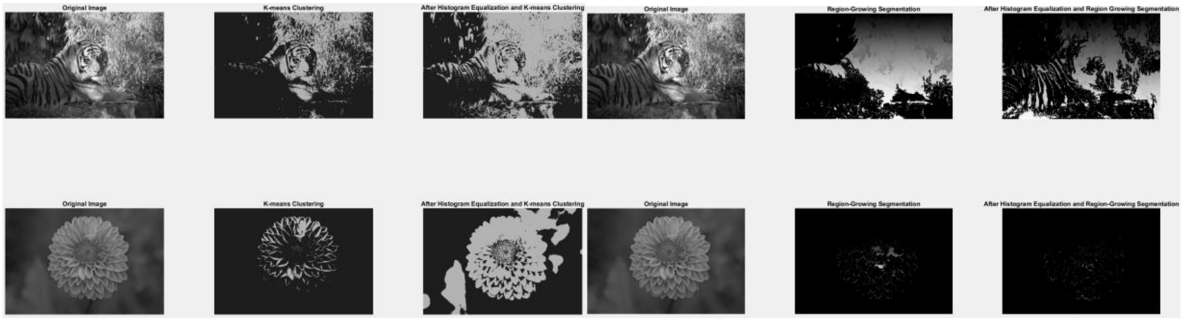
The reason why I chose the enhancement technique is that it can help improve the results of the segmentation methods by enhancing the contrast and making the features of interest more distinguishable from the background. The effect of histogram equalization varied depending on the specific image and parameters used. An example output is provided in the repository under /images/example output, but also below.



For example, in thresholding, histogram equalization can help in cases where the image has low contrast, and the features of interest are not clearly separated from the background – like the tiger picture. By spreading out the pixel values in the image, histogram equalization can help to better separate the features of interest from the background.



Similarly, in k-means clustering and region-growing, histogram equalization can help to improve the separation of clusters or segments by making their respective features more distinguishable from one another. This can lead to more accurate segmentation results and better preservation of the original features in the image, which it did to some degree with the tiger picture.



However, it is worth noting that histogram equalization can also introduce artifacts or noise into the image, particularly in regions with low pixel intensity. Therefore, after evaluating the results I would not accept the trade-offs in the flower picture. On the other hand the tiger picture is more debatable since different features of the tiger is highlighted before and after the enhancement. More of the body is distinguishable after the enhancement, while the head is more visable without it.

In conclusion, the report discusses two fundamental problems in computer vision: image segmentation and image enhancement. The aim of image segmentation is to partition an image into multiple regions or segments, while image enhancement aims to improve the quality of an image. The report examines three segmentation methods: thresholding, k-means clustering, and region-growing. The selection of appropriate parameters significantly affects the segmentation results and execution time. The report also discusses the histogram equalization technique for image enhancement, which can help improve segmentation results by enhancing contrast and making features more distinguishable from the background. However, histogram equalization can introduce artifacts or noise into the image, particularly in regions with low pixel intensity. The report provides examples and observations for each method and concludes that selecting the appropriate technique and parameters is critical to achieve the desired results.

References

Gonzalez, R. C. & Wods, R. E. (2017). *Digital Image Processing* (4 edition). Pearson.