Lab 3 – Quantization and Histogram Manipulation

Goals:

- *Getting familiar with Histogram techniques.*
- *Introduction to video tracking using Camshift algorithm.*

Introduction

In this lab we will practice image color histogram manipulations. Analyzing the color histogram, i.e., the distribution of gray levels, of given images stands in the core of many Image Processing and Computer Vision algorithms. We start by performing several simple operations on image histograms, and then study an example of a well-known video tracking algorithm, *Camshift*, which highly utilizes image histograms.

Preliminary report

Part 1 – Quantization

1) Define image quantization. What is it used for?
2) Write a python function: ***quant_img(img, N)*** which performs **uniform** quantization of the *image* to *N* gray levels ($1 \leq N \leq 256$) and returns a quantized image. **use only *Numpy* functions for this implementation**. Check your code on an image and make sure that it works.
3) Explain the problem of "false contours" that occurs in image quantization. Suggest a method to reduce this artifact.
4) Quantization using K-Means algorithm – read about sklearn.cluster.KMeans function. Give a short explanation about K-means algorithm.

Part 2 – Histogram and Histogram Manipulation

1) Define contrast stretching. What is contrast stretching used for?
2) Define histogram equalization. What is histogram equalization used for?
3) Write a python function: ***my_hist(img)*** which takes a *uint8 image* and returns a 1×256 vector with the corresponding histogram count. **use only *Numpy* for this implementation**.

4) Write a python function: ***my_hist_eq(img)*** which takes a *uint8 image* and returns the *uint8 equalized image*. Use **only *Numpy* and previously implemented functions** in the implementation.
5) Explain why the histogram of an image undergoing histogram equalization is not perfectly equalized (the intensity distribution is not uniform).
6) How will contrast stretching and histogram equalization affect the histogram of the image? What are the similarities and differences between them?
7) Find a grayscale image to be used in the lab and send it to your email. (Hint: to demonstrate the contrast-correcting behavior of the histogram equalization, it is recommended to choose a low-contrast color image.)
8) Send also the python functions **quant_img**(*img*, *N*), ***my_hist(img)*** and ***my_hist_eq(img)*** that you prepared.
9) Consider the *continuous* case of histogram specification:

$$g = S^{-1}(u) = S^{-1}[T(f)]$$

Explain what are g, f, u, S, T.

10) Explain why S^{-1}, T are monotonic. Are they increasing or decreasing?
11) *Figure 1* describes the matching process with *continuous* CDFs. With *discrete* CDFs however, the equation $G(x_i) = H(x_j)$ doesn't necessarily hold.
Suggest a way to overcome this problem. How will you assign the corresponding $G(x_i)$ for a given $H(x_j)$?

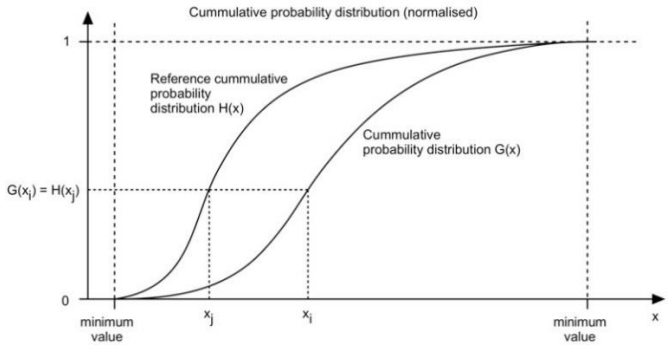

Figure 1- Histogram specification

Hint: How does a CDF of a discrete function looks like? Sketch two discrete CDFs for yourself and try to find $G(x_i)$ which corresponds best to $H(x_j)$.

Part 3: Video Tracking

Intro - Camshift algorithm

One of the common tasks in Computer Vision is tracking an object within a video, that is, finding the location of a moving object in all video frames. *OpenCV* proposed the *Camshift* algorithm [4] for tracking a user-selected object. This algorithm consists of 4 steps:

1) **Initialization**: Create a histogram of the hue (color) component of the selected object. Normalize the histogram to form a probability density function (the "statistical signature" of the object).
2) **Histogram back-projection**: For each pixel in the frame, compare its hue value with the "signature" of the object and find the probability that this pixel belongs to the object. (For instance, if the pixel's hue is equal to 23 and in the "signature" the value 23 corresponds to probability 81%, then the probability of this pixel to belong to the object is equal to 81%.) At the end of this stage, we get a probability map measuring each pixel's probability to belong to our object.
3) **Mean-shift algorithm** [2]: We now wish to find the peak in the probability map derived in the previous step, as it is the most probable location of our object.

   The mean-shift algorithm performs an iterative search in a predefined rectangular window (the "search window"), normally initialized around the location of the object at the previous frame.

   The location of the search window's center of gravity (mean) is then iteratively calculated, and the window is shifted such that it is centered around it. This process stops either when a predefined number of iterations has been reached, or when the displacement of the search window becomes smaller than a predefined threshold. Eventually, the search window is centered at the center of gravity of the peak of the probability map, that is, at the supposed location of the center of object within the frame.

4) **Adaptive adjustment**: Using the statistical moments computed within the search window, compute:
    - the parameters of the rectangle (size, rotation angle) that is centered at the center of gravity and delineates the peak, that is, the supposed object.
    - the size of the "search window" that will be used for the next frame.

   Now display the frame and plot the rectangle on the frame.

   For the next frames we repeat the stages (2-4) of this algorithm.

   For more information, refer to [1].

Questions

1) During the lab, we will run the *Camshift* algorithm on a sample video from the MOT16 (Multiple Object Tracking) dataset [3]. Review the code given in **initialize_camshift_sol.ipynb**. The script displays the first frame of the video with the initial position of the object to be tracked (ROI – Range of Interest). Modify the code so that the blue bounding box captures an object to your liking, which will be tracked (see figure 1). Keep the initial ROI coordinates you found; they will be used to initialize *Camshift* during the experiment. You may prepare several initial ROIs. **Attach your generated images and corresponding ROI coordinates to your preliminary report.**

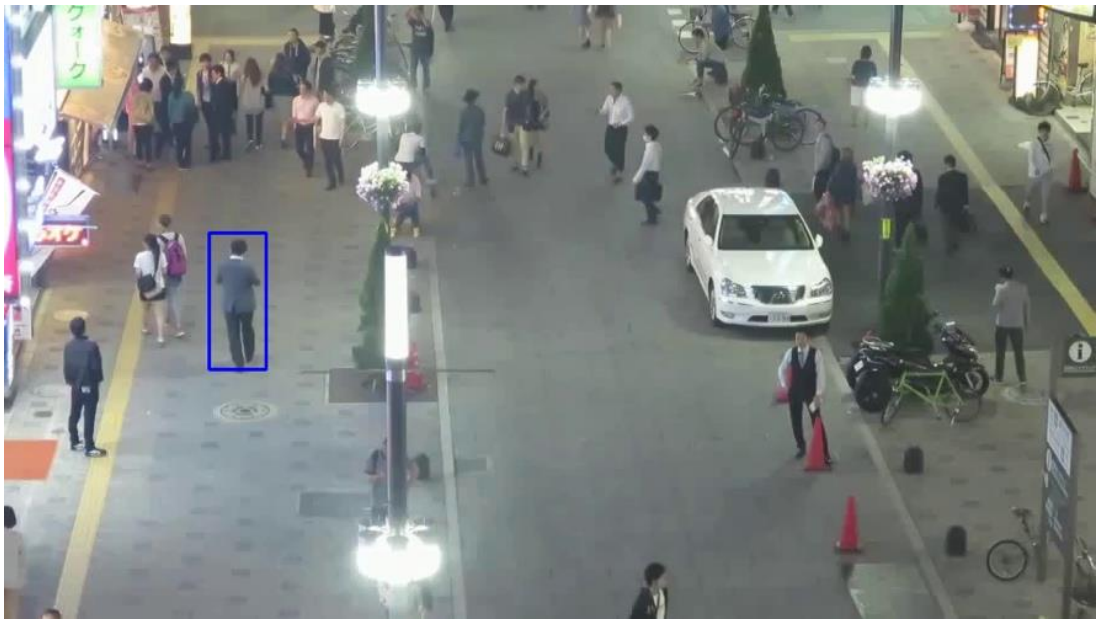

Figure 2: initial frame and ROI bounding box.

Description of the experiment

Open **Lab3.ipynb** in Google Colab, and follow the instructions:

Part 1 – Quantization

1) Complete the missing code and answer the related questions given in the notebook.

Part 2 – Histogram Manipulation

1) Complete the missing code and answer the related questions given in the notebook.
2) Answer the following additional questions:
    a) Suppose the histogram of an image has two sharp peaks. The histogram with two sharp peaks is called bimodal. Explain how bimodal histogram can be used for binarization.
    b) Explain the main difference between contrast stretching and histogram equalization. Which problem may occur in histogram equalization and why doesn't it happen in contrast stretching? (Hint: the problem occurs mostly in smooth background areas).

Part 3 - Video Tracking

1) Complete the missing code and answer the related questions given in the notebook.

Final report

Submit 1 PDF file. Your final report should include your code, outputs, and answers to all questions given in the notebook, as well as in this manual. Make sure you add titles to your figures and axes, and that your report is well organized and clear.

Good luck!

References

[1] G. R. Bradski, "Computer video face tracking for use in a perceptual user interface," *Intel Technology Journal*, Q2, 705–740, 1998, available at http://opencv.jp/opencv-1.0.0_org/docs/papers/camshift.pdf

[2] D. Comaniciu, P. Meer, "Mean shift: a robust approach toward feature space analysis," *IEEE Trans. Pattern Analysis Machine Intell.*, Vol. 24, No. 5, 603-619, 2002, available at https://courses.csail.mit.edu/6.869/handouts/PAMIMeanshift.pdf

[3] MILAN, Anton, et al. MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831, 2016.

[4] https://scikit-image.org/docs/stable/auto_examples/color_exposure/plot_histogram_matching.html