Middle East Technical University | Department of Computer Engineering

CENG 483

Introduction to Computer Vision

Fall 2023-2024

Take-Home Exam 1

Instance Recognition with Color Histograms

Due date: **12 November 2023, 23:59**

1 Objectives

The purpose of this take-home exam is to familiarize yourselves with the simple instance recognition task with color histograms. The assignment is expected to make you gain insight about the computer vision research and evaluation methods.

2 Background

In this assignment you are required to implement an Instance Recognition system based on different types of color histograms and evaluate it with the provided dataset using *top-1 accuracy*. The evaluation results and discussions are required to be reported in an approximately 4 pages long paper using the given LaTeX template.

The text below continues with detailed explanations of the methods and requirements.

2.1 Instance Recognition

Instance recognition is a visual recognition task that involves identifying a specific instance of an object. The goal is to locate additional images of a particular object within a dataset based on a given single image of that object. Although instance recognition is typically accomplished using more advanced techniques nowadays, in this assignment, you will use simple color histograms to measure the similarity of image pairs.

2.2 Color Histogram

A histogram is a vector that counts how many instances of a given property exist in the image. First step of creating a histogram is to define ranges to determine bins, then each instance is assigned into one of these bins. For example, the property in the hand may be the intensity values, then the length of the histogram will be 256 (for values between 0 and 255, with step size 1) for grayscale images and a

histogram is obtained by counting how many instances occur for each bin in the image. It is possible to play and experiment with the interval of quantization (to control number of bins) by defining ranges with different sizes, you are expected to select several different quantization intervals (for example, interval of 2 means 128 bins, interval of 32 means 8 bins for grayscale intensity) and fill in the corresponding parts in your report with your results. More detailed information can be found in the lecture videos.

In this take-home exam, you will be implementing two types of histograms for color values, as explained in the following sections.

2.2.1 Per-Channel Color Histogram

In the per-channel color histogram scheme, you need to compute a histogram for each color channel separately. For a given pair of images, you may first compute the similarity of each histogram separately (explained below) and then take the average of the similarity values.

2.2.2 3D Color Histogram

The color channel histogram can be obtained by first quantizing pixels in each color channel separately and then assigning pixels into a combination of bins from these three histograms. In other words, each combination of quantization in the three separate channels is treated as a single bin in the resulting histogram. For example, if there are 10 bins for each color channel, combining all bins from all histograms results in a histogram with 1000 different bins. For further information, you can check `https://en.wikipedia.org/wiki/Color_histogram`.

2.3 HSV Color Space

HSV (Hue, Saturation, Value) color space is a color representation that separates color information into three components: hue, saturation, and value. Hue represents the dominant color, saturation measures the intensity of the color, and value indicates the brightness of the color. In computer vision, HSV color space can be used for color-based feature extraction since it provides a more intuitive way to describe and manipulate colors compared to the RGB color space. HSV color space is visualized in Fig. 1.

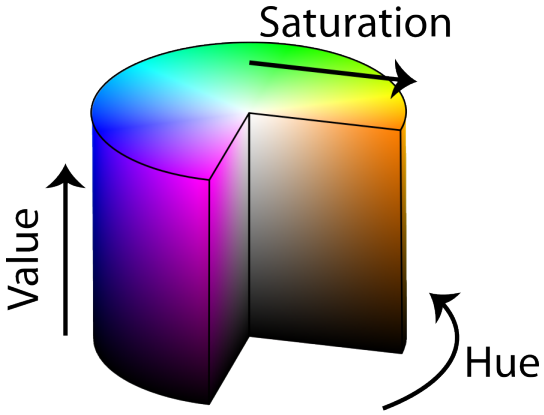

Figure 1: A cylinder representing HSV (Hue, Saturation, Value) color space.[1]

Note that both RGB and HSV color spaces can be used to create color histograms.

Assume that a color is given in RGB space, represented as a tuple (R, G, B) where R, G and B are in range $[0, 1]$ (normalized values). Using the following formulae, this color can be converted to HSV color

[1]Image is cropped from Wikipedia. Licensed under CC BY-SA 3.0.

space. First, the intermediary values $C_{\max}$, $C_{\min}$ and ΔC need be calculated as follows:

$$C_{\max} = \max(R, G, B)$$
$$C_{\min} = \min(R, G, B)$$
$$\Delta C = C_{\max} - C_{\min}$$

Then H, S and V values can be computed using equations (1), (2) and (3) respectively. Note that H, S and V values given by these equations are also in $[0, 1]$ range. You can convert them to $[0, 255]$ range before constructing the color histograms, or quantize the $[0, 1]$ range directly.

$$H = \begin{cases} 0 & \text{if } \Delta C = 0 \\ \frac{1}{6} \cdot \left(\frac{G-B}{\Delta C} \mod 6\right) & \text{if } C_{\max} = R \\ \frac{1}{6} \cdot \left(\frac{B-R}{\Delta C} + 2\right) & \text{if } C_{\max} = G \\ \frac{1}{6} \cdot \left(\frac{R-G}{\Delta C} + 4\right) & \text{if } C_{\max} = B \end{cases} \tag{1}$$

Note that when $\Delta C = 0$, the color is grayscale, and thus the hue value is not meaningful. The formula above assumes $H = 0$ for consistency (to ensure that all grayscale pixels have the same hue), but this is not a strict requirement.

$$S = \begin{cases} 0 & \text{if } V = 0 \\ \frac{\Delta C}{C_{\max}} & \text{if } V > 0 \end{cases} \tag{2}$$

$$V = C_{\max} \tag{3}$$

2.4 Grid Based Feature Extraction

A simple option is to compute histograms globally over the whole image. Alternatively, you can split the image into regions (grid cells) within a regular grid, and compute histograms separately at these cells. You can see an overview of the idea in Fig. 2.

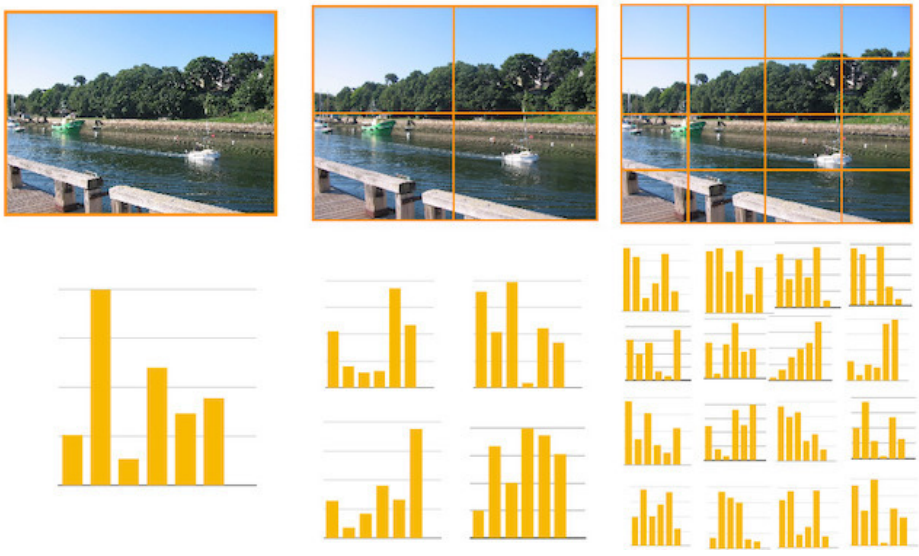

Figure 2: Grid based feature extraction using 1×1, 2×2 and 4×4 grids. In each case, a histogram with 6 bins is computed at each grid cell. Note that the values in the histograms are selected arbitrarily to visualize the concept, so they are not meaningful.

2.5 Computing Image Similarity

Using the schemes described above you will compute multiple histograms for each image. To compute the similarity between two images, you need to measure the similarity of the corresponding histogram pairs across the images. In this assignment, you are required to use **histogram intersection** for this purpose, which is defined as:

$$f(Q, S) = \sum_{i \in \text{NumHistogramBins}} \min\left(Q(i), S(i)\right) \tag{4}$$

where Q and S are both probability distributions corresponding to the query image (the input image whose other images are being searched for) and support image (an image that is being compared against the query image). Note that $f(Q, S) \in [0, 1]$ since both Q and S are probability distributions. Furthermore, $f(Q, S) = 1$ if and only if $Q = S$.

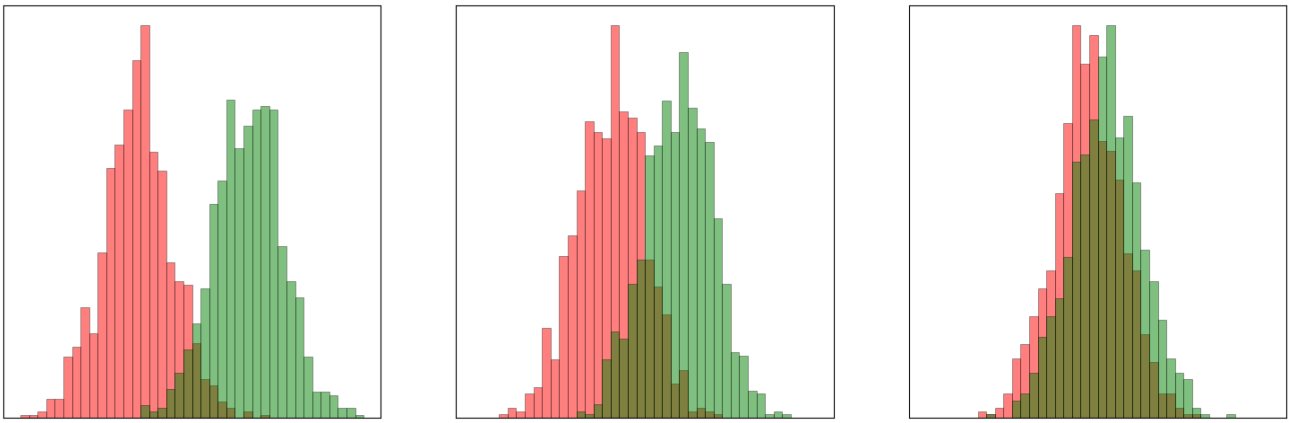

Figure 3: Intersection between the histograms of two probability distributions. The minimum value in each bin yields the intersection for that bin. As the distributions get closer, the intersection area gets larger.

There are a few things that you need to be careful about while using histogram intersection as a similarity metric:

- Histogram intersection in (4) is defined for probability distributions. You can convert a histogram into a categorical probability distribution by applying ℓ_1 *normalization* to the histogram. This basically means dividing each entry in a single histogram by ℓ_1 norm of that histogram. You must do this normalization *before* computing the histogram intersection in (4).
- When calculating per-channel color histogram and/or using a spatial grid, you will be computing multiple histograms per image. In this case, compute histogram intersections between each corresponding pair of histograms, and then take the average of the resulting histogram intersections to calculate the similarity between the images.

2.6 Evaluation Methodology

You are provided four sets of images: 1 support set of images (where you do the search) and 3 different query sets (inputs to your search algorithm). At each one of your experiments, you will be using one of the query sets and a particular configuration of your approach (guidance is provided in the template).

In a particular experiment (with a particular query set), for each query image, you need to search the most similar support image as follows: First, calculate the similarity of the query image to each one of the support images, as described above. Then choose the image with the highest similarity value as the

retrieved image. You need to repeat this process for each query image in the query set. Once this process is completed, compute the top-1 accuracy as follows:

$$\text{accuracy}_{\text{top-1}} = \frac{\text{\# of correctly retrieved images}}{\text{\# of query images}} \tag{5}$$

You can understand whether a retrieved support image is correct or not using the filenames.

2.7 Programming and Interpretation Tasks

You should evaluate your instance recognition system with different configurations by using the provided query datasets. The configurations are given in detail in your report template. Although you are expected to do experiments with these configurations, you are free to add your own configurations as well, just do not forget to mention it in your report and keep additional experiments in a separate section.

Along with the implementation of a Instance Recognition system, you are also submit a report that explains your work. A template is given to you, and reports in any other format will not be accepted. **In your report you are expected to discuss and highlight relative strengths and weaknesses of certain configurations on each of the query datasets (query1, query2 and query3) and explain why these can be the case. Focus on the top performing and worst performing configurations, try to explain why that is the case conscisely.**

2.8 Support and Query Datasets

Each image set contains 200 images of size 96×96. Image names in every dataset is same and they will be given to you as a txt file. You are expected to match images with same names in query datasets with their support counterparts. Each query dataset consists of images from support set with various transformations applied on top of them. The report will be based on the observations in the experiments for these queries. **Investigate the query sets to understand what kind transformations applied to them, which is crucial for the discussions that you are expected to present in the report.**

3 Restrictions and Tips

- Your implementation should be in Python 3.
- You must solely use `numpy` to implement the core homework tasks. You can use other tools to convert images into `numpy` arrays. But in the rest of the implementation you should use `numpy`.
- **Histogram, RGB to HSV conversion, grid based feature extraction, and instance recognition** implementations must be of your own.
- You **can not use np.histogram**.
- You are not allowed to use any built-in method to convert images to HSV color space. You must implement RGB to HSV conversion in `numpy` according to equations (1), (2) and (3).
- Do not use any available Python repository files without referring to them in your report.
- Note that the code you are going to submit will also be subject to manual inspection.
- Coming up with a correct implementation is important. We may randomly check for random configurations.
- $N \times N$ grid means a spatial grid of N^2 cells. For example, consider an image that has 96×96 pixels. A 12×12 grid of that image will have 144 cells each of which consist of 9×9 pixels.

- It is part of the challenge to implement the pipeline efficiently, for which you probably want to (i) leverage broadcasting in `numpy` (as explained early in the semester), which is typically much faster than naive for loops where possible, and (ii) cache features, i.e. don't re-extract them from scratch every time you process an image.

4 Submission

- **Late Submission:** As in the syllabus.
- You are expected to upload 2 files to ODTUClass:
  1. `code.tar.gz`: A compressed archive file of your implementation. Alongside the code, it should contain a README file that includes instructions on how to run the experiments. Feel free to create subdirectories, modules, configuration files, etc. to organize your code. However, the archive must contain no directories on top of the README file, i.e., when extracted with `tar xzfv code.tar.gz`, the README file must appear in the current directory.
  2. `report.pdf`: An approximately 4 pages long report based on the provided template.
- Do not include the dataset or any unmentioned files in your submission.

5 Regulations

1. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations.
2. **Newsgroup:** You must follow the course web page and ODTÜCLASS (odtuclass.metu.edu.tr) for discussions and possible updates on a daily basis.