CM 30080: Computer Vision

2020 Coursework: Filtering & Object Recognition

Submission

This coursework is worth 25 points from the overall mark of 100 for this unit. Marks are given beside each task. A report will be your main method of assessment. Students should form groups of 2 individuals and submit one report per group. Students can choose their partners until Feb 21st. Each group will receive a total mark for this course work and individuals in that group will share this mark (with weights) according to their contributions.

The submission deadline for your report is:
2nd April 2020 18:00: online Moodle submission of your final report

The main part of your report should not exceed 3000 word limit. After the main part please attach a Table of individuals' contributions and appendices including ONLY the codes (these are excluded from the word limit). The report should be submitted in PDF format. Table of contributions should include both students' names/university IDs, the list of contributions of each student, and finally the contribution percentage for each student. This percentage should be agreed between both individuals and it will be used to weigh their marks. We highly encourage individuals to evenly share the workload 50%-50%. Further, you should include all codes relevant to your implementations as an appendix or appendices with a clear referencing to the main body of your report. Codes must be well commented (indicating major steps).

First and foremost your report should demonstrate and evaluate your results. It must include figures and screenshots with appropriate resolutions. Evaluation can be qualitative (how it looks) and, wherever possible, quantitative (tested numerically). Second you should provide evidence that you understood the mathematics behind the assignment in each task. You should concisely explain your approach to solve each task, the choices you make (e.g. hyper-parameters etc), and answer questions in each part.

Other Details

You may implement the coursework in any language, but you are encouraged to use MATLAB if you are not familiar with another language's image processing and linear algebra tools.
Usual university rules apply, please check your undergraduate handbook, particularly regarding plagiarism and delayed deliveries.

Introduction

In this lab you will implement algorithms for matching images, similar to the principle of playing Dobble. This will be split into two main tasks, 1. Implementing 2D Image convolution, and 2. Intensity-based template matching. A Training dataset of images (icons) as shown in Figure 1 and a Testing dataset (various combinations) as shown in Figure 2 will be provided. The icons from the Training dataset have to be recognised and matched in Test dataset provided. You should also have completed the MATLAB skills lab before starting these tasks.

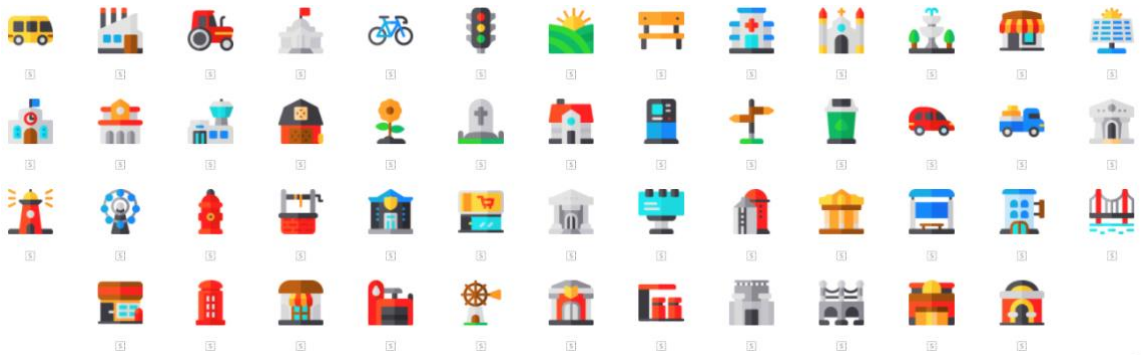

Figure 1 Training images: icons of the village dataset

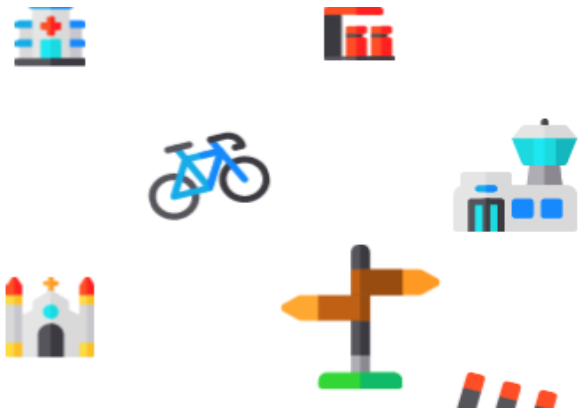

Figure 2 One of the test images (i.e. card patterns) to be used for recognition

Task 1: Image convolution (worth 7 pts)

Write a function which takes an RGB image and a kernel/filter as inputs, and computes the convolution of the two according to the double summation convolution formula in the lectures.

Output 1: Your function should output an RGB image of the same size as the original, although you may offer other options if you wish. You should treat areas outside of the image as zero, but again you may let the user select other options. Demonstrate the consistency of your function's outputs to the in-built conv2 function of MATLAB (or an equivalent function in another language).
Next, choose an image(s) from the Training dataset and test your function with two types of kernels: 1. blurs of different sizes, 2. horizontal, vertical and diagonal edge detectors in different directions. Demonstrate the outcomes and explain why the chosen kernels are appropriate for each task.

Task 2: Intensity-based template matching (worth 18 pts)

Write a function which takes an image from Test dataset and outputs the following:

Output 2: Detect objects in the Test images, recognize to which class (from the 50 Training image classes) they belong, and identify their scales and orientations. For visual demonstration the function should open a box around each detected object and indicate its class label. This box is scaled and rotated according to the object's scale and orientation. Demonstrate example images(s) of the outcome detection in you report.

Evaluate your algorithm on all Test images and report the overall False Positive (FP), True Positive (TP) and accuracy (ACC) rates, and the average runtime. Show and explain cases where this scheme finds difficulty to perform correctly. Give insight about the complexity (memory and runtime) of your algorithm in terms of the problem size e.g. how does the complexity scale in terms of the number of image pixels, pyramid scales and rotations?

Creating scaled & rotated templates: You should create a Gaussian pyramid of appropriate scale levels for each RGB Training image. This task includes implementing appropriate Gaussian blurs and subsampling of the training images through a hierarchy. You may use your own convolution algorithm to apply the Gaussian blur, or you may use the faster in-built version. In your report please explain the steps you take to build your Gaussian pyramids and show an example pyramid for a Training image.

After creating scaled templates, you should append this set by creating Gaussian pyramids for an appropriate number of orientations per class. This creates the overall scaled & rotated templates. Justify your choice of parameters (e.g. Gaussian kernel specs, initial/final scales, number of scale levels e.g. how many octaves, number of rotations etc). To justify the number of scales and rotations in your report plot the overall object detection performance against these parameters and pick parameters that achieve a reasonably good runtime vs. accuracy trade off.

Pre-processing: For each (scaled and rotated) template set the background to zero, similarly for the Test data.

Intensity-based matching (related to output 2): Slide each (scaled and rotated) template over the Test image to measure their similarities across x-y axes. An intensity-based similarity score could use the in-built normxcorr2 function in MATLAB (or an equivalent function in another language). Explain why this function is suitable to measure image similarities?
Define appropriate thresholds on this similarity measure, a proper non-maxima suppression strategy to avoid false positives within the neighbourhood of the detected objects, and return the **output 2.1** in the specified format.

Side notes: You're allowed to use in-built MATLAB functions such as imrotate,conv2/imfilt (in task 2) and possibly many more, as long as they don't conflict with the tasks you're asked to implement. An example could be using default Imresize function to build Gaussian pyramids which contradicts the specific guideline on how to do resizing using scaled Gaussian blurs. Another example, you are asked to open a box (localisation) around the detected object with proper scale and rotation. This box function (does not matter if it is in-built or written by you) should get the Location, Scale, Rotation of the object from *your* detection algorithm and not running another in-built detection algorithm in parallel.