Queen Mary
University of London

Introduction to Computer Vision

Coursework

For question 1) and 2) use Dataset A; for question 3) use Dataset B; for question 4) and 5) use Dataset C; for question 6) use Dataset A and Dataset B.

1) **Transformation.** Rotation, translation and skew are useful operations for matching and tracking. Write a function that takes as input an image *I*, rotates it with an angle θ_1 and horizontally skews it with a second angle, θ_2. Write the matrix formulation for image rotation and skewing (define all the variables).

   a) Create an image containing your name written in Arial, point 72, capital letters. Rotate the image you created by $\theta = 30$, $\theta = 60$, $\theta = 120$ and $\theta = -50$ degrees clockwise. Skew the same image by $\theta = 10$, $\theta = 40$ and $\theta = 60$ degrees. Comment the results.

   b) Analyse the results when you change the order of the two operators i.e. R(S(*I*)) and S(R(*I*)), where R is the rotation and S is the skew. Are the results of (i) and (ii) the same? Why?
      i) Rotate the image by $\theta_1 = 20$ clockwise and then skew the result of $\theta_2 = 50$.
      ii) Skew the image of $\theta_2 = 50$ and then rotate the result by $\theta_1 = 20$ clockwise.

2) **Filtering and Convolution.** Image filtering allows you to apply various effects on images. In this assignment you will focus on filtering in the spatial domain, also known as convolution. Convolution provides a way of multiplying together two arrays of numbers, to produce a third array of numbers. In the image processing context, one of the input arrays is normally a grey-level image. The second array is usually much smaller, and is also two-dimensional (matrix), and is known as the convolution kernel. Depending on the designed filter and intended effect, the kernel can be a square matrix, e.g., of 3x3, 5x5 or 7x7 dimensions.
   Write a filtering function that takes an input image, performs convolution using a given kernel, and returns the resulting image.

   a) Design a convolution kernel that enables the computation of average intensity value in a 3x3 region for each pixel. Use this kernel and the filtering function above, save the resulting image.

   b) Use the kernels provided below, apply the filtering function and save the resulting images.

kernel A		
1	2	1
2	4	2
1	2	1

kernel B		
0	1	0
1	-4	1
0	1	0

   Comment on the effect each kernel has on the input image.

   c) Use the filtering function for the following filtering operations: (i) A followed by A; (ii) A followed by B; (iii) B followed by A. Comment the results.

   d) Keeping the effect of the kernels in b) the same, discuss how to extend them to larger filter kernels 5x5 and 7x7. Using the extended Kernels and repeat the operations in c). Comment on the results and compare them with the ones obtained in c).

3) **Histogram.** A colour histogram is a representation of the distribution of colours in an image. Given a discrete predefined colour space (e.g. red, green, blue), the colour histogram is obtained by counting the number of times each colour occurs in the image array.
Histogram Intersection is a technique that can be used to match a pair of histograms. Given a pair of histograms, e.g., of an input image ***I*** and a model ***M***, each containing **n** bins, the intersection of the histograms is defined to be $\sum_{j=1}^{n} \mathbf{min}(I_j, M_j)$ where **j** ranges over each colour in the histograms. The result of the intersection of a model histogram with an image histogram is the number of pixels from the model that have corresponding pixels of the same colour in the image.

a) Write a histogram function that returns the colour histogram of an input image. Visualize the RGB histogram and save the figure(s). For a given video sequence, use the above function to construct and visualise the histograms of each frame.

b) Write a function that returns the intersection of a pair of histograms. For a given video sequence, use the histogram intersection function to calculate the intersection between the consecutive frames, e.g., between I_t and I_{t+1}, between I_{t+1} and I_{t+2} etc. Visualise and present the intersection values. Find a way to normalize the intersection. Does that cause any changes in the results?

c) What does the intersection value represent for a given input video? Can you use it to make a decision about where the scene in the video changes? How robust is the histogram intersection technique to changes? Where does it fail? What would be other application areas where histogram calculation and histogram intersection can be used?

4) **Motion estimation.** Block matching is a commonly used motion estimation approach. Block matching compares each block of a frame with blocks in a past or future frame in order to find the corresponding block (i.e. the block minimising an error measure, such as the Mean Square Error or the Mean Absolute Error). The search is usually restricted to a window centred on the position of the target block in the current frame. The search window defines an upper limit on how fast objects can move between frames (maximum displacement), if they are to be coded effectively.
Given two consecutive frames of a video sequence (I_t and I_{t+1}), implement a full-search block-matching algorithm that uses different block dimensions. The output of the algorithm will be (i) the motion field and (ii) the image P_{t+1}, the prediction of I_{t+1}.

a) Display the motion vectors as a set of arrows starting from the centres of the blocks and superimposed on I_{t+1}. Use a block size of 16x16 pixels and a search window of 20x20 pixels.

b) Based on the motion vectors you have estimated, display the image P_{t+1}, predicted version of I_{t+1}, using a block size of 16x16 pixels and a search window of 20x20 pixels.

c) Use a 16x16 search window and block sizes of 4x4, 8x8 and 16x16 pixels. Display the results (you can zoom on a portion of the image) and comment the differences between the results.

d) Use an 8x8 block size and search windows of 8x8, 16x16, and 32x32 pixels. Display the results (you can zoom on a portion of the image) and comment the differences between the results.

e) Plot the execution time of your software (i) against the variation of the block size and (ii) against the variation of the search window. Comment the results.

5) **Objects.** Moving objects captured by fixed cameras are of great importance in several computer vision applications.

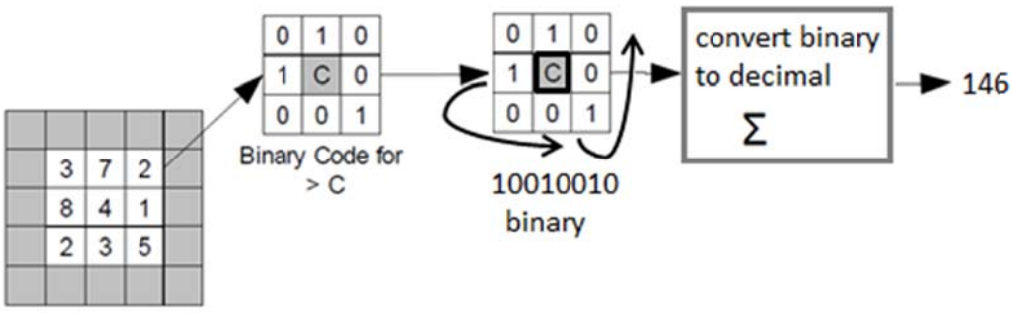
a) Write a function that performs pixel-by-pixel frame differencing using the first frame of an image sequence as reference frame. Apply a classification threshold and save the results to disk.

b) Repeat the exercise using the previous frame as reference frame (use frame I_{t-1} as reference frame for frame I_t, for each t). Comment the results.

c) Write a function that generates a reference frame (background) for the sequence using for example frame differencing and a weighted temporal averaging algorithm.

d) Write a function that automatically counts the number of moving objects in each frame of a sequence. Generate a bar plot that visualizes the number of objects for each frame of the sequence.

6) **Texture:** Local binary pattern (LBP) is a feature used for texture representation and classification. The LBP operator describes the surroundings of a pixel by generating a bit-code from the binary derivatives of a pixel. The operator is usually applied to greyscale images and the derivative of the intensities. An example of how the LBP in its simplest form works is illustrated in the figure below. Note how a pixel with a value > C is assigned '1' and a pixel with a value < C is assigned '0'.



a) Write a function that divides an input (greyscale) image into equally sized non-overlapping windows, and returns the feature descriptor for each window as distribution of LBP codes. For each pixel in the window, compare the pixel to each of its 8 neighbours (on its left-top, left-middle, left-bottom, right-top, etc.). Follow the pixels in a counter-clockwise manner. Compute the histogram over the window - as the frequency of each "number" occurring. Normalize the histogram. The histogram is now a feature descriptor representing the window at hand.

b) Come up with a *descriptor* to represent the whole image as consisting of multiple windows. *Hint:* Think of combining several local descriptions into a global description.

c) Using the global descriptor you have created, implement a classification process to classify the images in the provided dataset into two categories: face images and car images. Comment the results, include images if necessary. *Hint:* You can use simple histogram similarities. Is the global descriptor able/unable to represent whole images of different types, e.g. faces vs. cars? Identify problems (if any). Suggest solutions if possible, include images if necessary.

d) Decrease the window size and perform classification again. Comment the results, include images if necessary.

e) Increase the window size and perform classification again. Comment the results, include images if necessary.

f) Discuss how LBP can be used/modified for dynamic texture analysis, e.g. in a video.