CS-512 – Assignment 2

Filtering and feature detection

Due by: February 20, 2023

Review questions

Answer the following questions. Make sure that your answers are concise. In questions requiring explanation, make sure your explanation is brief.

1. Noise and filtering

   (a) Explain how to estimate the signal to noise ratio (SNR) in an image.

   (b) Explain the difference between Gaussian and impulsive noise. Which filter handles better impulsive noise: an averaging filter or a median filter.

   (c) Given an image having the value of 2 in each cell, write the value of the pixels in this image after applying a 3×3 convolution filter having all 1-s in its entries.

   (d) Given that we need the derivative of an image convolved with a filter explain how the operation can be applied more efficiently.

   (e) Explain the three different ways to handle boundaries during convolution.

   (f) Write a basic 3×3 smoothing filter. What is the sum of all entries in this filter? Explain the reason for the sum to be selected as it is.

   (g) Explain how to implement a 2D convolution with a Gaussian using two 1D convolution filters. Which option is more efficient? Is it possible to implement any 2D filter in this way?

   (h) Given a 1D Gaussian filter with $\sigma = 2$, what should be the size of this filter?

   (i) Explain how a Gaussian image pyramid is produced. What is the reason for producing such pyramids? What is the amount of additional processing done in a pyramid compared with a single image?

   (j) Explain how the Laplacian pyramid is produced and its use.

2. Edge detection

   (a) Why is edge detection useful? What are the desired properties of edge detection?

   (b) Explain the basic steps of edge detection and the need for them: smoothing, enhancement, localization.

   (c) Describe two filters for computing the image gradient. What is the meaning of the image gradient? What is it used for?

   (d) Explain how the Sobel filter can be produced from a smoothing and derivative filters.

   (e) Explain how to generate a more accurate derivative filter with an arbitrary σ. Write the elements of a filter for more accurate derivative computation with $\sigma = 2$.

(f) Explain how an edge can be localized using the first or second order derivative of the image.

(g) Let $\sigma = 1$. Write the Laplacian of Gaussian (LOG) filter using this σ. Explain how to use LOG to detect edges.

(h) Explain the main difference between the Canny edge detection algorithm and a standard edge detection that does not use directional derivatives. What is the condition for detecting an edge candidate in Canny?

(i) Explain the non-maximum suppression and hysteresis thresholding parts of the Canny algorithm.

3. Corner detection

(a) Explain the basic principle of corner detection. How is the number of principal directions assessed?

(b) Explain how PCA (Principal Component Analysis) is used to find principal directions of gradient orientations in a local patch.

(c) Given the gradient vectors $\{(0,0),(0,1),(0,2),(0,3),(0,4),(1,0),(1,1),(1,2),(1,3)\}$ compute the correlation matrix that can be used for corner detection.

(d) Write the condition on the eigenvalues of the gradient correlation matrix that is used for corner detection.

(e) Explain how non-maximum suppression works for corner detection.

(f) Explain how Harris corner detection avoids computing the eigenvalues of the gradient correlation matrix directly.

(g) Write and explain the formula for computing better localization of a corner. What is the condition for the solution in the formula to exist?

(h) Explain how feature points can be characterized using HOG (Histogram of Oriented Gradients). What are the requirements from a good characterization of feature points?

(i) Explain how SIFT features are computed.

Programming questions

In this part you need to write a program (in Python using OpenCV) to perform filtering and feature detection operations. The program should load an image and convert it to grayscale. Whenever your implementation is slow (e.g. due to double loops) accelerate it using Cython. The program should work on any size image. Make sure to properly evaluate the correctness of your implementation.

1. Smooth an image using the OpenCV `filter2D` function and show the resulting image. Use a track bar to control the amount of smoothing.

2. Repeat the previous step using your own implementation of a convolution function with a suitable filter. In this question (only) you must implement the convolution operation yourself. Make sure to use Cython to speedup your implementation.

3. Down-sample the image by a factor of 2 with smoothing (before down-sampling) and show it.

4. Up-sample the image from the previous step by a factor of 2 with smoothing (after up-sampling). Show the resulting image and the difference from the original image. When showing the difference, normalize difference values if necessary so that they are visible.

5. Compute the x- and y-derivatives of an image and display the resulting images. When displaying the derivative images, normalize the obtained values if necessary so that derivative values are visible.

6. Compute the magnitude of the image gradient you computed in the previous step, and display it. Normalize the displayed values as needed.

7. Plot the image gradient vectors on top of the original image every N pixels (e.g. using short red lines segments of length K) and display the resulting image. Use a track bar to control N.

8. Detect and display corners in an image using the OpenCV Harris corner detection function (`cornerHarris`). Show the results using red dots. Use a track bar to control the number of control points shown.

9. Scan the image with a window. At each window location, compute the correlation matrix and its eigen values. Multiply the eigenvalues you computed to get a "cornerness" measure. Display the "cornerness" measure making sure to normalize values as needed so that the measure is visible.

10. Train a simple convolutional network (code provided) to perform MNIST digit classification and report the accuracy you receive on a validation set. Smooth the images (thus degrading them), train the network, evaluate results, and report accuracy. Repeat the smoothing several times (thus increasingly degrading the images) and report the accuracy.

11. Repeat the previous step, except that now you compute the magnitude of image gradients before feeding them into the network. Compare the results in this step to the results from the previous step.

Submission instructions

1. Write a report (using the sample report of assignment 1) summarizing the results you obtained in your program. Make sure to evaluate in your report the results you obtain and demonstrate their correctness. Use both simple images where the expected result is clear and more complex images.

2. Follow the submission instructions of assignment 1.