EE6310 Image and Video Processing, Fall 2021

Indian Institute of Technology Hyderabad

Homework 6, Assigned 19.11.2021, Due **11:59 pm on 28.11.2021**

There are two possible outcomes: if the result confirms the hypothesis, then you've made a measurement. If the result is contrary to the hypothesis, then you've made a discovery. – Enrico Fermi

Instructions:

- For the first question, use the images *first_frame.png* and *second_frame.png* posted along with this HW.
- For the remaining questions, use *lighthouse.png* that is also posted with this HW.
- Please turn in Python Notebooks with the following notation for the file name: `your-roll-number`-hw6.ipynb.

1 Motion Estimation

In this problem you will implement the most critical part of the video codec – the motion estimator. Do the following:

1. Use the 3-step search to find motion vectors (check slides for description). (10)
   - Use mean absolute distance (MAD) as your metric.
   - Step 1: Search at 8 location ± 4 pixels around current macroblock including (0, 0) (relative to current macroblock).
   - Step 2: search at 8 location ± 2 pixels around best match location in Step 1 including best match location.
   - Step 3: search at 8 location ± 1 pixels around best match location in Step 2 including best match location.
2. Plot the motion vector at each macroblock. You can use the *arrow* function in *matplotlib*. (1)
3. Generate the motion compensated predicted frame using the motion vectors and the first frame. (3)
4. Compute the error between the second frame and its motion compensated predicted version and display it. (1)

   Instructions:

   - Do not use built-in functions.
   - Divide each frame into *non-overlapping macroblocks* of size 16×16 pixels. Note that the images are of size 176×144.
   - Generate motion vectors at each macroblock in the *second* frame from the *first*.

2 SSIM index

Write a program to implement the SSIM index. Your program should also display the SSIM map. Generate test images by adding noise, blurring and compressing the reference image (all separately) (10).

3 Edge Detection

In this problem, you will write a program to compute the edge map of an image. See Remarks for image.

3.1 Gradient Edge Detectors

1. Use the following gradient operators:
   (a) Centered 2-D differencing. (2)
   (b) Roberts operator. (2)
   (c) Prewitt operator. (2)
   (d) Sobel operator. (2)
2. Estimate gradient magnitude using the following definitions:
   (a) $M(i,j) = \sqrt{\Delta_x^2(i,j) + \Delta_y^2(i,j)}$ (2)
   (b) $M(i,j) = |\Delta_x(i,j)| + |\Delta_y(i,j)|$ (2)
   (c) $M(i,j) = \max\{|\Delta_x(i,j)|, |\Delta_y(i,j)|\}$ (2)
3. Threshold the magnitude map using an empirical threshold τ to find the edge map E. (1)

3.2 Laplacian Edge Detectors

1. Compute the Laplacian using the convolution template (2)

$$\begin{bmatrix} 0 & +1 & 0 \\ +1 & -4 & +1 \\ 0 & +1 & 0 \end{bmatrix}$$

2. Compute the edge map E as the output of a zero crossing detector. (3)

3.3 Laplacian of Gaussian (LoG)

LoG was motivated by the sensitivity of gradient and Laplacian edge detectors to noise. In this problem, you will implement an edge detector using the LoG operator. Work with the *lighthouse.png* image that is corrupted with AWGN whose $\sigma_n = 10$. Experiment with different values of σ for the Gaussian pre-filter and compute the edge map E after zero crossing detection. Compare your result with gradient based techniques implemented above and verify the robustness of LoG. (5)