CS 4/510: Computer Vision & Deep Learning

Programming Assignment #2

A. Rhodes

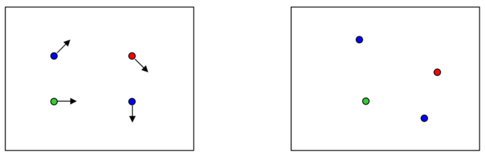
Note: This assignment is **due by Sunday, 7/17 at 1000pm**; you will turn in the assignment by email to our TA.

Exercise #2: Optical Flow

Dataset: Two pairs of images from video clips: frame1_a.png, frame1_b.png and frame2_a.png and frame2_b.png.

This exercise requires you to implement Lucas-Kanade Optical Flow method (by hand).

Recall that optical flow aims to estimate motion from one frame to the next in a video sequence.



$I(x,y,t)$ $I(x,y,t+\Delta t)$

If your image is not already grayscale (i.e. one channel), convert it to grayscale. For each pair of images: (frame_1, frame_2), we want to compute optical flow, a 2D vector (V_x,V_y) with respect to each pixel (x,y) in frame_1. To do this, we solve the following system of equations for each pixel (x,y) in frame_1 for V_x and V_y:

$$\underbrace{\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_9) & I_y(p_9) \end{bmatrix}}_{9\times 2} \underbrace{\begin{bmatrix} V_x \\ V_y \end{bmatrix}}_{2\times 1} = -\underbrace{\begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_9) \end{bmatrix}}_{9\times 1}$$

where $p_1,\ldots,p_9$ is a set of nine points surrounding the central pixel (x,y) in a 3x3 grid:

p_1	p_2	p_3
p_4	$p_5=(x,y)$	p_6
p_7	p_8	p_9

Above, I_x and I_y (on the left-hand side of the system of equations) denote the result of convolving the input image (frame_1) with the x and y gradient filters, respectively:

$$g_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, g_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Note that you should zero-pad (width=1) your input image, so you are able to perform convolution over pixels on the periphery of the image.

The notation $I_t(p)$ above (right-hand side of system of equations) stands for the "temporal" derivative. To estimate this we simple take the difference of the image intensities between the frames: $I(\text{frame}_2) - I(\text{frame}_1)$ where $I(\bullet)$ denotes pixel intensity; so then $I_t(p)$ is the difference in pixel intensities between the frames with respect to pixel *p*.

Finally, <u>for each pixel</u> in frame_1 we can generate the 9x2 system of equations shown above. Next, we leverage OLS to solve each 9x2 system:

$$\rightarrow \underbrace{\begin{bmatrix} V_x \\ V_y \end{bmatrix}}_{x} = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

Giving us V_x and V_y , the x and y components of optical flow, respectively, for each pixel in the original image, as desired.

For each pair of images provided on Canvas, calculate V_x and V_y and visualize each of these values (in their own 2D image, over all pixels); also include a 2D visualization of

$$\sqrt{(V_x)^2 + (V_y)^2}.$$

Report: Your report should include a short description of your experiments, along with the plots and discussion paragraphs requested above and any other relevant information to help shed light on your approach and results.

Here is what you need to turn in:

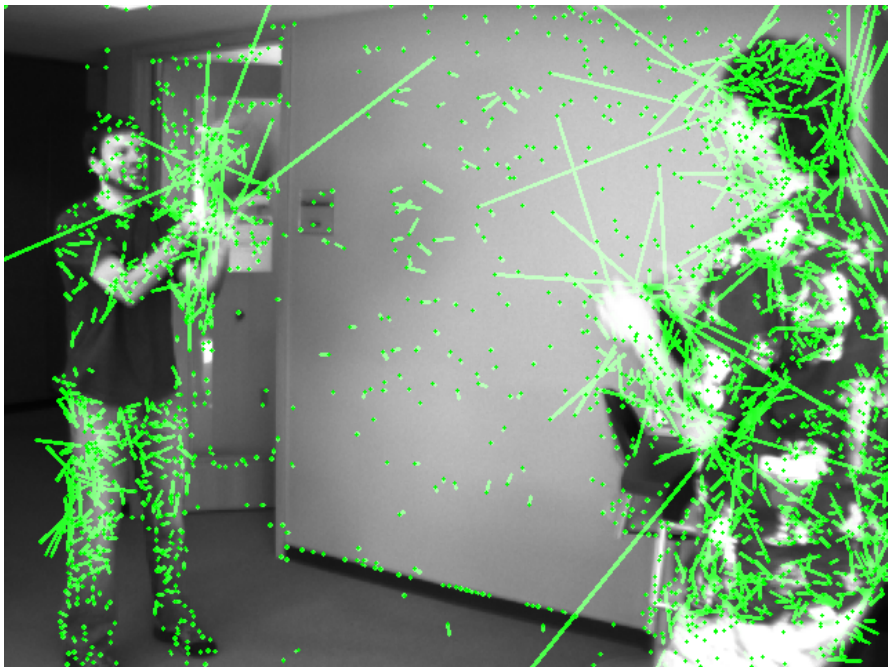
- Your report.
- Readable code.

Report : -

- This experiment contains 2 collections of images which are given to detect the optical flow. The optical flow works on the basis of brightness intensity detection and constraints that during frames moving objects have the same brightness intensity as per the start of the frame.
- Firstly, in this program, images are loaded and converted into grayscale, with padding added for the further process of Lucas-Kanade Optical Flow. Firstly, the derivative of x using gx and derivative of y using gy for the first image is created then intensity difference between two frames is calculated. Furthermore, the matrix calculation of inverse matrix and right side matrix is completed using numpy. This at the end, gives the optical flow as vx and vy which are used and significantly changing points are selected to draw over the image. This mask of drawing contains points and lines.
- Gained output displays the motion points in green. Some points do not significantly display the motion but the better improved model and algorithm can be developed.

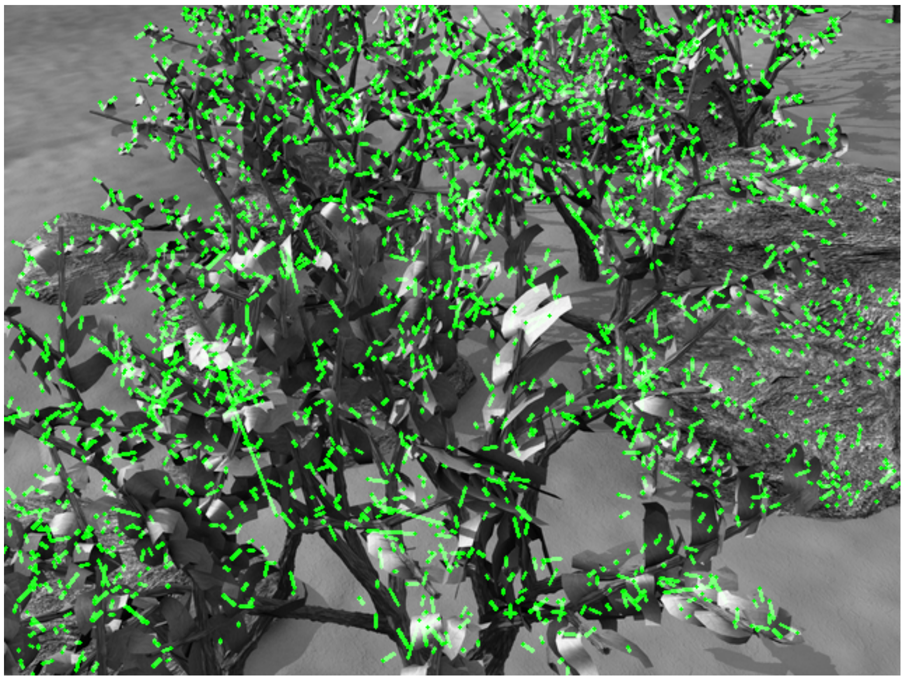
Output : -

Frame Collection 1



Square root form

Frame Collection 2



Square root form