Shreyash Agrawal
M.Tech CSE
2021201074
Computer Vision Assignment-2 Solutions

1.1 (1) DESCRIBE HOW OPTICAL FLOW COULD BE USED TO CREATE A SLOW-MOTION VIDEO

The Optical Flow Slow Motion effect delivers the most effective, smooth slow motion possible.

By using the optical flow, we are basically interpolating all missing frames. We use motion estimation to generate new frames from the source. This is done by analyzing and comparing pixel data from surrounding frames, and then predicting and creating what would come there.

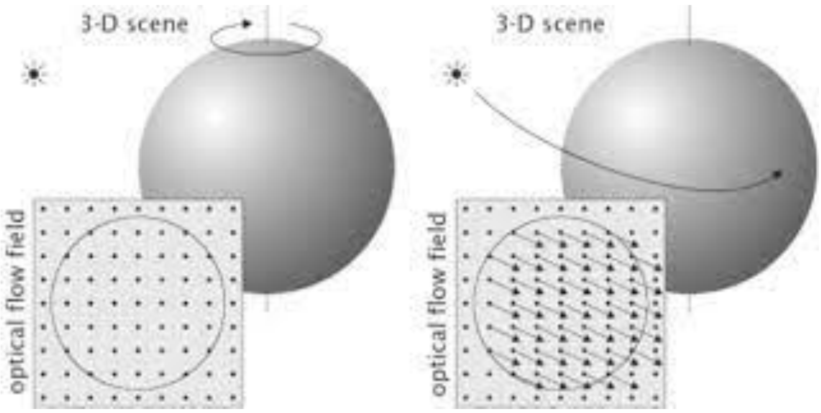
1.1 (2) EXPLAIN BRIEFLY HOW OPTICAL FLOW IS USED HERE

The effect shown in matrix was named as the 'Bullet time' Effect. It makes use of temporal motion, so the scene progresses in slow and variable motion. The visual effect team, brilliantly make use of the timings in which each camera is used to record the scene with a fraction of second difference of each other, so that all the cameras could record the scene as it progressed. This created a slow-motion effect when all these frames were put up together. Also, some other cameras were used to film the scenes with normal frame rate. Now these together produced the award-winning scene.

1.1 (3) CONSIDER A LAMBERTIAN BALL THAT IS: (I) ROTATING ABOUT ITS AXIS IN 3D UNDER CONSTANT ILLUMINATION AND (II) STATIONARY AND A MOVING LIGHT SOURCE. WHAT DO THE 2D MOTION FIELD AND OPTICAL FLOW LOOK LIKE IN BOTH CASES?

(I) In 1st case since only the ball is moving, its 2d motion field will be that of rotatory motion. Since the surface of this ball is uniform all over, thus we will observe that the optical flow will be zero.

(II) In 2nd case since the light source is moving, the 2d motion field will be zero. Since the light source is moving, the optical flow will be observed in the direction of the motion of the light source.



1.2 (1) WHAT DOES THE OBJECTIVE FUNCTION IMPLY ABOUT THE NOISE DISTRIBUTION?

The objective function of the classical optical flow problem is to estimate the optical flow components *(u, v)* which minimize the error between the estimate and the actual values. Considering the image as a function of position and time, *I(x, y, t)*, then according to the brightness constancy assumption, the intensity of the image should not change for tiny steps in position and time

$$f(x+\delta x) = f(x) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f^2}{\partial x^2}\frac{\delta x^2}{2!} + \ldots + \frac{\partial f^n}{\partial x^n}\frac{\delta x^n}{n!}$$

If δx is small,

$$f(x+\delta x) = f(x) + \frac{\partial f}{\partial x}\delta x + \boxed{0(\delta x^2)} \longrightarrow \text{Almost zero}$$

For a function of three variables with small δx, δy and δt ,

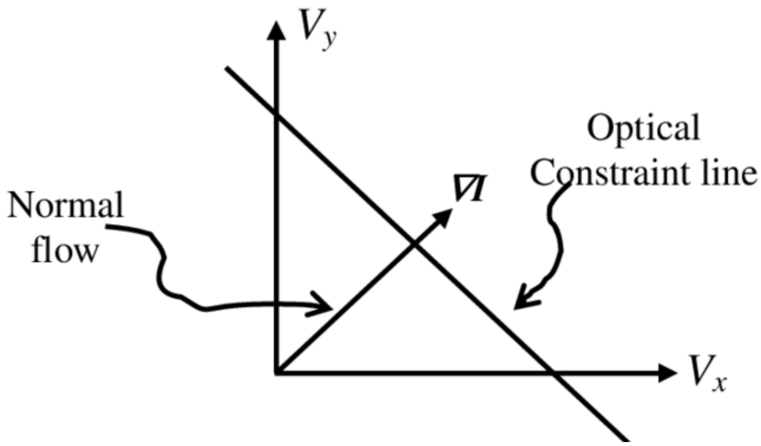
$$f(x+\delta x, y+\delta y, t+\delta t) \approx f(x,y,t) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial y}\delta y + \frac{\partial f}{\partial t}\delta t$$

1.2 (2) IN OPTIMIZATION, WHY IS THE FIRST-ORDER TAYLOR SERIES APPROXIMATION DONE?

The Taylor series provides an approximation or series expansion for a function. Some reasons are as follows: -

- The math tends to work out more simply with a linear approximation.
- in image processing, high order derivatives are corrupted by noise and virtually unusable.
- In order to minimize computation time, use the smallest order Taylor polynomial for which the approximation error is acceptably small.

1.2 (3) Geometrically show how the optical flow constraint equation is ill-posed. Also, draw the normal flow clearly



Ill-poised because, if we consider a particular pixel and then calculate the above equation, we will not get the actual flow information because there are infinitely many values of u and v which satisfy the equation.

2.3 (1) Why is optical flow only valid in the regions where local structure-tensor $A^T A$ has a rank of 2? What role does the threshold (tau) play here?

A^T A needs to have rank 2 in order to stay invertible. Being invertible is a primal condition under the defined constraints in order to be solvable to return direction vectors (u,v).

$$\underbrace{\begin{bmatrix} I_x(1,1) & I_y(1,1) \\ I_x(k,l) & I_y(k,l) \\ \vdots & \vdots \\ I_x(n,n) & I_y(n,n) \end{bmatrix}}_{\substack{\text{Known} \\ A \\ n^2 \times 2}} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\substack{\text{Unknown} \\ \mathbf{u} \\ 2 \times 1}} = -\underbrace{\begin{bmatrix} I_t(1,1) \\ I_t(k,l) \\ \vdots \\ I_t(n,n) \end{bmatrix}}_{\substack{\text{Known} \\ B \\ n^2 \times 1}}$$

And the Product $A^T A$ becomes -

$$\underbrace{\begin{bmatrix} \Sigma_W I_x I_x & \Sigma_W I_x I_y \\ \Sigma_W I_x I_y & \Sigma_W I_y I_y \end{bmatrix}}_{A^T A} \underbrace{\begin{bmatrix} u \\ v \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} -\Sigma_W I_x I_t \\ -\Sigma_W I_y I_t \end{bmatrix}}_{A^T B}$$

means that the eigen values are both positive and non-zero, and as the determinant is the product of eigen values, the determinant is also > 0.

The threshold τ ignores the bad patches. In our implementation, since we have implemented the sparse version of Lucas-Kanade, we have already defined good selective features and hence varying τ does not have a huge effect. In case of dense, varying τ would have filtered noise and we would be able to control the number of features to be considered for optical flow.

2.3 (2) In the experiments, did you use any modifications and/or thresholds? Does this algorithm work well for these test images? If not, why?

I used threshold τ in Lucas-Kanade Algo

- if we decrease the value of the threshold τ lesser than 0.01, for example at $\tau = 0.005$, then we end up calculating optical flow at more regions than before, the algorithms works well but there is no reason for selecting the points with eigen values very small, which will add nothing to the optical flow.
- if we increase the threshold, for example at $\tau = 0.05$, only few points are selected, but the optical flow calculated at those regions give very good response, because we are thresholding and rejecting all the points whose smallest eigen value is less than that.

2.3 (3) Try experimenting with different window sizes. What are the trade-offs associated with using a small versus a large window size? Can you explain what's happening?

When we only take a small window, we will consider a very small neighborhood, which is good when the velocity of the center pixel/corner is very small. Then we get better estimates for the slow-moving points as we are not considering many points that might add noise

The problem arises when the corner moves so fast, that the center pixel is completely outside of the window in the next frame, which will give us completely wrong estimates of optical flow which when visualized give us completely random directions for the optical flow.

When we take a larger window, we will consider a larger neighborhood which is good for the corner/interest points moving very fast, which result in huge translation from frame to frame. Larger windows give us better estimates for fast moving interest points.

The problem is the larger window might also add so much noise, because there can be cases where the large window is also considering other interest points, which might be moving in various directions, which can give erroneous estimates for slow moving points or regions which include objects moving in opposite directions.

2.3 (4) DID YOU OBSERVE THAT THE GROUND TRUTH VISUALIZATIONS ARE IN HSV COLOR SPACE? TRY TO REASON IT.

The visualizations of the optical flow are done in HSV instead of RGB because,

- We can represent the angle of the optical flow with the hue value in HSV, which denotes different color at different angles, starting from Red at 0°, Yellow at 60°, Green at 120°, Cyan at 180°, Blue at 240°, Magenta at 300° and again ending at Red.
- We can represent the magnitude of the optical flow vector with saturation, low saturation means slow movement and high saturation (pure color) means the point's movement is fast.

The main reason for using HSV other than using Hue and saturation components is that we can comprehend colors in HSV space like real life where we use intensity in our color description, which is the V(value) component, and the color is the hue component.

Using RGB space to visualize optical flow vectors is a very painful task, as we must set ranges for different values for u and v, their angle and their magnitude. RGB is just a triplet of numbers which says nothing about the brightness/intensity of the color, or at least tell what is the pure color.

In HSV, for example, if the color at a particular pixel is light blue, we can understand that the optical flow direction is in the bottom-left (240°) direction, and the lightness says that the saturation is low, which means the velocity is also low. It is impossible to know the optical flow if it were visualized in RGB.