Computer Vision
Assignment 2 Solutions

Roll: 2019101017
Name: Shaurya Dewan

1.

1.1

1. Optical flow can be used to create slow motion videos by synthesizing intermediate frames. We can estimate the flow between any two consecutive frames of the original video and use this to interpolate the flow for intermediate frames. This interpolated flow can then be used to generate the intermediate frames. Thus, we can create a slow motion effect by obtaining frames that capture intermediate information such as miniscule movements without reducing the fps, etc. This also makes the video much more smooth than what we would obtain by simple manipulations such as changing the fps as instead of increasing the time period between frames, we are inserting the interim frames which further break down the transition between two of the original frames.
2. In the video, they make use of multiple cameras placed around the actor/scene where the cameras capture images in a specific sequence one after the other with some delay between each click. The video produced can be made smoother and slower by interpolating the frames in between each of the captured frames using optical flow in a similar fashion as described above in the previous question.
3. In order to recreate the ‘painterly effect’ observed in the video, we can estimate the optical flow between frames as 2x1 vectors for each point and then apply a smearing/stroke effect along these vectors.

4. When the sphere is rotated, there is no visible/apparent change in the intensity and distribution of pixels due to the constant illumination and hence there is no observable optical flow. However, the 2D motion field will be in the direction of motion.
   When the light source is moved, then there is a visible change where we observe windows of lighter pixels moving across. Here

1.2

1. The first assumption that is made is the brightness constancy assumption where we assume that pixels retain the same brightness/intensity from one frame to another and only their positions change due to the motion of objects in the scene and/or motion of the camera/observer. This does not take into account changes due to change in lighting, etc.
   The second primary assumption is the spatial coherence constraint/assumption where it is assumed that the neighbors of a pixel will have the same shift/flow as the pixel under consideration, i.e, patches of pixels shift together in a similar manner and not just individual pixels. This is done to obtain sufficient equations to solve for the unknowns and avoid dealing with an under-constrained problem.
2.

The objective function is:

$$I(x, y, t) = I(x+u, y+v, t+1)$$

where (x, y) is the original pixel location & (u, v) is the flow for that pixel. This is based on the brightness constancy assumption which states that the pixel intensity remains constant with only a shift in pixel locations.

After Taylor Series approximation,

$$\nabla I \, [u \; v]^T + I_t = 0$$

where $\nabla I = [I_x \; I_y]$

& $I_t = I(t+1) - I(t)$

Here, I_t is the data term as it is simply the difference in data at two different samples/times. $\nabla I\,[u\ v]^T$ is the spatial term as it involves the derivative at a specific point.

If we apply the spatial coherence constraint over 5×5 windows, i.e, assume the flow to be the same for all pixels in a 5×5 window around the pixel under consideration,

$$Ad = b \quad -(1)$$

where,

$$A = \begin{bmatrix} I_{x_1} & I_{y_1} \\ \vdots & \\ I_{x_{25}} & I_{y_{25}} \end{bmatrix},$$

$$d = \begin{bmatrix} u \\ v \end{bmatrix}$$

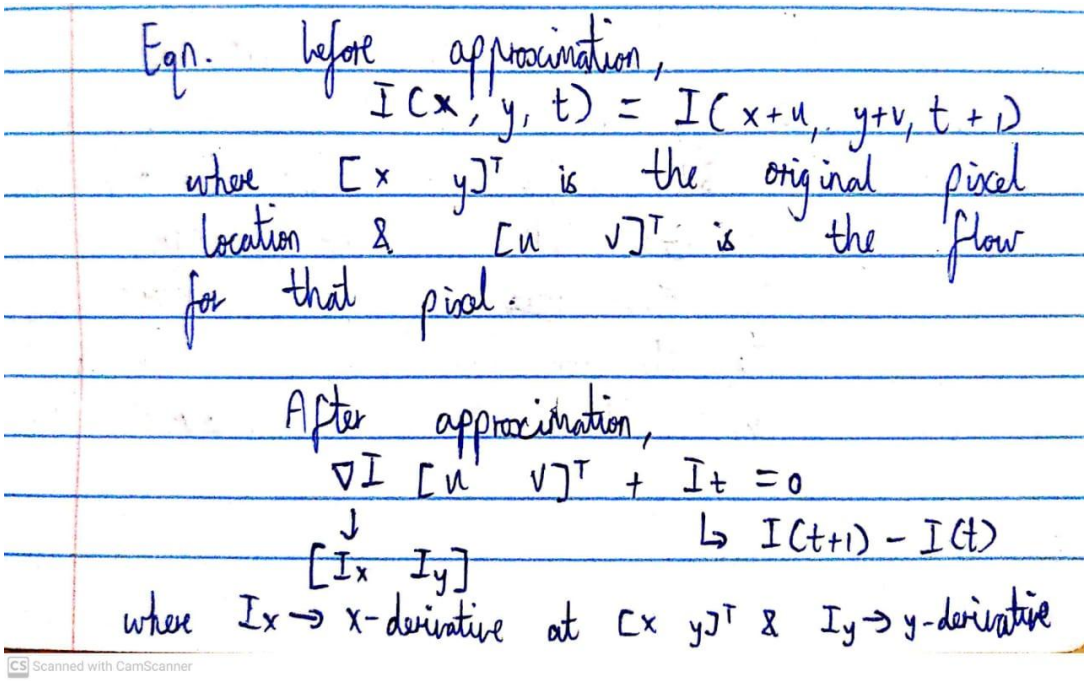
$$b = \begin{bmatrix} I_{t_1} \\ \vdots \\ I_{t_{25}} \end{bmatrix}$$

Lucas-Kanade eqn.:

$$(A^TA)\,d = A^Tb \quad (\text{(1) pre-multiply with } A^T)$$

3. This is done in order to simplify the equation in terms of the derivative of the image and the flow as separate terms.

Without this simplification it is not possible for us to estimate the flow as the original equation involves a function of the flow, I(x + u, y + v, t + 1), and not the flow itself [u, v] as a separate term. The approximation enables us to estimate the flow directly as we obtain a linear equation where we can solve for the flow. Moreover, it involves terms which are easy to compute such as the derivatives of the first image and the derivative/difference between the two images.



4. Geometrically, the objective function/equation obtained after simplification (Taylor Series approximation) fails as motion perpendicular to the gradient at any point/pixel cannot be measured/captured by the equation.

Here, I_t is the data term as it is simply the difference in data at two different samples/times. $\nabla I [u\ v]^T$ is the spatial term as it involves the derivative at a specific point.

If we apply the spatial coherence constraint over 5×5 windows, i.e, assume the flow to be the same for all pixels in a 5×5 window around the pixel under consideration,

$$A d = b \quad - (1)$$

where,

$$A = \begin{bmatrix} I_{x_1} & I_{y_1} \\ \vdots & \\ I_{x_{25}} & I_{y_{25}} \end{bmatrix},$$

$$d = \begin{bmatrix} u \\ v \end{bmatrix}$$

$$b = \begin{bmatrix} I_{t_1} \\ \vdots \\ I_{t_{25}} \end{bmatrix}$$

∴ Lucas-Kanade eqn.:

$$(A^T A)\, d = A^T b \quad (\text{pre-multiply with } A^T)$$

2.

2.3.

1. This is because if it has a rank less than 2, then it is not full rank and hence is not invertible. This means that no solution exists for the Lucas-Kanade equation at such points.
   The threshold used on the lower eigen value ensures that the determinant is not too small as matrices with determinant zero are not invertible. Thus as the determinant approaches zero, the solutions obtained become less reliable.
2. There is a trade-off in using larger thresholds. It returns better results as only well-conditioned points such as corners satisfy the threshold and the flow estimation is best for these points but then the overall flow estimation for the image becomes sparse.
   I found that the algorithm best worked for the 'RubberWhale' sequence as maximum points satisfied the threshold while it performed relatively poorly for the 'Grove3' sequence. The 'Grove3' sequence has many edges and corners but they are highly unorganized and the flow varies highly from point to point. The flow turns out to be a bit erroneous and noisy. In the 'RubberWhale' sequence the flow is similar over regions and highly organized. The images can be roughly divided into 4 regions based on flow and our algorithm performs very well for this type of sequence.
3. I observed that larger windows generally returned better results, especially for organized sequences like 'RubberWhale' as we obtain more equations to finetune/constrain our flow estimates per point when we use larger windows. The trade-off is that as we increase the window size, the computations become more expensive and complex and it takes longer and longer for the algorithm to run to completion. Thus, smaller windows are faster but larger windows give better results.
4. This method fails for rotations and occlusions. When the motion of objects in the scene and/or the camera's motion

results in certain points which were visible in prior frames being hidden in consequent frames, i.e, results in certain points occlusion, then this method fails as it requires the points location in the consequent frame to estimate the flow.

5. This is done as flow is represented by a 2x1 vector. Thus it can be used as/mapped to a point on a 2D plane such as the 2D color wheel which is used in the HSV representation where the angle/direction determines the hue (H) while the magnitude determines the value (V) for a constant saturation (S).