CS7.505.S22: Computer Vision | Assignment 2

Author: Aditya Kumar Singh
ID: 2021701010

March 22, 2022

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Contents

Question 1: Basics of Optical Flow

Estimating the motion of every pixel in a sequence of images is a problem with many applications in computer vision, such as image segmentation, object classification, and visual odometry. In general, optical flow describes a sparse or dense vector field, where a displacement vector is assigned to certain pixel position, that points to where that pixel can be found in another image.

Thought Experiments

1. The number-one use of optical flow in visual effects is for retiming a sequence — speeding it up or slowing it down. Describe how optical flow could be used to create slow motion video. You can find the answer in Amazing Slow Motion Videos With Optical Flow video on YouTube.

   It's movie time now! Let's watch two epic movie clips from two of the Academy Award winning movies for the Best Visual Effects - What Dreams May Come (1998), and The Matrix (1999).

2. In The Matrix, one of the most remembered, iconic use of the tactic comes during the rooftop scene where Neo effortlessly dodges one bullet after another. (Re-)watch "Bullet Time" here and explain briefly how optical flow is used. You may also find this amazing video on bullet-time interesting.

3. So breathtaking, heartbreaking and brimming with emotion, WDMC is a journey into the afterlife and deals with a dead man's attempt to reunite with his wife. Catch a glimpse of the "Painted World" here! You'll now describe briefly on how optical flow is used to create this "painterly effect".

4. Consider a Lambertian ball that is:

   (a) rotating about its axis in 3D under constant illumination and

   (b) stationary and a moving light source.

   What does the 2D motion field and optical flow look like in both cases?

Answer:

1. Part 1

   - Slow-motion (also called *slow-mo*) videos demands a higher frame rate in comparison to its normal counterpart. Because extra frames per sec in slow-mo videos accounts for smoother change in subject's action than in normal videos.
   - For example, the normal video is 2 fps while displaying it in slow-mo would make it (say) 1 fps which would seem more like as if video is getting stuck. To avoid such discrete display of frames we add more frames per sec to have smooth transitions.
   - Now optical flow plays a major role in deciding what extra information to add in those intermediate new frames. First, we estimate the Optical Flow (OF) between two adjacent frames. This way the transition of one specific feature (say a bird in the sky) from one frame to next frame seem smooth rather than abrupt.
   - One possible way with interpolation is to take the average of the two neighbouring frames, and then cross-dissolves to smooth out the artifacts in the video.

2. Part 2

   - This is a nice case of using OF not only to interpolate the frames for dodging the bullet, but also to account for rotation effect (i.e., changing of camera position to take shot from different angle).

- Anyway the flying bullet scenes are video edits and cinematography effects. While the rest i.e., the shot of falling Neo while dodging is captured *with a lot of cameras circling around the scene that programmed to click picture at some short interval* to create a revolving camera effect.
- Further, techniques like decomposing the actor into limbs, using optical flow to do interpolation based on the motion data of consecutive frames (probably average or bi-linear interpolation), and generating the intermediate frames and stabilizing the parts, and then rebuilding the actor again by combining the stabilized parts of the actor.

3. Part 3

- Firstly, the scenes created with “painterly effect” where awesome not only due to its beauty but also due to its simplicity. No doubt that the background (excluding protagonist and dog in “dream” scene) in each shot (= continuous footage or sequence between two edits or cuts) is digitally painted and to add lively effects, OF for each pixel (which may be manually made or may be taken from where there is garden fluttering scene in present) is added to those static image (paints).
- Not only one, rather a sequence of OF vectors are created for each pixel (say for time t, $t+1$, $t+2$, and $t+3$, four different OF vectors are made per pixel) which is repeated again and again to produce oscillatory (periodic) motion of flowers or leaves. Finally the actor is put back into the sequence of frames created.
- Here the main challenge is to assign a perfectly matching OF to the subjects in background. Because the nature of motion can for different subjects and in different parts of the videos. One way to tackle this problem is to shot the scene in real-world whose background can be correlated to the painting. Obtain the optical flow and put that back into painting (warp the paint) to create lively scene.

4. Part 4

(a) Since the ball is physically rotating, of course there is a motion field associated with it. But due to its shape (i.e., w.r.t to axis of rotation every slice going through the axis is same) as well as its nature (Lambertian) with constant illumination, the video would exactly look as if it were an image of the ball in that illumination (meaning there is no intensity change between the frames). Hence there won't be any optical flow. Mathematically,

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\cancelto{0}{\begin{bmatrix} I_t1 \\ I_t2 \\ \vdots I_tn \end{bmatrix}}$$

The only way the LHS side of the equation equates to 0 is when the optical flow is 0 because the image gradients cannot be zero at the edges of the sphere.

(b) While for the second case, since the ball is not moving there are no motion fields. But due to changing illumination (i.e., moving light source) the ball would seem to be rotating which would say there is an OF. Formally, since there is a difference in intensity between the frames, which will give a non zero solution for the displacements u and v.

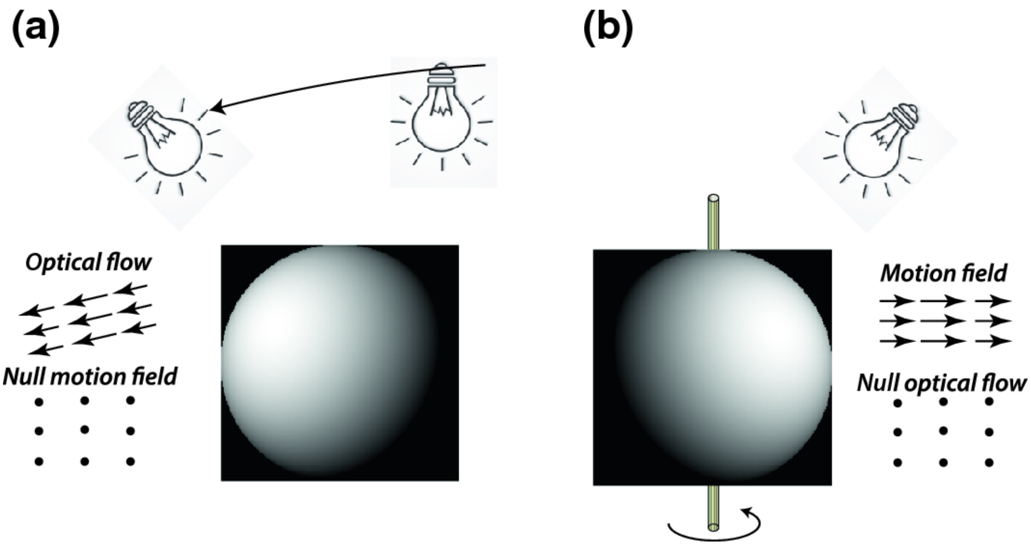

Figure 1: Source: Handbook of Image Processing and Computer Vision pp 479–598

Concept Review

1. List down the important assumptions made in optical flow estimation. Describe each one of them in one-two lines.
2. Formalize the objective function of the classical optical flow problem. Clearly mark the data term and the spatial term. What does the objective function imply about the noise distribution?
3. In optimization, why is the first-order Taylor series approximation done?
4. Geometrically show how the optical flow constraint equation is ill-posed. Also, draw the normal flow clearly.

Answer:

1. Part 1
   (a) **Illumination Constancy:** The brightness of any image point remains constant over time. In other words, the projection the same point in real-world looks the same in every frame.
   (b) **Small motion:** The displacement $(\delta x, \delta y)$ and time step δt are assumed to be very small. This enable us to apply Taylor series expansion and do all the approximations and further problem formulation to arrive at the estimates of $(\delta x, \delta y)$.
   (c) **Spatial Coherence:** To solve the *Aperture problem* we pretend the pixel's neighbors have the same (u, v) i.e., the points move like their neighbors. Plus, to solve for the under-constrained problem taking window make it over-constrained system which then posed as *least squares* problem.

2. Part 2 (*Source: On the Spatial Statistics of Optical Flow*)
   - Like LK and LKT (Lucas-Kanade-Tomashi) alogorithm to estimate OF, there is one such algorithm that globally formulate the optical flow problem introduced by *Horn and Schunck (=**HS**) (1981).* It relies on both brightness constancy and spatial smoothness assumptions, but suffers from the fact that their quadratic formulation is not robust to outliers.
   - To counter the above issue a lot of works has been demonstrated and published which is derived from HS. Hence all such spatially-discrete formulations derived from HS as "classical."
   - HS equation:
   $$E(\mathbf{u}, \mathbf{v}) = \int_I \left[\rho_D(\mathbf{I}_x\mathbf{u} + \mathbf{I}_y\mathbf{v} + \mathbf{I}_t) + \lambda.\rho_S\left(\sqrt{|\nabla \mathbf{u}|^2 + |\nabla \mathbf{v}|^2}\right)\right] dxdy \tag{1}$$
   where we need to minimize the energy function E to obtain the estimates $\hat{\mathbf{u}}, \hat{\mathbf{v}}$. ρ_D and ρ_S are robust penalty functions. I_x, I_y, and I_t denote the spatial and temporal derivatives of the image sequence. The first term in [1] is the so-called **data term** that enforces the brightness constancy assumption. The second term is the so-called **spatial term**, which enforces (piece-wise) spatial smoothness.
   - The noise distribution implied by the above energy (or error) function can be modeled using *Mixture of Gaussians.*

3. Part 3
   (a) Mathematically, *Taylor's* theorem gives an approximation of a k-times differentiable function around a given point, say a, (i.e., in the neighbourhood of that point) by a polynomial expression of degree k which is nothing but the truncation at the order k of the Taylor series of the function.
   $$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots = \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!}(x-a)^k \tag{2}$$
   (b) Since we assumed little displacement $0 < (\delta x, \delta y) \ll 1$, the higher-order terms diminishes in value and hence ignored. When truncation happens at $k = 2$, that linear approximation is termed as *first-order* Taylor polynomial as shown below.
   $$f(a) + \frac{f'(a)}{1!}(x-a) + \mathcal{O}(x^3) \approx f(a) + \frac{f'(a)}{1!}(x-a) \tag{3}$$
   (c) This approximation removes the dependency upon intensity terms like $I(x, y, t)$ completely from the equation. In order to estimate $(\delta u, \delta v)$ explicitly, the Taylor expansion pops out the δu and δv from $I(x + \delta x, y + \delta y, t + 1)$ (since $\delta u = \delta x$ as $\delta t = 1$) by approximating it in terms of $I(x, y, t + 1)$ that get cancelled with $I(x, y, t)$ due to brightness constancy assumption. Plus the expression for $(\delta u, \delta v)$ remains simple for first-order approximation.
   $$\cancel{I(x + \delta x, y + \delta y, t + 1)} = \cancel{I(x, y, t + 1)} + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}$$

4. Part 4
   - The *optical flow constraint equation* ($\mathbf{I}_x\mathbf{u} + \mathbf{I}_y\mathbf{v} + \mathbf{I}_t = 0$) expresses a constraint on the components u and v of the optical flow. Seeing this equation with u and v as variables, geometrically it can be interpreted as a straight line or constraint line. For any point (x, y) in the image, its OF $(\mathbf{u}, \mathbf{v})$ lies on the line. [picture]
   - The issue is we don't know where exactly the $(\mathbf{u}, \mathbf{v})$ lies on the line, which makes it a under constraint problem. Geometrically speaking, we have the information for the normal component of $(\mathbf{u}, \mathbf{v})$ i.e., $\mathbf{U}_n \left(= \frac{|I_t|}{(I_x^2 + I_y^2)}(I_x, I_y) \right)$ but not about the parallel component $(\mathbf{U}_p)$, which is why the $(\mathbf{u}, \mathbf{v})$ is constrained to a line and spans it but not to a point. This is the *ill-posedness* of this optical flow constraint equation.
   - This not only limited to algorithms but also applicable to humans. One such problem is *Aperture problem* where we perceive the normal component of actual motion of a object when viewed through a window/aperture whose size is very small compared to that of object. In other words, locally, we can only determine the normal flow.
   - Hence, to overcome this under-constraint nature (i.e., 2 unknowns and one equation) we add more constraints (e.g., taking 5×5 window around x, y gives 24 more constraints) to solve this two variable problem.

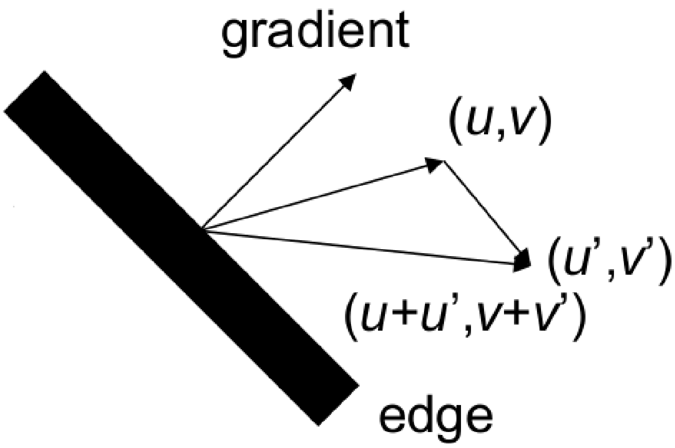

Figure 2: Problem of parallel flow: Ill-posed problem in LK OF estimation.

Question 2: Single-Scale Lucas-Kanade Optical Flow

Analyzing Lucas-Kanade Method

1. Why is optical flow only valid in the regions where local structure-tensor A^TA has a rank 2? What role does the threshold τ play here?
2. In the experiments, did you use any modifications and/or thresholds? Does this algorithm work well for these test images? If not, why?
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?
4. There are locations in an image where *Lucas–Kanade* optical flow will fail, regardless of choice of window size and sigma for Gaussian smoothing. Describe two such situations.You are not required to demonstrate them.
5. Did you observe that the ground truth visualizations are in HSV colour space? Try to reason it.

Answer:

1. Part 1

    - Having a full-rank square matrix ($A^T A$), makes it invertible due to which the explicit expression for velocity $(u, v)^T = (A^T A)^{-1} A^T b$ becomes solvable. Otherwise, this would mean $A^T A$ is non-invertible or singular (i.e., $\det(A^T A) = 0$). And this in turn implies atleast one of the eigen value is 0 which is not desirable. As this can happen only if we have either an uniform or an edgy region.
    - Threshold τ avoids the ill-conditioned patches like uniform intensity patches or patches consisting of edge only, where the Lucas-Kanade system is unsolvable (i.e., (u, v) can't be estimated properly). Ideally speaking, eigen-vector corresponding to the largest eigen-value accounts for maximum change (or captures maximum variation) and the patch whose both the eigen-values are significant enough implies that there is large variation in intensities along two major perpendicular directions. And this is sufficient to make it trackable through time. Whereas in case of edge patches or uniform patches, they are partially or not at all trackable.

2. Part 2

    (a) Lowering the threshold τ, would allow more patches to be considered as "interesting" patch, which may allow some "edge patches" for computing Lucas-Kanade system of equations to estimate (u, v). As a result of which the estimate would accumulate error and can cause a problem for advanced version of Lucas-Kanade like `MultiScaleLucasKanade`. So lowering threshold is more like relaxing the condition.

    (b) Whereas, increasing threshold would definitely allows only those, where "dots" type feature or something that have rough like texture is more prevalent. Further, increasing it too much would result in selection of fewer interesting patches due to which we may have OF information for only a small part of the image (which may not be a representative part of entire image).

3. Part 3

    (a) With small window sizes we may not be able to track the motion of interest points properly as a result of which OF vectors are oriented randomly (in any direction). One possible reason for this was the amount of shift (in terms of pixels) happened for image points while going from one frame to next. And the patch size we take around any interest point (or image point) to solve the LK system should be comparable to that amount of shift.

    (b) Like lower bound there is also an upper bound that decides what should be the upper limit. Taking windows of size $> (100 - 150)$ pixels results in unwanted pattern of OFs and may give OF at some points which point polar opposite to what actually should be. Not all parts of that patch move in same proportion, like there might be some points whose velocity is bit different from others, while there are some whose velocity direction is totally opposite, which anyway **violates** the 3^{rd} assumption made during OF estimation. And this is why a lot of aberrations can be observed in velocity estimates. [Add PICTURES]

4. Part 4

    - Locations where pixel intensity is "uniform" (i.e., gradient is zero or approximately zero in all directions) will bound to fail no matter what window size and sigma is chosen for Gaussian smoothing.
    - For locations, where there is an edge (i.e., uniform pixel intensity in one direction while large change in intensity in its perpendicular direction) will also fail regardless of any choice we make.

5. Part 5

    (a) 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 differeWnt 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.

    (b) 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.

    (c) Using RGB space to visualize optical flow vectors is a very painful task, as we have to 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 atleast tell what is the pure color.

(d) 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.