Chapter 9 – Motion Estimation & Optical Flow

Author: Gianmarco Scarano

gianmarcoscarano@gmail.com

1. Introduction

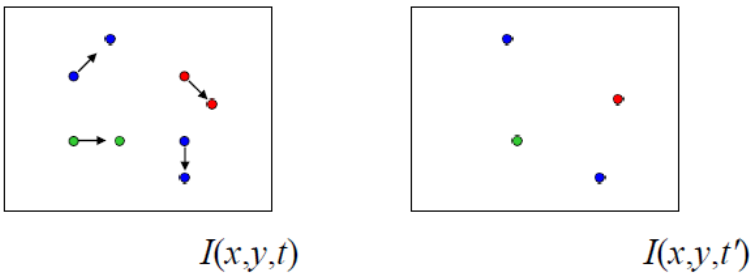
We know that if an image is (x, y), a video is of a form (x, y, t) where t is the time. We use motion (estimation of every pixel in sequence) for improving video quality, super-resolution tasks, tracking objects/body parts and so on and so forth.

We have different motion estimation techniques, but the most important are Optical Flow and Feature Tracking (the last one being about extracting features such as corners etc. from image).

1.1 Optical flow

Optical flow means recovering image motion at each pixel through image brightness variations and understand how every point in the image is moving frame to frame. Also, we want to underline the fact that motion can be caused either by the movement of a scene or by the movement of the camera.

- Sparse Optical Flow:
  - Only the most interesting pixels from the entire image are taken, within a frame.
- Dense Optical Flow:
  - All pixels from the entire frame are processed. Slower, but more accurate.

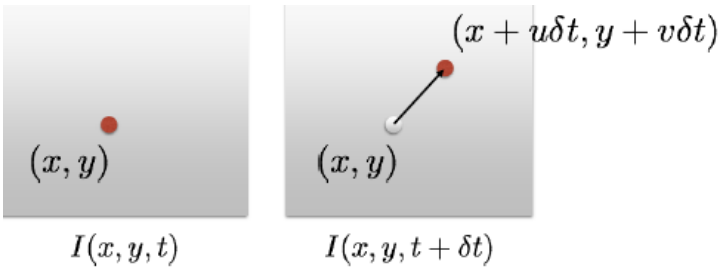


When we have to estimate optical flow between two frames (like the one on the left), we might also take into consideration that points do not move so far, there is some sort of spatial coherence and that the projection of the same point looks the same in every frame.

The approach, then, aims at looking for nearby pixels with the same color.

Since we said that there is brightness consistency, we can write something like this:

$I(x(t), y(t), t) = C$ where C is a constant, meaning that the brightness of the point will remain the same.



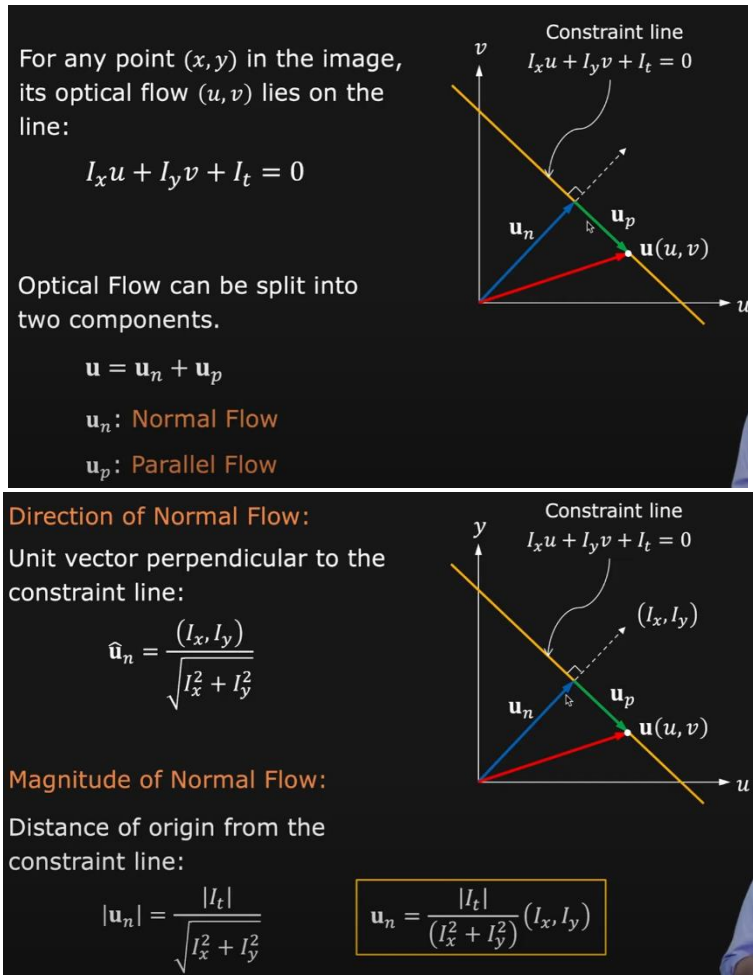
As for the small motion, we have an optical velocity (u, v) which correspond to x and y and a displacement in act during the two frames $(\delta x, \delta y) = (u\delta t, v\delta t)$ while

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

brightness is still unchanged.

If the time step is really small, we can linearize the intensity function with first order approximation of the Taylor series, as follows:

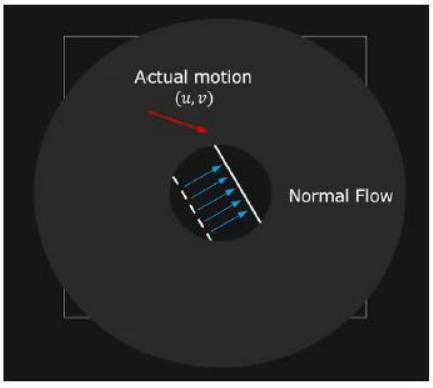
$$I_x u + I_y v + I_t = 0$$



And so, we can conclude that:

$$I(x+u, y+v) \approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

From these two images, we can see that on the upper side, we want to recover the true flow (u, v) (red line), but through the equation we can only know the yellow line. We know that the true flow vector lies in that yellow line, so we can divide this vector in two sub-vectors u_n and u_p, from where we can compute the Normal Flow only. This is due also to the aperture problem, where we might see the "Normal Flow" only and not the actual flow, when we zoom-in a region of interest as below:



Locally we can only determine normal flow

1.2 Constant (Lucas-Kanade) Optical Flow

Here, the flow is constant for all the pixels, based on the assumption that the optical flow (u, v) is constant within a small neighborhood (in other words they produce the same motion field and hence the same optical flow in the image).

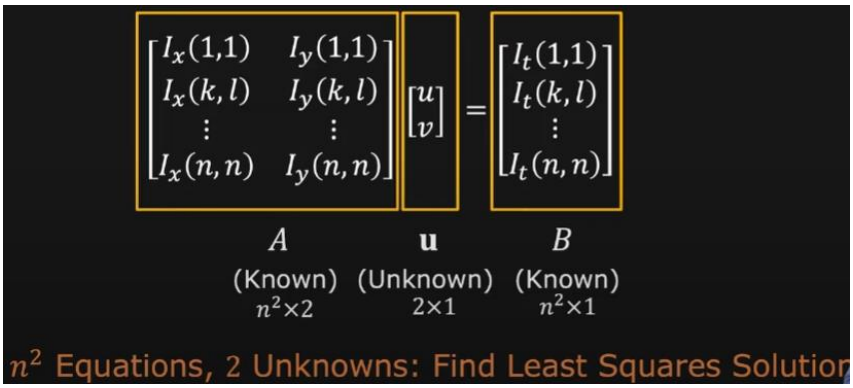
So, imagine we take a point $(k, l) \in W$, where W is a patch, we have:

- The derivative of the Intensity in the x-direction, multiplied by u => $I_x(k,l) \cdot u$
- The derivative of the Intensity in the y-direction, multiplied by v => $I_y(k,l) \cdot v$
- The derivative of the time direction => $I_t(k,l)$

Hence:

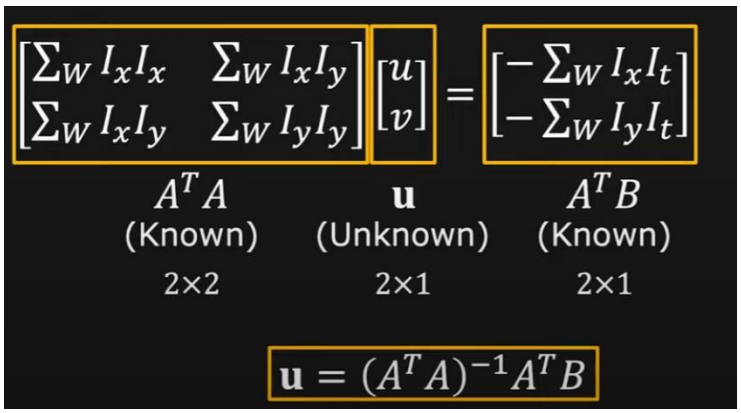
$$I_x(k,l) \cdot u + I_y(k,l) \cdot v + I_t(k,l) = 0$$

This is just for one point (k, l), but since we have multiple points in W, we'll have a system of equations based on the size of the window W (if W = 5x5, then we have 25 equations).



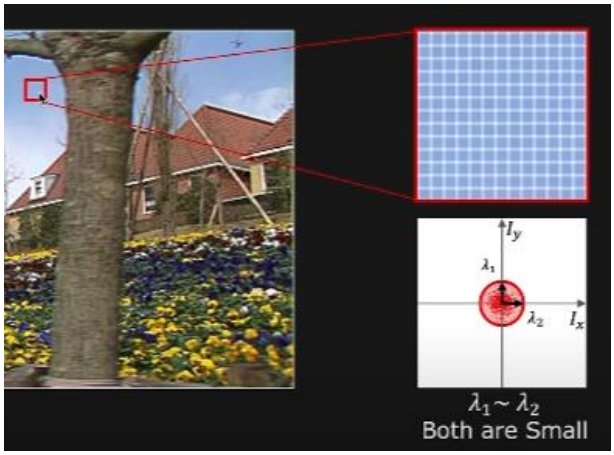
We can rewrite this system of equations in matrix form. The u (optical flow) is what we are trying to achieve and we can achieve that through the Least Squares method, solving $Au = b$, which is equal to $A^T Au = A^T B$, and finally arriving to the conclusion that:

$$u = (A^T A)^{-1} A^T B$$



But this only works if $A^T A$ is invertible, meaning that the determinant must be different from 0 and when $A^T A$ is well-conditioned, meaning that if we make changes in the input and nothing happens in the output, then that is NOT a well-conditioned system --> In the end, this system is well-conditioned when λ_1 and λ_2 (eigenvalues of $A^T A$) have the following properties: $\lambda_1 \geq \lambda_2$ *but not* $\lambda_1 \gg \lambda_2$ (λ_1 *not significantly larger than* λ_2).

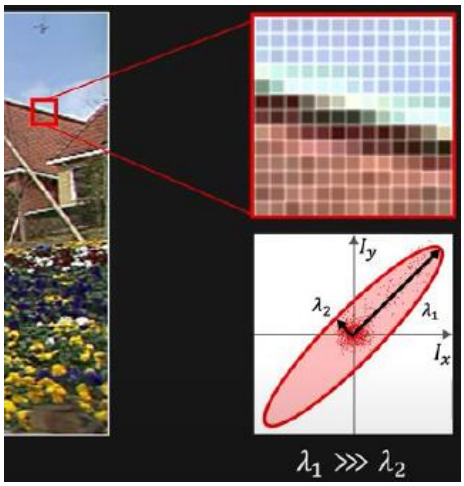
1.2.1 Smooth regions



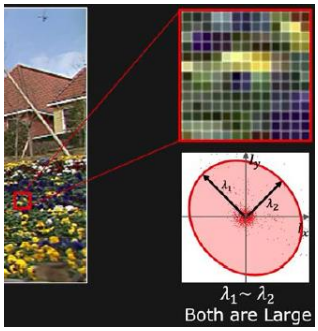
Let's pretend we have a patch and we want to compute the optical flow. We can see from this plot that λ_1 and λ_2 are small and so they are close to zero, in fact in the patch we don't have that much of a texture.

So, if λ_1 and λ_2 are really small (since the gradients are really small) then we are not really able to compute the optical flow in the correct way.

1.2.2 Edges



In this case, one eigenvalue is significantly larger than the other one, meaning that there is a prominent gradient in one direction. So, in this case it is also very difficult to compute the optical flow correctly.



This is a well-conditioned problem, since we have lots of changes in different brightness, values etc. in both directions -> Gradients are different and we can correctly compute the optical flow.

1.3 Smooth (Horn-Schunck) Optical Flow

Horn and Schunck said that they wanted to come up with an optical flow which in the end minimized some error, which is the following:

$$e_c = \iint_{image} (I_x u + I_y v + I_t)^2 \, dx \, dy$$

The first term that we have says how much we violate the brightness constancy constraint equation. Basically, that term is supposed to be 0, so when we say "violate", we mean that that specific term is somehow different from 0 (could be positive/negative as well). We square it and sum it up with the whole image and that tells us how bad my flow is w.r.t the constraint equation -> **CONSTRAINT ERROR**

When we have something moving around, the motion vectors from one point to the next change slowly.

So, they came up with a penalty term that penalizes the velocity fields when this change drastically and quickly from one location to the neighboring location:

$$e_s = \iint_{image} (u_x^2 + u_y^2) + (v_x^2 + v_y^2) \; dx \, dy$$

The first term is the derivative in the x direction, the second in the y direction (u, v). We would like it to be zero or relatively close to it. If it is 0, that

would mean that the whole image was moving towards the same EXACT direction. Any other (u) and (v) will cause this term to have some sort of value and in that case, we want to minimize this whole thing. -> **SMOOTHNESS ERROR**

Finally, we want to find the (u, v) at every pixel that **minimizes** the total error which is explicated as follows:

$$e = e_s + \lambda e_c$$

Where λ is the weighting factor. The higher the value is, the more importance we give to the brightness constancy constraint. The lower that value is, the more importance we give to the smoothness error.

That value depends on the image, how much noise there is etc.