Lucas-Kanade Tracking

Rohith Krishna

May 23, 2021

Computer Vision deals with extracting information from image and video data, and fundamentally aims at training machines to do tasks performed by the human visual cortex. The study involves a variety engaging topics such as motion estimation, feature tracking, image restoration etc. It is heavily founded upon mathematical ideas from Vector Calculus to Linear Algebra and Optimization. In this chapter we make a self-contained attempt at elucidating the computer vision problem of Lucas-Kanade Tracking, and bring out its similarity to regularized least squares and the filtering problem.

1 Displacement Tracking

In this section we discuss and solve the computer vision problem of feature tracking. The basic question is that given a video stream data and a feature of interest (a small object or region within each image), how can one effectively capture the displacement of the feature/object as it moves through each frame? The data provided to us, is a video sequence V which is a collection of images ordered sequentially in time, $V = \{I_1, I_2, \ldots, I_n\}$. In each image there is a region or object of interest to us, which we refer to as a feature. This feature at given reference position in an image begins to move as time progresses; therefore in subsequent images the feature has displaced. Our objective is to track this displacement over time. We shall solve the problem by tracking the feature between two given frames/images.

Consider the above illustration. Each image in the video is of resolution 45×45 pixels. We divide each image into 5×5 regions of interest to us, labelled by p_{ij}. Tracking cannot be done at the pixel level because the pixel brightness inadvertently changes every instant, and the presence of noise is unavoidable. It is therefore that we rather track features that cover a region.

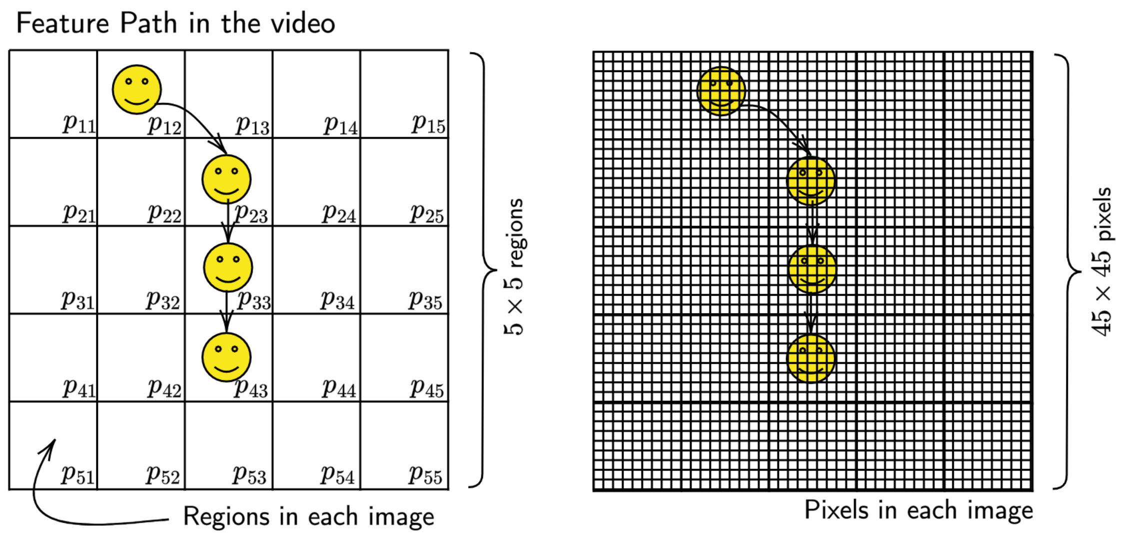


Figure 1: Path taken by the feature through the video.

The object of interest to us in the video starts at the region p_{12}. Then over the course of the video it moves to regions $p_{23} \rightarrow p_{33} \rightarrow p_{43}$. This path of the feature is shown in the figure below. The pixels highlighed at each step can also be seen.

In order to track the feature at each step, we break down the video into its constituent images and arrange them frame by frame. We take any two images, say I_1 and I_2 at instances $t = 1$ and $t = 2$ and track the diplacement of the feature at that instant. In repeatedly doing so, once can track the feature throughout the video.

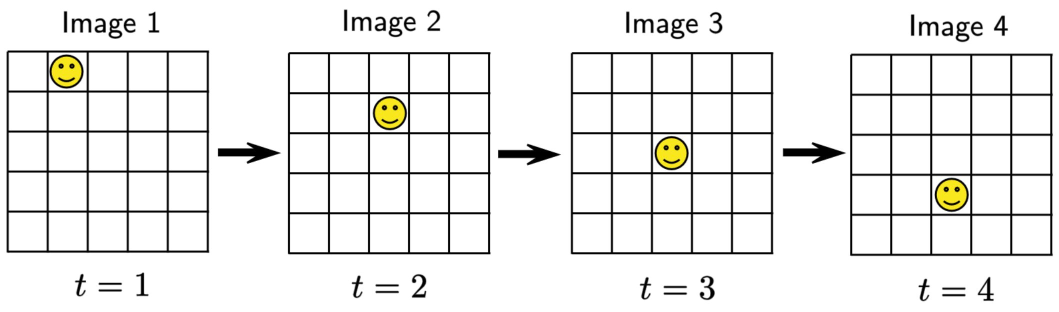
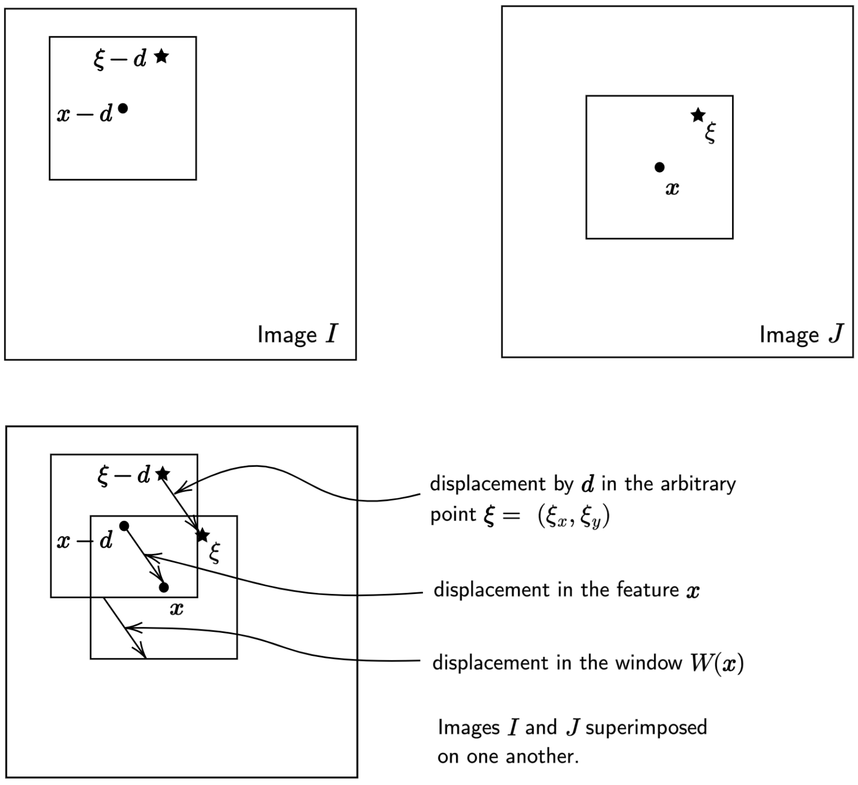


Figure 2: Path taken by the feature through the video.

1.1 Problem Statement

Consider two images I and J that occur sequentially in a video. Let $\boldsymbol{x} = (x, y)^T$ be the vector representing the coordinates of the feature to be tracked in image J. We consider a window $W(\boldsymbol{x})$ which is a collection of pixels that is being tracked. For simplicity assume that the position $\boldsymbol{x}$ is at the centre of our tracking window $W(\boldsymbol{x})$. Corresponding to this feature, the displaced feature must be present in the previous image I. If $\boldsymbol{d} = (d_x, d_y)^T$ denotes the displacement of the window from $\boldsymbol{x}$, then the position of the center of the window in image I must be $(\boldsymbol{x} - \boldsymbol{d})$.

At any instant, every pixel lights up. The intensity of birghtness of every pixel is function of its position relative to a fixed coordinate system. The function $I(x, y)$ refers to the intensity of pixel at position (x, y) in image I. Likewise function $J(x, y)$ refers to the intensity of pixel at position (x, y) in image J. Any arbitrary point $\boldsymbol{\xi}$ in image J is diplaced to $(\boldsymbol{\xi} - \boldsymbol{d})$ in image I. When object has not moved, then $\boldsymbol{d} = \mathbf{0}$ and images $I = J$. In order to create a scenario that satisfies this condition, we come up with notion of error between two images.



$\xi - d$
$x - d$
Image I
ξ
x
Image J
$\xi - d$
$x - d$
ξ
x
displacement by d in the arbitrary point $\xi = (\xi_x, \xi_y)$
displacement in the feature x
displacement in the window $W(x)$
Images I and J superimposed on one another.

Figure 3: Displacement $\boldsymbol{d}$ between images I and J.

Definition. (Error) The error $\mathcal{E}$ between two images J and I where an arbitrary point $\boldsymbol{\xi}$ has been displaced from $(\boldsymbol{\xi} - \boldsymbol{d})$, is defined as the sum of squared differences between all corresponding points in the window, given the fixed feature of interest $\boldsymbol{x}$. That is,

$$\mathcal{E}(\boldsymbol{x}, \boldsymbol{d}) = \iint_{W(\boldsymbol{x})} \Big[J(\boldsymbol{\xi}) - I(\boldsymbol{\xi} - \boldsymbol{d})\Big]^2 d\boldsymbol{\xi} \quad \text{(continuous)} \tag{1}$$

$$\mathcal{E}(\boldsymbol{x}, \boldsymbol{d}) = \sum_{(x,y)\in W} \Big[J(\xi_x, \xi_y) - I(\xi_x - d_x, \xi_y - d_y)\Big]^2 \quad \text{(discrete)} \tag{2}$$

The double integral in Equation 1 represents the integration over the window W entered at $\boldsymbol{x}$. It can be seen that when error $\mathcal{E} = 0$ the windows completely overlap and are identical.

Objective. Find displacement $\boldsymbol{d}$ so as to minimize error $\mathcal{E}$. There exists an optimal value of displacement $\boldsymbol{d}^*$ which minimized $\mathcal{E}$. Note: This minimization is for a fixed feature $\boldsymbol{x}$ in image J.

$$\boldsymbol{d}(\boldsymbol{x}) = \arg\min_{\boldsymbol{d}} \Big[\mathcal{E}(\boldsymbol{x}, \boldsymbol{d})\Big] = \boldsymbol{d}^* \tag{3}$$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \left[\iint_{W(\boldsymbol{x})} \Big[J(\boldsymbol{\xi}) - I(\boldsymbol{\xi} - \boldsymbol{d})\Big]^2 d\boldsymbol{\xi}\right] \tag{4}$$

1.2 Solution

The minimization problem in Equation 4 is solved using the First-Order Condition (FOC), which states $\frac{\partial \mathcal{E}}{\partial \boldsymbol{d}} = 0$. We implement the chain rule in differentiating the RHS of Equation 1.

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{d}} = 2\iint_{W(\boldsymbol{x})} \underbrace{\Big[J(\boldsymbol{\xi}) - I(\boldsymbol{\xi} - \boldsymbol{d})\Big] d\boldsymbol{\xi}\ \big\{-1\boldsymbol{\nabla}_d I\big\}\big\{-1\big\}}_{\text{chain rule of differentiation}} = \mathbf{0} \tag{5}$$

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{d}} = 2\iint_{W(\boldsymbol{x})} \Big[J(\boldsymbol{\xi}) - I(\boldsymbol{\xi} - \boldsymbol{d})\Big] \boldsymbol{\nabla}_d I\ d\boldsymbol{\xi} = \mathbf{0} \tag{6}$$

Recall that for small values of h Taylor's theorem can be approximated to the following. We use this to rewrite $I(\boldsymbol{\xi} - \boldsymbol{d})$. Note that the gradient arising here is with respect to $\boldsymbol{\xi}$, whereas the gradient in the FOC equation was with respect to $\boldsymbol{d}$.

$$\begin{aligned}\text{Since, } f(x+h) &\simeq f(x) + hf'(x) &(7)\\ \implies I(\boldsymbol{\xi} - \boldsymbol{d}) &\simeq I(\boldsymbol{\xi}) - [\boldsymbol{\nabla}_{\boldsymbol{\xi}} I]^T \cdot \boldsymbol{d} &(8)\end{aligned}$$

Applying 8 to 6, we have

$$\iint_{W(\boldsymbol{x})} \left[J(\boldsymbol{\xi}) - I(\boldsymbol{\xi}) + [\boldsymbol{\nabla}_{\boldsymbol{\xi}} I]^T \cdot \boldsymbol{d} \right] \boldsymbol{\nabla}_{\boldsymbol{d}} I \; d\boldsymbol{\xi} = \mathbf{0} \tag{9}$$

Since the integral operator is linear we split the term within the integral into two and rearrange.

$$\begin{aligned}\iint_{W(\boldsymbol{x})} \left[J(\boldsymbol{\xi}) - I(\boldsymbol{\xi}) \right] \boldsymbol{\nabla}_{\boldsymbol{d}} I \; d\boldsymbol{\xi} \quad &+ \quad \iint_{W(\boldsymbol{x})} \left[\boldsymbol{\nabla}_{\boldsymbol{d}} I \cdot [\boldsymbol{\nabla}_{\boldsymbol{\xi}} I]^T \right] \; \boldsymbol{d} \; d\boldsymbol{\xi} = \mathbf{0} &(10)\\ \underbrace{\iint_{W(\boldsymbol{x})} \left[\boldsymbol{\nabla}_{\boldsymbol{d}} I \cdot [\boldsymbol{\nabla}_{\boldsymbol{\xi}} I]^T d\boldsymbol{\xi} \right] \; \boldsymbol{d}}_{\text{matrix } A} \quad &= \quad \underbrace{- \iint_{W(\boldsymbol{x})} \left[J(\boldsymbol{\xi}) - I(\boldsymbol{\xi}) \right] \boldsymbol{\nabla}_{\boldsymbol{\xi}} I \; d\boldsymbol{\xi}}_{\text{error between images, vector } \boldsymbol{e}} &(11)\\ A\boldsymbol{d} \quad &= \quad \boldsymbol{e} &(12)\end{aligned}$$

We see that the equation reduces to the form $A\boldsymbol{d} = \boldsymbol{e}$ which can be solved for $\boldsymbol{d}$ using Gaussian Elimination or LU factorization. The vector $\boldsymbol{e}$ is similar to the error between the two images. The vector $\boldsymbol{e}$ much like $\boldsymbol{d}$ is of the order 2×1. The nature and order of matrix A is to be determined.

$$\begin{aligned}A &= \iint_{W(\boldsymbol{x})} \left[\boldsymbol{\nabla}_{\boldsymbol{d}} I \cdot [\boldsymbol{\nabla}_{\boldsymbol{\xi}} I]^T d\boldsymbol{\xi} \right] &(13)\\ A &= \begin{bmatrix} \sum_x \sum_y \left(\frac{\partial I}{\partial x} \right)^2 & \sum_x \sum_y \frac{\partial I}{\partial x} \frac{\partial I}{\partial y} \\ \sum_x \sum_y \frac{\partial I}{\partial y} \frac{\partial I}{\partial x} & \sum_x \sum_y \left(\frac{\partial I}{\partial y} \right)^2 \end{bmatrix} &(14)\end{aligned}$$

Since the gradient in this case is 2×1 vector, the inner product of two gradients of I yields a 2×2 Gramian matrix. Since $\boldsymbol{\nabla} I = (\partial I / \partial x, \partial I / \partial y)^T$, we compute the product of the gradients. Further, summing over the exent of the window gives matrix A.

$$(\boldsymbol{\nabla} I)(\boldsymbol{\nabla} I)^T = \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial y}\frac{\partial I}{\partial x} & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \tag{15}$$

Note that in $A\boldsymbol{d} = \boldsymbol{e}$, the expansion for $\boldsymbol{e}$ depends on both images I and J, whereas the matrix A depends only on image I. This causes a significant problem, when A is a rank deficient matrix and A^{-1} does not exist. This is typically seen in edges where $(\partial I/\partial x)$ vanishes, thereby making A uninvertible. The explicit form for vector $\boldsymbol{e}$ is also calculated in a similar fashion.

$$\boldsymbol{e} = \iint_{W(\boldsymbol{x})} \Big[I(\boldsymbol{\xi}) - J(\boldsymbol{\xi})\Big] \boldsymbol{\nabla}_{\boldsymbol{\xi}} I \; d\boldsymbol{\xi} \tag{16}$$

$$\boldsymbol{e} = \begin{bmatrix} \sum_x \sum_y \Big(I(x,y) - J(x,y)\Big)\frac{\partial I}{\partial x} \\ \sum_x \sum_y \Big(I(x,y) - J(x,y)\Big)\frac{\partial I}{\partial y} \end{bmatrix} \tag{17}$$

1.3 Bilinear Interpolation

On solving $A\boldsymbol{d} = \boldsymbol{e}$ for a window W with points (x_p, y_p) as the centre, we obtain some value of displacement $(d_x p, d_y p)$. These displacement are typically small values. The process of finding displacement using Lucas-Kanade tracking method involves bilinear interpolation, which is an iterative step through which individual window displacements are computed.

So far we have taken a single window of size w in image I and J, performed the first step of displacement iteration. We create a new image $I_1(x, y)$ from the original image $I(x, y)$ by subtracting out the displacement $\boldsymbol{d} = (d_x 1, d_y 1)$ obtained in the previous iteration. It is done using the bilinear interpolation formula given by

$$\begin{aligned} I(r+u, c+v) &= I(r,c).(1-u).(1-v) + I(r+1,c).u.(1-v) \\ &\quad + I(r,c+1).(1-u).v + I(r+1,c+1).u.v \end{aligned} \tag{18}$$

where r indexes rows (y axis) and c indexes columns (x axis). u denotes d_y and v denotes d_x obtained in the previous step of the iteration. Note that an image specified as $I(y, x)$ in row-column notation is denoted by $I(r, c)$. The bilinear interpolation method for computing $I(r + d_y, c + d_x)$ from the intensities at neighbouring pixels and displacement (d_x, d_y) is shown in Figure 4.

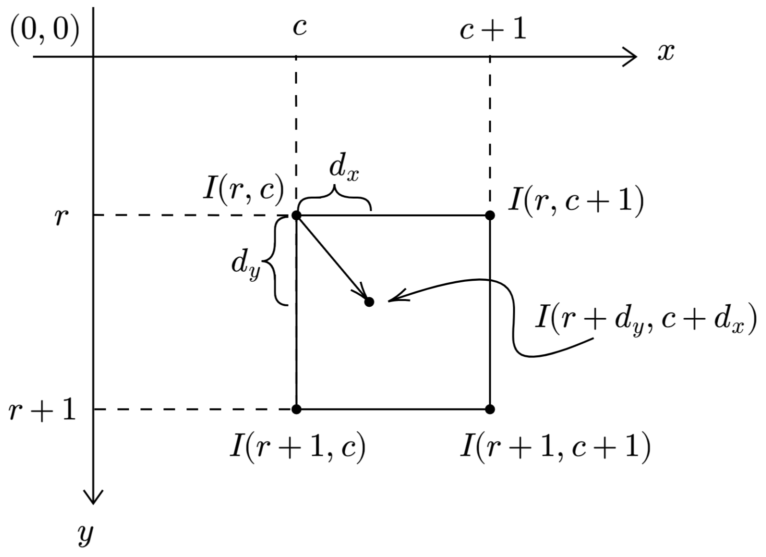


Figure 4: Bilinear Interpolation method for computing $I(r+d_y, c+d_x)$ from the intensities at neighbouring pixels and $\boldsymbol{d}$.

On using Equation 18, we obtain the intermediate pixel intensity for $I(x+d_x, y+d_y)$ after the first step of the iteration. Repeating this for all pixels in window of size w, we get a new intermediate image $I_1(\boldsymbol{\xi}) = I(\boldsymbol{\xi} - \boldsymbol{d})$. Subtracting the intermediate displacements at each step amounts to bringing the image I closer to J, which would in fact reduce the magnitude of displacements in the next step. Thus, after k iterations, the intermediate displacements would converge to 0. The final displacement is the sum of intermediate displacements:

$$\boldsymbol{d} = \boldsymbol{d_1} + \boldsymbol{d_2} + \cdots + \boldsymbol{d_k} \tag{19}$$

1.4 Displacement Maps

The algorithm presented above computes the displacement for a given window of size w within an image. The displacement for a window is treated as the displacement of the pixel that is centered on the window. One can slide a single window in the x and y directions and generate multiple windows within the image. Applying the tracking algorithm for all windows of the image results in a new image (with dimensions of the original image reduced by the window size, to avoid edge effects), called the *displacement map*. Each point in the displacement map is tiny vector with magnitudes and directions of displacements between images I and J.

1.5 Computing Gradients

Since the algorithm to solve $A\boldsymbol{d} = \boldsymbol{e}$ involves computing A and $\boldsymbol{e}$ vectors, which in turn involves calculating the gradients of I with respect to x and y, one must first come up with a method of finding gradients on a computer. Computers inherently do not deal with continuous variables and can only take discrete values, the approximate the definition of gradient and apply it for the smallest difference in x and y that is possible. Since x and y denote the positions of pixels along the respective axes, these variables can only be incremented by a single unit. Thus necessarily we have $\Delta x = 1$ and $\Delta y = 1$.

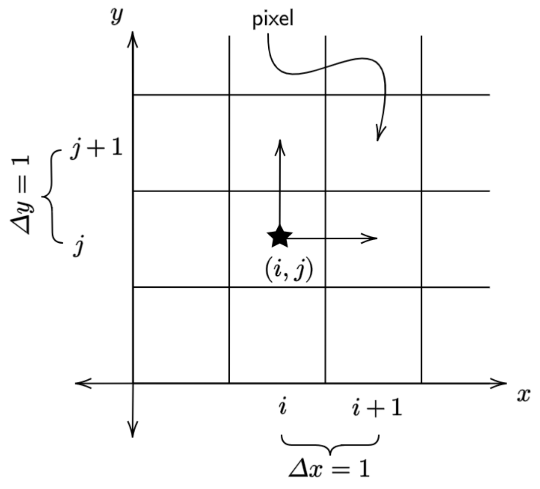


Figure 5: Gradient is defined over pixel-wise change in intensity.

From the definition of gradient,

$$\left.\frac{\partial I(x,y)}{\partial x}\right|_{y} = \lim_{\Delta x \to 0} \frac{I(x+\Delta x, y) - I(x,y)}{\Delta x} \tag{20}$$

$$\left.\frac{\partial I(x,y)}{\partial y}\right|_{x} = \lim_{\Delta y \to 0} \frac{I(x, y+\Delta y) - I(x,y)}{\Delta y} \tag{21}$$

Given an initial point $(x, y) = (i, j)$, moving along the x-axis by 1 unit increments $x = i$ by $\Delta x = 1$ units; thereby the coordinate shifts to $(i+1, j)$. A similar argument can be made for shift in y coordinate by $\Delta y = 1$ unit. In essence the denominator in the equation for gradient becomes unity. Therefore one can obtain gradients using,

$$\left.\frac{\partial I(x,y)}{\partial x}\right|_{\substack{x=i,\\ y=j}} = \frac{I(i+1,j) - I(i,j)}{1}, \quad \left.\frac{\partial I(x,y)}{\partial y}\right|_{\substack{x=i,\\ y=j}} = \frac{I(i,j+1) - I(i,j)}{1} \tag{22}$$

2 Motion Flow

Consider a point $\boldsymbol{p} \in \mathbb{R}^2$ in a 2D image. This point corresponds to a *world point* $\boldsymbol{P} \in \mathbb{R}^3$ in 3D in the real world. As $\boldsymbol{P}$ moves in 3 dimensions, so does its projection, the *image point* $\boldsymbol{p}$ in 2 dimensions. The image point is associated with a vector $\boldsymbol{u}$ to indicate its motion in the image plane. A *motion flow* is a collection of such vectors $\boldsymbol{u}$, and *optical flow* describes the motion of brightness patters in the image and consequently captures the motion of *isointensity curves* in a image.

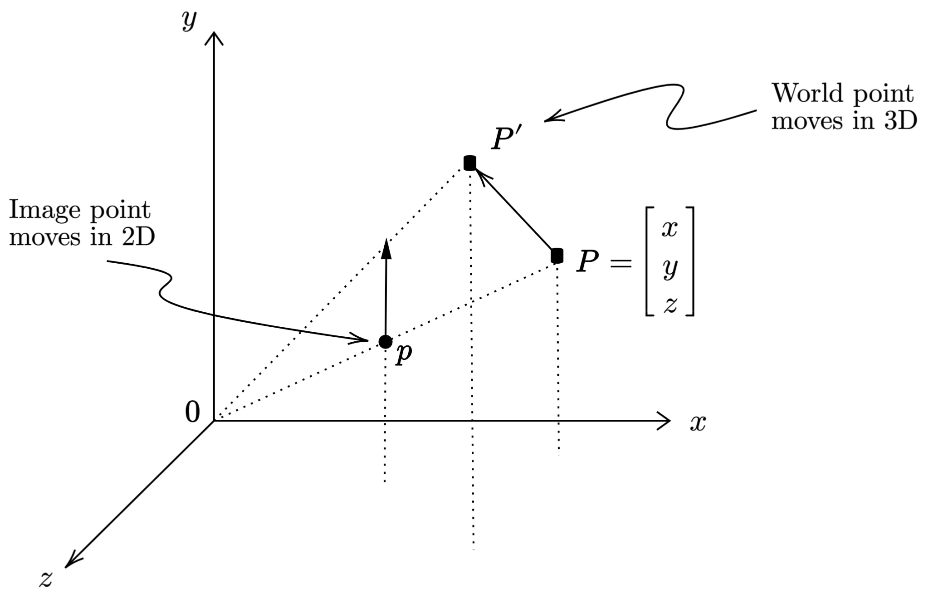
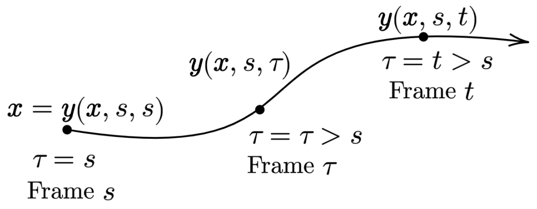


Figure 6: Motion of world point $\boldsymbol{P}$ corresponds to the motion of image point $\boldsymbol{p}$ with velocity $\boldsymbol{u}$.

2.1 Euler-Lagrange Description

Let $\boldsymbol{x} = (x, y)$, a 2D image point be the projection of some 3D world point $\boldsymbol{y} = (\boldsymbol{x}, s, t)$. $\boldsymbol{y}$ denotes the trajectory which the projection of motion of world point traces in the 2D sensor plane as a function of time.



$\boldsymbol{y}(\boldsymbol{x}, s, t)$
$\boldsymbol{y}(\boldsymbol{x}, s, \tau)$
$\tau = t > s$
Frame t
$\boldsymbol{x} = \boldsymbol{y}(\boldsymbol{x}, s, s)$
$\tau = \tau > s$
Frame τ
$\tau = s$
Frame s

Figure 7: Euler-Lagrange Description of motion of world point captured by the motion of image point.

Here, the object starts at point $\boldsymbol{x} = (x, y) = \boldsymbol{y}(\boldsymbol{x}, s, t)$ at time s, moves to an intermediate point $\boldsymbol{y}(\boldsymbol{x}, s, \tau)$ and further to the final point $\boldsymbol{y}(\boldsymbol{x}, s, t)$ at time t where $t > s$. The notation used is,

$$\boldsymbol{y}(\textsf{position in 2D, end time, start time})$$

Euler looks at a fixed pixel $\boldsymbol{x}$ and observes how its intensity changes with time. Lagrange on the contrary follows a fixed 3D world point and measures the change in intensity of the projection of the world point. These two descriptions are identical ways to obtain the velocity of the object in motion.

In the Lagrangian description, the image velocity of point $\boldsymbol{y}(\boldsymbol{x}, s, t)$ at time t is simply the derivative,

$$\boldsymbol{W}(\boldsymbol{x}, s, t) = \frac{\partial \boldsymbol{y}(\boldsymbol{x}, s, t)}{\partial t} \tag{23}$$

In the Eulerian description, the motion field is simply the image velocity of the fixed point $\boldsymbol{y}(\boldsymbol{x}, s, t)$ at time $t = s$,

$$\boldsymbol{V}(\boldsymbol{x}, s) = \boldsymbol{W}(\boldsymbol{x}, s, s) \tag{24}$$

Since at end time $t > s$, the point $\boldsymbol{x}$ is at position $\boldsymbol{y}(\boldsymbol{x}, s, t)$, the displacement is obtained using,

$$\boldsymbol{y}(\boldsymbol{x}, s, t) = \int_{\tau=s}^{\tau=t} \boldsymbol{W}(\boldsymbol{x}, s, \tau) d\tau \tag{25}$$

$$\boldsymbol{d}(\boldsymbol{x}, s, t) = \big[\boldsymbol{y}(\boldsymbol{x}, s, t) - \boldsymbol{y}(\boldsymbol{x}, s, s)\big] \tag{26}$$

Note that the 3D point visible at a particular position at start time s may become obstructed or occluded at a later time $t > s$ or it may leave the field of view of the camera altogether. Conversely, points that are visible at time t may be occluded at earlier time $s < t$. For world points that are obstructed at either start or end time (s or t), the displacement $\boldsymbol{d}$ is undefined. For all other points not obstructed at one time have a corresponding point at the other time. When $\tau = s$, the initial time period, the Lagrangian and Eulerian descriptions are the same.

The Eulerian description of the motion field $\boldsymbol{V}(\boldsymbol{x}, s)$ is not observable in a video. That is, not every change in the video results in a corresponding change in the field. For example, consider a smooth uniformly colored sphere rotating around its axis. The would produce no trace of its motion in the image (zero optical field), while in actuality a rotating sphere has a non-zero motion field. Alternatively, consider a static scene (zero motion field) being lit by a moving light source. This produces changes in the image and thereby non-zero optical flow, without any object in the scene actually moving. Thus the estimation of motion field from a video has a fundamental observability issue. Motion field, or the projection of 3D world point onto the 2D image place is true motion, while optical flow obtained from video frames is a measured motion.

2.2 Problem statement & Assumptions

The core objective of Motion Flow problem is to measure motion field from video data. We interpret changes in the observed brightness or intensity patter as motion by assuring that the different frames are related views in the same scene.
The *weak assumption of constant appearance* states that the appearance of a given 3D world point does not change over small intervals of time as the 3D world point moves across the field of view. This assumption, however, can be violated if the lighting on the object changes. Mathematically this is expressed in terms of constant irradiance,

$$\boldsymbol{e}(\boldsymbol{x}(t), t) = \textsf{constant over time} \tag{27}$$

This implies that, for a 3D world point,

$$\frac{D\boldsymbol{e}(\boldsymbol{x}(t), t)}{Dt} = \mathbf{0} \tag{28}$$

By definition of derivative, the partial derivative properties,

$$\frac{D\boldsymbol{e}(\boldsymbol{x}(t), t)}{Dt} = \lim \frac{\boldsymbol{e}\big[\boldsymbol{x}(t+\Delta t), t+\Delta t\big] - \boldsymbol{e}\big[\boldsymbol{x}(t), t+t\big]}{\Delta t} = \mathbf{0} \tag{29}$$

$$\frac{D\boldsymbol{e}(\boldsymbol{x}(t), t)}{Dt} = \frac{\partial \boldsymbol{e}}{\partial t}\bigg|_{\boldsymbol{x}} + \frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}} \underbrace{\frac{\partial \boldsymbol{x}}{\partial t}}_{\boldsymbol{v}} = \mathbf{0} \tag{30}$$

This simplifies to the *optical flow constraint equation* given by,

$$\underbrace{\frac{D\boldsymbol{e}}{Dt}}_{\text{Lagrange}} = \underbrace{\frac{\partial \boldsymbol{e}}{\partial t}\bigg|_{\boldsymbol{x}}}_{\text{Euler}} + \underbrace{\boldsymbol{v} \cdot \big[\boldsymbol{\nabla} \boldsymbol{e}\big]}_{\substack{\text{motion field} \\ \text{additive term}}} = \mathbf{0} \tag{31}$$

In the optical flow constraint, there are 3 equations, one for each color - red, green and blue. The two unknowns to be determined are velocity $\boldsymbol{v} = (v_x, v_y)$ in the motion field.

There are two derivatives of $\boldsymbol{e}$ in Equation 31, one with respect to time t and the other with respect to space $\boldsymbol{x}$. The term $\frac{\partial \boldsymbol{e}}{\partial t}\big|_{\boldsymbol{x}}$ is estimated from the video. It is the derivative from 2 frames of the video for a fixed pixel $\boldsymbol{x}$. Thus,

$$\left.\frac{\partial \boldsymbol{e}}{\partial t}\right|_{\boldsymbol{x}} = \frac{\boldsymbol{e}(\boldsymbol{x}, t+1) - \boldsymbol{e}(\boldsymbol{x}, t)}{1} \tag{32}$$

The gradient $\boldsymbol{\nabla e}\big|_t$ is estimated from within a given frame, at a fixed time. This is called the spatial gradient and is a Jacobian matrix of order 3×2, where the columns represent each spatial component and the rows represent the partial derivative corresponding to a particular color.

$$\underset{\text{gradient}}{\overset{\text{spatial}}{}} \quad \boldsymbol{\nabla e} = \left.\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\right|_t = \begin{bmatrix} \frac{\partial e_1}{\partial x} & \frac{\partial e_1}{\partial y} \\ \frac{\partial e_2}{\partial x} & \frac{\partial e_2}{\partial y} \\ \frac{\partial e_3}{\partial x} & \frac{\partial e_3}{\partial y} \end{bmatrix} \begin{cases} \rightarrow e_1 = R \\ \rightarrow e_2 = G \\ \rightarrow e_3 = B \end{cases} \tag{33}$$

$$\underset{\text{gradient}}{\overset{\text{spatio-temporal}}{}} \quad \boldsymbol{\nabla e} = \left[\left.\frac{\partial \boldsymbol{e}}{\partial \boldsymbol{x}}\right|_t, \left.\frac{\partial \boldsymbol{e}}{\partial t}\right|_{\boldsymbol{x}} \right] = \begin{bmatrix} \frac{\partial e_1}{\partial x} & \frac{\partial e_1}{\partial y} & \frac{\partial e_1}{\partial t} \\ \frac{\partial e_2}{\partial x} & \frac{\partial e_2}{\partial y} & \frac{\partial e_1}{\partial t} \\ \frac{\partial e_3}{\partial x} & \frac{\partial e_3}{\partial y} & \frac{\partial e_1}{\partial t} \end{bmatrix} \tag{34}$$

The spatio-temporal Jacobian in given by Equation 34. Note that the RGB color components are correlated only with the spatial gradient and not with the temporal gradient. Therefore the matrix columns in Equation 34 are not linearly independent.

2.3 Aperture Problem

Optical flow constraint leads to a problem in computer vision referred to as the *aperture problem.* It states that one can only detect motions that are normal to the edge, or in other words, motion is detected only in the direction of the gradient. One is unable to detect motion perpendicular to the direction of the gradient. The optical flow constraint equation in 31 degenerates into a single equation in I.

$$\left.\frac{\partial I}{\partial t}\right|_{\boldsymbol{x}} + \boldsymbol{v} \cdot \boldsymbol{\nabla} I = 0 \tag{35}$$

$$\underbrace{\boldsymbol{v} \cdot \boldsymbol{\nabla} I}_{\substack{\text{only the dot} \\ \text{product is observed}}} = \underbrace{-\left.\frac{\partial I}{\partial t}\right|_{\boldsymbol{x}}}_{\substack{\text{computed} \\ \text{between frames}}} \tag{36}$$

$\boldsymbol{v}$ is unknown, but I is known. We can measure only $\boldsymbol{v} \cdot \boldsymbol{\nabla} I$ only in the direction of the gradient. Thus $\boldsymbol{v} \cdot \boldsymbol{\nabla} I$, nonetheless $\boldsymbol{v}$ is unknown. If $\boldsymbol{\nabla} I \neq 0$ at point $\boldsymbol{x}$ then we can rearrange the terms to obtain the component of the motion field along the spatial gradient of I, given by

$$\mathcal{V}(\boldsymbol{x}) = \frac{\boldsymbol{v} \cdot \boldsymbol{\nabla} I}{\|\boldsymbol{\nabla} I\|_2} = \frac{-1}{\|\boldsymbol{\nabla} I\|_2} \left.\frac{\partial I}{\partial t}\right|_{\boldsymbol{x}} \tag{37}$$

This component of motion field observable along the spatial gradient of I is the *normal component of motion field.* The term on the right hand side of Equation 37 is computed from the video data, frame by frame. Therefore, the component of $\boldsymbol{v}$ orthogonal to $\boldsymbol{\nabla} I$ is not observable. This is called the *aperture problem.*

2.4 Solution

The objective of the motion flow problem is the estimation of the velocity term $\boldsymbol{v}$. To solve this, we being by redefining the optical flow constraint equation and differentiating it with respect to spatial component. Let $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{\nabla}(I(\boldsymbol{x}) = [I_x, I_y]$. The optical flow constraint equation goes as,

$$\frac{DI}{Dt} = I_t + \boldsymbol{v}^T \cdot \boldsymbol{g} = \mathbf{0} \tag{38}$$

Differentiating both sides with respect to $\boldsymbol{x}$,

$$\frac{\partial}{\partial \boldsymbol{x}} \frac{DI}{Dt} = \frac{\partial I_t}{\partial \boldsymbol{x}} + \frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{v}^T \cdot \boldsymbol{g}) = \frac{\partial \mathbf{0}}{\partial \boldsymbol{x}} = \mathbf{0} \tag{39}$$

$$\frac{\partial}{\partial \boldsymbol{x}} \frac{DI}{Dt} = \frac{\partial}{\partial \boldsymbol{x}} \frac{\partial I}{\partial t} + \frac{\partial \boldsymbol{g}^T}{\partial \boldsymbol{x}} \cdot \boldsymbol{v} + \frac{\partial \boldsymbol{v}^T}{\partial \boldsymbol{x}} \cdot \boldsymbol{g} = \mathbf{0} \tag{40}$$

Rearranging the terms and assuming $\frac{\partial \boldsymbol{v}^T}{\partial \boldsymbol{x}} \cdot \boldsymbol{g} = \mathbf{0}$, because we do not want deformations in image intensity (i.e, no first order terms) we have,

$$\frac{\partial^2 I}{\partial \boldsymbol{x} \partial t} + \frac{\partial \boldsymbol{g}^T}{\partial \boldsymbol{x}} \cdot \boldsymbol{v} + \underbrace{\frac{\partial \boldsymbol{v}^T}{\partial \boldsymbol{x}} \cdot \boldsymbol{g}}_{= \mathbf{0}, \text{ assume.}} = \mathbf{0} \tag{41}$$

$$\frac{\partial^2 I}{\partial \boldsymbol{x} \partial t} + \frac{\partial \boldsymbol{g}^T}{\partial \boldsymbol{x}} \cdot \boldsymbol{v} = \mathbf{0} \tag{42}$$

Thus with Equation 38 and Equation 42, we have in total 3 equations to solve for 2 unknown. Let the notation be the following: time gradient between two frames is $\frac{\partial I}{\partial t} = I_t$, the term $\frac{\partial^2 I}{\partial \boldsymbol{x} \partial t} = I_{\boldsymbol{x}t}$ and the term $\frac{\partial \boldsymbol{g}^T}{\partial \boldsymbol{x}} = H$ is simply the Hessian matrix. Thus we have,

$$\boldsymbol{v}^T \cdot \boldsymbol{g} + I_t = 0 \implies \boldsymbol{g}^T \cdot \boldsymbol{v} = -I_t \tag{43}$$

$$\frac{\partial^2 I}{\partial \boldsymbol{x} \partial t} + \frac{\partial \boldsymbol{g}^T}{\partial \boldsymbol{x}} \cdot \boldsymbol{v} = \boldsymbol{0} \implies H\boldsymbol{v} = -I_{\boldsymbol{x}t} \tag{44}$$

Consider the matrices,

$$A = \begin{bmatrix} \boldsymbol{g}^T \\ H \end{bmatrix}_{3\times 2}, \quad \boldsymbol{b} = \begin{bmatrix} I_t \\ I_{\boldsymbol{x}t} \end{bmatrix}_{3\times 1} \tag{45}$$

Thus the motion flow problem reduces to solving the matrix equation of the form $A\boldsymbol{v} = \boldsymbol{b}$, where,

$$\underbrace{\begin{bmatrix} \boldsymbol{g}^T \\ H \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} v_x \\ v_y \end{bmatrix}}_{\boldsymbol{v}} = \underbrace{\begin{bmatrix} I_t \\ I_{\boldsymbol{x}t} \end{bmatrix}}_{\boldsymbol{b}} \tag{46}$$

Recall from the optical flow constraint equation that

$$\frac{DI}{Dt} = I_t + \boldsymbol{g}^T \boldsymbol{v} = \boldsymbol{0} \tag{47}$$

Solving this equation is tantamount to the minimization of the least-squares objective function with respect to motion flow,

$$\min_{\boldsymbol{v}} \iint_{W(\boldsymbol{x})} \left[I_t + \boldsymbol{g}^T \boldsymbol{v} \right]^2 \boldsymbol{x} d\boldsymbol{x} \tag{48}$$

If we have prior information that the motion flow $\boldsymbol{v}$ is smooth then the objective can be modified to include the regularization term with penalty λ

$$\min_{\boldsymbol{v}} \iint_{W(\boldsymbol{x})} \left[\underbrace{\left[I_t + \boldsymbol{g}^T \boldsymbol{v} \right]^2 \boldsymbol{x}}_{\text{data term}} + \underbrace{\lambda \left\| \frac{\partial \boldsymbol{v}^T}{\partial \boldsymbol{x}} \right\|_F}_{\text{regularization term}} \right] d\boldsymbol{x} \tag{49}$$

Setting $\lambda = 0$, the problem is solved by $A\boldsymbol{v} = \boldsymbol{b}$ where,

$$A = \iint_{W(\boldsymbol{x})} \boldsymbol{g} \cdot \boldsymbol{g}^T d\boldsymbol{x}, \quad \boldsymbol{b} = -\iint_{W(\boldsymbol{x})} \boldsymbol{g} I_t d\boldsymbol{x} \tag{50}$$

3 Optical Flow

Recall that the motion field is the velocity of the projection of 3D world point motion onto the 2D image plane, while optical flow is the apparent motion of brightness/intensity patterns in the image frames. Ideally we want optical flow to be equivalent to the motion flow, however this need not be the case, because apparent motion in the image frame can be caused by lighting changes without any actual motion. If one has uniformly rotating sphere (motion flow $\neq \mathbf{0}$) under fixed lighting (optical flow $= \mathbf{0}$ or in the case when a stationary sphere (motion flow $= \mathbf{0}$) under a moving illumination (optical flow $\neq \mathbf{0}$), the ideal condition is not satisfied.

The optical flow problem is the estimation of optical flow from 2 subsequent frames by estimating the apparent motion field $\boldsymbol{d}(\boldsymbol{x})$, where $\boldsymbol{x} = (x, y)$ and $\boldsymbol{d} = (d_x, d_y)$. We make the following assumptions:

1. *Brighness Constancy* - The projection of the 3D world point looks the same in every frame, and leads to the optical flow constraint equation.
2. *Small Motion* - Points do not move very far in between two frames. That is $\boldsymbol{d}$ is small. This means one could use first order Taylor series to linearize.
3. *Spatial Coherence* - Points move alike their neighbours, thus one tracks the optical flow of small regions in the frame rather than individual pixel, which sets in the notion of windows.

Let the point $\boldsymbol{x} = (x, y)$ at time $t-1$ displace by an amount $\boldsymbol{d} = (d_x, d_y)$. In time t it moves to point $(x + d_x, y + d_y)$, as shown in Figure 8.

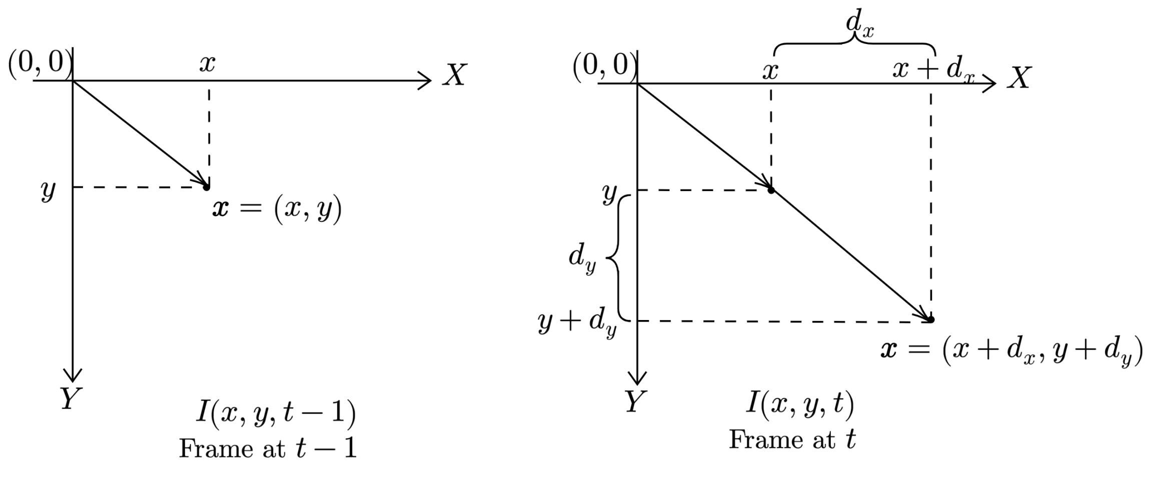


Figure 8: Displacement of the point $\boldsymbol{x}$ between 2 frames

The *Brightness Constancy assumption* means that the intensity of the 3D object and therefore its projection, remains the same while displaced between 2 frames.

$$I(x, y, t-1) = I(x + d_x, y + d_y, t) \tag{51}$$

Using the *Small motions assumption*, one can apply the 2D Taylor expansion in the above equation, with only linear terms,

$$\begin{aligned}
I(x, y, t-1) &= I(x + d_x, y + d_y, t) && (52)\\
I(x, y, t-1) &= I(x, y, t) + \underbrace{d_x I_x(x, y, t) + d_y I_y(x, y, t) + \dots}_{\text{Taylor expansion}} && (53)\\
I(\boldsymbol{x}, t-1) &= I(\boldsymbol{x}, t) + \boldsymbol{d} \cdot \boldsymbol{\nabla} I(\boldsymbol{x}, t) && (54)\\
I(\boldsymbol{x}, t-1) &- I(\boldsymbol{x}, t) \;-\; \boldsymbol{d} \cdot \boldsymbol{\nabla} I(\boldsymbol{x}, t) \;=\; \mathbf{0} && (55)\\
-I_t(\boldsymbol{x}, t) &- \boldsymbol{d} \cdot \boldsymbol{\nabla} I(\boldsymbol{x}, t) \;=\; \mathbf{0} && (56)
\end{aligned}$$

This leads to the *optical flow constraint equation*, which is 1 equation in 2 unknowns $d_x(\boldsymbol{x})$ and $d_y(\boldsymbol{x})$ for a given pixel $\boldsymbol{x}$.

$$\boldsymbol{d}(\boldsymbol{x}_i) \cdot \boldsymbol{\nabla} I(\boldsymbol{x}_i, t) = -I_t(\boldsymbol{x}_i, t) \tag{57}$$

We estimate the left hand side of Equation 57 from $I_t(\boldsymbol{x}_i, t)$ which can be got by differencing subsequent frames in the video. Once again, we note that we have fewer equations than unknowns. Now we apply the *Spatial Coherence assumption* that the pixel neighbours of a $\boldsymbol{x}_i$ have the same displacement $\boldsymbol{d}(\boldsymbol{x}_i)$.

$$d_x I_x(\boldsymbol{x}_i, t) + d_y I_y(\boldsymbol{x}_i, t) = -I_t(\boldsymbol{x}_i, t) \tag{58}$$

Suppose we take a window around pixel $\boldsymbol{x}_i$ of size $n \times n = 5 \times 5 = 25$, we can write the *spatial coherence* equation 58 for all 25 pixels in the window. Expressed in matrix form, it reduces to solving $A\boldsymbol{d} = \boldsymbol{b}$ where,

$$\underbrace{\begin{bmatrix} I_x(\boldsymbol{x}_1, t) & I_y(\boldsymbol{x}_1, t) \\ I_x(\boldsymbol{x}_2, t) & I_y(\boldsymbol{x}_2, t) \\ \vdots & \vdots \\ I_x(\boldsymbol{x}_3, t) & I_y(\boldsymbol{x}_3, t) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} d_x \\ d_y \end{bmatrix}}_{\boldsymbol{d}} = - \underbrace{\begin{bmatrix} I_t(\boldsymbol{x}_1, t) \\ I_t(\boldsymbol{x}_2, t) \\ \vdots \\ I_t(\boldsymbol{x}_3, t) \end{bmatrix}}_{\boldsymbol{b}} \tag{59}$$

This is solved using the least squares normal equation $(A^T A)\boldsymbol{d} = A^T \boldsymbol{b}$. On performing a similar exercise for all windows in the image, and using bilinear interpolation for intermediate displacements, we can obtain the optical flow displacement map, as seen earlier. The matrix computation for A^TA is given by

$$A^TA = \begin{bmatrix} \sum_x \sum_y \left(\frac{\partial I}{\partial x}\right)^2 & \sum_x \sum_y \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_x \sum_y \frac{\partial I}{\partial y}\frac{\partial I}{\partial x} & \sum_x \sum_y \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \tag{60}$$

The vector $A^T\boldsymbol{b}$ is obtained by computing

$$A^T\boldsymbol{b} = \begin{bmatrix} \sum_x \sum_y I_x I_t \\ \sum_x \sum_y I_y I_t \end{bmatrix} \tag{61}$$

Using this we solve the equation $\boldsymbol{d} = (A^TA)^{-1}A^T\boldsymbol{b}$ for $\boldsymbol{d} = (d_x, d_y)$ for a given pixel. The real eigenvalues λ_1, λ_2 of A^TA is an important metric for this algorithm to work. They should not be too small and one of the eigenvalue should not be too larger than the other. The matrix A^TA should be well conditioned such that if $\lambda_1 > \lambda_2$, then $\lambda_1/\lambda_2 \not>\not> 1$

We run into problems with the Lucas-Kande algorithm if the matrix A^TA is not invertible. In certain areas when the motion is large, one has to use iterative refinement. If our assumptions are violated, then too there arises errors in the optical flow problem. For instance if a point does not move like its neighbours. If brightness constancy does not hold, then we try to rectify by performing feature tracking from frame to frame following optical flow. The real challenge in such a case is to find good features to track.