Optical Flow

Shree K. Nayar

Monograph: FPCV-4-3

Module: Reconstruction II

Series: First Principles of Computer Vision

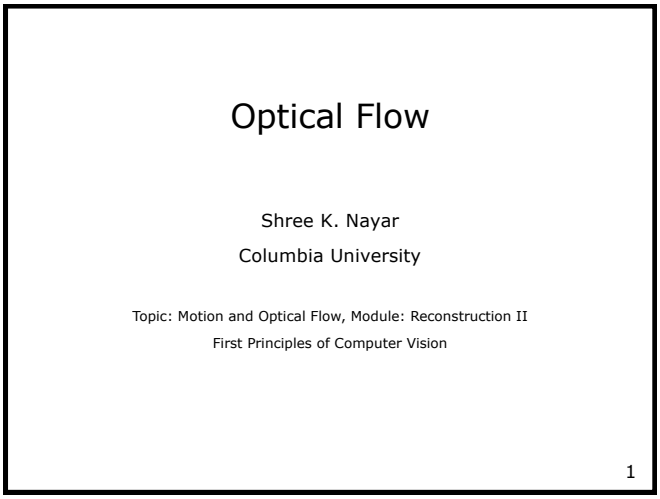
Computer Science, Columbia University

April, 2025

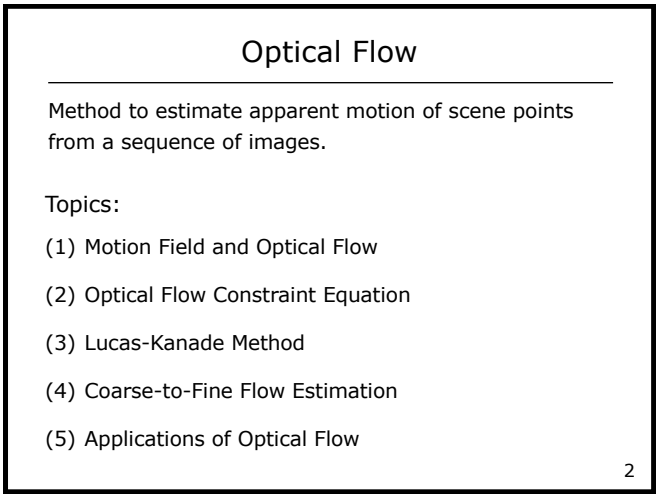
FPCV Channel

FPCV Website

In all the methods discussed thus far, we have assumed that the camera and the objects in the scene are stationary. However, we know that our visual world is highly dynamic due to either the motion of the observer and/or the objects in the scene. While we cannot directly measure the 3D motion of points in a scene, we can attempt to measure the 2D motion of their image projections. This is referred to as the problem of computing optical flow.

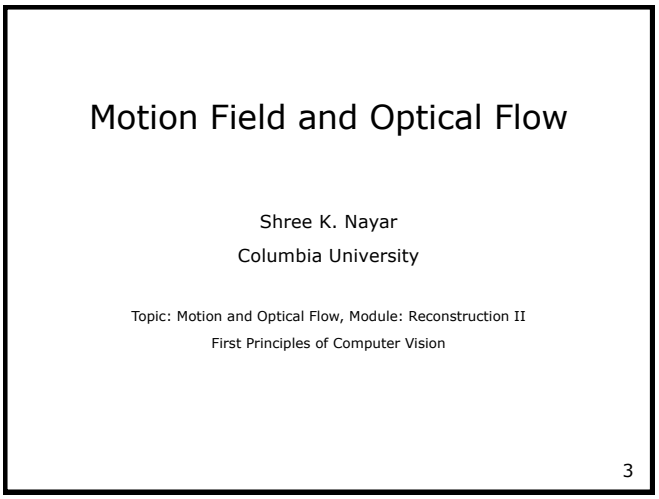


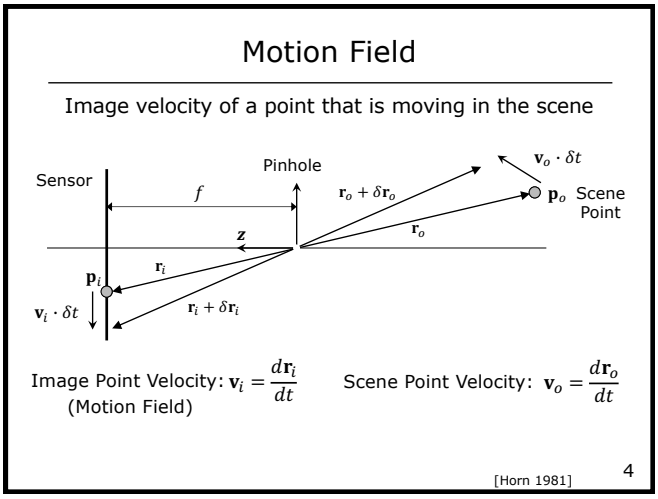
We will develop a method for estimating the motion of scene points in the image, from a sequence of images taken in quick succession. When scene points move with respect to the camera, the projection of their motion in 3D onto the 2D image plane is called the motion field. Although we can't directly measure the motion field, we can measure the motion of brightness patterns in the image, which is referred to as the optical flow. We discuss when optical flow corresponds to motion field, and when it does not. It turns out that we cannot uniquely determine optical flow at a pixel by simply looking at its brightness variation over time. However, we can derive an optical flow constraint equation that constrains the optical flow at a pixel. To uniquely solve for the optical flow at each pixel, the Lucas-Kanade method uses a small neighborhood of pixels.

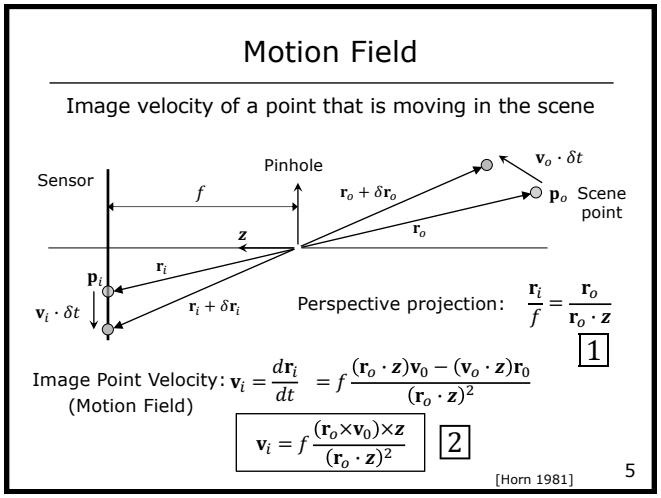


The optical flow constraint equation relies on the assumption that local spatial and temporal derivatives of the image can be accurately computed. This assumption falls apart when the motion is large, i.e., the motion of a scene is significant between two consecutive captured images. To solve this problem, we use an image representation called the resolution pyramid. We use the pyramid to first compute optical flow at a low resolution, and then propagate the computed flow to a higher resolution. Repeating this process, in steps, to the finest resolution results in a computed flow that includes a wide range of motions. Finally, we will look at a few interesting applications of optical flow.

Consider a point in the 3D scene which is moving in some direction. The projection of that motion onto the 2D image plane is referred to as the motion field corresponding to the scene point. Unfortunately, there is no guarantee that we can measure this motion field; all we can measure from captured images is the motion of brightness patterns in the image, which is referred to as the optical flow. We will now discuss the relationship between motion field and optical flow.



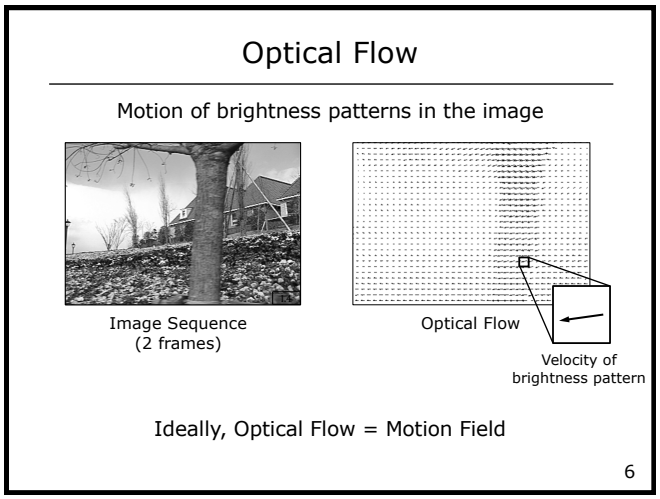




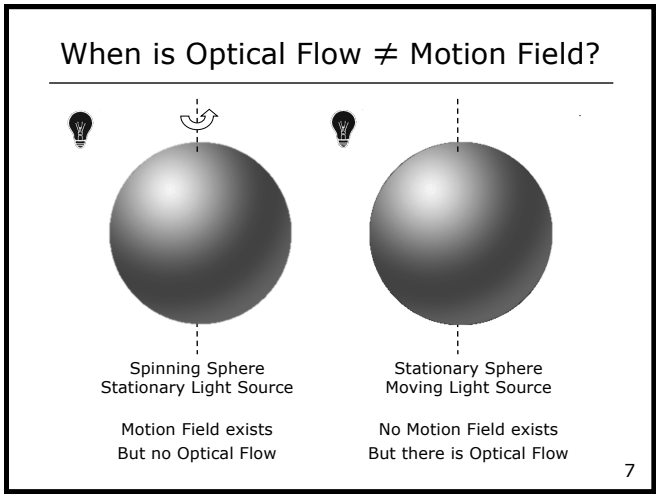
Consider the scenario shown on the left. We have a point $\mathrm{p_o}$ in the 3D scene, that maps via perspective projection to the point $\mathrm{p_i}$ in the 2D image. $\mathrm{p_o}$ and $\mathrm{p_i}$ can be represented by the vectors $\mathbf{r}_o$ and $\mathbf{r}_i$, respectively. Assume that in time δt, $\mathrm{p_o}$ moves to a new location which is given by the vector $\mathbf{r}_o + \delta\mathbf{r}_o$. The velocity $\mathbf{v}_o$ of $\mathrm{p_o}$ is the rate of change of the vector $\mathbf{r}_o$, and the velocity $\mathbf{v}_i$ of $\mathrm{p_i}$ is the rate of change of the vector $\mathbf{r}_i$. We want to derive an expression for the image velocity $\mathbf{v}_i$, which is called the motion field.

Using perspective projection (see 1), we get $\mathbf{r}_i$ divided by the effective focal length f equals $\mathbf{r}_o$ divided by the depth of $\mathrm{p_o}$, which is the dot product of $\mathbf{r}_o$ and $\boldsymbol{z}$. Using this expression to substitute for $\mathbf{r}_i$ in the expression for $\mathbf{v}_i$, and using the quotient rule of derivatives, we get expression 2 for the motion field $\mathbf{v}_i$.

Unfortunately, there is no guarantee that we can measure the motion field $\mathbf{v}_i$. We can only hope to measure the motion of brightness patterns in the image. Consider two images of the scene shown here taken in quick succession by a camera moving in the horizontal direction. From our lectures on stereo, we know that the motion of each point in scene between the two images would depend on its depth in the scene—the points closer to the camera, such as ones on the tree, would move faster than distant ones, such as those on the house. We would like to develop an algorithm that takes a small window around each pixel in the first image and figure out where it moved to in the second image. If we have a successful algorithm for doing that, we would get the result shown on the right. At each pixel, we have a vector that represents the motion of the local brightness pattern. This is called the optical flow vector—its length represents the speed of the point and its direction reveals the direction along which the point is moving. Ideally, optical flow is equal to the motion field. While this is often the case, it is not always true. Let us now look at a few cases where optical flow does not correspond to motion field.

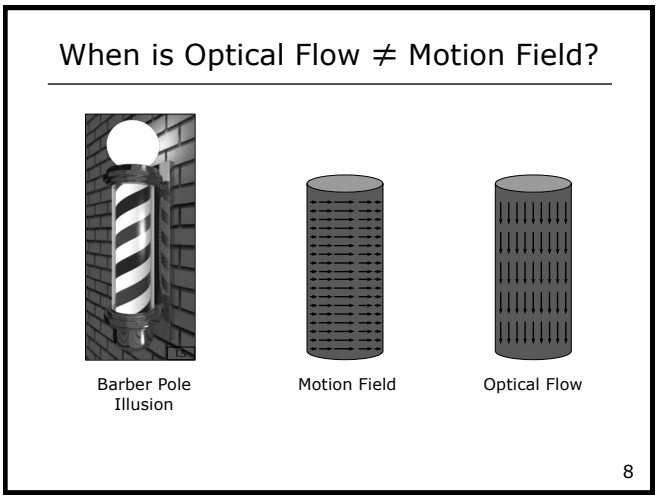


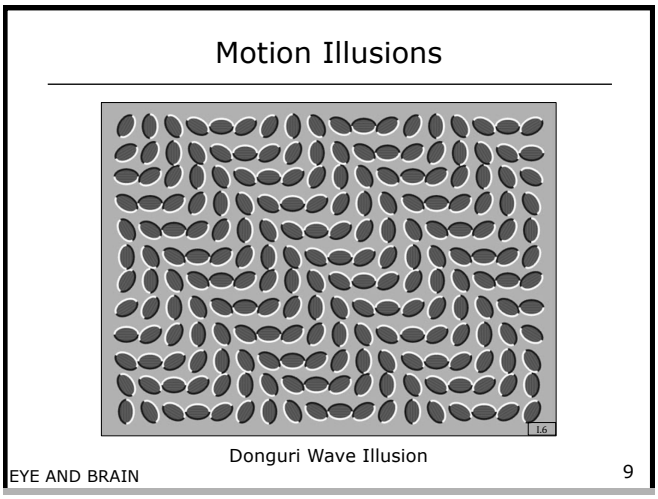
Consider the setup shown here. We have a sphere made of a single material, i.e., it has the same reflectance properties everywhere. Assume that the sphere is spinning about the vertical axis that passes through its center. The sphere is lit by a point light source, which results in some shading over its surface. Since the sphere is spinning, there is motion field, i.e., points on its surface are moving. However, despite this motion, the image of the sphere does not change, i.e., consecutive images taken by a stationary camera will be identical. Therefore, there is no optical flow. This is an example where we have motion field, but no optical flow.

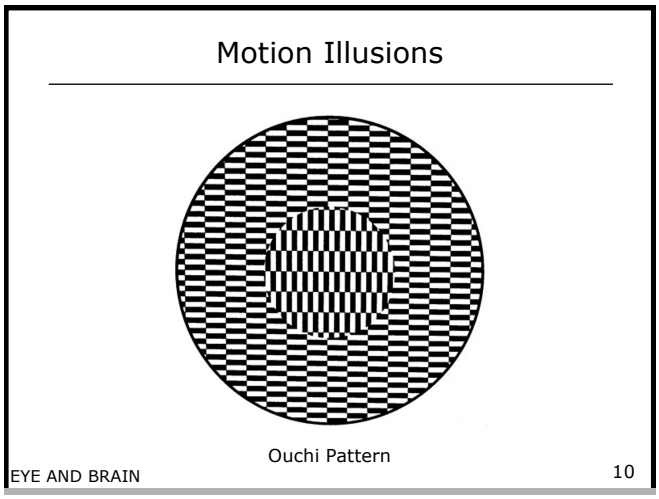


Now, consider the case on the right where the sphere is stationary and the light source moves around it. The motion of the source causes the shading of the sphere to change. In other words, brightness patterns in the image move as a function of time, but there is no actual physical motion of points on the sphere. This is an example where there is no motion field, but there is optical flow. These two examples show that there can be optical flow and no motion field, or there can be motion field and no optical flow.

The barber poll illusion is an interesting example where we have both optical flow and motion field, but they do not correspond to each other. In this case, we have a cylinder spinning about a vertical axis, so we know that the motion field is horizontal. However, if we look at the pattern in consecutive images (see online lecture video), the pattern appears to move in the vertical direction. Therefore, we have both motion field and optical flow, but they do not correspond to each other.







This phenomenon of incorrectly perceiving motion also applies to us, humans, and is referred to as motion illusion. Consider the Donguri wave Illusion on the left. This is obviously a static image, but if we move our eye around the image, we see that the leaves appear to move. That is, our visual system detects optical flow even though there is no motion field.

The example on the right is called the Ouchi pattern. In this case, we have a little disc in the center with a pattern and an outer ring with a different pattern. When we move our eyes around this image, the inner disc appears to move with respect to the outer ring. Once again, there is no motion field, but we perceive optical flow.

Now, let's consider the problem of estimating optical flow. We will first derive the optical flow constraint equation and then used it to develop an algorithm for estimating the optical flow at each point in the image.

Optical Flow Constraint Equation

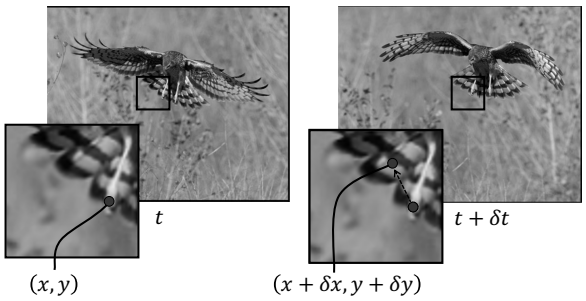
Shree K. Nayar
Columbia University

Topic: Motion and Optical Flow, Module: Reconstruction II
First Principles of Computer Vision

11

Consider the following scenario where we have two images of a bird in flight, taken in quick succession. The first image is taken at time t and second one at time $t + \delta t$, where δt is small. Let us now focus our attention on a single point which is on the foot of the bird. Assume that the location of the point is (x, y) in the first image. In the second image, taken at time $t + \delta t$, the point has moved to a new location $(x + \delta x,\ y + \delta y)$. The displacement of the point is therefore $(\delta x, \delta y)$. The speeds of the point in the x and y directions are $u = \delta x/\delta t$ and $v = \delta y/\delta t$, respectively. $(u,\ v)$ is the optical flow we wish to measure.

Optical Flow



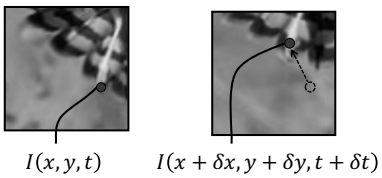
Displacement: $(\delta x, \delta y)$ Optical Flow: $(u, v) = \left(\frac{\delta x}{\delta t}, \frac{\delta y}{\delta t}\right)$

[Horn 1981]

12

In order to measure the flow $(u,\ v)$, we need to make a few assumptions. The first assumption is that the image brightness of a scene point remains constant over time. The problem becomes hard to solve if the brightness of the scene point changes significantly between consecutive images. Since the time between the images, δt, is small, this is a reasonable assumption. Under this assumption, the intensity of point (x, y) at time t is equal to the intensity of point $(x + \delta x, y + \delta y)$ at time $t + \delta t$.

Optical Flow Constraint Equation



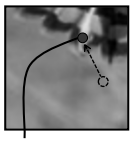
Assumption #1:
Brightness of image point remains constant over time

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

13

The second assumption we make is that the spatial displacements δx and δy are small and the timestep δt is also small. Since the images are taken in quick succession, as in the case of a video, this assumption is often valid. The assumption is important to our derivation of the optical flow constraint equation as it allows us to use a linear approximation via the Taylor series expansion.

Optical Flow Constraint Equation

$I(x, y, t)$ $I(x + \delta x, y + \delta y, t + \delta t)$

Assumption #2:

Displacement $(\delta x, \delta y)$ and time step δt are small

14

The Taylor series expansion applies to any infinitely differentiable function, i.e., one for which all the derivatives exist. If $f(x)$ is such a function, then $f(x + \delta x)$ can be expressed as $\boxed{1}$. Now, if δx is small, we can assume the second and higher order terms are very small and can be assumed to be zero. Then $f(x + \delta x)$ can be expressed in terms of $f(x)$ and its first-order derivative. This is called the first-order Taylor approximation and it is linear in the displacement δx. If f is a function of three variables, x, y, and t, the linear approximation is given by $\boxed{2}$.

Taylor Series Expansion

Expand a function as an infinite sum of its derivatives

$$f(x + \delta x) = f(x) + \frac{\partial f}{\partial x}\delta x + \frac{\partial^2 f}{\partial x^2}\frac{\delta x^2}{2!} + \cdots + \frac{\partial^n f}{\partial x^n}\frac{\delta x^n}{n!} \quad \boxed{1}$$

If δx is small:

$$f(x + \delta x) = f(x) + \frac{\partial f}{\partial x}\delta x + \boxed{O(\delta x^2)} \rightarrow \text{Almost Zero}$$

For a function of three variables with small $\delta x, \delta y, \delta t$:

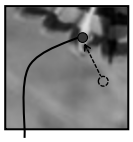
$$f(x + \delta x, y + \delta y, t + \delta t) \approx f(x, y, t) + \frac{\partial f}{\partial x}\delta x + \frac{\partial f}{\partial \cdot}\delta y + \frac{\partial f}{\partial t}\delta t$$

$\boxed{2}$

MATH PRIMER 15

Applying the above linear approximation to the image brightness I in the second image allows us to express it in terms of the brightness in the first image and its derivatives with respect to the x direction, y direction, and time t. These derivates are denoted I_x, I_y, and I_t, respectively.

Optical Flow Constraint Equation

$I(x, y, t)$ $I(x + \delta x, y + \delta y, t + \delta t)$

Assumption #2:

Displacement $(\delta x, \delta y)$ and time step δt are small

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t$$

$$\boxed{I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + I_x\delta x + I_y\delta y + I_t\delta t}$$

16

Slides 13 and 16 give us the two equations shown at the top here. Subtracting them gives us expression 1. Simplifying this equation and substituting u and v for the derivatives of x and y with respect to time, respectively, gives us the optical flow constraint equation. Given two images taken in quick succession, we can find the three derivatives, I_x, I_y, and I_t, for each pixel using finite differences. Plugging these values into the optical flow constraint equation gives us a straight-line equation with u and v as the variables.

Optical Flow Constraint Equation

$I(x+\delta x, y+\delta y, t+\delta t) = I(x,y,t)$ ..... (1)

$I(x+\delta x, y+\delta y, t+\delta t) = I(x,y,t) + I_x\delta x + I_y\delta y + I_t\delta t$ ..... (2)

Subtract (1) from (2): $I_x\delta x + I_y\delta y + I_t\delta t = 0$ 1

Divide by δt and take limit as $\delta t \to 0$: $I_x\frac{\partial x}{\partial t} + I_y\frac{\partial y}{\partial t} + I_t = 0$

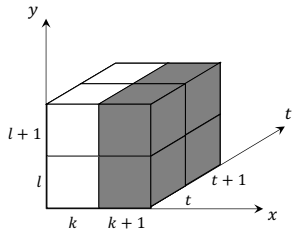
Constraint Equation: $I_x u + I_y v + I_t = 0$ (u, v): Optical Flow

(I_x, I_y, I_t) can be easily computed from two frames

17

Now, let us look at how we would compute I_x, I_y, and I_t given two images taken in quick succession. For simplicity, we consider a 2x2x2 cube of pixels. We can find I_x by subtracting the average of the four brightness values in the black pixels from the average of the four brightness values in the white pixels. Similarly, we can calculate I_y and I_t. Therefore, given two consecutive frames in a video, we can calculate I_x, I_y, and I_t for each pixel.

Computing Partial Derivatives I_x, I_y, I_t



$I_x(k,l,t) =$

$\frac{1}{4}[I(k+1,l,t) + I(k+1,l,t+1) + I(k+1,l+1,t) + I(k+1,l+1,t+1)]$

$-\frac{1}{4}[I(k,l,t) + I(k,l,t+1) + I(k,l+1,t) + I(k,l+1,t+1)]$

Similarly find $I_y(k,l,t)$ and $I_t(k,l,t)$

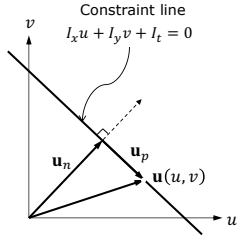
18

Shown here is a geometric interpretation of the optical flow constraint equation. Consider the u-v space show on the right. Let the optical flow at a particular image point be $\mathbf{u}(u,v)$. We want to find the vector **u** but, unfortunately, we only have the straight line given by the constraint equation. We know that **u** lies on this line, but we do not know exactly where on the line it lies. This makes the optical flow estimation problem an under-constrained problem. We can split up the unknown optical flow vector **u** into two

Geometric Interpretation

For any point (x,y) in the image, its optical flow (u,v) lies on the line:

$I_x u + I_y v + I_t = 0$



Optical Flow can be split into two components.

$\mathbf{u} = \mathbf{u}_n + \mathbf{u}_p$

$\mathbf{u}_n$: Normal Flow

$\mathbf{u}_p$: Parallel Flow

19

orthogonal components, the normal flow $\mathbf{u}_n$ which is normal to the constraint line, and the parallel flow $\mathbf{u}_p$ which is parallel to the constraint line.

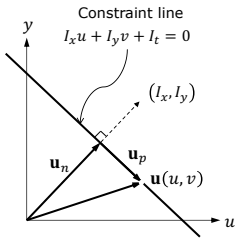
It turns out we can compute the normal flow $\mathbf{u}_n$. The direction of $\mathbf{u}_n$ is given by a unit vector that is perpendicular to the constraint line. The magnitude of $\mathbf{u}_n$ is the shortest distance of the origin from the constraint line. Therefore, given the constraint line, the normal flow can always be computed. However, we have no way of computing the parallel flow needed to find the optical flow vector $\mathbf{u}_p$. Note that both $\mathbf{u}_n$ and $\mathbf{u}_p$ are needed to determine the optical flow **u.**

Normal Flow

Direction of Normal Flow:

Unit vector perpendicular to the constraint line:

$$\hat{\mathbf{u}}_n = \frac{(I_x, I_y)}{\sqrt{I_x^2 + I_y^2}}$$



Magnitude of Normal Flow:

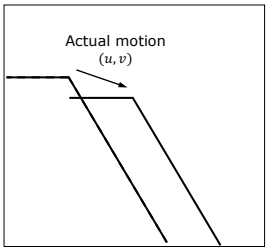
Distance of origin from the constraint line:

$$|\mathbf{u}_n| = \frac{|I_t|}{\sqrt{I_x^2 + I_y^2}}$$

$$\mathbf{u}_n = \frac{|I_t|}{(I_x^2 + I_y^2)}(I_x, I_y)$$

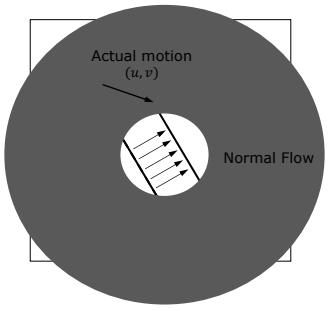
20

Aperture Problem



22

Aperture Problem



Locally, we can only determine normal flow!

23

It turns out that the under-constrained nature of the optical flow problem is applicable to us, humans, as well. Consider the moving object shown in slide 22. The direction of motion (arrow) is towards the bottom-right corner of the slide. This is obvious when we view the entire object. However, since flow can vary significantly over the image, we need to compute flow locally using a small window of attention, or "keyhole," like the one shown in slide 23. When we view the motion in slide 22 through this keyhole, we perceive the local motion to be in the direction normal to the linear boundary of the object, which is different from the actual motion in slide 22. In other words, we humans too are not able to always measure the actual flow. Locally, we can only perceive the normal flow. This is referred to as the aperture problem associated with the estimation of optical flow.

So, we have our single constraint equation with two unknowns. Clearly, to solve for flow we need additional constraints.

Optical Flow is Under Constrained

Constraint Equation: $I_x u + I_y v + I_t = 0$

2 unknowns, 1 equation.

We need additional constraints.

24

To address the above problem, the Lucas-Kanade method assumes that the optical flow for all points within each small window in the image is the same.

Lucas-Kanade Method

Shree K. Nayar
Columbia University

Topic: Motion and Optical Flow, Module: Reconstruction II
First Principles of Computer Vision

25

Lucas-Kanade Solution

Assumption: For each pixel, assume Motion Field, and hence Optical Flow (u, v), is constant within a small neighborhood W.

That is for all points $(k, l) \in W$:

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

[Lucas 1981]

26

Lucas-Kanade Solution

For all points $(k, l) \in W$: $I_x(k,l)u + I_y(k,l)v + I_t(k,l) = 0$

Let the size of window W be $n \times n$

In matrix form:

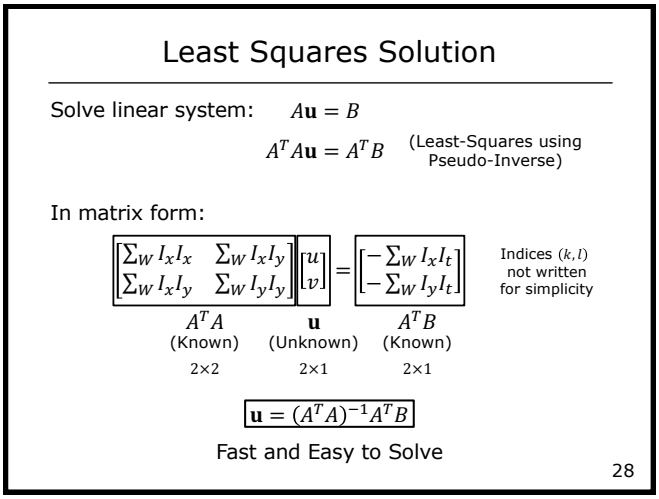
$$\underset{\substack{A \\ (\text{Known}) \\ n^2 \times 2}}{\begin{bmatrix} I_x(1,1) & I_y(1,1) \\ I_x(k,l) & I_y(k,l) \\ \vdots & \vdots \\ I_x(n,n) & I_y(n,n) \end{bmatrix}} \underset{\substack{\mathbf{u} \\ (\text{Unknown}) \\ 2 \times 1}}{\begin{bmatrix} u \\ v \end{bmatrix}} = \underset{\substack{B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

n^2 Equations, 2 Unknowns: Find Least Squares Solution

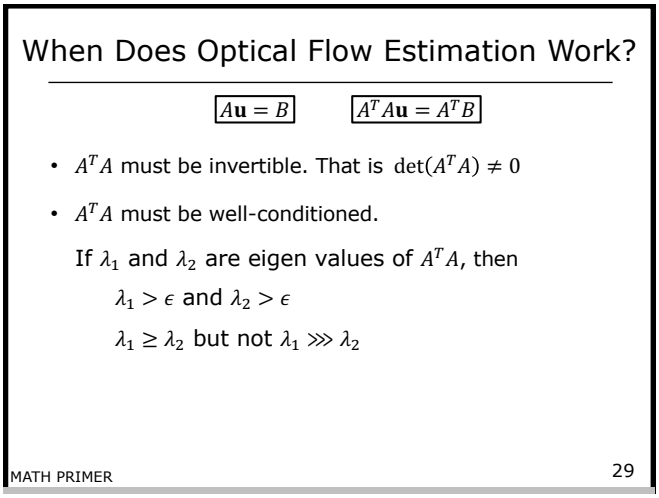
27

We assume that for each pixel in the image, the motion field, and hence the optical flow, is constant within a small neighborhood around the pixel. Consider the small patch *W* in the image on the left. The constraint equation for a single pixel (k, l) in *W* is given at the bottom. We get one such equation for each pixel in *W.* Since the optical flow within *W* is constant, we get a large system of equations, which is shown on the right. If *W* has n^2 pixels, we have n^2 equations and just two unknowns, (u, v). Note that if the patch *W* is textured, pixels within it are likely to differ in their derivatives I_x and I_y. In other words, we can expect our system of equations to have enough linearly independent equations to solve for (u, v). We know how to solve this overdetermined system of equations using the least squares method.

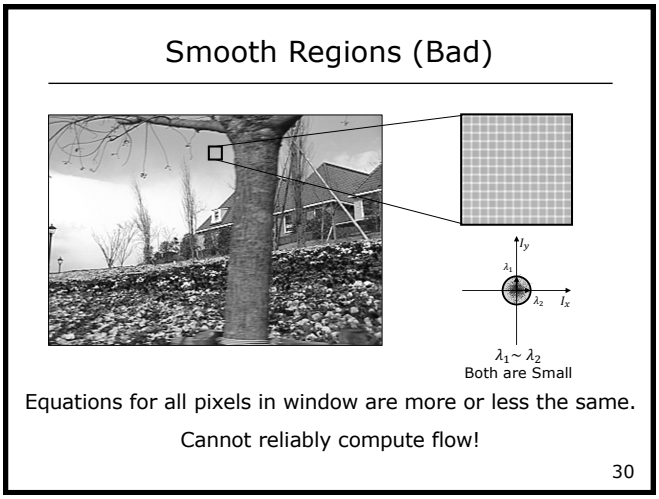
The matrix form of the above system of equations is $A\mathbf{u} = B$. We multiply both sides of the equation by A^T and then multiply both sides by the inverse of $A^T A$ to solve for **u**. This solution is computationally efficient because $A^T A$ is a 2x2 matrix. The elements of $A^T A$ are functions of the spatial derivatives and the time derivatives only appear in B.



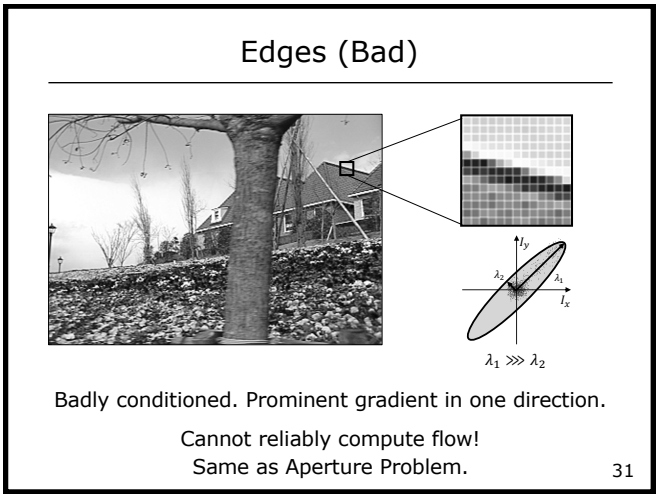
We know that this approach will not work when $A^T A$ is not invertible. More generally, it will produce reliable results only when the matrix $A^T A$ is well-conditioned, which is true when both the eigenvalues of $A^T A$ are significant enough, i.e., neither eigenvalue is close to zero and neither is significantly greater than the other.



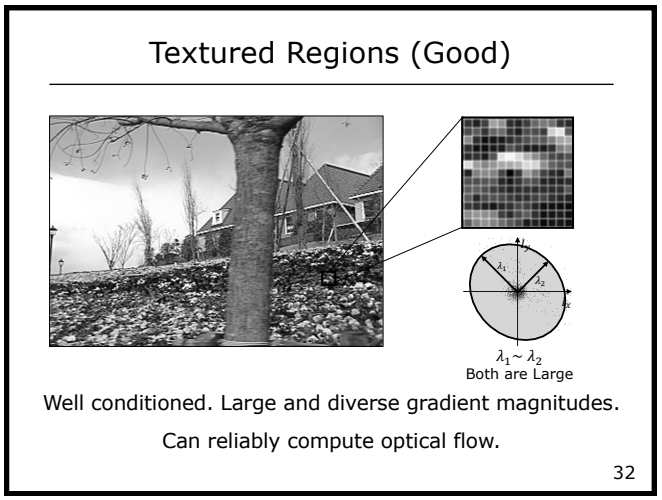
Let's look at how the conditioning of A^TA relates to real images. Consider the patch in the sky shown here. We take the spatial gradients I_x and I_y at each pixel in the patch and plot them as shown on the right. We fit an ellipse to the distribution of points in I_x-I_y space. The semi-major axis and semi-minor axis of the best fit ellipse correspond to the eigenvalues λ_1 and λ_2 of the matrix A^TA. Since the sky is texture-less, the points in I_x-I_y space form a compact cluster around the origin. The best fit ellipse in this case is a small circle. Hence both λ_1 and λ_2 are small, indicating that the matrix A^TA is not well-conditioned and hence optical flow cannot be computed reliably.



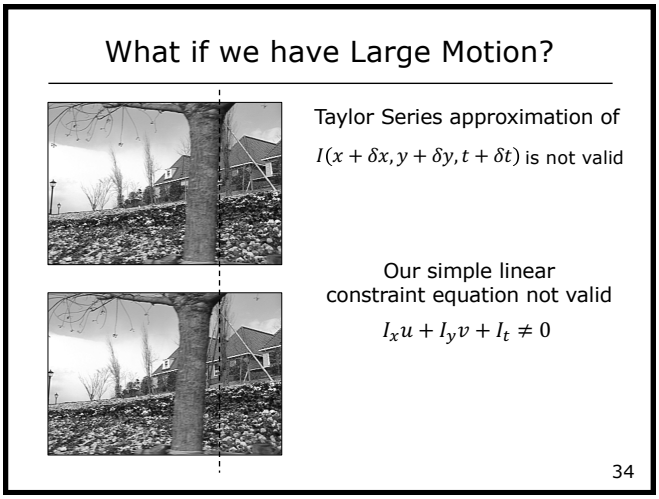
Shown here is a patch that sits on the edge of the roof of the house. The roof edge has a strong gradient in one direction but a weak gradient in the orthogonal direction. Note that moving the patch along the edge will not significantly change its content. This gives rise to the aperture problem we discussed in slide 23. In this case, the best fit ellipse in I_x- I_y space is thin and long, yielding a large value for λ_1 and a small value for λ_2. Again, the matrix A^TA is not well-conditioned, and hence optical flow cannot be computed reliably.



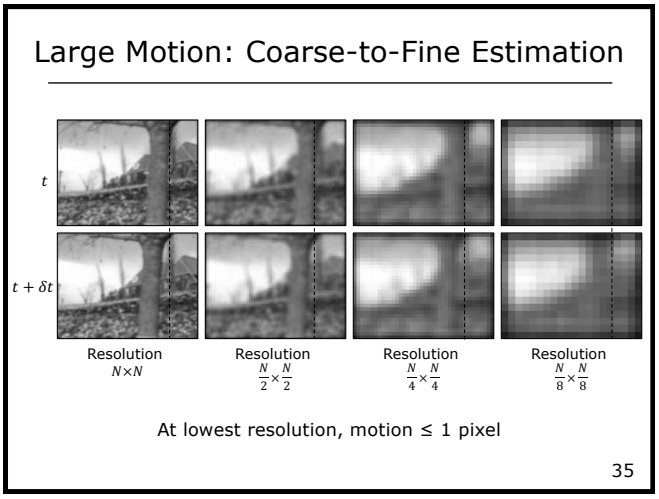
The Lucas-Kanade method works when you have a richly textured patch with strong gradients in various directions. For the patch on the ground shown here, we get a well-scattered distribution of points in I_x-I_y space, resulting in a large ellipse and hence large values for both λ_1 and λ_2. In this case, A^TA is well-conditioned and the computed optical flow is reliable.



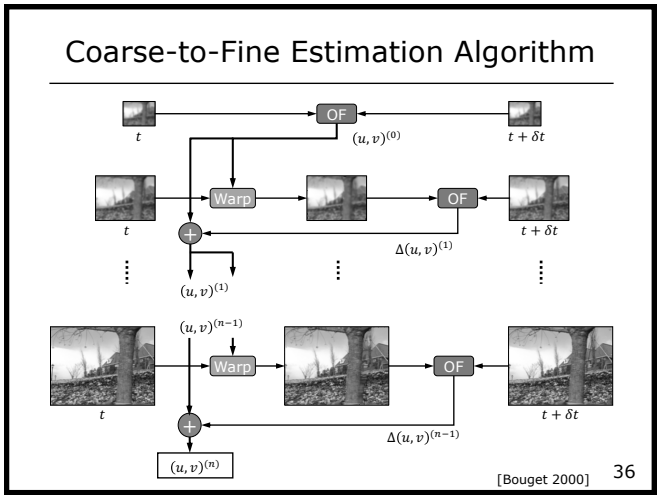
When we derived the optical flow constraint equation, we assumed that the displacements δx and δy over time δt are small. What happens if we have large motion between consecutive images? Consider the two images shown here. Assume that the camera's movement is the only cause of motion in this case. Since the tree is close to the camera, its motion is substantial (tens of pixels). In this case, we can't assume that δx and δy are small. Hence, the Taylor series approximation we made in slide 16 is no longer valid, making our optical flow constraint equation invalid.

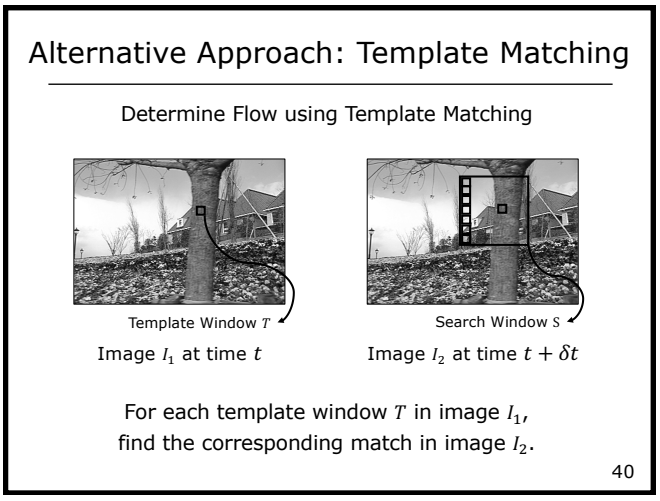


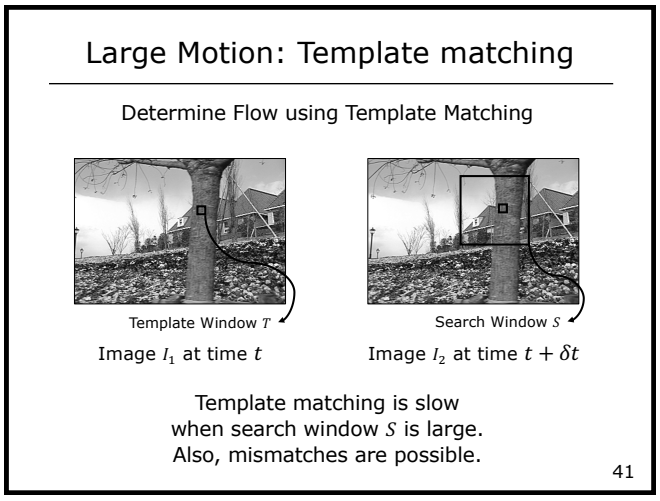
We can address the problem of large motion using a resolution pyramid. Assume that we have two images taken at time t and $t+\delta t$. The resolution of each captured image is $N \times N$. We now compute lower-resolution versions ($\frac{N}{2} \times \frac{N}{2}$, $\frac{N}{4} \times \frac{N}{4}$, $\frac{N}{8} \times \frac{N}{8}$, etc.) of these two images, as shown here. At each resolution, each pixel is the average of four pixels in the higher resolution image. An important thing to note is that if the motion of a scene point in the original $N \times N$ image is 8 pixels, its motion is 1 pixel in the $\frac{N}{8} \times \frac{N}{8}$ image. We can therefore create a resolution pyramid such that at its lowest resolution the motion at all pixels is less than 1 pixel. At that resolution, the optical flow constraint equation becomes valid again.



Using the resolution pyramid, we develop a coarse-to-fine optical flow estimation algorithm. We start with the lowest resolution and apply the Lucas–Kanade algorithm to the images taken at time t and $t + \delta t$. We then use this computed optical flow to warp the image at time t in the next higher resolution. This warping is done by "pushing" each pixel in the direction of its computed flow. Since the optical flow was computed at a lower resolution, the warped image is not going to be exactly the same as the corresponding pyramid image at time $t + \delta t$, but it will be close to it. Then, we compute the optical flow between the warped image and the corresponding pyramid image at time $t + \delta t$. This flow computation works well because the flow between these two images is small, since the large motions are taken care of by the warping. We now add this newly computed optical flow to the previous one and use the resultant flow to warp the image at time t at the next higher resolution. This process is repeated all the way down to the highest resolution, at which point we have the final flow at every pixel. A few examples of optical flow computed using this method can be seen in the online video lecture.







There is another brute-force approach to compute optical flow that is based on template matching. Consider the two images I_1 and I_2 shown here, taken at time t and $t + \delta t$, respectively. For each pixel in image I_1, we can take a small window T around it and use it as a template to find the best matching window in image I_2. The difference between the location of the original window in I_1 and the best match window in I_2 gives us the optical flow vector. The problem with using this approach is that it could lead to mismatches—a patch in I_1 could match multiple patches in I_2 equally well.

We will conclude by discussing a few applications of optical flow.

Applications of Optical Flow

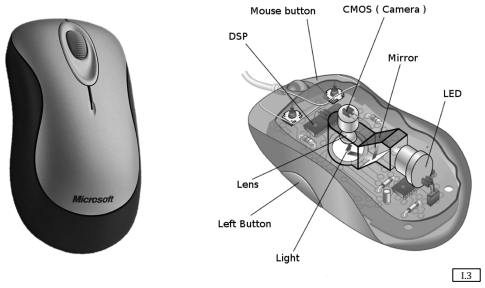
Shree K. Nayar
Columbia University

Topic: Motion and Optical Flow, Module: Reconstruction II
First Principles of Computer Vision

42

The optical mouse has a complete computer vision system inside it. It has a light source, which illuminates the surface that the mouse sits on. This surface is imaged by a camera that is low in resolution (e.g., 64x64 pixels) but high in frame-rate (e.g., 2000 frames a second). Successive images captured by the camera are used to estimate optical flow, i.e., the motion of the mouse between successive frames. This information is used, for example, to control the cursor on a computer screen.

Optical Mouse

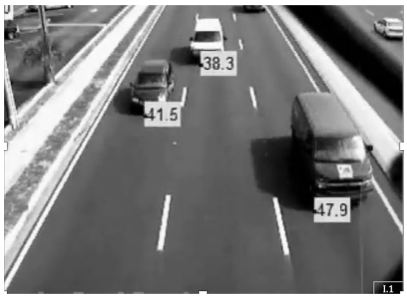


Estimating Mouse Movements

43

Here we see the use of optical flow for traffic monitoring. We have a stationary camera looking down at a highway. We know a-priori the plane of the highway in the camera's coordinate frame. The computed aggregate flow of each vehicle is projected onto the highway plane to determine the speed of the vehicle.

Traffic Monitoring

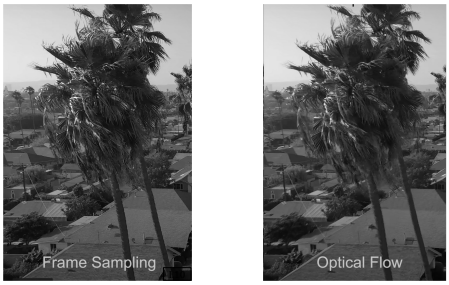


Finding Velocities of Vehicles

44

Another interesting application of optical flow is video retiming—increasing the frame-rate of an already captured video. Fast moving objects in a scene can cause a captured video to appear choppy. To compute a smooth video that appears like it was taken with a high-speed camera, we compute the optical flow between successive frames of the original video and use the flow vectors to interpolate intermediate frames. The end result is a smoother video. This method is widely used in smartphones to create slow-motion effects.

Video Retiming



Optical Flow is used to determine the intermediate frames to produce slow-motion effect.

45

Optical flow can be used to remove camera shake and stabilize a video. Imagine taking a video of a scene while walking through it. The video is likely to be shaky. Optical flow is computed between consecutive images of the video. The dominant flow, which is how most points in the scene are moving, is then computed. The dominant flow usually corresponds to the motion of the background of the scene. This dominant flow is then compensated for to reduce the shake in the video. This process is called stabilization.

Image Stabilization

Captured Video Stabilized Video

Optical Flow is used to remove camera shake.

[Liu 2014]

46

Another interesting application is face tracking. Here, we compute the flow of points on a face. This gives us a lot of useful information that can be used to figure out when a person blinks, how the lips are moving, recognize the person's expression, etc.

Face Tracking

Tracking of Facial Features

47

References and Credits

Shree K. Nayar

Columbia University

Topic: Motion and Optical Flow, Module: Reconstruction II

First Principles of Computer Vision

49

References

[Barron 2005] J. L. Barron, D. J. Fleet, and S. Beauchemin, "Performance of optical flow techniques". IJCV, 2005.

[Black 1993] M. J. Black and P. Anandan, "A framework for the robust estimation of optical flow." ICCV, 1993.

[Bouget 2000] J. Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker", Intel Corporation 2000.

[Brox 2004] T. Brox, A. Bruhn, N. Papenberg, and J.Weickert, "High accuracy optical flow estimation based on a theory for warping." ECCV, 2004.

[Horn 1981] B. K. P. Horn and B. G. Schunck, "Determining optical flow." Artificial Intelligence, 1981.

[Liu 2014] S. Liu, L. Yuan, P. Tan and J. Sun, "SteadyFlow: Spatially Smooth Optical Flow for Video Stabilization." CVPR 2014.

[Lucas 1981] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision". Proceedings of Imaging understanding workshop, 1981.

50

Image Credits

I.1 http://www.youtube.com/watch?v=49BQVszT5mI . Janaka Liyanage.

I.2 Simon Baker. Used with permission.

I.3 http://howthingswork.org/electronics-how-an-optical-mouse-works/

I.5 https://commons.wikimedia.org/wiki/File:Barber-pole-0 Barber pole illusion. Licensed under CC BY-SA.

I.6 Akiyoshi Kitaoka. Used with permission.

I.7 Robot Vision. B.K.P. Horn. McGraw-Hill 1986. Used with permission.

I.8 http://vimeo.com/30779242. Alex M. White.

I.9 Hidetoshi Shimodaira. Used with permission.

I.10 https://www.youtube.com/watch?v=HE9NkiSjXRY . Austin Caccavo.

51

Acknowledgements: Thanks to Pranav Sukumar, Ayush Sharma and Nikhil Nanda for their help with transcription, editing and proofreading.

References

[Barron 2005] J. L. Barron, D. J. Fleet, and S. Beauchemin, Performance of optical flow techniques, IJCV, 2005.

[Black 1993] M. J. Black and P. Anandan, A framework for the robust estimation of optical flow, ICCV, 1993.

[Bouguet 2000] J. Y. Bouguet, Pyramidal Implementation of the Lucas Kanade Feature Tracker, Intel Corp., 2000.

[Brox 2004] T. Brox, A. Bruhn, N. Papenberg, and J.Weickert, High accuracy optical flow estimation based on a theory for warping, ECCV, 2004.

[Horn 1981] B. K. P. Horn and B. G. Schunck, Determining optical flow, Artificial Intelligence, 1981.

[Liu 2014] S. Liu, L. Yuan, P. Tan, and J. Sun, SteadyFlow: Spatially Smooth Optical Flow for Video Stabilization, CVPR 2014.

[Lucas 1981] B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision, Proceedings of Imaging understanding workshop, 1981.

[Nayar 2022B] Image Formation, Nayar, S. K., Monograph FPCV-1-1, First Principles of Computer Vision, Columbia University, New York, February 2022.

[Nayar 2022C] Image Sensing, Nayar, S. K., Monograph FPCV-1-2, First Principles of Computer Vision, Columbia University, New York, February 2022.

[Nayar 2022E] Image Processing I, Nayar, S. K., Monograph FPCV-1-4, First Principles of Computer Vision, Columbia University, New York, March 2022.

[Nayar 2022F] Image Processing II, Nayar, S. K., Monograph FPCV-1-5, First Principles of Computer Vision, Columbia University, New York, March 2022.

[Nayar 2022G] Edge Detection, Nayar, S. K., Monograph FPCV-2-1, First Principles of Computer Vision, Columbia University, New York, May 2022.

[Nayar 2025K] Structure from Motion, Nayar, S. K., Monograph FPCV-4-4, First Principles of Computer Vision, Columbia University, New York, April 2025.

[Nayar 2025L] Object Tracking, Nayar, S. K., Monograph FPCV-5-1, First Principles of Computer Vision, Columbia University, New York, May 2025.