Lecture Notes: Optical Flow

Subhransu Maji

April 4, 2025

Contents

1 Overview

In the next few lectures, we will learn about optical flow, which refers to the apparent motion of brightness patterns in images. Motion is an important cue for recognition—and sometimes the only cue! Think of a camouflaged animal that becomes easy to spot when it moves. Motion also provides valuable information about scene structure, such as the depth of objects, as well as their direction and speed. It is also used in modern video compression formats such as MPEG, and in applications like video stabilization.

We will begin by defining the concept of optical flow between a pair of images and developing a simple method to estimate it. We'll then use this approach to compute depth maps from stereo image pairs, assuming knowledge of certain camera parameters, and explore additional applications of optical flow. Finally, we'll discuss some of the limitations of estimating depth using optical flow, and how these can be addressed by designing better cameras.

2 Motion field and Optical flow

The motion field is the projection of 3D scene motion onto the image plane. Optical flow, on the other hand, refers to the apparent motion of brightness patterns within an image. Ideally, optical flow would match the motion field, but this is not always the case. Apparent motion can occur without any actual 3D motion—for example, due to changes in lighting conditions. Think of shadows or highlights moving as the lighting direction changes.

2.1 Estimating Optical Flow

Given two frames $I(x, y, t-1)$ and $I(x, y, t)$, our goal is to estimate the apparent motion field $u(x, y)$ and $v(x, y)$ between them. The notation here implies that a pixel (x, y) in $I(x, y, t-1)$ is displaced by (u, v) to its new position $(x+u, y+v)$ in the image $I(x, y, t)$. In other words, $u(x, y)$ and $v(x, y)$ represent the horizontal and vertical components of the flow, respectively.

2.1.1 The brightness constancy constraint

Assuming that the pixel appearance is unchanged leads to the **brightness constancy constraint**.

$$I(x, y, t-1) = I(x + u(x, y), y + v(x, y), t). \tag{1}$$

For small displacements we can linearize the right side of the equation using a Taylor series expansion:

$$I(x, y, t-1) \approx I(x, y, t) + \frac{\partial I}{\partial x} u(x, y) + \frac{\partial I}{\partial y} v(x, y). \tag{2}$$

Rearranging the equation gives us:

$$\frac{\partial I}{\partial t} + \frac{\partial I}{\partial x} u(x, y) + \frac{\partial I}{\partial y} v(x, y) \approx 0, \tag{3}$$

where $\frac{\partial I}{\partial t} = I(x, y, t) - I(x, y, t-1)$. Denoting I_x, I_y, and I_t as the partial derivatives of the image I with respect to x, y, and t, respectively, and dropping the dependence on (x, y) allows us to shorten the above as:

$$I_x u + I_y v + I_t \approx 0. \tag{4}$$

2.1.2 Understanding the brightness constancy constraint

Thus, the brightness constancy assumption imposes a linear constraint on the optical flow (u, v) at each point. But what does this constraint imply? Specifically, we have a single linear equation with two unknowns, which means there are infinitely many possible solutions.

If we denote $\nabla I = (I_x, I_y)$ as the image gradient vector, the constraint can be written as:

$$\nabla I \cdot (u, v) + I_t \approx 0. \tag{5}$$

This equation depends only on the dot product between the gradient vector ∇I and the flow (u, v). As a result, the component of the flow that is perpendicular to the gradient—that is, parallel to image edges—remains undetermined.

2.1.3 Adding more constraints

To estimate optical flow, we need additional constraints. One common approach is to assume *spatial coherence*—that is, we divide the image into small neighborhoods and assume that all pixels within a neighborhood share the same flow. The Lucas-Kanade[1] optical flow estimation method assumes rectangular windows. For example, a 5×5 window gives as 25 linear constraints corresponding to each pixel $\mathbf{x}_i = (x_i, y_i)$ in the window, i.e.,

$$\nabla I(\mathbf{x}_i) \cdot (u, v) + I_t(\mathbf{x}_i) = 0. \tag{6}$$

We can then solve for (u, v) using in a least squares manner:

$$\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_n) & I_y(\mathbf{x}_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{x}_1) \\ I_t(\mathbf{x}_2) \\ \vdots \\ I_t(\mathbf{x}_n) \end{bmatrix}. \tag{7}$$

2.2 The aperture problem

But when is this system solvable? To do this let's analyze the structure of the linear system.

If we write the system of linear equations above as $Ax = b$, the least-squares solution satisfies $(A^\top A)x = A^T b$:

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}. \tag{8}$$

[1] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the International Joint Conference on Artificial Intelligence, pp. 674–679, 1981.

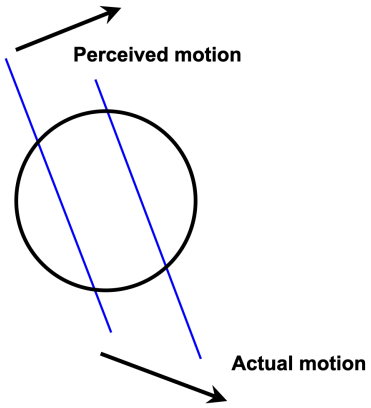

Figure 1: **The aperture problem**. The line moves diagonally, but the perceived motion is horizontal if the analysis is constrained to be within the circle. The circle or the aperture is meant to denote the scale of the local analysis.

The least-squares solution is given by $x = (A^\top A)^{-1} A^\top b$. The matrix $M = A^\top A$ is the familiar *second moment matrix*. The system is solvable when M is invertible, which occurs when both of its eigenvalues are large. Not surprisingly, this is exactly the condition used in *corner detection*. Optical flow is easiest to estimate at corners, while at edges or in flat regions, flow estimation becomes ambiguous.

This ambiguity is often referred to as the *aperture problem*. It is difficult to determine the true motion of a point when viewed through a small hole—an aperture—as different motion directions can appear identical.

For example, in Figure 1, the line moves diagonally, but the perceived motion within the circle is horizontal. This is also the basis of the *barber pole illusion*[2], where we perceive the motion to be vertical even though the actual motion is horizontal.

One way to address the aperture problem is to enlarge the aperture (or window size) to capture more context. However, doing so naively can lead to blocky or over-smoothed motion estimates.

In addition to the aperture problem, optical flow estimation can fail for several other reasons:

- If the motion is large, the Taylor series approximation used in the derivation becomes inaccurate.
- Assuming constant flow within a window breaks down at motion boundaries.
- The brightness constancy assumption may not hold under illumination changes or non-Lambertian surfaces.

2.3 Alternate optical flow techniques

The method introduced in the lecture was originally proposed by **Lucas and Kanade** in their seminal 1981 paper. While effective in many cases, this baseline approach has several limitations, particularly in handling large motions, motion boundaries, and low-texture regions.

To address these shortcomings, a number of extensions have been proposed:

- **Iterative refinement** and **coarse-to-fine estimation** strategies help estimate motion in less textured regions and allow the method to handle large displacements more effectively.
- **Local feature matching** techniques are used to establish correspondences across larger distances, improving robustness to large motions.
- **Image segmentation-based methods** group pixels into coherent regions, helping to preserve motion boundaries and reduce smoothing across object edges.

Beyond these extensions, entirely different families of approaches have also been developed. One such example is the **Horn–Schunck method**, which introduces a global **smoothness constraint** on the flow field and formulates optical flow estimation as a **global optimization problem** over the entire

[2] `https://en.wikipedia.org/wiki/Barberpole_illusion`

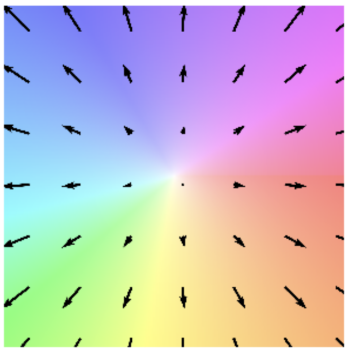

Figure 2: Visualization of optical flow as color-coded flow fields.

image. This approach allows for more consistent flow estimates, especially in regions with little texture or ambiguous motion cues.

Modern approaches leverage **deep neural networks** trained on large datasets of image pairs with ground-truth optical flow to predict flow in a fully **data-driven** manner. Interestingly, many of these datasets are synthetically generated—using *video games*, *rendered 3D scenes*, or simulation environments—where accurate ground-truth optical flow can be easily computed.

3 Visualizing optical flow

Optical flow is typically visualized using color-coded flow fields, where each pixel is assigned a color that represents the direction and magnitude of motion. A common scheme uses the HSV color wheel: the hue encodes the direction of motion (e.g., rightward motion might be red, upward motion blue), while the intensity indicates the speed (with brighter colors representing faster motion) as seen in Figure 2. This provides an intuitive way to understand motion patterns across an image. Flow vectors are also shown as arrows overlaid on the image, though this can become cluttered for dense flow fields.

4 Applications of Optical Flow

Below, we review some key applications of optical flow.

4.1 Depth from Disparity

Given a pair of cameras (a stereo pair) related by pure translation, we can show that the depth of a point in the scene is inversely related to the disparity—i.e., the magnitude of the optical flow between the two views.

In Figure 3, the two cameras, with centers O and O', observe a 3D point X, which is projected to image points x and x' in the respective views. Using similar triangles, we derive the following relationship:

$$\frac{x - x'}{O - O'} = \frac{f}{z}, \tag{9}$$

and therefore,

$$\text{disparity} = x - x' = \frac{B \cdot f}{z}, \tag{10}$$

where $B = O - O'$ is the baseline—the distance between the two camera centers—and f is the focal length. Thus, the depth z of the point is inversely proportional to the disparity, or equivalently, the magnitude of the optical flow between the two images. We can use this to derive a dense depth map given a dense optical flow.

4.2 Tracking Points in Videos

To obtain long-term trajectories of points in a video, one can chain together successive optical flow estimates between pairs of frames. This is the basic idea behind the Kanade-Lucas-Tomasi (KLT) tracker, which consists of the following steps:

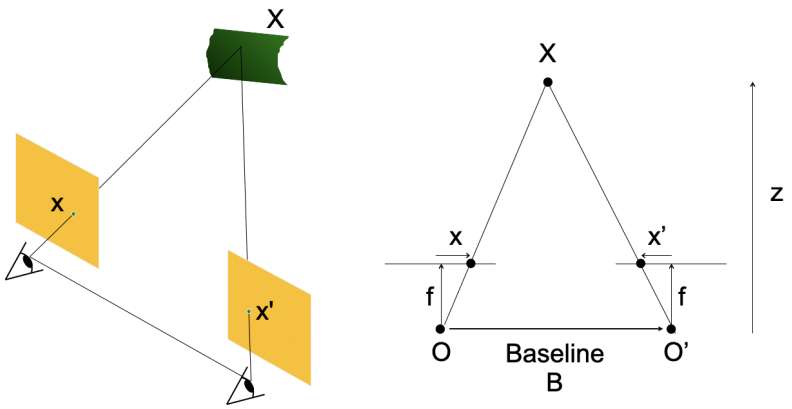

Figure 3: Depth from disparity.

1. Identify corners (or other good features to track) in the image.
2. Estimate optical flow using the Lucas-Kanade method.
3. Compute the displacements of the corners based on the flow.
4. Chain the successive displacements of each corner across frames to form longer trajectories.

4.3 Frame Interpolation Using Optical Flow

Given two video frames, we can estimate one or more intermediate frames that would appear between them. This allows us to either increase the video's frame rate (making it smoother) or play it in slow motion by generating additional frames.

A common approach to estimate these intermediate frames is to assume that the motion between the two original frames is smooth and occurs at a uniform speed. This idea is similar to *keyframe animation* in computer graphics.

Suppose we know the position of an object is at x_0 at time $t = 0$, and it moves to x_1 at time $t = 1$. Then, for any intermediate time $0 < \alpha < 1$, we can estimate the object's position using **linear interpolation**:

$$x_\alpha = (1 - \alpha)x_0 + \alpha x_1$$

Using *optical flow*, which tells us how pixels move from one frame to another, we can apply this idea to all pixels and *warp* them accordingly to generate intermediate frames.

However, this approach can struggle near **object boundaries**, where motion causes *occlusion* (something gets hidden) or *disocclusion* (something is newly revealed). In these regions, simple interpolation may lead to visible artifacts. To improve the results, we need to reason carefully about which input frame—I_0 or I_1—to borrow pixel values from.

Modern methods such as **Super SloMo**[3] use deep neural networks to refine the optical flow driven frame interpolation to better handle difficult cases like motion blur, occlusions, and ambiguous boundaries. These models are trained to produce more accurate and visually pleasing intermediate frames by learning from data, going beyond simple motion assumptions.

[3] https://jianghz.me/projects/superslomo/