XII
UNDERSTANDING MOTION

Images are observed by a moving camera recording a dynamic world, but we have devoted little space to discuss the analysis of image sequences. We will focus on understanding motion in this set of chapters. This part is composed of four chapters:

- **Chapter 46** introduces the problem of motion estimation and provides a very simple approach to estimate motion across two frames of a video.
- **Chapter 47** explains the image formation process and how the three-dimensional (3D) motion in the scene projects into a sequence of two-dimensional (2D) images.
- **Chapter 48** goes deeper into optical flow estimation, describing classical methods for motion estimation.
- **Chapter 49** finally introduces supervised and unsupervised learning-based methods for motion estimation.

Notation

We will continue using the same notation as in the previous chapters:

- Moving points: $\mathbf{P}(t)$ for 3D points, and $\mathbf{p}(t)$ for 2D points. Where t is time.
- Temporal derivatives: to simplify the equations, we will use the dot notation for temporal derivatives, $\dot{(t)} = \partial\mathbf{P}/\partial t$.

46Motion Estimation

46.1Introduction

An important task in both human and computer vision is to model how images (and the underlying scene) change over time. Our visual input is constantly moving, even when the world is static. Motion tells us how objects move in the world, and how we move relative to the scene. It is an important grouping cue that lets us discover new objects. It also tells us about the three-dimensional (3D) structure of the scene.

Look around you and write down how many things are moving and what are they doing. Take note of the things that are moving because you interact with them (such as this book or your computer) and the things that move independently of you.

The first observation you might make is that not much is happening. Nothing really moves. Most of the world is remarkably static, and when something moves it attracts our attention. However, motion perception becomes extremely powerful as soon as the world starts to move. Our visual system can form a detailed representation of moving objects with complex shapes. Even in front of a static image, we form a representation of the dynamics of an object, as shown in the photograph in figure 46.1.

Figure 46.1: Even from a static picture we form a rich representation of the dynamics of the scene. *Source*: Photograph by Fredo Durand.

Looking at the power of that static image to convey motion, one wonders if seeing movies is really necessary. From the notes you took about what moves around you, probably you deduced that the world is, most of the time, static.

And yet, biological systems need motion signals to learn. Hubel and Wiesel [501] observed that a paralyzed kitten was not capable of developing its visual system properly. The human eye is constantly moving with saccades and microsaccades. Even when the world is static, the eye is a moving camera that explores the world.

Motion tells us about the temporal evolution of a 3D scene, and is important for predicting events, perceiving physics, and recognizing actions. Motion allows us to segment objects from the static background, understand events, and predict what will happen next. Motion is also an important grouping cue that our visual system uses to understand what parts of the image are connected. Similarly moving scene points are likely to belong to the same object. For example, the movement of a shadow accompanying an object, or various parts of a scene moving in unison—even when the connecting mechanism is concealed—strongly suggests that they are physically linked and form a single entity.

Motion estimation between two frames in a sequence is closely related to disparity estimation in stereo images. A key difference is that stereo images incorporate additional constraints, as only the camera moves—imagine a stereo pair as a sequence with a moving camera while everything else remains static. The displacements between stereo images respect the epipolar constraint, which allows the estimated motions to be more robust. In contrast, optical flow estimation doesn't assume a static world.

> Disparity from stereo and optical flow estimations are closely related. Stereo benefits from the epipolar constraint to make estimation easier. For a rectified stereo pair the vertical component of motion between the stereo frames is zero.

Another distinction is that optical flow generally presumes small displacements between consecutive frames due to the short time gap between them. In stereo images, feature displacements tend to be larger. Despite these differences, the remaining steps are similar, and the same architectures can address both tasks.

46.2Motion Perception in the Human Visual System

The eye is constantly moving, fixating different scene locations every 300 ms. Therefore, the nature of the input signal to the brain is a sequence of ever-changing visual information. It is not a surprise then that motion perception is a key component of visual perception.

> Only when the eye tracks a moving object can it move continuously. Otherwise, the eye jumps from one location to another in saccades. Try moving your eyes smoothly and you will notice that you cannot. However, if you look at your finger you will see that you can follow its motion smoothly.

What do we know about the human perception of motion? Much is known and many advances

from cognitive psychology have impacted imaging technology. One example is movies. The fact that we learned to create the illusion of continuous motion by displaying a sequence of static images (a phenomenon called **apparent motion**) was a remarkable discovery.

There are a number of visual illusions associated with motion perception that are intriguing and offer a window into how motion perception is implemented in the brain. One famous illusion is the **waterfall illusion**. When looking at a constant motion (such as a waterfall) our brain adapts to the motion in such a way that if, immediately after adaptation, we look at a static texture we will see it drifting in the opposite direction. The waterfall illusion was already known to the Greeks and was reported by Aristotle. Another remarkable and surprising visual illusion is that it is possible to create static images that produce the sensation of motion. One beautiful example is the **Rotating Snakes** visual illusion created by cognitive psychologist Akiyoshi Kitaoka. Figure 46.2 shows an example a motion-inducing visual illusion, using very simple elements to produce the illusion. The effect becomes more intense as we move our eyes to explore different parts of the bicycle.

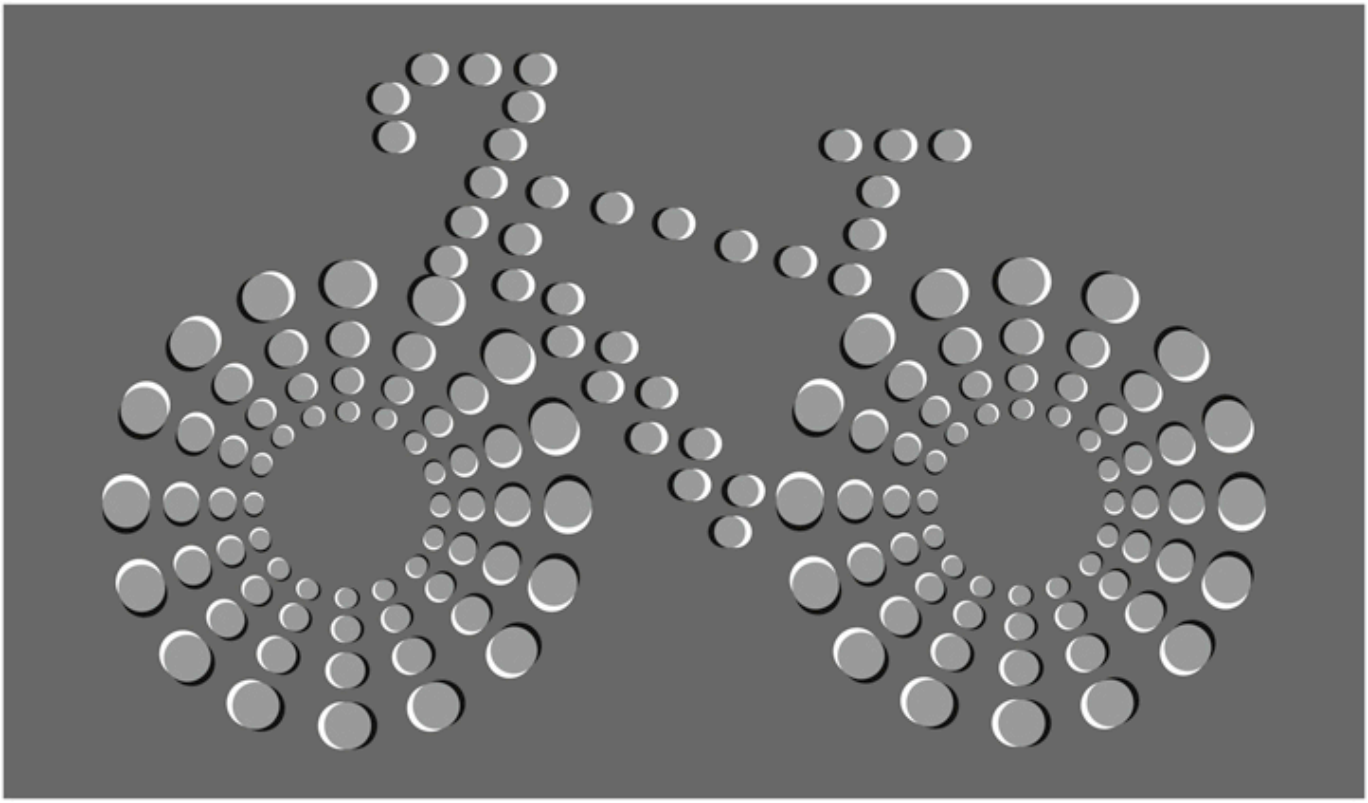


Figure 46.2: Motion-induced visual illusion, after [344]. The illusion becomes stronger when viewed peripherally rather than looking directly at the image. Changing the contrast of this image can change the direction of perceived motion.

The opposite visual illusion can also be achieved: perceiving no motion when there is movement. By creating sequences with isoluminant and textureless patterns [451], an observer can perceive them as perfectly still, even though they are moving. This illusion requires precise calibration and only works for the specific observer the system is tuned for; other observers will still see motion. The illusion can be thought of as an adversarial attack on a single observer's motion estimation system.

What mechanisms does the brain use to translate the sequence of images projected on the retina into actual motion in the 3D scene? This question has long been studied by neuroscientists and psychologists.

In area V1 of the visual cortex, most visual neurons respond to moving stimuli and exhibit selectivity to specific motion directions. A motion-selective neuron responds strongly when an edge moves across its receptive field in a particular orientation, and its response diminishes as the motion deviates from the preferred direction. These motion-selective cells project to other specialized areas, with the middle temporal area (area MT) and the medial superior temporal area (area MST) playing significant roles in motion processing. The exact functions and roles of these areas are not yet fully understood. This organization suggests that motion is processed by specialized visual pathways, indicating a modular architecture for the visual system.

One of the early computational models of motion perception was proposed by Hassenstein and Reichardt [189] when studying the motion detectors in the fly's visual system. Another computational model of human motion perception is the energy model proposed by Adelson and Bergen [9] and briefly discussed in chapter 22. In this chapter we will focus on motion estimation algorithms developed by the computer vision community without trying to follow biologically plausible mechanisms.

46.3Matching-Based Motion Estimation

Let's get our hands dirty quickly by trying to estimate how pixels move in a video. Let's consider two frames of a video sequence that contains a few moving objects, as shown in figure 46.3.

(Frame 1) (Frame 2)

Figure 46.3: Two frames of a video sequence captured from a moving car driving along a busy street in Palma de Mallorca (Spain).

These two frames belong to a sequence captured from a moving car driving along a busy street. In this sequence there are cars moving on both sides of the road, some moving away from the camera and others moving toward it. How can we compute the motion between the two frames? One way of

representing the motion is by computing the displacement for each pixel between the two frames.

Under this formulation, the task of motion estimation consists of finding, for each pixel in frame 1, the location of the corresponding pixel in frame 2. Just as in the chapter on stereo matching, we need first to define how we will compare pixels to find **correspondences**.

Using the color of each individual pixel will be insufficient as many pixels are likely to have very similar colors. Instead we will represent each pixel by a color patch of size $C = 3 \times (2s + 1) \times (2s + 1)$ centered on each pixel. That is, for pixel in location (n, m) in image $\boldsymbol{\ell}$, we will use the patch $\boldsymbol{\ell}[n - s : n + s, m - s : m + s]$ to represent the local appearance in that location. Small values of s will result is small patches that might be less distinctive ($s = 0$ corresponds to use individual pixels), while large values of s will result in large discriminative patches but might fail if there is sufficient geometric distortion between the two frames due to motion. By representing each pixel with a patch, we are transforming the image into a feature map of size $3 \times (2s + 1) \times (2s + 1)$. As in chapter 31, we could also use other patch embeddings such as DINO or SIFT descriptors. But since the image transformation between two consecutive frames is usually small, using red-green-blue (RGB) patches can work well. Another constraint we can use to simplify the matching is to assume that motion will be small between the two frames so that we only need to look for matches inside a small neighborhood of size $L \times L$ in the second frame around the original pixel location. To compute distances between two patches we will use the Euclidian distance between them. We will then compute the motion at each pixel in the first frame by searching for the patch with the smallest distance in the second frame. We can implement this with the following algorithm:

Algorithm 46.1: Patch matching motion estimation. The algorithm starts by chopping the two frames into overlapping patches. Then, for every patch from the first frame, we compute the distance to all the nearby patches in frame 2. Finally, for each input patch we select the closest patch from frame 2 and we record the relative displacement between the two patches. The pseudocode can be rearranged to be more memory efficient.

Input frames: $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2 \ ^{N\times M\times 3}$, **Output motion flow:** $\mathbf{u}, \mathbf{v} \ ^{N\times M}$
Parameters: L = Maximum displacement, s = Patch size parameter
for $i = 0, \ldots, M-1$ **do**
  for $j = 0, \ldots, N-1$ **do**
    $\mathbf{r}_1[i,j] = \boldsymbol{\ell}_1[i-s:i+s, j-s:j+s]$ ◁ Chop up frame 1 into patches
    $\mathbf{r}_2[i,j] = \boldsymbol{\ell}_2[i-s:i+s, j-s:j+s]$ ◁ Chop up frame 2 into patches
for $i = L, \ldots, M-L-1$ **do**
  for $j = L, \ldots, N-L-1$ **do**
    for $di = -L, \ldots, L$ **do**
      for $dj = -L, \ldots, L$ **do**
        $C[i,j][d_i, d_j] = \text{dist}(\mathbf{r}_2[i+di, j+dj], \mathbf{r}_1[i,j])$ ◁ Compute matching cost
for $i = L, \ldots, M-L$ **do**
  for $j = L, \ldots, N-L$ **do**
    $\mathbf{u}[i,j] = \arg\min(\min(C[i,j], axis = 1))$ ◁ Estimate displacement x
    $\mathbf{v}[i,j] = \arg\min(\min(C[i,j], axis = 0))$ ◁ Estimate displacement y

Note that padding must be applied if we want to compute the output optical flow near the image boundaries. Algorithm 46.1 could be written more compactly but we prefer this form for its clarity. The following images shows the matching result for one input location. In this example, $s = 5$ (patch size of 11×11 pixels), and $L = 16$ (search window of size 33×33 pixels).

In the example shown in figure 46.4, the input patch is centered on the car logo. The matching cost displays the distance using a reversed grayscale map, with smaller values appearing as brighter spots. In this case, the search identifies a unique match. However, it is worth noting that the best matching patch, although correctly detecting the same logo, is not identical to the input patch. This discrepancy can be attributed to factors such as nondiscrete motion and the slight enlargement of the logo as the car approaches the camera.

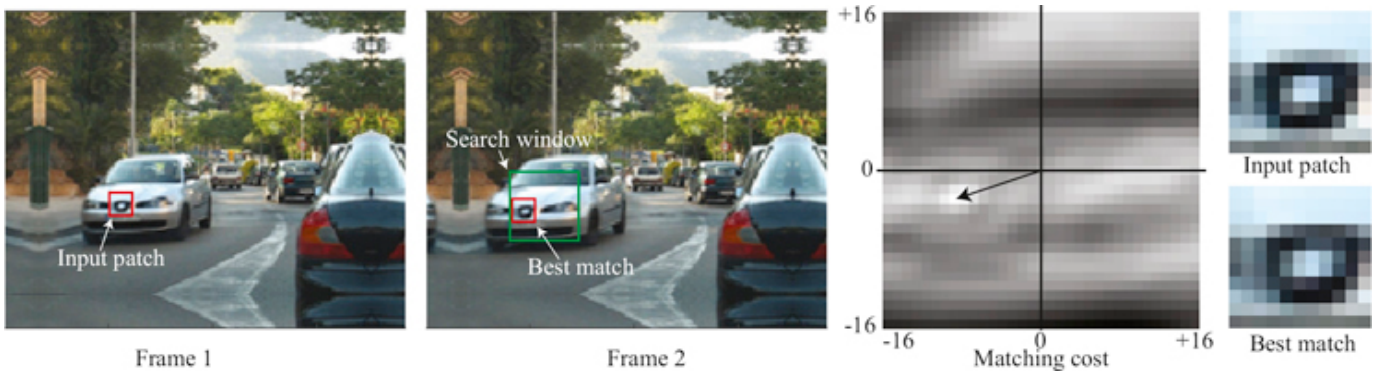


Figure 46.4: Two frames and best match for an input patch from frame 1 within frame 2. Search is done only within a small neighborhood.

The patch size used to represent each pixel is the most important parameter in this algorithm. Figure 46.5 shows the effect of the choice of the parameter *s* on the estimated optical flow.

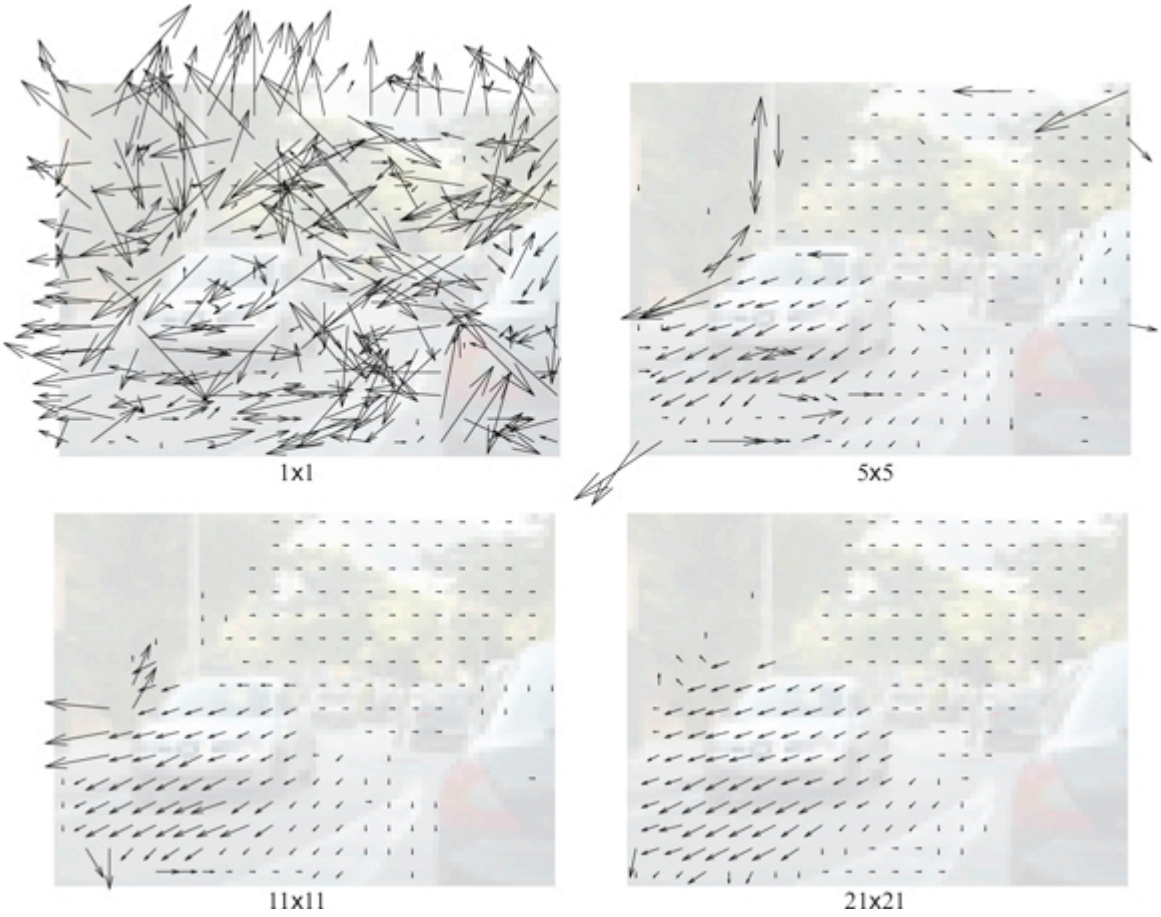


Figure 46.5: Effect of the choice of the patch size parameter, *s*, on the estimated optical flow. When the patch size is just one pixel ($s = 0$), the approach fails as there are many similar pixels that correspond to different parts of the scene. Only when the patches are large enough, the image matches correspond to the same scene elements. Large patch sizes are necessary. But too large patches lead to oversmoothing.

When using single pixels to represent each location, the matching fails in detecting true correspondences and the estimated motion field is very noisy. Just making the patches 5×5 pixels is already capable of detecting many correct matches and the motion field seems to mostly capture the true motion between the two frames. Further increasing the patch size eliminates some of the errors. However, using very big patches also introduces new problems. In this example we can see how the motion of the gray car extends over the road. This is due to patches overlapping with the car on top, and as the road is mostly uniform, the motion of the car propagates to all the nearby pixels. One of the challenges in this algorithm is that the ideal value of s will depend on the sequence.

This approach has many shortcomings. To start, we assumed that motion is discrete (it can only take on integer values). Therefore, the current approach does not compute displacements of pixels to subpixel accuracy that might be important if we need precision or if motions are very small. We could improve the approach by interpolating the cost function or by computing subpixel displacements using bilinear or bicubic interpolation, but it would become more computationally expensive. In addition, the patch matching method gives poor results near motion discontinuities such as object edges.

One advantage of this algorithm is that motion is computed in a way that is independent of the objects present in the scene. We did not make any assumption about what objects are moving or how. We did not introduce any grouping cues (as in chapter 31) to presegment the image into candidate objects. Therefore, we could use the computed motion as new cue for grouping.

46.4Does the Human Visual System Use Matching to Estimate Motion?

While researchers aren't sure of the precise computations involved in human motion processing, some experiments can distinguish between classes of algorithms that the visual system may use. The previous method is an example of **pattern matching methods** [9], which are often based on image correlations. A second class of motion algorithms are based on **spatiotemporal filtering** [9] and their principles were briefly described in chapter 22. Spatiotemporal filtering uses the responses of velocity-tuned filters to estimate the motion.

Adelson and Bergen proposed a beautiful motion illusion that distinguishes between two classes of motion algorithms that might be used by the visual system (figure 46.6). The illusion involves temporal filtering, motion processing, and aliasing and thus provides a good review of the material in this chapter and also chapters 20 and 19.

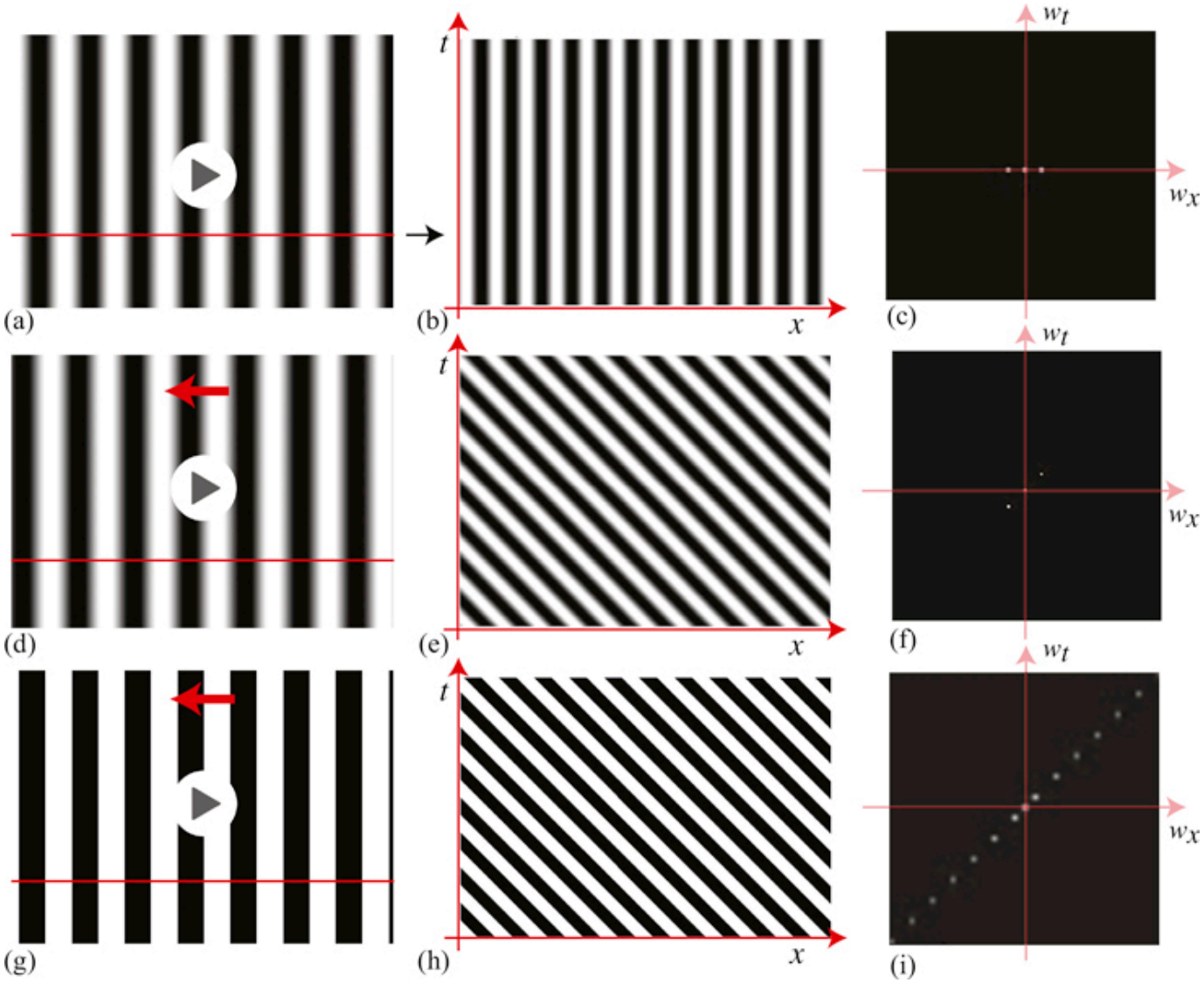


Figure 46.6: Space-time signals, building toward the fluted square-wave motion illusion. The first row shows a stationary sine wave. (a) Movie of a motionless sine wave. (b) Space-time plot shows only vertical structure. (c) Spatiotemporal Fourier transform shows all energy on the zero temporal frequency axis because nothing is moving. (d) The second row shows a moving sine wave. (e) In the space-time plot, speed corresponds to local orientation. (f) The Fourier transform energy is sheared according to the sine wave's speed. (g–i) The third row shows a moving square wave. The additional harmonics required to form a square wave are visible in (i) the spatiotemporal Fourier transform.

The signals, and magnitudes of their space-time Fourier transforms, are developed in figures 46.6 and 46.7, building-up from simpler signals. The three rows of figure 46.6 show a stationary sinusoid, a moving sinusoid, and a moving square wave. The spatiotemporal Fourier transform of the stationary sinusoid, $f(x, y, t) = \cos(\pi\omega x)$, is $(\delta(w_x + \omega) + \delta(w_x - \omega))\delta(w_y)\delta(w_t)$. We have added a constant bias to the sinusoid to avoid negative intensity values, leading to an impulse at the center of the Fourier transform, as shown in figure 46.6(c). The resulting three colinear impulses are along the temporal frequency, $w_t = 0$ line. A space-time plot of the signal (figure 46.6[b]) shows only vertical structures, indicating no motion.

A moving sinusoid has a similar Fourier transform magnitude (figure 46.6[f]) but with the spatiotemporal energies along a line perpendicular to the moving structures in the spatiotemporal signal (figure 46.6[e]). A moving square wave is similar, but the extra harmonics needed to construct the square wave visible in the Fourier transform (figure 46.6[i]).

Continuing the development of the illusion, figure 46.7(c) shows the Fourier transform and space-time plot of a square wave moving in 1/4 period jumps each time increment. This signal can be formed from figure 46.6(h) by applying a periodic sample-and-hold function, resulting in the spectrum of figure 46.6(i) replicated over temporal frequencies, and multiplied by a sinc function over temporal frequency. The resulting Fourier transform magnitude is shown in figure 46.7(c).

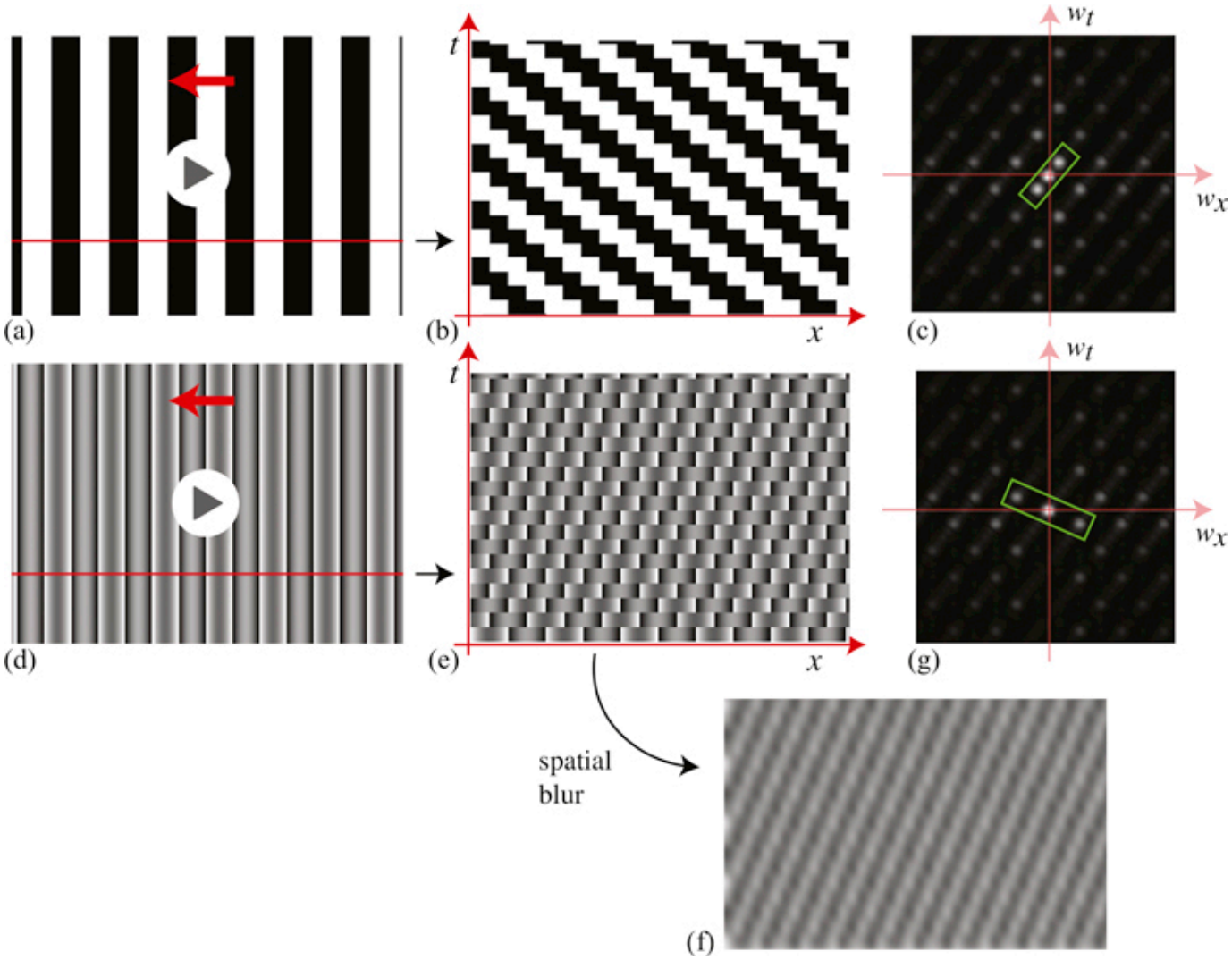


Figure 46.7: Derivation of the fluted square-wave motion illusion, continued from figure 46.6. Top row shows that the square wave moves in 1/4 wavelength jumps, instead of continuously. This staggered motion generates the additional spatiotemporal frequencies shown in (c). The lowest spatiotemporal frequency (green rectangle) still indicates motion to the left. Second row shows that if we remove the lowest spatial frequency sine wave of the square wave, creating a *fluted square wave*, then the lowest spatio-emporal frequency now moves in the other direction. This is also visible from (e) the space time plot and especially in (f) the spatiotemporally low-pass filtered version.

Because orientation in space-time tells motion direction (section 19.2) the space-time plot of figure 46.7(b) shows that the motion should be perceived to the left. This will be consistent with the behavior of velocity tuned filters (section 22.4.2) responding to the lowest spatiotemporal frequency impulses shown in figure 46.7(c). However, if we remove the lowest spatial frequency sinusoid from the signal, the result is shown in figure 46.7(e), with spatiotemporal Fourier transform shown in figure 46.7(g). Now the lowest spatiotemporal frequency cosine wave is oriented in the other direction. This opposite slope is also visible in the spatial domain, in the space-time plot of figure 46.7(e), and especially if we apply a low-pass filter, resulting in figure 46.7(f).

The signal of the second row of figure 46.7 poses a conundrum. It can be argued that the signal moves to the left, just as does the signal of row 1 of figure 46.7. The pattern match, that is, the minimum correlation signal indeed moves to the left. But the vision system examining the orientation of the lowest spatiotemporal frequency components of the signal in figure 46.7(g), or looking at the dominant orientations in the space-time plots of figures 46.7(e and g), would find a signal moving to the right! Videos showing each signal are available on the book's web page. These illusions give support to the spatiotemporal energy models for human motion processing.

46.5Concluding Remarks

In this section we have introduced a conceptually simple approach to compute the motion in sequences. But are the estimated patch displacements meaningful? Do they correspond in any way to the motion in the 3D world?

We have computed motion between two frames before really understanding the motion formation process (i.e., how a camera looking at a moving 3D scene produces two-dimensional [2D] sequences). How should the correct motion look? And what do we want to do with it? So, before we move into more sophisticated motion estimation algorithms, let's revisit the image formation process and examine how motion on the image plane emerges from the perspective projection of a dynamic 3D scene into a moving camera.

473D Motion and Its 2D Projection

47.1Introduction

As objects move in the world, or as the camera moves, the projection of the dynamic scene into the two-dimensional (2D) camera plane produces a sequence of temporally varying pixel brightness. Before diving into how to estimate motion from pixels, it is useful to understand the image formation process. Studying how three-dimensional (3D) motion projects into the camera will allow us to understand what the difference is between a moving camera or a moving object and what types of constraints one might be able to use to estimate motion.

47.23D Motion and Its 2D Projection

A 3D point will follow a trajectory $\mathbf{P}(t) = (X(t), Y(t), Z(t))$, in camera coordinates (figure 47.1). As the point moves, it has an instantaneous 3D velocity of $= (\ (t),\ \ (t), \dot{Z}(t))$. The projection of this point into the image plane location is $\mathbf{p}(t) = (x(t), y(t))$ and its projection will move with the 2D instantaneous velocity $= ((t), (t))$, where all the derivatives are done with respect to time, t.

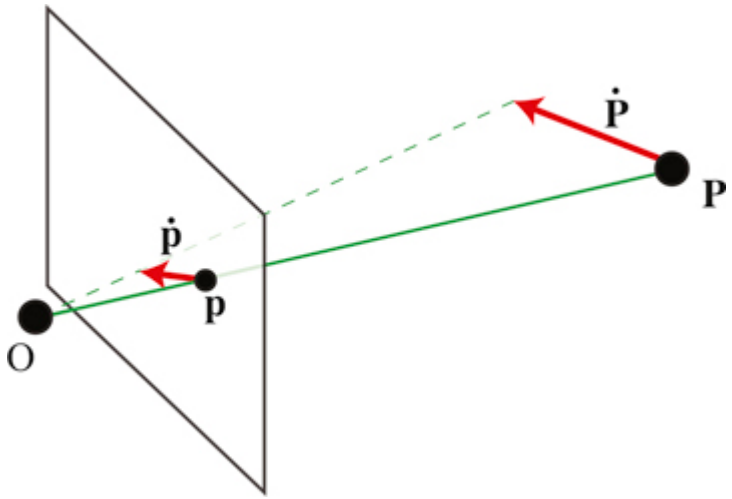


Figure 47.1: A 3D point, **P**, moving in the world projects a 2D moving point, **p**, into the camera plane.

Using the equations of perspective projection $x = fX/Z$ and $y = fY/Z$ (assuming that the camera is at the origin of the world-coordinate system), we can derive the equations of how the instantaneous velocity in the camera plane relates to the point motion in 3D world coordinates:

$$\dot{x} = f\frac{\dot{X}Z - \dot{Z}X}{Z^2} = \frac{f\dot{X} - \dot{Z}x}{Z} \tag{47.1}$$

$$\dot{y} = f\frac{\dot{Y}Z - \dot{Z}Y}{Z^2} = \frac{f\dot{Y} - \dot{Z}y}{Z} \tag{47.2}$$

The second expression is obtained by using $x = fX/Z$ and $y = fY/Z$, which removes the dependency on the world coordinates X and Y. Note that f is the focal length. In many derivations, the notation is simplified by setting the focal length $f = 1$. Here we will keep it to make explicit which factors depend on the camera parameters and which ones do not. We can write the last two equations in matrix form as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix}\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} \tag{47.3}$$

This expression reveals a number of interesting properties of the optical flow and how it relates to motion in the world. For instance, points that move parallel to the camera plane ($\dot{Z} = 0$) will project to a motion parallel to the motion in 3D but with a magnitude that will be inversely proportional to the distance Z:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} \tag{47.4}$$

For points moving parallel to the Z axis (= = 0) we get:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\frac{\dot{Z}}{Z}\begin{bmatrix} x \\ y \end{bmatrix} \tag{47.5}$$

Points at the same distance Z from the camera, moving away or towards the camera ($\dot{Z} \neq 0$, and = = 0), with the same velocity will project into points moving at different velocities on the image plane.

Figure 47.2 illustrates the geometry of the projection of 3D motion into the camera for points moving parallel to the camera plane (figure 47.2[left]) and parallel to the optical axis of the camera (figure 47.2[right]). The arrows at the image plane show the imaged scene velocities.

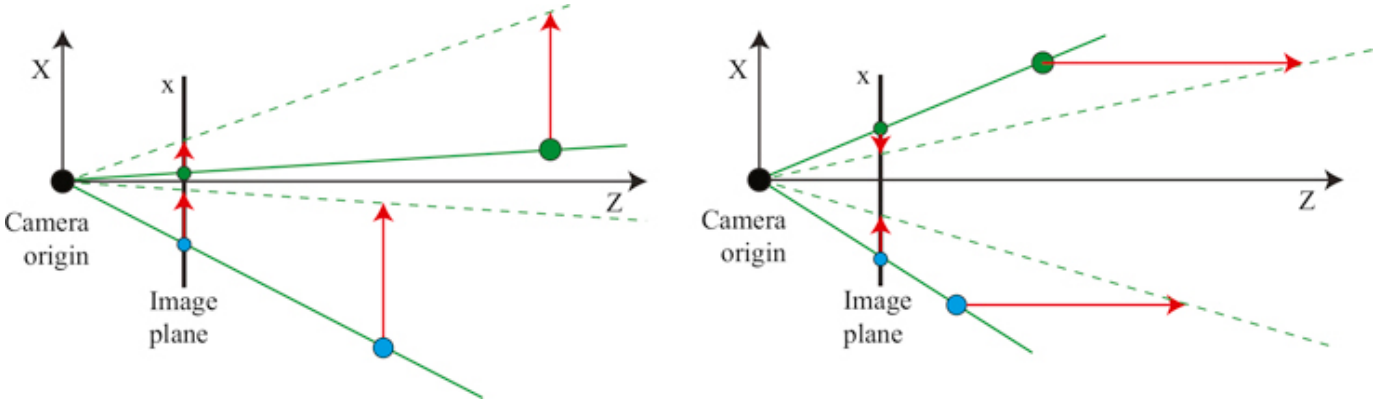


Figure 47.2:(left) Geometry of the projection of 3D motion into the camera for points moving parallel to the camera plane, and (right) parallel to the optical axis of the camera.

Let's examine a few scenarios in a bit more detail to gain some familiarity with the relationship between 3D motion and the projected 2D motion field.

47.2.1Vanishing Point

Let's consider a point moving in a straight line in 3D with constant velocity over time: $= (V_X, V_Y, V_Z)^T$. At each time instant, t, the point location will be $\mathbf{P}(t) = (X + V_X t, Y + V_Y t, Z + V_Z t)^T$, and its 2D projection:

$$x(t) = f\frac{X + V_X t}{Z + V_Z t} \tag{47.6}$$

$$y(t) = f\frac{Y + V_Y t}{Z + V_Z t} \tag{47.7}$$

If $\dot{Z} = 0$, then the projected point will move with constant velocity over time, as shown in equation (47.4). If $\dot{Z} \neq 0$, then, as time goes to infinity, the point will converge to a **vanishing point**:

$$\lim_{t\to\infty} x(t) = f\frac{V_X}{V_Z} = x_\infty \tag{47.8}$$

$$\lim_{t\to\infty} y(t) = f\frac{V_Y}{V_Z} = y_\infty \tag{47.9}$$

The following sketch (figure 47.3) shows Gibson's bird (see figure 1.9) flying away from the camera along a straight line. In the camera the bird gets smaller as it flies away until it disappears at the vanishing point. In this drawing the vanishing point is within the view of the camera.

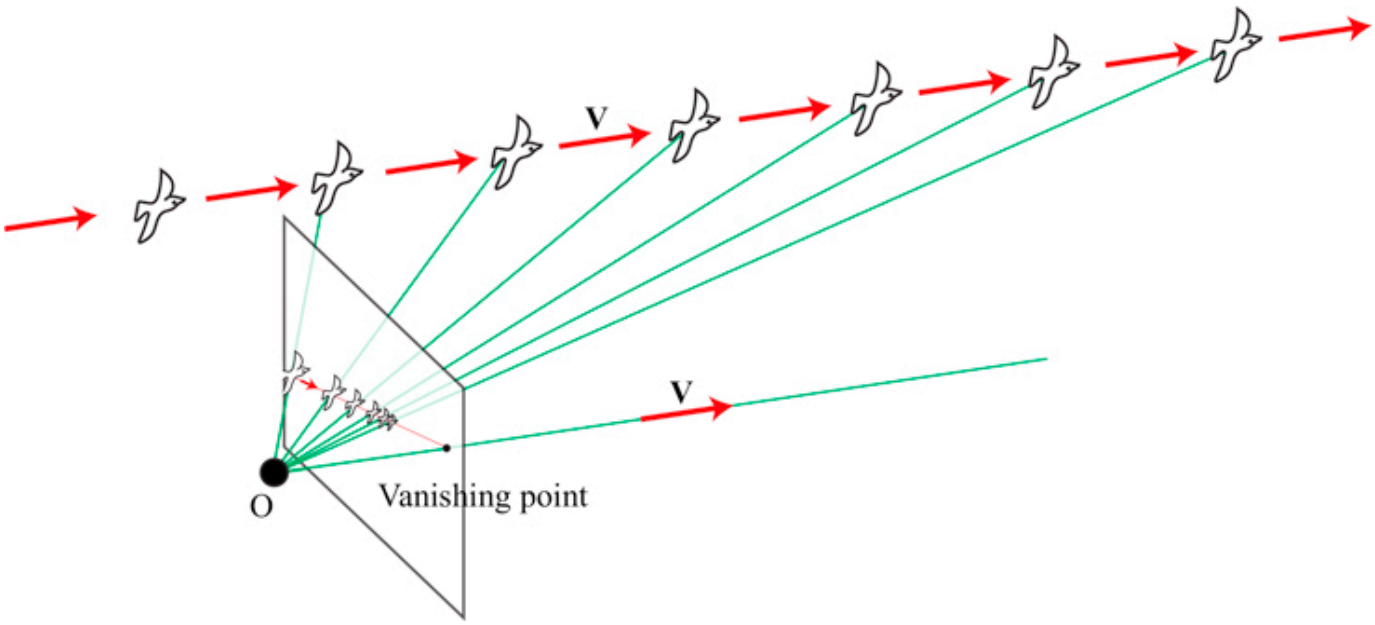


Figure 47.3: Projection onto the camera plane of the sequence produced by a bird flying away. The bird will vanish at the **vanishing point**.

The vanishing point is the location $\mathbf{p}_\infty = (x_\infty, y_\infty)^\mathsf{T}$ where the moving point slowly converges to. The location of the vanishing point is independent of the point location at time $t = 0$, and it only depends on the 3D velocity vector, $\mathbf{V}$. Therefore, if the scene contains multiple points at different locations moving with the same velocity, they will converge to the same vanishing point.

47.2.2Camera Translation

Let's now assume that the scene is static and that only the camera is moving. In this case, all the observed motion in the image will be due to the motion of the camera.

Let's assume the camera is moving in a straight line with a velocity $= \mathbf{V} = (V_X, V_Y, V_Z)^\mathsf{T}$. The translation of the camera after time t will be $\mathbf{T} = \mathbf{V}t$. A point in space $\mathbf{P} = (X, Y, Z)^\mathsf{T}$, will move with velocity, relative to the camera, equal to $= -\mathbf{V}$. The moving camera is equivalent to the case where all the scene points move relative to the camera with the same velocity.

The 2D motion field by using equation (47.3) is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} \tag{47.10}$$

We can also express the same relationship by making the contribution of the camera coordinates more explicit. We can do this by rearranging the terms in equation (47.10), resulting in:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\frac{f}{Z} \begin{bmatrix} V_X \\ V_Y \end{bmatrix} + \frac{V_Z}{Z} \begin{bmatrix} x \\ y \end{bmatrix} \tag{47.11}$$

Equation (47.10) gives a generic expression for the observed motion in the camera plane produced by a moving camera undergoing a translation (we will see later what happens if we also have camera rotation). But let's first look at a few specific scenarios with a camera following simple translation trajectories (and no rotations).

47.2.2.1Lateral camera motion

Consider a camera translating laterally, as shown in figure 47.4. This will happen if you are looking through a side window of a car at the scene passing by. In this case, the forward velocity is zero, $V_Z = 0$.

Figure 47.4: Lateral camera motion parallel to the camera plane.

Under lateral camera motion, using equation (47.4), we have the following relationship between

the velocity of a 3D point and the apparent velocity of its projection in the image plane:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\frac{f}{Z} \begin{bmatrix} V_X \\ V_Y \end{bmatrix} \tag{47.12}$$

The motion field depends on the depth at each location Z as illustrated in figure 47.5. Objects close to the camera will appear moving faster than objects farther away. Objects that are very far will appear as not moving. This parallax effect is the same one used in stereo vision to recover depth. The 2D motion in the image place is in the opposite direction to the camera motion.

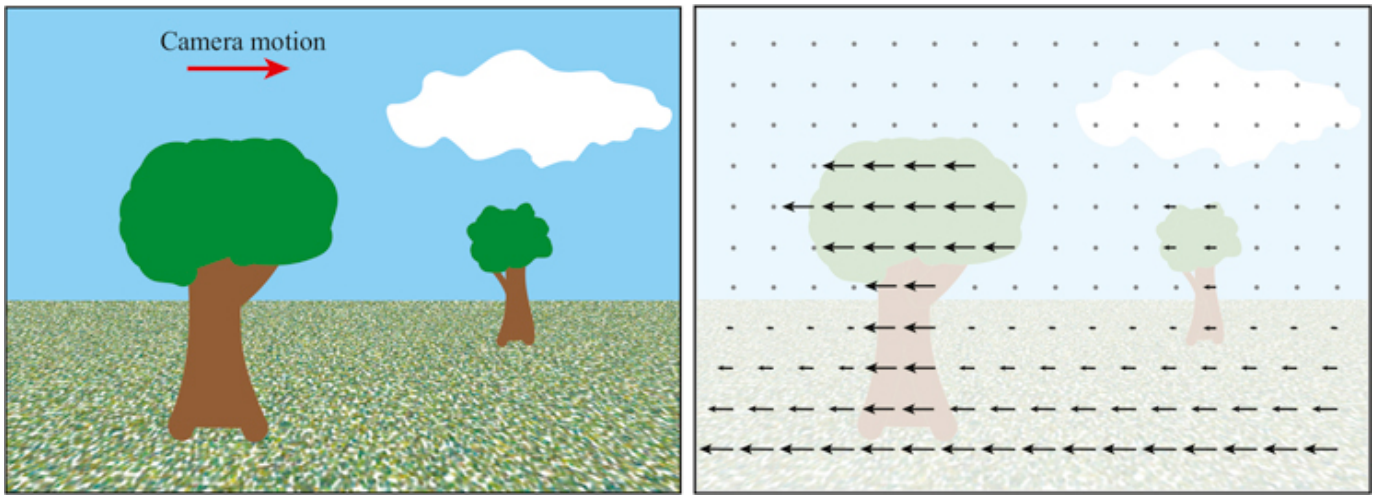


Figure 47.5: Sketch of the motion field under lateral camera motion. Objects close to the camera will appear to be moving faster than objects farther away. Objects that are very far (like the cloud) will appear to be nearly stationary.

47.2.2.2Camera forward motion and focus of expansion

For a camera moving forward, as illustrated in figure 47.6, note that $V_X = V_Y = 0$. In this case, the motion is only along the Z-axis, $V_Z \neq 0$.

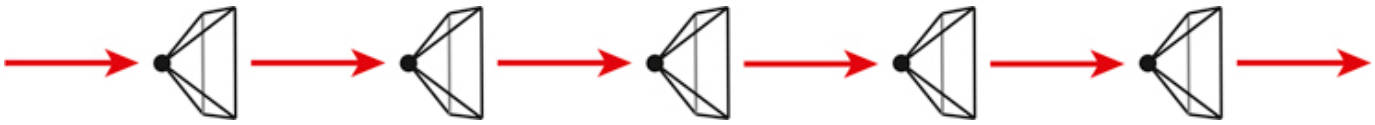


Figure 47.6: Camera moving forward, along the camera axis.

Using equation (47.5), we get:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{V_Z}{Z} \begin{bmatrix} x \\ y \end{bmatrix} \tag{47.13}$$

Equation (47.13) provides a few interesting insights. First, the rate of expansion does not depend on the focal length f. Second, the observed motion only depends on the ratio V_Z/Z, which is the

inverse of the **time to contact**. The time to contact, V_Z/Z, is the time it will take the camera to reach the object located a distance Z when moving at velocity V_Z.

For a camera moving in an arbitrary direction, that is, with $V_X \neq 0$, $V_Y \neq 0$, and $V_Z \neq 0$, using equation (47.3) and equation (47.9) we get:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{V_Z}{Z} \begin{bmatrix} x - x_\infty \\ y - y_\infty \end{bmatrix} \tag{47.14}$$

The observed motion is zero at the **focus of expansion**, (x_∞, y_∞).

Figure 47.7 illustrates the apparent motion field for a camera moving toward the center of a wall. Points near the center (which will be the point of impact) appear stationary, while points in the periphery appear to move faster and away from the center. The whole wall expands over time.

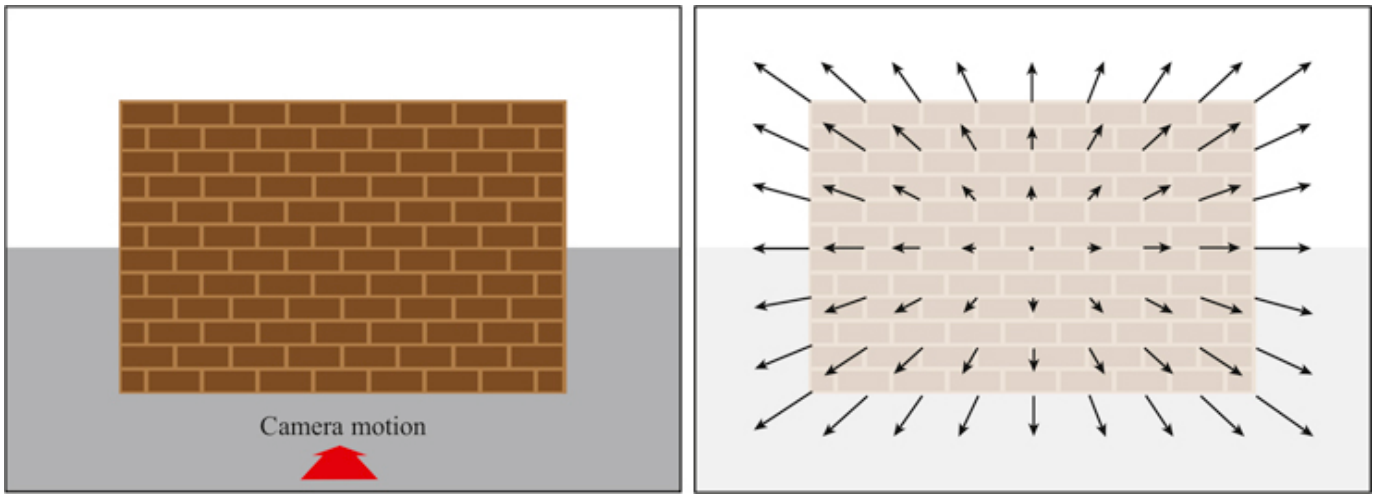


Figure 47.7: Sketch of the motion field when the camera approaches a planar surface. The motion field indicates the rate of expansion of the image, and it is a function of the time to contact. In this example, the focus of expansion is on the center.

47.2.3Camera Rotation

Let's consider a general camera motion undergoing both translation and rotation. To compute the motion field with a compact expression, we will do a number of simplifications assuming a small motion between consecutive frames. After a small time interval, Δt, the camera will move, generating a displacement in the points with respect to the camera coordinate system equal to:

$$\mathbf{P}_{t+\Delta t} = -\mathbf{T}_{\Delta t} + \mathbf{R}_{\Delta t}\mathbf{P}_t \tag{47.15}$$

where $\mathbf{T}_{\Delta t}$ is the camera translation and $\mathbf{R}_{\Delta t}$ is the camera rotation that took place over that time interval, Δt. The velocity of a 3D point with respect to the camera will be:

$$\dot{\mathbf{P}} = \frac{\mathbf{P}_{t+\Delta t} - \mathbf{P}_t}{\Delta t} = -\mathbf{V} + \frac{\mathbf{R}_{\Delta t} - \mathbf{I}}{\Delta t}\mathbf{P}_t \tag{47.16}$$

To derive the rotation, we consider the **Euler angles** (figure 47.8) and decompose the rotation using rotations along the three axes (yaw, pitch, roll):

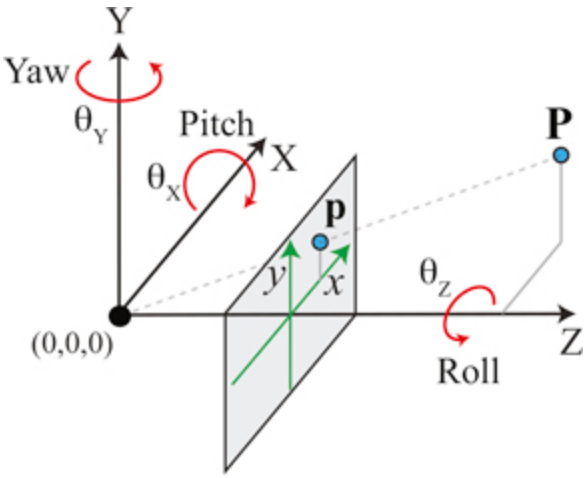


Figure 47.8: Rotation expressed by Euler angles (yaw, pitch, roll).

Each angle measures the rotation along the camera-coordinate axes. Using this representation of the rotation, the rotation matrix can be written as:

$$\mathbf{R}_{\Delta t} = \begin{bmatrix} \cos\theta_Z & \sin\theta_Z & 0 \\ -\sin\theta_Z & \cos\theta_Z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_Y & 0 & -\sin\theta_Y \\ 0 & 1 & 0 \\ \sin\theta_Y & 0 & \cos\theta_Y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_X & \sin\theta_X \\ 0 & -\sin\theta_X & \cos\theta_X \end{bmatrix} \tag{47.17}$$

In this equation, the sign of the angles are chosen to reflect that a rotation of the camera is equivalent to the opposite rotation of the 3D point.

For a small Δt, the angles will be small, and we can approximate the trigonometric functions, cos and sin, by $\cos\alpha \approx 1$ and $\sin\alpha \approx \alpha$. We can also approximate the product $\sin\alpha \sin\beta \approx 0$ as it will result in a second-order term.

$$\mathbf{R}_{\Delta t} \approx \begin{bmatrix} 1 & \theta_Z & 0 \\ -\theta_Z & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -\theta_Y \\ 0 & 1 & 0 \\ \theta_Y & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \theta_X \\ 0 & -\theta_X & 1 \end{bmatrix} \approx \begin{bmatrix} 1 & \theta_Z & -\theta_Y \\ -\theta_Z & 1 & \theta_X \\ \theta_Y & -\theta_X & 1 \end{bmatrix} \tag{47.18}$$

$$\mathbf{P}_{t+\Delta t} - \mathbf{P}_t = -\mathbf{T}_{\Delta t} + (\mathbf{R}_{\Delta t} - \mathbf{I})\mathbf{P}_t = -\mathbf{T}_{\Delta t} - \begin{bmatrix} 0 & -\theta_Z & \theta_Y \\ \theta_Z & 0 & -\theta_X \\ -\theta_Y & \theta_X & 0 \end{bmatrix} \mathbf{P}_t \tag{47.19}$$

The last term corresponds to the cross product in matrix form (note that we changed the sign of the matrix to make the cross product form more obvious). Therefore, we can rewrite the previous expression as:

$$\mathbf{P}_{t+\Delta t} - \mathbf{P}_t = -\mathbf{T}_{\Delta t} - \boldsymbol{\theta} \times \mathbf{P}_t \tag{47.20}$$

where $\boldsymbol{\theta} = (\theta_X, \theta_Y, \theta_Z)$. Substituting this expression into equation (47.16), we get the expression of the motion of a 3D point:

$$\dot{\mathbf{P}} = -\mathbf{V} - \mathbf{W} \times \mathbf{P}_t = - \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} - \begin{bmatrix} -W_Z Y + W_Y Z \\ W_Z X - W_X Z \\ -W_Y X + W_X Y \end{bmatrix} \tag{47.21}$$

where $\mathbf{W}$ is the angular velocity $\mathbf{W} = (W_X, W_Y, W_Z)$. Now we are ready to compute the 2D motion field by using equation (47.3):

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\frac{1}{Z} \begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} - \frac{1}{Z} \begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix} \begin{bmatrix} -W_Z Y + W_Y Z \\ W_Z X - W_X Z \\ -W_Y X + W_X Y \end{bmatrix} \tag{47.22}$$

This expression can be rewritten as, using $x = fX/Z$ and $y = fY/Z$:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -f & 0 & x \\ 0 & -f & y \end{bmatrix} \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} + \frac{1}{f} \begin{bmatrix} xy & -f^2 - x^2) & fy \\ f^2 + y^2 & -xy & -fx \end{bmatrix} \begin{bmatrix} W_X \\ W_Y \\ W_Z \end{bmatrix} \tag{47.23}$$

This is the expression we were looking for. It relates the 2D motion field with the camera velocity and rotation. The matrices are only a function of the intrinsic camera parameters (focal length, f) and the camera coordinates. Note that this expression is only valid for small displacements.

In the previous section we saw what happens when there is no rotation; now we can focus on the case when there is only camera rotation, that is $V_X = V_Y = V_Z = 0$,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{f} \begin{bmatrix} xy & -f^2 - x^2) & fy \\ f^2 + y^2 & -xy & -fx \end{bmatrix} \begin{bmatrix} W_X \\ W_Y \\ W_Z \end{bmatrix} \tag{47.24}$$

The first thing to notice is that the 2D motion field does not depend on the 3D scene structure, Z, and it is only a function of the rotational velocity and the camera parameters. Therefore, under camera rotation we can not learn anything about the scene by observing the motion field. The only thing we can learn from the 2D motion field is about the motion of the camera.

Let's consider first rotation along the camera optical axis, that is $W_X = W_Y = 0$. In this case the motion field is:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} y \\ -x \end{bmatrix} W_Z \tag{47.25}$$

The 2D motion field at each location (x, y) will point in the orthogonal direction to the vector that connects that point with the origin (figure 47.9). The motion field does not depend on the focal length, f.

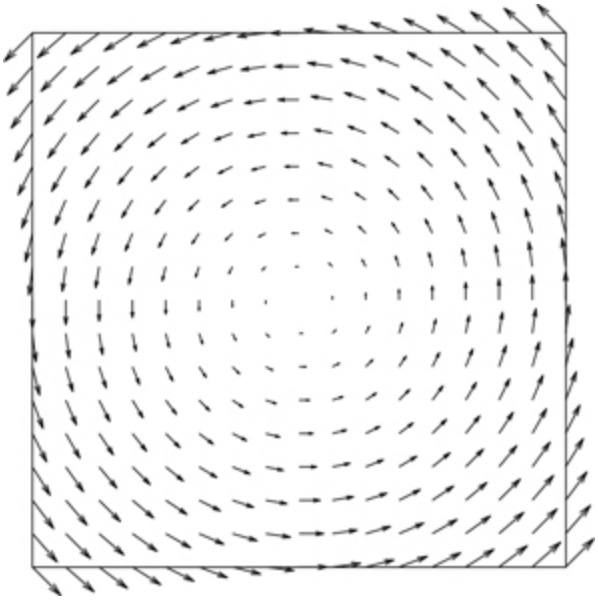


Figure 47.9: Motion field for a rotating camera around the optical axis, $W_X = W_Y = 0$.

If $W_Z = 0$, then the rotation along W_X or W_Y will produce similar 2D motion fields, so let's consider $W_X = 0$. We have:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = -\frac{1}{f} \begin{bmatrix} f^2 + x^2 \\ xy \end{bmatrix} W_Y \tag{47.26}$$

In this case, the focal length f will have a strong effect on the appearance of the motion field. For very large f, we can approximate the 2D motion field by $\approx fW_Y$ and ≈ 0. The resulting motion field is approximately constant across the entire image and looks like lateral translation motion. For very small f, the motion field will be similar to the one produced by a homography and it will be very different to a lateral camera motion. The flows shown in figure 47.10 correspond to $f = 1/3$, $f = 1$, and $f = 3$.

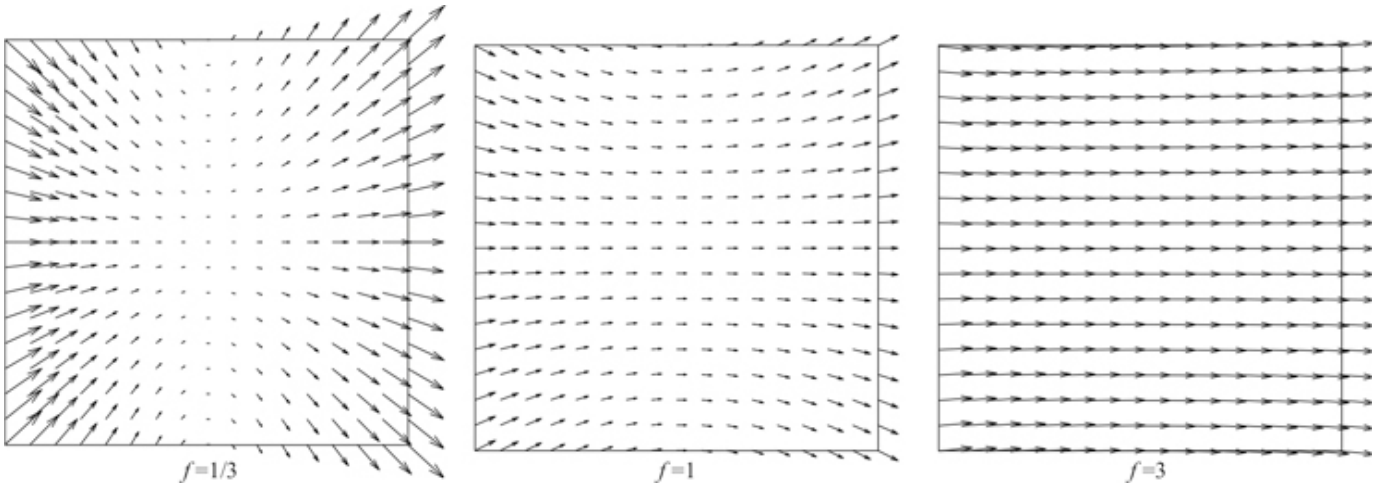


Figure 47.10: Motion flows corresponding to a rotation around the Y-axis. (left) $f = 1/3$. (middle) $f = 1$. (right) $f = 3$.

Camera rotation around the Y-axis does not inform about the scene structure, but it informs about the camera parameters. The magnitude of W_Y only affects the scaling of the motion vectors, but it does not change their orientation.

In the case of camera rotation, for a large angle (or a large Δt), the relationship between the image at time t and the image at time $t + \Delta t$ is a homography.

47.2.4 Motion under Varying Focal Length

Before moving into motion estimation, let's consider one final scenario: a static camera observing a static scene, but the focal length changes over time. What will the motion field be? In this setting, despite that there is not motion in the scene, the focal length of the camera changes over time, producing motion in the image plane. If the focal length increases, it will seem as if we are zooming into the scene. Would it be similar to a forward motion?

Starting from the perspective projection equation,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{47.27}$$

If we compute the temporal derivative where only f varies over time, we get:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{\dot{f}}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} = \frac{\dot{f}}{f} \begin{bmatrix} x \\ y \end{bmatrix} \tag{47.28}$$

The last expression is obtained by using equation (47.27). As the equation shows, changing the focal length only results in a scaling of the projected image on the image plane. It does not create any parallax. As the sensor has finite size, changing the focal length results in a zoom and a crop. The motion field does not depend on the 3D scene structure. Therefore, images taken by a pinhole camera

from the same viewpoint but with different focal lengths do not provide depth information about the scene. This is an example where there is a 2D motion field even when there is no motion in the scene.

47.3Concluding Remarks

As we did in chapter 5, in this chapter we have focused on formulating the problem of image formation: How does the 3D motion in the world appear once it is projected on the image plane?

But the goal of vision is to inverse this projection and recover the 3D scene structure. In the upcoming chapters, we will proceed on the path begun in chapter 46, and we will study how to estimate motion from pixels.

48Optical Flow Estimation

48.1Introduction

Now that we have seen how a moving three-dimensional (3D) scene (or camera) produces a two-dimensional (2D) motion field on the image, let's see how can we measure the resulting 2D motion field using the recorded images by the camera. We want to measure the 2D displacement of every pixel in a sequence.

Unfortunately, we do not have a direct observation of the 2D motion field either, and not all the displacements in image intensities correspond to 3D motion. In some cases, we can have scene motion without producing changes in the image, such as when the camera moves in front of a completely white wall; in other cases, we will see motion in the image even when there is not motion in the scene such as when the illumination source moves.

In the previous chapter we discussed a matching-based algorithm for motion estimation but it is slow and assumes that the motion is only on discrete pixel locations. In this chapter we will discuss gradient-based approaches that allow for estimating continuous displacement values. These methods introduce many of the concepts later used by learning-based approaches that employ deep learning.

48.22D Motion Field and Optical Flow

Before we discuss how to estimate motion, let's introduce a new concept: optical flow.

> James J. Gibson, presented in chapter 1, introduced the concept of optical flow in 1947 [159].

Optical flow is an approximation to the 2D motion field computed by measuring displacement of image brightness (figure 48.1). The ideal optical flow is defined as follows: given two images $\boldsymbol{\ell}_1$ and $\boldsymbol{\ell}_2$ $^{N\times M\times 3}$, the optical flow $[\mathbf{u}, \mathbf{v}]$ $^{N\times M\times 2}$ indicates the relative position of each pixel in $\boldsymbol{\ell}_1$ and the corresponding pixel in $\boldsymbol{\ell}_2$. Note that optical flow will change if we reverse time. This definition assumes that there is a one-to-one mapping between two frames. This will not be true if an object appears in one frame, or when it disappears behind occlusions. The definition also assumes that one motion explains all pixel brightness changes. That assumption can be violated for many reasons, including, for example, if transparent objects move in different directions, or if an illumination source moves.

Figure 48.1 shows two frames and the optical flow between them. This visualization using a color code was introduced in [29]. In this chapter we will use arrows instead as it provides a more direct visualization and it is sufficient for the examples we will work with.

Figure 48.1: Two frames of a sequence, ground-truth optical flow (color coded), and the color code to read the vector at each pixel.

48.2.1 When Optical Flow and Motion Flow Are Not the Same

There are a number of scenarios where motion in the image brightness does not correspond to the motion of the 3D points in the scene. Here there are some examples where it is unclear how motion should be defined:

- Rotating Lambertian sphere with a static illumination will produce no changes in the image. If the sphere is static, and the light source moves, we will see motion in spite of the sphere being static.
- Moving in front of a textureless wall produces no change on the image.
- Waves in water: waves appear to move along the surface but the actual motion of the water is up and down (well, it is even more complicated than that).
- A rotating mirror will produce the appearance of a faster motion. And this will happen in general with any surface that has some specular component.
- A camera moving in front of a specular planar surface will not produce a motion field corresponding to a homography.

Motion estimation should not just measure pixel motion, it should also try to assign to each source of variation in the image the physical cause for that change. The models we will study in this chapter will not attempt to do this.

48.2.2 The Aperture Problem

One classical example of the limitations of motion estimation from images is the **aperture problem**. The aperture problem happens when observing the motion of a one-dimensional (1D) structure larger than the image frame, as shown in figure 48.2. If none of the termination points are in view within the observation window (i.e., the aperture), it is impossible to measure the actual image motion. Only the component orthogonal to the moving structure can be measured.

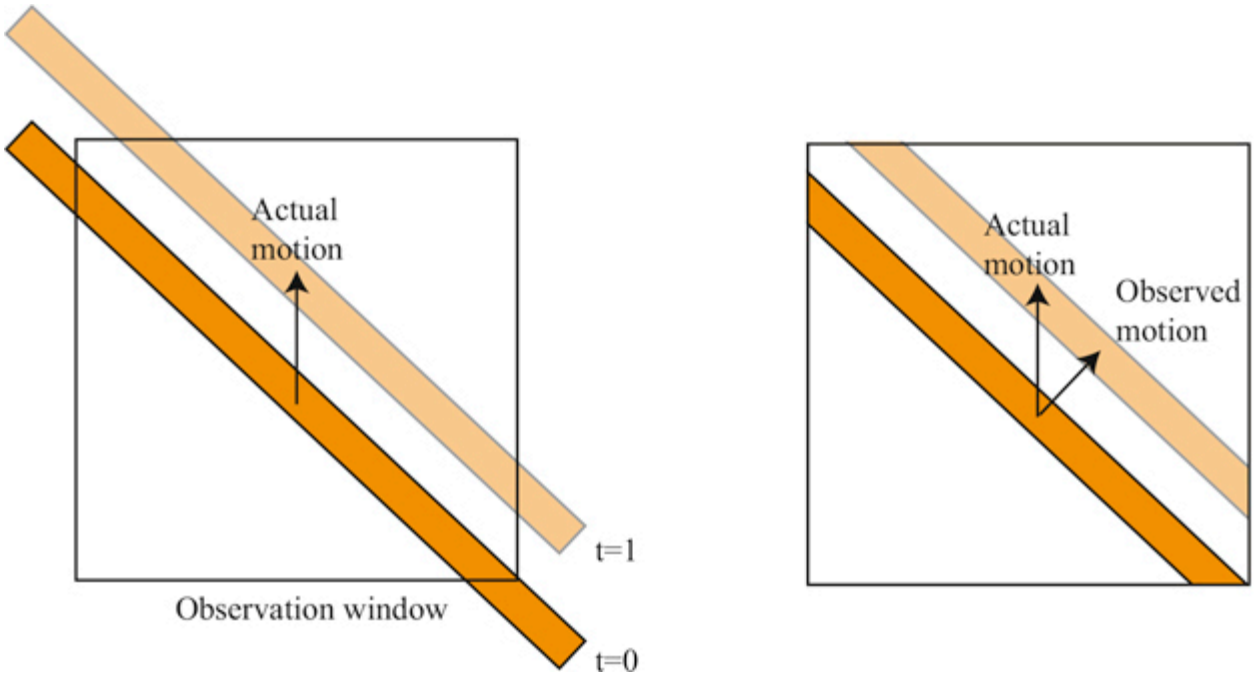
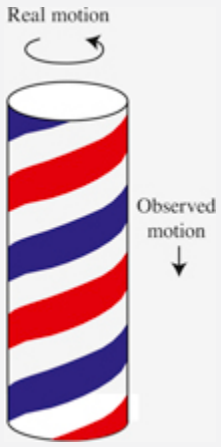


Figure 48.2: Aperture problem when observing the motion of a one-dimensional (1D) structure larger than the image frame. The actual motion of the bar is upward, but the perception, when vision is limited to what is visible within the observation window, appears as if the motion of the bar is in the direction perpendicular to the bar.

The **Barber-pole illusion** is an illustration of the aperture problem. The barber pole induces the illusion of downward motion while rotating.



48.2.3 Representation of Optical Flow

We can model motion over time as a sequence of **dense optical flow** images:

$$\left[u(x, y, t), v(x, y, t)\right] \tag{48.1}$$

The quantities $u(x, y, t)$ and $v(x, y, t)$ indicate that a pixel at image coordinates (x, y) at time t moves

with velocity (u, v). The problem with this formulation is that we do not have an explicit representation of the trajectory followed by points in the scene over time. As time t changes, the location (x, y) might correspond to different scene elements.

An alternative representation is to model motion as a **sparse set of moving points** (tracks):

$$\{\mathbf{P}_t^{(i)}\}_{i=1}^{N} \tag{48.2}$$

This has the challenge that we have to establish the correspondence of the same scene point over time (tracking). The appearance of a 3D scene point will change over time due to perspective projection and other variations such as illumination changes over time. It might be difficult to do this on a dense array. Therefore, most representations of motion as sets of moving points use a sparse set of points.

Both of those motion representations have limitations, and depending on the applications, one representation may be preferred over the other. Choosing the right representation might be challenging in certain situations. For instance, what representation would be the most appropriate to describe the flow of smoke or water over time? (We do not know the answer to this question, and the answer might depend on what do we want to do with it.)

In the rest of this chapter we will use the dense optical flow representation.

48.3Model-Based Approaches

Let's now describe a set of approaches for motion estimation that are not based on learning. Motion estimation equations will be derived from first principles and will rely on some simple assumptions.

In section 46.3 we discussed matching-based optical flow estimation. We will discuss now gradient-based methods. These approaches rely on the brightness constancy assumption and use the reconstruction error as the loss to optimize.

48.3.1Brightness Constancy Assumption

The **brightness constancy assumption** says that as a scene element moves in the world, the brightness captured by the camera from that element does not change over time. Mathematically, this assumption translates into the following relationship, given a sequence $\ell(x, y, t)$:

$$\ell(x+u, y+v, t+1) = \ell(x, y, t) \tag{48.3}$$

where u and v are the element displacement over one unit of time and are also a function of pixel location, $u(x, y)$ and $v(x, y)$, but we will drop that dependency to simplify the notation. The previous relationship is equivalent to saying that $\ell(x, y, t + 1) = \ell(x - u, y - v, t)$. To make sure that the reader remains onboard without getting confused by the indices and how the translation works, here is a simple toy example of a translation with $(u = 1, v = 0)$:

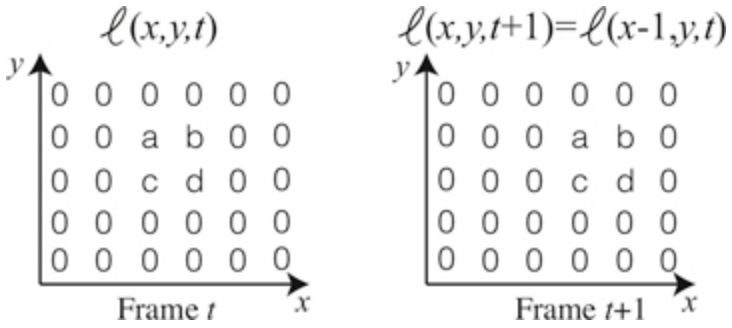


Figure 48.3: Translation to the right of a simple 6×6 size image.

The constant brightness assumption only approximately holds in reality. For it to be exact, we should have a scene with Lambertian objects illuminated from a light source at infinity, with no occlusions, no shadows, and no interreflections. Few real scenes check any of those boxes. This equation assumes that all the pixels in $\ell(x, y, t)$ are visible in $\ell(x, y, t + 1)$, but in reality, some pixels might be occluded in the first frame and new pixels might appear around the image boundaries and behind occlusions.

48.3.2Gradient-Based Optical Flow Estimation

The most popular version of a gradient-based method for optical flow estimation was introduced by Lucas and Kanade in 1981 [311].

Let's start describing the method in words, and we will see next how it translates into math. We will start by approximating the change between two consecutive frames by a linear equation using a Taylor approximation. This linear approximation combined with the constant brightness assumption will result in a linear constraint for the optical flow at each pixel. We will then derive a big system of linear equations for the entire image that, when solved, will result in an estimated optical flow for each image pixel. Let's now see, step by step, how this algorithm works.

If the motion (u, v) is small in comparison to how fast the image ℓ changes spatially, we can use a first-order Taylor expansion of the image $\ell(x + u, y + v, t + 1)$ around (x, y, t):

$$\ell(x+u, y+v, t+1) \simeq \ell(x, y, t) + u\ell_x + v\ell_y + \ell_t + \text{h. o. t.} \tag{48.4}$$

Combining equations (48.3) and (48.4), and ignoring higher order terms, we arrive at the **gradient constraint equation**:

$$u\ell_x + v\ell_y + \ell_t = 0 \tag{48.5}$$

This equation constrains the motion (u, v) in location (x, y) to be along a line perpendicular to the image gradient $\ell = \ell_x, \ell_y$ at that location. This is the same relationship that we discussed before when describing the aperture problem. This is not enough to estimate motion and we will need to add additional constraints. The second assumption that we will add is that the motion field is constant (or smoothly varying) over an extended image region. By solving for the gradient constraints over an

image patch, we hope there will be only a unique velocity that will satisfy all the equations. We can implement this constraint in a different way. One simple way of implementing this constraint is by summing over a neighborhood using a weighting function $g(x, y)$. If $g(x, y)$ is a Gaussian window centered in the origin, the optical flow, (u, v), at the location (x, y) can be estimated by minimizing:

$$\mathcal{L}(u,v) = \sum_{x',y'} g(x'-x, y'-y) \left| u(x,y)\ell_x(x',y',t) + v(x,y)\ell_y(x',y',t) + \ell_t(x',y',t) \right|^2 \tag{48.6}$$

The previous equation is a bit cumbersome because we wanted to make explicit the spatial variables, (x', y'), over which the sum is being made and the factors that are function of the location (x, y) at which the flow is computed. From now, to simplify the derivation, we will drop all the arguments and write the loss in a single location as:

$$\mathcal{L}(u,v) = \sum g \left| u\ell_x + v\ell_y + \ell_t \right|^2 \tag{48.7}$$

where only u and v are constant.

The solution that minimizes this loss can be obtained by computing where the derivatives of the loss with respect to u and v are equal to zero:

$$\frac{\partial \mathcal{L}(u,v)}{\partial u} = \sum g \left(u\ell_x^2 + v\ell_x\ell_y + \ell_x\ell_t \right) = 0 \tag{48.8}$$

$$\frac{\partial \mathcal{L}(u,v)}{\partial v} = \sum g \left(u\ell_x\ell_y + v\ell_y^2 + \ell_y\ell_t \right) = 0 \tag{48.9}$$

We can write the previous two equations in matrix form at each location (x', y') as:

$$\begin{bmatrix} \sum g\ell_x^2 & \sum g\ell_x\ell_y \\ \sum g\ell_x\ell_y & \sum g\ell_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum g\ell_x\ell_t \\ \sum g\ell_y\ell_t \end{bmatrix} \tag{48.10}$$

We will have an equivalent set of equations for each image location. The solution at each location can be computed analytically as $\mathbf{u} = \mathbf{A}^{-1}\mathbf{b}$ where $\mathbf{A}$ is the 2×2 matrix of equation (48.10). The motion at location x, y will only be uniquely defined if we can compute the inverse of $\mathbf{A}$. Note that the matrix $\mathbf{A}$ is a function of the image structure around location (x, y). If the rank of the matrix is 1, we can not compute the inverse and the optical flow will be constrained along a 1D line, this is the aperture problem. This will happen if the image structure is 1D inside the region of analysis that will be defined by the size of the Gaussian window $g(x, y)$.

In order to implement this approach we can make use of convolutions in order to compute all the quantities efficiently in a compact algorithm 48.1:

Algorithm 48.1: Gradient-based optical flow estimation using two input frames.

Input: $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2$ $^{N\times M\times 3}$, **Output:** $\mathbf{u}, \mathbf{v}$ $^{N\times M}$
Parameters: g = integration window
Compute: ℓ_x, ℓ_y, and ℓ_t
Compute A: $A_{1,1} = \ell_x^2 \circ g, A_{1,2} = \ell_x \ell_y \circ g, A_{2,2} = \ell_y^2 \circ g$
Compute b: $b_1 = \ell_x \ell_t \circ g, b_2 = \ell_y \ell_t \circ g$
for $i = 0, \ldots, M-1$ **do**
  for $j = 0, \ldots, N-1$ **do**
    $u[i,j], v[i,j] = inv(\mathbf{A}[i,j])\mathbf{b}[i,j]$

To see how optical flow is computed from gradients, let's consider a simple sequence with two moving squares as shown in figure 48.4 where the top square moves with a velocity of (0, 0.5) pixels/frame and the bottom one moves at (–0.5, –0.5) pixels/frame (the figure shows frames 0 and 10 to make the motion more visible).

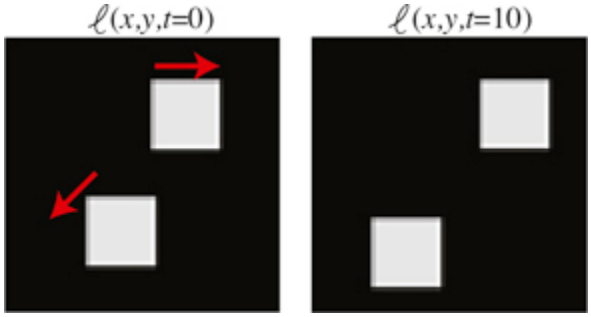


Figure 48.4: Toy sequence with two moving squares. The red arrows indicate the direction of motion of each square.

To apply the gradient-based optical flow algorithm we need to compute the gradients along x, y, and t. In practice, the image derivatives, ℓ_x and ℓ_y, are computed by convolving the image with Gaussian derivatives (see chapter 18). In the experiments here we first blur the two input frames with a Gaussian of $\sigma = 1$, approximated with a five-tap kernel (i.e., a kernel with size 5 × 5 values). For the spatial derivatives we use the kernel [1, –8, 0, 8, –1]/12 for the x-derivative and its transposed for the y -derivative. The temporal derivative can be computed as the difference of two consecutive blurred frames. Choosing the appropriate filters to compute the derivatives is critical to get correct motion estimates (the three derivatives need to be centered at the same spatial and temporal location in the $x - y - t$ volume). For the moving squares sequence, the spatial and temporal derivatives are shown in figure 48.5.

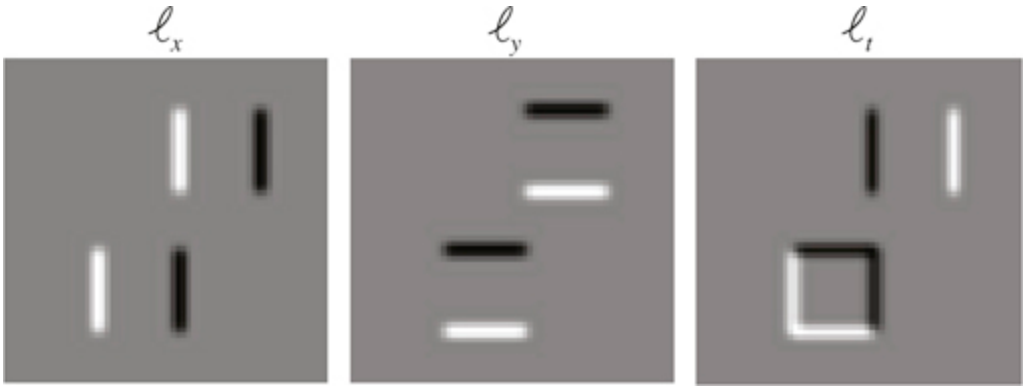


Figure 48.5: Spatial and temporal derivatives for the sequence from figure 48.4.

The next step consists of computing ℓ_x^2, $\ell_x\ell_y$, ℓ_y^2, $\ell_x\ell_t$, and $\ell_y\ell_t$ and blurring them with the Gaussian kernel, g, which will then be used to build the matrix $\mathbf{A}$ at each pixel. Figure 48.6 shows the results.

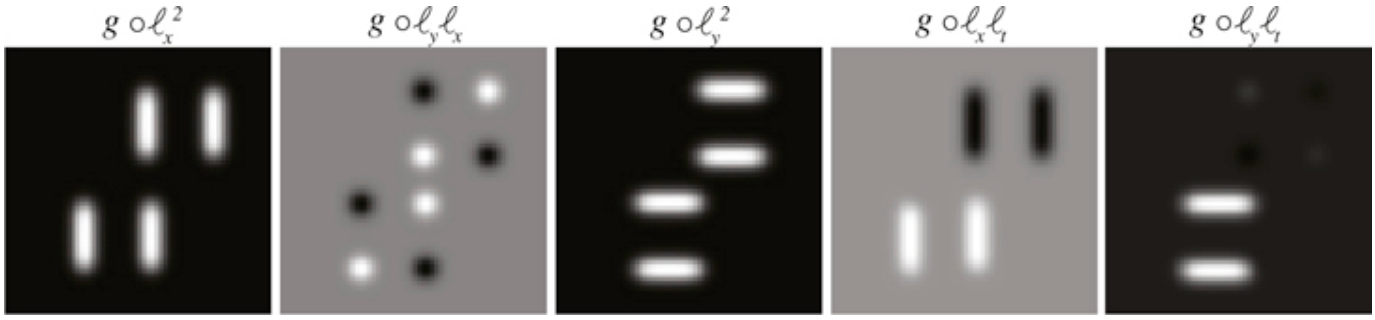


Figure 48.6: Computation of all the products between derivatives from figure 48.5.

In order to compute the optical flow, we need to compute the matrix inverse $\mathbf{A}^{-1}$. This matrix depends only on the spatial derivatives and thus is independent of the motion present in the scene. If we compute optical flow at each pixel, the result looks like the image in figure 48.7.

Figure 48.7: Estimated optical flow for the sequence in figure 48.5.

We can see that the result seems to be wrong for the motion estimated near the center of the side of

each square. The inverse can only be computed in image regions with sufficient texture variations (i.e., near the corners). As in the Harris corner detector (see section 40.3.2.1), the eigenvector with the smallest eigenvalue indicates the direction in which the image has the smallest possible change under translations. Regions with a small minimum eigenvalue are regions with a 1D image structure and will suffer from the aperture problem. To identify regions where the motion will be reliable, we can use the following quantity (proposed by Harris), which relates to the conditioning of the matrix **A**:

$$R = \det(\mathbf{A}) - \lambda \mathrm{Tr}(\mathbf{A})^2 \tag{48.11}$$

where $\lambda = 0.05$ (which is within the range of values used in the Harris corner detector). Figure 48.8 shows R and the estimated optical flow in the regions with $R > 2$. Harris proposed this formulation to avoid the computation of the eigenvalues at each pixel because it is computational expensive.

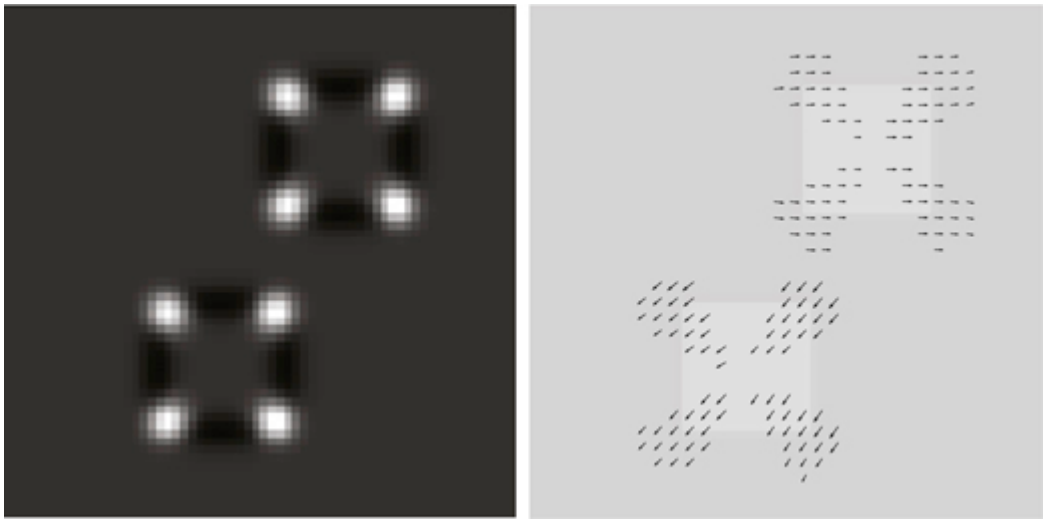


Figure 48.8: Estimated optical flow in the regions with $R > 2$ (around *good features to track* [436]).

The regions for which $R > 2$ will increase by making the apertures larger, which is achieved by using a Gaussian filter, g, with larger σ. However, this will result in smoother estimated flows and the estimated motion will not respect the object boundaries.

One advantage of this approach over the matching-based algorithm is that the estimated flow is not discrete, but it does not work well if the displacement is large, as the Taylor approximation is not valid anymore. One solution to the problem of large displacements is to compute a Gaussian pyramid for the input sequence. The low-resolution scales make the motion smaller and the gradient-based approach will work better.

48.3.3Iterative Refinement for Optical Flow

The gradient-based approach is efficient but it only provides an approximation to the motion field because it ignores higher order terms in the Taylor expansion in equation (48.4). A different approach consists of directly minimizing the **photometric reconstruction** error:

$$L(\ell_1, \ell_2, \mathbf{u}, \mathbf{v}) = \sum_{x,y} \left| \ell_1(x+u, y+v, t+1) - \ell_2(x, y, t)) \right|^2 \tag{48.12}$$

We can now run gradient descent on this loss. Running only one iteration would be similar to the gradient-based optical flow algorithm described in the previous section. However, adding iterations will provide a more accurate estimate of optical flow. At each iteration n, the estimated optical flow will be used to compute the warped frame $\ell_1(x + u_n, y + v_n, t + 1)$ and we will compute an update, Δu_n and Δv_n, of the optical flow: $u_{n+1} = u_n + \Delta u_n$ and $v_{n+1} = v_n + \Delta v_n$. To improve the results, the optical flow estimation is done on a Gaussian pyramid. First, we run a few iterations on the lowest resolution scale of the pyramid (where the motion will be the smallest). The estimated motion is then upsampled and used as initialization at the next level. We iterate this process until arriving at the highest possible resolution. This process is shown in figure 48.9.

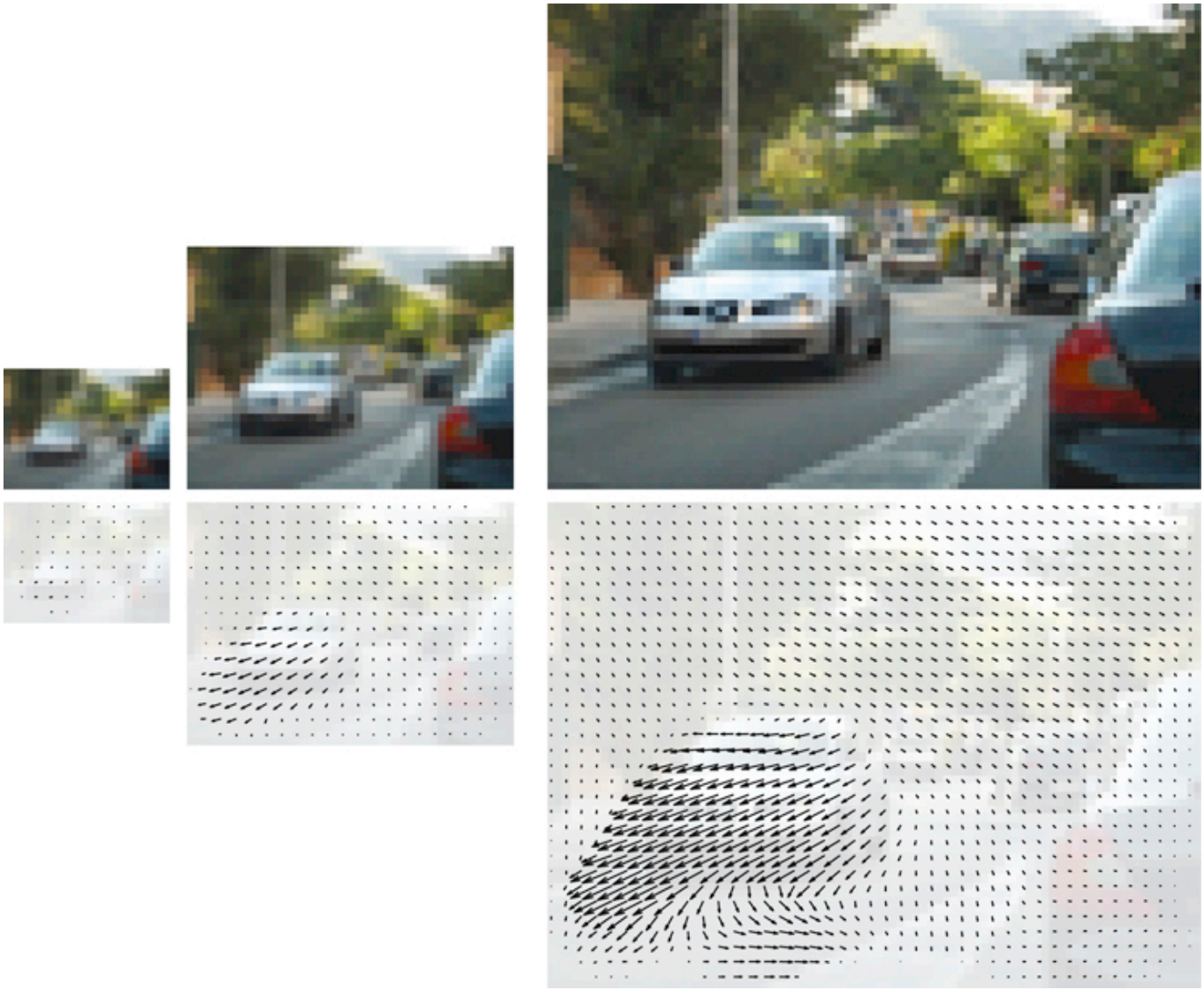


Figure 48.9: Multiscale iterative refinement for optical flow. Optical flow estimation is done on a Gaussian pyramid. (left) First, we run a few iterations on the lowest resolution scale of the pyramid (where the motion will be the smallest). The estimated motion is then upsampled and used as the initialization at the next level. (right) We iterate this process until arriving at the highest possible resolution.

Figure 48.10 compares the optical flow computed using the gradient-based algorithm (i.e., one iteration) and the multiscale iterative refinement approach. Note how the gradient-based approach underestimates the motion of the left car. The displacement between consecutive frames is close to four pixels and that makes the first-order Taylor approximation very poor. The multiscale method is capable of estimating large displacements.

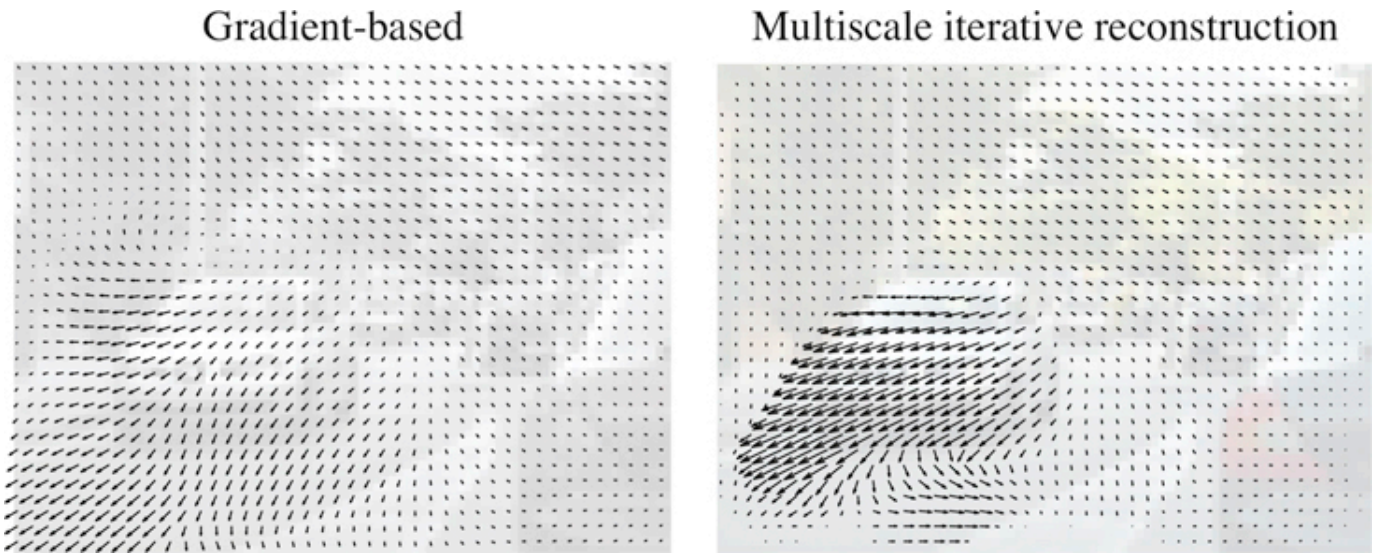


Figure 48.10: Comparison between the optical flow estimated using the gradient-based algorithm and the multiscale iterative refinement approach.

The photometric reconstruction can incorporate a regularization term penalizing fast variations on the estimated optical flow:

$$\mathcal{L}(\mathbf{u}, \mathbf{v}) = L(\ell_1, \ell_2, \mathbf{u}, \mathbf{v}) + \lambda R(\mathbf{u}, \mathbf{v}) \tag{48.13}$$

This problem formulation was introduced by Horn and Schunck in 1981 [218].

There are several popular regularization terms. One penalizes large velocities (**slow prior**):

$$R(\mathbf{u}, \mathbf{v}) = \sum_{x,y} (u(x, y))^2 + (v(x, y))^2 \tag{48.14}$$

Another penalizes variations on the optical flow (**smooth prior**):

$$R(\mathbf{u}, \mathbf{v}) = \sum_{x,y} \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{48.15}$$

The photometric loss plays an important role in unsupervised learning-based methods for optical flow estimation as we will discuss later.

48.3.4Layer-Based Motion Estimation

Until now we have not made use of any of the properties of the motion field derived from the 3D projection of the scene. One way of incorporating some of that knowledge is by making some strong assumptions about the moving scene. If the scene is composed of rigid objects, then we can assume that the motion field within each object will have the form described by equation (47.23).

In this case, instead of a moving camera we have rigid moving objects (which is equivalent). The 2D motion field can then be represented as a collection of superimposed layers, each layer containing one object and occluding the layers below. Each layer will be described by a different set of motion parameters. The parametric motion field can be incorporated into the gradient-based approach described previously. The motion parameters can then be estimated iteratively using an expectation-maximization (EM) style algorithm. At each step we will have to estimate, for each pixel, which layer it is likely to belong to, and then estimate the motion parameters of each layer. The idea of using layers to represent motion was first introduced by Wang and Adelson in 1994 [493].

48.4Concluding Remarks

Motion estimation is an important task in image processing and computer vision. It is used in video denoising and compression. In computer vision it is a key attribute to understand scene dynamics and 3D structure. Despite being studied for a long time, accurate optical flow remains challenging, even when using state-of-the-art deep-learning techniques.

The approaches presented here require no training. In the next chapter, we will study several learning-based methods for motion estimation. The approaches presented in this chapter will become useful when exploring unsupervised learning methods.

49Learning to Estimate Motion

49.1Introduction

We have discussed in the previous sections a number of model-based methods for motion estimation. If these models describe the equations of motion based from first principles, why is that we need learning based methods at all? The reason is that the models make a number of assumptions that are not always true. Also, there are other sources of information that can reveal properties about motion that cannot be modeled but that can be learned.

Causes of modeling errors include failure of the brightness constancy assumption; the presence of occlusions, shadows and changes in illumination; new structures appearing due to changes in the resolution as a result of motion, deformable surfaces; and so on. Many of the motion computations involved approximations such as approximating the derivatives with finite size discrete convolutions. There could also be other motion-relevant cues present in the image, such as monocular depth cues that provide information about the three-dimensional (3D) scene structure and the presence of familiar objects for which we can have strong priors about their motion. These could include that buildings do not move, walls are solid and usually featureless, people are deformable, trees leaves have huge number of occlusions, and so on. Those semantic properties can be implicitly exploited by a learning-based model.

49.2Learning-Based Approaches

Learning-based approaches rely on many of the concepts we introduced in the previous chapters. We will differentiate between two big families of models: supervised models that learn to estimate motion using a database of training examples, and unsupervised models that learn to estimate motion without training data.

49.2.1Supervised Models for Optical Flow Estimation

The simplest formulation for learning to estimate optical flow is when we have available a dataset of image sequences with associated ground truth optical flow. Researchers have used synthetic data [68], using lidar [157] or human annotations [302] to build datasets with ground truth motion. In previous approaches, ground truth data could be used for evaluation; however, we will use it here to train a model to predict motion directly from the input frames.

49.2.1.1Architectures

As in the case of stereo, we can train a function to estimate optical flow from two frames:

$$\left[\hat{\mathbf{u}}, \hat{\mathbf{v}}\right] = h_\theta\left(\ell_1, \ell_2\right) \tag{49.1}$$

One of the first approaches to use this formulation with neural networks was FlowNet [108]. The architecture is simple.

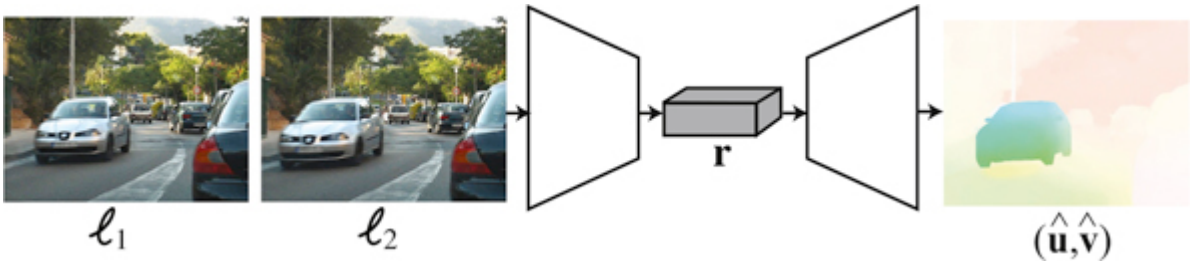


Figure 49.1: In FlowNet the direct approach estimates optical flow directly from a pair of frames.

The direct approach depicted in figure 49.1 learns to estimate optical flow directly from a pair of frames. This architecture makes no assumptions about which architectural priors are needed to compute optical flow from images. The architecture is trained end-to-end using ground truth optical flow.

Another common approach, depicted in the block diagram shown in figure 49.2, is to define an architecture that follows the same steps as traditional approaches:

- Extract features from each image using a pair of networks with shared weights. This can be done by a feature pyramid [297].
- Form a 3D cost volume indicating the local visual evidence of a match between the two images for each possible pixel position. This 3D cost volume can be referenced to the H and V positions of one of the input images (generally the first frame is the reference frame).
- Train and apply a CNN to aggregate (process) the costs over the cost volume in order to estimate a single best optical flow for each pixel position.
- Use a coarse-to-fine estimation procedure where optical flow estimated at a coarse scale is used to warp the features at a finer scale to compute a refined cost volume. Then, estimate an update to the optical flow to warp the features and the next finer level of the pyramid.

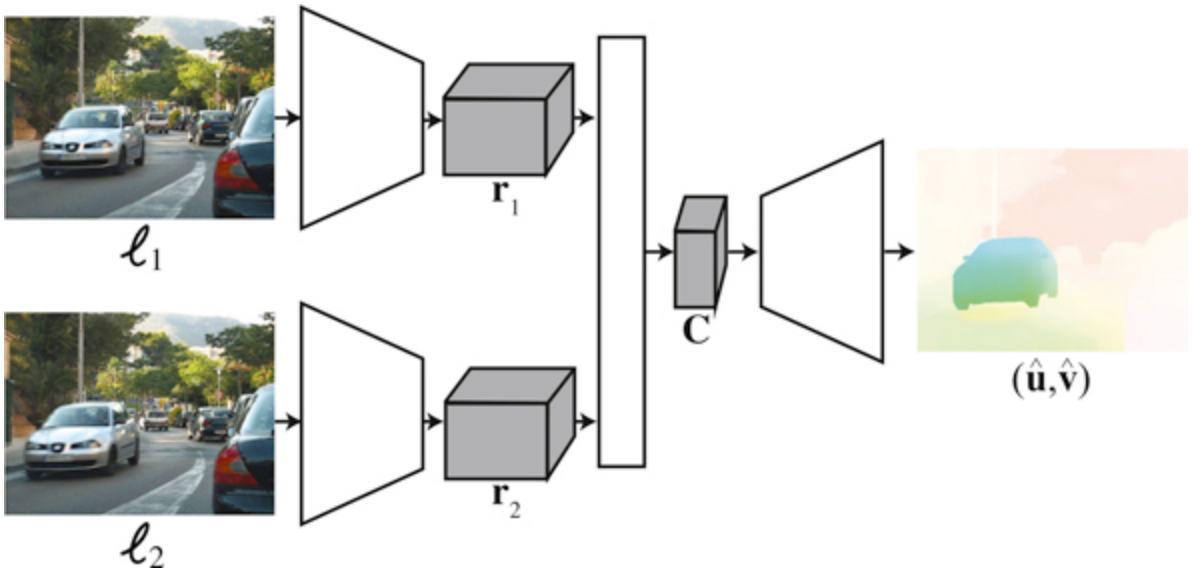


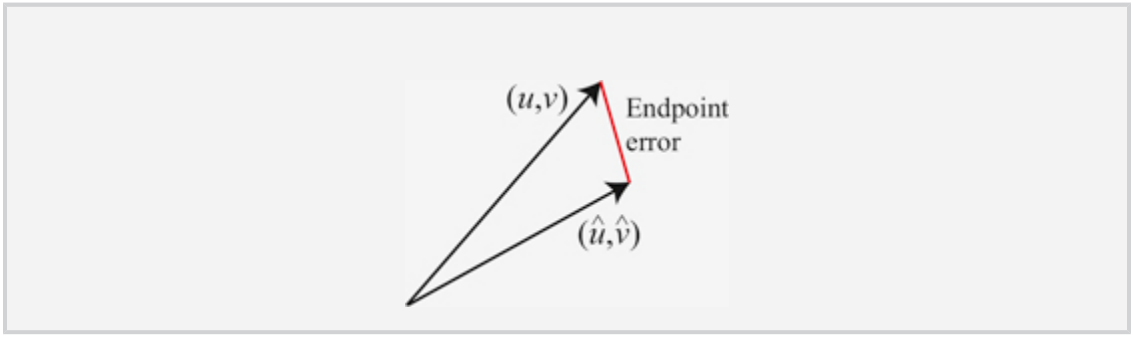
Figure 49.2: Motion estimation. (1) Extract features from each image; (2) compute a 3D cost volume; and (3) aggregate the cost volume in order to estimate the best optical flow for each pixel.

Other variations over this architecture incorporate some of the concepts we studied before, such as coarse-to-fine refinement, matching, and smoothing. Different approaches will differ in some of the details of how each step is implemented and how training takes place. The main building blocks can be implemented with convolutional neural networks or transformers. The main difference between the matching-based and gradient-based methods described earlier is that instead of using predefined functions, the architectures are trained end-to-end to minimize the optical flow error when compared with ground truth data.

49.2.1.2Loss functions

In supervised optical flow estimation, the most common loss is the **endpoint error**, which is the average, over the whole image, if the distance between the estimated optical flow vector, $(\hat{u}, \hat{v})$ and the ground-truth vector, (u, v):

$$\mathcal{L}(\hat{\mathbf{u}}, \hat{\mathbf{v}}, \mathbf{u}, \mathbf{v}) = \sum_{n,m} (\hat{u}[n,m] - u[n,m])^2 + (\hat{v}[n,m] - v[n,m])^2 \tag{49.2}$$



The sum is over all the pixels in the image. Each pixel has an estimated optical flow $(\hat{u}[n,m], \hat{v}[n,m])$.

49.2.1.3Handling occlusions

One of the challenges for estimating optical flow is that, as objects move, they will occlude some pixels from the background and reveal new ones. Therefore, together with the estimated flow it is also convenient to detect occluded pixels. If we have ground truth data, we can train a function to estimate optical flow and the occlusion map from two frames:

$$\left[\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{o}}\right] = h_\theta\left(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2\right) \tag{49.3}$$

49.2.1.4Training set

The biggest challenge of using a supervised model for motion estimation is that ground truth data is very hard to collect. This is probably one of the main limitations of these approaches. There are some small existing datasets, although this might change in a few years.

The largest existing datasets are synthetic 3D scenes with moving objects that can be rendered, which will give us perfect ground truth data to train the regression function. There are several examples of existing datasets such as this, like the Middelbury dataset [425], which contains six real and synthetic sequences with ground truth optical flow. The optical flow for the real sequences was obtained by tracking hidden fluorescent texture. The KITTI dataset [157] contains real motion recorded from a moving car. The MPI Sintel [68] contains synthetic sequences made with great effort to make the scenes look realistic. Finally, the Flying Chairs dataset is an interesting synthetic dataset that consists of pasting the image of a random number of chairs over a background image [108]. Motion is created by applying different affine transformations to the background and the chairs. These sequences are easy to generate and pay little attention to their realism. This makes it possible to generate a very large number of sequences for training, allowing for competitive performance when used to train a neural network.

49.2.2Unsupervised Learning of Optical Flow

Collecting ground truth data is the Achilles heel for learning-based approaches. This is particularly true for optical flow as it can not be recorded directly. Ground truth data optical flow can be obtained on synthetic data only, and for real data one needs to create specific recording scenarios that allow inferring accurate optical flow or relying on noisy human annotations [302]. As a consequence, real data collection is expensive and nonscalable.

Is it possible to learn to estimate optical flow by just looking at movies without using ground truth data?

Unsupervised methods for training an optical flow model will make some assumptions about dynamic image formation. Those assumptions will be similar to the ones we have presented all along this chapter: (1) when the motion is due only to camera motion, the optical flow will have to fit the equations of the projected motion that provide constraints that can be used to train a model; (2) we can assume that the appearances of objects and surfaces in the scene do not change due to motion (brightness constancy assumption); and (3) we can expect the optical flow to be smooth over regions, although with sharp variations along occlusion boundaries.

One typical formulation consists in learning to predict the displacement from frame 1 to frame 2, so that if we warp frame 1 we minimize the reconstruction error of frame 2. This is achieved by using the photometric loss:

$$L_{photo}(\ell_1, \ell_2, \mathbf{u}, \mathbf{v}) = \sum_{x,y} \left| \ell_2(x + \hat{u}, y + \hat{v}) - \ell_1(x, y)) \right|^2 \tag{49.4}$$

where now $[\hat{\mathbf{u}}, \hat{\mathbf{v}}] = h_\theta(\ell_1, \ell_2)$.

The learning is done by searching over the parameter space for the parameters θ that minimize the photometric loss over a large collection of videos. The photometric loss can also be replaced by the L1 or other robust norms. If the network also predicts occlusions, the photometric loss can include a weight that cancels the contribution of occluded pixels to the loss.

The network can also take as input multiple frames and not just two.

49.3Concluding Remarks

Supervised and unsupervised learning-based methods are now the state-of-the-art in motion estimation. But an accurate solution is still missing. One important question is, do we really need learning in order to solve this problem? Should we abandon the derivation of physically motivated algorithms for motion estimation that require no training? Our answer is that we should pursue both directions of work.

XIII
UNDERSTANDING VISION WITH LANGUAGE

Visual understanding consists of inferring scene properties from images. Some properties might refer to mid-level scene attributes such as motion or depth, while others may relate to high-level features like semantic segmentation.

In this part we will focus on the semantics of visual processing; that is, the association between visual stimuli and meaning. The goal is to infer from visual input, what is the signification of what we see. Therefore, there is a strong connection between semantic visual processing and natural language processing.

- **Chapter 50** describes how learn to recognize and localize objects in images, assigning words to them.
- **Chapter 51** explores the role of language as a representation of the visual world and its connection with vision systems.