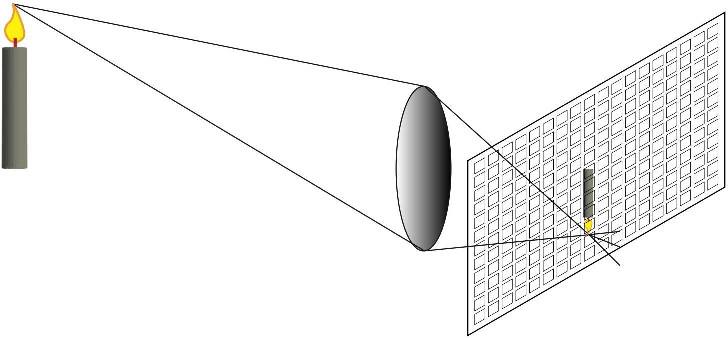
Image Motion

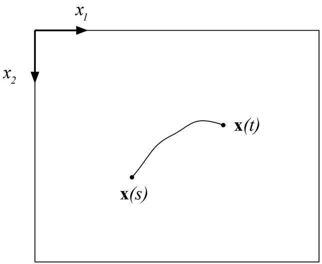
COMPSCI 527 — Computer Vision

Outline

Continuous and Discrete Image



Motion Field and Displacement

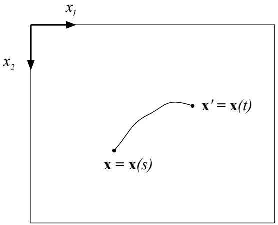


- Follow the image projection $\mathbf{x}(t)$ of a single world point
- *Displacement*: $\mathbf{d}(t, s) = \mathbf{x}(t) - \mathbf{x}(s)$, a difference in positions
- *Motion field*: $\mathbf{v}(t) = \frac{d\mathbf{x}(t)}{dt}$, an instantaneous velocity
- A *field* b/c it can be defined for every $\mathbf{x}$ in the image plane

Constancy of Appearance

- *Images do not move*
- What is assumed to remain constant across images?
- Motion estimation is impossible without such an assumption
- Most generic assumption: The appearance of a point does not change with time or viewpoint
- If two image points in two images correspond, they look the same
- "Appearance:" Image *irradiance* $e(\mathbf{x}, t)$ (brightness)
- If colors differ, so do brightnesses most of the time, so color does not help much
- We only consider gray images and video from now on

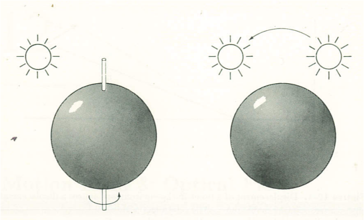
Constancy of Appearance



- If two image points in two images correspond, they look the same
- If $\mathbf{x}$ at time s and $\mathbf{x}'$ at time t correspond, then $e(\mathbf{x}, s) = e(\mathbf{x}', t)$ (finite-displacement formulation)
- Equivalently, $\frac{de(\mathbf{x}(t),t)}{dt} = 0$ (differential formulation)
- This is the key constraint for motion estimation

Motion Field and Optical Flow

- Extreme violations of constancy of appearance:



B. K. P. Horn, *Robot Vision*, MIT Press, 1986

- Ill-defined distinction:
  - Motion field $\approx$ true motion
  - Optical flow $\approx$ locally observed motion
- Still assume constancy of appearance almost everywhere
- What else can we do?

The Brightness Change Constraint Equation

- The appearance of a point does not change with time or viewpoint: $\frac{de(\mathbf{x}(t),t)}{dt} = 0$
- Total derivative, not partial:

$$\frac{de(\mathbf{x}(t),\ t)}{dt} \stackrel{\text{def}}{=} \lim_{\Delta t \to 0} \frac{e(\mathbf{x}(t+\Delta t),\ t+\Delta t) - e(\mathbf{x}(t),\ t)}{\Delta t}$$

- Use chain rule on $\frac{de(\mathbf{x}(t),t)}{dt} = 0$ to obtain the *Brightness Change Constraint Equation* (BCCE)

$$\frac{\partial e}{\partial \mathbf{x}^T}\frac{d\mathbf{x}}{dt} + \frac{\partial e}{\partial t} = 0$$

- $\mathbf{v} \stackrel{\text{def}}{=} \frac{d\mathbf{x}}{dt}$ is the unknown motion field
- *This is the key constraint for motion estimation*

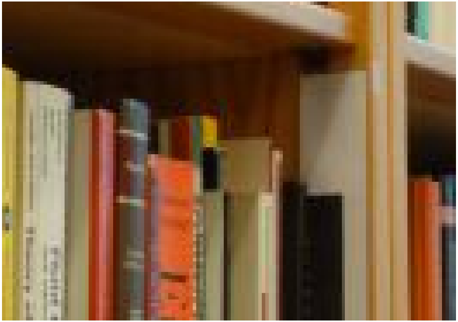
(Compare: $\frac{\partial e(\mathbf{x}(t),t)}{\partial t} \stackrel{\text{def}}{=} \lim_{\Delta t \to 0} \frac{e(\mathbf{x}(t),\ t+\Delta t) - e(\mathbf{x}(t),\ t)}{\Delta t}$)

The Aperture Problem

- Issues arise even when the appearance is constant

$$\text{BCCE:} \qquad \frac{\partial e}{\partial \mathbf{x}^T}\mathbf{v} + \frac{\partial e}{\partial t} = 0$$

- One equation in two unknowns: the *aperture problem*



The Aperture Problem

$$\text{BCCE:} \qquad \frac{\partial e}{\partial \mathbf{x}^T}\mathbf{v} + \frac{\partial e}{\partial t} = 0$$

- The BCCE is always under-determined: the *aperture problem*
- Cannot recover motion based on point measurements alone
- Can at most recover the *normal component* along the gradient $\nabla e(\mathbf{x}) = \frac{\partial e}{\partial \mathbf{x}^T}$ (if the gradient is nonzero):

  $$v(\mathbf{x}) \stackrel{\text{def}}{=} \|\nabla e(\mathbf{x})\|^{-1} \, [\nabla e(\mathbf{x})]^T \, \mathbf{v}(\mathbf{x})$$

Estimating the Motion Field

- Because of the aperture problem, we can only estimate several displacement vectors **d** or motion field vectors **v** simultaneously, not each individually
- Estimation problems are *coupled* across the image
- *Global* estimation methods
    - A *data term* measures deviations from BCCE at every pixel in the image
    - A *smoothness term* measures deviations of the motion field $\mathbf{v}(\mathbf{x})$ from smoothness
    - Minimize a linear combination of the two types of terms, integrated over the image
    - Tend to blur the solution near motion boundaries (discontinuities in the motion field)
    - Will see some global methods later

Local Estimation Methods

- Local methods are an alternative to global ones
- Basic idea:
  - The image displacement $\mathbf{d}$ in a small window around a pixel $\mathbf{x}$ is assumed to be constant over the window (extreme local smoothness)
  - Write one BCCE for every pixel in the window
  - Solve for the one displacement that satisfies all these equations as much as possible
  - A *linear system* to be solved (in the LSE sense)
- These are *(feature) window tracking* methods
- Any method needs to account for the difference between velocity and displacement

Window Tracking

- Given images $f(\mathbf{x})$ and $g(\mathbf{x})$, a point $\mathbf{x}_f$ in image f, and a square window $W(\mathbf{x}_f)$ of side-length $2h+1$ centered at $\mathbf{x}_f$, what are the coordinates $\mathbf{x}_g = \mathbf{x}_f + \mathbf{d}^*(\mathbf{x}_f)$ of the corresponding window's center in image g?
- $\mathbf{d}^*(\mathbf{x}_f) \in \mathbb{R}^2$ is the *displacement* of that point feature
- Assumption 1: The whole window translates
- Assumption 2: $\mathbf{d}^*(\mathbf{x}_f) \ll h$

General Window Tracking Strategy

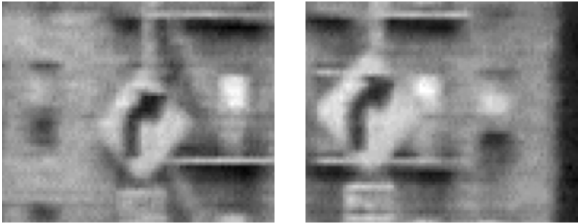
- Let $w(\mathbf{x})$ be the indicator function of $W(\mathbf{0})$
- Measure the *dissimilarity* between $W(\mathbf{x}_f)$ in f and a candidate window $W(\mathbf{x}_f + \mathbf{d})$ in g with the *loss*

$$L(\mathbf{x}_f, \mathbf{d}) = \sum_{\mathbf{x}} [g(\mathbf{x} + \mathbf{d}) - f(\mathbf{x})]^2 \, w(\mathbf{x} - \mathbf{x}_f)$$

- Minimize $L(\mathbf{x}_f, \mathbf{d})$ over $\mathbf{d}$: $\quad \mathbf{d}^*(\mathbf{x}_f) = \arg\min_{\mathbf{d} \in R} L(\mathbf{x}_f, \mathbf{d})$
- The *search range* $R \subseteq \mathbb{R}^2$ is a square centered at the origin
- Half-side of R is $\ll h$ (the half-side of W)

Obvious Failure Points

- Multiple motions in the same window



(Less dramatic cases arise as well)

- Actual motion large compared with h

(We'll come back to this later)

A Softer Window

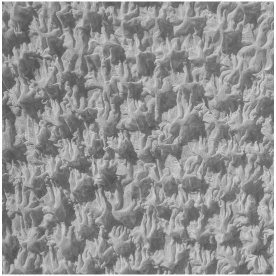
- Make $w(\mathbf{x})$ a (truncated) Gaussian rather than a box

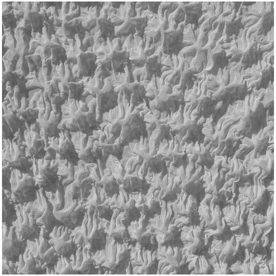
$$w(\mathbf{x}) \propto \begin{cases} e^{\frac{1}{2}\left(\frac{\|\mathbf{x}\|}{\sigma}\right)^2} & \text{if } |x_1| \leq h \text{ and } |x_2| \leq h \\ 0 & \text{otherwise} \end{cases}$$

- Dissimilarity $L(\mathbf{x}_f, \mathbf{d}) = \sum_{\mathbf{x}} [g(\mathbf{x} + \mathbf{d}) - f(\mathbf{x})]^2 \; w(\mathbf{x} - \mathbf{x}_f)$ depends more on what's around the window center
- Reduces the effects of multiple motions
- Does not eliminate them

How to Minimize $L(\mathbf{x}_f, \mathbf{d})$?

- Method 1: Exhaustive search over a grid of $\mathbf{d}$
- Advantages: Unlikely to be trapped in local minima





- Disadvantage: Fixed resolution
- Accurate motion is sometimes necessary
- Using a very fine grid would be very expensive
- Exhaustive search may provide a good initialization

How to Minimize $L(\mathbf{x}_f, \mathbf{d})$?

- Method 2: Use a gradient-descent method
- Search space has low dimension ($\mathbf{d} \in \mathbb{R}^2$), so we can use Newton's method for faster convergence
- Compute gradient and Hessian of
  $L(\mathbf{d}) = \sum_{\mathbf{x}} [g(\mathbf{x} + \mathbf{d}) - f(\mathbf{x})]^2 \; w(\mathbf{x} - \mathbf{x}_f)$
  (omitted $\mathbf{x}_f$ from arguments of L for simplicity)
- Take Newton steps
- Technical difficulty: the unknown $\mathbf{d}$ appears inside $g(\mathbf{x} + \mathbf{d})$, and computing a Hessian would require computing second-order derivatives of an image, which is available only through its pixels
- Second derivatives of images are very sensitive to noise

The Lucas-Kanade Tracker, 1981

- Instead of computing the Hessian of $L(\mathbf{d}) = \sum_{\mathbf{x}} [g(\mathbf{x} + \mathbf{d}) - f(\mathbf{x})]^2 \; w(\mathbf{x} - \mathbf{x}_f)$, linearize $g(\mathbf{x} + \mathbf{d}) \approx g(\mathbf{x}) + [\nabla g(\mathbf{x})]^T \mathbf{d}$
- This brings $\mathbf{d}$ "outside g"
- $L(\mathbf{d})$ is now quadratic in $\mathbf{d}$, and we can find a minimum in closed form by taking the gradient (no Hessian required)
- Only differentiate the image once to get $\nabla g(\mathbf{x})$
- Since the solution $\mathbf{d}_1$ relies on an approximation, we iterate: Shift g by $\mathbf{d}_1$ to make the residual $\mathbf{d}$ smaller, and repeat
- This method works for losses that are sums of squares, and is called the *Newton-Raphson method*

Lucas-Kanade Overall Scheme

- Initialize: $\mathbf{d}_0 = \mathbf{0}$

- Find a displacement $\mathbf{s}_1$ by minimizing linearized $L(\mathbf{d}_0 + \mathbf{s})$
- Shift g by $\mathbf{s}_1$ to obtain g_1
- Accumulate: $\mathbf{d}_1 = \mathbf{d}_0 + \mathbf{s}_1$

- Find a displacement $\mathbf{s}_2$ by minimizing linearized $L(\mathbf{d}_1 + \mathbf{s})$
- Shift g_1 by $\mathbf{s}_2$ to obtain g_2
- Accumulate: $\mathbf{d}_2 = \mathbf{d}_1 + \mathbf{s}_2$

- $\ldots$

Lucas-Kanade Derivation

- Let $\mathbf{d}_t = \mathbf{s}_1 + \ldots + \mathbf{s}_t$ (accumulated shifts, initially $\mathbf{0}$)
- Let $g_t(\mathbf{x}) \stackrel{\text{def}}{=} g(\mathbf{x} + \mathbf{d}_t)$
- We seek $\mathbf{d}_{t+1} = \mathbf{d}_t + \mathbf{s}_{t+1}$ by minimizing the following over $\mathbf{s}$
  $L(\mathbf{d}_t + \mathbf{s}) = \sum_{\mathbf{x}} [g_t(\mathbf{x} + \mathbf{s}) - f(\mathbf{x})]^2 \, w(\mathbf{x} - \mathbf{x}_f)$
  with linearization $g_t(\mathbf{x} + \mathbf{s}) \approx g_t(\mathbf{x}) + [\nabla g_t(\mathbf{x})]^T \mathbf{s}$, so that

$$
\begin{aligned}
L(\mathbf{d}_t + \mathbf{s}) &= \sum_{\mathbf{x}} [g_t(\mathbf{x} + \mathbf{s}) - f(\mathbf{x})]^2 \, w(\mathbf{x} - \mathbf{x}_f) \\
&\approx \sum_{\mathbf{x}} [g_t(\mathbf{x}) + [\nabla g_t(\mathbf{x})]^T \mathbf{s} - f(\mathbf{x})]^2 \, w(\mathbf{x} - \mathbf{x}_f) \; ,
\end{aligned}
$$

a quadratic function of $\mathbf{s}$

Lucas-Kanade Derivation, Cont'd

- Gradient of
  $L(\mathbf{d}_t + \mathbf{s}) \approx \sum_{\mathbf{x}} \{g_t(\mathbf{x}) + [\nabla g_t(\mathbf{x})]^T \mathbf{s} - f(\mathbf{x})\}^2 \, w(\mathbf{x} - \mathbf{x}_f)$ is
  $\nabla L(\mathbf{d}_t + \mathbf{s}) \approx 2 \sum_{\mathbf{x}} \nabla g_t(\mathbf{x}) \{g_t(\mathbf{x}) + [\nabla g_t(\mathbf{x})]^T \mathbf{s} - f(\mathbf{x})\} \, w(\mathbf{x} - \mathbf{x}_f)$
- Setting to zero yields

The Core System of Lucas-Kanade

Linear, 2×2 system

$$A\mathbf{s} = \mathbf{b}$$

where

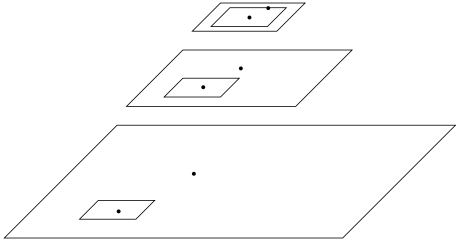
$$A = \sum_{\mathbf{x}} \nabla g_t(\mathbf{x})[\nabla g_t(\mathbf{x})]^T \; w(\mathbf{x} - \mathbf{x}_f)$$

and

$$\mathbf{b} = \sum_{\mathbf{x}} \nabla g_t(\mathbf{x})[f(\mathbf{x}) - g_t(\mathbf{x})] \; w(\mathbf{x} - \mathbf{x}_f) \; .$$

- Solution yields $\mathbf{s}_t$ (real-valued)
- Shift image g_t is by $\mathbf{s}_t$ by *bilinear interpolation* $\rightarrow g_{t+1}$
- Accumulate shifts $\mathbf{d}_{t+1} = \mathbf{d}_t + \mathbf{s}_t \quad (g_{t+1}$ is g shifted by $\mathbf{d}_t)$
- This shift makes f and g_t more similar within the windows
- Repeat until convergence. Final $\mathbf{d}_t$ is the answer

If Motion is Large, Track in a Pyramid



- A large motion at fine level is small at coarse level
- (Only drawing one frame per level, for simplicity)
- Start at the coarsest level (same window size at all levels)
- Multiply solution **d** by 2 to initialize tracking at the next level
- Motion is progressively refined at every level