Estimating Optic Flow by Optimization

D.A. Forsyth,

University of Illinois at Urbana Champaign

Photometric consistency

- Assume:
  - Color of a surface does not change when it moves
    - usually, intensity
    - multiple if's, and's, but's

Flow estimation rests on the assumption that the intensity of a point on an object does not change when the object or the camera move. This assumption means you can rule out certain flows, because they imply that the object has changed intensity. It is mostly sound, but there are cases where the assumption doesn't apply. There are a variety of reasons the amount of light reflected from the surface of the object to the camera might change when the object moves by a small amount. If the material of the object has a strong specular or glossy component, the intensity might change because the angles to sources will change. Alternatively, the object might move into a shadow volume. Finally, the illumination intensity varies across space (for example, as a result of interreflections).

Photometric consistency - II

This assumption produces a very important constraint on flow. At time t, a point $\mathbf{V}$ on an object projects to the point $\mathbf{x}$ in the image, producing intensity $I(\mathbf{x}, t)$. The camera then moves by a small amount and at time $t + \delta t$ a new image is obtained. The point now projects to $\mathbf{x} + (\delta t)\mathbf{u}$, where $\mathbf{u}$ is the flow. Then $I(\mathbf{x} + (\delta t)\mathbf{u}, t + \delta t)$ is the same as $I(\mathbf{x}, t)$, so

$$I(\mathbf{x} + (\delta t)\mathbf{u}, t + \delta t) - I(\mathbf{x}, t) = 0.$$

I will refer to this property as *photometric consistency*. The form is inconvenient, because the constraint is not linear.

If the flow and δt are small enough, a Taylor series is a good approximation, and you find

$$I(\mathbf{x} + (\delta t)\mathbf{u}, t + \delta t) \approx (I(\mathbf{x}, t) + (\delta t)\left[(\nabla I)^T \mathbf{u} + \frac{\partial I}{\partial t}\right]$$

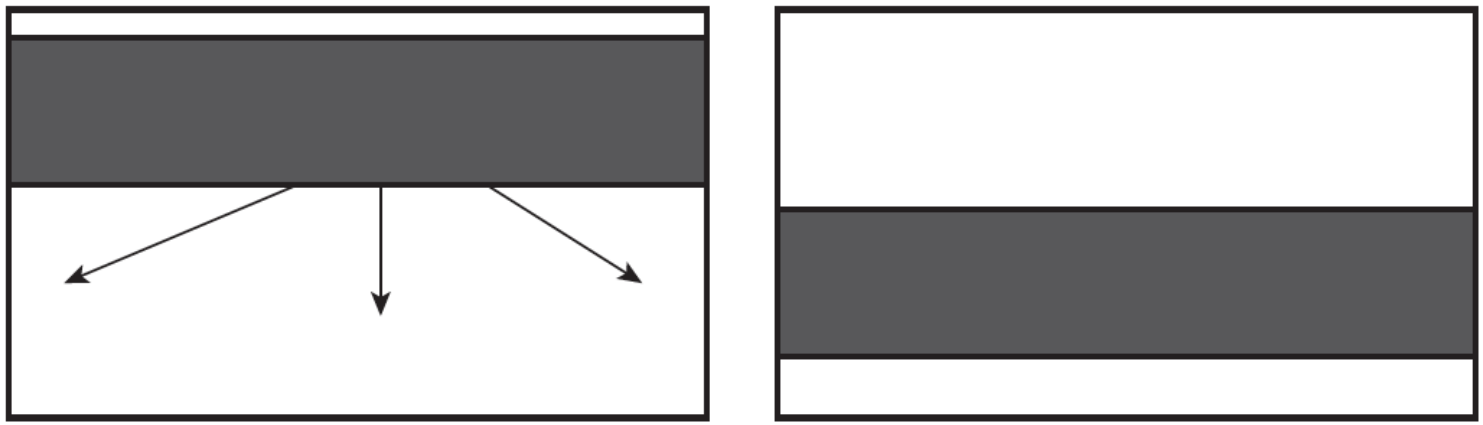
so that for sufficiently small δt

$$(\nabla I)^T \mathbf{u} + \frac{\partial I}{\partial t} = 0.$$

The Optical Flow Equation

$$\nabla \mathcal{I}^T \mathbf{u} + \frac{\partial \mathcal{I}}{\partial t} = 0$$

Still ambiguous – one constraint on two variables



Largely unambiguous at corners

- So smooth:
  - compute flow that has
    - very small gradient magnitudes
    - meets optic flow equation

Horn-Schunk (now obsolete)

$$D(u,v) = \sum_{i,j \in \text{pixels}} \left((\nabla I_{ij})^T \mathbf{u}_{ij} + \frac{\partial I_{ij}}{\partial t} \right)^2$$

as a data term. Here the subscripts refer to the value at the locations (so ∇I_{ij} is the gradient of I at i, j and so on). A simple penalty term follows from assuming that the gradient of each flow component is small. Flow components are u, v, so

$$P(u,v) = \sum_{i,j \in \text{pixels}} \left([\nabla u_{ij}]^T [\nabla u_{ij}] + [\nabla v_{ij}]^T [\nabla v_{ij}] \right)$$

should be small. Then $C(u,v) = D(u,v) + \alpha P(u,v)$ where α weights the penalty term against the data term. This cost function is quadratic in u and v, so a solution is obtained with linear algebra. Solving the resulting flow equation is straightforward, but you should expect reasonable results only for very small flows (**exercises**). The method I have described is known as the *Horn-Schunk* method, and was originally described in [].

Farneback

- Assume small image windows translate
  - flow is (close to) constant in a small window
- Approximate image windows with quadratic

$$\mathcal{I}_1(\mathbf{u}) = \mathbf{u}^T \mathcal{A}_1 \mathbf{u} + \mathbf{b}_1^T \mathbf{u} + c_1$$

- If window translates, you get

$$\mathbf{u}^T \mathcal{A}_2 \mathbf{u} + \mathbf{b}_2^T \mathbf{u} + c_2 = (\mathbf{u} - \mathbf{t})^T \mathcal{A}_1 (\mathbf{u} - \mathbf{t}) + \mathbf{b}_1^T (\mathbf{u} - \mathbf{t}) + c_1$$

so that $\mathcal{A}_2 = \mathcal{A}_1$, $\mathbf{b}_2 = (\mathbf{b}_1 - 2\mathcal{A}_1\mathbf{t})$, $c_2 = c_1 - \mathbf{b}_1^T\mathbf{t} + \mathbf{t}^T\mathcal{A}_1\mathbf{t}$, which yields a linear system in $\mathbf{t}$.

Farneback - II

- Quadratic approximation at each pixel
- Find flow that minimizes errors

In turn, a flow estimation procedure follows. Compute the best quadratic approximation at each location in $\mathcal{I}_1$ and $\mathcal{I}_2$, yielding $\mathcal{A}_1(\mathbf{x})$, and so on. Here $\mathbf{x}$ identifies the center pixel of the window for which you computed the quadratic approximation. Then an estimate of flow at each location is given by solving

$$\frac{1}{2}\left(\mathcal{A}_1(\mathbf{x}) + \mathcal{A}_2(\mathbf{x})\right)\mathbf{t}(\mathbf{x}) = -\frac{1}{2}(\mathbf{b}_2(\mathbf{x}) - \mathbf{b}_1(\mathbf{x}))$$

to obtain $\mathbf{t}(\mathbf{x})$, the flow field. Although $\mathcal{A}_1(\mathbf{x})$ should be the same as $\mathcal{A}_2(\mathbf{x})$, you should use

$$\frac{1}{2}\left(\mathcal{A}_1(\mathbf{x}) + \mathcal{A}_2(\mathbf{x})\right)$$

Farneback - III

- Average error over windows
  - and weight (gaussian weights)

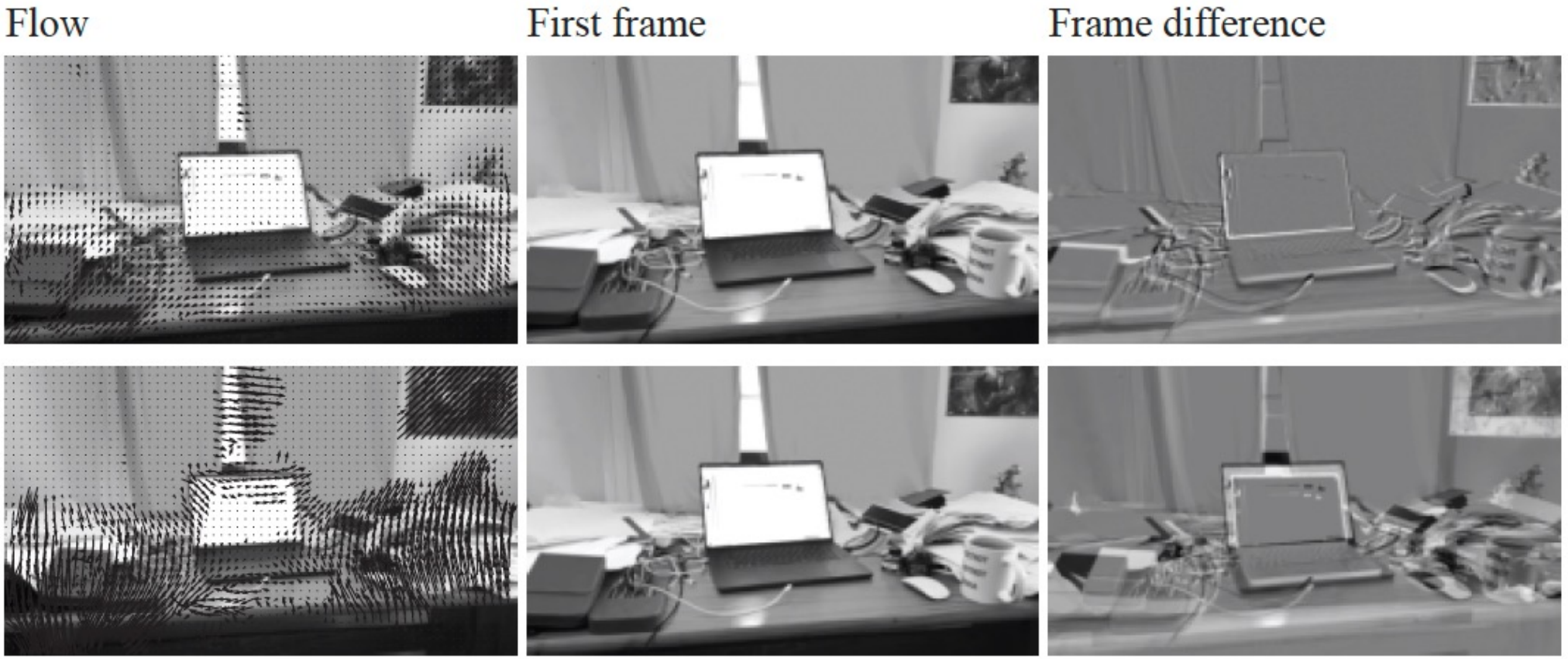
be noisy. An improvement is available by assuming that the flow field is smooth, so that nearby windows can contribute to the estimate. At $\mathbf{x}$, choose a flow that minimizes a weighted average of errors over a local window given by some set of

offsets δx. This gives the optimization problem

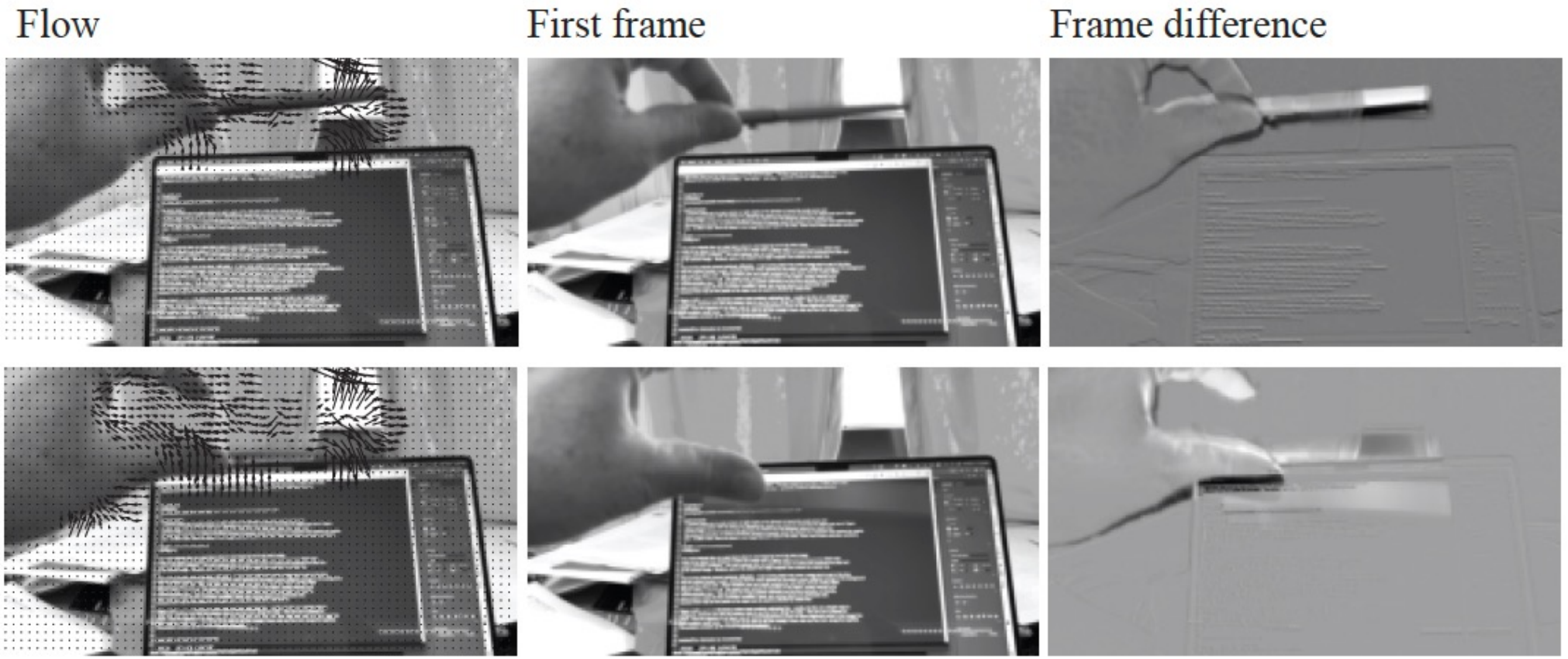
$$\mathbf{t}(\mathbf{x}) = \underset{\mathbf{u}}{\text{argmin}} \sum_{\delta x \in \text{window}} w(\delta_x) \| \mathcal{A}(\mathbf{x}+\delta x)\mathbf{u} + \frac{1}{2}(\mathbf{b}_2(\mathbf{x}+\delta x) - \mathbf{b}_1(\mathbf{x}+\delta x)) \|_2^2 .$$

Typically, one uses gaussian weights. This method is often referred to as the *Farneback method*, and originates in []. Example flow estimates appear in Figure 33.11. As Figure **??** shows, this method is susceptible to motion blur (as are other methods).

Works for small flows



But doesn’t like motion blur…



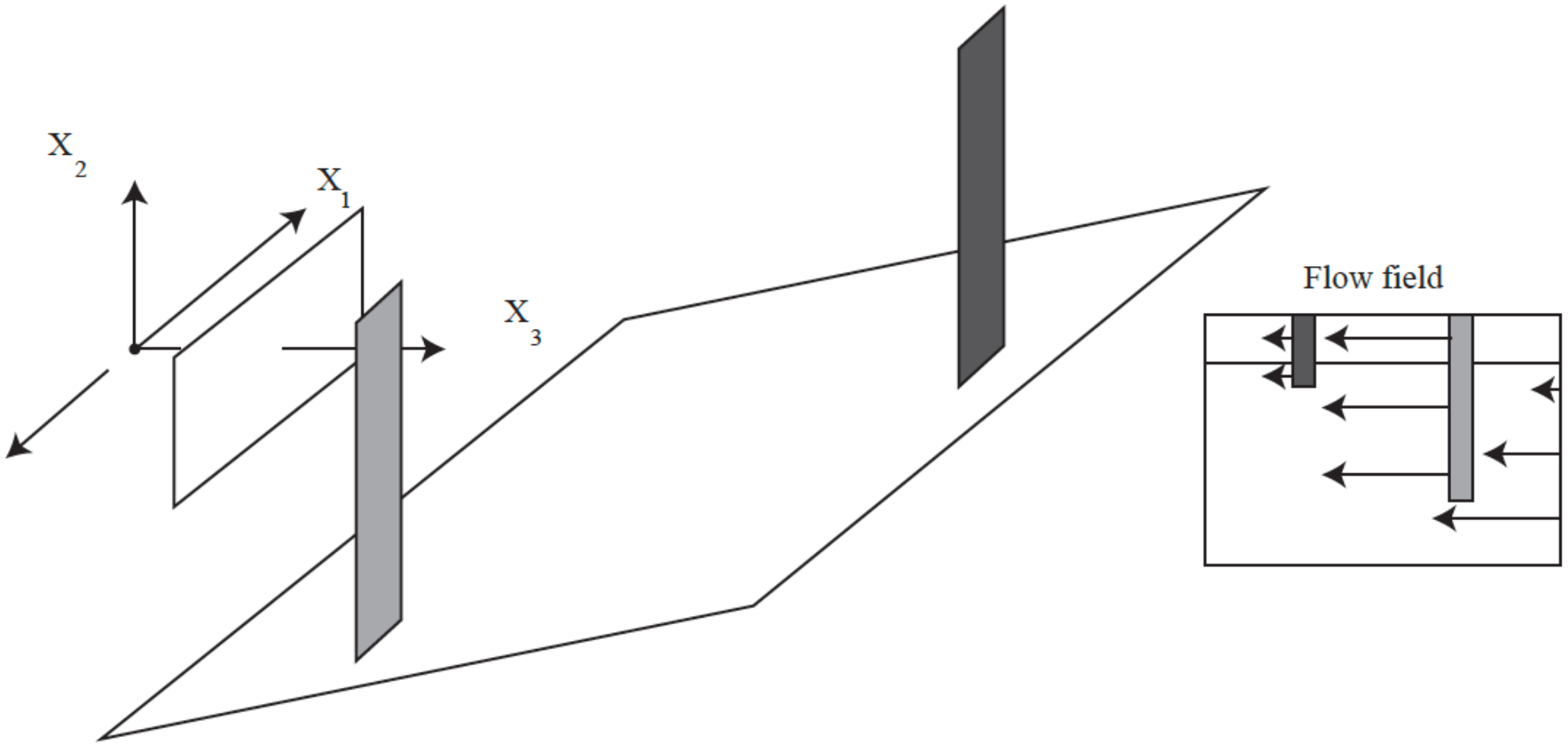
Potential improvements

- Robust versions of the cost function
  - replace squared intensity error with something better
- Iterative re-estimation of flow

$$I(x+u^{n+1}, y+v^{n+1}, t+\delta t) \approx I(x+u^{n}, y+v^{n}, t) + \frac{\partial I}{\partial x}\delta u + \frac{\partial I}{\partial y}\delta v + \frac{\partial I}{\partial t}\delta t.$$

  - note change in derivative estimate locations
- Flow estimates at multiple scales
  - 1 pixel flow in low res image == many pixels in hi-res
  - start at coarsest scale (N), zero flow
  - iterate
    - estimate flow at scale s using warped image 1
      - (original for N)
    - upsample flow estimate to s-1
    - warp image 1 at s-1
  - key idea: each flow is quite small, so…

Parametric flow models



Parametric flow models

- Apply the pinhole camera

Multiple objects on a plane: A calibrated camera translates in the x direction at a fixed height above a ground plane. The ground plane is at $Y = c$ in camera coordinates, so the image plane is perpendicular to the ground plane. A collection of N flat objects stick up out of the ground plane at fixed depths (Figure 33.4; the i'th object is at $Z = d_i$). For this setup, the flow in the y direction in the image plane is always zero. Check that, if the pixel at $(x, y)^T$ sees the ground plane, the z coordinate is c/y. The x component of flow at a pixel takes one of $N + 1$ values

$$\begin{cases} -\frac{t_x c}{y} & \text{if the pixel sees the ground plane} \\ -\frac{t_x}{c_1} & \text{if the pixel sees object 1} \\ \dots & \dots \\ -\frac{t_x}{c_N} & \text{if the pixel sees object N} \end{cases}$$

Affine flows

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \mathcal{M} \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{bmatrix}$$

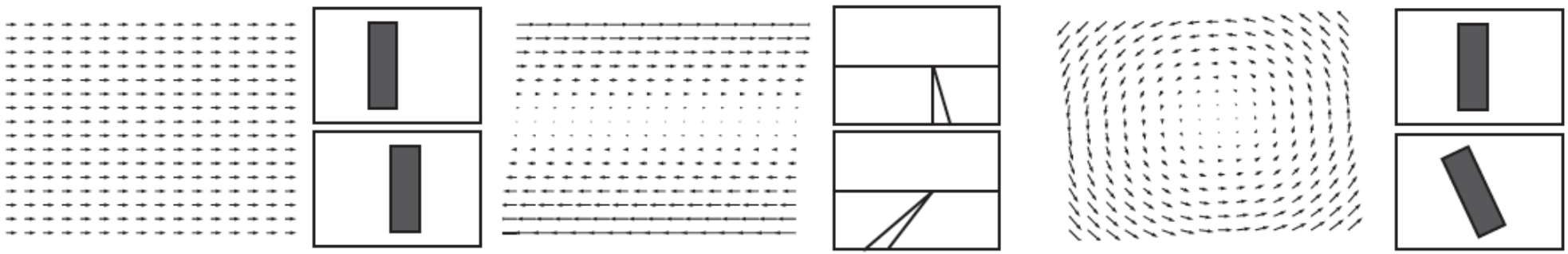


FIGURE 33.5: *A straightforward affine flow model in Section* **??** *encodes many useful cases. On the* **left**, *constant flow (the first column of* $\mathcal{M}$ *gives the flow direction, all others are zeros). In the* **center**, *the flow viewed from a train window in Figure 33.3 (here the first three columns of* $\mathcal{M}$ *are non-zero, others are zeros). On the* **right**, *an object rotates (again, the first three columns of* $\mathcal{M}$ *are non-zero, others are zeros).* **Top** *image frame is frame 1,* **bottom** *is frame 2. Details in* **exercises** .

Affine flows - II

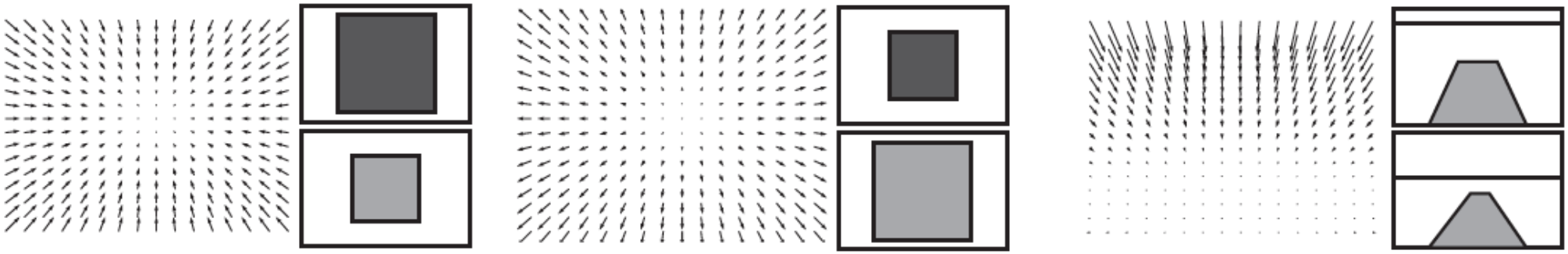


FIGURE 33.6: *A straightforward affine flow model in Section* **??** *encodes many useful cases. On the* **left**, *the viewer leaves an object; in the* **center**, *a viewer is heading towards an object; and on the* **right**, *the plane on which the rectangle sits is tilted backwards (or, equivalently, the viewer tilts backwards).* **Top** *image frame is frame 1,* **bottom** *is frame 2. Details of the relevant* $\mathcal{M}$ *for each case in* **exercises** .

Simplest estimation

Write $\mathbf{t}(\theta) = (u(x, y; \theta), v(x, y; \theta))^T$ for a parametric flow field, where θ are the parameters of the field. To obtain a flow field, you must choose some function $D(\delta I)$ for the data term to penalize the photometric inconsistency, another function – which you might even omit – $P(\theta)$ which penalizes the flow for being unrealistic. You then minimize

$$C(\theta) = D(I_2(x + u(x, y; \theta), y + v(x, y; \theta)) - I_1(x, y))) + \alpha P(\theta)$$

as a function of θ. In the simplest case, $C(\delta I) = (\delta I)^2$, you set $\alpha = 0$, and you use the linearized photometric inconsistency. If the parametric flow model is linear in the parameters, you will need to solve a linear system (**exercises**). The

Estimation using Farneback

- Previously :

$$\mathbf{t}(\mathbf{x}) = \underset{\mathbf{u}}{\operatorname{argmin}} \sum_{\delta x \in \text{window}} w(\delta_x) \| \mathcal{A}(\mathbf{x}+\delta x)\mathbf{u} + \frac{1}{2}(\mathbf{b}_2(\mathbf{x}+\delta x) - \mathbf{b}_1(\mathbf{x}+\delta x)) \|_2^2 .$$

- Now u is a parametric flow model

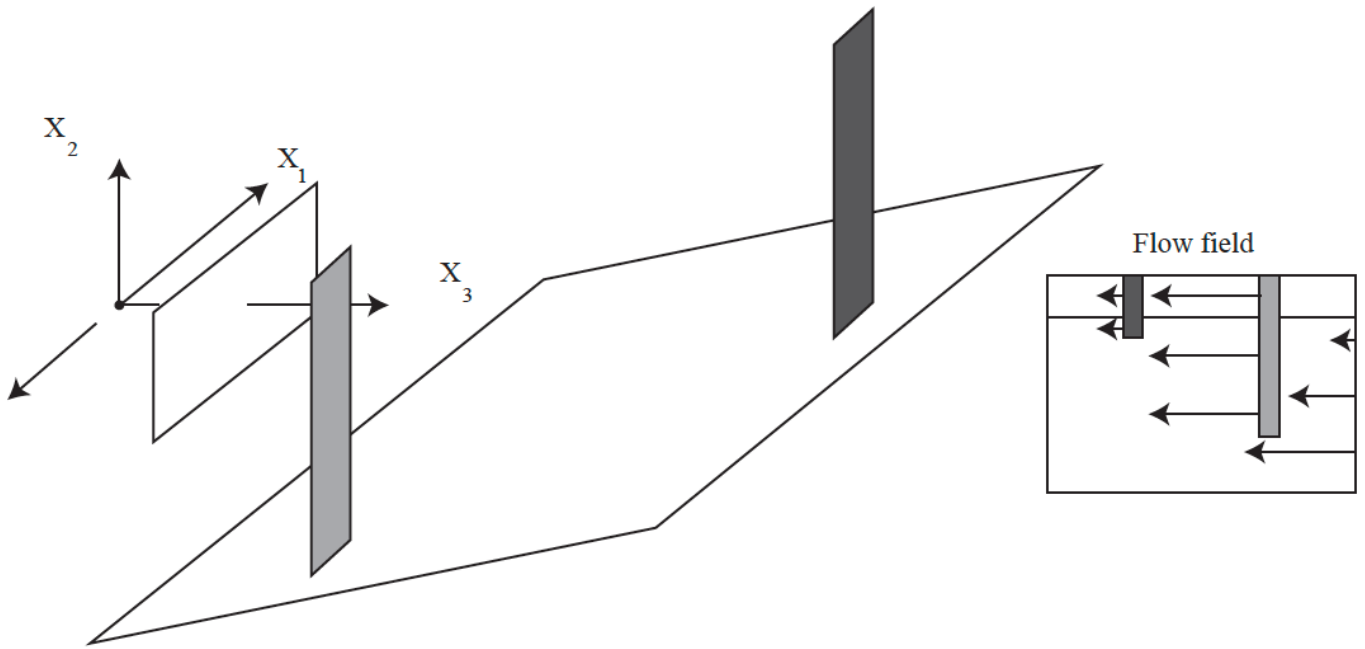
$$\theta = \underset{\phi}{\operatorname{argmin}} \sum_{\delta x \in \text{window}} w(\delta_x) \| \mathcal{A}(\mathbf{x}+\delta x)\mathcal{M}\phi + \frac{1}{2}(\mathbf{b}_2(\mathbf{x}+\delta x) - \mathbf{b}_1(\mathbf{x}+\delta x)) \|_2^2 .$$

Potential improvements

- Robust versions of the cost function
  - replace squared intensity error with something better
- Iterative re-estimation of flow
  - note change in derivative estimate locations
- Smooth estimates
  - or use overlapping windows
- Segment image using flow

Layered motion

- One model may not work for whole image
  - eg the train viewing trees on ground plane
- Decompose image into regions
  - all pixels that share a parametric flow model
  - eg tree 1, tree 2, ground plane



Re-estimation

- Given set of parametric flow models
  - you can segment
  - using photometric loss
    - eg at pixel, move along the flow from each model
      - choose model where next intensity is closest to start intensity
- Given a segmentation
  - you can estimate parametric flow models
    - use all pixels in the segment to estimate the models

Re-estimation

- This is k-means
  - (in fairly heavy disguise)
  - soft allocation is useful (text)
- Can phrase as estimating a mixture of Gaussians

Sequence

Reconstruction without tree layer

Layers

Segmentation