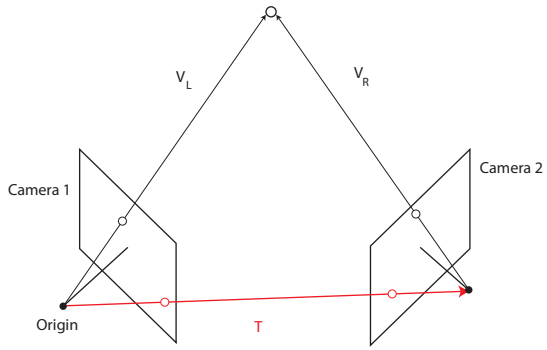
Two and more cameras: Stereo, Optic Flow and Structure from Motion

D.A. Forsyth, UIUC

Stereopsis

- Generically:
  - recover depth map from two images of scene
    - cameras may be calibrated/uncalibrated
      - may have large/small baseline
      - if uncalibrated, recover from fundamental matrix, above
  - do so by
    - finding correspondences
    - constructing depth map using correspondences
- Huge literature, with multiple important tricks, etc.
  - I'll mention a small set



Camera translation

$$\mathbf{V}_R = \mathcal{R}(\mathbf{V}_L - \mathbf{T})$$

Camera rotation

$$\mathcal{S} = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{pmatrix}$$

$$\mathbf{T}^T \mathcal{S} = \mathbf{0}$$

What we have

- 3D points: $\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ And $\begin{pmatrix} x_1^t \\ x_2^t \\ x_3^t \end{pmatrix} = \mathcal{R}\left[\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} - \mathbf{t}\right]$

Original point in camera two's coordinate system

- normalized image points:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1/x_3 \\ x_2/x_3 \end{pmatrix} \qquad \begin{pmatrix} y_1^t \\ y_2^t \end{pmatrix} = \begin{pmatrix} x_1^t/x_3^t \\ x_2^t/x_3^t \end{pmatrix}$$

Recovering the point in 3D

- Write

$$\mathcal{R} = \begin{pmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{pmatrix}$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix}$$

- Then

$$x_3 = \frac{(\mathbf{r}_1 - y_1^t \mathbf{r}_3)^T \mathbf{t}}{(\mathbf{r}_1 - y_2^t \mathbf{r}_3)^T \mathbf{y}}$$

And we have everything in 3D!

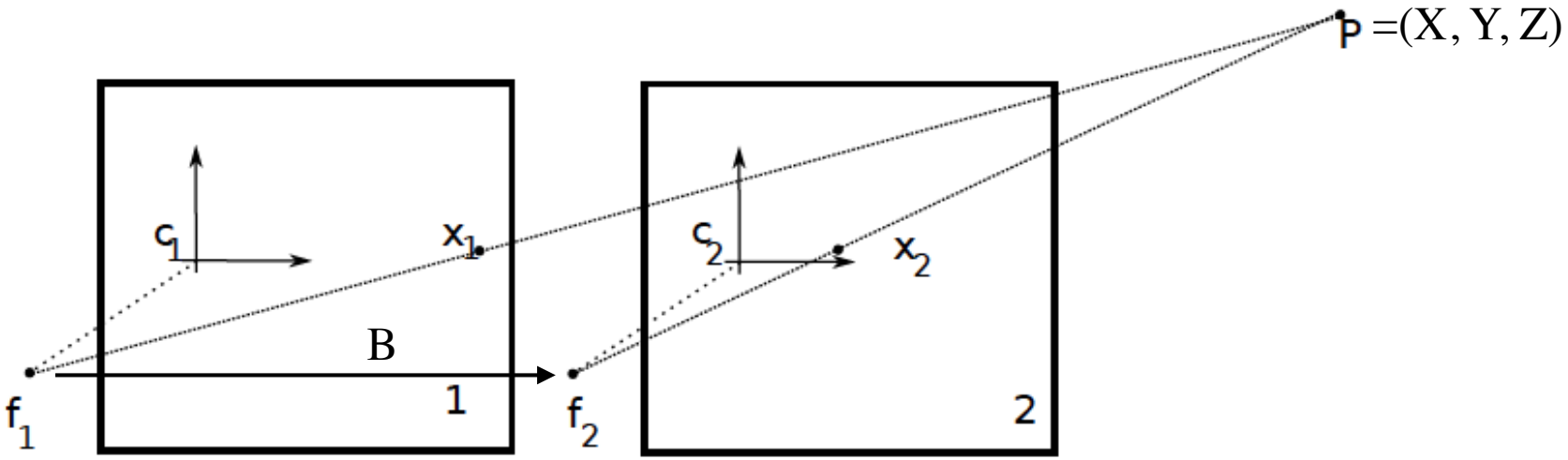
The effect of scale

$$x_3 = \frac{(\mathbf{r}_1 - y_1^t \mathbf{r}_3)^T \mathbf{t}}{(\mathbf{r}_1 - y_2^t \mathbf{r}_3)^T \mathbf{y}}$$

- If we know R, t, we can reconstruct point in 3D
  - IF we can match points across more than two cameras

Disparity

Assume camera 2 is translated wrt camera 1, both are calibrated



Disparity

$$d = x_2 - x_1 = f\frac{(X - B) - X}{Z} = -f\frac{B}{Z}$$

Baseline

Depth

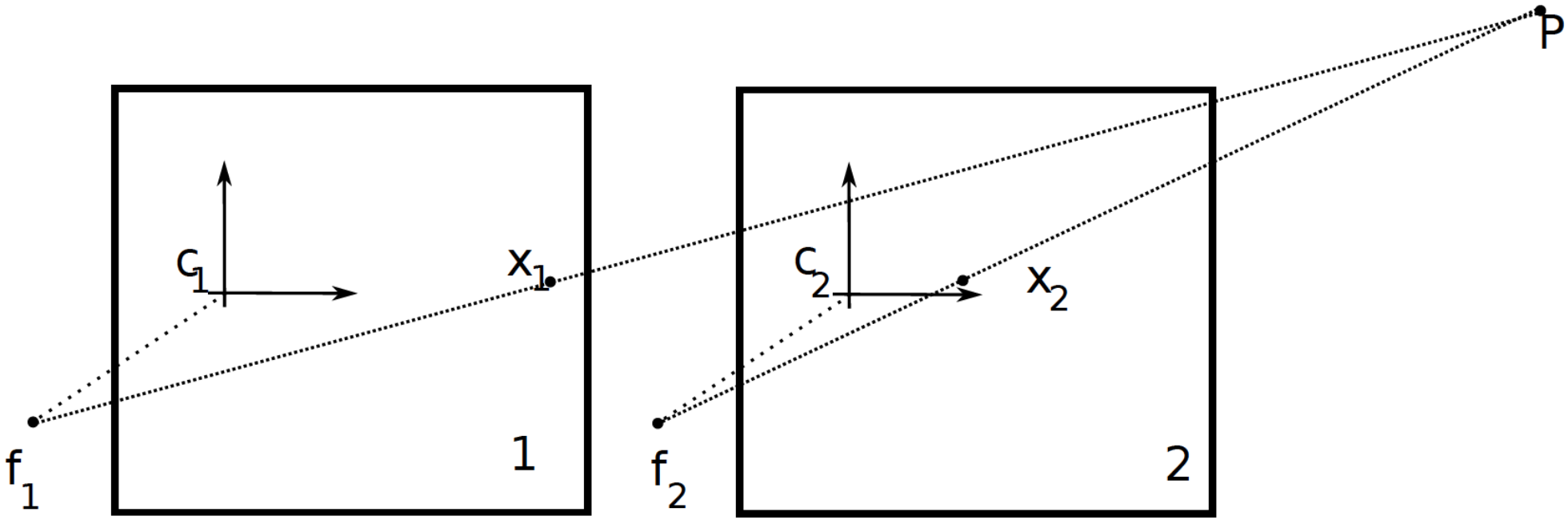
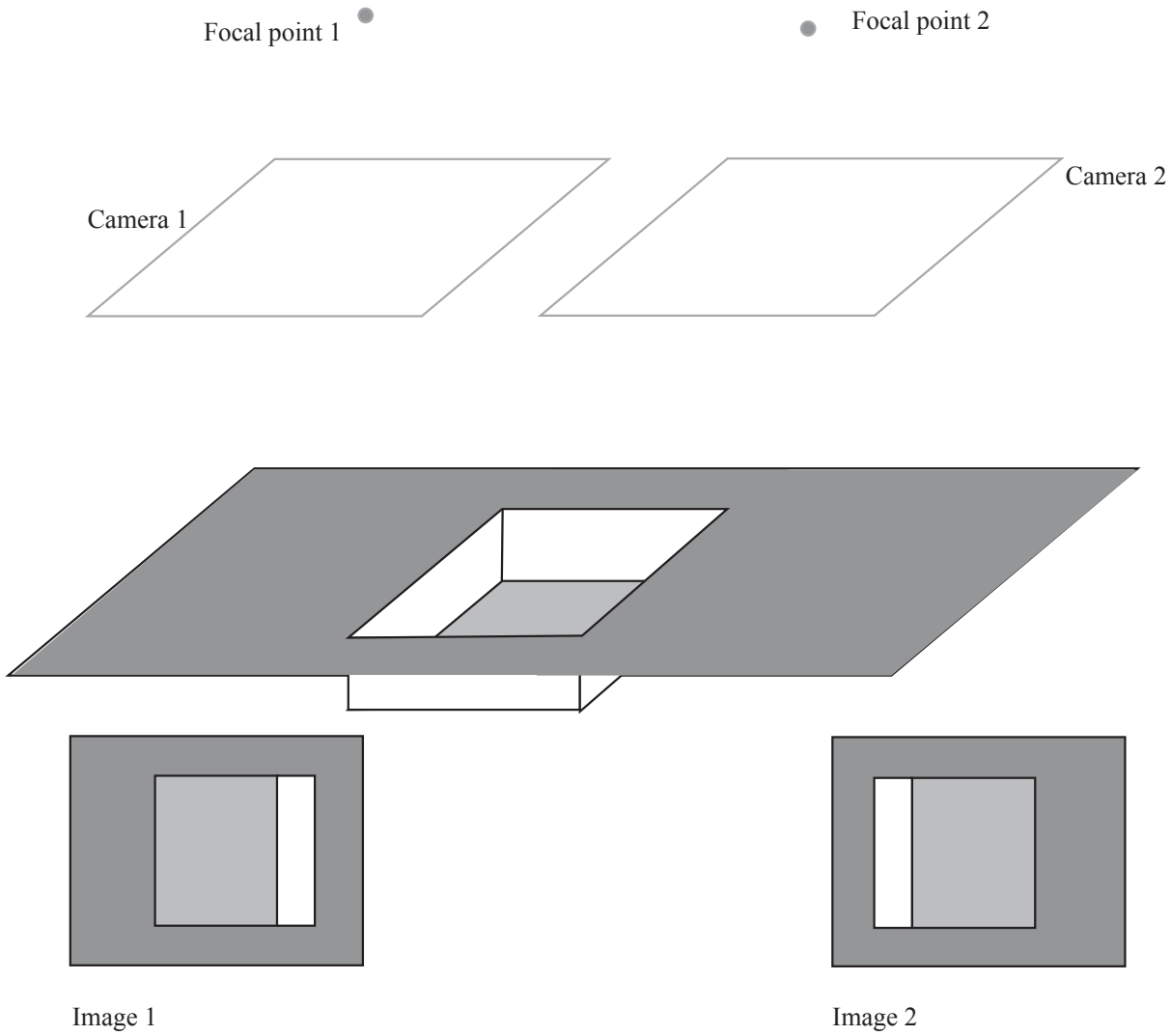


FIGURE 2.11: *When two pinhole cameras view a point, the 3D coordinates of the point can be reconstructed from the two images of that point. This applies for almost every configurations of the cameras. It is an elementary exercise in trigonometry (exercises) to determine* $\mathbf{P}$ *from the positions of the two focal points, the locations of the point in the two images, and the distance between the focal points. Considerable work can be required to find appropriate matching points, but the procedures required are now extremely well understood (Chapters 36.2). One can now buy camera systems that use this approach to report 3D point locations (often known as RGBD cameras). Here we show a specialized camera geometry, chosen to simplify notation. The second camera is translated with respect to the first, along a direction parallel to the image plane. The second camera is a copy of the first camera, so the image planes are parallel. In this geometry, the point being viewed shifts somewhat to the left in the right camera.*

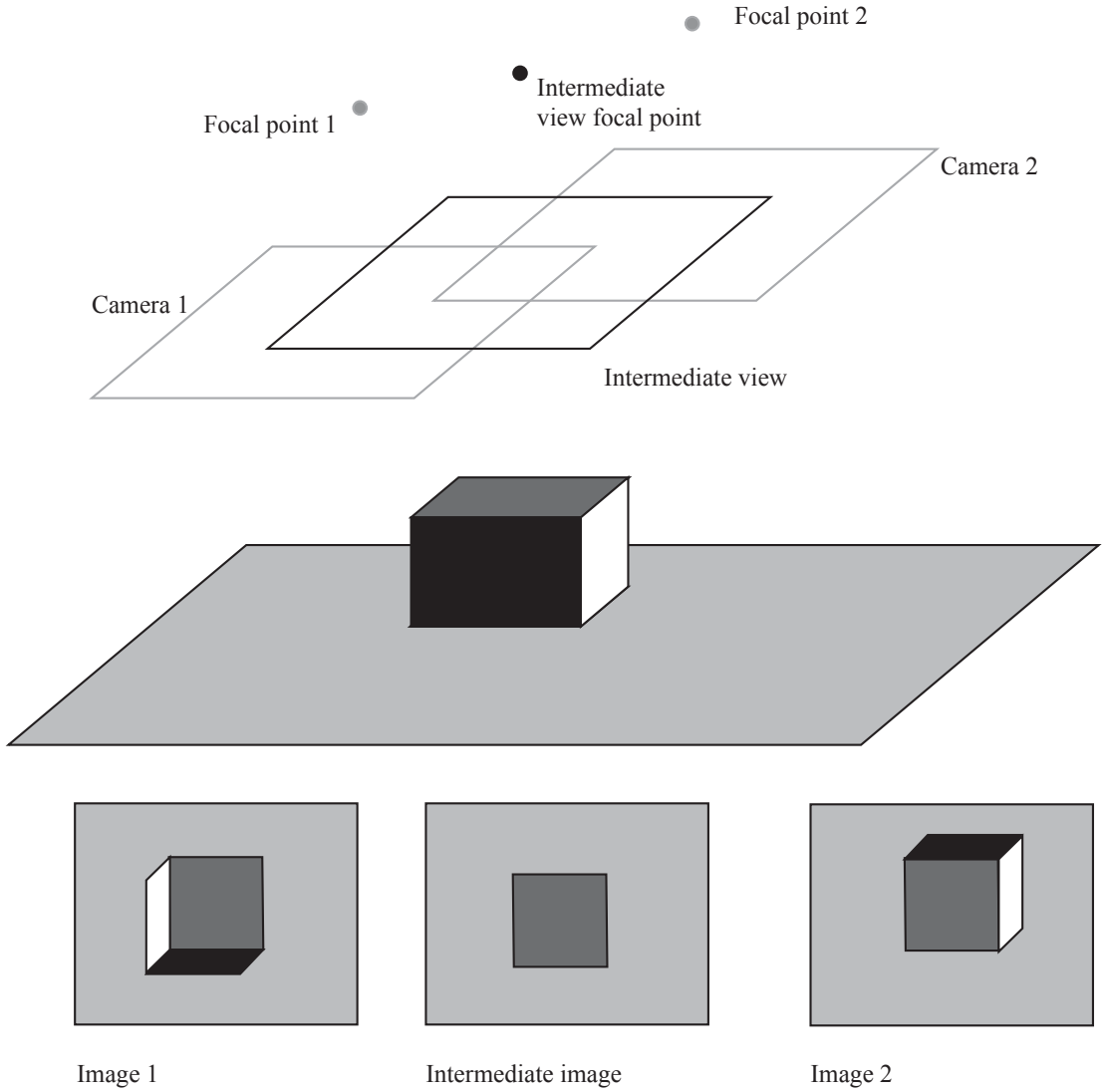
Pragmatics

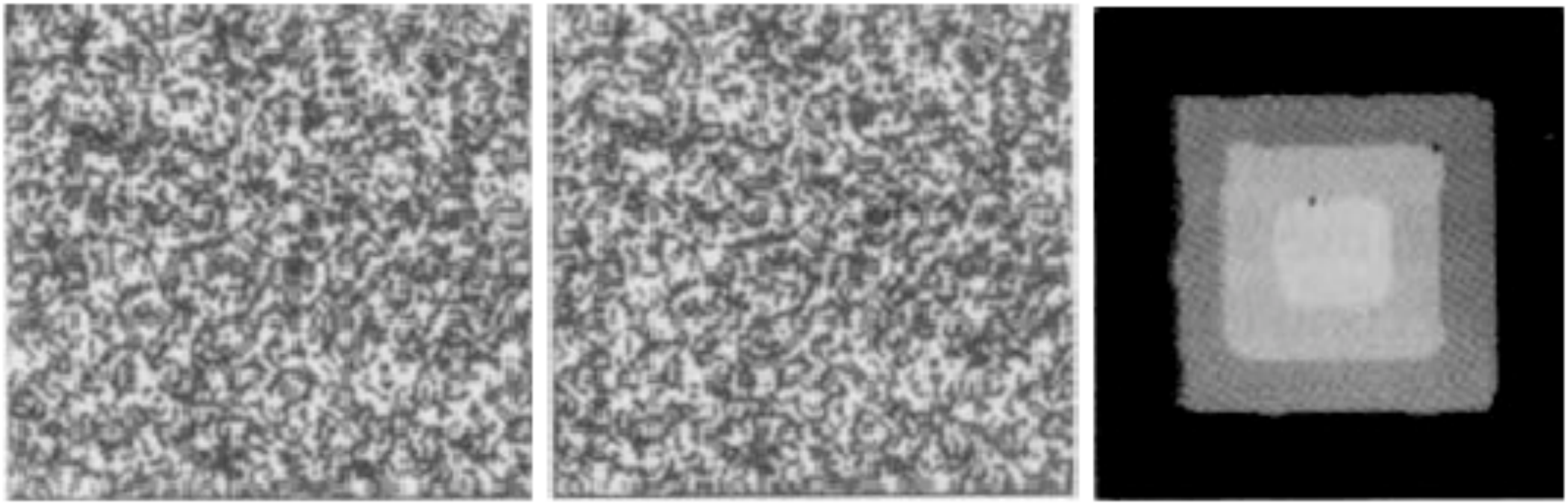
- Issue
  - Match points
- Strategy
  - correspondences occur only along relevant epipolar line
  - represent points with local representation
    - color + surface smoothness; filters; some kind of deep feature
- Issue
  - some points don’t have correspondences (occlusion)
- Match left to right, then right to left
  - if they don’t agree, break match

Some points don’t have matches

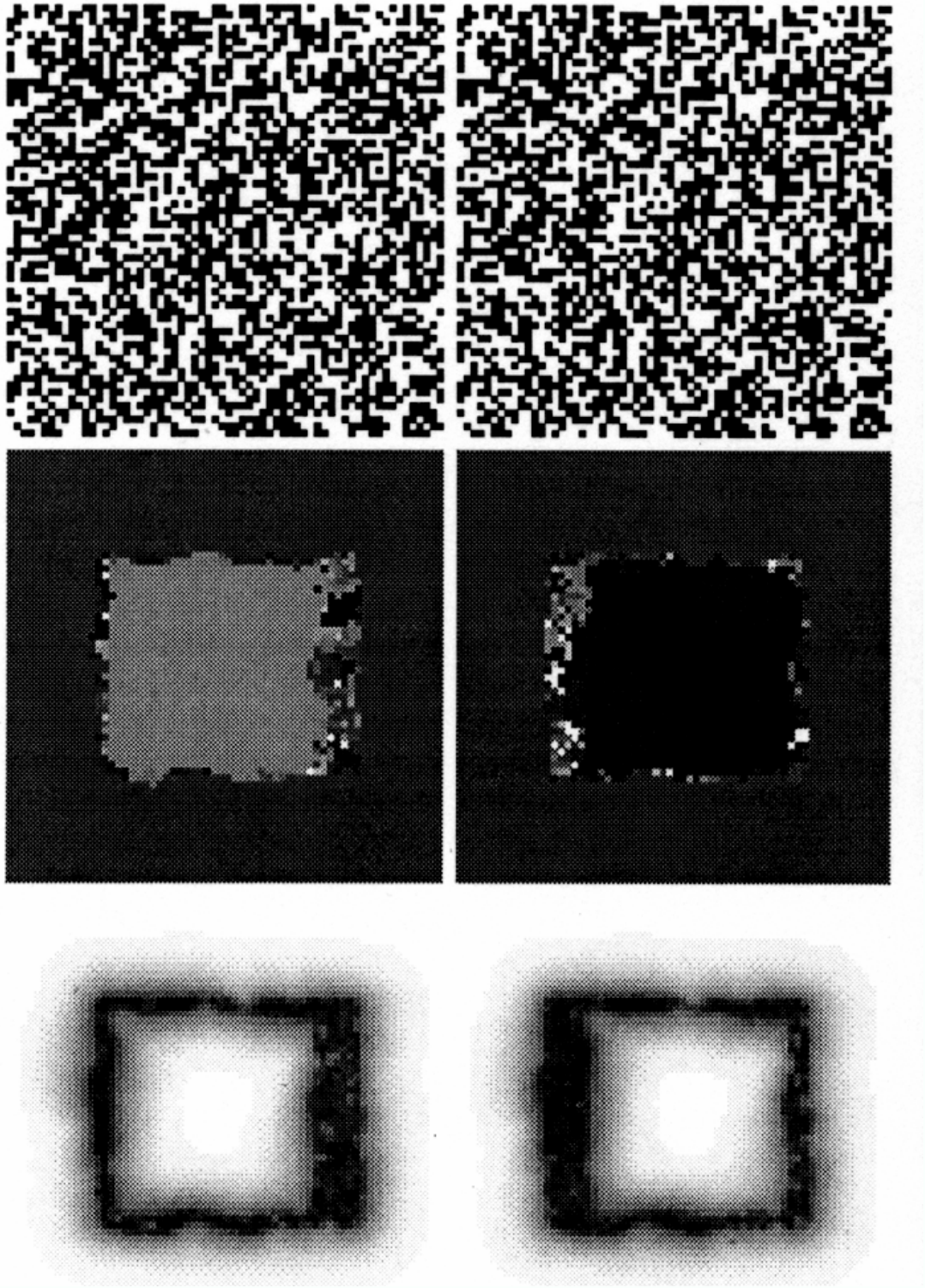


Some points don’t have matches

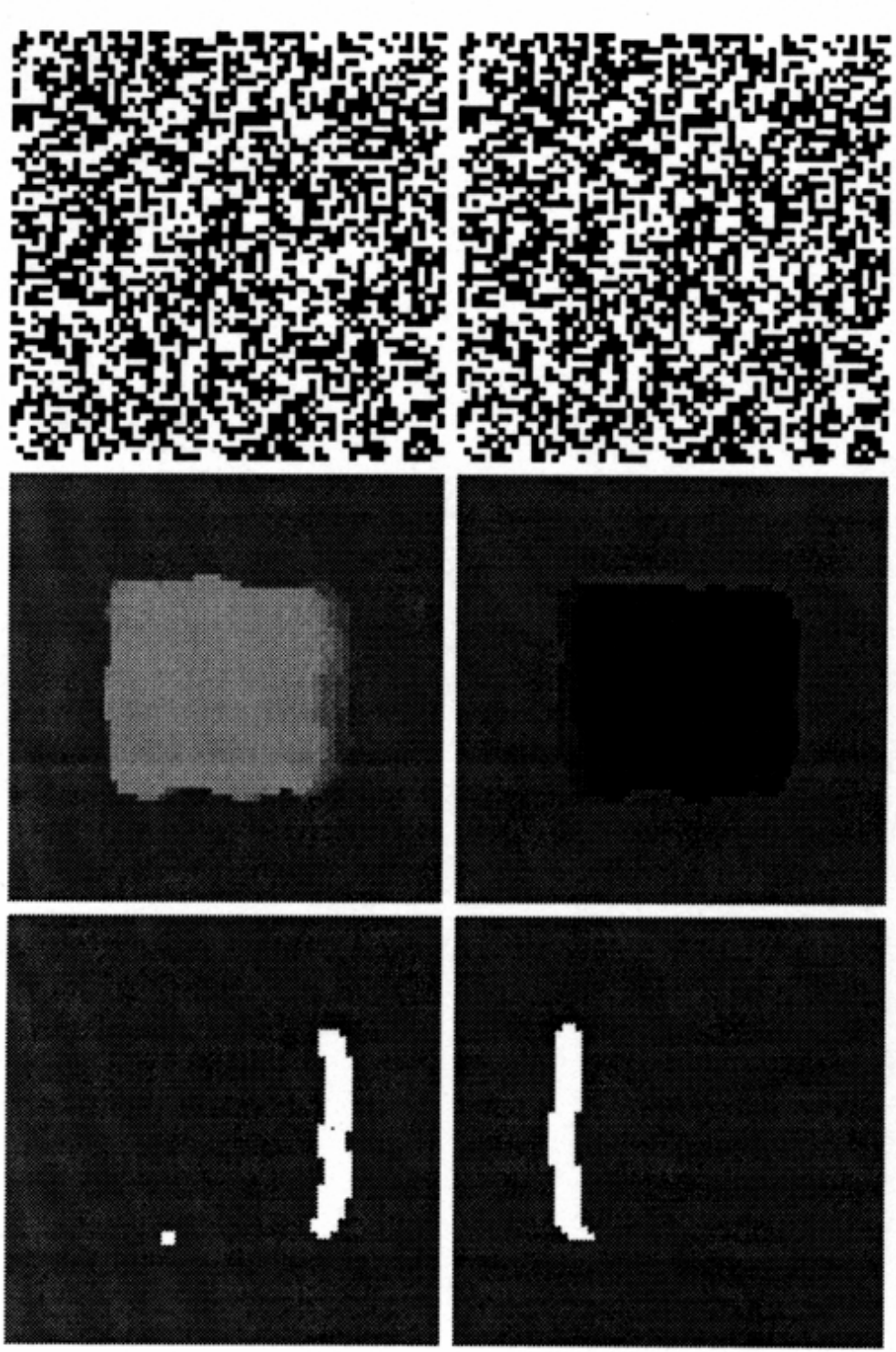




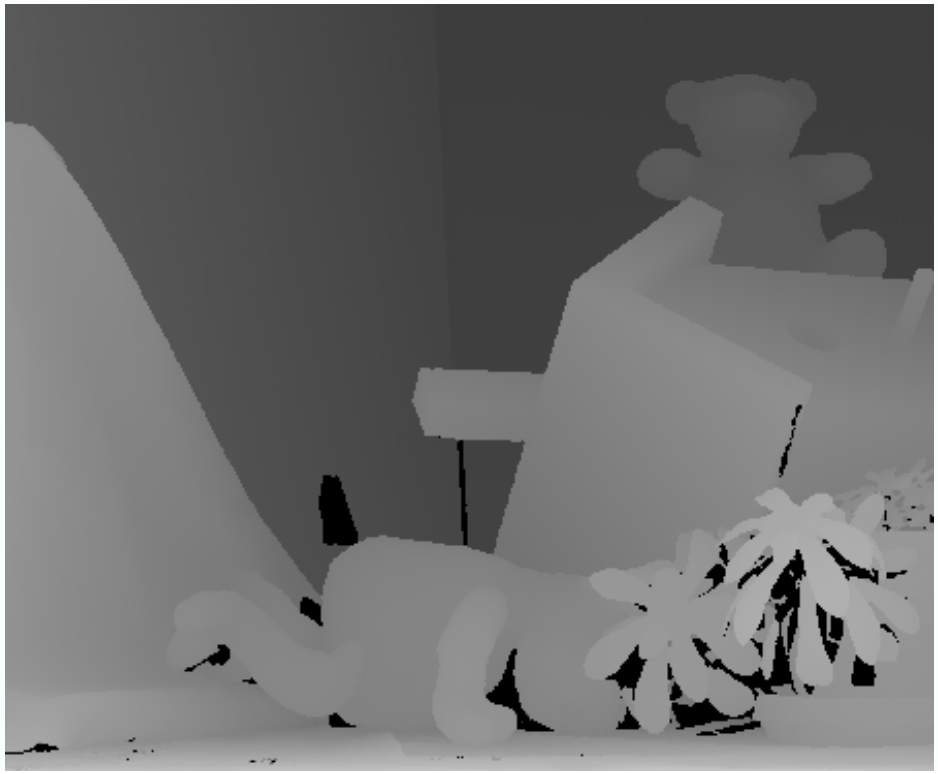
From Jones and Malik, “A computational framework for determining
Stereo correspondences from a set of linear spatial filters

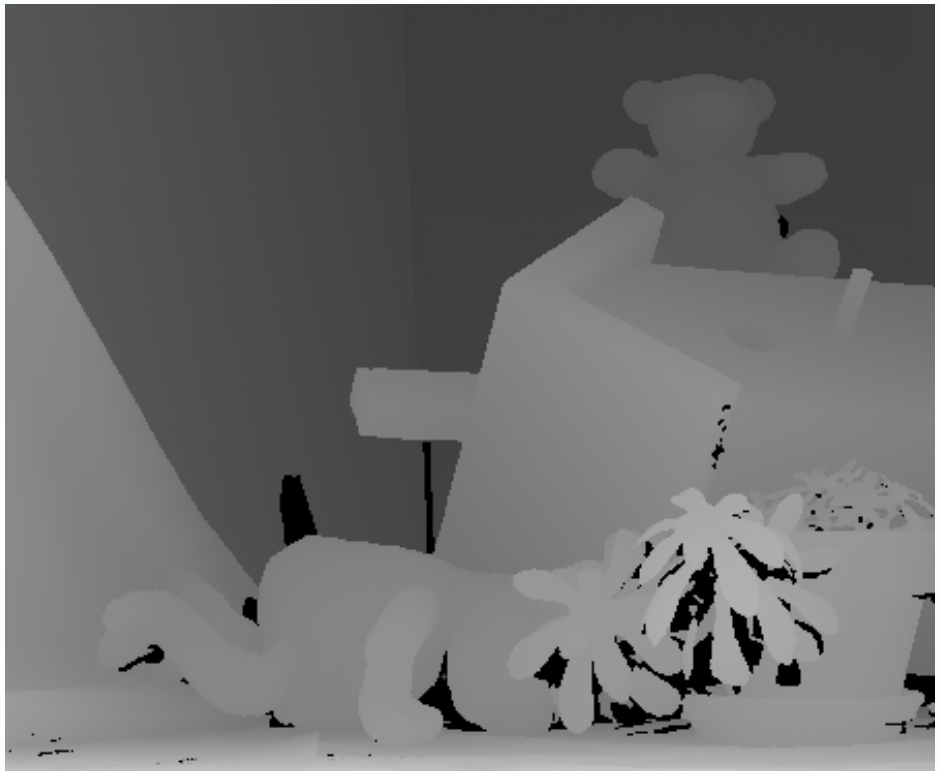


From Jones and Malik, “A computational framework for determining
Stereo correspondences from a set of linear spatial filters

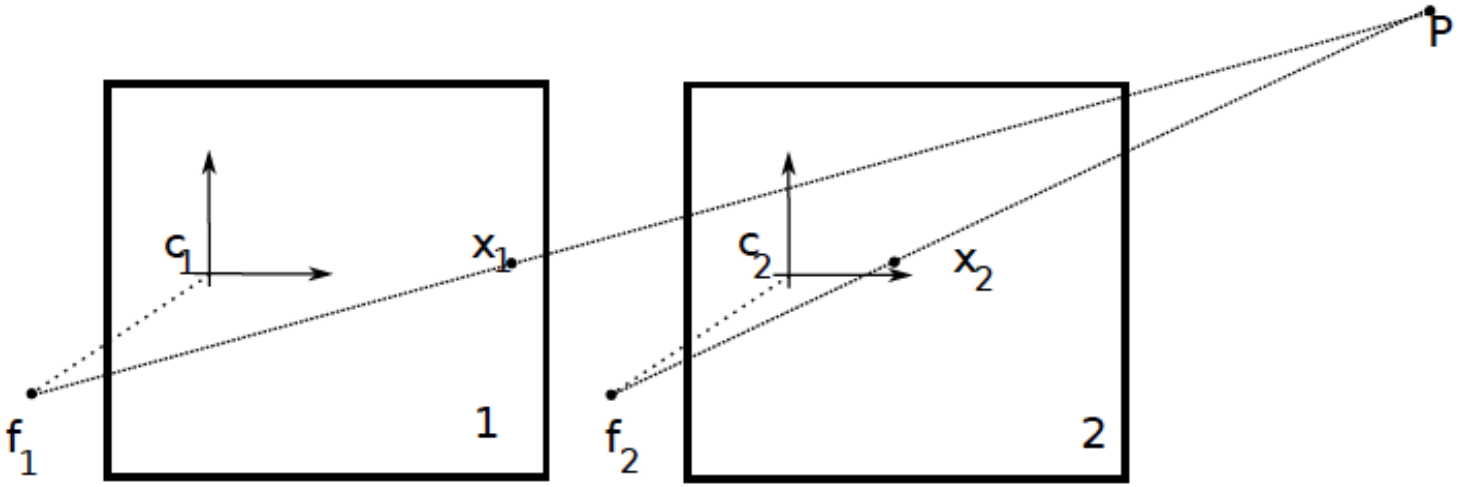


From Jones and Malik, "A computational framework for determining
Stereo correspondences from a set of linear spatial filters





Stereo as an optimization problem

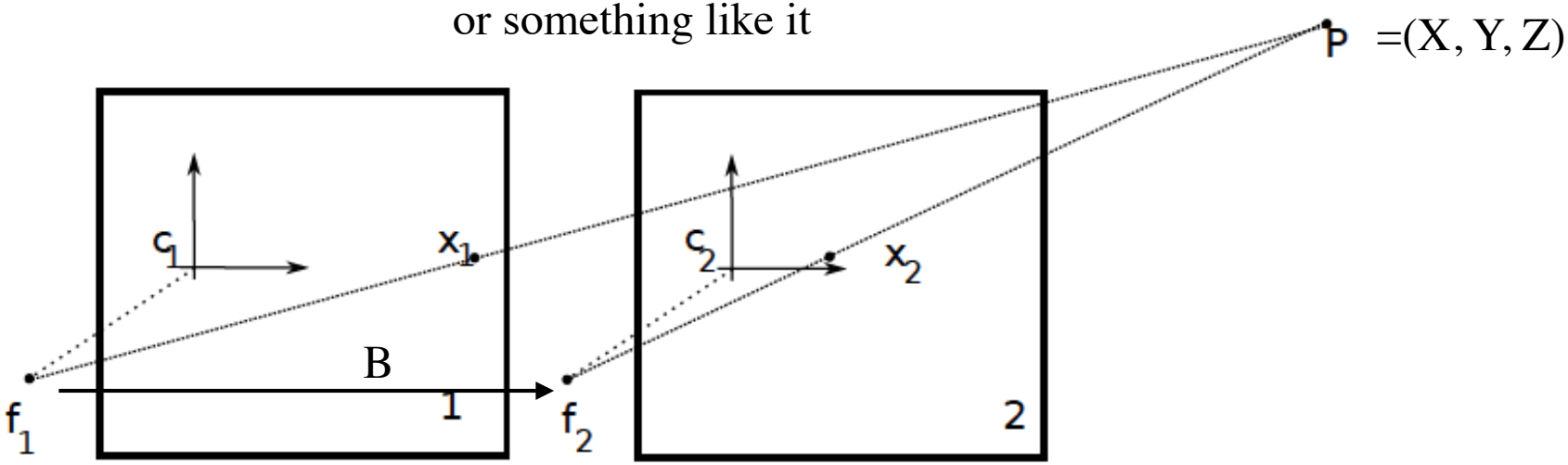


- Original:
  - find q, q' that match, and infer depth
- Now:
  - choose value of depth at q; then quality of match at q' is cost
  - optimize this

Stereo as an optimization problem

Assume camera 2 is translated wrt camera 1, both are calibrated

IF you choose the right depth for x_1, then:
you know disparity, so you know x_2
and you can optimize
||color(x_1)-color(x_2(d))||^2
or something like it



Disparity

$$d = x_2 - x_1 = f\frac{(X - B) - X}{Z} = -f\frac{B}{Z}$$

Baseline

Depth

Stereo as an optimization problem

- Typically:
  - quantize depth to a fixed number of levels
  - two costs:
    - compare x_1, x_2(d) — unary cost
      - color match (photometric consistency constraint)
      - it can be helpful to match intensity gradients, too
    - compare x_1, x_1'
      - typically, smoothness constraint on recovered depths
      - eg depth gradient not too big, etc.
  - massive discrete quadratic program

Discrete Quadratic Programs

- Minimize:
  - x^T A x + b^t x
  - subject to: x is a vector of discrete values
- Summary:
  - turn up rather often in early vision
    - from Markov random fields; conditional random fields; etc.
  - variety of cases:
    - some instances are polynomial
    - most are NP hard
      - but have extremely efficient, fast approximation algorithms
      - typically based on graph cuts, qv

Stereo as an optimization problem (II)

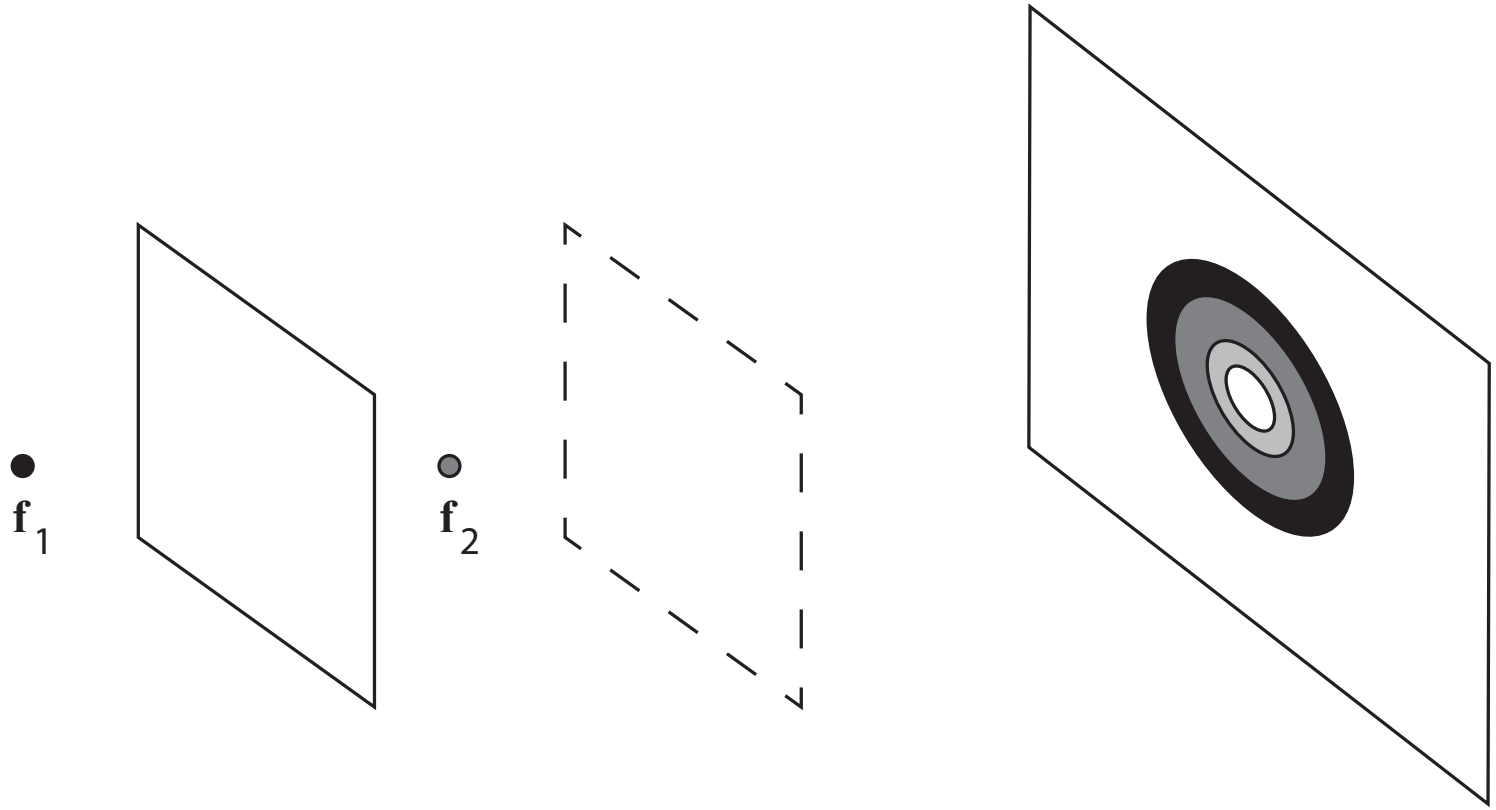
- Segment images into regions
  - NOT semantic; small, constant color+texture
- Each region is assumed to have a linear disparity
  - $d(x, y)=a\ x + b\ y+c$
- Find a quantized "vocabulary" of such disparities
  - eg by initial disparity, incremental fitting
- For each region, choose the "best" in the "vocabulary"
  - This is a discrete optimization problem
  - It's quadratic
    - unary term - does the chosen vocab item "agree" with color data?
    - binary term - are neighboring pairs of models "similar" on boundary?

Stereo resources

- Datasets and evaluations:
  - Middlebury stereo page has longstanding
    - datasets
    - evaluations with leaderboards
    - datasets with groundtruth
    - refs to other such collections
      - (but this is the best known, by a long way)
  - https://vision.middlebury.edu/stereo/

Optic flow

- Generically:
  - a “small” camera movement yields image 2 from image 1
  - determine where points in image 1 move
- Assume we’re moving rigidly in a stationary environment
  - then points will move along their epipolar lines
    - where the epipolar lines follow from fundamental matrix
      - so from camera movement
- Main point of contrast with stereo
  - Images are not usually simultaneous
    - so objects might have moved



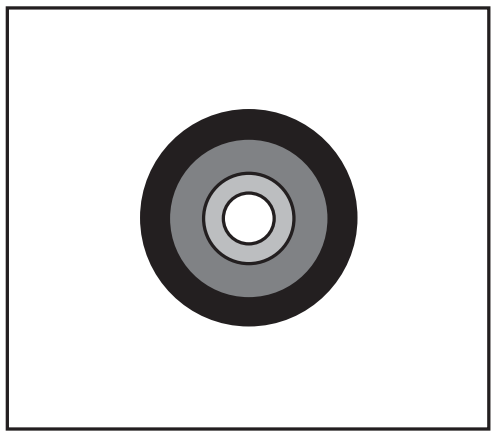


Image 1

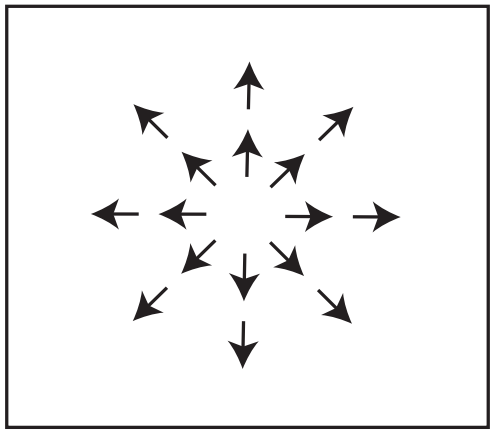


Image 1 optic flow

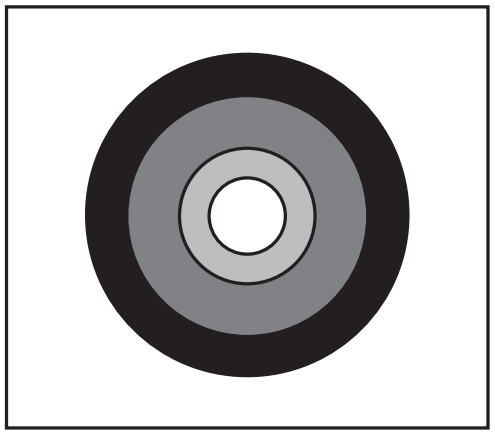
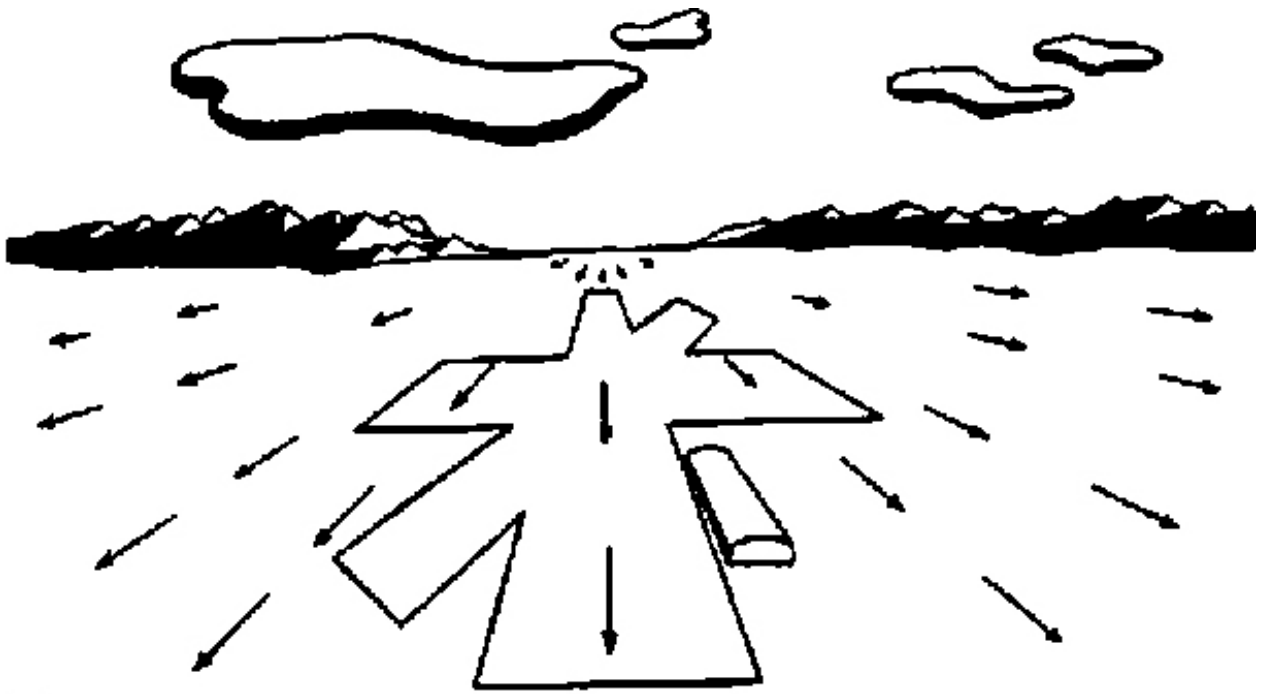
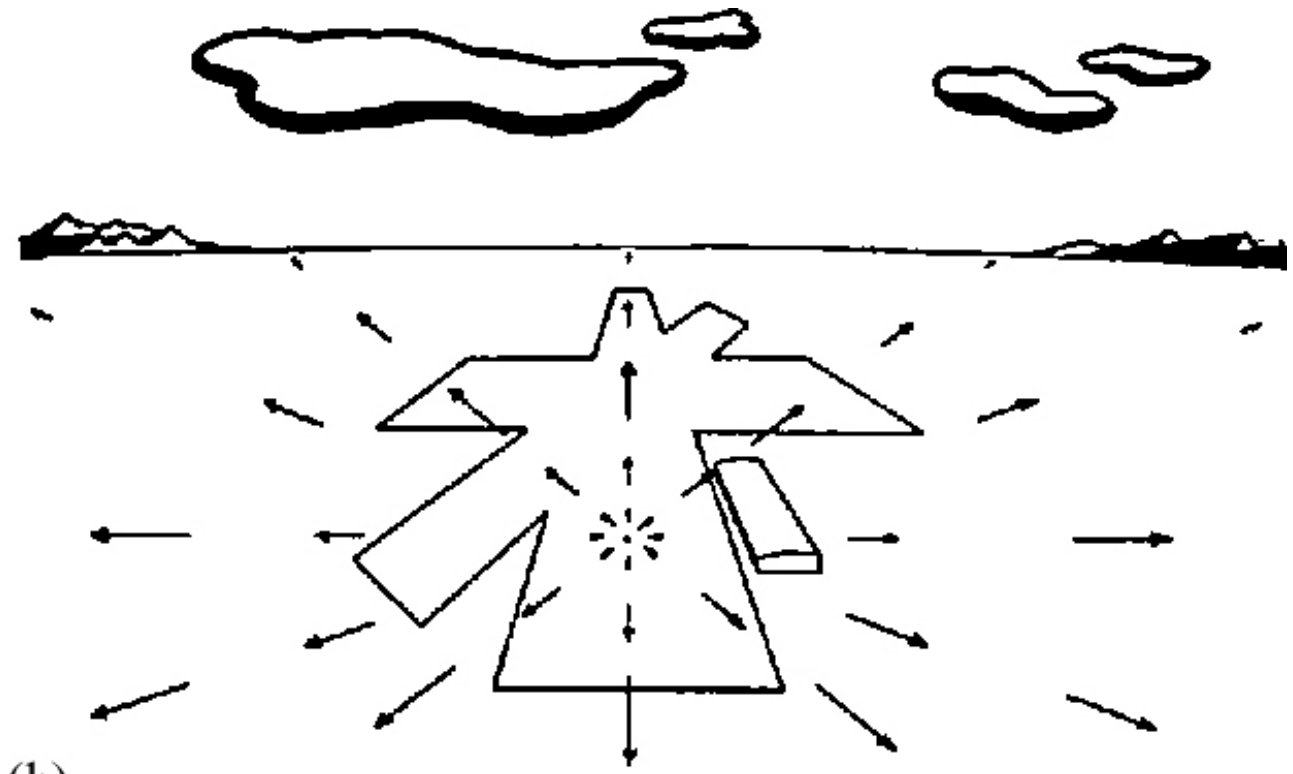


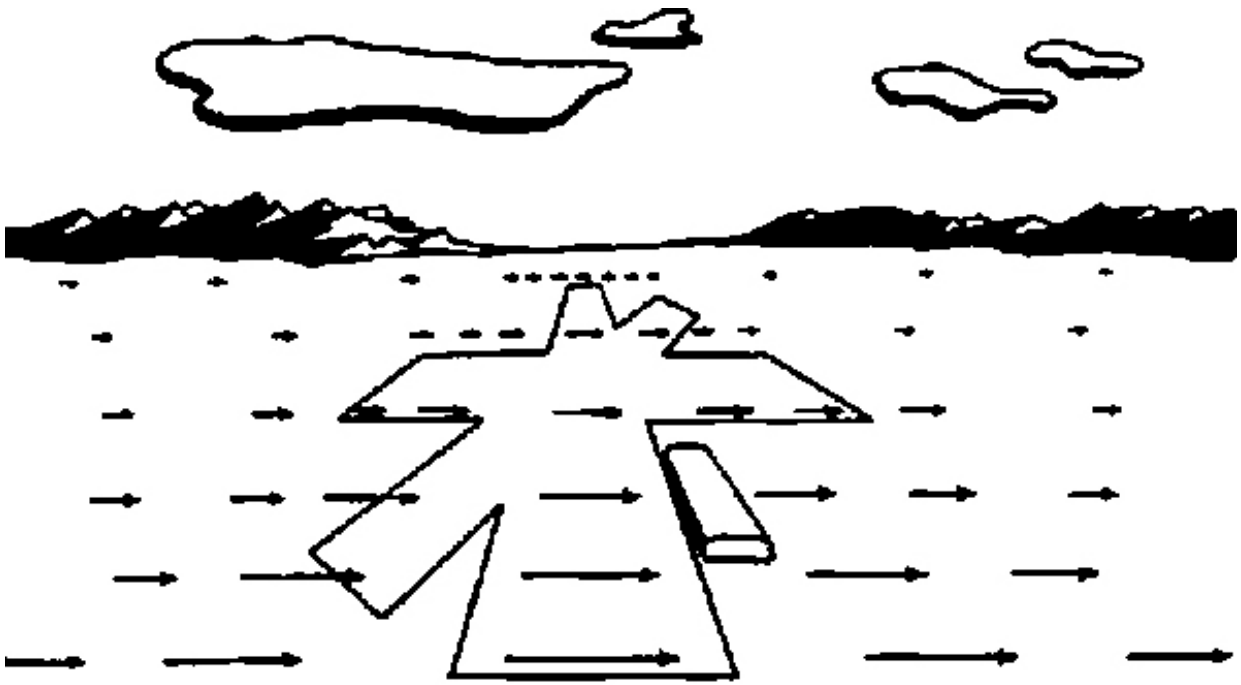
Image 2



(a)

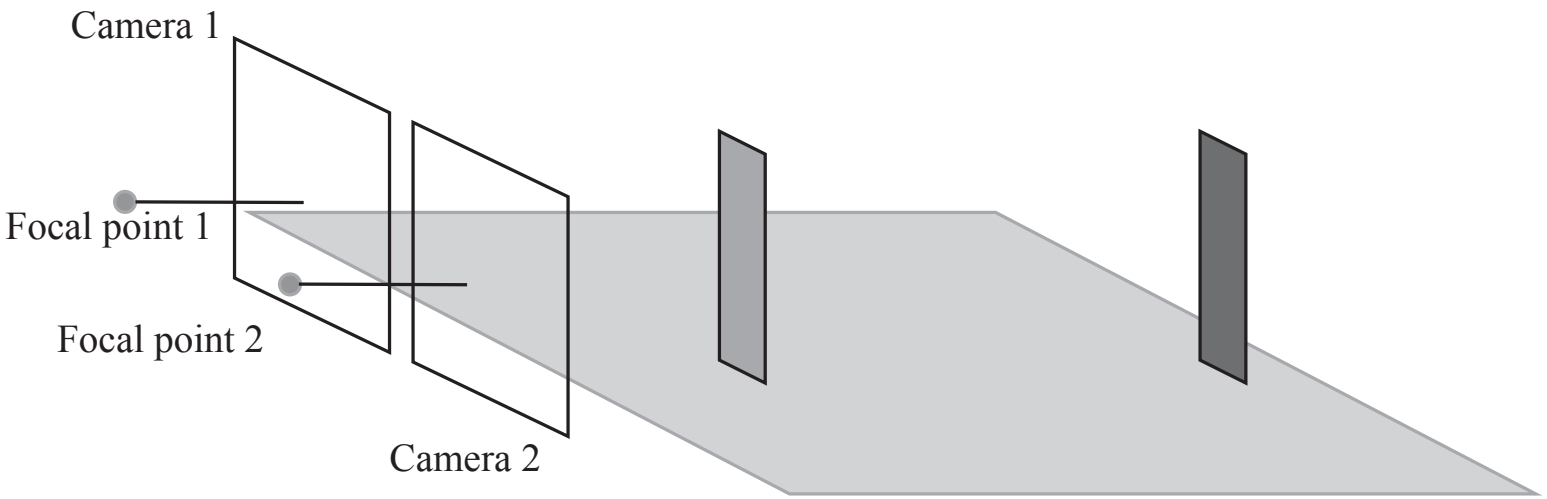


(b)



Optical flow

- Generically:
  - a “small” camera movement yields image 2 from image 1
  - determine where points in image 1 move
- Assume we’re moving rigidly in a stationary environment
  - then points will move along their epipolar lines
    - where the epipolar lines follow from fundamental matrix
      - so from camera movement
- As we saw, HOW FAR they move is determined by depth
  - and by their movement!!!



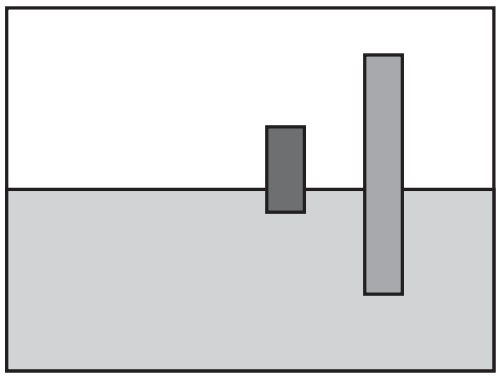


Image 1

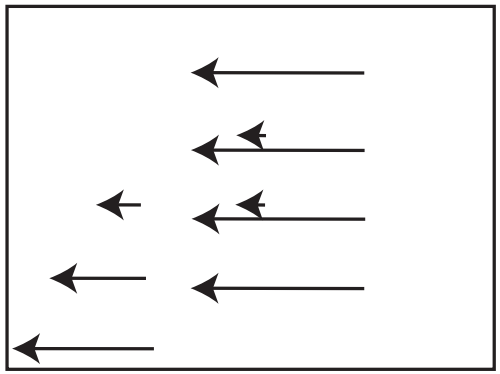


Image 1

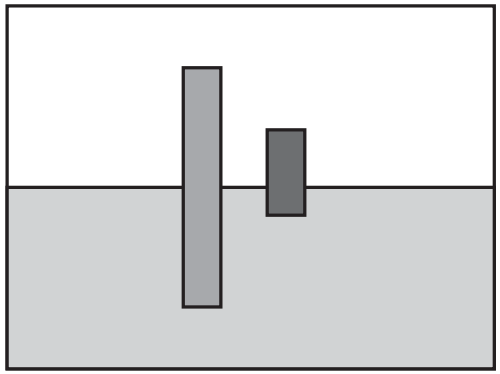


Image 2

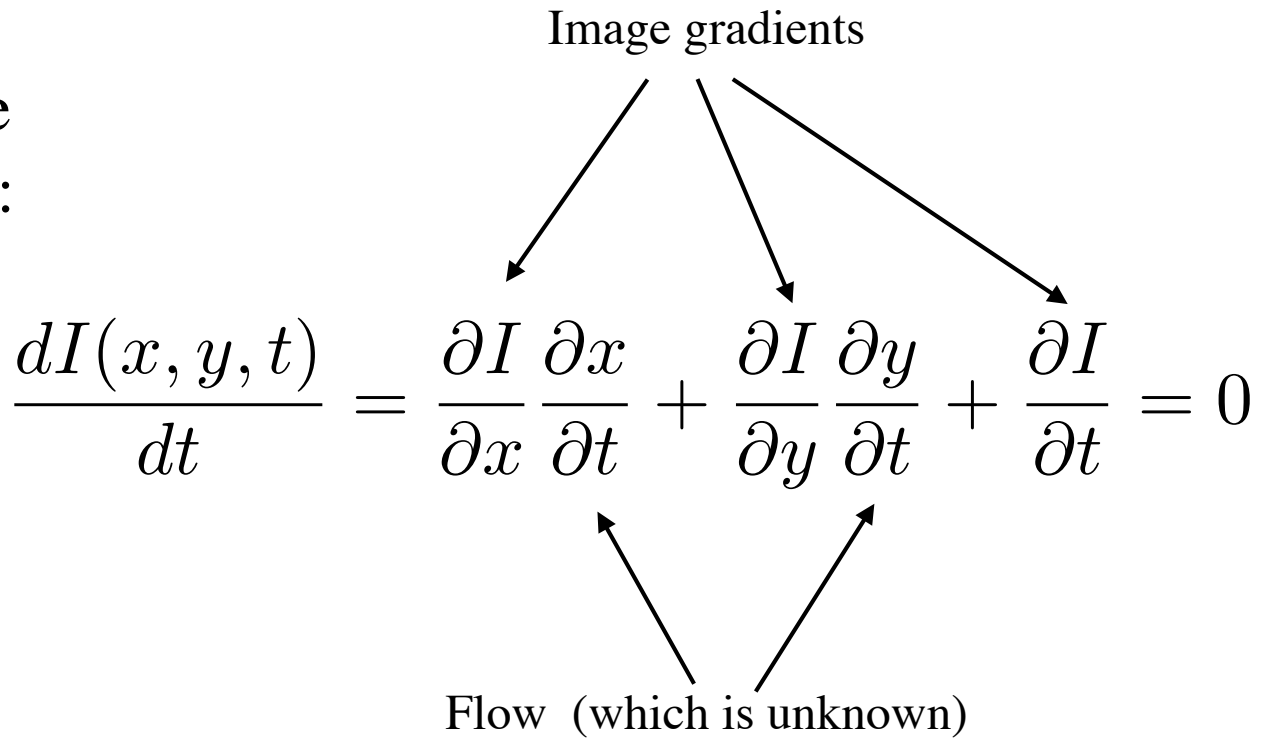

WOMEN IN MA
AND ENG
CORNELL U
dV
V

There is flow here!

For camera motions in a rigid scene, you can determine ground truth.
Evaluation is then by comparison to ground truth.

Recovering optic flow

- Huge literature
- Initial strategy:
  - Assume



$$I_x u + I_y v + I_t = 0$$

Recovering optic flow

$$I_x u + I_y v + I_t = 0$$

- Strategies:
  - find u(x, y), v(x, y) that minimizes some smoothness cost
    - subject to constraint on flow
    - what smoothness cost?
    - how to impose constraint?
  - assume flow has some parametric form within windows (eg. constant)
    - choose parameters to minimize error in window
    - what parametric model?
    - what windows?
  - If few or no objects move
    - impose a parametric depth model, and use that


Middlebury College
Computer Science
annual high school
computition day
Middlebury College
Computer Science
annual high school
computition day

If objects are moving, much harder to determine ground truth.

IDEA: Interpolate flow to get intermediate frame.

Evaluation is then by comparing interpolate to ground truth frame.

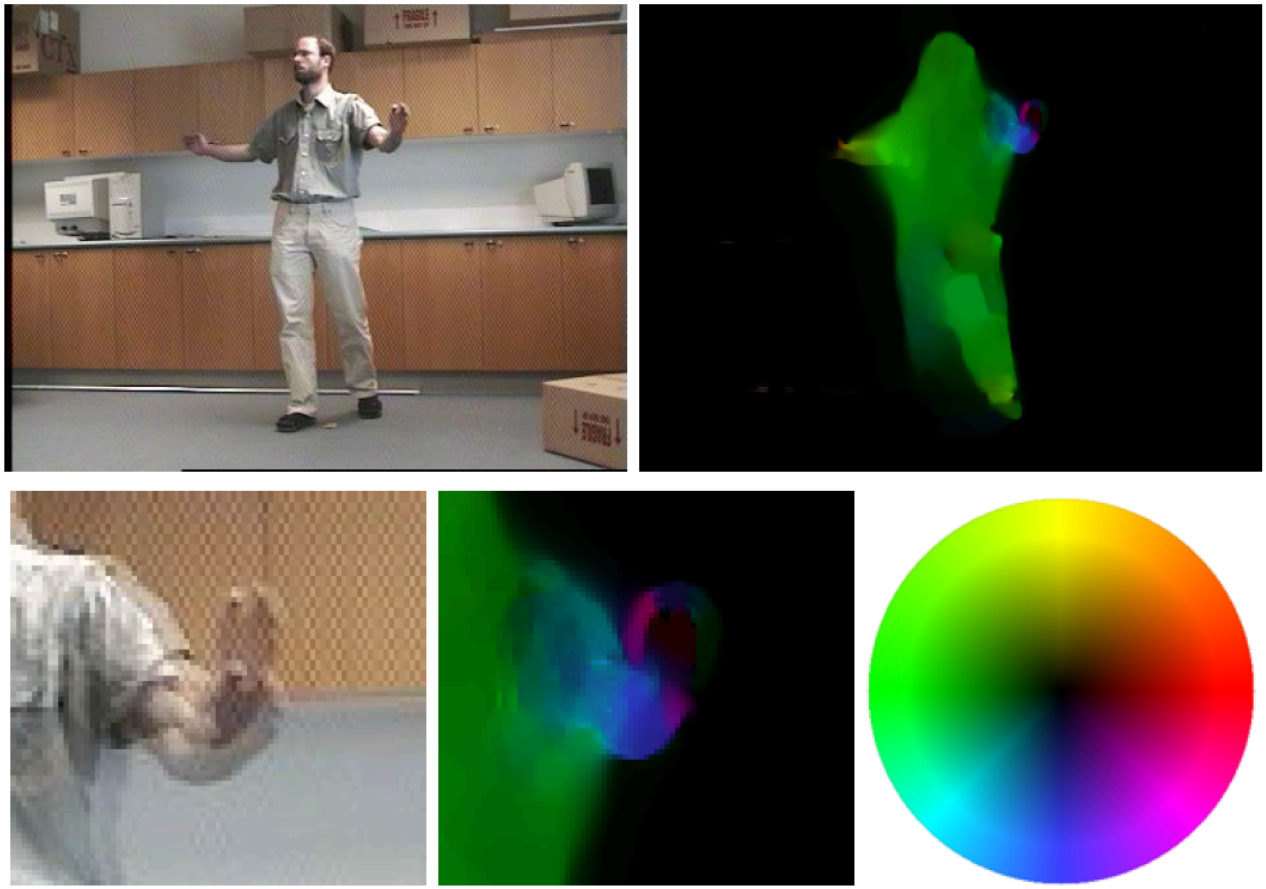


Figure 1. **Top row:** Image of a sequence where the person is stepping forward and moving his hands. The optical flow estimated with the method from [4] is quite accurate for the main body and the legs, but the hands are not accurately captured. **Bottom row, left:** Overlay of two successive frames showing the motion of one of the hands. **Center:** The arm motion is still good but the hand has a smaller scale than its displacement leading to a local minimum. **Right:** Color map used to visualize flow fields in this paper. Smaller vectors are darker and color indicates the direction.

Brox et al 09

Strategy

- Segment into regions, estimate region correspondences
  - use to inform flow estimate

Figure 9. **Left:** Two overlaid images of a tennis player in action. **Center left:** Region correspondences. **Center right:** Result with optical flow from [4]. The motion of the right leg is too fast to be estimated. **Right:** The proposed method captures the motion of the leg.

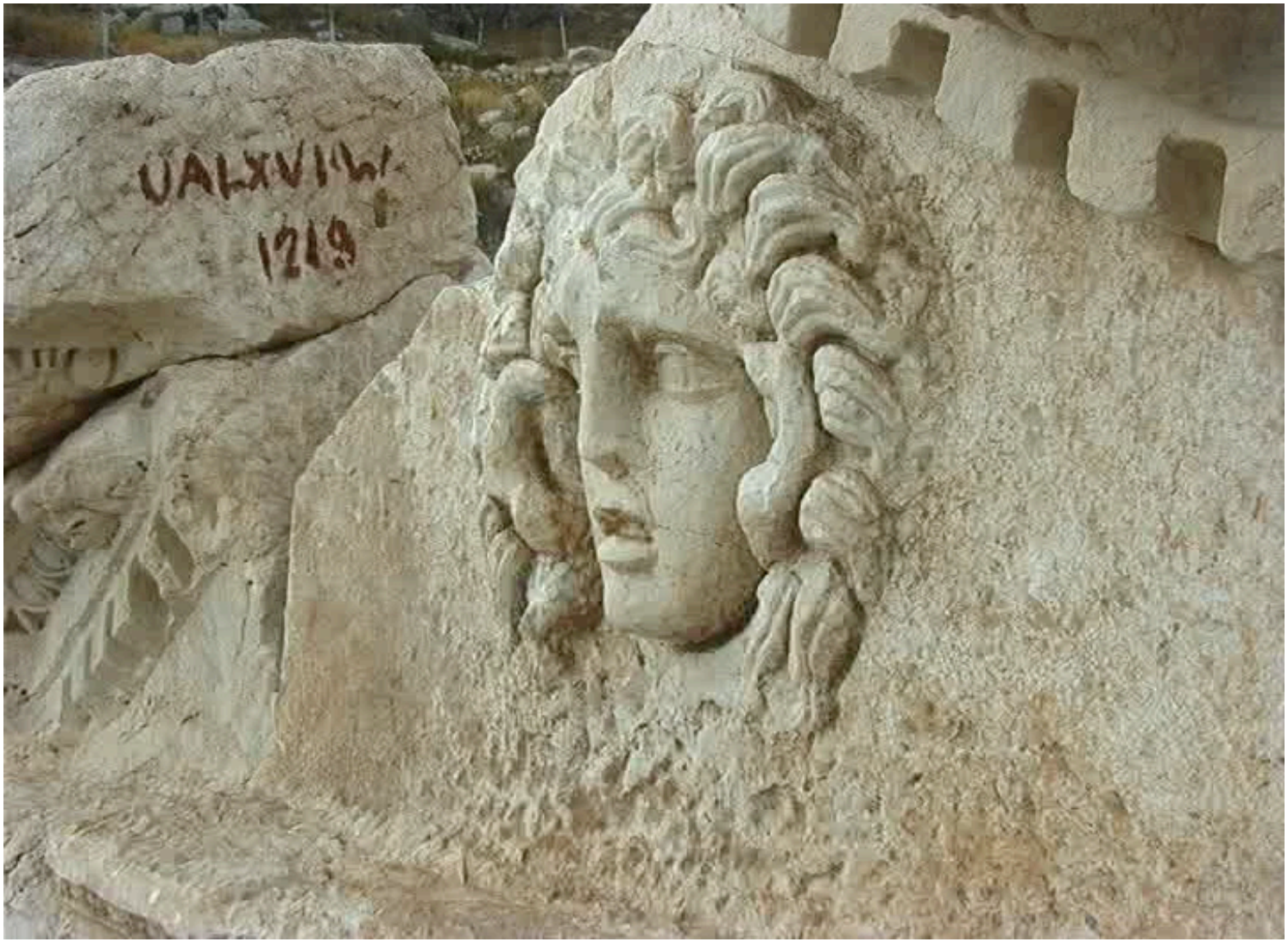
Brox et al 09

Optical flow resources

- Datasets and evaluations:
  - Middlebury optical flow page has longstanding
    - datasets
    - evaluations with leaderboards
    - datasets with groundtruth
    - refs to other such collections
      - (but this is the best known, by a long way)
  - https://vision.middlebury.edu/flow/

Camera and structure from motion

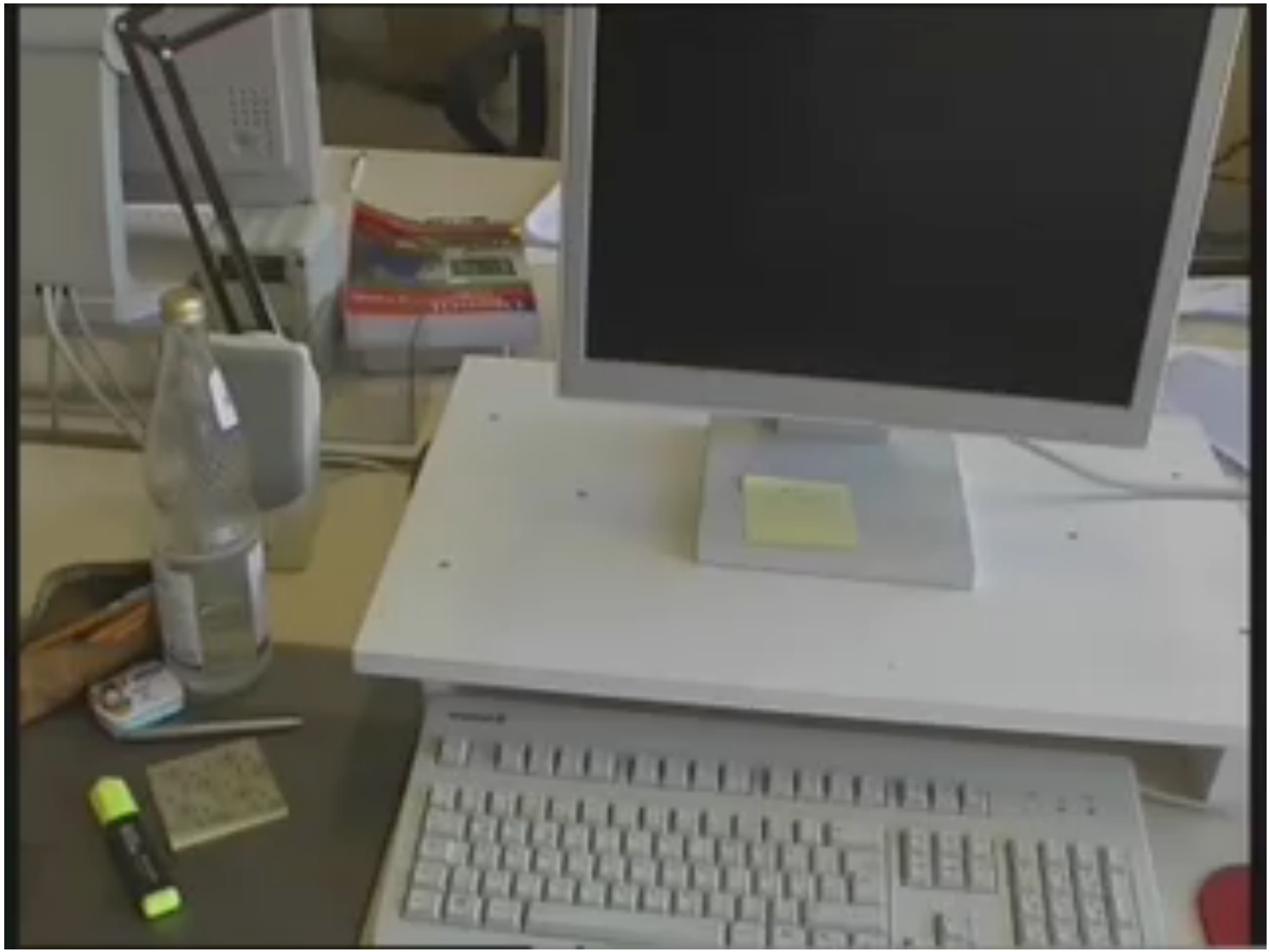
- Assume:
  - a moving camera views a static scene
  - the camera is orthographic OR
    - weak perspective applies with one scale for all
  - all points can be seen in all views AND all correspondences are known
- Can get:
  - the positions of all points in the scene
  - the configuration of each camera
- Applications
  - Reconstruction: Build a 3D model out of the reconstructed points
  - Mapping: Use the camera information to figure out where you went
  - Object insertion: Render a 3D model using the cameras, then composite the videos

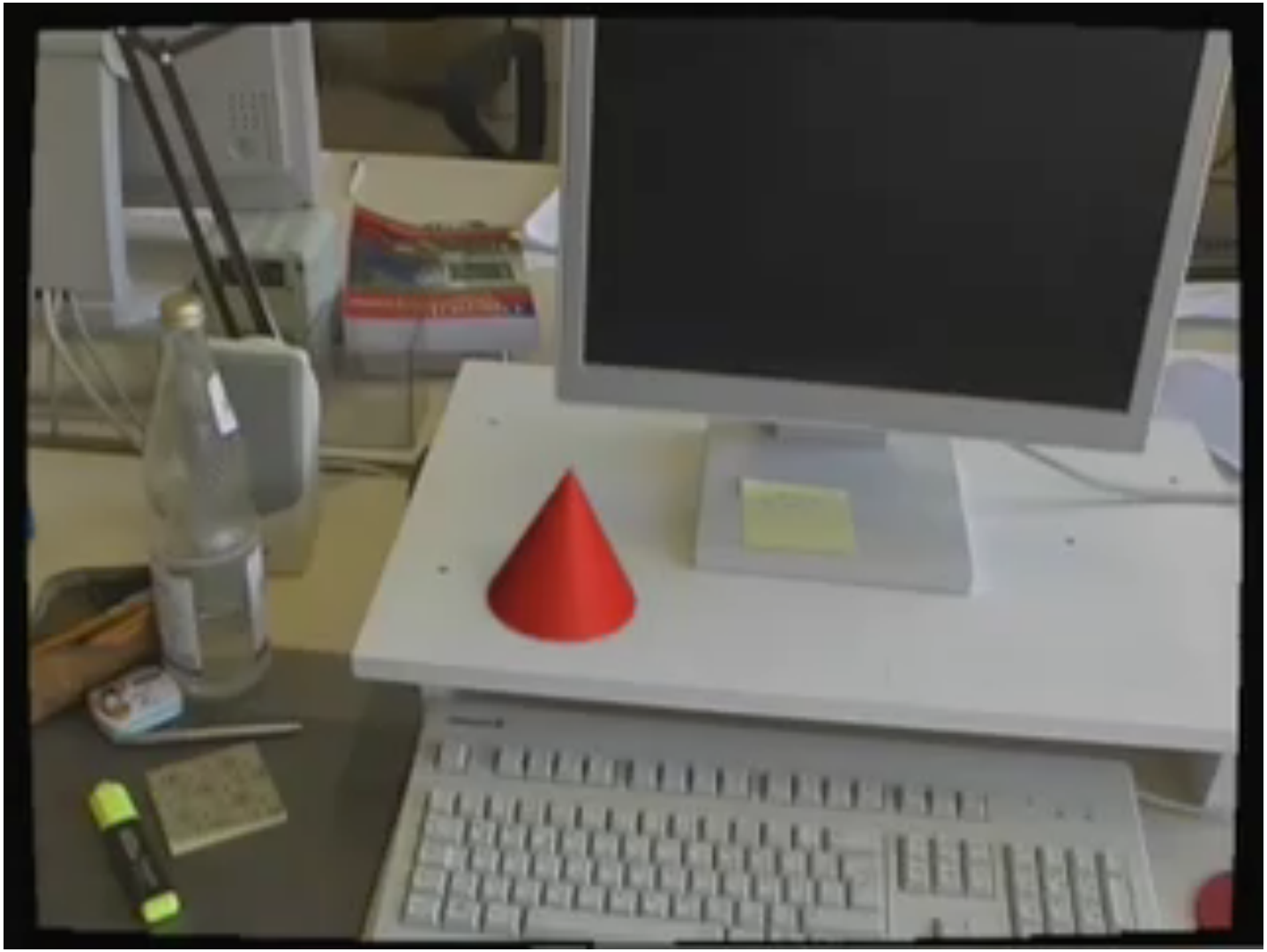


M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, R. Koch, Visual modeling with a hand-held camera, International Journal of Computer Vision 59(3), 207-232, 2004

Rendering and compositing

- Rendering:
  - take camera model, object model, lighting model, make a picture
  - very highly developed and well understood subject
  - many renderers available; tend to take a lot of skill to use (Luxrender)
- Compositing:
  - place two images on top of one another
  - new picture using some pixels from one, some from the other
  - example:
    - green screening
      - take non-green pixels from background, non-bg pixels from top





Simple case in some detail

- Cameras are calibrated scaled orthographic cameras
- We see points that don't move
- All points are seen in all views
- All correspondences are known

- Actually
  - all these constraints can be relaxed
    - any camera OK
    - point motion tricky, but do-able
    - all points in all views isn't required
    - can make correspondences relatively easily

Scaled orthographic projection - II

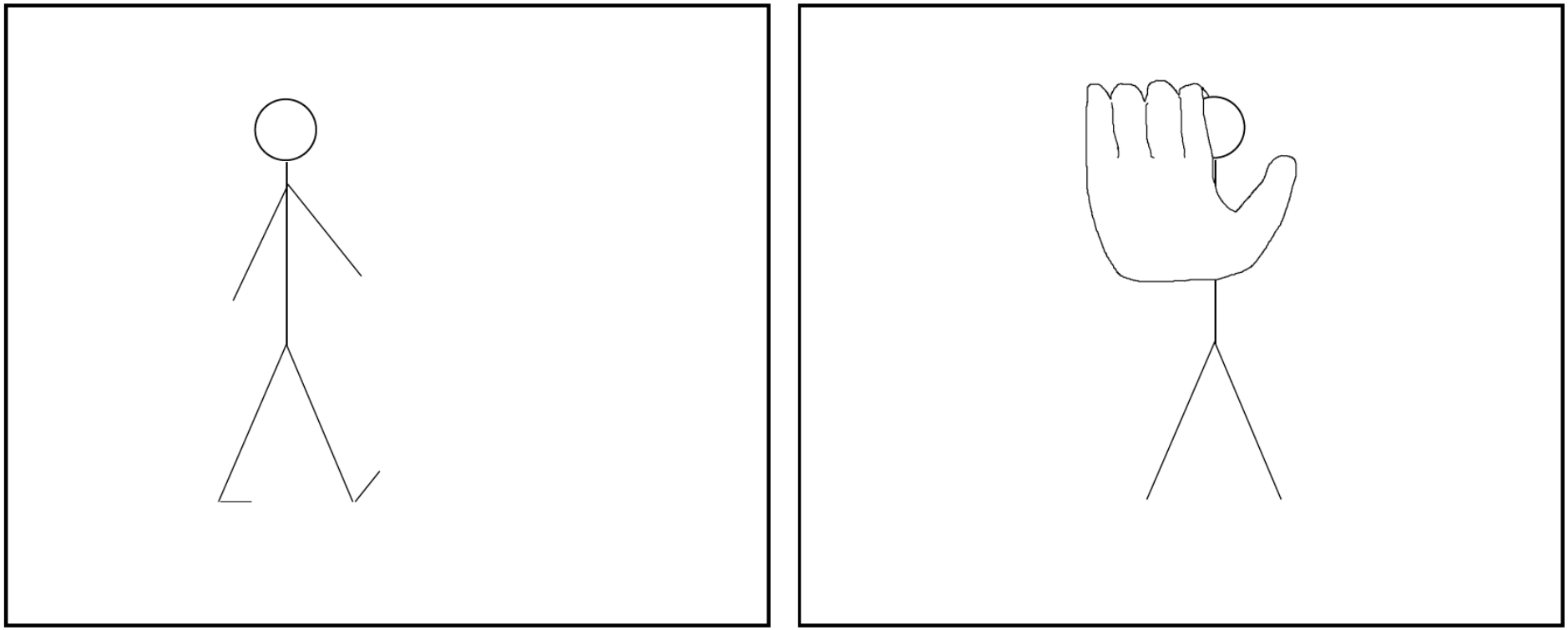


FIGURE 2.5: *The pedestrian on the* **left** *is viewed from some way away, so the distance to the pedestrian is much larger than the change in depth over the pedestrian. In this case, which is quite common for views of people, scaled orthography will apply. The celebrity on the* **right** *is holding a hand up to prevent the camera viewing their face; the hand is quite close to the camera, and the body is an armslength away. In this case, perspective effects are strong. The hand looks big because it is close, and the head looks small because it is far.*

Remember this: *Scaled orthographic projection maps*

$$(X, Y, Z) \rightarrow s(X, Y)$$

where s is some scale. The model applies when the distance to the points being viewed is much greater than their relief. Many views of people have this property.

Scaled Orthographic camera matrix

- And this becomes (for the relevant group of points)

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \mathcal{C} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathcal{W} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

Intrinsics: Hide the scale in here

Extrinsics - rotation and translation wrt world coords

- We will see further simplifications soon

Scaled orthographic cameras

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \mathcal{C} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \mathcal{W} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

- Alternatively
  - the camera film plane has
    - two axes, u and v
    - an origin, at (tx, ty)
  - axes are at right angles
  - axes are the same length
  - point in 3D is $(x, y, z) = \mathbf{x}$
  - equation:

$$\mathbf{x} \rightarrow (\mathbf{u} \cdot \mathbf{x} + t_x, \mathbf{v} \cdot \mathbf{x} + t_y)$$

Simplify

- Now place the 3D origin at center of gravity of points
  - ie mean of x over all points is zero, mean of y is zero, mean of z is zero
- Camera origin at center of gravity of image points
  - we see all of them, so we can compute this
  - this is the projection of 3D center of gravity
- Now camera becomes

$$\mathbf{x} \rightarrow (\mathbf{u} \cdot \mathbf{x}, \mathbf{v} \cdot \mathbf{x})$$

- Index for points, views

$$\mathbf{x}_j \rightarrow (\mathbf{u}_i \cdot \mathbf{x}_j, \mathbf{v}_i \cdot \mathbf{x}_j)$$

Multiple views

- More notation:
  - write $x_{i,j}$ for the first (x) coordinate of the i'th picture of the j'th point
  - write $y_{i,j}$ for the second (y) coordinate of the i'th picture of the j'th point
- We had: $\mathbf{x}_j \rightarrow \left(\mathbf{u}_i \cdot \mathbf{x}_j, \mathbf{v}_i \cdot \mathbf{x}_j\right)$
- Rewrite:

$$\begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_i^T \\ \mathbf{v}_i^T \end{pmatrix} \mathbf{x}_j$$

Multiple views

Point

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \cdots & & & \\ y_{m,1} & y_{m,2} & \cdots & y_{m,n} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,n} \\ \cdots & & & \\ y_{m,1} & y_{m,2} & \cdots & y_{m,n} \end{pmatrix}$$

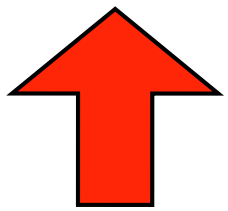
Picture

Data matrix

$$\mathcal{D} = \mathcal{V}\mathcal{X}$$

Multiple views

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \dots & & & \\ y_{m,1} & y_{m,2} & \dots & y_{m,n} \\ y_{1,1} & y_{1,2} & \dots & y_{1,n} \\ y_{2,1} & y_{2,2} & \dots & y_{2,n} \\ \dots & & & \\ y_{m,1} & y_{m,2} & \dots & y_{m,n} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \dots \\ \mathbf{u}_m^T \\ \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \dots \\ \mathbf{v}_m^T \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{pmatrix}$$



Data - observed!

$$\mathcal{D} = \mathcal{V}\mathcal{X}$$

Multiple views

- The data matrix has rank 3!
  - so we can factor it into an mx3 factor and a 3xn factor
  - (tall+thin)x(short+fat)
  - so we know what to do; SVD -> factors
    - recall SVD from IRLS!
- These factors are not unique
  - assume A is 3x3 with rank 3, we get symmetry below

$$\mathcal{D} = \mathcal{T}\mathcal{S} = (\mathcal{T}\mathcal{A})(\mathcal{A}^{-1}\mathcal{S})$$

Camera and reconstruction

- Can choose factors uniquely
  - recall v_i, u_i are
    - at right angles
    - same length
- Algorithm
  - form D
  - factor
  - now choose A so that v_i, u_i are at right angles, same length
    - by numerical optimization
- What if there are missing points?
  - Fairly simple optimization trick, following slides

Factoring without all points

- Write D for the data matrix, W for a mask matrix
  - W_ij=0 if that entry of D is unknown, =1 if it is known
- Strategy:
  - choose S, T to minimize

$$\sum_{i,j} W_{ij}(D_{ij} - \sum_{k} T_{ik}S_{kj})^2$$

  - now multiply these S, T - the result is the whole of D
    - i.e. holes are filled in
  - we expect this to work even if D has many holes in it because
    - there are few parameters in S, T

Factors with missing points

- How to minimize? set the gradient to zero
- gradient with respect to T_uv is

$$2\sum_j W_{uj}(D_{uj} - \sum_k T_{uk}S_{kj})S_{vj}$$

- gradient with respect to S_uv is

$$2\sum_i W_{iv}(D_{iv} - \sum_k T_{ik}S_{kv})T_{iu}$$

Notice there are TWO products here

$$\begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \cdots & & & \\ y_{m,1} & y_{m,2} & \cdots & y_{m,n} \\ y_{1,1} & y_{1,2} & \cdots & y_{1,n} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,n} \\ \cdots & & & \\ y_{m,1} & y_{m,2} & \cdots & y_{m,n} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_1^T \\ \mathbf{u}_2^T \\ \cdots \\ \mathbf{u}_m^T \\ \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \cdots \\ \mathbf{v}_m^T \end{pmatrix} \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \end{pmatrix}$$

Points in 3D

Estimates of camera rotation

What happened to translation?

State of the art (ish)

- Reconstructions at extremely large scales
  - scale=large city
  - from “scattered” pictures
  - very fine metric accuracy
- “Symmetries” can create serious problems
  - eg indoor corridors, etc
- Low feature environments are a nuisance

- Very little impact of deep learning (so far)
  - mostly, major changes in the type of 3D reconstruction recovered.

Software

- Colmap
  - open source SFM at very large scale
  - backbone of many other projects
  - https://demuc.de/colmap/
- MicMac
  - scale photogrammetry software from French national geographic inst.
  - https://micmac.ensg.eu/index.php/Accueil
- OpenSFM
  - open source SFM at very large scale
  - https://opensfm.org