Optical Flow and Motion

Computer Vision I

CSE 252A

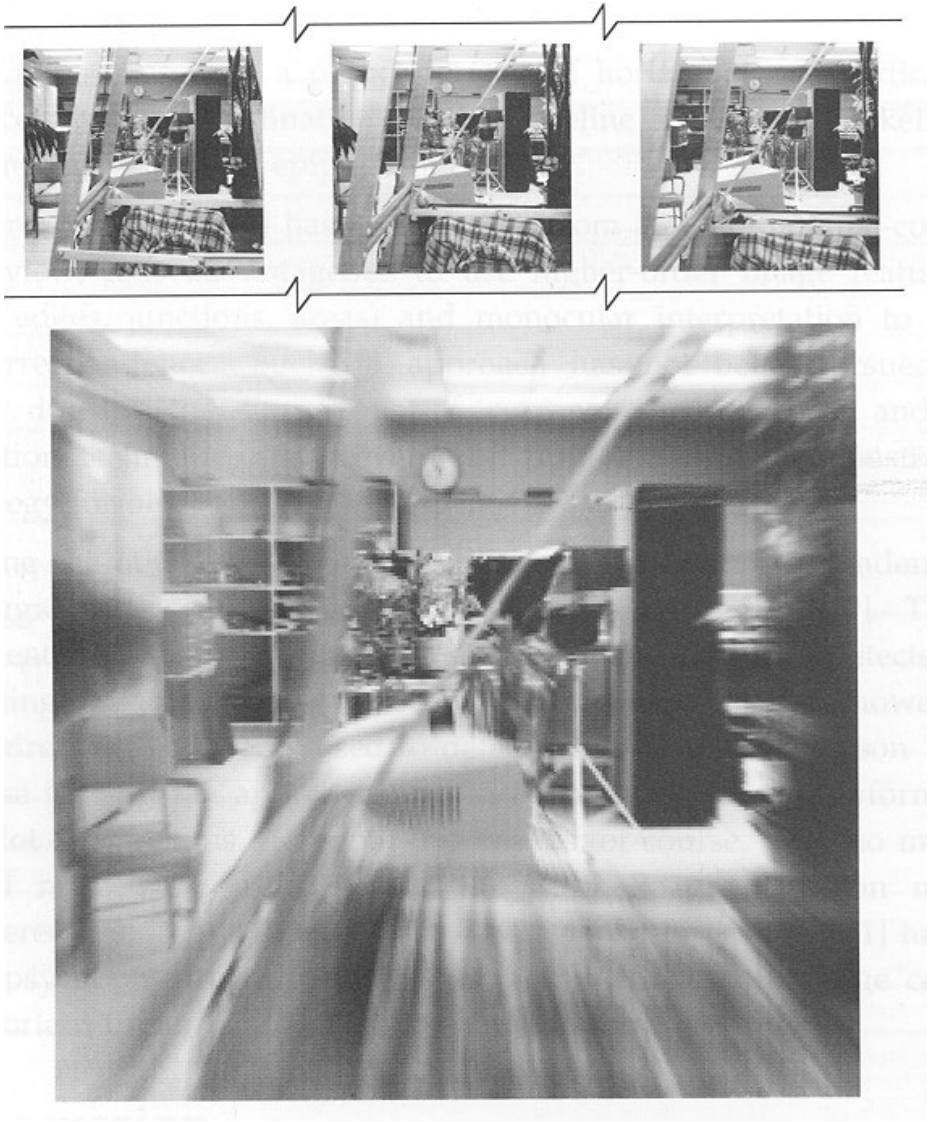
Lecture 12

Announcements

- Assignment 2 is due today, 11:59 PM
- Assignment 3 will be released today
  - Due Nov 22, 11:59 PM

Motion

- Consider a video camera moving continuously along a trajectory (rotating & translating).
- How do points in the image move?
- What does that tell us about the 3D motion & scene structure?



Motion

Structure-from-Motion (SFM)

Given two or more images or video without any information on camera position/motion as input, estimate camera motion and 3-D structure of a scene

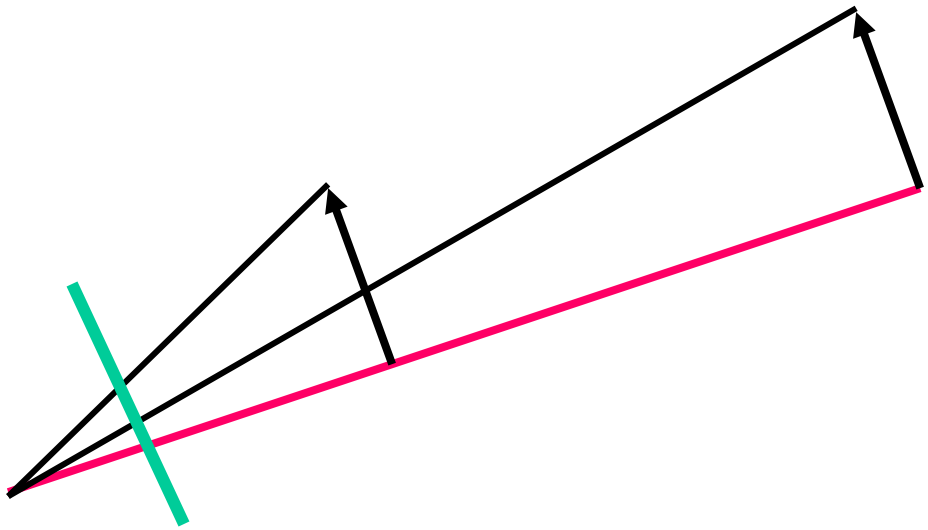
Two Approaches

1. Discrete motion (wide baseline)
2. Continuous (Infinitesimal) motion usually from video

Motion

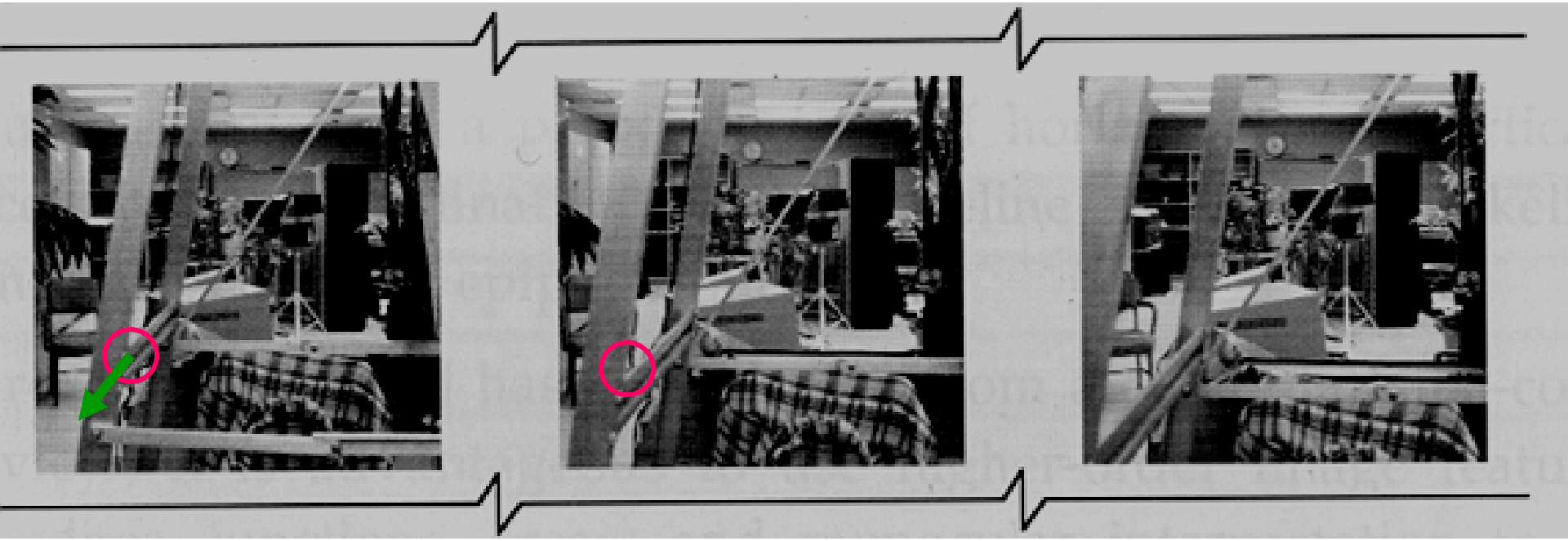
"When objects move at equal speed, those more remote seem to move more slowly."

- Euclid, 300 BC

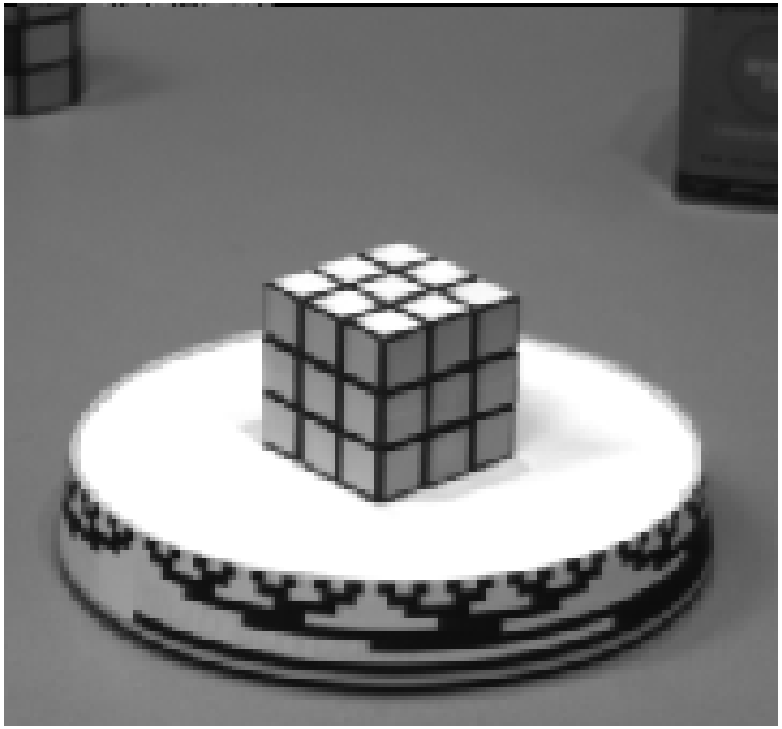


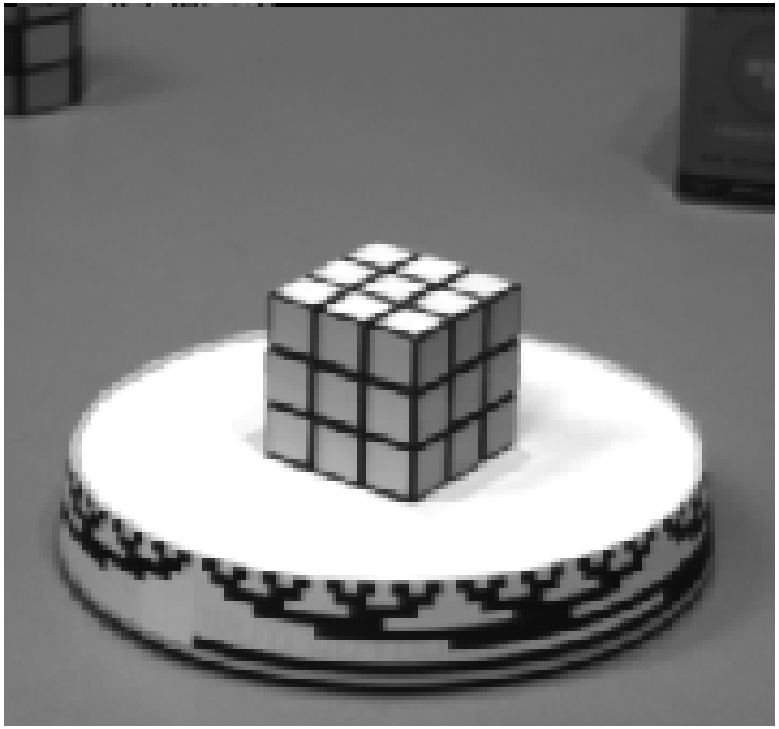
The Motion Field

Where in the image did a point move?



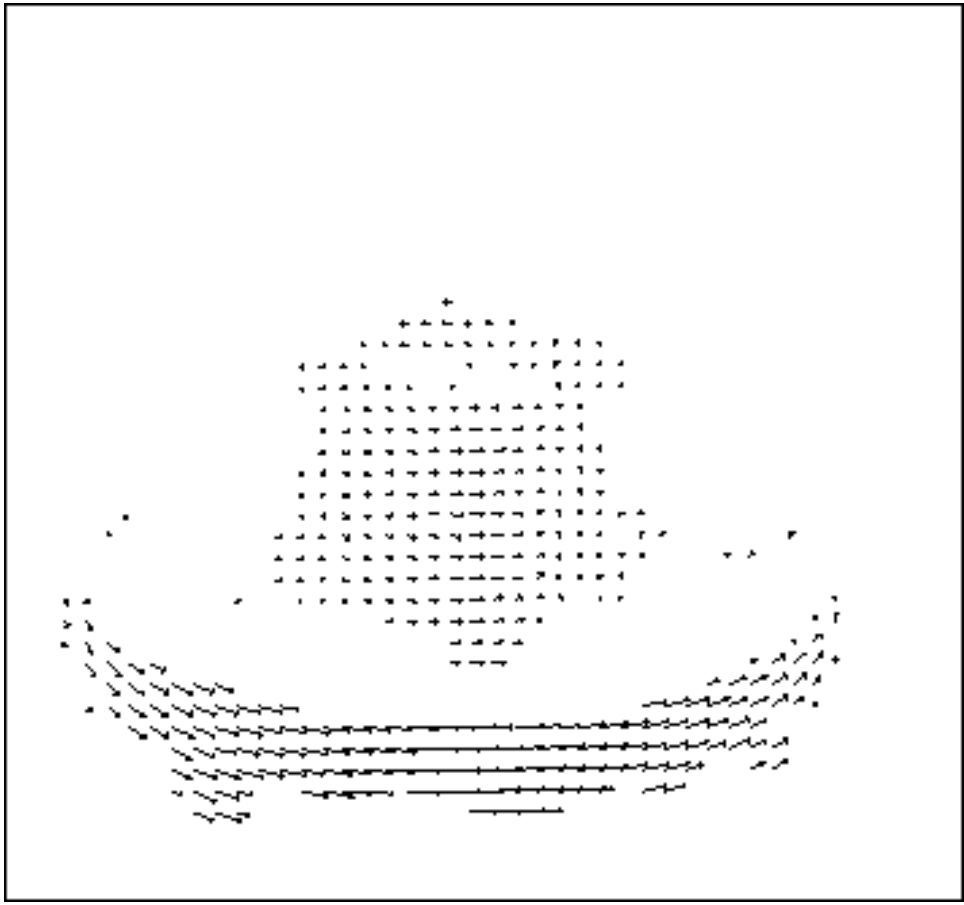
Down and left





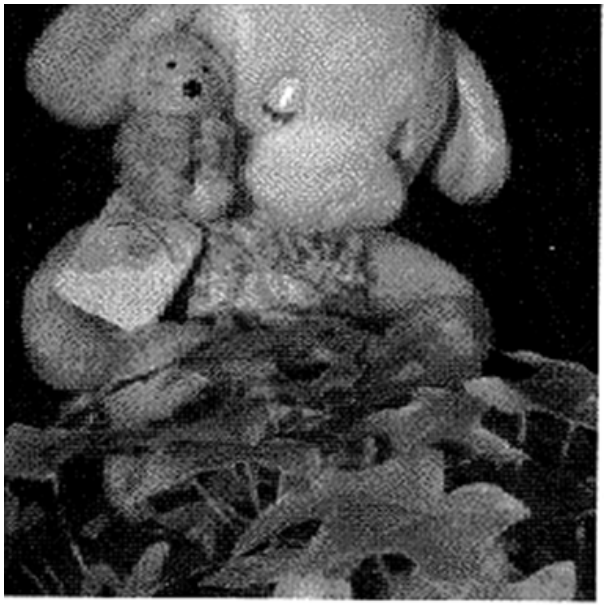
Motion field

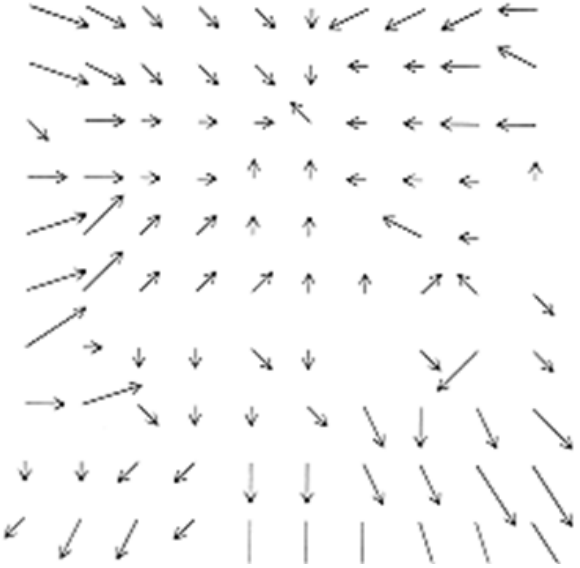
- The motion field is the projection of the 3D scene motion into the image



Motion blur.
Usually in direction of motion field

The Motion Field





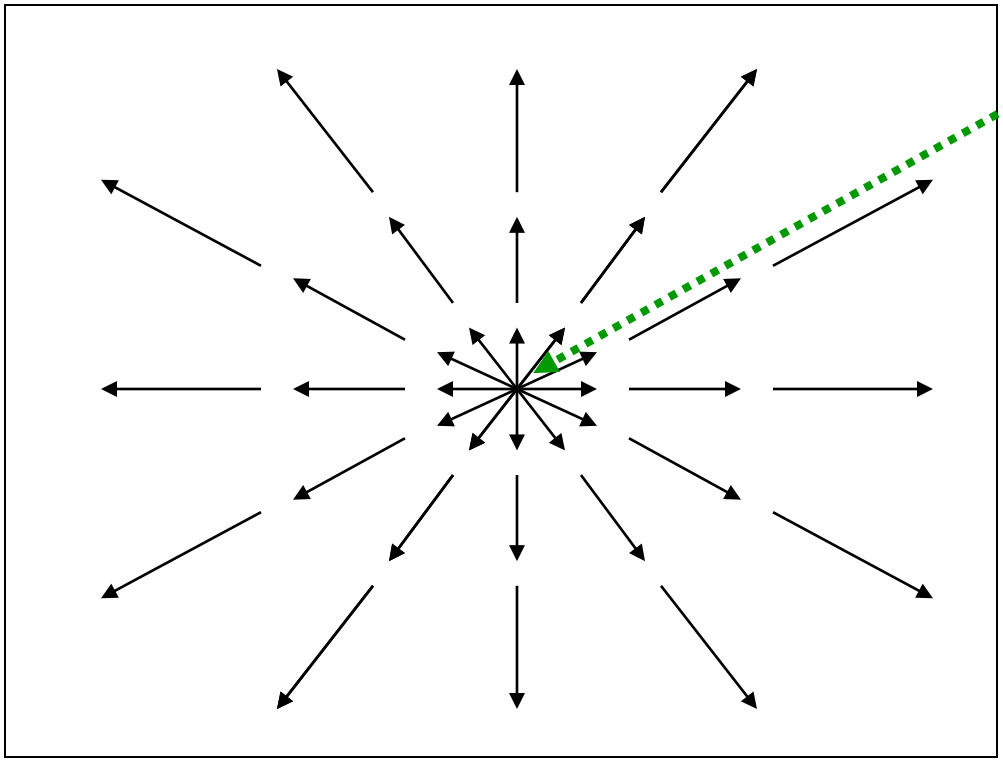
What causes a motion field?

1. Camera moves (translates, rotates)
2. Objects in scene move rigidly
3. Objects articulate (pliers, humans, animals)
4. Objects bend and deform (fish)
5. Blowing smoke, clouds
6. Multiple movements

An example motion field: Camera moving straight along optical axis

The "instantaneous" velocity of all points in an image

LOOMING



The *Focus of Expansion (FOE)*

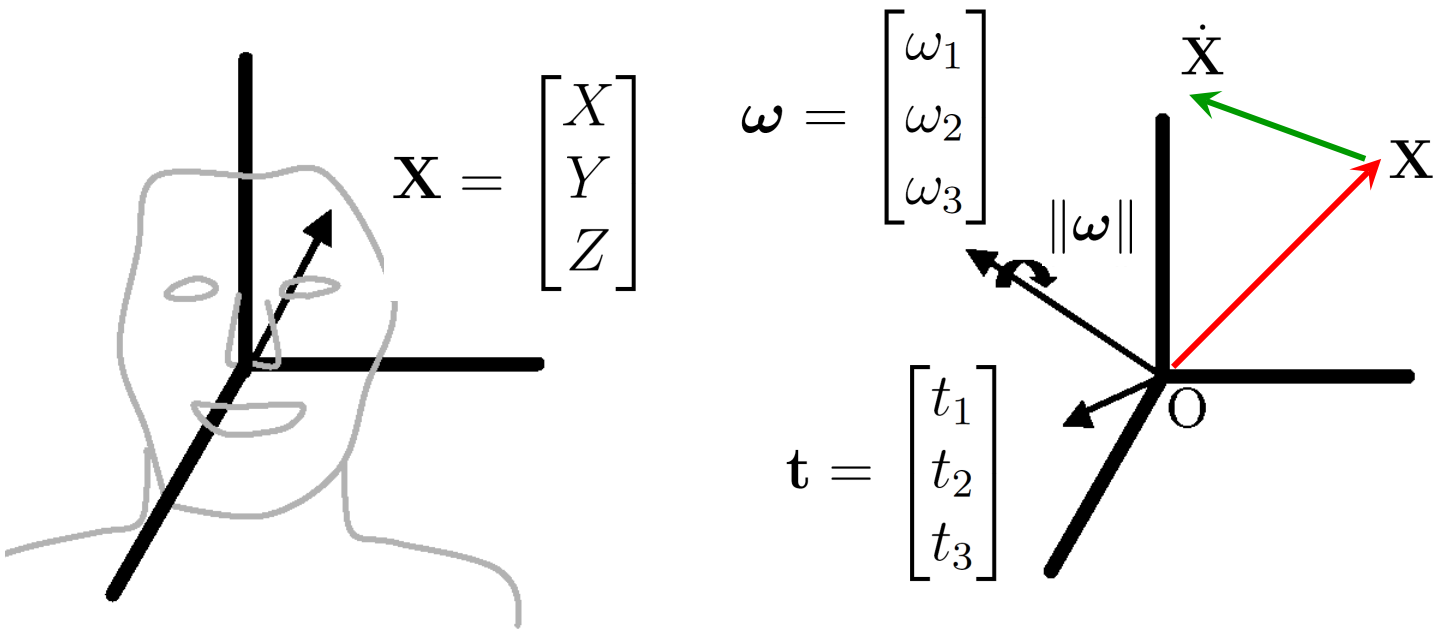
Intersection of velocity vector with image plane

With just this information it is possible to calculate:

1. Direction of motion
2. Time to collision

Rigid Motion: General Case

$$\dot{\mathbf{X}} = -(\boldsymbol{\omega} \times \mathbf{X} + \mathbf{t})$$



Position and orientation of a rigid body
Rotation Matrix & Translation vector

Rigid Motion:
Velocity Vector: $\mathbf{t}$
Angular Velocity Vector: $\boldsymbol{\omega}$

General Motion

Pixel coordinates $\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}$

Taking the time derivative of both sides yields

Motion field pixel coordinates $\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} - \frac{f\dot{Z}}{Z^2} \begin{bmatrix} X \\ Y \end{bmatrix}$

Substitute $\dot{\mathbf{X}} = -(\boldsymbol{\omega} \times \mathbf{X} + \mathbf{t})$

Then, substitute $x = \frac{f}{Z}X$

$y = \frac{f}{Z}Y$

Motion Field Equation

- Motion field pixel coordinates

$$\dot{x} = \frac{t_3 x - t_1 f}{Z} - \omega_2 f + \omega_3 y + \frac{\omega_1 xy - \omega_2 x^2}{f}$$

$$\dot{y} = \frac{t_3 y - t_2 f}{Z} + \omega_1 f - \omega_3 x + \frac{\omega_1 y^2 - \omega_2 xy}{f}$$

Pure Translation

If camera is just translating with velocity $\mathbf{t} = (t_1, t_2, t_3)^\top$,
Then there is no rotation:

$$\boldsymbol{\omega} = \mathbf{0}$$

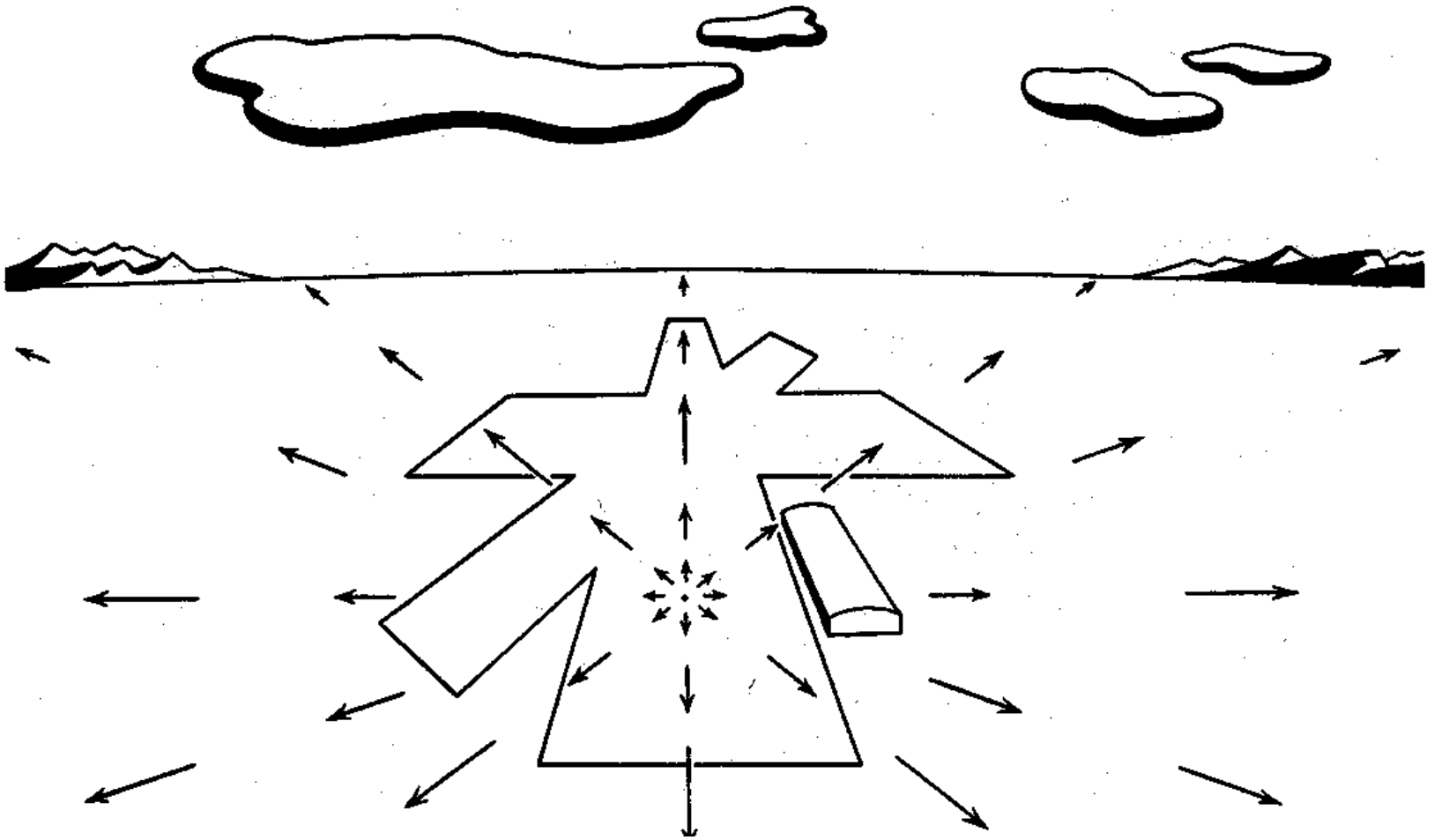
$$\dot{x} = \frac{t_3 x - t_1 f}{Z} - \omega_2 f + \omega_3 y + \frac{\omega_1 xy - \omega_2 x^2}{f}$$

$$\dot{y} = \frac{t_3 y - t_2 f}{Z} + \omega_1 f - \omega_3 x + \frac{\omega_1 y^2 - \omega_2 xy}{f}$$

- $\dot{x}, \dot{y}$ is inversely proportional to Z (remember Euclid)
- Focus of expansion is located at (x, y) where $\dot{x} = 0, \dot{y} = 0$

Forward Translation & Focus of Expansion

[Gibson, 1950]



Focus of Expansion (FOE)

$$\dot{x} = \frac{t_3 x - t_1 f}{Z} - \omega_2 f + \omega_3 y + \frac{\omega_1 xy - \omega_2 x^2}{f}$$

$$\dot{y} = \frac{t_3 y - t_2 f}{Z} + \omega_1 f - \omega_3 x + \frac{\omega_1 y^2 - \omega_2 xy}{f}$$

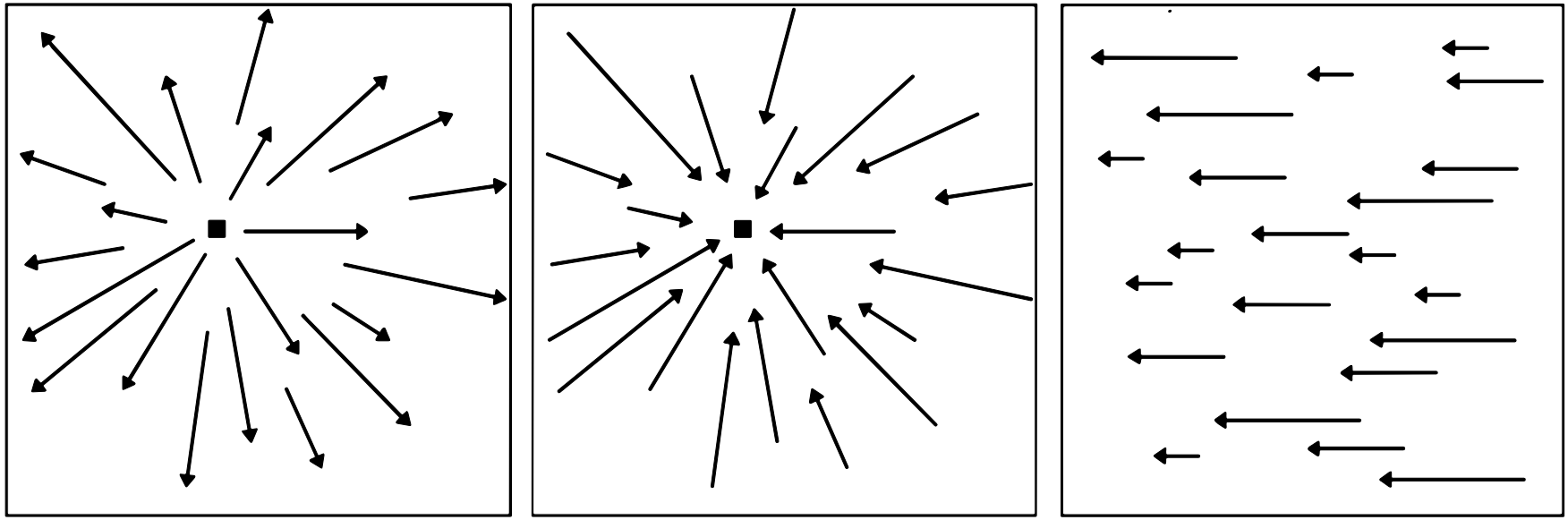
- Focus of expansion is located at (x, y) where $\dot{x} = 0, \dot{y} = 0$
- Substitute and solve for (x, y)

$$x = f\frac{t_1}{t_3}$$

$$y = f\frac{t_2}{t_3}$$

Insight: The FOE is the perspective projection of the linear velocity vector $\mathbf{t} = (t_1, t_2, t_3)^\top$

Pure Translation

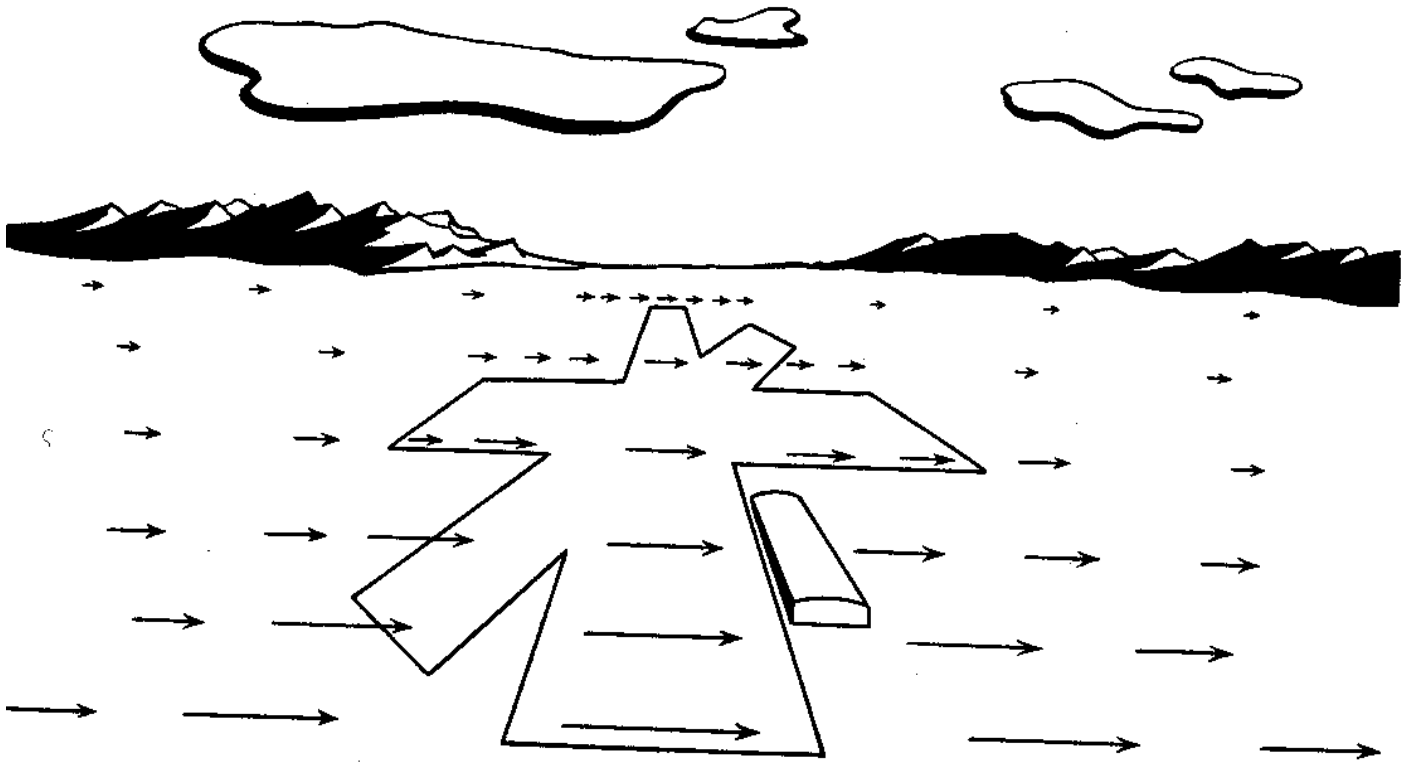


Radial
about FOE

Parallel
(FOE point at infinity)
$t_3 = 0$
Motion parallel to image plane

Sideways Translation

[Gibson 1950]



Parallel

(FOE point at infinity)

$t_3 = 0$

Motion parallel to image plane

Pure Rotation: $\mathbf{t} = \mathbf{0}$

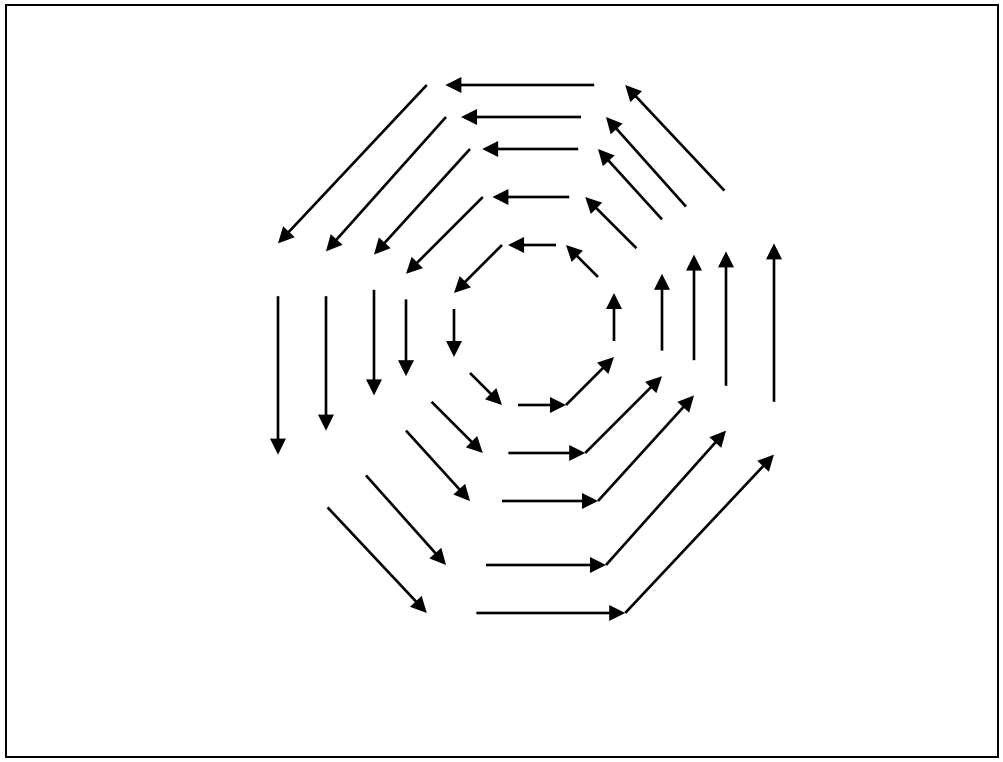
$$\dot{x} = \frac{t_3 x - t_1 f}{Z} - \omega_2 f + \omega_3 y + \frac{\omega_1 xy - \omega_2 x^2}{f}$$

$$\dot{y} = \frac{t_3 y - t_2 f}{Z} + \omega_1 f - \omega_3 x + \frac{\omega_1 y^2 - \omega_2 xy}{f}$$

- Independent of $\mathbf{t} = (t_1, t_2, t_3)^\top$
- Independent of Z
- Only function of x, y, f, and $\boldsymbol{\omega}$

Rotational MOTION FIELD

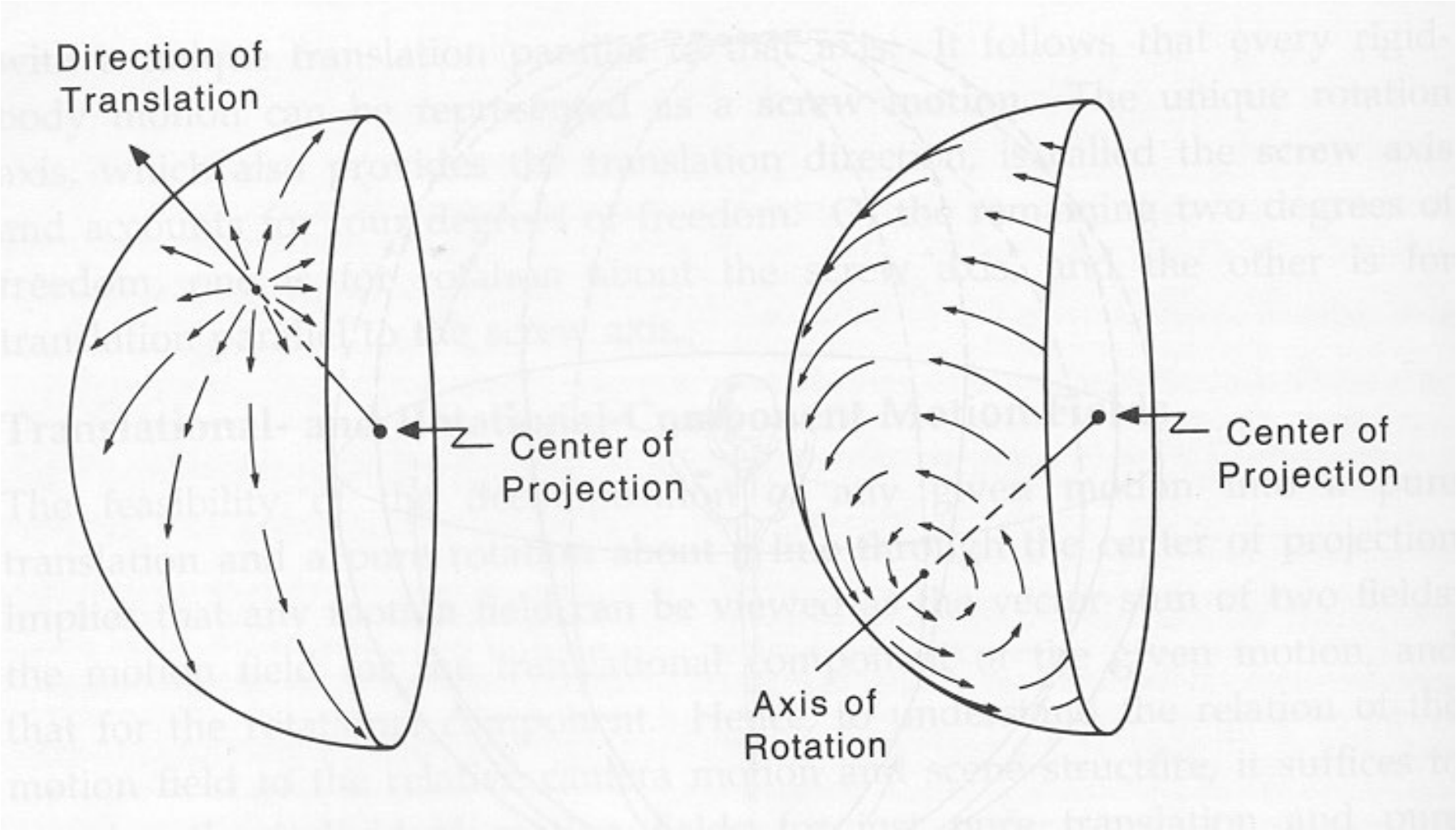
The “instantaneous” velocity of points in an image



PURE ROTATION

$\omega = (0,0,1)^T$

Pure translation and pure rotation: Motion Field on Sphere



Motion Field Equation: Estimate Depth

$$\dot{x} = \frac{t_3 x - t_1 f}{Z} - \omega_2 f + \omega_3 y + \frac{\omega_1 xy - \omega_2 x^2}{f}$$

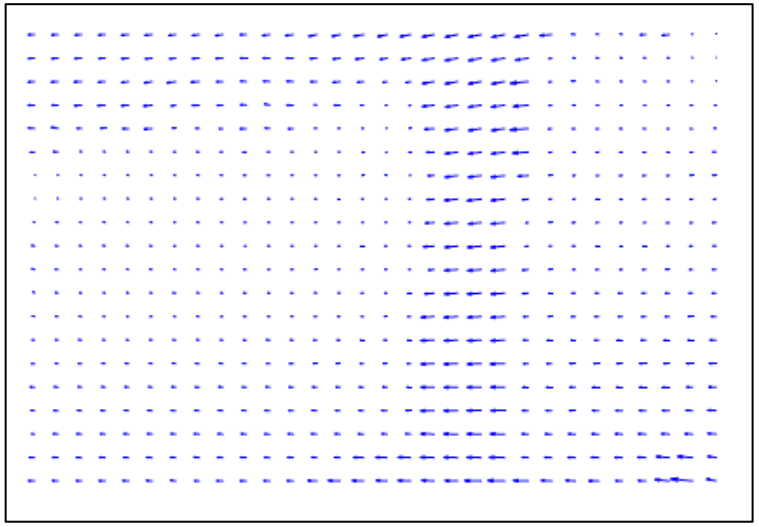
$$\dot{y} = \frac{t_3 y - t_2 f}{Z} + \omega_1 f - \omega_3 x + \frac{\omega_1 y^2 - \omega_2 xy}{f}$$

If $\mathbf{t}$, $\boldsymbol{\omega}$, and f are known or measured, then for each image point (x, y), one can solve for the depth Z given measured motion $\dot{u}, \dot{v}$ at (x, y)

Depth is inversely proportional to image velocity

Measuring Motion

Optical Flow Field



Estimating the motion field from images

1. Feature-based (Sect. 8.4.2 of Trucco & Verri)
    1. Detect (corner-like) features in an image
    2. Search for the same features nearby (feature tracking)

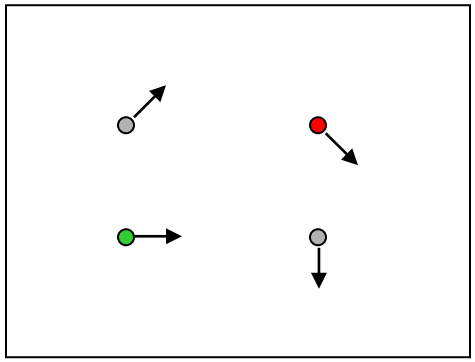
2. Differential techniques (Sect. 8.4.1)

Optical Flow

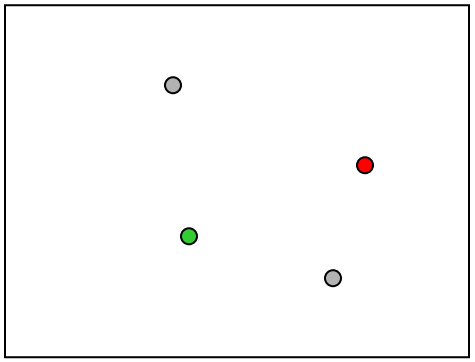
- Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and a scene

- Ideally, the optical flow is the projection of the three-dimensional velocity vectors on the image (i.e., the motion field)
  - As we will see, it is not

Problem Definition: Optical Flow



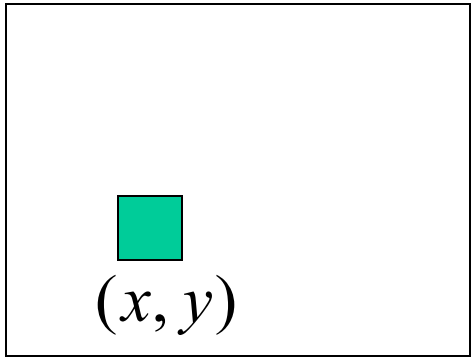
$H(x, y)$



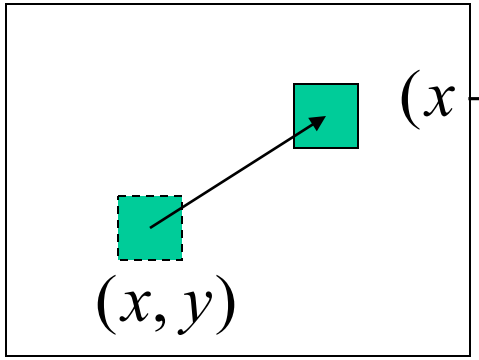
$I(x, y)$

- How to estimate pixel motion from image H to image I?
  - Find pixel correspondences
    - Given a pixel in H, look for nearby pixels of the same color in I
- Key assumptions
  - **Color constancy**: a point in H looks "the same" in image I
    - For grayscale images, this is **brightness constancy**
  - **Small motion**: points do not move very far

Optical Flow Constraint Equation



time t



time $t + \delta t$

Optical Flow: (u, v)

Displacement:

$$(\delta x, \delta y) = (u\,\delta t, v\,\delta t)$$

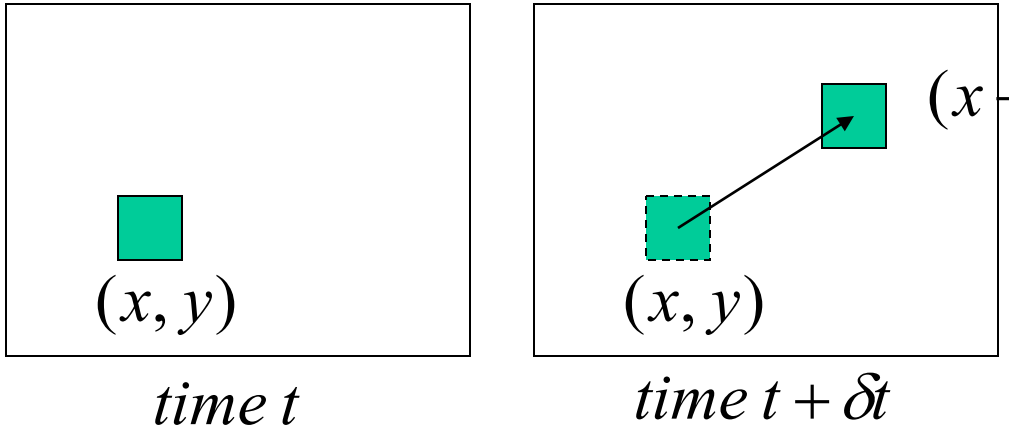
1. Assume brightness of patch remains same in both images:

$$I(x + u\,\delta t, y + v\,\delta t, t + \delta t) = I(x, y, t)$$

2. Assume small motion: (Taylor expansion of LHS up to first order)

$$I(x, y, t) + \delta x \frac{\partial I}{\partial x} + \delta y \frac{\partial I}{\partial y} + \delta t \frac{\partial I}{\partial t} = I(x, y, t)$$

Optical Flow Constraint Equation



Optical Flow: (u, v)

Displacement:

$$(\delta x, \delta y) = (u\,\delta t, v\,\delta t)$$

3. Subtracting $I(x,y,t)$ from both sides and dividing by δt

$$\frac{\delta x}{\delta t}\frac{\partial I}{\partial x} + \frac{\delta y}{\delta t}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$$

4. Assume small interval, this becomes:

$$\frac{dx}{dt}\frac{\partial I}{\partial x} + \frac{dy}{dt}\frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0$$

Solving for flow

Optical flow constraint equation :

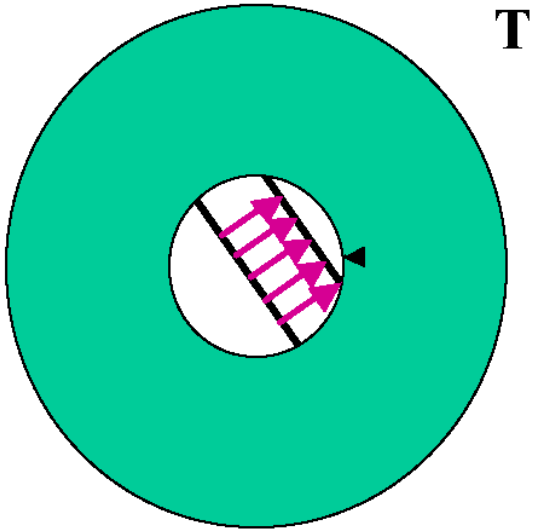
$$\frac{dx}{dt}\frac{\partial I}{\partial x}+\frac{dy}{dt}\frac{\partial I}{\partial y}+\frac{\partial I}{\partial t}=0$$

- We can measure $\frac{\partial I}{\partial x},\frac{\partial I}{\partial y},\frac{\partial I}{\partial t}$
  - $\frac{\partial I}{\partial x}$ Filter image with kernel [-1, 0, 1]
  - $\frac{\partial I}{\partial y}$ Filter image with $[-1, 0, 1]^T$
  - $\frac{\partial I}{\partial t}$ Consider stacking 3 images at (t-1, t, t+1), then filter over time with kernel [-1,0,1]; or backward difference
- We want to solve for $\frac{dx}{dt},\frac{dy}{dt}$
- One equation, two unknowns → Cannot solve it

Aperture Problem and Normal Flow

We measure:

$$I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y}, \quad I_t = \frac{\partial I}{\partial t}$$

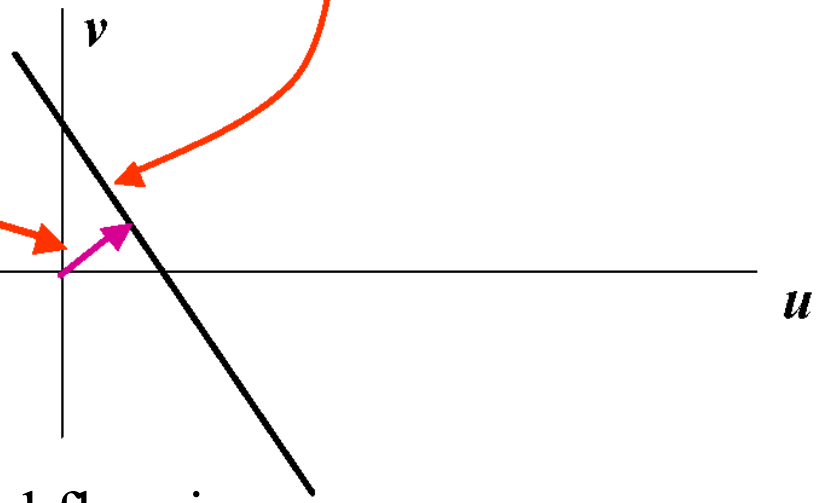


The gradient constraint:

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

$$\nabla I \bullet \vec{U} = 0$$

Defines a line in the *(u,v)* space



We want to estimate Flow vector

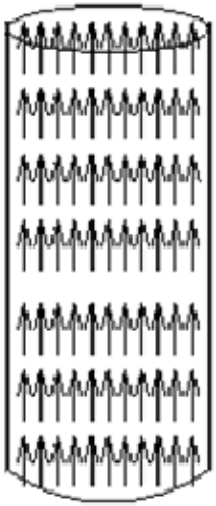
$$u = \frac{dx}{dt}, \quad v = \frac{dy}{dt}$$

Normal Flow:

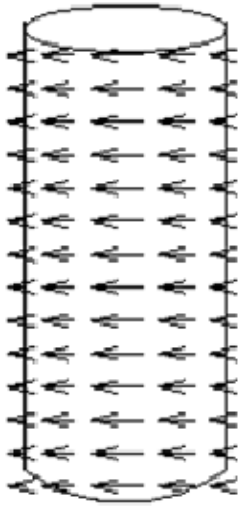
$$u_\perp = -\frac{I_t}{|\nabla I|}\frac{\nabla I}{|\nabla I|}$$

The component of the optical flow in the direction of the image gradient

Barber Pole Illusion



Optical Flow

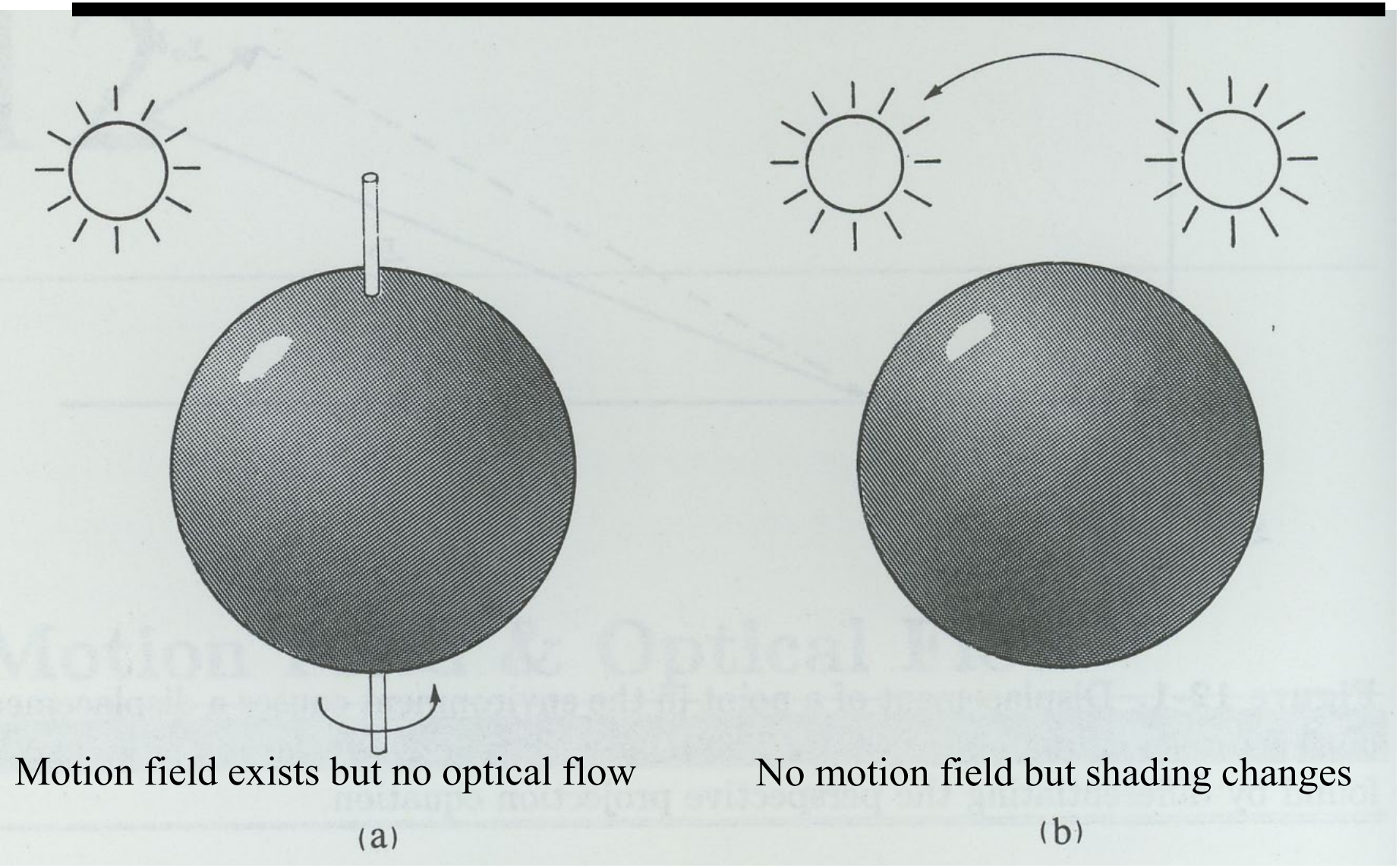


Motion Field

Optical flow field is not always the same as the motion field

http://www.opticalillusion.net/optical-illusions/the-barber-pole-illusion/

Optical Flow ≠ Motion Field



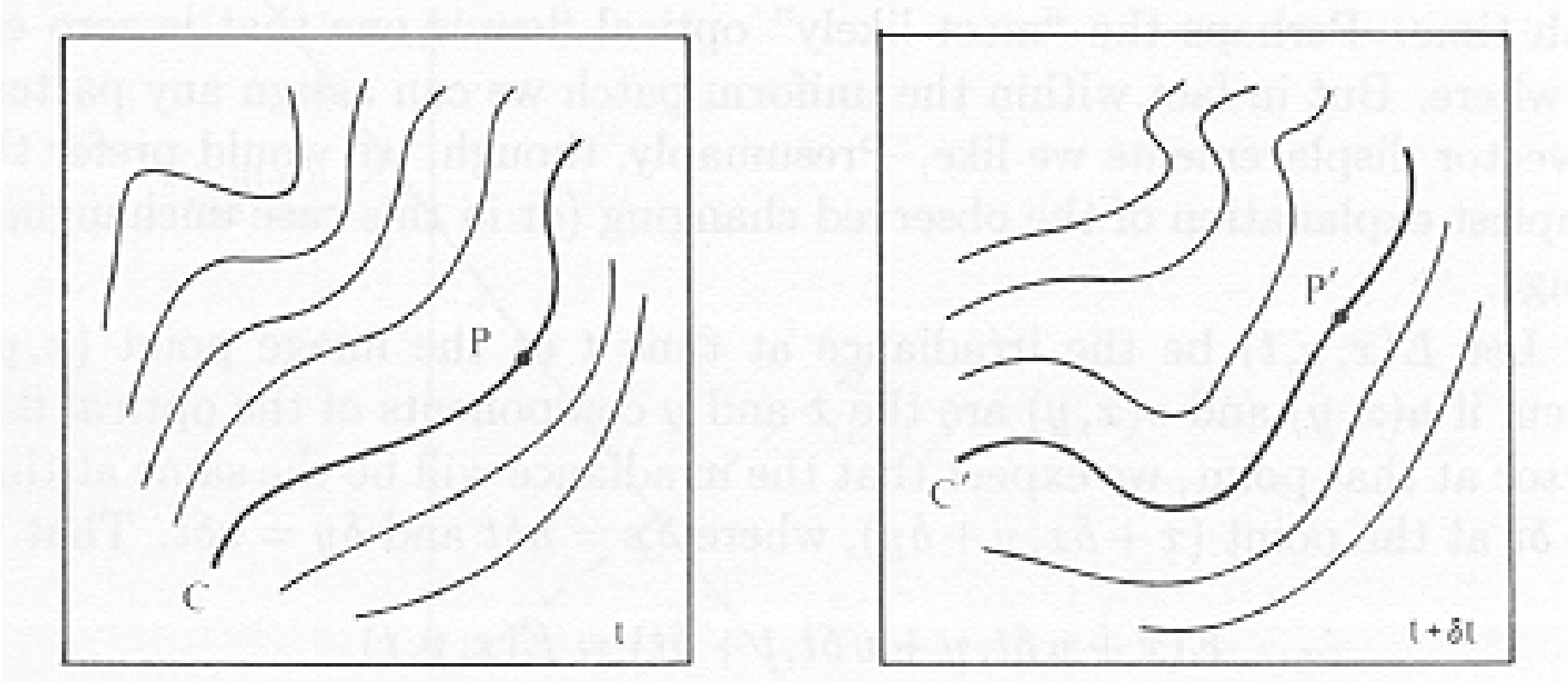
Motion field exists but no optical flow

(a)

No motion field but shading changes

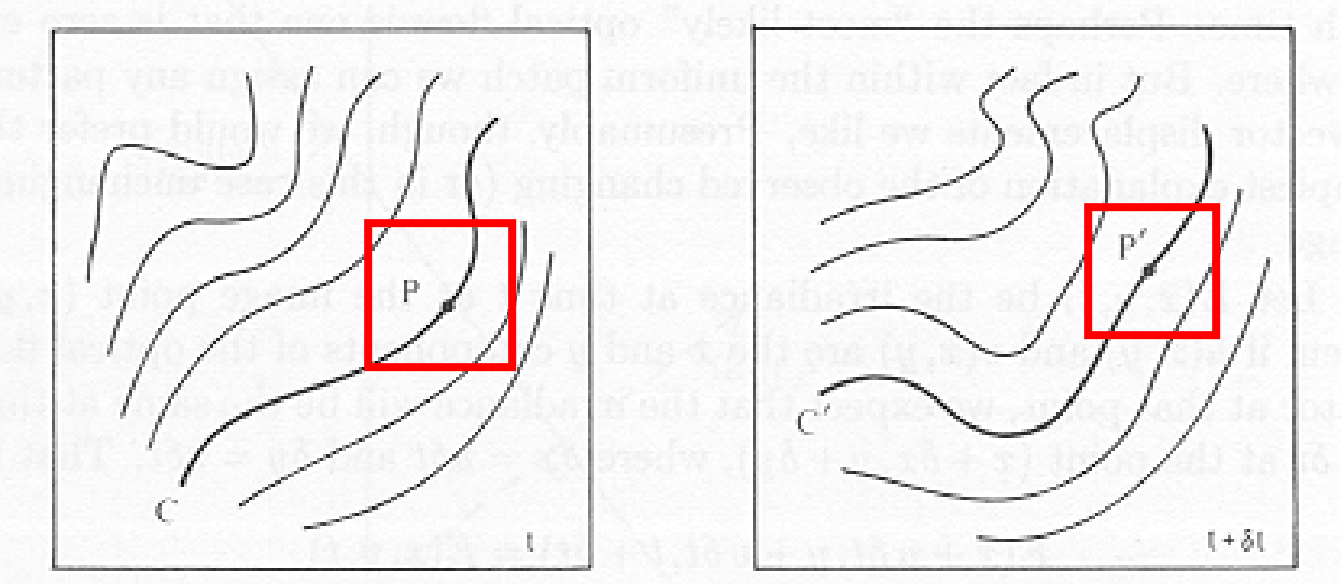
(b)

What is the correspondence of P & P’



Contour plots of image intensity in two images

Two ways to get flow



1. Think globally, and regularize over image

2. Look over window and assume constant motion in the window

Horn & Schunck algorithm

Additional smoothness constraint :

$$e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2))dxdy,$$

besides OF constraint equation term

$$e_c = \iint (I_x u + I_y v + I_t)^2 dxdy,$$

minimize $e_s + \lambda e_c$

Lucas-Kanade: Integrate over a Patch

Assume a single velocity (u,v) for pixels within an image patch Ω

$$E(u,v) = \sum_{x,y \in \Omega} \left(I_x(x,y)u + I_y(x,y)v + I_t \right)^2$$

E(u,v) is minimized when partial derivatives equal zero.

$$\frac{dE(u,v)}{du} = \sum 2I_x \left(I_x u + I_y v + I_t \right) = 0$$

$$\frac{dE(u,v)}{dv} = \sum 2I_y \left(I_x u + I_y v + I_t \right) = 0$$

In matrix form:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$

On the LHS: sum of the 2x2 outer product tensor of the gradient vector

$$\left(\sum \nabla I \nabla I^T \right) \vec{U} = -\sum \nabla I I_t$$

Lukas-Kanade

$$\text{Let} \quad M = \sum (\nabla I)(\nabla I)^T \qquad \text{and} \qquad b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

- So, the optical flow U = (u,v) can be written as

$$MU=b$$

- And optical flow is just $U=M^{-1}b$

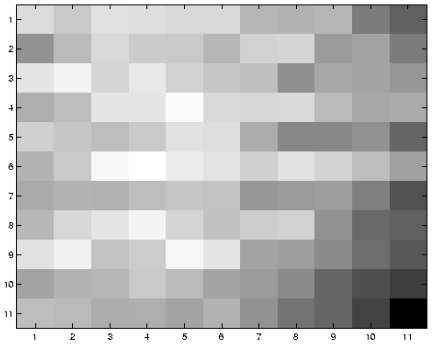
Lukas-Kanade: Singularities & Aperture Problem

Let $M = \sum (\nabla I)(\nabla I)^T$ and $b = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$

- Algorithm: At each pixel compute U by solving $MU = b$

- M is singular if
  - constant brightness in image: $\nabla I = 0$
  - Window is one pixel
  - Along an edge (where the direction of ∇I is the same (or zero) in the window)
    - Aperature problem still exists
- M is full rank for regions of bidirectional texturedness (e.g., corners)

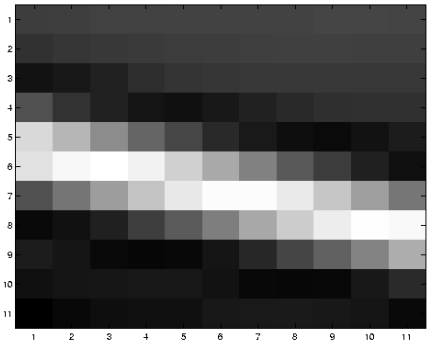
Low texture region



$$M = \sum (\nabla I)(\nabla I)^T$$

– Eigenvalues of M: small λ_1, small λ_2

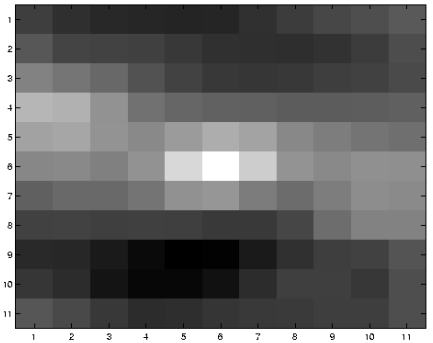
Edge



$$M = \sum (\nabla I)(\nabla I)^T$$

– Eigenvalues of M: large λ_1, small λ_2

High textured region



$$M = \sum (\nabla I)(\nabla I)^T$$

– Eigenvalues of M: large λ_1, large λ_2

Some variants

- Iterative refinement
- Coarse to fine (image pyramids)
- Local/global motion models
- Robust estimation

Revisiting the small motion assumption

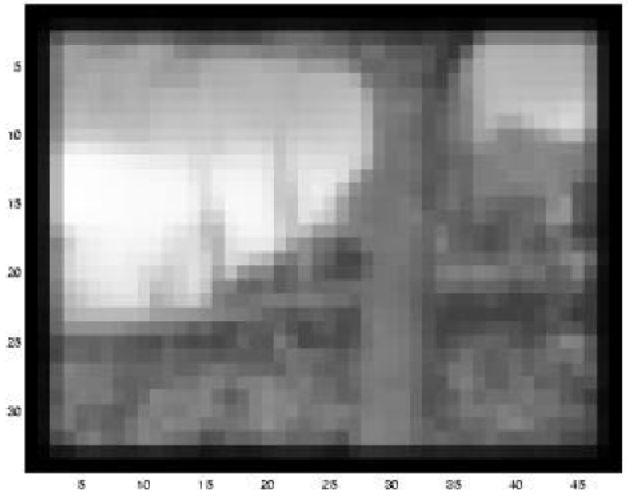
- Is this motion small enough?
  - Probably not—it's much larger than one pixel (2^{nd} order terms dominate)
  - How might we solve this problem?

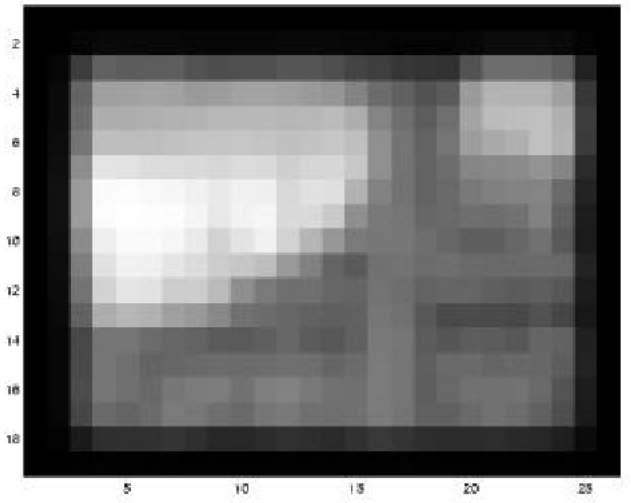
Iterative Refinement

- Estimate velocity at each pixel using one iteration of Lucas and Kanade estimation
- Warp one image toward the other using the estimated flow field

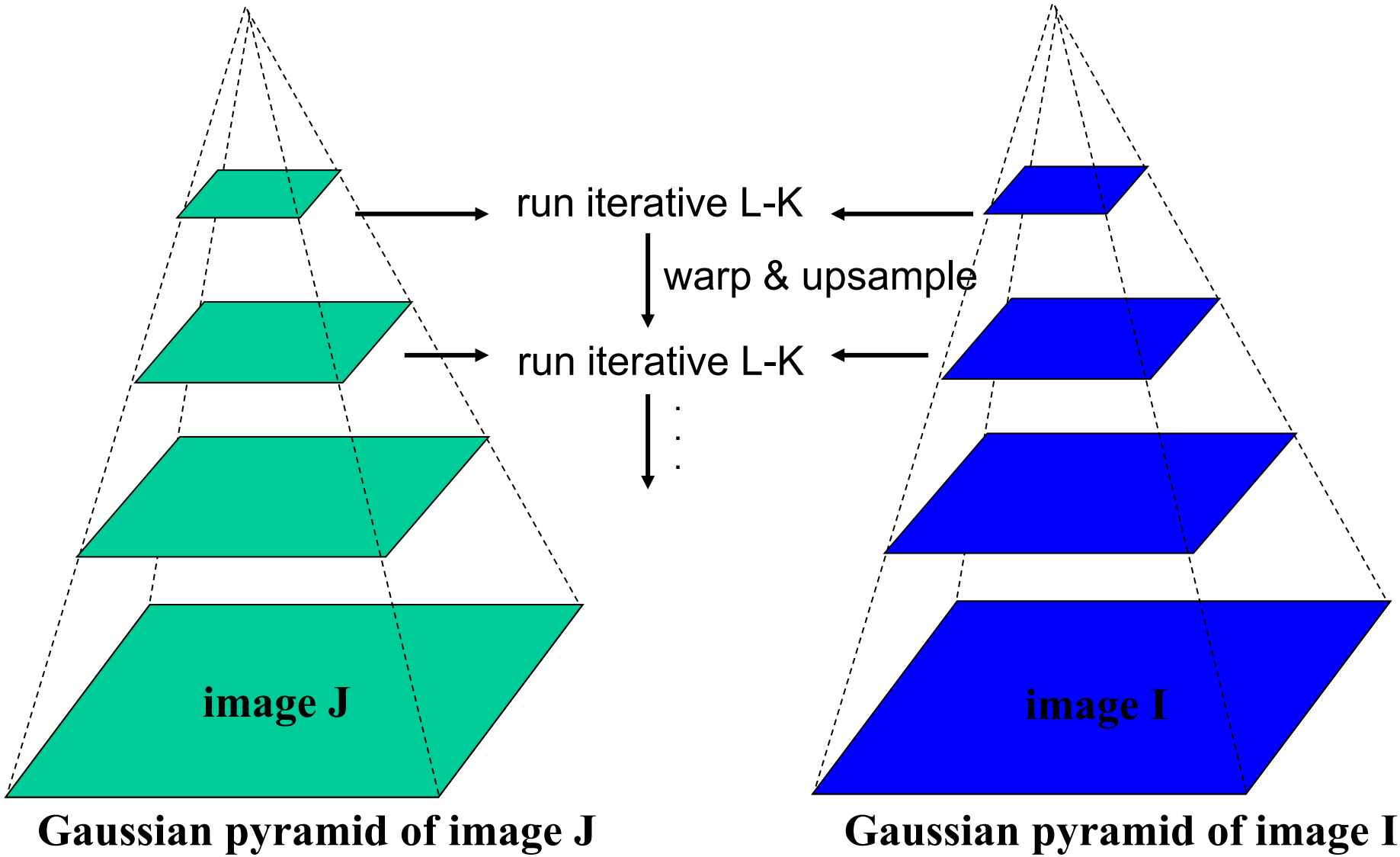
  (easier said than done)
- Refine estimate by repeating the process

Pyramid / “Coarse-to-fine”





Coarse-to-fine optical flow estimation



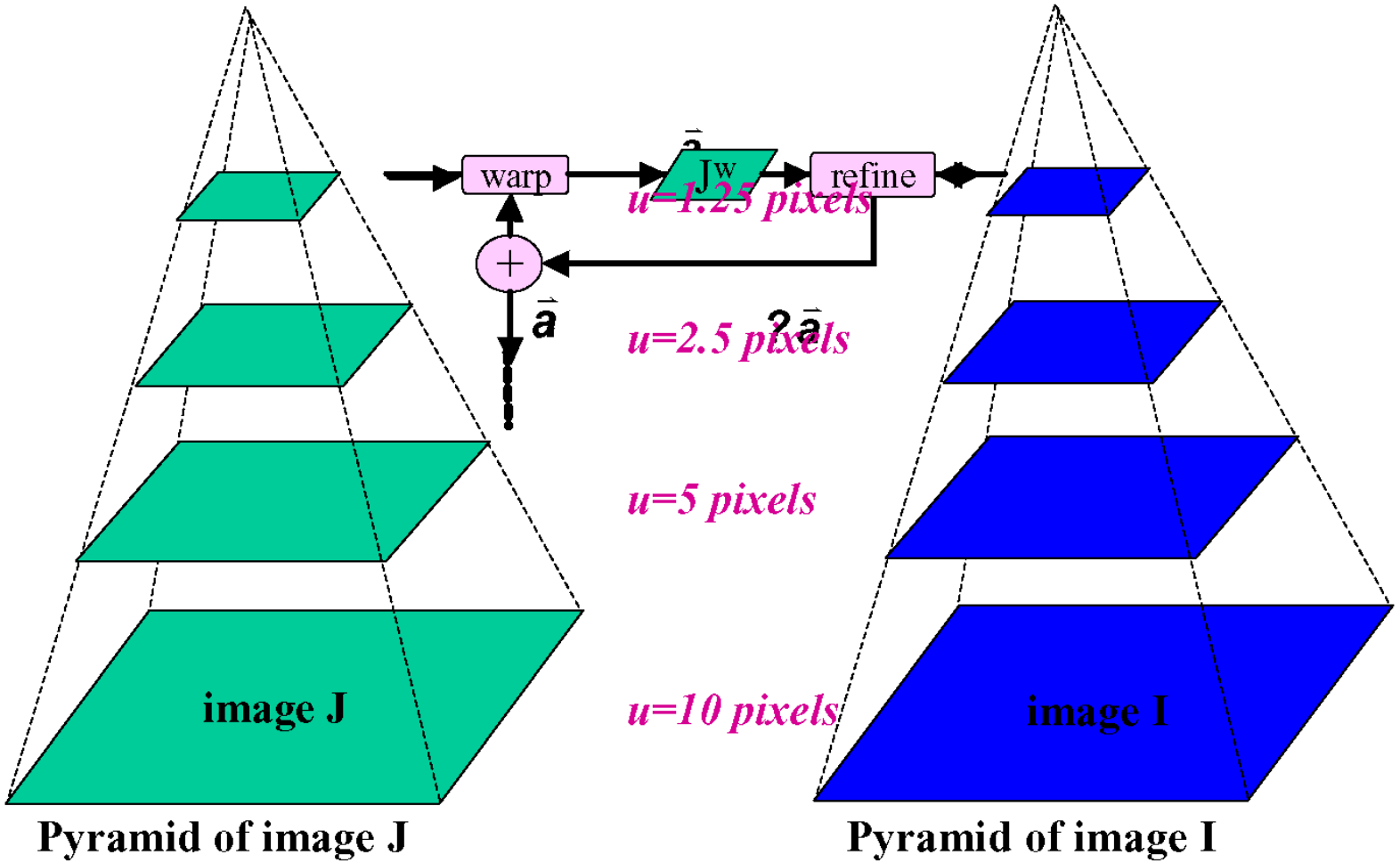
Gaussian pyramid of image J

Gaussian pyramid of image I

Coarse-to-Fine Estimation

$$I_x \cdot u + I_y \cdot v + I_t \approx 0$$

==> small *u* and *v* ...



Multi-resolution Lucas Kanade Algorithm

- Compute 'simple' LK at highest level
- At level i
  - Take flow u_{i-1}, v_{i-1} from level i-1
  - bilinear interpolate it to create u_i^*, v_i^* matrices of twice resolution for level i
  - multiply u_i^*, v_i^* by 2
  - compute f_t from a block displaced by $u_i^*(x,y)$, $v_i^*(x,y)$
  - Apply LK to get $u_i'(x, y)$, $v_i'(x, y)$ (the correction in flow)
  - Add corrections u_i' v_i', *i.e.* $u_i = u_i^* + u_i'$, $v_i = v_i^* + v_i'$.

Parametric (Global) Motion Models

2D Models:

(Translation)

Affine

Quadratic

Planar projective transform (Homography)

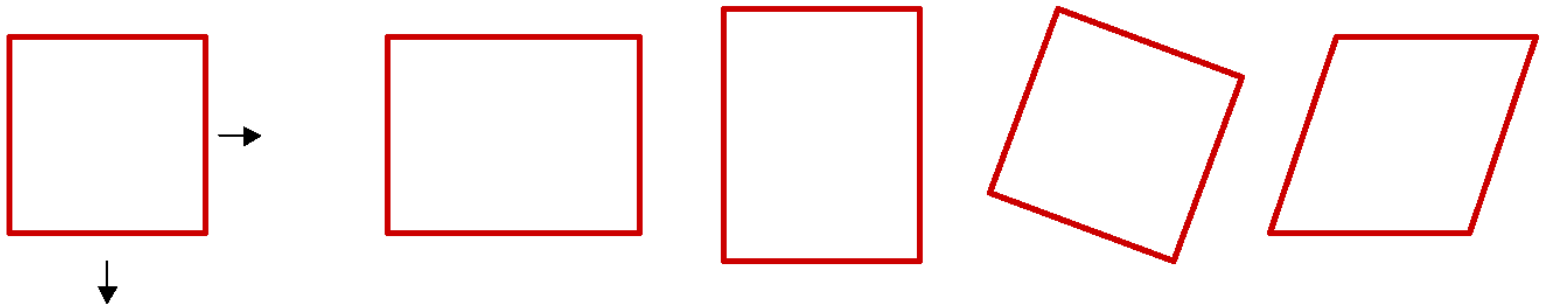
3D Models:

Instantaneous camera motion models

Homography+epipole

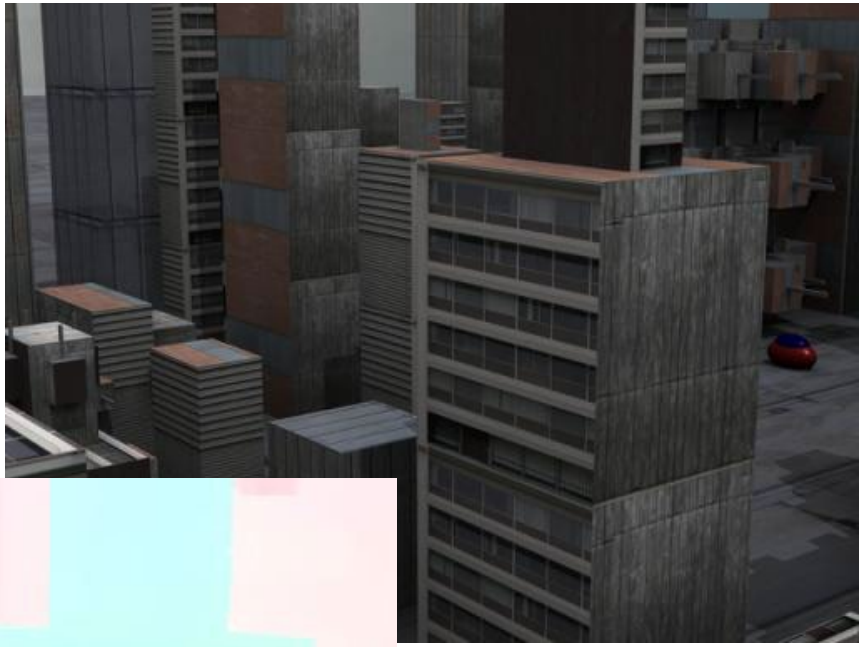
Plane+Parallax

Motion Model Example: Affine Motion

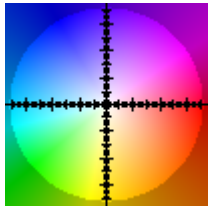


Affine: $\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\mathbf{h} = \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$

Optical flow



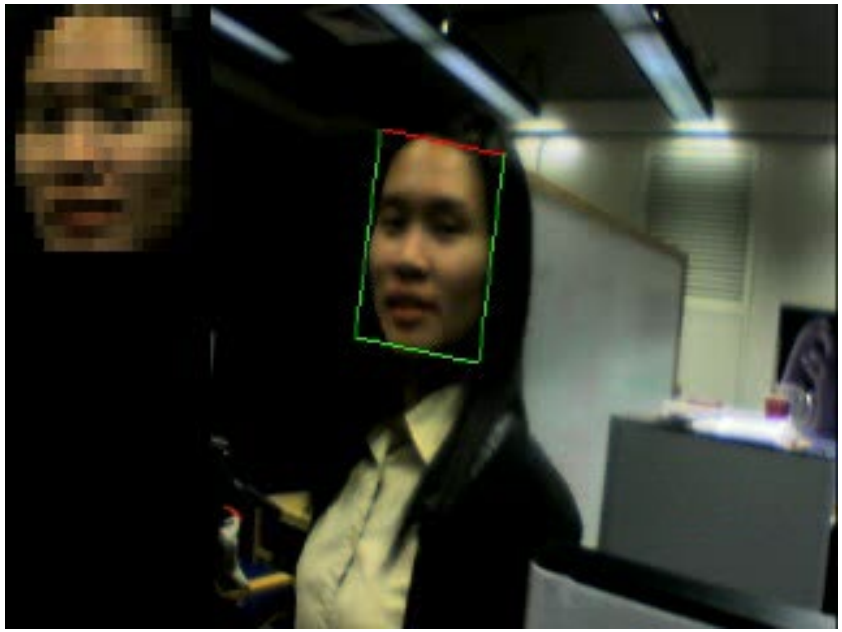
Deep neural networks have become an essential component of high performance dense optical flow algorithms



Color encoding of flow vectors

Visual Tracking

Optical flow is pixel-level tracking.
Now we consider tracking objects



Main Challenges

1. 3D pose variation
2. Target occlusion
3. Illumination variation
4. Camera jitter
5. Expression variation

   etc.

[Ho, Lee, Kriegman]

Main tracking notions

- **State $\boldsymbol{\phi}(t)$**: usually a finite number of parameters (a vector) that characterizes the "state" (e.g., location, size, pose, deformation of thing being tracked.
- **Dynamics $\dot{\boldsymbol{\phi}}(t)$**: How does the state change over time? How is that changed constrained?
- **Prediction**: Given the state $\boldsymbol{\phi}(t)$ at time *t,* what is an estimate $\boldsymbol{\phi}_p(t+1)$? Use $\boldsymbol{\phi}(t)$ and $\dot{\boldsymbol{\phi}}(t)$.
- **Representation**: How do you represent the thing being tracked
- **Data Association:** Which measurements correspond to which object?
- **Correction**: Given the predicted state $\boldsymbol{\phi}_p(t+1)$ at time *t+1*, and a measurement at time *t+1*, update the state $\boldsymbol{\phi}_c(\mathrm{t+1}) = \mathrm{f}(\boldsymbol{\phi}_p(t+1), \mathrm{M(t+1)})$
- **Initialization** – what is the state at time t=0?

Tracking by detection

- Example: Structured Output Tracking with Kernels

https://youtu.be/gnT34hJwdjM

Next Lectures

- Recognition, detection, and classification