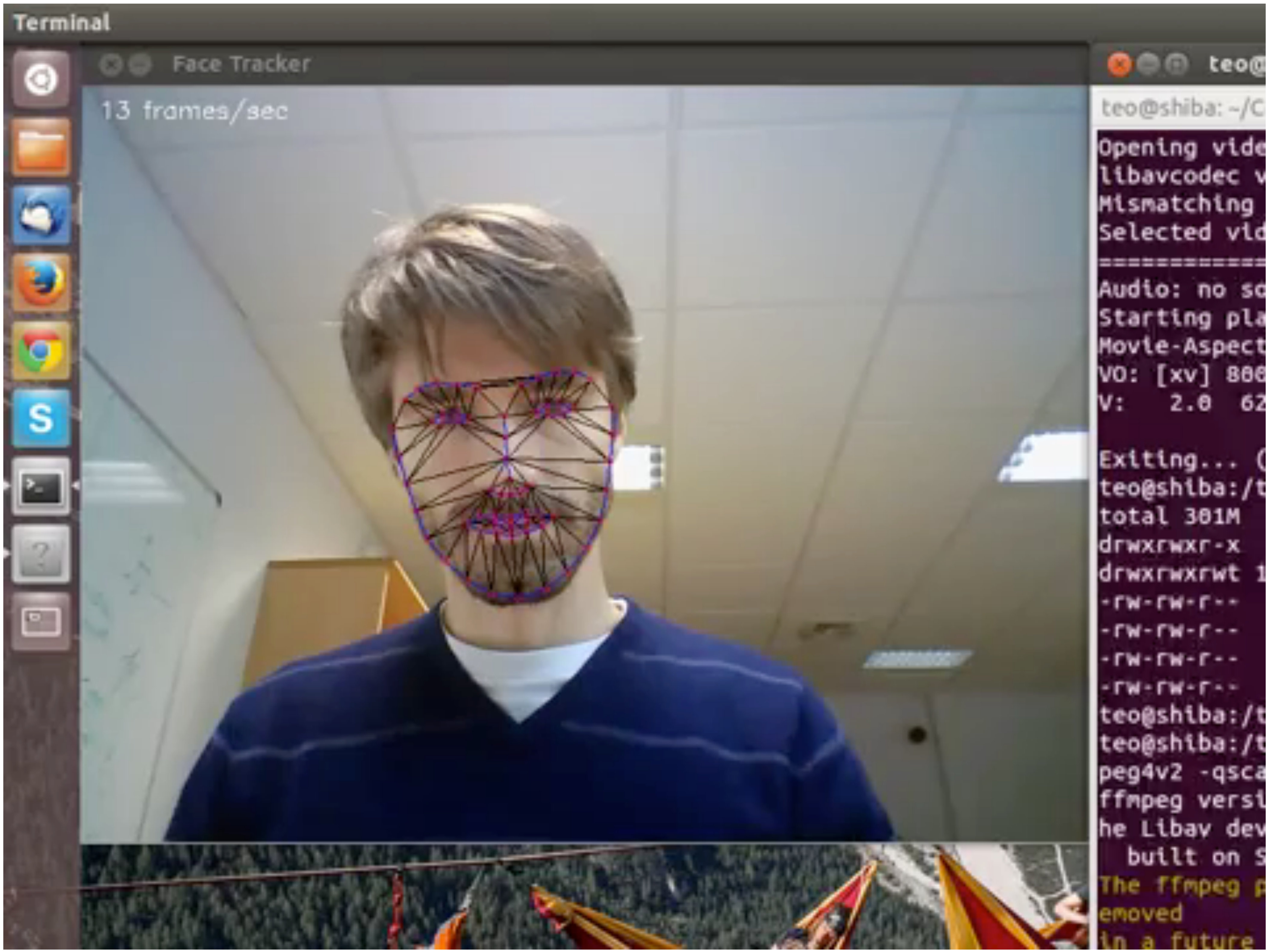
Motion and flow

Outline

- Lucas Kanade
- Moving cameras (egomotion)
- Estimating flow

Peicewise affine-tracking



Nonlinear least squares

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x}) \right]^2$$

$$I(\mathbf{x};p) = I(\mathbf{W}(\mathbf{x};p))$$

shorthand notation

$$I(\mathbf{x};\tilde{\mathbf{p}}+\boldsymbol{\Delta}\mathbf{p}) \approx I(\mathbf{x};\tilde{\mathbf{p}}) + \left[\frac{\partial I(\mathbf{x},\tilde{\mathbf{p}})}{\partial x} \quad \frac{\partial I(\mathbf{x},\tilde{\mathbf{p}})}{\partial y}\right] \left[\begin{matrix} \frac{\partial W_x(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}} \\ \frac{\partial W_y(\mathbf{x},\mathbf{p})}{\partial \mathbf{p}} \end{matrix}\right]_{\tilde{\mathbf{p}}} \Delta\mathbf{p}$$

current warped image

current warped image gradient

jacobian matrix

parameter update vector

Nonlinear least squares (cont'd)

$$\sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x}) \right]^2 \approx \sum_{\mathbf{x}} \left[I(\mathbf{W}(\mathbf{x};\mathbf{p})) + \nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - T(\mathbf{x}) \right]^2$$

Set derivative of above (wrt delta p) = 0

Error Image

$$\Delta\mathbf{p} = \sum_{\mathbf{x}} H^{-1} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p})) \right]$$

Gradient

(Approx) Hessian

$$H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]^{\mathrm{T}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \right]$$

Jacobian

Example: jacobian of affine warp

affine warp function (6 parameters)

$$W([x,y];P) = \begin{pmatrix} 1+p_1 & p_3 & p_5 \\ p_2 & 1+p_4 & p_6 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \Rightarrow \frac{\partial W}{\partial P} = \frac{\partial \begin{bmatrix} x + xP_1 + yP_3 + P_5 \\ xP_2 + y + yP_4 + P_6 \end{bmatrix}}{\partial P}$$

$$= \begin{bmatrix} x & 0 & y & 0 & 1 & 0 \\ 0 & x & 0 & y & 0 & 1 \end{bmatrix}$$

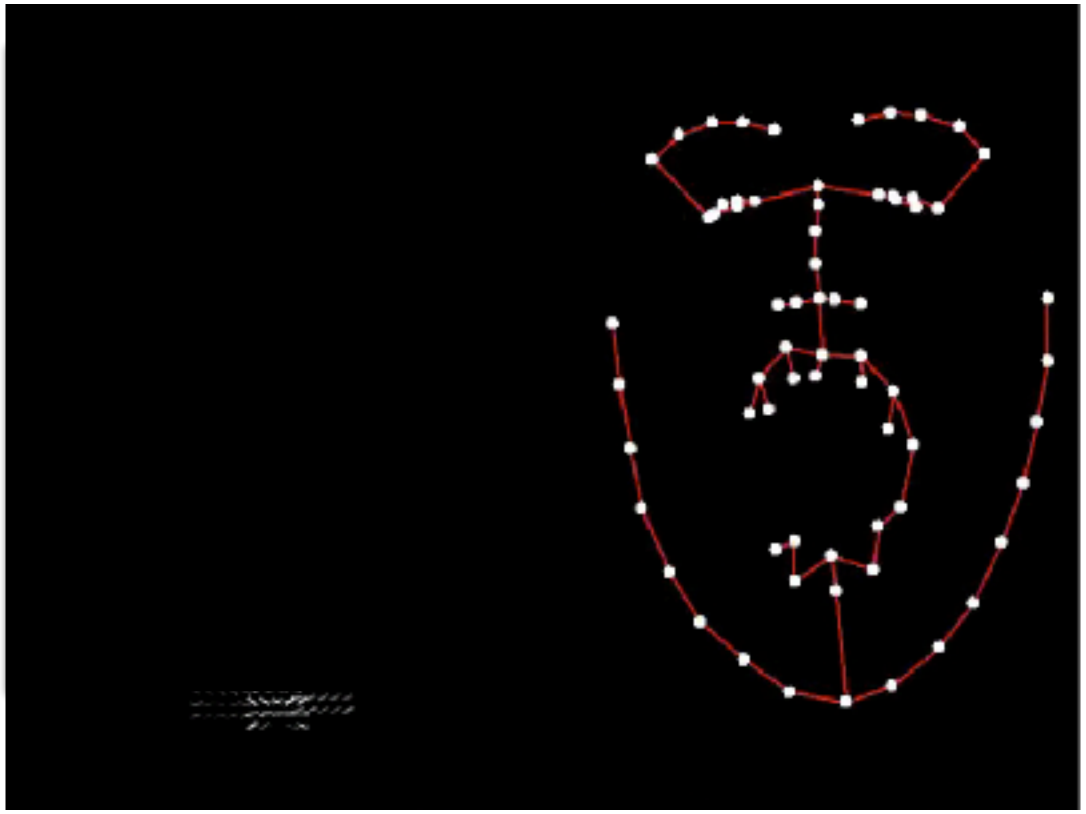
Notes:

1) Above parameterization is better conditioned because all-zero parameters defaults to identity

2) Jacobian matrix is a function of (x,y) coordinate

Peicewise warps

$$W(\mathbf{x}; p)$$



$$\mathbf{s} = \begin{bmatrix} x_1 & y_2 & x_2 & y_2 \dots \end{bmatrix}$$

$$\mathbf{s} = \mathbf{s}_0 + p_1\mathbf{s}_1 + p_2\mathbf{s}_2 + \dots$$

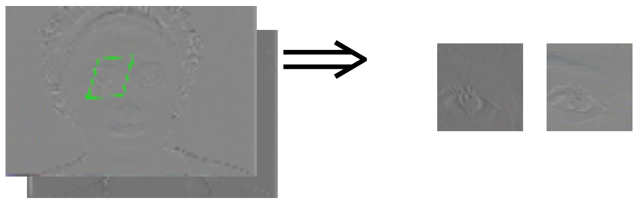
$$Pr(\mathbf{s}) = N(\mathbf{s}; \mu, \Sigma)$$

Lucas-Kanade Algorithm

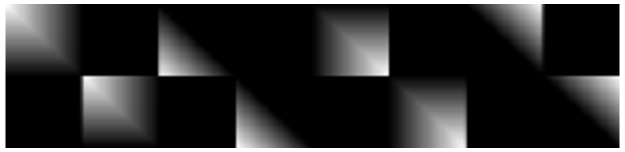
1. Warp I with $\mathbf{W}(\mathbf{x};\mathbf{p}) \Rightarrow I(\mathbf{W}(\mathbf{x};\mathbf{p}))$

2. Compute error image $T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))$

3. Warp gradient of I to compute ∇I

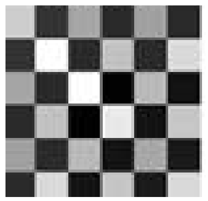


4. Evaluate Jacobian $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$

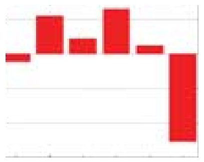


5. Compute steepest descent images $\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}$

6. Compute Hessian $H = \sum_{\mathbf{x}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$



7. Compute updates $\Delta \mathbf{p} = \sum_{\mathbf{x}} H^{-1} \left[\nabla I \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$



8. Update parameters $\mathbf{p} \leftarrow \mathbf{p} + \Delta \mathbf{p}$

Detailed reference

Lucas-Kanade 20 Years On: A Unifying Framework

SIMON BAKER AND IAIN MATTHEWS
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
simonb@cs.cmu.edu
iainm@cs.cmu.edu

IJCV 2004

Variations

Additive warp: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x})]^2$ $\quad\quad$ $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}.$

Variations

Additive warp: $$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x})]^2$$ $$\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}.$$

Compositional warps: $$\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{W}(\mathbf{x}; \Delta\mathbf{p}); \mathbf{p})) - T(\mathbf{x})]^2$$

$$\mathbf{W}(\mathbf{x}; \mathbf{p}) = \begin{pmatrix} (1 + p_1) \cdot x & + & p_3 \cdot y & + & p_5 \\ p_2 \cdot x & + & (1 + p_4) \cdot y & + & p_6 \end{pmatrix}$$

$$\mathbf{W}(\mathbf{x}; \mathbf{\Delta p}) = \begin{pmatrix} (1 + \Delta p_1) \cdot x & + & \Delta p_3 \cdot y & + & \Delta p_5 \\ \Delta p_2 \cdot x & + & (1 + \Delta p_4) \cdot y & + & \Delta p_6 \end{pmatrix}$$

Variations

Additive warp: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x}; \mathbf{p} + \Delta\mathbf{p})) - T(\mathbf{x})]^2$ $\qquad \mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}.$

Compositional warps: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{W}(\mathbf{x}; \Delta\mathbf{p}); \mathbf{p})) - T(\mathbf{x})]^2$

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & + & p_3 \cdot y & + & p_5 \\ p_2 \cdot x & + & (1+p_4)\cdot y & + & p_6 \end{pmatrix}$$

$$\mathbf{W}(\mathbf{x};\Delta\mathbf{p}) = \begin{pmatrix} (1+\Delta p_1)\cdot x & + & \Delta p_3 \cdot y & + & \Delta p_5 \\ \Delta p_2 \cdot x & + & (1+\Delta p_4)\cdot y & + & \Delta p_6 \end{pmatrix}$$

$$\mathbf{W}(\mathbf{W}(\mathbf{x}; \Delta\mathbf{p}); \mathbf{p})$$

$$= \begin{pmatrix} (1+p_1)\cdot((1+\Delta p_1)\cdot x + \Delta p_3 \cdot y + \Delta p_5) \\ \quad + p_3 \cdot (\Delta p_2 \cdot x + (1+\Delta p_4)\cdot y + \Delta p_6) \\ \quad + p_5 \\ p_2 \cdot ((1+\Delta p_1)\cdot x + \Delta p_3 \cdot y + \Delta p_5) \\ \quad + (1+p_4)\cdot(\Delta p_2 \cdot x + (1+\Delta p_4)\cdot y \\ \quad + \Delta p_6) + p_6 \end{pmatrix}$$

Work out Taylor expansion; it turns out Jacobian is evaluated at $\Delta\mathbf{p} = 0$, which means it can be precomputed

Variations

Additive warp: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x})]^2$ $\quad\quad$ $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}.$

Notation:

Compositional warps: $\sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{W}(\mathbf{x};\Delta\mathbf{p});\mathbf{p})) - T(\mathbf{x})]^2$ $\quad\quad$ $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p}),$

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot x & + & p_3 \cdot y & + & p_5 \\ p_2 \cdot x & + & (1+p_4)\cdot y & + & p_6 \end{pmatrix}$$

$$\mathbf{W}(\mathbf{x};\mathbf{\Delta p}) = \begin{pmatrix} (1+\Delta p_1)\cdot x & + & \Delta p_3 \cdot y & + & \Delta p_5 \\ \Delta p_2 \cdot x & + & (1+\Delta p_4)\cdot y & + & \Delta p_6 \end{pmatrix}$$

$$\mathbf{W}(\mathbf{W}(\mathbf{x};\Delta\mathbf{p});\mathbf{p}) = \begin{pmatrix} (1+p_1)\cdot((1+\Delta p_1)\cdot x + \Delta p_3 \cdot y + \Delta p_5) \\ \quad + p_3 \cdot (\Delta p_2 \cdot x + (1+\Delta p_4)\cdot y + \Delta p_6) \\ \quad + p_5 \\ p_2 \cdot ((1+\Delta p_1)\cdot x + \Delta p_3 \cdot y + \Delta p_5) \\ \quad + (1+p_4)\cdot(\Delta p_2 \cdot x + (1+\Delta p_4)\cdot y \\ \quad + \Delta p_6) + p_6 \end{pmatrix}$$

Work out Taylor expansion; it turns out Jacobian is evaluated at $\Delta\mathbf{p} = 0$, which means it can be precomputed

Overview

Additive warp: $\sum_{\mathbf{x}}[I(\mathbf{W}(\mathbf{x};\mathbf{p}+\Delta\mathbf{p}))-T(\mathbf{x})]^2$ $\qquad$ $\mathbf{p} \leftarrow \mathbf{p}+\Delta\mathbf{p}.$

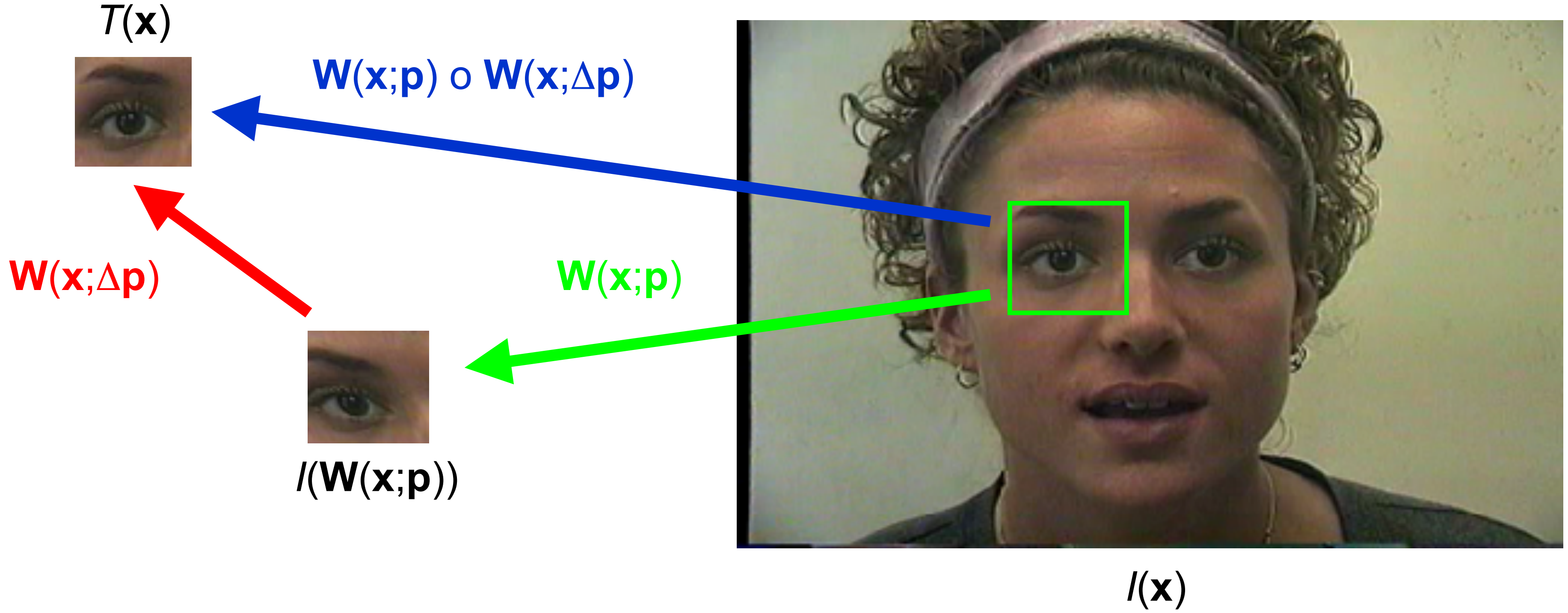
Compositional warp: $\sum_{\mathbf{x}}[I(\mathbf{W}(\mathbf{W}(\mathbf{x};\Delta\mathbf{p});\mathbf{p}))-T(\mathbf{x})]^2$ $\qquad$ $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p})\circ\mathbf{W}(\mathbf{x};\Delta\mathbf{p}),$

Inverse compositional warp: $\sum_{\mathbf{x}}[\,T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p}))-I(\mathbf{W}(\mathbf{x};\mathbf{p}))\,]^2$ $\qquad$ $\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p})\circ\mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}.$

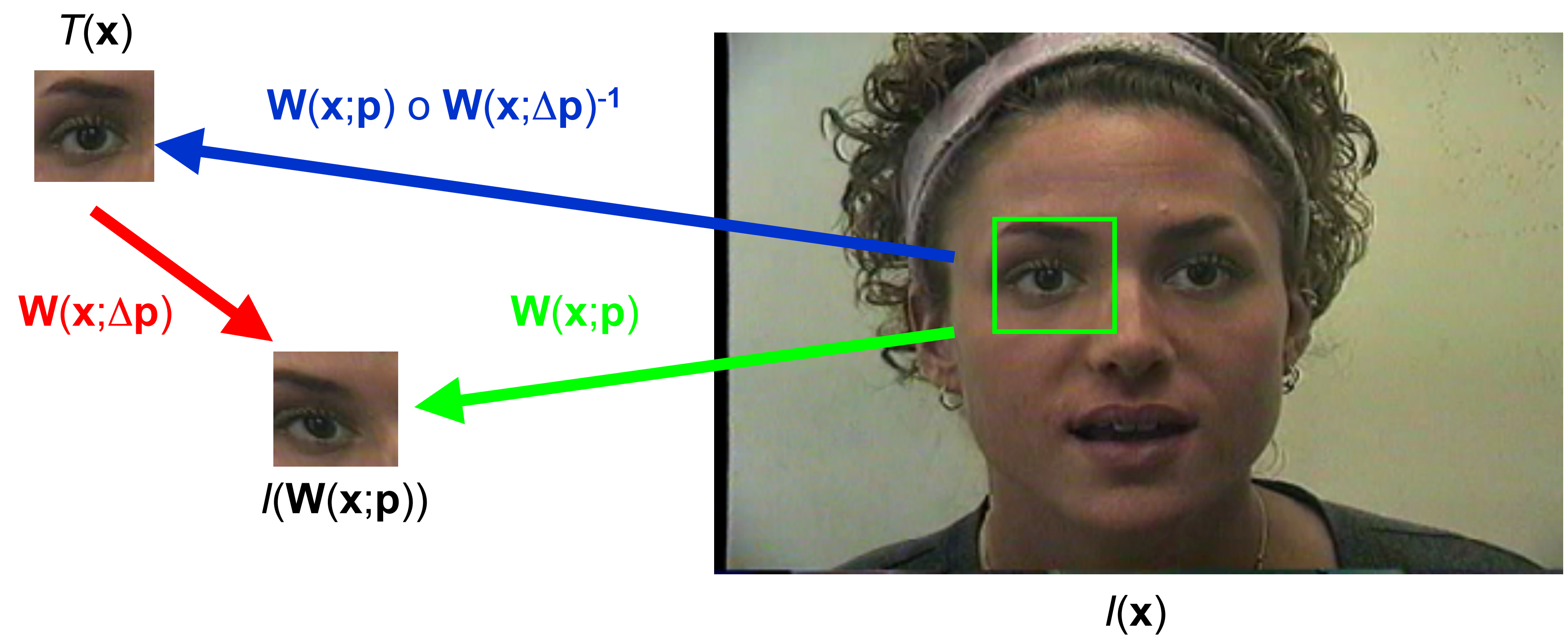
Work out Taylor expansion;
both Jacobian and Hessian are not a function of current **p** and so can be precomputed

Forward and Inverse Compositional

- Forwards compositional



- Inverse compositional



Inverse Compositional

- Minimise,

$$\sum_{\mathbf{x}} [T(\mathbf{W}(\mathbf{x};\Delta\mathbf{p})) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]^2 \approx \sum_{\mathbf{x}} \left[T(\mathbf{W}(\mathbf{x};\mathbf{0})) + \nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \Delta\mathbf{p} - I(\mathbf{W}(\mathbf{x};\mathbf{p}))\right]^2$$

- Solution

$$H = \sum_{\mathbf{x}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]$$

$$\Delta\mathbf{p} = -\sum_{\mathbf{x}} H^{-1} \left[\nabla T \frac{\partial \mathbf{W}}{\partial \mathbf{p}}\right]^{\mathrm{T}} [T(\mathbf{x}) - I(\mathbf{W}(\mathbf{x};\mathbf{p}))]$$

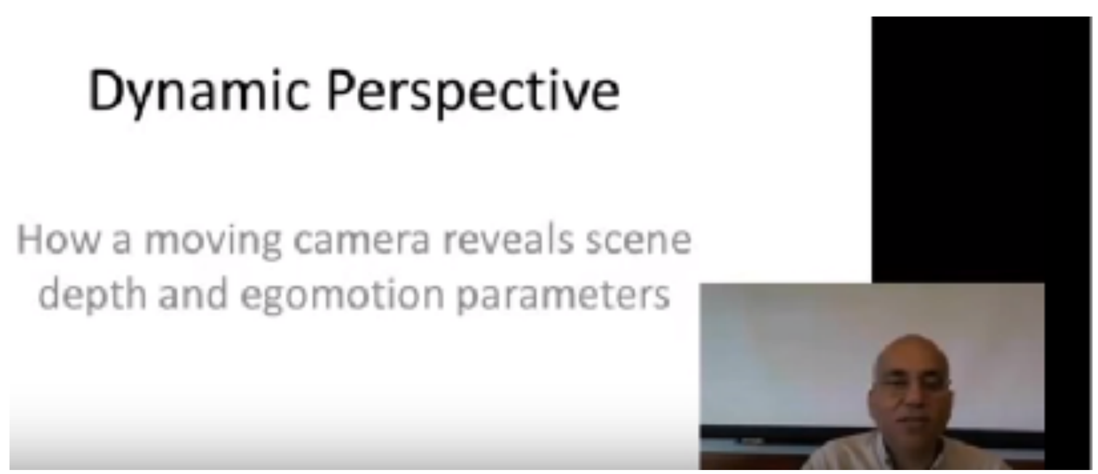
- Update

$$\mathbf{W}(\mathbf{x};\mathbf{p}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}) \circ \mathbf{W}(\mathbf{x};\Delta\mathbf{p})^{-1}$$

Crucial observation: we're always performing taylor expansion of template **@ the identity warp,** so precompute Jacobian, Steepest Descent Images, Hessian (everything but error image!)

Outline

- Lucas Kanade
- Moving cameras



https://www.youtube.com/watch?v=iz9UVIo_ZUo&list=PLc0IeyeoGt2xtmfaF2ST_uNdeptre3f9s&index=10

Moving is a part of life!

Sea squirt:

Starting off as an egg, the sea squirt quickly develops into a tadpole-like creature, complete with a spinal cord connected to a simple eye and a tail for swimming. It also has a primitive brain that helps it locomote through the water. But, the sea squirt's mobility doesn't last long. Once it finds a suitable place to attach itself, its brain is absorbed by its body. Being permanently attached to a home makes the sea squirt's spinal cord and the neurons that control locomotion superfluous. Once the sea squirt becomes stationary, it literally eats its own brain.

Human development:
Crawling teaches babies depth perception

After 1 week of crawling:

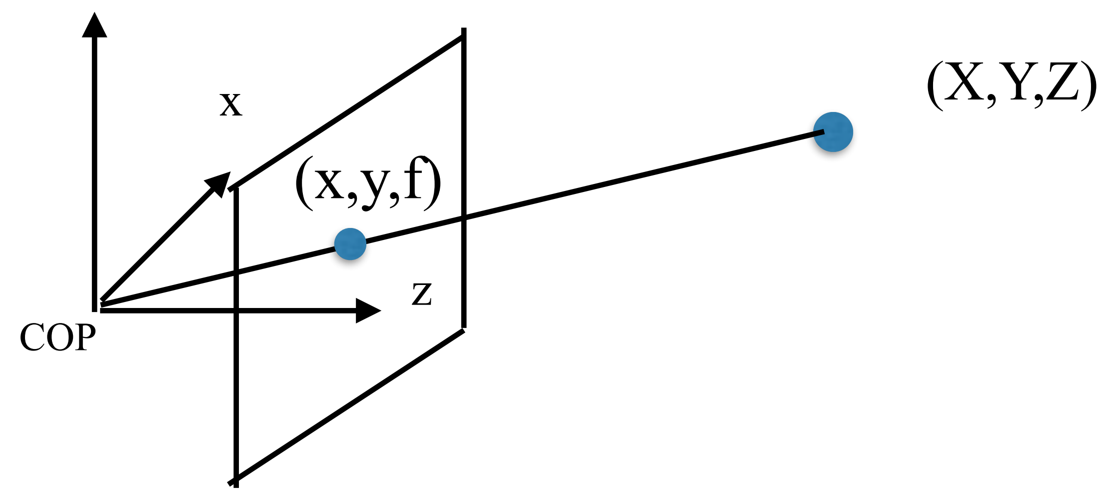
After several weeks of crawling:

Gibson, 1960s

Today's goal

Understand the space of image transformations that we see when we move

Recall: pinhole cameras



$$x = \frac{f}{Z}X$$

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

Assume calibrated cameras (f=1)

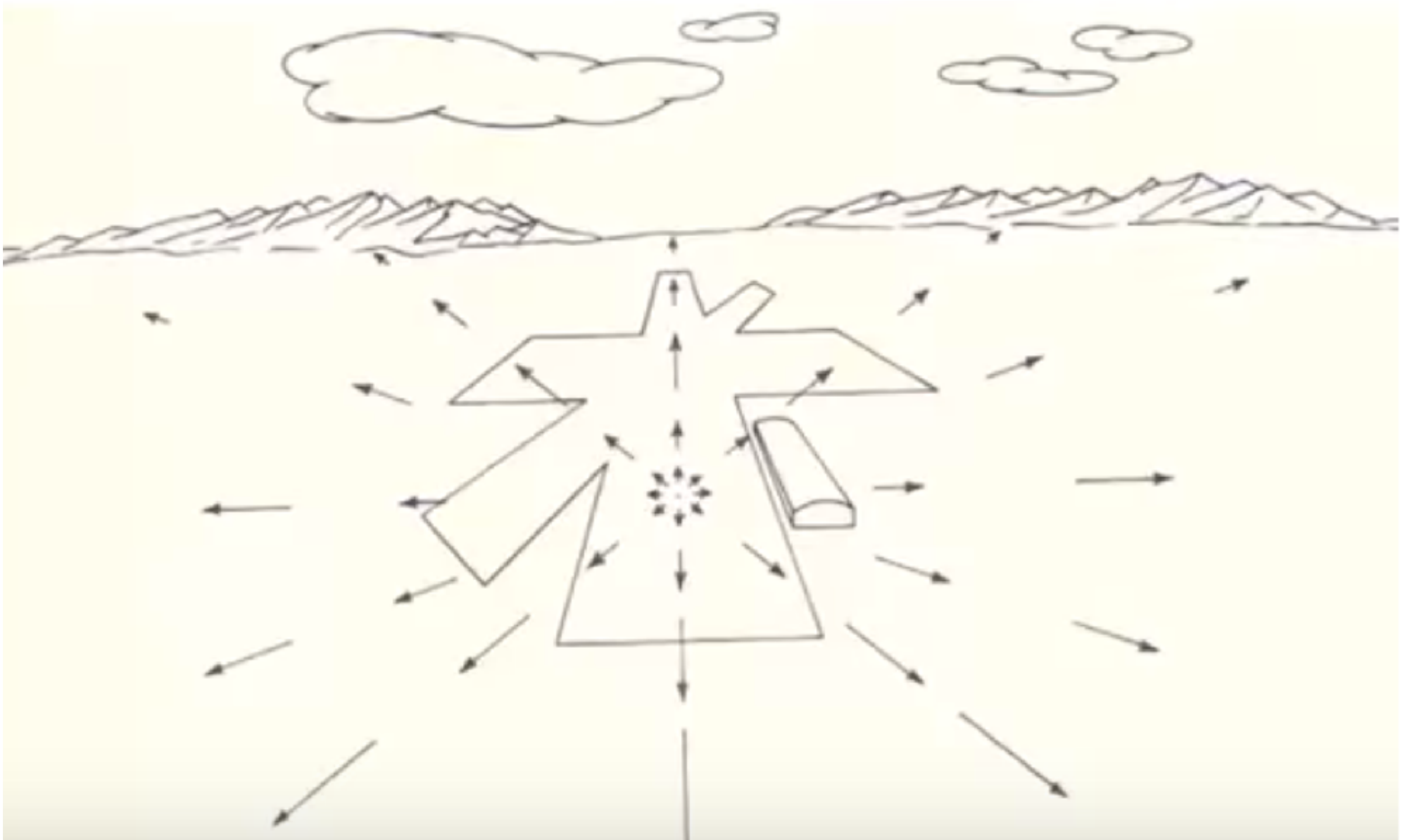
Suppose the camera moves with respect to the world…

- When a point (X,Y,Z) in the world moves relative to the camera, its projection in the image (x,y) moves as well.
- This movement in the image plane is called optical flow. Suppose the point (x,y) moves to $(x+\Delta x, y+\Delta y)$ in time Δt, then

$$u = \frac{\Delta x}{\Delta t}, v = \frac{\Delta y}{\Delta t}$$

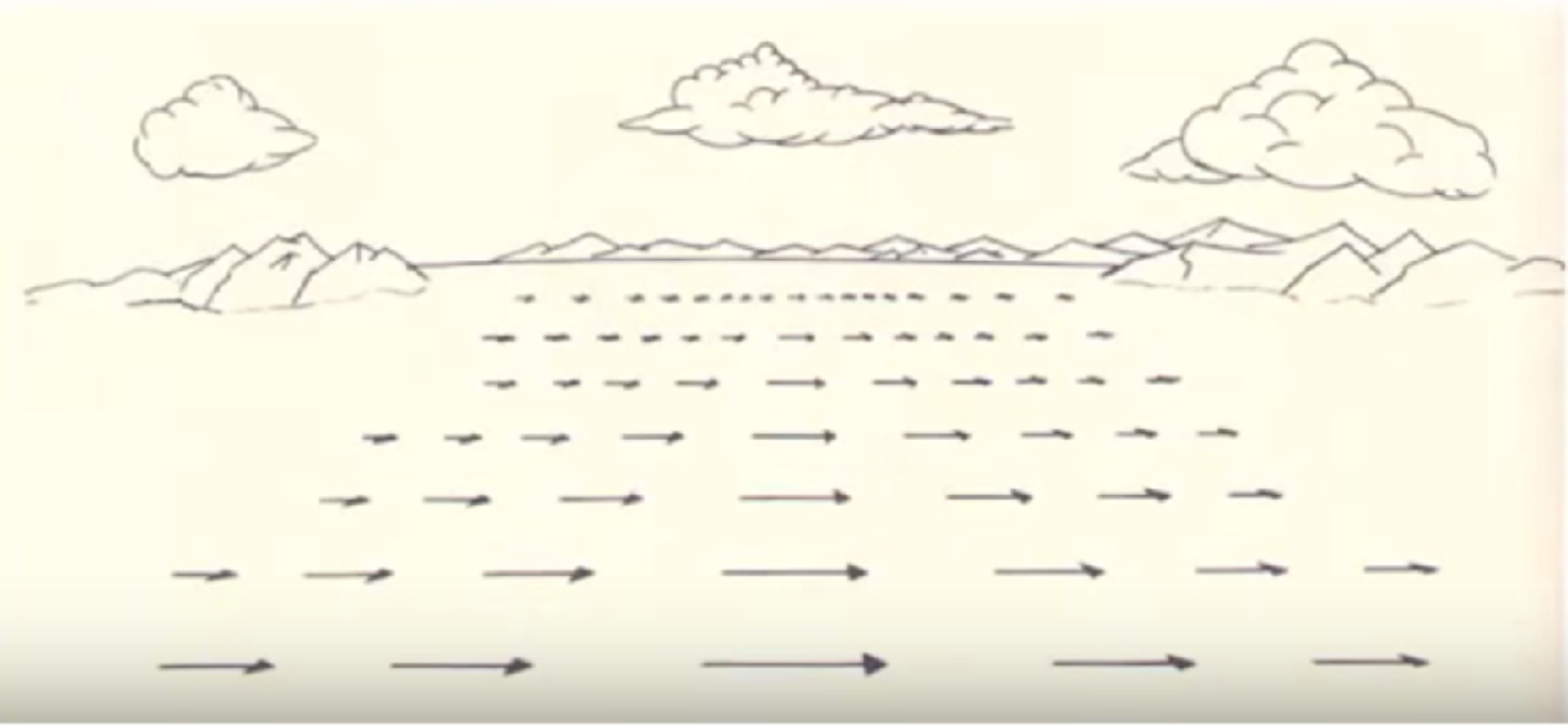
are the two components of the optical flow at (x,y)

Gibson's example 1: optical flow for a pilot landing a plane

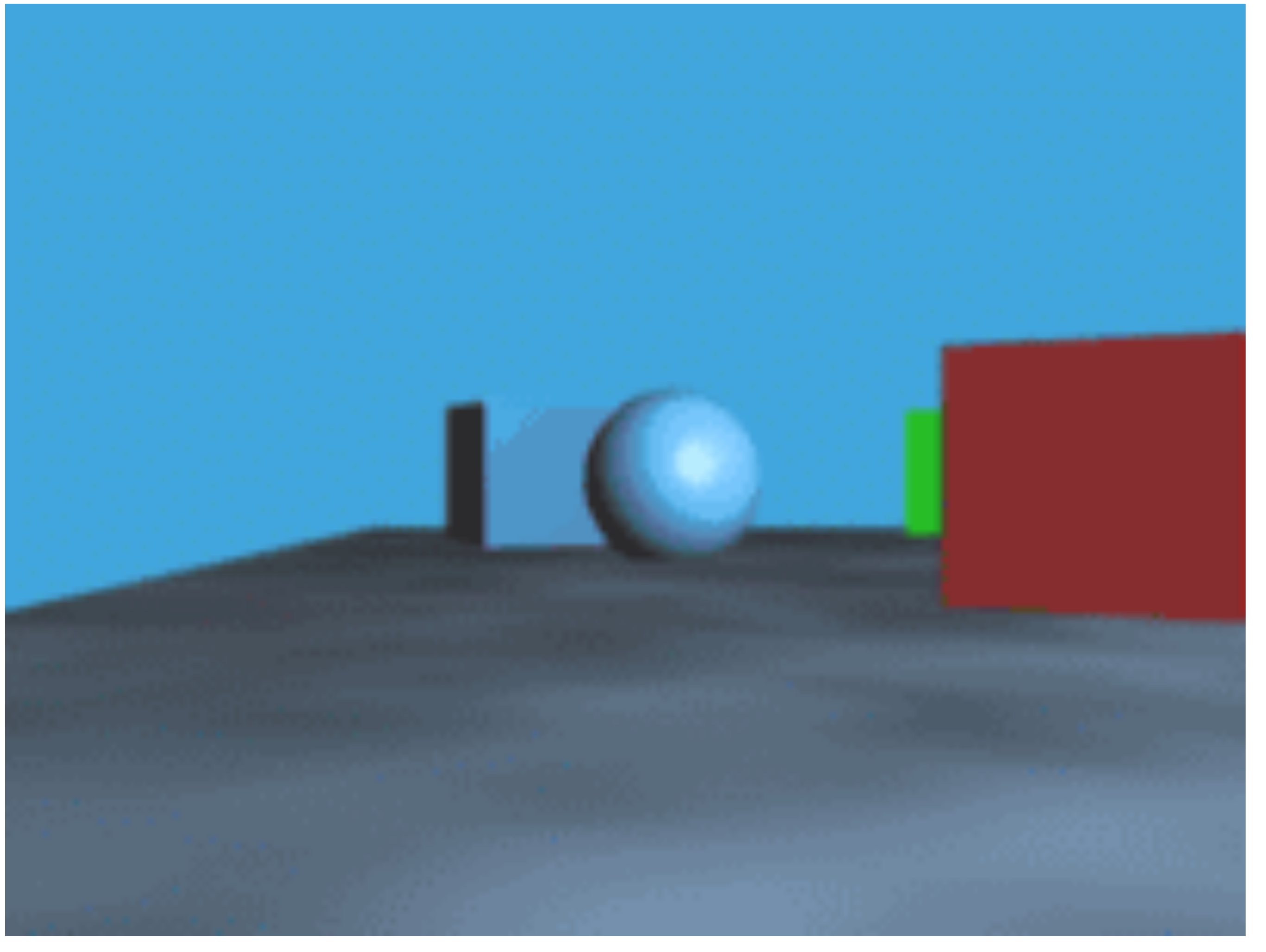


Optical flow is a vector feild (like a gradient map)

Gibson's example II:
Optical flow from the side window of a car



Parallax

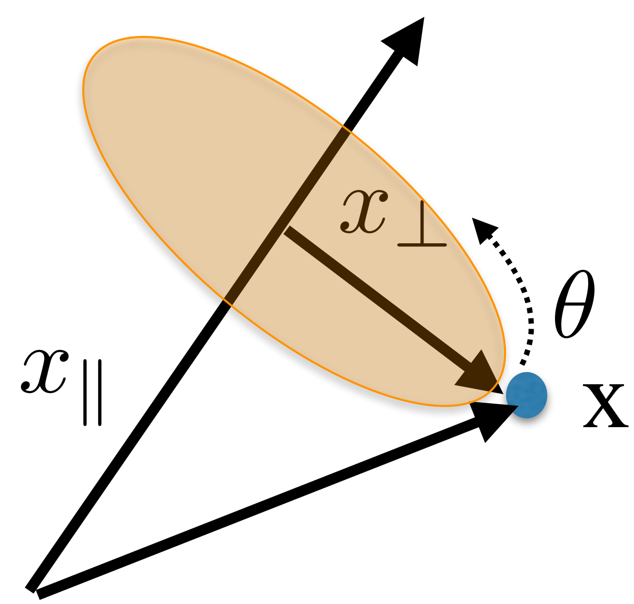


Outline

- Derive equation relating optical flow field to scene depth $Z(x,y)$ and the motion of the camera t, ω
- The translational component of the flow field is the more important one – it is what tells $Z(x,y)$ and the translation t
- The rotational component of the flow field reveals information about ω

Recall: 3D rotations

$$\omega \in R^3, \quad ||\omega|| = 1$$



$$R = I + \hat{w}\sin\theta + \hat{w}\hat{w}(1 - \cos\theta)$$ Rodrigues' formula

$$R = \exp(\hat{v}), \quad \text{where} \quad v = \omega\theta$$ (derive from above by Taylor series expansion of sine + cosine)

$$= I + \hat{v} + \frac{1}{2!}\hat{v}^2 + \ldots$$

Implication: we can approximate change in position due to a small rotation as $v \times x$.

Angular velocity

https://en.wikipedia.org/wiki/Angular_velocity

Notation switch: let's call the scaled direction vector as "angular velocity":

$$\omega, \quad ||\omega|| = \Delta\theta, \quad \mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

We can approximate change in position due to a small rotation as:

$$\dot{\mathbf{X}} = \omega \times \mathbf{X} = \hat{w}\mathbf{X}$$

Recall: exponentials are solutions of linear differential equations

$$\dot{x}(t) = ax(t)$$

$$x(t) = e^{at}x(0)$$

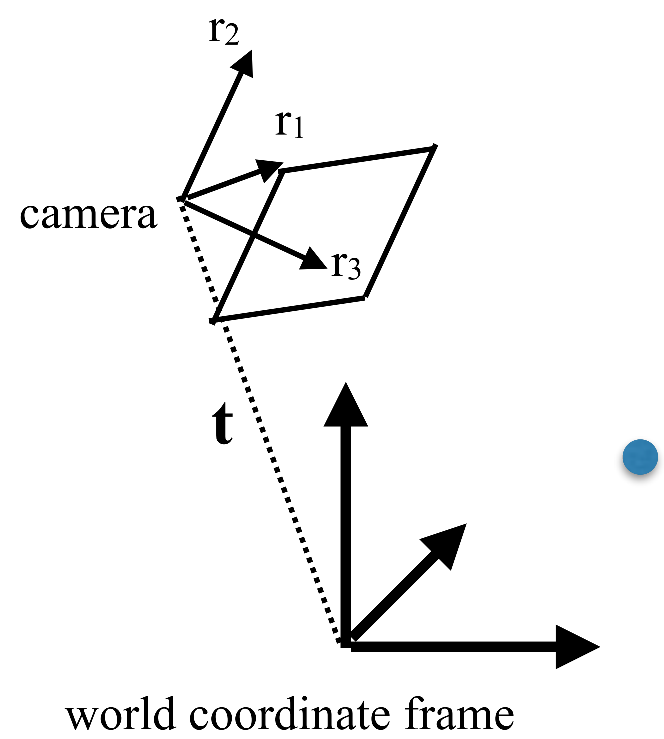
Matrix exponentials are solutions of matrix linear differential equations

$$\dot{\mathbf{X}}(t) = \hat{\omega}\mathbf{X}(t)$$

$$\mathbf{X}(t) = e^{\hat{\omega}t}\mathbf{X}(0)$$

Alternative derivation of an exponential map representation of rotations!

How does a fixed scene point **X** move wrt camera?



Let camera be moving at translational velocity of **t** and rotating with angular velocity ω

$$\dot{\mathbf{X}} = -\mathbf{t} - \omega \times \mathbf{X}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = -\begin{bmatrix} t_x \\ t_y \\ t_y \end{bmatrix} - \begin{bmatrix} \omega_y Z - \omega_z Y \\ \omega_z X - \omega_x Z \\ \omega_x Y - \omega_y X \end{bmatrix}$$

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = - \begin{bmatrix} t_x \\ t_y \\ t_y \end{bmatrix} - \begin{bmatrix} \omega_y Z - \omega_z Y \\ \omega_z X - \omega_x Z \\ \omega_x Y - \omega_y X \end{bmatrix}$$

If we assume f=1, x(t) = X(t)/Z(t) and y(t) = Y(t)/Z(t), (dx/dt, dy/dt) = ?

$$\dot{x} = \frac{\dot{X} Z - \dot{Z} X}{Z^2}, \quad \dot{y} = \frac{\dot{Y} Z - \dot{Z} Y}{Z^2}$$

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -1 & 0 & X \\ 0 & -1 & Y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} XY & -(1 + X^2) & Y \\ 1 + Y^2 & -XY & -X \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

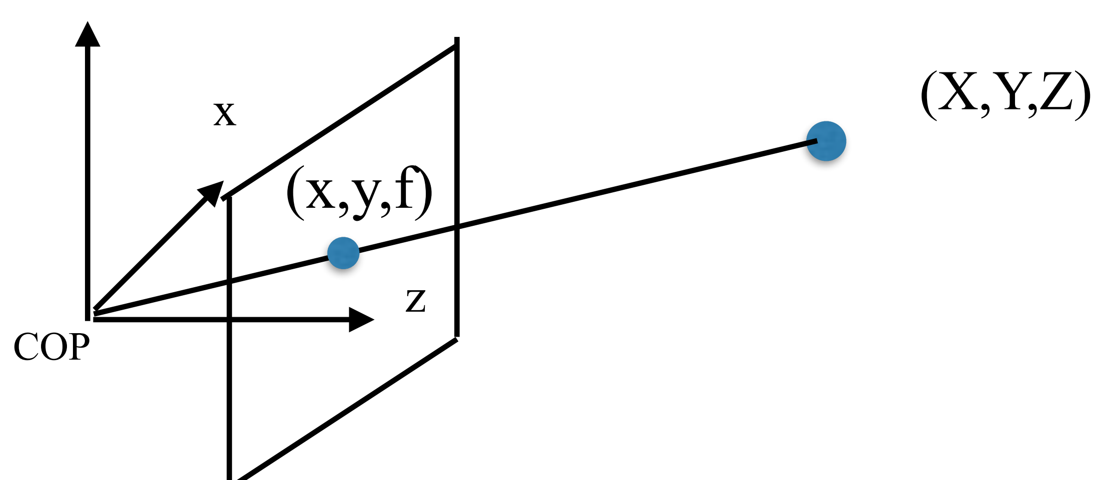
Egomotion optical flow

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z}\begin{bmatrix} -1 & 0 & X \\ 0 & -1 & Y \end{bmatrix}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} XY & -(1+X^2) & Y \\ 1+Y^2 & -XY & -X \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

translation component rotation component

$$u(x,y) = \frac{1}{Z(x,y)}(-t_x + Xt_z) + XY\omega_x - (1+X^2)\omega_y + Y\omega_z$$

$$v(x,y) = \frac{1}{Z(x,y)}(-t_y + Yt_z) + (1+Y^2)\omega_x - XY\omega_y - X\omega_z$$



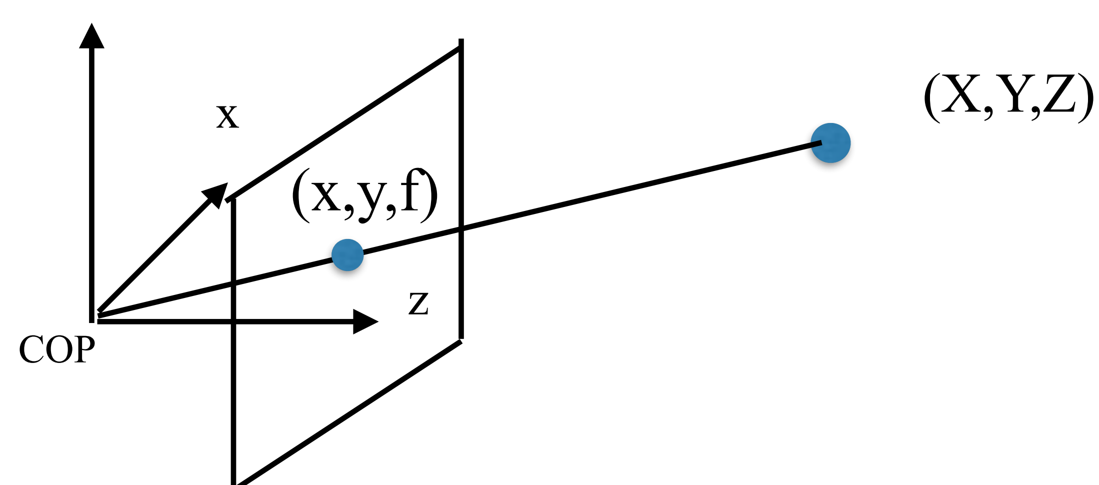
Optical flow for translation along camera's z-axis

If the motion of the camera is purely translational, the terms due to rotation in Eq. (3.4) can be dropped and the flow field becomes

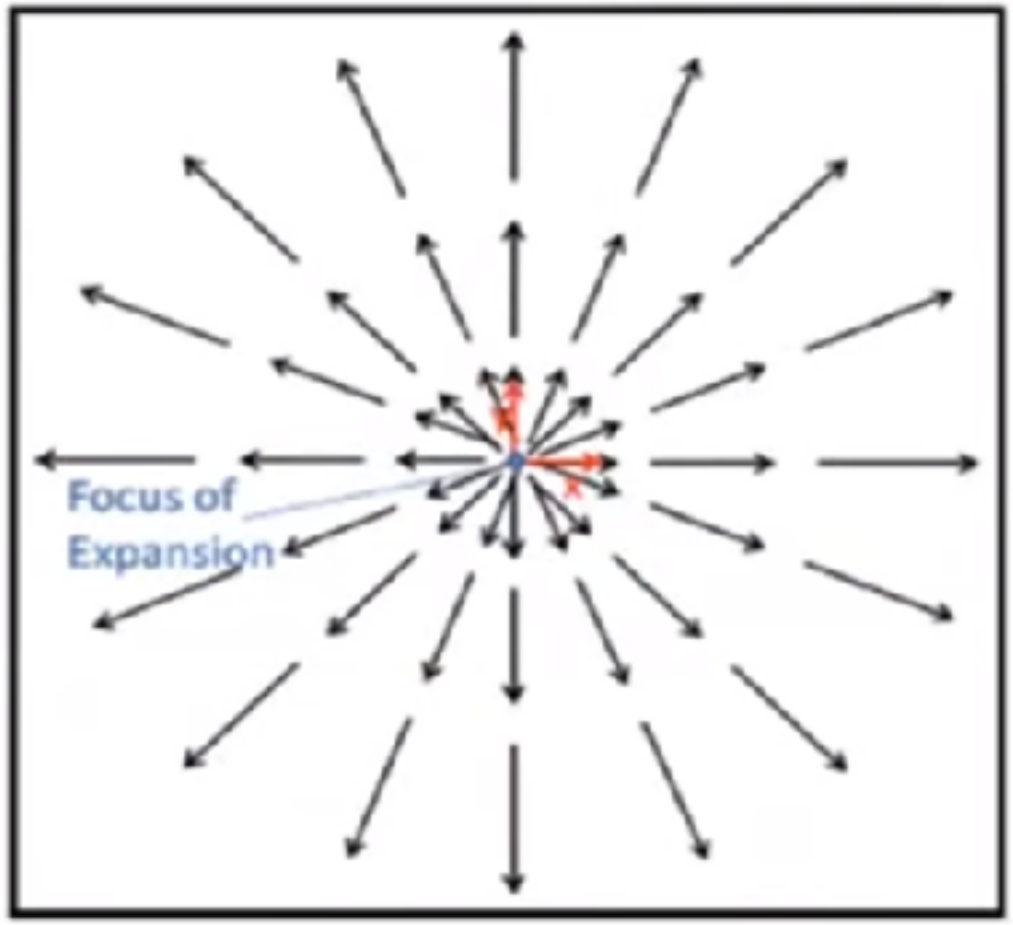
$$u(x,y) = \frac{-t_x + xt_z}{Z(x,y)}, v(x,y) = \frac{-t_y + yt_z}{Z(x,y)}. \tag{3.5}$$

We can gain intuition by considering the even more special case of translation along the optical axis, i.e. $t_z \neq 0, t_x = 0, t_y = 0$, the flow field in Eq.(3.5) becomes

$$u(x,y) = \frac{xt_z}{Z(x,y)}, v(x,y) = \frac{yt_z}{Z(x,y)}; \tag{3.6}$$



Optical flow for translation along camera's z-axis

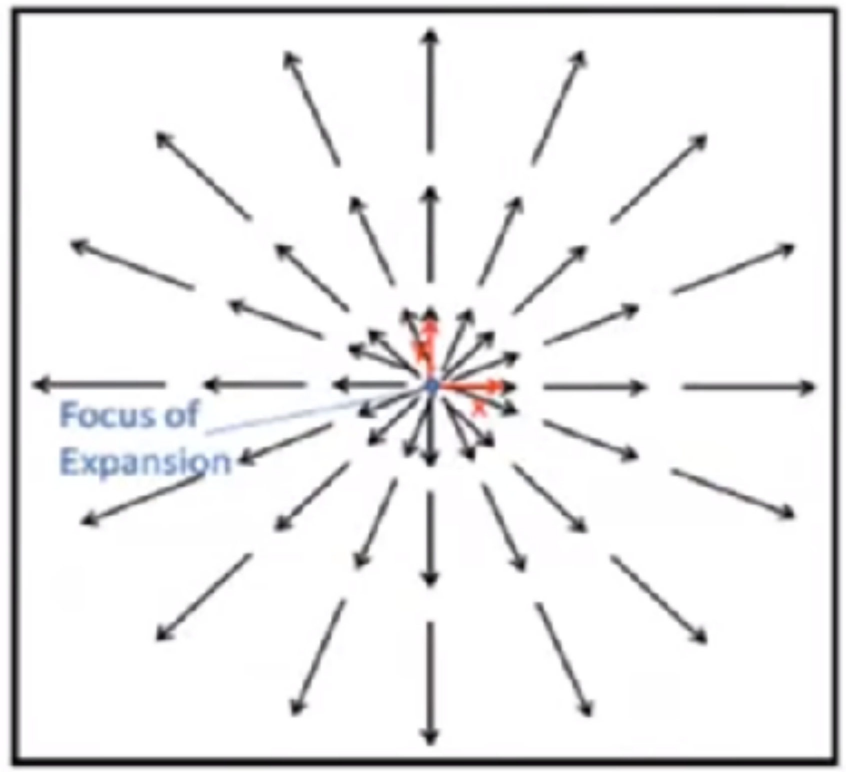


$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \frac{t_z}{Z(x,y)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{at origin}$$

Optical flow vector is a scalar multiple of position vector

(where scalar depends upon the depth of scene point)

Moving toward a wall



$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \frac{t_z}{Z} \begin{bmatrix} x \\ y \end{bmatrix}$$

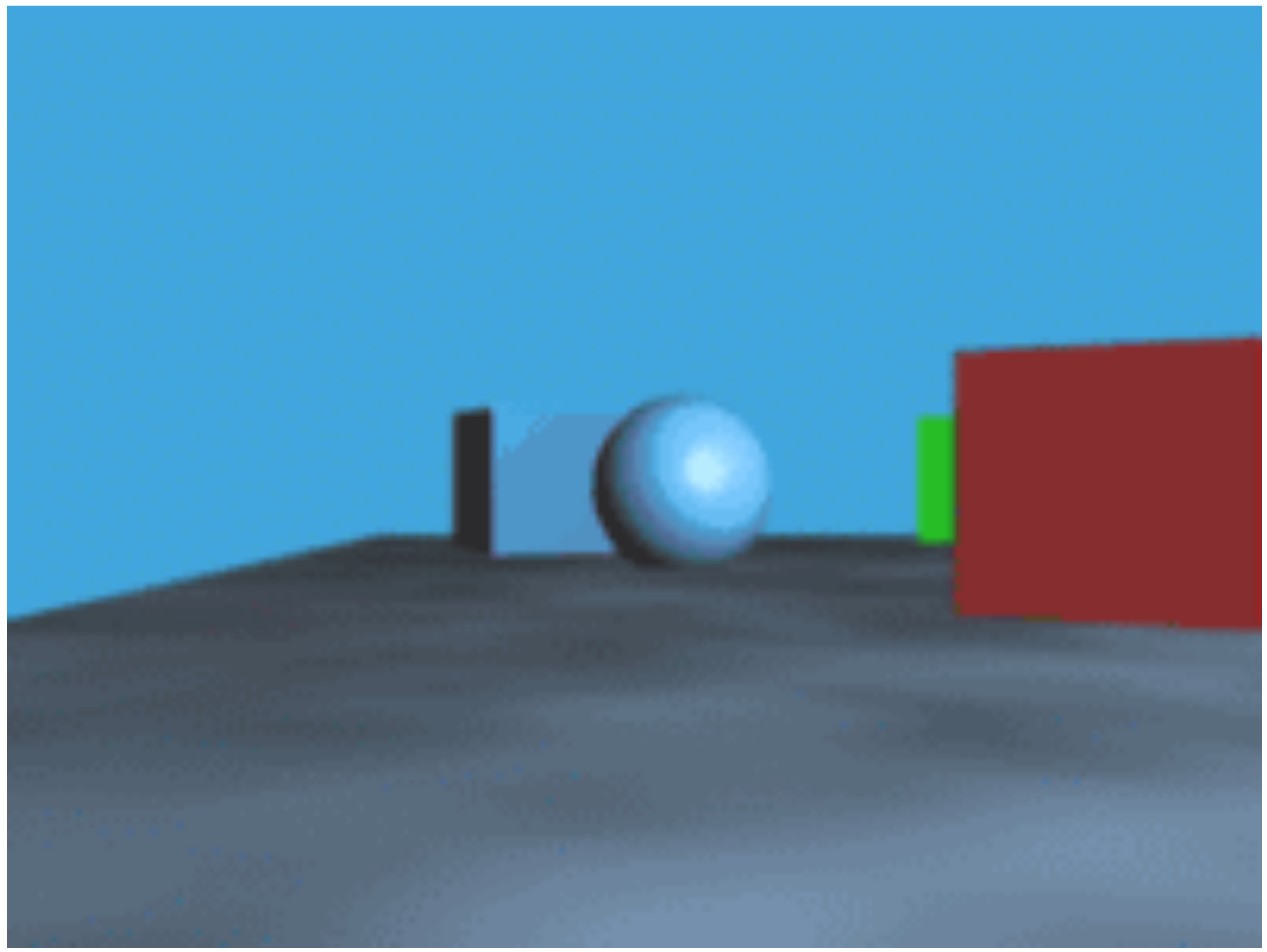
If speed (t_z) and distance (Z) to wall doubles, what happens to flow?

Nothing => fundamental scale ambiguity

If we are travelling 2 ft/sec, and the wall is 4 ft away, what's time-to-contact?

Time to contact = Z / t_z Implies time-to-contact (assuming constant depth) can be computed from 2D flow!

Optical flow for translation along camera's x-axis



$$u(x,y) = \frac{-t_x + xt_z}{Z(x,y)}, v(x,y) = \frac{-t_y + yt_z}{Z(x,y)}.$$

$v(x,y) = 0$

$u(x,y) = -t_x / Z(x,y)$

Optical flow for general translation

$$u(x, y) = \frac{-t_x + xt_z}{Z(x, y)}, v(x, y) = \frac{-t_y + yt_z}{Z(x, y)}.$$

When is this (0,0)?

Optical flow for general translation

$$u(x,y) = \frac{-t_x + xt_z}{Z(x,y)}, v(x,y) = \frac{-t_y + yt_z}{Z(x,y)}.$$

When is this (0,0)?

$$-t_x + xt_z = 0 \Rightarrow x = \frac{t_x}{t_z}$$

$$-t_y + yt_z = 0 \Rightarrow y = \frac{t_y}{t_z}$$

At $\left(\frac{t_x}{t_z}, \frac{t_y}{t_z}\right)$, optical flow $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Implies FOE is projection of translation vector

With respect to the FOE, the flow vectors are radially outward

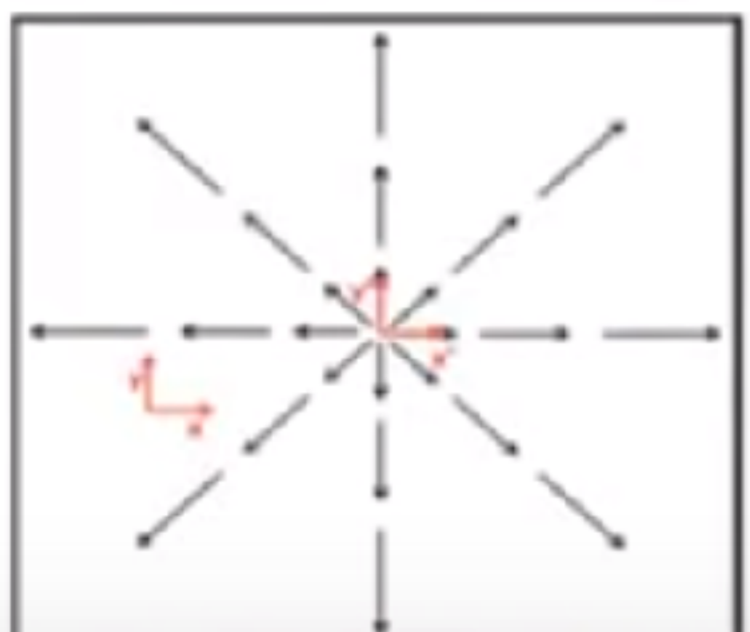
Suppose we change the origin to the FOE by applying the following coordinate change to Eq.(3.5),

$$x' = x - \frac{t_x}{t_z}, y' = y - \frac{t_y}{t_z}, \tag{3.9}$$

then the optical flow field becomes

$$[u, v]^T(x', y') = \frac{t_z}{Z}[x', y']^T. \tag{3.10}$$

which should look very familiar. Thus the general case too corresponds to optical flow vectors pointing outwards from the FOE, justifying the choice of the term. Figure 3.3 shows such an optical vector field.



Rotational component

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} -1 & 0 & X \\ 0 & -1 & Y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} XY & -(1+X^2) & Y \\ 1+Y^2 & -XY & -X \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

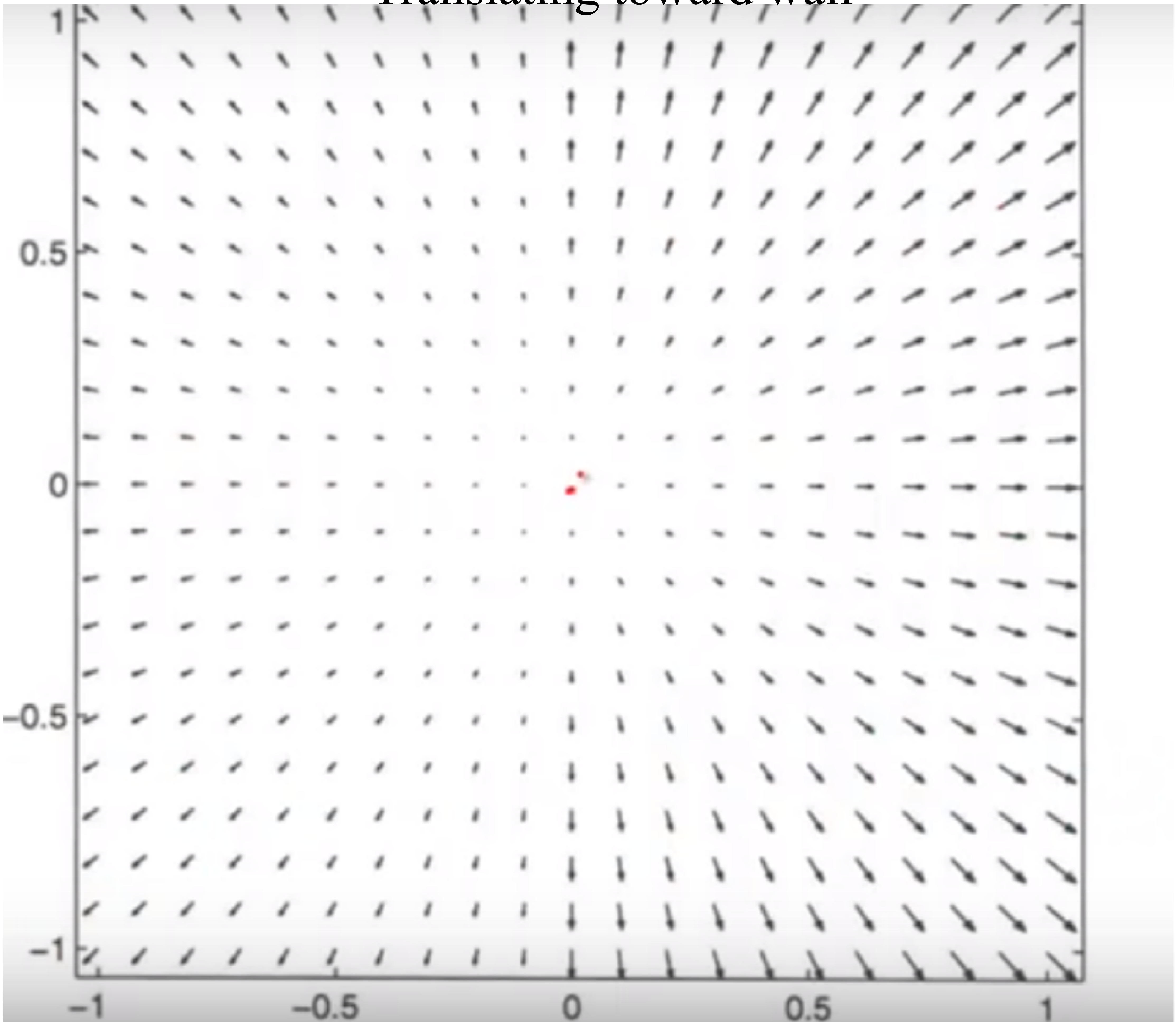
translation component   rotation component

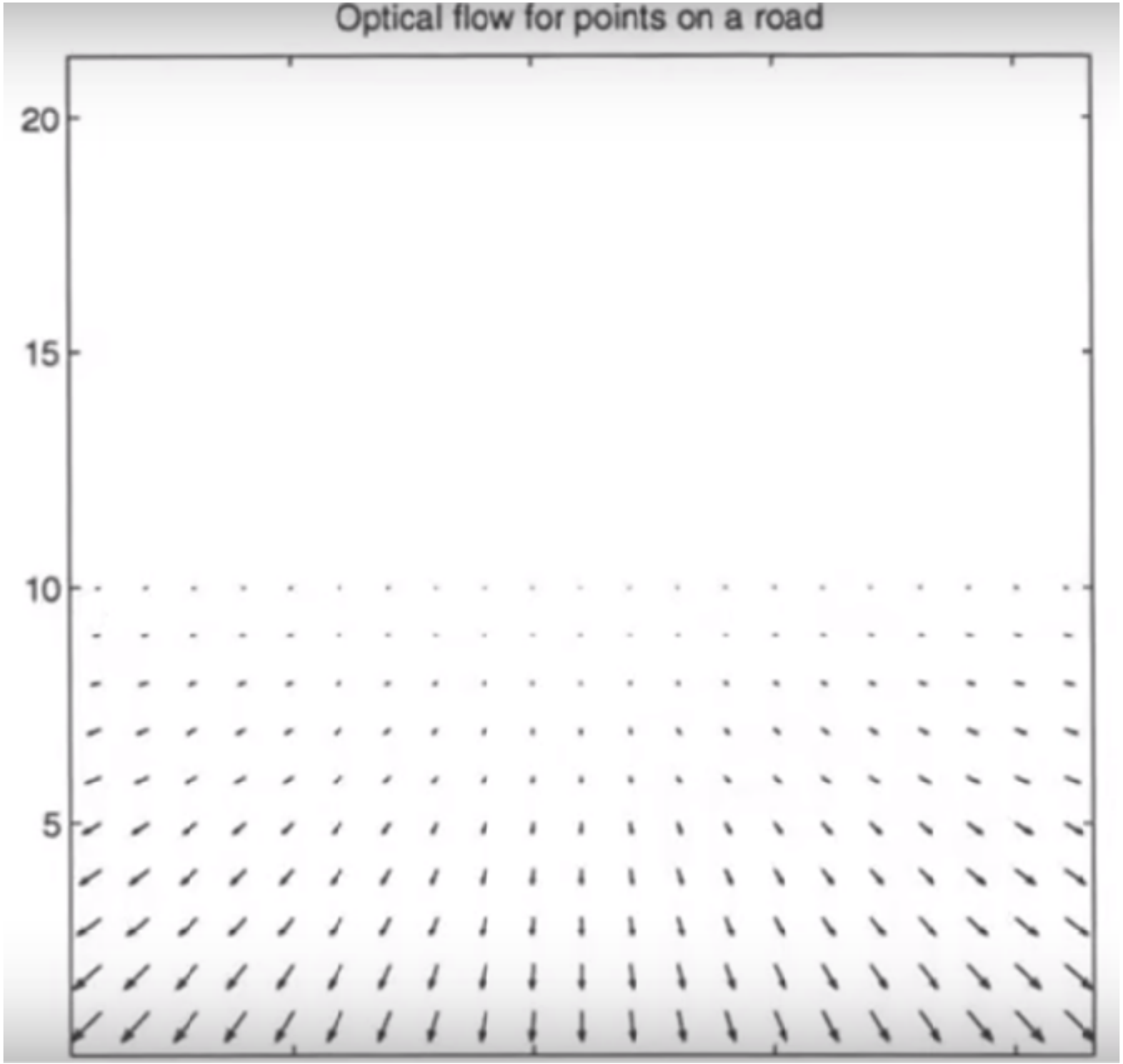
Rotational component does not depend on scene depth Z

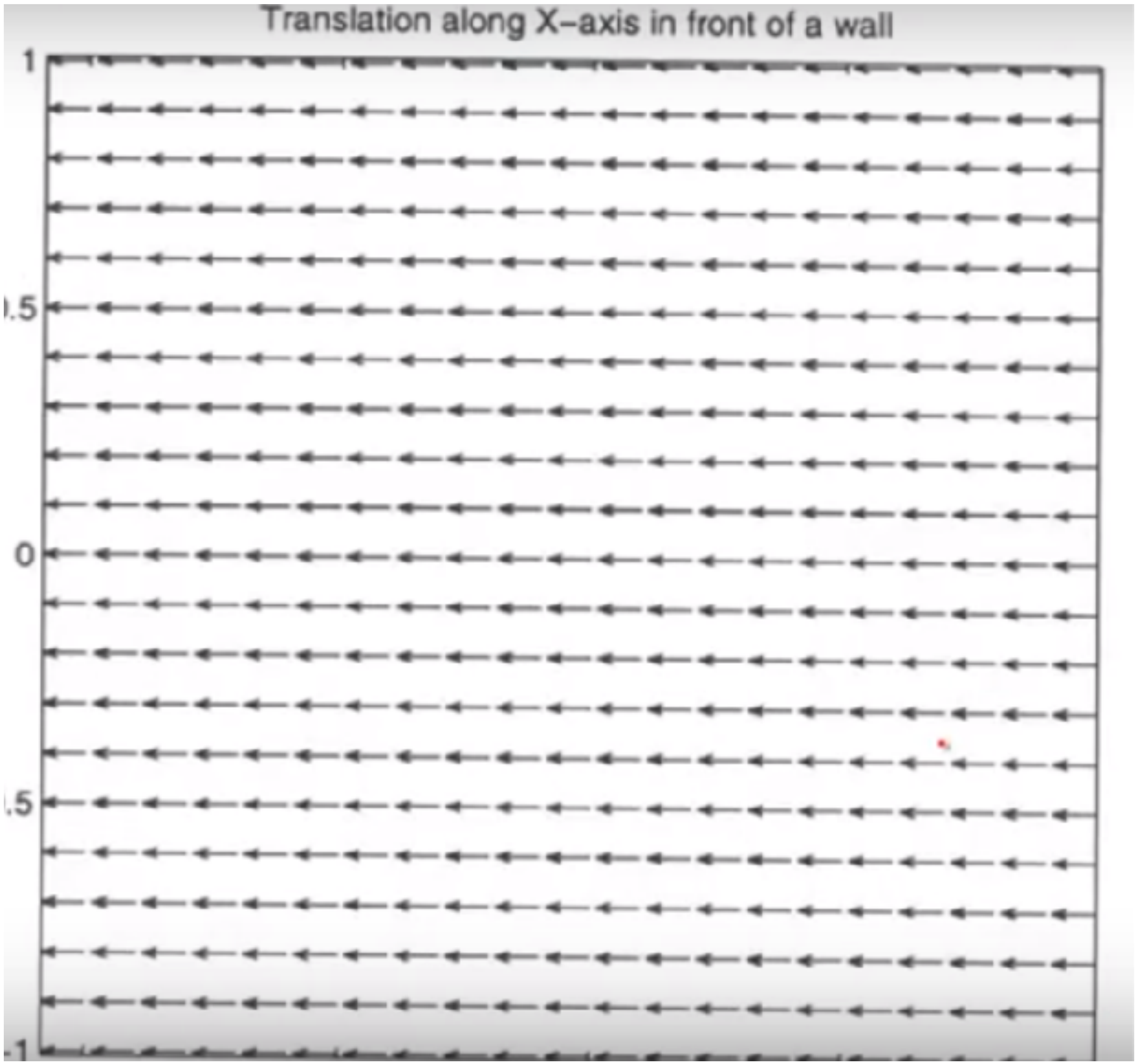
(exploited when fitting a homography to a rotating camera!)

Angular velocity can be recovered from rotational component

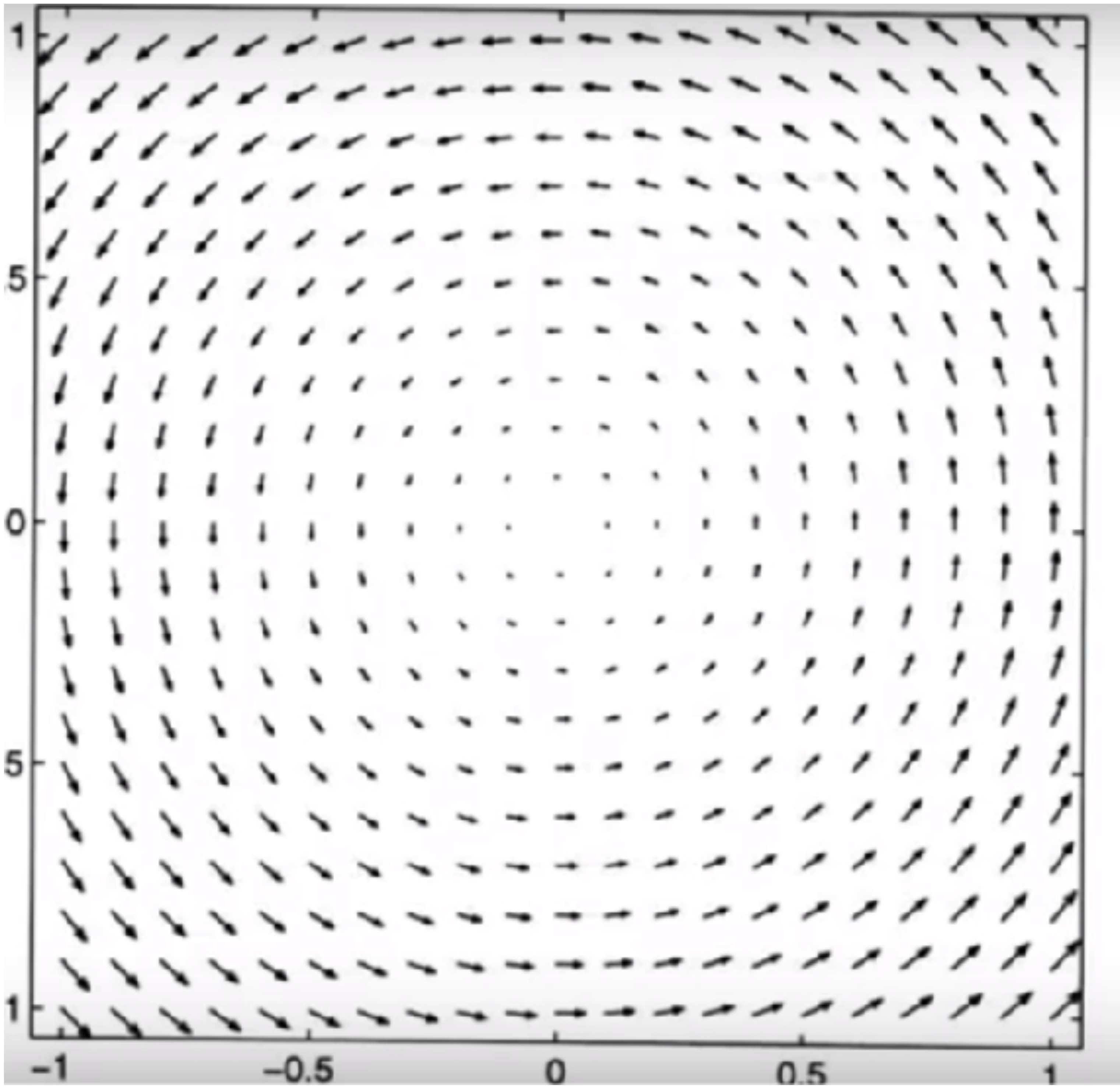
Translating toward wall



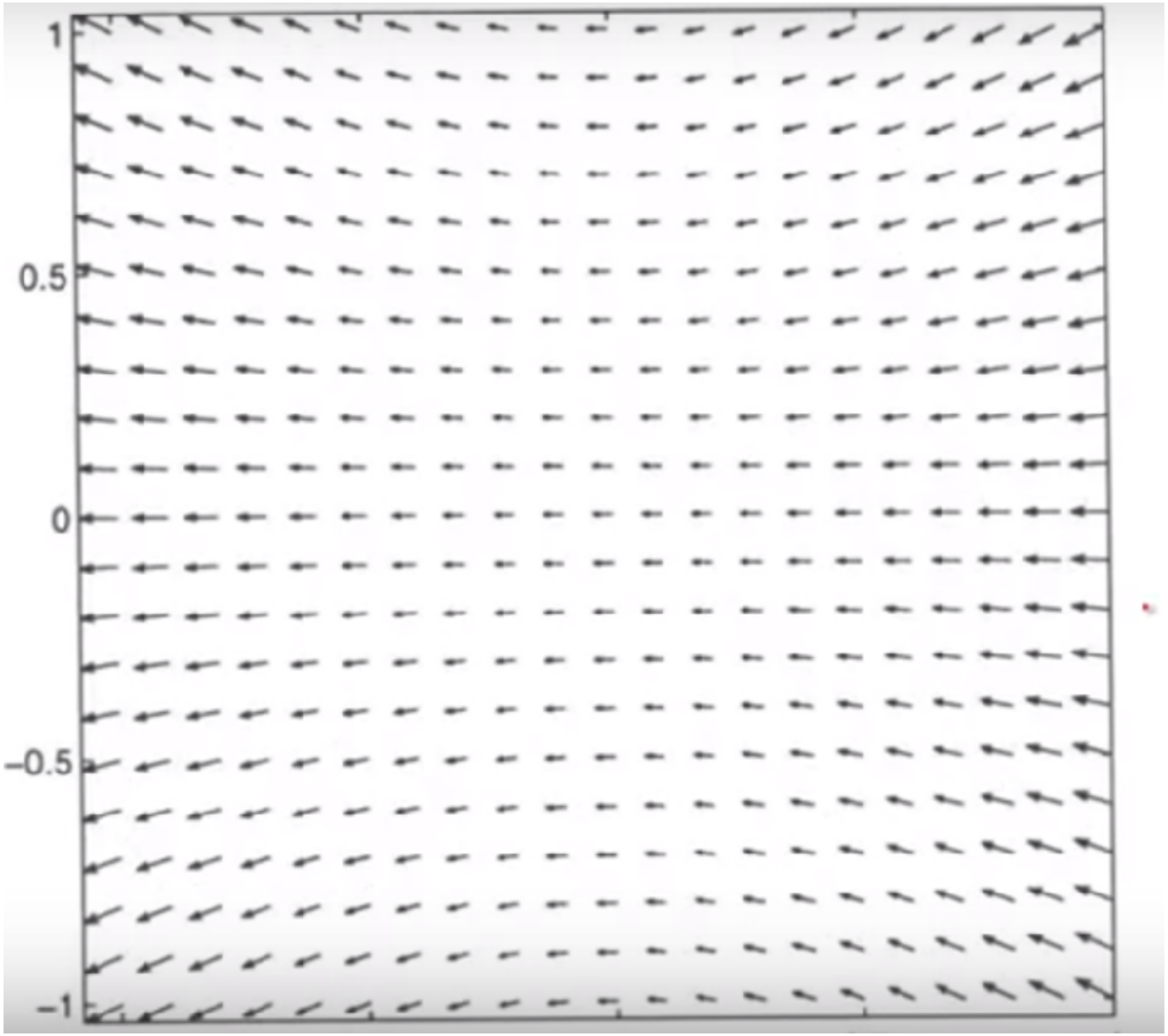

Optical flow for points on a road
20
15
10
5


Translation along X−axis in front of a wall
1
.5
0
.5
-1

Rotating about z-axis



Rotating about y-axis



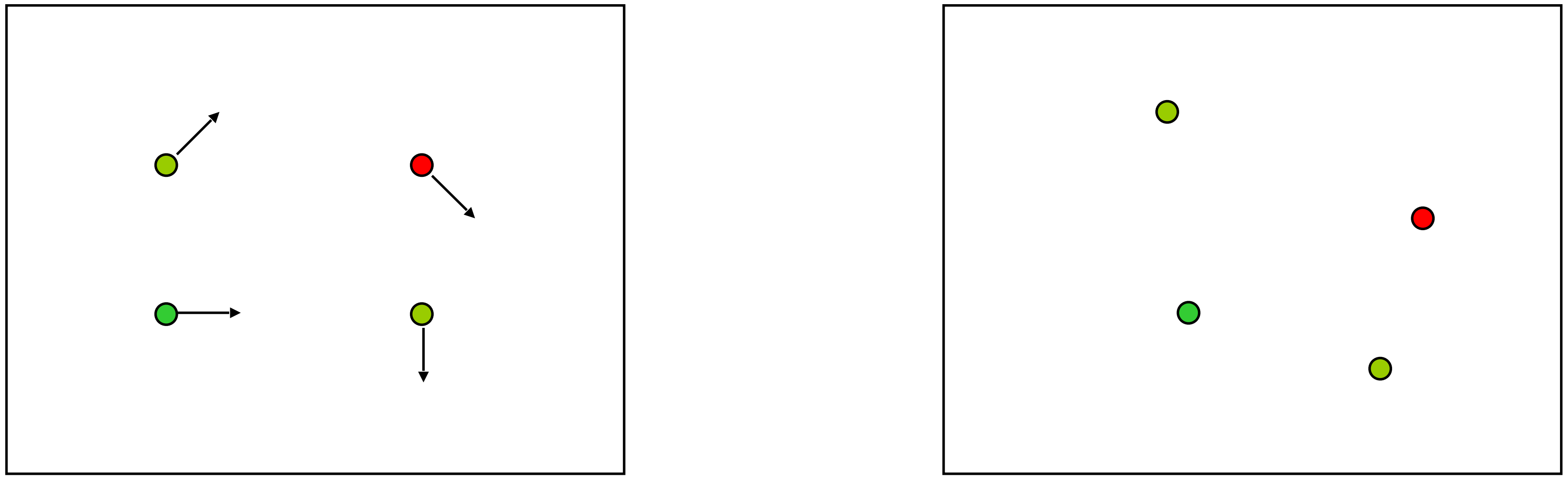
Concluding Remarks

- Suppose an animal or a robot could analyze its video signal to measure the optical flow field, it could use this as data to compute depth and egomotion. t, ω $Z(x,y)$
- There is considerable evidence that many animals can (a) measure optical flow (b) use it to control their movement, avoid obstacles etc.
- We will study later how to measure optical flow.

Outline

- Lucas Kanade
- Moving cameras (egomotion)
- Estimating flow

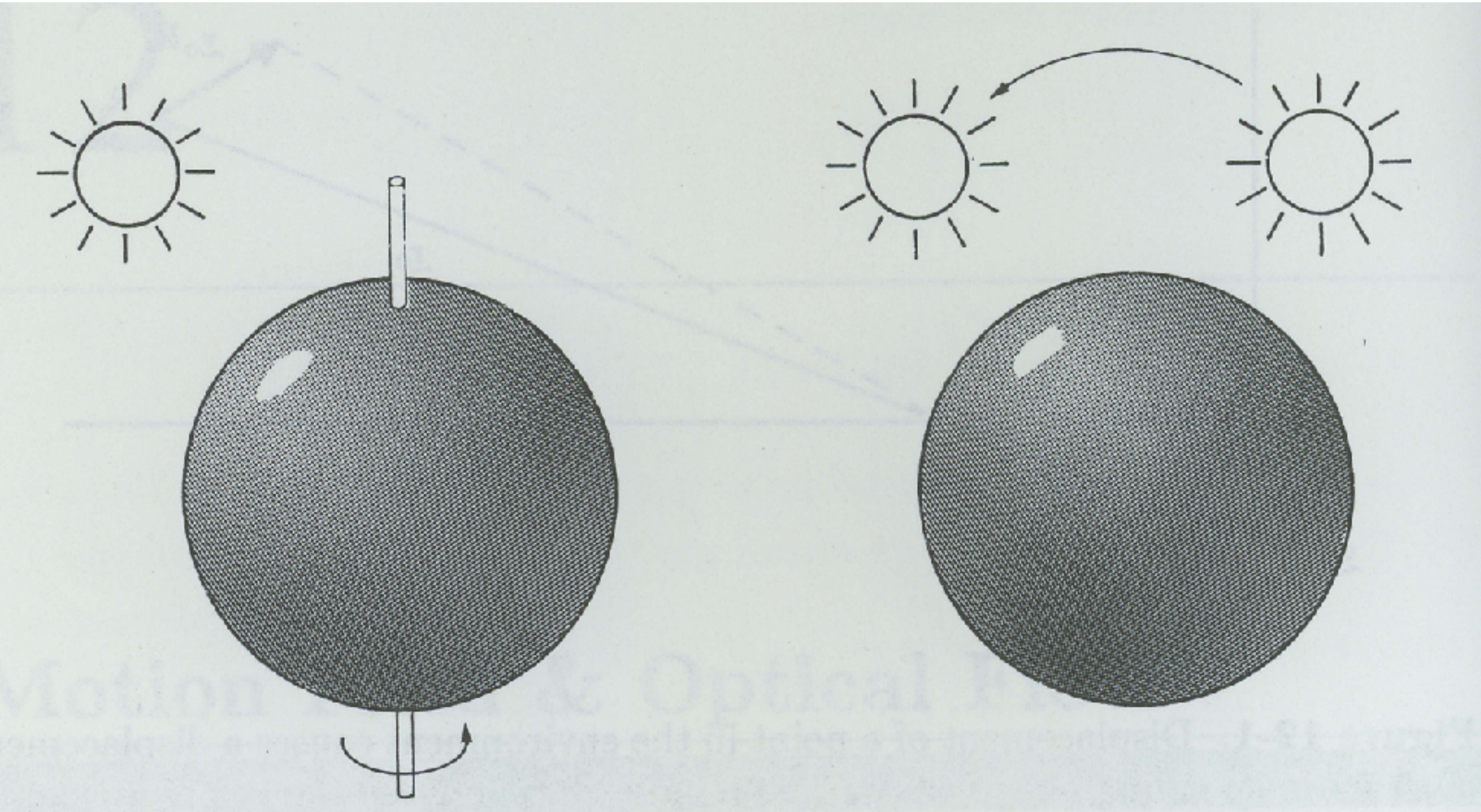
Problem Definition: Optical Flow



$H(x, y)$ $I(x, y)$

- 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 assumption
  - **color constancy:** a point in H looks "the same" in image I
    - For grayscale images, this is **brightness constancy**

Caution:
2D measured optical flow $\neq$ 3D scene flow



Motion field exists but no optical flow

(a)

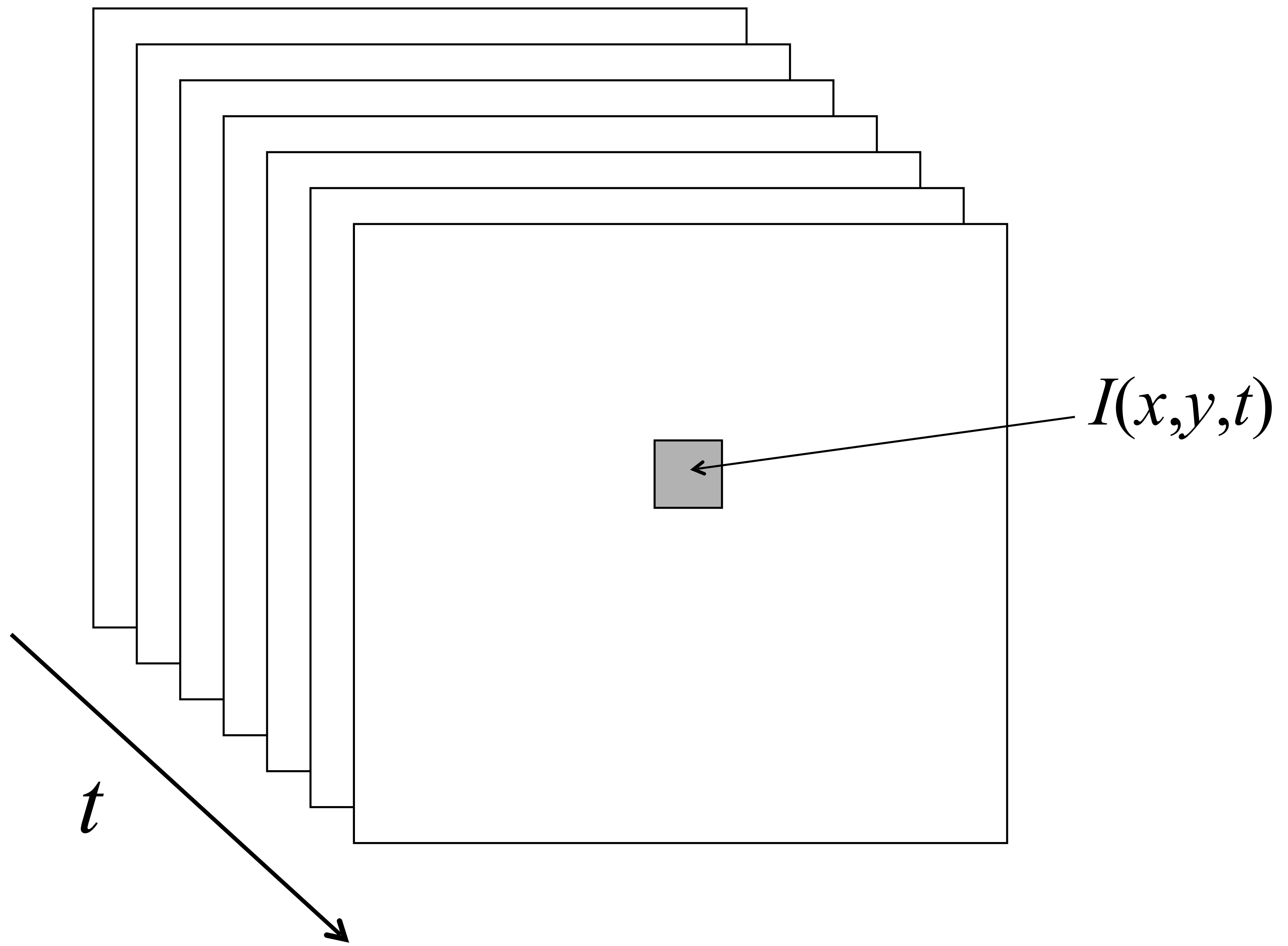
No motion field but shading changes

(b)

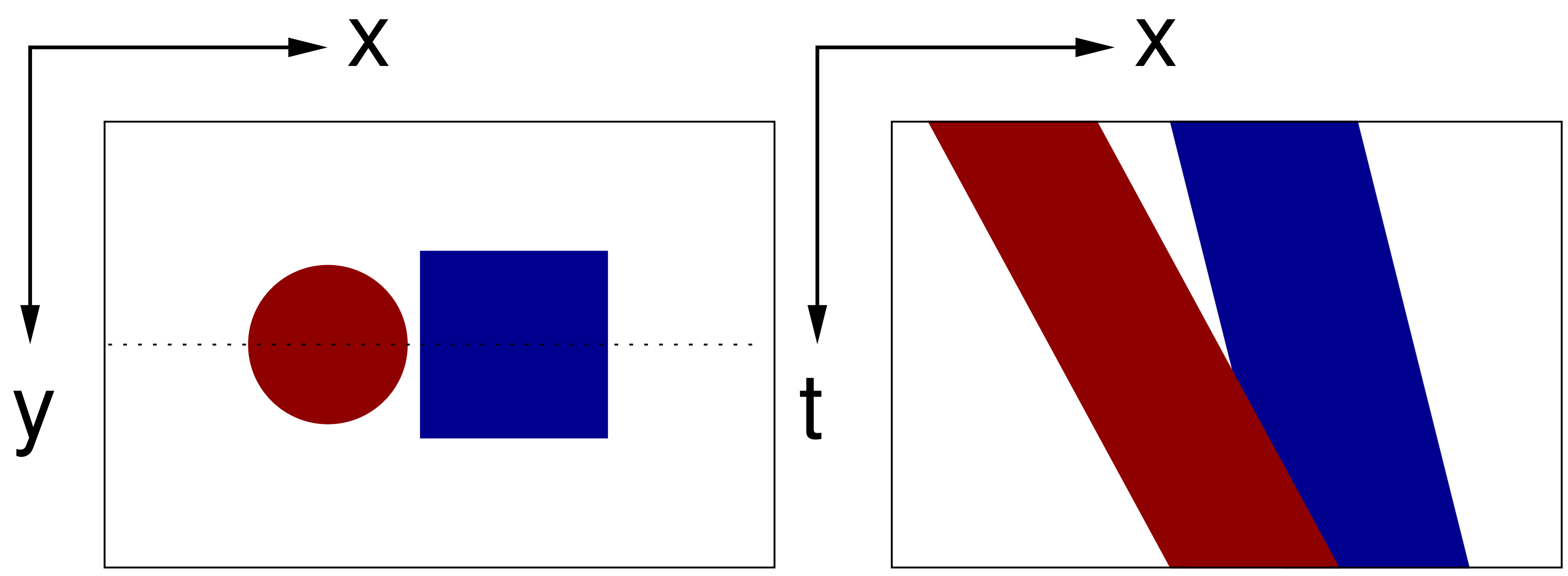
Importance of low-level motion

(for inferences beyond camera motion)

Videos as spacetime cubes



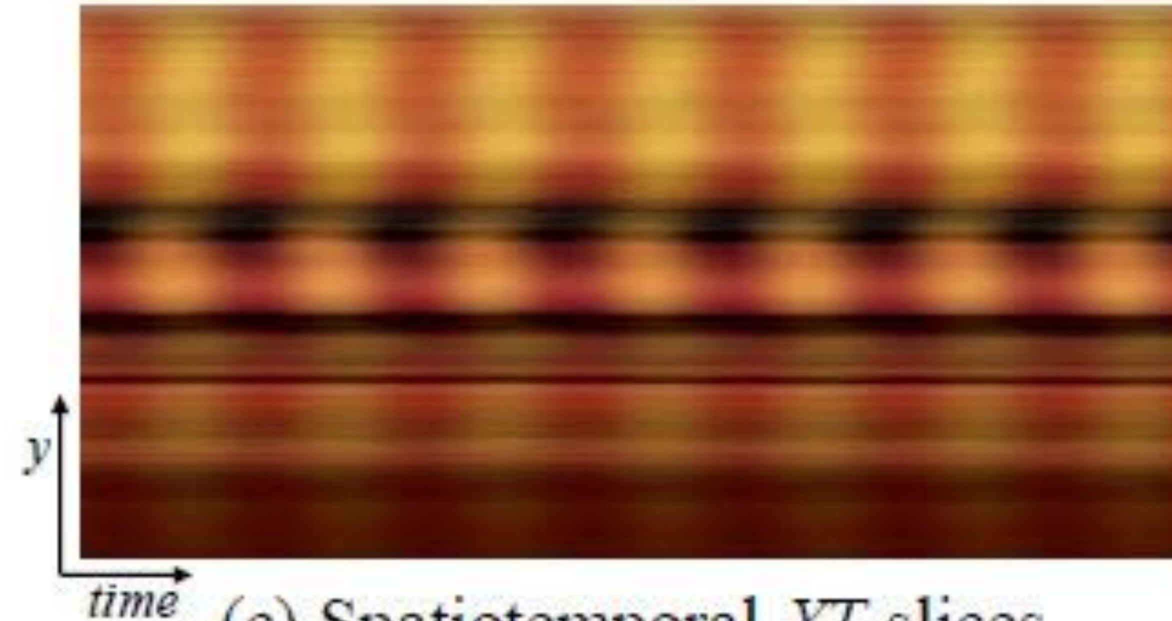
Visualizing spacetime cubes



In this example, the circle is in front of the square and the camera is moving horigontally to the left

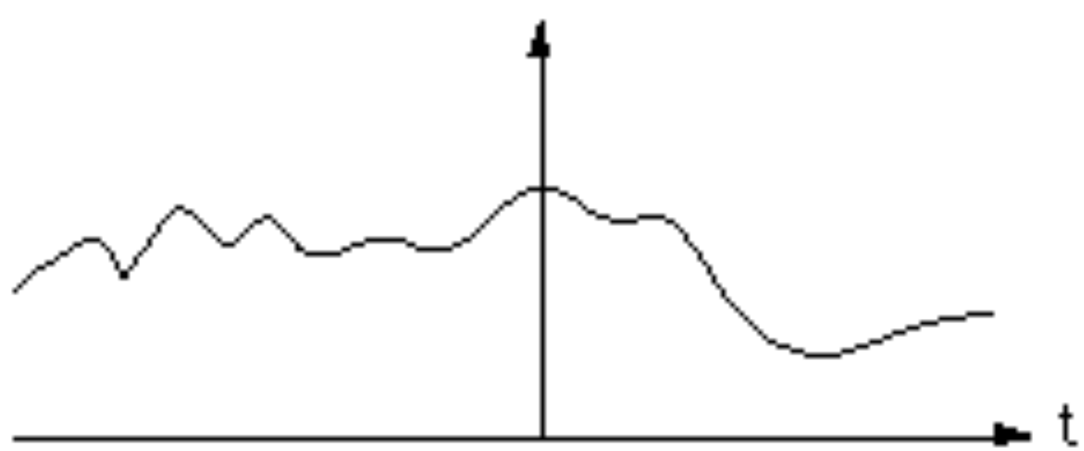
Digression: visualizing space-time cube

Plot I(x,y,t) for a fixed t



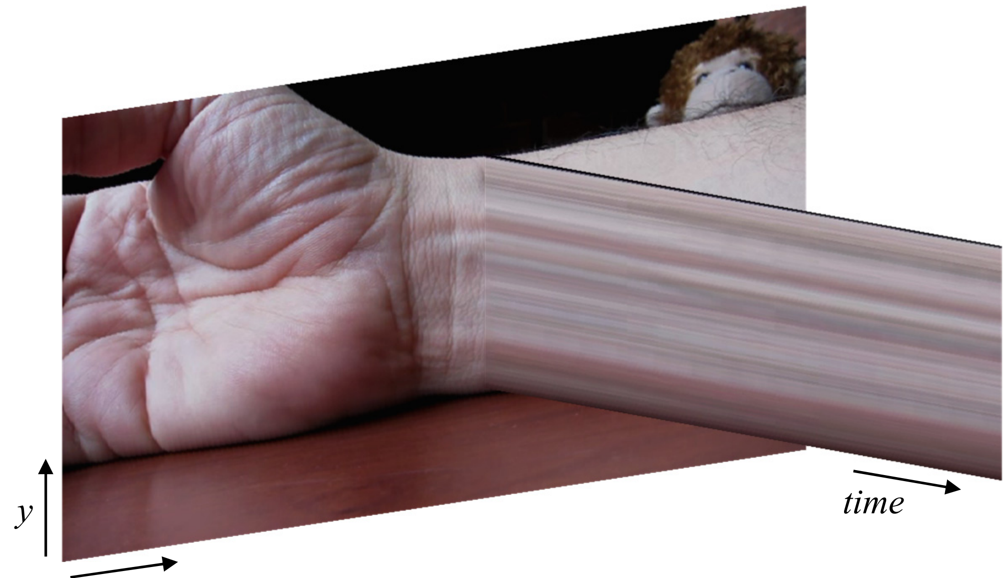
(c) Spatiotemporal *YT* slices

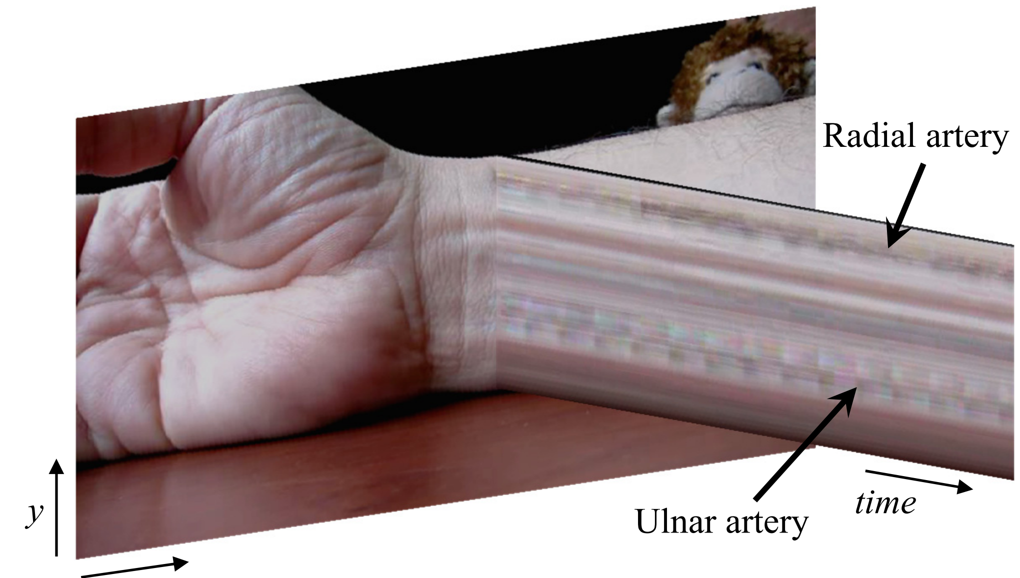
Plot I(x,y,t) for a fixed x



Plot I(x,y,t) for a fixed (x,y)

Amplifying temporal signals





Eulerian Video Magnification for Revealing Subtle Changes in the World