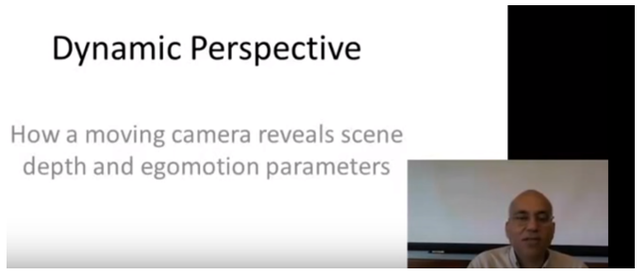
Motion and flow

Outline



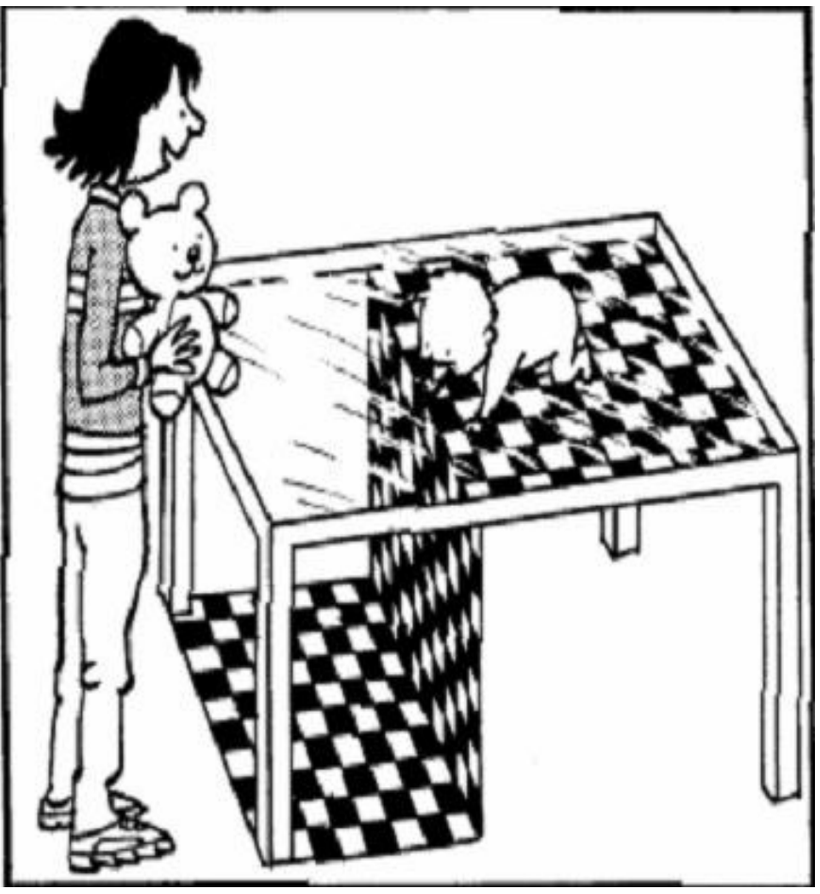
- **Egomotion**
  - Motivation
  - Time-to-Contact, Parallax, Focus-of-Expansion
- Optical Flow
  - Logistics
  - Motivation, aperture problem
  - Sparse (KLT) vs Dense (variational)
  - Optimization tools: robust statistics, coarse-to-fine, markov-random feilds
  - Segmentatiion (dominant motion estimation, background subtraction, layered models

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

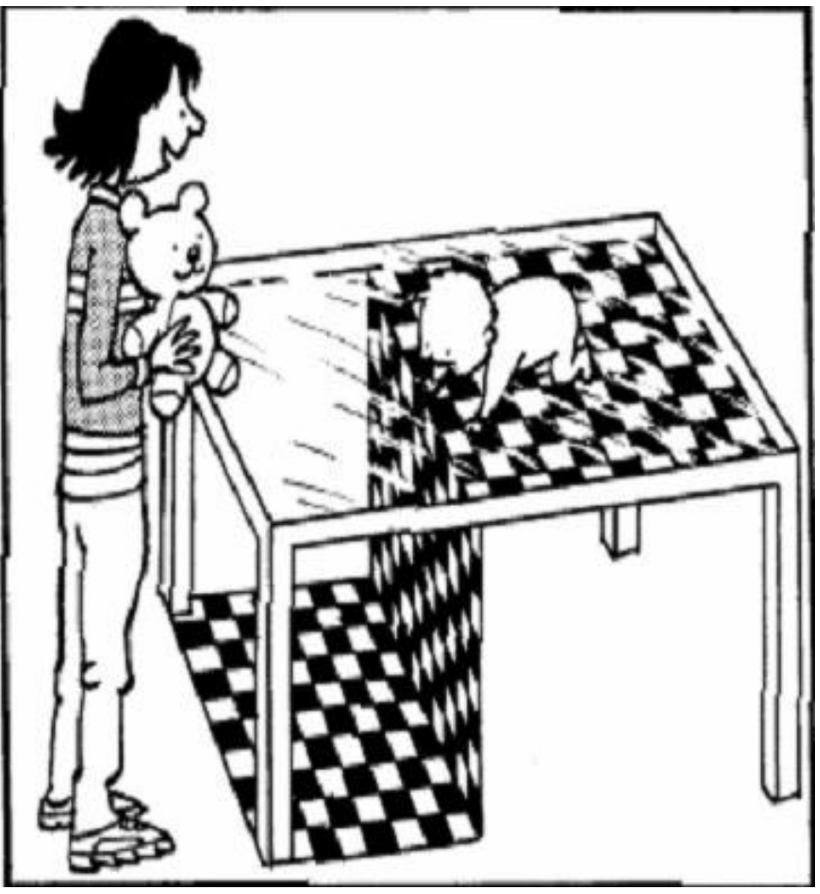


Gibson, 1960s

After 1 week of crawling:

After several weeks of crawling:

Human development:
Crawling teaches babies depth perception

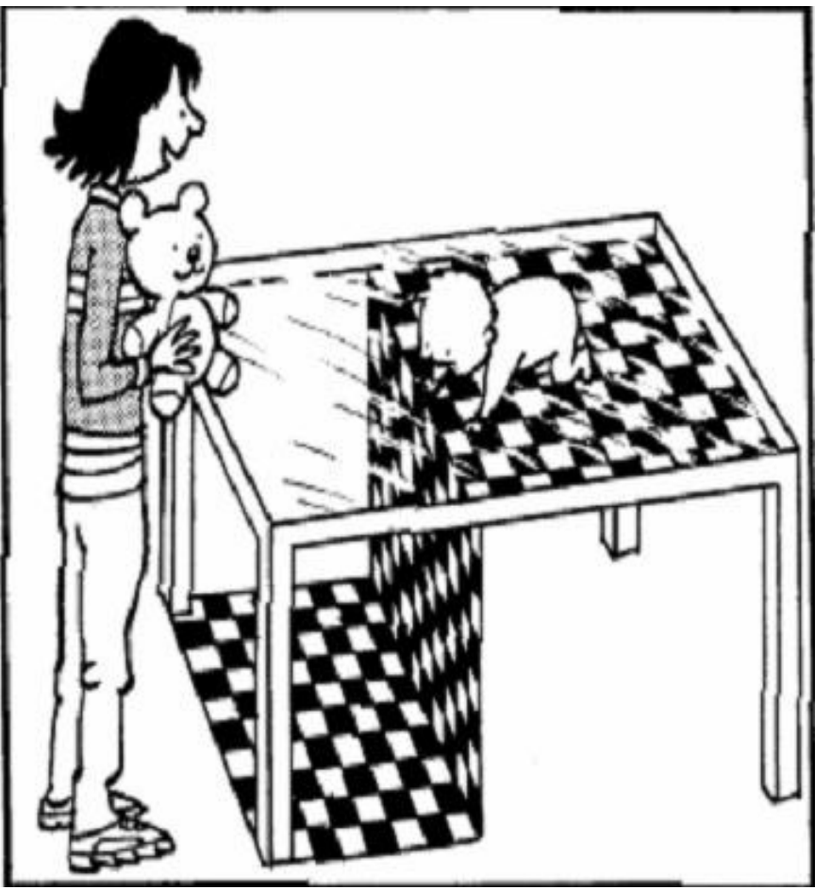


Gibson, 1960s

After 1 week of crawling:

After several weeks of crawling:

Human development:
Crawling teaches babies depth perception

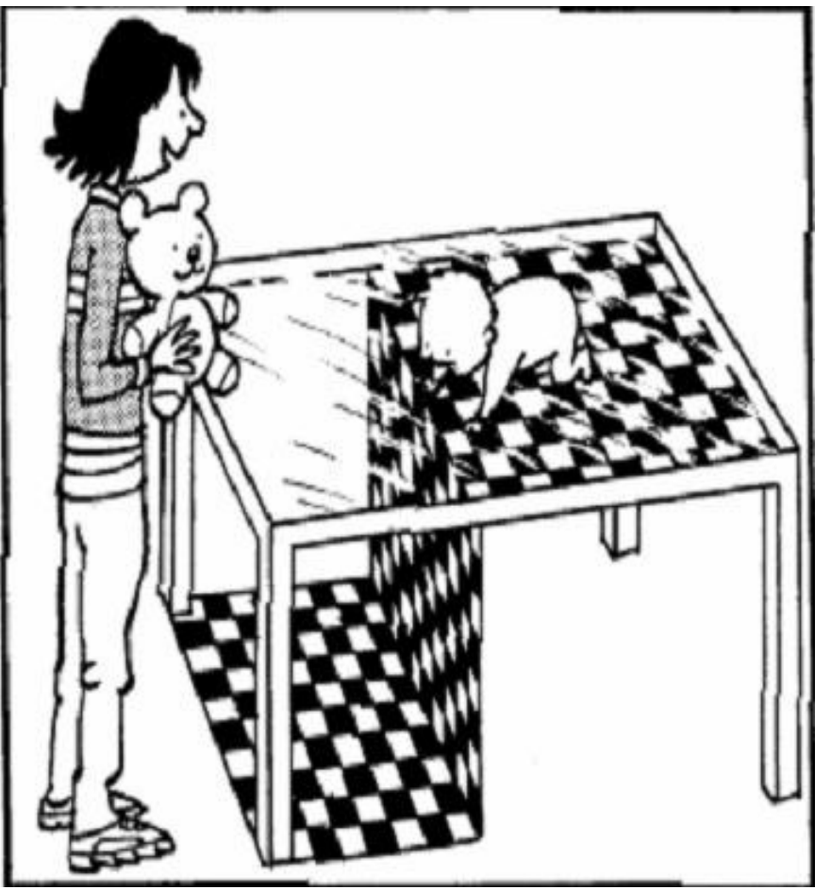


Gibson, 1960s

After 1 week of crawling:

After several weeks of crawling:

Human development:
Crawling teaches babies depth perception

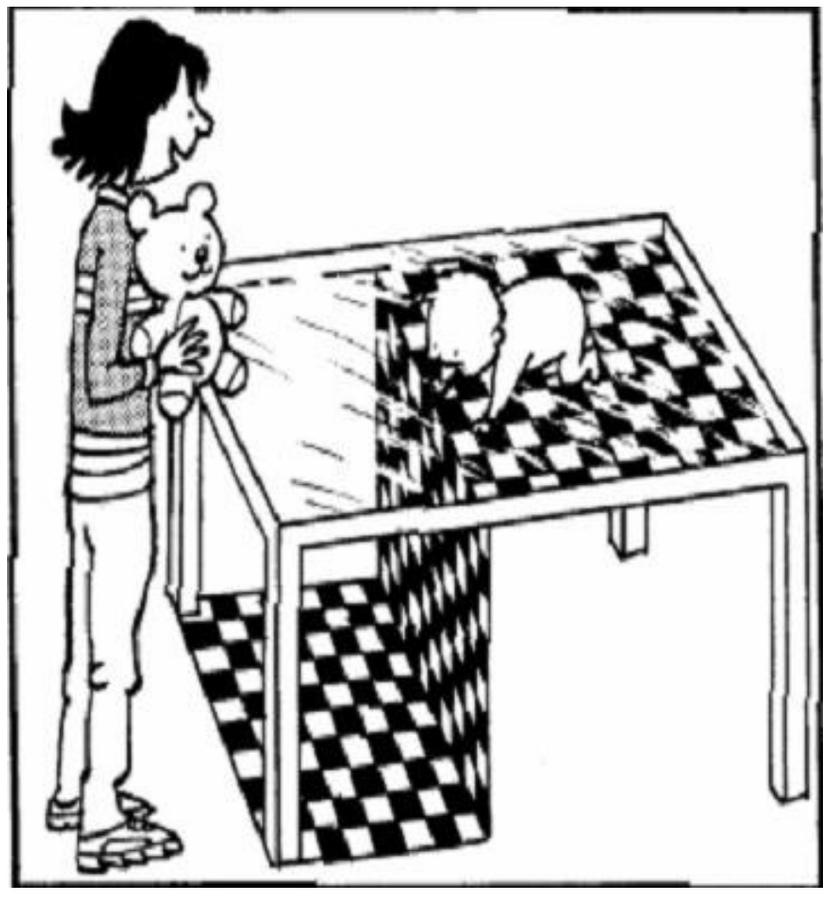


Gibson, 1960s

After 1 week of crawling:

After several weeks of crawling:

Today’s goal



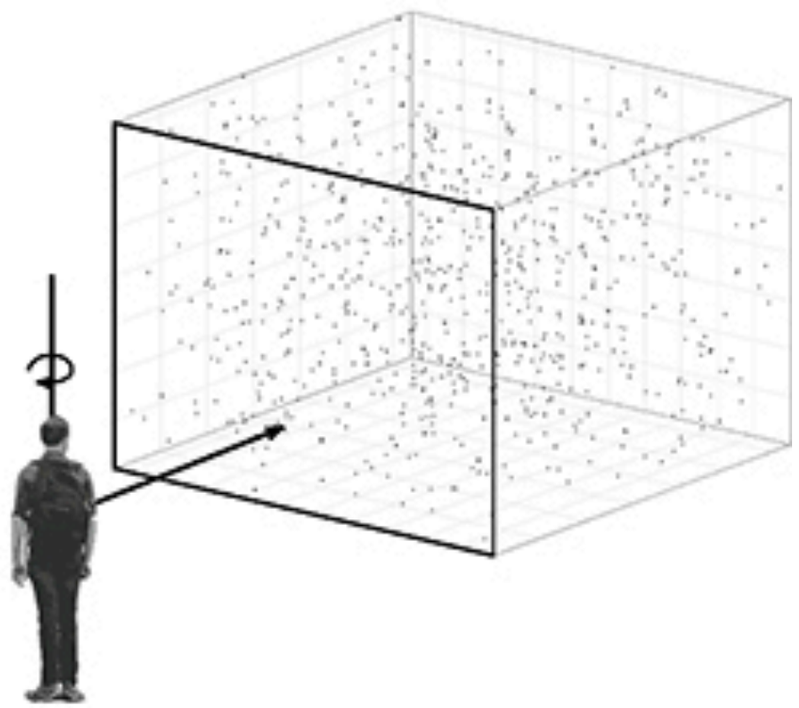
Understand the space of image transformations that we see when we move

Sometimes referred to *egomotion*

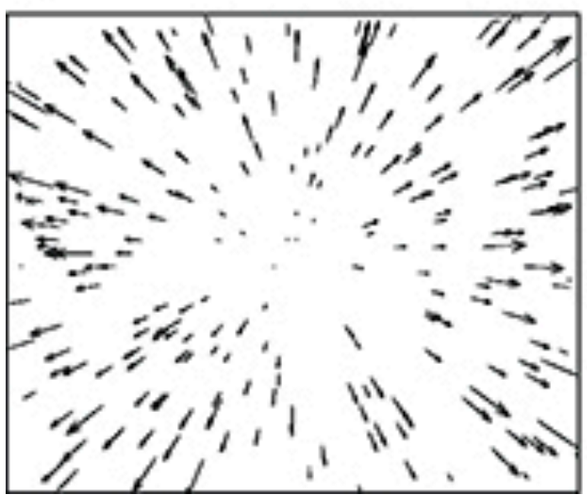
How does camera movement affect pixel “flow”?

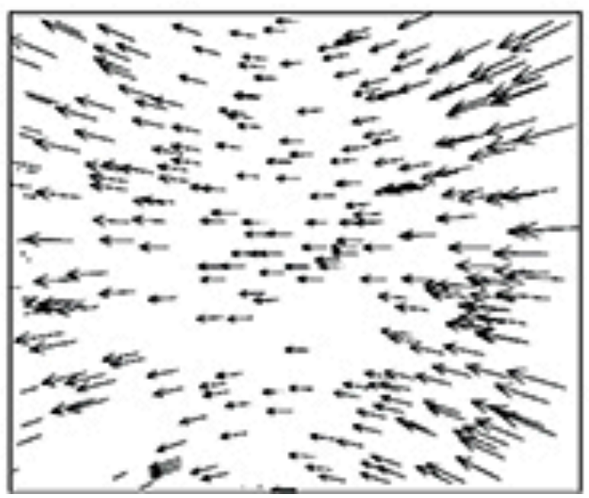
A.

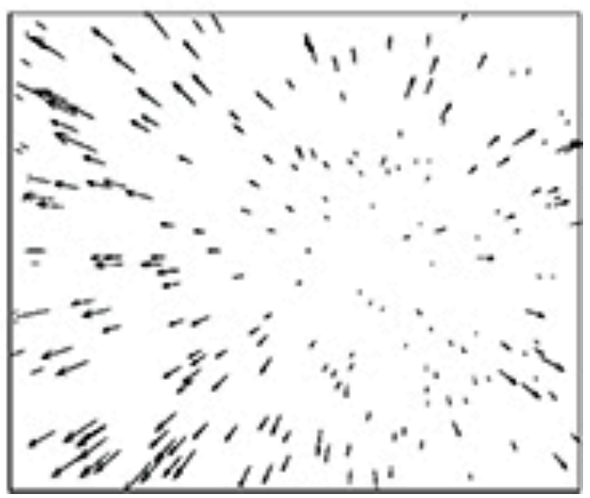
B.



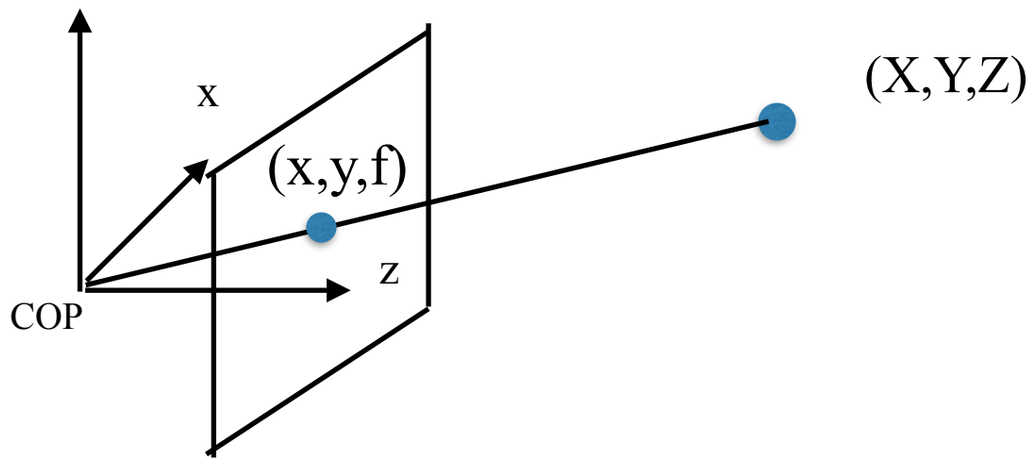
C.







Recall: pinhole cameras



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

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

Assume calibrated cameras (with *known* intrinsics K).
Implies we can compute normalized image with f=1

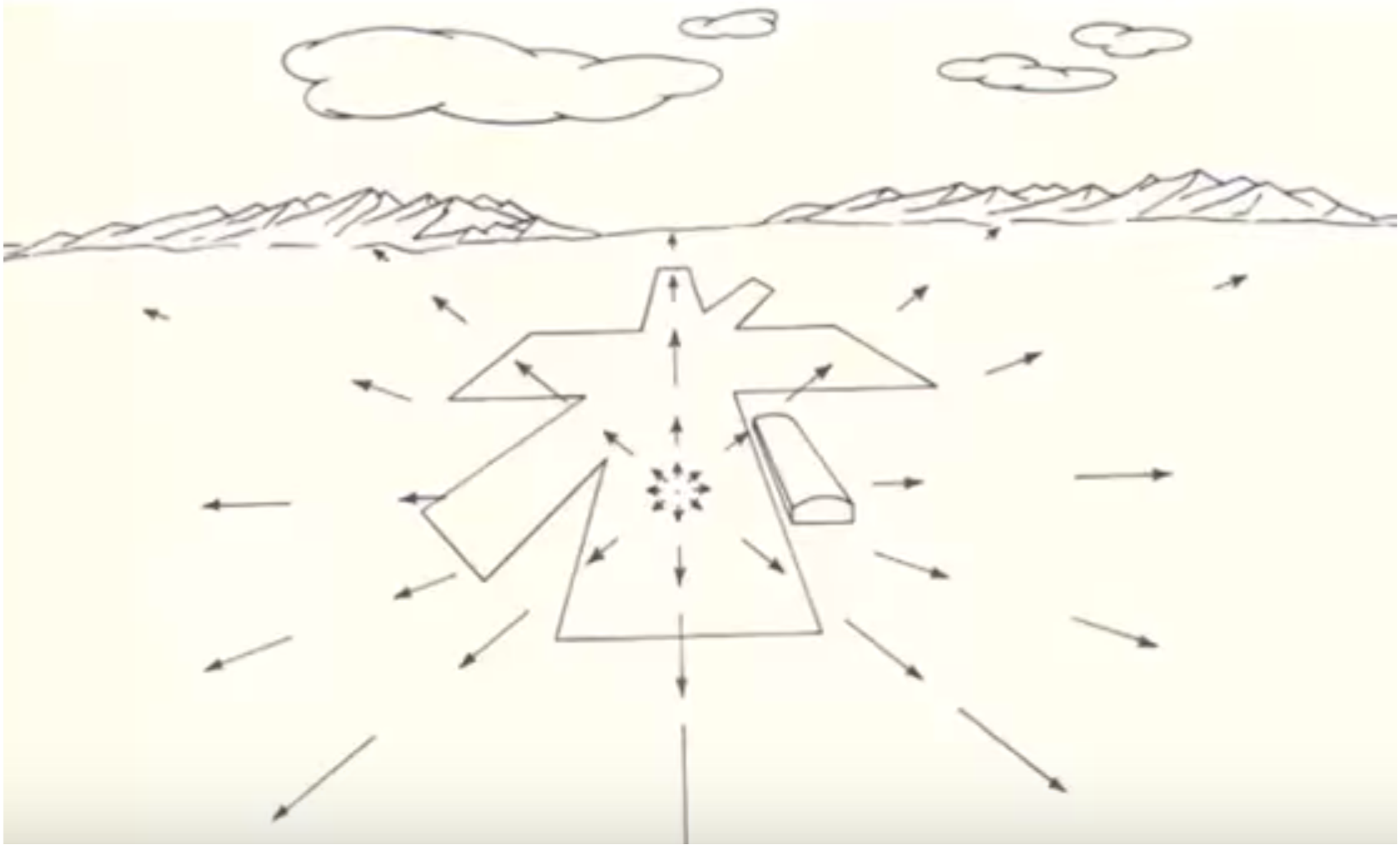
Suppose the camera moves with respect to the world…

- 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*+Δ*x*, *y*+Δ*y*) in time Δ*t*, then

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

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

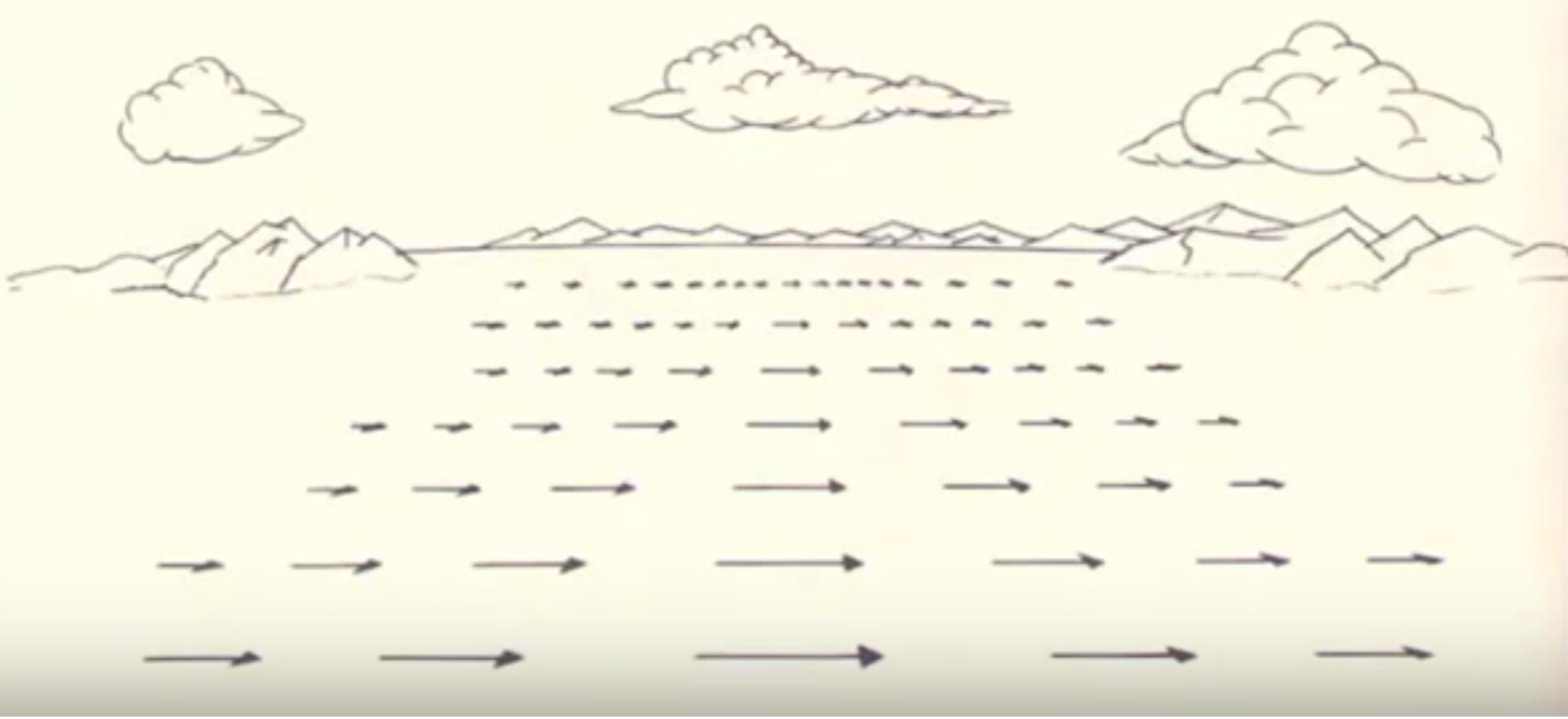
Gibson's example 1: optical flow for a pilot landing a plane



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

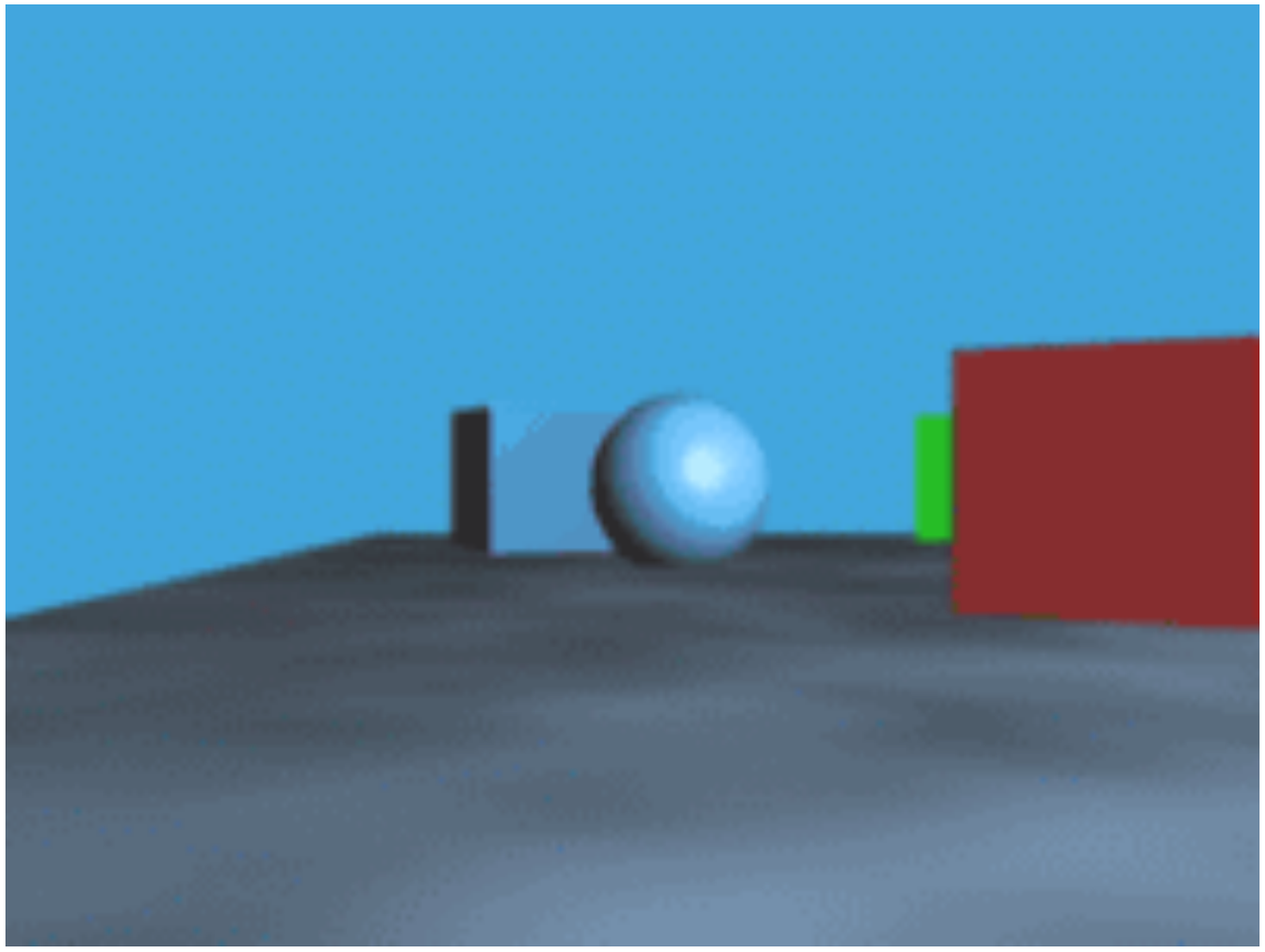
Gibson's example II:

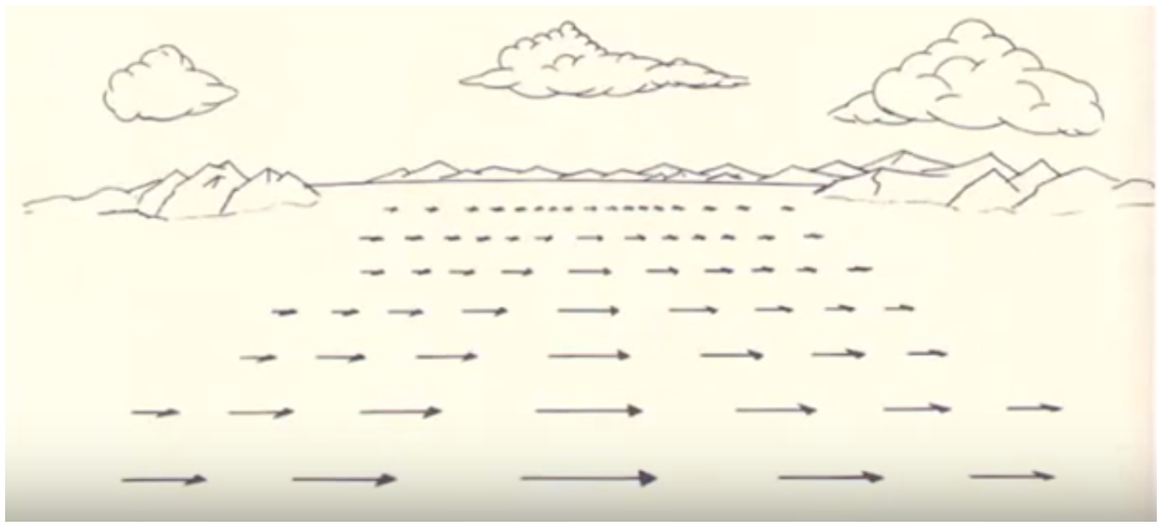
Optical flow from the side window of a car



Parallax

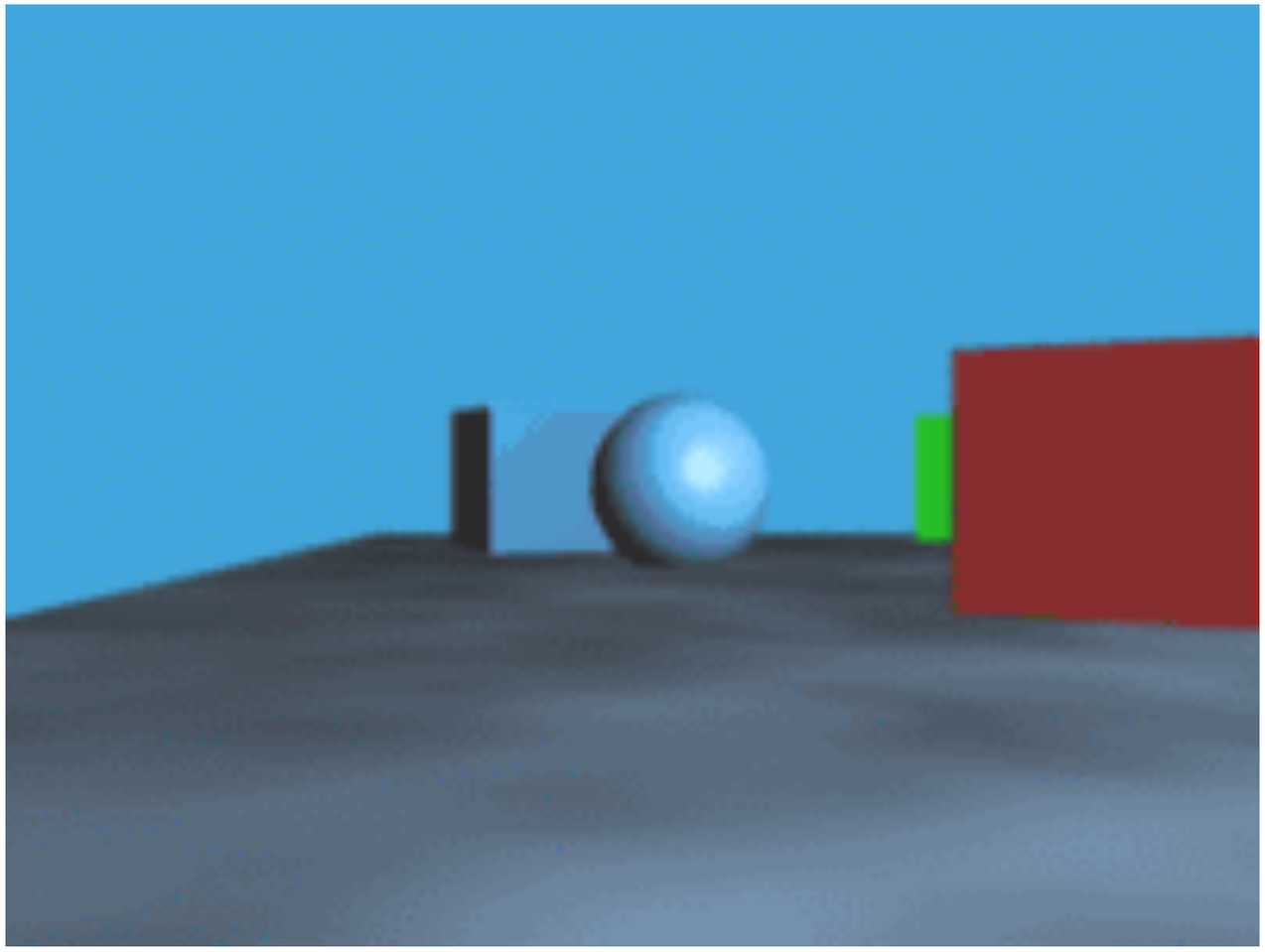
Intuition: flow due to translating cameras reveals depth about scene

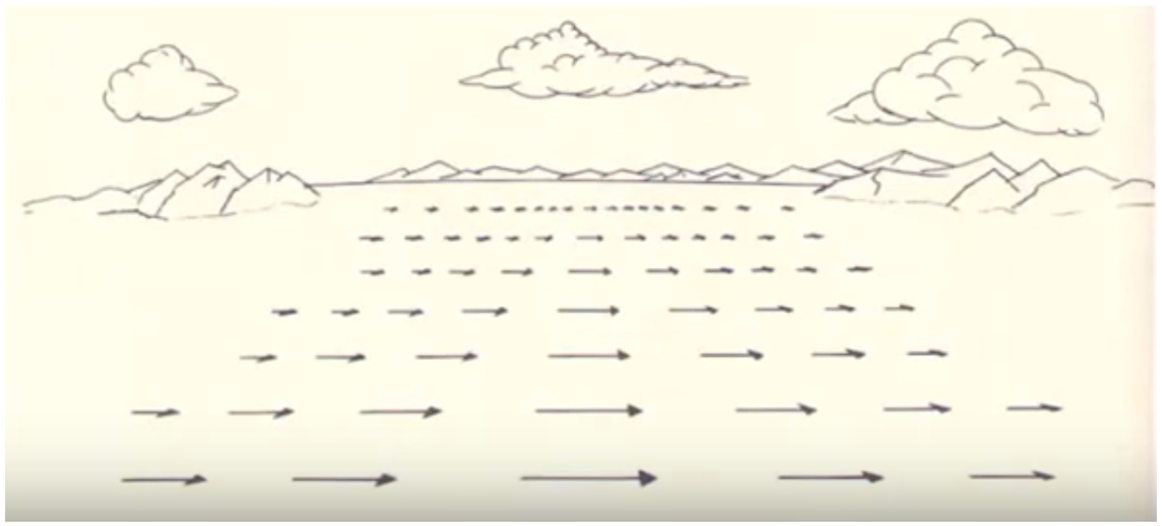




Parallax

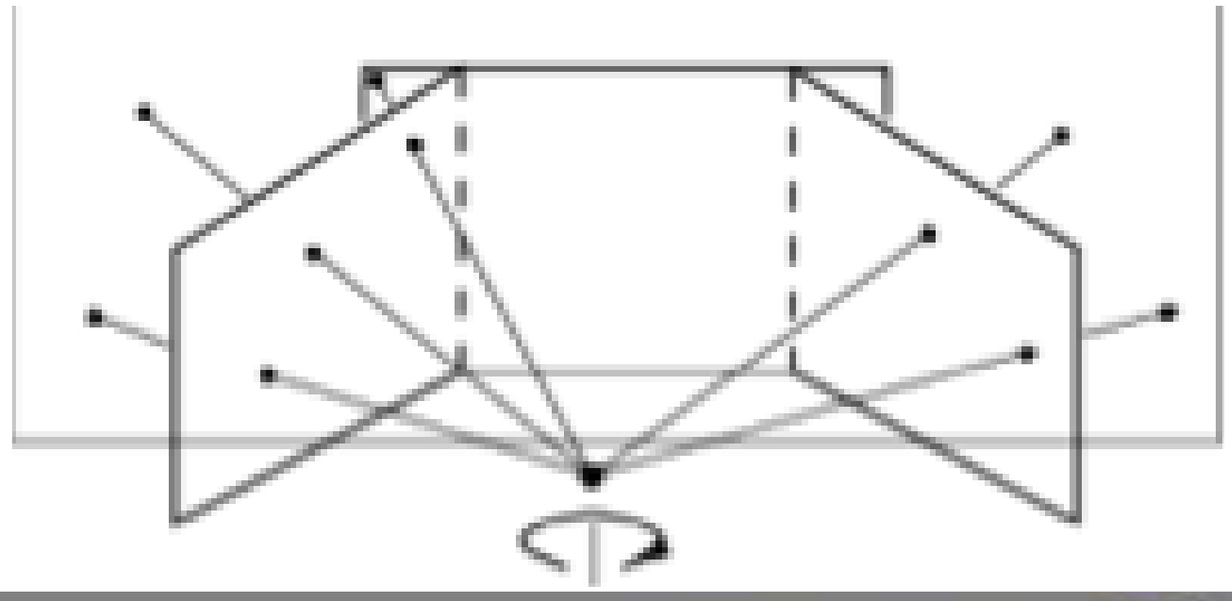
Intuition: flow due to translating cameras reveals depth about scene

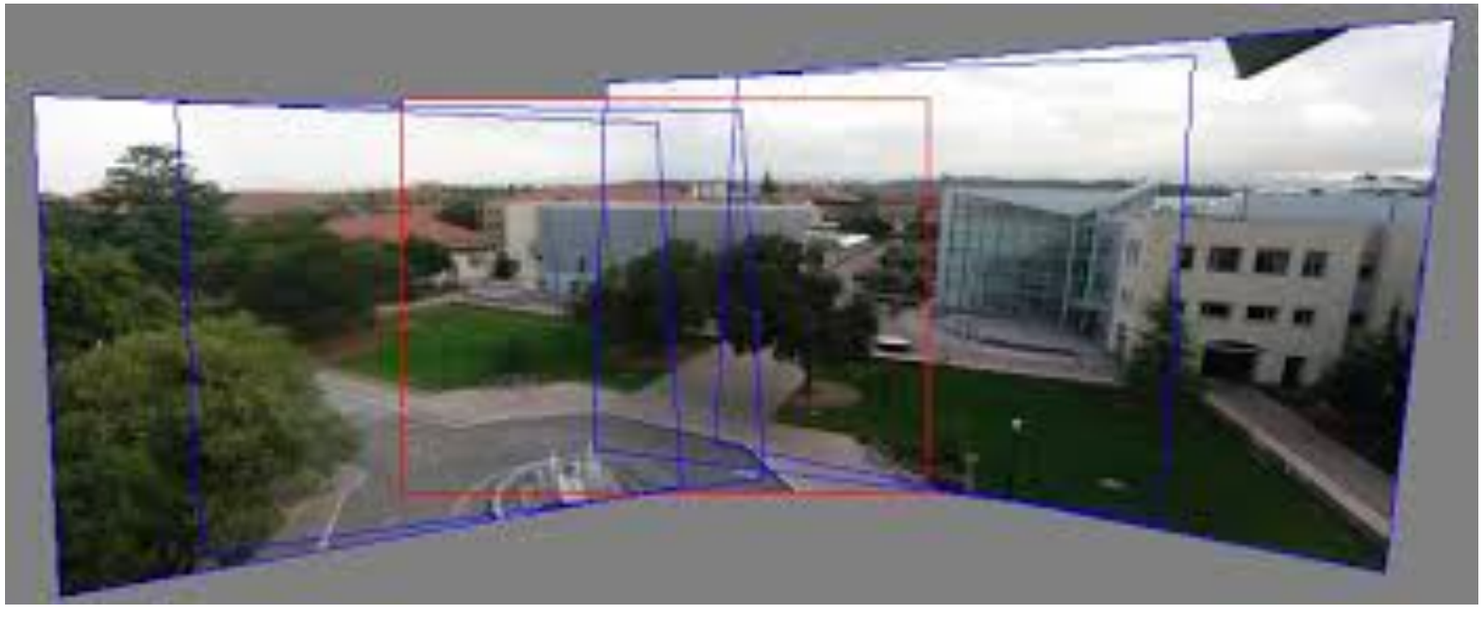




Rotating cameras

What does a rotating camera tell us about scene depth?

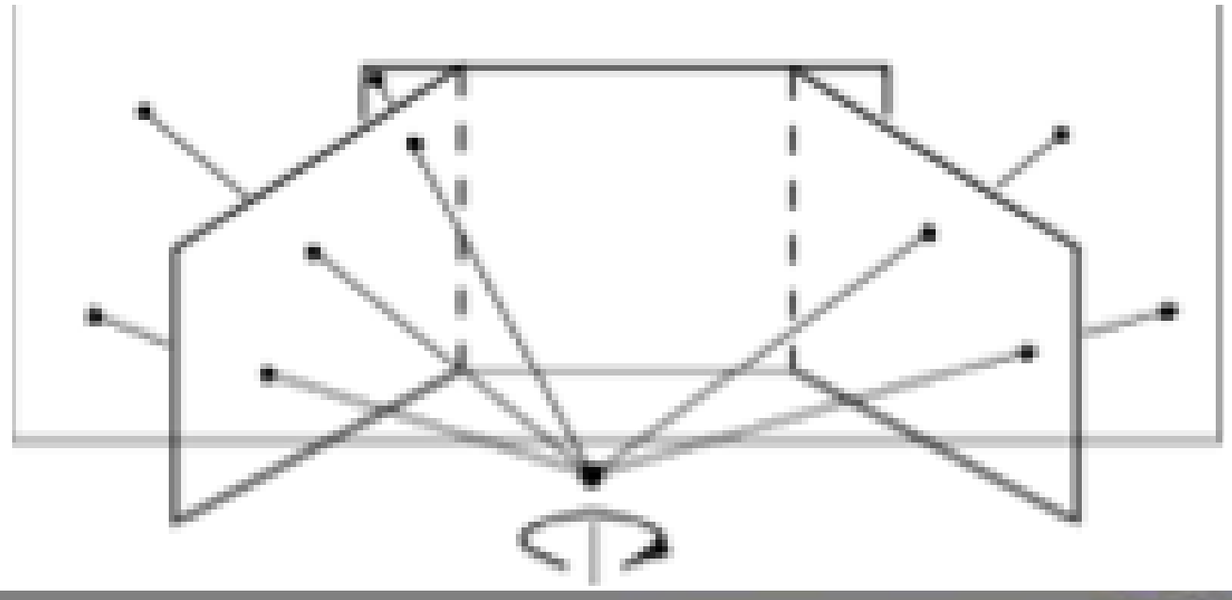


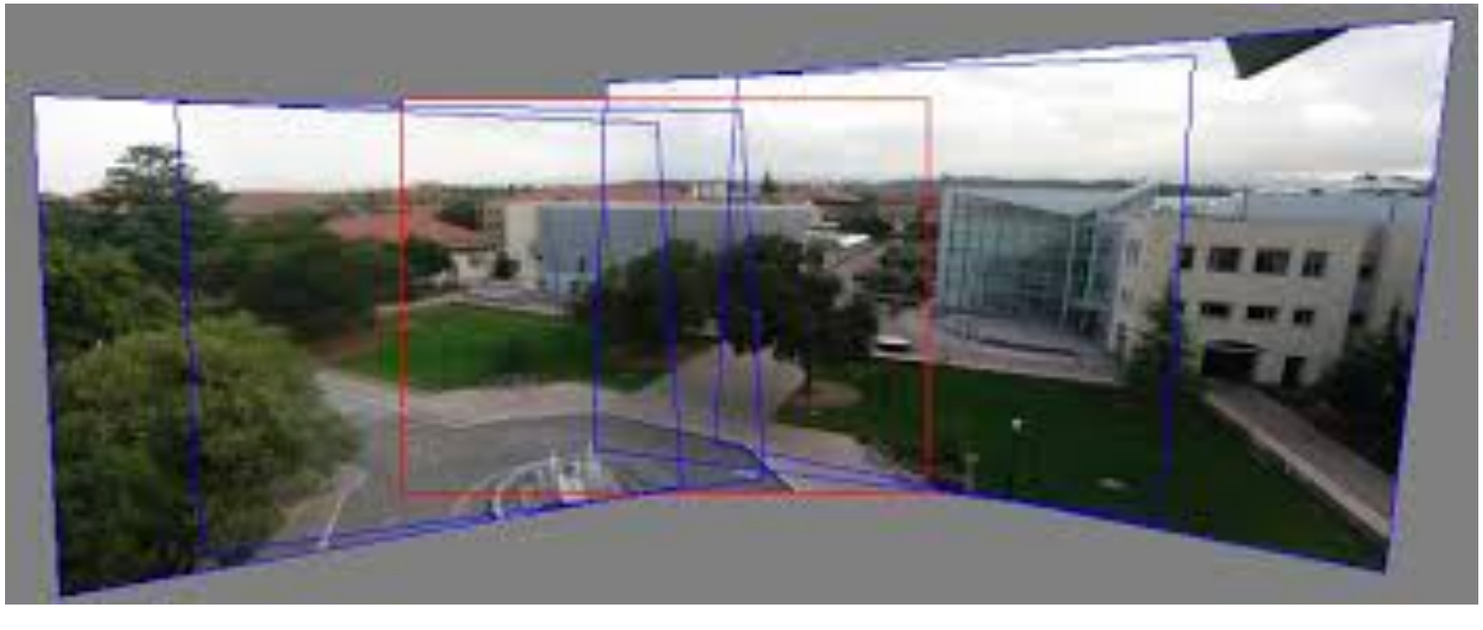


Rotating cameras

What does a rotating camera tell us about scene depth?

Since pixels are related by a homography transformation, nothing!





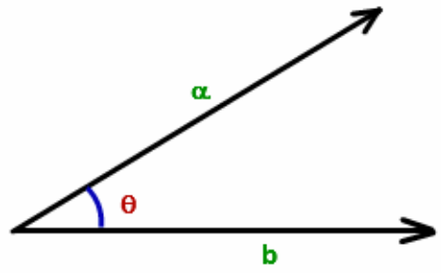
Roadmap

- Let's paramaterize rigid-body camera movement by translation velocity **t** and axis-angle rotation ω
  - We'll look at rigid body transformations (R and T) over Δt as $\Delta t \Rightarrow 0$
  - Translation => translational velocity **t**, Rotation => angular velocity ω
- We will look at flow-feilds produced by this movement: u[x,y] and v[x,y]
- We'll explicitly derive eqn that shows how egomotion flow depend on camera movement (**t**,ω) and scene depth z[x,y]
  - **Punchline 1**: translation component of flow field is the important one; it reveals info about scene depth z[x,y] and camera translational velocity **t**
  - **Punchline 2**: rotation component of flow field reveals info about ω, but not scene depth

We'll start with a bunch of review slides on angular velocity
[can be ignored if we just want punchline]

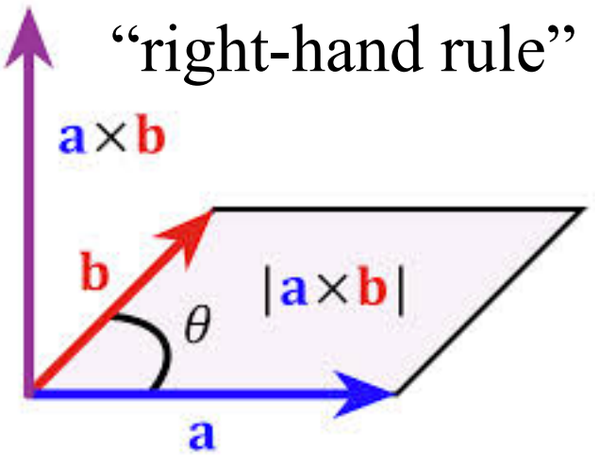
Review: dot and cross products

Dot product: $\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}|| \ ||\mathbf{b}|| cos\theta$



Cross product: $\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin\theta \, \mathbf{n}$

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ b_1 a_3 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

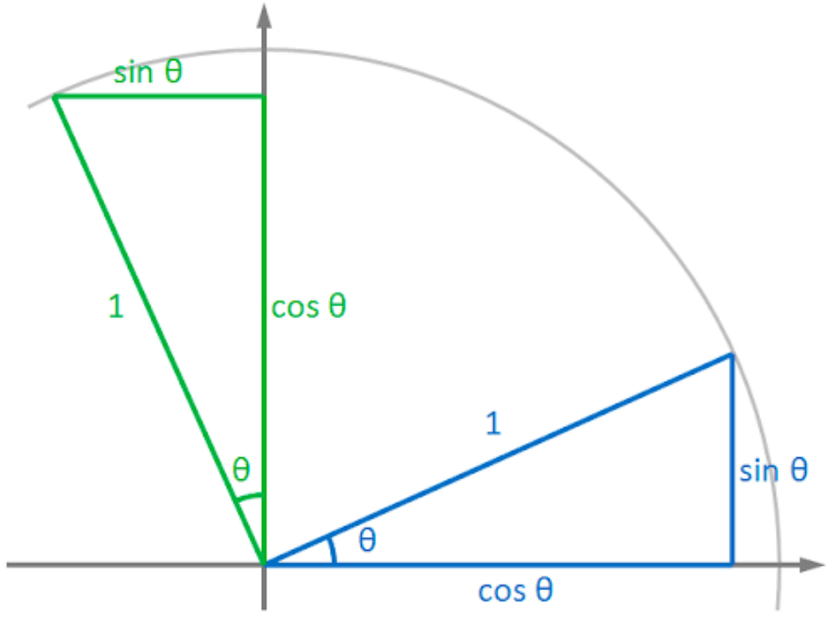


Cross product *matrix*: $\mathbf{a} \times \mathbf{b} = \hat{\mathbf{a}}\mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$

$\hat{a}^T = -\hat{a}$ (skew-symmetric matrix)

Review: 2D Rotations

$$R_\theta \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

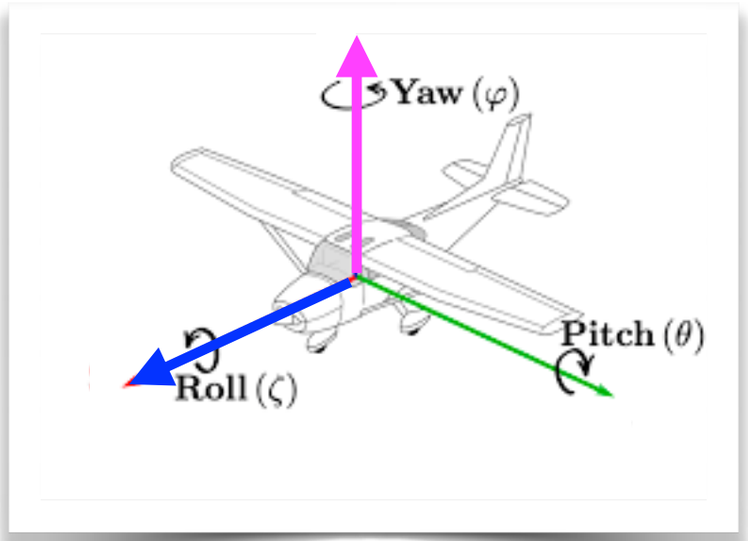


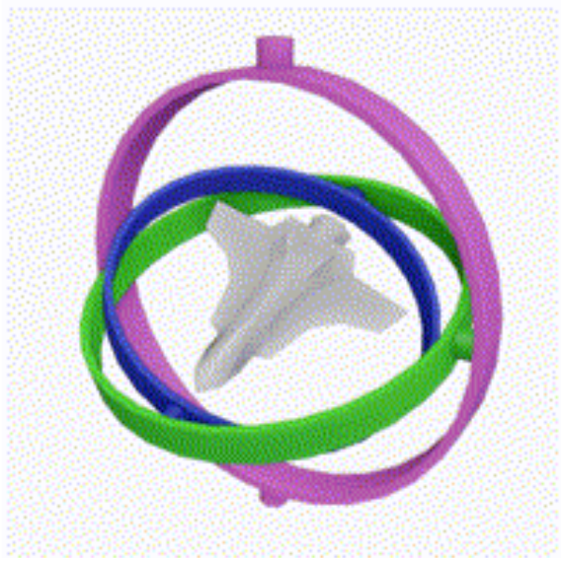
3D Rotations

[Aside] Can’t we just represent as 3 “roll, pitch,yaw” rotations along x,y,z axis?

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

$$R = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) = \overset{\text{yaw}}{\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{\text{pitch}}{\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}} \overset{\text{roll}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}}$$





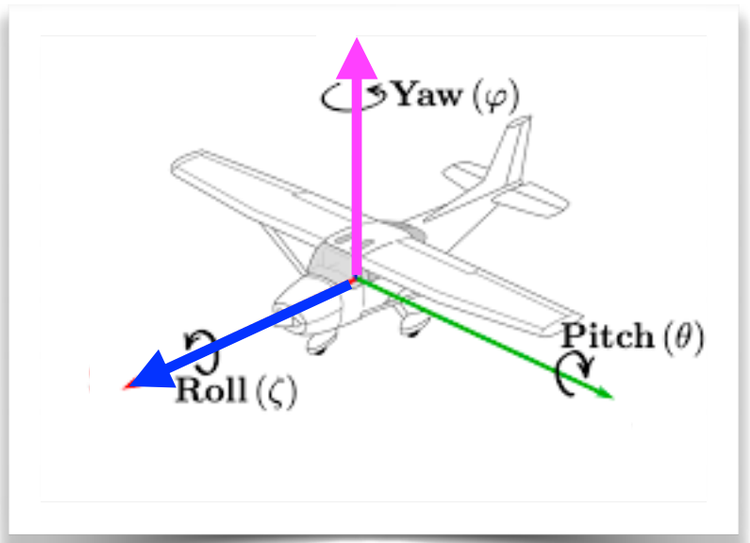
Problem: what happens to yaw and roll axes when pitch is at 90 degrees?

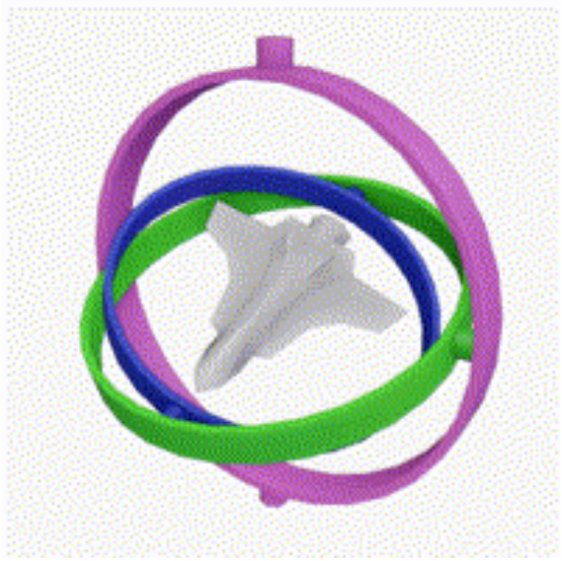
3D Rotations

[Aside] Can’t we just represent as 3 “roll, pitch,yaw” rotations along x,y,z axis?

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

$$R = R_z(\alpha)\,R_y(\beta)\,R_x(\gamma) = \overset{\text{yaw}}{\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}} \overset{\text{pitch}}{\begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}} \overset{\text{roll}}{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}}$$





Problem: what happens to yaw and roll axes when pitch is at 90 degrees?

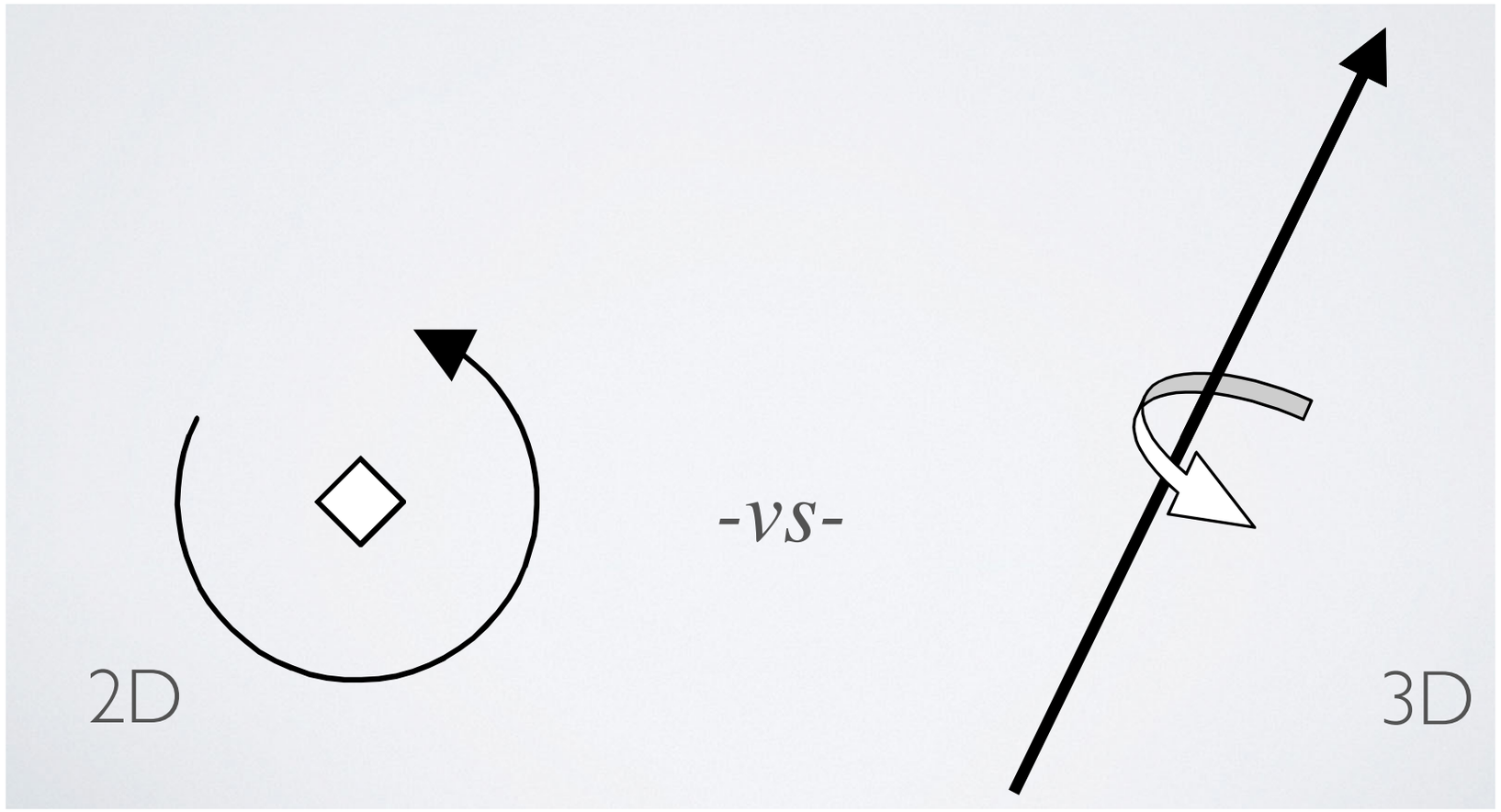
Gimbal lock: lose a degree of freedom since yaw and roll apply same rotation

3D Rotations

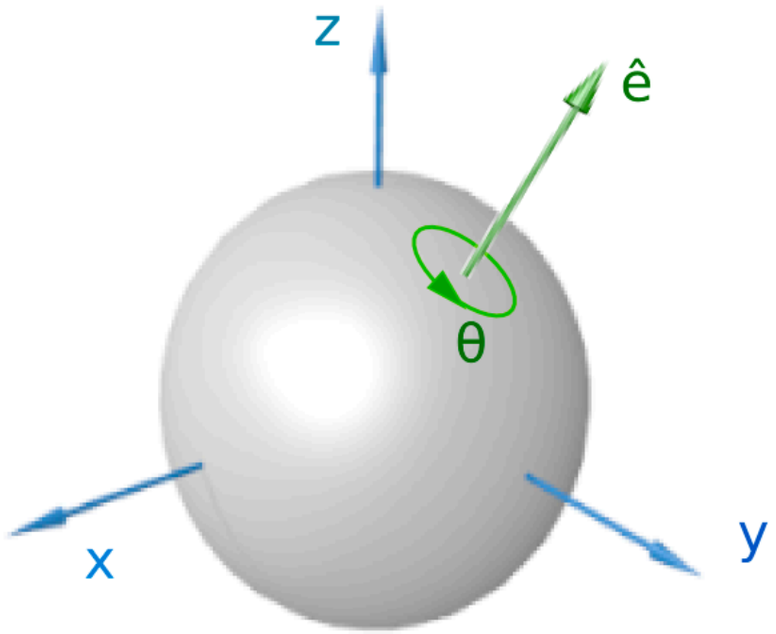
Lots of parameterizations that try to capture 3 DOFs

Helpful ones for vision: *orthonormal matrix*, *axis-angle, exponential maps*, quaternions

Represent a 3D rotation with a unit vector pointed along the axis of rotation, and an angle of rotation about that vector



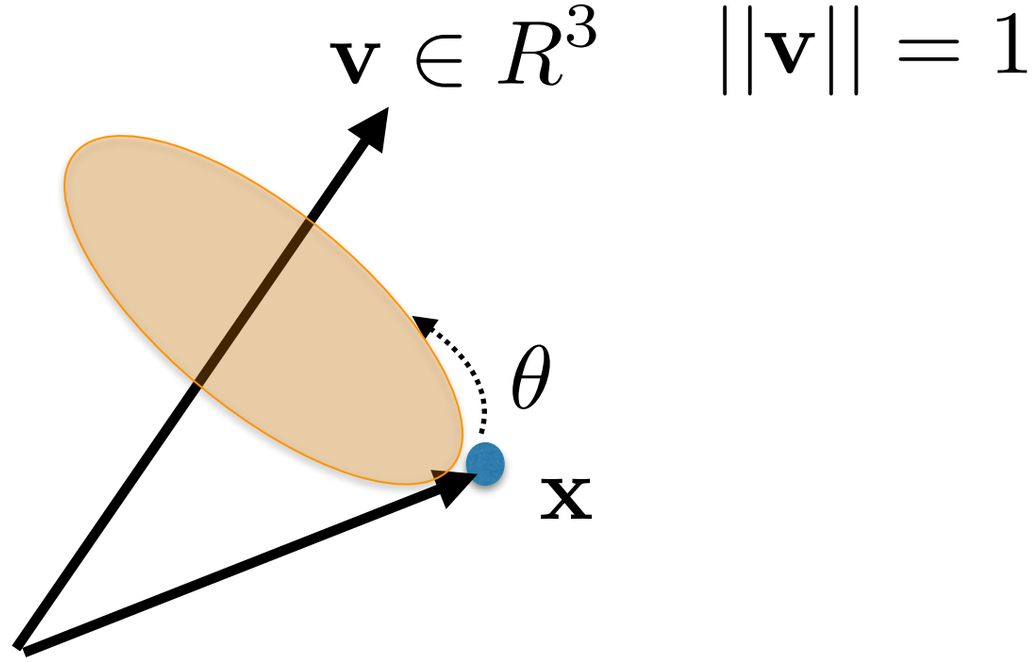
Background:
euler's rotation theorm



Any rotation of a rigid body in a three-dimensional space is equivalent to a pure rotation about a single fixed axis

https://en.wikipedia.org/wiki/Euler's_rotation_theorem

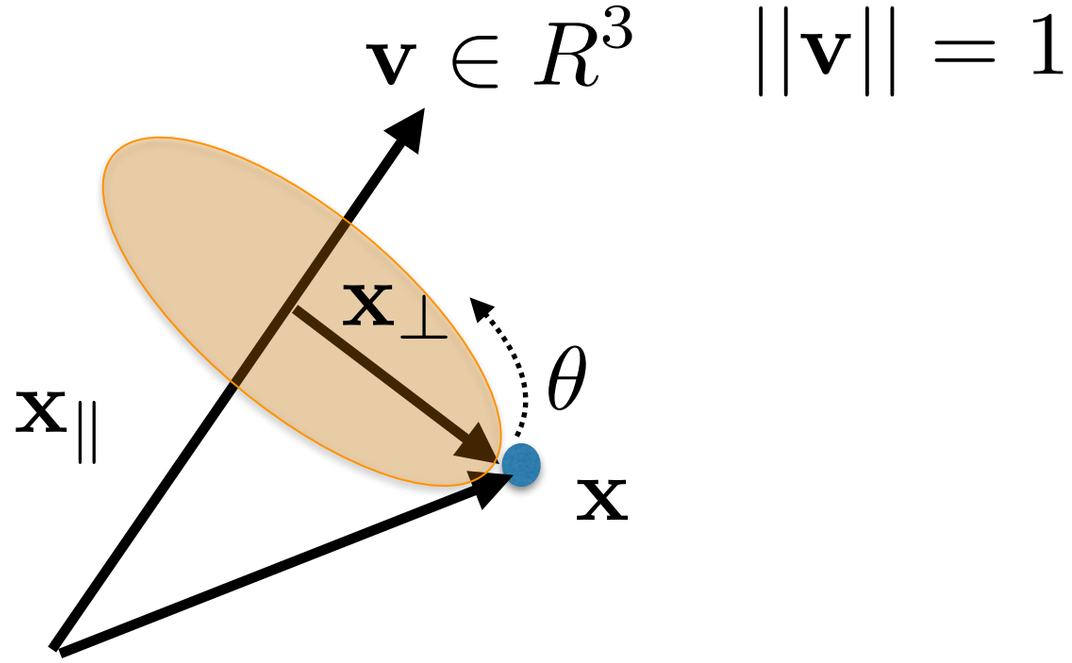
Axis-angle representations of rotations



https://en.wikipedia.org/wiki/Axis-angle_representation

Background: Rodrigues' rotation formula

https://en.wikipedia.org/wiki/Rodrigues'_rotation_formula_rotation_formula



1. Write as **x** as sum of parallel and perpendicular component to **v**

2. Rotate perpendicular component by 2D rotation of theta in plane orthogonal to **v**

$$R = I + \hat{\mathbf{v}} \sin\theta + \hat{\mathbf{v}}\hat{\mathbf{v}}(1 - \cos\theta)$$

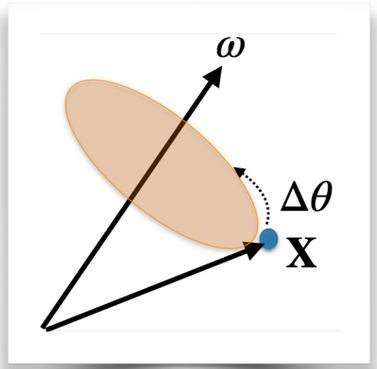
[$R\boldsymbol{x}$ can simplify to cross and dot product computations]

Why is this convenient for vision? Straightforward to compute derivatives of R with respect to θ and $\hat{\mathbf{v}}$.

Angular velocity

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

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



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

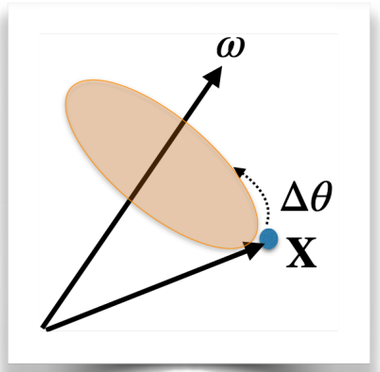
Claim: one can approximate the change in position of **X** due to a small rotation as a *translation*:

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

Angular velocity

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

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



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

Claim: one can approximate the change in position of **X** due to a small rotation as a *translation*:

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

"Sketch" of proof: exponentials are solutions of

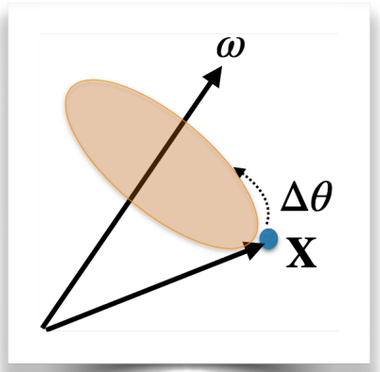
$$\dot{x}(t) = ax(t)$$

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

Angular velocity

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

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



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

Claim: one can approximate the change in position of **X** due to a small rotation as a *translation*:

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

"Sketch" of proof: exponentials are solutions of

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

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

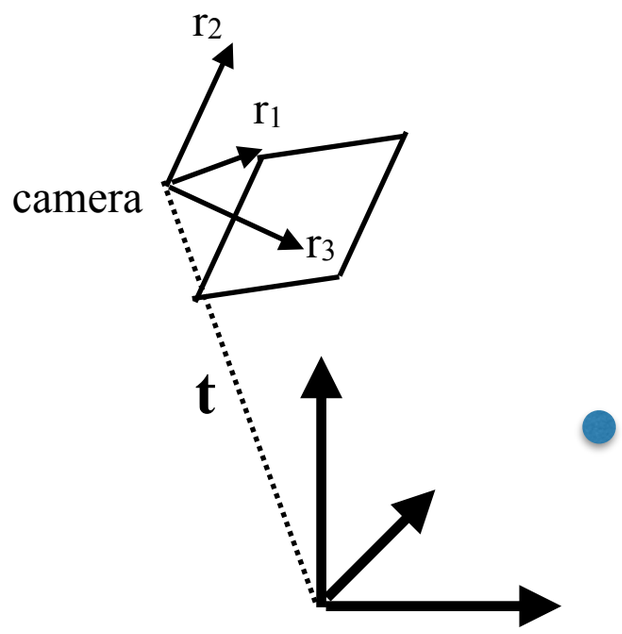
Matrix exponentials are solutions of matrix

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

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

Alternative derivation of an exponential map $R(t) = \exp^{\hat{\omega}t}$

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



Let camera move at translational velocity of **t** and rotate with angular velocity ω

Punchline for the past few slides: the apparant 3D motion of point **X** in the camera coordinate frame is given by

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

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

Gritty details

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

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

Gritty details

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

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

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

Gritty details

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

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

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

$$\begin{aligned}
\dot{x} &= \frac{1}{Z^2}[-t_x Z - \omega_y ZZ - \omega_z YZ + t_z X + \omega_x YX + \omega_y XX] \\
&= \frac{1}{Z^2}[-t_x Z + t_z X] + \frac{1}{Z^2}[\omega_x YX - \omega_y(ZZ + XX) + \omega_z YZ] \\
&= \frac{1}{Z}[-t_x + t_z x] + [w_x xy - \omega_y(1 + x^2) + \omega_z y] \\
&= \frac{1}{Z}\begin{bmatrix}-1 & 0 & x\end{bmatrix}\begin{bmatrix}t_x\\ t_y\\ t_z\end{bmatrix} + \begin{bmatrix}xy & -(1+x^2) & y\end{bmatrix}\begin{bmatrix}\omega_x\\ \omega_y\\ \omega_z\end{bmatrix}
\end{aligned}$$

Gritty details

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

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

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

$$\begin{aligned} \dot{x} &= \frac{1}{Z^2}[-t_x Z - \omega_y ZZ - \omega_z YZ + t_z X + \omega_x YX + \omega_y XX] \\ &= \frac{1}{Z^2}[-t_x Z + t_z X] + \frac{1}{Z^2}[\omega_x YX - \omega_y(ZZ + XX) + \omega_z YZ] \\ &= \frac{1}{Z}[-t_x + t_z x] + [w_x xy - \omega_y(1 + x^2) + \omega_z y] \\ &= \frac{1}{Z}\begin{bmatrix} -1 & 0 & x \end{bmatrix}\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} + \begin{bmatrix} xy & -(1+x^2) & y \end{bmatrix}\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \end{aligned}$$

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

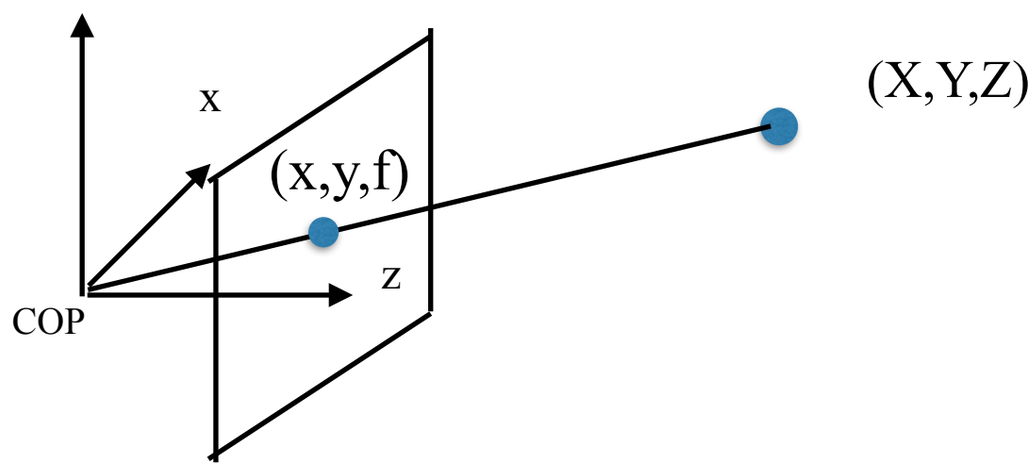
Egomotion optical flow

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

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

$$u(x,y) = \frac{1}{Z(x,y)}(-t_x + xt_z) + xy\omega_x - (1+x^2)\omega_y + y\omega_z$$

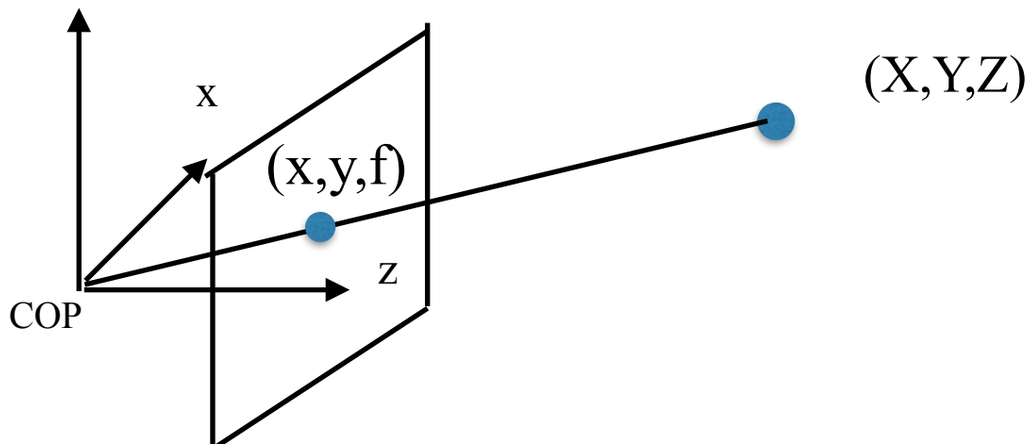
$$v(x,y) = \frac{1}{Z(x,y)}(-t_y + yt_z) + (1+y^2)\omega_x - xy\omega_y - x\omega_z$$

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

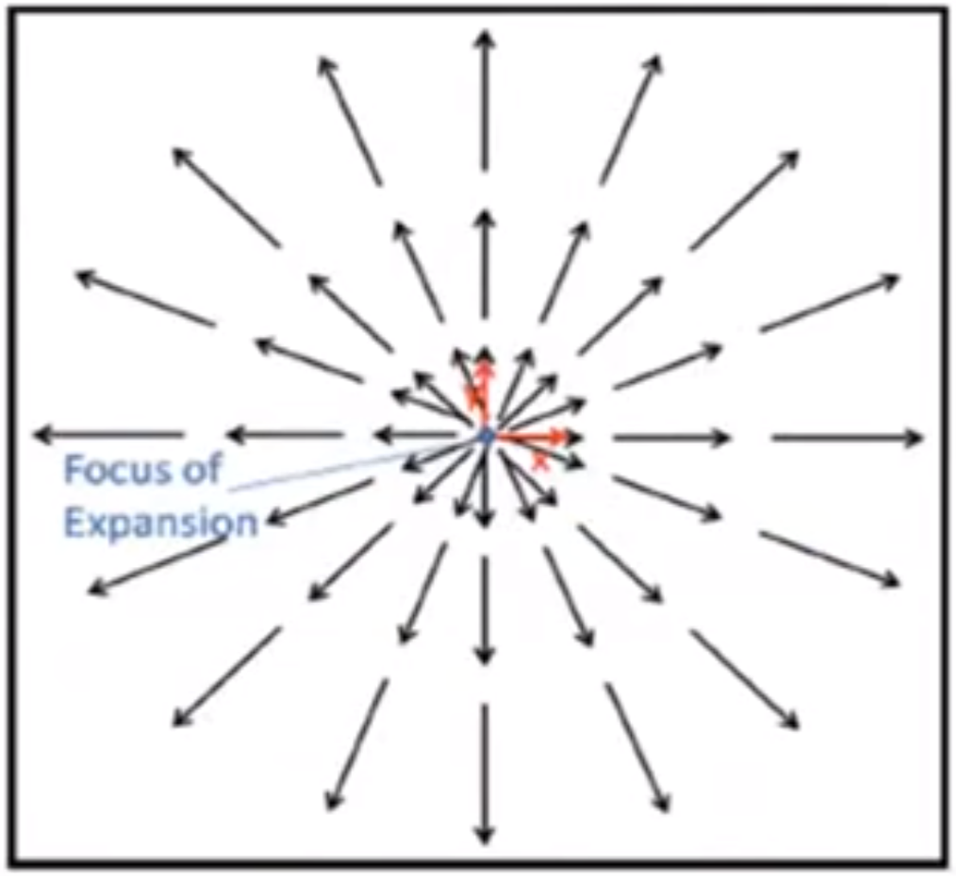
$$u(x,y) = \frac{-t_x + xt_z}{Z(x,y)}, v(x,y) = \frac{-t_y + yt_z}{Z(x,y)}. \tag{3.5}$$

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

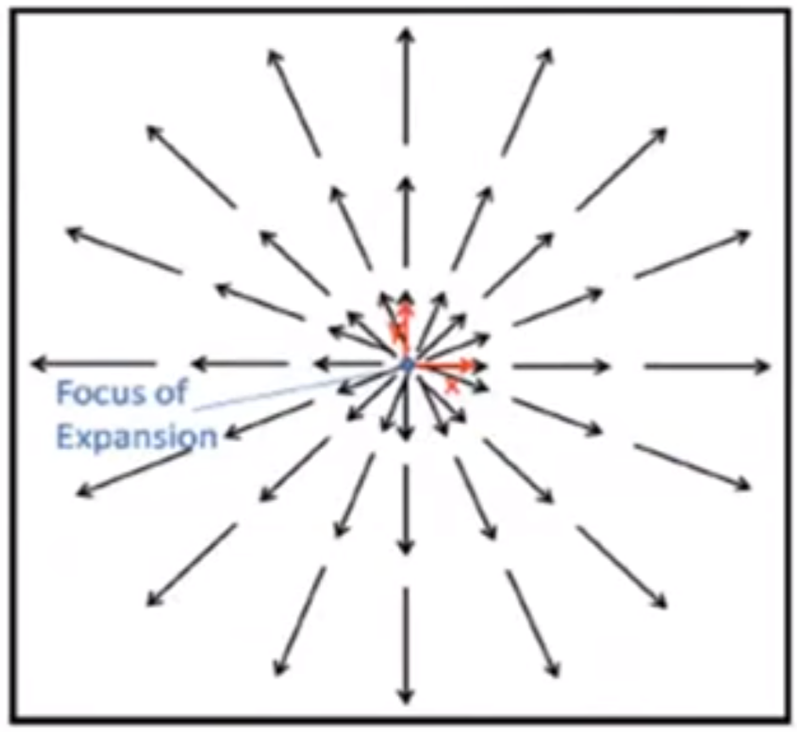


$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \frac{t_z}{Z(x,y)} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{at origin}$$

Optical flow vector is a scalar multiple of position vector

(where scalar depends upon speed of camera translation *and* depth of scene point)

Moving toward a wall

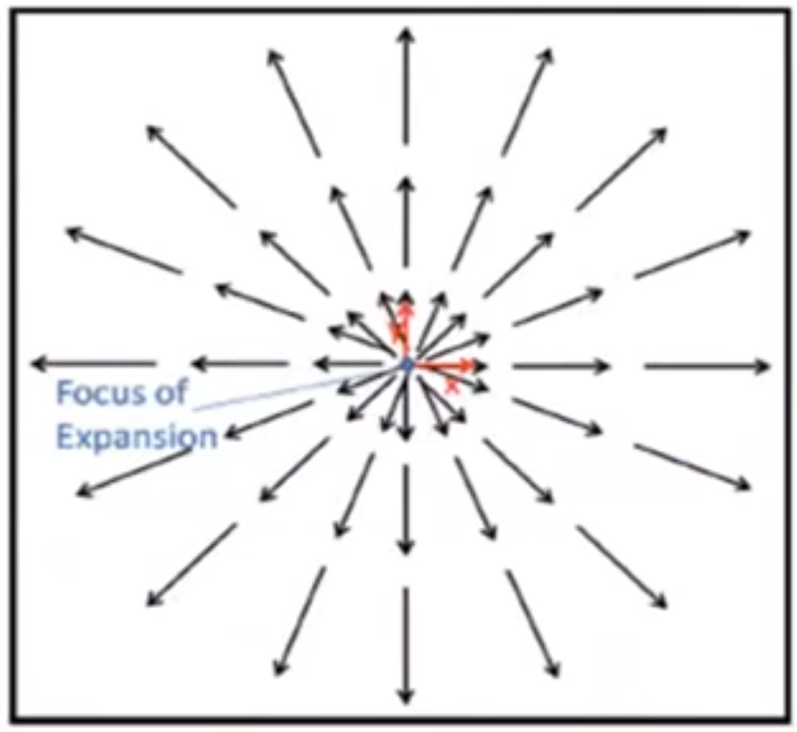


$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \frac{t_z}{Z} \begin{bmatrix} x \\ y \end{bmatrix}$$

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

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

Moving toward a wall



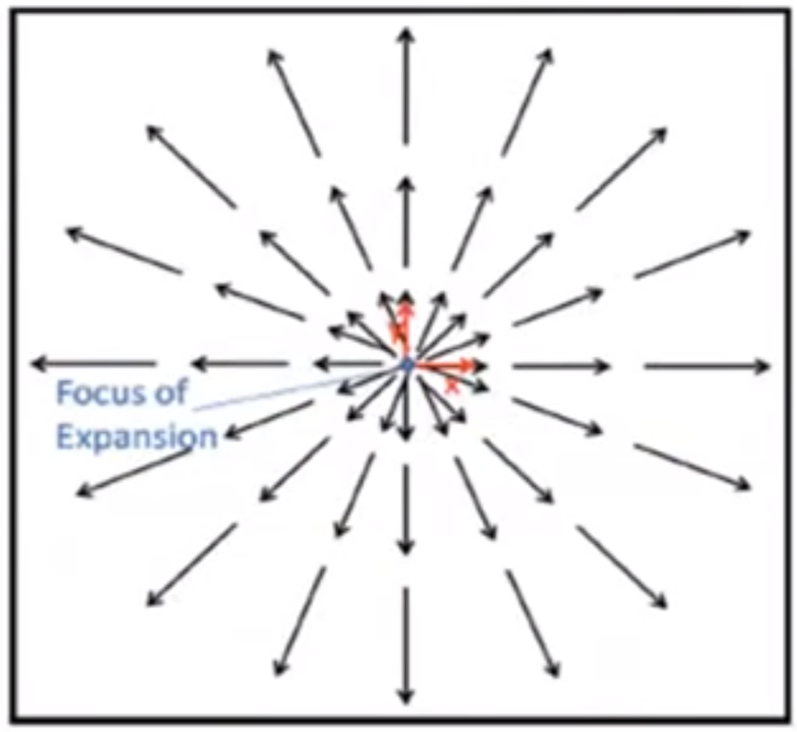
$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \frac{t_z}{Z} \begin{bmatrix} x \\ y \end{bmatrix}$$

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?

Moving toward a wall



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

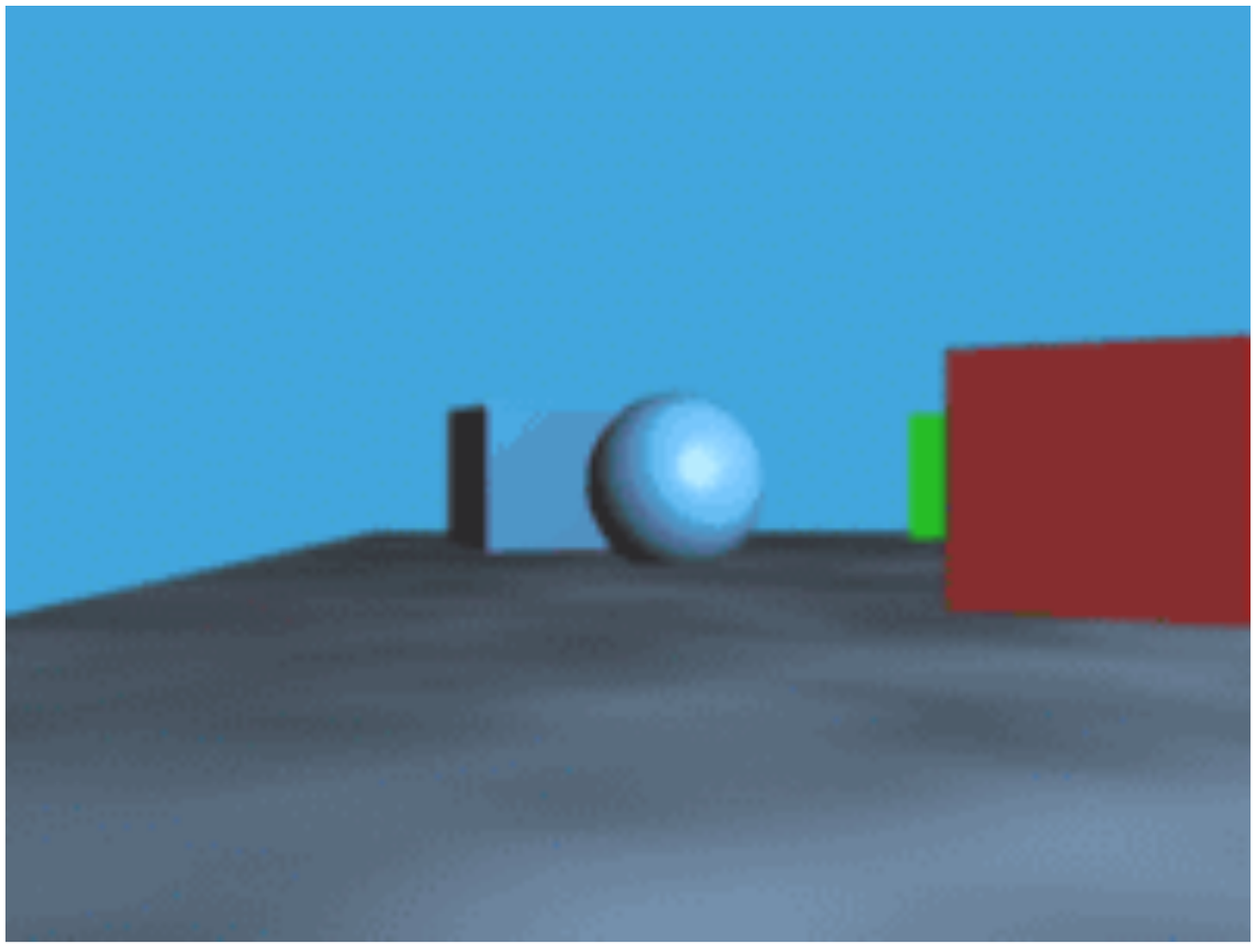
If speed (t_z) and distance (Z) to wall doubles, what happens to flow?

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 can be computed from 2D flow!

Optical flow for translation along camera's x-axis ($t_y, t_z = 0$)

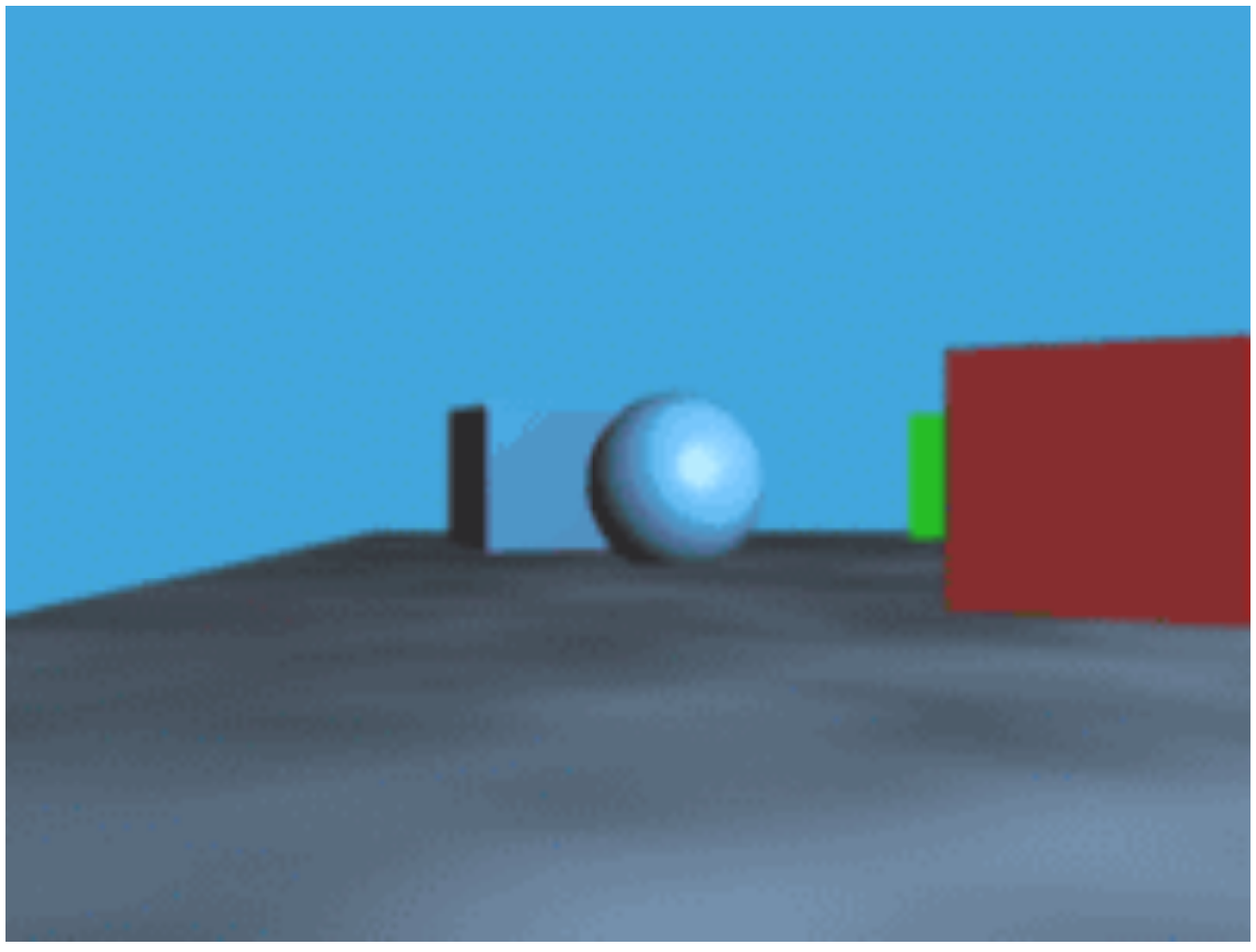


$$u(x,y) = \frac{-t_x + xt_z}{Z(x,y)}, v(x,y) = \frac{-t_y + yt_z}{Z(x,y)}.$$

$$v(x,y) = 0$$

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

Optical flow for translation along camera's x-axis ($t_y, t_z = 0$)



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

$$v(x,y) = 0$$

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

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)?

$$-t_x + x t_2 = 0 \Rightarrow x = \frac{t_x}{t_2}$$

$$-t_y + y t_2 = 0 \Rightarrow y = \frac{t_y}{t_2}$$

At $\left(\frac{t_x}{t_2}, \frac{t_y}{t_2}\right)$, optical flow $\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Implies FOE is projection of translation vector

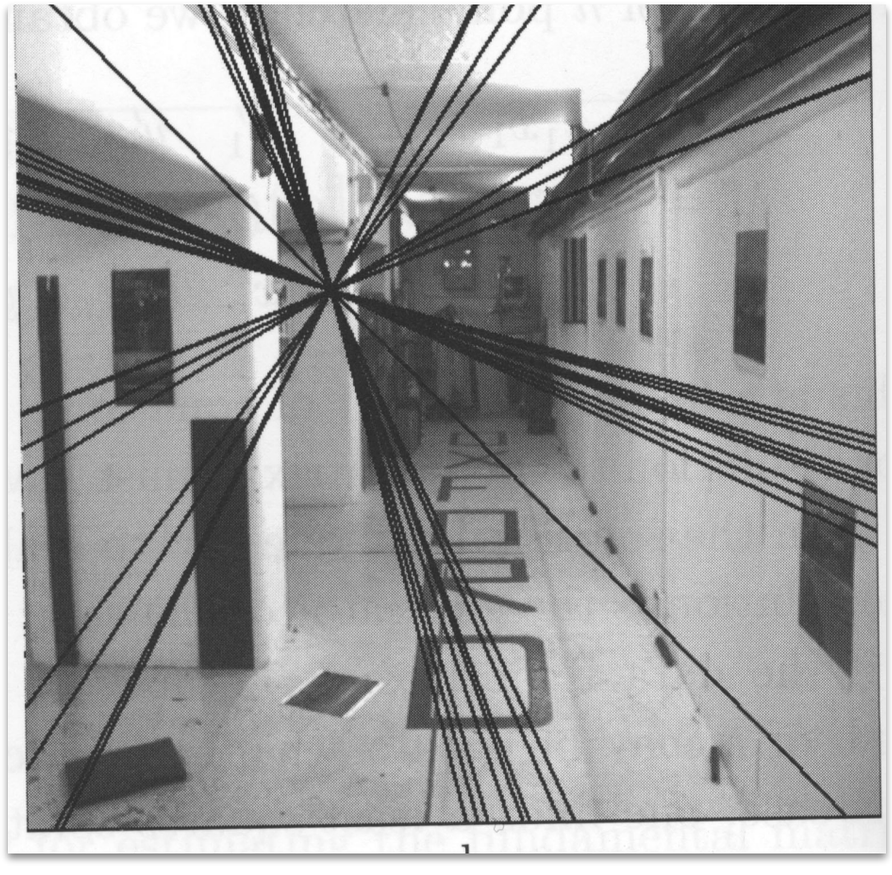
Optical flow radially extends out from field of expansion (FOE)

$$x' = x - \frac{t_x}{t_z}$$

$$y' = y - \frac{t_y}{t_z}$$

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

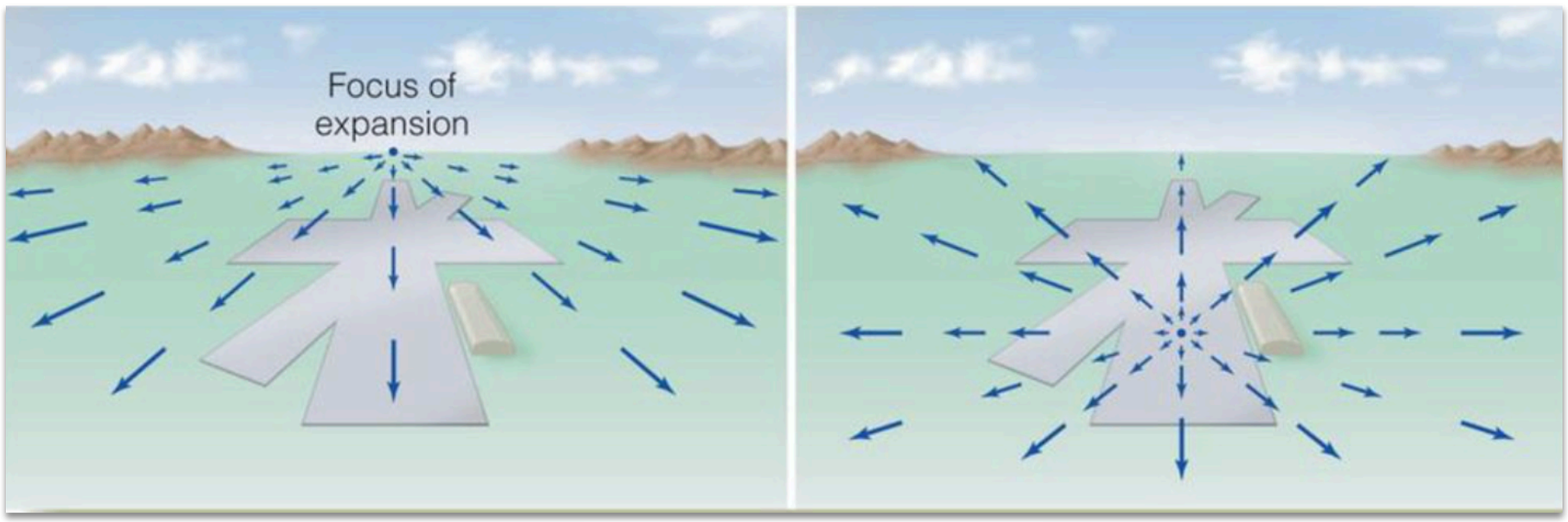
(once we shift pixel coordinates by FOE, we obtain radial flow as before)

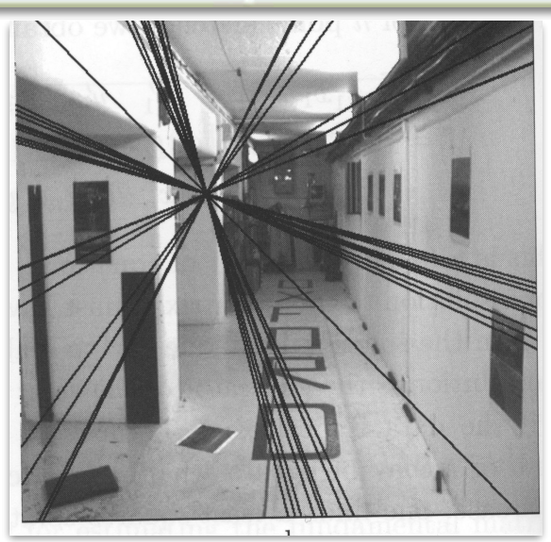


Focus of expansions

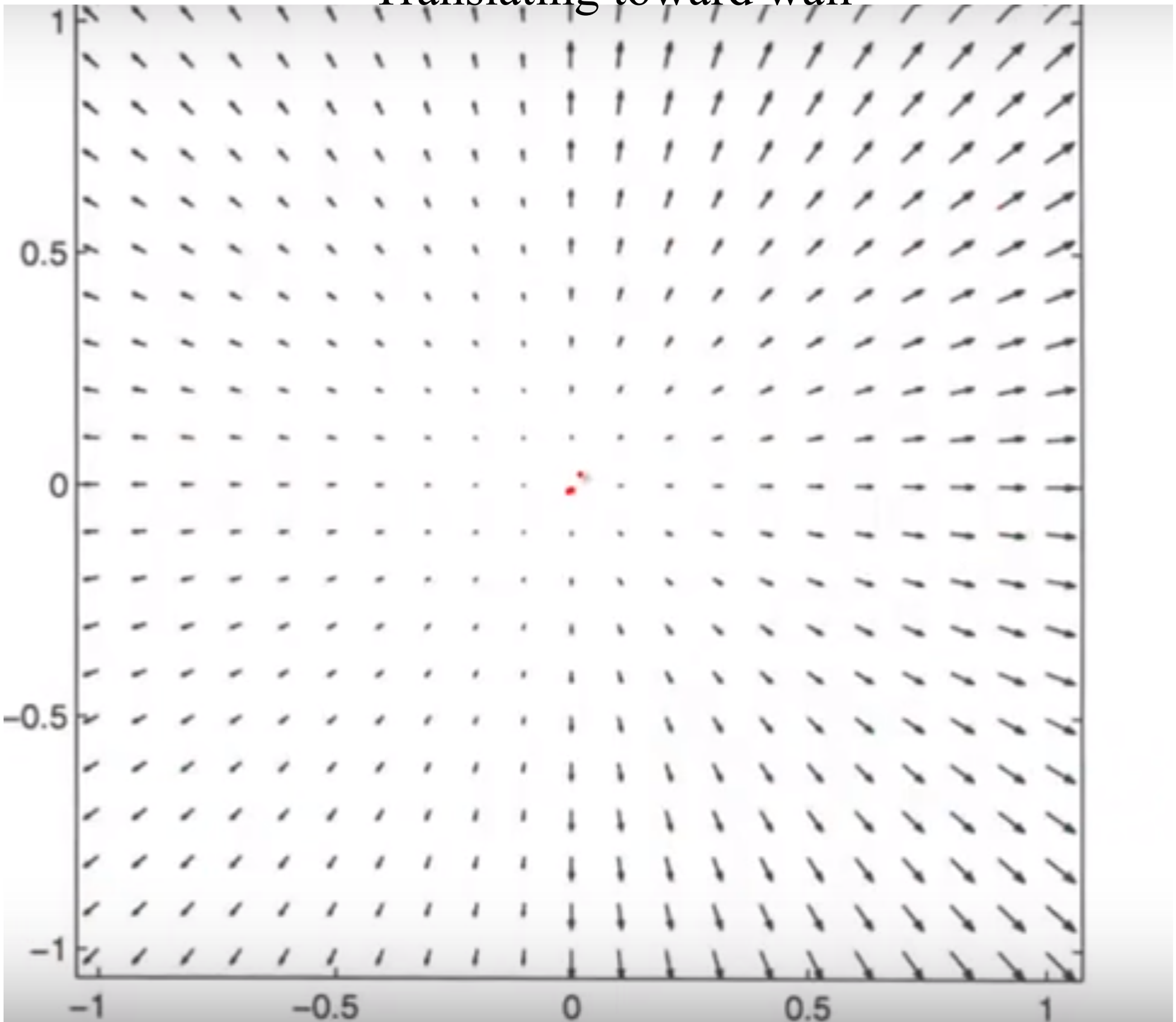
reveal projection of camera translation velocity vector into image

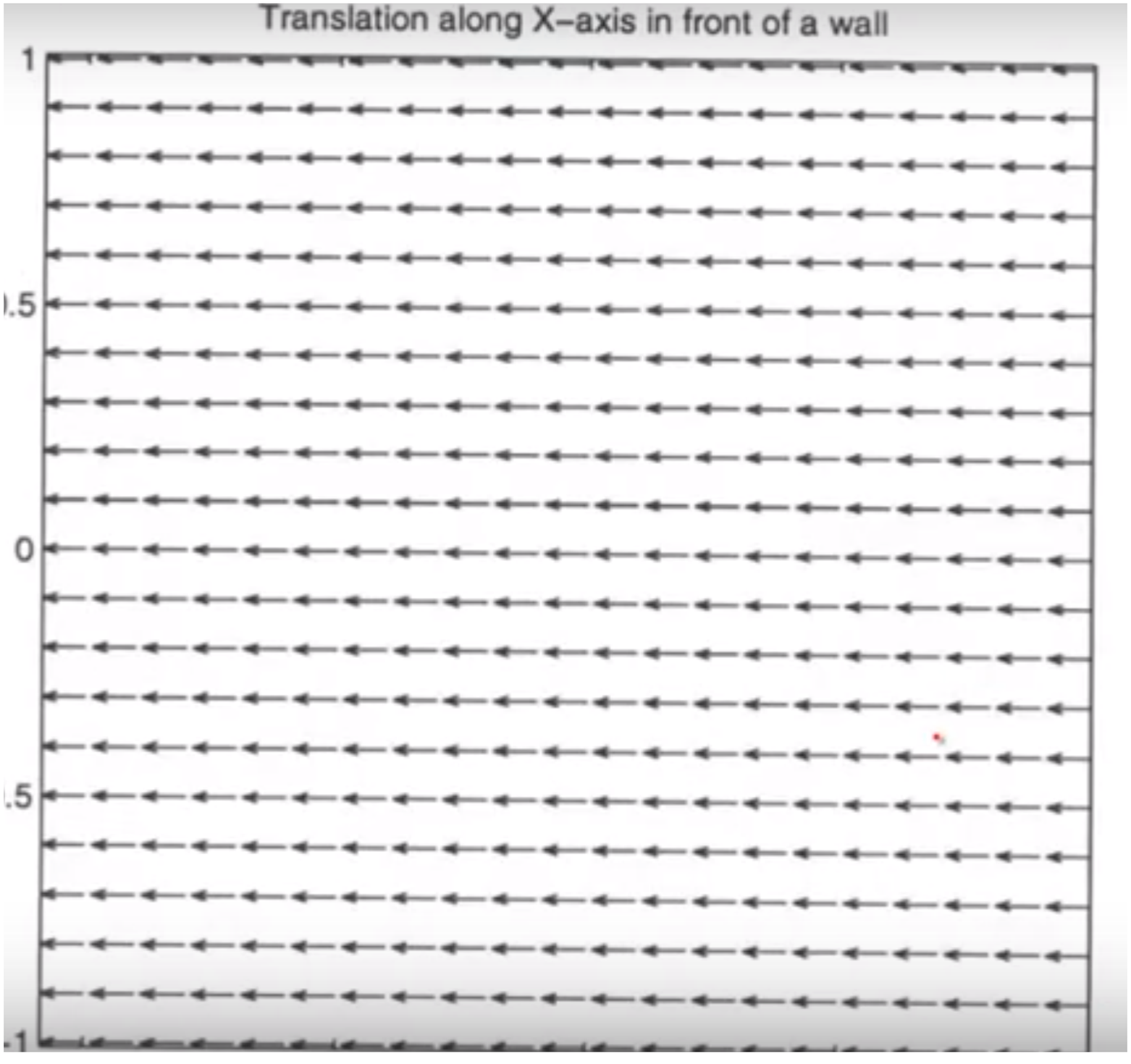
$$\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \Rightarrow \begin{bmatrix} \frac{t_x}{t_z} & \frac{t_y}{t_z} \end{bmatrix}$$



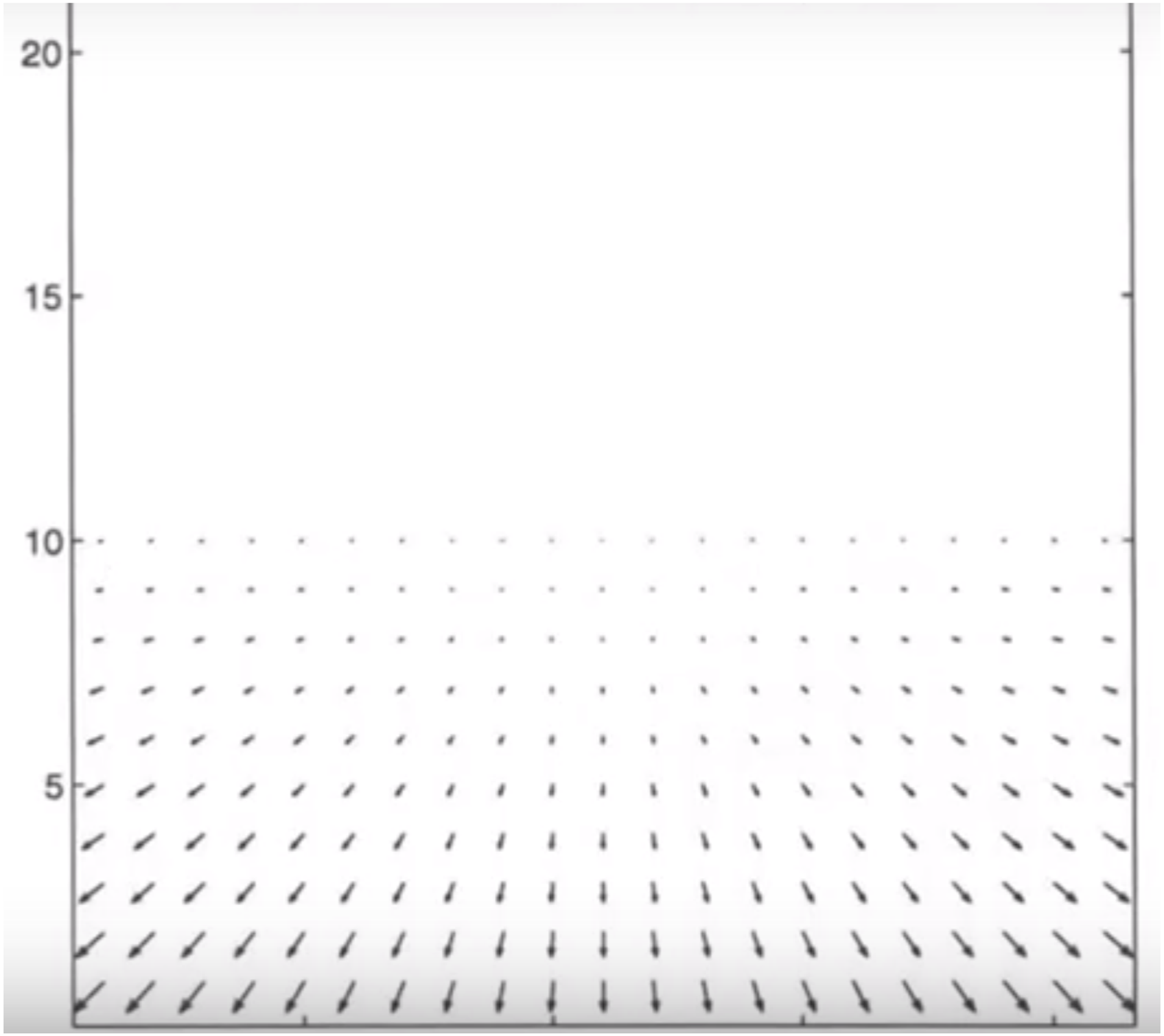


Translating toward wall

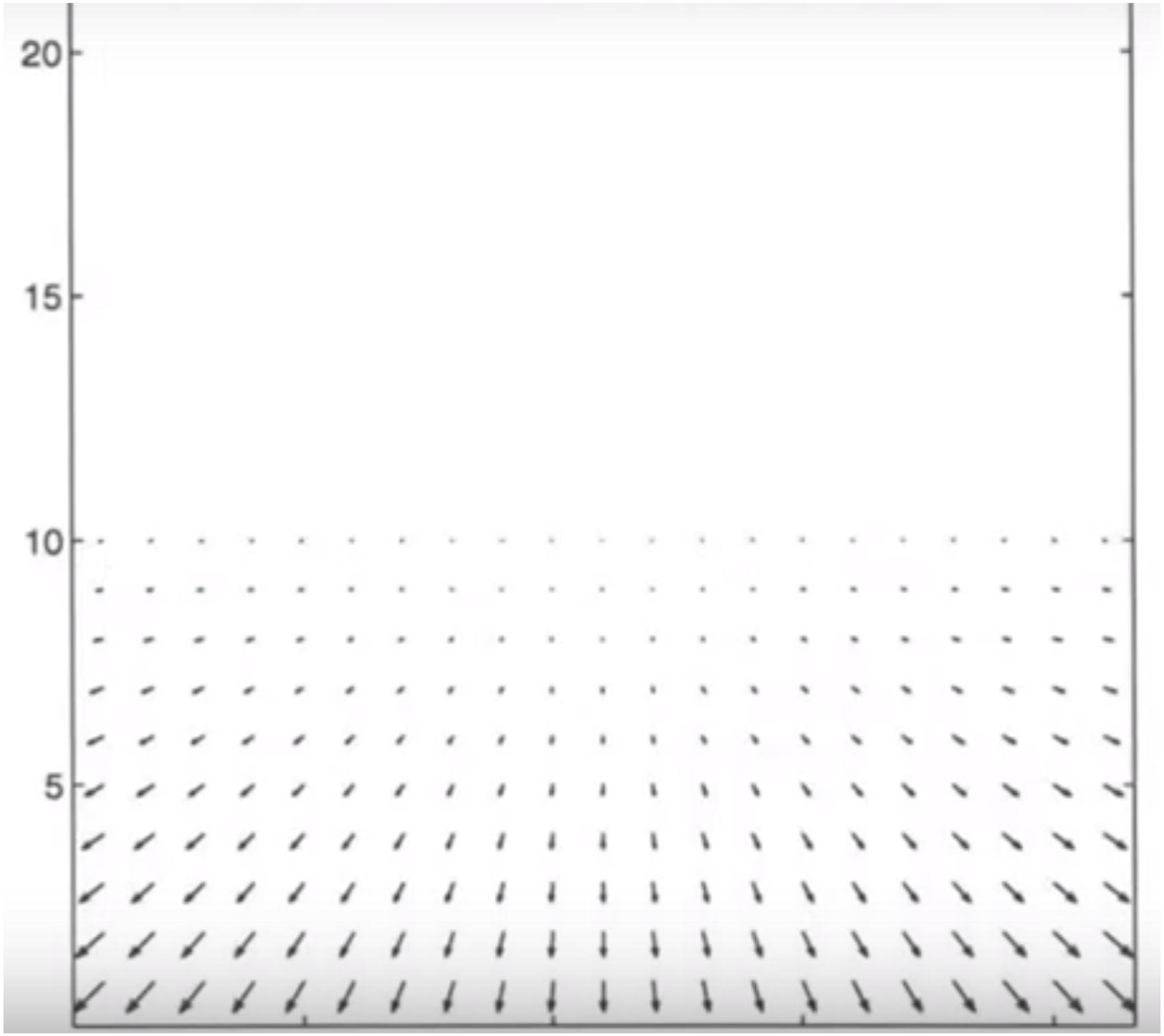



Translation along X–axis in front of a wall
1
.5
0
.5
-1

What kind of motion (and scene) would generate this flow?



What kind of motion (and scene) would generate this flow?



Walking forward along a ground plane

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

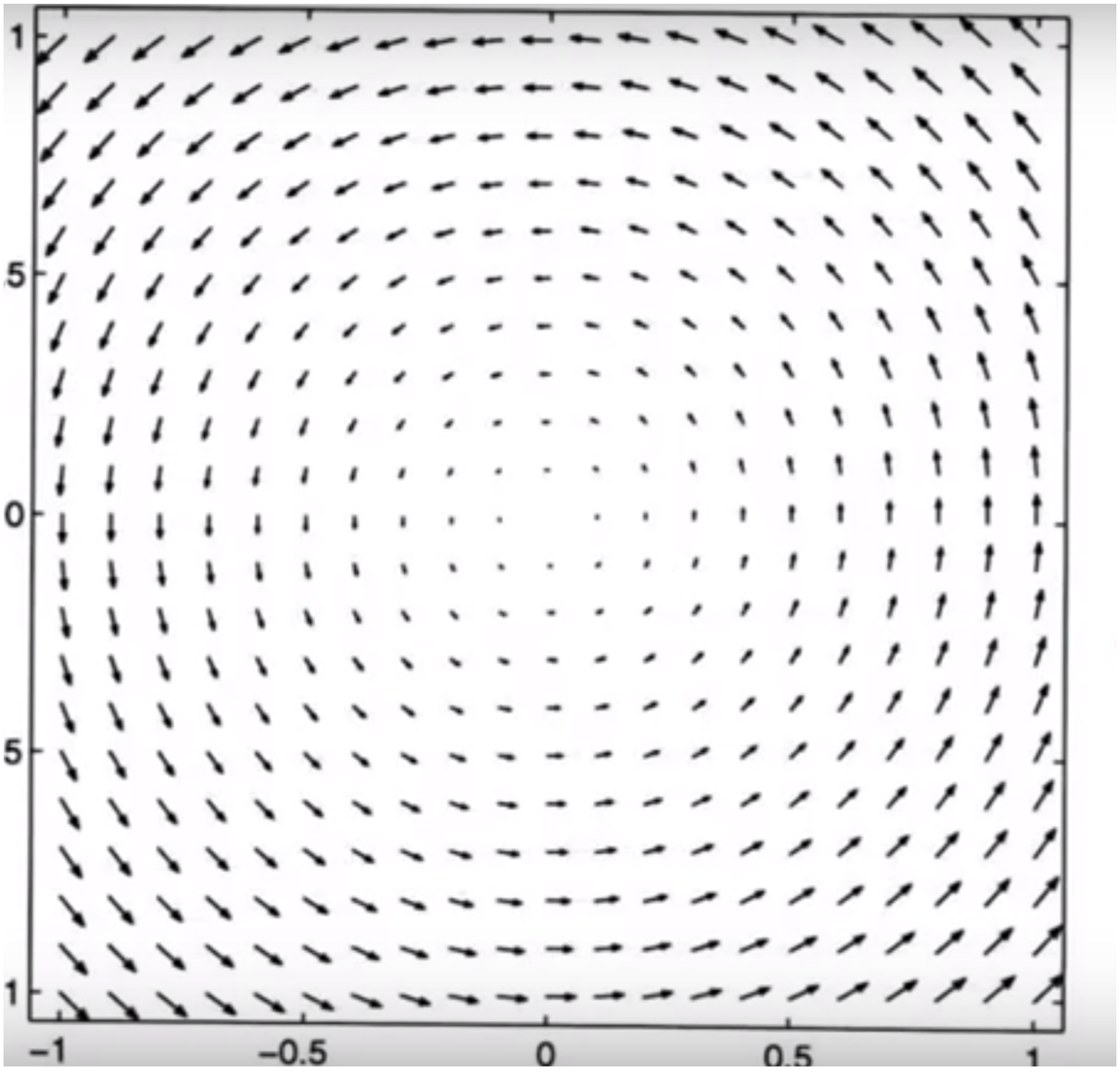
rotation component

Rotational component does not depend on scene depth Z

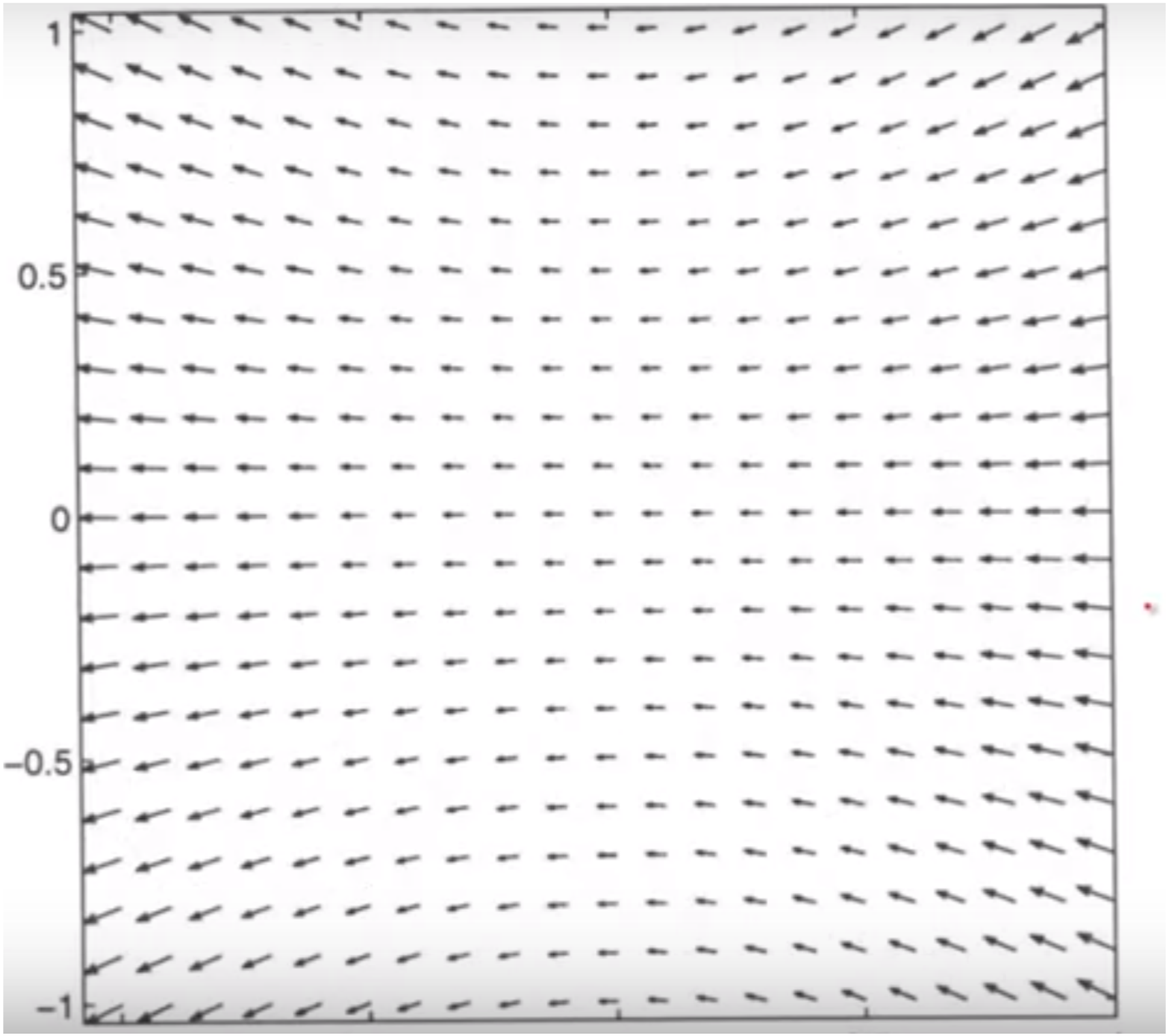
(exploited when fitting a homography to a rotating camera!)

Angular velocity can be recovered from rotational component

Rotating about z-axis



Rotating about y-axis

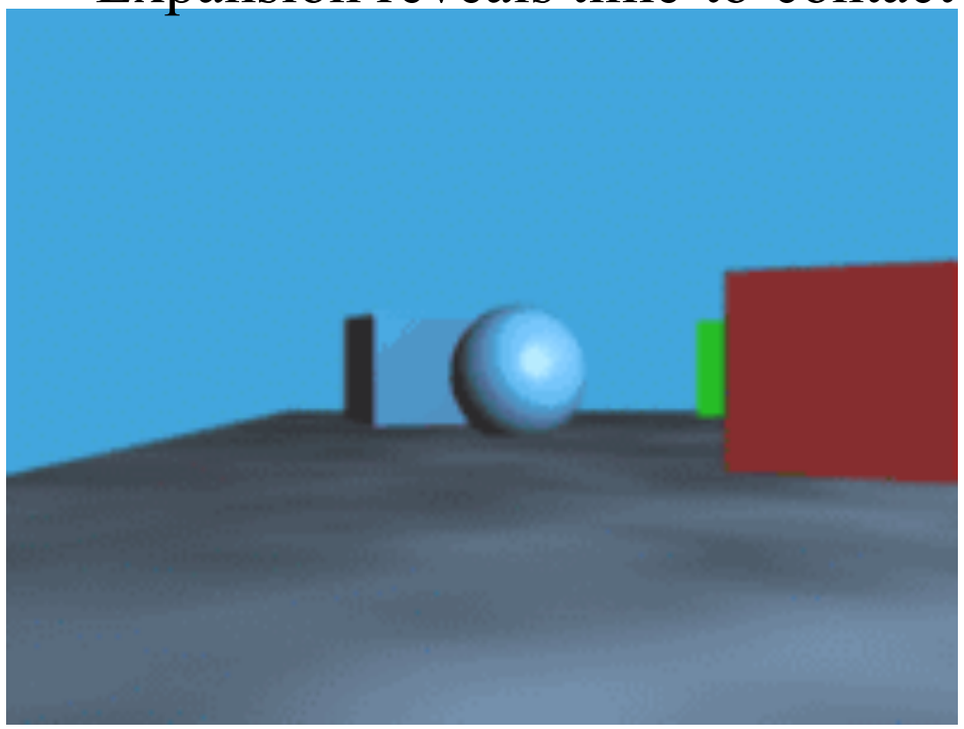


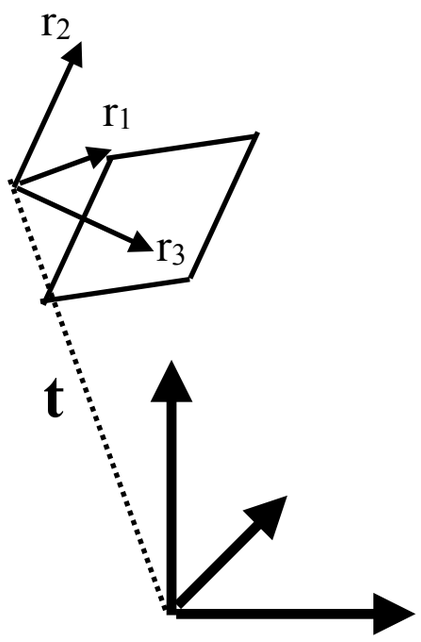
Concluding Remarks

- 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.
- 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.

A look back: deriving image flow from egomotion

- Parallex reveals depth
- Expansion reveals time-to-contact





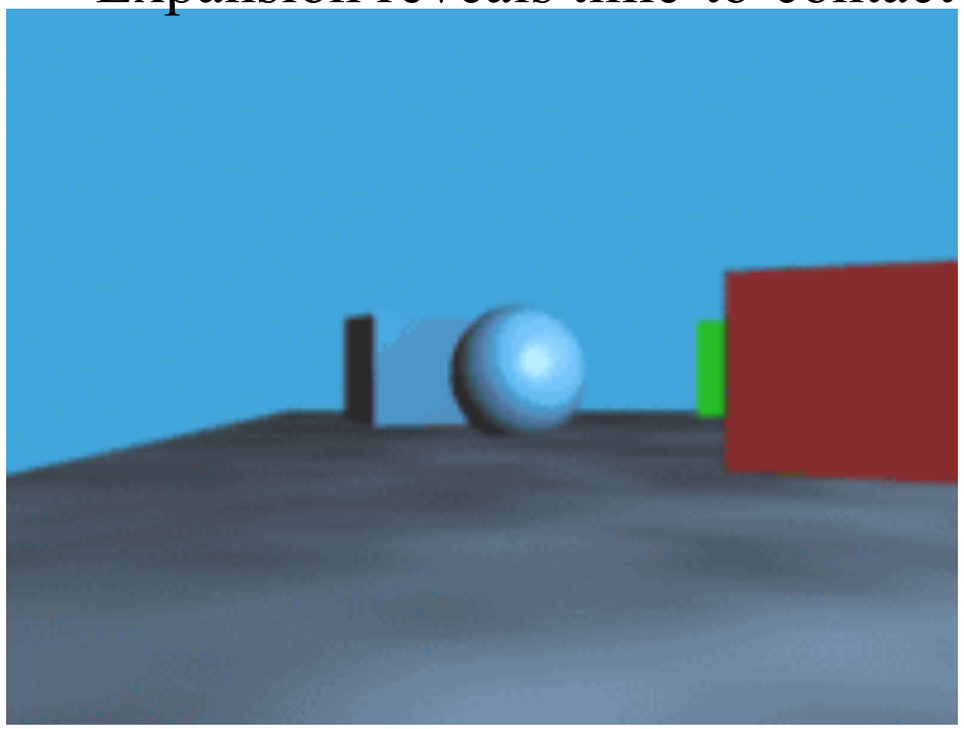
$$\dot{\mathbf{X}} = -\mathbf{t} - \omega \times \mathbf{X}$$

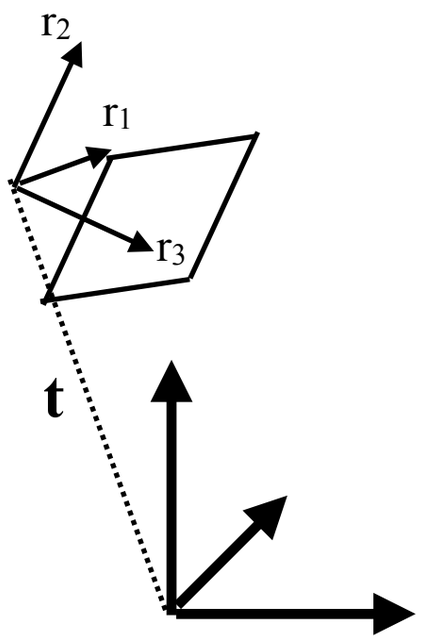
$$\dot{x} = \frac{1}{Z(x,y)}(-t_x + xt_z) +$$

$$\dot{y} = \frac{1}{Z(x,y)}(-t_y + yt_z) + \vdots$$

A look back: deriving image flow from egomotion

- Parallex reveals depth
- Expansion reveals time-to-contact





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

$$\dot{x} = \frac{1}{Z(x,y)}(-t_x + xt_z) +$$

$$\dot{y} = \frac{1}{Z(x,y)}(-t_y + yt_z) + \vdots$$

Outline

- Egomotion
    - Motivation
    - Time-to-Contact, Parallax, Focus-of-Expansion
- **Optical Flow**
    - Logistics
    - Motivation, aperture problem
    - Sparse (KLT) vs Dense (variational)
    - Optimization tools: robust statistics, coarse-to-fine, markov-random feilds
    - Segmentatiion (dominant motion estimation, background subtraction, layered models)

General motion feilds

Why do we want to compute motion fields beyond egomotion?

Simple evidence that “single image” analysis won’t always suffice

General motion feilds

Why do we want to compute motion fields beyond egomotion?

Simple evidence that “single image” analysis won’t always suffice

Moving light sequences

https://www.youtube.com/watch?v=1F5ICP9SYLU

Johansson displays (1971)

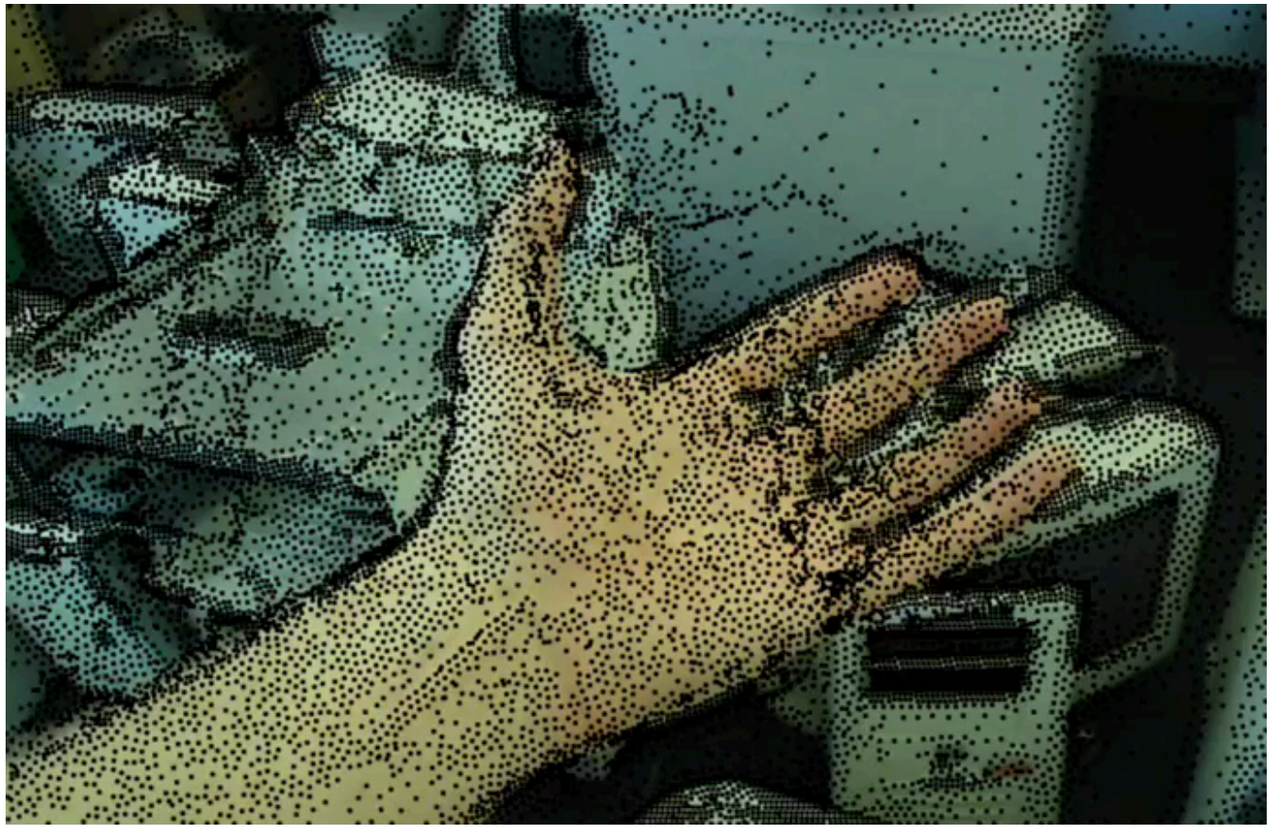
Moving light sequences

https://www.youtube.com/watch?v=1F5ICP9SYLU

Johansson displays (1971)

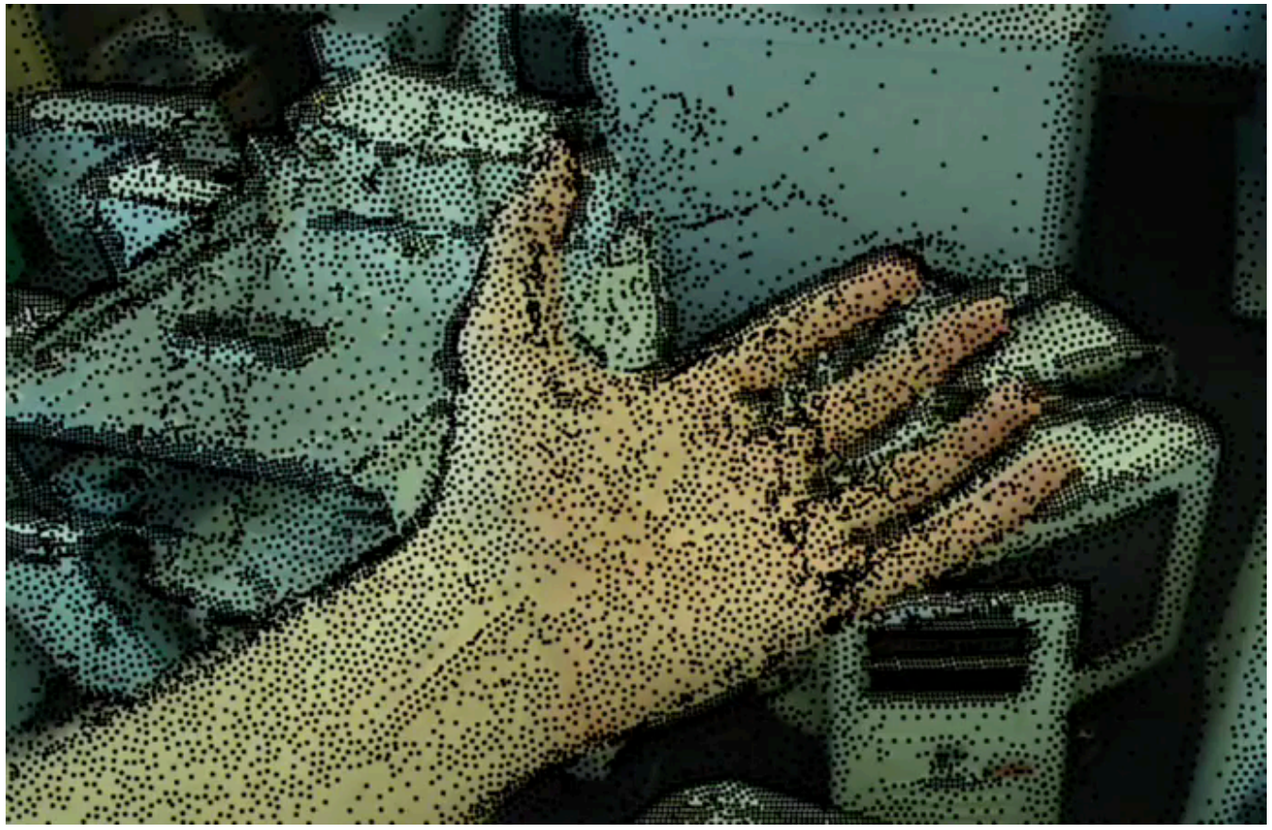
Detailed “particle videos” IJCV 08

(http://rvsn.csail.mit.edu/pv/)

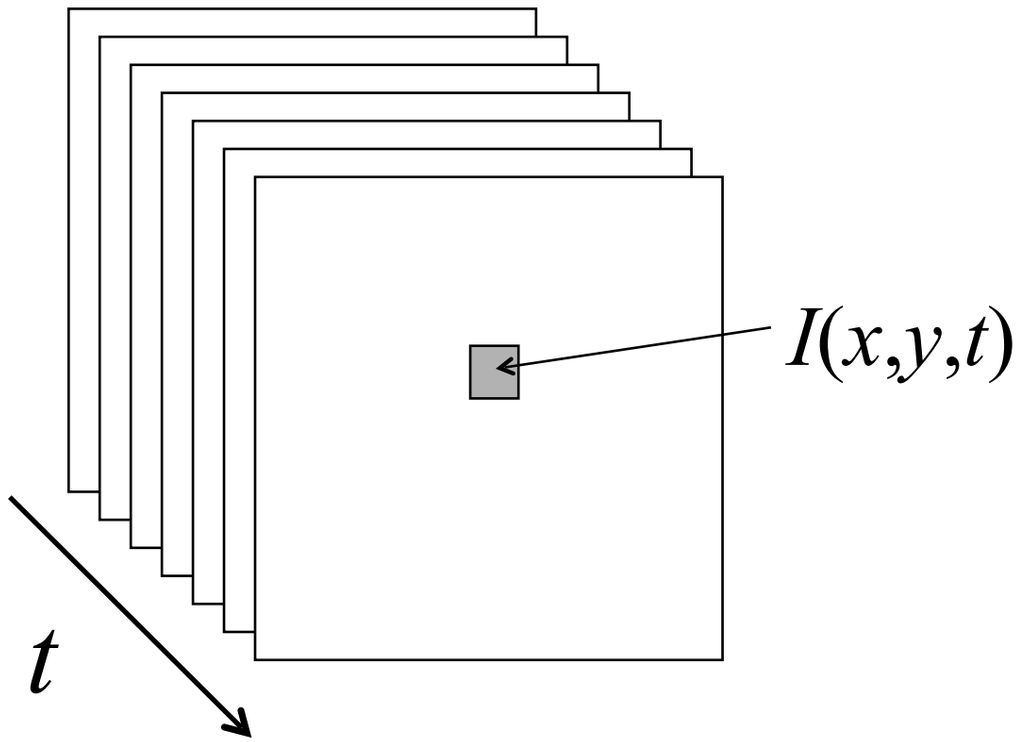


Detailed “particle videos” IJCV 08

(http://rvsn.csail.mit.edu/pv/)



Another perspective we'll embrace today: videos as spacetime cubes



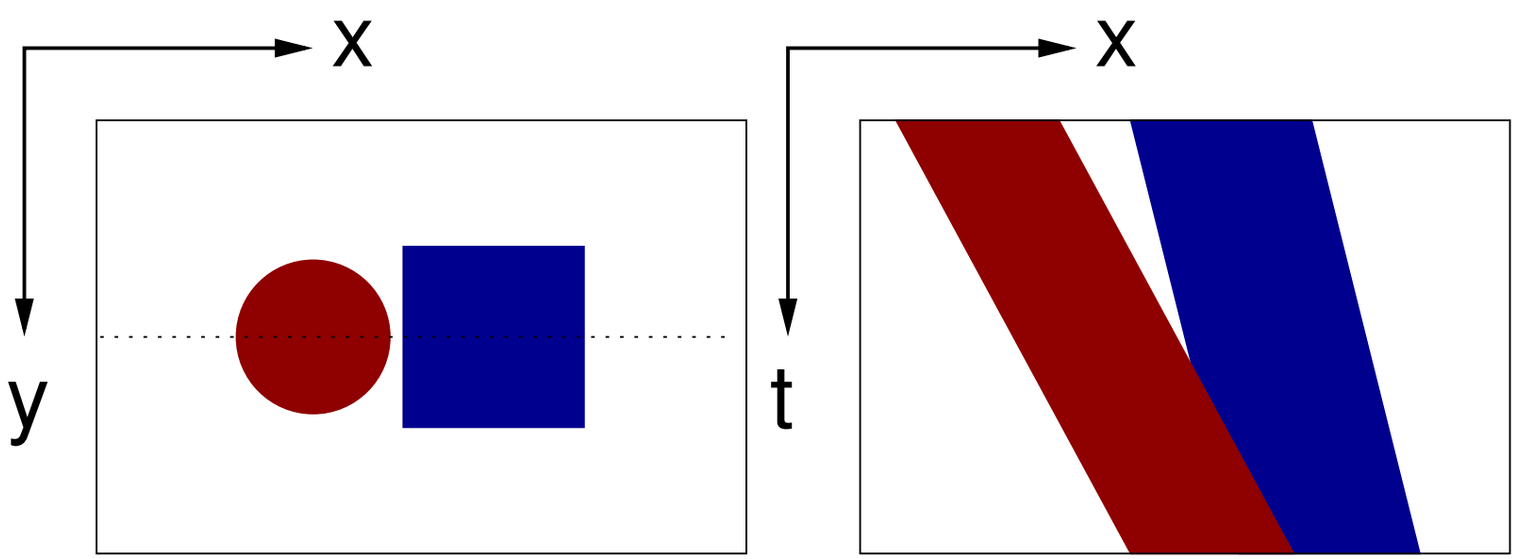
Different approaches:

- "slice" this cube in different ways
- apply 3D frequency analysis
- taylor expansion in (x,y,t)

Challenges:

- can't hold large videos in (GPU) memory
- offline vs (streaming) online processing
- (my hunch) we need to embrace stateful "recurrent" processing

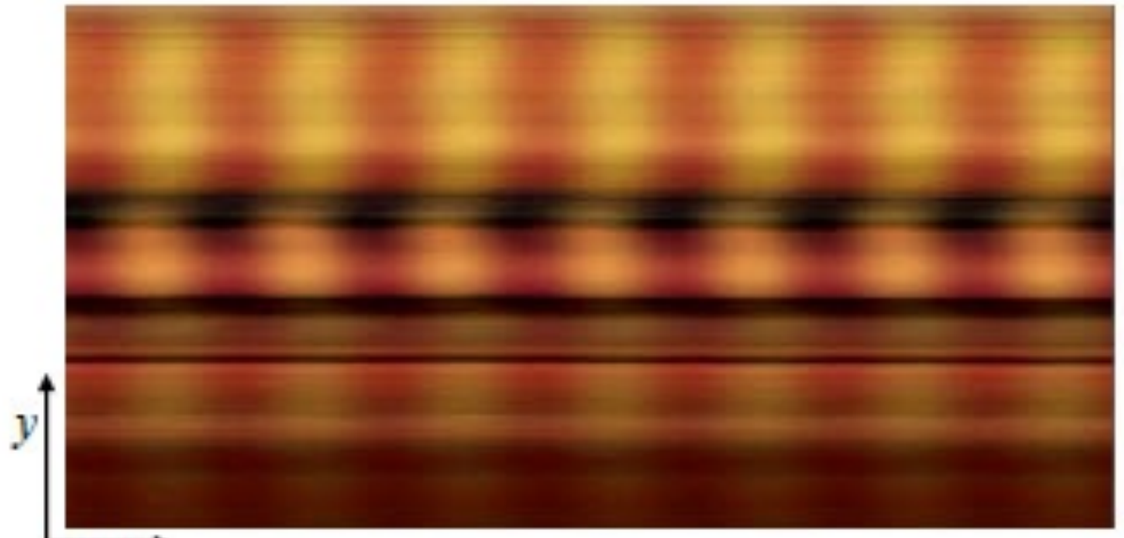
Visualizing spacetime cubes



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

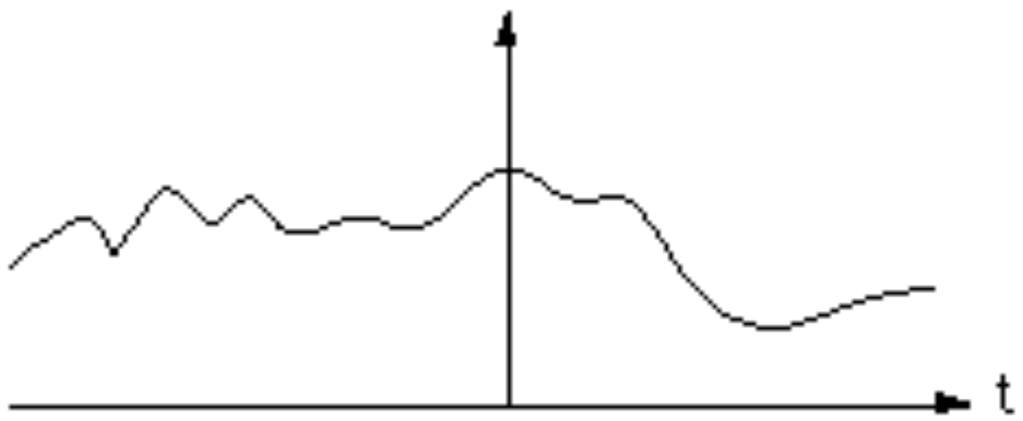
Digression: visualizing space-time cube

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



(c) Spatiotemporal *YT* slices

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



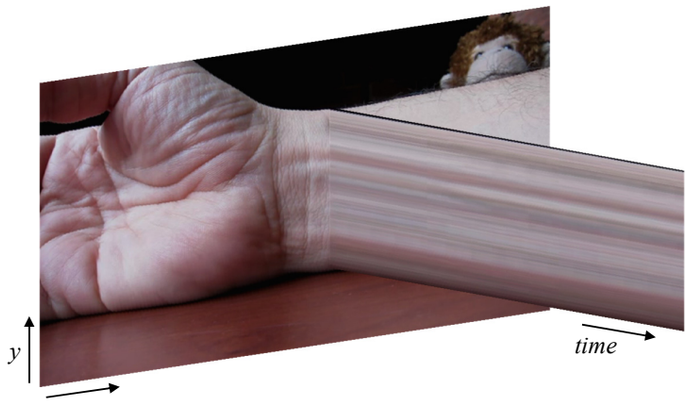
Plot I(x,y,t) for a fixed (x,y)

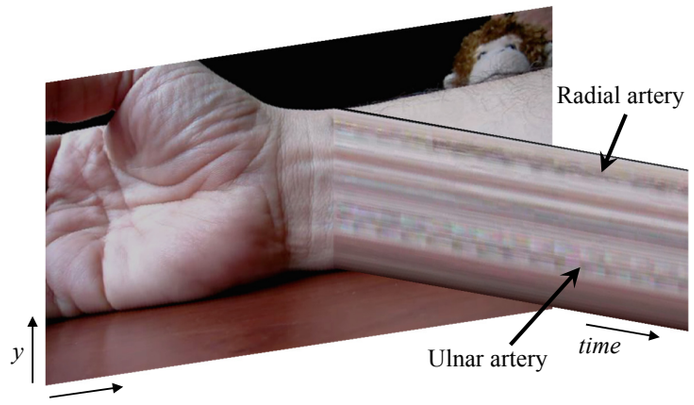
Amplifying temporal signals

By (3D!) frequency modification or by optical flow tracking (today)

Eulerian Video Magnification for Revealing Subtle Changes in the World

Hao-Yu Wu[1] Michael Rubinstein[1] Eugene Shih[2] John Guttag[1] Frédo Durand[1] William Freeman[1]
[1]MIT CSAIL [2]Quanta Research Cambridge, Inc.



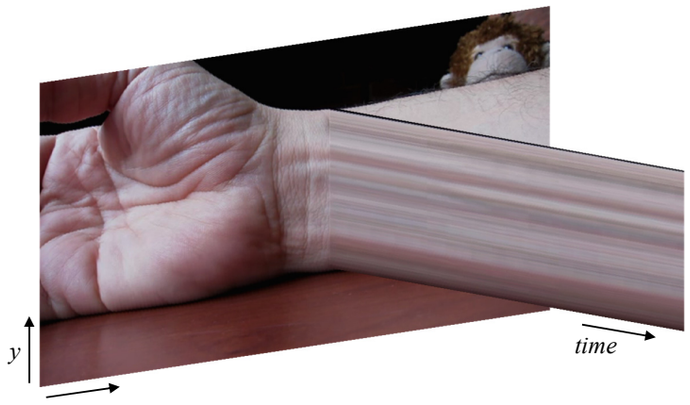


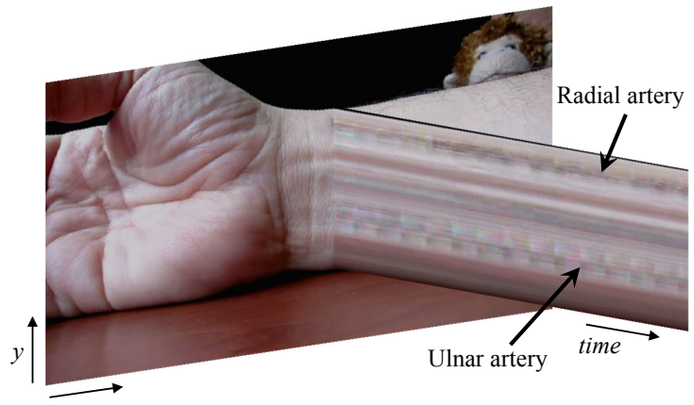
Amplifying temporal signals

By (3D!) frequency modification or by optical flow tracking (today)

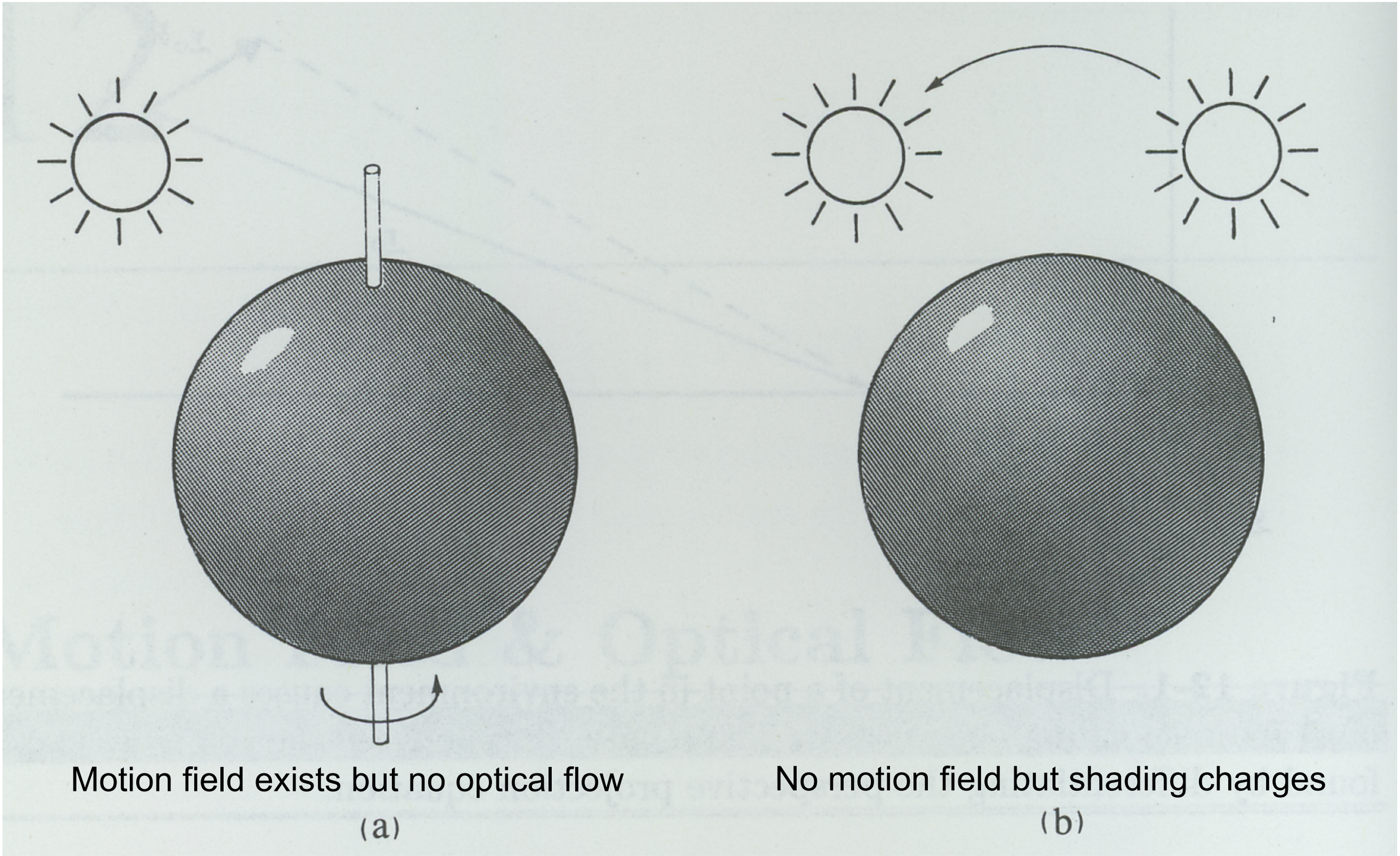
Eulerian Video Magnification for Revealing Subtle Changes in the World

Hao-Yu Wu[1] Michael Rubinstein[1] Eugene Shih[2] John Guttag[1] Frédo Durand[1] William Freeman[1]
[1]MIT CSAIL [2]Quanta Research Cambridge, Inc.





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



Motion field exists but no optical flow

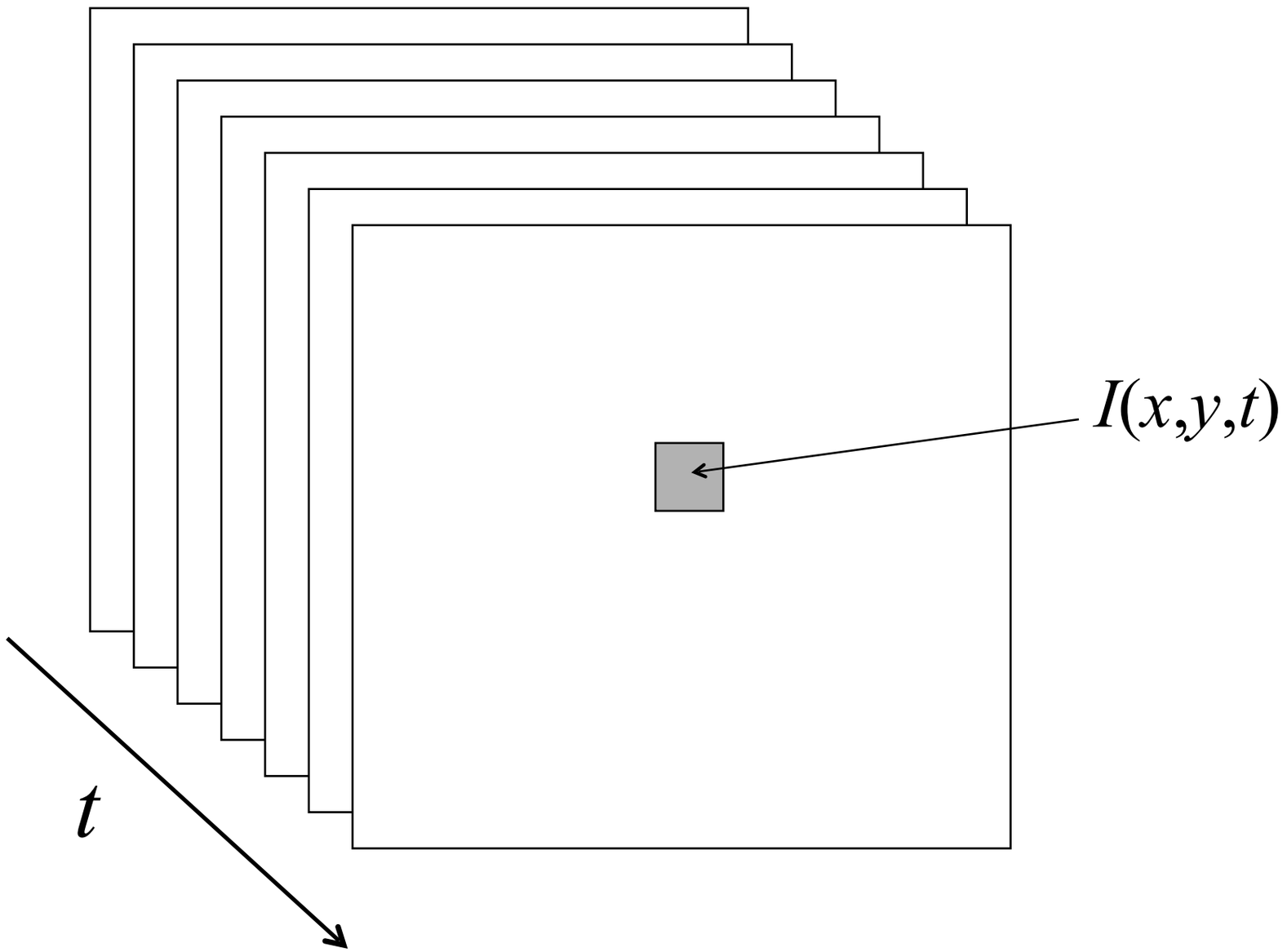
(a)

No motion field but shading changes

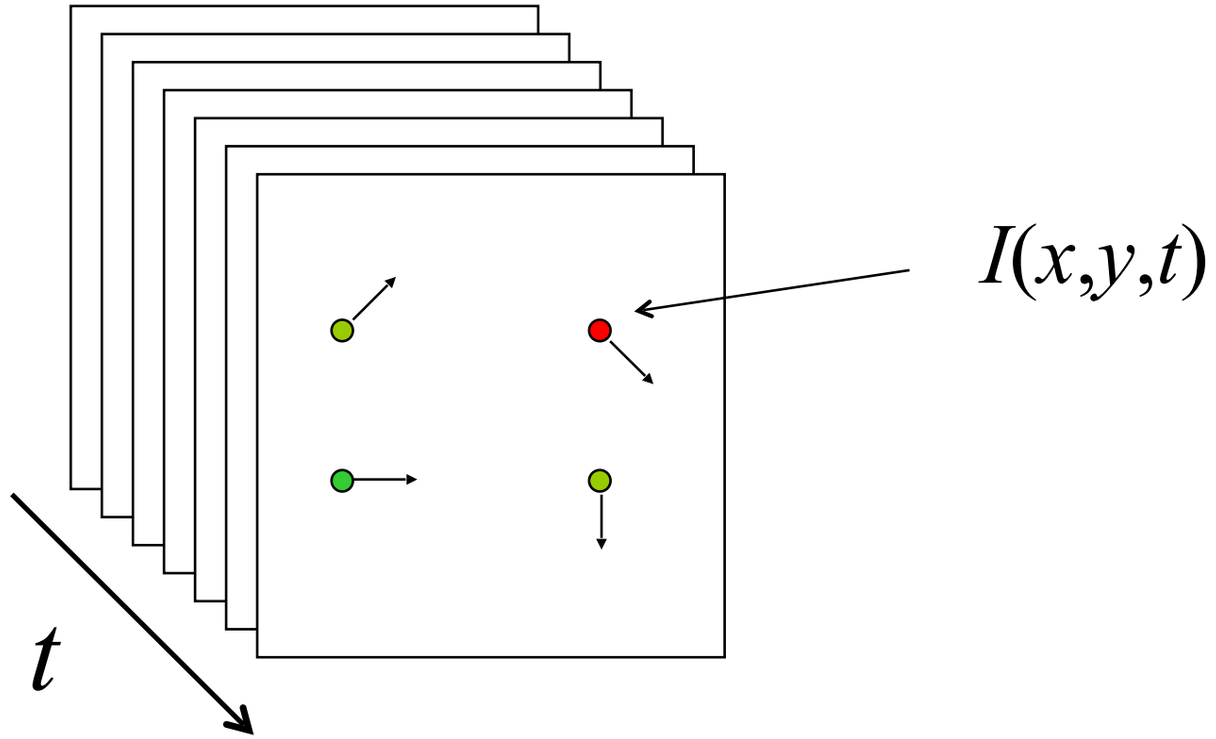
(b)

Approach: embrace videos as spacetime cubes

Under what conditions can we measure the motion of a pixel?

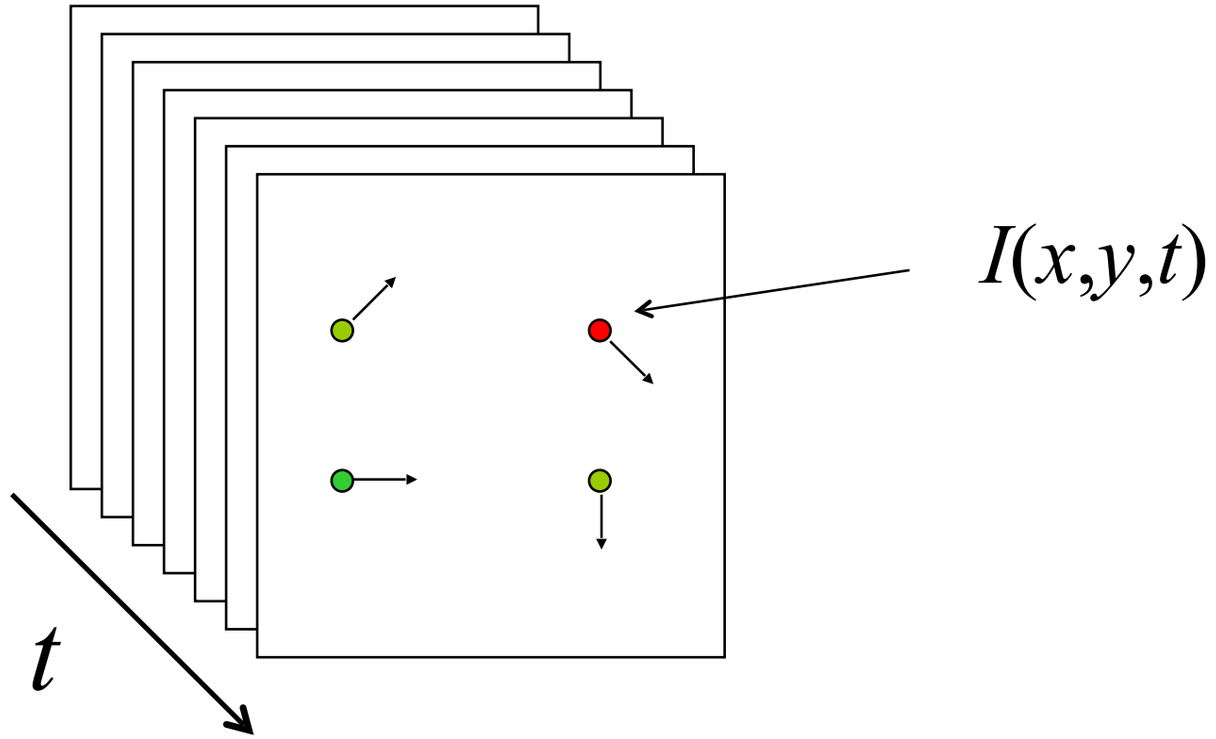


Problem Definition: Optical Flow



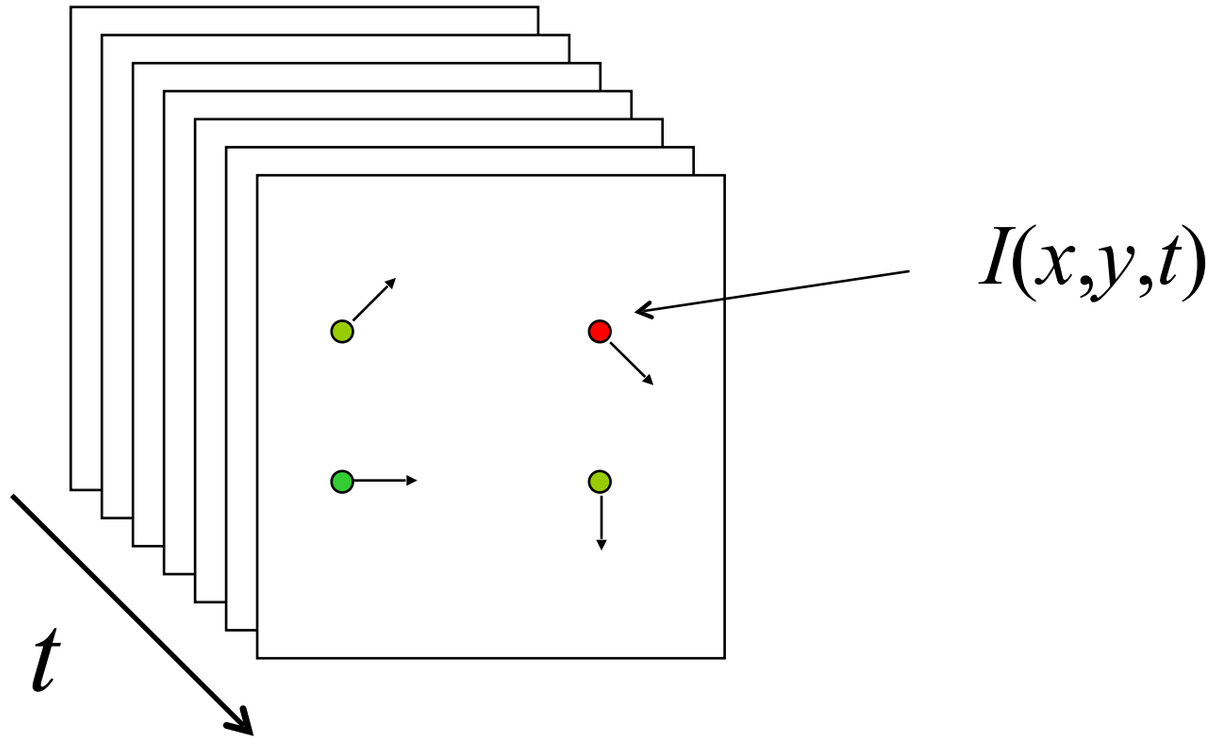
- How to estimate pixel motion from I(x,y,t) to I(x,y,t+1)?
  - Find pixel correspondences
    - Given a pixel in frame t, look for nearby pixels of the same color in frame t+1

Problem Definition: Optical Flow



- How to estimate pixel motion from I(x,y,t) to I(x,y,t+1)?
  - Find pixel correspondences
    - Given a pixel in frame t, look for nearby pixels of the same color in frame t+1

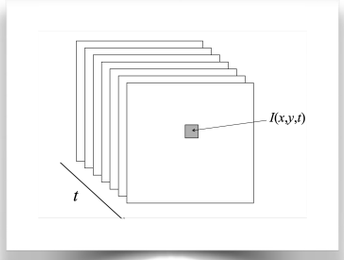
Problem Definition: Optical Flow



- How to estimate pixel motion from I(x,y,t) to I(x,y,t+1)?
  - Find pixel correspondences
    - Given a pixel in frame t, look for nearby pixels of the same color in frame t+1

- Key assumption
  - **color constancy**: a point in frame t looks "the same" in frame t+1
    - For grayscale images, this is **brightness constancy**

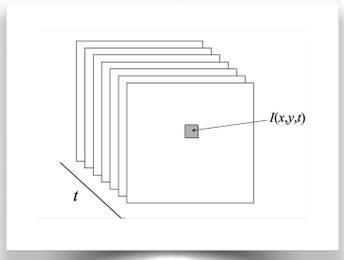
Brightness constancy



Embrace continuous view of I(x,y,t)

$$I(x + \Delta x, y + \Delta y, t + \Delta t) - I(x, y, t) = 0$$

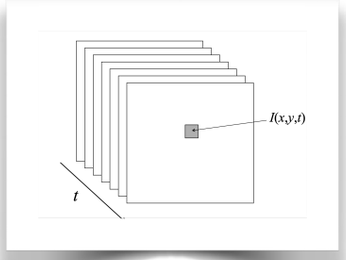
Brightness constancy



Embrace continuous view of I(x,y,t)

$$I(x + \Delta x, y + \Delta y, t + \Delta t) - I(x, y, t) = 0$$

Brightness constancy

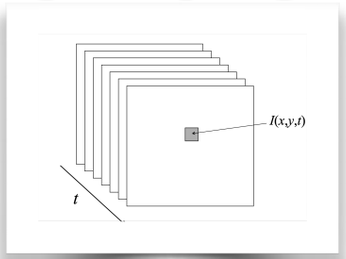


Embrace continuous view of I(x,y,t)

$$I(x + \Delta x, y + \Delta y, t + \Delta t) - I(x, y, t) = 0$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \approx 0$$

Brightness constancy



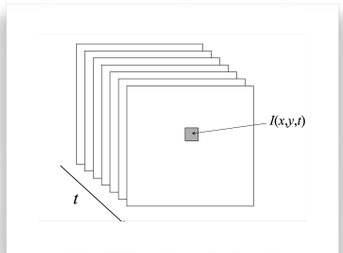
Embrace continuous view of I(x,y,t)

$$I(x+\Delta x, y+\Delta y, t+\Delta t) - I(x,y,t) = 0$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \approx 0$$

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \quad \text{where} \quad u = \frac{\Delta x}{\Delta t}, v = \frac{\Delta y}{\Delta t}$$

Brightness constancy



Embrace continuous view of I(x,y,t)

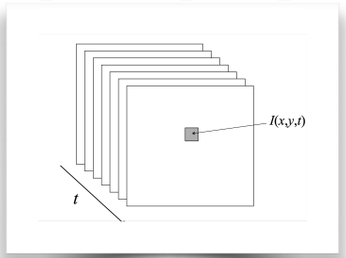
$$I(x+\Delta x, y+\Delta y, t+\Delta t) - I(x,y,t) = 0$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \approx 0$$

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \quad \text{where} \quad u = \frac{\Delta x}{\Delta t}, v = \frac{\Delta y}{\Delta t}$$

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$$

Brightness constancy



Embrace continuous view of I(x,y,t)

$$I(x+\Delta x, y+\Delta y, t+\Delta t) - I(x,y,t) = 0$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \approx 0$$

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0 \quad \text{where} \quad u = \frac{\Delta x}{\Delta t}, v = \frac{\Delta y}{\Delta t}$$

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$$

Brightness constancy equation gives us:

a linear constraint on flow vector (u,v)

Question: video 2 frames of video, we can easily compute image gradients and temporal derivatives. When will these values uniquely reveal the flow?

Outline

- Lucas Kanade
- Egomotion
    - Motivation
    - Time-to-Contact, Parallax, Focus-of-Expansion
- Optical Flow
    - Logistics
    - **Motivation, aperture problem**
    - Sparse (KLT) vs Dense (variational)
    - Optimization tools: robust statistics, coarse-to-fine, markov-random feilds
    - Segmentatiion (dominant motion estimation, background subtraction, layered models)

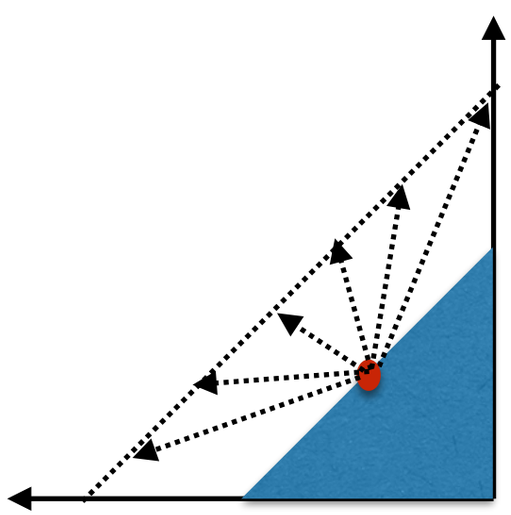
Aperature problem $\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$

We can only determine (u,v) flow in direction parallel to gradient

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_\perp \\ v_\perp \end{bmatrix} + \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} =$$

$$=$$



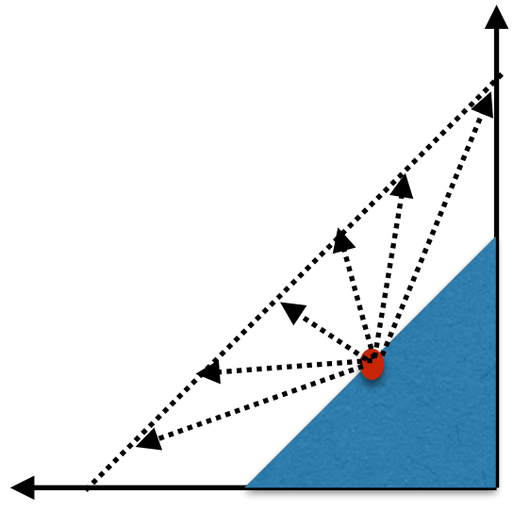
Aperature problem $\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$

We can only determine (u,v) flow in direction parallel to gradient

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_\perp \\ v_\perp \end{bmatrix} + \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} = \nabla I \cdot \begin{bmatrix} u_\perp \\ v_\perp \end{bmatrix} + \nabla I \cdot \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$

$$= \nabla I \cdot \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$



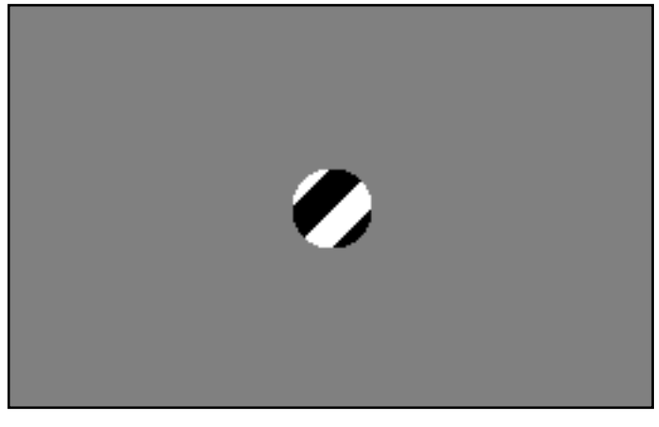
Aperature problem $\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$

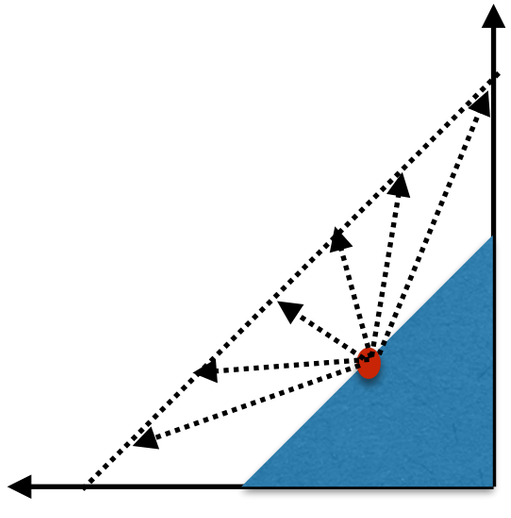
We can only determine (u,v) flow in direction parallel to gradient

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_\perp \\ v_\perp \end{bmatrix} + \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} = \nabla I \cdot \begin{bmatrix} u_\perp \\ v_\perp \end{bmatrix} + \nabla I \cdot \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$

$$= \nabla I \cdot \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$





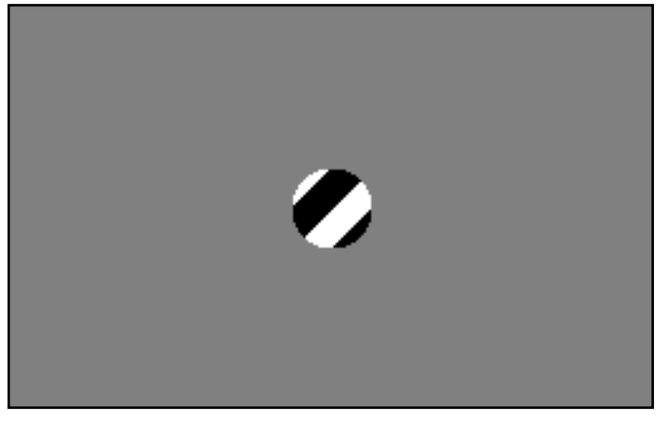
Aperature problem $\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$

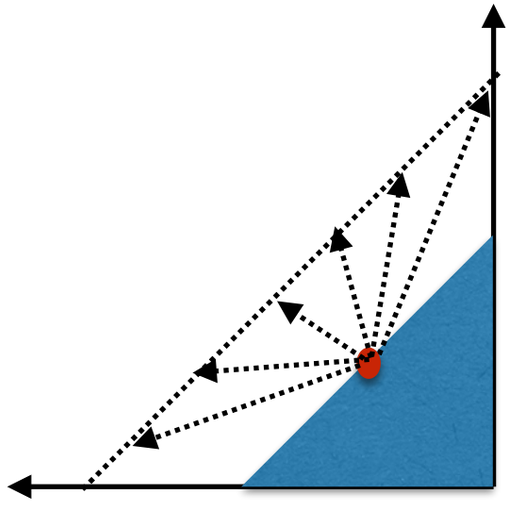
We can only determine (u,v) flow in direction parallel to gradient

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u_\perp \\ v_\perp \end{bmatrix} + \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} = \nabla I \cdot \begin{bmatrix} u_\perp \\ v_\perp \end{bmatrix} + \nabla I \cdot \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$

$$= \nabla I \cdot \begin{bmatrix} u_\parallel \\ v_\parallel \end{bmatrix}$$



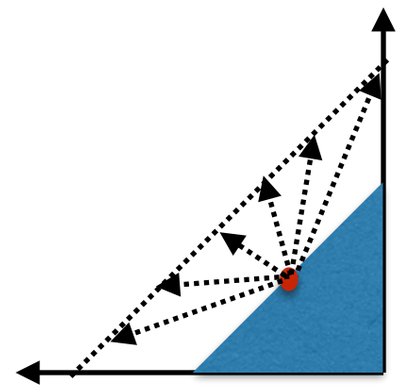


Challenges

- Aperture problem

Soln to brightness constancy equation may not be unique

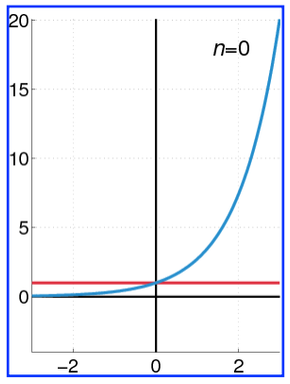
$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$



- Small motion assumption

First-order taylor approximation does not hold for large motions
(or slow framerates)

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$$

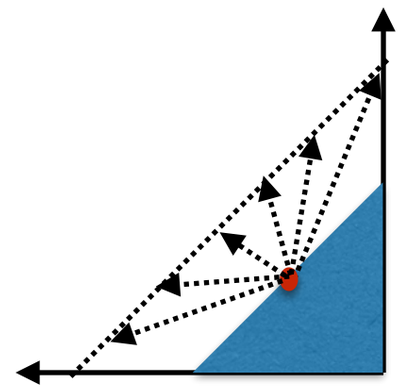


Challenges

- Aperture problem

Soln to brightness constancy equation may not be unique

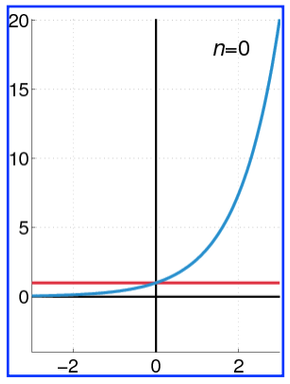
$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$



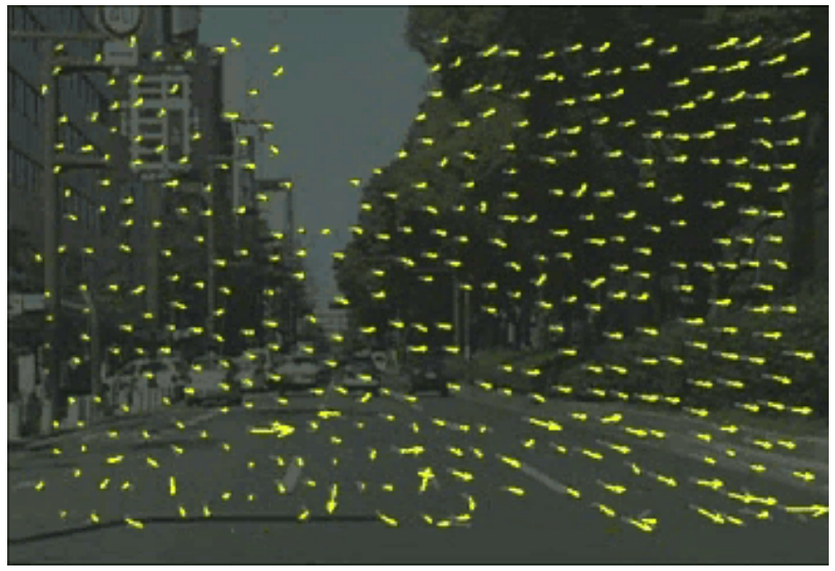
- Small motion assumption

First-order taylor approximation does not hold for large motions
(or slow framerates)

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$$



Soln for aperture problem



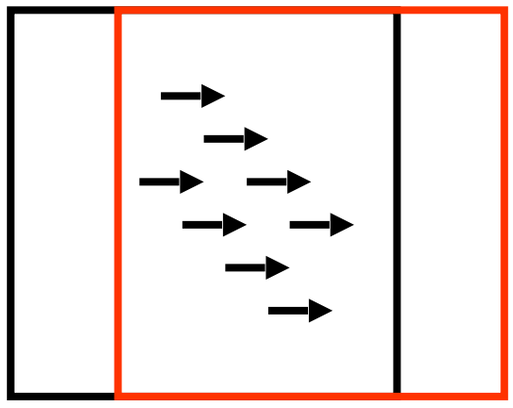
1. Don’t try to estimate flow at unreliable points (sparse flow; similar to feature point alignment of corners!)

2. Assume neighboring flow vectors are similar (enforce *spatial smoothness* in dense flow feild)

Simple approach:
assume flow is constant over local patch

KLT (Kanade Lucas Tomasi) tracker: run Lucas Kanade tracker (HW2) on interest points (HW3) from frame 1

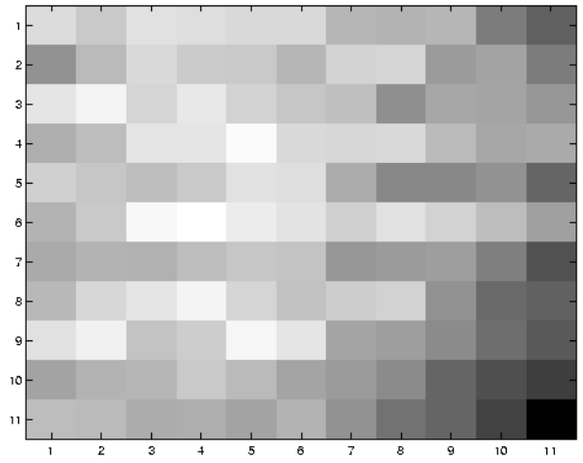
$$\min_{u,v} \sum_{x,y \in W} \Big(I_2(x+u, y+v) - I_1(x,y) \Big)^2$$

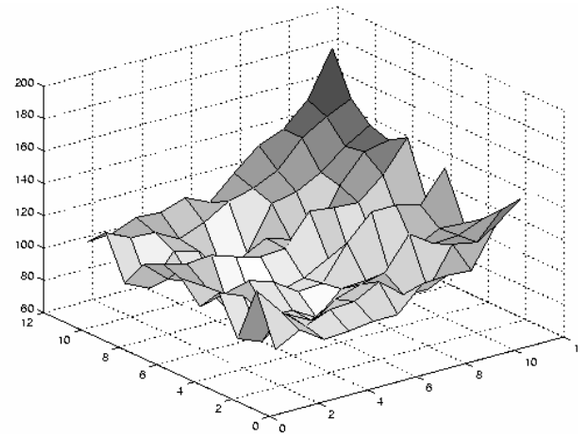


$u(x,y) = u$

$v(x,y) = v$

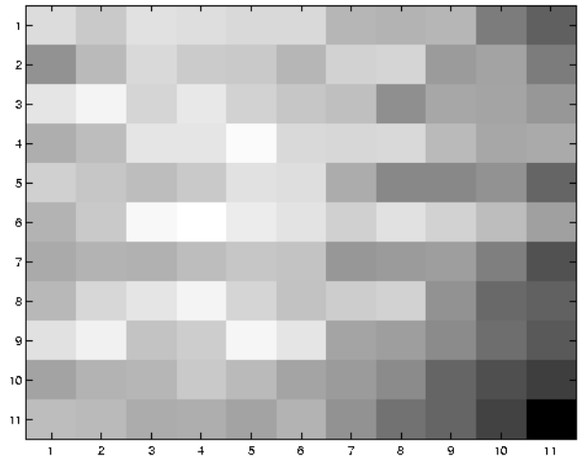
Low Texture Region - Bad

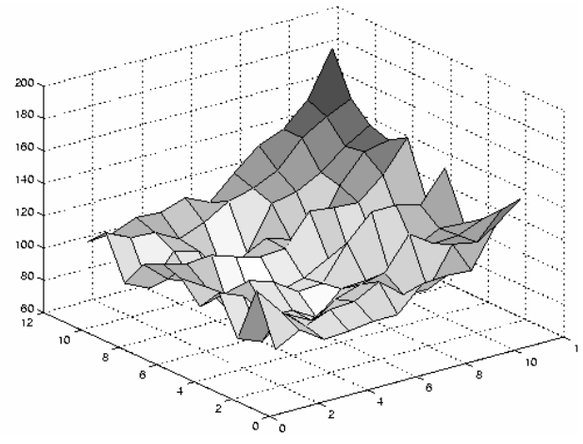




SSD surface

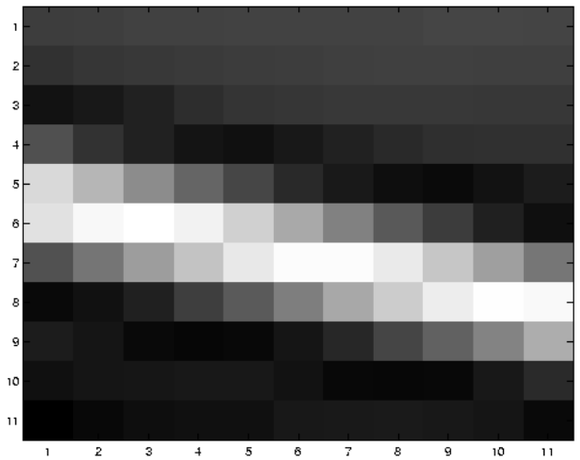
Low Texture Region - Bad

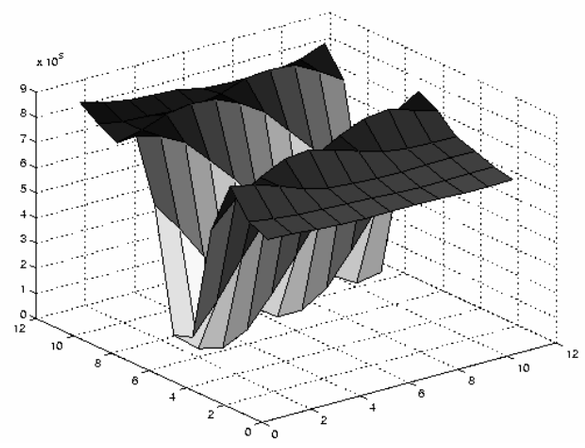




SSD surface

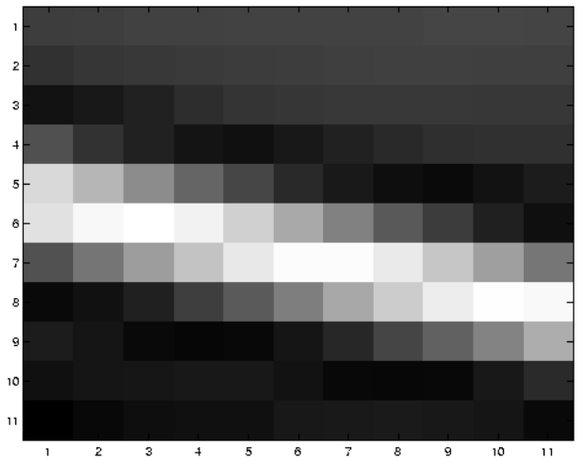
Edges – so,so (aperture problem)

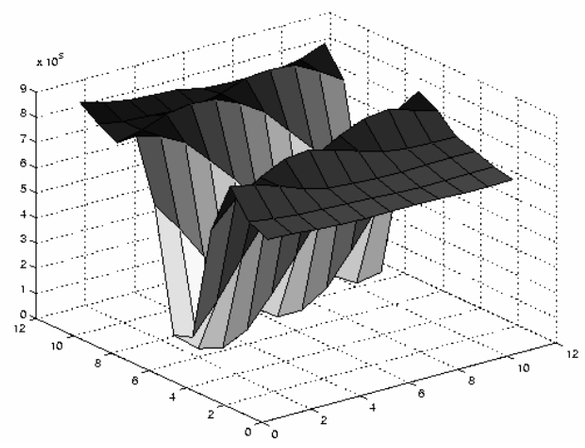




SSD surface

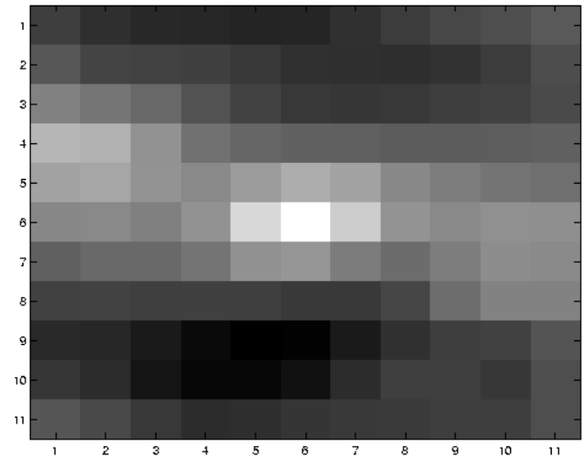
Edges – so,so (aperture problem)

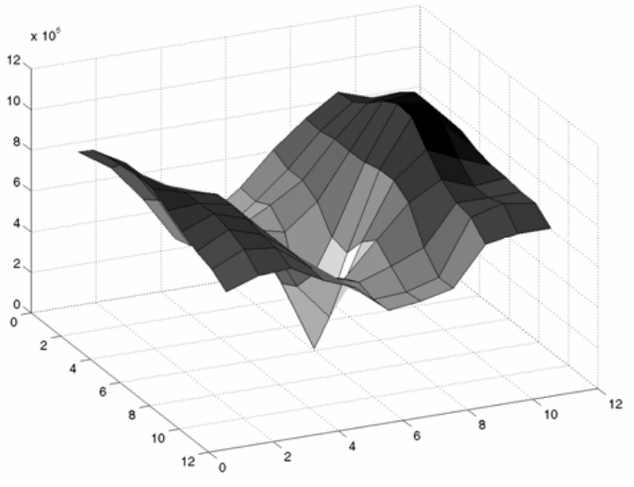




SSD surface

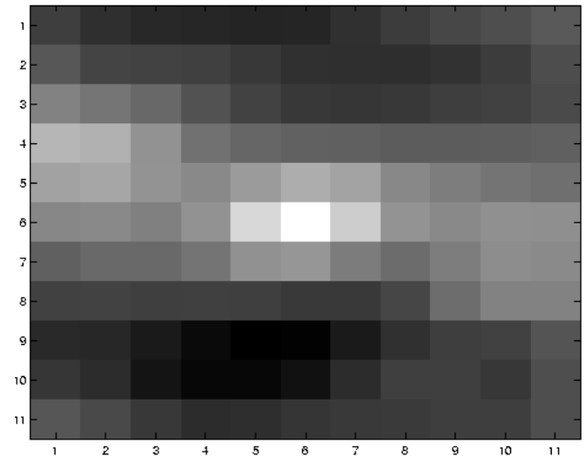
High Textured Region - Good

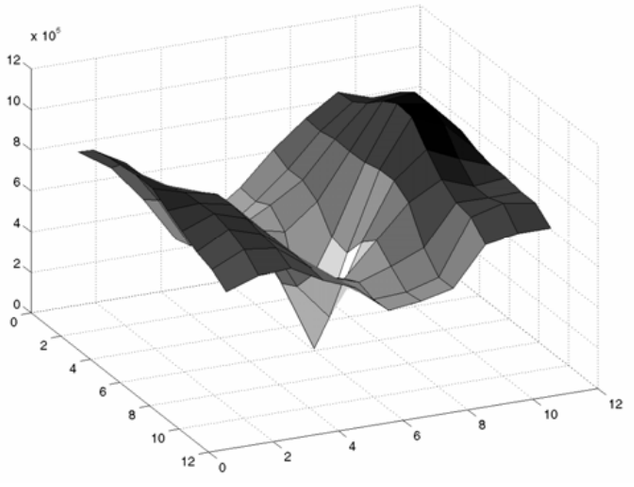




SSD surface

High Textured Region - Good





SSD surface

Sparse flow estimation (feature tracking)

1. User Harris corner score to find trackable patches

$$I_2(x+u, y+v) - I_1(x,y) \approx \nabla I(x,y) \begin{bmatrix} u \\ v \end{bmatrix} + I_t(x,y)$$

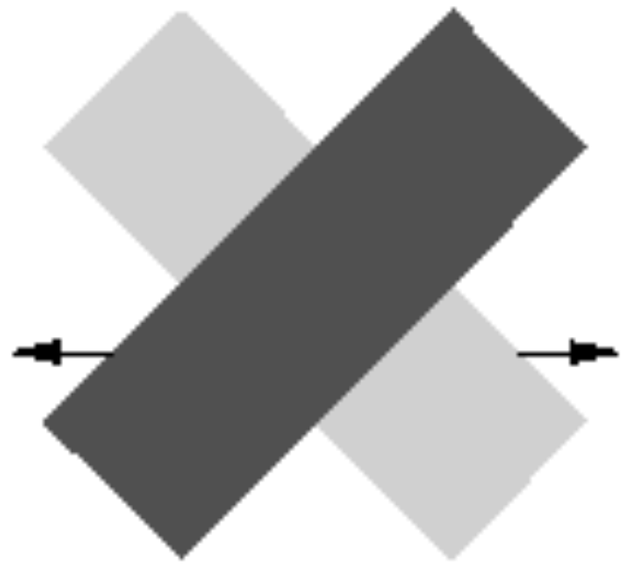
2. Appy Lucas Kanade on those patches

Good Features to Track

Jianbo Shi
Computer Science Department
Cornell University
Ithaca, NY 14853

Carlo Tomasi
Computer Science Department
Stanford University
Stanford, CA 94305

Problem: interest points can *still* have ambiguious motion



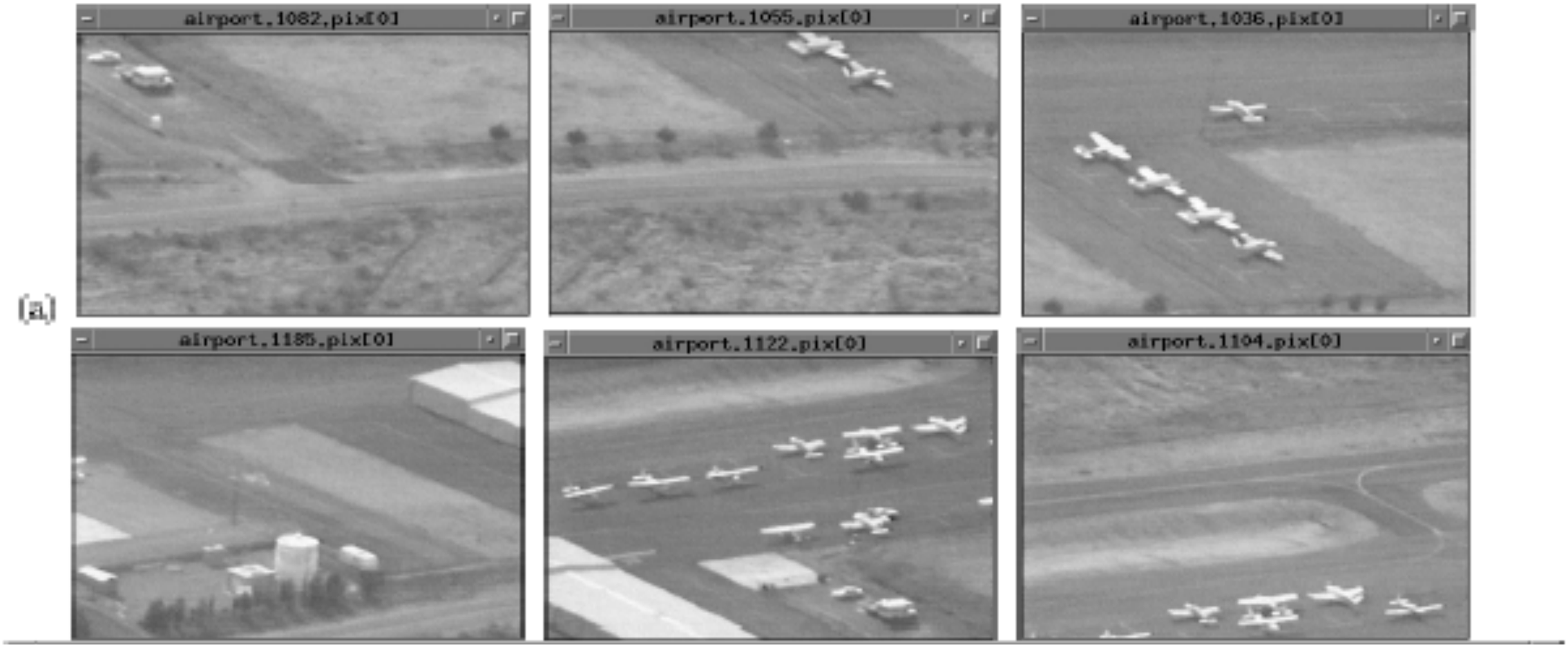
Where does false “t-junctions” appear to move?

We’d like to integrate local signals *globally* instead of locally

Outline

- Lucas Kanade
- Egomotion
    - Motivation
    - Time-to-Contact, Parallax, Focus-of-Expansion
- **Optical Flow**
    - Motivation, aperture problem
    - Sparse (KLT) vs **Dense** (variational)
    - Optimization tools: robust statiistics, coarse-to-fine, markov-random feilds
    - Segmentatiion (dominant motion estimation, background subtraction, layered models)

Dense flow (I)

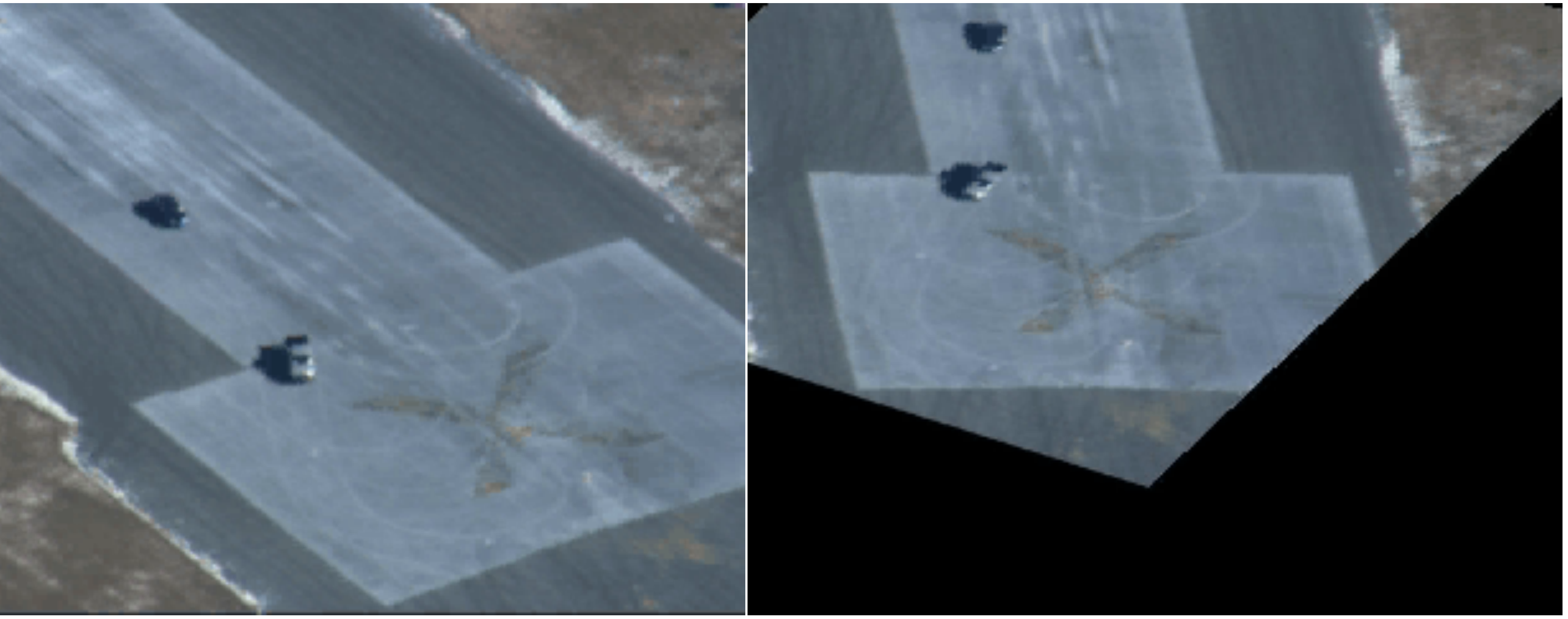


$$E(\mathbf{p}) = \sum_{\mathbf{x}} [I(\mathbf{W}(\mathbf{x};\mathbf{p})) - T(\mathbf{x})]^2$$

Estimate a global warp over successive frames of a video sequence (with Lucas Kanade optimization)

Generalize translation to other 2D warps (affine, homographies,…)

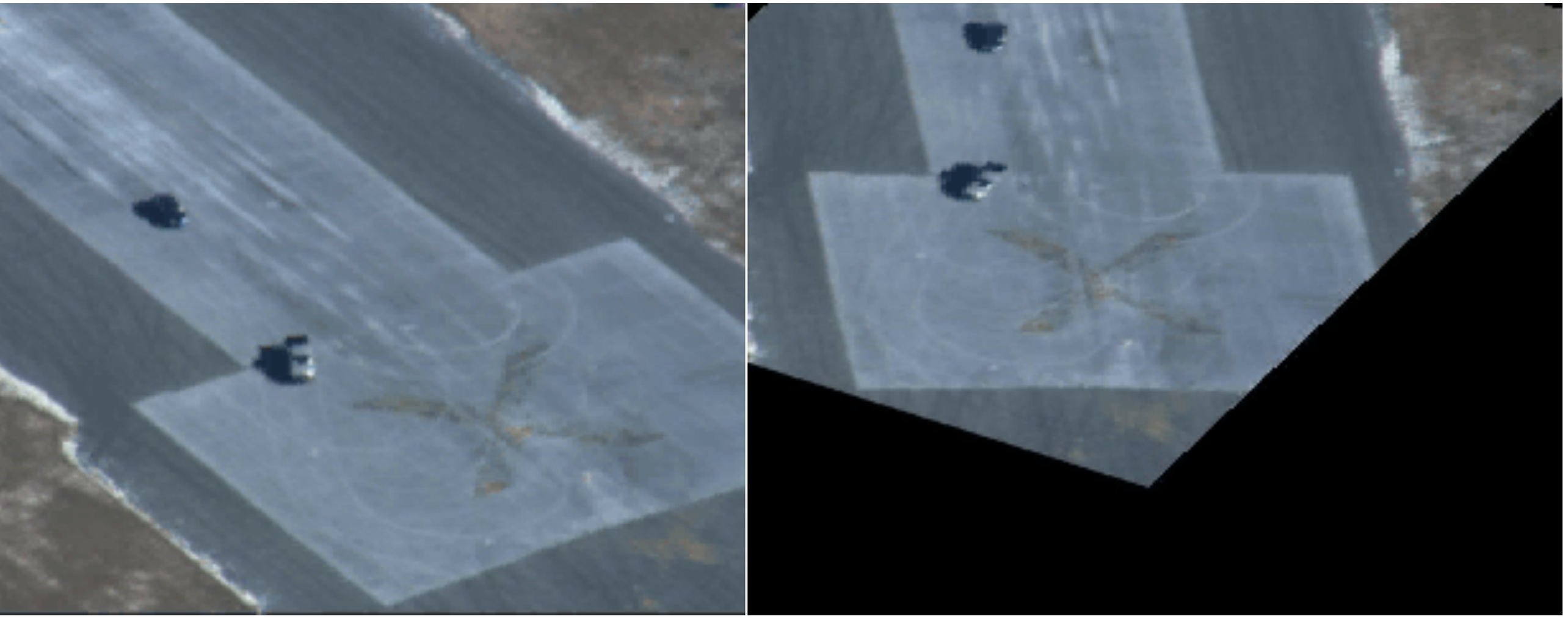
Applications: mosaicing (HW3)



Global homography warp works for some cases (rotations, planar scenes)

… but won't capture moving objects.
Can we come up with an "elegant" and general mathematical formulation of dense flow?

Applications: mosaicing (HW3)



Global homography warp works for some cases (rotations, planar scenes)

… but won't capture moving objects.
Can we come up with an "elegant" and general mathematical formulation of dense flow?

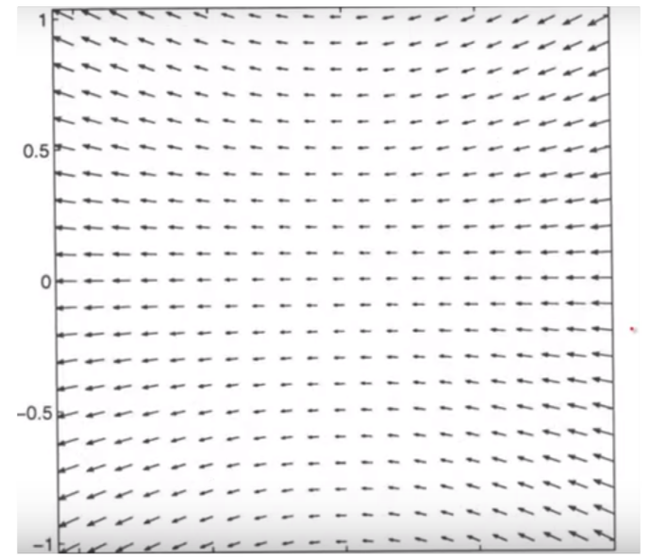
Dense flow (II)

Solve for global flow *feild*

$I_1(x,y)$, $I_2(x,y)$

$$\min_{\substack{u(x,y)\\v(x,y)}} \sum_{x,y} [I_2(x+u(x,y), y+v(x,y)) - I_1(x,y)]^2$$

$u(x,y)$, $v(x,y)$



Where we are headed: we'll add a regularization term that favors *smooth* fields

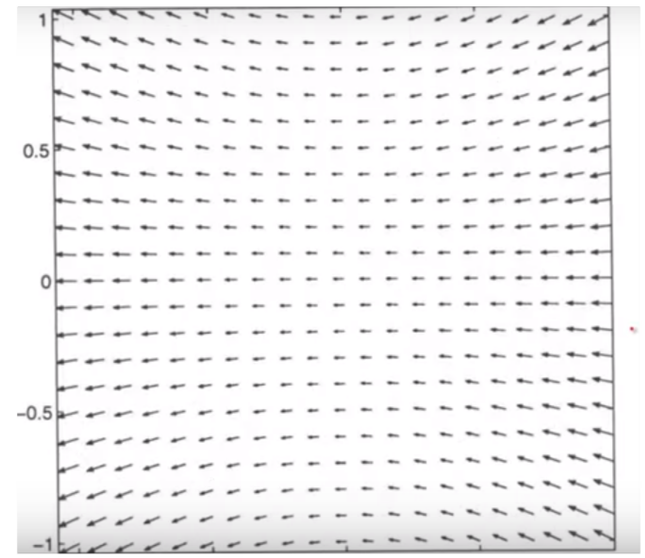
Dense flow (II)

Solve for global flow *feild*

$I_1(x,y)$, $I_2(x,y)$

$$\min_{\substack{u(x,y)\\v(x,y)}} \sum_{x,y} [I_2(x+u(x,y), y+v(x,y)) - I_1(x,y)]^2$$

$u(x,y)$, $v(x,y)$



Where we are headed: we'll add a regularization term that favors *smooth* fields

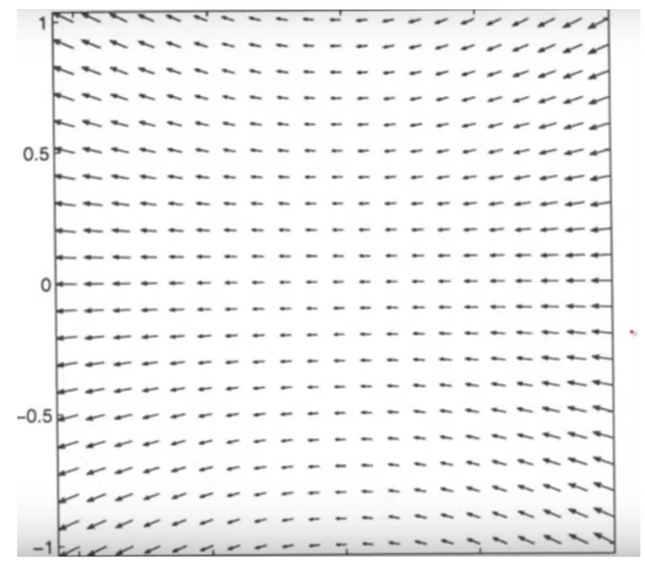
Dense flow (II)

Solve for global flow *feild*

$I_1(x,y)$, $I_2(x,y)$

$$\min_{\substack{u(x,y)\\v(x,y)}} \sum_{x,y} [I_2(x+u(x,y), y+v(x,y)) - I_1(x,y)]^2$$

u(x,y), v(x,y)



Where we are headed: we'll add a regularization term that favors *smooth* fields

Aside: continuous case (historically more common in flow literature)

$$\min_{u,v} \int\int \big(I_2(x+u, y+v) - I_1(x,y)\big)^2 dxdy$$

Formal math is known as *calculus of variations* (we're minimizing over the *space of functions*)

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

Dense *variational* flow

If we assume small motions….

$$I_2(x+u, y+v) - I_1(x,y) \approx \nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t$$

$$\min_{u,v} \int\int (\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t)^2 dx dy$$

above is “shorthand” for…

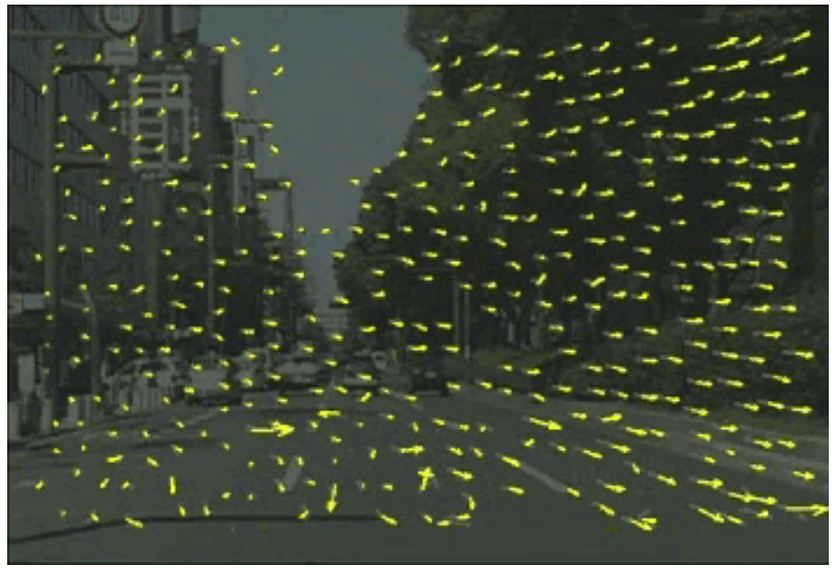
$$\min_{\substack{u(x,y) \\ v(x,y)}} \sum_{x,y} \left[\nabla I(x,y) \cdot \begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} + I_t(x,y) \right]^2$$

Punchline: spatial regularization

Penalize differences in nearby flow vectors: Horn-Schunk Optical Flow

$$\min_{u,v} E_{intensity} + E_{smooth}$$

$$E_{intensity}(u,v) = \int\int (\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t)^2 dxdy \qquad E_{smooth}(u,v) = \int\int ||\nabla u||^2 + ||\nabla v||^2 dxdy$$



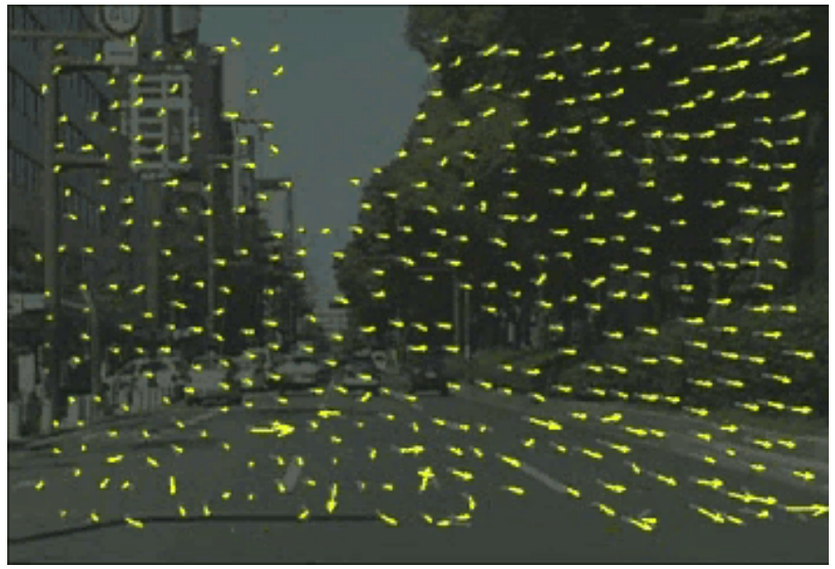
$$\min_{\substack{u(x,y) \\ v(x,y)}} \sum_{x,y} \Big(I_x(x,y)u(x,y) + I_y(x,y)v(x,y) + I_t(x,y) \Big)^2 + (u(x+1,y) - u(x,y))^2 + \dots$$

Punchline: spatial regularization

Penalize differences in nearby flow vectors: Horn-Schunk Optical Flow

$$\min_{u,v} E_{intensity} + E_{smooth}$$

$$E_{intensity}(u,v) = \int\int (\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t)^2 dxdy \qquad E_{smooth}(u,v) = \int\int ||\nabla u||^2 + ||\nabla v||^2 dxdy$$



1. Unknowns (u,v) appear quadratically in above expression => discretize above and solve for them with a giant linear system

$$\min_{\substack{u(x,y) \\ v(x,y)}} \sum_{x,y} \Big(I_x(x,y)u(x,y) + I_y(x,y)v(x,y) + I_t(x,y) \Big)^2 + (u(x+1,y) - u(x,y))^2 + \dots$$

2. Challenge: outliers will overwhelm squared error term

Challenges with regularization

https://en.wikipedia.org/wiki/Horn-Schunck_method

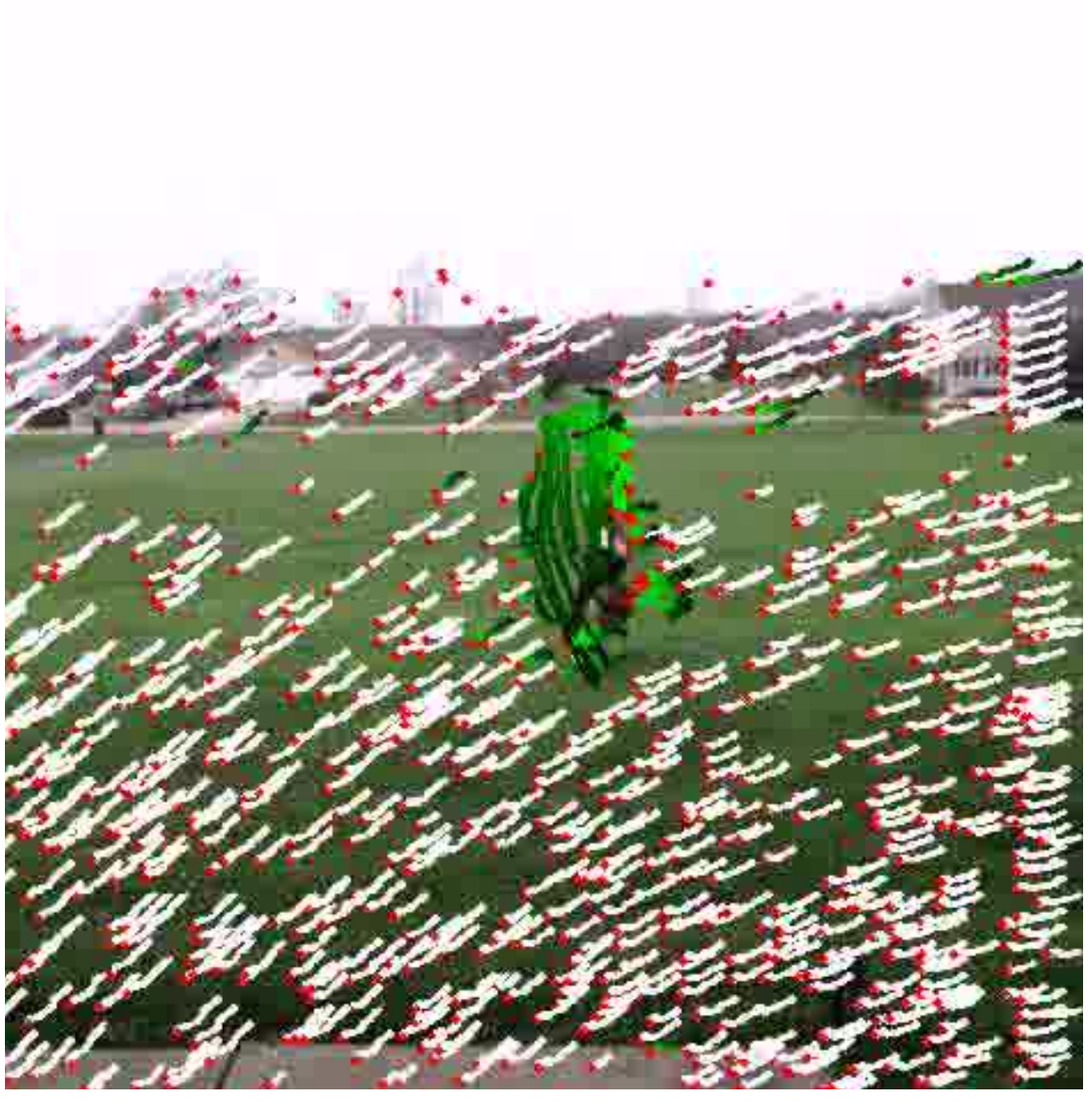
$$\min_{u,v} E_{\text{intensity}} + E_{\text{smooth}}$$

estimated flow ground-truth flow

$$E_{intensity}(u,v) = \int\int (\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t)^2 dxdy$$

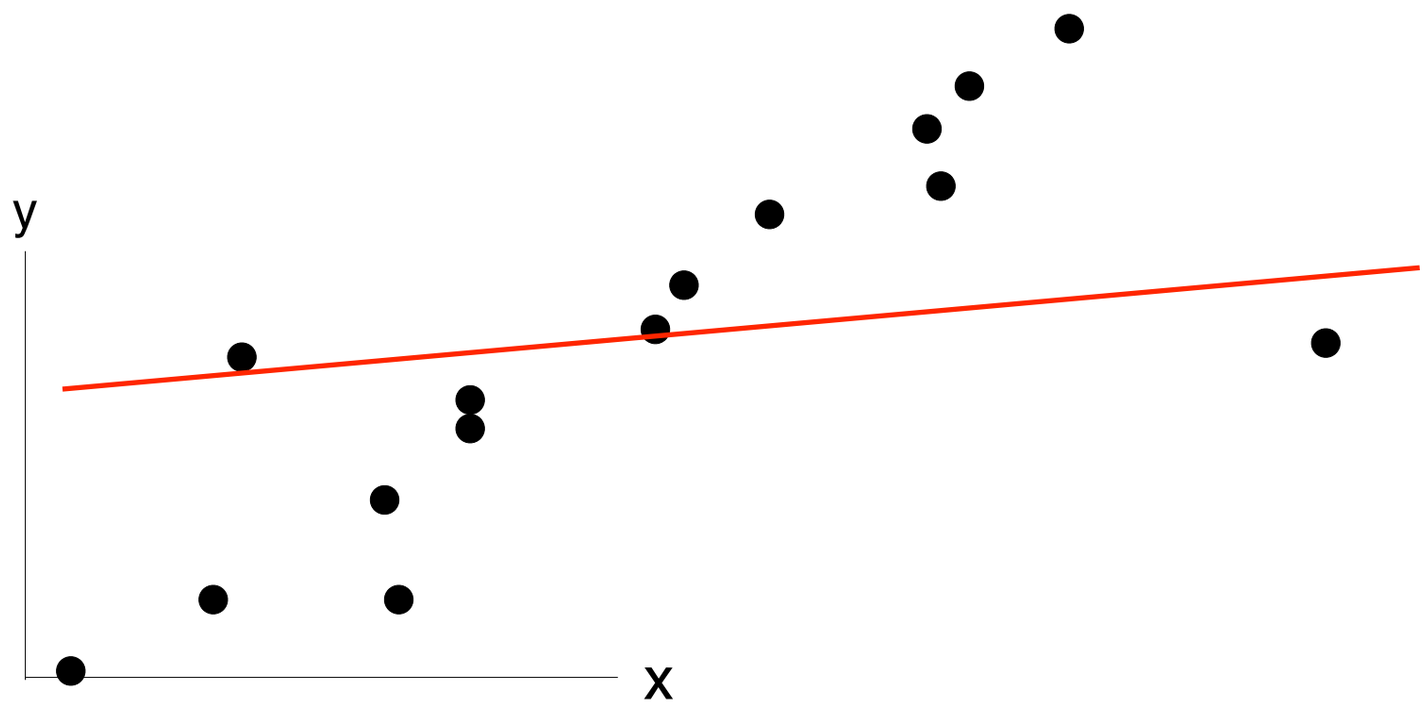
$$E_{smooth}(u,v) = \int\int ||\nabla u||^2 + ||\nabla v||^2 dxdy$$

Can we use ransac to fit flow?

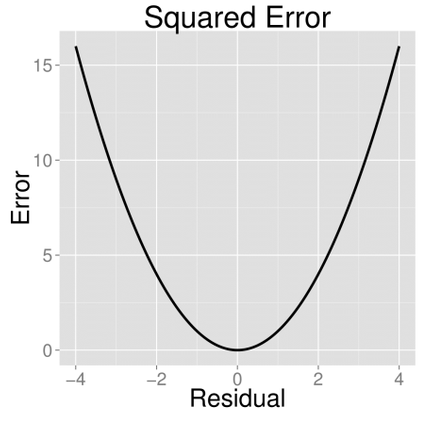


Yes, but historically optical flow methods take a more continuous optimization (variational) perspective

Recall RANSAC



Underlying problem: squared error penalizes outliers too much

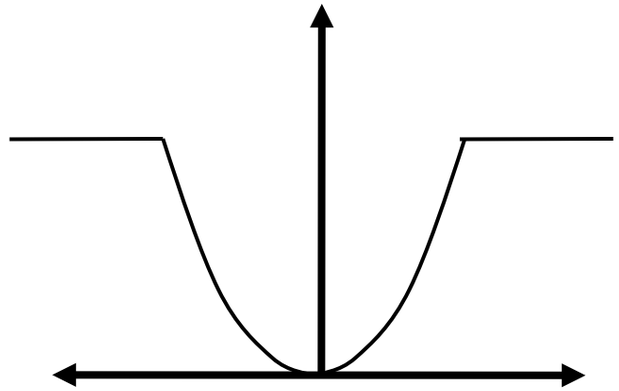


Deeper reason: Gaussian statistics ($e^{-\|x-u\|^2}$) are too simplistic for real world

RANSAC as robust model-fitting

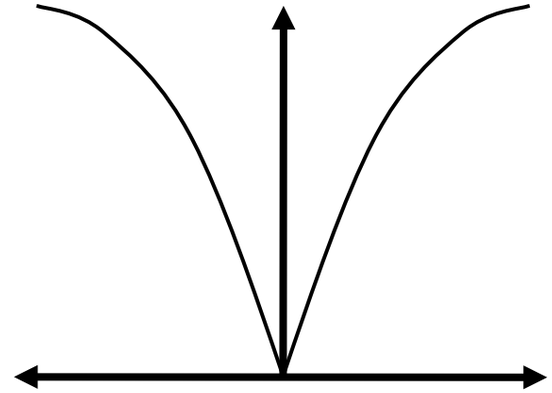
Instead of scoring a candidate model with # of inliers, score it under robust error function

It turns out, we can generalize RANSAC to an algorithm for *maximum-likelihood* model fitting with robust error models (MLESAC, Torr & Zisserman 00)



truncated quadratic

$$\rho_{\alpha,\lambda}(x) = \begin{cases} \lambda x^2 & \text{if } |x| < \frac{\sqrt{\alpha}}{\sqrt{\lambda}} \\ \alpha & \text{otherwise} \end{cases}$$



lorentzian

$$\rho_\sigma(x) = log\left(1 + \frac{1}{2}(\frac{x}{\sigma})^2\right)$$

Where we are headed: we'll insert such robust error terms into our variational optimization

Robust variational optical flow

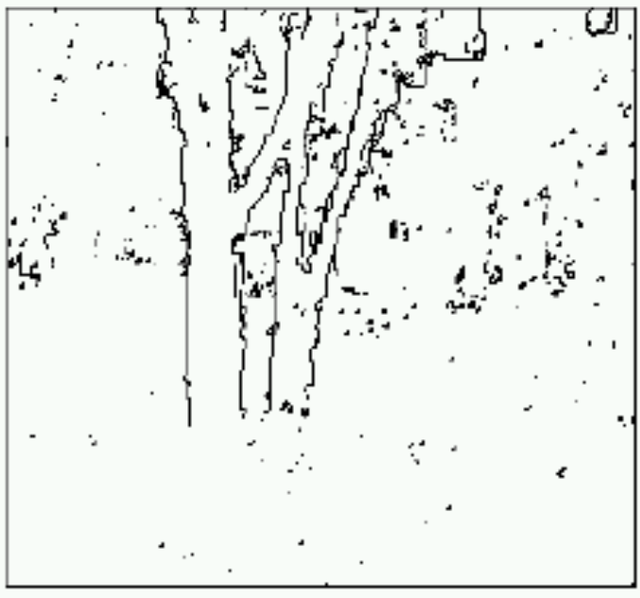
$$\min_{u,v} \int\int \rho(I_2(x+u, y+v) - I_1(x,y)) + \rho(||\nabla u||) + \rho(||\nabla v||) dx dy$$

where ρ = robust error function (instead of quadratic error)

first image

quadratic flow

lorentzian flow



detected outliers

Typical approach: initialize solution with quadratic flow and locally optimize

Reference

- Black, M. J. and Anandan, P., A framework for the robust estimation of optical flow, *Fourth International Conf. on Computer Vision* (ICCV), 1993, pp. 231-236 http://www.cs.washington.edu/education/courses/576/03sp/readings/black93.pdf

Outline

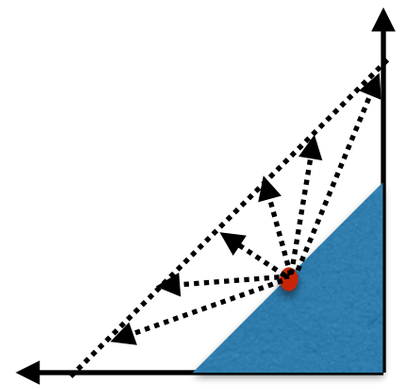
- Lucas Kanade
- Egomotion
    - Motivation
    - Time-to-Contact, Parallax, Focus-of-Expansion
- **Optical Flow**
    - Motivation, aperture problem
    - Sparse (KLT) vs Dense (variational)
    - Optimization tools: robust statistics, **coarse-to-fine**, markov-random feilds
    - Segmentatiion (dominant motion estimation, background subtraction, layered models)

Recall our challenges

- Aperture problem

Soln to brightness constancy equation may not be unique

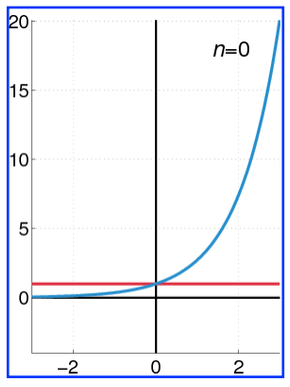
$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$



- Small motion assumption

First-order taylor approximation does not hold for large motions

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$$

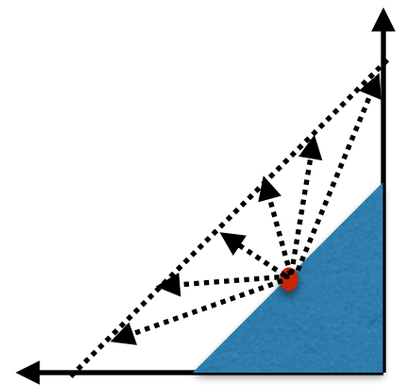


Recall our challenges

- Aperture problem

Soln to brightness constancy equation may not be unique

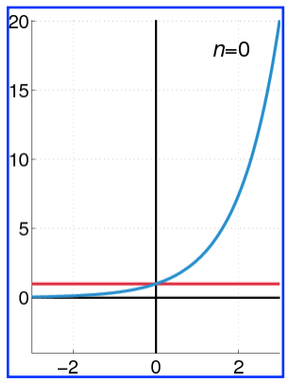
$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$



- Small motion assumption

First-order taylor approximation does not hold for large motions

$$\nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + \frac{\partial I}{\partial t} = 0$$



Revisting the small motion assumption

$$I_2(x+u, y+v) - I_1(x,y) \approx \nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t$$

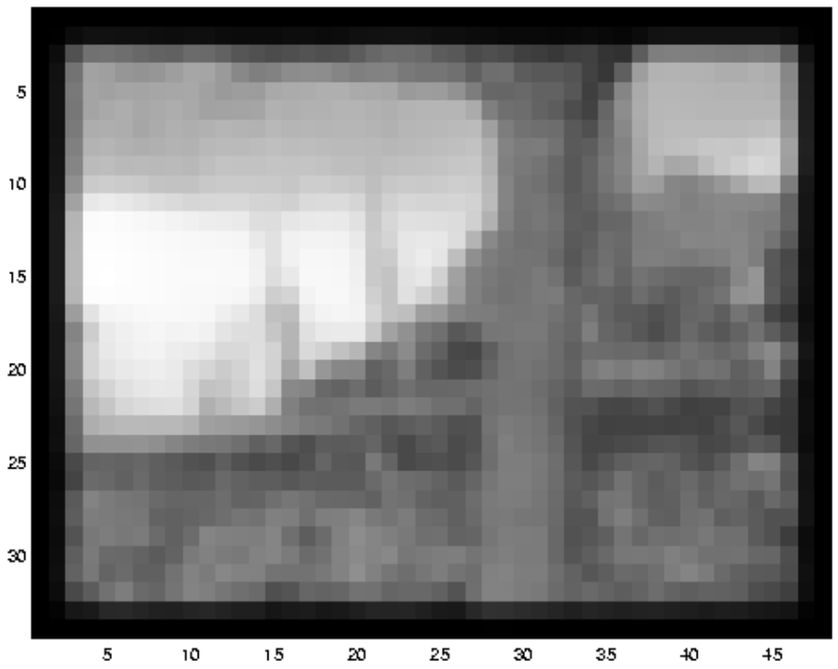
Is the motion small enough to make Taylor-series linearization valid?

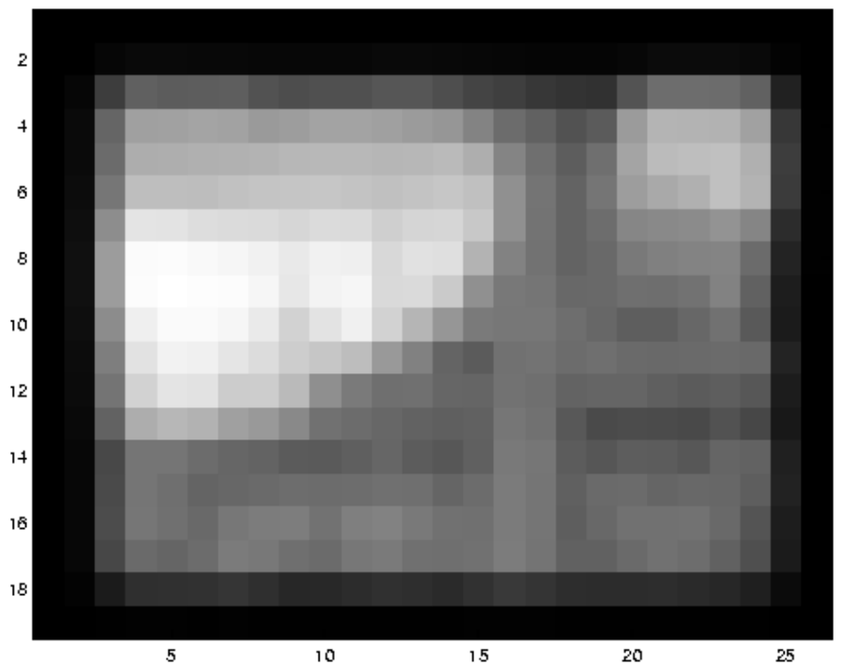
Revisting the small motion assumption

$$I_2(x+u, y+v) - I_1(x,y) \approx \nabla I \cdot \begin{bmatrix} u \\ v \end{bmatrix} + I_t$$

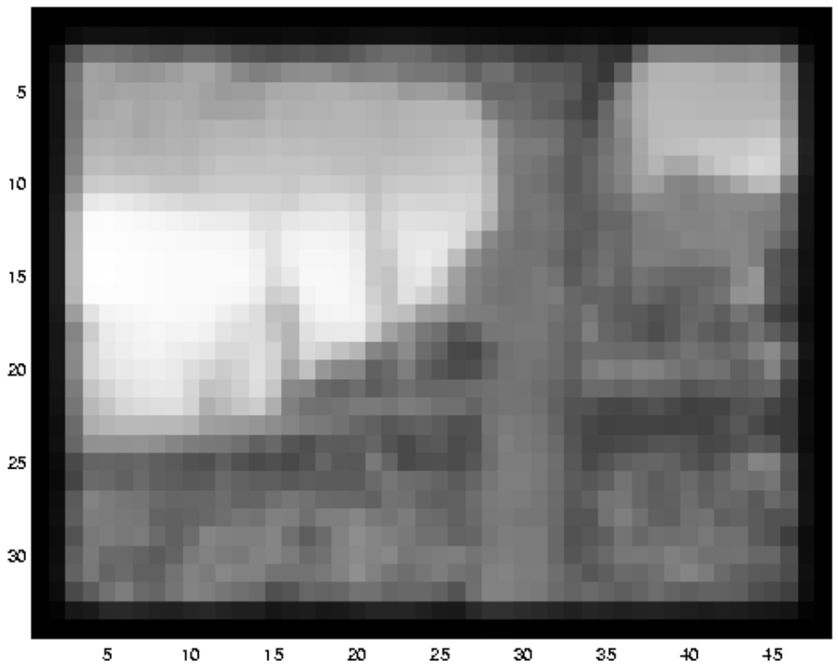
Is the motion small enough to make Taylor-series linearization valid?

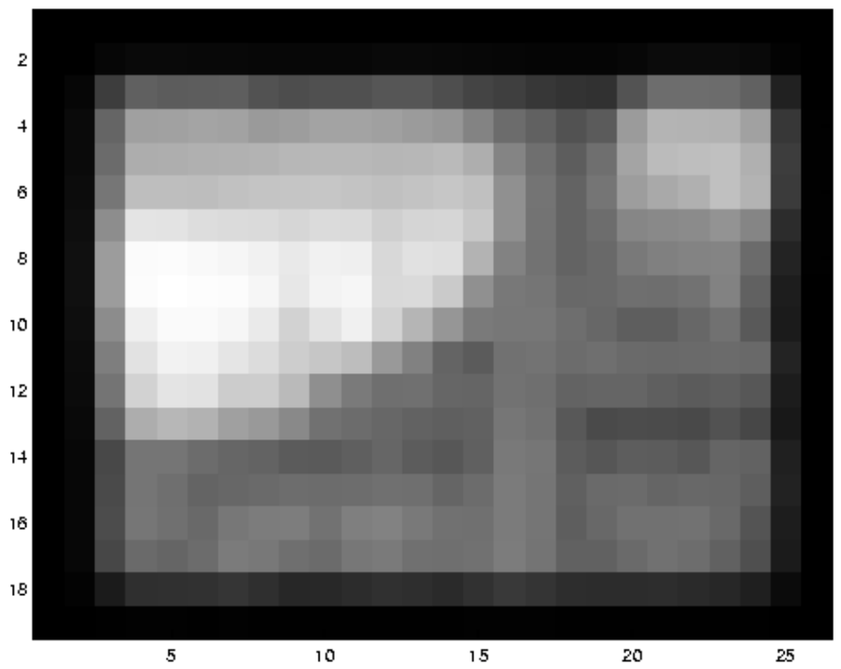
One soln: reduce the resolution!



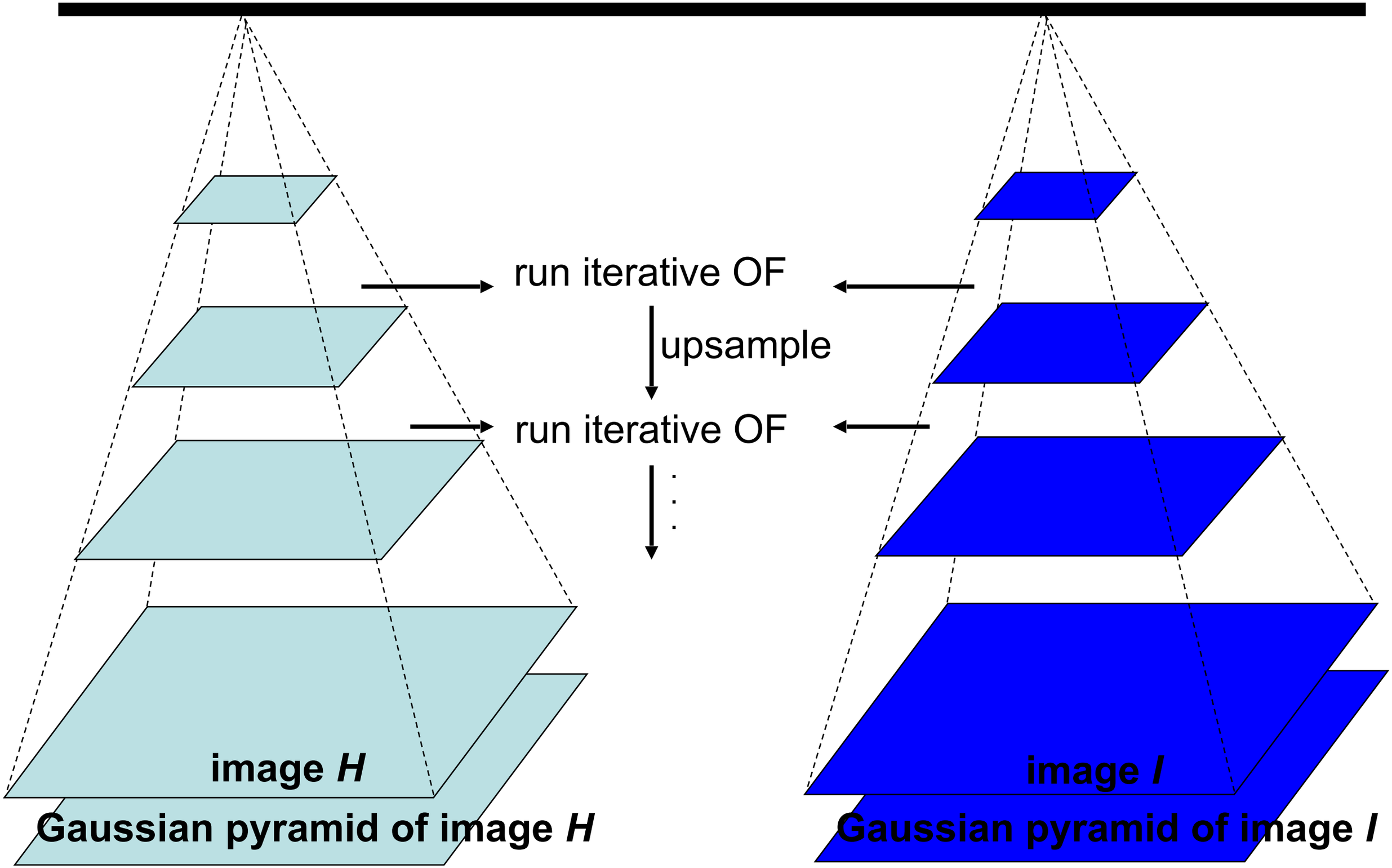


One soln: reduce the resolution!





Soln 1: Coarse-to-fine Optical Flow



Soln 2: discrete optical flow estimation

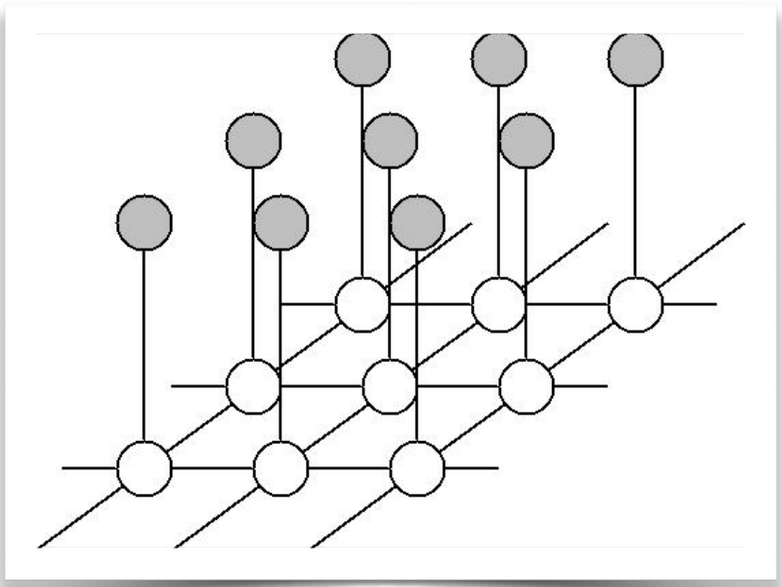
Intuition: turn optical flow estimation into a discrete graph coloring problem

1. Discretize the space of motions into K (e.g., 100) labels or colors.
2. Define cost of a coloring = image reprojection error + # of neighbors with disagreeing color
3. Find minimum cost coloring

$$u_i \in \{-5 \dots 5\}$$
$$v_i \in \{-5 \dots 5\}$$
$$z_i = (u_i, v_i)$$

$$\phi_i(z_i) = \rho(||I_2(x_i + u_i, y_i + v_i) - I(x_i, y_i)||)$$
$$\psi_{ij}(z_i, z_j) = \rho(u_i - u_j, v_i - v_j)$$

$$E(z) = \sum_{i \in V} \phi_i(z_i) + \sum_{ij \in E} \psi_{ij}(z_i, z_j)$$



Discrete Markov Random Feild (MRF) with pixel-grid graph G=(V,E)

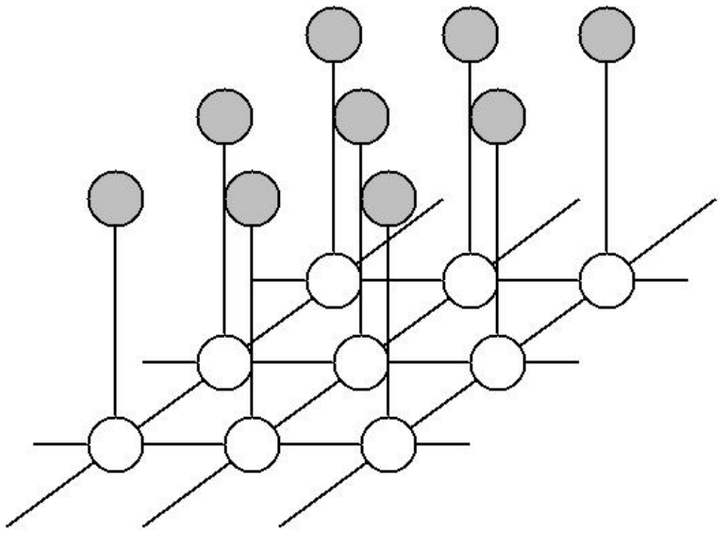
A Database and Evaluation Methodology for Optical Flow

Simon Baker · Daniel Scharstein · J.P. Lewis ·
Stefan Roth · Michael J. Black · Richard Szeliski

Example: SIFTFlow

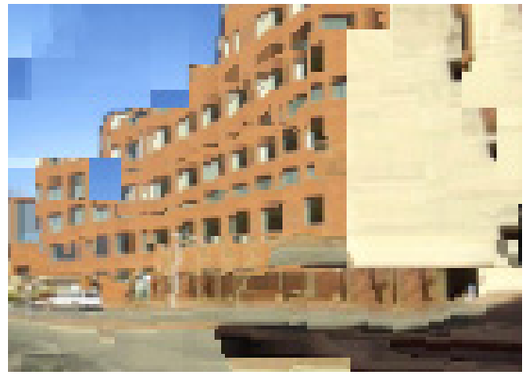
Measure local appearances of patches using SIFT descriptors

Turns out that this can be used to align images of different scenes!



Liu et al, PAMI 2011



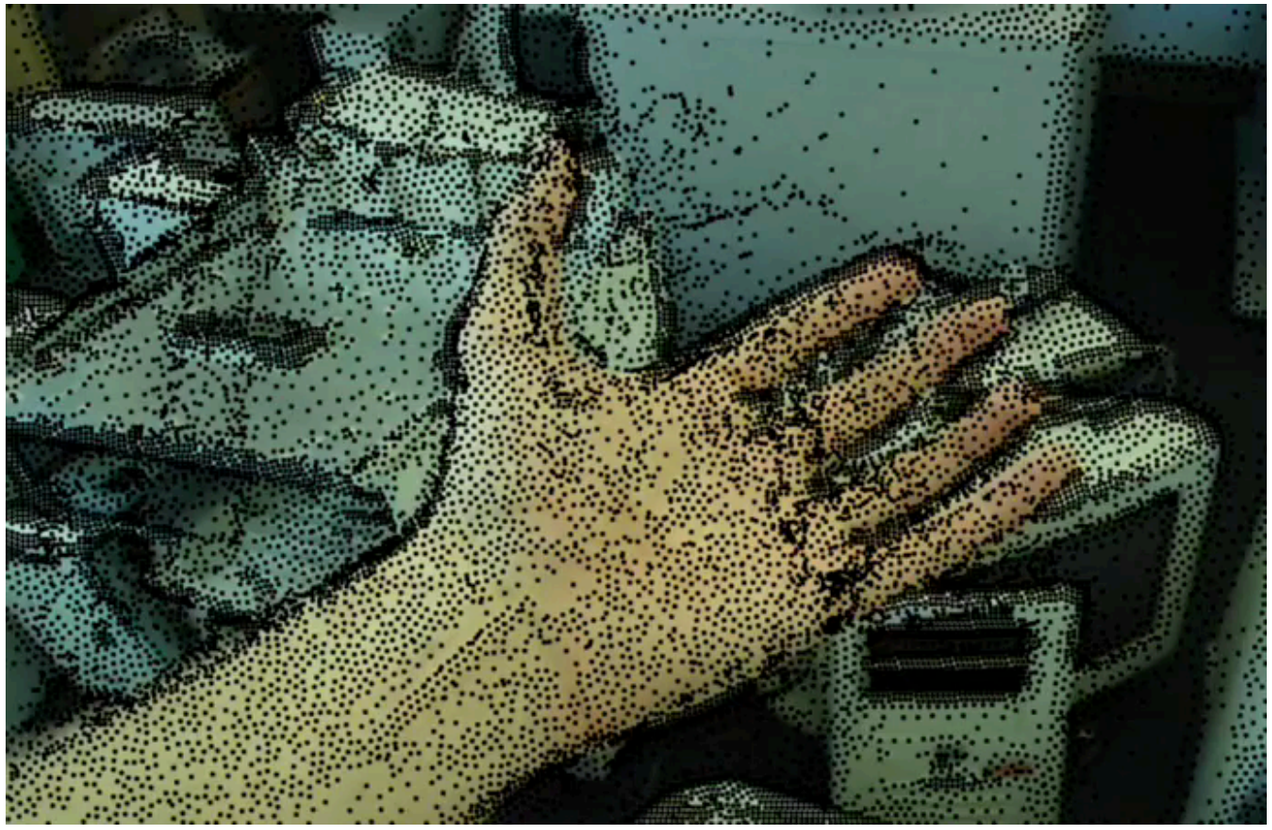



Allows us to do nearest-neighbor label transfer for scene analysis

Remaining challenges: long-term optical flow

Combine long-term sparse feature tracking with variational flow regularization

(http://rvsn.csail.mit.edu/pv/)

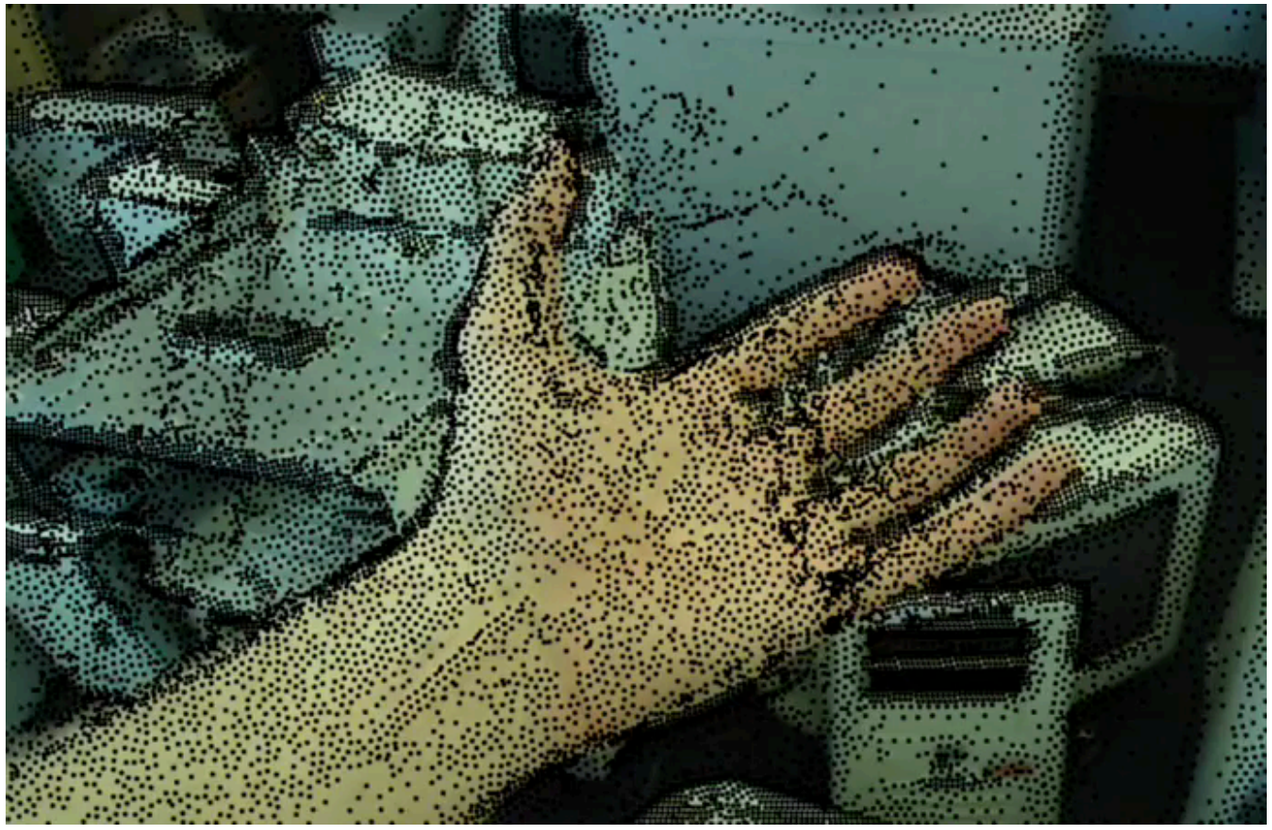


Note the difficulty in getting regularization “right”!

Remaining challenges: long-term optical flow

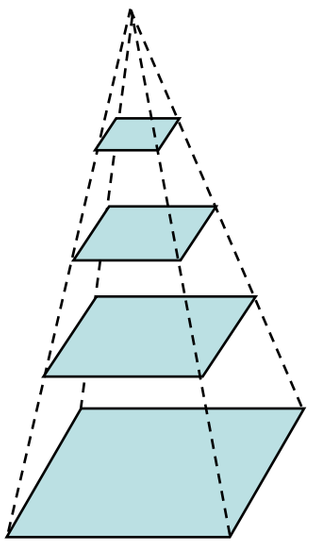
Combine long-term sparse feature tracking with variational flow regularization

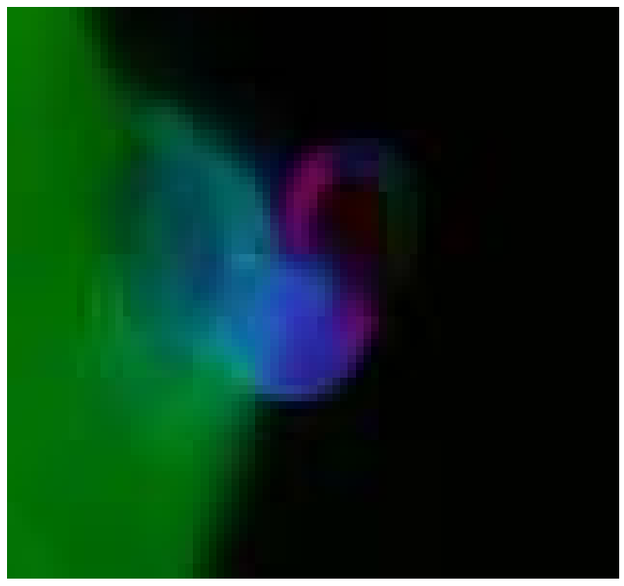
(http://rvsn.csail.mit.edu/pv/)



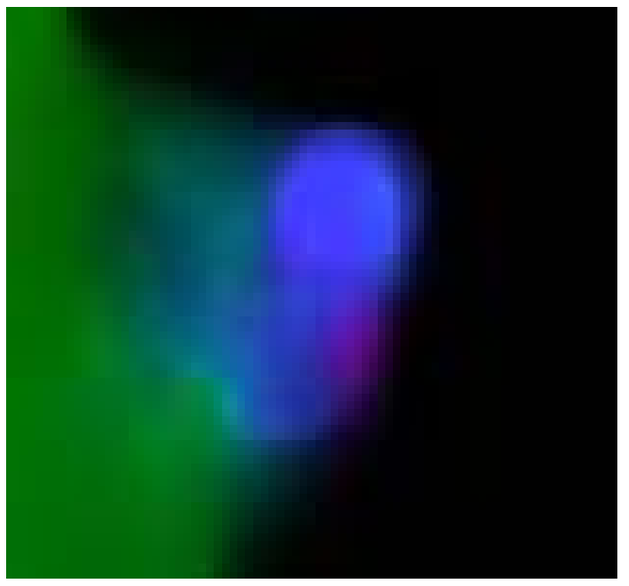
Note the difficulty in getting regularization “right”!

Remaining challenges: small things that move fast





Coarse-to-fine output



Ideal flow

Challenge: coarse-to-fine can’t generate the correct coarse flow for small things
(because they will be blurred out in coarse image)

Large Displacement Optical Flow*

Thomas Brox[1] Christoph Bregler[2] Jitendra Malik[1]

[1]University of California, Berkeley
Berkeley, CA, 94720, USA
{brox,malik}@eecs.berkeley.edu

[2]Courant Institute, New York University
New York, NY, 10003, USA
bregler@courant.nyu.edu

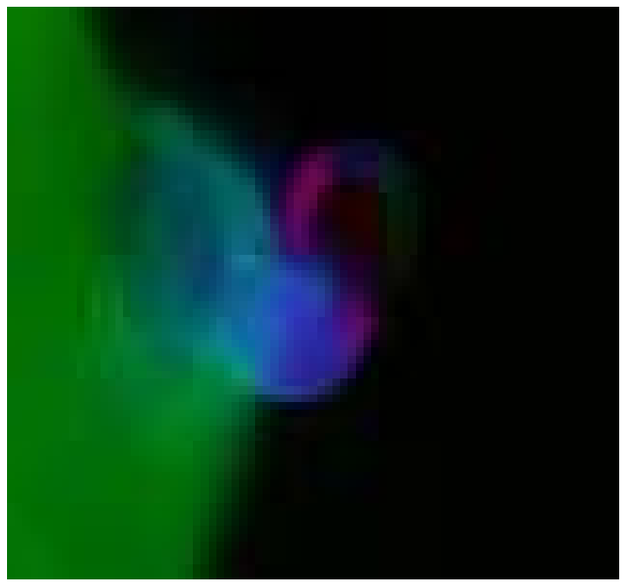
1. Estimate sparse correspondences across 2 frames with “classic” interest-points + descriptor matching
2. Use sparse matches as “priors” in dense variational optimization

Set of matchable points and estimated offsets: $\{(x_i, y_i, u_i, v_i)\}$

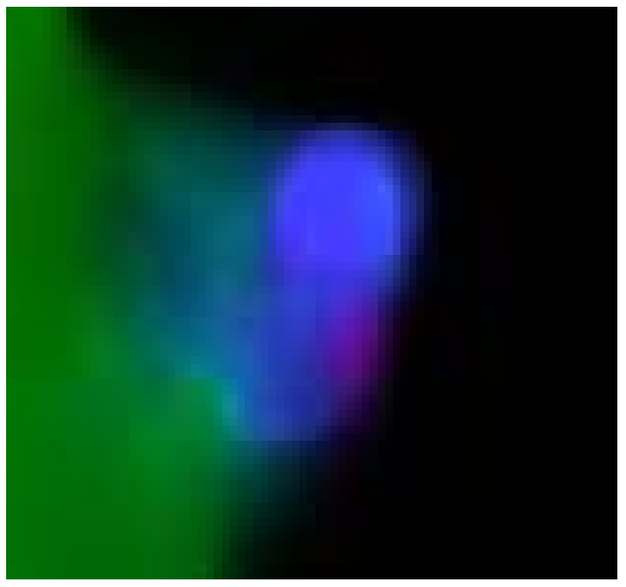
$$E_{match}(u,v) = \sum_i (u(x_i,y_i) - u_i)^2 + (v(x_i,y_i) - v_i)^2$$

$$\min_{u,v} E_{intensity} + E_{smooth} + E_{match}$$

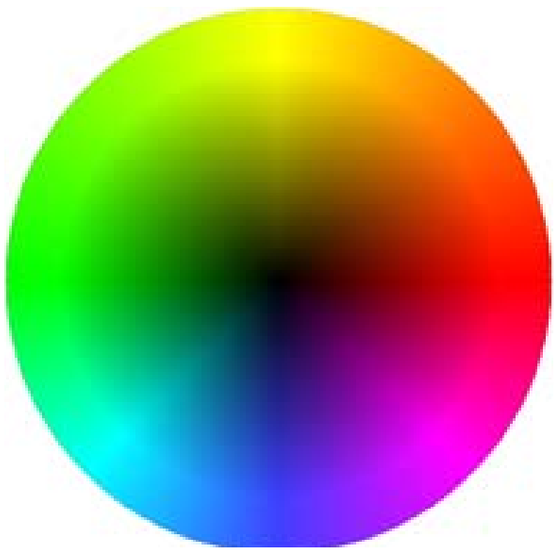
Examples



no E_{match}



with E_{match}

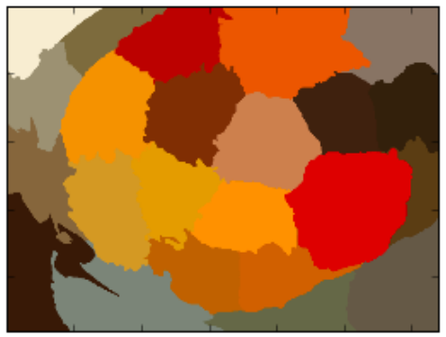


Outline

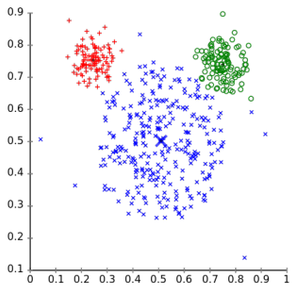
- Lucas Kanade
- Egomotion
    - Motivation
    - Time-to-Contact, Parallax, Focus-of-Expansion
- Optical Flow
    - Motivation, aperture problem
    - Sparse (KLT) vs Dense (variational)
    - Optimization tools: robust errors, varational coarse-to-fine, markov-random feilds
    - **Segmentation** (dominant motion estimation, background subtraction, layered models)

Motion segmentation

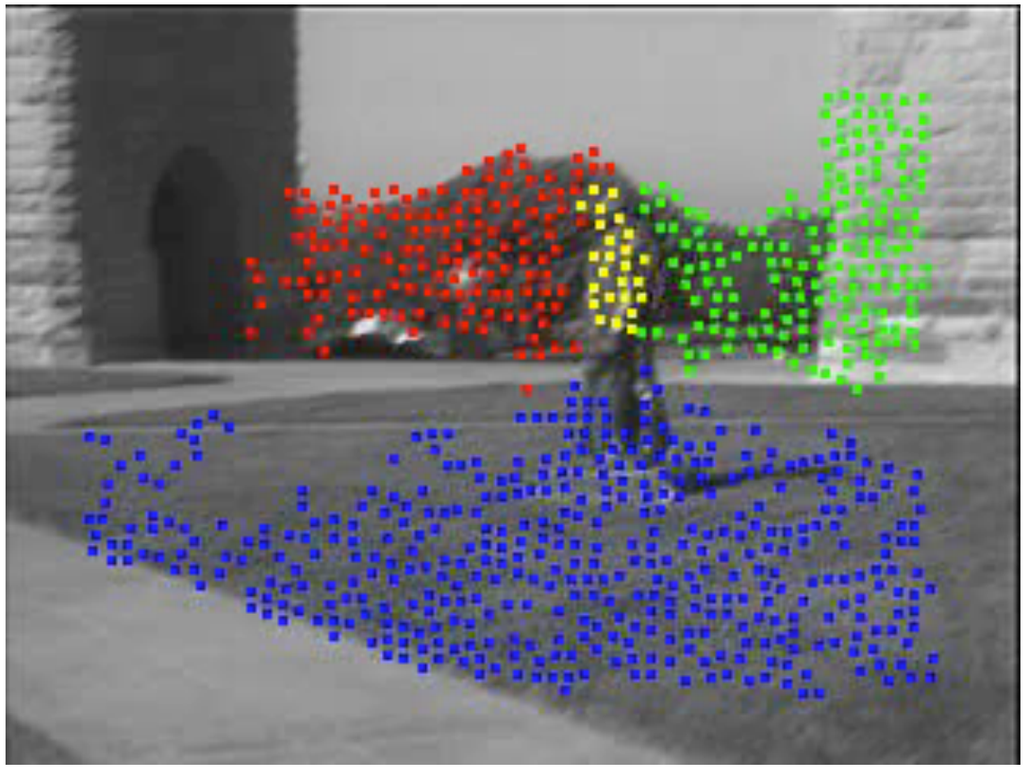
Treat as pixel-clustering problem



pixel color clustering



pixel motion clustering



1. Obtain an initial estimate of flow (sparse or dense)
2. Cluster pixels using feature vectors (consisting of flow, RGB, etc.)

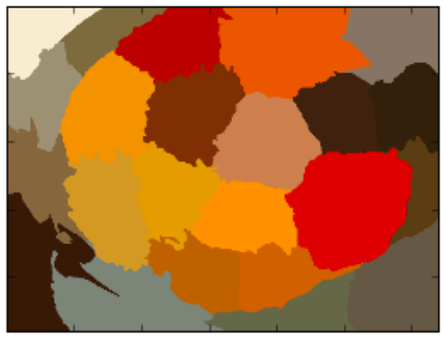
Generalize K-means to fit a parametric model (e.g., affine warp) rather than a centroid

Weiss & Adelson, CVPR 96

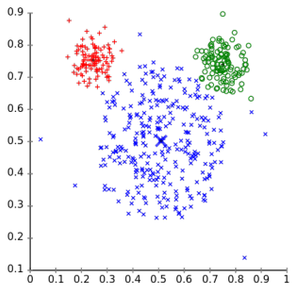
Uses “soft” K-means or EM algorithm

Motion segmentation

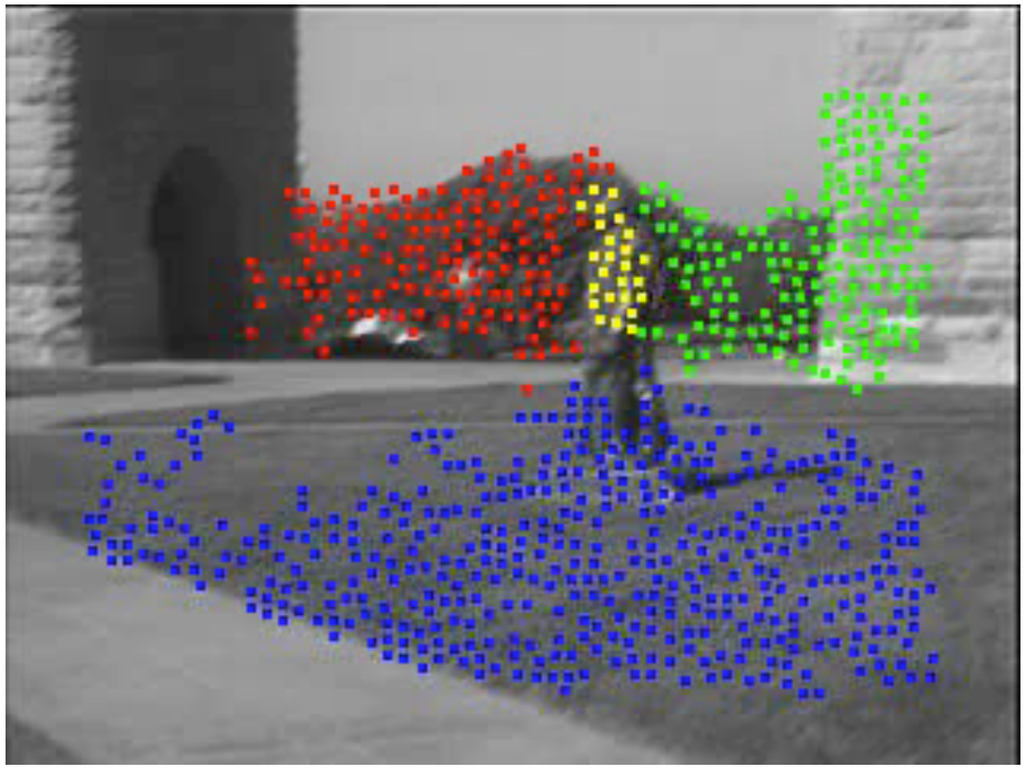
Treat as pixel-clustering problem



pixel color clustering



pixel motion clustering



1. Obtain an initial estimate of flow (sparse or dense)
2. Cluster pixels using feature vectors (consisting of flow, RGB, etc.)

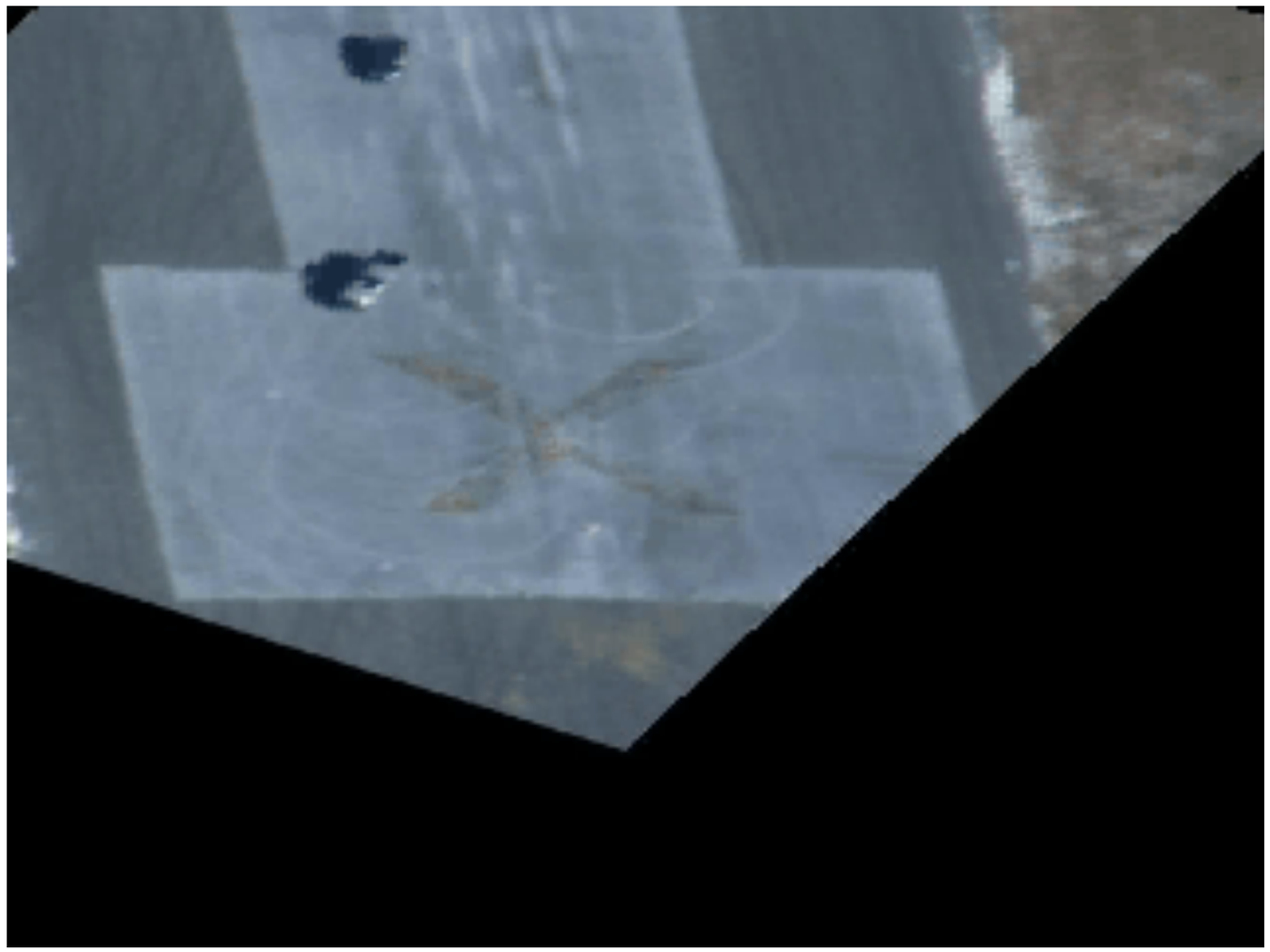
Generalize K-means to fit a parametric model (e.g., affine warp) rather than a centroid

Weiss & Adelson, CVPR 96

Uses “soft” K-means or EM algorithm

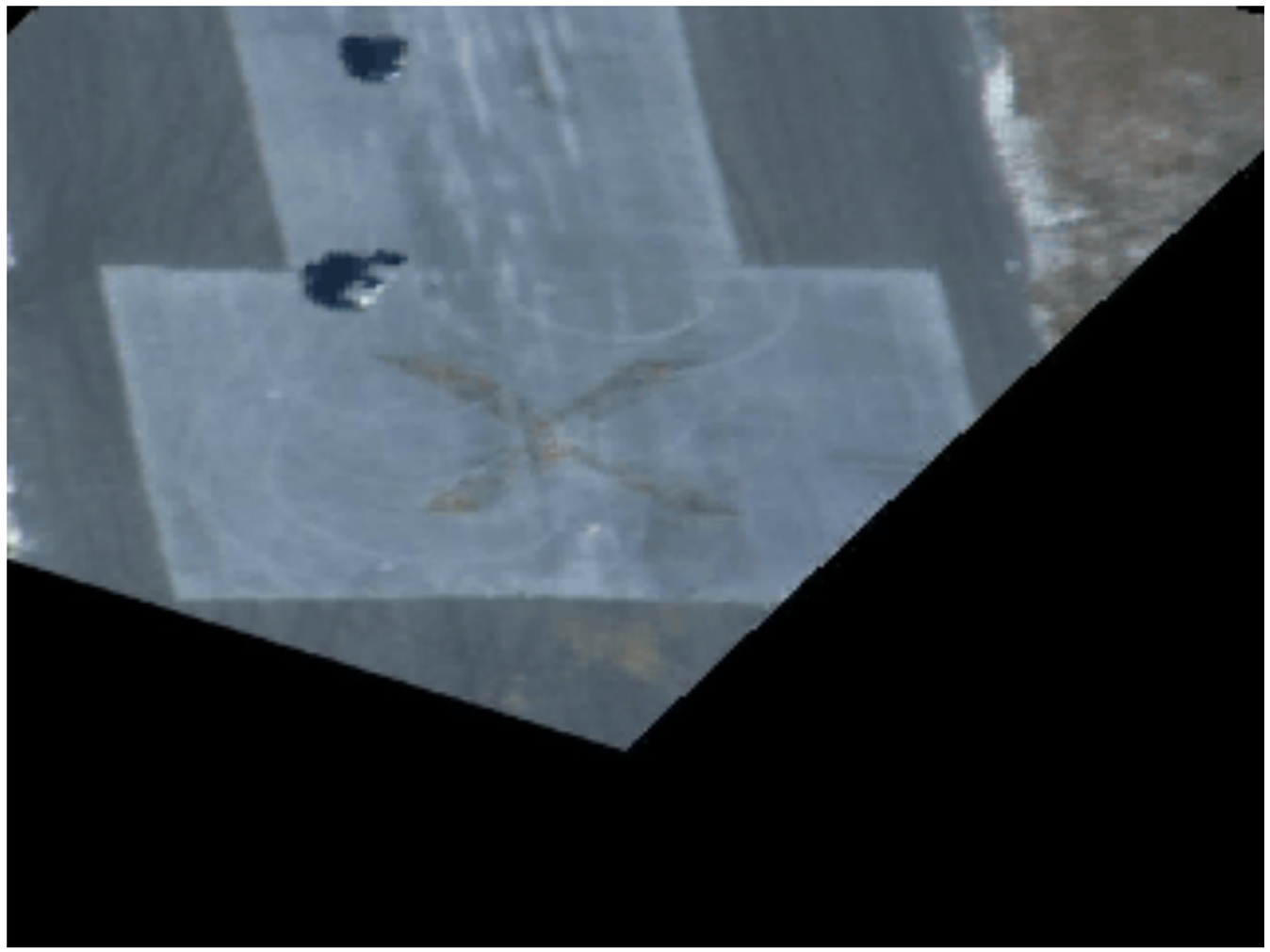
Motion segmentation: special case (I)

Treat objects as *outliers* when estimating a global homography alignment



Motion segmentation: special case (I)

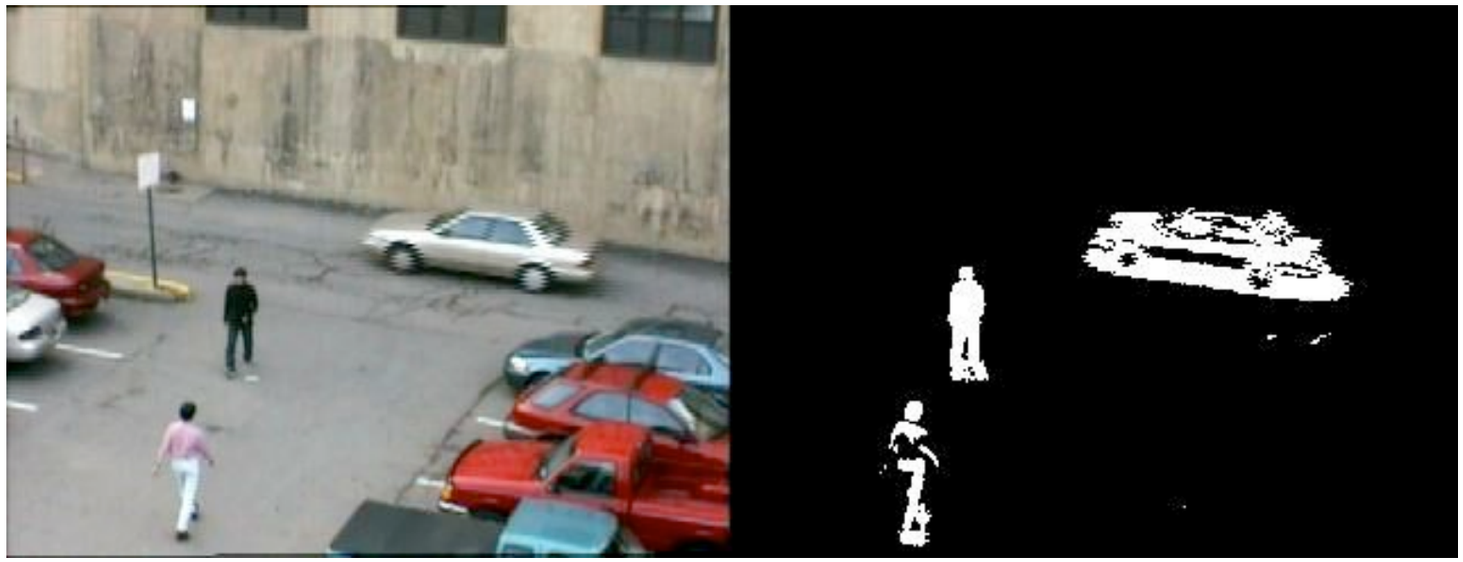
Treat objects as *outliers* when estimating a global homography alignment

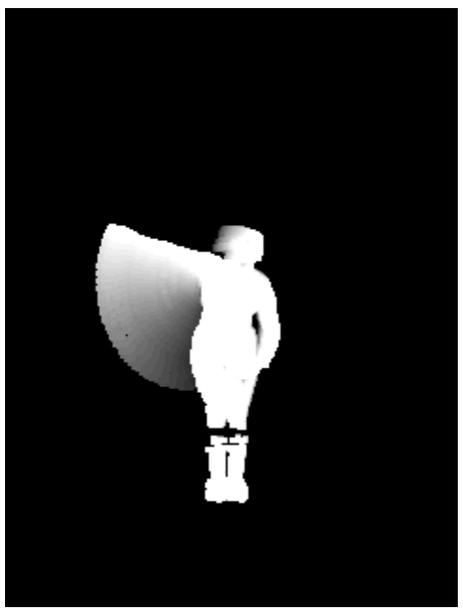


Motion segmentation: special case (II)

Background subtraction

Once we have background image/mosiac (trivial for a stationary camera), how do we identify foreground?

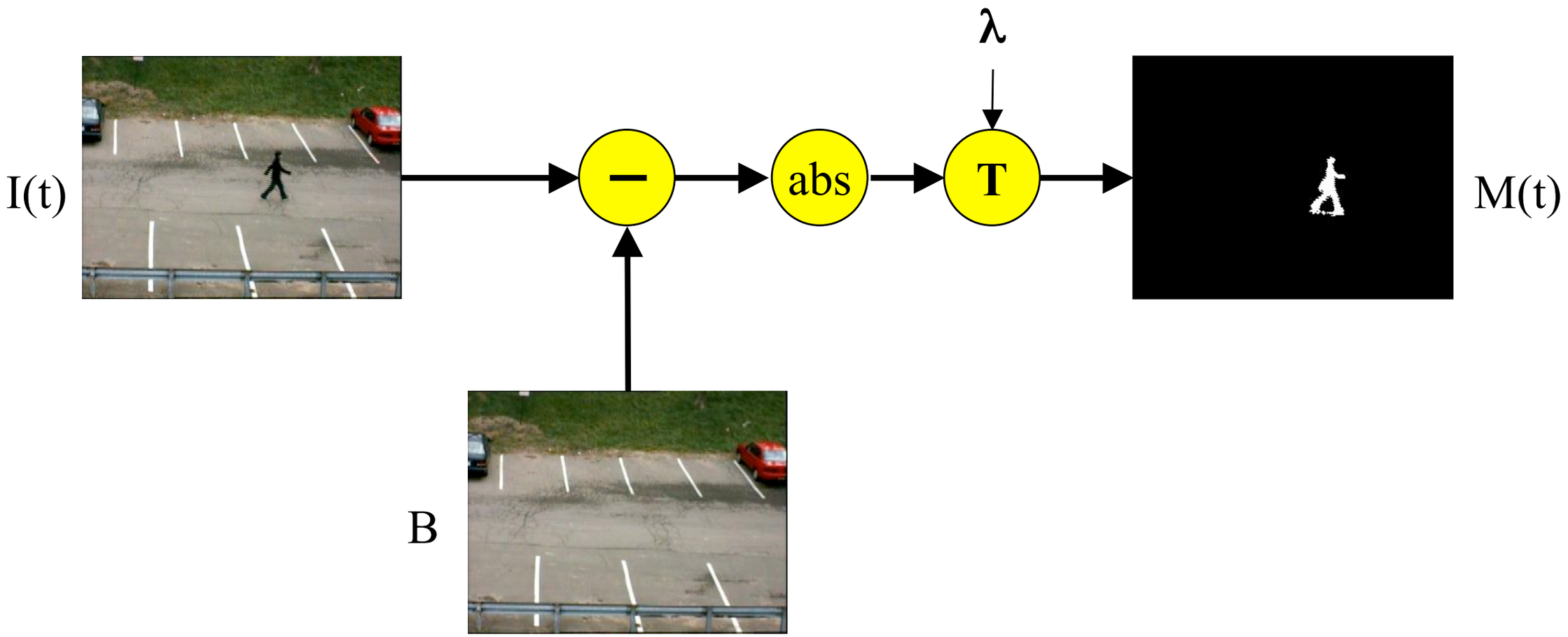




Very commonly-used technique, so we'll spend a few slides on it…

A naive approach

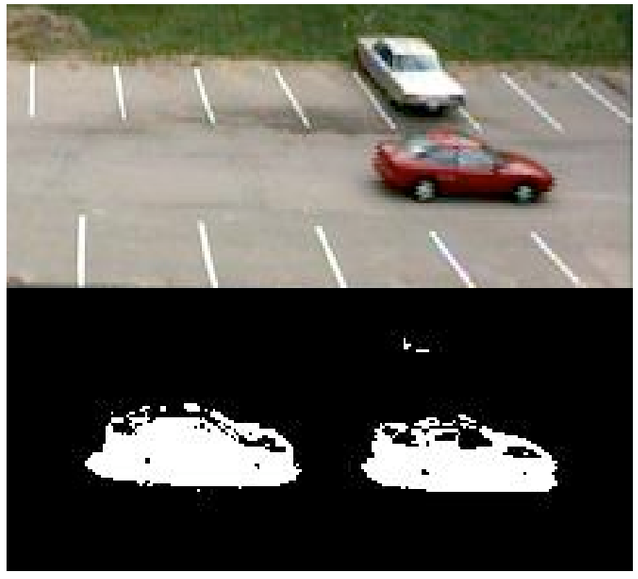
$$M(t) = ||B - I(t)|| > \lambda$$



Simplest approach: assume we have a backgorund image B

Difficulties

Overlapping foreground objects are merged together



Formerly static objects (that now move) result in ghosting

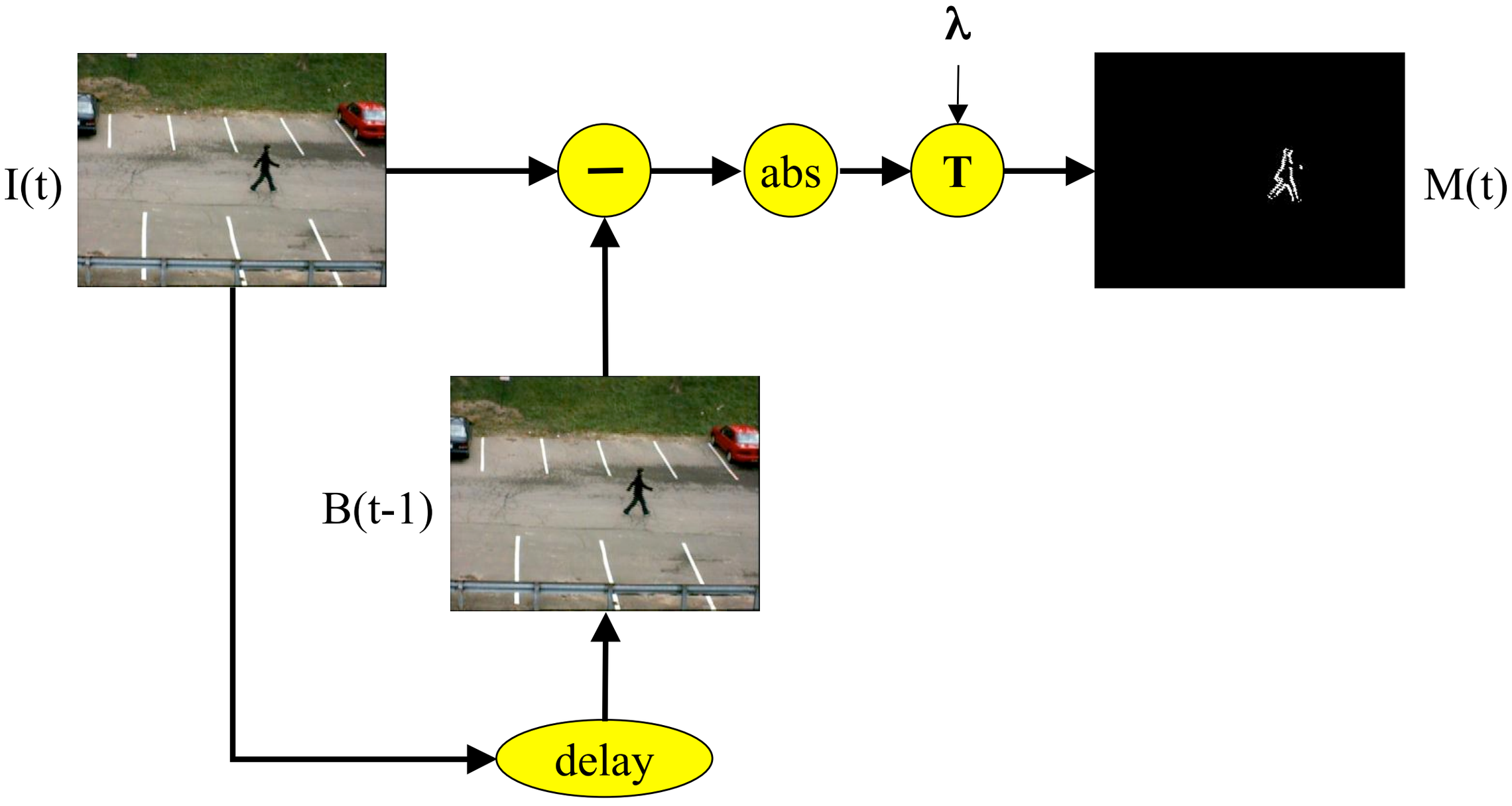
Senstive to small movements in scene (trees) and changes in illumination (sunlight)

Senstive to small movements of camera

One soln: frame-differencing

$$M(t) = ||B - I(t)|| > \lambda$$

$$B = I(t-1)$$



Mod 1: Use previous frame as background image

Adjusting temporal scale of differencing

Note what happens when we adjust the temporal scale (frame rate) at which we perform two-frame differencing …

Define D(N) = || I(t) - I(t+N) ||

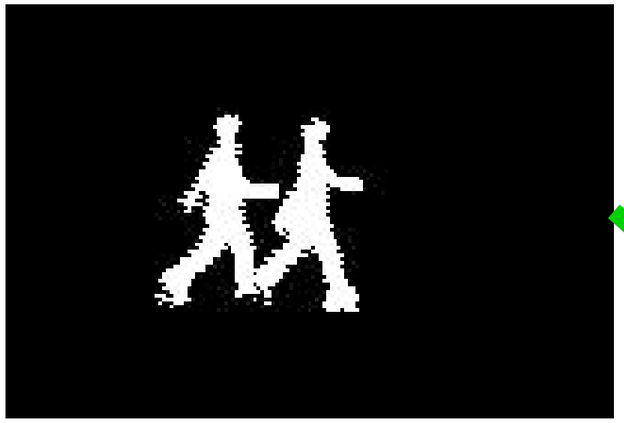
I(t) **D(-1)** **D(-3)** **D(-5)** **D(-9)** **D(-15)**

more complete object silhouette, but two copies (one where object used to be, one where it is now).

A neat “trick”: 3-frame differencing

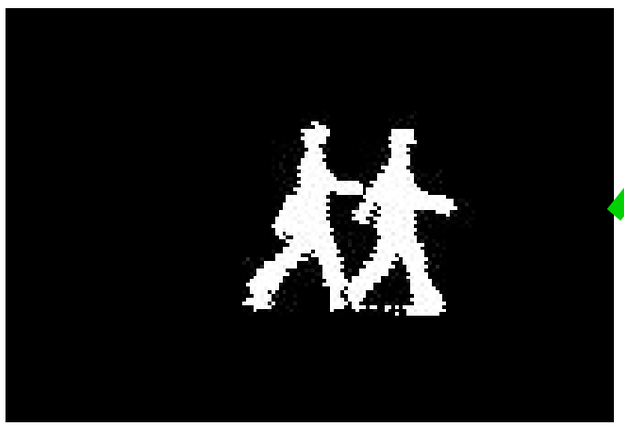
The previous observation is the motivation behind three-frame differencing

D(-15)



where object was,
and where it is now

D(+15)



where object is now,
and where it will be

AND

where object is now!

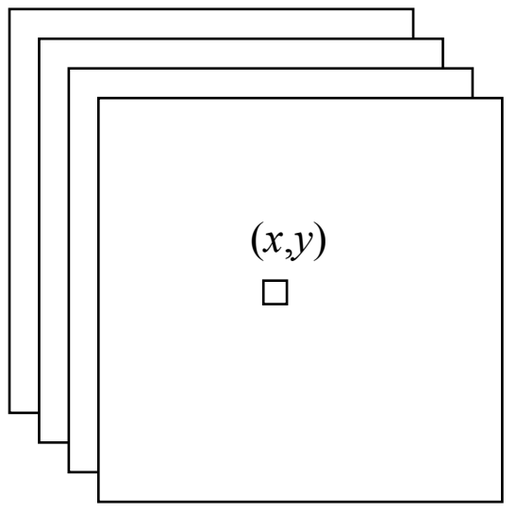
But its hard to find a good frame rate

Choice of good frame-rate for three-frame differencing depends on the size and speed of the object

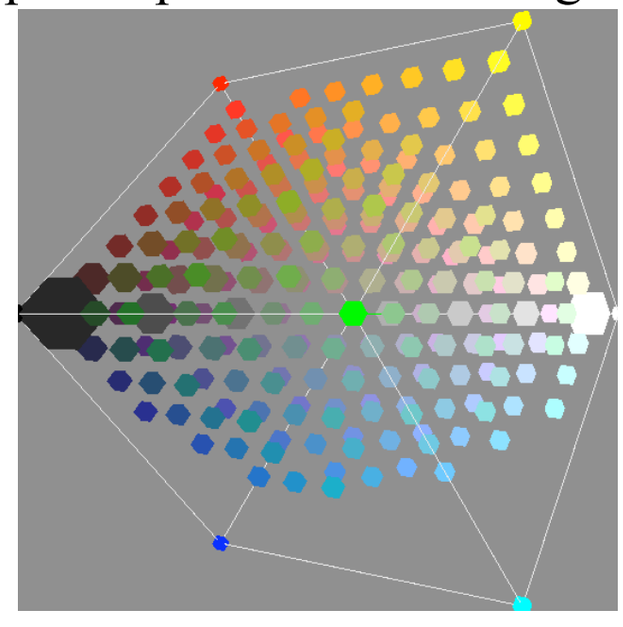
# frames skipped			
1			**35**
5			**45**
15 ← This worked well for the person			**55**
25			**65**

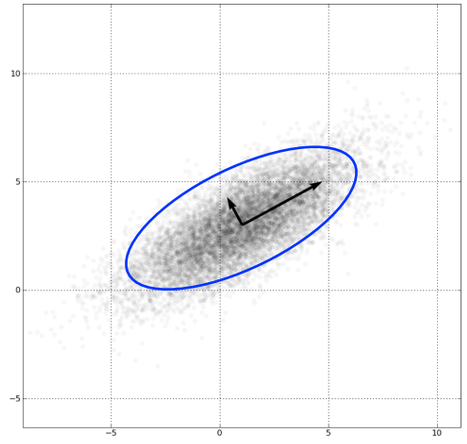
What's a "principled" way to build background model?

Statistical color models: $P(I(x,y)|bg) > \lambda$

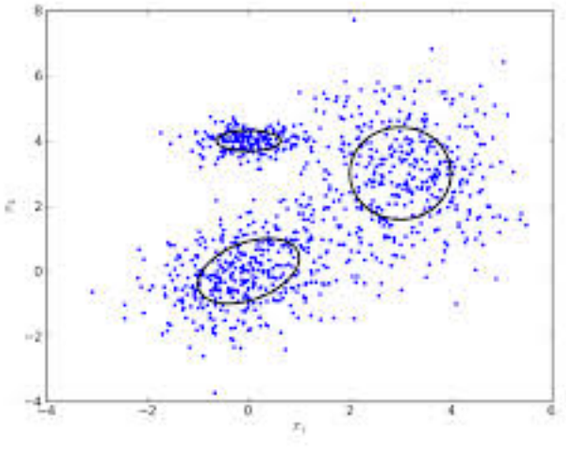


pixel-specific color histogram





$$P(I) = N(I; \mu, \Sigma)$$



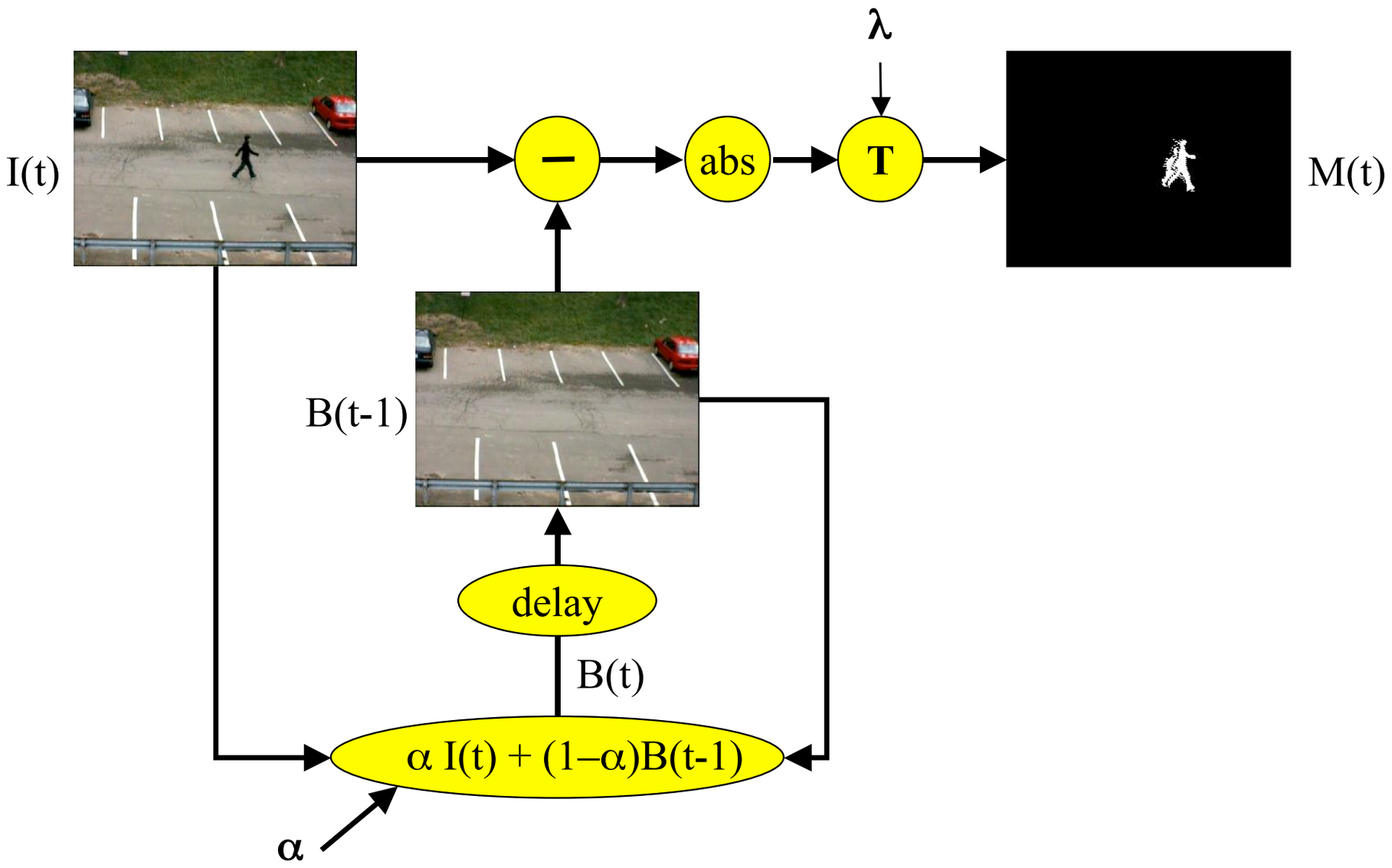
$$P(I) = \sum_i \pi_i N(I; \mu, \Sigma)$$

Online statistical learning
…of say, mean of distribution at each pixel (x,y)

$$M(t) = [B(t-1) - I(t)] > \lambda$$

$$B(t) = \alpha I(t) + (1-\alpha)B(t-1)$$

alpha=1: frame differencing
alpha=0: fixed (initial) background image

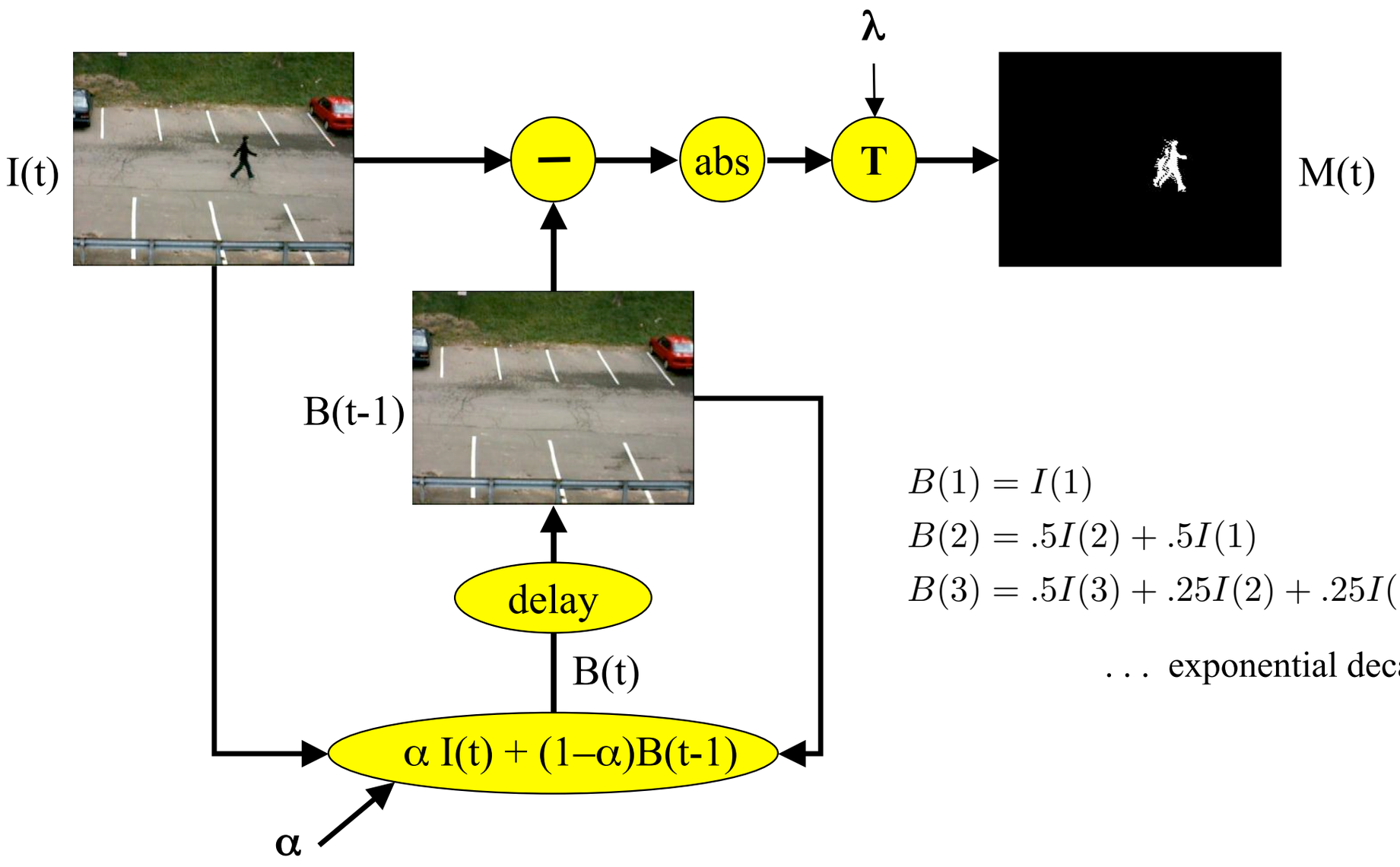


Online statistical learning
…of say, mean of distribution at each pixel (x,y)

$$M(t) = [B(t-1) - I(t)] > \lambda$$

$$B(t) = \alpha I(t) + (1-\alpha)B(t-1)$$

alpha=1: frame differencing
alpha=0: fixed (initial) background image



$$B(1) = I(1)$$
$$B(2) = .5I(2) + .5I(1)$$
$$B(3) = .5I(3) + .25I(2) + .25I(1)$$

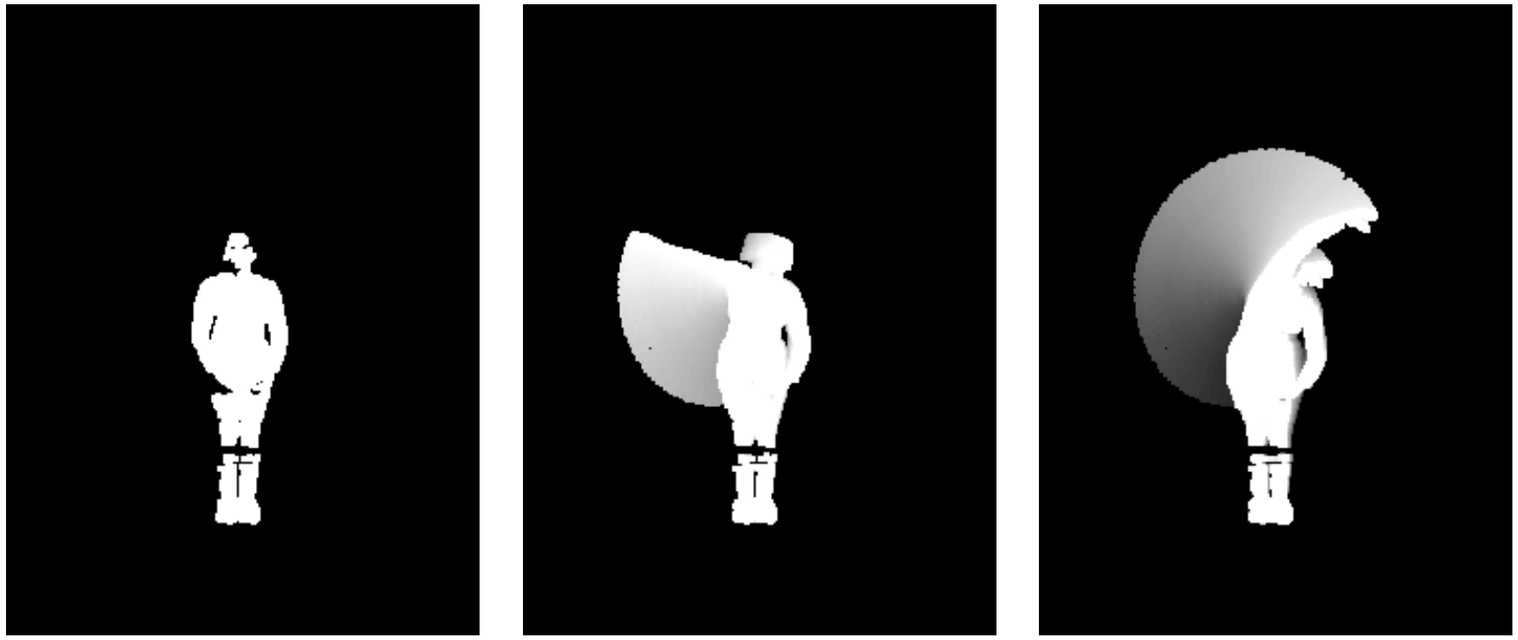
. . . exponential decay

Adaptive background subtraction

Adaptive background subtraction

Nifty visualizations: persistant frame differencing

FRAME-0 FRAME-35 FRAME-70



MHI-0 MHI-35 MHI-70

Use some previous method to identify foreground/background pixels

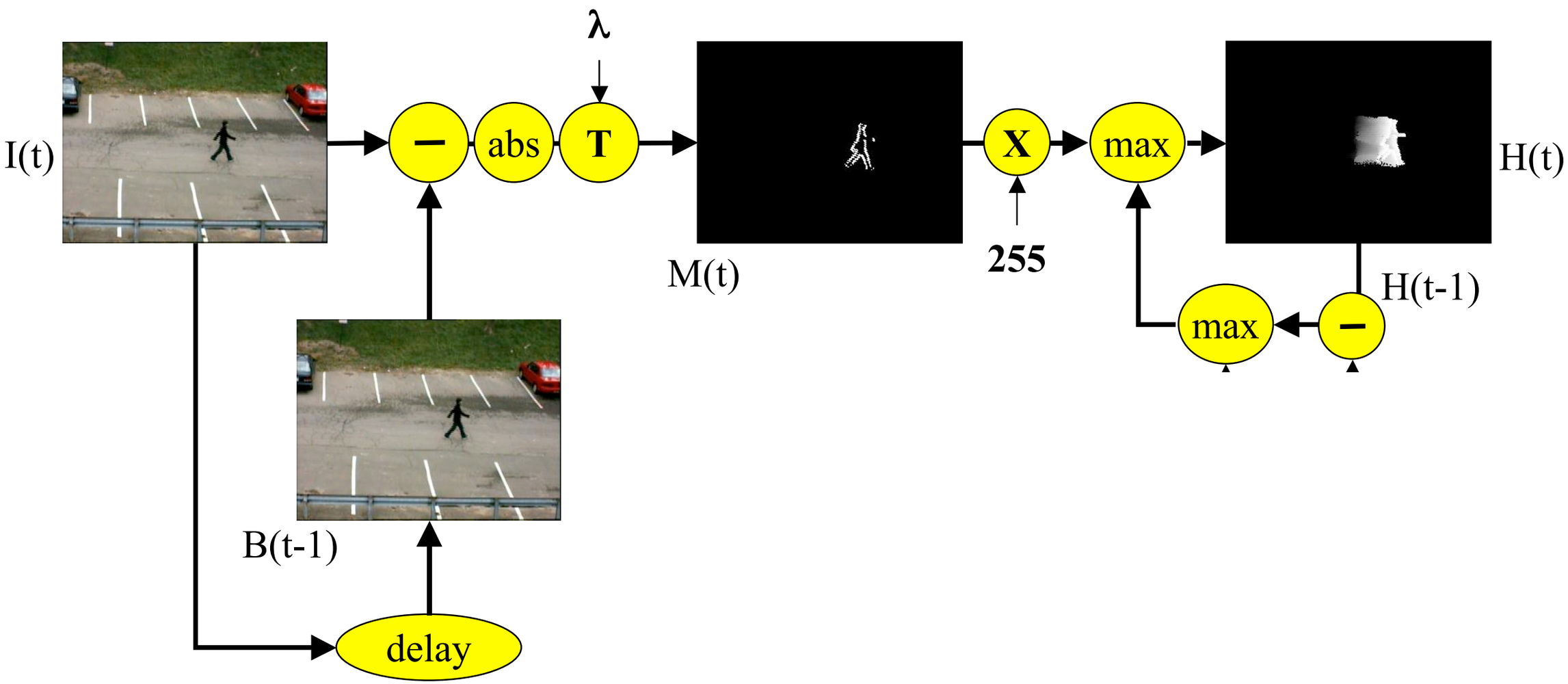
Mark each pixel with the last “time” is was declared foreground

Motion History Images

[Bobick & Davis]

Update history image with current binary mask + old (decremented) history image

$$H(t) = \max(255 * M(t), \max(H(t) - 1, 0))$$



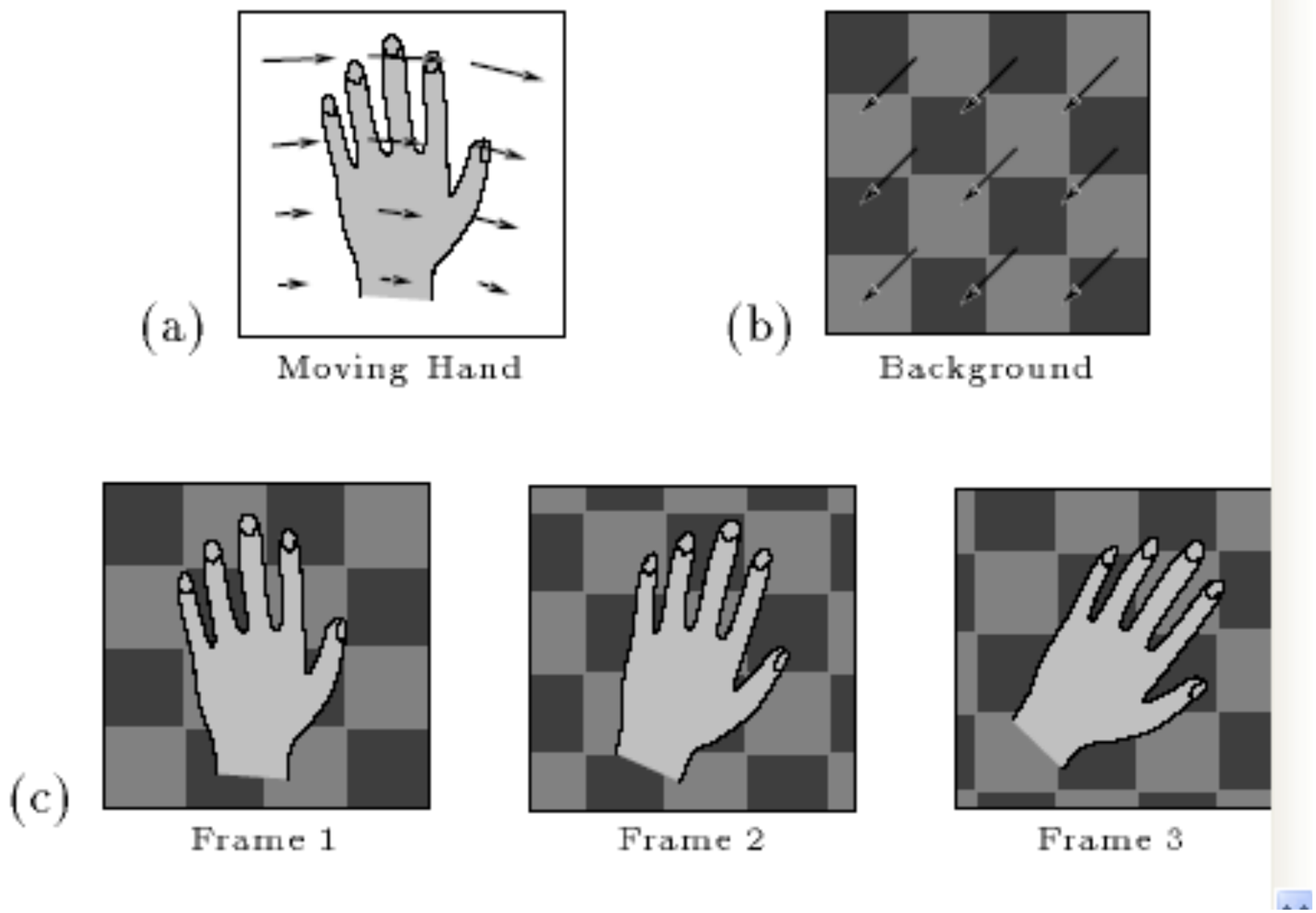
Motion History Images

Motion History Images

Outline

- Lucas Kanade
- Egomotion
    - Motivation
    - Time-to-Contact, Parallax, Focus-of-Expansion
- Optical Flow
    - Motivation, aperture problem
    - Sparse (KLT) vs Dense (variational)
    - Optimization tools: varational coarse-to-fine, markov-random feilds
    - Segmentation (dominant motion estimation, background subtraction, l**ayered models**)

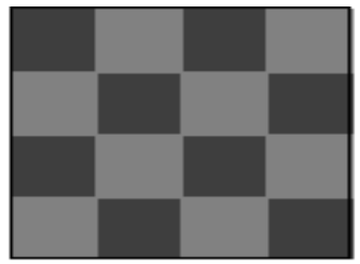
Layered model



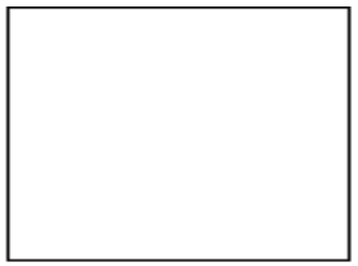
Direct analogy with *layers* in photoshop

Mathematical formalism

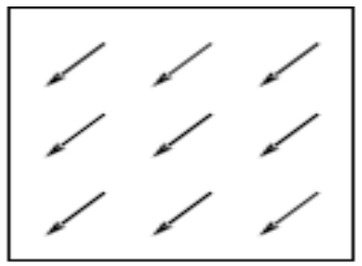
Layer 0 (BG)



Intensity map

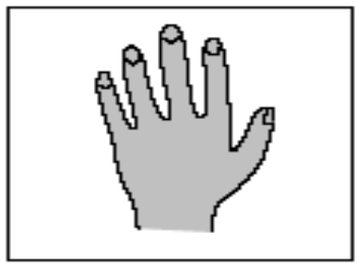


Alpha map



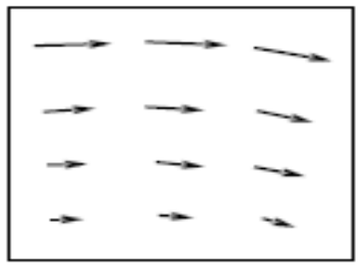
Velocity map

Layer 1



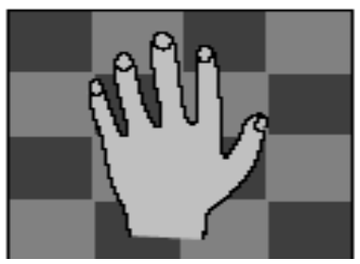
Intensity map

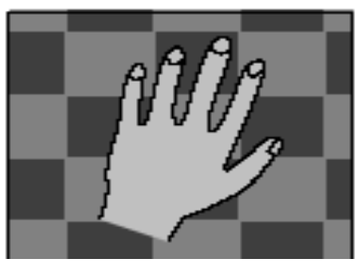
Alpha map



Velocity map

Alpha composite

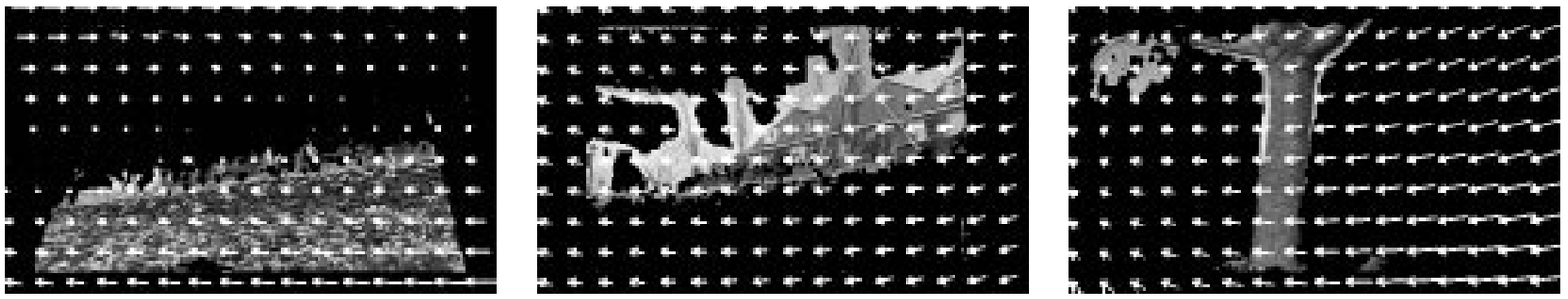




$$I_i(x,y) = \alpha_i(x,y)L_i(x,y) + (1-\alpha_i(x,y))I_{i-1}(x,y)$$

Representing Moving Images with Layers

John Y. A. Wang AND Edward H. Adelson



(a) (b) (c)

Figure 12: The layers corresponding to the tree, the flower bed, and the house shown in figures (a-c), respectively. The affine flow field for each layer is superimposed.

(a) (b) (c)

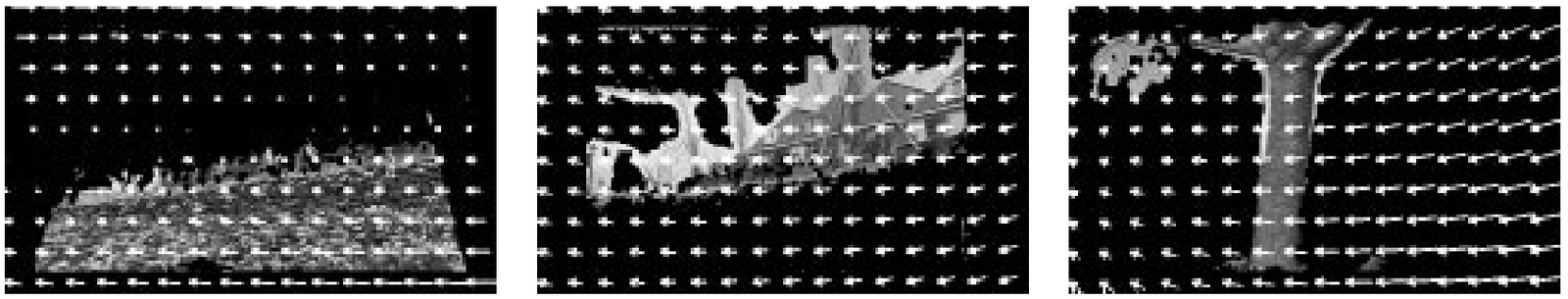
Figure 13: Frames 0, 15, and 30 as reconstructed from the layered representation shown in figures (a-c), respectively.

(a) (b) (c)

Figure 14: The sequence reconstructed without the tree layer shown in figures (a-c), respectively.

Representing Moving Images with Layers

John Y. A. Wang AND Edward H. Adelson



(a) (b) (c)

Figure 12: The layers corresponding to the tree, the flower bed, and the house shown in figures (a-c), respectively. The affine flow field for each layer is superimposed.

(a) (b) (c)

Figure 13: Frames 0, 15, and 30 as reconstructed from the layered representation shown in figures (a-c), respectively.

(a) (b) (c)

Figure 14: The sequence reconstructed without the tree layer shown in figures (a-c), respectively.

Inferring layers, motion, and appearance with EM clustering

Learning Flexible Sprites in Video Layers

Nebojsa Jojic
Microsoft Research
http://www.ifp.uiuc.edu/~jojic

Brendan J. Frey
University of Toronto
http://www.psi.toronto.edu

Inferring layers, motion, and appearance with EM clustering

Learning Flexible Sprites in Video Layers

Nebojsa Jojic
Microsoft Research
http://www.ifp.uiuc.edu/~jojic

Brendan J. Frey
University of Toronto
http://www.psi.toronto.edu

Outline

- Lucas Kanade
- Egomotion
    - Motivation
    - Time-to-Contact, Parallax, Focus-of-Expansion
- Optical Flow
    - Motivation, aperture problem
    - Sparse (KLT) vs Dense (variational)
    - Optimization tools: varational coarse-to-fine, markov-random feilds
    - Segmentation (dominant motion estimation, background subtraction, layered models)