Computer Vision I

Motion Estimation - 26.06.2013

TECHNISCHE UNIVERSITÄT DARMSTADT

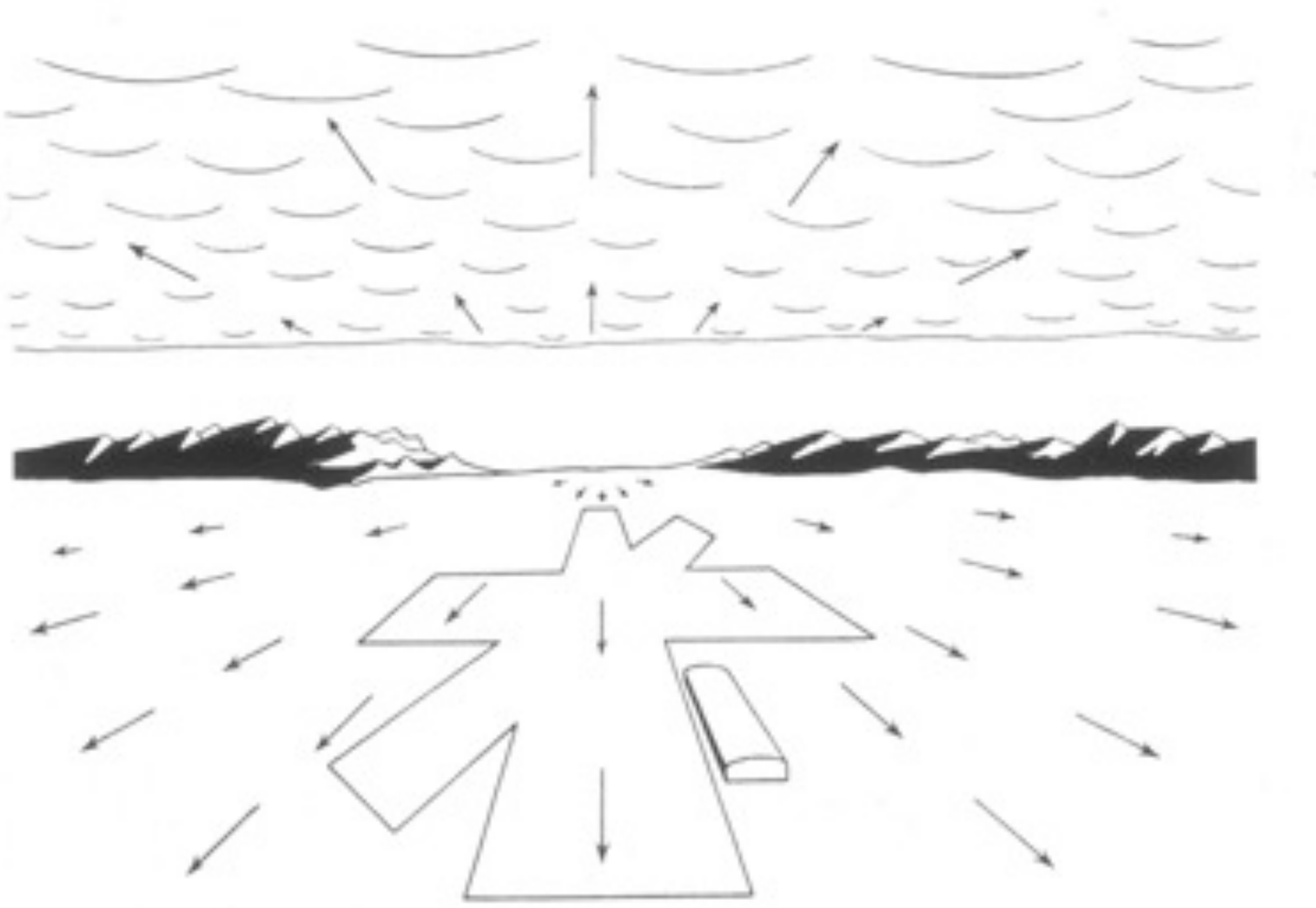
with slides from:

Michael Black
Rick Szeliski
Yair Weiss

vi visual inference

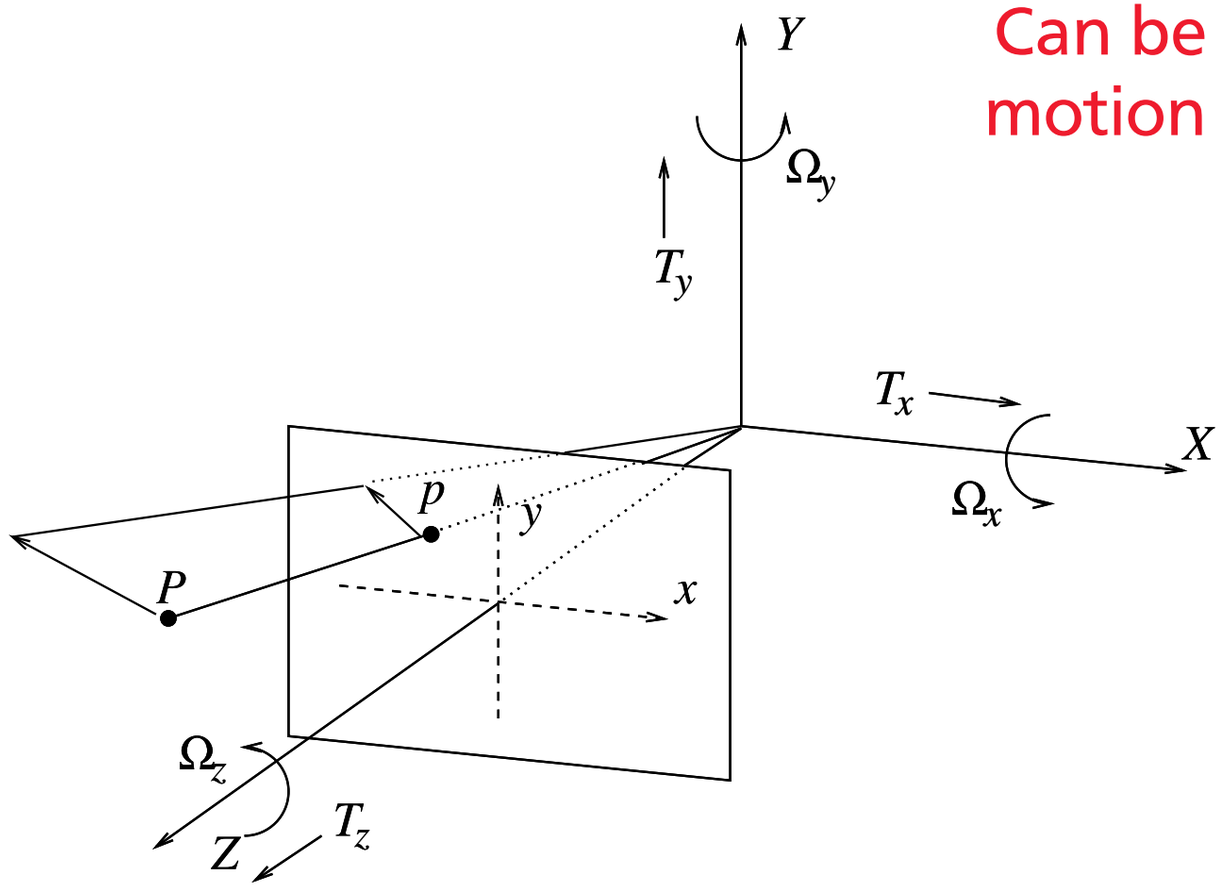
What is Image Motion?

Optical Flow



J. J. Gibson, The Ecological Approach to Visual Perception

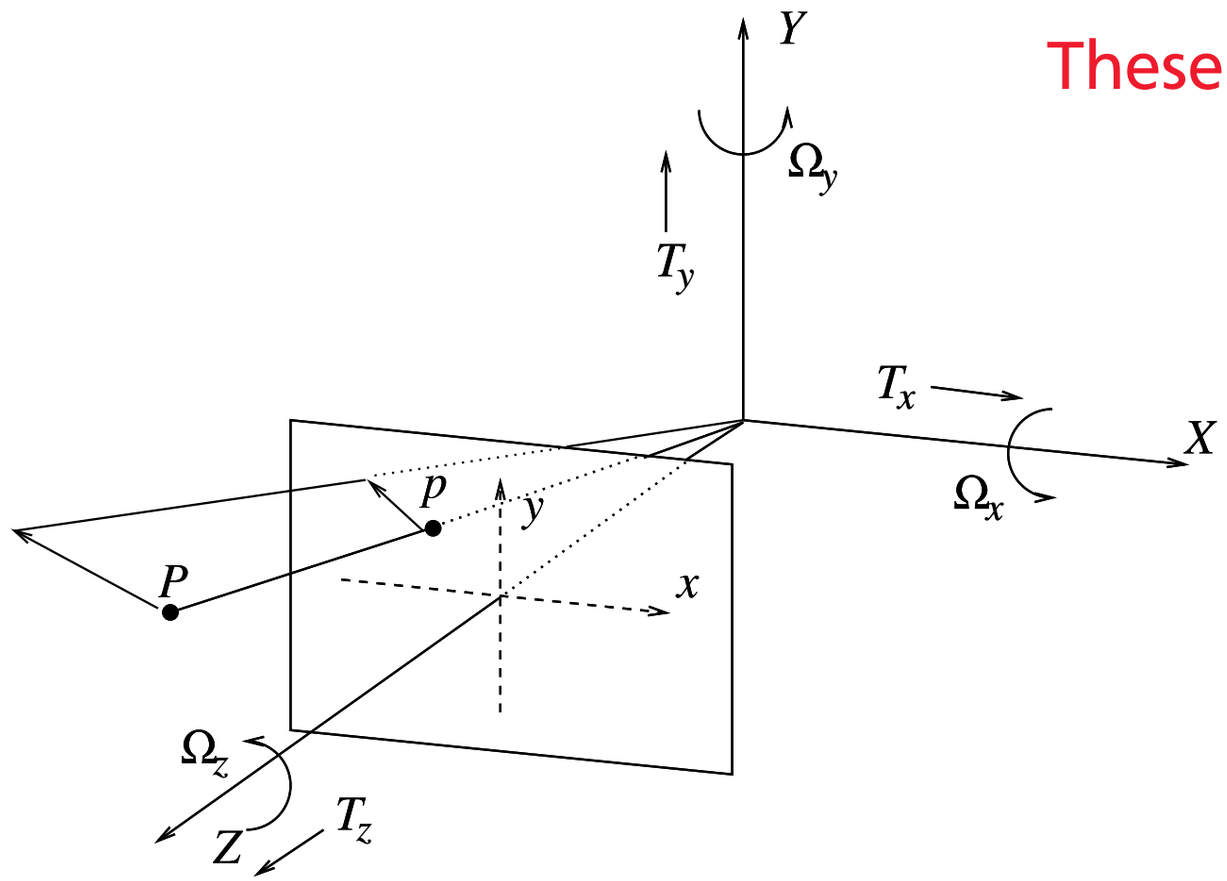
Motion Field



Can be the result of camera motion or object motion (or both)!

Motion field = 2D motion field representing the projection of the 3D motion of points in the scene onto the image plane.

Optical Flow



These are not the same!
Why?

Optical flow = 2D velocity field describing the apparent motion in the images.

Thought Experiment 1

TECHNISCHE UNIVERSITÄT DARMSTADT

Lambertian ball rotating in 3D

What does the 2D motion field look like?

What does the 2D optical flow field look like?

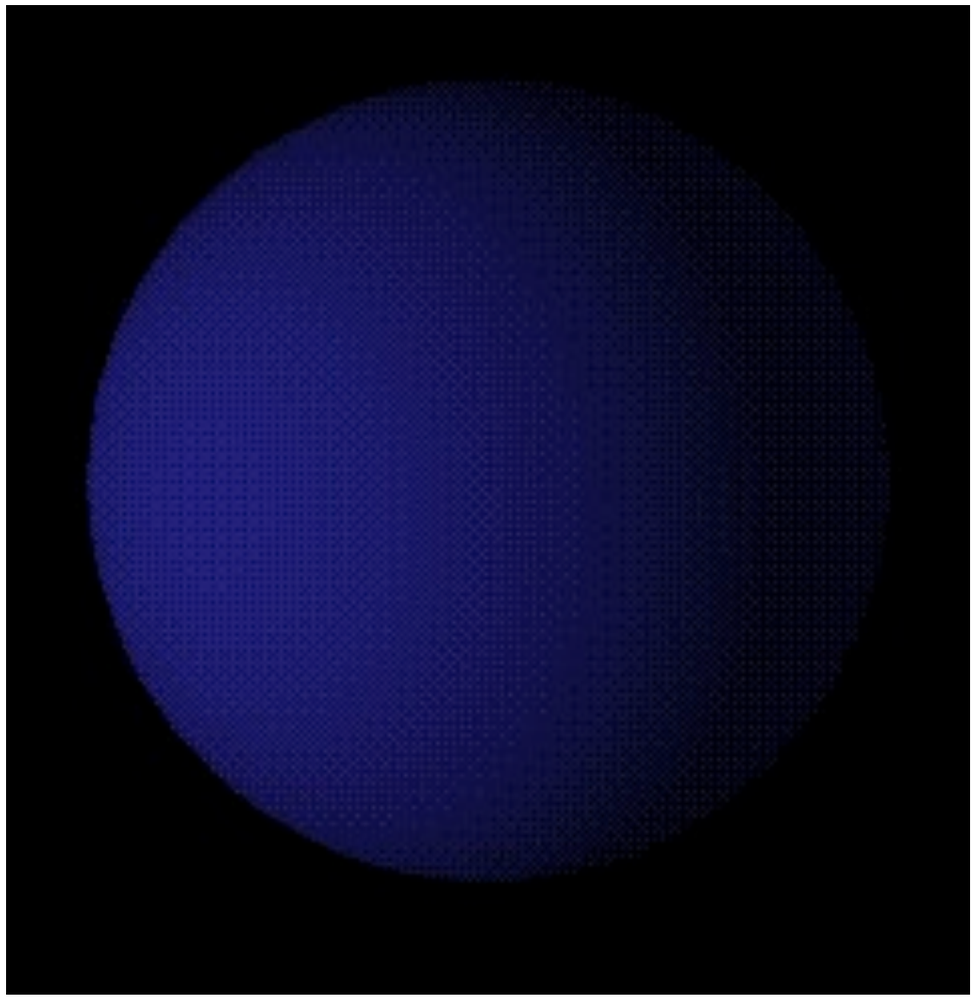


Image source: http://www.evl.uic.edu/aej/488/lecture12.html

Thought Experiment 2

TECHNISCHE UNIVERSITÄT DARMSTADT

Stationary specular ball, moving light source.

What does the 2D motion field look like?

What does the 2D optical flow field look like?

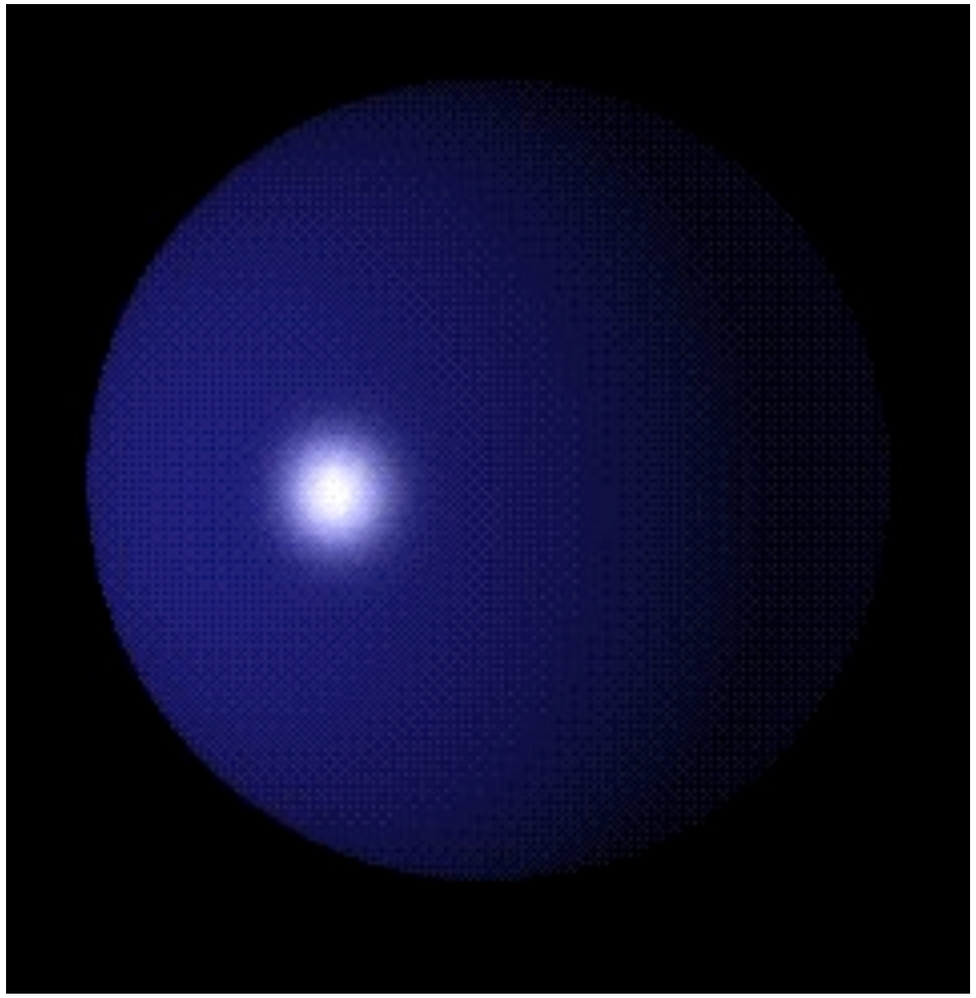
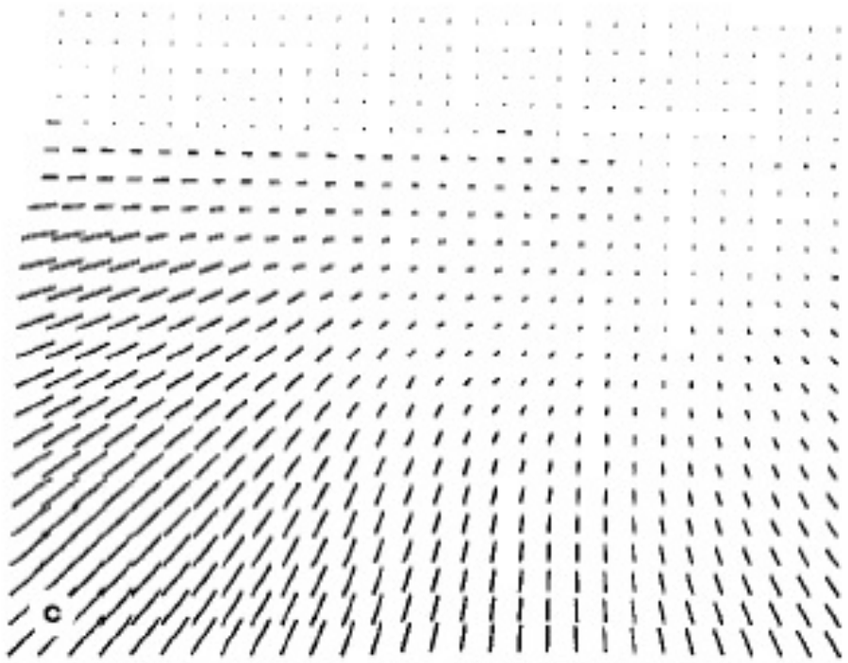


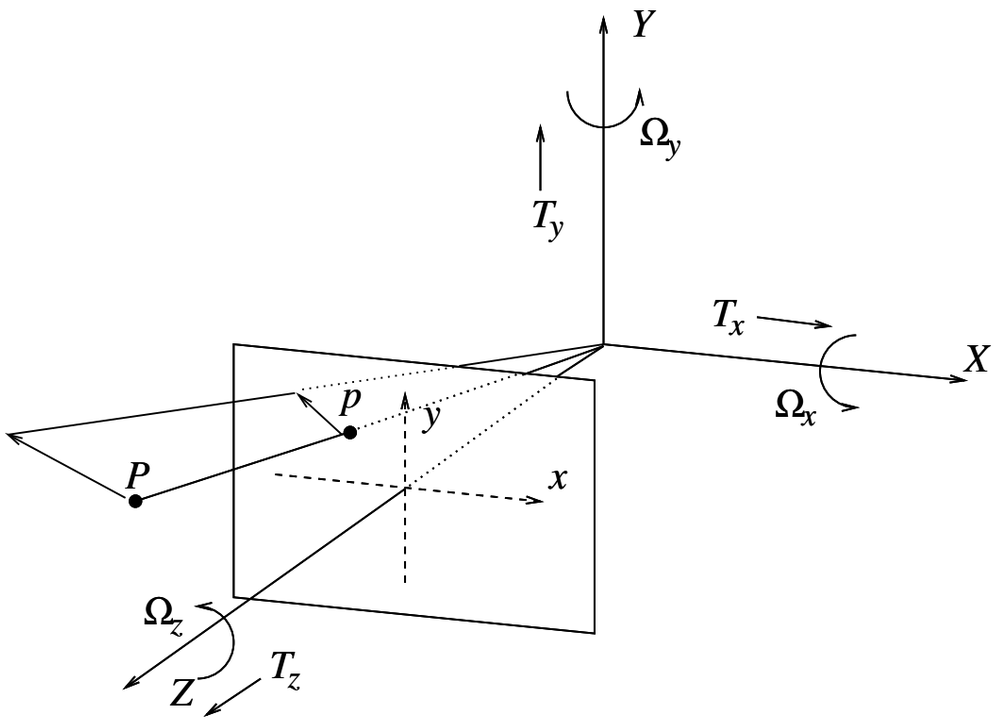
Image source: http://www.evl.uic.edu/aej/488/lecture12.html

Optical Flow Field

Image brightness at time t and location $\mathbf{x} = (x, y)$:

$$I(x, y, t)$$



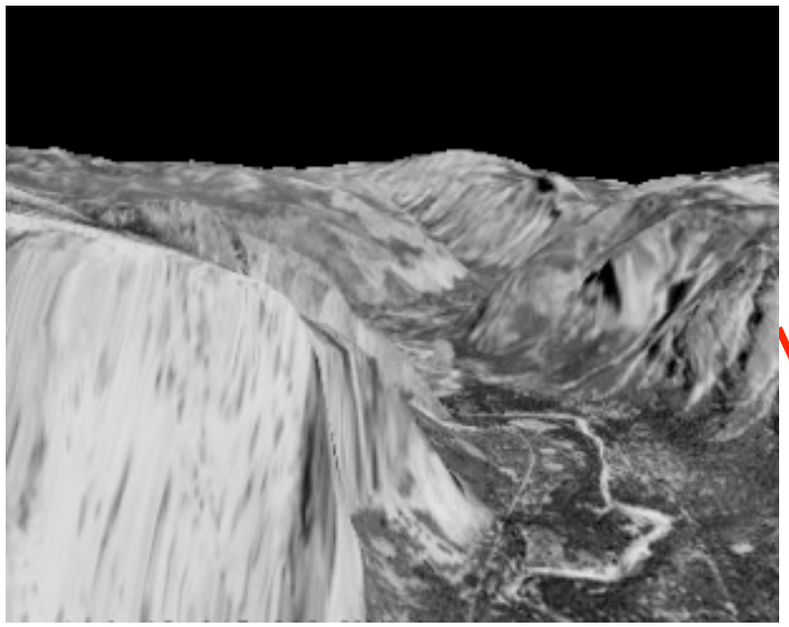


$u(x, y)$ Horizontal component

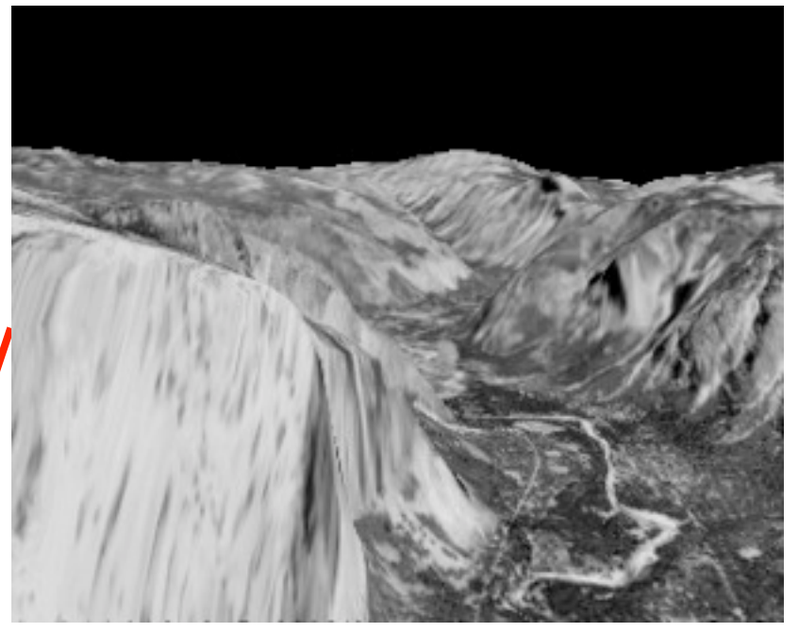
$v(x, y)$ Vertical component

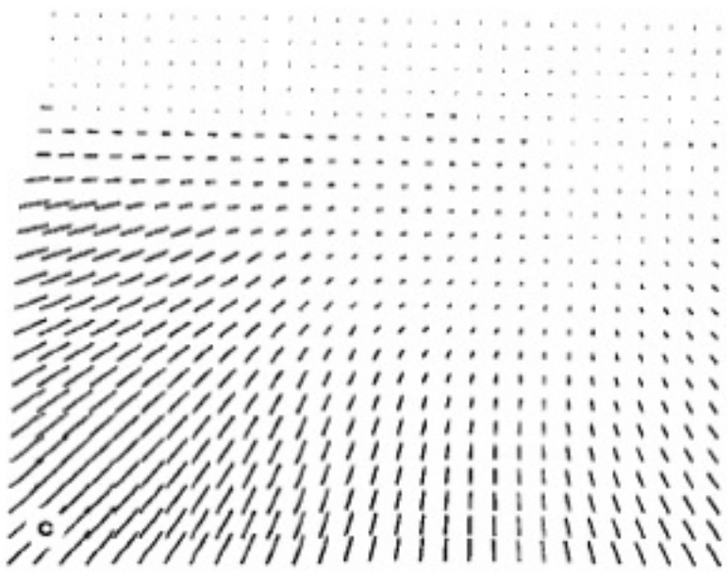
Optical Flow Estimation

time t

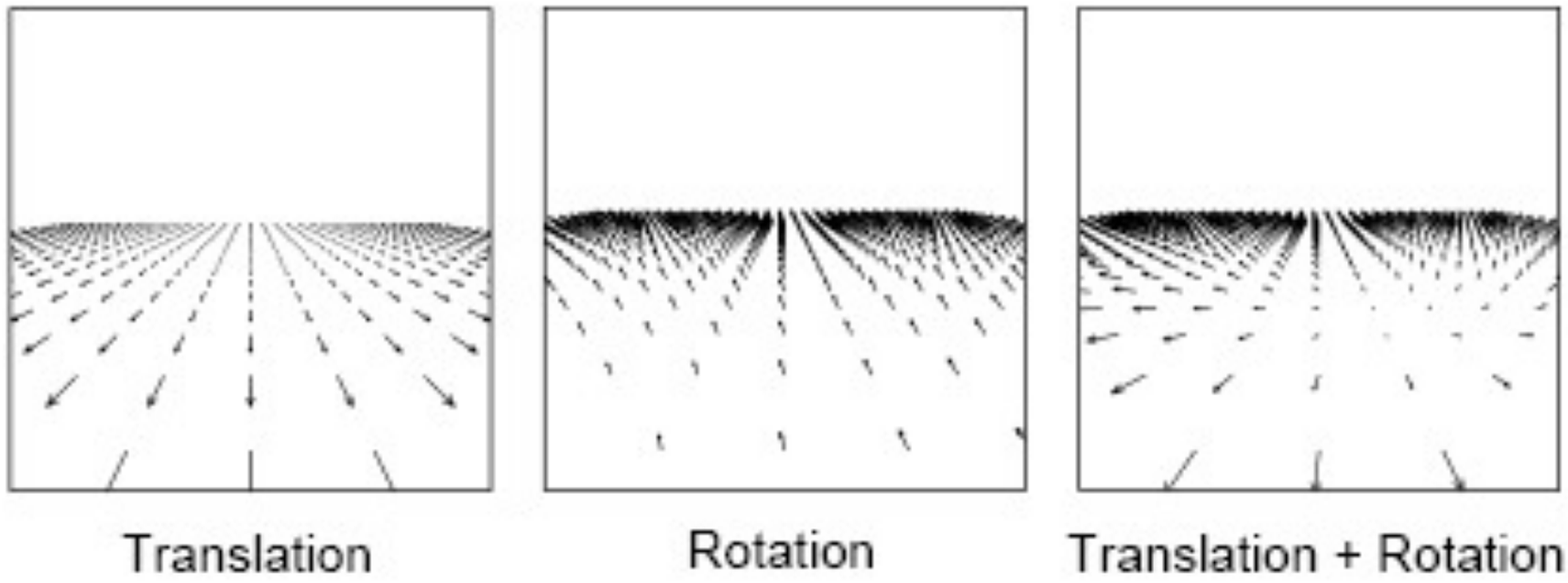


time t+1





Why Compute Flow?



Tells us something (maybe ambiguous) about the 3D structure of the world and the motion of the observer (if any).

Why Compute Flow?

TECHNISCHE UNIVERSITÄT DARMSTADT

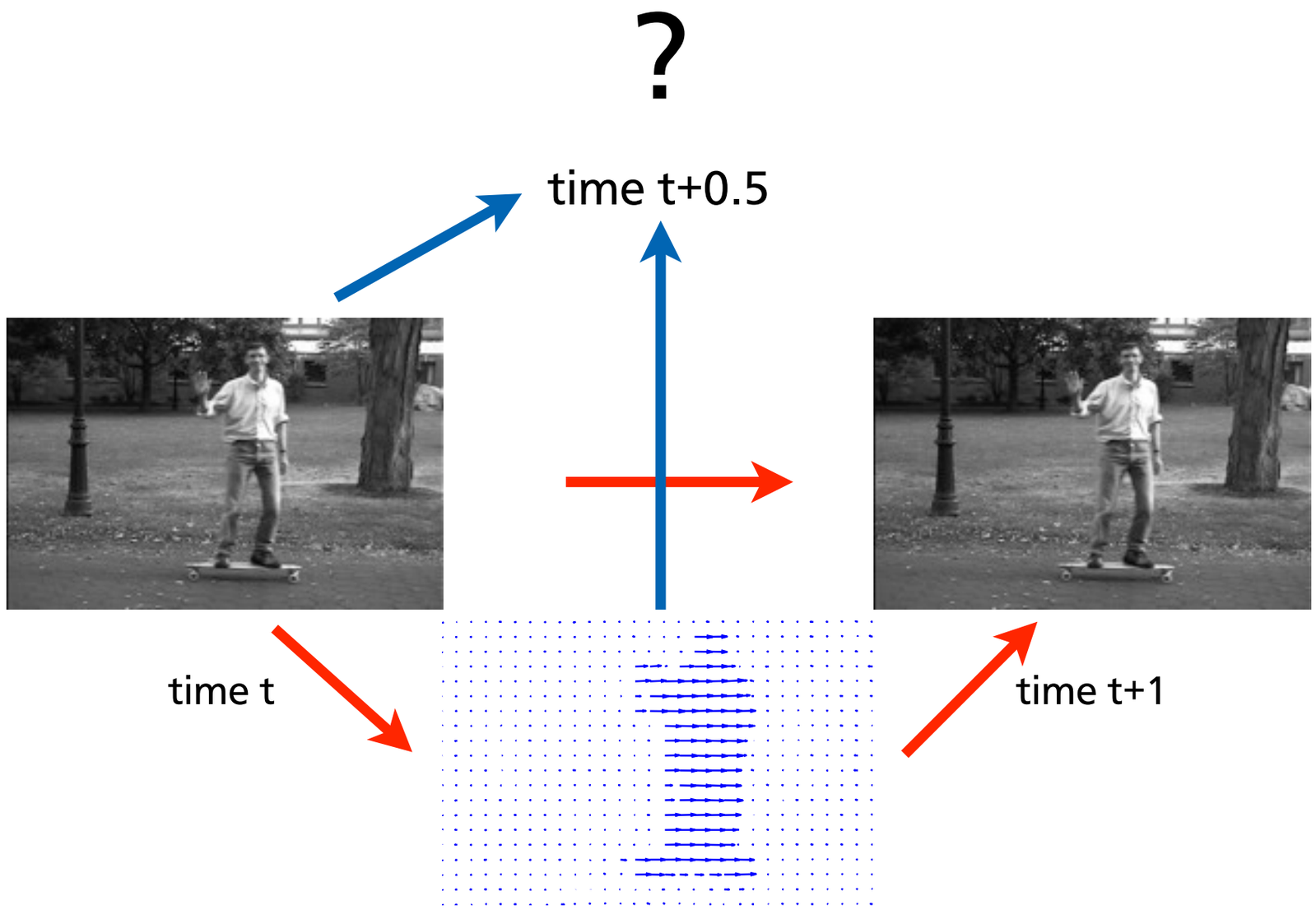
- Big application area: Interpolation
  - Morphing, warping...
  - Frame rate doubling, video compression, ...

Bullet time in “The Matrix”

Flow for Frame Rate Adaption

TECHNISCHE UNIVERSITÄT DARMSTADT

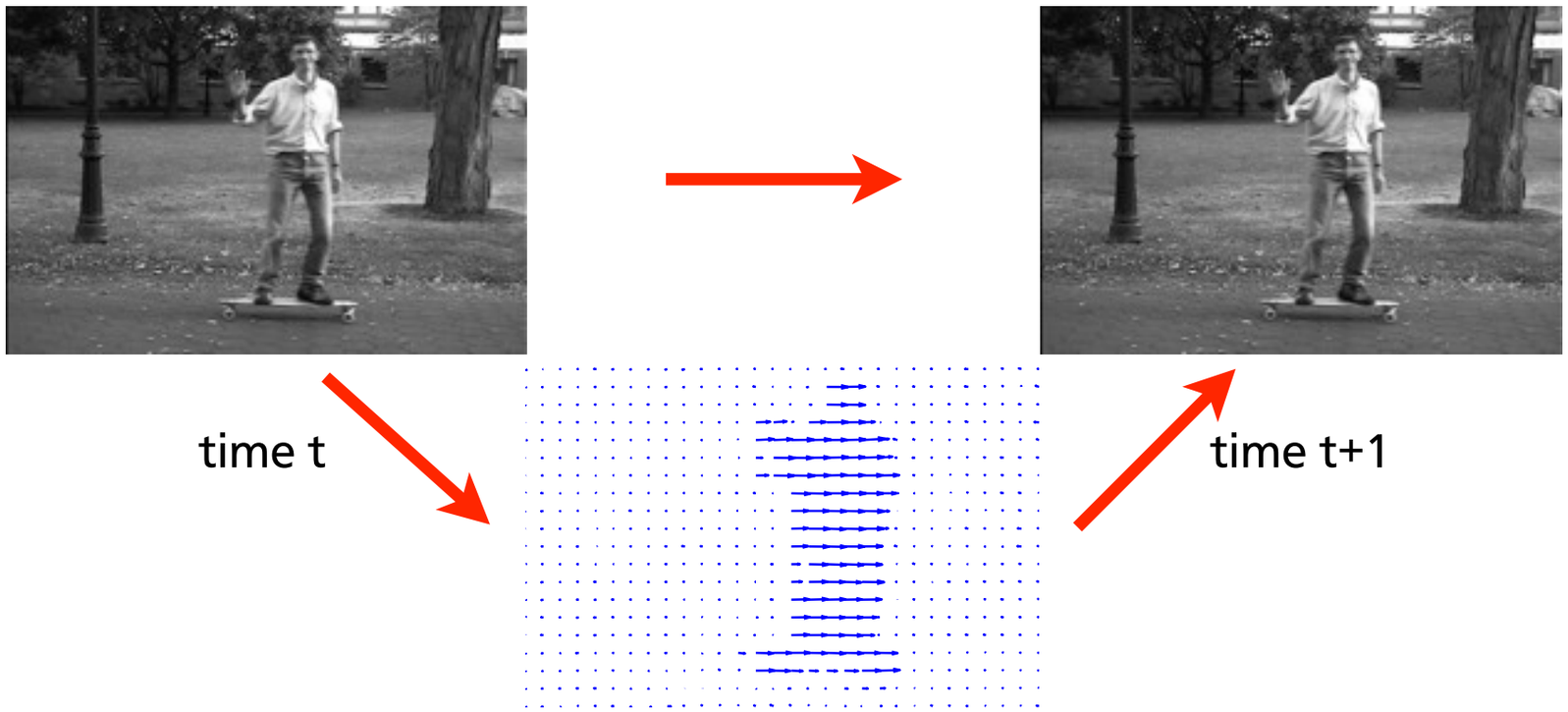
- If we know the image motion, we can compute images at intermediate time steps:



Flow for Video Compression

TECHNISCHE UNIVERSITÄT DARMSTADT

- If we want to compress an image sequence, we can predict new frames using the flow, and only store how to "fix" this prediction.
- The flow field is relatively easy to compress...



Why Compute Flow?

TECHNISCHE UNIVERSITÄT DARMSTADT

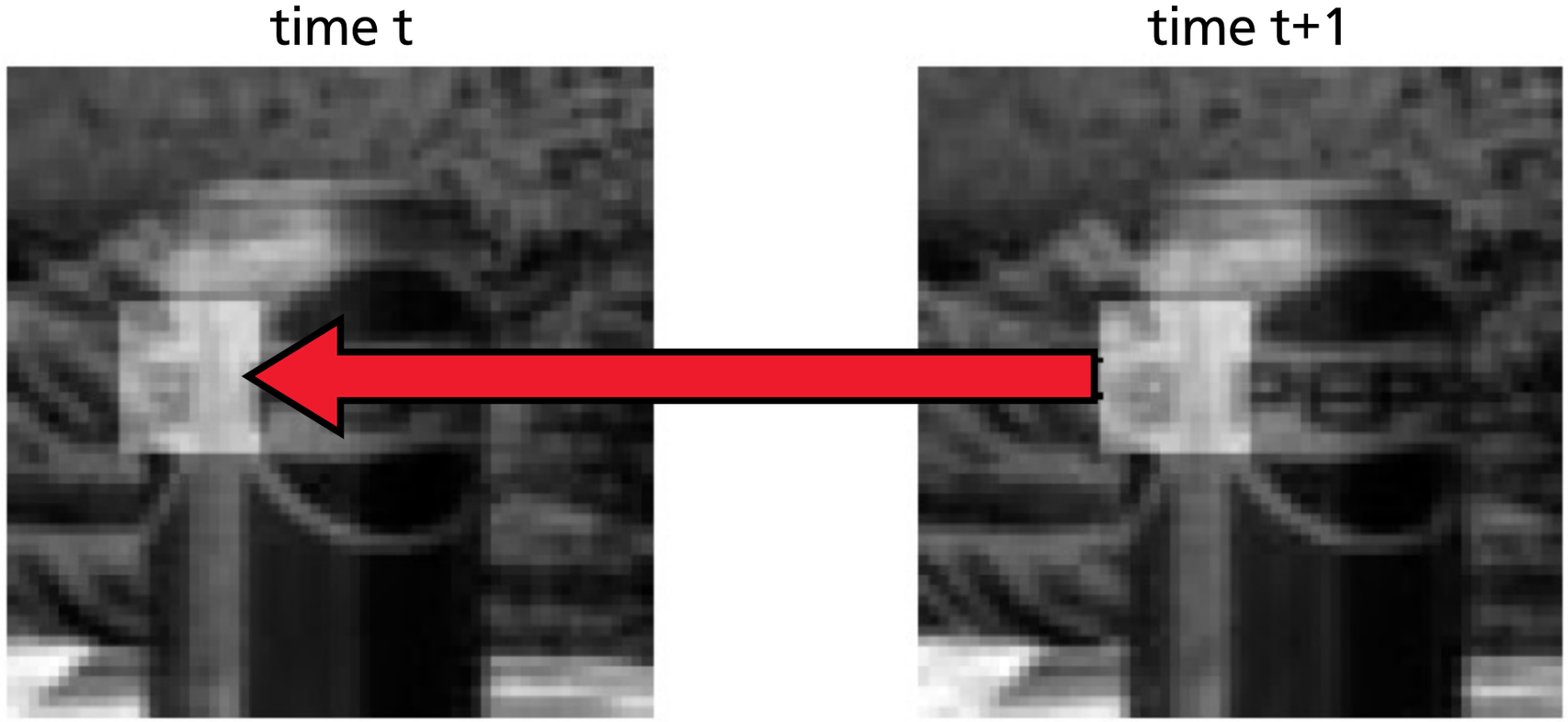
- Estimation of motion parameters in robotics.
- Reconstruction of the 3-D world from an image sequence (structure-from-motion).
- tracking of moving objects, e.g. human body motion.

How do we compute flow?

TECHNISCHE UNIVERSITÄT DARMSTADT

- We will make 3 important assumptions first.
- Then we will exploit those computationally.

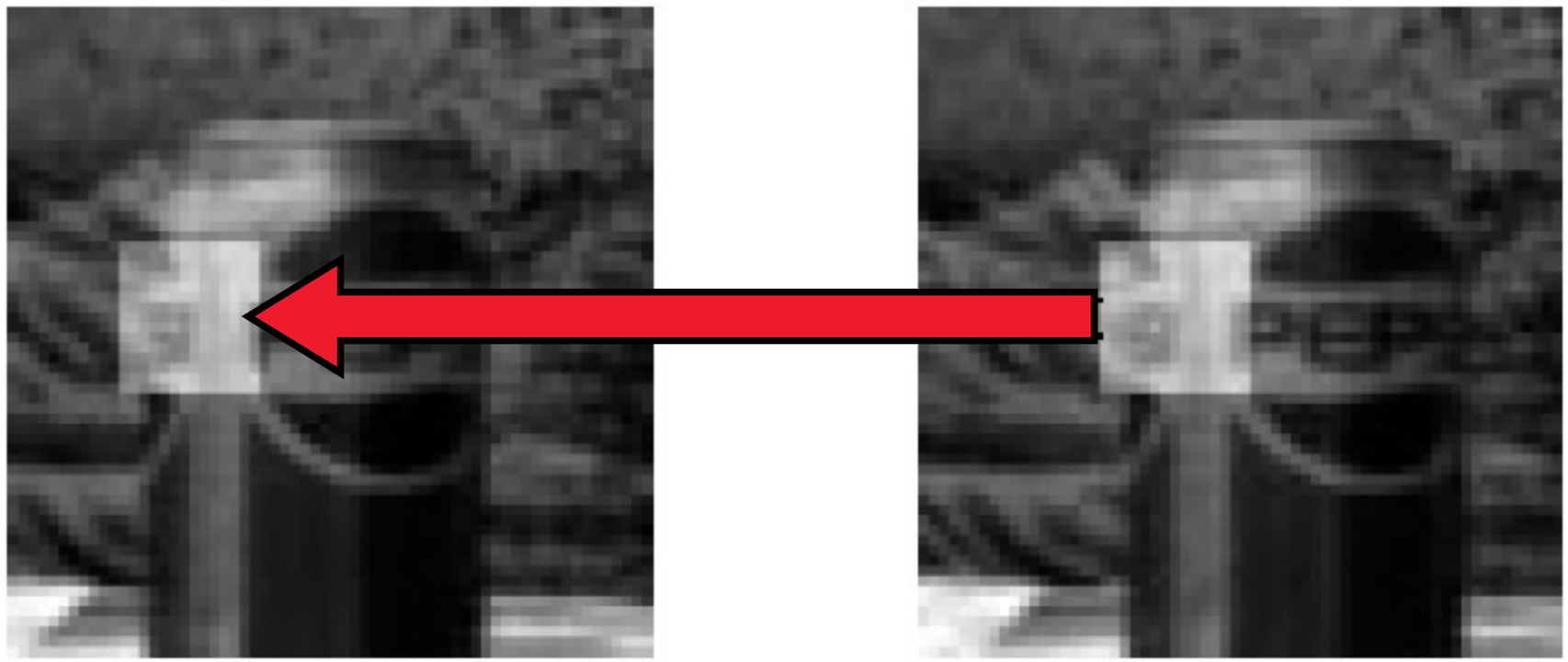
Brightness Constancy



Assumption 1:

Image measurements (e.g. brightness) in a small region remain the same although their location may change.

Brightness Constancy



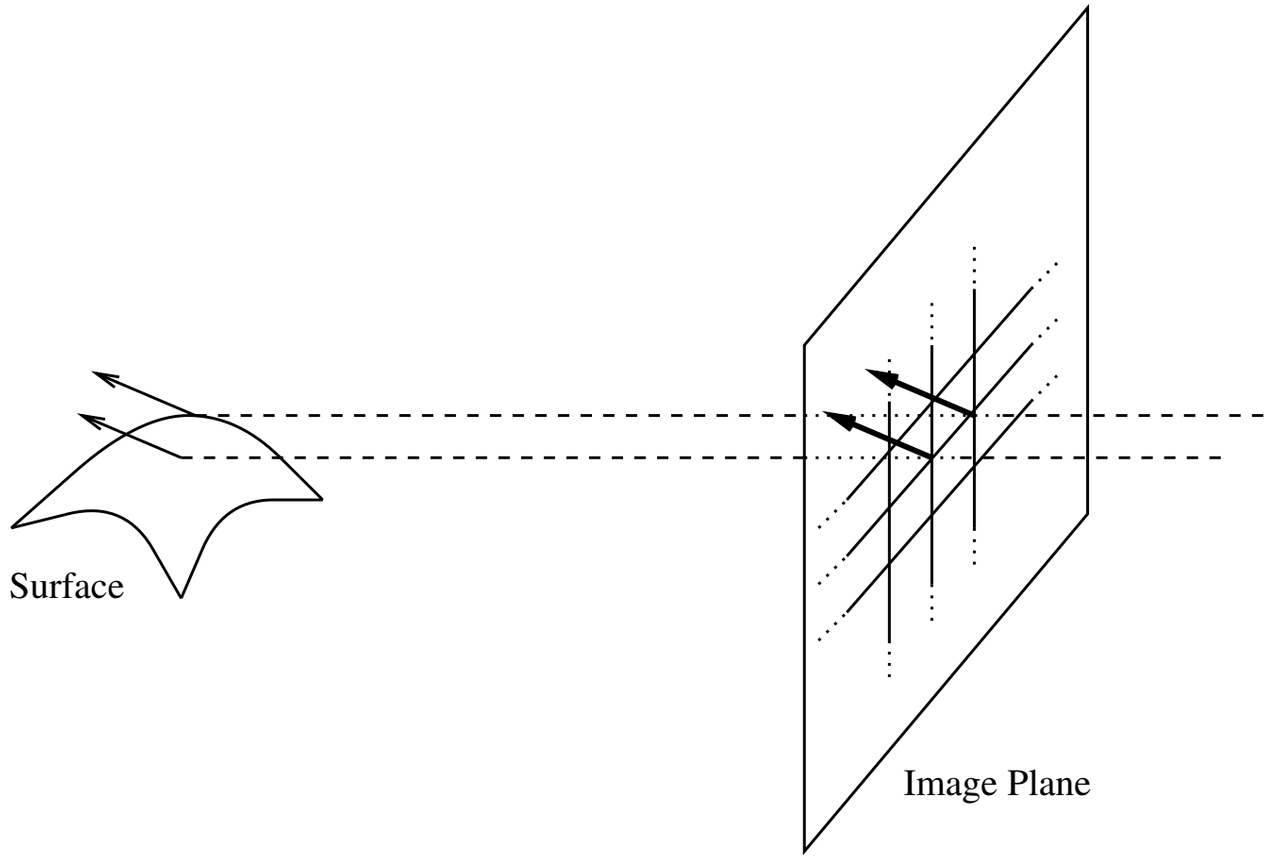
Assumption 1:

$$I(x + u(x, y), y + v(x, y), t + 1) = I(x, y, t)$$

shifted by horizontal flow

shifted by vertical flow

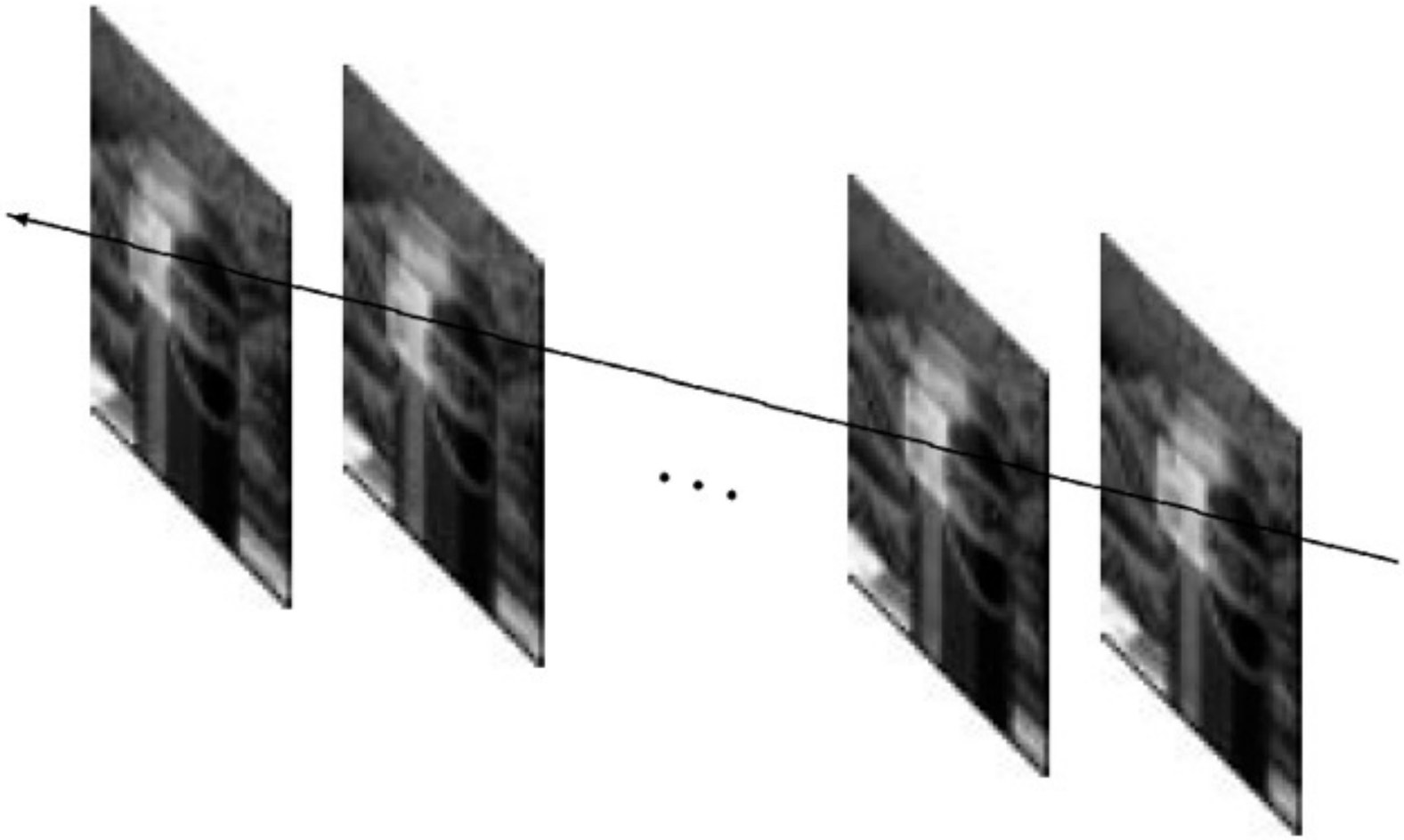
Spatial Coherence



Assumption 2:

- Neighboring points in the scene typically belong to the same surface and hence typically have similar 3D motions.
- Since they also project to nearby points in the image, we expect spatial coherence in the image flow.

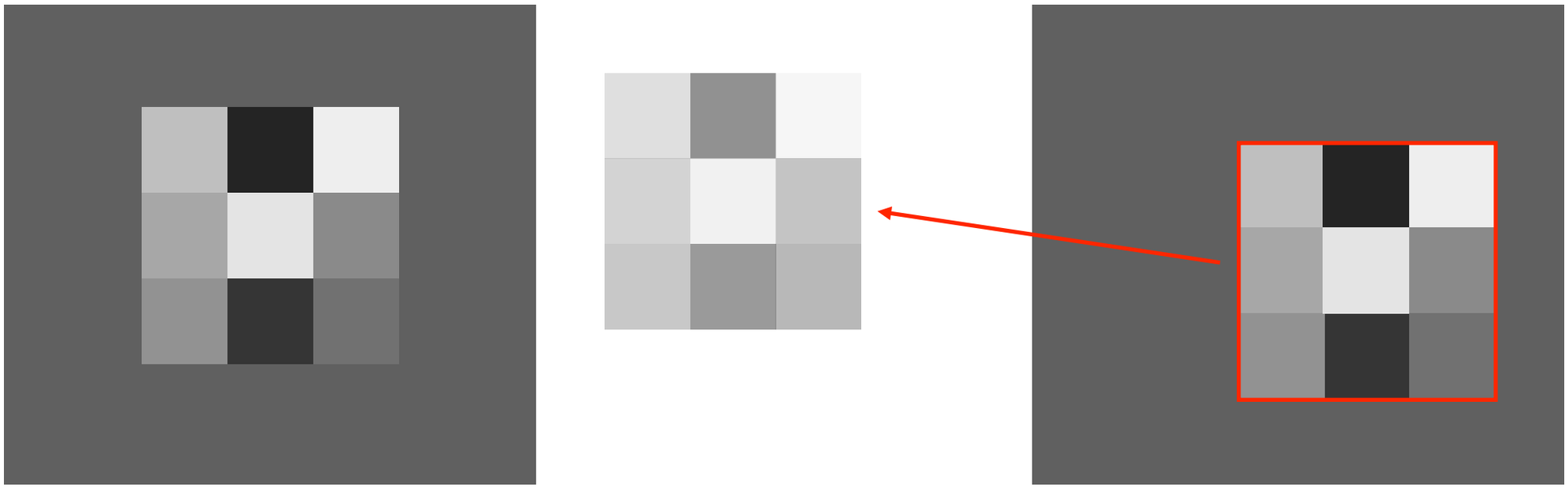
Temporal Persistence



Assumption 3:

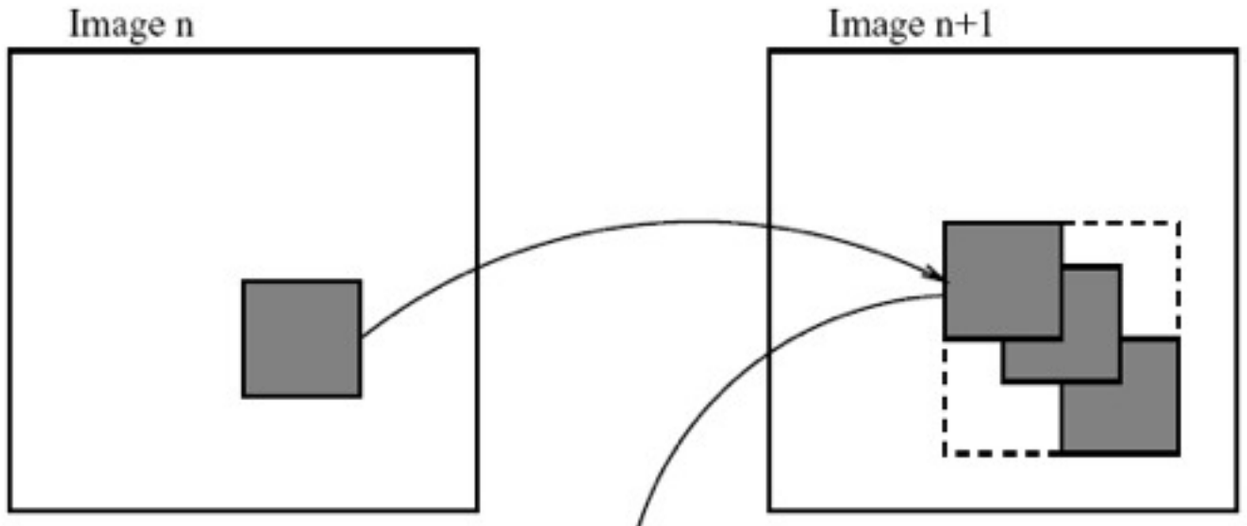
The image motion of a surface patch changes gradually over time.
(we do not use this cue for now)

Minimize Brightness Difference

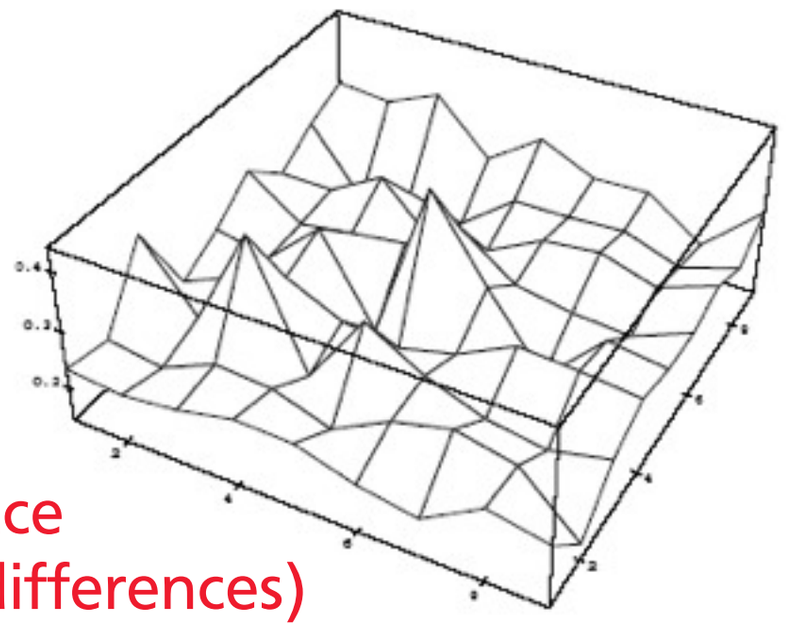


$$E_{SSD}(u, v) = \sum_{(x,y)\in R} \left(I(x + u, y + v, t + 1) - I(x, y, t)\right)^2$$

Sum of Squared Differences



$$E_{SSD}(u,v) = \sum_{(x,y)\in R} \big(I(x+u,y+v,t+1) - I(x,y,t)\big)^2$$



SSD surface
(sum of squared differences)

Simple Flow Estimation Algorithm

$$E_{SSD}(u,v) = \sum_{(x,y)\in R} \big(I(x+u, y+v, t+1) - I(x,y,t)\big)^2$$

- Simple way of flow estimation:
  - Discretize the space of possible motions.
  - For each pixel, (take its neighborhood region and) try all possible motions.
  - Take the one that minimizes the SSD.
- Problems with this approach:
  - Very inefficient.
  - The motions are discrete, which is usually not true in practice.
- So how can we optimize this nonlinear and non-convex function?

Can we approximate this?

$$E_{SSD}(u,v) = \sum_{(x,y)\in R} \left(I(x+u, y+v, t+1) - I(x,y,t) \right)^2$$

Let us look at the general case:

$$I(x+\Delta_x, y+\Delta_y, t+\Delta_t)$$

Taylor series approximation:

$$I(x,y,t) + \Delta_x \frac{\partial}{\partial x} I(x,y,t) + \Delta_y \frac{\partial}{\partial y} I(x,y,t) + \Delta_t \frac{\partial}{\partial t} I(x,y,t) + \epsilon(\Delta_x^2, \Delta_y^2, \Delta_t^2)$$

Approximation error

Can we approximate this?

$$E_{SSD}(u,v) = \sum_{(x,y)\in R} \big(I(x+u,y+v,t+1) - I(x,y,t)\big)^2$$

Assume small motion

Brightness varies smoothly

$$I(x,y,t) + u\frac{\partial}{\partial x}I(x,y,t) + v\frac{\partial}{\partial y}I(x,y,t) + \frac{\partial}{\partial t}I(x,y,t) + \epsilon(u,v,1)$$

SSD approximation:

$$\begin{aligned}
E_{SSD}(u,v) &\approx \sum_{(x,y)\in R} \big(I(x,y,t) + u\frac{\partial}{\partial x}I(x,y,t) + v\frac{\partial}{\partial y}I(x,y,t) + \frac{\partial}{\partial t}I(x,y,t) - I(x,y,t)\big)^2 \\
&= \sum_{(x,y)\in R} \big(u\frac{\partial}{\partial x}I(x,y,t) + v\frac{\partial}{\partial y}I(x,y,t) + \frac{\partial}{\partial t}I(x,y,t)\big)^2 \\
&= \sum_{(x,y)\in R} \big(u\cdot I_x(x,y,t) + v\cdot I_y(x,y,t) + I_t(x,y,t)\big)^2
\end{aligned}$$

Optical Flow Constraint Equation

$$E_{SSD}(u,v) \approx \sum_{(x,y)\in R} \left(u \cdot I_x(x,y,t) + v \cdot I_y(x,y,t) + I_t(x,y,t)\right)^2$$

- This approximated SSD objective is a convex function of the motion u and v.
- It becomes much easier to optimize.
  - We will see that shortly.
- But, this only holds for small motions!

Optical Flow Constraint Equation

$$E_{SSD}(u,v) \approx \sum_{(x,y)\in R} \left(u \cdot I_x(x,y,t) + v \cdot I_y(x,y,t) + I_t(x,y,t)\right)^2$$

- By minimizing this Taylor series approximation to the SSD, we are trying to enforce the so-called optical flow constraint equation (OFCE):

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

- This is also called the linearized brightness constancy constraint.

Notation

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

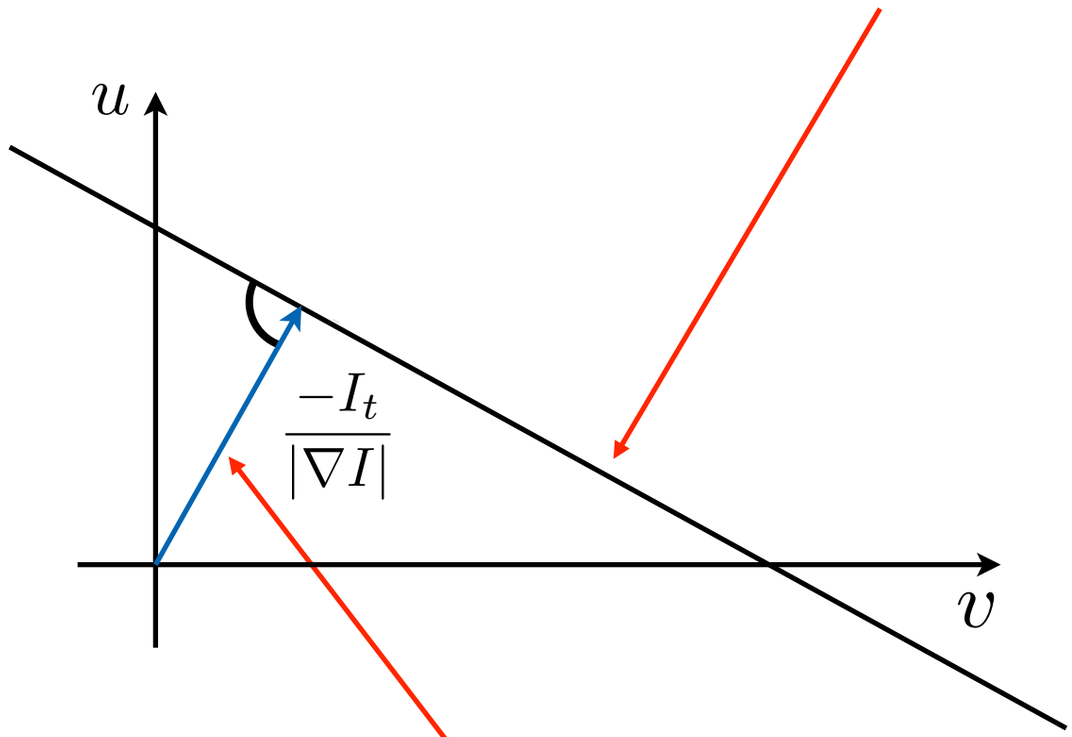
$$\nabla I^{\mathrm{T}} \mathbf{u} = -I_t$$

$$\mathbf{u} = \begin{pmatrix} u \\ v \end{pmatrix} \qquad \nabla I = \begin{pmatrix} I_x \\ I_y \end{pmatrix}$$

OFCE

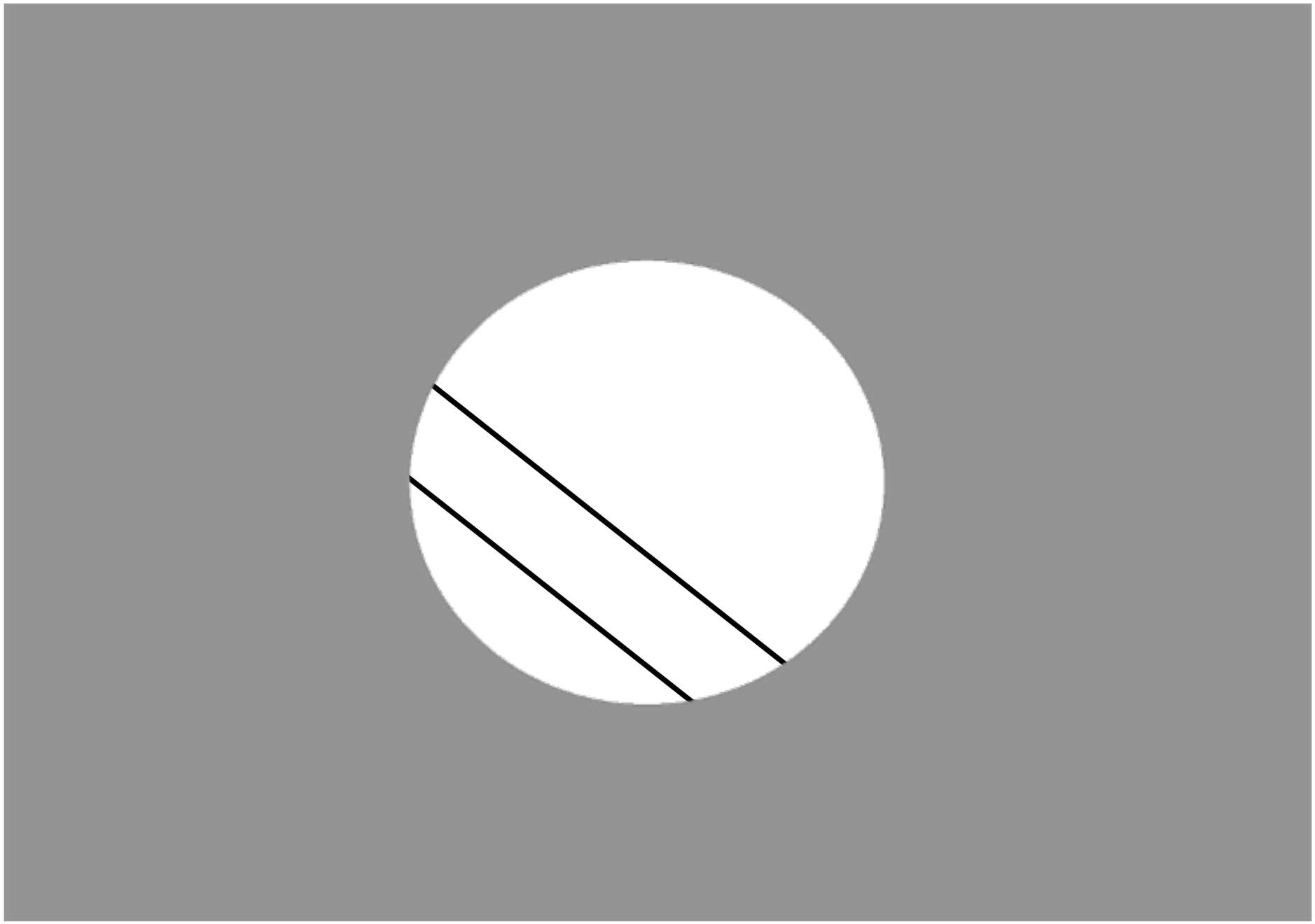
If we take a single image pixel, we get a constraint line:

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

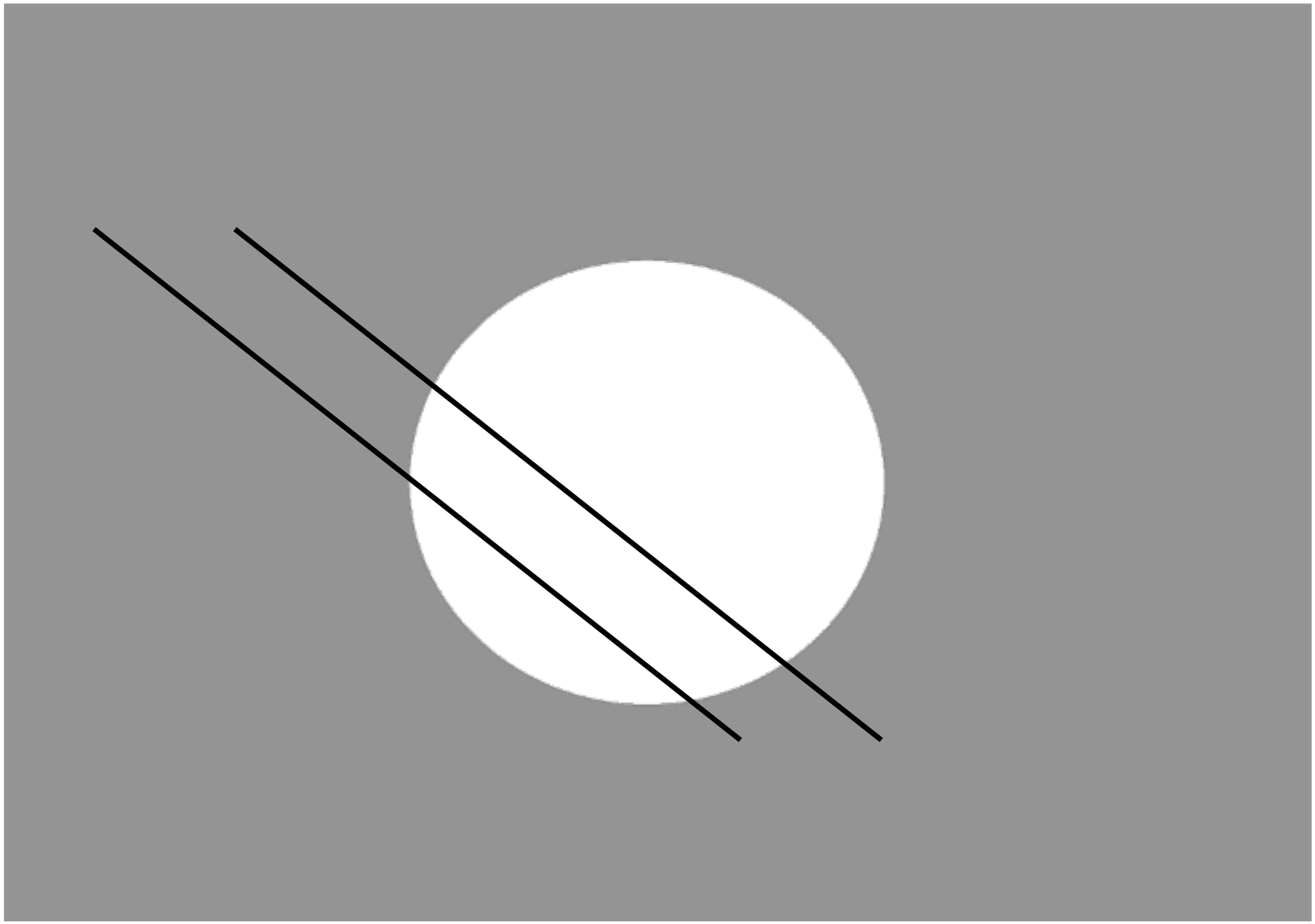


"Normal flow"

Aperture Problem

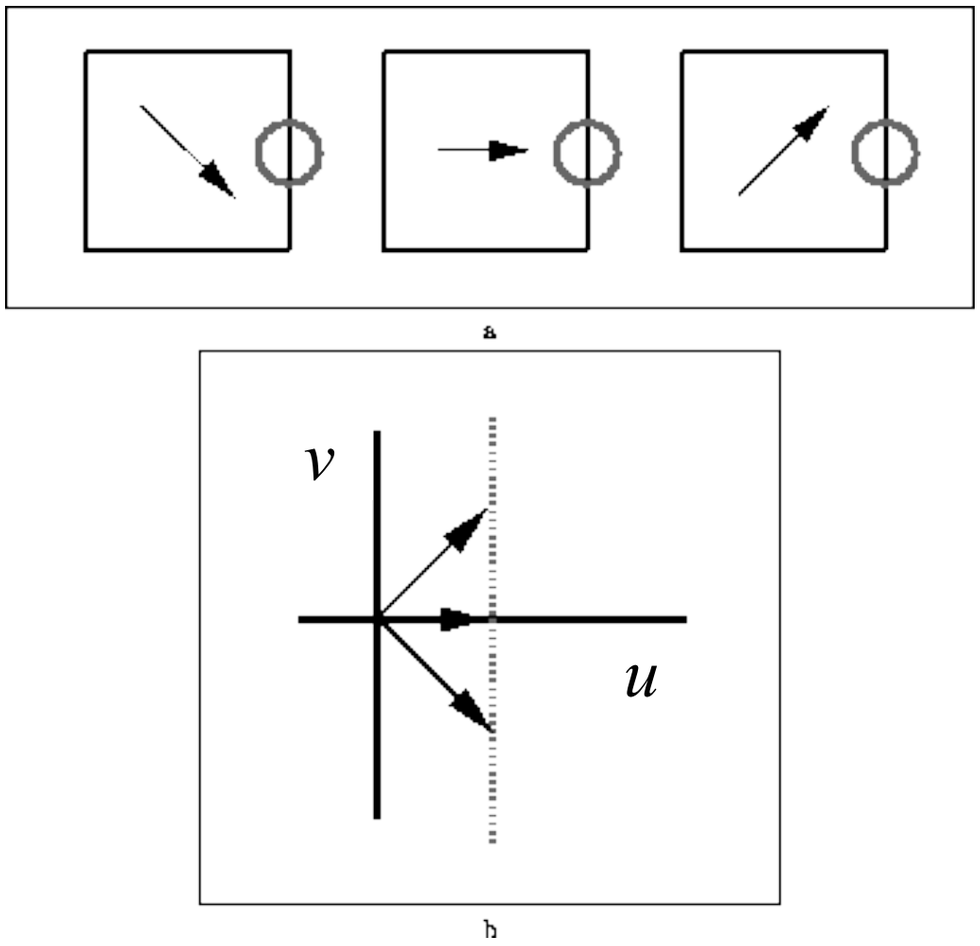


Aperture Problem



Aperture Problem

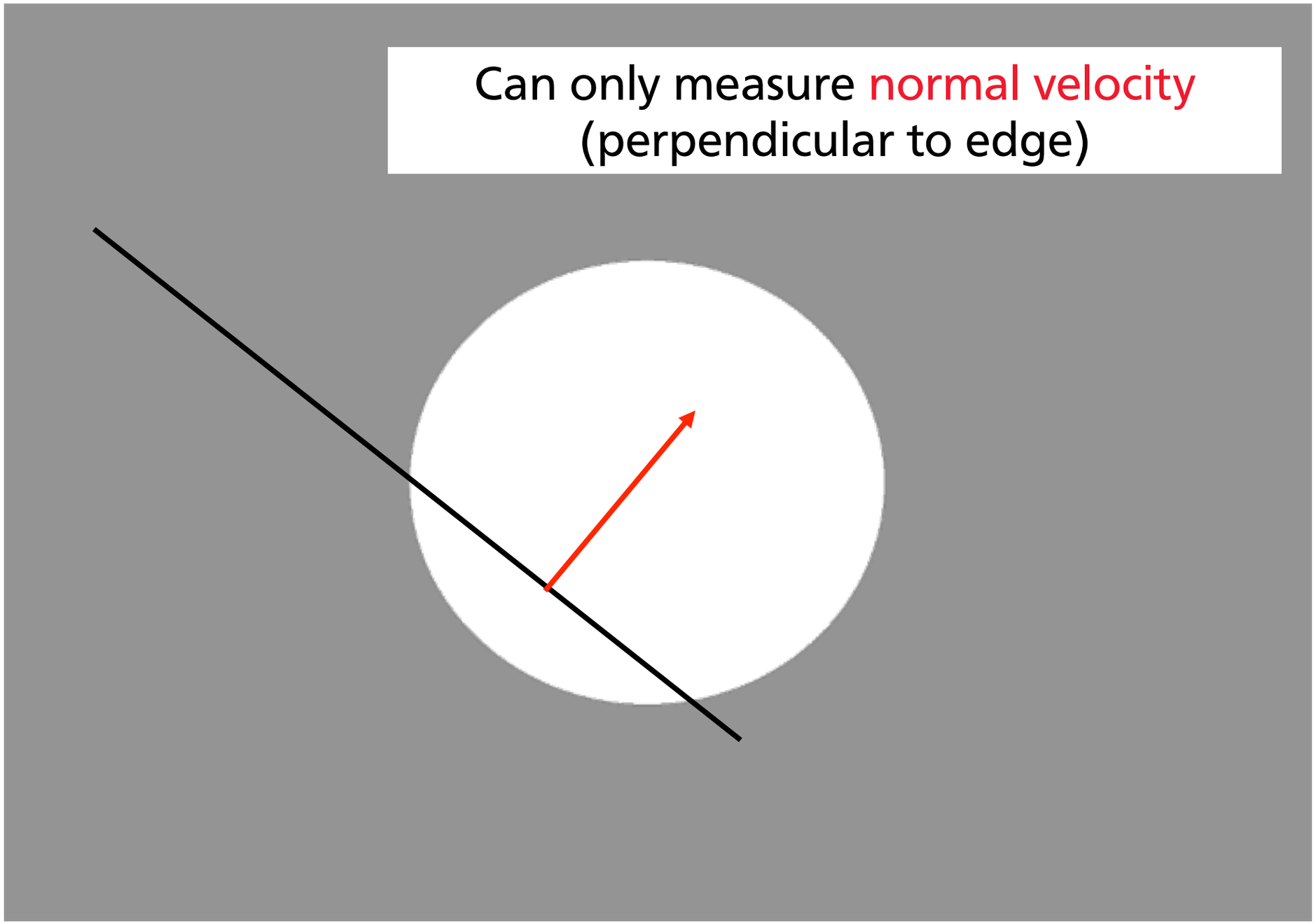
$$
\begin{aligned}
u \cdot I_x + v \cdot I_y &= -I_t \\
u \cdot I_x + 0 \cdot I_y &= -I_t \\
u \cdot I_x &= -I_t \\
u &= -\frac{I_t}{I_x}
\end{aligned}
$$



[Yair Weiss]

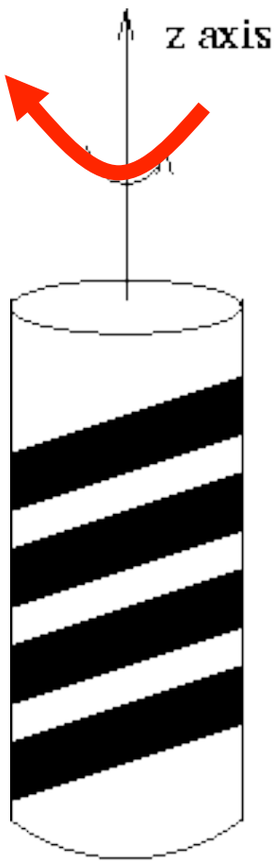
v could be anything!

Aperture Problem



Aperture Problem

Barber pole illusion:



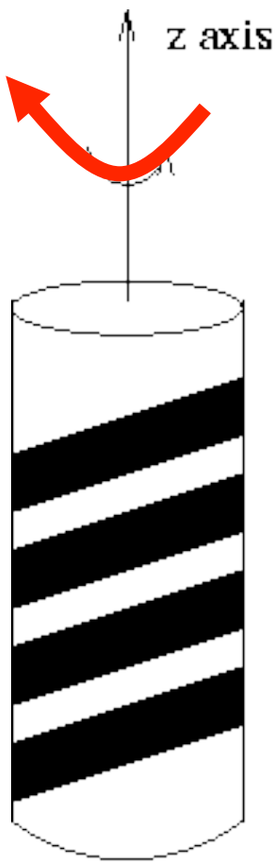
Barber's pole

What is the motion field?

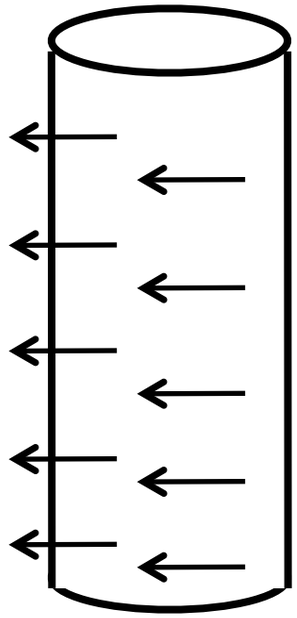
http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html

Aperture Problem

Barber pole illusion:



Barber's pole

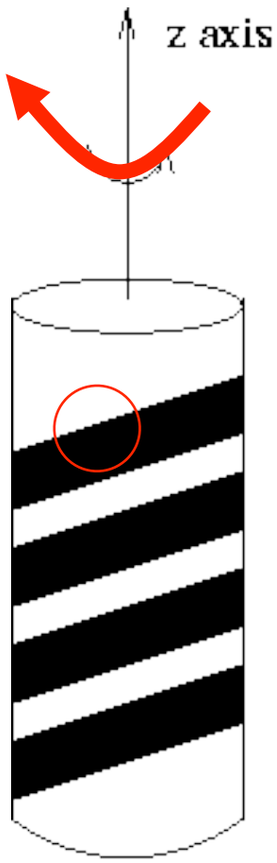


Motion field

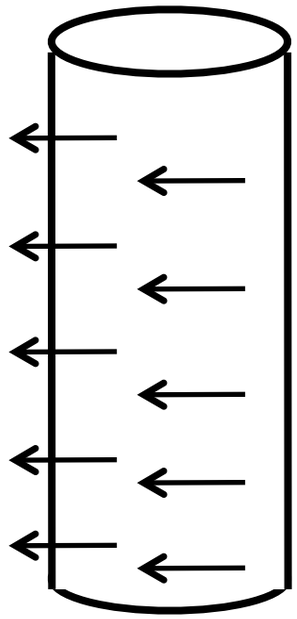
http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html

Aperture Problem

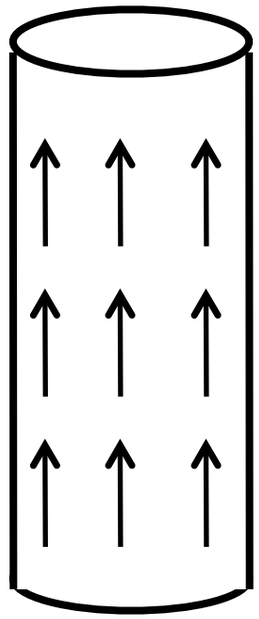
Barber pole illusion:



Barber's pole



Motion field

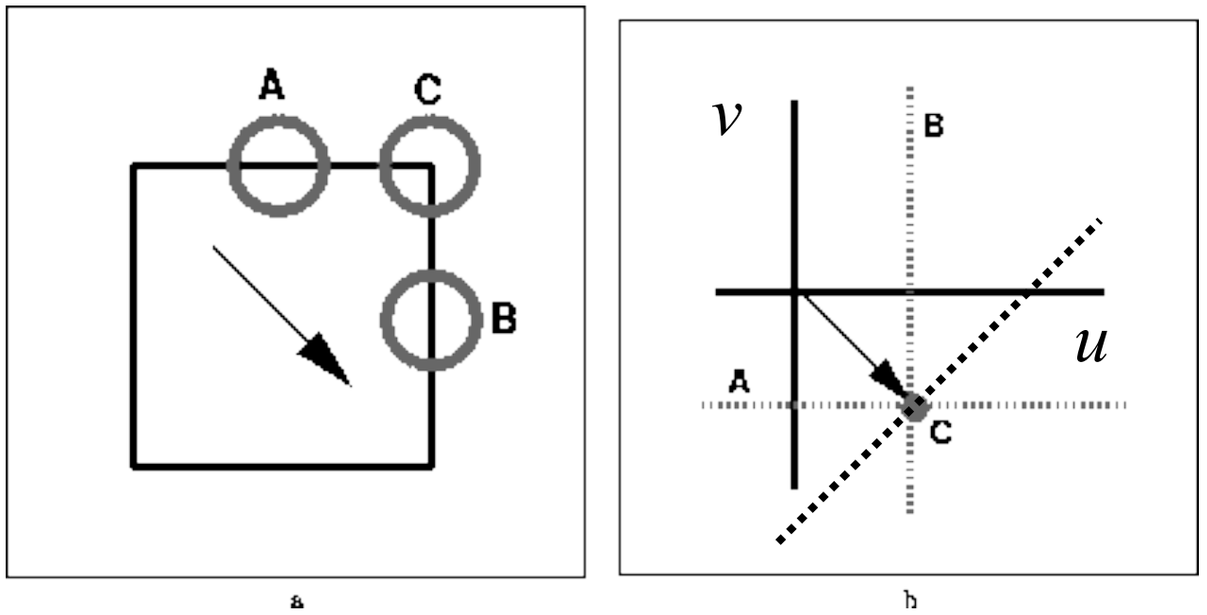


Optical flow

Why?

http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/OWENS/LECT12/node4.html

Multiple Constraints

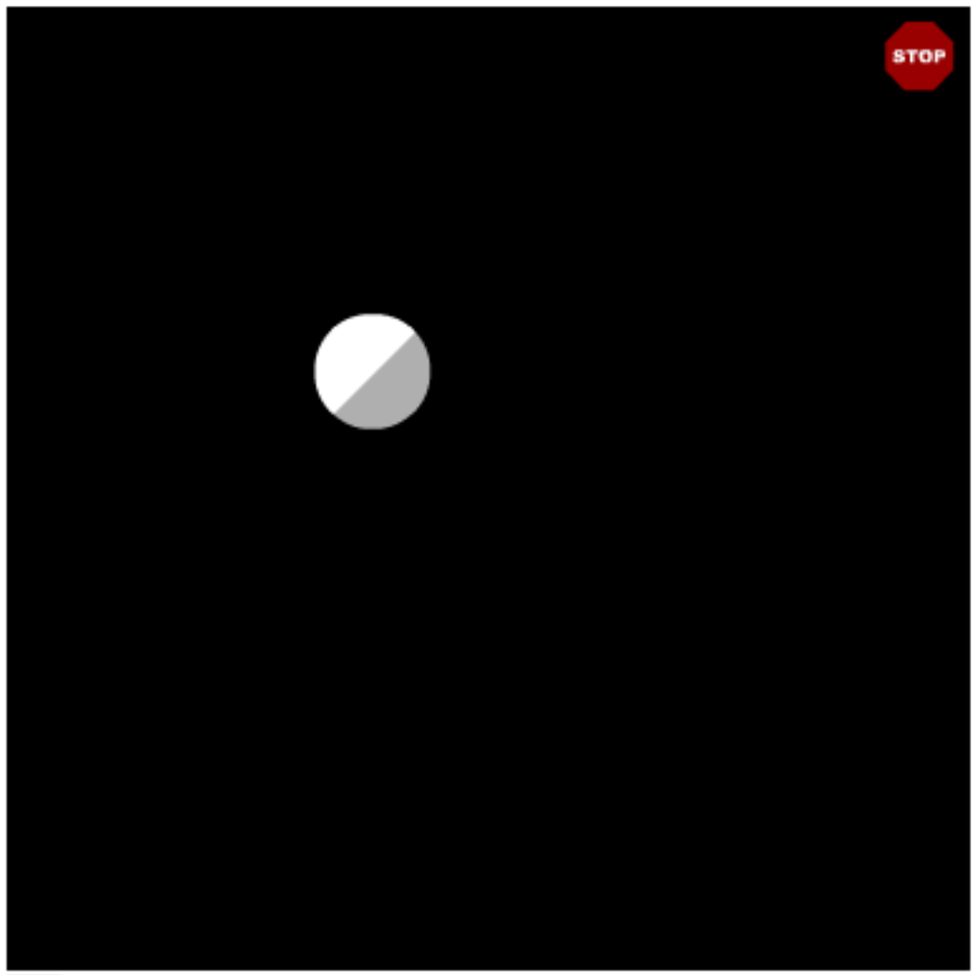


[Yair Weiss]

Combine multiple constraints to get an estimate of the velocity (not only the normal component).

Multiple Constraints

ONE SQUARE



a one edge
b whole square
c a corner
d four edges

Source: McDermott, J., Weiss, Y., Adelson, E.H. (2001). Beyond junctions: Nonlocal form contraints on motion interpretation. Perception, 30: 905-923.

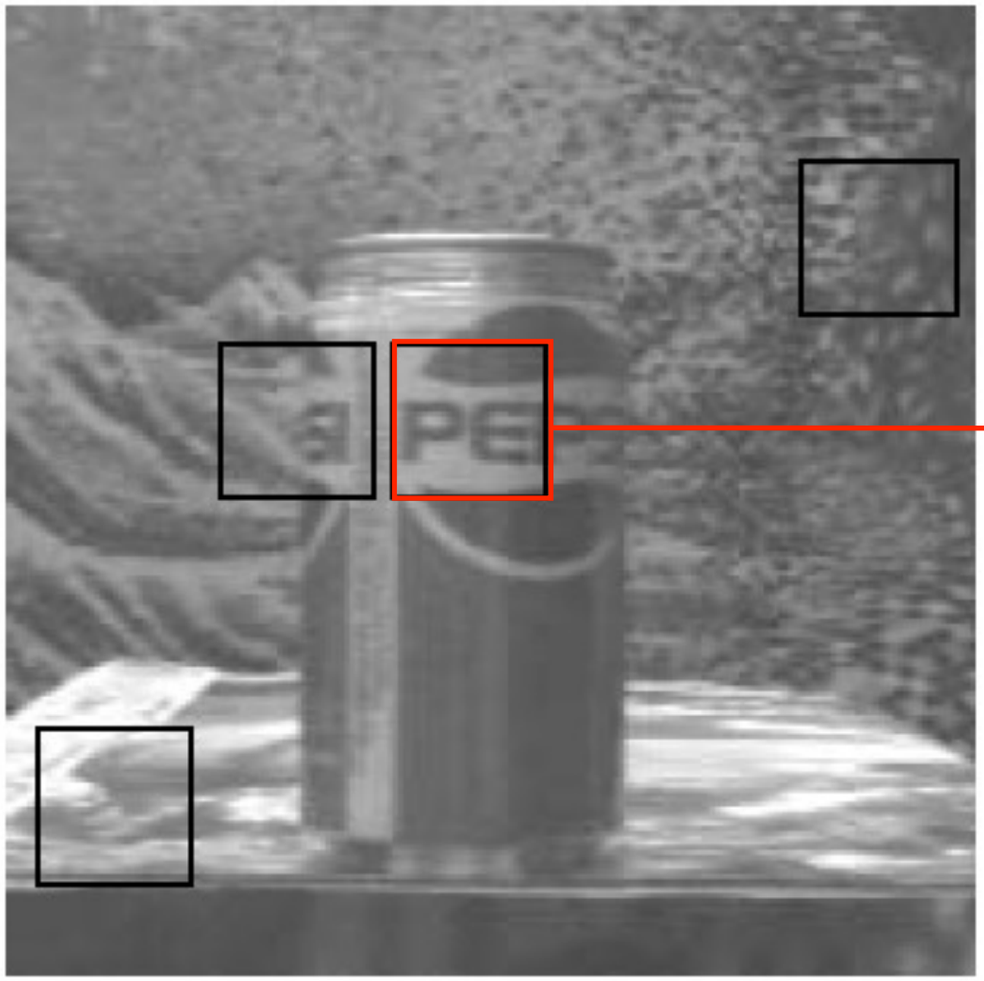
The motion of an edge seen through an aperture, as in (a) is inherently ambiguous. In this case the edge is physically moving upwards, as can be seen in (b), but the edge motion alone is consistent with many other possible motions, and in this case the edge in (a) generally appears to move diagonally. This ambiguity is known as the aperture problem. The upshot is that local motion measurements, such as those made by neurons in the early stages of the visual system, often do not specify the direction of motion of objects in the world.

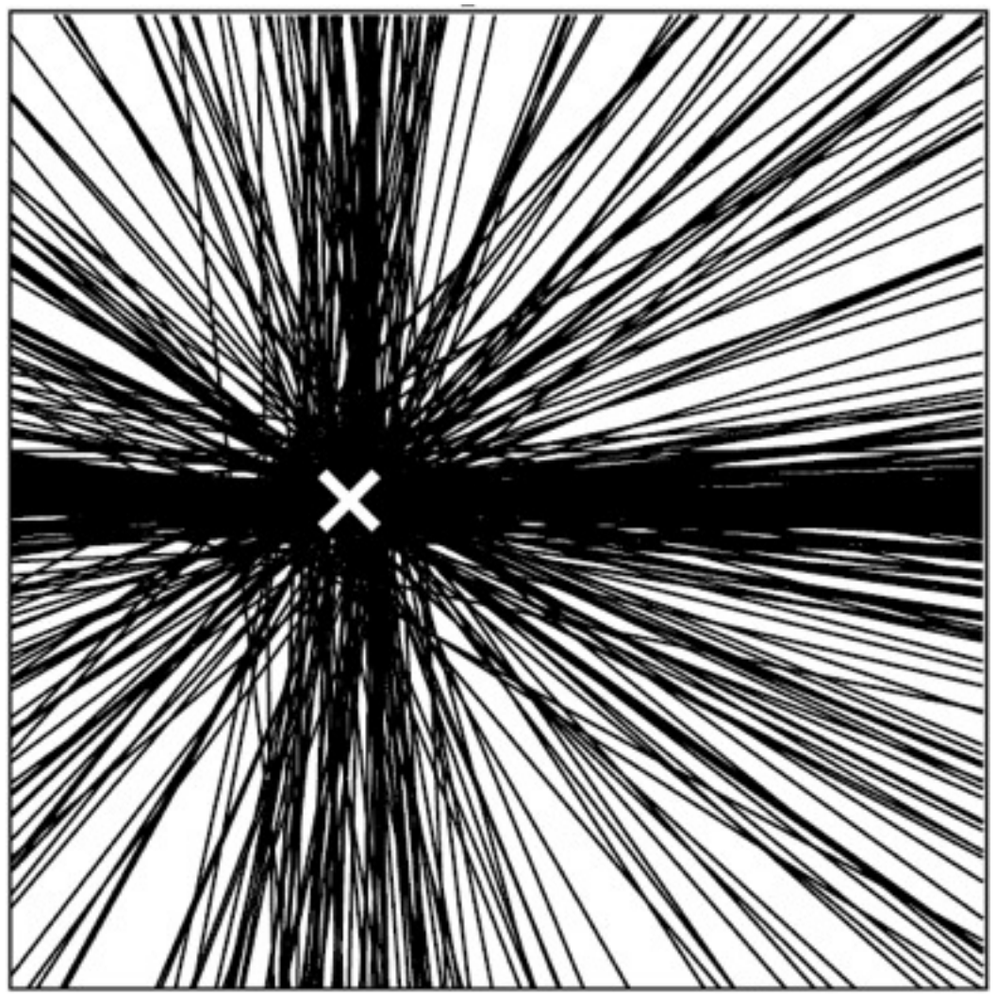
Two solutions are generally proposed to the aperture problem. The first involves finding 2D features, such as a corner of the square (c), whose motions are unambiguous. The second involves combining motion measurements made at different locations in space. As is evident in (d), when information from all four edges is available, the correct vertical motion can be extracted.

However, both of these strategies for solving the aperture problem run into further problems in real world scenes with multiple objects, as can be seen in the next demo.

http://web.mit.edu/persci/demos/Motion&Form/mini.html

Multiple Constraints





Area-Based Flow

- How do we combine multiple constraints?
  - Remember our assumptions.
  - We will assume spatial smoothness (coherence) of the flow.
- More specifically:
  - Assume that the flow is constant in a region.

$$E_{SSD}(u,v) \approx \sum_{(x,y)\in R} \left(u \cdot I_x(x,y,t) + v \cdot I_y(x,y,t) + I_t(x,y,t)\right)^2$$

  - This is what we have been doing (sliding window).
- But how do we solve for the motion?

Optimization

$$E_{SSD}(u,v) \approx \sum_{(x,y)\in R} \big(u \cdot I_x(x,y,t) + v \cdot I_y(x,y,t) + I_t(x,y,t)\big)^2$$

Differentiate with respect to u and v and set this to zero.

$$\frac{\partial}{\partial u} E_{SSD}(u,v) \approx 2 \sum_{(x,y)\in R} \big(u \cdot I_x(x,y,t) + v \cdot I_y(x,y,t) + I_t(x,y,t)\big) I_x(x,y,t) = 0$$

$$\frac{\partial}{\partial v} E_{SSD}(u,v) \approx 2 \sum_{(x,y)\in R} \big(u \cdot I_x(x,y,t) + v \cdot I_y(x,y,t) + I_t(x,y,t)\big) I_y(x,y,t) = 0$$

Optimization

$$\frac{\partial E_{SSD}}{\partial u} \approx 2\sum_R \big(u \cdot I_x + v \cdot I_y + I_t\big) I_x = 0$$

$$\frac{\partial E_{SSD}}{\partial v} \approx 2\sum_R \big(u \cdot I_x + v \cdot I_y + I_t\big) I_y = 0$$

Rearrange the terms and drop the constant:

$$\left[\sum_R I_x^2\right] u + \left[\sum_R I_x I_y\right] v = -\left[\sum_R I_x I_t\right]$$

$$\left[\sum_R I_x I_y\right] u + \left[\sum_R I_y^2\right] v = -\left[\sum_R I_y I_t\right]$$

Optimization

$$\left[\sum_R I_x^2\right] u + \left[\sum_R I_x I_y\right] v = -\left[\sum_R I_x I_t\right]$$

$$\left[\sum_R I_x I_y\right] u + \left[\sum_R I_y^2\right] v = -\left[\sum_R I_y I_t\right]$$

System of 2 equations in 2 unknowns:

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

Symmetric positive definite (invertible)

Structure tensor

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

Structure tensor

- The structure tensor is an important tool in vision.
  - The eigenvectors give the principal directions of the local image variation.
  - The eigenvalues indicate their strength.
- We've use it before to find interest points.

Optimization

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

- Rewrite using the abbreviations from before:

$$\left(\sum_R \nabla I \nabla I^{\mathrm{T}} \right) \mathbf{u} = -\sum_R I_t \nabla I$$

- Invert structure tensor to obtain flow:

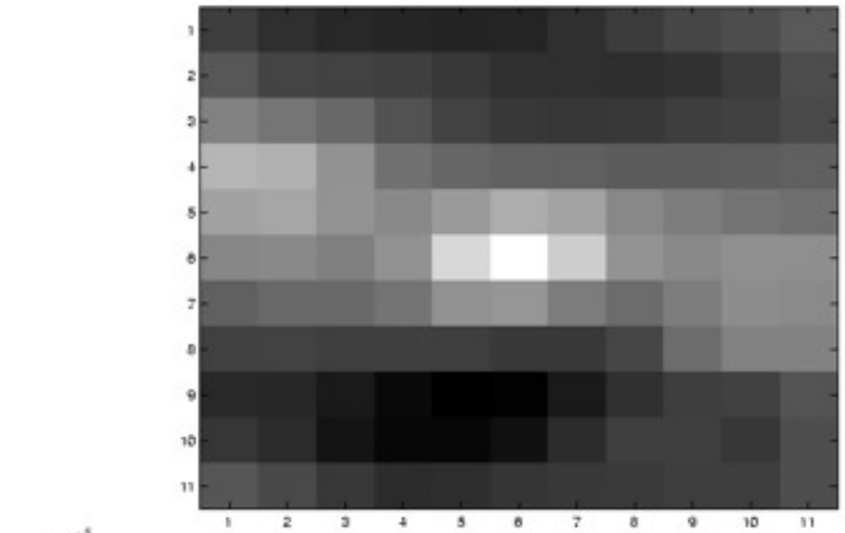
$$\mathbf{u} = -\left(\sum_R \nabla I \nabla I^{\mathrm{T}} \right)^{-1} \left(\sum_R I_t \nabla I \right)$$

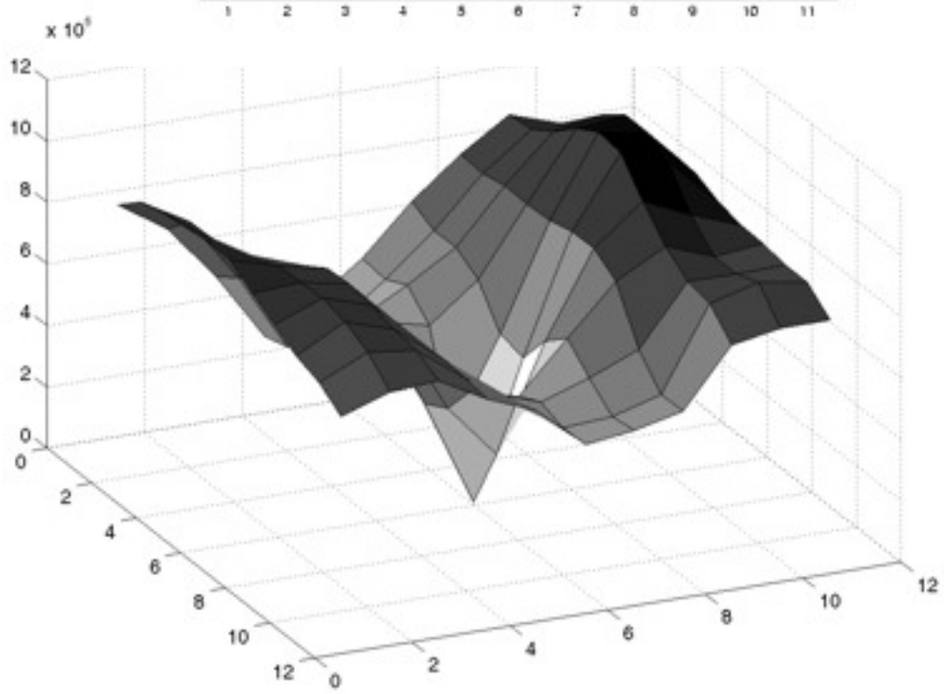
Solving for u

$$\mathbf{u} = -\left(\sum_R \nabla I \nabla I^{\mathrm{T}}\right)^{-1} \left(\sum_R I_t \nabla I\right)$$

- This is a classical flow technique:
  B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. IJCAI, pp. 674–679, 1981.
- What happens if:
  - The region is homogeneous?
  - There is a single edge?
  - There is a corner?

SSD Surface - Textured area

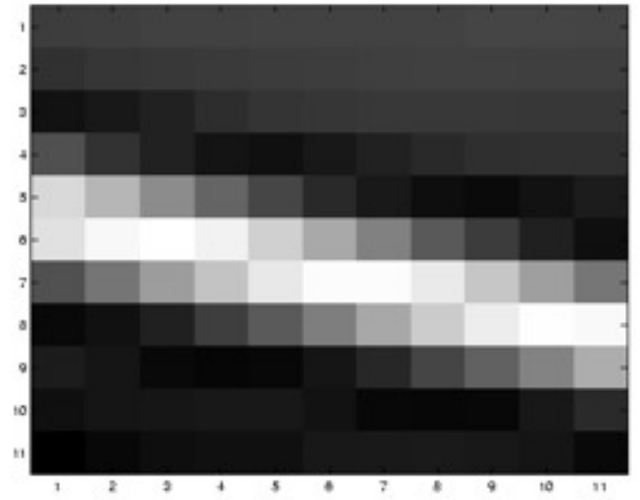


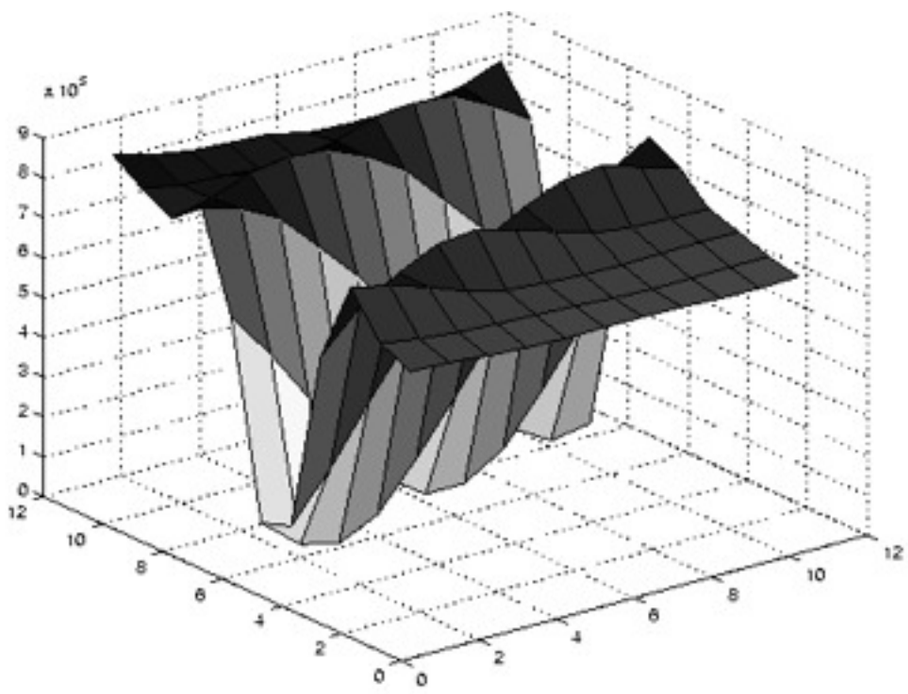


$$\begin{bmatrix} \sum_R I_x^2 & \sum_R I_x I_y \\ \sum_R I_x I_y & \sum_R I_y^2 \end{bmatrix}$$

Gradients in x and y.

SSD Surface - Single Edge

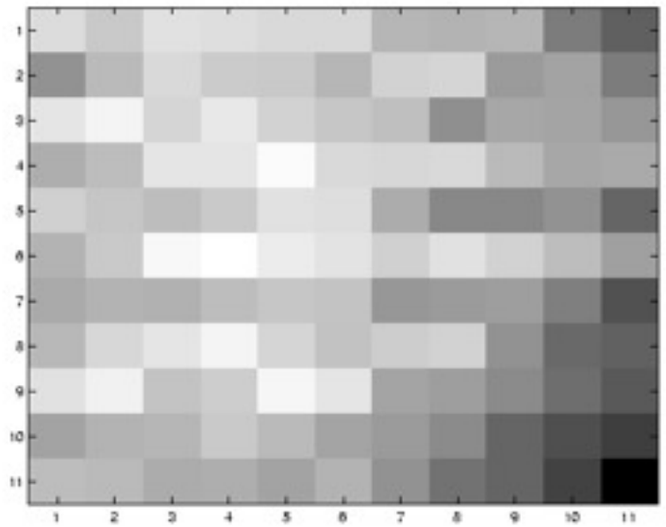


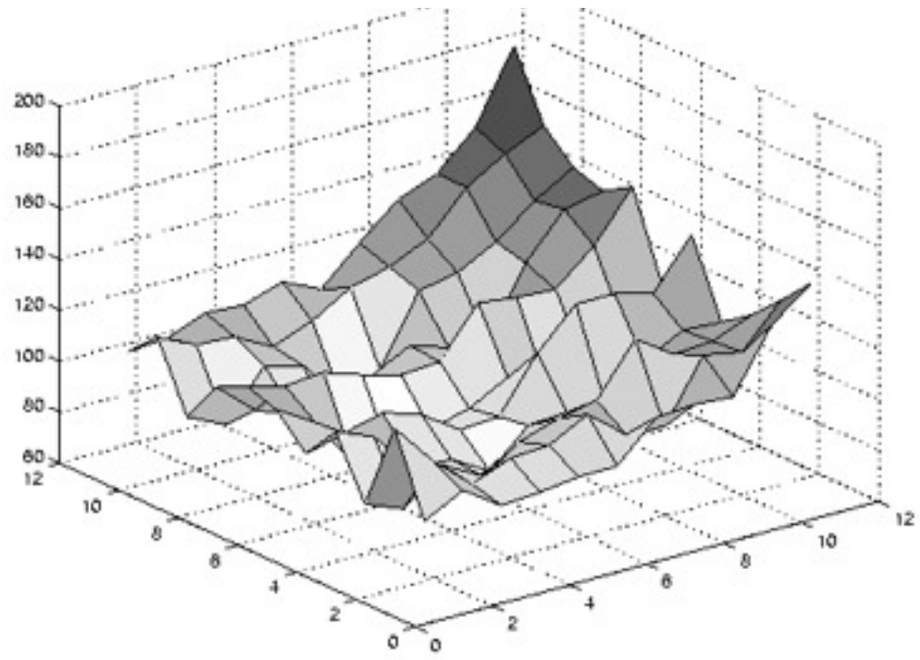


$$\begin{bmatrix} \sum_R I_x^2 & \sum_R I_x I_y \\ \sum_R I_x I_y & \sum_R I_y^2 \end{bmatrix}$$

Gradients oriented in one direction.

SSD Surface - Homogeneous area





$$\begin{bmatrix} \sum_R I_x^2 & \sum_R I_x I_y \\ \sum_R I_x I_y & \sum_R I_y^2 \end{bmatrix}$$

Weak gradients everywhere.

SSD Surface – Surface Boundary

?

$$\begin{bmatrix} \sum_R I_x^2 & \sum_R I_x I_y \\ \sum_R I_x I_y & \sum_R I_y^2 \end{bmatrix}$$

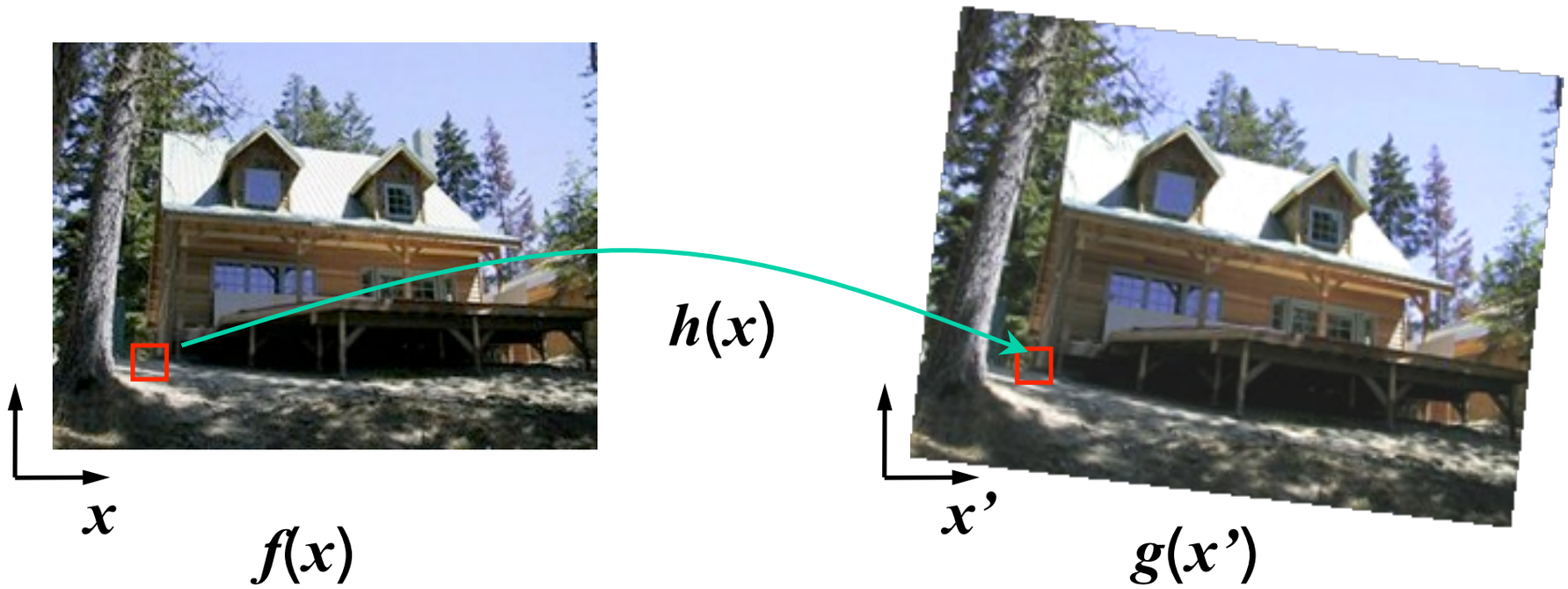
Gradients in x and y.

Image registration

- Remember the title of the Lucas-Kanade paper?
  - "An iterative image registration technique with an application to stereo vision."
  - What does that have to do with optical flow?
- We can use this to register (i.e. align) images:
  - Compute the flow with the entire images.
  - Shift the second image toward the first based on the flow.
  - Iterate until convergence.
- How do we "shift" an image?

Image Warping

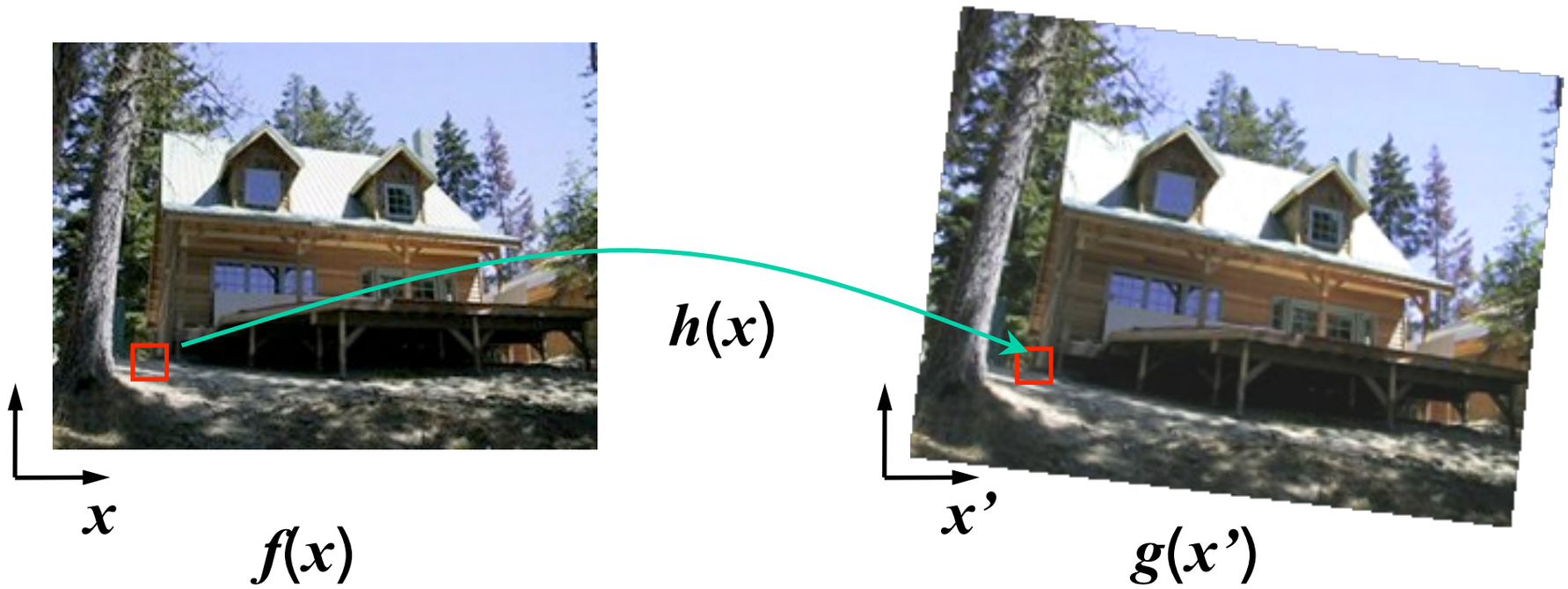
- Given a coordinate transform $\mathbf{x}' = h(\mathbf{x})$ and a source image $f(\mathbf{x})$, how do we compute a transformed image $g(\mathbf{x}') = f(g(\mathbf{x}))$?
- Note that we only need translations for now, but it's good to know the general case.



[Szeliski and Fleet]

Forward Warping

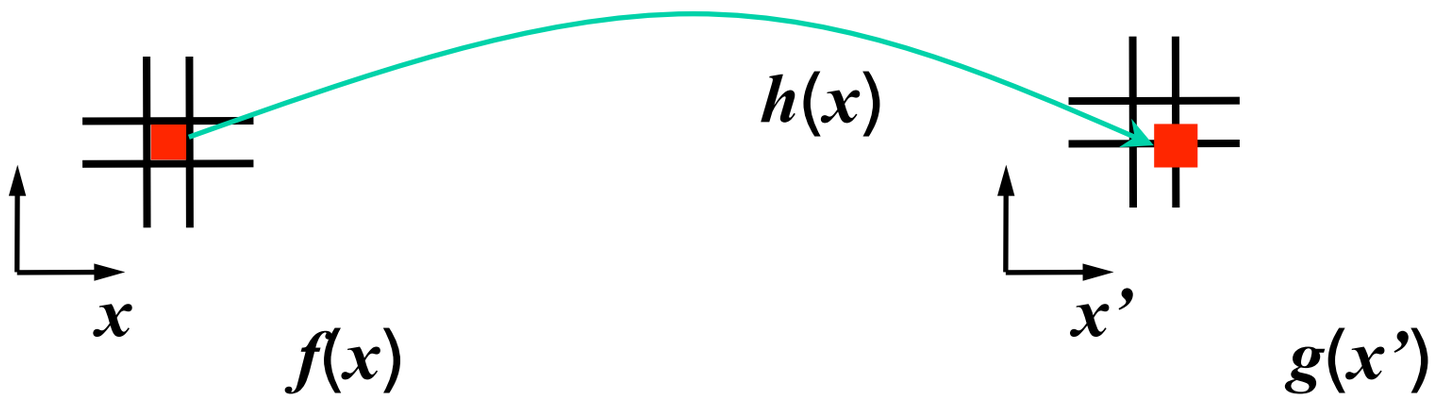
- Send each pixel $f(\mathbf{x})$ to its corresponding location $\mathbf{x}' = h(\mathbf{x})$ in $g(\mathbf{x}')$



[Szeliski and Fleet]

Forward Warping

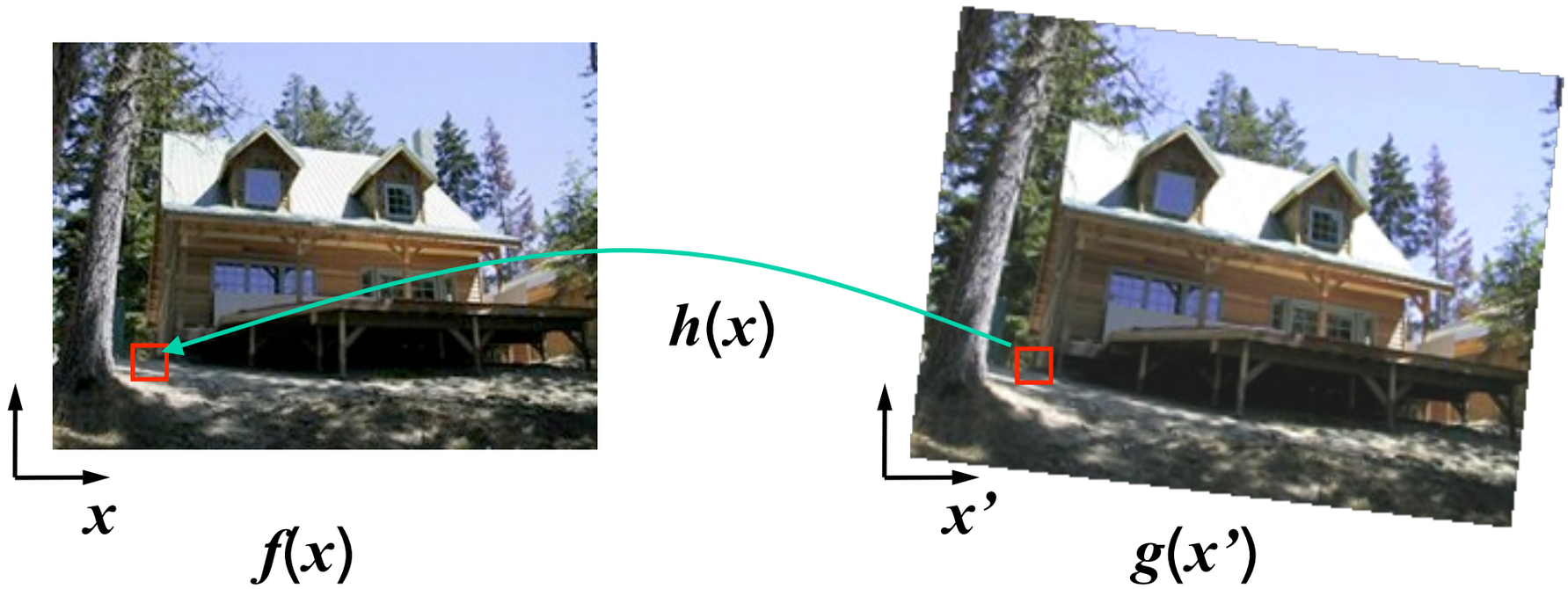
- What if pixel lands “between” two pixels?
- Answer: Add “contribution” to several pixels, normalize later (*splatting*).



[Szeliski and Fleet]

Inverse Warping

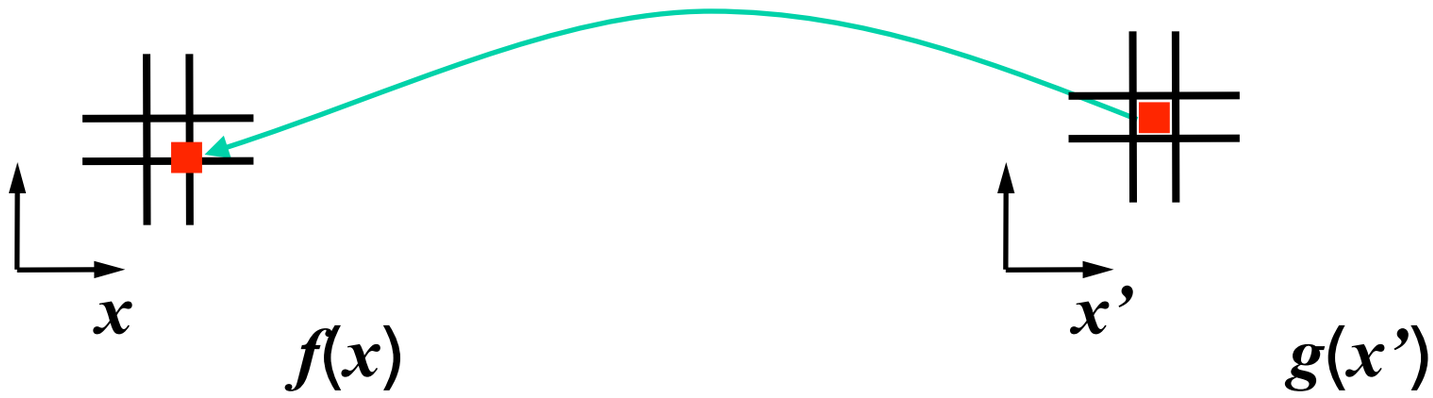
- Get each pixel $f(\mathbf{x})$ from its corresponding location $\mathbf{x}' = h(\mathbf{x})$ in $g(\mathbf{x}')$



[Szeliski and Fleet]

Inverse Warping

- What if pixel comes from “between” two pixels?
- Answer: Resample pixel value from interpolated source image



[Szeliski and Fleet]

Interpolation

- Possible interpolation filters:
  - nearest neighbor
  - bilinear
  - bicubic (interpolating)

- Needed to prevent “jaggies”.
- When iteratively warping, always warp the original image

[Szeliski and Fleet]

Warping

Example warps:

translation

rotation

aspect

affine

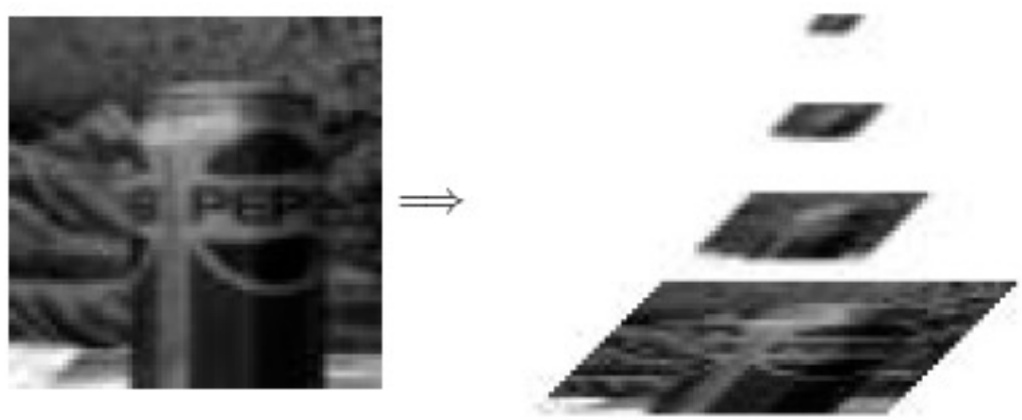
perspective

cylindrical

[Szeliski and Fleet]

Image registration revisited

- How do we “shift” an image?
  - We use image warping to shift the image.
- But there is still one problem:
  - The LK-model only holds for small motions.
- Solution: Coarse-to-fine estimation
  - Build a Gaussian pyramid
  - Start with the lowest resolution (motion is small!)
  - Use this motion to pre-warp the next finer scale
  - Only compute motion increments (small!)

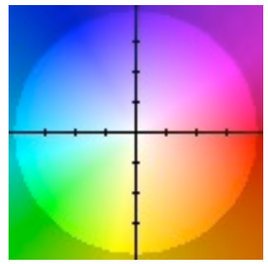


Dense LK Flow

- What if we now want to estimate dense flow?
  - We can just take the region R to be a small region around every pixel and compute a flow vector for every pixel.
- But we face the same problems as before:
  - The LK method only works for small motions.
- Two workarounds (use both):
  - Iterative estimation.
  - Coarse-to-fine estimation.

Iterative Estimation

- Take image pair and compute LK flow (21x21 window):



Frame 1

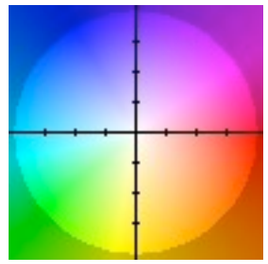
Frame 2

Frame 2 (warped)

Frame 1

Iterative Estimation

- Backward warp the second image toward the first:



Iterative Estimation

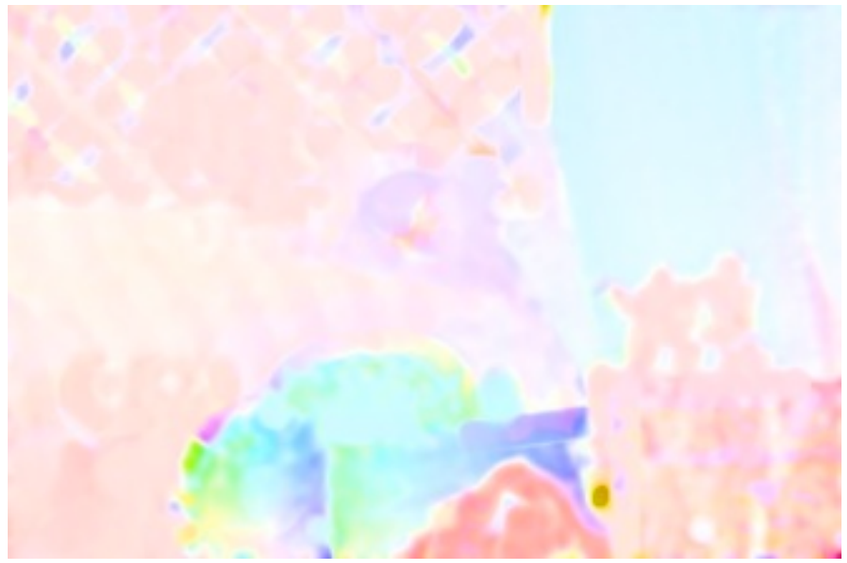
- Estimate incremental flow from warped image pair:

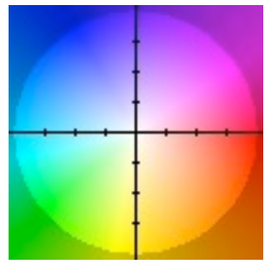
Iterative Estimation

- Add incremental flow to previous estimate:

Iterative Estimation

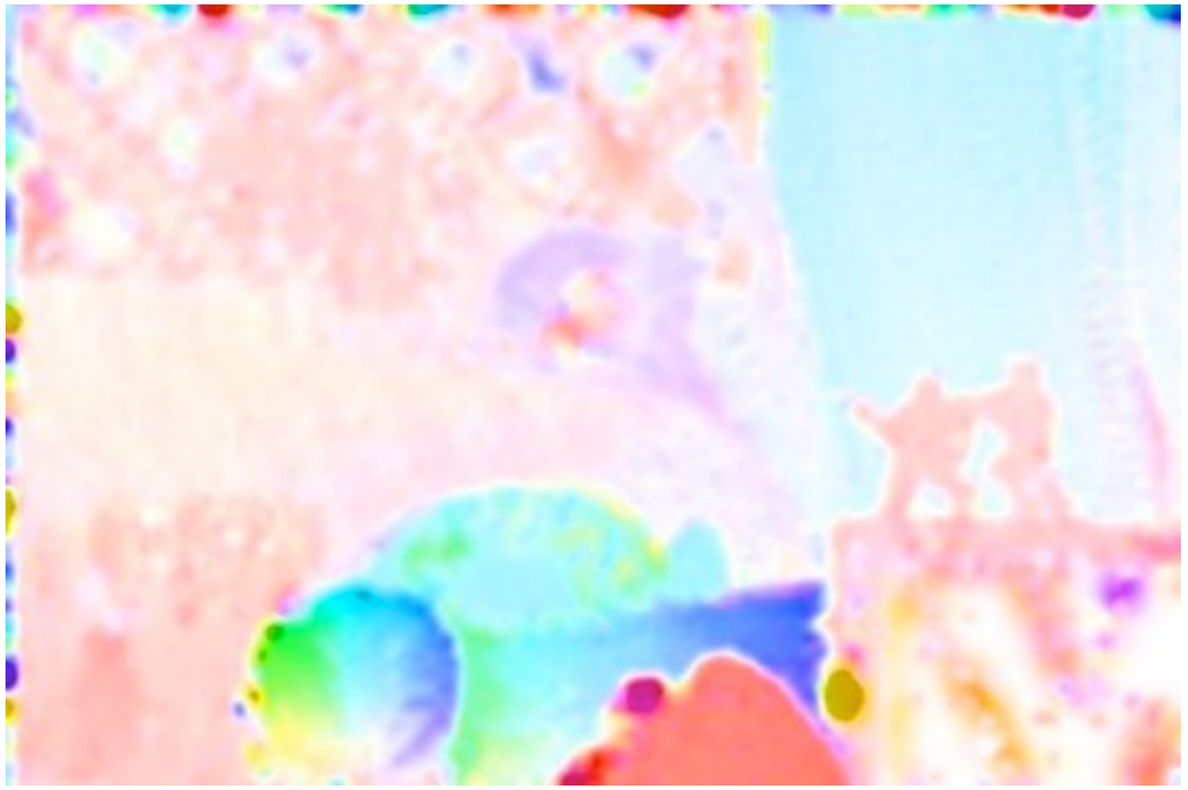
- Warp again and so on... until convergence.





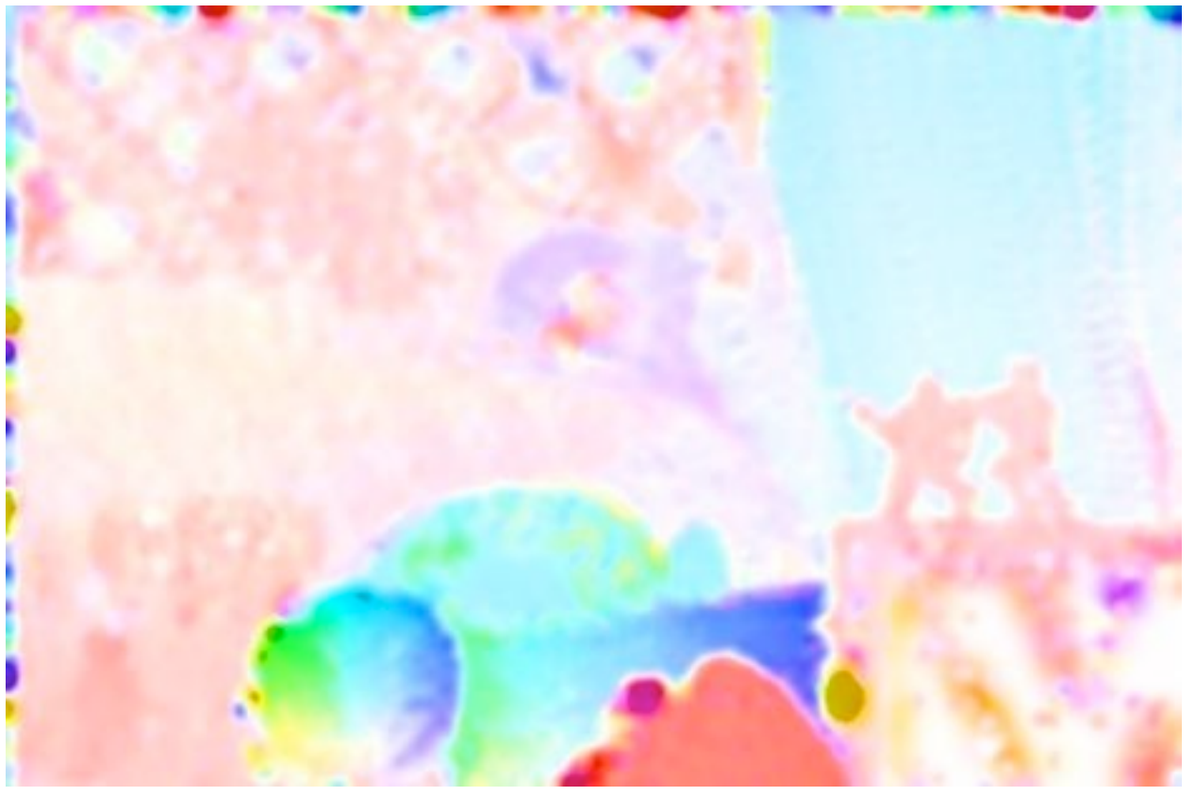
Coarse-to-fine Estimation

- Of coarse, we also apply our previous trick and use a Gaussian pyramid:
  - Initialize with the flow from a coarser level.
- If we do this on the previous image pair, we get this:

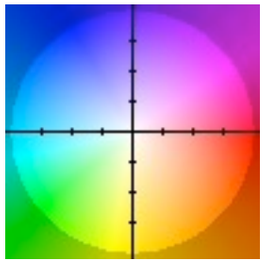


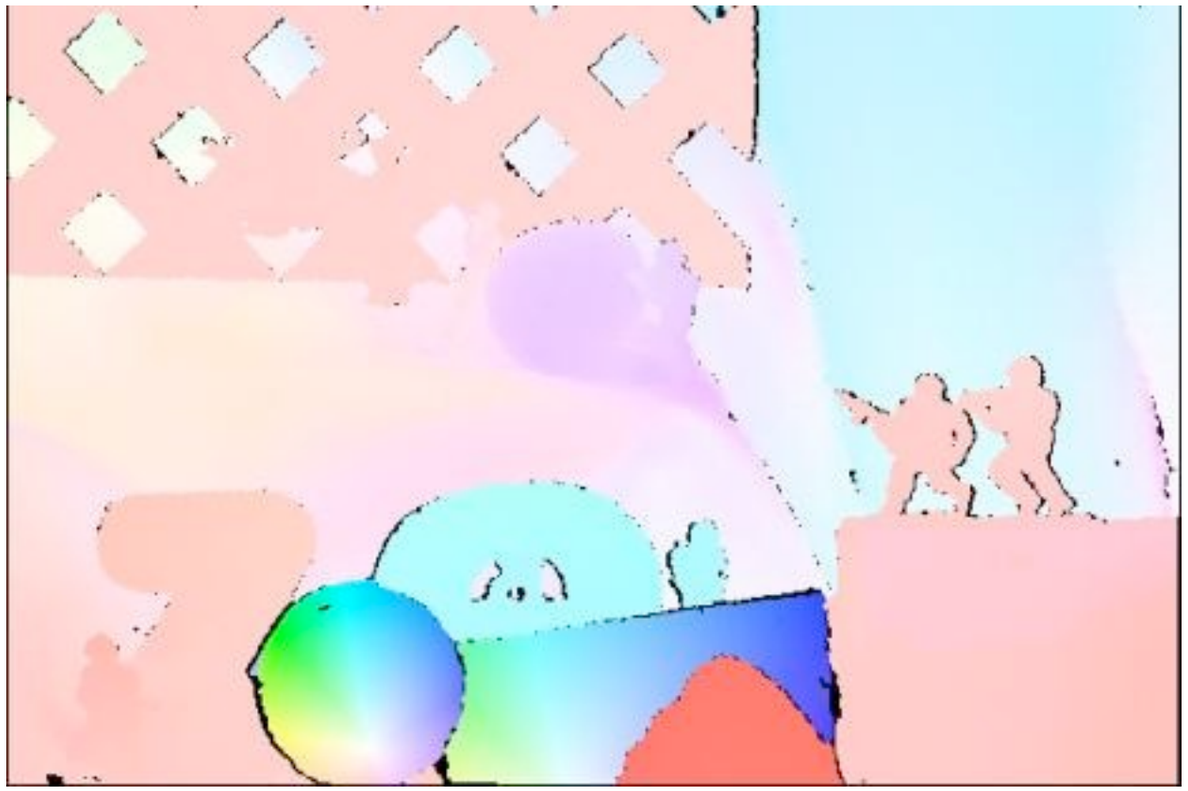
Is that a good result?

- Let's see:
  - Maybe not...



Pyramid LK method

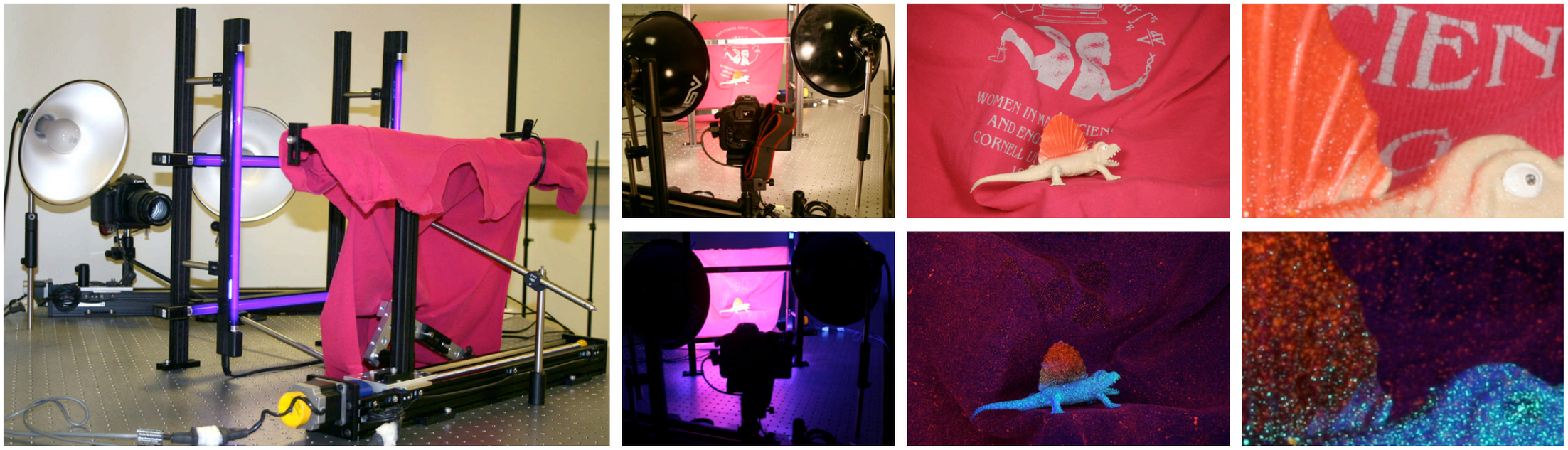




Ground truth

Aside: How to get the ground truth?

- Until recently, there were only very few image sequences with ground truth flow available.
  - All of them were synthetic and furthermore very simple.
- Middlebury optical flow benchmark [Baker et al., 07]:
  - Contains a number of complex synthetic and real sequences with ground truth.
  - Real sequences are captured using normal + UV light:



What is the problem?

- The window is too big!
  - All the discontinuities in the flow are smoothed over.
  - But discontinuities do exist, e.g., at motion boundaries.
- But: The window is also too small!
  - In some areas, the flow is really bad, because there is not enough image information in the window.
  - This happens in areas with little texture.
- LK is a local optical flow method.
  - Global flow methods to the rescue (CV II)