Univerza v Ljubljani

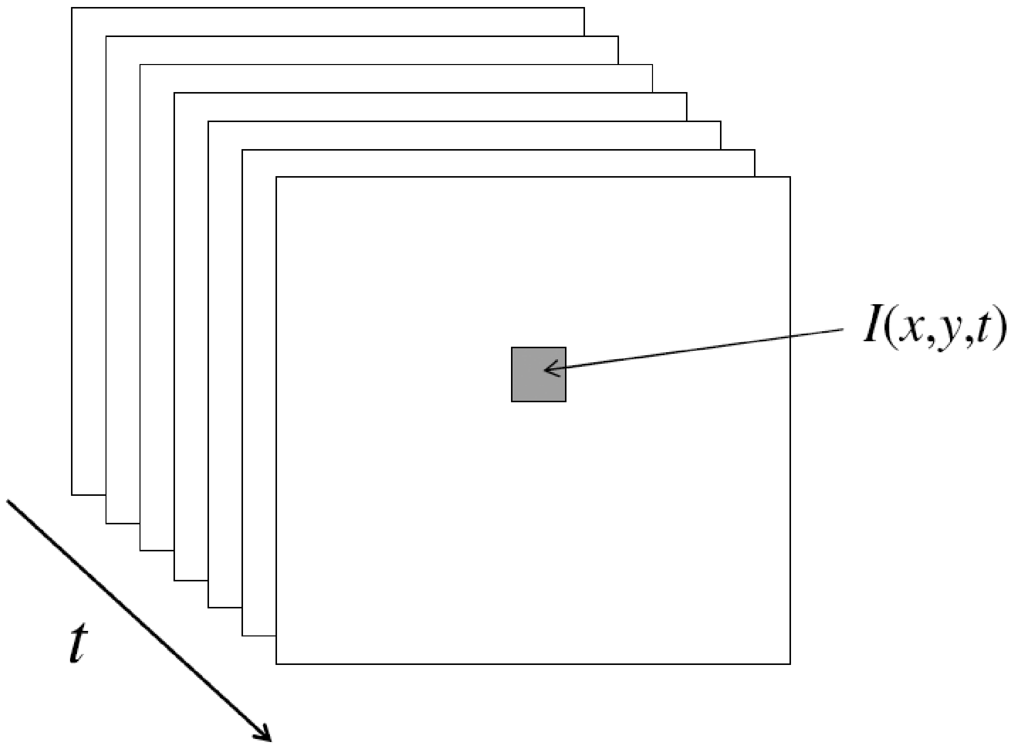
Advanced CV methods
Optical flow 1

Matej Kristan

Laboratorij za Umetne Vizualne Spoznavne Sisteme,
Fakulteta za računalništvo in informatiko,
Univerza v Ljubljani

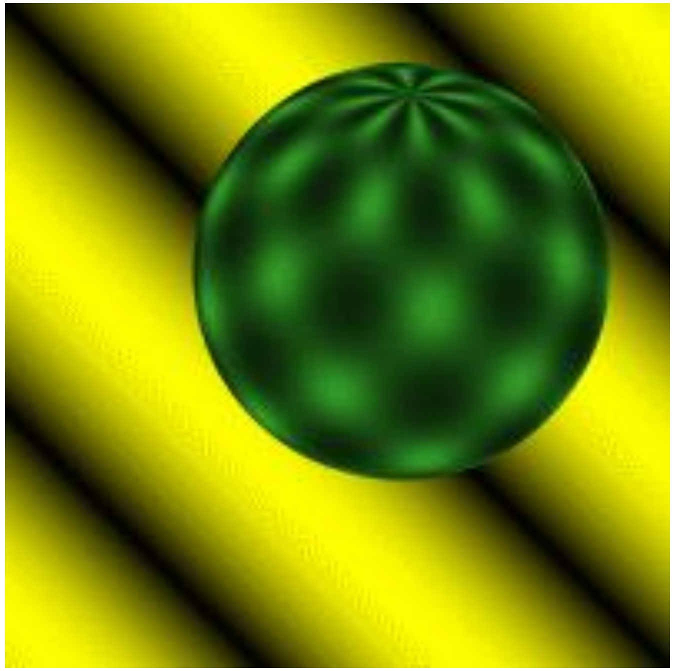
Video analysis

- Video is a sequence of images
- A pixel is located in space (x,y) and time (t): $I(x, y, t)$

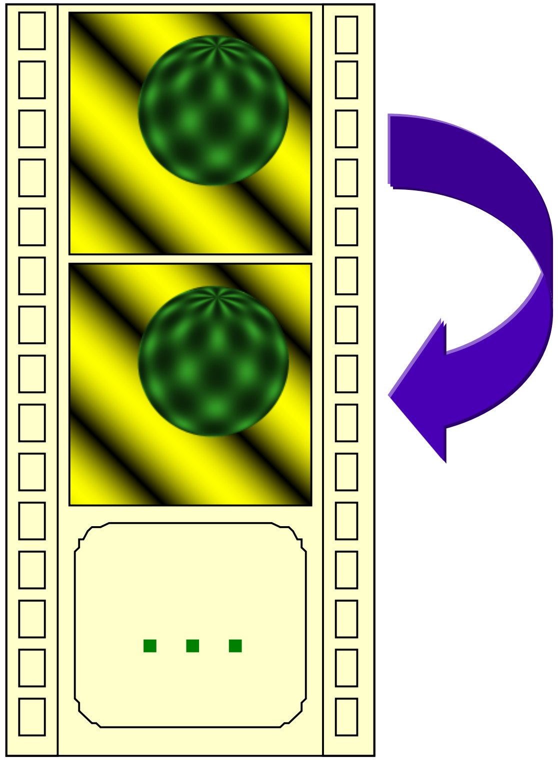


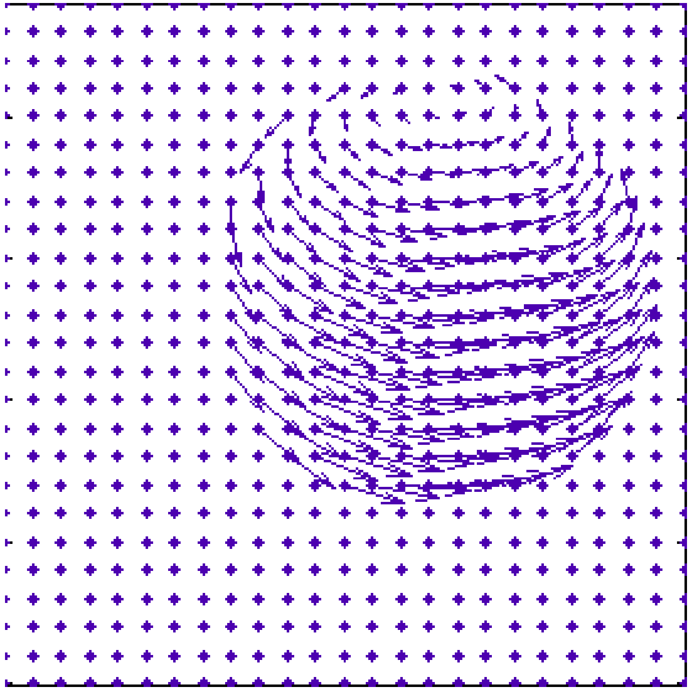
Motion perception: Motion field

- Minimum number of images to analyze a video is 2
- Calculate displacements over pair of frames



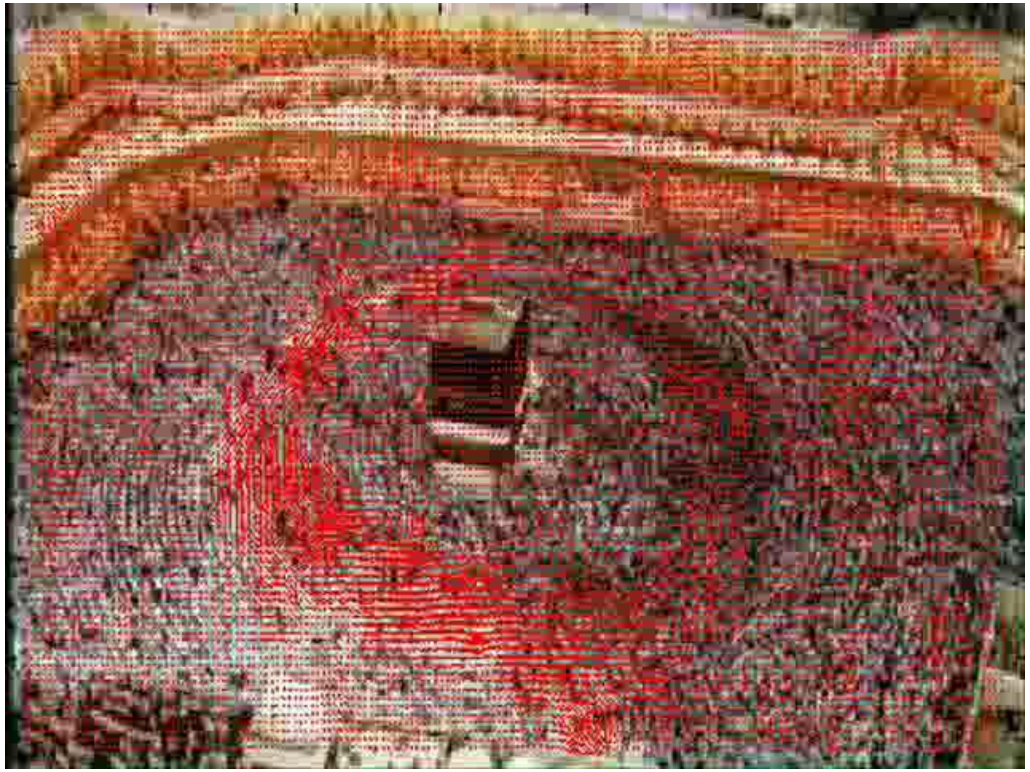
Video





Motion field examples

Dense motion field



http://www.cs.cmu.edu/~saada/Projects/CrowdSegmentation/

Sparse motion field

http://www.youtube.com/watch?v=ckVQrwYIjAs

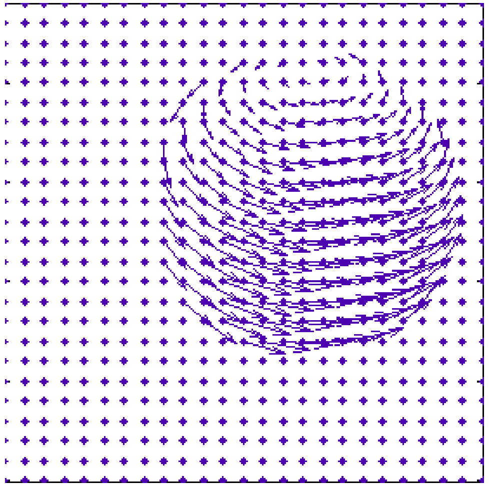
Application: surveillance, multimedia

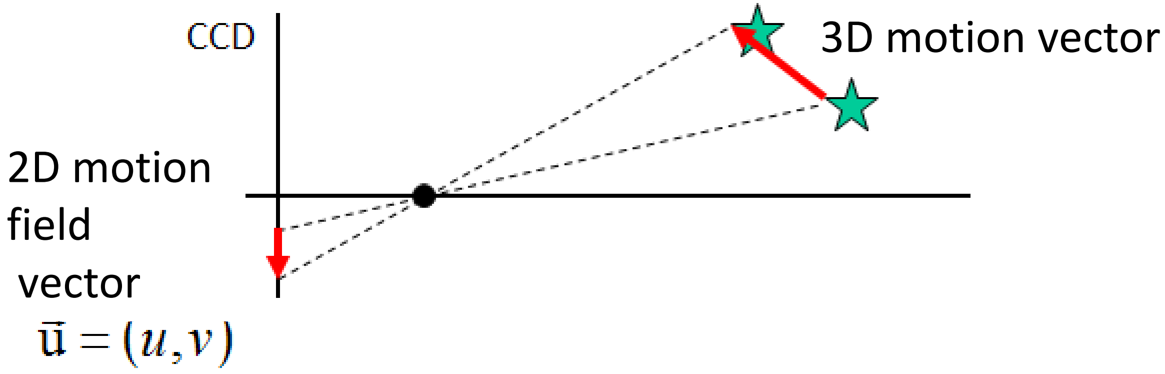
Tracking with occlusions via Graph cuts, N. Papadakis and A. Bugeau. *TPAMI 2011*

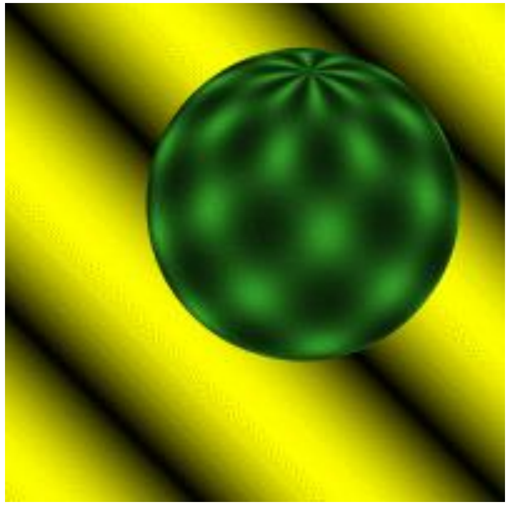
Motion perception: Motion field

- The motion field is a *projection* of 3D motion to image

  [Horn&Schunck]





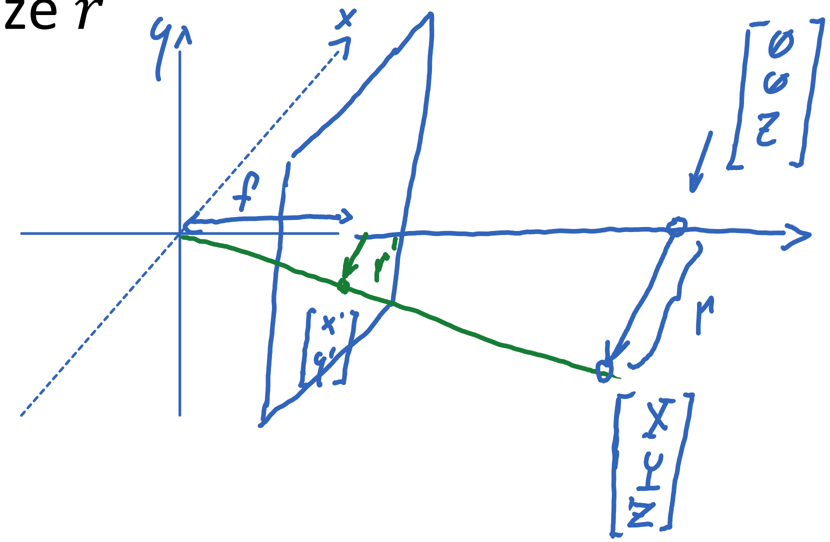


In this case, the 2D motion
field vector is equal to optical flow vector

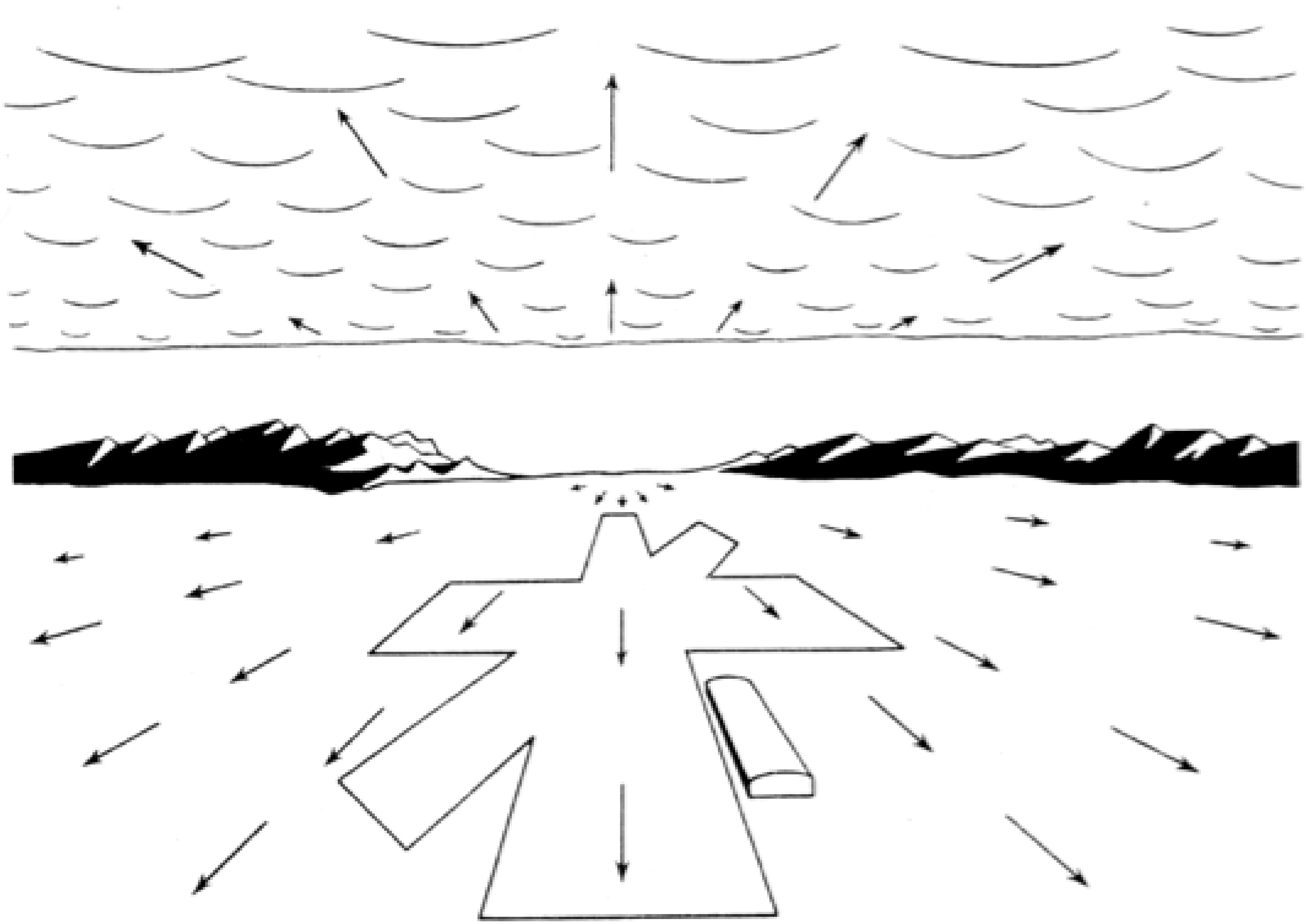
How do constant motions appear from far away and how do they appear close by?
*(*See your notes*)*

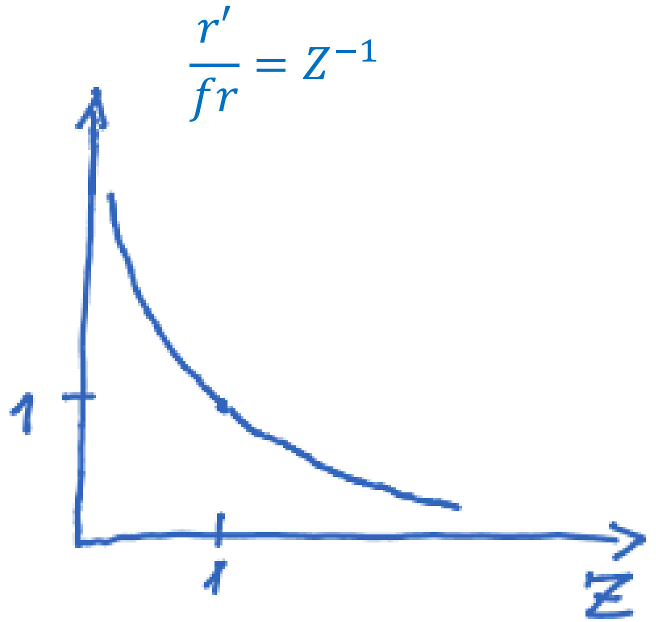
Depth and motion parallax

- Relation between 3D motion size r and its 2D projection size r'
- Assume a parallel translation



Depth and motion paralax

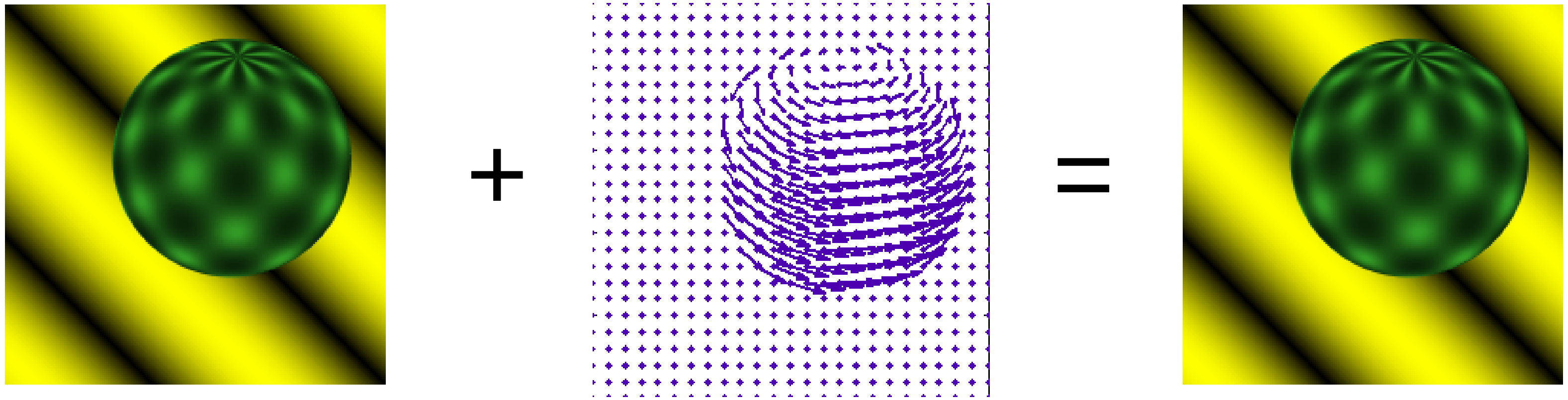




Motion vector length is inversely proportional to depth of 3D point.

Optical flow

- Definition: *optical flow is a velocity field in the image which transforms one image into the next image in a sequence* [Horn&Schunck]

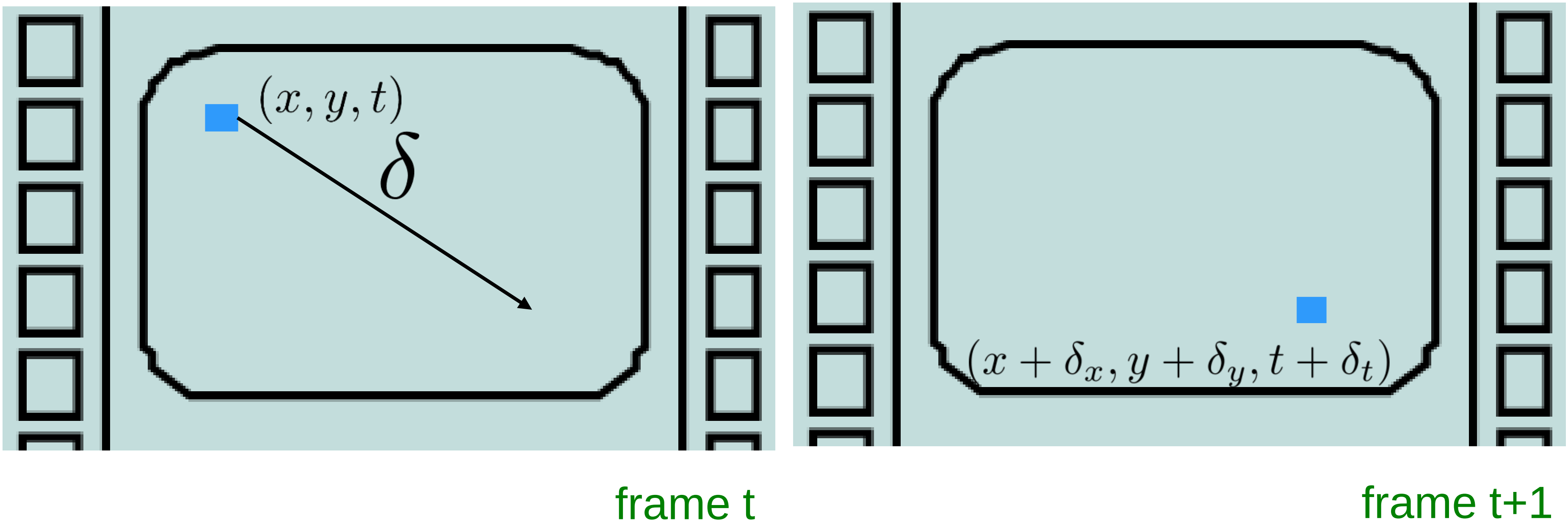


frame #1 flow field frame #2

- Ideally optical flow equals motion field
- Careful: the *apparent motion* is not always induced by the actual motion!

Optical flow: problem definition

- Optical flow introduced by Horn&Schunk (1981)
- Task: Estimate the pixel motion from time t to t+1 *given the intensity measurements at pixels*

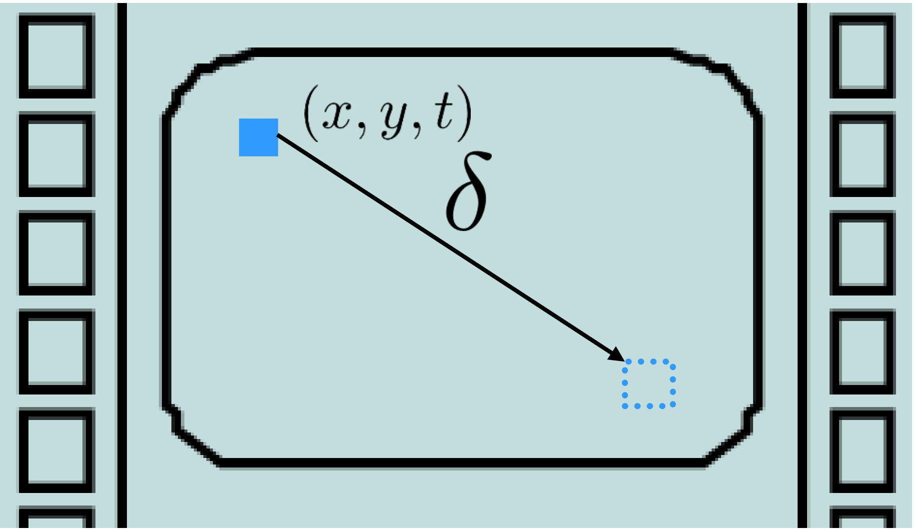


Horn and Schunck, "***Determining Optical Flow***," Artificial Intelligence, 17 (1981), pp. 185-203

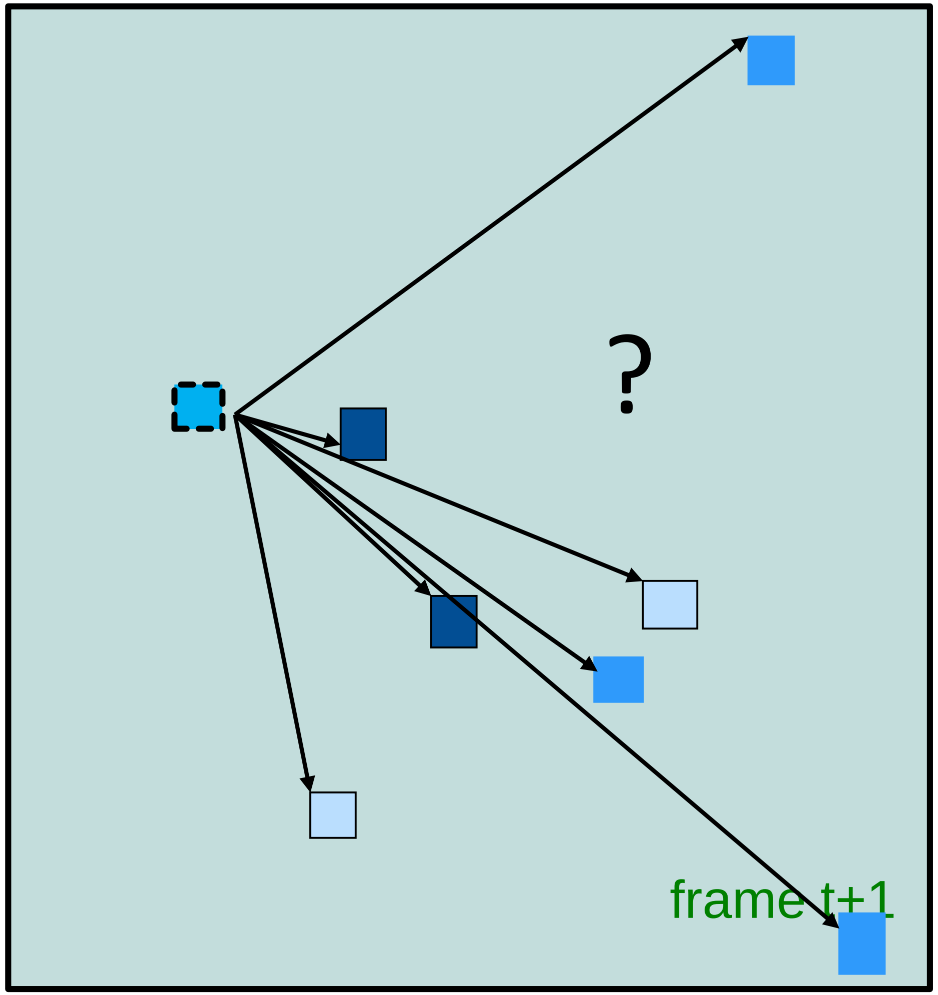
Optical flow: problem definition

- How to find the correct displacement?

$$\delta = [\delta_x, \delta_y, \delta_t]^T$$



frame t

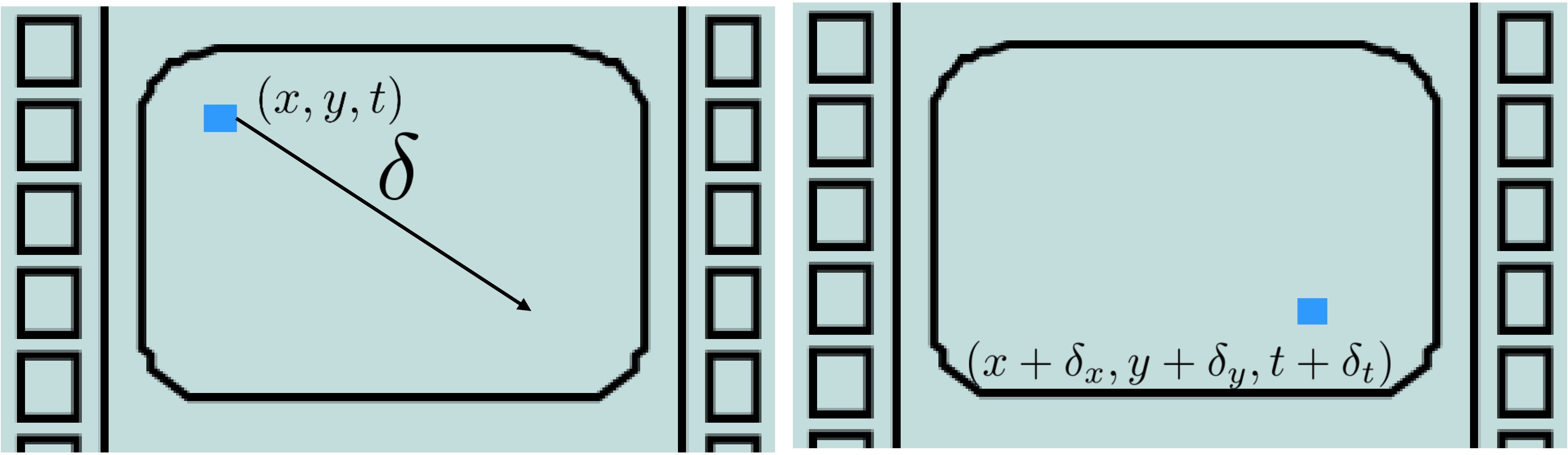


Assumptions required to constrain the space of solutions!

Assumption 1: Brightness constancy

- Intensity of a point does not change during motion

$$\delta = [\delta_x, \delta_y, \delta_t]^T$$

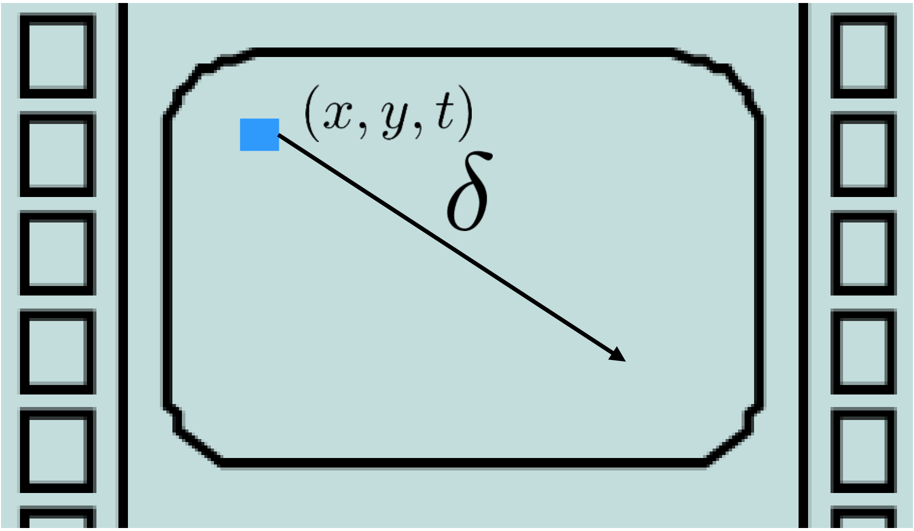


frame t

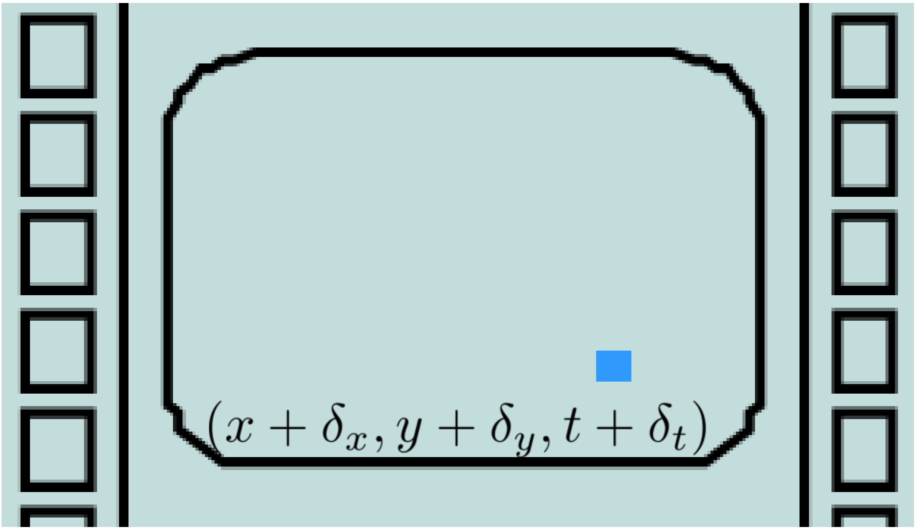
frame t+1

Assumption 2: Small displacements

- The displacement vector $\delta = [\delta_x, \delta_y, \delta_t]^T$ is sufficiently small.
- Actually, assume that the length $\|[\delta_x, \delta_y]\|$ is small.



frame t



frame t+1

Derivation at single pixel

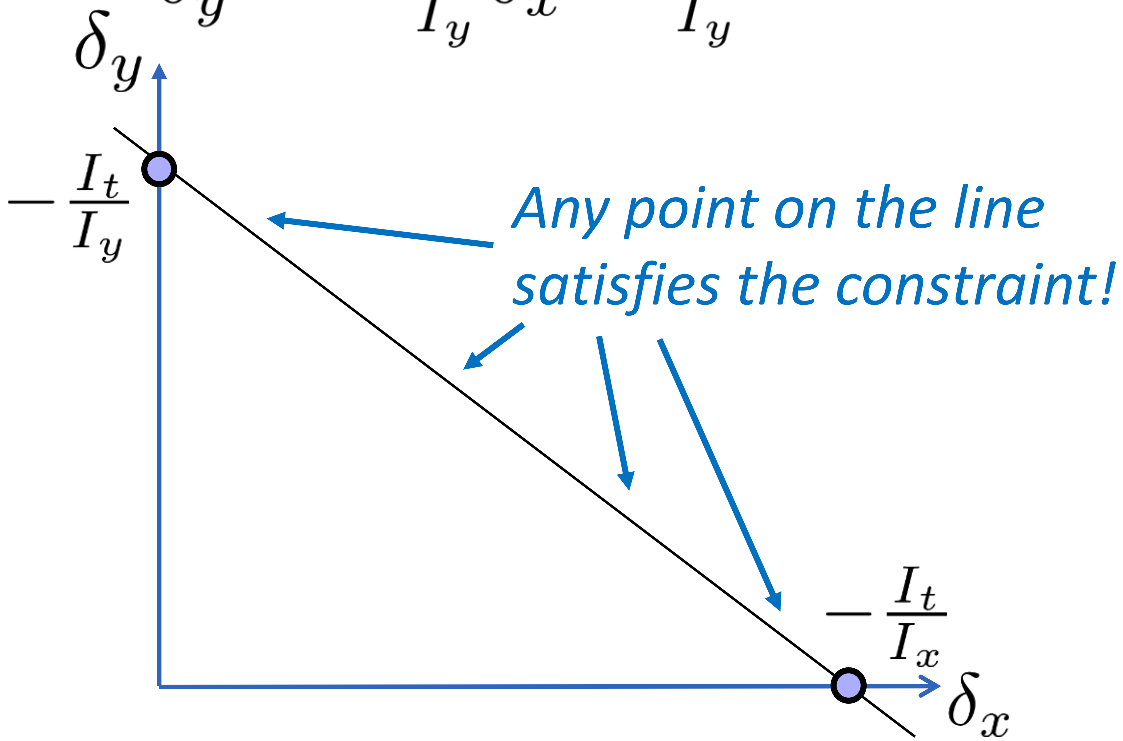
Optical flow constraint equation

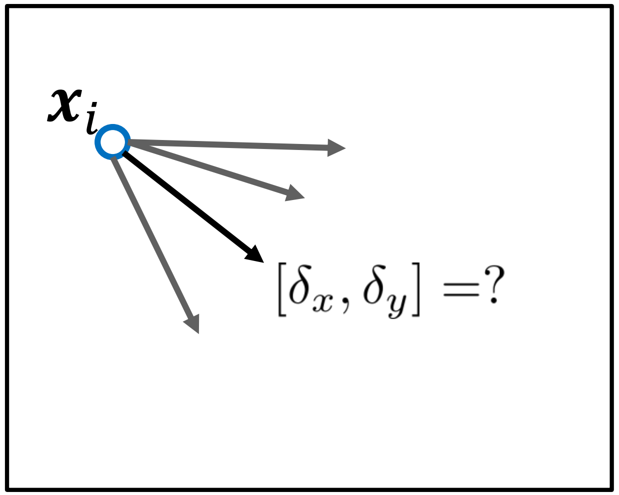
- Optical flow constraint where we set $\delta_t = 1$:

$$I_x(\mathbf{x}_i)\delta_x + I_y(\mathbf{x}_i)\delta_y + I_t(\mathbf{x}_i) = 0$$

- This is a line equation with parameters (δ_x, δ_y):

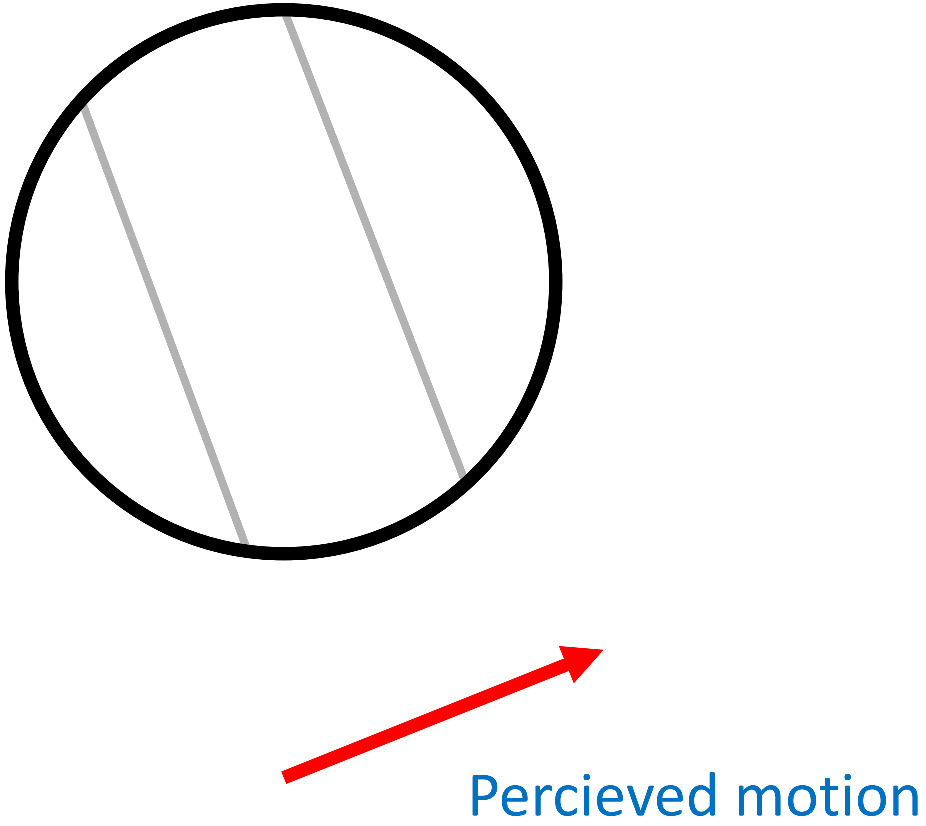
$$\delta_y = -\frac{I_x}{I_y}\delta_x - \frac{I_t}{I_y}$$





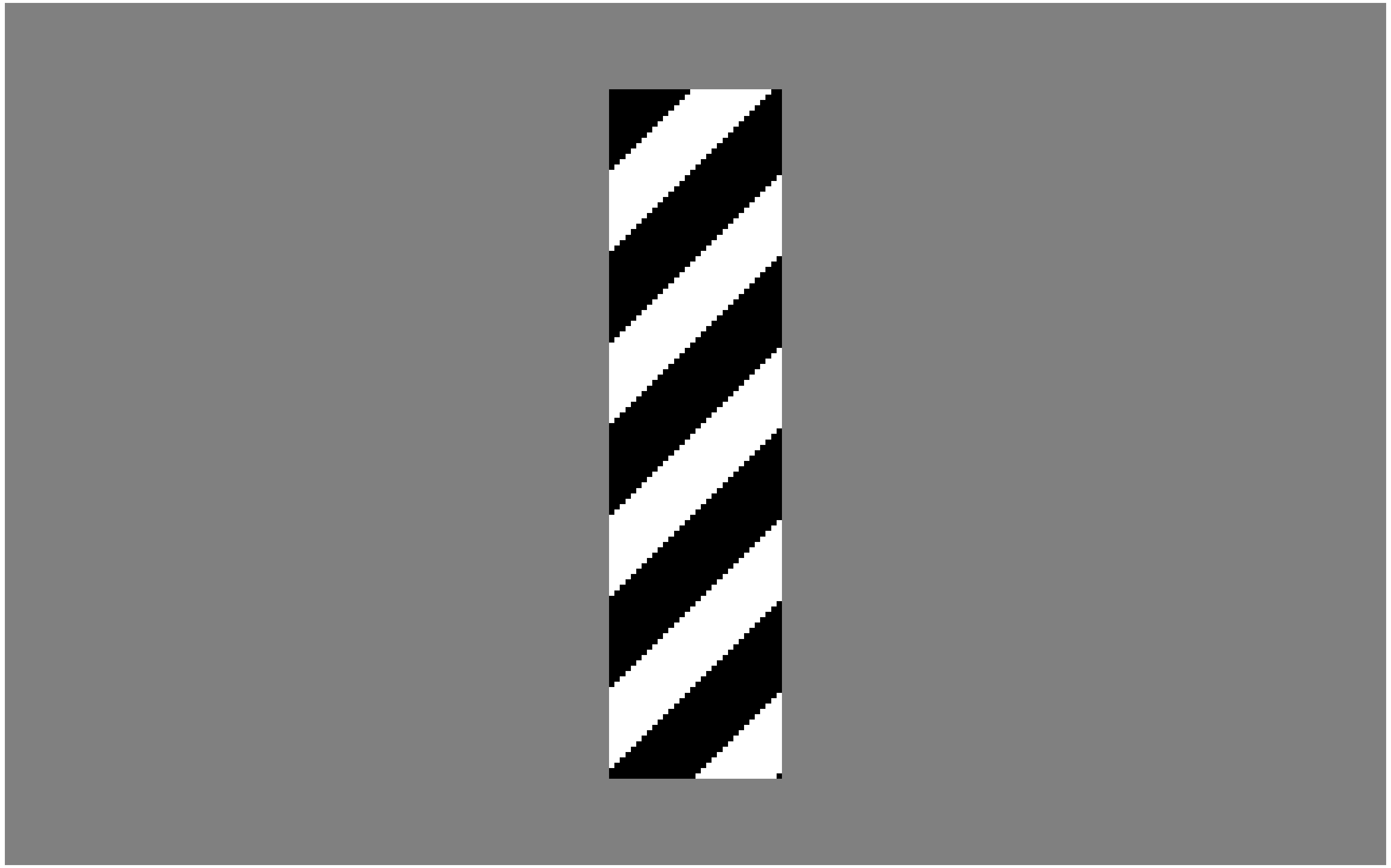
This is the aperture problem

Component parallel to the edge unknown...



Barber poll illusion

The aperture problem!

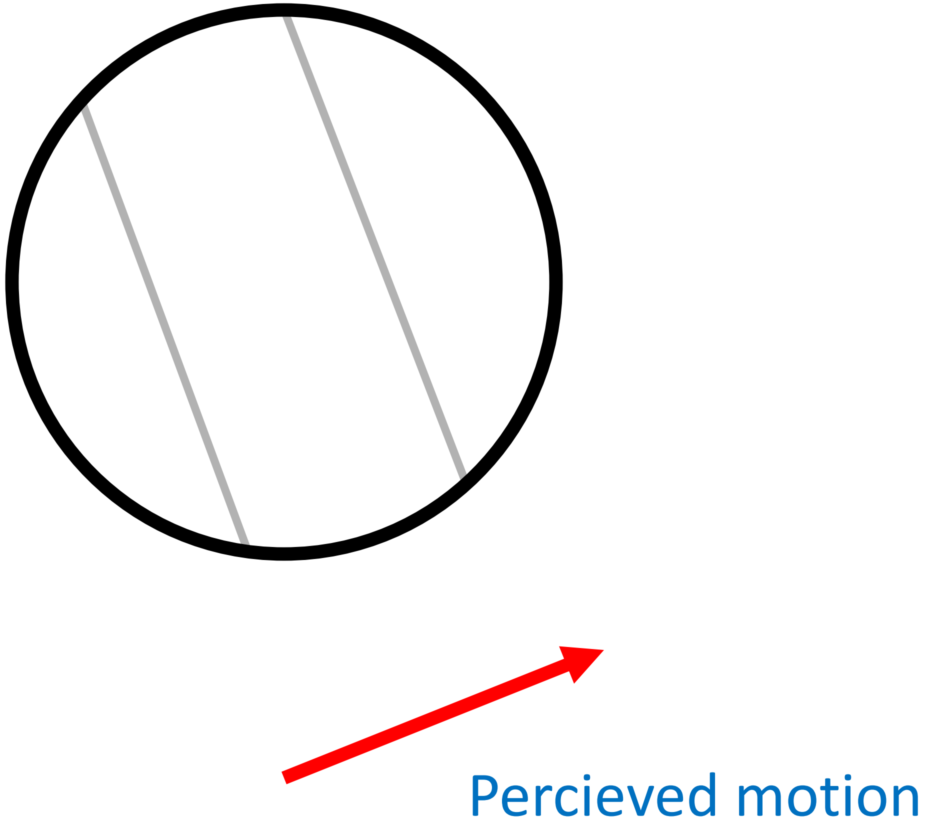


http://www.sandlotscience.com/Ambiguous/Barberpole_Illusion.htm

http://en.wikipedia.org/wiki/Barber's_pole

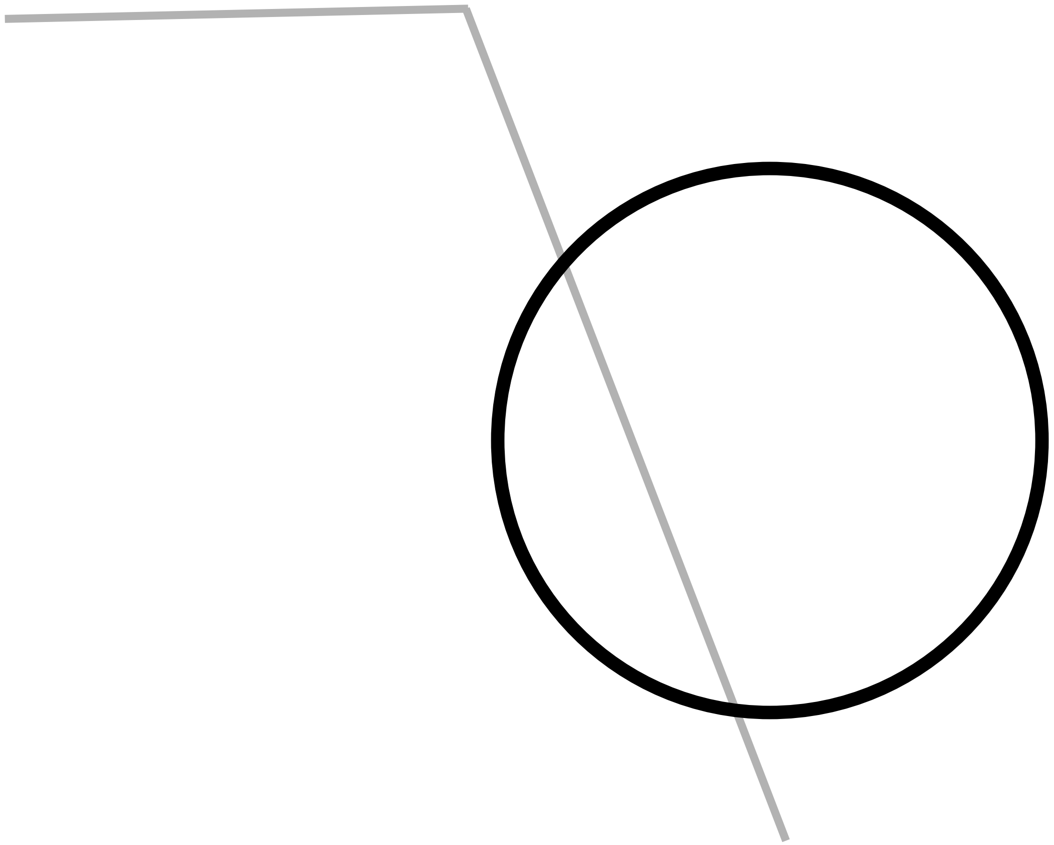
This is the aperture problem

The motion component parallel to the edge is unknown...



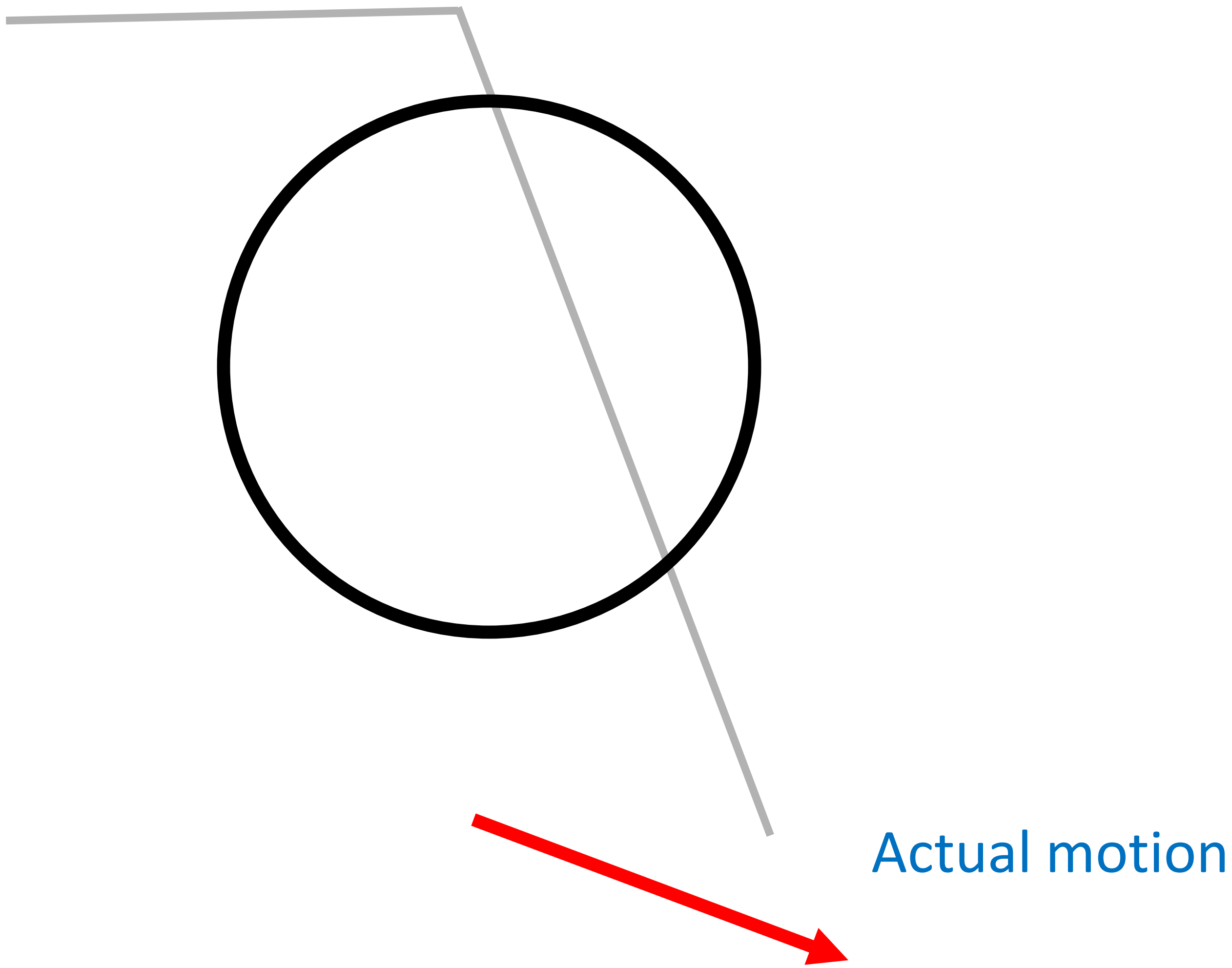
This is the aperture problem

Component parallel to the edge unknown...



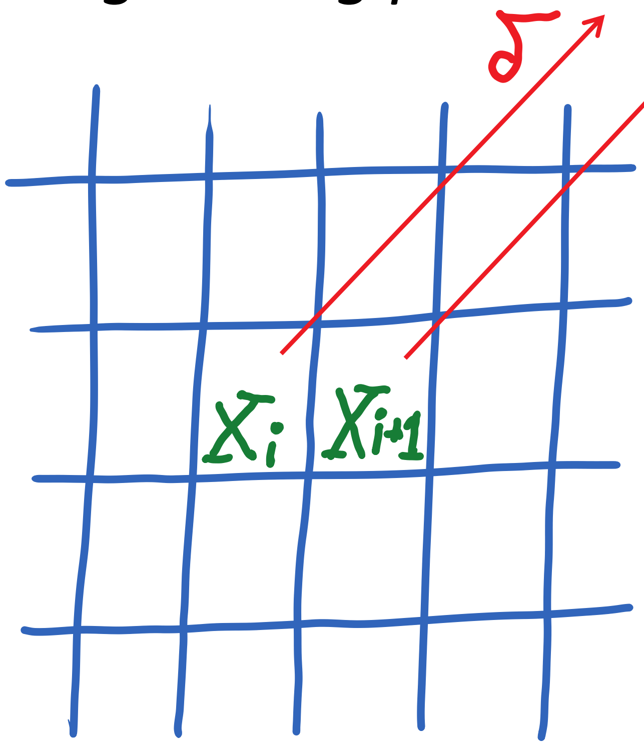
This is the aperture problem

Component parallel to the edge unknown...



Solving the aperture problem

- More equations per pixel are required!
- Assumption 3: Local motion coherency constraint -- *assume that neighboring pixels have equal displacements.*



$$\delta = [\delta_x, \delta_y, \delta_t]^T$$

Further assume that frames are sampled discrete timesteps, i.e., $\delta_t = 1 \;\forall\; t$.

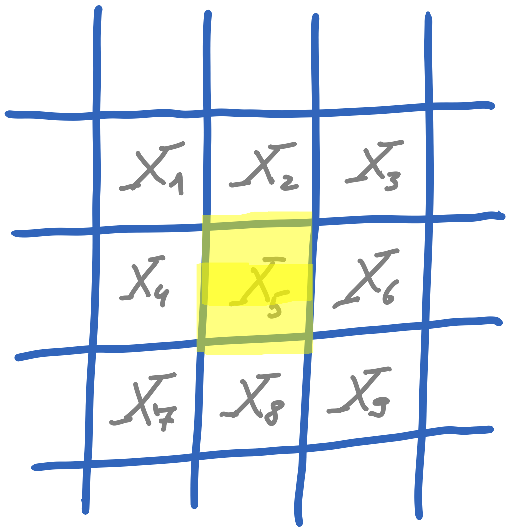
Lucas and Kanade *"An Iterative Image Registration Technique with an Application to Stereo Vision"* IJCAI '81 pp. 674-679

Solving the aperture problem

- $\boldsymbol{x}_i$ … *i*-th pixel coordinates; discrete time-steps $(\delta_t = 1)$

$$I_x(\mathbf{x}_i)\delta_x + I_y(\mathbf{x}_i)\delta_y = -I_t(\mathbf{x}_i)1$$

- Consider a small 3×3 window:



$$\Rightarrow$$

$$\begin{aligned}
I_x(\mathbf{x}_1)\delta_x + I_y(\mathbf{x}_1)\delta_y &= -I_t(\mathbf{x}_1) \\
I_x(\mathbf{x}_2)\delta_x + I_y(\mathbf{x}_2)\delta_y &= -I_t(\mathbf{x}_2) \\
\ldots \\
I_x(\mathbf{x}_9)\delta_x + I_y(\mathbf{x}_9)\delta_y &= -I_t(\mathbf{x}_9)
\end{aligned}$$

Solve the aperture problem

- Rewrite into a matrix form:

$$
\begin{aligned}
I_x(\mathbf{x}_1)\delta_x + I_y(\mathbf{x}_1)\delta_y &= -I_t(\mathbf{x}_1) \\
I_x(\mathbf{x}_2)\delta_x + I_y(\mathbf{x}_2)\delta_y &= -I_t(\mathbf{x}_2) \\
\ldots \\
I_x(\mathbf{x}_9)\delta_x + I_y(\mathbf{x}_9)\delta_y &= -I_t(\mathbf{x}_9)
\end{aligned}
$$

$$
\underbrace{\begin{bmatrix} I_x(\mathbf{x}_1) & I_y(\mathbf{x}_1) \\ I_x(\mathbf{x}_2) & I_y(\mathbf{x}_2) \\ \vdots & \vdots \\ I_x(\mathbf{x}_9) & I_y(\mathbf{x}_9) \end{bmatrix}}_{9\text{X}2}
\underbrace{\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}}_{2\text{X}1}
= -
\underbrace{\begin{bmatrix} I_t(\mathbf{x}_1) \\ I_t(\mathbf{x}_2) \\ \vdots \\ I_t(\mathbf{x}_9) \end{bmatrix}}_{9\text{X}1}
$$

$$
\mathbf{A}\mathbf{d} = \mathbf{b}
$$

Solve the aperture problem

Problem: We have more equations than unknowns

$$\mathbf{A}\mathbf{d} = \mathbf{b} \longrightarrow \tilde{\mathbf{d}} = \arg\min_{\mathbf{d}} \|\mathbf{A}\mathbf{d} - \mathbf{b}\|^2$$

Least-squares solution by pseudo inverse!

$$\underbrace{A^T A}_{\text{SQUARE}} d = A^T b \quad / (A^T A)^{-1}$$

$$d = (A^T A)^{-1} A^T b$$

Structure of the solution

- In principle one could compute $\mathbf{d} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ at each pixel.
- But this can be done much more efficiently!
- Possible to work out the equations independently for δ_x and δ_y at each pixel!
- START HERE:
  We can show that $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d} = \boldsymbol{A}^T\boldsymbol{b}$ equals to (show for *home exercise!*):

$$\begin{bmatrix} \sum_{i=1:9} I_x(\mathrm{x}_i)^2 & \sum_{i=1:9} I_x(\mathrm{x}_i) I_y(\mathrm{x}_i) \\ \sum_{i=1:9} I_x(\mathrm{x}_i) I_y(\mathrm{x}_i) & \sum_{i=1:9} I_y(\mathrm{x}_i)^2 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = -\begin{bmatrix} \sum_{i=1:9} I_x(\mathrm{x}_i) I_t(\mathrm{x}_i) \\ \sum_{i=1:9} I_y(\mathrm{x}_i) I_t(\mathrm{x}_i) \end{bmatrix}$$

Solve the aperture problem

$$\begin{bmatrix} \sum_{i=1:9} I_x(\mathrm{x}_i)^2 & \sum_{i=1:9} I_x(\mathrm{x}_i) I_y(\mathrm{x}_i) \\ \sum_{i=1:9} I_x(\mathrm{x}_i) I_y(\mathrm{x}_i) & \sum_{i=1:9} I_y(\mathrm{x}_i)^2 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = - \begin{bmatrix} \sum_{i=1:9} I_x(\mathrm{x}_i) I_t(\mathrm{x}_i) \\ \sum_{i=1:9} I_y(\mathrm{x}_i) I_t(\mathrm{x}_i) \end{bmatrix}$$

- We will drop $\boldsymbol{x}_i$ and index i in interest of compact notation:

$$\begin{bmatrix} \sum I_x{}^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y{}^2 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Solve the aperture problem

- Compact notation:

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = -\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- Now invert:

$$\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = -\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

Derive the inverse yourself

- Equation from previous slide:

$$\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = -\begin{bmatrix} \sum I_x{}^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y{}^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- Recall the matrix inversion rule:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad A^{-1} = \frac{1}{|\mathrm{A}|} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$\begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = ?$$

Now write the solution of $\boldsymbol{d}$

- Applying the inversion rule:

$$\begin{bmatrix}\delta_x \\ \delta_y\end{bmatrix} = \frac{1}{(\sum I_x^{\,2})(\sum I_y^{\,2}) - (\sum I_x I_y)^2}\begin{bmatrix}\sum I_y^{\,2} & -\sum I_x I_y \\ -\sum I_x I_y & \sum I_x^{\,2}\end{bmatrix}\begin{bmatrix}-\sum I_x I_t \\ -\sum I_y I_t\end{bmatrix}$$

- Results in the following solution:

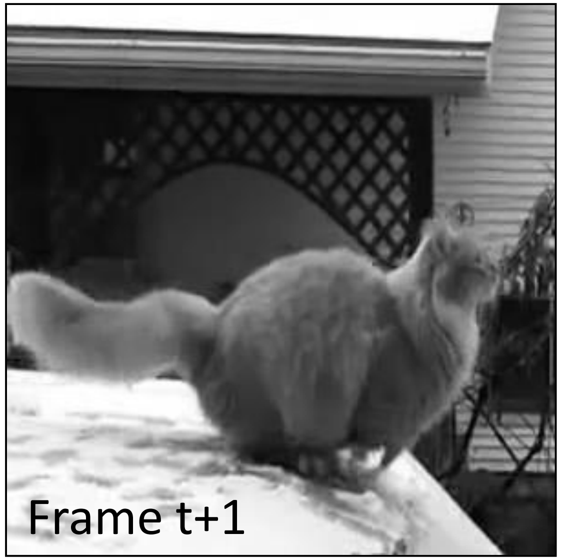
$$\delta_x = \frac{-(\sum I_y^{\,2})\sum I_x I_t + (\sum I_x I_y)\sum I_y I_t}{(\sum I_x^{\,2})\sum I_y^{\,2} - (\sum I_x I_y)^2}$$

$$\delta_y = \frac{(\sum I_x I_y)\sum I_x I_t - (\sum I_x^{\,2})\sum I_y I_t}{(\sum I_x^{\,2})\sum I_y^{\,2} - (\sum I_x I_y)^2}$$

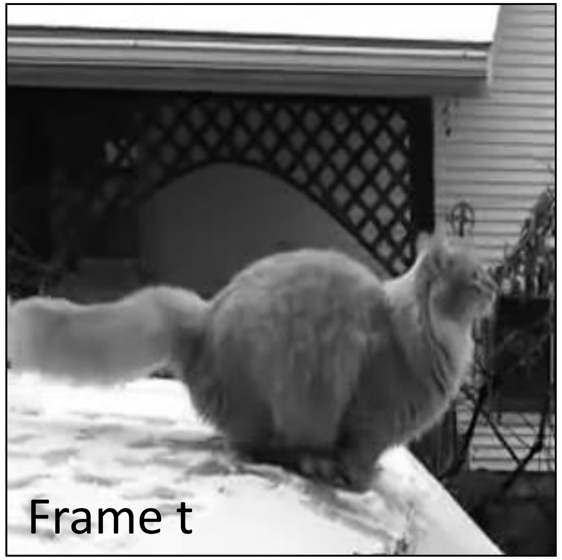
That's great!
…Why?!
We'll see soon.

Implementation by example

- The following video will be considered as an example



Frame t+1



Frame t

Implementation by example

- How to compute $I_x(x, y, t), I_y(x, y, t), I_t(x, y, t)$?
- Start with an easy one: I_t

$$\delta_x = \frac{-(\sum I_y^2)\sum I_x I_t + (\sum I_x I_y)\sum I_y I_t}{(\sum I_x^2)\sum I_y^2 - (\sum I_x I_y)^2}$$

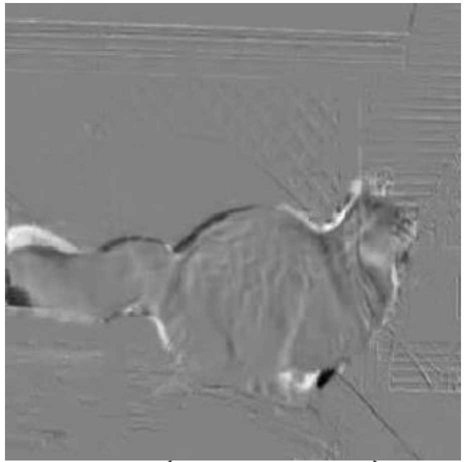
$$\delta_y = \frac{(\sum I_x I_y)\sum I_x I_t - (\sum I_x^2)\sum I_y I_t}{(\sum I_x^2)\sum I_y^2 - (\sum I_x I_y)^2}$$

$I(x, y, t+1)$

-

$I(x, y, t)$

=

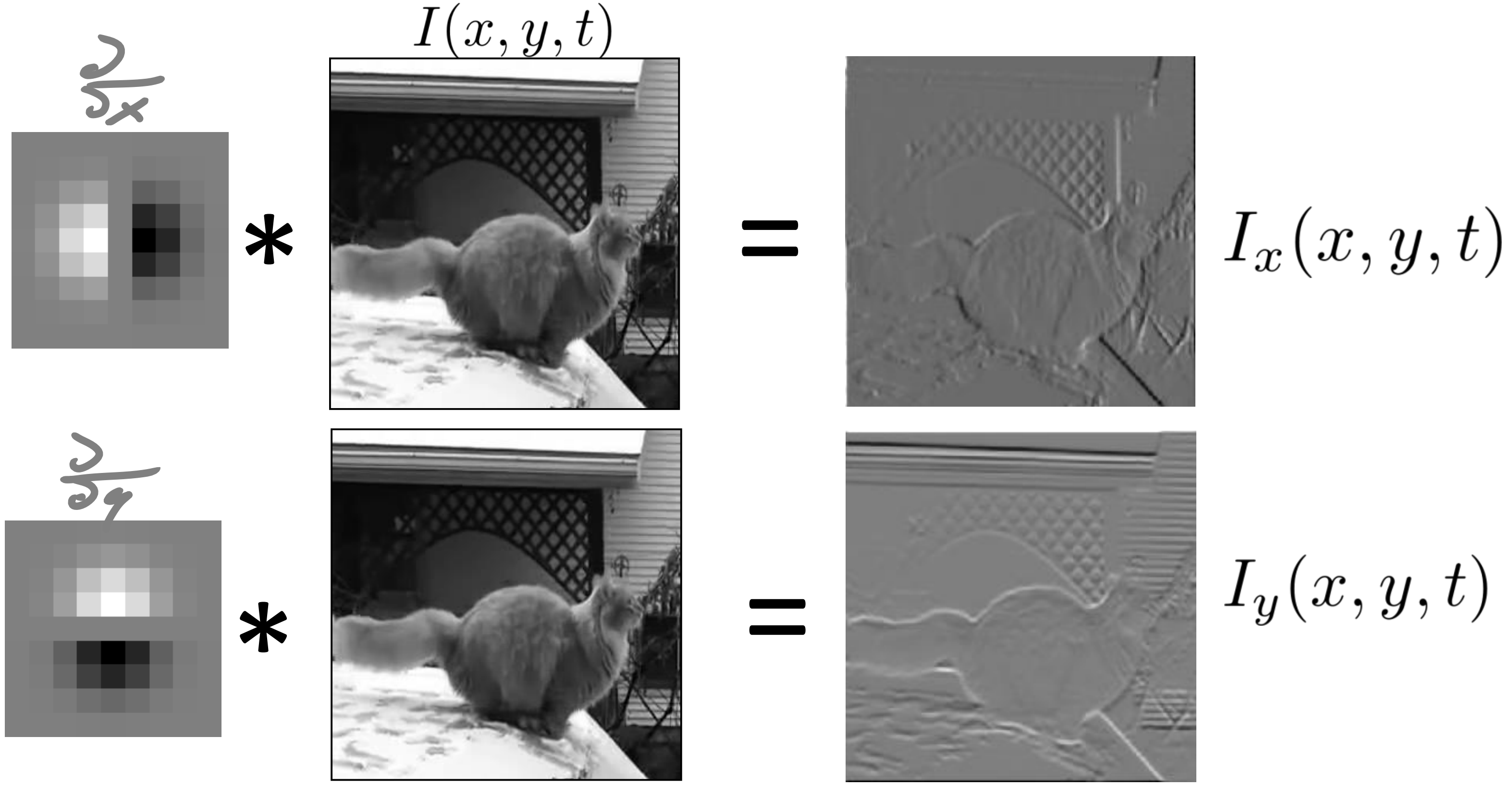


$I_t(x, y, t)$

Temporal derivative is approximated by difference between consecutive images.

Implementation by example

- How to compute $I_x(x, y, t), I_y(x, y, t), I_t(x, y, t)$?
- Approximate spatial derivatives I_x, I_y by convolution



If this is a mystery to you, check Prince’s book, Sec. 13.1.3. or Szeliski, Sec. 4.2.1.

Implementation – putting it together

$I(x,y,t)$ $\quad$ $I(x,y,t+1)$

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

$\frac{\partial I}{\partial y} = I_y$

1

$\Rightarrow$

$I_x \cdot I_t$ $\quad$ $I_y \cdot I_t$

2

$I_x^2 = I_x \cdot I_x$ $\quad$ $I_y^2 = I_y \cdot I_y$ $\quad$ $I_x \cdot I_y$

$\Downarrow$

3

$$\delta_x = \frac{-(\sum I_y{}^2)\sum I_x I_t + (\sum I_x I_y)\sum I_y I_t}{(\sum I_x{}^2)\sum I_y{}^2 - (\sum I_x I_y)^2}$$

$$\delta_y = \frac{(\sum I_x I_y)\sum I_x I_t - (\sum I_x{}^2)\sum I_y I_t}{(\sum I_x{}^2)\sum I_y{}^2 - (\sum I_x I_y)^2}$$

$\Rightarrow$

$\delta_x(x,y)$ $\quad$ $\delta_y(x,y)$

x and y flow components!

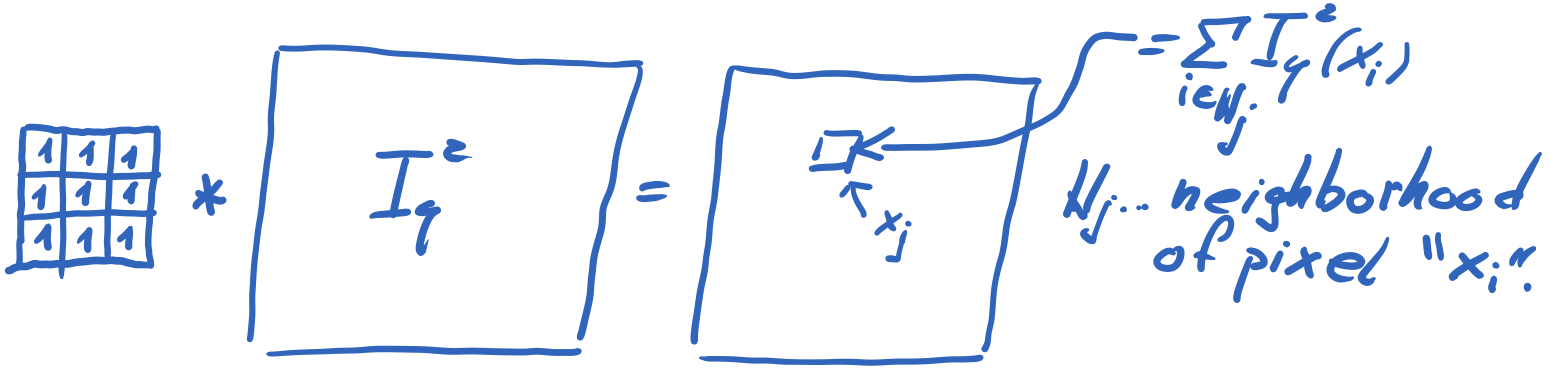
A note on summations

- Recall that the equations require summing over neighboring pixels:

$$\delta_x = \frac{-(\sum I_y^2)\sum I_x I_t + (\sum I_x I_y)\sum I_y I_t}{(\sum I_x^2)\sum I_y^2 - (\sum I_x I_y)^2}$$

$$\delta_y = \frac{(\sum I_x I_y)\sum I_x I_t - (\sum I_x^2)\sum I_y I_t}{(\sum I_x^2)\sum I_y^2 - (\sum I_x I_y)^2}$$

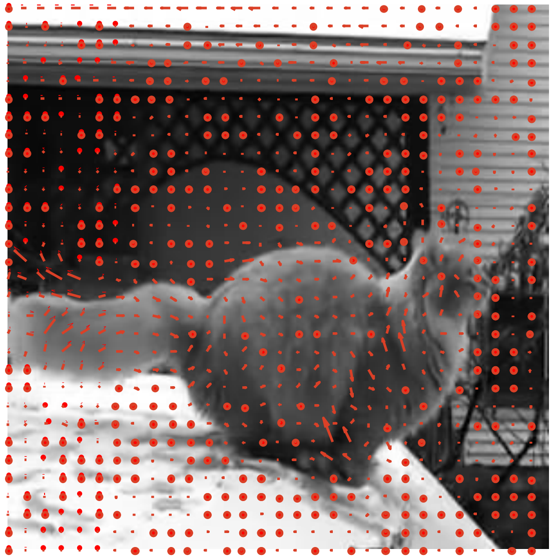
- This can be trivially implemented by convolution, e.g., for $\sum I_y^2$:



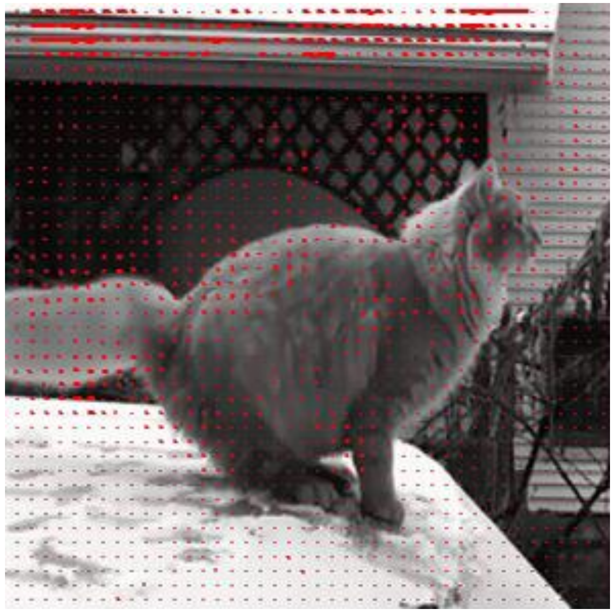
Back to Waffle the terrible

Frame t

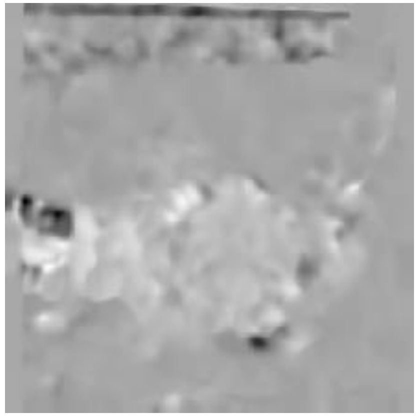
Frame t+1



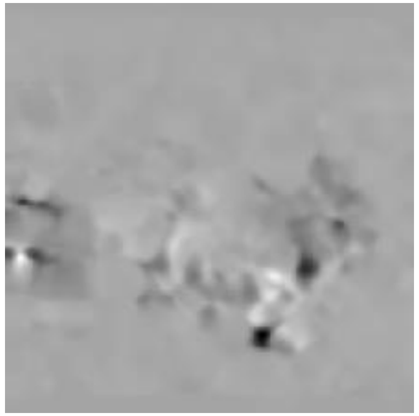
Flow (δ_x, δ_y)



δ_x



δ_y



Flow computation reliability

- Flow cannot be computed just at any point
- Recall that the following equation is implicitly solved:

$$\begin{bmatrix} \sum I_x^{\ 2} & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^{\ 2} \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

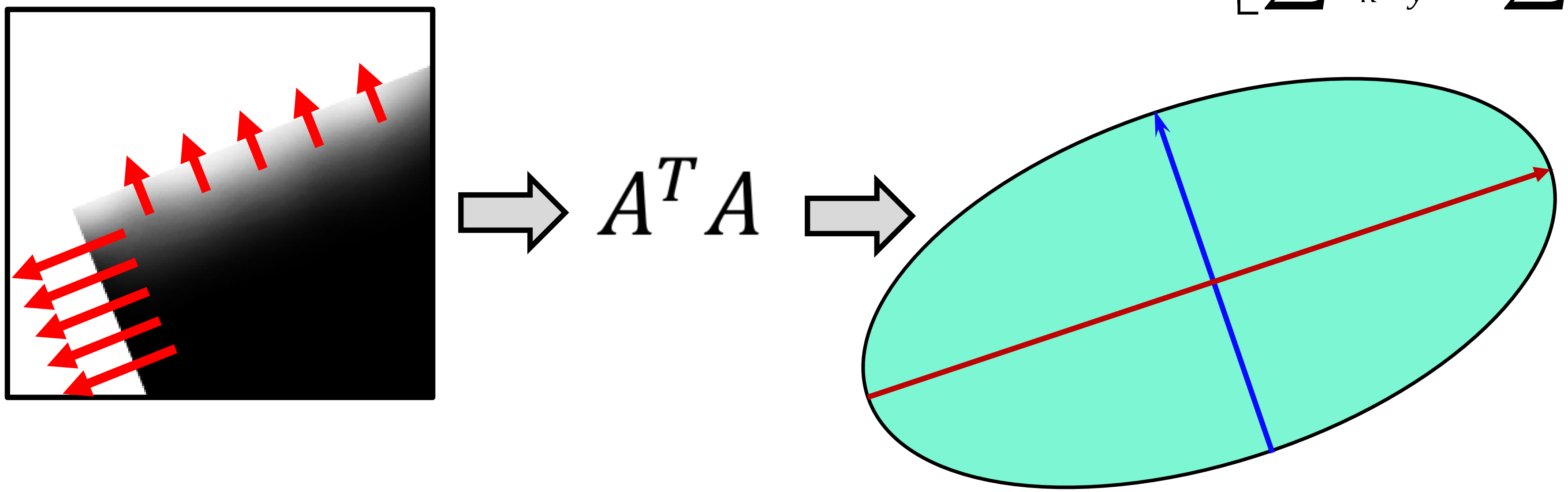
$$\mathbf{A}^T \mathbf{A} \mathbf{d} = \mathbf{A}^T \mathbf{b}$$

When is this system solvable?

- $\mathbf{A}^T\mathbf{A}$ must not be singular, (cannot invert it otherwise)
  - Eigenvalues λ_1 and λ_2 of $\mathbf{A}^T\mathbf{A}$ must not be too small
- $\mathbf{A}^T\mathbf{A}$ has to be well conditioned
  - Ratio λ_1 / λ_2 must not be too large
    (λ_1 = the larger eigenvalue)

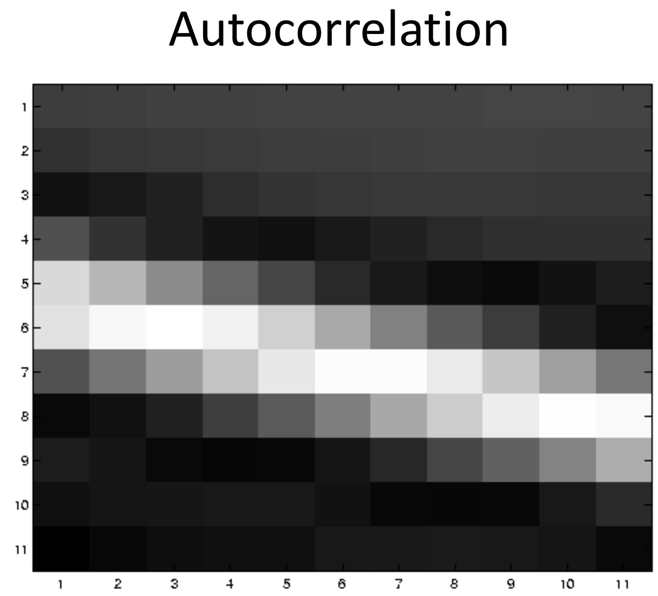
Eigenvalues of A^TA

- $A^T A$ is a covariance matrix of local gradients: $$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$



- Same as in the Harris corner detection!
- Note: If you are unfamiliar with the Harris corner detection, see Prince (Sec. 13.2.2) or Szeliski (Sec. 4.1.1)

Eigenvalues of A^TA



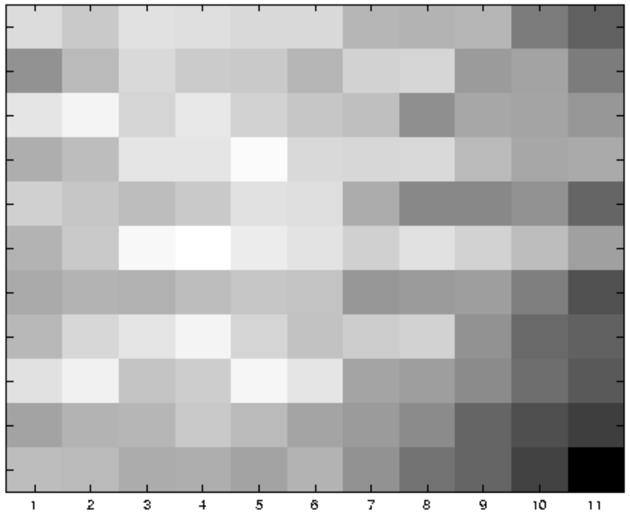
$$A^T A = \begin{bmatrix} \sum I_x^{\ 2} & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^{\ 2} \end{bmatrix}$$

– large gradient in one direction

– large λ_1, small λ_2

Eigenvalues of A^TA

Autocorrelation



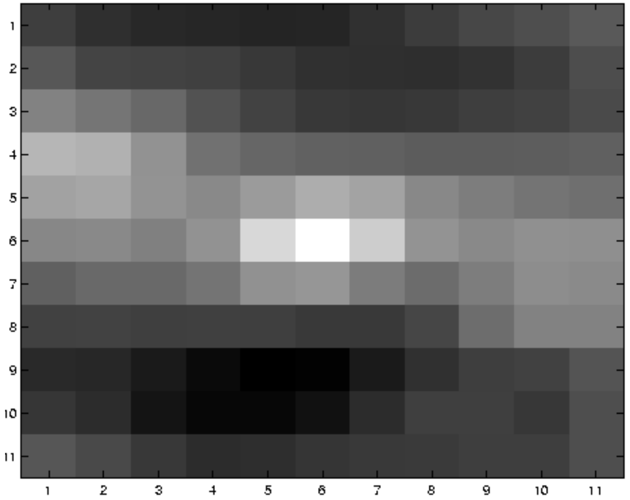
$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

– gradients with small magnitude

– small λ_1, small λ_2

Eigenvalues of A^TA

Autocorrelation



$$A^T A = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

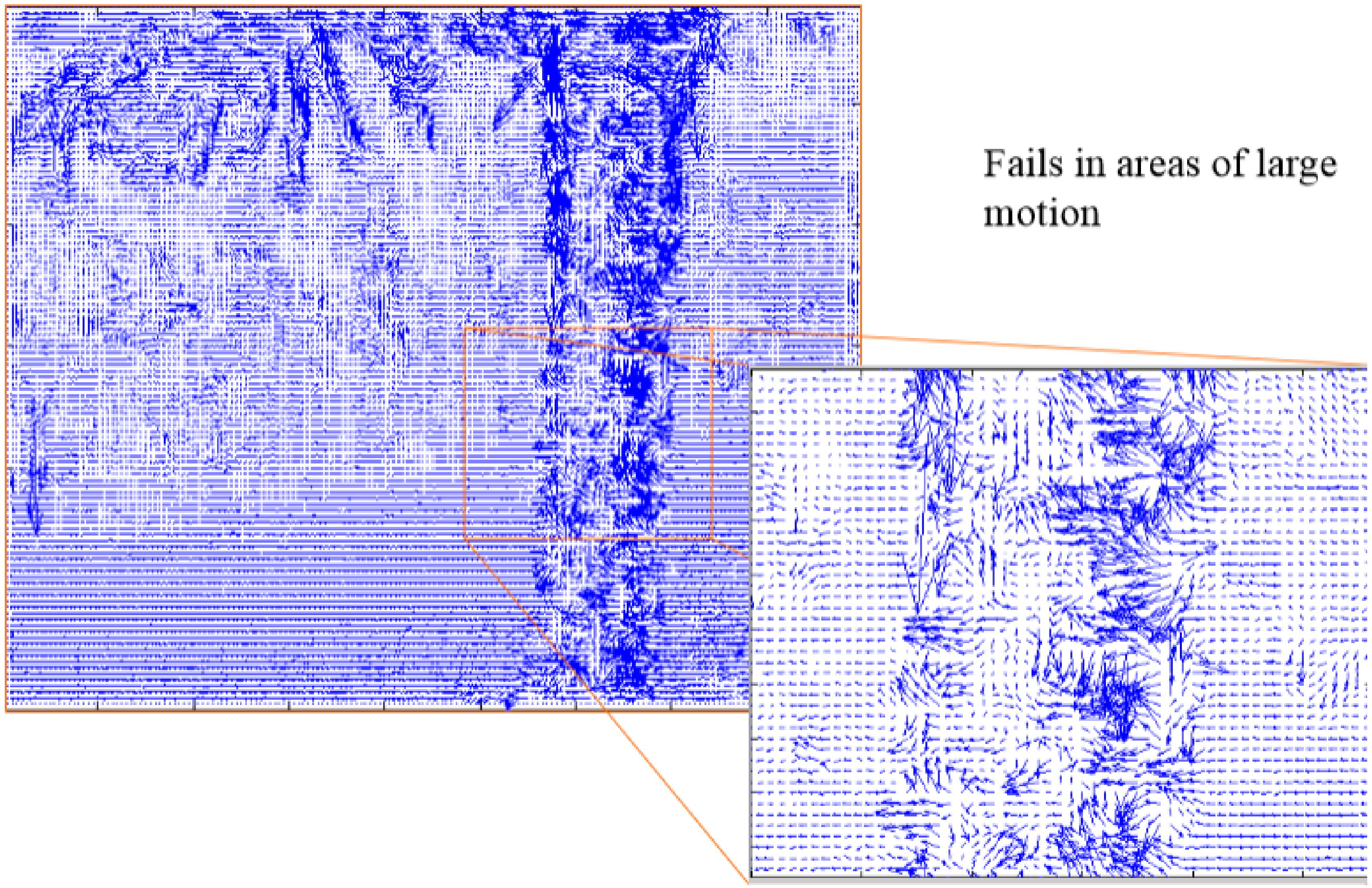
– large gradient magnitudes

– large λ_1, large λ_2

Small motion assumption

- Lucas-Kanade works well only for small motions.
- If an object moves fast, the small motion assumption is violated.
- 2x2 or 3x3 convolution kernels fail to estimate the spatio-temporal derivatives.

Small motion assumption violated

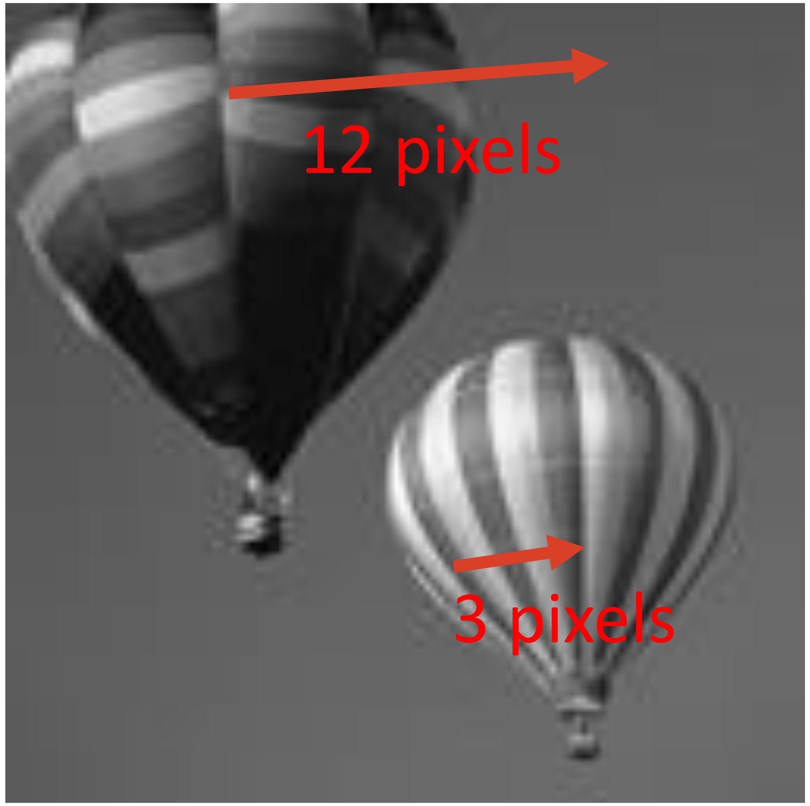


Fails in areas of large motion

- So how to improve estimation of large motions?

Accounting for large motions

- Assume that we can estimate well motions below 3 pixels in length



Can't detect this motion

Reduce the size 4 times

3 pixels

0 pixels !

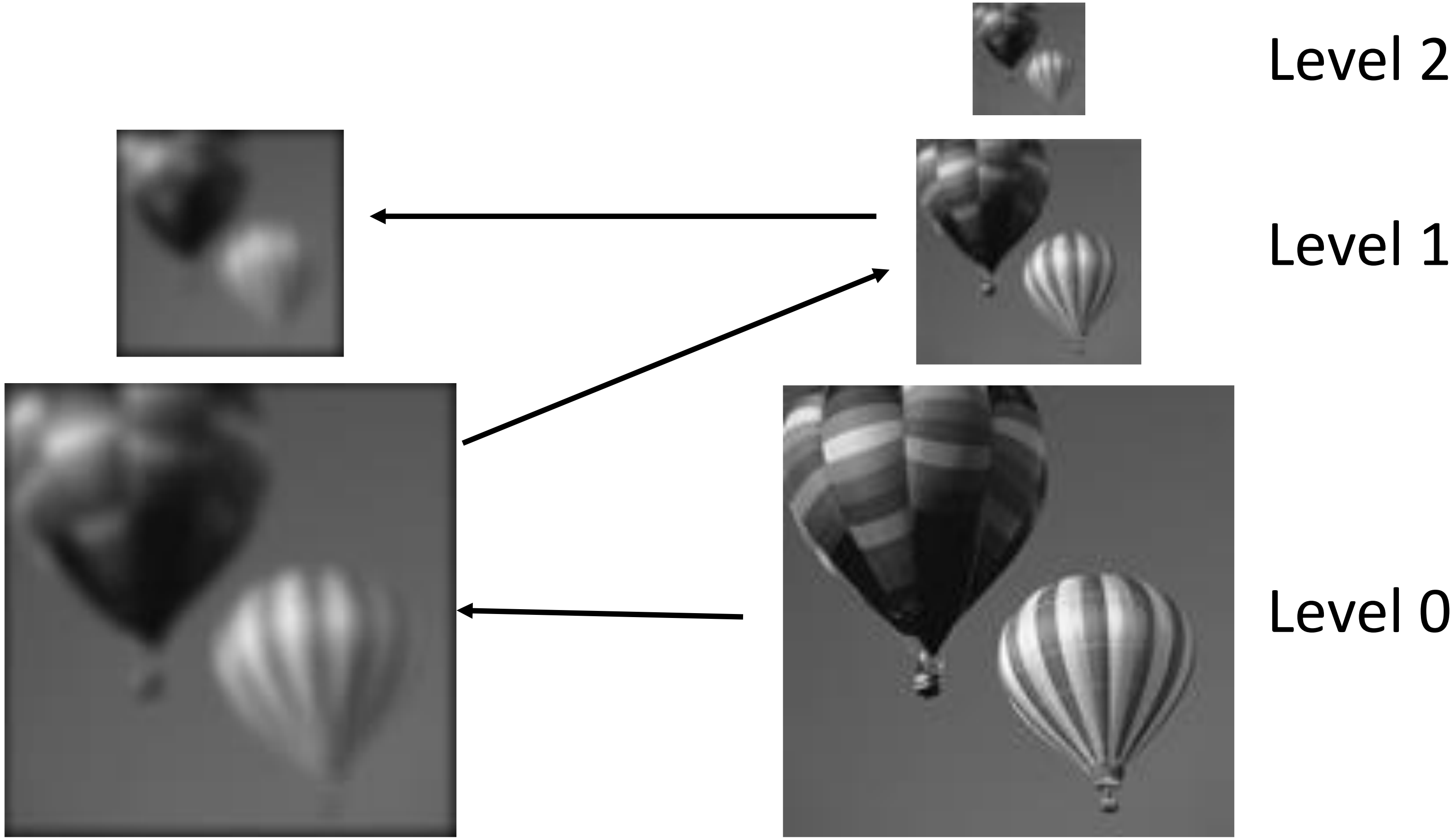
Detection successful

No free lunch?

- But, we will not be able to detect small motions....

Create an image pyramid

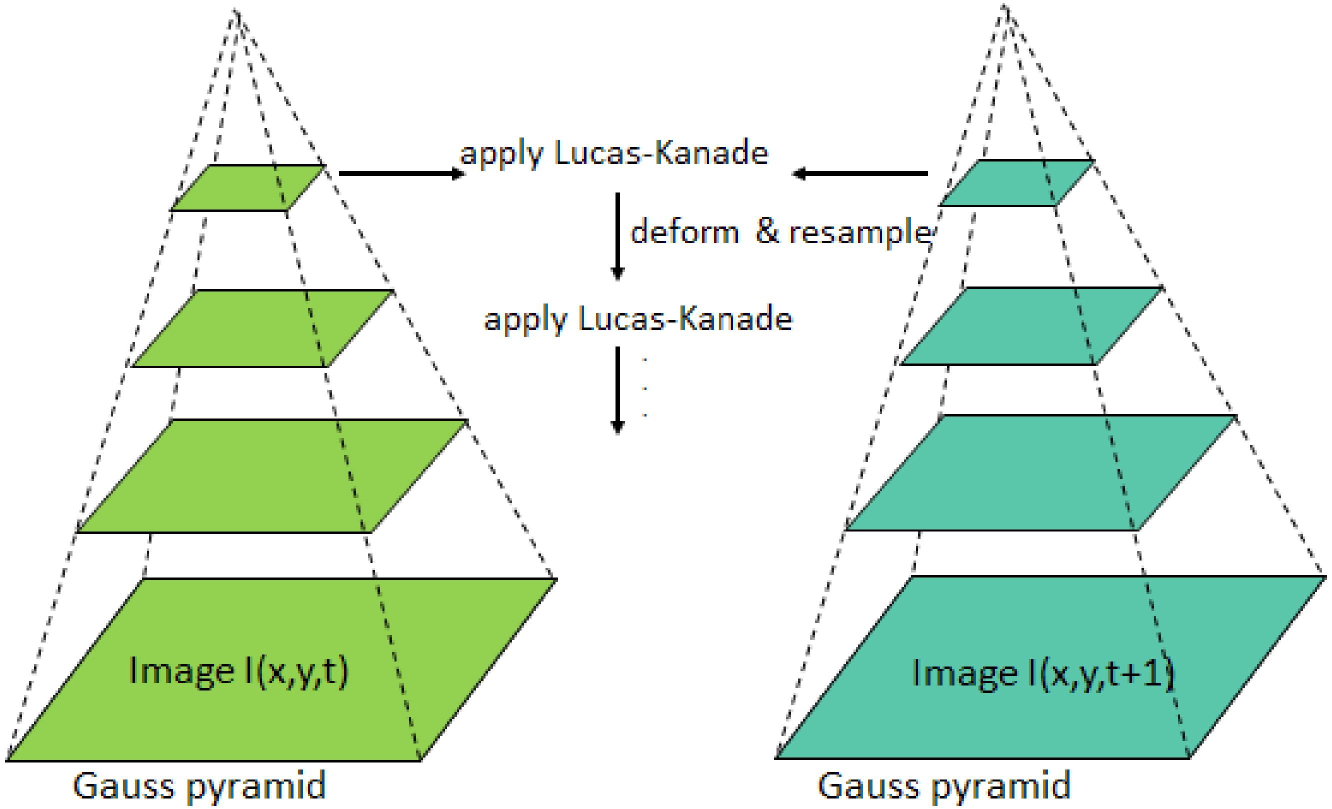
- From one level to the next: smooth image by Gaussian filter and reduce by half



See Szeliski, Sec. 8.1.1. and Sec. 3.5.3.

Improve flow by iterations

- Iteratively solve Lucas Kanade:
  - Calculate rough estimate at low resolution
  - Increase resolution and gradually improve flow estimates



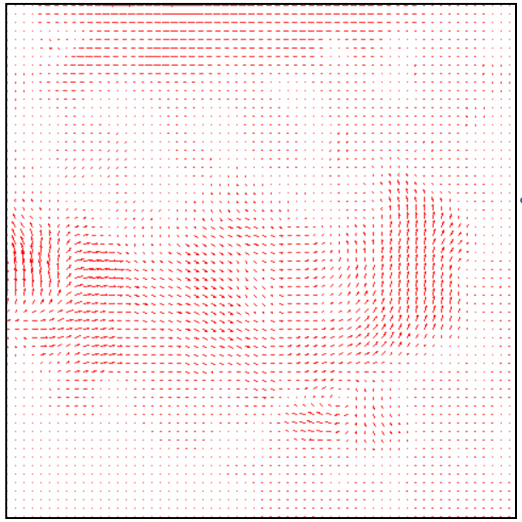
Example of warp/deformation application

Frame t

Frame t+1

note the similarity to Frame t!

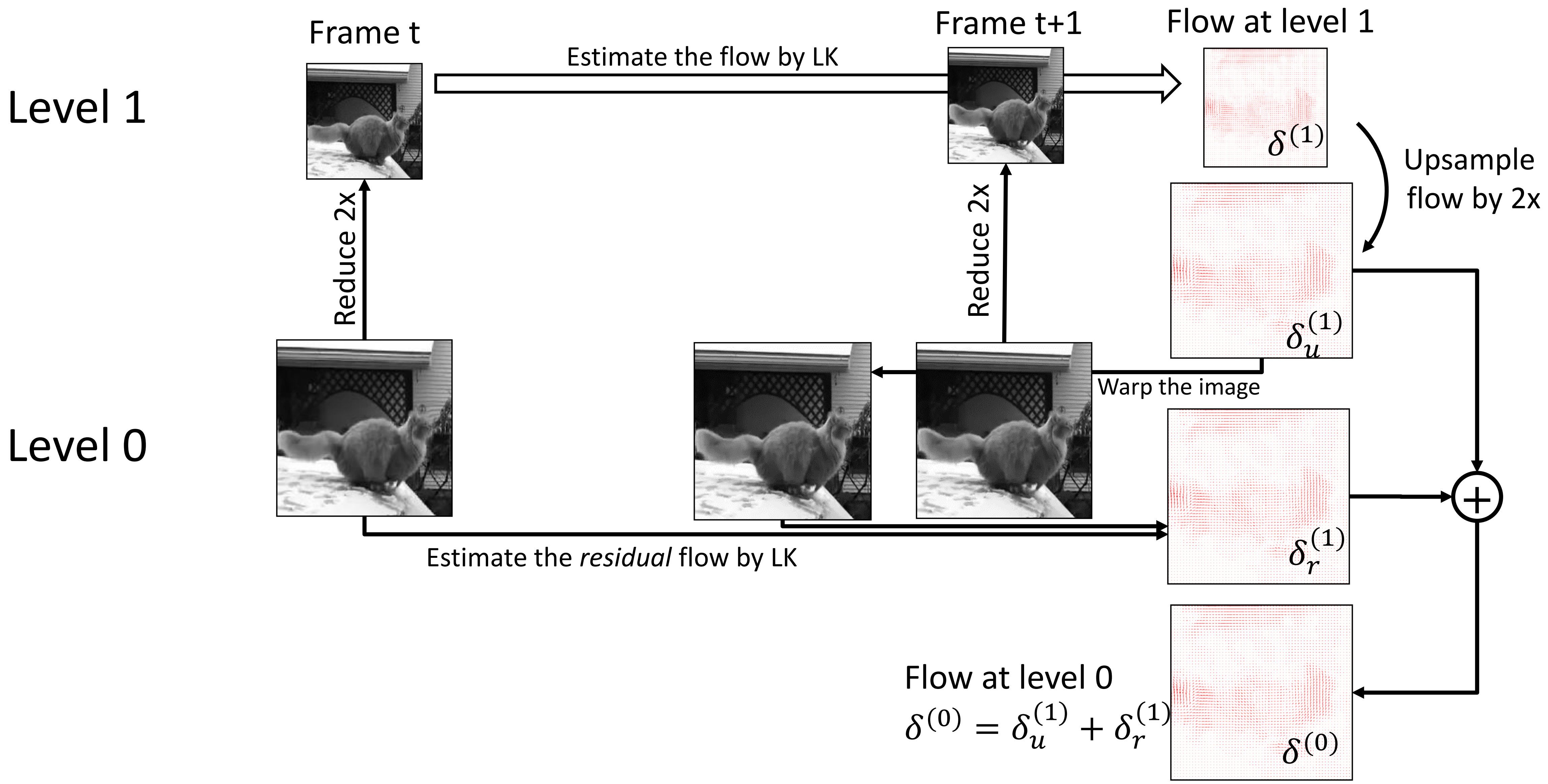
Flow



WARP IMAGE

Warped Frame t+1

Example of using a pyramid with 2 levels



May try to improve the derivative estimates

- Smooth temporal derivative by a small Gaussian:

$$\hat{I}_t = g(x, y) * I_t$$

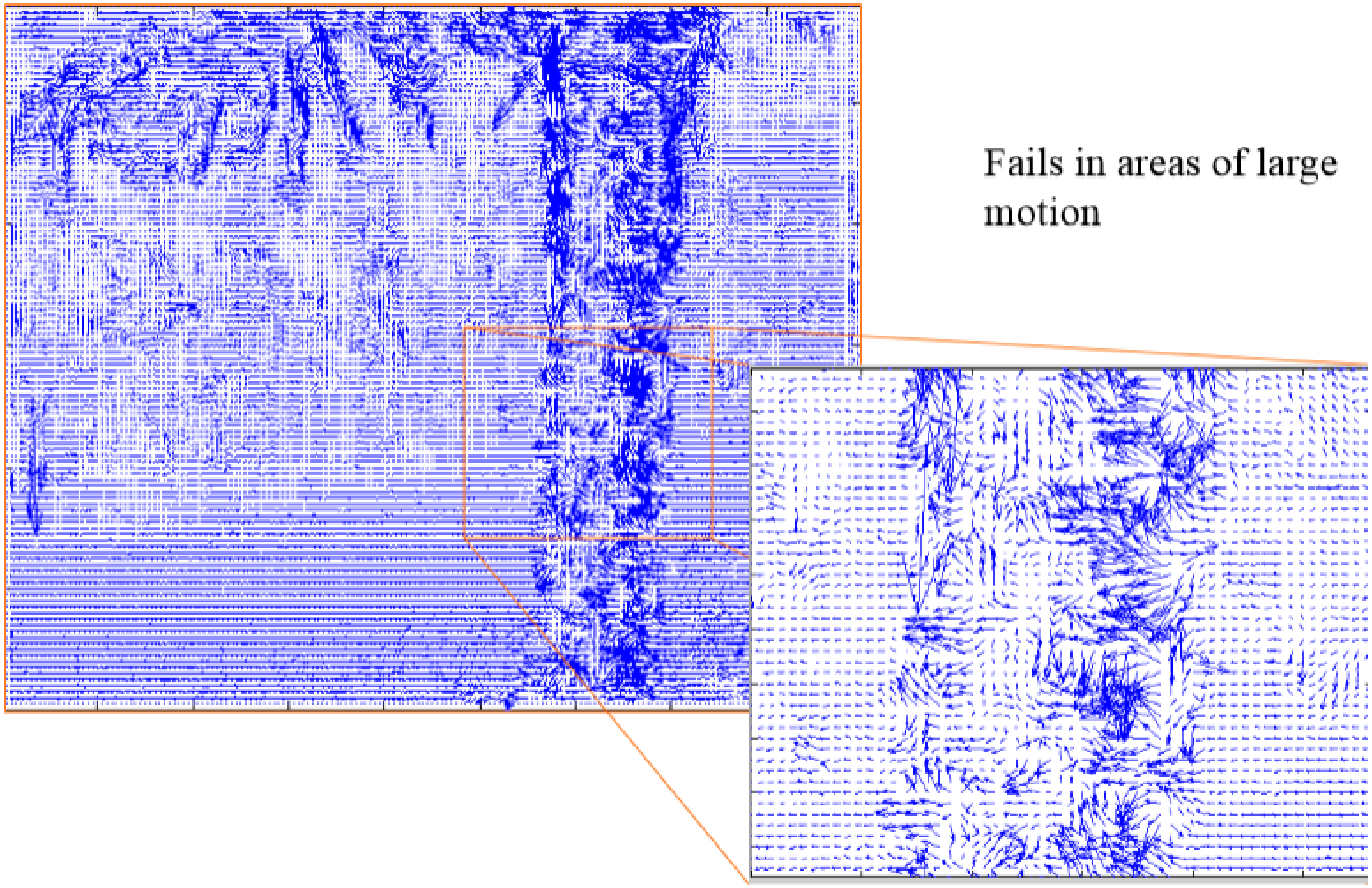
- Average spatial derivative in frame t and t+1:

  (mathematically incorrect, but could help in some situations)

$$\hat{I}_x = \frac{1}{2}(I_x(x, y, t) + I_x(x, y, t+1))$$
$$\hat{I}_y = \frac{1}{2}(I_y(x, y, t) + I_y(x, y, t+1))$$

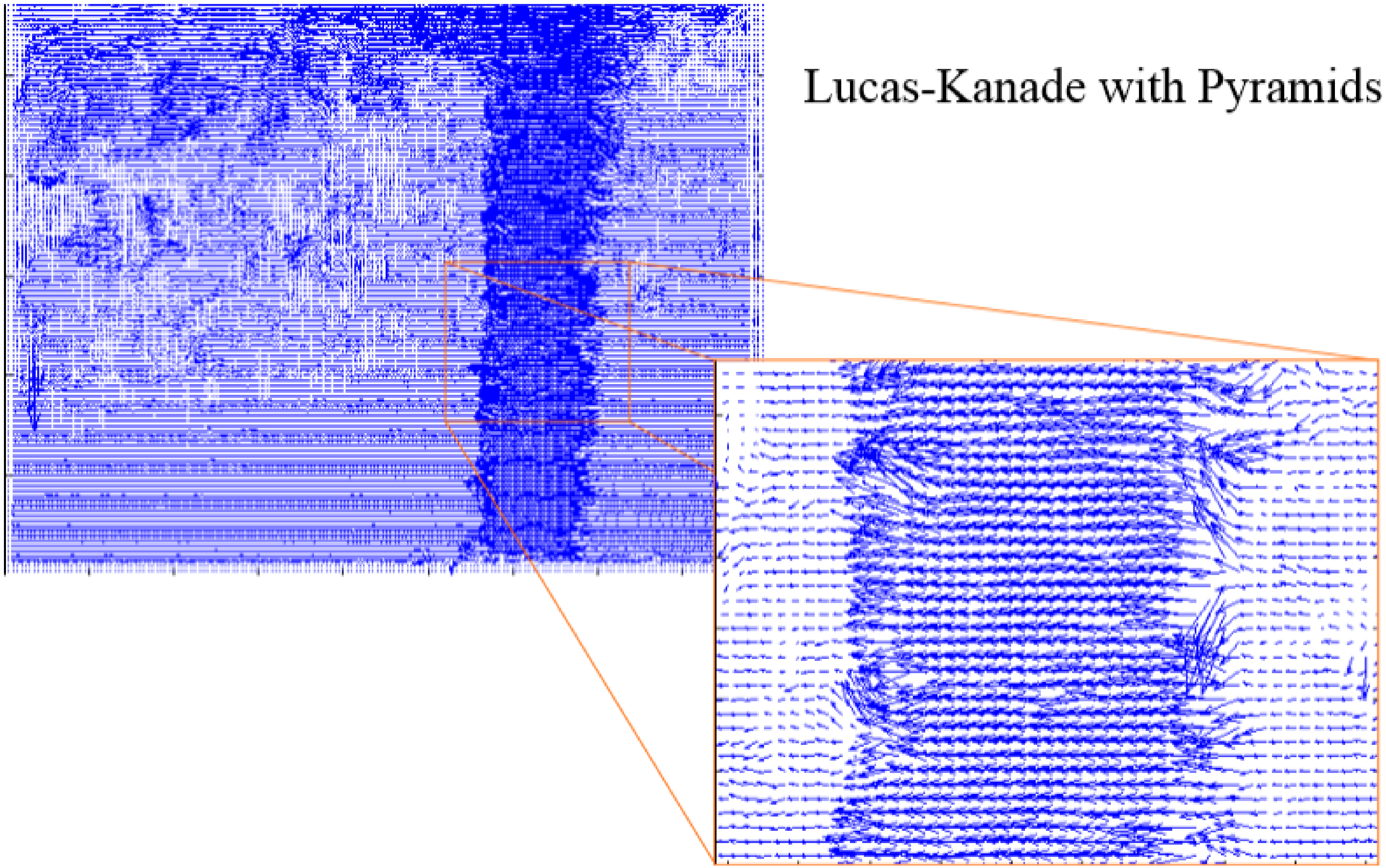
- Iterate between warping and flow estimation at a single level of the pyramid.

Without using the pyramids



Fails in areas of large motion

By using the pyramids



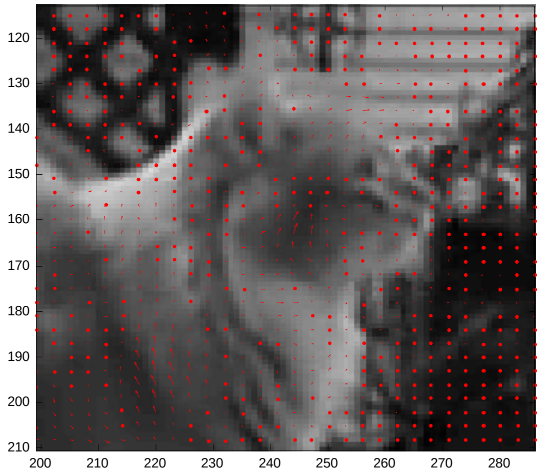
Lucas-Kanade with Pyramids

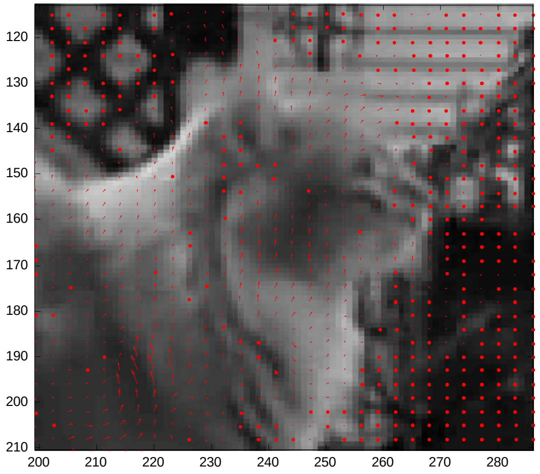
Back to Waffle the terrible

Standard derivatives

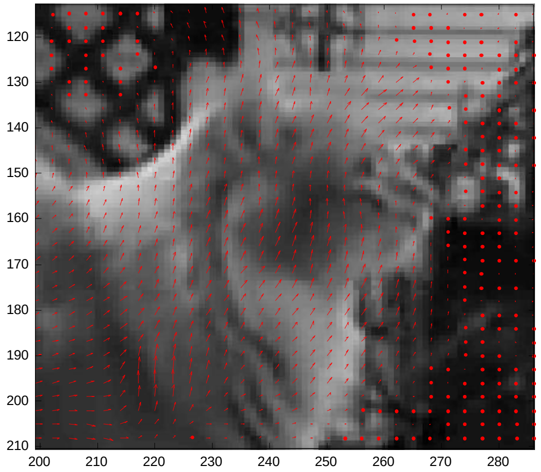
"Improved" i.e., hacked derivatives

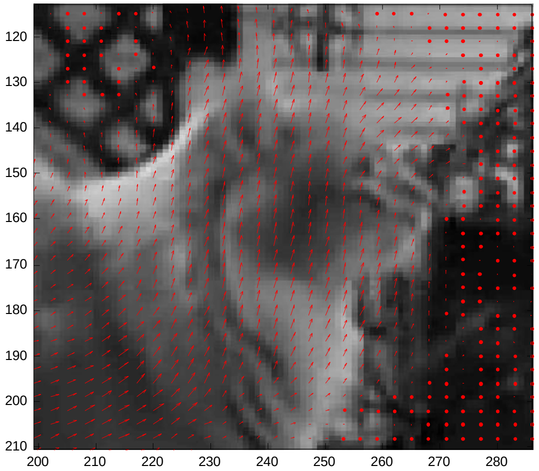
Without pyramid





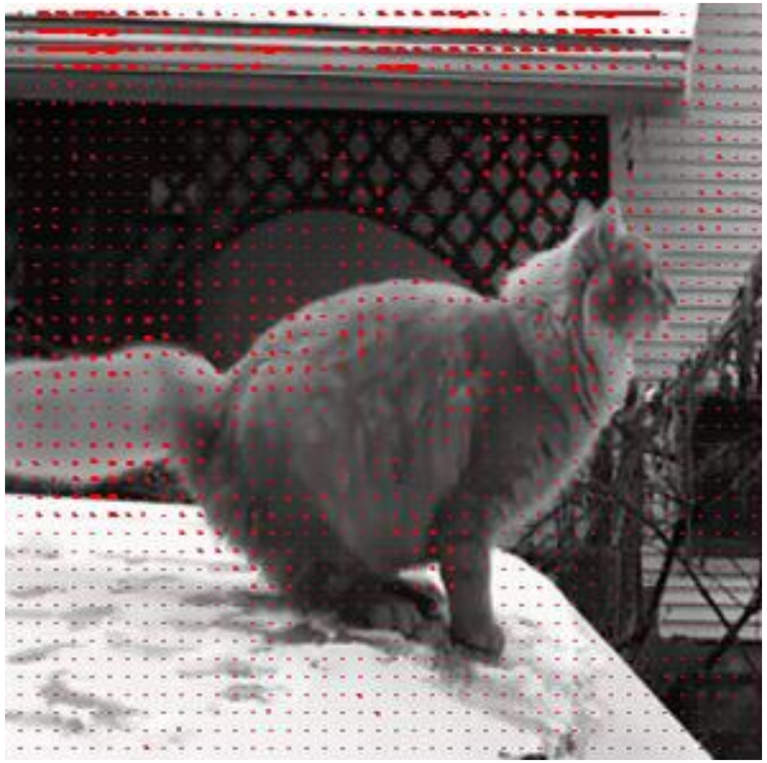
With pyramid



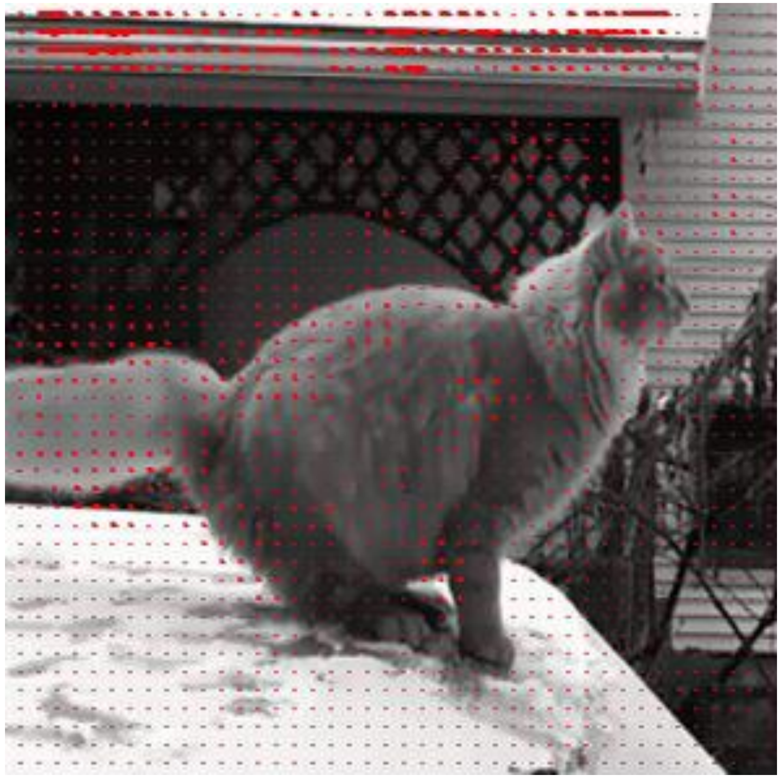


Back to Waffle the terrible

Standard derivatives,
without pyramid



“Improved” derivatives,
with pyramid



Recap on the Lucas Kanade flow

- Brightness constancy assumption:

$$I(\mathbf{x}) = I(\mathbf{x} + \delta)$$

- Small displacement assumption:

$$I(\mathbf{x} + \delta) \approx I(\mathbf{x}) + \nabla I^T \mathbf{J} \delta$$

- Optical flow equation (underdetermined system):

$$I_x(\mathbf{x}_i)\delta_x + I_y(\mathbf{x}_i)\delta_y + I_t(\mathbf{x}_i) = 0$$

- LK solution: neighboring points move similarly, so we can solve for the displacements via least squares.
- Large motions violate the small motion assumption -> Pyramids!
- Pay attention to implementation efficiency

Further info on LK flow estimation

- B.D. Lucas and T. Kanade "*An Iterative Image Registration Technique with an Application to Stereo Vision*" IJCAI '81
  Pay attention to pages: pp. 674-679