# Optic Flow Estimation

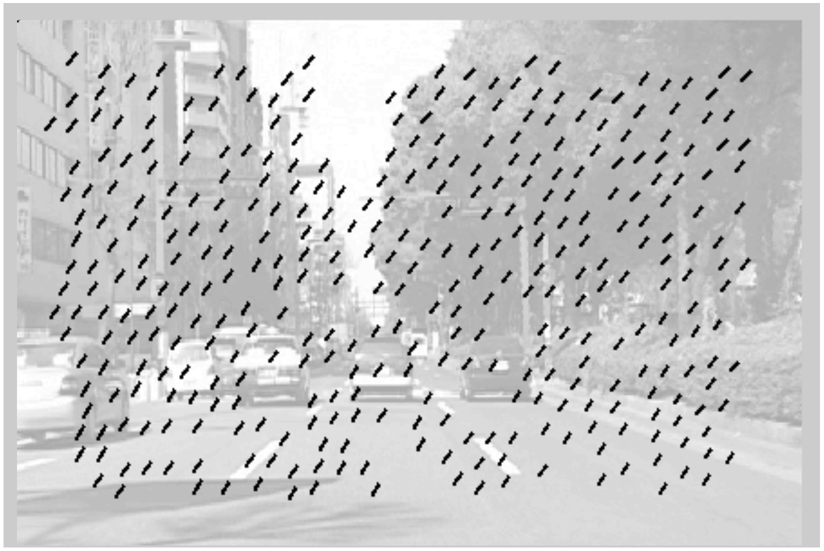
Key points for today:
- Brightness constancy equation
- Aperture problem
- Lucas-Kanade algorithm

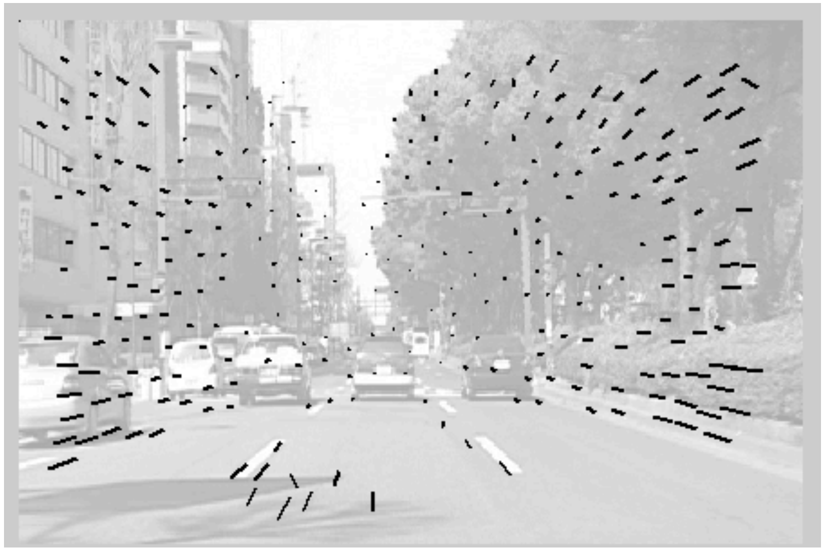
## Review: Flow Due to Self-Motion

Flow:

$$\begin{aligned} u' - u &= \boxed{f\omega_Y - v\omega_Z + \frac{u^2}{f}\omega_Y - \frac{uv}{f}\omega_X} + \boxed{f\frac{T_X}{Z} - u\frac{T_Z}{Z}} \\ v' - v &= \boxed{-f\omega_X + u\omega_Z - \frac{v^2}{f}\omega_X + \frac{uv}{f}\omega_Y} + \boxed{f\frac{T_Y}{Z} - v\frac{T_Z}{Z}} \end{aligned}$$

Rotation Translation


**Rotation component**


**Translation component**

rotation component does not depend on scene structure.

translational component does vary as scene *Z*-value varies. That is, it exhibits motion <u>parallax.</u>

## Motion Field vs Optic Flow

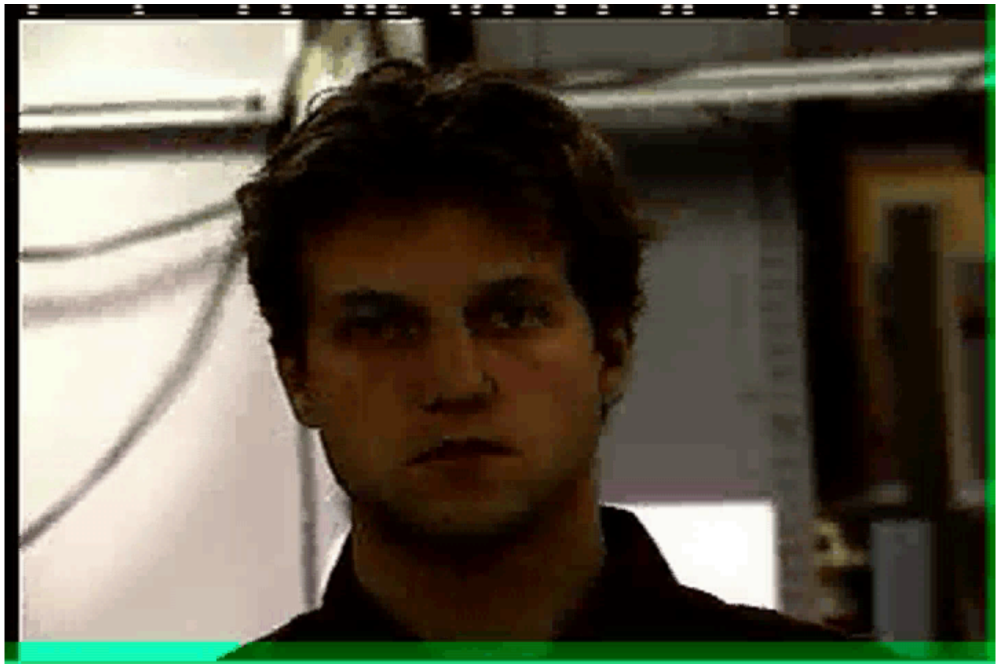
Motion Field: projection of 3D relative velocity vectors onto the 2D image plane.

Optic Flow: observed 2D displacements of brightness patterns in the image.

Motion field is what we want to know.
Optic flow is what we can estimate.

## Optic Flow ≠ Motion Field

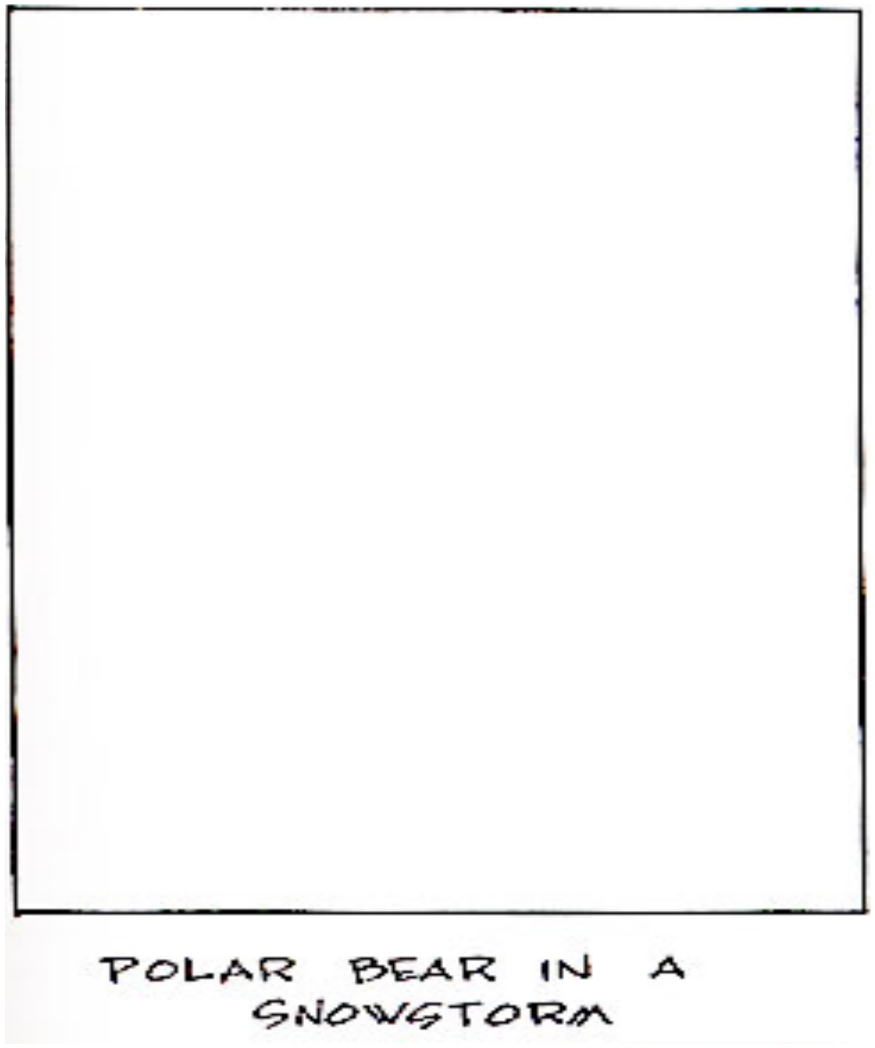
Consider a moving light source:



MF = 0 because points on the scene are not moving

OF ≠ 0 because there is a moving pattern in the images

# Optic Flow ≠ Motion Field



MF ≠ 0 because the bear is charging right at you!!!!

OF = 0 because you can't see it

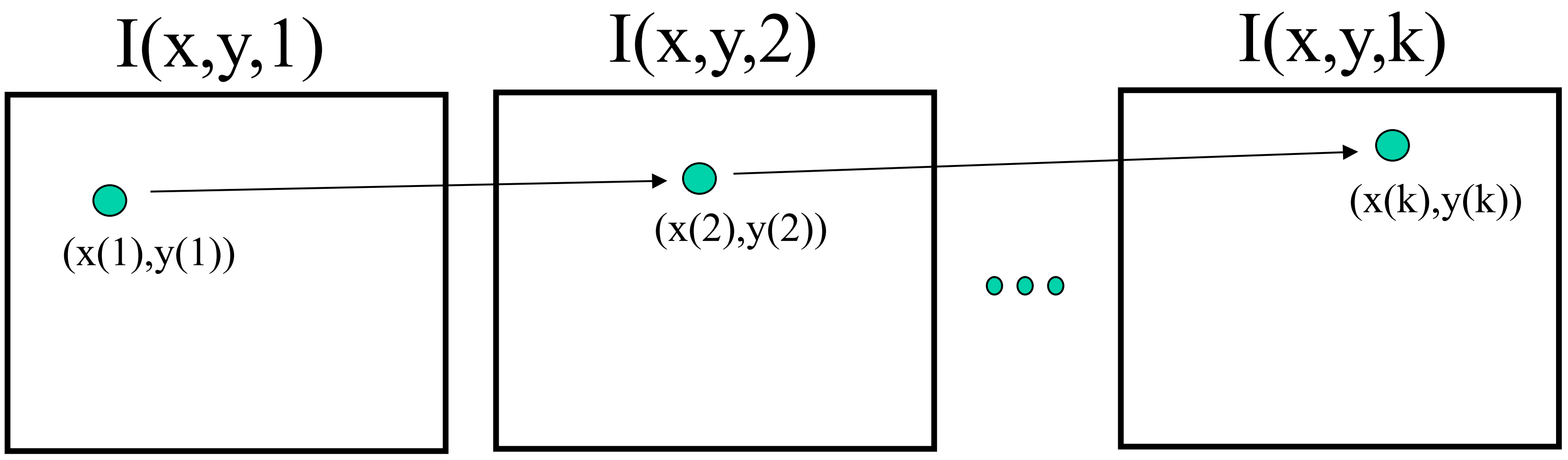
## Approximating MF with OF

Nevertheless, we will estimate OF (since MF cannot really be observed!).

To avoid apparent flow due to changing illumination, <u>assume that the apparent brightness of moving objects remains constant.</u>

## Brightness Constancy Equation

consider a scene point moving through an image sequence



claim: its brightness/color will remain the same (that's partly how we can recognize that it IS the same point)

$$\boxed{I\left(x(t), y(t), t\right) = Constant}$$

## Brightness Constancy Equation

$$I\left(x(t), y(t), t\right) = Constant$$

Take derivative of both sides wrt time:

$$\frac{d\, I\left(x(t), y(t), t\right)}{dt} = 0$$

(using chain rule)

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

# Brightness Constancy Equation

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

$\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$ (spatial gradient; we can compute this!)

$\frac{dx}{dt}, \frac{dy}{dt}$ = (u, v) (optic flow, what we want to find)

$\frac{\partial I}{\partial t}$ (derivative across frames. Also known, e.g. frame difference)

## Brightness Constancy Equation

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

Becomes:

$$\boxed{(\nabla I)^T . v + I_t = 0}$$

Optic Flow is CONSTRAINED to be on a line !
(equation a $u$ + b $v$ + c = 0)

What is the practical implication of this?

## Brightness Constancy Equation

$$\boxed{I_x}\,\boxed{u} + \boxed{I_y}\,\boxed{v} + \boxed{I_t} = 0$$

<u>Known</u> (spatial and temporal gradients)

Unknowns (components of flow vector)

Optic Flow solution is CONSTRAINED to be on a line (equation says $a\,u + b\,v + c = 0$)

What is the practical implication of this?

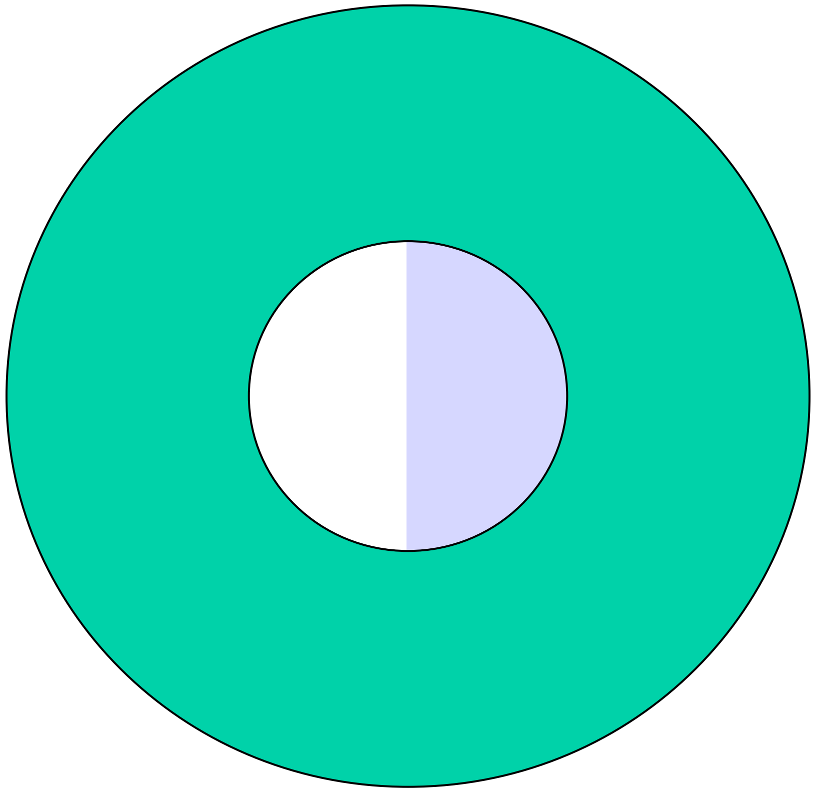
## Implications

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

- Q: how many unknowns and equations per pixel?

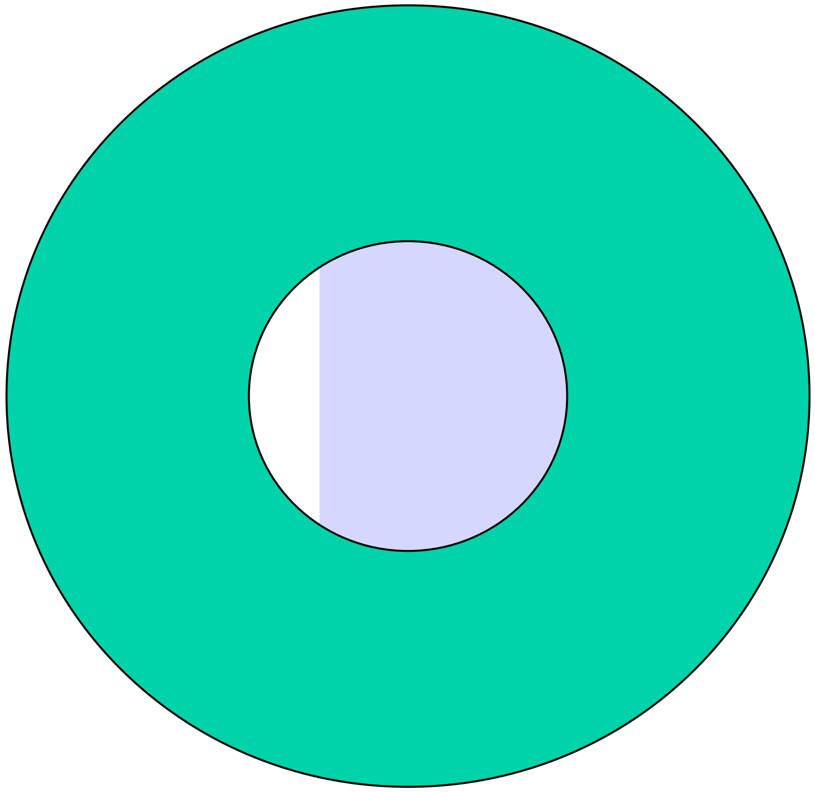
- Intuitively, this constraint means that
  - The component of the flow in the gradient direction is determined **(called Normal Flow)**
  - The component of the flow parallel to an edge is unknown

## The Aperture Problem

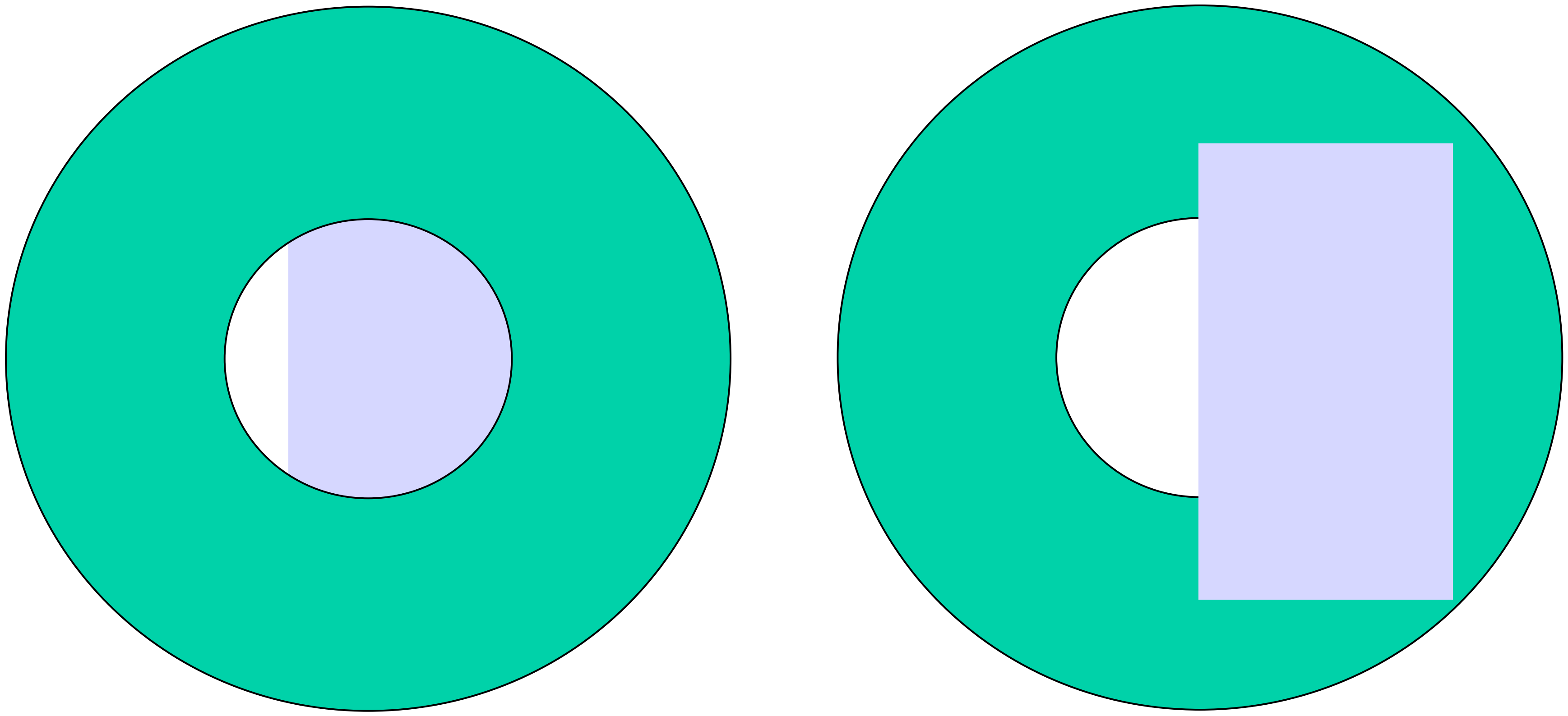


Camps, PSU

## The Aperture Problem

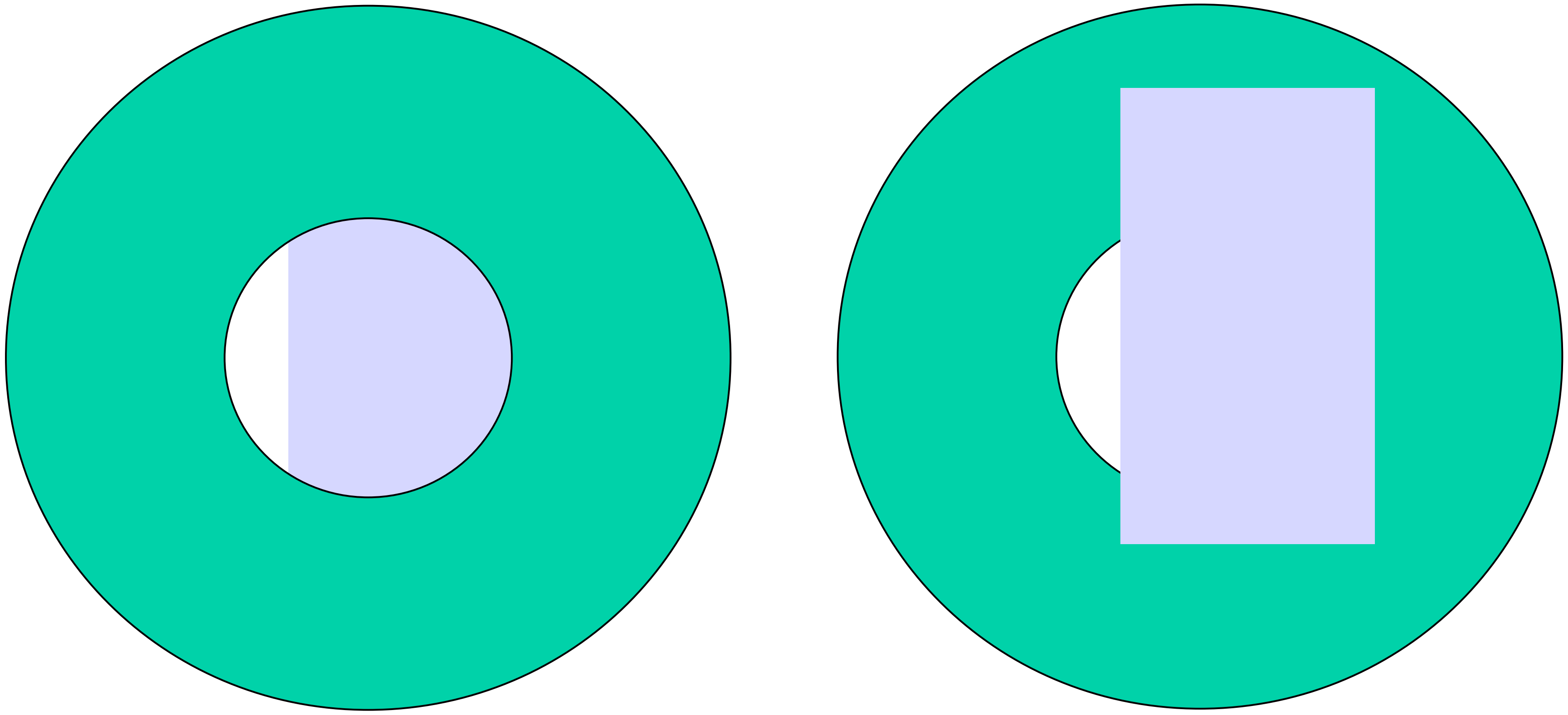


# The Aperture Problem



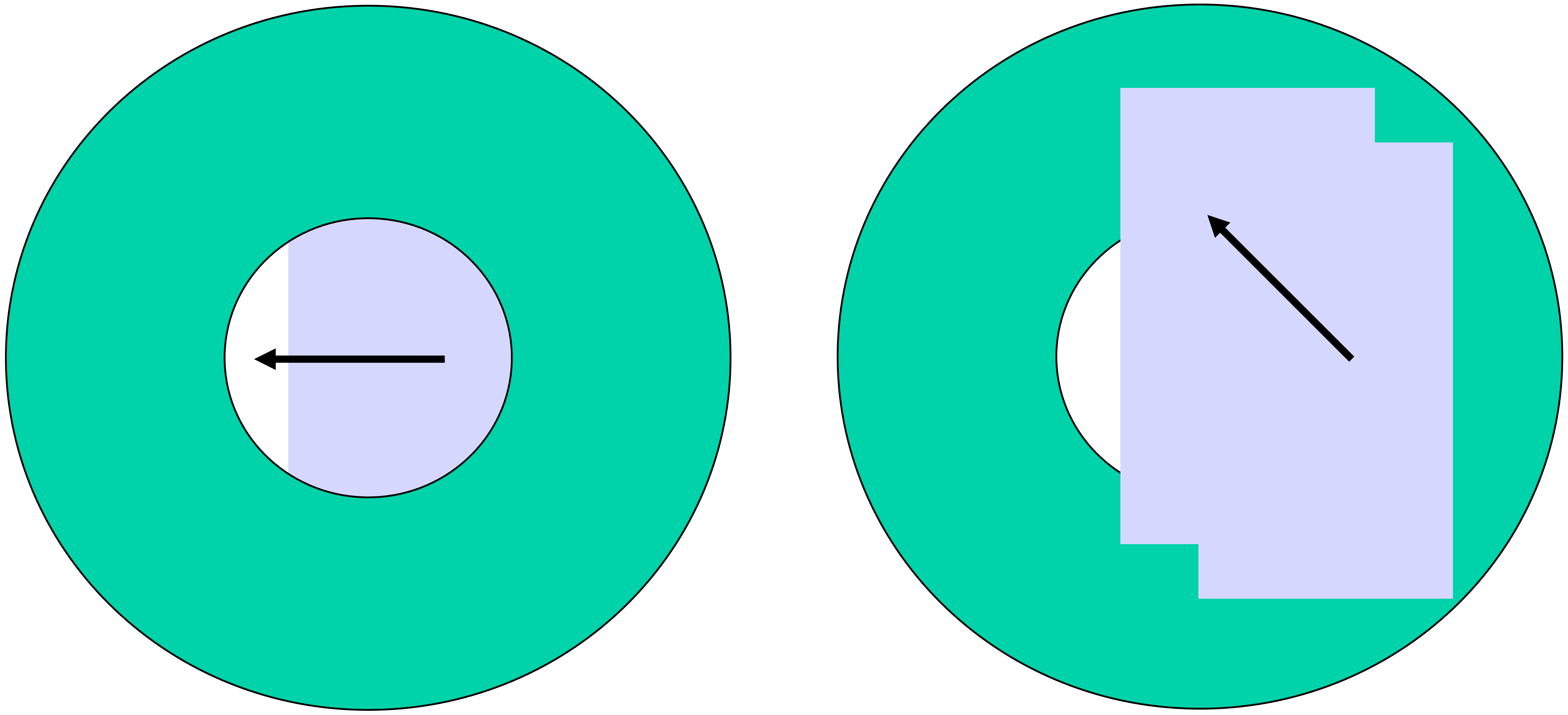
Camps, PSU

## The Aperture Problem



Camps, PSU

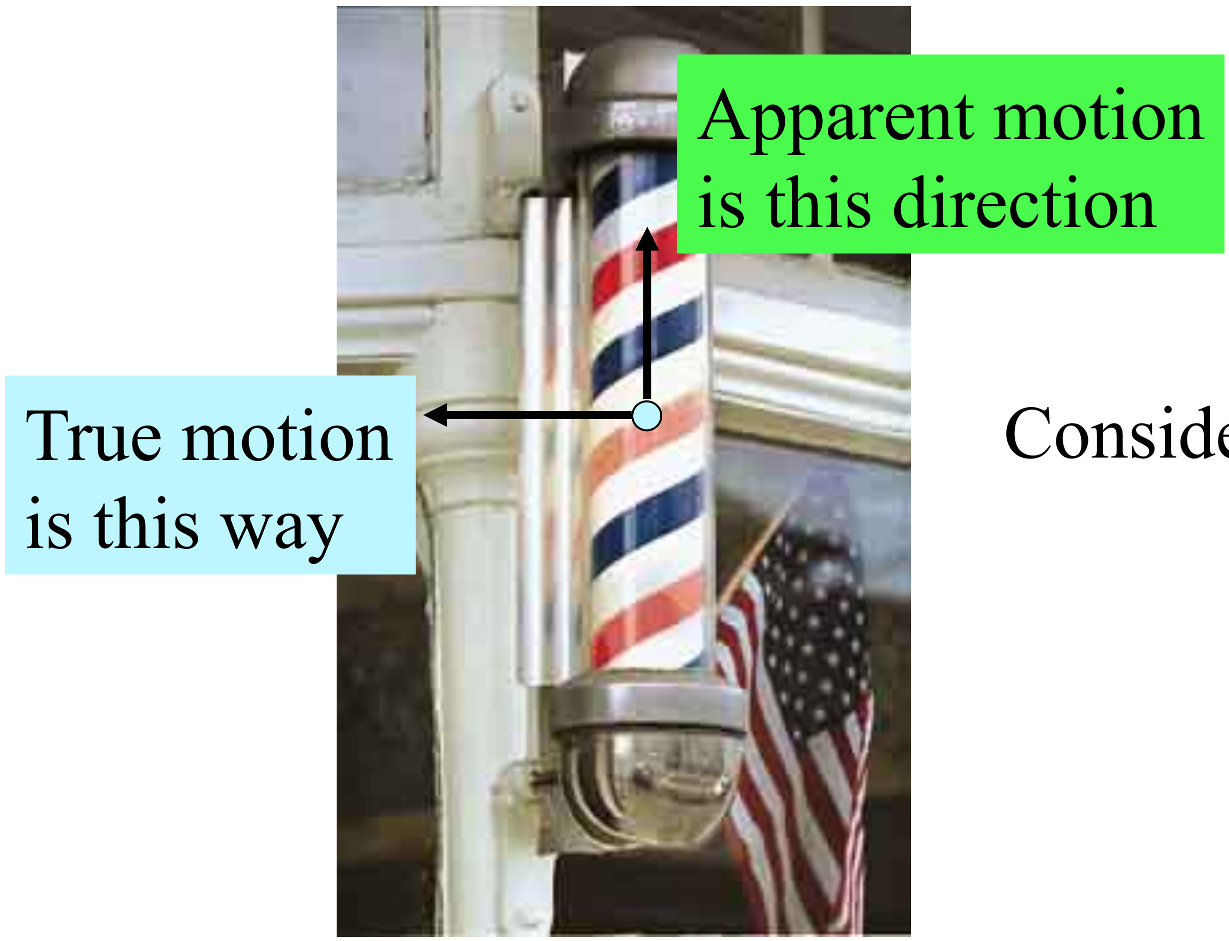
## The Aperture Problem



The Image Brightness Constancy Assumption only provides the Optic Flow component in the direction of the spatial image gradient

## Aperture Problem

- Another example is the barber-pole illusion.



Consider a point

## Optic Flow ≠ Motion Flow

- The aperture problem reminds us once again that optic flow is not the same thing as motion flow.
  - – near an edge, we can only observe (measure) the component of flow perpendicular to the edge
  - – cannot measure the component of flow parallel to the edge.
  - – another case of non-observability of optic flow is in areas of constant intensity. No flow is observed.
  - – and don't forget: when illumination is changing you can observe non-zero flow, even when motion is really zero!

## Computing Optic Flow

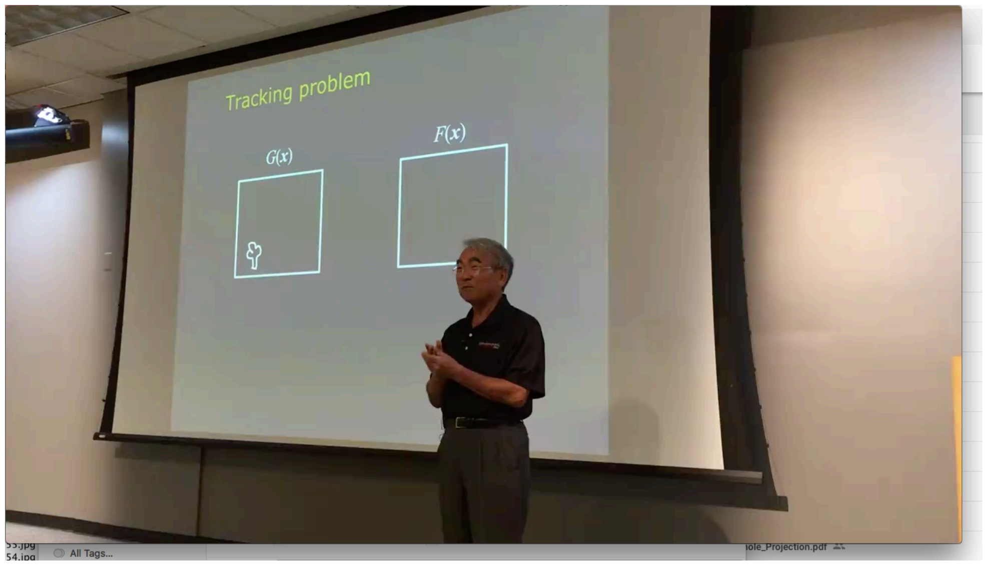
- Algorithms for computing OF are of two types:
  - Differential Techniques
    - Based on spatial and temporal variations of the image brightness at all pixels.
    - Used to compute DENSE flow.
  - Matching Techniques
    - Similar to stereo feature matching, compute disparities.
    - Used to compute SPARSE flow.

## A Differential Technique: Constant Flow, aka Lucas-Kanade

## Aside

- I’ve posted a short video of Takeo Kanade discussing how the Lucas-Kanade algorithm was developed. He’s a charming speaker; take a look if you are interested.

https://drive.google.com/open?id=0B1U0n0n5uFhpNFRWVkpGakhjbG8



## Lucas-Kanade Motivation

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

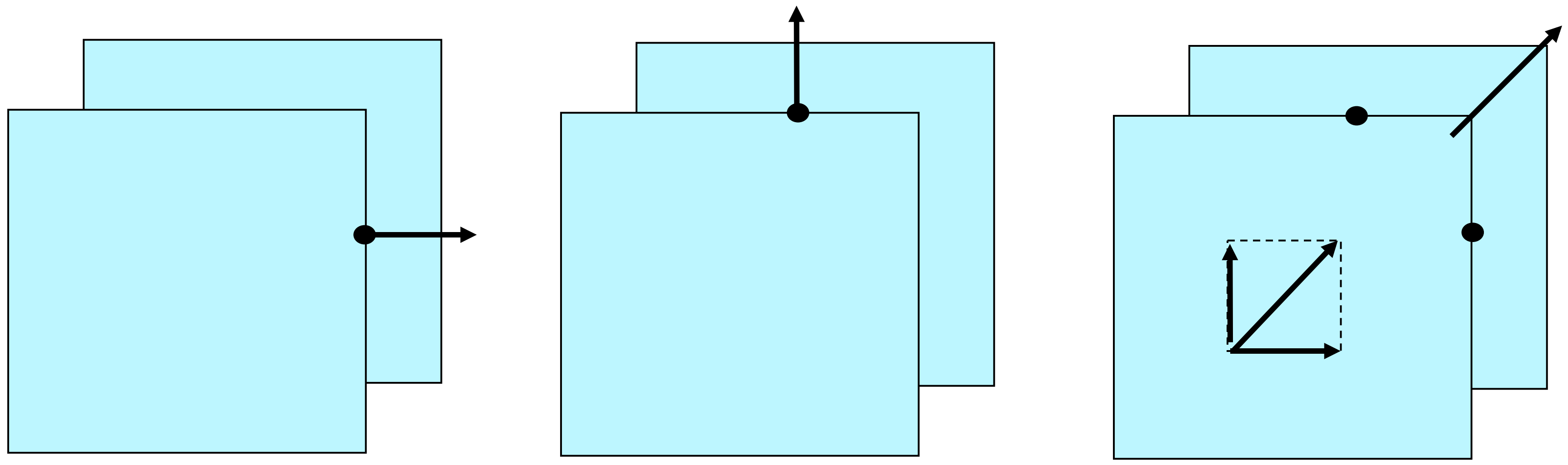
- Q: how many unknowns and equations per pixel?

- We need two or more pixels to solve.

- Due to aperture problem, we would like to include pixels with different gradient directions.

## Solving the Aperture Problem

- Using gradients with two or more directions



aperture problem with single gradient direction

can infer correct solution using two or more directions

Robert Collins
CMPEN454

## Solving the aperture problem

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

- How to get more equations for a pixel?
  - Basic idea: impose additional constraints
    - most common is to assume that the flow field is smooth locally
    - e.g. pretend the pixel's neighbors have same displacement (u,v)
      - If we use a 5x5 window, that gives us 25 equations per pixel!

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

Seitz, UW

## Lucas-Kanade flow

- We have more equations than unknowns: solve least squares problem. This is given by:

$$A \quad d = b \longrightarrow \text{minimize } \|Ad - b\|^2$$

$$(A^T A) \; d = A^T b$$

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

– Summations over all pixels in the KxK window

Seitz, UW

## Conditions for solvability

– Optimal (u, v) satisfies Lucas-Kanade equation

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

### When is This Solvable?

- **$A^T A$** should be invertible
- **$A^T A$** should not be too small due to noise
  - eigenvalues $\lambda_1$ and $\lambda_2$ of **$A^T A$** should not be too small
- **$A^T A$** should be well-conditioned
  - $\lambda_1 / \lambda_2$ should not be too large ($\lambda_1$ = larger eigenvalue)

## Does $A^T A$ seem familiar?

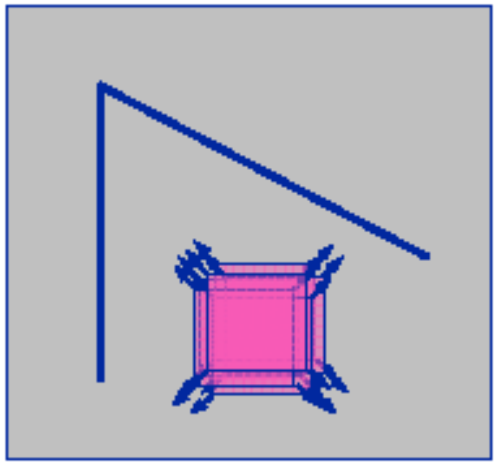
Look at the matrix:

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$$

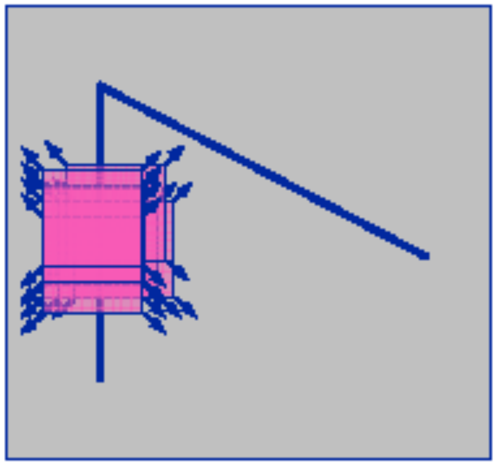
Harris Corner
Detector Matrix!

## Review: Harris Corner Detector

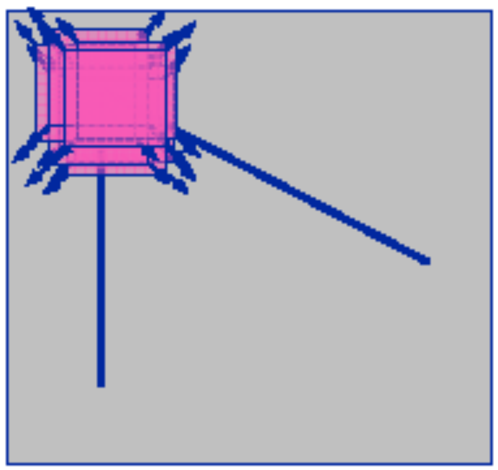
Earlier in the course we derived the Harris corner detector by considering SSD of shifted intensity patches…



"flat" region:
no change in
all directions

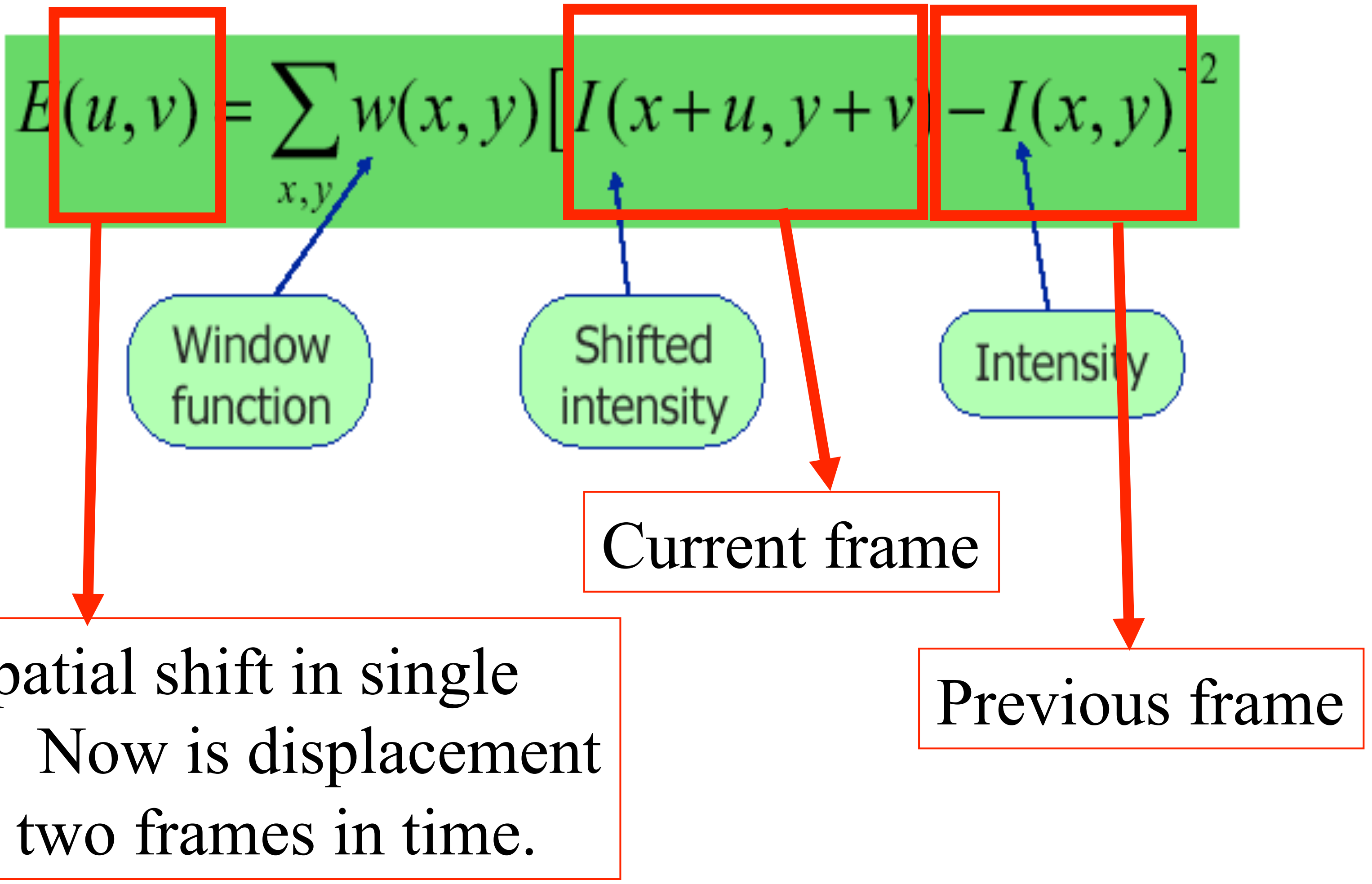


"edge":
no change along
the edge direction

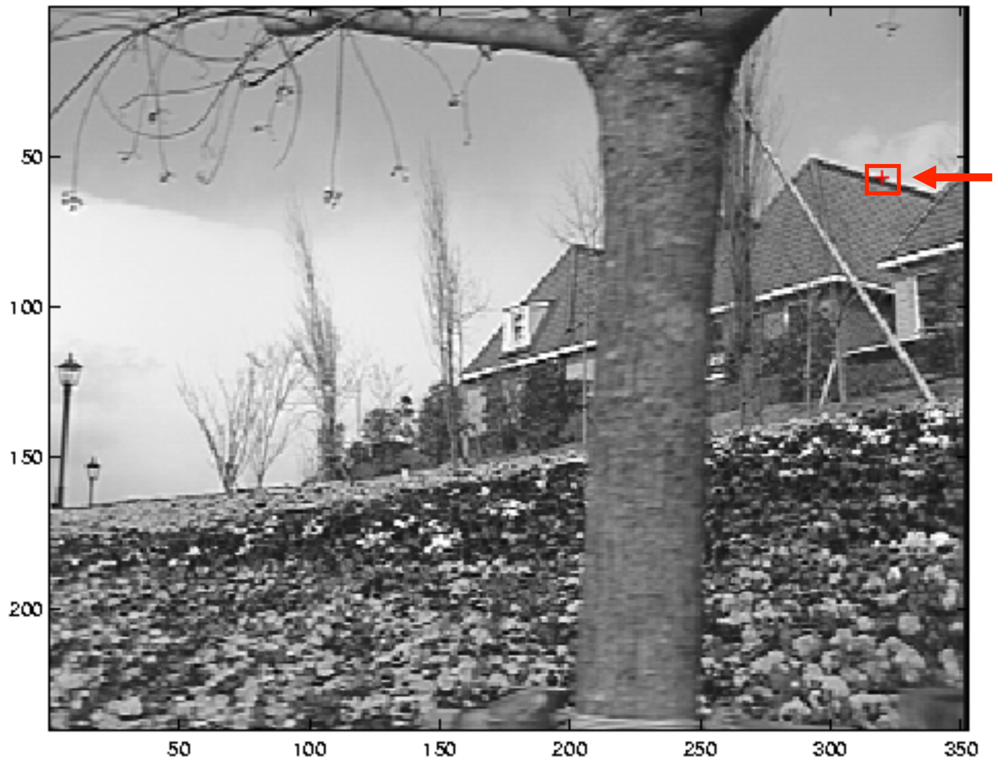


"corner":
significant change
in all directions

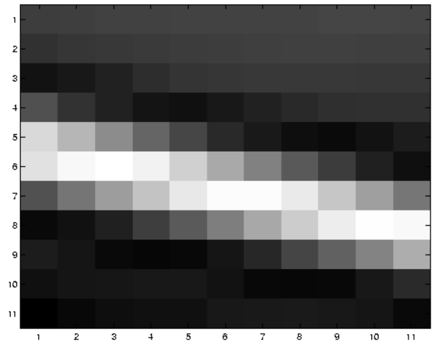
# Tie in Harris with Motion Analysis

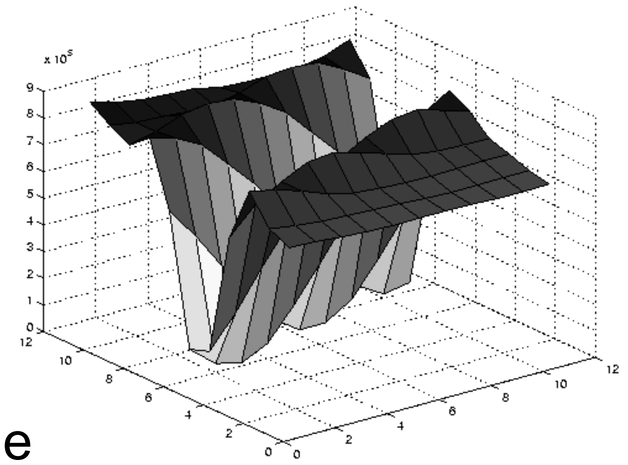

$E(u,v)=\sum_{x,y} w(x,y)\left[I(x+u,y+v)-I(x,y)\right]^2$
Window function
Shifted intensity
Intensity
Current frame
Previous frame
Was spatial shift in single frame. Now is displacement across two frames in time.

# Tie in with Harris



Edge





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

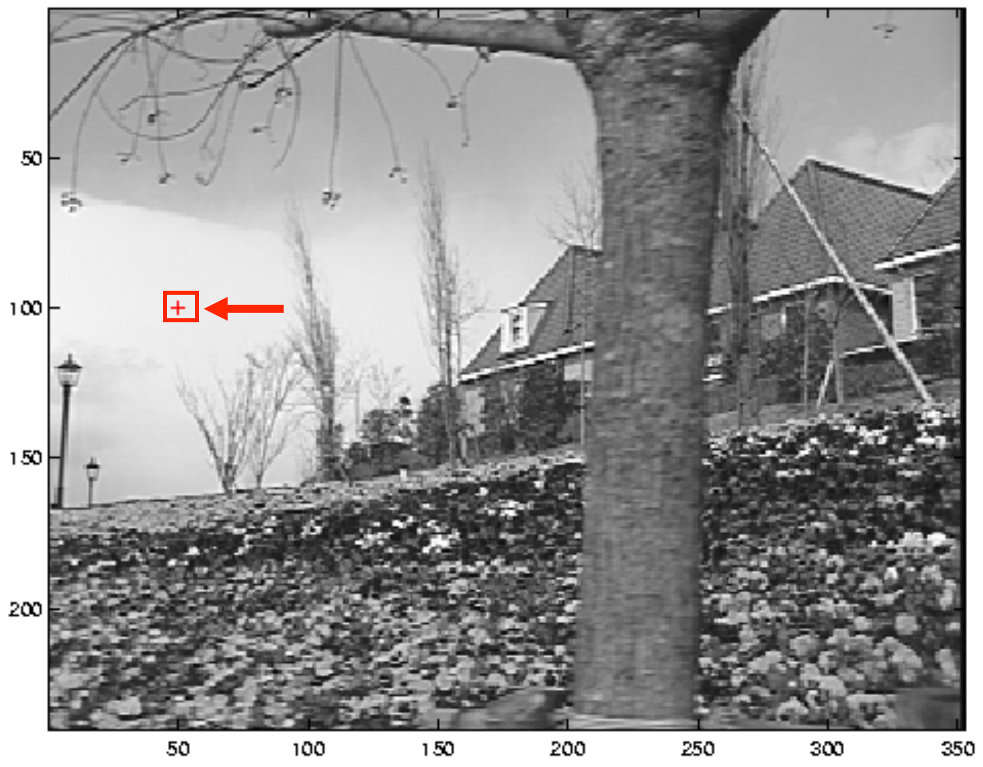
– large gradients, all the same

– large $\lambda_1$, small $\lambda_2$

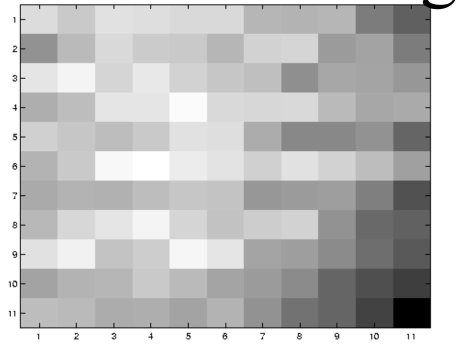
**Will have aperture problem**

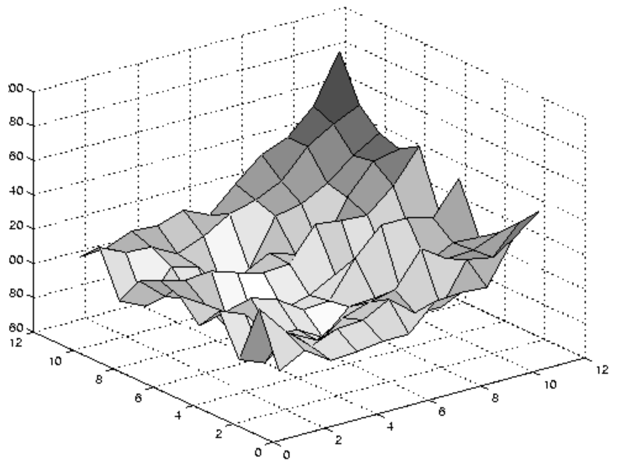
Seitz, UW

## Tie in with Harris



Low Texture Region





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

– gradients have small magnitude

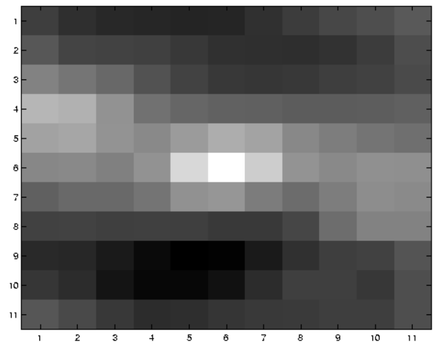
– small $\lambda_1$, small $\lambda_2$

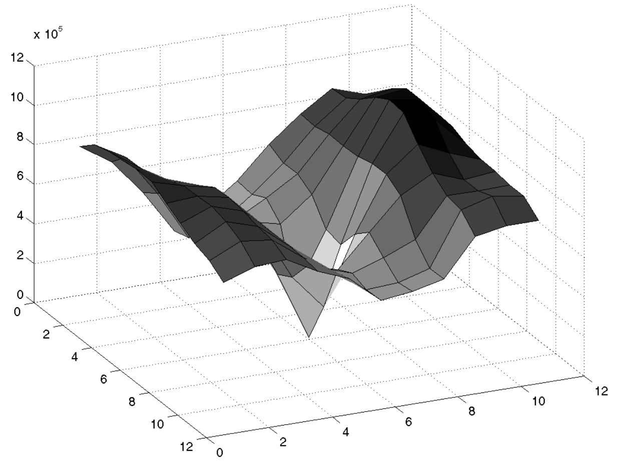
**Ill-conditioned matrix. Likely to compute garbage answer.**

Seitz, UW

# Tie in with Harris

Corner





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

– gradients are different, large magnitudes

– large $\lambda_1$, large $\lambda_2$

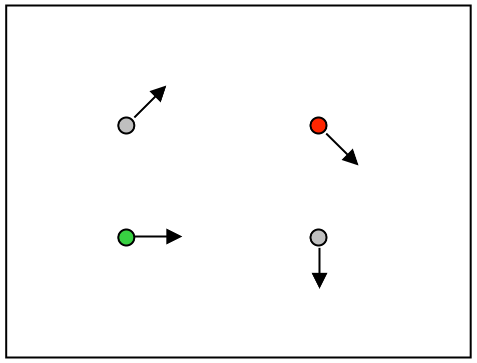
**Good corner feature. Also good place to estimate flow!**

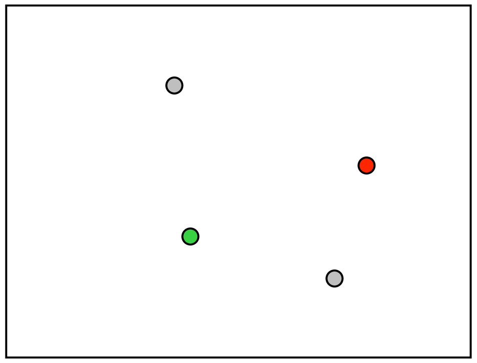
Seitz, UW

## Implications

- Corners are when $\lambda 1$, $\lambda 2$ are big; this is also when Lucas-Kanade works best.
- Corners are regions with two different directions of gradient (at least).
- Aperture problem disappears at corners.
- Corners are good places to compute flow!

## Feature Matching for Sparse Flow


I(x,y,t)


I(x,y,t+1)

General idea:
 Find Corners in one image (because these are good areas to estimate flow)
 Search for Corresponding intensity patches in second image.

## KLT Algorithm

Public-domain code by Stan Birchfield.
Available from

http://www.ces.clemson.edu/~stb/klt/

Tracking corner features through 2 or more frames

# KLT Algorithm

1. Find corners in first image
2. Extract intensity patch around each corner
3. Use Lucas-Kanade algorithm to estimate constant displacement of pixels in patch

Niceties

1. Iteration and multi-resolution to handle large motions
2. Subpixel displacement estimates (bilinear interp warp)
3. Can track feature through a whole sequence of frames
4. Ability to add new features as old features get “lost”

## Correlation-Based Matching

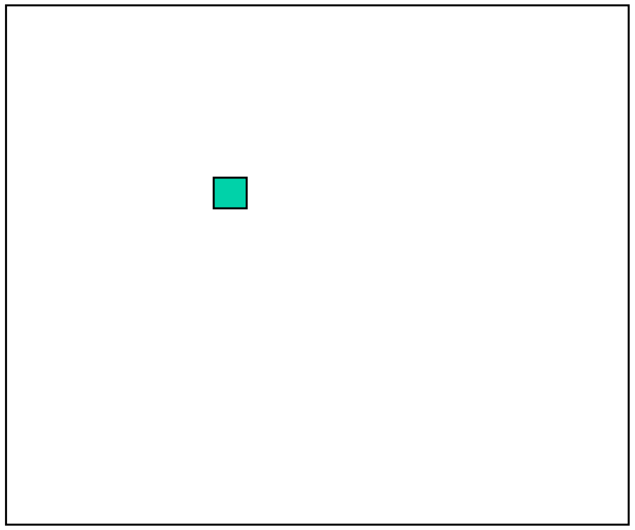
- Another approach is to match intensity patches using correlation or NCC.
  - We talked about this for stereo as well:

## Correlation-based Matching

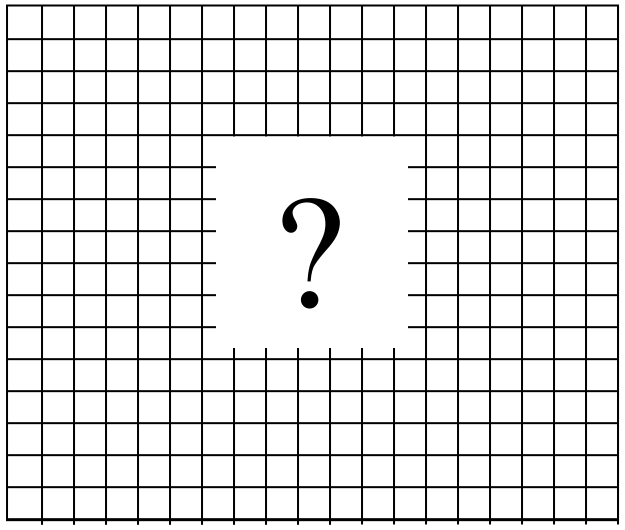
1. Find corners in first image
2. Extract intensity patch around each corner
3. Use NCC to compute match scores within
   the search window in second image
4. Identify highest score (best match)

# Important Note on Efficiency

Given image patch in one image



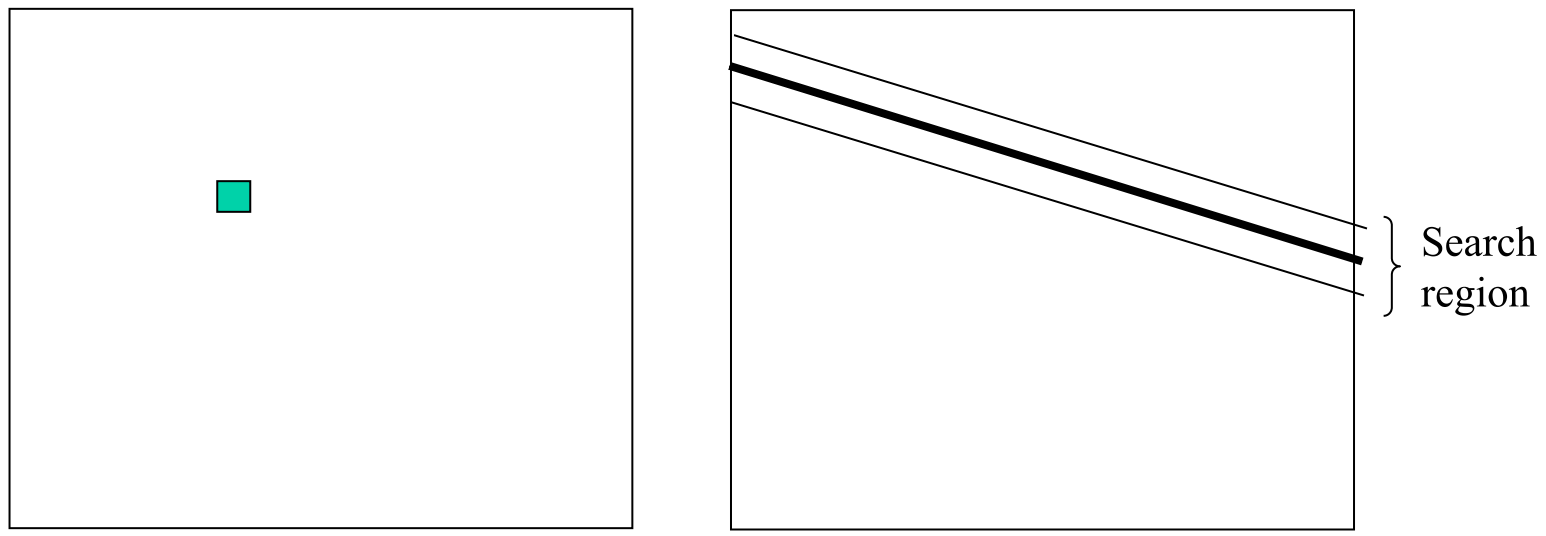
We don’t want to search everywhere in the second image for a match.



# Important Note on Efficiency

Given image patch in one image

We don't want to search everywhere in the second image for a match.

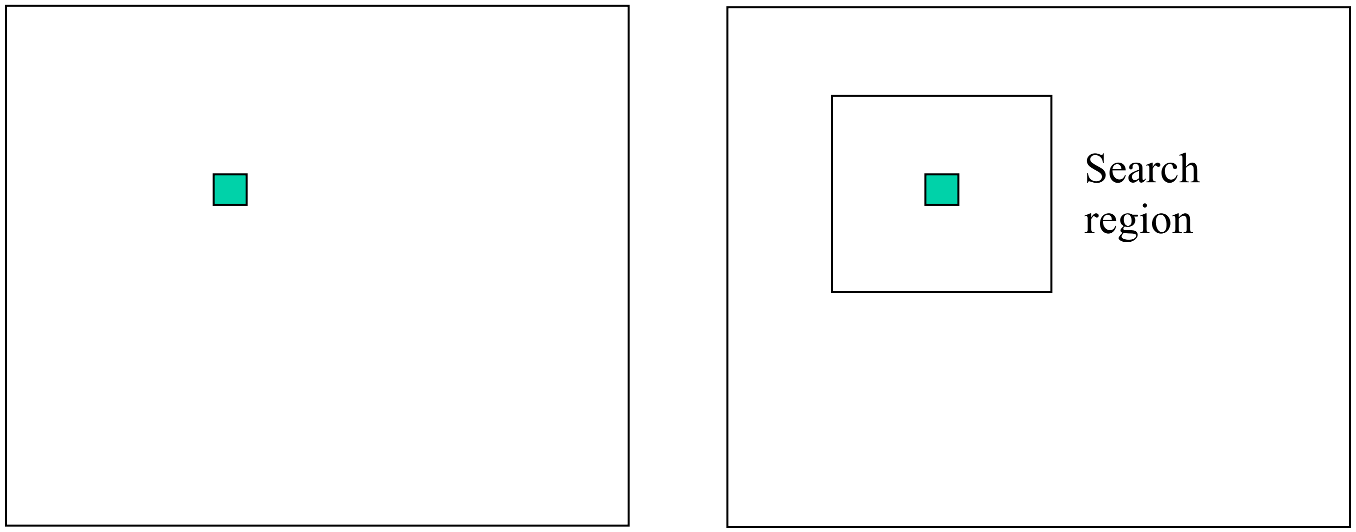


With stereo, we had an epipolar line constraint.

# Important Note on Efficiency

Given image patch in one image

We don't want to search everywhere in the second image for a match.



We don't know the relative R,T here, so no epipolar constraint.
But… motion is known to be "small", so can still bound the search region.

## Note on using Normalized Correlation

If we use normalized correlation to match feature patches, we can relax the constant brightness assumption!

We thus remove some potential error sources
(changes in illumination or camera gain)

## Another Correlation-based Algorithm

Due to David Nister, “Visual Odometry”, CVPR 2004

Observation: corners in one image tend to stay corners over short periods of time.

Therefore, we only need to match corners to corners.

## Nister's Algorithm

1. Find corners in first image
2. Extract intensity patch around each corner
3. Find corners in second image
4. Extract intensity patch around each of them
5. Find matching pairs (c1,c2) such that
    - C1 is a corner patch from image 1
    - C2 is a corner patch from image 2
    - C2 is the best match for C1
    - C1 is the best match for C2

## Nister’s Algorithm

1. Find corners in first image
2. Extract intensity patch around each corner
3. Find corners in second image
4. Extract intensity patch around each of them
5. Find matching pairs (c1,c2) such that
   - C1 is a corner patch from image 1
   - C2 is a corner patch from image 2
   - C2 is the best match for C1
   - C1 is the best match for C2

Question: Why do this too?
Answer: Approx solution to the linear assignment problem -- you get better matches.

## Linear Assignment Problem

Aka the "Marriage Problem"

Simplest form:
1) given k boys and k girls
2) ask each boy to rank the girls in order of desire
3) ask each girl to also rank order the boys
4) The marriage problem determines the pairing of boys and girls to maximize sum of overall pairwise rankings.

Note: in general you may not get your most desirable spouse, but on average you rarely get the least.

## Linear Assignment Problem

The optimal solution to the LAP is too computationally intensive for real-time feature matching/tracking.

Therefore, we use a heuristic solution where features can pair up only if each is the best match for the other (i.e. boy-girl pairs that each listed each other as #1)

This throws away a lot of potential point matches, but the ones that are left are usually pretty good.