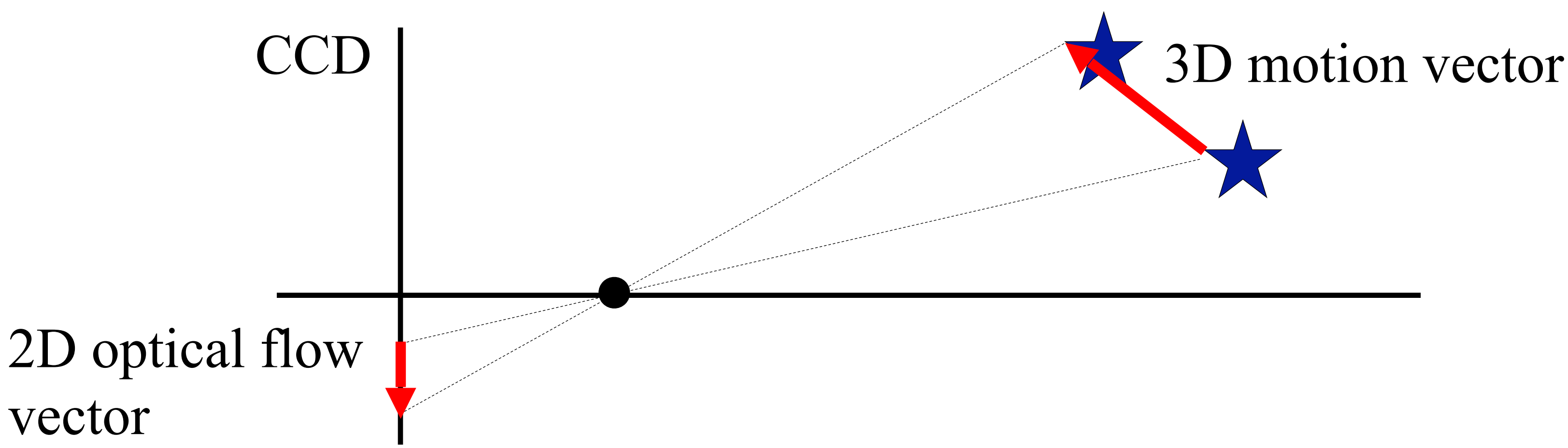
CSE 152: Computer Vision

Hao Su

Lecture 18: Lucas-Kanade Algorithm

Motion Field & Optical Flow Field

- Motion Field = Real world 3D motion
- Optical Flow Field = Projection of the motion field onto the 2d image

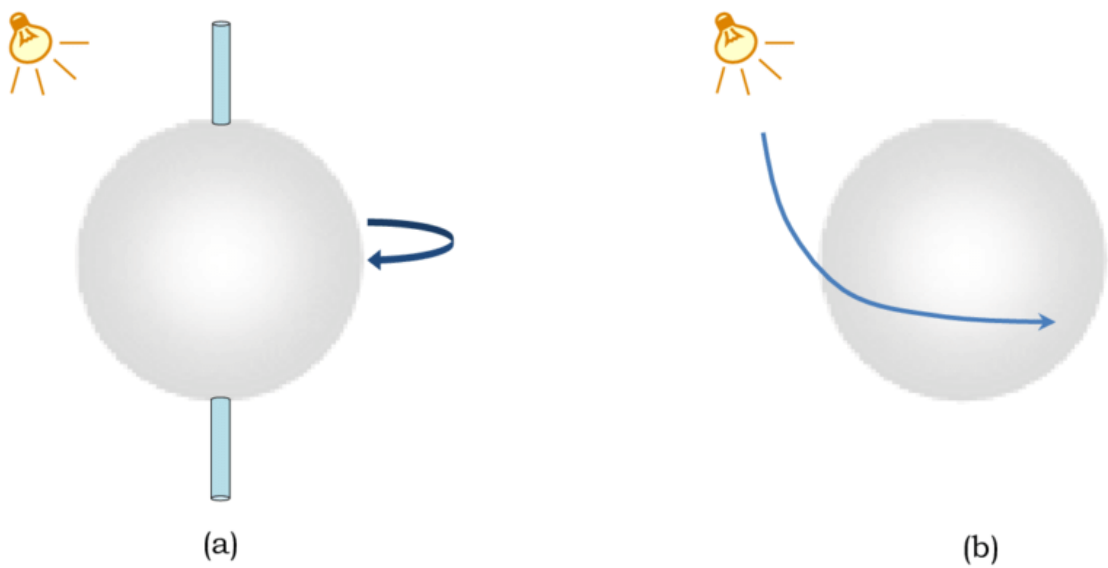


$$\vec{u} = (u, v)$$

Slide adapted from Savarese.

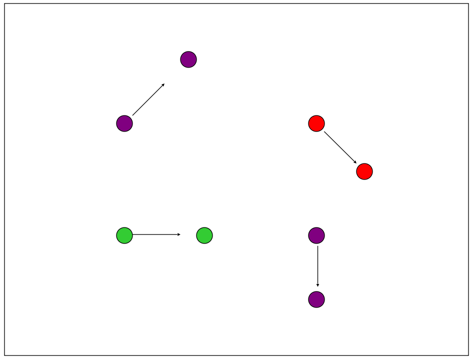
Apparent Motion

- Optical flow differs from actual motion field:
  - (a) intensity remains constant, so that no motion is perceived;
  - (b) no object motion exists, however moving light source produces shading changes.

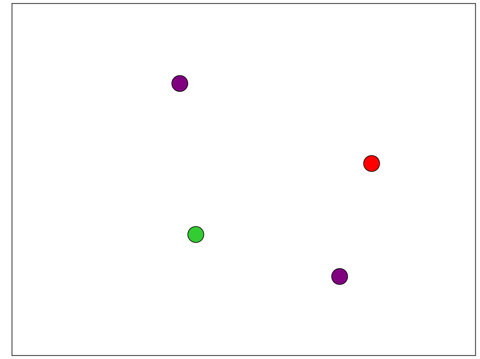


Source: Silvio Savarese

Estimating Optical Flow



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

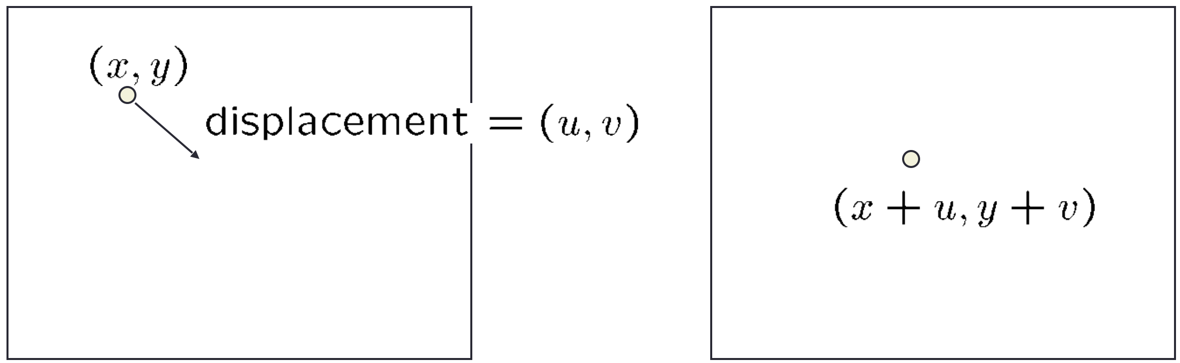


$I(x,y,t)$

- Given two subsequent frames, estimate the apparent motion field u(x,y), v(x,y) between them
- **Key assumptions**
  - Brightness constancy: projection of the same point looks the same in every frame
  - Small motion: points do not move very far
  - Spatial coherence: points move like their neighbors

Source: Silvio Savarese

Optical Flow Constraints (grayscale images)



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

- Let's look at these constraints more closely
  - Brightness constancy constraint (equation)

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

  - Small motion: (u and v are less than 1 pixel, or smoothly varying)
    Taylor series expansion of I:

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

Brightness Constancy Constraint Equation

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

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

The Brightness Constancy Constraint

Can we use this equation to recover image motion (u,v) at each pixel?

$$0 = I_t + \nabla I \cdot < u, v >$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns (u,v)

Source: Silvio Savarese

The Brightness Constancy Constraint

Can we use this equation to recover image motion (u,v) at each pixel?

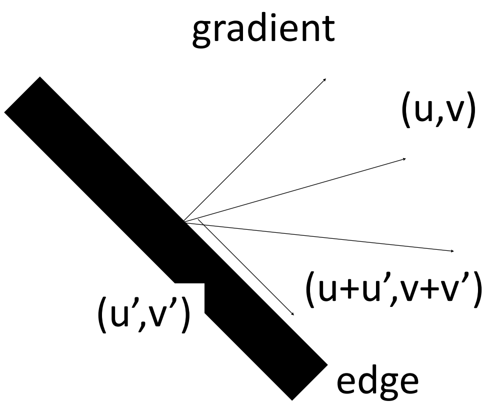
$$0 = I_t + \nabla I \cdot < u, v >$$

- How many equations and unknowns per pixel?
  - One equation (this is a scalar equation!), two unknowns (u,v)

The component of the flow perpendicular to the gradient (i.e., parallel to the edge) cannot be measured

If (u, v) satisfies the equation,
so does (u+u', v+v') if

$$\nabla I \cdot \begin{bmatrix} u' & v' \end{bmatrix}^T = 0$$

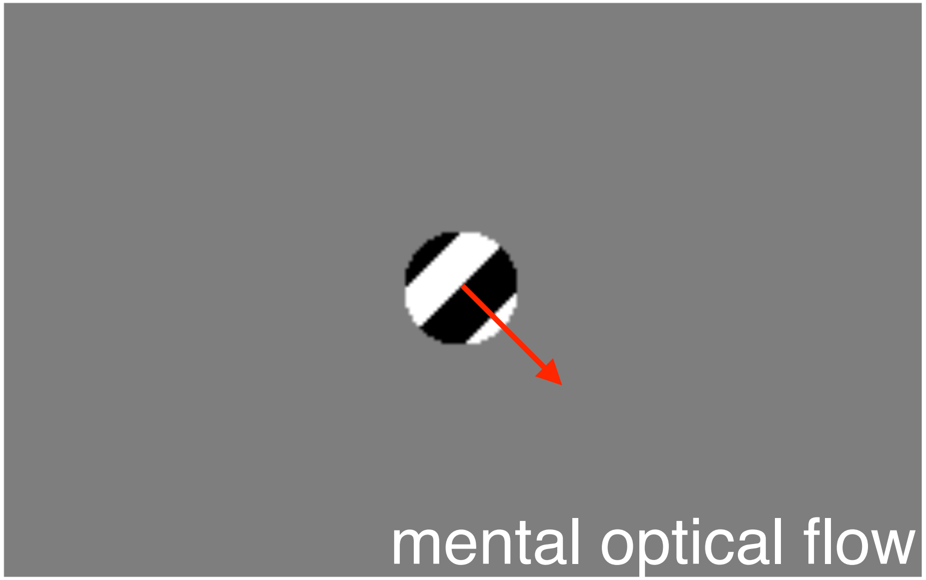


Source: Silvio Savarese

The Barber Pole Illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

The Barber Pole Illusion

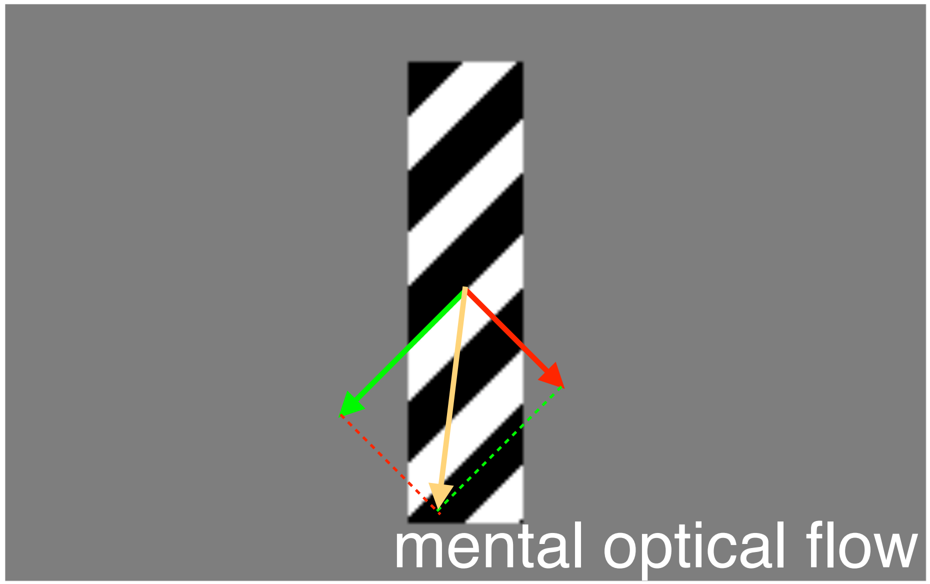


http://en.wikipedia.org/wiki/Barberpole_illusion

The Barber Pole Illusion

http://en.wikipedia.org/wiki/Barberpole_illusion

The Barber Pole Illusion



http://en.wikipedia.org/wiki/Barberpole_illusion

Solving the Ambiguity…

B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

- How to get more equations for a pixel?
- **Spatial coherence constraint**

  Assume the pixel's neighbors have the same (u,v)

  If we use a 5x5 window, that gives us 25 equations per pixel

$$0 = I_t(\mathbf{p_i}) + \nabla I(\mathbf{p_i}) \cdot [u \;\; v]$$

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

Solving the Ambiguity…

- Least squares problem:

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

$$\underset{25\times 2}{A} \; \underset{2\times 1}{d} = \underset{25\times 1}{b}$$

Matching Matches Across Images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \qquad \underset{25\times 2}{A} \; \underset{2\times 1}{d} = \underset{25\times 1}{b}$$

Least squares solution for *d* given by $(A^T A) \; d = A^T b$

Matching Matches Across Images

- Overconstrained linear system

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix} \qquad \underset{25\text{x}2}{A} \; \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Least squares solution for *d* given by $(A^T A)\; d = A^T b$

$$A = \begin{bmatrix} (\nabla I(p_1))^T \\ \vdots \\ (\nabla I(p_n))^T \end{bmatrix} \Rightarrow A^T A = \sum_i \nabla I(p_i)(\nabla I(p_i))^T = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}$$

The summations are over all pixels in the K x K window

Conditions for Solvability

Optimal (u, v) satisfies Lucas-Kanade equation

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

$$A^T A \qquad\qquad\qquad\qquad A^T b$$

When is this solvable? What are good points to track?

- **A^{T}A** should be invertible
- **A^{T}A** should not be too small due to noise
  - eigenvalues λ_1 and λ_2 of **A^{T}A** should not be too small
- **A^{T}A** should be well-conditioned
  - λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

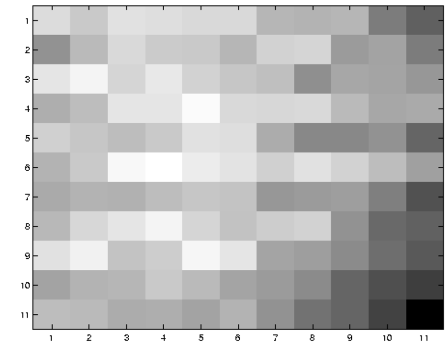
Link Linear Algebra with Pixels

Let $A = \begin{bmatrix} (\nabla I(p_1))^T \\ \vdots \\ (\nabla I(p_n))^T \end{bmatrix} = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$

$\forall w \in \mathbb{R}^2$, we have $w^T(A^TA)w = \|Aw\|^2 = \sum (a_i^T w)^2$

$$\lambda_1 = \lambda_{max}(A^TA) \qquad \lambda_2 = \lambda_{min}(A^TA)$$

Low Texture Region



$$A = \begin{bmatrix} (\nabla I(p_1))^T \\ \vdots \\ (\nabla I(p_n))^T \end{bmatrix} = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$$

Gradients (a_i) have small magnitude

Low Texture Region

Gradients (a_i) have small magnitude

$$A = \begin{bmatrix} (\nabla I(p_1))^T \\ \vdots \\ (\nabla I(p_n))^T \end{bmatrix} = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$$

$$w^T(A^TA)w = \sum (a_i^T w)^2$$

↓

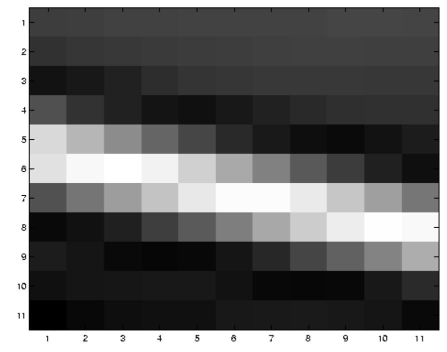
small for any w with $\|w\|^2 = 1$

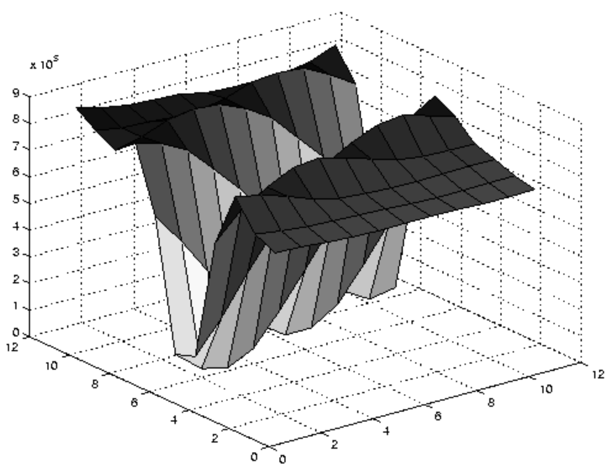
Because $\underset{\|w\|^2=1}{\text{maximize}}\ w^T(A^TA)w = \lambda_1$ (Recall 8-point algo. and HW1)

↓

λ_1 and λ_2 are both small

Edge





large gradients, all the same

Edge

Large gradients (a_i), all the same

$$A = \begin{bmatrix} (\nabla I(p_1))^T \\ \vdots \\ (\nabla I(p_n))^T \end{bmatrix} = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$$

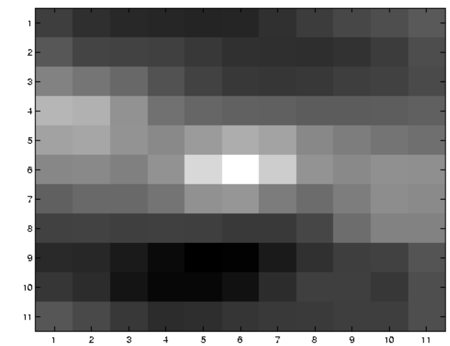
$$w^T(A^TA)w = \sum (a_i^T w)^2$$

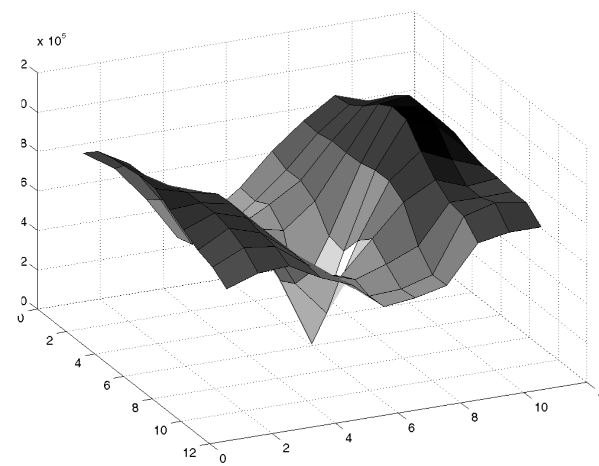
big for w parallel to gradients
small for w orthogonal to gradients

Because $\underset{\|w\|^2=1}{\text{maximize}}\ w^T(A^TA)w = \lambda_1$, λ_1 is big

Because $\underset{\|w\|^2=1}{\text{minimize}}\ w^T(A^TA)w = \lambda_2$, λ_2 is small

High Textured Region





gradients are different,
large magnitudes

High Textured Region

gradients (a_i) are different, large magnitudes

$$A = \begin{bmatrix} (\nabla I(p_1))^T \\ \vdots \\ (\nabla I(p_n))^T \end{bmatrix} = \begin{bmatrix} a_1^T \\ \vdots \\ a_n^T \end{bmatrix}$$

$$w^T(A^TA)w = \sum (a_i^T w)^2$$

↓

for any w there are some $a_i's$ with signficant component along it

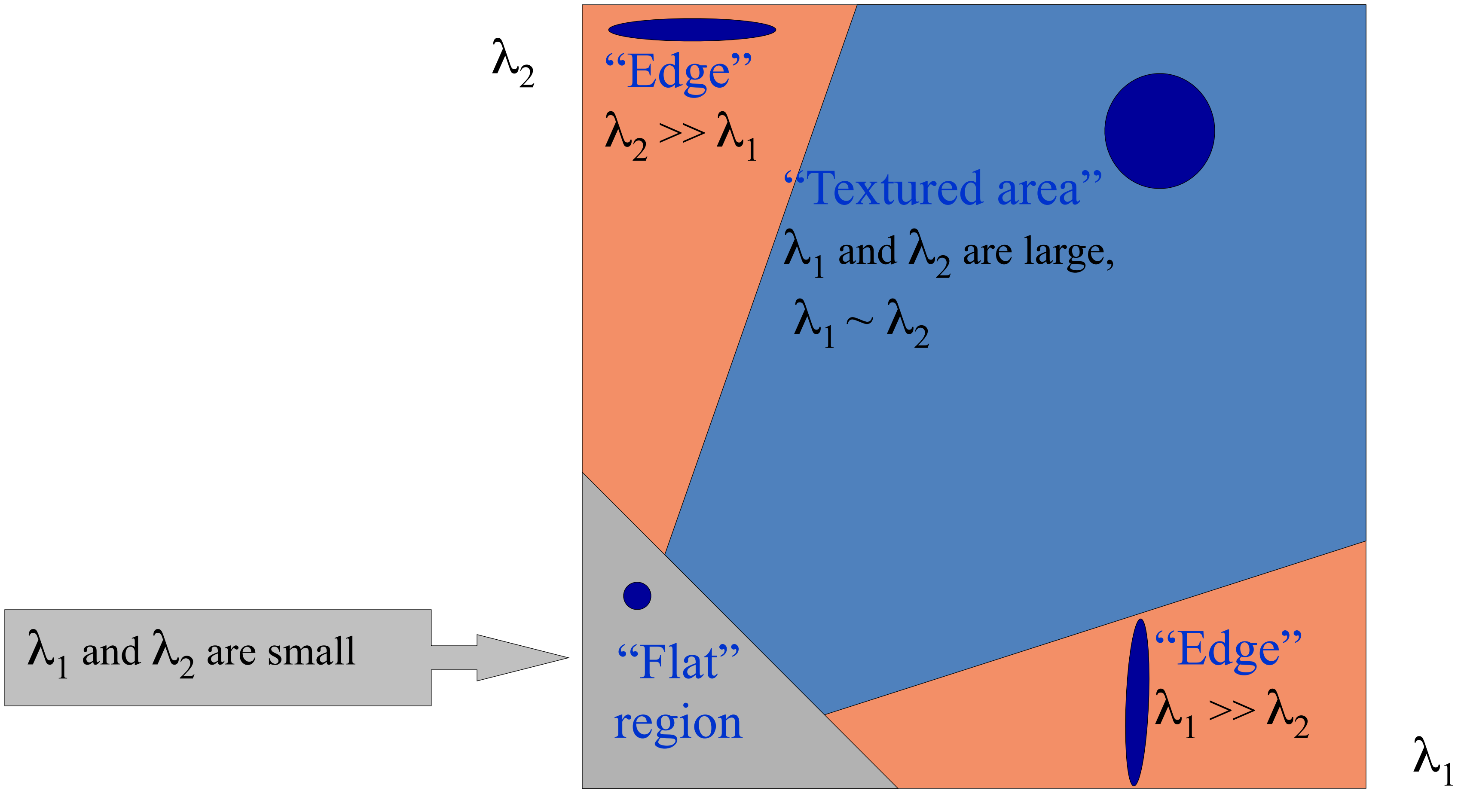
In other words, this quantity is never small

Because $\text{maximize}_{\|w\|^2=1} \; w^T(A^TA)w = \lambda_1$, λ_1 is big

Because $\text{minimize}_{\|w\|^2=1} \; w^T(A^TA)w = \lambda_2$, λ_2 is also big

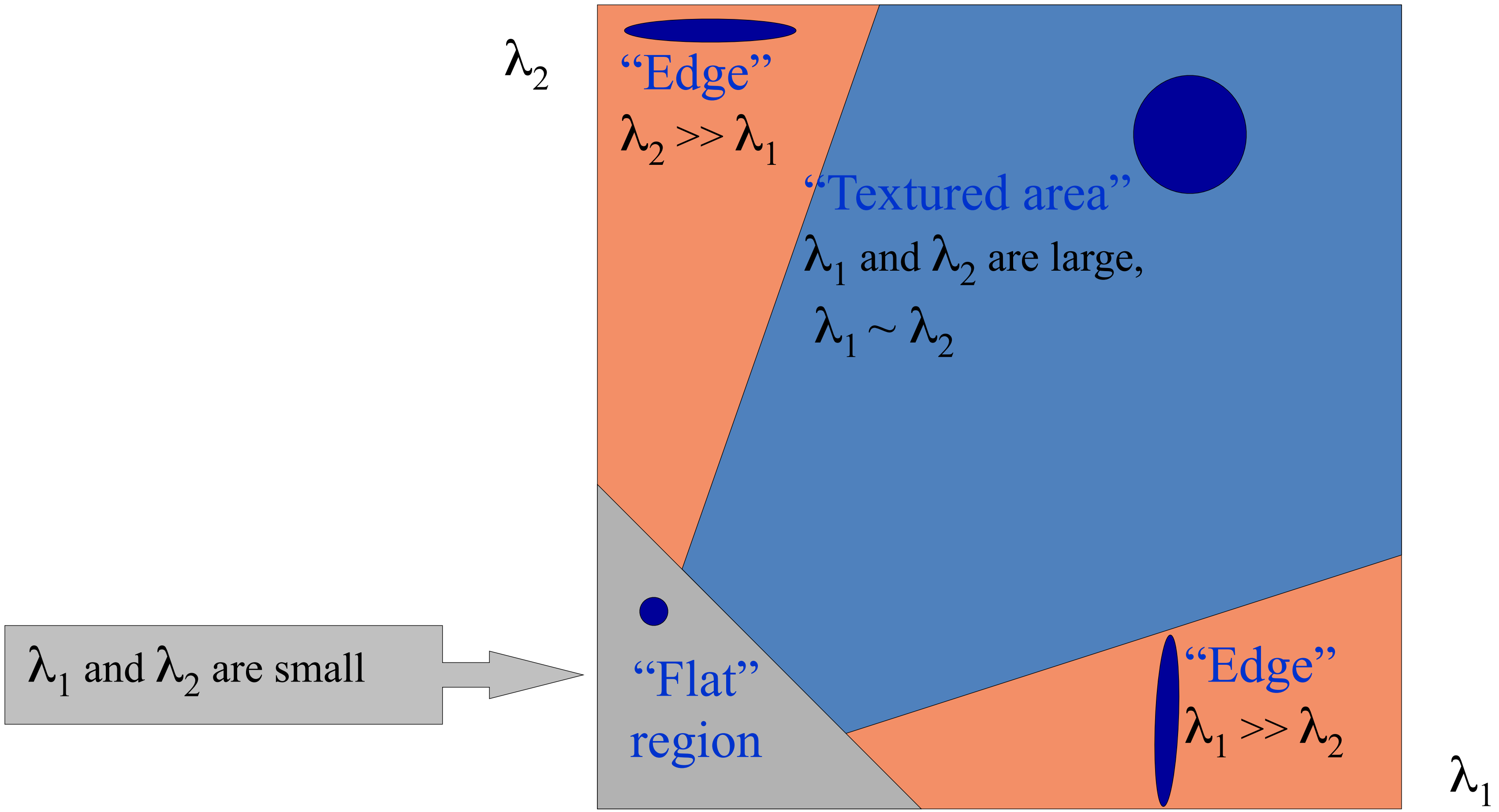
Interpreting the Eigenvalues

Classification of image points using eigenvalues of $A^T A$



Interpreting the Eigenvalues

Classification of image points using eigenvalues of $A^T A$

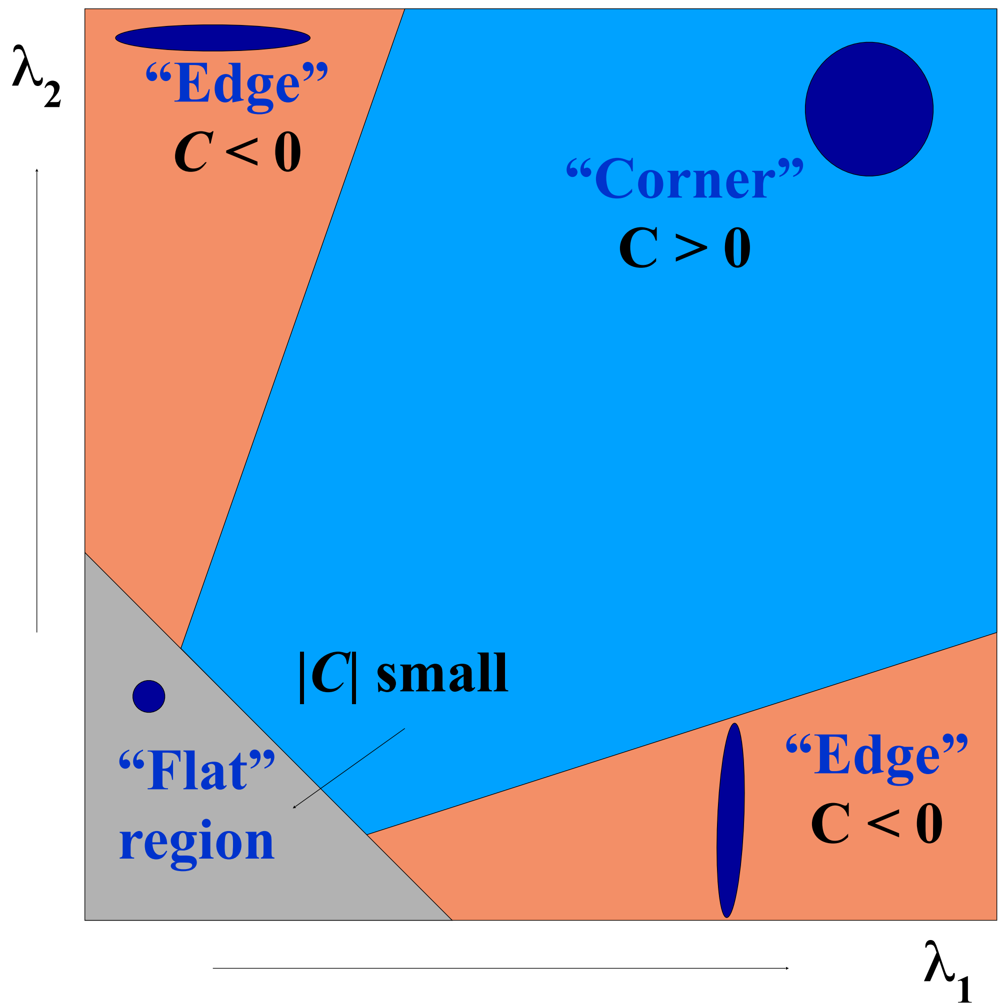


Harris Corner Detector

Cornerness

$$C = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)



Harris Corner Detector

Harris Corner Detector

Errors in Lukas-Kanade

- What are the potential causes of errors in this procedure?
  - Suppose $A^T A$ is easily invertible
  - Suppose there is not much noise in the image

- When our assumptions are violated (Taylor expansion fails)
  - Brightness constancy is **not** satisfied
  - The motion is **not** small
  - A point does **not** move like its neighbors
    - window size is too large

* From Khurram Hassan-Shafique CAP5415 Computer Vision 2003