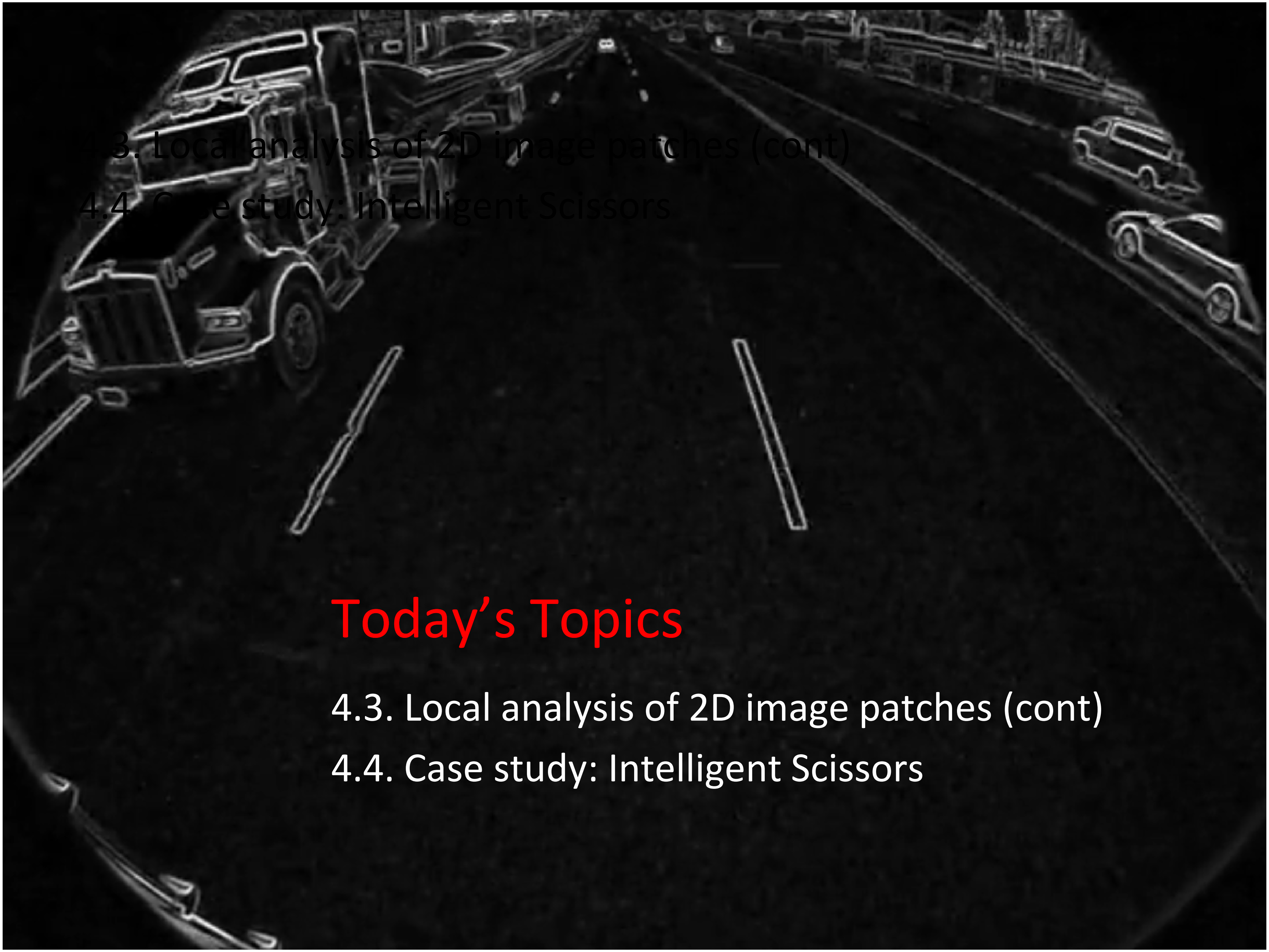


4.3. Local analysis of 2D image patches (cont)
4.4. Case study: Intelligent Scissors

Today’s Topics

4.3. Local analysis of 2D image patches (cont)

4.4. Case study: Intelligent Scissors

Announcements

Marks for A1 are already available on line, through Blackboard

Next week is reading week.

No class, office hours only on Wednesday 11-12.

Topic 4.3:

Local analysis of 2D image patches

- Images as surfaces in 3D
- Directional derivatives
- Image Gradient
  - Painterly rendering
- Edge detection & localization
  - Gradient extrema
  - Laplacian zero-crossings
- Local geometry at image extrema
- The Image Hessian
- Eigenvectors & eigenvalues
- Corner & feature detection
  - Lowe feature detector
  - Harris/Forstner detector

Analysing Special 2D Image Patches

Goal: To mathematically characterize salient image patches

Analysing Special 2D Image Patches

Goal: To mathematically characterize salient image patches

Edges

Gradient magnitude and direction.

Zero-crossings of Laplacian

Analysing Special 2D Image Patches

Goal: To mathematically characterize salient image patches

Edges

Corners

↓

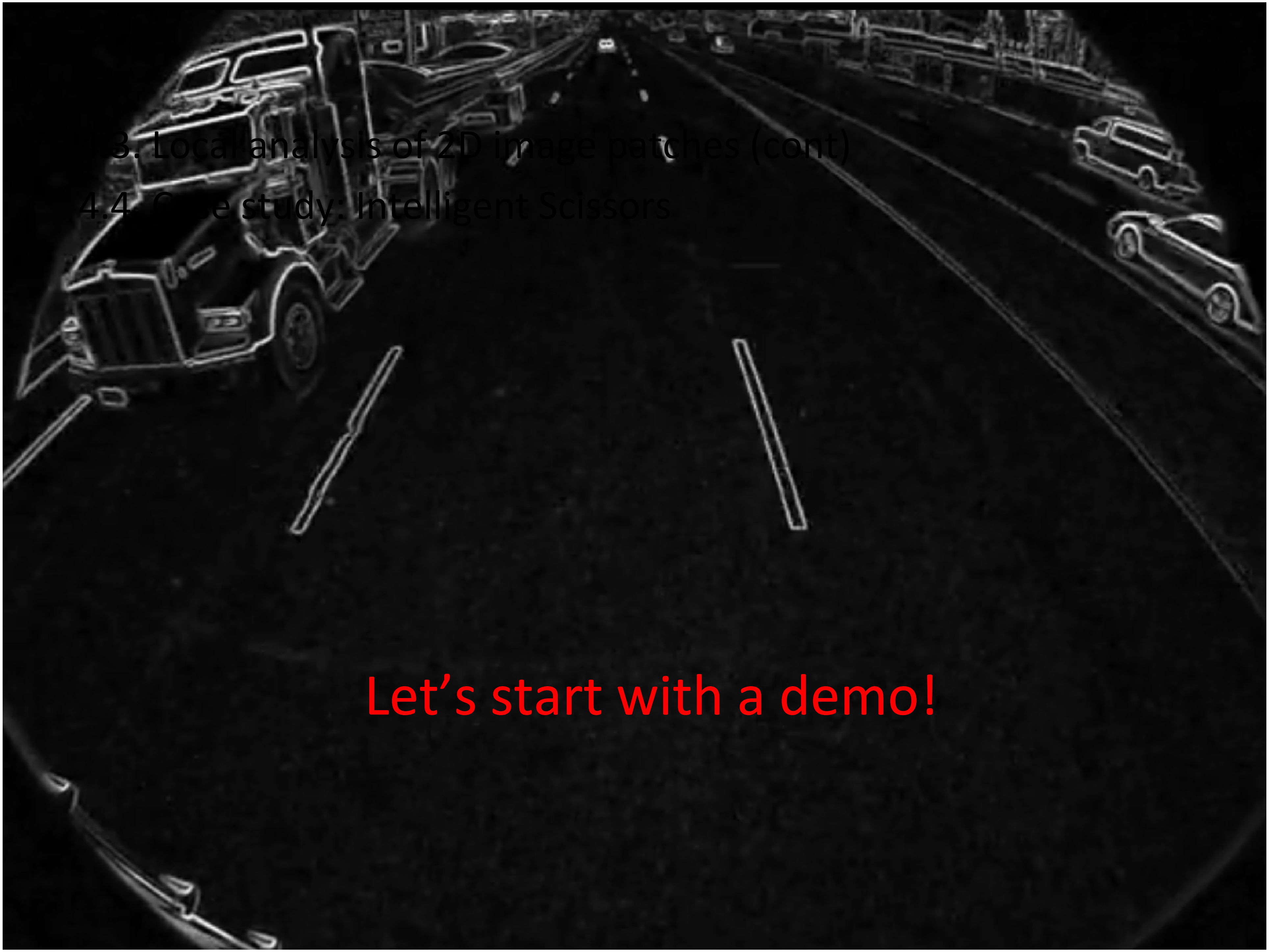
??

Gradient magnitude and direction.

Zero-crossings of Laplacian

4.3. Local analysis of 2D image patches (cont)

4.4. Case study: Intelligent Scissors



Let’s start with a demo!

Topic 4.3:

Local analysis of 2D image patches

- Images as surfaces in 3D
- Directional derivatives
- Image Gradient
  - Painterly rendering
- Edge detection & localization
  - Gradient extrema
  - Laplacian zero-crossings
- Local geometry at image extrema
- The Image Hessian
- Eigenvectors & eigenvalues
- Corner & feature detection
  - Lowe feature detector
  - Harris/Forstner detector

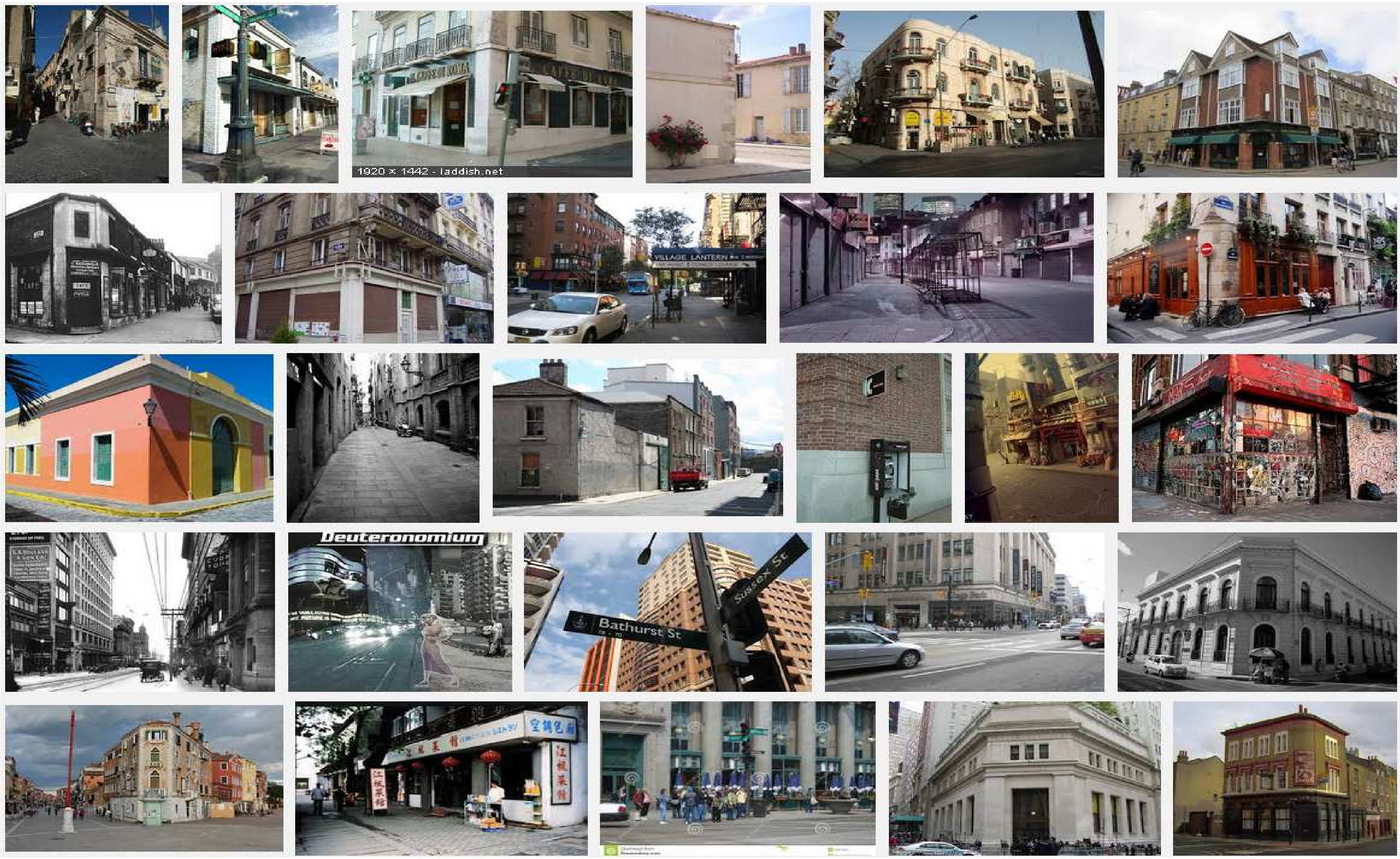
Analysing Special 2D Image Patches

Goal: To mathematically characterize salient image patches

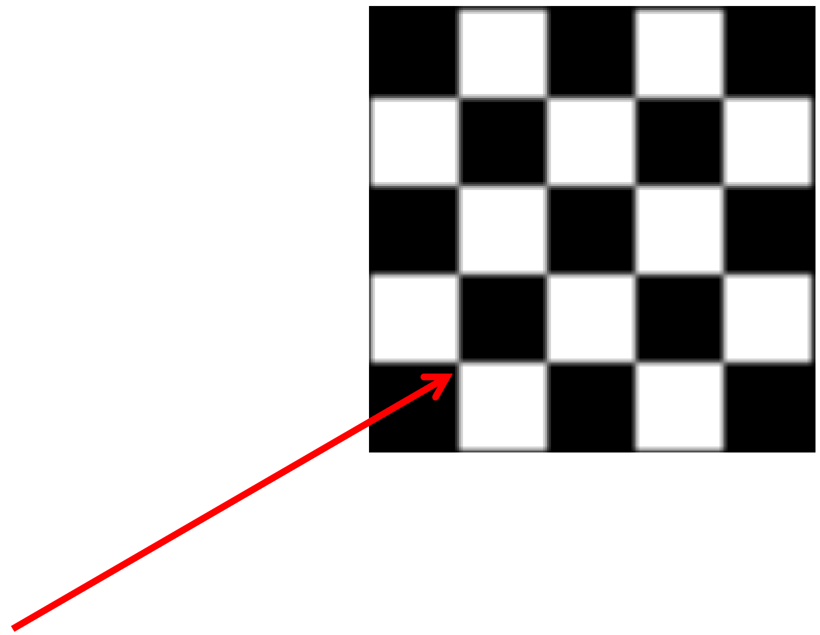
Corners

??

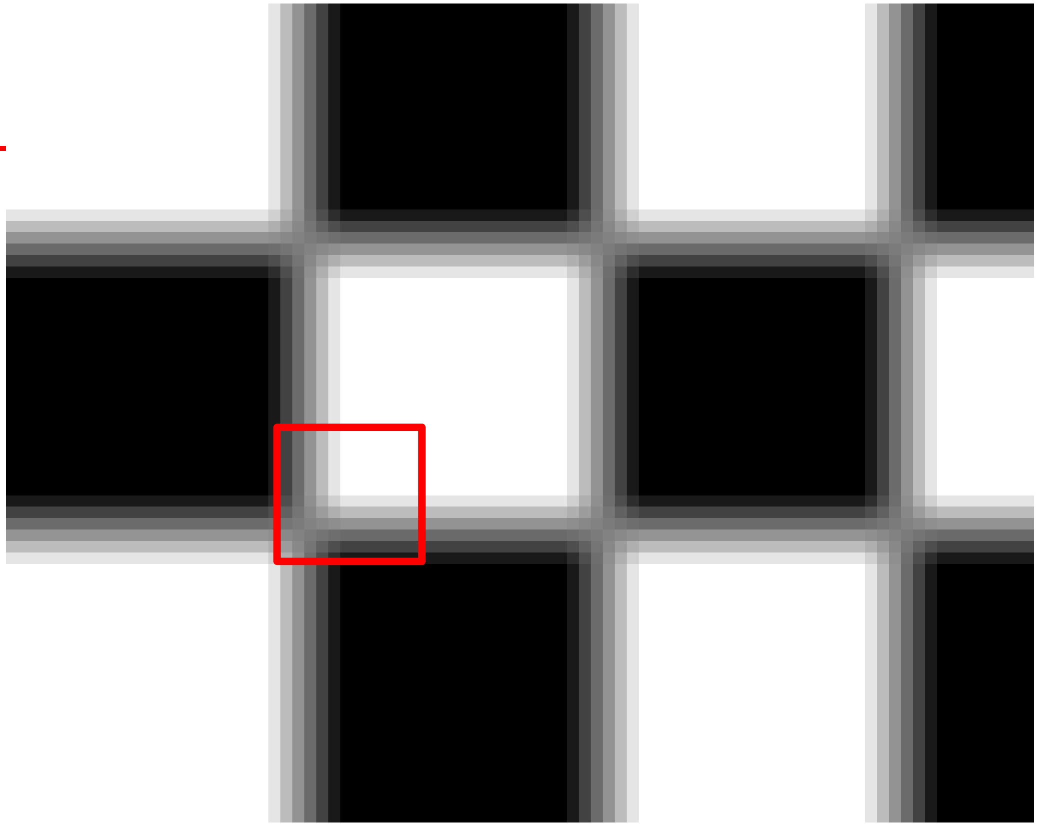
Not these corners…



Corners



Corners

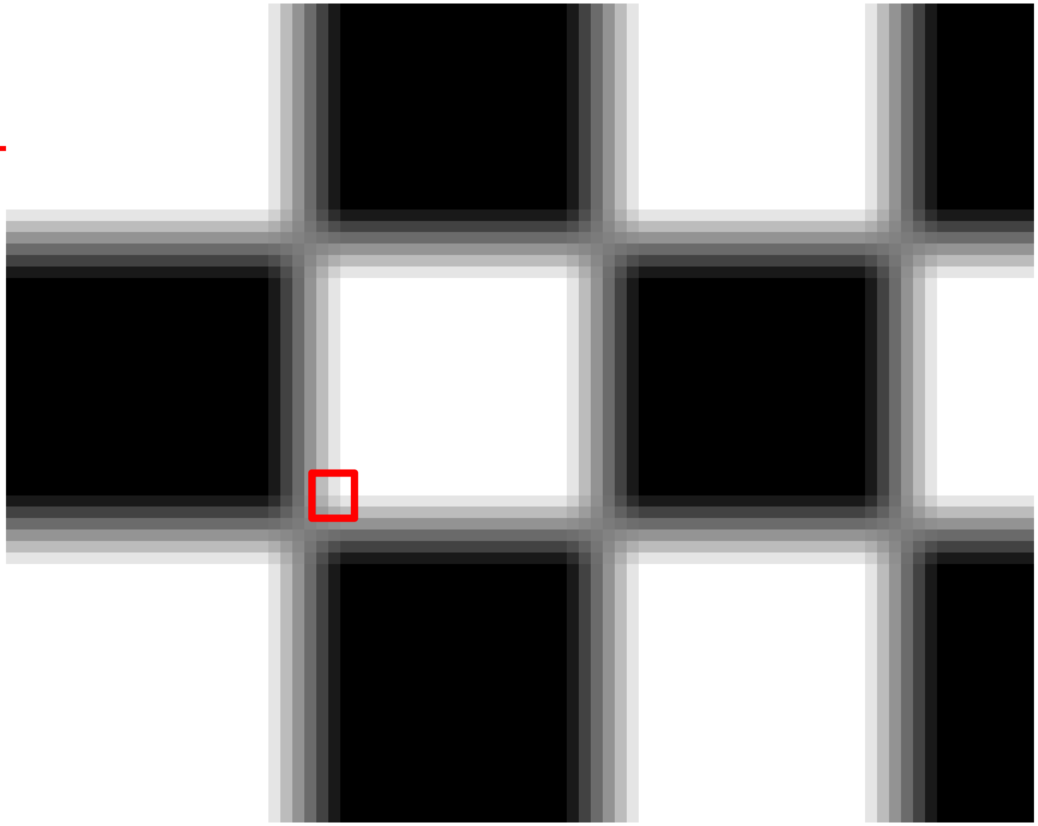


What is a corner?

How is this image patch special?

a corner patch is one where two edges intersect

Corners



will the image gradient be useful?

$$\nabla I(x,y) = \left[\frac{\partial I}{\partial x}(x,y) \quad \frac{\partial I}{\partial y}(x,y) \right]$$

(what was the intuition behind the image gradient?)

Reminder: Partial Derivative along x

$$\nabla I(x,y) = \left[\boxed{\frac{\partial I}{\partial x}(x,y)} \quad \frac{\partial I}{\partial y}(x,y) \right]$$

Local metric of image intensity variation in the horizontal direction

$$\left| \frac{\partial I}{\partial x}(x,y) \right|$$

Reminder: Partial Derivative along y

$$\nabla I(x,y) = \left[\frac{\partial I}{\partial x}(x,y) \quad \boxed{\frac{\partial I}{\partial y}(x,y)} \right]$$

Local metric of image intensity variation in the vertical direction

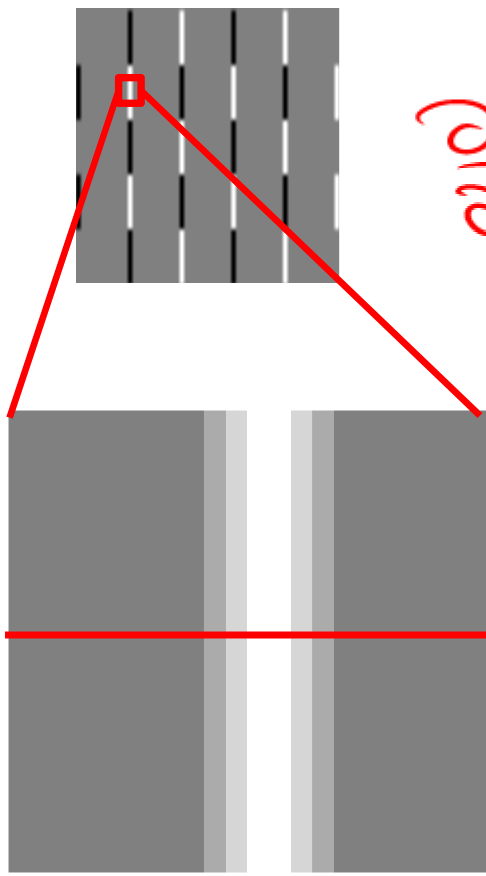
$$\left| \frac{\partial I}{\partial y}(x,y) \right|$$

And the action is at the gradient's…

… extrema (maxima and minima)

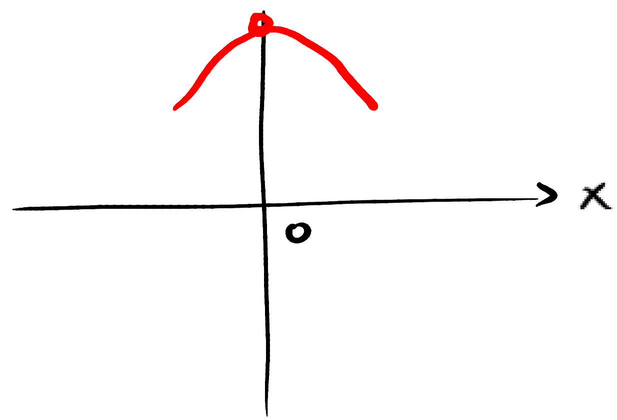
$I(x,y)$

$$\nabla I(x,y) = \left[\frac{\partial I}{\partial x}(x,y) \quad \frac{\partial I}{\partial y}(x,y) \right]$$



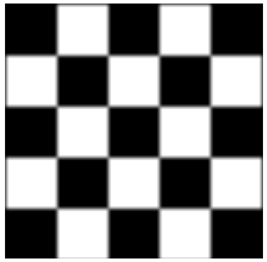
$$\frac{\partial I}{\partial x}$$

$$\frac{\partial I}{\partial x}(x_0, y)$$

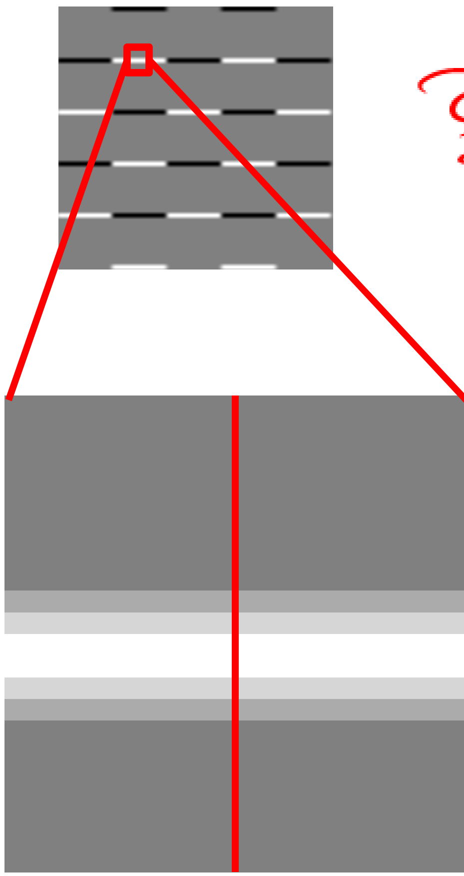


... extrema (maxima and minima)

$I(x,y)$

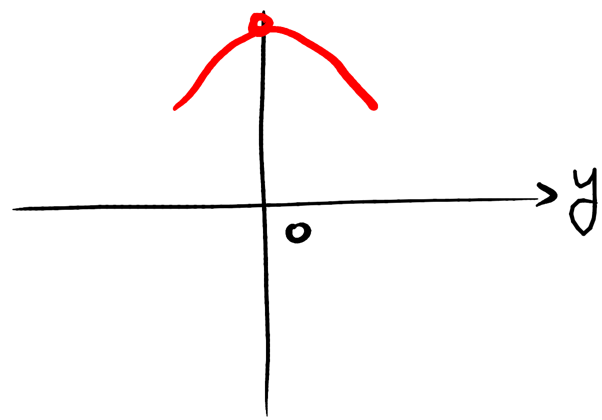


$$\nabla I(x,y) = \left[\frac{\partial I}{\partial x}(x,y) \quad \frac{\partial I}{\partial y}(x,y) \right]$$



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

$\frac{\partial I}{\partial y}(x_0, y)$



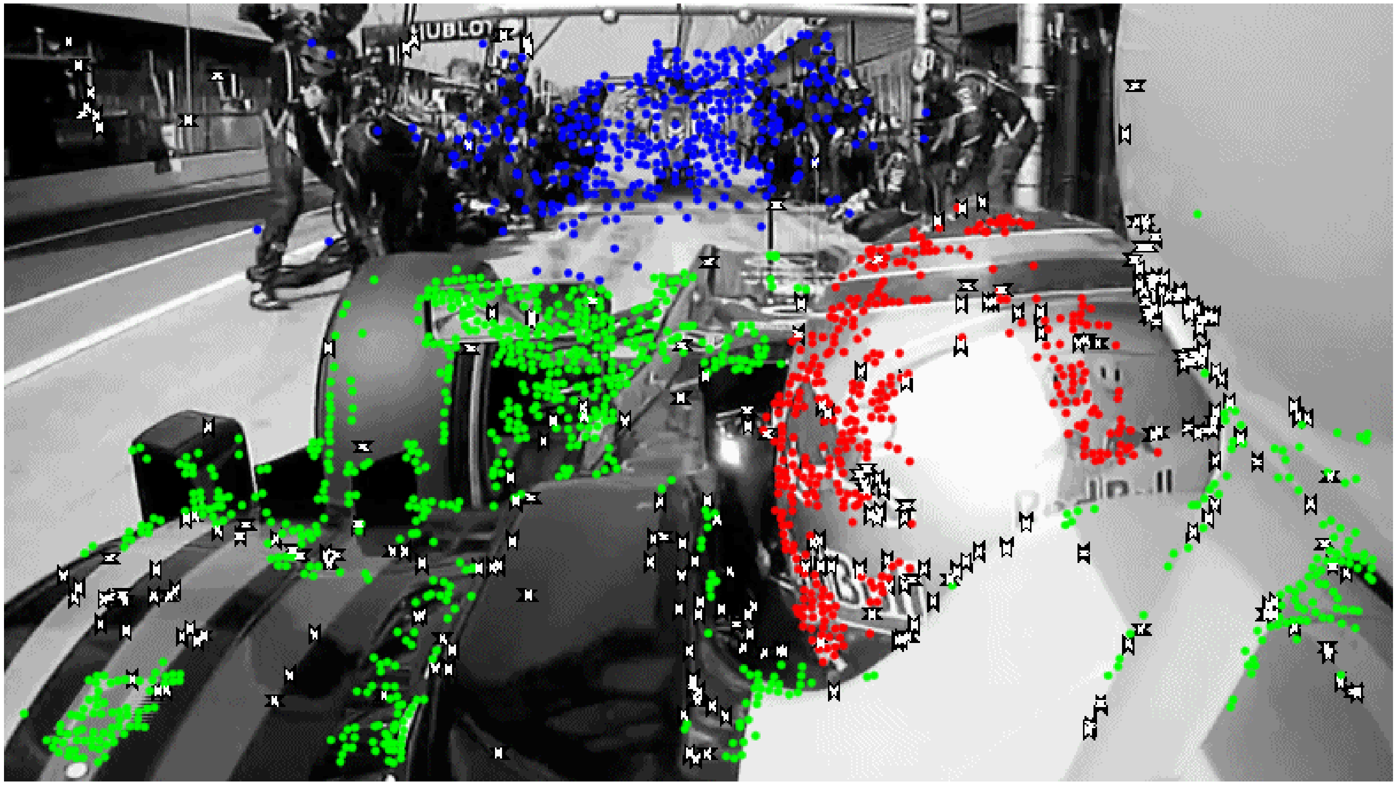
but what happens at a corner?

but what happens at a corner?

and why do we care?

why we care: feature tracking

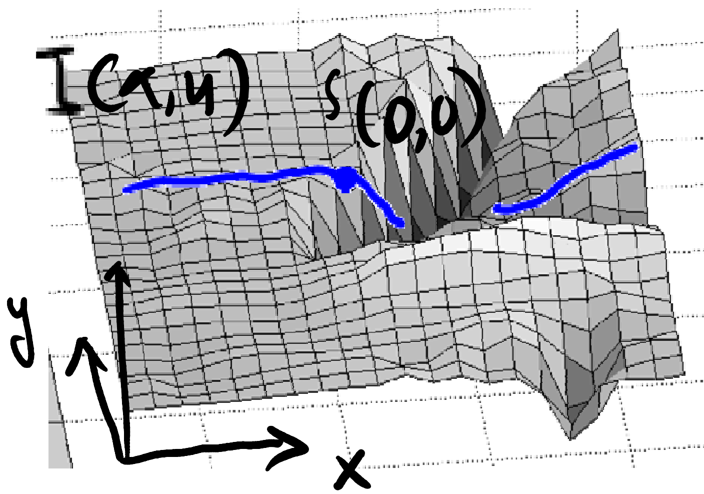
why we care: motion segmentation



a corner patch is one where two edges intersect

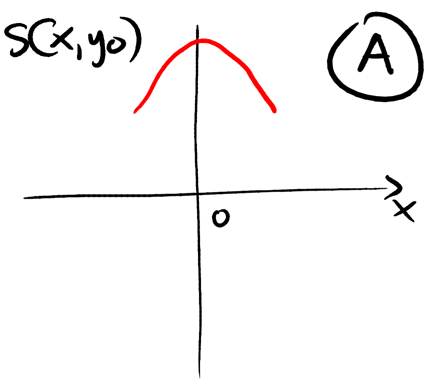
how can they be found using a computer?

Analysis in Neighborhood of Function Extrema

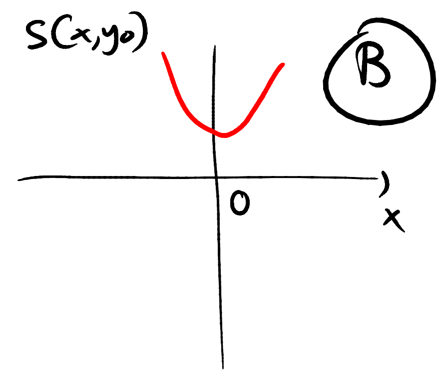


cross-section along x

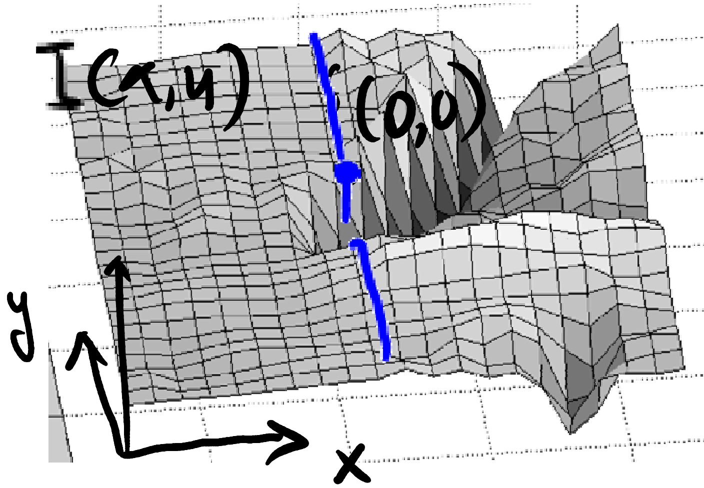
local maximum



local minimum

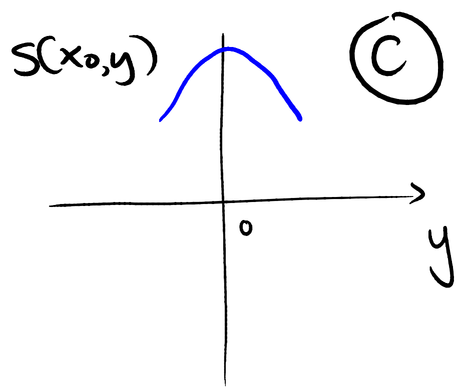


Analysis in Neighborhood

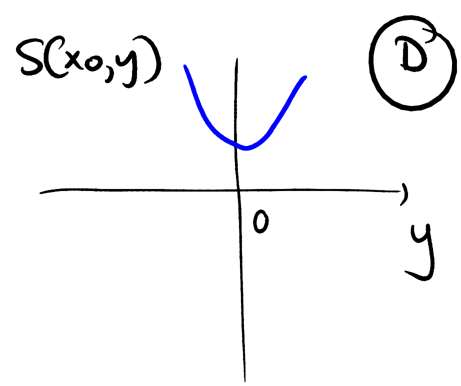


cross-section along y

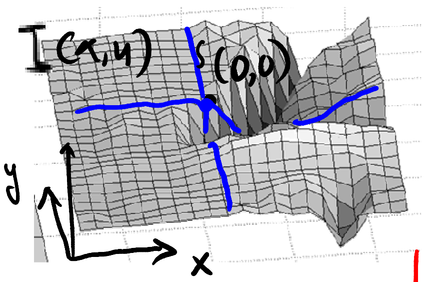
local maximum



local minimum



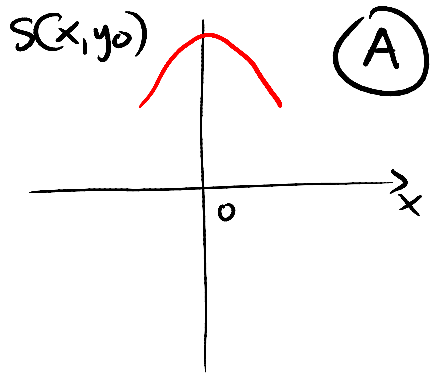
Analysis in Neighborhood of Function Extrema

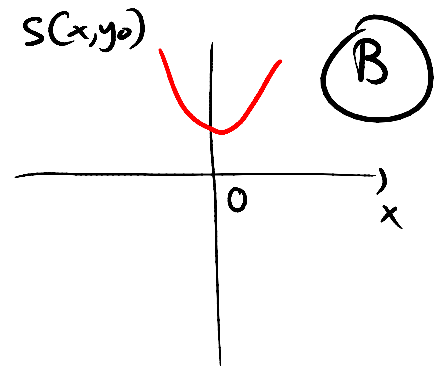


local maximum

local minimum

cross-section along x

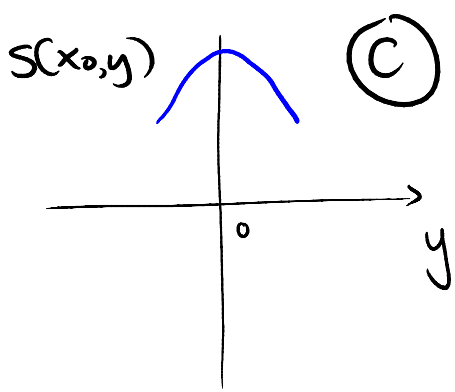


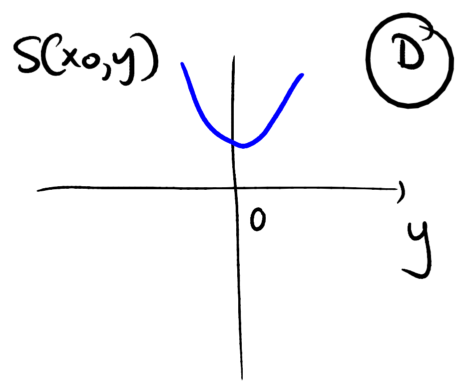


local maximum

local minimum

cross-section along y





4 possible combinations of cross sections along x and y axes

Function extrema

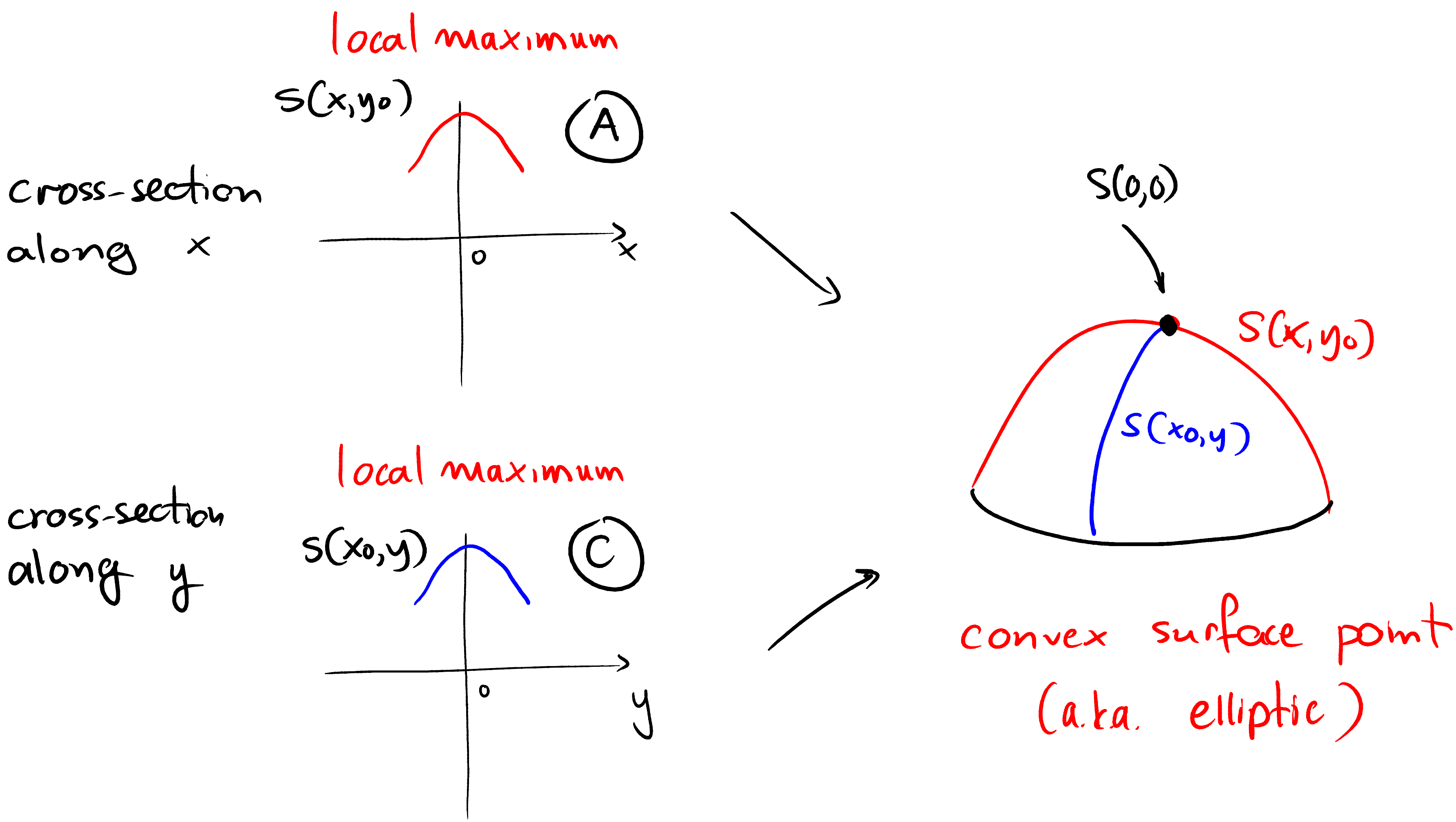
Conditions for gradient extrema:

$$\frac{\partial^2 I}{\partial x^2}(x,y) = 0$$

$$\frac{\partial^2 I}{\partial y^2}(x,y) = 0$$

Case A-C: Elliptical Points

If $\frac{\partial^2 I}{\partial x^2}(x,y) = \frac{\partial^2 I}{\partial y^2}(x,y) = 0$ it could be because:

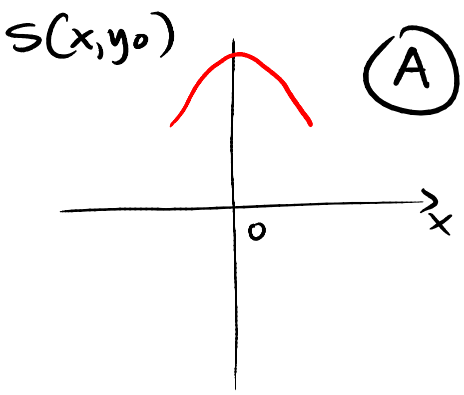


Case A-C: Elliptical Points

If $\frac{\partial^2 I}{\partial x^2}(x,y) = \frac{\partial^2 I}{\partial y^2}(x,y) = 0$ it could be because:

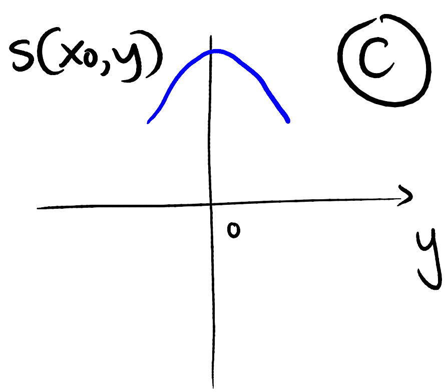
local maximum

cross-section along x

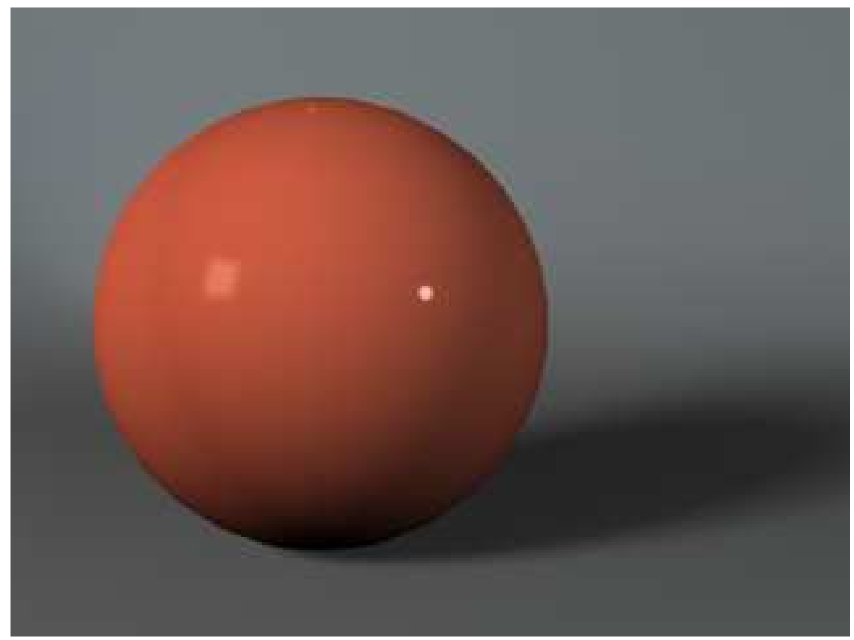


local maximum

cross-section along y



Like in:

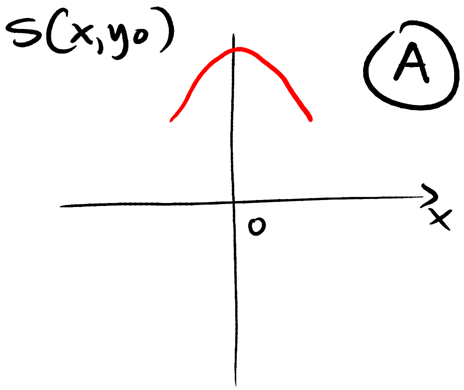


Case A-C: Elliptical Points

If $\frac{\partial^2 I}{\partial x^2}(x,y) = \frac{\partial^2 I}{\partial y^2}(x,y) = 0$ it could be because:

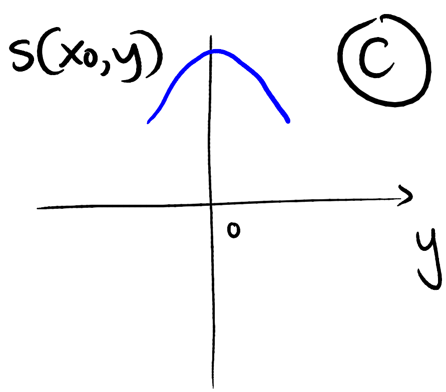
local maximum

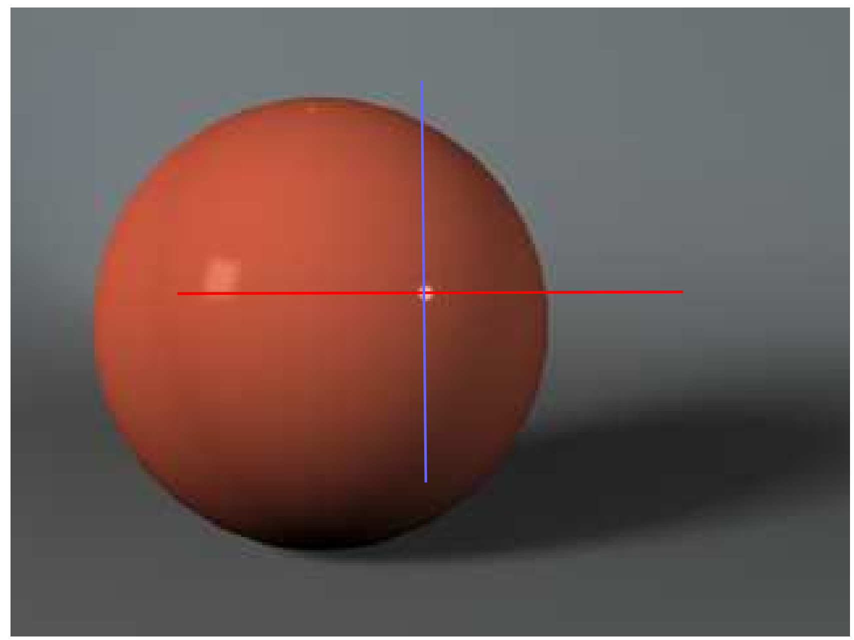
cross-section along x



local maximum

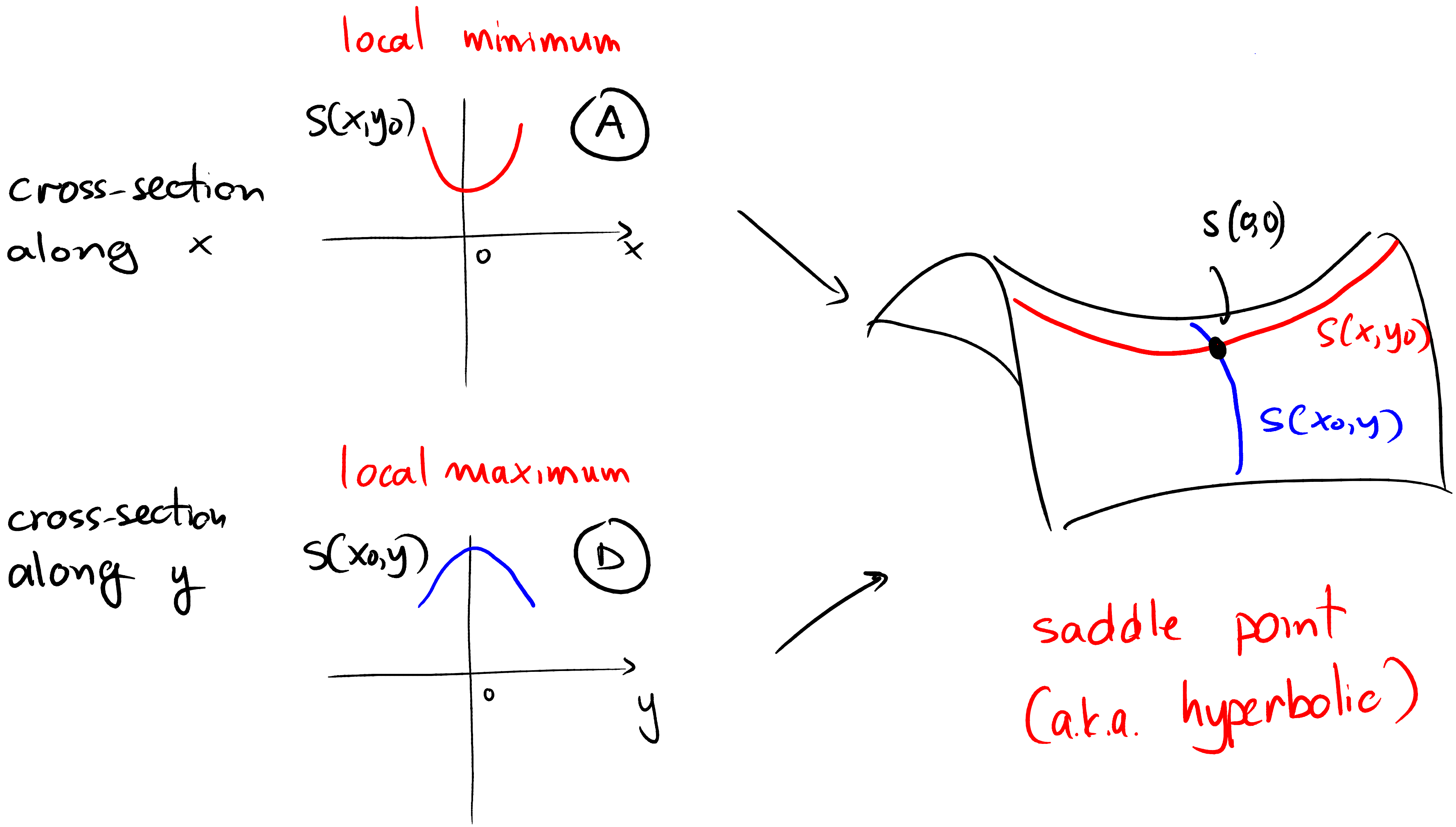
cross-section along y





Case A-D: Hyperbolic Points

If $\dfrac{\partial^2 I}{\partial x^2}(x,y) = \dfrac{\partial^2 I}{\partial y^2}(x,y) = 0$ it could be because:

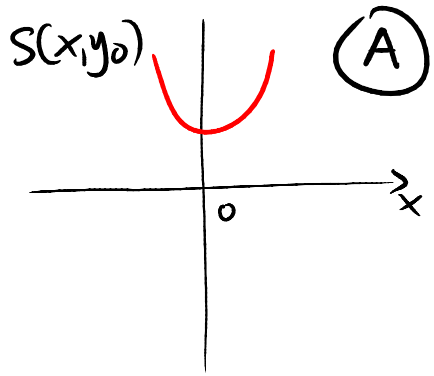


Case A-D: Hyperbolic Points

If $\frac{\partial^2 I}{\partial x^2}(x,y) = \frac{\partial^2 I}{\partial y^2}(x,y) = 0$ it could be because:

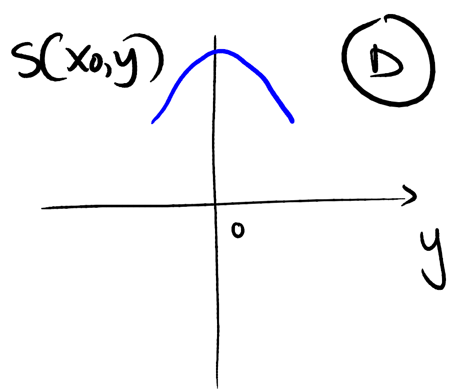
local minimum

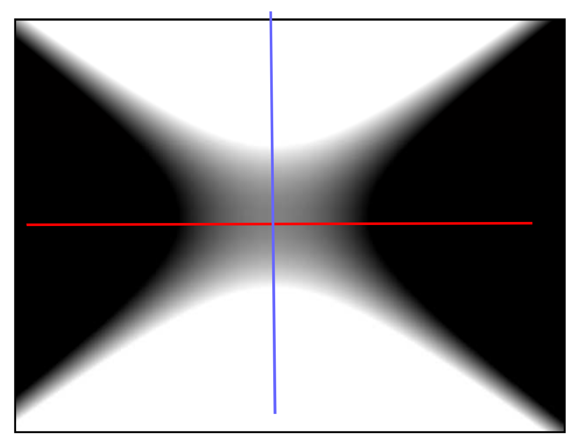
cross-section along x



local maximum

cross-section along y



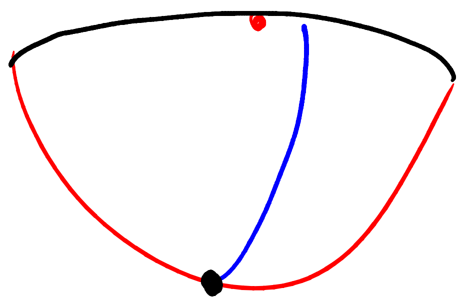


saddle point (a.k.a. hyperbolic)

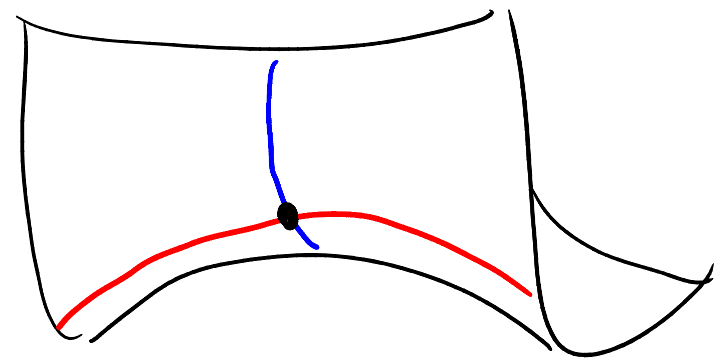
Cases B-C and B-D

If $\dfrac{\partial^2 I}{\partial x^2}(x,y) = \dfrac{\partial^2 I}{\partial y^2}(x,y) = 0$ it could be because:

The other two cases (B-C and B-D) produce "upside-down" versions of these two shapes



convex surface point (a.k.a. elliptic)



saddle point (a.k.a. hyperbolic)

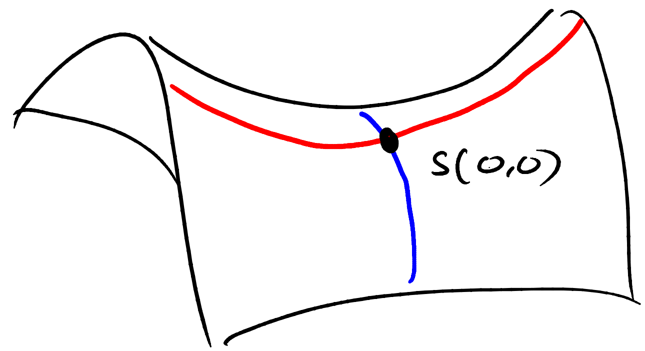
Local Geometry Near Surface Extrema

What if we wanted to approximate the image close to these extreme points?

Taylor series expansion of S

$$S(x,y) = S(0,0) + \cancelto{0}{x \frac{\partial S}{\partial x}(0,0) + y \frac{\partial S}{\partial y}(0,0)}$$

$$+ \frac{1}{2}\left[x^2 \frac{\partial^2 S}{\partial x^2}(0,0) + xy \frac{2\,\partial^2 S}{\partial x \partial y}(0,0) + y^2 \frac{\partial^2 S}{\partial y^2}(0,0) \right]$$

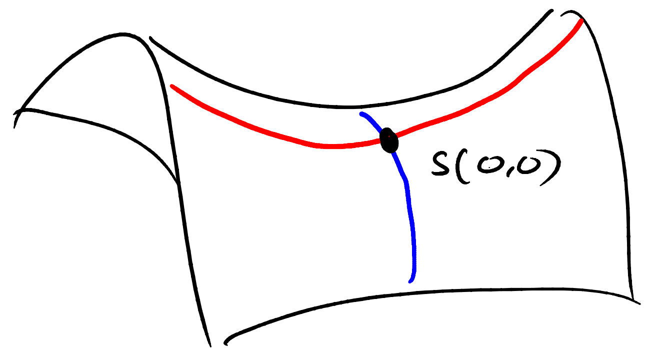


Local Geometry Near Surface Extrema

What if we wanted to approximate the image close to these extreme points?

Taylor series expansion of S

$$S(x,y) = S(0,0) + + \frac{1}{2}\left[x^2 \frac{\partial^2 S}{\partial x^2}(0,0) + xy \frac{2\,\partial^2 S}{\partial x \partial y}(0,0) + y^2 \frac{\partial^2 S}{\partial y^2}(0,0)\right]$$



Note that local shape is determined by the 2nd derivative only!

Topic 4.2:

Local analysis of 2D image patches

- Images as surfaces in 3D
- Directional derivatives
- Image Gradient
    - Painterly rendering
- Edge detection & localization
    - Gradient extrema
    - Laplacian zero-crossings
- Local geometry at image extrema
- The Image Hessian
- Eigenvectors & eigenvalues
- Corner & feature detection
    - Lowe feature detector
    - Harris/Forstner detector

The Hessian Matrix

If $\frac{\partial^2 I}{\partial x^2}(x,y) = \frac{\partial^2 I}{\partial y^2}(x,y) = 0$ at a point (x,y), then the Taylor Series expansion of the function at that point is:

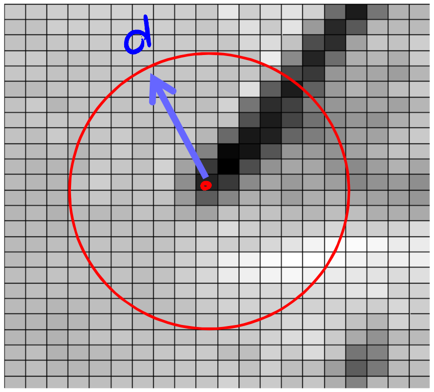
$$S(x,y) = S(0,0) + \frac{1}{2}\left[x^2 \frac{\partial^2 S}{\partial x^2}(0,0) + xy\,\frac{2\,\partial^2 S}{\partial x \partial y}(0,0) + y^2 \frac{\partial^2 S}{\partial y^2}(0,0)\right]$$

Hessian matrix

$$= S(0,0) + \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \frac{\partial^2 S}{\partial x^2} & \frac{\partial^2 S}{\partial x \partial y} \\ \frac{\partial^2 S}{\partial x \partial y} & \frac{\partial^2 S}{\partial y^2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

The Hessian Matrix: Intuition

The Hessian matrix H determines how S(x,y) changes from a unit-length displacement *d*, in a given direction

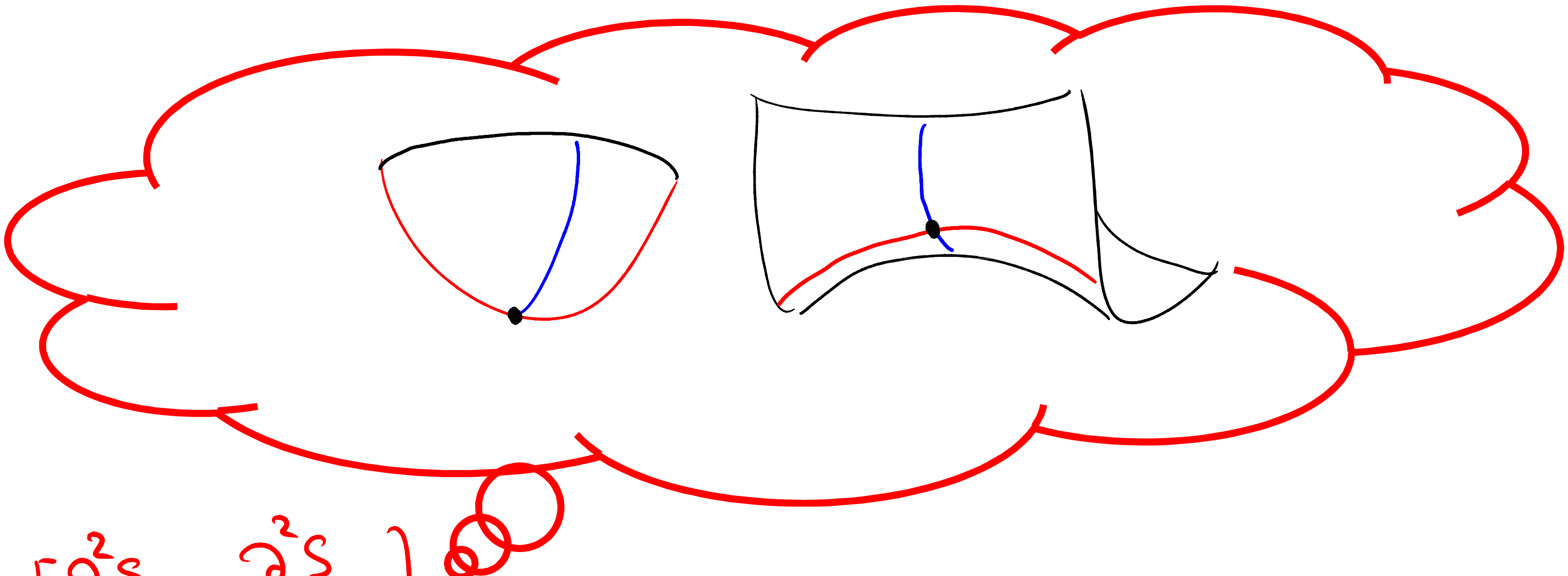


$$S(x,y) = S(0,0) + \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \frac{\partial^2 S}{\partial x^2} & \frac{\partial^2 S}{\partial x \partial y} \\ \frac{\partial^2 S}{\partial x \partial y} & \frac{\partial^2 S}{\partial y^2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$S(x,y) = S(0,0) + d^T H d \text{ with } d = \begin{bmatrix} x \\ y \end{bmatrix}$$

The Hessian Matrix: Intuition

In other words, the Hessian knows about the local shape of S



$$\begin{bmatrix} \frac{\partial^2 S}{\partial x^2} & \frac{\partial^2 S}{\partial x \partial y} \\ \frac{\partial^2 S}{\partial x \partial y} & \frac{\partial^2 S}{\partial y^2} \end{bmatrix}$$

Local Geometry from the Hessian Matrix

$$S(x,y) = S(0,0) + \frac{1}{2}\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \frac{\partial^2 S}{\partial x^2} & \frac{\partial^2 S}{\partial x \partial y} \\ \frac{\partial^2 S}{\partial x \partial y} & \frac{\partial^2 S}{\partial y^2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

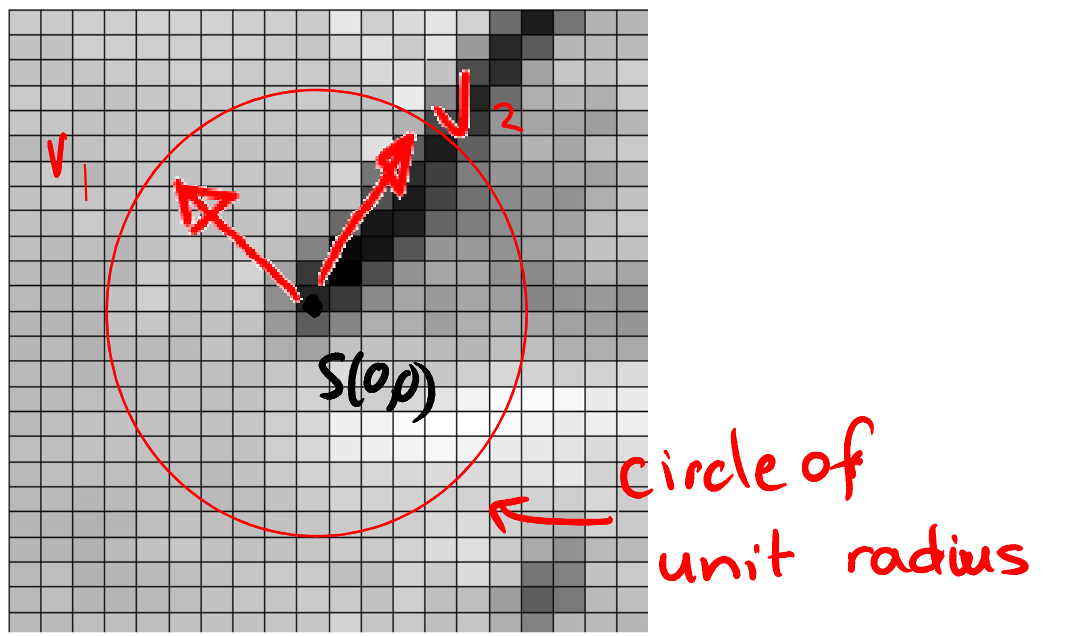
The Hessian (H) defines 2 orthogonal unit vectors v_1, v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

$$\text{angle}\left(\begin{bmatrix} x \\ y \end{bmatrix}, v_1\right) = \theta \implies S(x,y) - S(0,0) = \lambda_1 \cos^2\theta + \lambda_2 \sin^2\theta$$

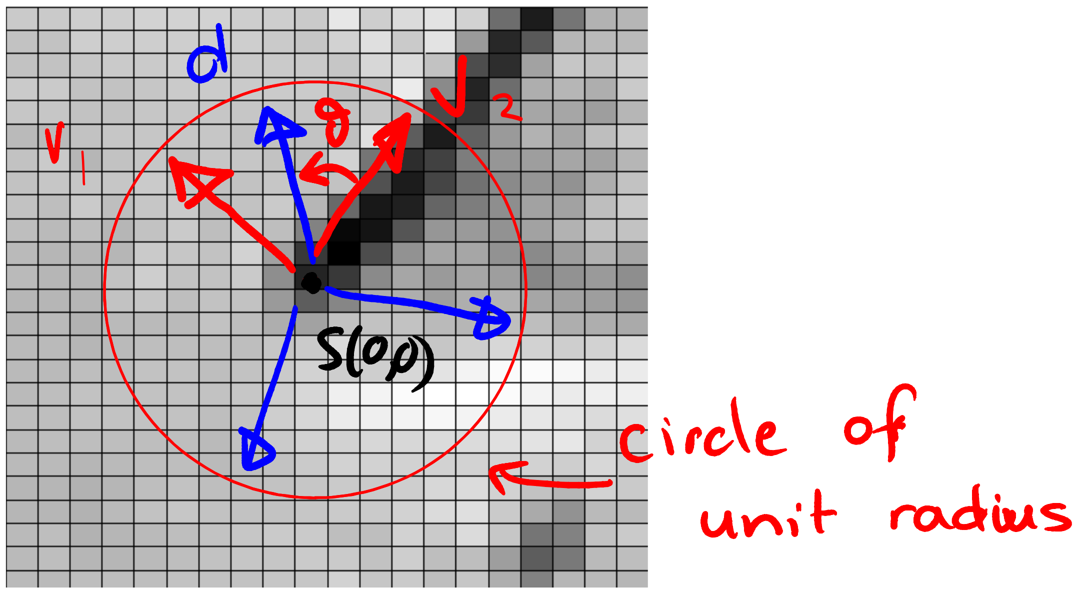
Local Geometry from the Hessian Matrix

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$



Local Geometry from the Hessian Matrix

$$\text{angle}\left(\begin{bmatrix} x \\ y \end{bmatrix}, v_1\right) = \theta \Rightarrow S(x,y) - S(0,0) = \lambda_1 \cos^2\theta + \lambda_2 \sin^2\theta$$



Local Geometry from the Hessian Matrix

- $\text{angle}\left(\begin{bmatrix} x \\ y \end{bmatrix}, v_1\right) = \vartheta \Rightarrow S(x,y) - S(0,0) = \lambda_1 \cos^2\vartheta + \lambda_2 \sin^2\vartheta$

Proof

If $\text{angle}(d, v_1) = \vartheta$, we have $d = \cos\vartheta \cdot v_1 + \sin\vartheta \cdot v_2$

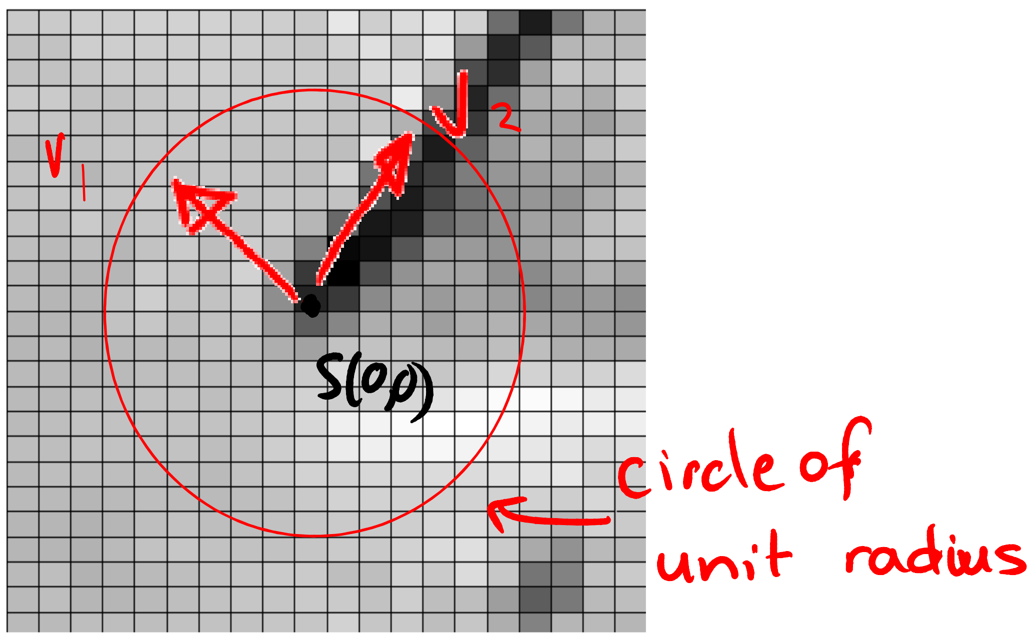
- $(\cos\vartheta \cdot v_1^T + \sin\vartheta \cdot v_2^T) H (\cos\vartheta \cdot v_1 + \sin\vartheta \cdot v_2) =$

$$\cos^2\vartheta \cdot v_1^T (\underbrace{H v_1}_{\lambda_1 v_1}) + \sin\vartheta \cdot \cos\vartheta \cdot \cancel{v_2^T (\underbrace{H v_1}_{\lambda_1 v_1})} + \cos\vartheta \cdot \sin\vartheta \cdot \cancel{v_1^T (\underbrace{H v_2}_{\lambda_2 v_2})} + \sin^2\vartheta \, v_2^T (\underbrace{H v_2}_{\lambda_2 v_2}) =$$

$$\cos^2\vartheta \cdot \lambda_1 + \sin^2\vartheta \cdot \lambda_2$$

Local Geometry from the Hessian Matrix

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 =$ MAXIMAL
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 =$ MINIMAL



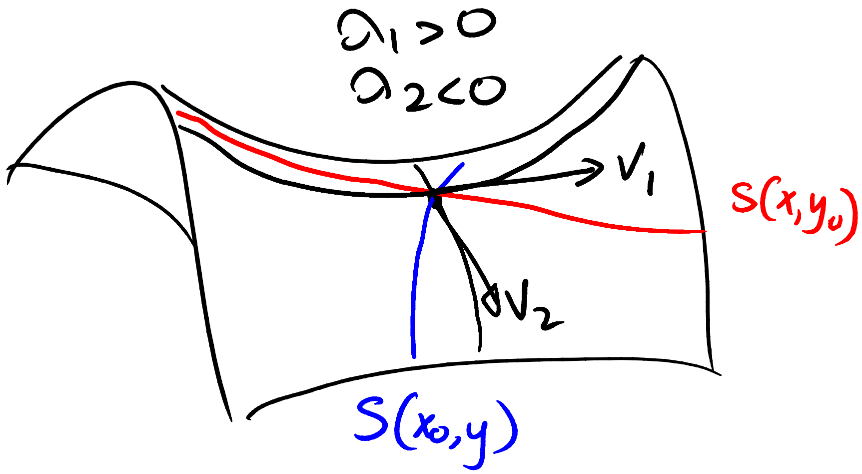
Local Geometry from the Hessian Matrix

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

- v_1, v_2 are unit Eigenvectors of H
- λ_1, λ_2 are Eigenvalues of H

Local Geometry from the Hessian Matrix

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$



v_1 = direction of greatest ascent

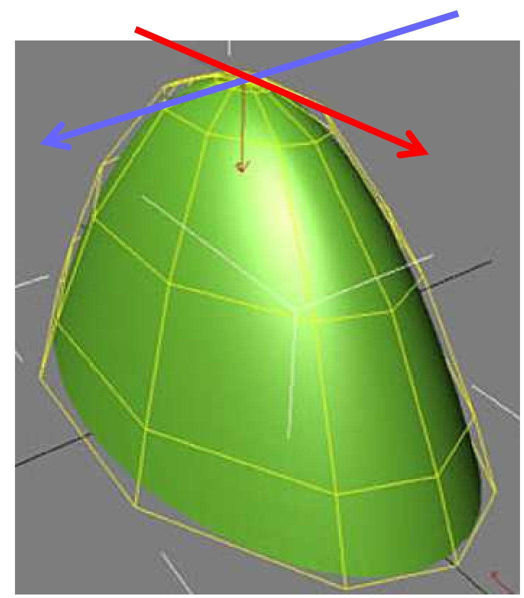
v_2 = direction of smallest ascent
(or greatest descent)

Local Geometry from the Hessian Matrix

$\cdot \begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$

$\cdot \begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

$\lambda_1 < 0$
$\lambda_2 < 0$



v1 = direction of steepest ascent (or less steep descent)

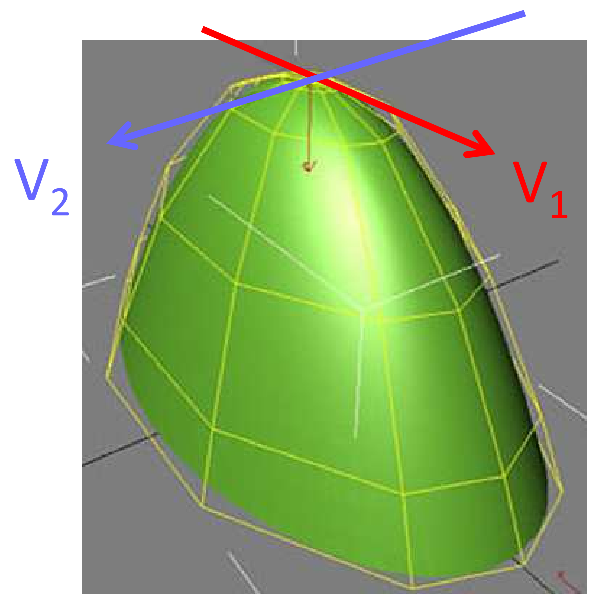
v2 = direction of steepest descent (or least steep ascent)

Local Geometry from the Hessian Matrix

$\cdot \begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$

$\cdot \begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

$\lambda_1 < 0$
$\lambda_2 < 0$

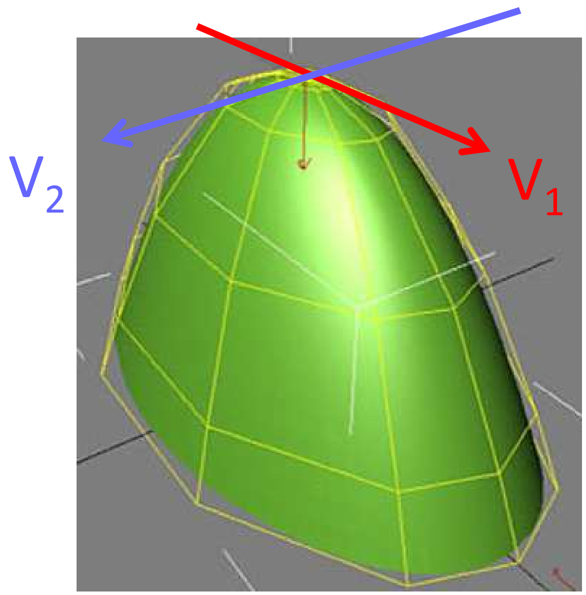


v1 = direction of steepest ascent (or less steep descent)

v2 = direction of steepest descent (or least steep ascent)

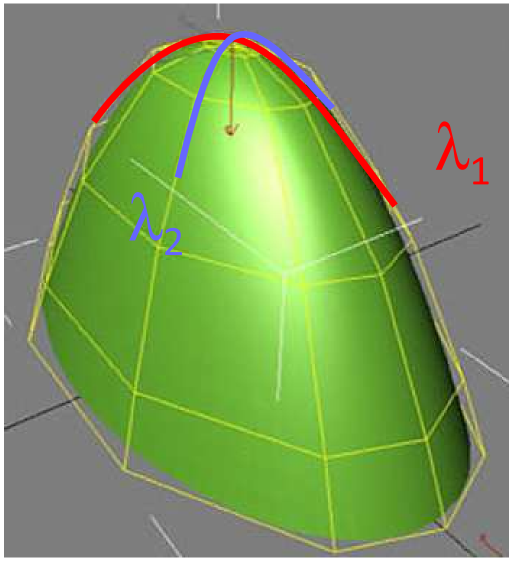
Principal Directions & Curvatures

- v_1 and v_2 are called the principal directions at the surface point S(0,0)



Principal Directions & Curvatures

- λ_1 and λ_2 are called the principal curvatures at S(0,0)



Topic 4.3:

Local analysis of 2D image patches

- Images as surfaces in 3D
- Directional derivatives
- Image Gradient
  - Painterly rendering
- Edge detection & localization
  - Gradient extrema
  - Laplacian zero-crossings
- Local geometry at image extrema
- The Image Hessian
  - Eigenvectors & eigenvalues
- Corner & feature detection
  - Lowe feature detector
  - Harris/Forstner detector

Background: Eigenvectors & Eigenvalues

Definition:

A non-zero vector $\mathbf{v}$ is an eigenvector of a matrix H if

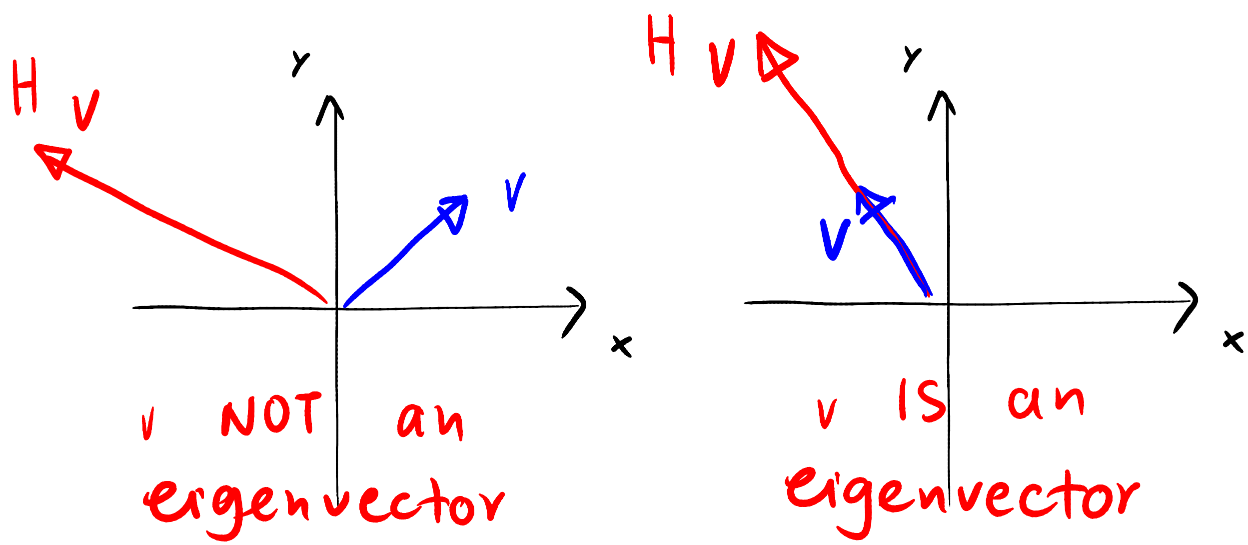
$$H\mathbf{v} = \lambda\mathbf{v}.$$

The scalar λ is the eigenvalue associated to $\mathbf{v}$.

Background: Eigenvectors & Eigenvalues

A non-zero vector **v** is an eigenvector of a matrix H if H**v** = λ**v.** The scalar λ is the eigenvalue associated to **v.**

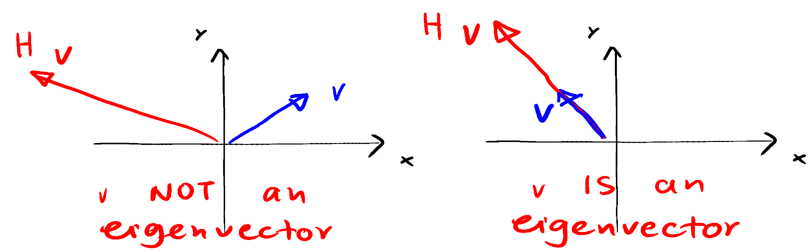
A matrix H transforms a vector **v** to the vector H**v**, so if:



Background: Eigenvectors & Eigenvalues

A non-zero vector **v** is an eigenvector of a matrix H if H**v** = λ**v.** The scalar λ is the eigenvalue associated to **v.**

A matrix H transforms a vector **v** to the vector H**v**, so if:



This also means that if **v** is an eigenvector and k ≠ 0 is a constant, then k**v** is also an eigenvector.
We will think of eigenvectors as unit-eigenvectors ||**v**||=1.

Eigenvectors of Symmetric Matrices

The hessian $H = \begin{bmatrix} \frac{\partial^2 S}{\partial x^2} & \frac{\partial^2 S}{\partial x \partial y} \\ \frac{\partial^2 S}{\partial x \partial y} & \frac{\partial^2 S}{\partial y^2} \end{bmatrix}$ is a symmetric matrix, which

means that H = H^T (where H^T denotes the transpose of H).

We said that if v is an eigenvector then Hv=λv, but when H is symmetric, then: $v^T H = \lambda v^T$.

Proof: Let $H = \begin{bmatrix} a & c \\ c & b \end{bmatrix}, v = \begin{bmatrix} x \\ y \end{bmatrix}$

$$\lambda \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a & c \\ c & b \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax+cy \\ cx+by \end{bmatrix}$$

$$\begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} a & c \\ c & b \end{bmatrix} = \begin{bmatrix} ax+cy & cx+by \end{bmatrix}$$

Eigenvectors of Symmetric Matrices

When the matrix H is symmetric ($H = H^T$ like the Hessian), its associated eigenvectors are orthogonal: $v_1^T v_2 = 0$.

Proof

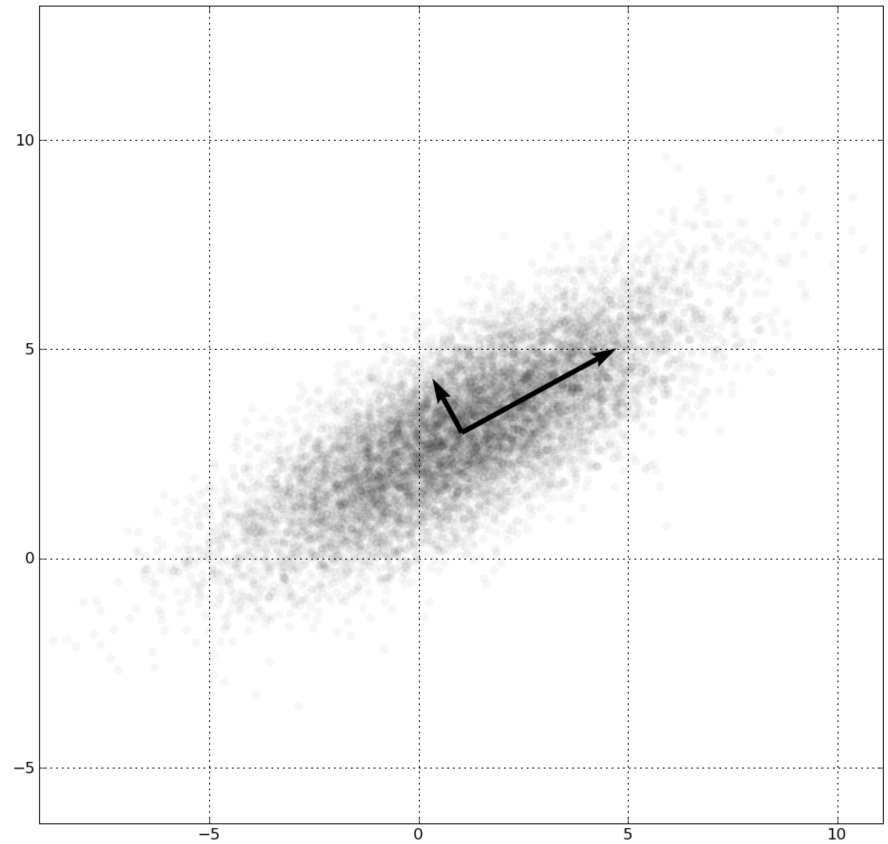
$$\lambda_1 v_1^T v_2 = (\lambda_1 v_1^T) v_2 = v_1^T H v_2 = v_1^T v_2 \cdot \lambda_2$$

$$(\lambda_1 - \lambda_2) v_1^T v_2 = 0$$

Since $\lambda_1 \neq \lambda_2$, $v_1^T v_2$ must be 0

BTW, what other matrices are symmetric

The covariance matrix!



$$\Sigma_{ij} = \mathrm{cov}(X_i, X_j) = \mathrm{E}\big[(X_i - \mu_i)(X_j - \mu_j)\big]$$

Eigenvalues & the Trace of a Matrix

The trace of a matrix (denoted as tr(H)) is the sum of the diagonal elements.

The sum of the eigenvalues of a matrix H is equal to its trace (regardless of H being symmetric).

For a 2x2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\quad a + d = \lambda_1 + \lambda_2$

This is important for computational efficiency

Eigenvalues & the Trace of a Matrix

The product of the eigenvalues of a matrix H is equal to its determinant (also regardless of H being symmetric).

For a 2x2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\quad ad - bc = \lambda_1 \cdot \lambda_2$

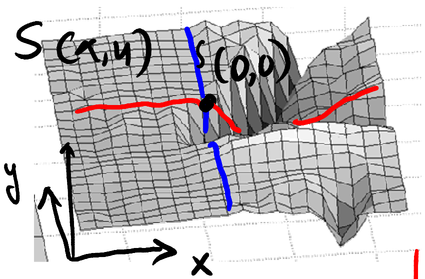
This is very important for computational efficiency

Topic 4.3:

Local analysis of 2D image patches

- Images as surfaces in 3D
- Directional derivatives
- Image Gradient
  - Painterly rendering
- Edge detection & localization
  - Gradient extrema
  - Laplacian zero-crossings
- Local geometry at image extrema
- The Image Hessian
- Eigenvectors & eigenvalues
- Corner & feature detection
  - Lowe feature detector
  - Harris/Forstner detector

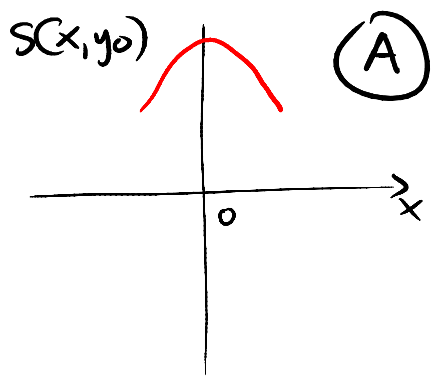
Analysis in Neighborhood of Function Extrema



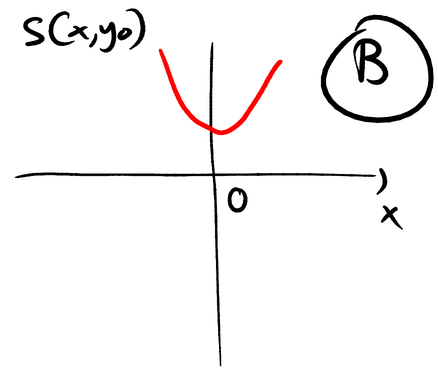
What type of extrema do we have?

cross-section along x

local maximum

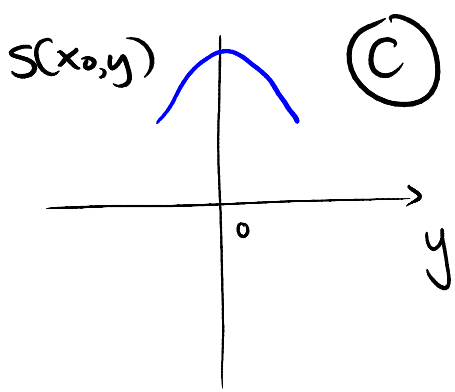


local minimum

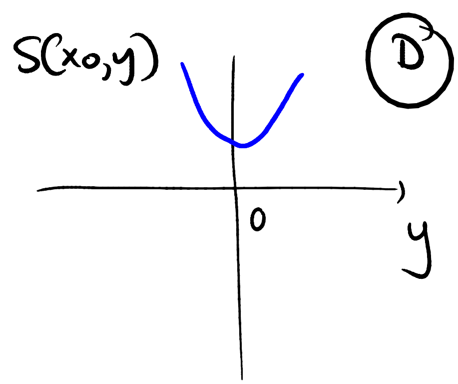


cross-section along y

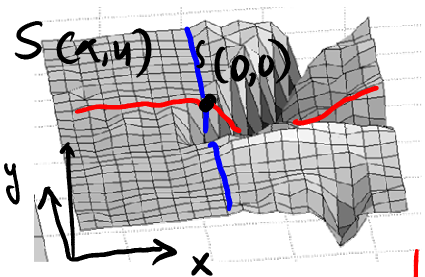
local maximum



local minimum



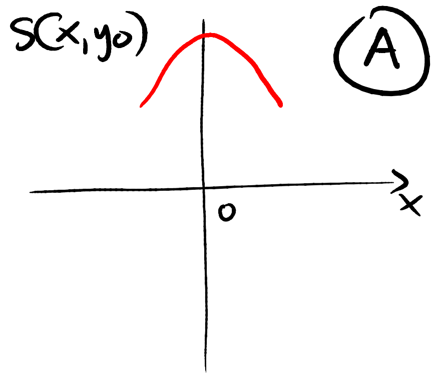
Analysis in Neighborhood of Function Extrema



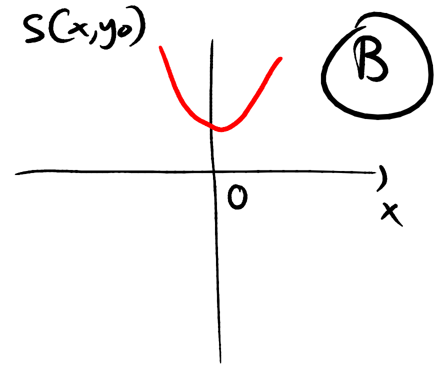
We can determine the type of extremum from tr(H) and det(H) !!

cross-section along x

local maximum

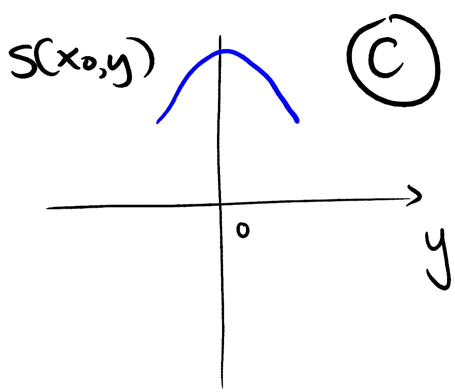


local minimum

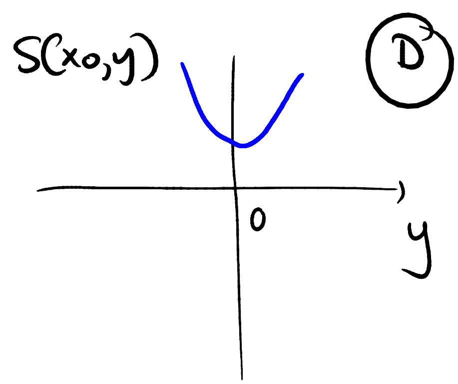


cross-section along y

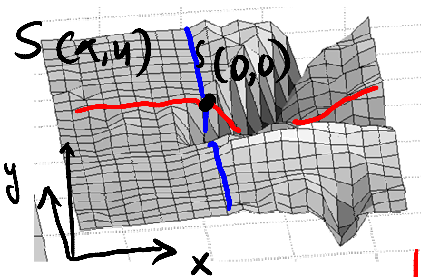
local maximum



local minimum



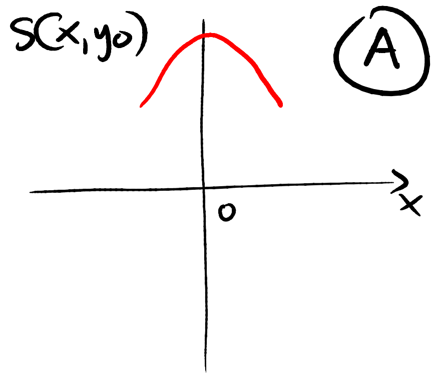
Analysis in Neighborhood of Function Extrema



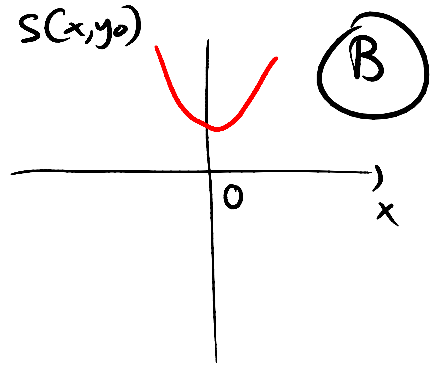
We can determine the type of extremum from tr(H) and det(H) !!

cross-section along x

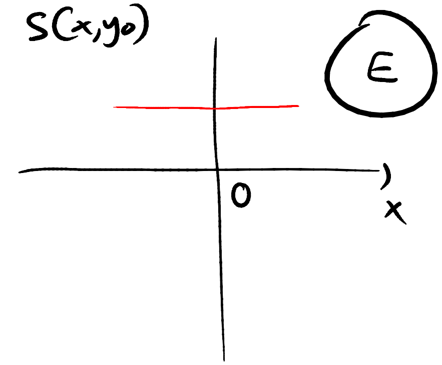
local maximum



local minimum

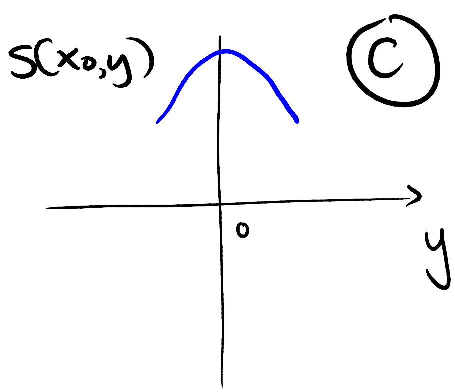


locally constant

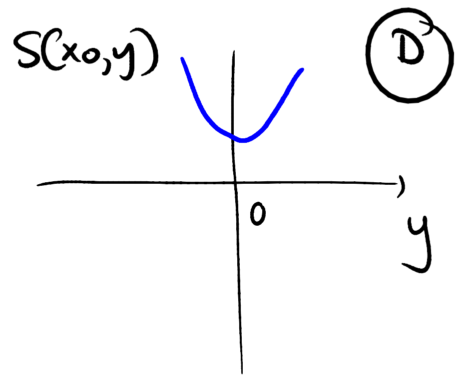


cross-section along y

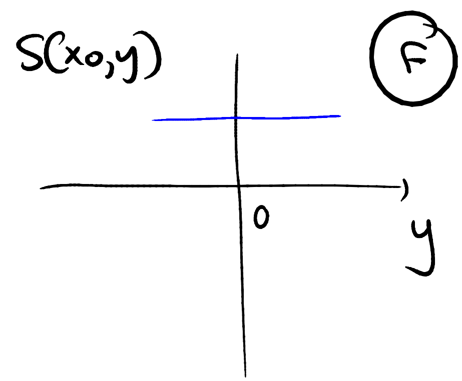
local maximum



local minimum



locally constant



The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

$\det(H) = \lambda_1 \cdot \lambda_2 > 0$?

The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

$\det(H) = \lambda_1 \cdot \lambda_2 > 0 \implies$ sign(λ1)=sign(λ2)

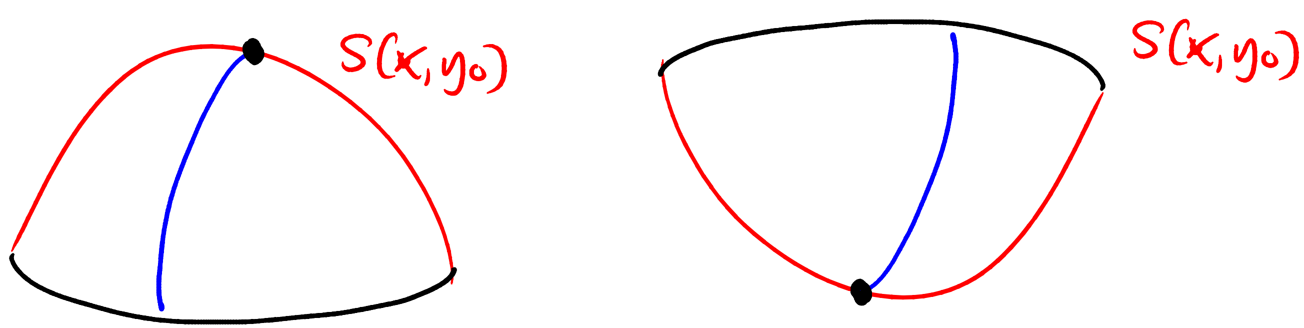
The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

$$\cdot \begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$$

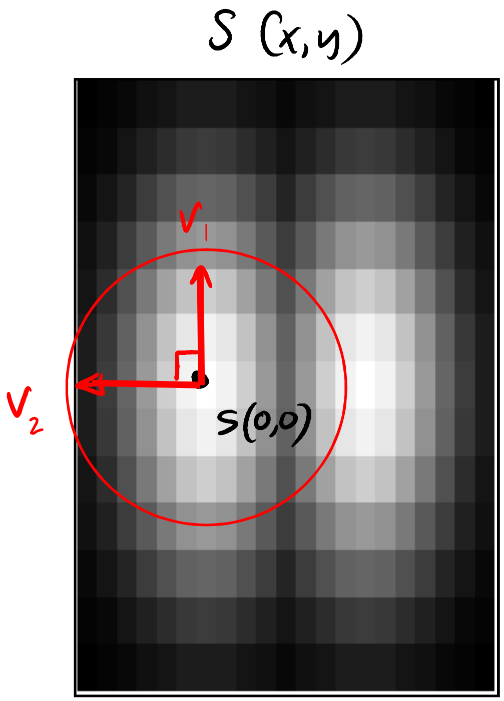
$$\cdot \begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$$

$\det(H) = \lambda_1 \cdot \lambda_2 > 0 \Rightarrow$ sign(λ1)=sign(λ2) $\Rightarrow$ Eliptical



The Hessian Matrix: Elliptical Points

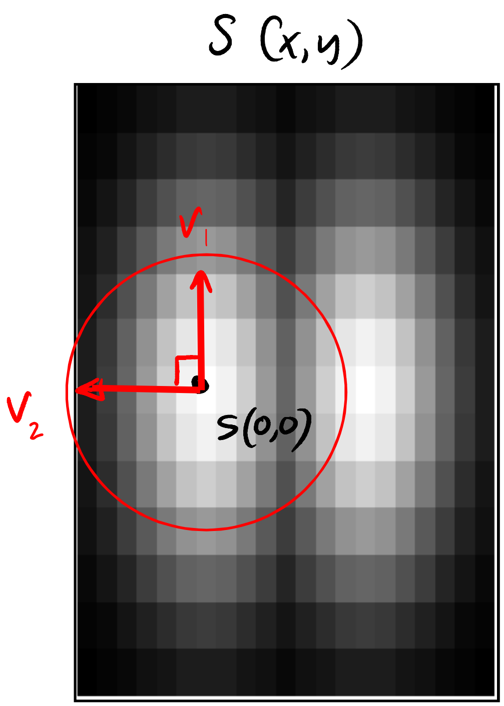
$\det(H) = \lambda_1 . \lambda_2 > 0 \Rightarrow$ sign(λ1)=sign(λ2) $\Rightarrow$ Eliptical

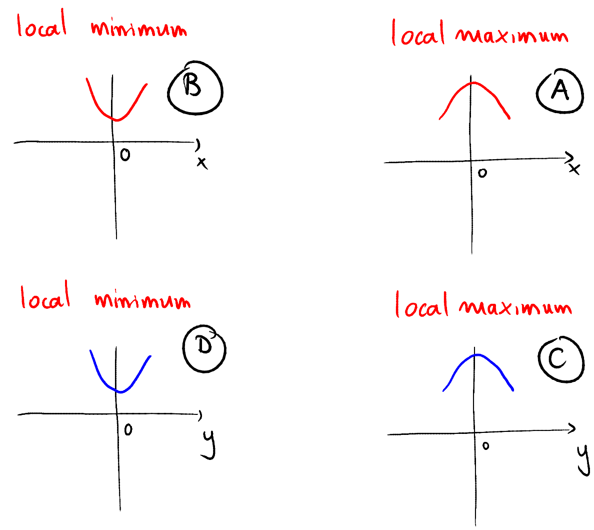


The Hessian Matrix: Elliptical Points

$\det(H) = \lambda_1 \cdot \lambda_2 > 0 \Rightarrow$ sign(λ1)=sign(λ2) $\Rightarrow$ Eliptical

$$\boxed{tr(H) = \begin{cases} >0 & \text{case B–D} \\ <0 & \text{case A–C} \end{cases}}$$





tr(H)>0 tr(H)<0

The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

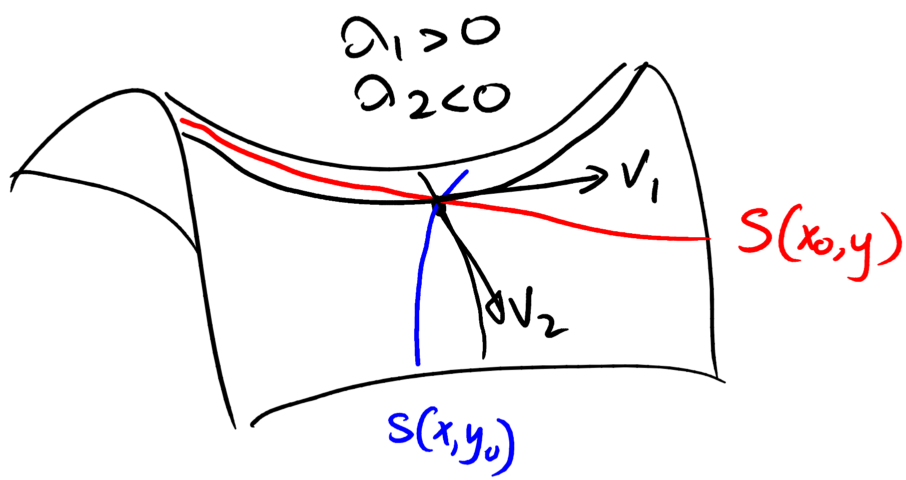
$$\det(H) = \lambda_1 \cdot \lambda_2 < 0$$

The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

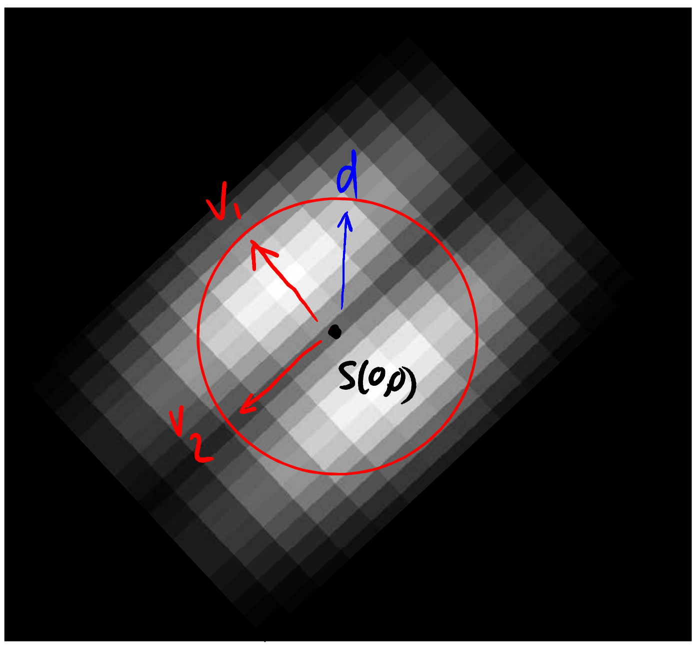
$\det(H) = \lambda_1 \cdot \lambda_2 < 0 \Rightarrow$ sign(λ1) $\neq$ sign(λ2) $\Rightarrow$ saddle points

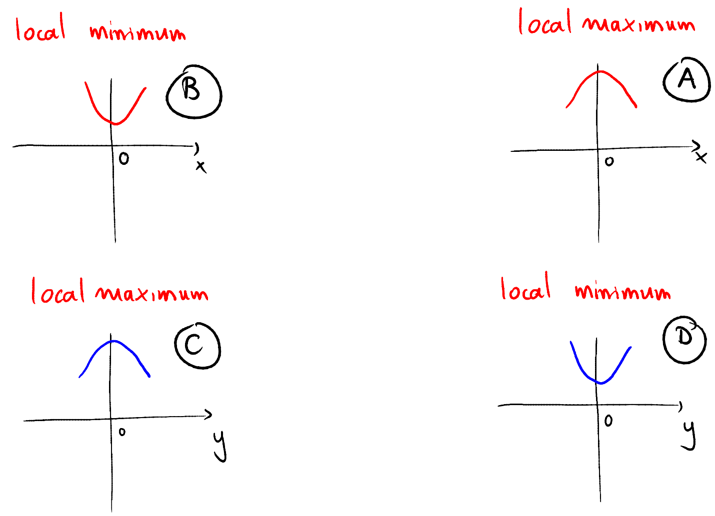


The Hessian Matrix: Elliptical Points

$\det(H) = \lambda_1 \cdot \lambda_2 < 0 \Rightarrow \mathrm{sign}(\lambda 1) \neq \mathrm{sign}(\lambda 2) \Rightarrow$ saddle points

$S(x,y)$





The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

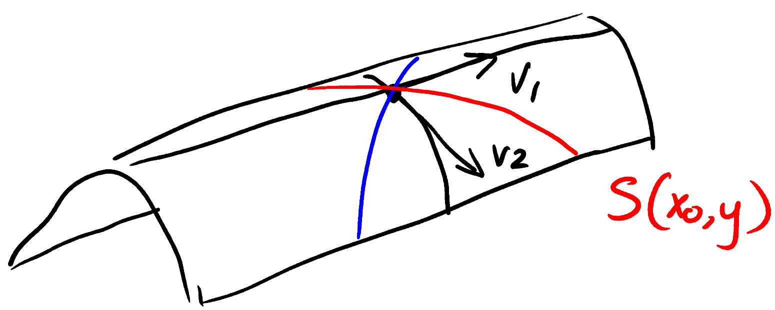
$\det(H) = \lambda_1 \cdot \lambda_2 = 0$ but $tr(H) \neq 0$

The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 =$ MAXIMAL
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 =$ MINIMAL

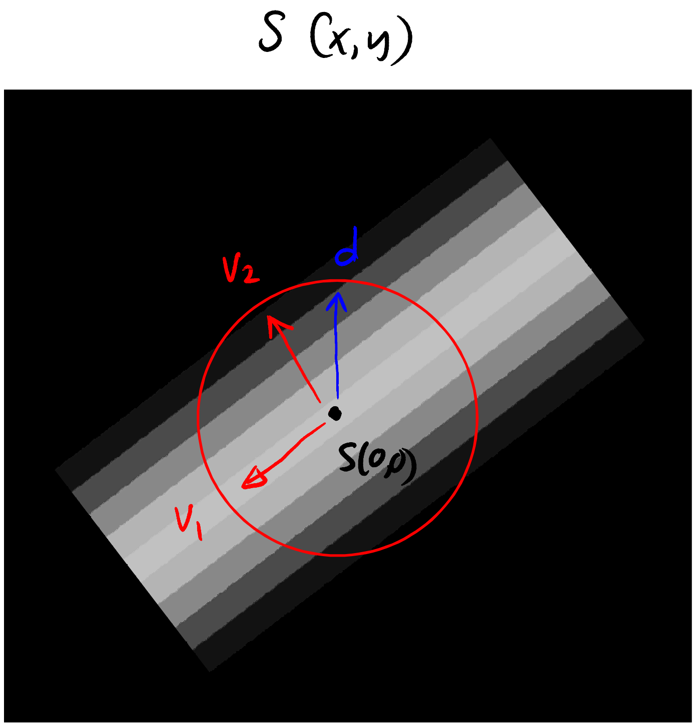
$\det(H) = \lambda_1 \cdot \lambda_2 = 0$ but $tr(H) \neq 0$



E-C, E-D, A-F, B-F

The Hessian Matrix: Elliptical Points

$\det(H) = \lambda_1 \cdot \lambda_2 = 0$ but $tr(H) \neq 0$



$$\boxed{tr(H) = \begin{cases} >0 & \text{case E-D, F-B} \\ <0 & \text{case E-C, F-A} \end{cases}}$$

Topic 4.3:

Local analysis of 2D image patches

- Images as surfaces in 3D
- Directional derivatives
- Image Gradient
  - Painterly rendering
- Edge detection & localization
  - Gradient extrema
  - Laplacian zero-crossings
- Local geometry at image extrema
- The Image Hessian
- Corner & feature detection
  - Lowe feature detector
  - Harris/Forstner detector

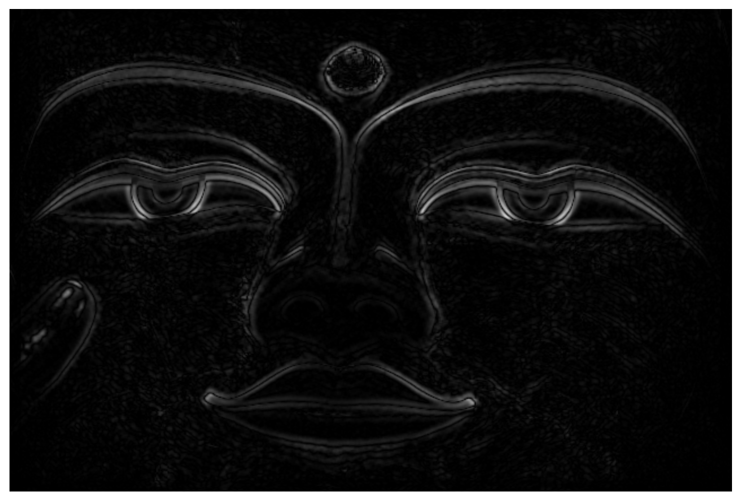
The (single-scale) Lowe Feature Detector

Goal of Lowe feature detector

Find pixels that are very different from their surroundings.

$I(x,y)$

$S(x,y) = \nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$

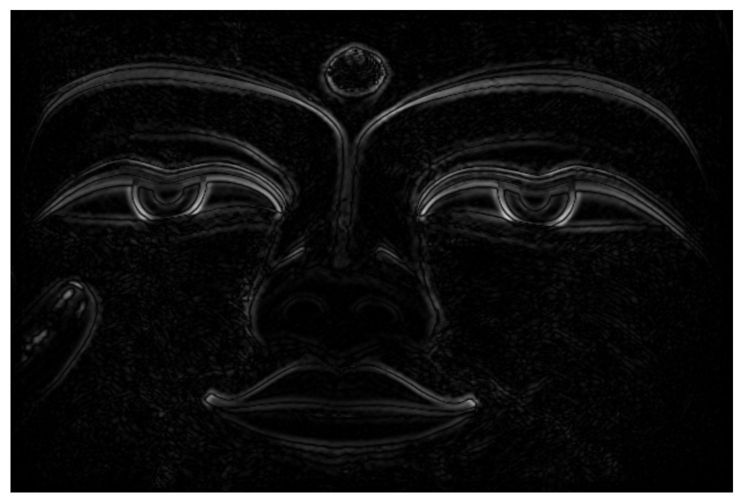


The (single-scale) Lowe Feature Detector

Idea: Apply the Hessian eigenvalue analysis to the Laplacian of an image

$I(x,y)$

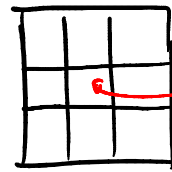
$S(x,y) = \nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$



The (single-scale) Lowe Feature Detector

Steps:

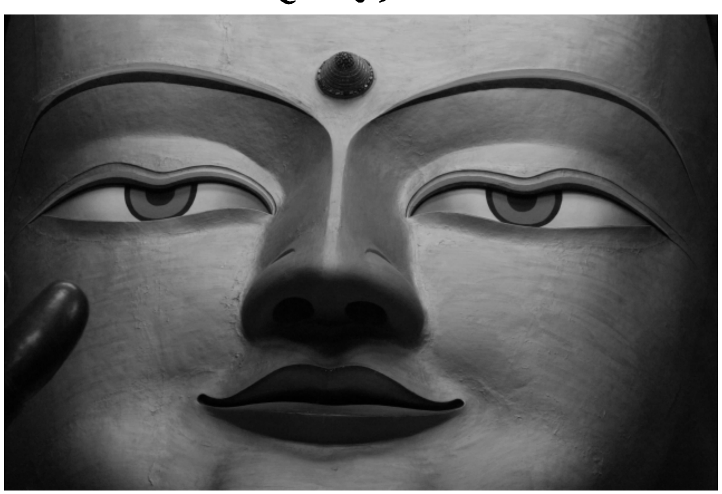
(1) Identify all pixels that are extrema of $S(x,y)$



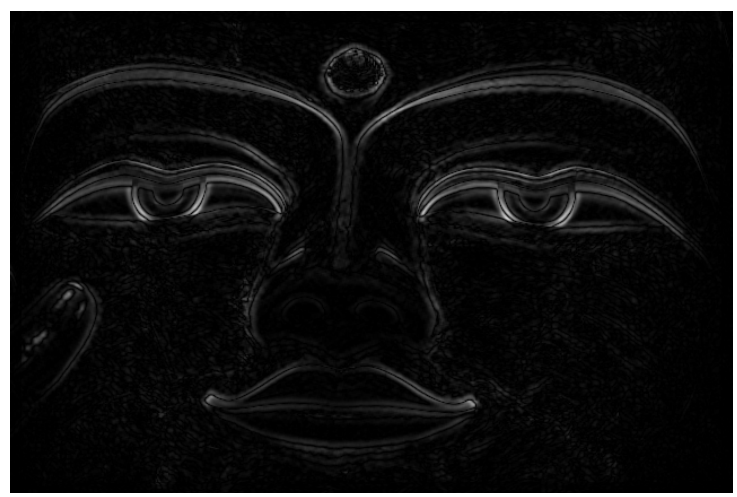
$S(x,y) >$ all 8 neighbors

$S(x,y) <$ all 8 neighbors

$I(x,y)$



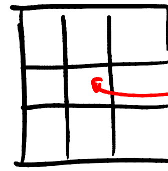
$$S(x,y) = \nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$



The (single-scale) Lowe Feature Detector

Steps:

① Identify all pixels that are extrema of $S(x,y)$



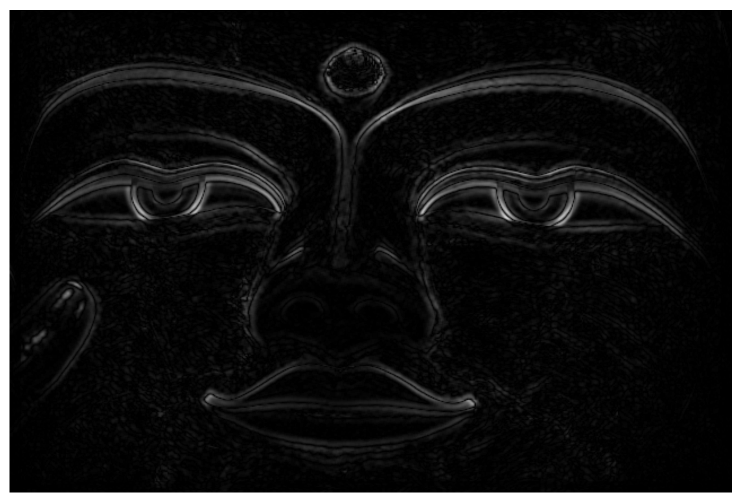
$S(x,y) >$ all 8 neighbors

$S(x,y) <$ all 8 neighbors

② Eliminate all extrema with $|S(x,y)| <$ threshold $(=0.03)$

$I(x,y)$

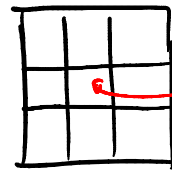
$$S(x,y) = \nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$



The (single-scale) Lowe Feature Detector

Steps:

1. Identify all pixels that are extrema of $S(x,y)$



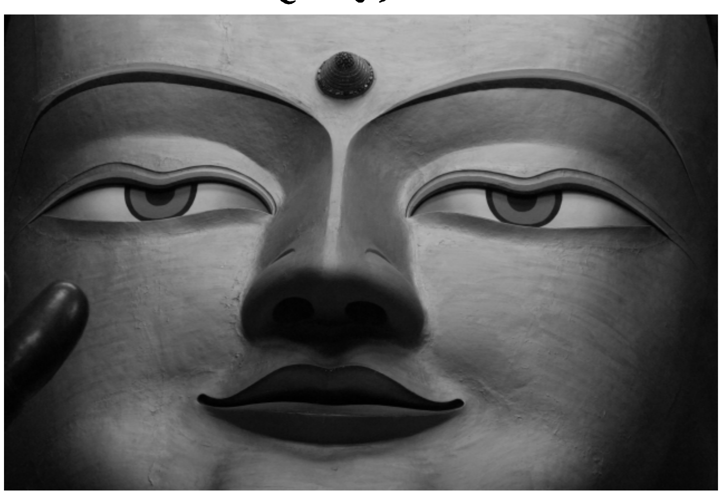
$S(x,y) >$ all 8 neighbors

$S(x,y) <$ all 8 neighbors

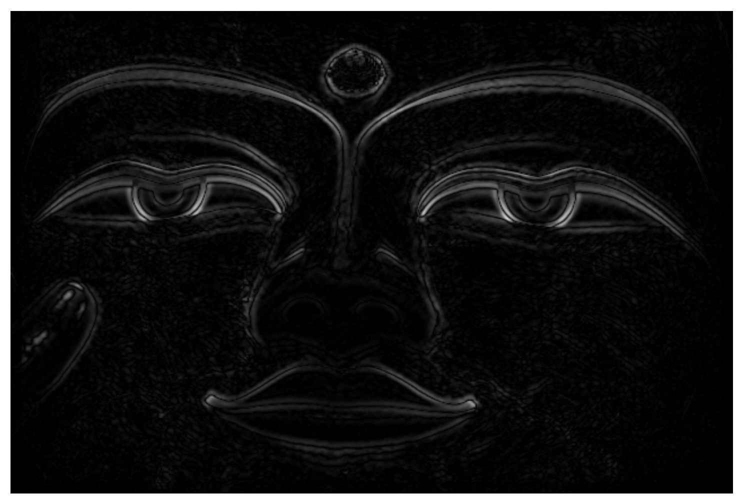
2. Eliminate all extrema with $|S(x,y)| <$ threshold $(=0.03)$

3. Eliminate all extrema whose local shape is near-cylindrical (i.e. "edge-like")

$I(x,y)$



$$S(x,y) = \nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$



The (single-scale) Lowe Feature Detector

3. Eliminate all extrema whose local shape is near-cylindrical (i.e. "edge-like")

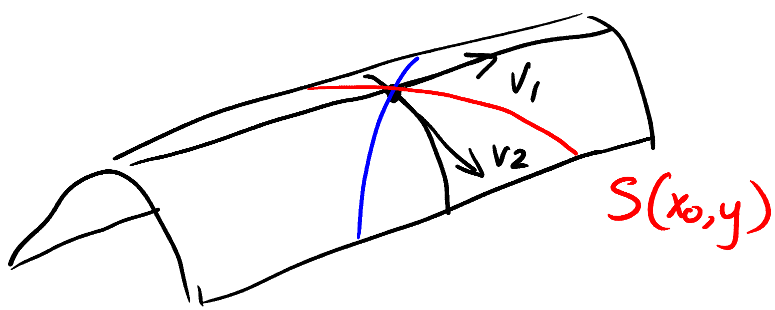
How?

The Hessian Matrix: Elliptical Points

H defines 2 orthogonal unit vectors v_1 and v_2 such that:

- $\begin{bmatrix} x \\ y \end{bmatrix} = v_1 \iff S(x,y) - S(0,0) = \lambda_1 = \text{MAXIMAL}$
- $\begin{bmatrix} x \\ y \end{bmatrix} = v_2 \iff S(x,y) - S(0,0) = \lambda_2 = \text{MINIMAL}$

$\det(H) = \lambda_1 \cdot \lambda_2 = 0$ but $tr(H) \neq 0$



E-C, E-D, A-F, B-F

The (single-scale) Lowe Feature Detector

(3) Eliminate all extrema whose local shape is near-cylindrical (i.e. "edge-like")

3a: Compute $\frac{\partial^2 S}{\partial x^2}, \frac{\partial^2 S}{\partial y^2}, \frac{\partial^2 S}{\partial x \partial y}$

3b: Compute $\mathrm{Det}(H)$ at each pixel

$$\boxed{\begin{array}{l} \det(H) = \lambda_1 \cdot \lambda_2 = 0 \\ \text{in practice:} \\ \text{check } |\lambda_1| \gg |\lambda_2| \text{ or } |\lambda_2| \gg |\lambda_1| \end{array}}$$

The (single-scale) Lowe Feature Detector

(3) Eliminate all extrema whose local shape is near-cylindrical (i.e. "edge-like")

3a: Compute $\frac{\partial^2 S}{\partial x^2}, \frac{\partial^2 S}{\partial y^2}, \frac{\partial^2 S}{\partial x \partial y}$

3b: Compute $Det(H)$ at each pixel

$$\boxed{\det(H) = \lambda_1 \cdot \lambda_2 = 0}$$

in practice:

check $|\lambda_1| \gg |\lambda_2|$ or $|\lambda_2| \gg |\lambda_1|$

The (single-scale) Lowe Feature Detector

③ Eliminate all extrema whose local shape is near-cylindrical (i.e. "edge-like")

But what we really want to know is if $\lambda 1$ is much bigger compared terms to $\lambda 2$, as in "r" times bigger, so if we do:

$$\lambda_1 = r\lambda_2$$

We can use the following equation to test for the ratio:

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\lambda_2 + r\lambda_2)^2}{r \cdot \lambda_2^2} = \frac{(r+1)^2}{r^2}$$

The (single-scale) Lowe Feature Detector

3. Eliminate all extrema whose local shape is near-cylindrical (i.e. "edge-like")

3a: Compute $\frac{\partial^2 S}{\partial x^2}, \frac{\partial^2 S}{\partial y^2}, \frac{\partial^2 S}{\partial x \partial y}$

3b: Compute $Det(H)$ at each pixel

3c: Compute $Tr(H)$

3d: Keep only pixels with

$$\frac{Tr(H)^2}{Det(H)} < \left(\frac{r+1}{r}\right)^2$$

usually: $r=10$

The (single-scale) Lowe Feature Detector

Result: Top 500 un-eliminated pixels
(ranked according to $|\nabla^2 I|$)

Topic 4.3:

Local analysis of 2D image patches

- Images as surfaces in 3D
- Directional derivatives
- Image Gradient
  - Painterly rendering
- Edge detection & localization
  - Gradient extrema
  - Laplacian zero-crossings
- Local geometry at image extrema
- The Image Hessian
- Corner & feature detection
  - Lowe feature detector
  - Harris/Forstner detector

The Harris/Forstner Corner Detector

Goal of Harris / Forstner corner detector:

Find pixels that are "very different" from their neighborhood

Idea:

Equivalent eigenvalue analysis on a slightly different $S(x,y)$ functional

The Harris/Forstner Corner Detector

The Harris/Forstner Corner Detection looks for (dis)-similarity between the center pixel and its neighborhood, literally

$$S(x,y) = \sum_u \sum_v w(u,v)\ \left(I(u+x,v+y) - I(u,v)\right)^2$$

The center pixels are more heavily weighted than the rest.

Now, because $I(u+x,v+y)$ can be approximated using a Taylor expansion

$$I(u+x,v+y) \approx I(u,v) + I_x(u,v)x + I_y(u,v)y$$

The Harris/Forstner Corner Detector

The dissimilarity metric

$$S(x,y) = \sum_u \sum_v w(u,v)\ \left(I(u+x, v+y) - I(u,v)\right)^2$$

can then be written as the approximation:

$$S(x,y) \approx \sum_u \sum_v w(u,v)\ \left(I_x(u,v)x + I_y(u,v)y\right)^2$$

or equivalently:

$$S(x,y) \approx \begin{pmatrix} x & y \end{pmatrix} A \begin{pmatrix} x \\ y \end{pmatrix}$$

with: $A = \sum_u \sum_v w(u,v) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$

The Harris/Forstner Corner Detector

The Harris/Forstner Corner detector looks for patches that are “very dissimilar” from their neighbors!

The Harris/Forstner Corner Detector

The Harris/Forstner Corner detector looks for patches that are "very dissimilar" from their neighbors!

Topic 4.4:

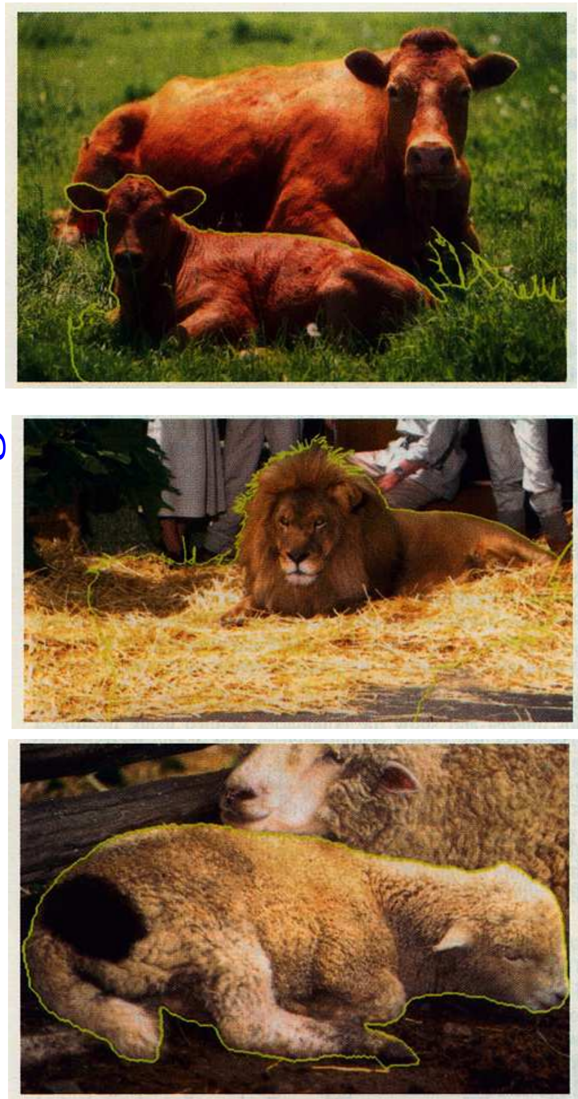
Application: Intelligent Scissors

- Assigning cost to “links” between pixels
- Contour tracing as a shortest-path problem

Image Scissoring: Motivation

By scissoring portions of one or more images & pasting them together we can create new, composite images

source images



composite image

Image Scissoring: Requirements

Interactive operation

"User is always right"

Scissoring system must allow user to select arbitrary regions

Scissoring operations must be performed efficiently

Scissoring interface must be simple & easy to use

Image Scissoring

In contrast the manual approach requires the user to manually delineate every single image pixel that defines the region boundary

What is proposed here is: Intelligent Scissors approach ("livewire") developed by Eric Mortensen & presented at SIGGRAPH'95

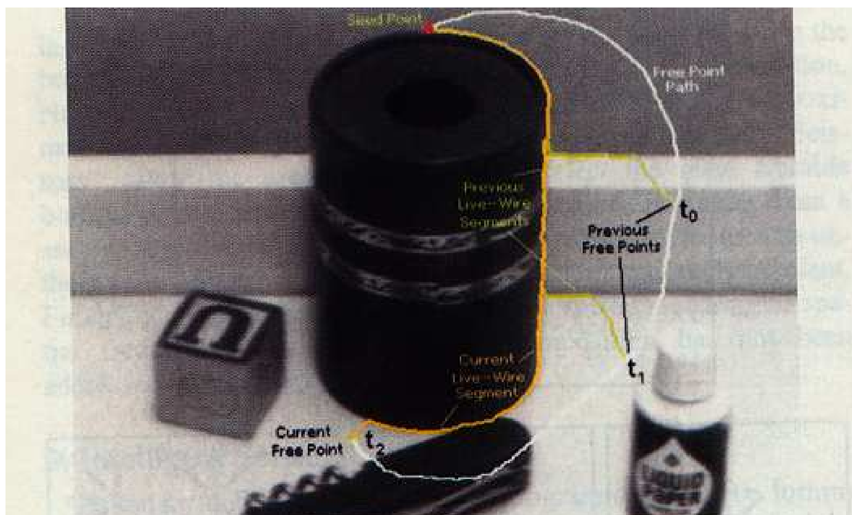


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Intelligent Scissors: Operation

User loads image & specifies a "seed" point on boundary that must be outlined

User then positions mouse close to object boundary

System automatically creates a "live-wire" that connects seed & current mouse position & follows boundary as much as possible

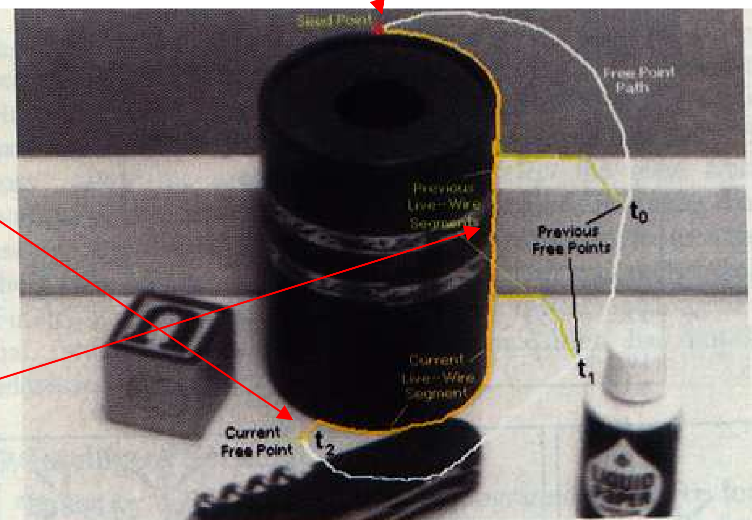


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Live wire updated whenever mouse moves

Intelligent Scissors: Basic Idea

Approach answers one basic question:

- How should we define a path from seed to mouse that follows an object boundary as closely as possible?

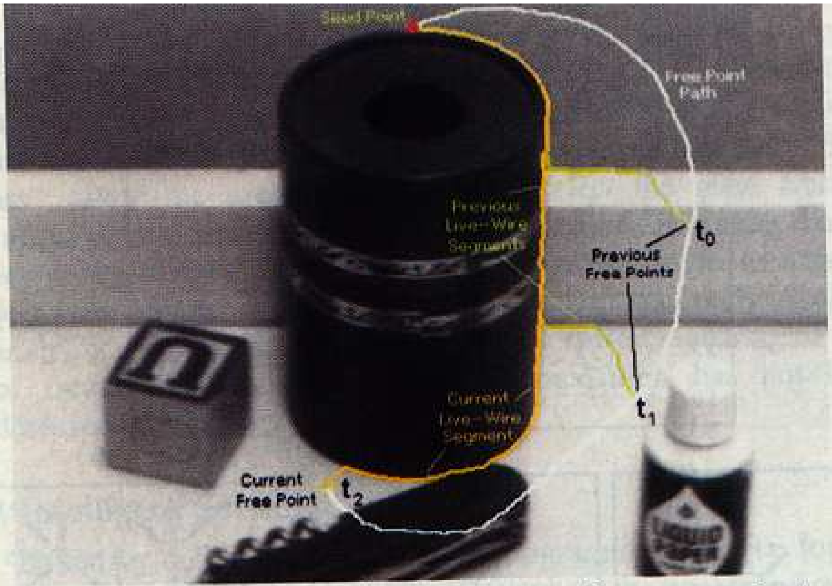


Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

- Answer: Define a path that is as close as possible to image edges

Intelligent Scissors: Basic Idea

Approach taken in intelligent scissors attempts to exploit user interaction while avoiding the need to detect edges corresponding to object boundaries very accurately

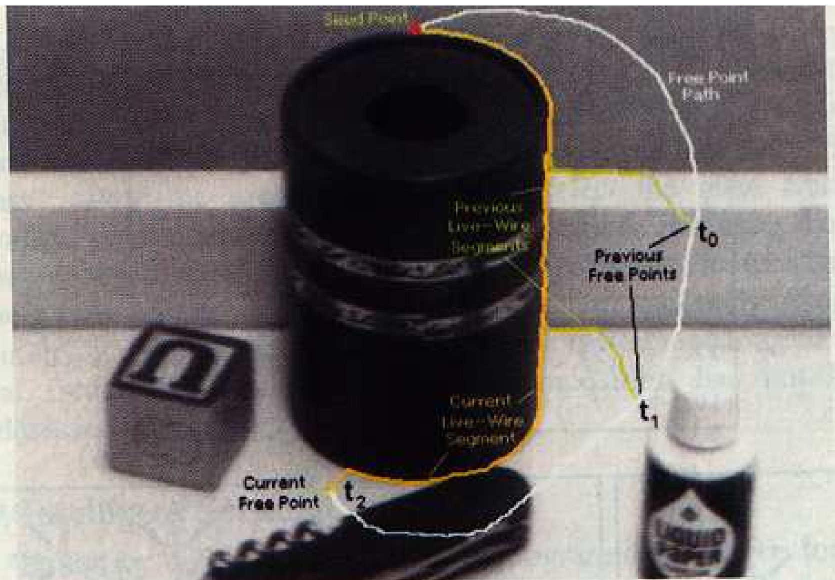
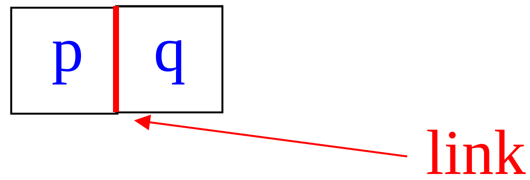
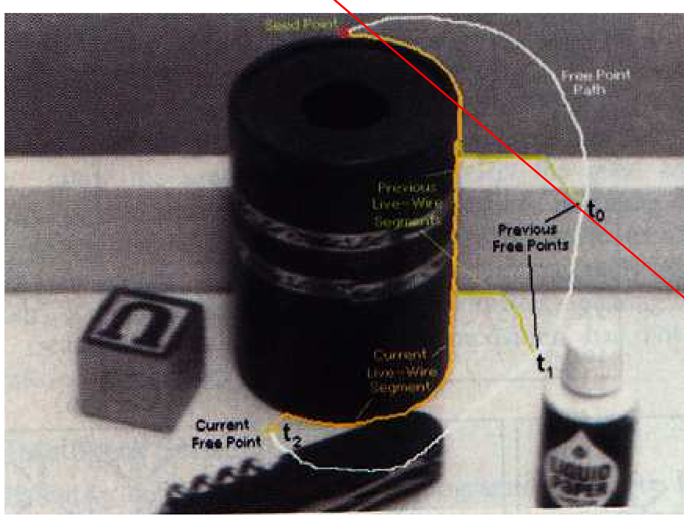


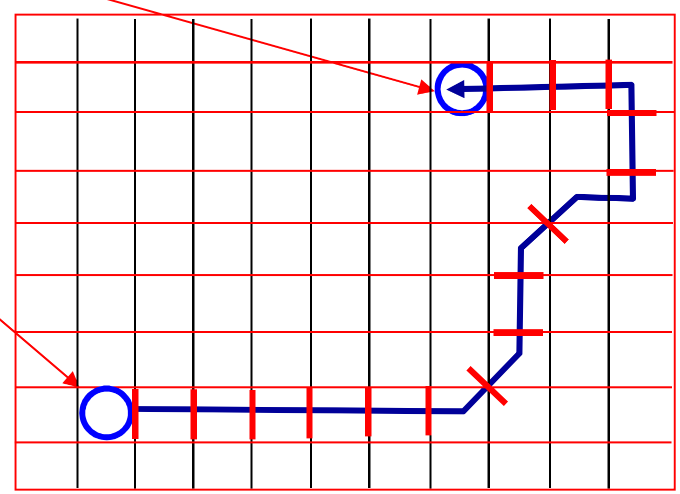
Figure 2: *Image demonstrating how the live-wire segment adapts and snaps to an object boundary as the free point moves (via cursor movement). The path of the free point is shown in white. Live-wire segments from previous free point positions (t_0, t_1, and t_2) are shown in green.*

Intelligent Scissors: Basic Idea



- Every pair of neighboring pixels is called a link and is assigned an "edgeness" weight
- Link weights defined so that pixel links along an edge have very low weights
- To connect seed & mouse positions, choose the path that minimizes the total weight of links along the path

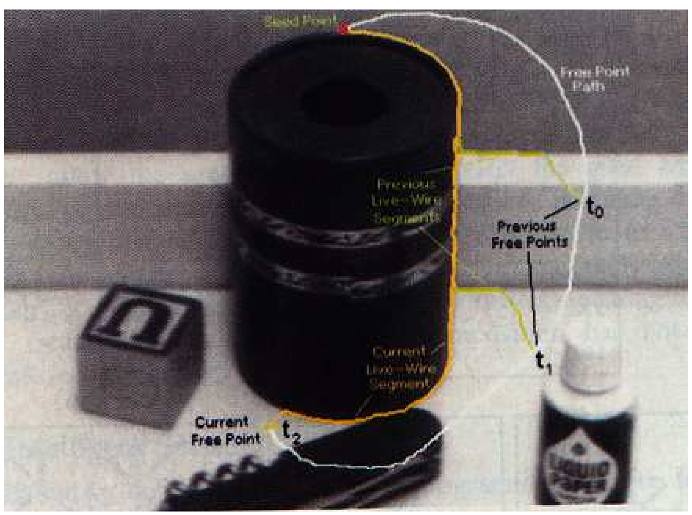


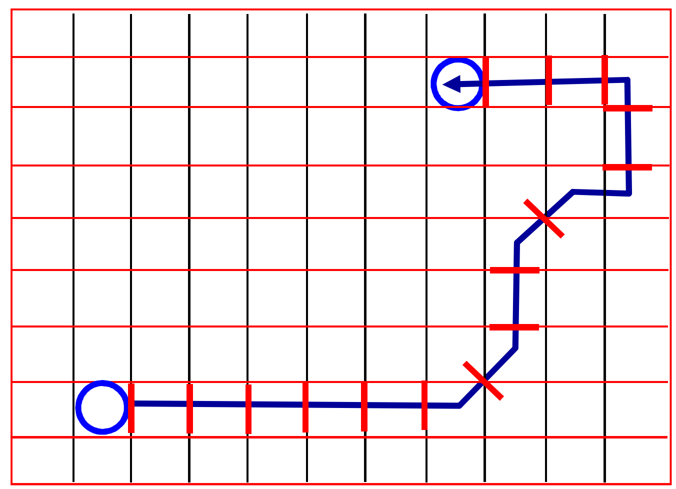


Intelligent Scissors: Basic Idea

Two questions must be answered to fully specify the algorithm:

- How do we assign a weight to a link?
- How do we find the lowest-cost path between any two image pixels?

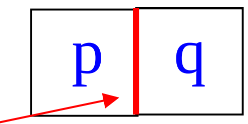




Intelligent Scissors: Weight Assignment

Path: a sequence of adjacent pixels in the image

Link: a pair of adjacent pixels along the path



Given a link defined by pixels p & q, its weight is defined to be

$$\boxed{l(p,q) = 0.43 f_Z(q) + 0.43 f_D(p,q) + 0.14 f_G(q)}$$

$f_Z(q) = 0$ if the Laplacian has a zero-crossing at q

$f_Z(q) = 1$ otherwise

term that penalizes links not consistent with the gradient direction at p and q

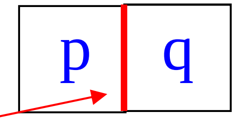
$$f_G(q) = 1 - \frac{|\nabla I|}{\max(|\nabla I|)}$$

largest value over the image

Intelligent Scissors: Weight Assignment

Path: a sequence of adjacent pixels in the image

Link: a pair of adjacent pixels along the path



Given a link defined by pixels p & q, its weight is defined to be

$$\boxed{l(p,q) = 0.43 f_Z(q) + 0.43 f_D(p,q) + 0.14 f_G(q)}$$

$$f_G(q) = 1 - \frac{|\nabla I|}{\max(|\nabla I|)}$$

largest value
over the image

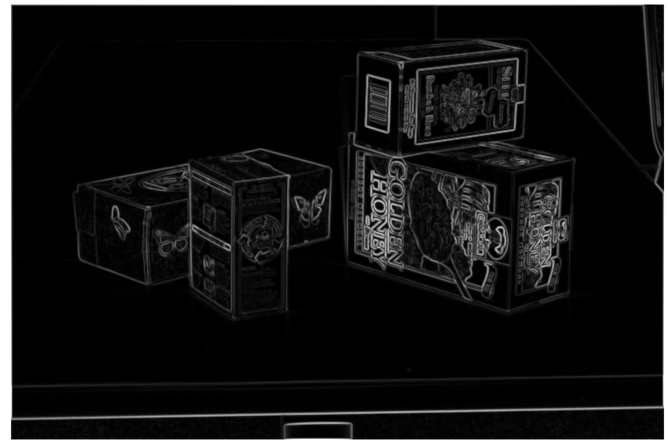
Weight Assignment: Gradient Term

Recall:

- can detect edges where 1st derivative is high

$$I_x \quad \begin{array}{|ccc|}\hline -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \\ \hline\end{array}$$

$$I_y \quad \begin{array}{|ccc|}\hline -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \\ \hline\end{array}$$



gradient magnitude

$$\left|\nabla I\right| = \left|(I_x, I_y)\right| = \sqrt{I_x^{\,2} + I_y^{\,2}}$$

$$f_G(q) = 1 - \frac{|\nabla I|}{\max(|\nabla I|)}$$

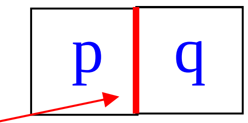
largest value
over the image

- high gradients produce low costs
- scaled by largest gradient in image, to lie in [0,1]

Weight Assignment: Laplacian Term

Path: a sequence of adjacent pixels in the image

Link: a pair of adjacent pixels along the path



Given a link defined by pixels p & q, its weight is defined to be

$$\boxed{l(p,q) = 0.43\, f_Z(q) + 0.43\, f_D(p,q) + 0.14\, f_G(q)}$$

$f_Z(q) = 0$ if the Laplacian has a zero-crossing at q

$f_Z(q) = 1$ otherwise

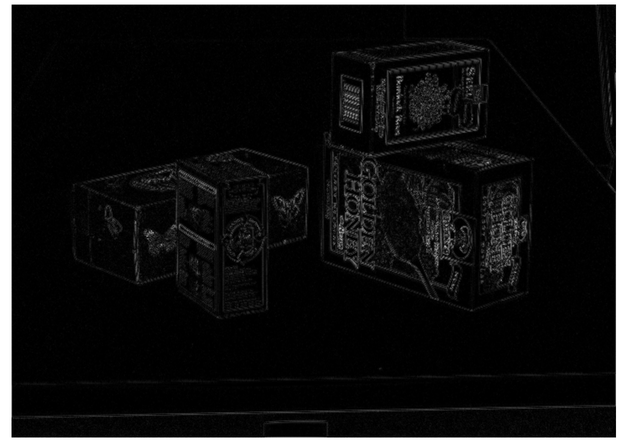
Intelligent Scissors: Weight Assignment

Recall:

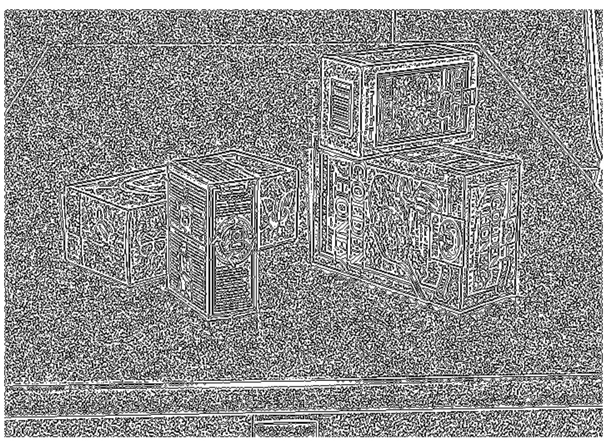
- can detect edges where 2nd derivative is zero (inflection points)
- find zero-crossings instead (sign change with 8-neighbors)

$$L = I_{xx} + I_{yy}$$

$$\boxed{\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}}$$



Laplacian $L = I_{xx} + I_{yy}$



$f_Z(q)$

$f_Z(q) = 0$ if the Laplacian has a zero-crossing at q

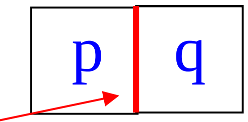
$f_Z(q) = 1$ otherwise

- zero-crossings produce low costs
- many zero-crossings are due to noise

Intelligent Scissors: Weight Assignment

Path: a sequence of adjacent pixels in the image

Link: a pair of adjacent pixels along the path



Given a link defined by pixels p & q, its weight is defined to be

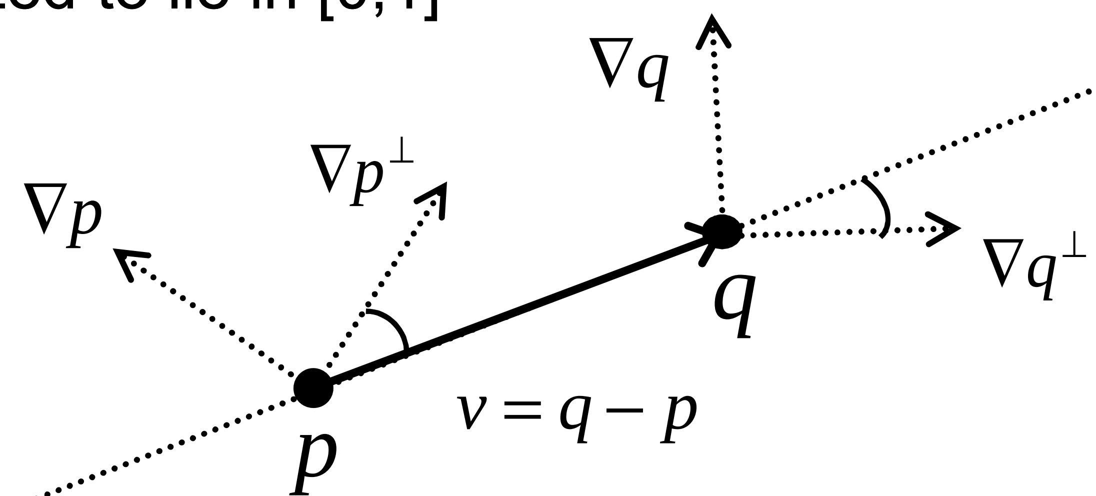
$$\boxed{l(p,q) = 0.43\, f_Z(q) + 0.43\, f_D(p,q) + 0.14\, f_G(q)}$$

term that penalizes links not consistent with the gradient direction at p and q

Weight Assignment: Direction Term

gradient direction term $f_D(p,q)$

- penalizes paths that do not follow edges in the image
- penalizes sharp changes in path direction (creases)
- normalized to lie in [0,1]



$$f_D(p,q) = \left(\text{angle}(v, \nabla p^{\perp}) + \text{angle}(v, \nabla q^{\perp})\right) / \pi$$

$$\text{angle}(u,w) = \arccos\left(\frac{|u \cdot w|}{|u||w|}\right) \in [0, \pi/2]$$

Intelligent Scissors: Path Optimizer

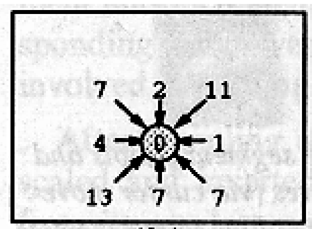
Path optimization formulated as a graph search algorithm that computes the minimum-cost path from seed to all other image pixels (use Dijkstra's algorithm)

Weights

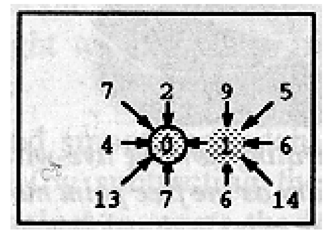
11	13	12	9	5	8	3	1	2	4	10
14	11	7	4	2	5	8	4	6	3	8
11	6	3	5	7	9	12	11	10	7	4
7	4	6	11	13	18	17	14	8	5	2
6	2	7	10	15	15	21	19	8	3	5
8	3	4	7	9	13	14	15	9	5	6
11	5	2	8	3	4	5	7	2	5	9
12	4	2	1	5	6	3	2	4	8	12
10	9	7	5	9	8	5	3	7	8	15

seed

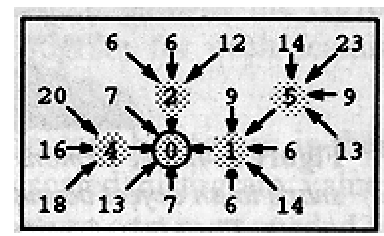
Step 1



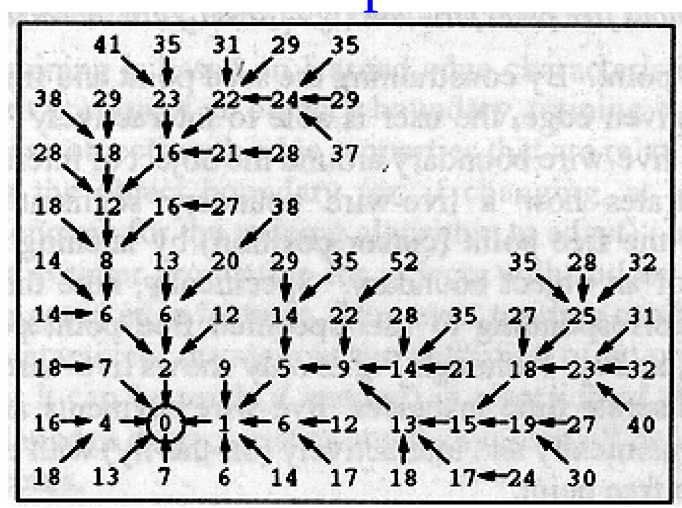
Step 2



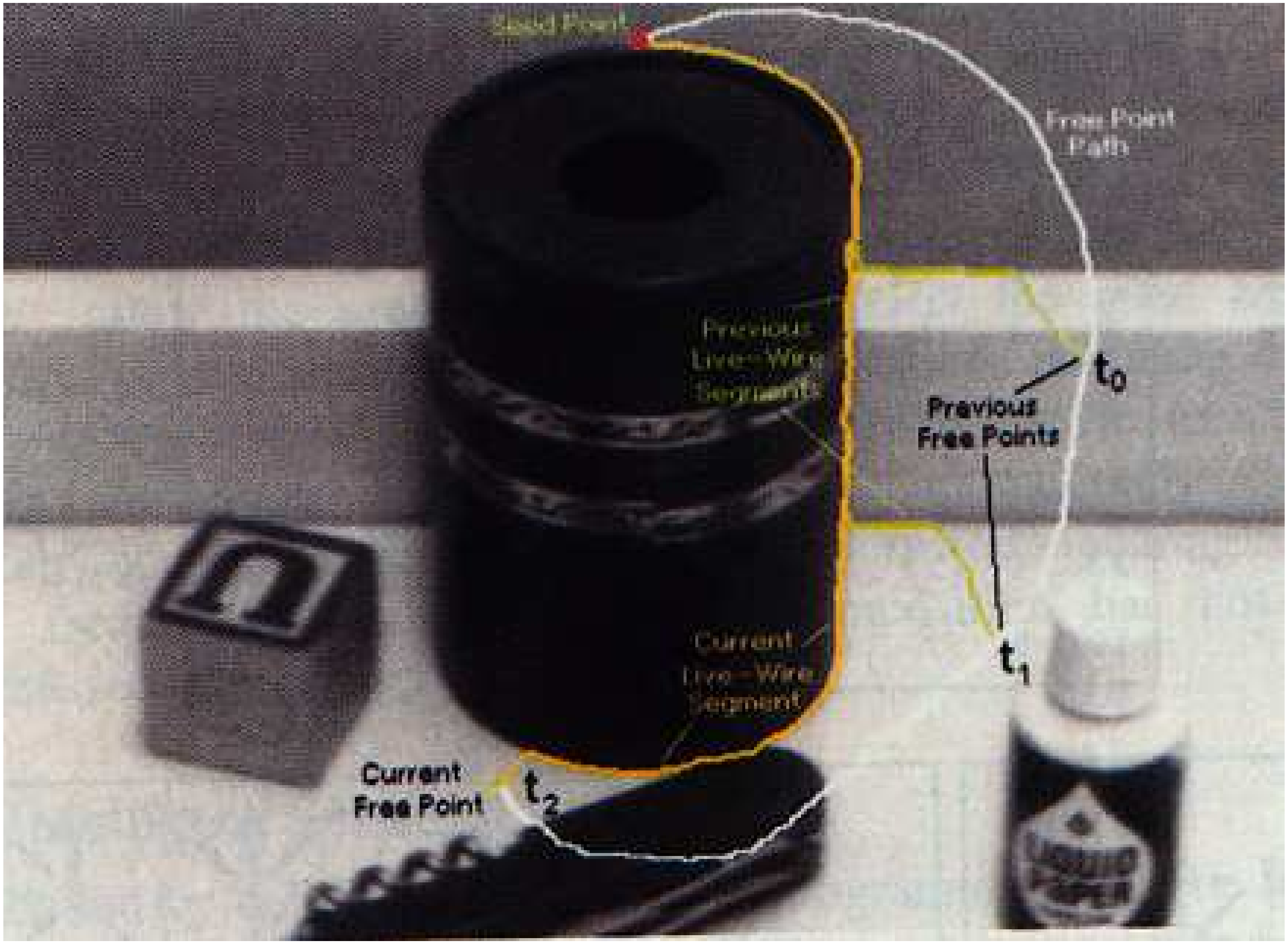
Step 3



Step n



Intelligent Scissors: Results

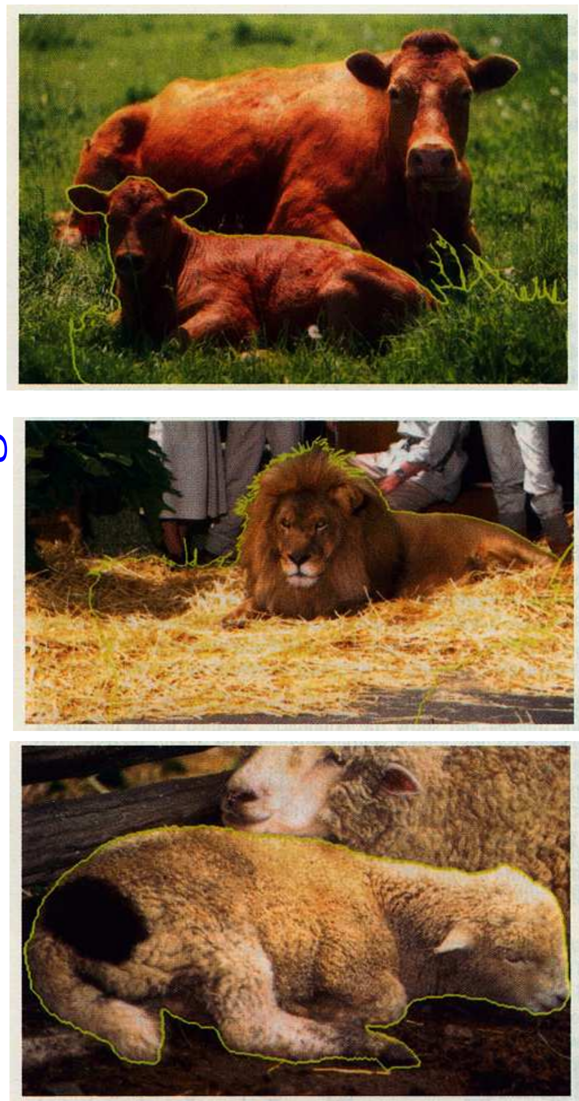


Intelligent Scissors: Results

Intelligent Scissors: Results

By scissoring portions of one or more images & pasting them together we can create new, composite images

source images



composite image