

COMS30030
Image Processing and Computer Vision

Motion – Modelling

Andrew Calway

andrew@cs.bris.ac.uk

Motion – Important Perceptual Cue



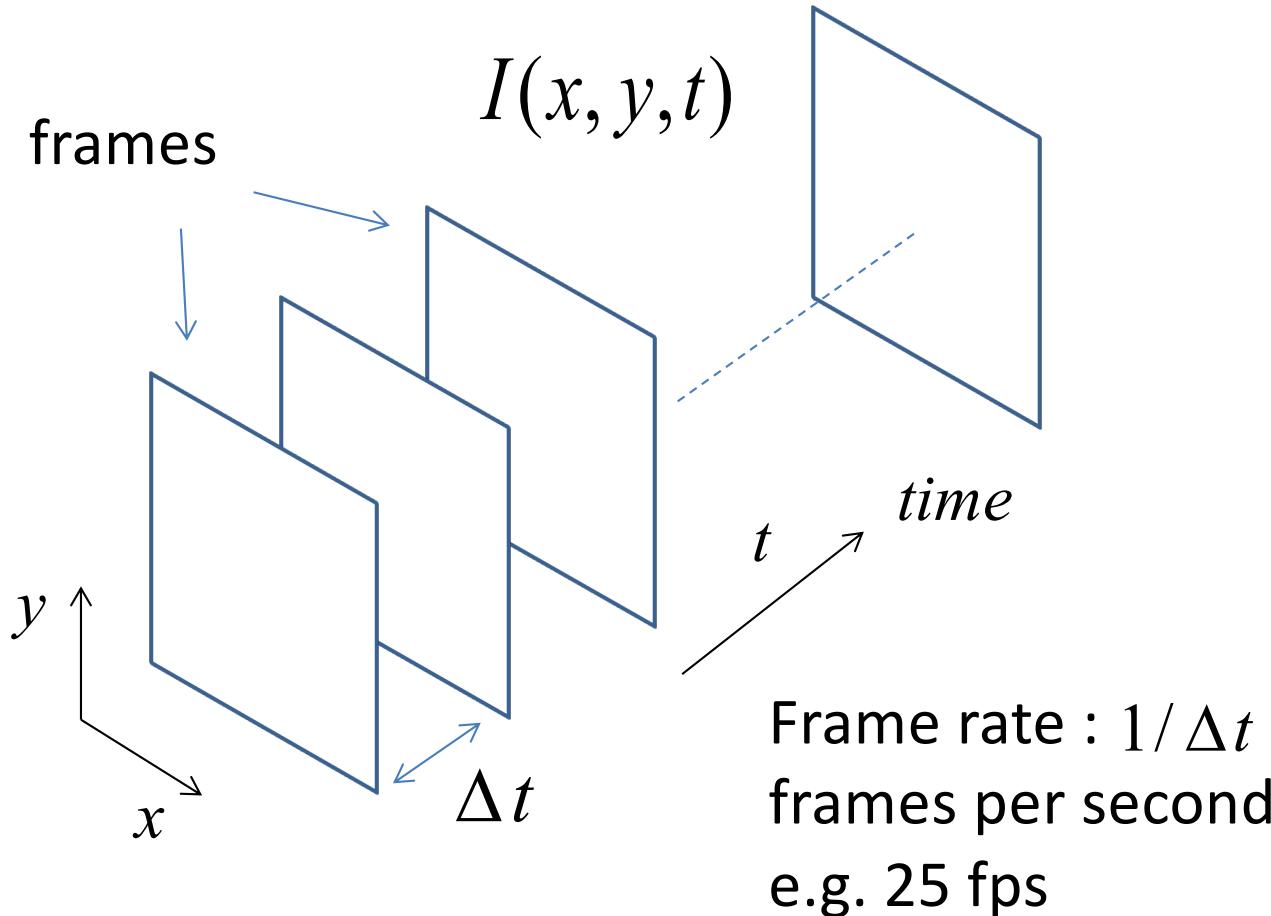
We are going to look at

- Modelling 2-D motion fields
- Optical flow
- The optical flow equation (OFE)
- Motion estimation
 - Lucas and Kanade method

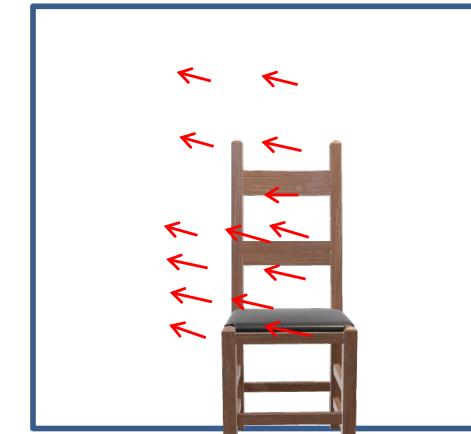
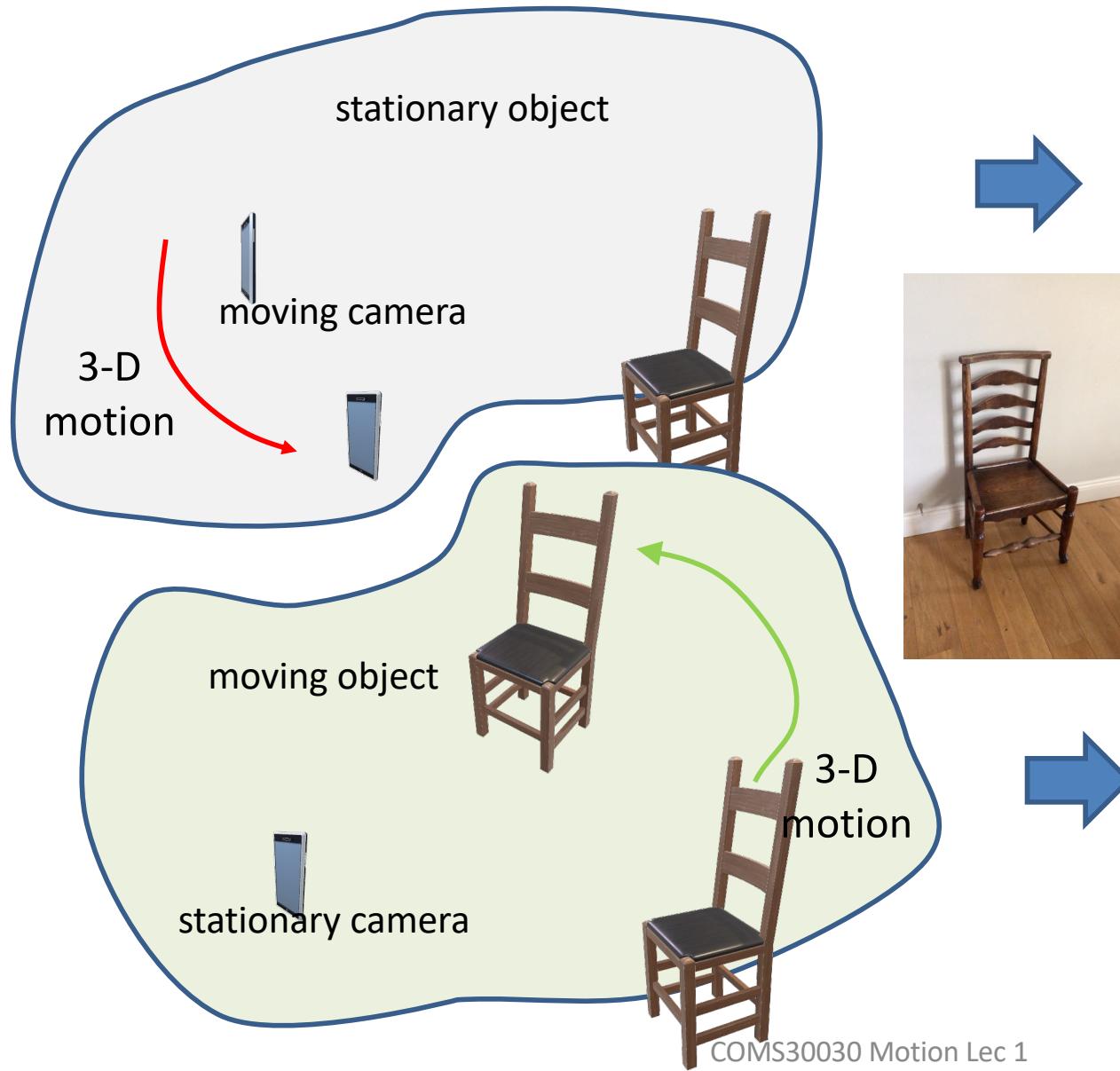


Video Sequences

First motion
picture camera
Kinetograph



Modelling 2-D Motion Fields



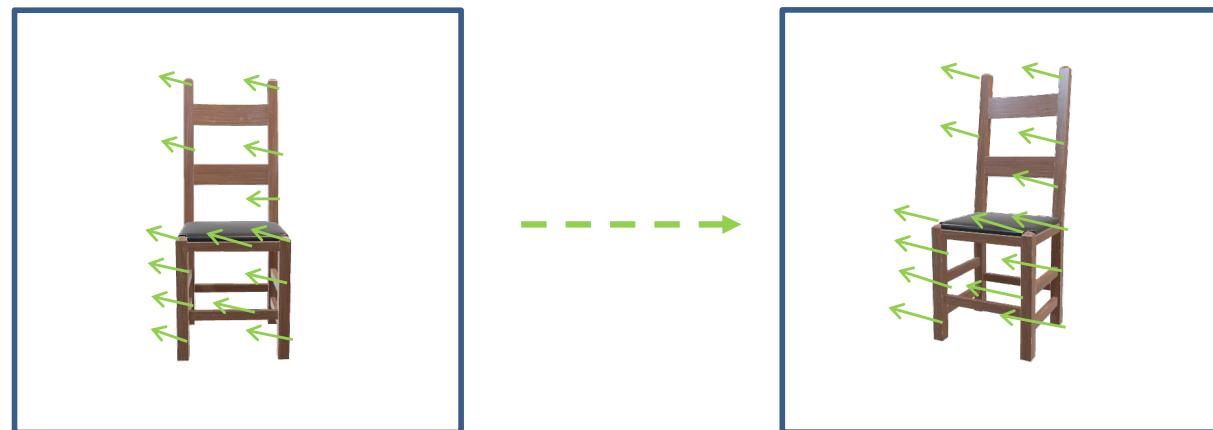
Incremental 2-D motion fields



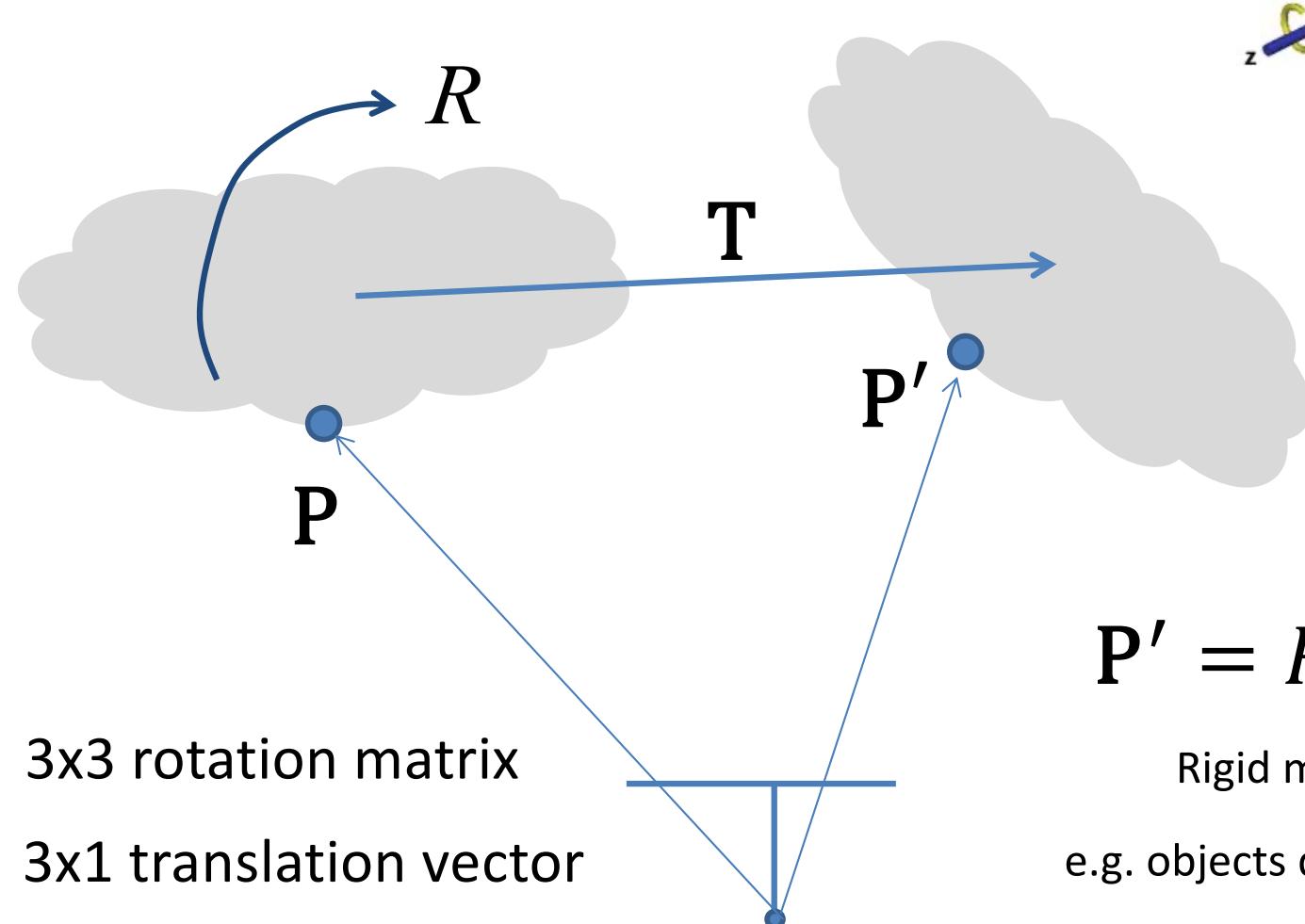
Incremental 2-D Motion Fields

What is the relationship between the:

- Position of a 3-D point and its 2-D motion?
- 2-D motion of different 3-D points?
- 3-D motion of a 3-D point and its 2-D motion?



3-D Rigid Motion

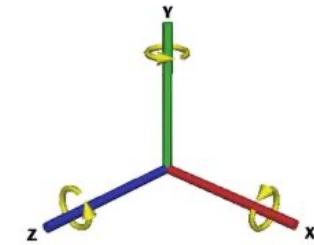


$$P' = RP + T$$

Rigid motion

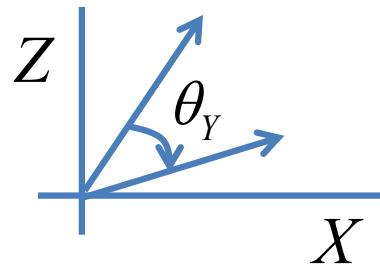
e.g. objects do not deform

Rotation Matrices



$R = R_X R_Y R_Z$ (for example) : Rotations about X , Y and Z axes

$$R_Y \mathbf{P} = \begin{bmatrix} \cos \theta_Y & 0 & \sin \theta_Y \\ 0 & 1 & 0 \\ -\sin \theta_Y & 0 & \cos \theta_Y \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X \cos \theta_Y + Z \sin \theta_Y \\ Y \\ Z \cos \theta_Y - X \sin \theta_Y \end{bmatrix}$$



For small θ_Y :
 $\cos \theta_Y \approx 1$
 $\sin \theta_Y \approx \theta_Y$

NB : for small θ_Y

$$R_Y \approx \begin{bmatrix} 1 & 0 & \theta_Y \\ 0 & 1 & 0 \\ -\theta_Y & 0 & 1 \end{bmatrix}$$

NB : for small $\theta_X, \theta_Y, \theta_Z$

$$R \approx \begin{bmatrix} 1 & -\theta_Z & \theta_Y \\ \theta_Z & 1 & -\theta_X \\ -\theta_Y & \theta_X & 1 \end{bmatrix}$$

3-D Motion Field

$$\mathbf{V} = \lim_{\Delta t \rightarrow 0} \{\mathbf{P}' - \mathbf{P} = (R - I)\mathbf{P} + \mathbf{T}\}$$

$$\mathbf{P}' = R\mathbf{P} + \mathbf{T}$$

For small angles:

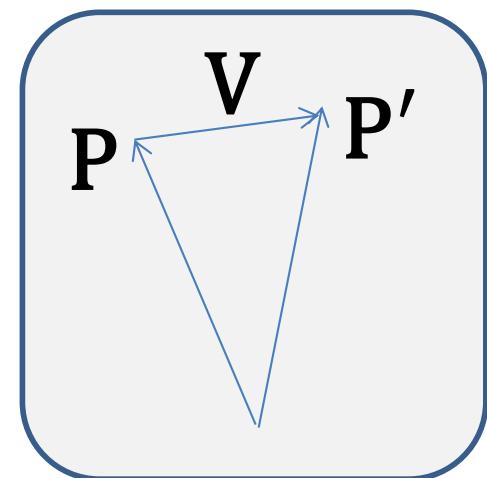
$$R \approx \begin{bmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{bmatrix}$$

Hence:

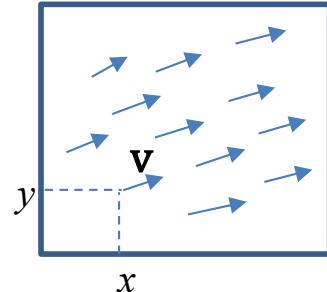
$$V_x = \theta_y Z - \theta_z Y + T_x \quad (\theta_x, \theta_y, \theta_z) \equiv \text{Angular velocity}$$

$$V_y = \theta_z X - \theta_x Z + T_y \quad (T_x, T_y, T_z) \equiv \text{Rectilinear velocity}$$

$$V_z = \theta_x Y - \theta_y X + T_z$$



2-D Motion Field Equations



For image point $\mathbf{p} = (x, y, f)$ Motion field $\mathbf{v} = (v_x, v_y)$

$$v_x = \frac{dx}{dt} = \frac{d}{dt} \left(\frac{fX}{Z} \right) = f \frac{V_X Z - X V_Z}{Z^2}$$

Quotient
rule

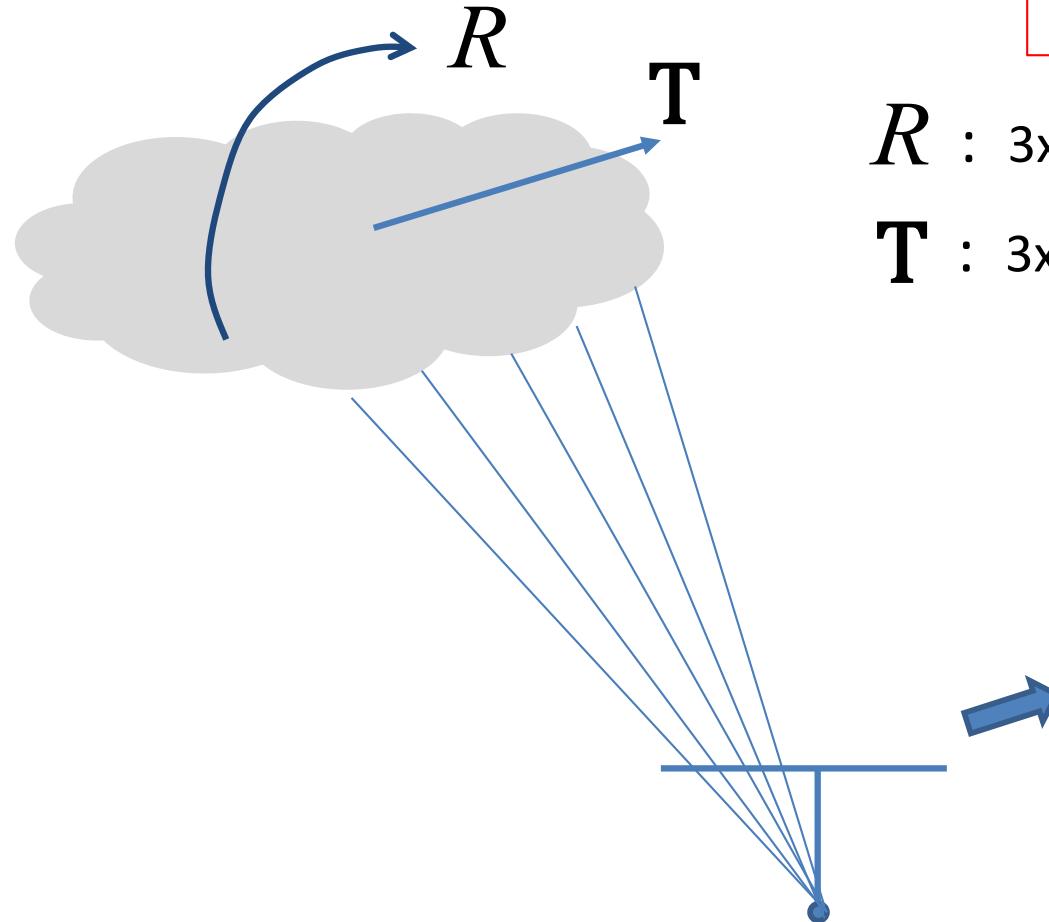
$$x = \frac{fX}{Z}$$

$$V_X = \frac{dX}{dt}$$

Substituting for V_X, V_Y, V_Z gives (previous slide)

$$v_x = (fT_X - xT_Z)/Z + f\theta_Y - \theta_Z y - (\theta_X xy - \theta_Y x^2)/f$$

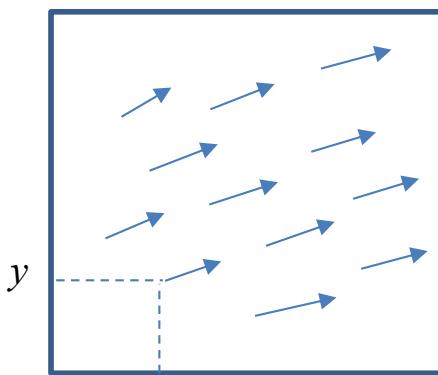
$$v_y = (fT_Y - yT_Z)/Z - f\theta_X + \theta_Z x + (\theta_Y xy - \theta_X y^2)/f$$



Small angle approximation

R : 3x3 rotation matrix $\rightarrow (\theta_X, \theta_Y, \theta_Z)$

T : 3x1 translation vector $\rightarrow (T_X, T_Y, T_Z)$



2-D motion field

$$\mathbf{v} = (v_x, v_y)$$

$$v_x = (fT_X - xT_Z)/Z + f\theta_Y - \theta_Z y - (\theta_X xy - \theta_Y x^2)/f$$

$$v_y = (fT_Y - yT_Z)/Z - f\theta_X + \theta_Z x + (\theta_Y xy - \theta_X y^2)/f$$

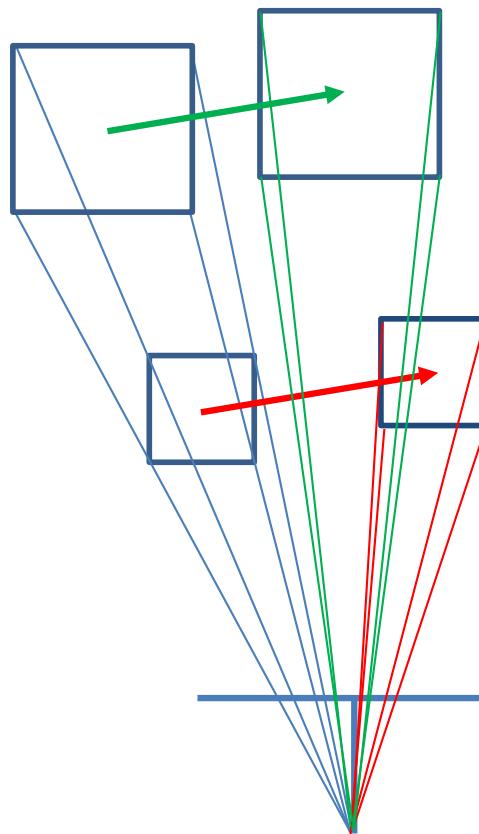
Two Components

$$v_x = (fT_X - xT_Z)/Z + f\theta_Y - \theta_Z y - (\theta_X xy - \theta_Y x^2)/f$$
$$v_y = (fT_Y - yT_Z)/Z - f\theta_X + \theta_Z x + (\theta_Y xy - \theta_X y^2)/f$$

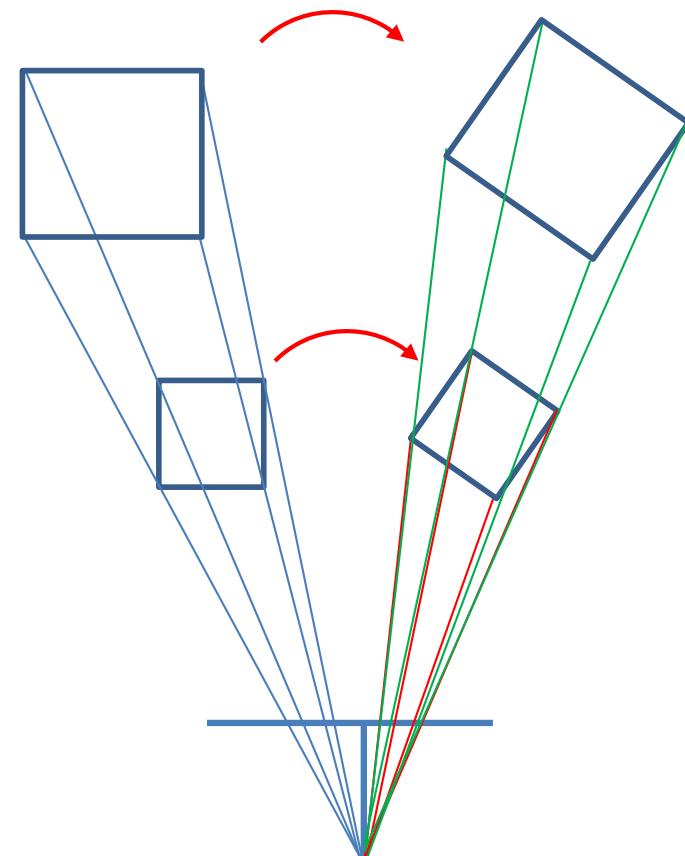
Translational – dependent on scene depth Z

Rotational – independent of scene depth Z

Translation, Rotation and Depth



Motion field \propto depth



Motion field $\not\propto$ depth

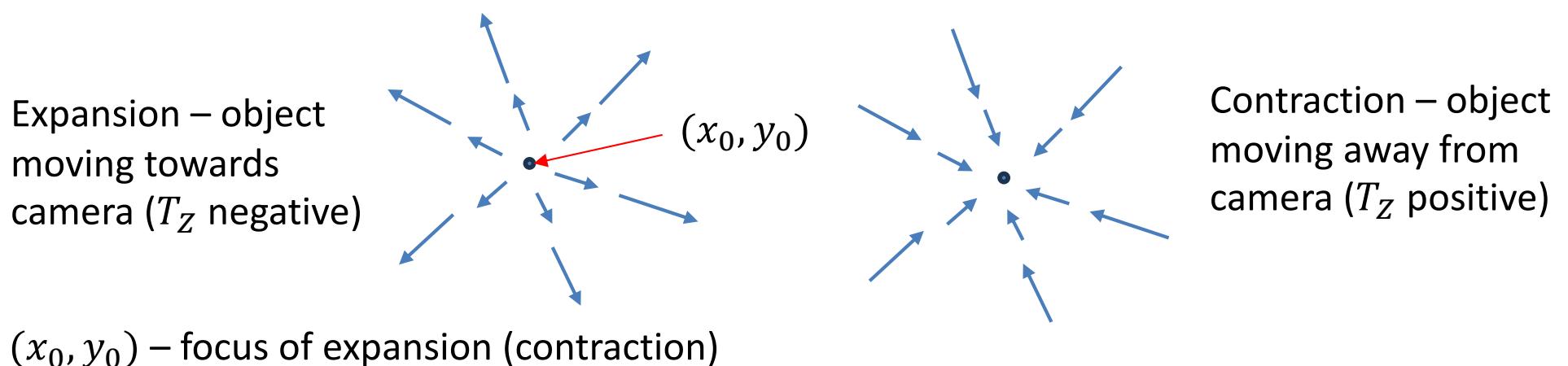
Special Case: Pure Translation

- Assume 3-D motion is only translational, $\theta = 0$, then

$$v_x = (fT_X - xT_Z)/Z \quad v_y = (fT_Y - xT_Z)/Z$$

- If $T_Z \neq 0$, $x_0 = fT_X/T_Z$ and $y_0 = fT_Y/T_Z$, then

$$v_x = -(x - x_0)T_Z/Z \quad v_y = -(y - y_0)T_Z/Z$$



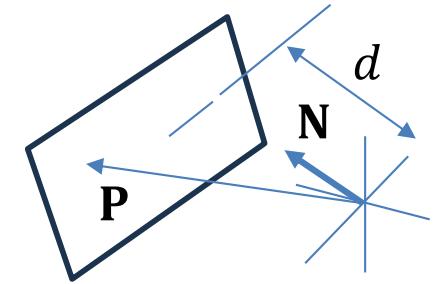
Special Case: Moving Plane

- Assume 3-D points lie in plane with unit surface normal \mathbf{N} , i.e. $\mathbf{N}^T \mathbf{P} = d$, where d is distance of plane from origin.
- Since $\mathbf{P} = Z\mathbf{p}/f$, this gives $Z(N_X x + N_Y y + N_Z f)/f = d$
- Substituting for Z in 2-D motion field:

$$v_x = \frac{1}{fd} (a_1 x^2 + a_2 xy + a_3 fx + a_4 fy + a_5 f^2)$$

$$v_y = \frac{1}{fd} (a_1 xy + a_2 y^2 + a_6 fy + a_7 fx + a_8 f^2)$$

- Motion field is a quadratic polynomial in 2-D spatial coordinates x and y



COMS30030
Image Processing and Computer Vision

Motion – Modelling

Andrew Calway
andrew@cs.bris.ac.uk

COMS30030 Motion Lec 1

1

Motion – Important Perceptual Cue



COMS30030 Motion Lec 1

2

We are going to look at

- Modelling 2-D motion fields
- Optical flow
- The optical flow equation (OFE)
- Motion estimation
 - Lucas and Kanade method

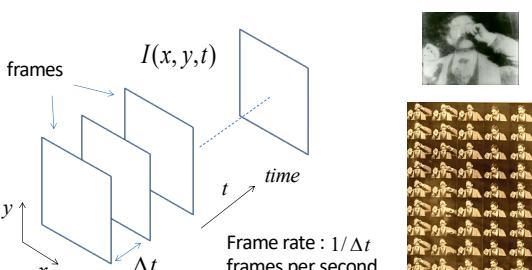


COMS30030 Motion Lec 1

3

Video Sequences

First motion picture camera
Kinetograph

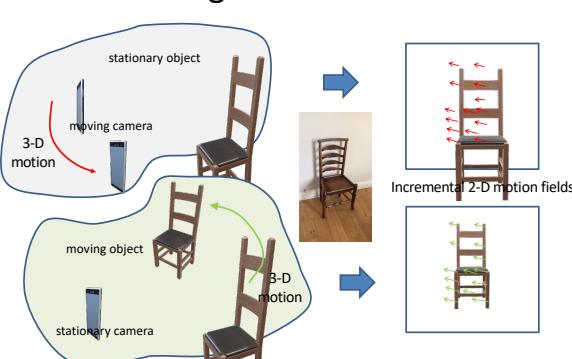


Frame rate : $1/\Delta t$
frames per second
e.g. 25 fps

COMS30030 Motion Lec 1

4

Modelling 2-D Motion Fields



stationary object
moving camera
3-D motion

moving object
stationary camera
3-D motion

Incremental 2-D motion fields

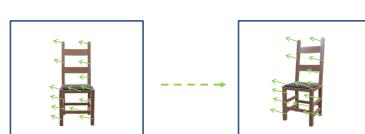
COMS30030 Motion Lec 1

5

Incremental 2-D Motion Fields

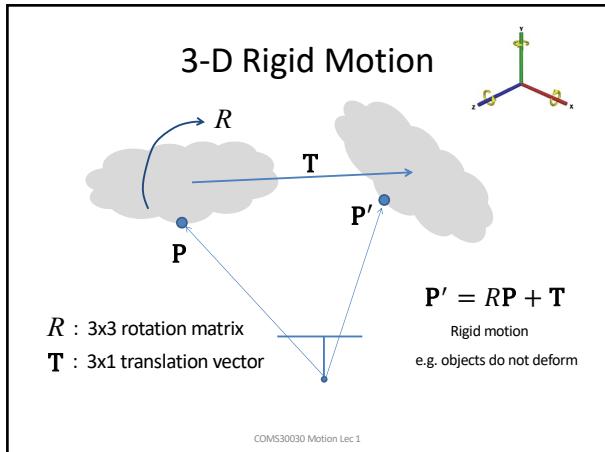
What is the relationship between the:

- Position of a 3-D point and its 2-D motion?
- 2-D motion of different 3-D points?
- 3-D motion of a 3-D point and its 2-D motion?



COMS30030 Motion Lec 1

6



7

Rotation Matrices

$R = R_X R_Y R_Z$ (for example) : Rotations about X , Y and Z axes

$$R_Y \mathbf{P} = \begin{bmatrix} \cos \theta_Y & 0 & \sin \theta_Y \\ 0 & 1 & 0 \\ -\sin \theta_Y & 0 & \cos \theta_Y \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X \cos \theta_Y + Z \sin \theta_Y \\ Y \\ Z \cos \theta_Y - X \sin \theta_Y \end{bmatrix}$$

Z θ_Y X

NB : for small θ_Y

$$R_Y \approx \begin{bmatrix} 1 & 0 & \theta_Y \\ 0 & 1 & 0 \\ -\theta_Y & 0 & 1 \end{bmatrix}$$

For small θ_Y :

$$\cos \theta_Y \approx 1$$

$$\sin \theta_Y \approx \theta_Y$$

NB : for small $\theta_X, \theta_Y, \theta_Z$

$$R \approx \begin{bmatrix} 1 & -\theta_Z & \theta_Y \\ \theta_Z & 1 & -\theta_X \\ -\theta_Y & \theta_X & 1 \end{bmatrix}$$

COMS30030 Motion Lec 1

8

3-D Motion Field

$\mathbf{V} = \lim_{\Delta t \rightarrow 0} \{\mathbf{P}' - \mathbf{P} = (R - I)\mathbf{P} + \mathbf{T}\}$

$\mathbf{P}' = R\mathbf{P} + \mathbf{T}$

For small angles:

$$R \approx \begin{bmatrix} 1 & -\theta_Z & \theta_Y \\ \theta_Z & 1 & -\theta_X \\ -\theta_Y & \theta_X & 1 \end{bmatrix}$$

Hence:

$$V_x = \theta_Y Z - \theta_Z Y + T_x \quad (\theta_X, \theta_Y, \theta_Z) \equiv \text{Angular velocity}$$

$$V_y = \theta_Z X - \theta_X Z + T_y \quad (T_x, T_y, T_z) \equiv \text{Rectilinear velocity}$$

$$V_z = \theta_X Y - \theta_Y X + T_z$$

COMS30030 Motion Lec 1

9

2-D Motion Field Equations

For image point $\mathbf{p} = (x, y, f)$ Motion field $\mathbf{v} = (v_x, v_y)$

$$v_x = \frac{dx}{dt} = \frac{d}{dt} \left(\frac{fX}{Z} \right) = f \frac{V_x Z - X V_z}{Z^2}$$

Quotient rule

$$x = \frac{fX}{Z}$$

$$V_x = \frac{dX}{dt}$$

Substituting for V_x, V_y, V_z gives (previous slide)

$$v_x = (fT_x - xT_z)/Z + f\theta_Y - \theta_Z y - (\theta_X xy - \theta_Y x^2)/f$$

$$v_y = (fT_y - yT_z)/Z - f\theta_X + \theta_Z x + (\theta_Y xy - \theta_X y^2)/f$$

COMS30030 Motion Lec 1

10

Small angle approximation

R : 3x3 rotation matrix $\Rightarrow (\theta_X, \theta_Y, \theta_Z)$
 \mathbf{T} : 3x1 translation vector $\Rightarrow (T_x, T_y, T_z)$

2-D motion field $\mathbf{v} = (v_x, v_y)$

$v_x = (fT_x - xT_z)/Z + f\theta_Y - \theta_Z y - (\theta_X xy - \theta_Y x^2)/f$
 $v_y = (fT_y - yT_z)/Z - f\theta_X + \theta_Z x + (\theta_Y xy - \theta_X y^2)/f$

COMS30030 Motion Lec 1

11

Two Components

$v_x = (fT_x - xT_z)/Z + f\theta_Y - \theta_Z y - (\theta_X xy - \theta_Y x^2)/f$

$v_y = (fT_y - yT_z)/Z - f\theta_X + \theta_Z x + (\theta_Y xy - \theta_X y^2)/f$

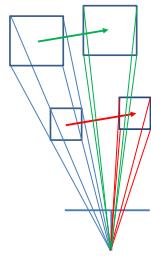
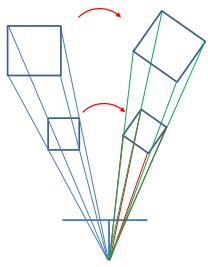
Translational – dependent on scene depth Z

Rotational – independent of scene depth Z

COMS30030 Motion Lec 1

12

Translation, Rotation and Depth

Motion field \propto depthMotion field $\not\propto$ depth

COMS30030 Motion Lec 1

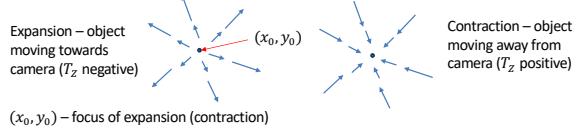
Special Case: Pure Translation

- Assume 3-D motion is only translational, $\Theta = 0$, then

$$v_x = (fT_X - xT_Z)/Z \quad v_y = (fT_Y - xT_Z)/Z$$

- If $T_Z \neq 0$, $x_0 = fT_X/T_Z$ and $y_0 = fT_Y/T_Z$, then

$$v_x = -(x - x_0)T_Z/Z \quad v_y = -(y - y_0)T_Z/Z$$



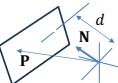
COMS30030 Motion Lec 1

13

14

Special Case: Moving Plane

- Assume 3-D points lie in plane with unit surface normal \mathbf{N} , i.e. $\mathbf{N}^T \mathbf{P} = d$, where d is distance of plane from origin.



- Since $\mathbf{P} = Z\mathbf{p}/f$, this gives $Z(N_x x + N_y y + N_z f)/f = d$

- Substituting for Z in 2-D motion field:

$$v_x = \frac{1}{fd}(a_1x^2 + a_2xy + a_3fx + a_4fy + a_5f^2)$$

$$v_y = \frac{1}{fd}(a_1xy + a_2y^2 + a_6fy + a_7fx + a_8f^2)$$

- Motion field is a quadratic polynomial in 2-D spatial coordinates x and y

COMS30030 Motion Lec 1

15

COMS30030
Image Processing and Computer Vision

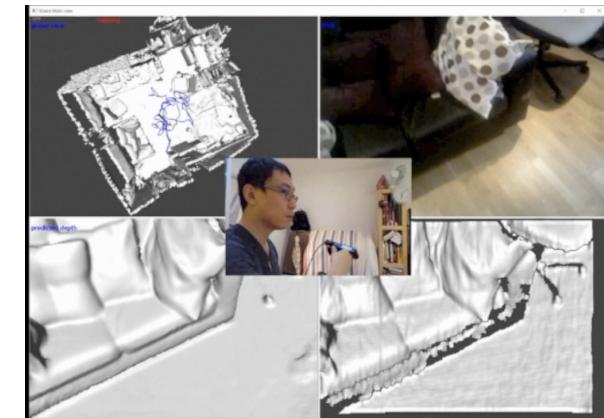
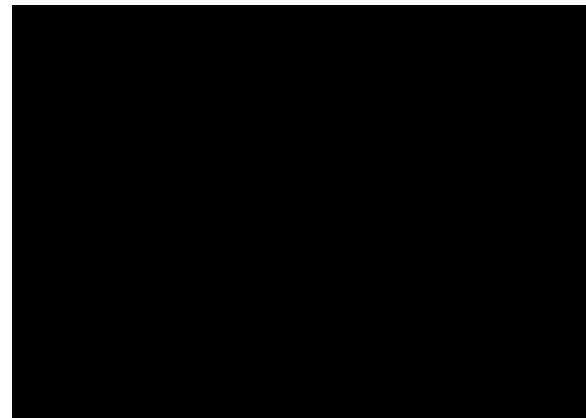
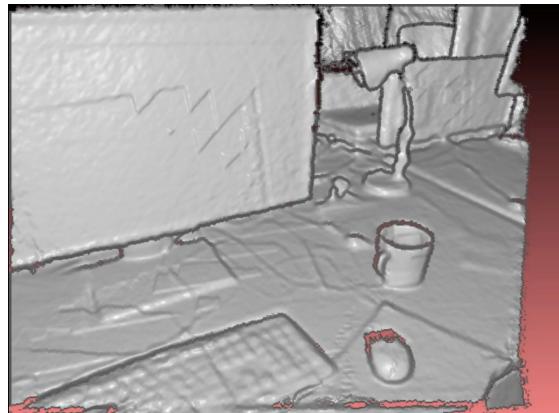
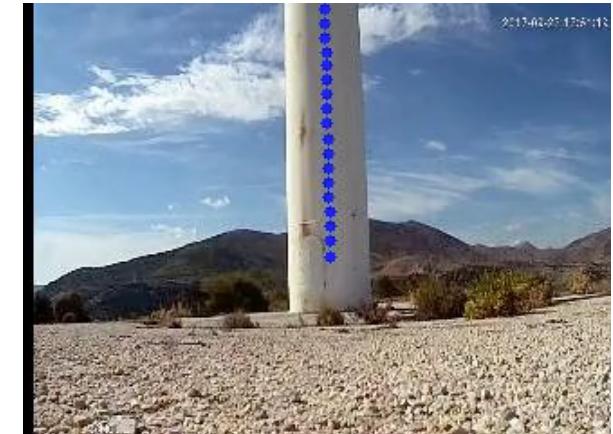
Part II: Stereo and Motion

Andrew Calway

andrew@cs.bris.ac.uk



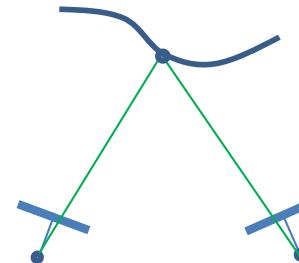
Andrew Calway
Professor of Computer Vision
www.cs.bris.ac.uk/~andrew
andrew@cs.bris.ac.uk
Computer Vision and Robotics



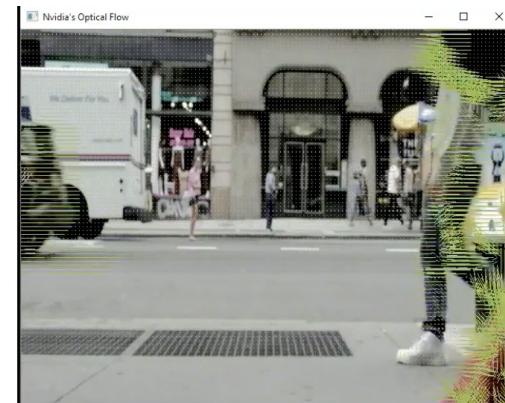
Vision-Based Simultaneous Localisation and Mapping (SLAM)

Stereo and Motion

- **Stereo**
 - Determining scene depth information from 2 (or more) images captured from different viewpoints
 - Geometry, correspondence matching and 3-D reconstruction

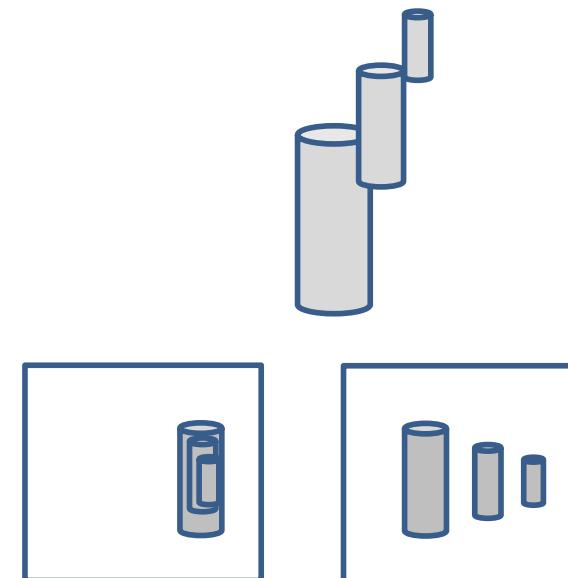
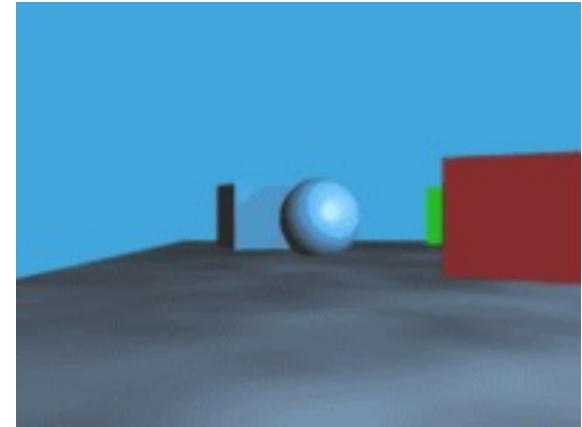


- **Motion**
 - Determining 2-D motion in video frames
 - Modelling, optical flow and motion estimation



Stereo Vision

- Stereo vision - 3-D from two images taken from different viewpoints.
- Objects appear in different positions in each viewpoint – **parallax**.
- Position of object in each image depends on its depth.
 - position difference (**disparity**) inversely proportional to depth
- If we know disparity & viewpoints
 - 3-D scene structure



Colour animated gif by **Nathaniel Domek**

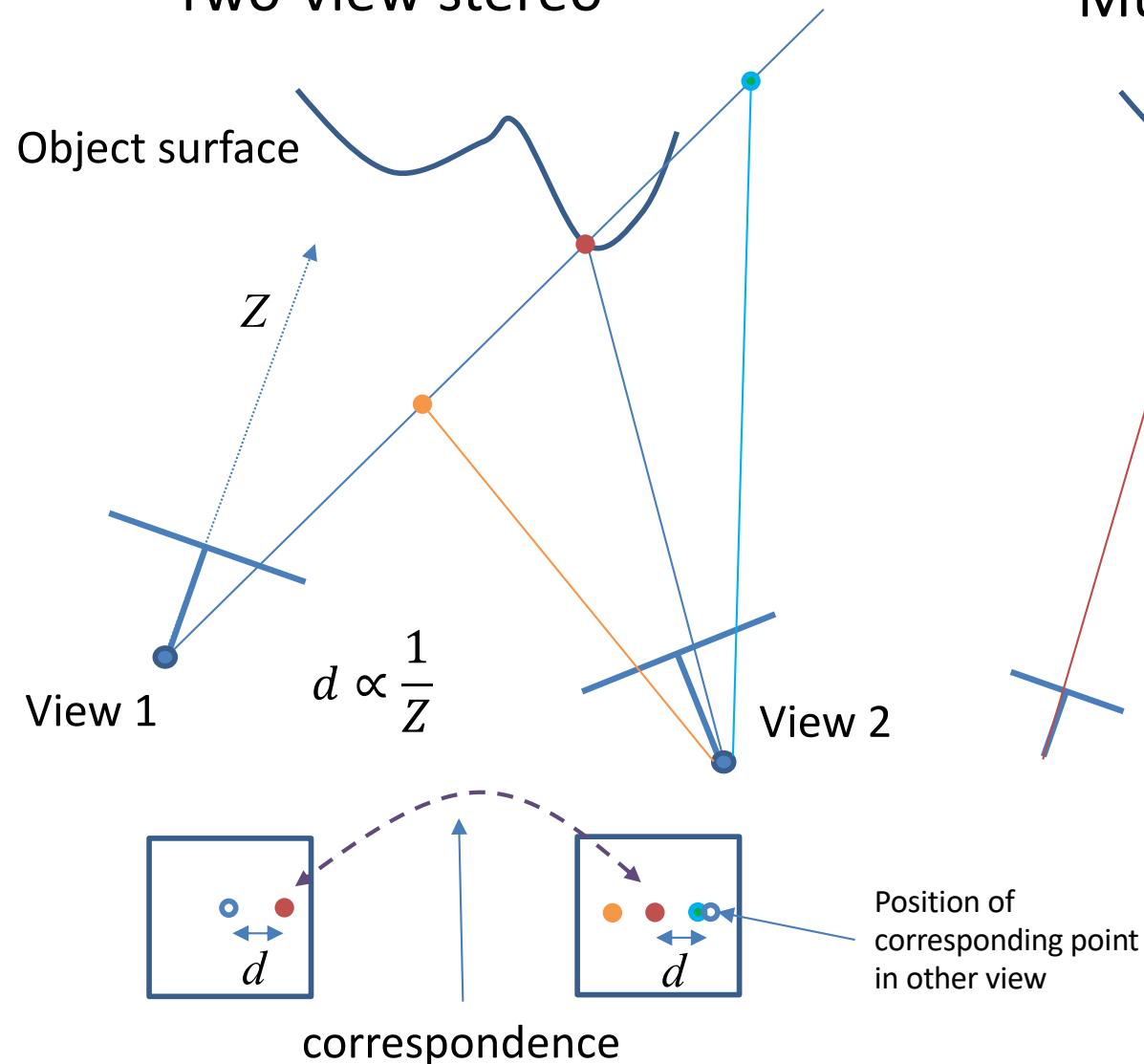
<https://en.wikipedia.org/wiki/Parallax#/media/File:Parallax.gif>

Two-View Stereo Examples

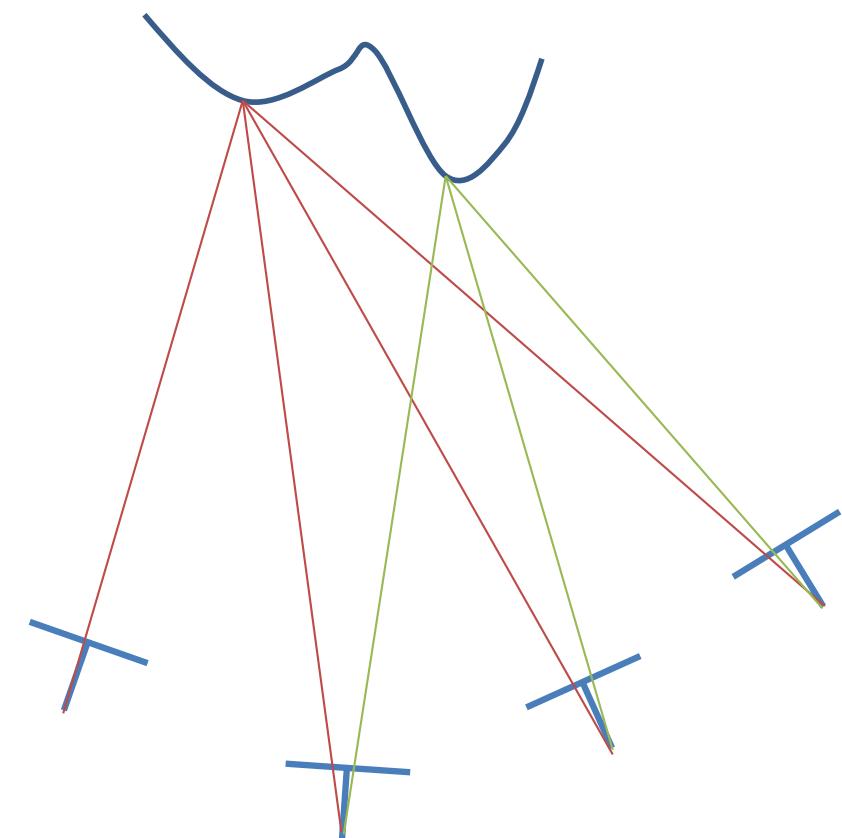


Stereo Computer Vision

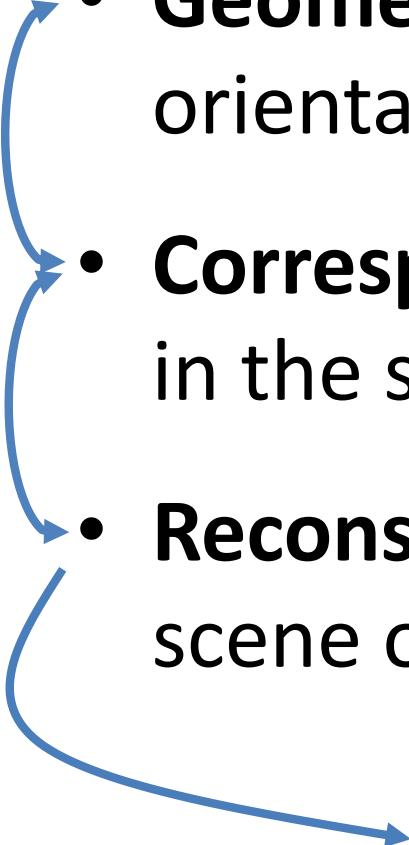
Two-view stereo



Multi-view stereo



Three Problems of Stereo

- **Geometry** – determine relative position and orientation of the cameras
 - **Correspondence** – determine matching points in the stereo views
 - **Reconstruction** – determine 3D location in scene of matched points via triangulation
- all interrelated**
- 

Stereo Vision – SOTA Examples

Two view



depth map

Group-wise Correlation Stereo Network,
Guo *et al*, CVPR 2019

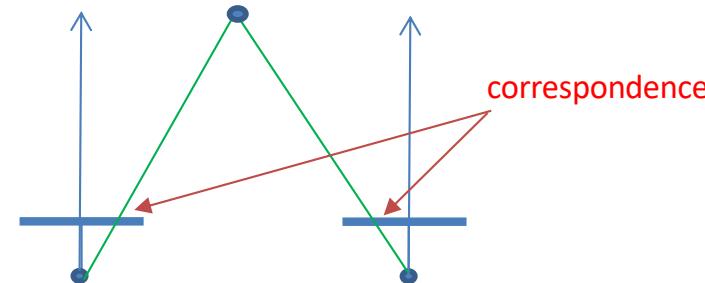
Multi-view



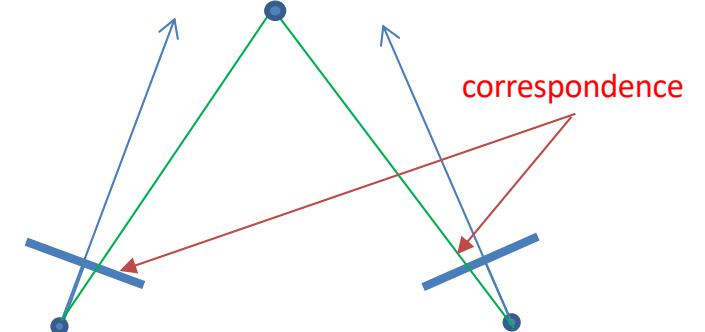
Depth from Gradients in Dense Light Fields for
Object Reconstruction, Yucer *et al*, 3DV 2016

Stereo Geometry

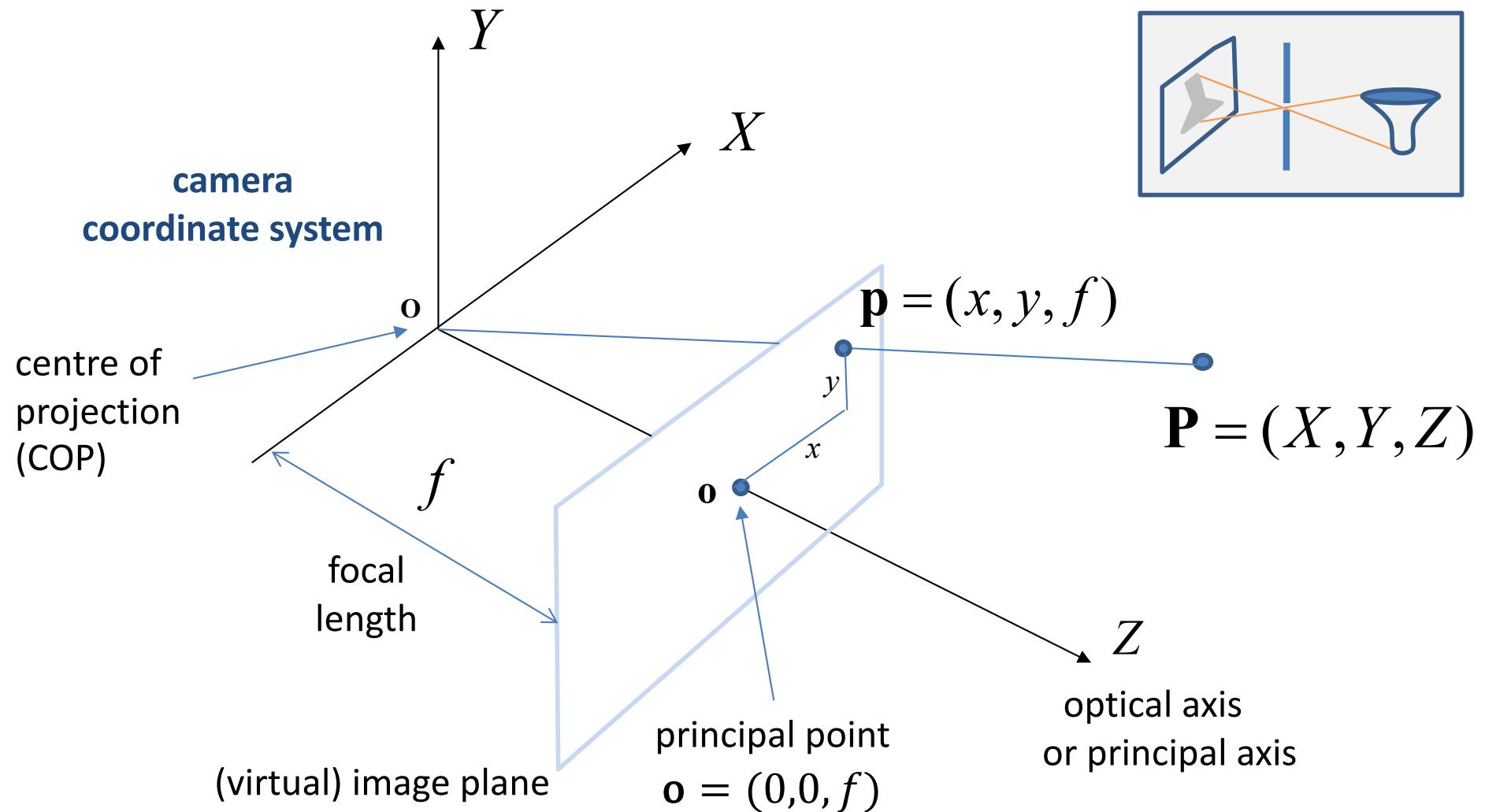
- Need to understand geometric relationship between cameras to allow 3-D reconstruction from correspondences
- Simple two-view stereo - coplanar image planes – geometry defined by similar triangles



- General stereo – geometry depends on **position** and **orientation** of cameras
 - **epipolar geometry**
- But we also need camera model



Pin Hole Camera Model



Perspective Projection Equations

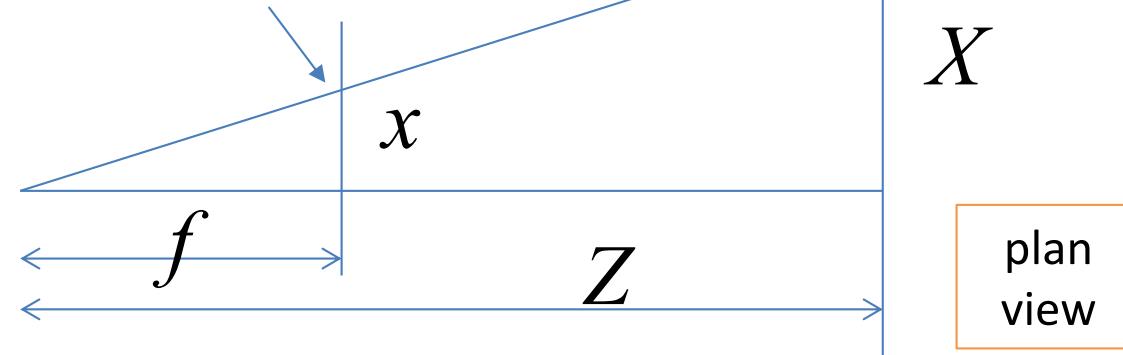
- 3D point: $\mathbf{P} = (X, Y, Z)$ (on surface of object)
- Projects to 2D point: $\mathbf{p} = (x, y, f)$ (in image)
- Using similar triangles (pinhole model):

$$x = \frac{fX}{Z} \quad y = \frac{fY}{Z}$$

$$\mathbf{p} = f \frac{\mathbf{P}}{Z}$$

$$\mathbf{p} = (x, y, f)$$

$$\mathbf{P} = (X, Y, Z)$$



Simple Two-View Stereo

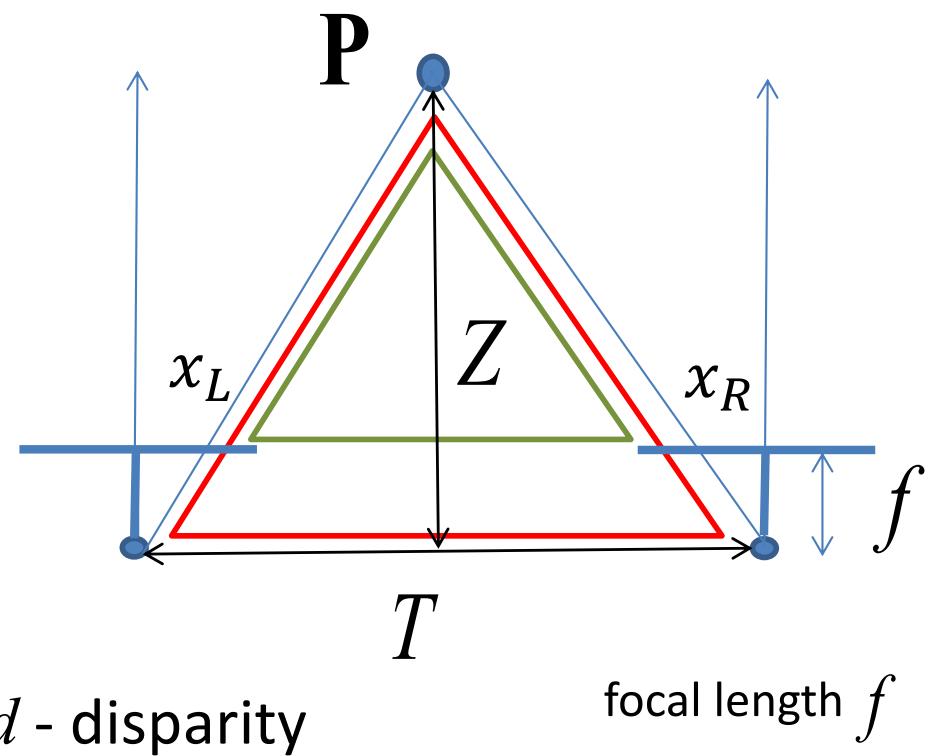
- Coplanar image planes, COPs in X-Z plane
- T – **baseline**, distance between COPs
- Similar triangles:

$$\frac{T}{Z} = \frac{T - x_L + x_R}{Z - f}$$

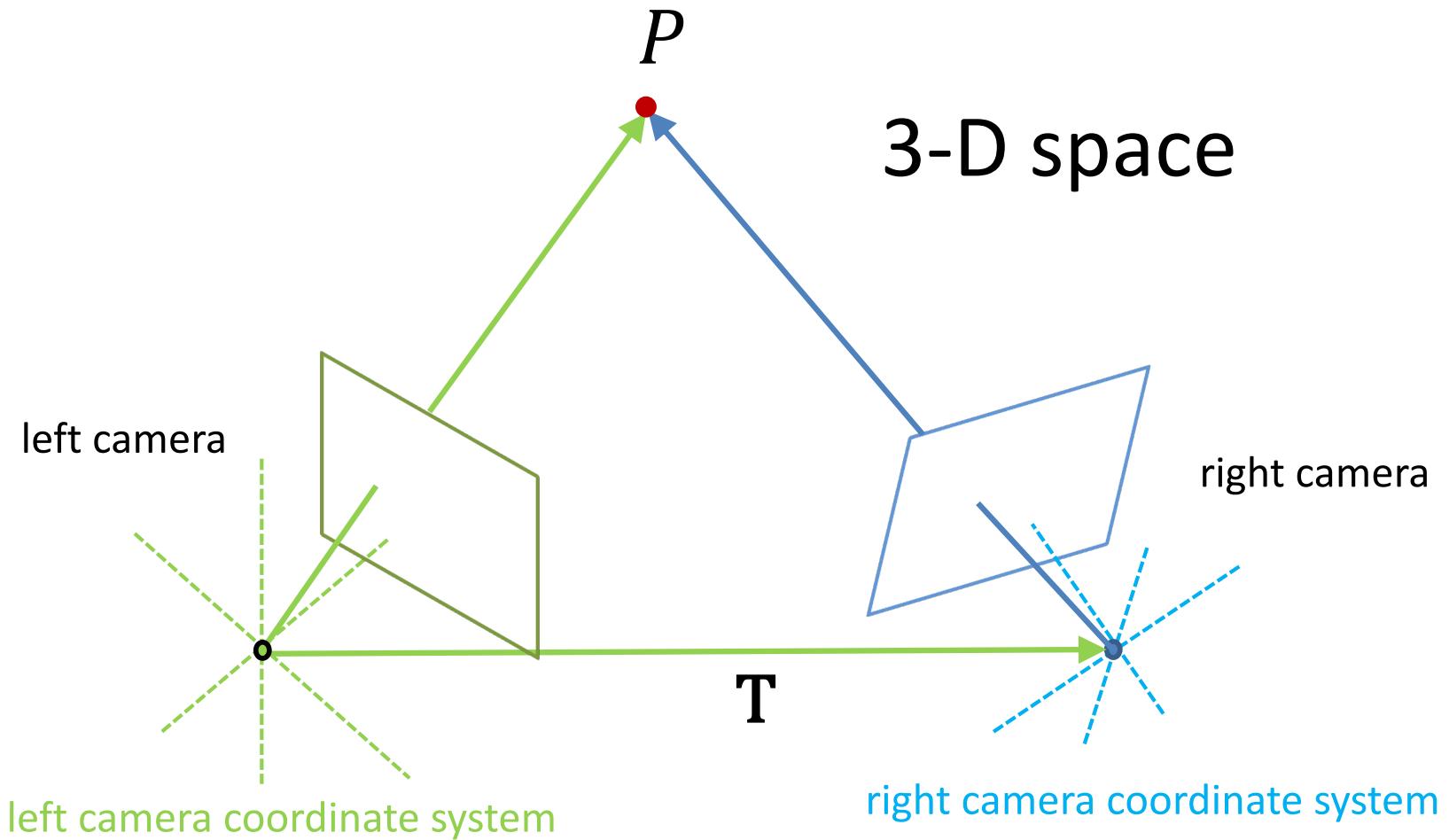
- Rearrange for depth:

$$Z = \frac{fT}{x_L - x_R} = \frac{fT}{d}$$

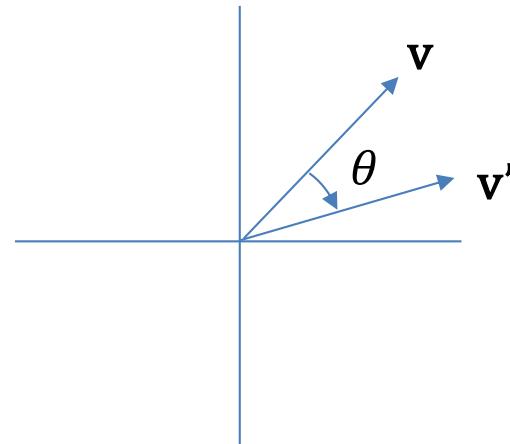
Point P has depth Z



General Two-View Stereo



Rotation Matrices



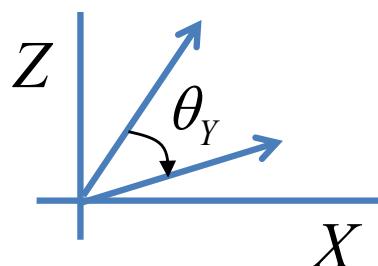
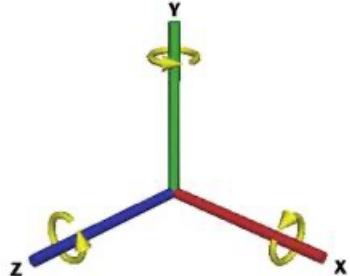
2-D clockwise rotation:

$$\mathbf{v}' = \begin{bmatrix} v'_x \\ v'_y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = R(\theta)\mathbf{v}$$

2x2 matrix

3-D rotation composed of rotations around X , Y and Z axes:

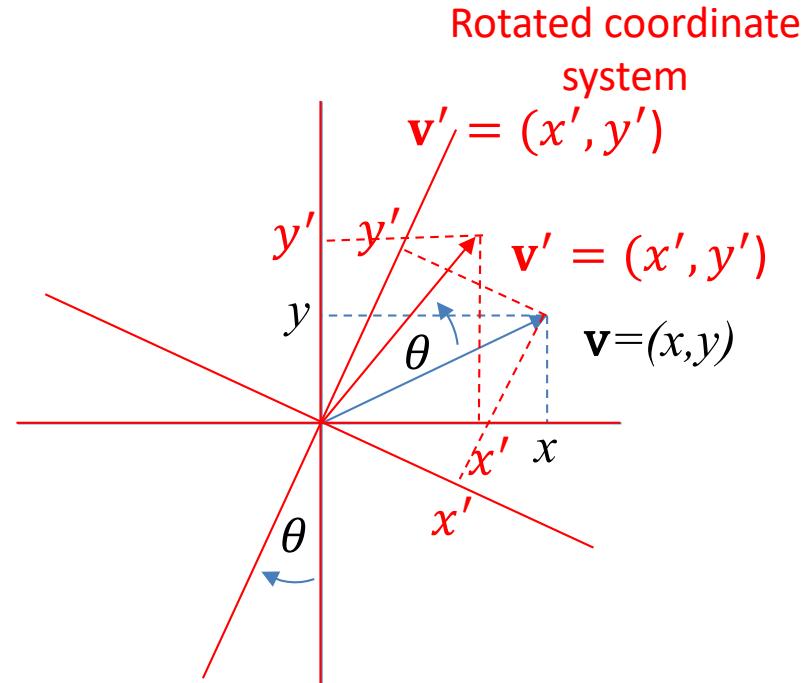
3x3 matrix: $R = R_X R_Y R_Z$



$$R_Y \mathbf{P} = \begin{bmatrix} \cos \theta_Y & 0 & \sin \theta_Y \\ 0 & 1 & 0 \\ -\sin \theta_Y & 0 & \cos \theta_Y \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

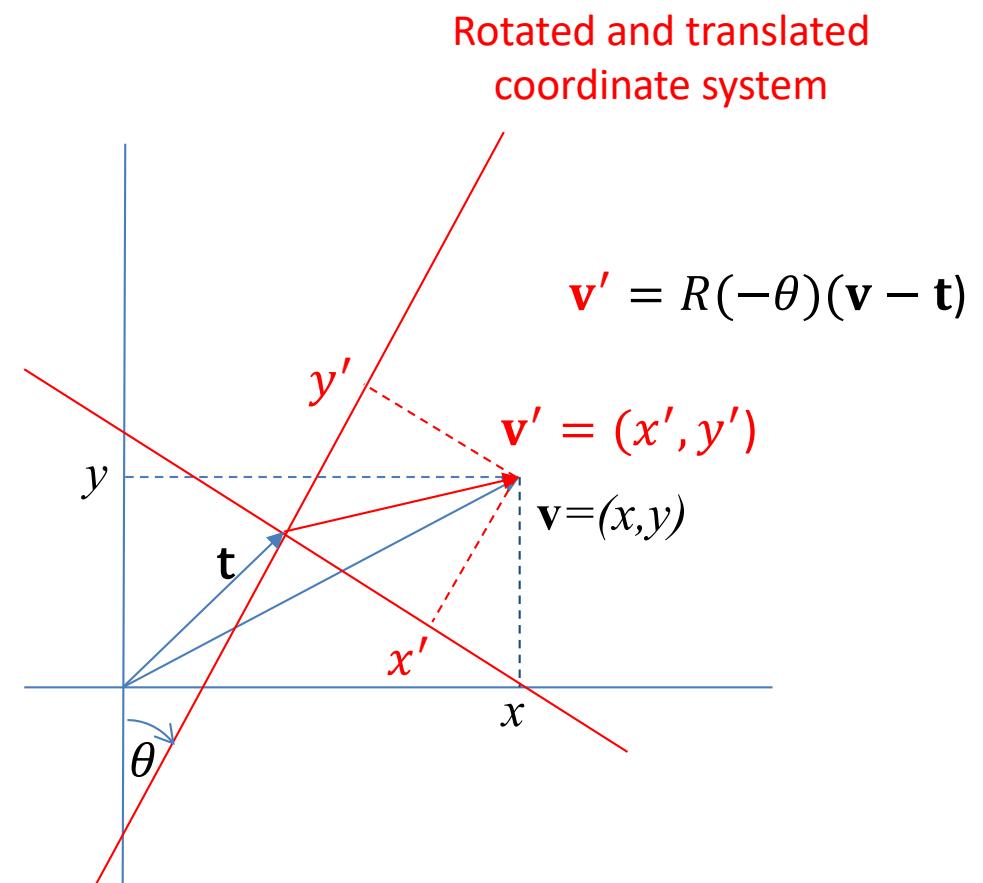
Clockwise rotation about Y axis

2-D Coordinate Transformations



$$\mathbf{v}' = R(-\theta)\mathbf{v}$$

Vector representation in rotated coordinate system



$$\mathbf{v}' = R(-\theta)(\mathbf{v} - \mathbf{t})$$

COMS30030
Image Processing and Computer Vision

Part II: Stereo and Motion

Andrew Calway
andrew@cs.bris.ac.uk

COMS30030 Stereo Lec 1

1

Andrew Calway
Professor of Computer Vision
www.cs.bris.ac.uk/~andrew
andrew@cs.bris.ac.uk
Computer Vision and Robotics

Vision-Based Simultaneous Localisation and Mapping (SLAM)

COMS30030 Stereo Lec 1

2

Stereo and Motion

- Stereo**
 - Determining scene depth information from 2 (or more) images captured from different viewpoints
 - Geometry, correspondence matching and 3-D reconstruction
- Motion**
 - Determining 2-D motion in video frames
 - Modelling, optical flow and motion estimation

COMS30030 Stereo Lec 1

3

Stereo Vision

- Stereo vision - 3-D from two images taken from different viewpoints.
- Objects appear in different positions in each viewpoint – **parallax**.
- Position of object in each image depends on its depth.
 - position difference (**disparity**) inversely proportional to depth
- If we know disparity & viewpoints – 3-D scene structure

Colour animated gif by Nathaniel Domek
<https://en.wikipedia.org/wiki/Parallax#/media/File:Parallax.gif>

COMS30030 Stereo Lec 1

4

Two-View Stereo Examples

COMS30030 Stereo Lec 1

5

Stereo Computer Vision

Two-view stereo

Multi-view stereo

$d \propto \frac{1}{Z}$

correspondence

Position of corresponding point in other view

Similar to SLAM and Structure-from-Motion (SfM)

COMS30030 Stereo Lec 1

6

Three Problems of Stereo

- **Geometry** – determine relative position and orientation of the cameras
 - **Correspondence** – determine matching points in the stereo views
 - **Reconstruction** – determine 3D location in scene of matched points via triangulation
- all interrelated

COMS30030 Stereo Lec 1

7

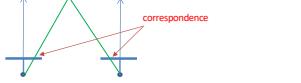
Stereo Vision – SOTA Examples



COMS30030 Stereo Lec 1

8

Stereo Geometry

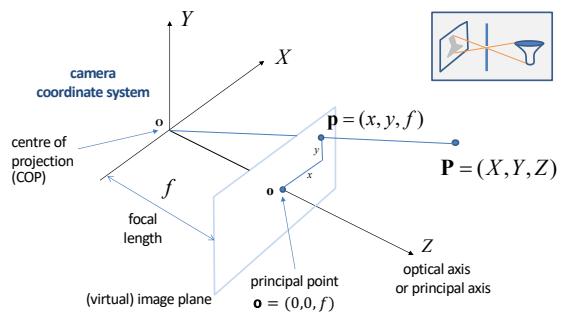
- Need to understand geometric relationship between cameras to allow 3-D reconstruction from correspondences
- Simple two-view stereo - coplanar image planes – geometry defined by similar triangles
 
- General stereo – geometry depends on **position** and **orientation** of cameras
 - **epipolar geometry**

- But we also need camera model

COMS30030 Stereo Lec 1

9

Pin Hole Camera Model



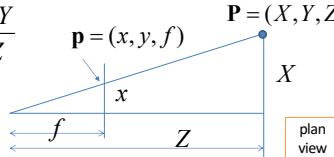
COMS30030 Stereo Lec 1

10

Perspective Projection Equations

- 3D point: $\mathbf{P} = (X, Y, Z)$ (on surface of object)
- Projects to 2D point: $\mathbf{p} = (x, y, f)$ (in image)
- Using similar triangles (pinhole model):

$$x = \frac{fX}{Z} \quad y = \frac{fY}{Z} \quad \mathbf{p} = (x, y, f) \quad \mathbf{P} = (X, Y, Z)$$

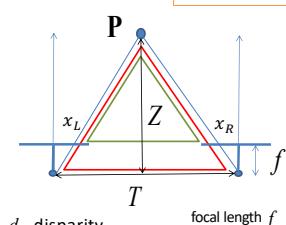
$$\mathbf{p} = f \frac{\mathbf{P}}{Z}$$


COMS30030 Stereo Lec 1

11

Simple Two-View Stereo

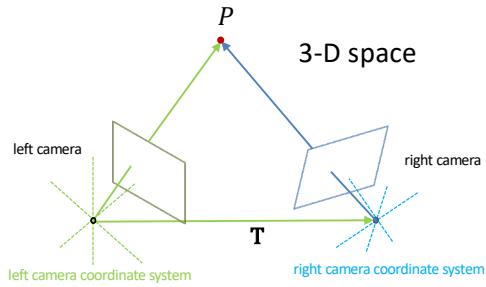
- Coplanar image planes, COPs in X-Z plane
 - T – **baseline**, distance between COPs
 - Similar triangles:
- $$\frac{T}{Z} = \frac{x_L + x_R}{Z - f}$$
- Rearrange for depth:

$$Z = \frac{fT}{x_L - x_R} = \frac{fT}{d}$$
- Point P has depth Z
- 

COMS30030 Stereo Lec 1

12

General Two-View Stereo



COMS30030 Stereo Lec 1

13

Rotation Matrices

2-D clockwise rotation:
 $v' = \begin{bmatrix} v'_x \\ v'_y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = R(\theta)v$

3-D rotation composed of rotations around X , Y and Z axes:

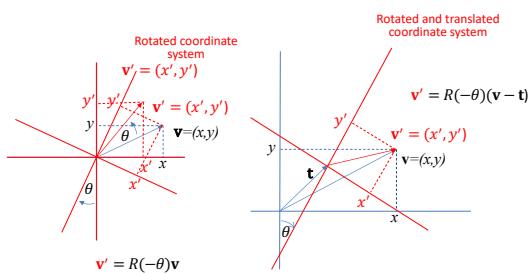
3x3 matrix: $R = R_X R_Y R_Z$
 $R_Y \mathbf{P} = \begin{bmatrix} \cos \theta_Y & 0 & \sin \theta_Y \\ 0 & 1 & 0 \\ -\sin \theta_Y & 0 & \cos \theta_Y \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$

Clockwise rotation about Y axis

COMS30030 Stereo Lec 1

14

2-D Coordinate Transformations



Vector representation in rotated coordinate system

COMS30030 Stereo Lec 1

15

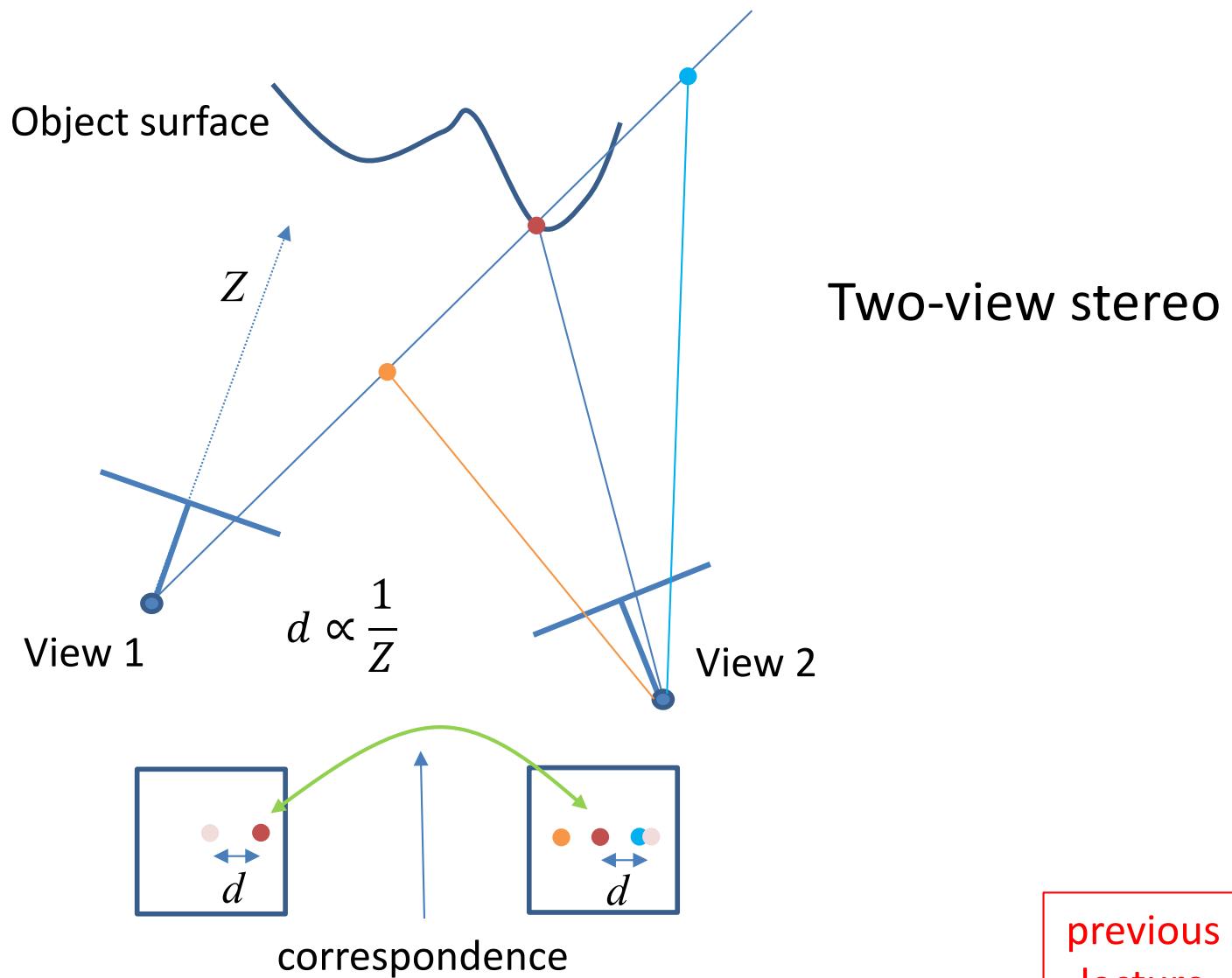
COMS30030
Image Processing and Computer Vision

Stereo 2 – Epipolar Geometry

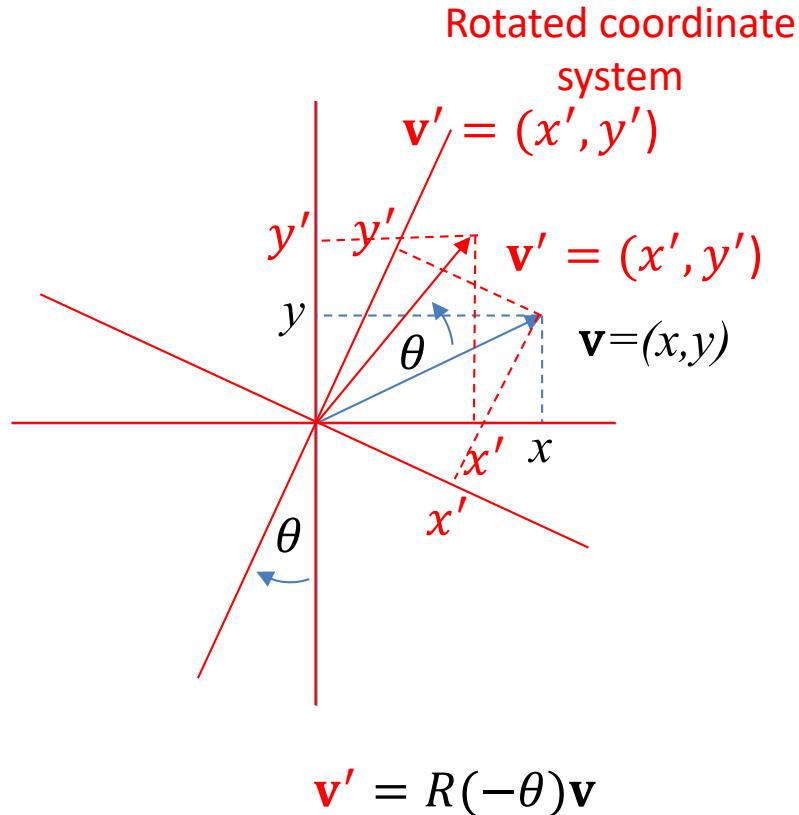
Andrew Calway

andrew@cs.bris.ac.uk

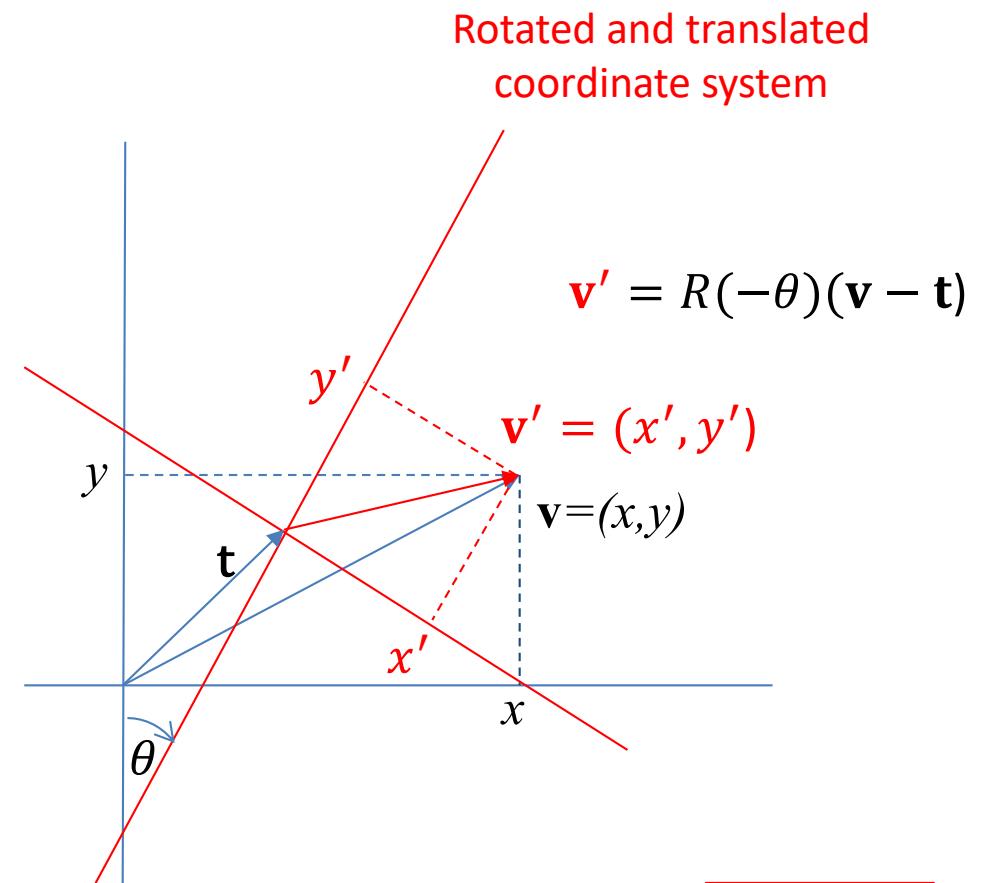
Stereo Computer Vision



2-D Coordinate Transformations



Vector representation in rotated coordinate system



previous
lecture

3-D Coordinate Transformations

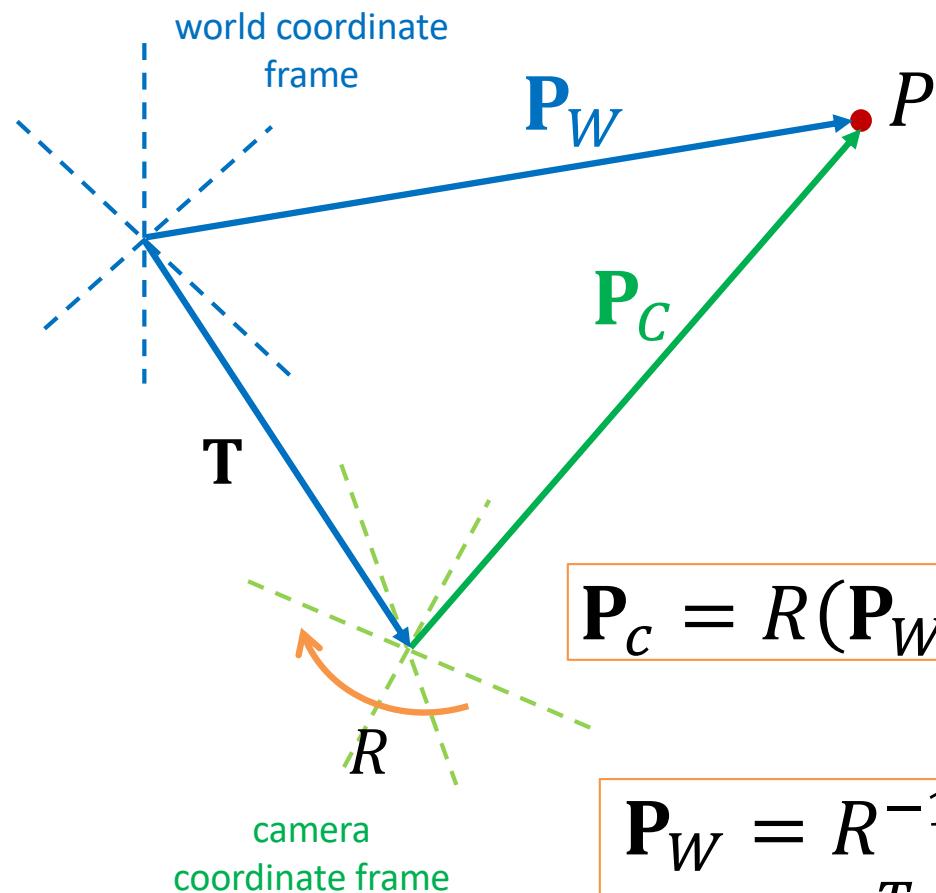
\mathbf{P}_W Vector defining P in world coordinates

\mathbf{P}_C Vector defining P in camera coordinates

\mathbf{T} : 3-D camera position vector

R : 3-D camera rotation matrix

R defines rotation to be applied to camera coordinate system to align it with world coordinate system



$$\mathbf{P}_c = R(\mathbf{P}_W - \mathbf{T})$$

$$\begin{aligned}\mathbf{P}_W &= R^{-1}\mathbf{P}_c + \mathbf{T} \\ &= R^T\mathbf{P}_c + \mathbf{T}\end{aligned}$$

Rotation matrices: $R^T = R^{-1}$

Homogeneous Coordinates

- Homogeneous coordinates allow coordinate transformations to be defined by 4x4 matrices:

$$\mathbf{P}'_W = \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_W \end{bmatrix} = \begin{bmatrix} R^T & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_C \end{bmatrix} = H_{CW} \mathbf{P}'_C \quad \Rightarrow \quad \mathbf{P}_W = R^T \mathbf{P}_C + \mathbf{T}$$

$$\mathbf{P}'_C = H_{CW}^{-1} \mathbf{P}'_W = H_{WC} \mathbf{P}'_W \quad H_{CW} = \begin{bmatrix} R_{00} & R_{10} & R_{20} & T_x \\ R_{01} & R_{11} & R_{21} & T_y \\ R_{02} & R_{12} & R_{22} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

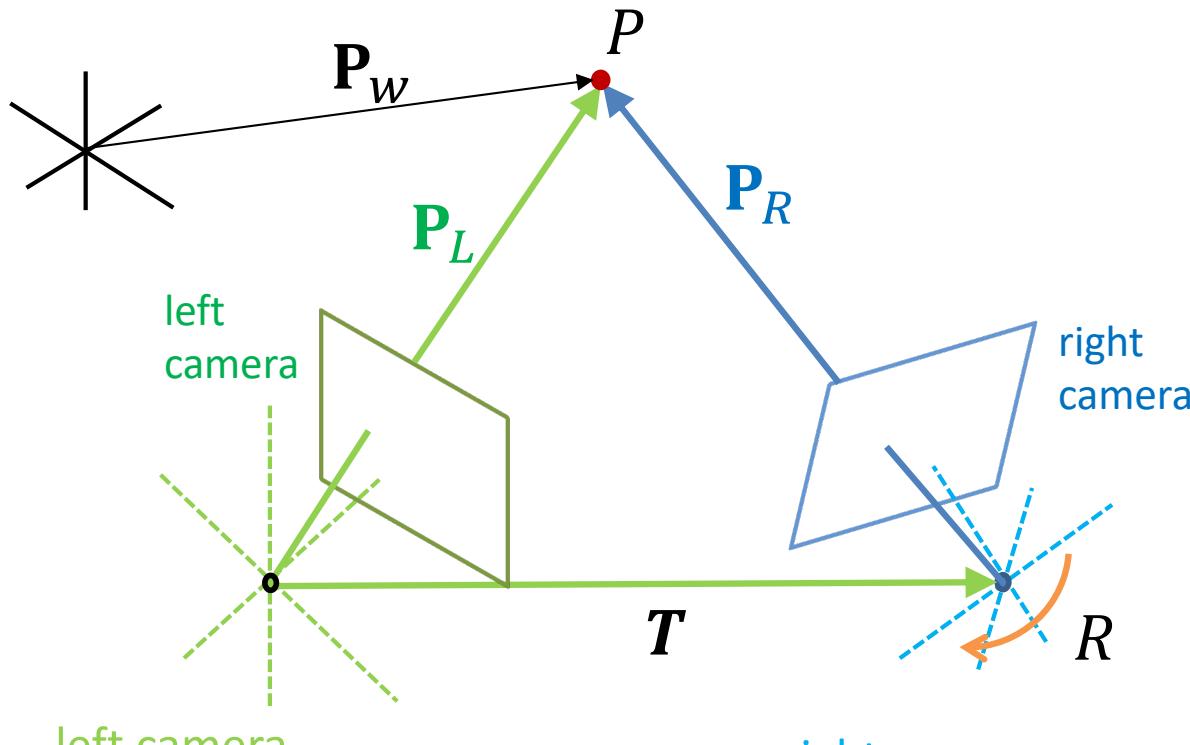
$$H_{WC} = \begin{bmatrix} R & -R\mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \quad \Rightarrow \quad H_{WC} H_{CW} = I$$

I - Identity matrix

$$\Rightarrow \quad \mathbf{P}_C = R (\mathbf{P}_W - \mathbf{T})$$

H_{CW} : camera to world
coordinate transformation matrix

Stereo Coordinate Systems



left camera
coordinate system

right camera
coordinate system

R defines rotation to be applied to right camera coordinate system to align it with left coordinate system

$$\mathbf{P}'_L = H_{WL} \mathbf{P}'_W$$

$$\mathbf{P}'_R = H_{WR} \mathbf{P}'_W$$

$$\mathbf{P}'_L = H_{WL} H_{WR}^{-1} \mathbf{P}'_R$$

$$\mathbf{P}'_L = H_{RL} \mathbf{P}'_R$$

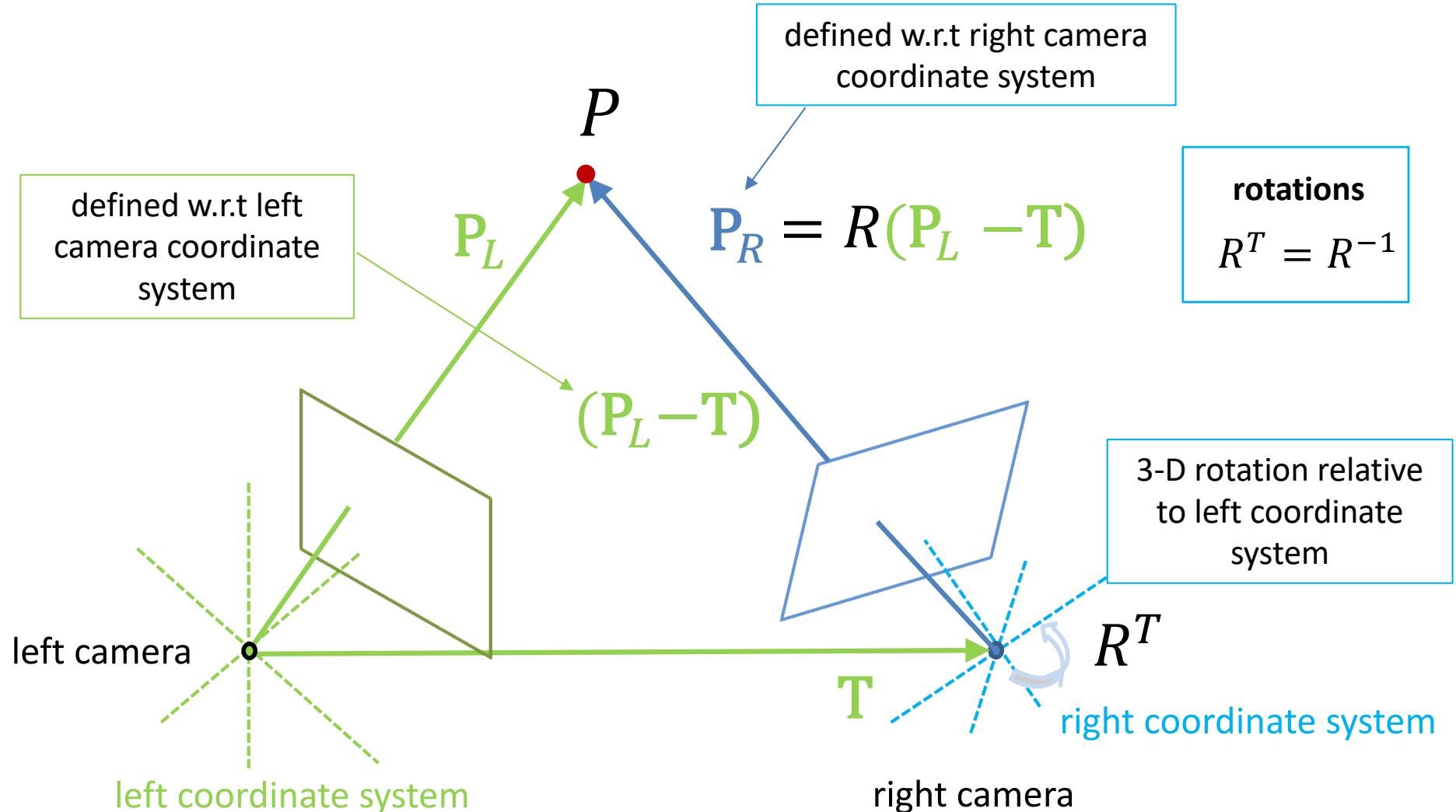
$$H_{RL} = \begin{bmatrix} R^T & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\mathbf{P}_L = R^T \mathbf{P}_R + \mathbf{T}$$

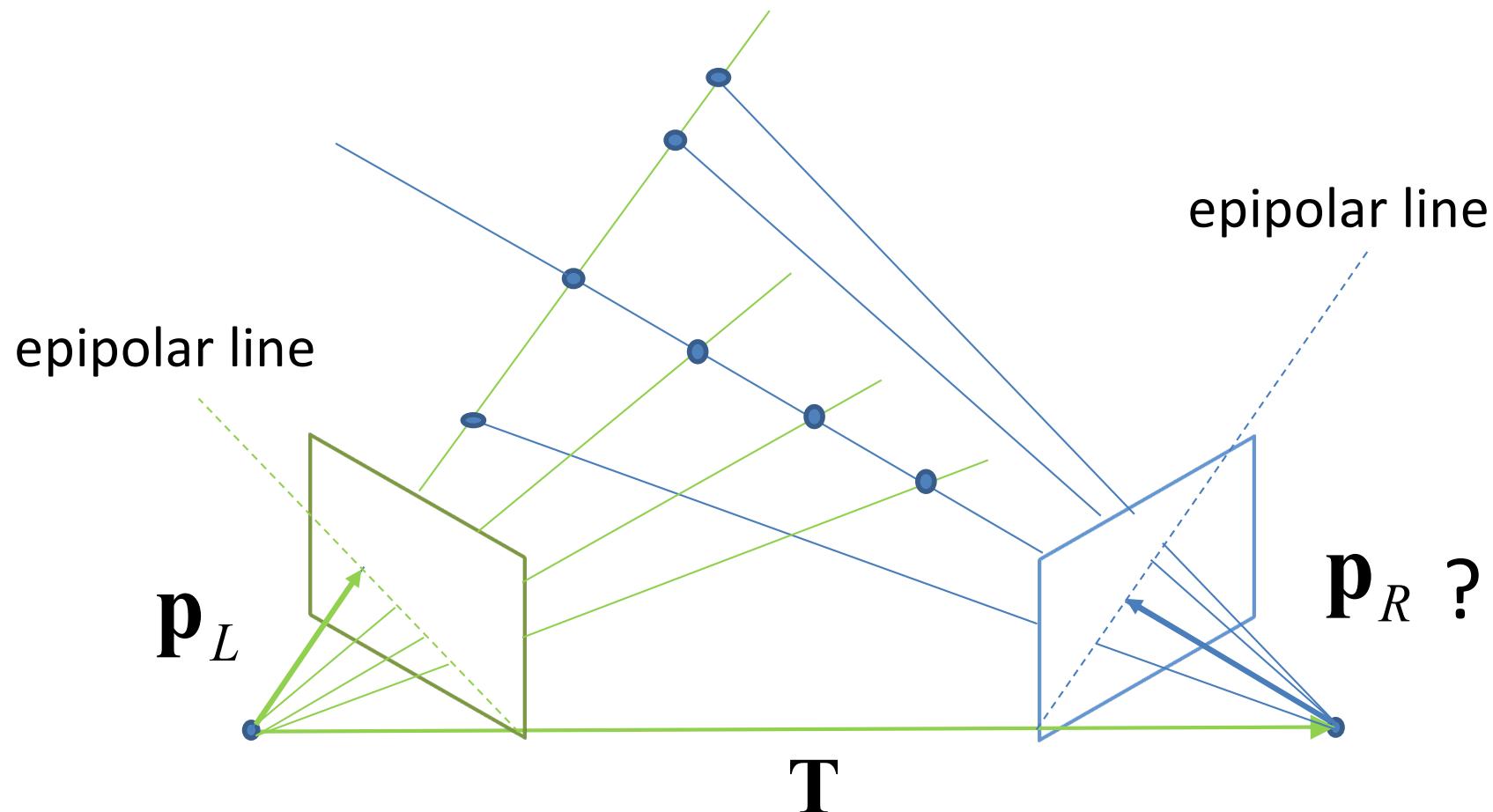
$$\mathbf{P}_R = R(\mathbf{P}_L - \mathbf{T})$$

General Two-View Stereo

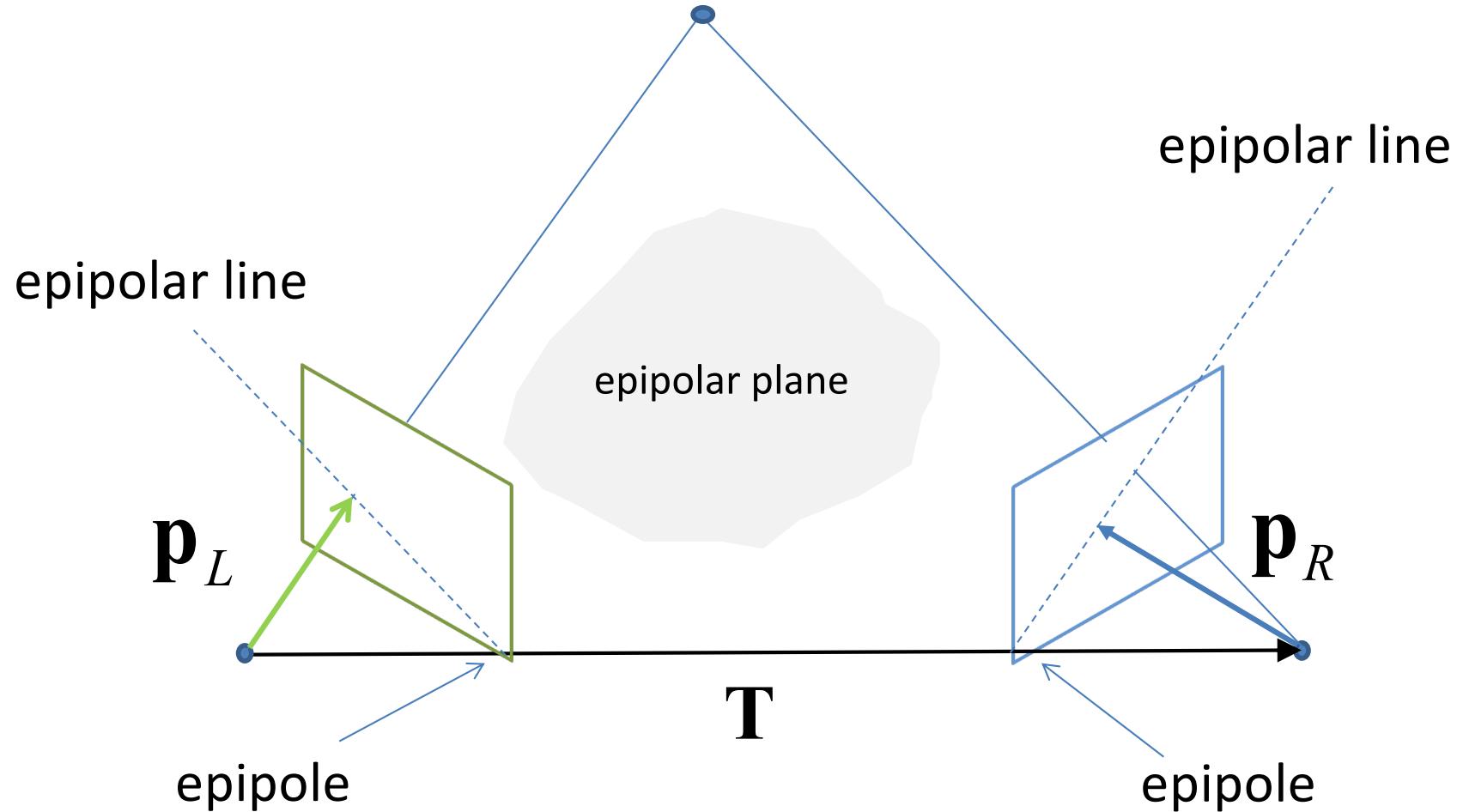
previous
lecture



Epipolar Lines



Epipolar Planes



Epipolar Geometry

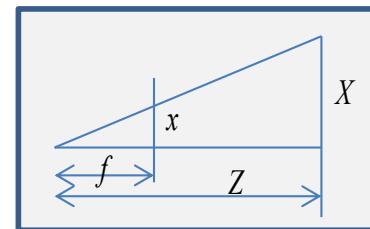
- Epipolar geometry defines relationship between two stereo views
- For known viewpoints:
 - it constrains matches to lie along epipolar lines
- For unknown viewpoints:
 - given matching points
 - it enables estimation of viewpoints

Epipolar Geometry - Maths

Rigid transformation between cameras:

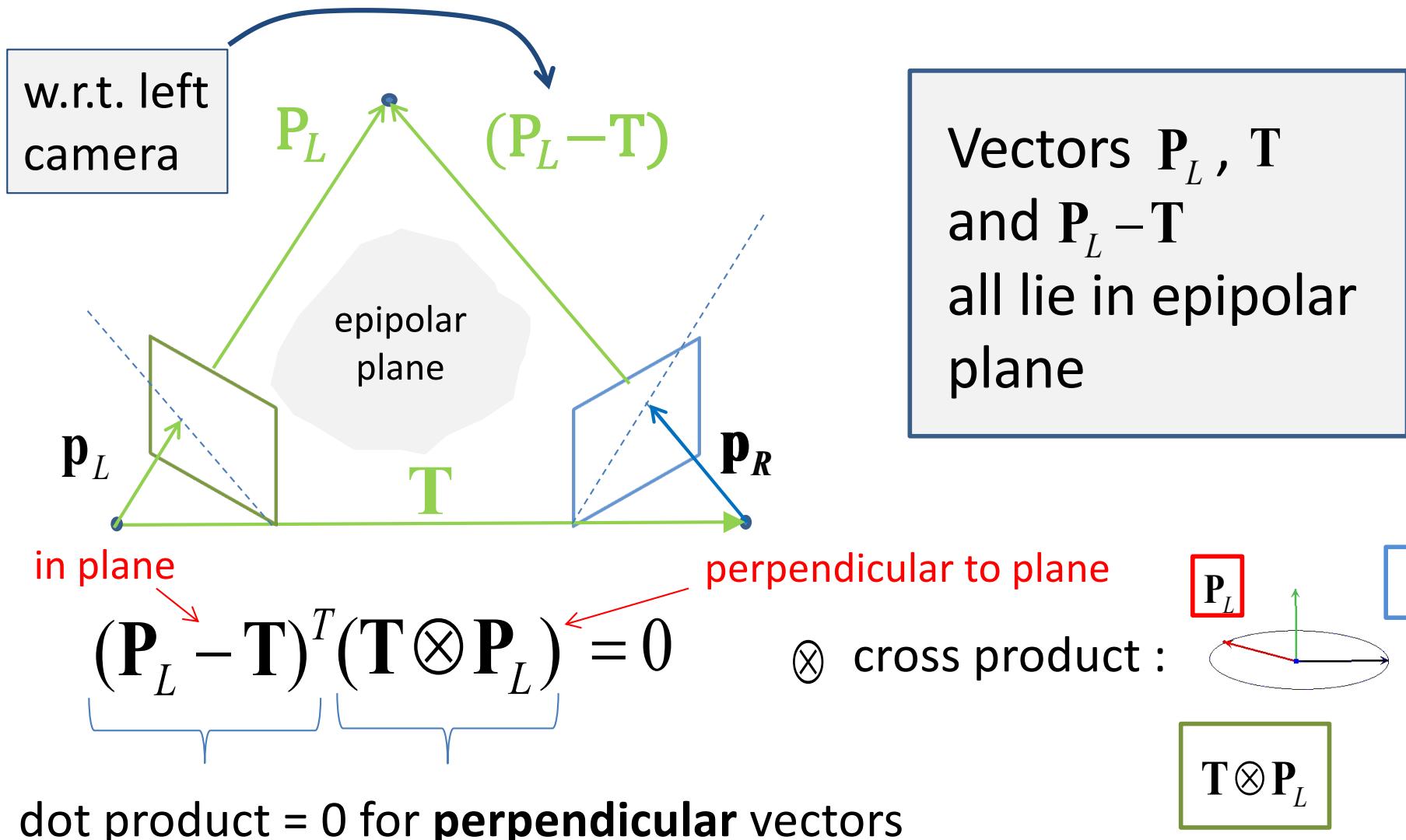
$$\mathbf{P}_R = R(\mathbf{P}_L - \mathbf{T})$$

Perspective projection:



$$\mathbf{P}_L = \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \quad \mathbf{p}_L = \begin{bmatrix} x_L \\ y_L \\ f \end{bmatrix} = \frac{f\mathbf{P}_L}{Z_L} \quad \mathbf{p}_R = \begin{bmatrix} x_R \\ y_R \\ f \end{bmatrix} = \frac{f\mathbf{P}_R}{Z_R}$$

Epipolar Geometry - Maths



Epipolar Geometry - Maths

$$(\mathbf{P}_L - \mathbf{T})^T (\mathbf{T} \otimes \mathbf{P}_L) = 0$$

$$(\mathbf{T} \otimes \mathbf{P}_L) = S \mathbf{P}_L$$

$$S = \begin{bmatrix} 0 & -T_Z & T_Y \\ T_Z & 0 & -T_X \\ -T_Y & T_X & 0 \end{bmatrix}$$

w.r.t. right camera

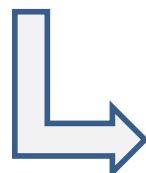
$$\mathbf{P}_R = R(\mathbf{P}_L - \mathbf{T})$$

$$R^T = R^{-1}$$

Rotation matrix

$$R^T \mathbf{P}_R = (\mathbf{P}_L - \mathbf{T})$$

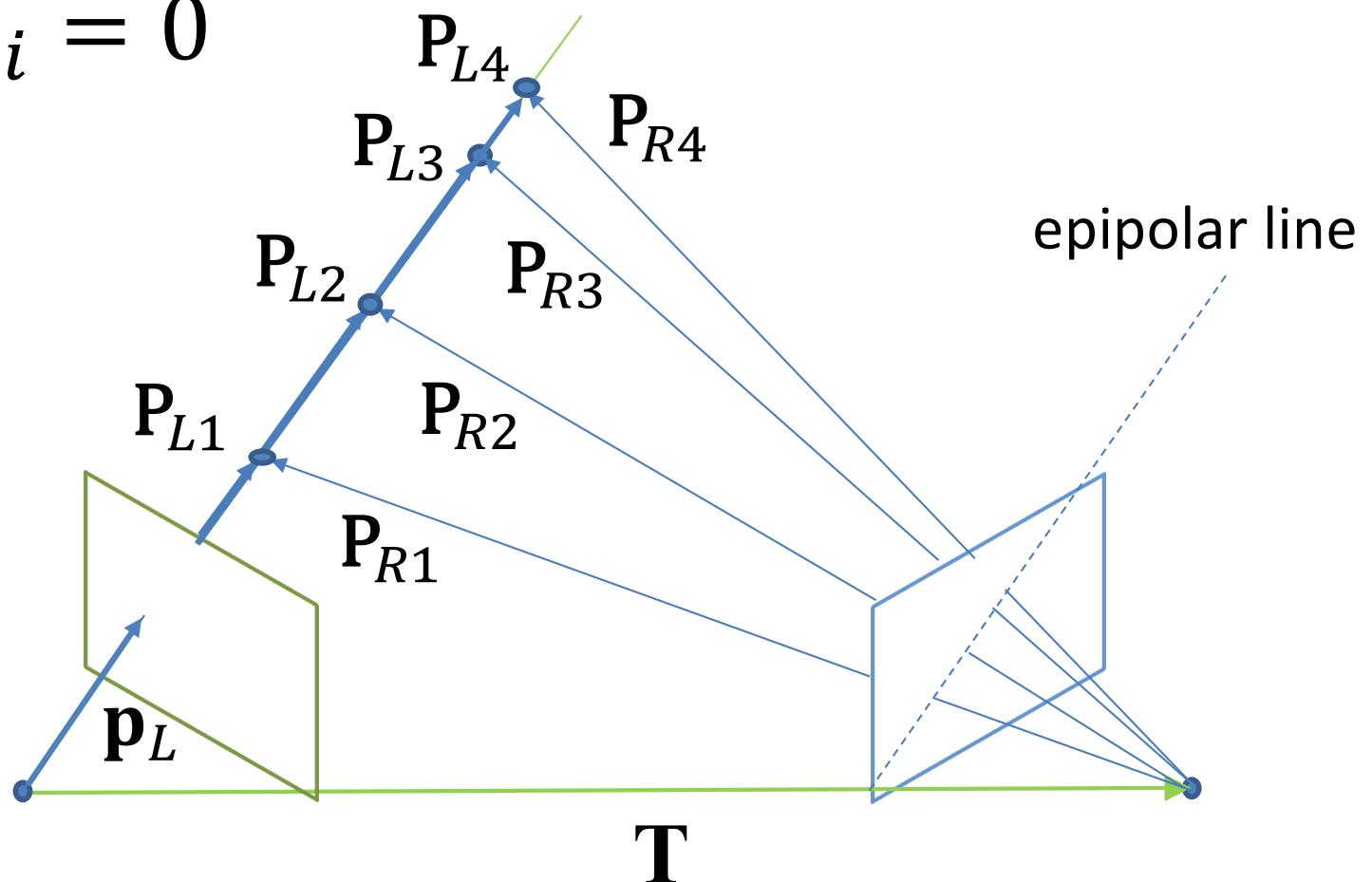
$$\mathbf{P}_R^T R = (\mathbf{P}_L - \mathbf{T})^T$$



$$\mathbf{P}_R^T R S \mathbf{P}_L = 0$$

Epipolar Line Constraint

$$\mathbf{P}_{Ri}^T R S \mathbf{P}_{Li} = 0$$



The Essential Matrix

$$\mathbf{P}_R^T \underbrace{R \ S}_{\text{ }} \mathbf{P}_L = 0 \quad \longrightarrow \quad \mathbf{P}_R^T E \mathbf{P}_L = 0$$

$E = RS \quad \longrightarrow \quad \text{the essential matrix}$

$$\mathbf{p}_L = \frac{f \mathbf{P}_L}{Z_L} \quad \mathbf{p}_R = \frac{f \mathbf{P}_R}{Z_R}$$

$$\longrightarrow \frac{\cancel{Z}_R}{\cancel{f}} \mathbf{p}_R^T E \frac{\cancel{Z}_L}{\cancel{f}} \mathbf{p}_L = 0$$

$$\mathbf{P}_L = \frac{Z_L \mathbf{p}_L}{f} \quad \mathbf{P}_R = \frac{Z_R \mathbf{p}_R}{f}$$

$$\longrightarrow \mathbf{p}_R^T E \mathbf{p}_L = 0$$

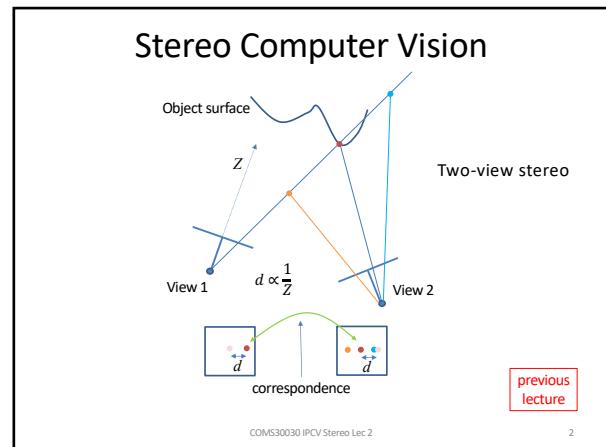
COMS30030
Image Processing and Computer Vision

Stereo 2 – Epipolar Geometry

Andrew Calway
andrew@cs.bris.ac.uk

COMS30030 IPCV Stereo Lec 2

1



2

2-D Coordinate Transformations

Rotated coordinate system
 $v' = (x', y')$

Rotated and translated coordinate system
 $v' = R(-\theta)(v - t)$

$v' = R(-\theta)v$

Vector representation in rotated coordinate system

previous lecture

COMS30030 Stereo Lec 1

3

3-D Coordinate Transformations

\mathbf{P}_W : Vector defining P in world coordinates

\mathbf{P}_C : Vector defining P in camera coordinates

\mathbf{T} : 3-D camera position vector

R : 3-D camera rotation matrix

R defines rotation to be applied to camera coordinate system to align it with world coordinate system

$\mathbf{P}_c = R(\mathbf{P}_W - \mathbf{T})$

$\mathbf{P}_W = R^{-1}\mathbf{P}_C + \mathbf{T} = R^T\mathbf{P}_C + \mathbf{T}$

Rotation matrices: $R^T = R^{-1}$

world coordinate frame

camera coordinate frame

previous lecture

COMS30030 Stereo Lec 1

4

Homogeneous Coordinates

- Homogeneous coordinates allow coordinate transformations to be defined by 4x4 matrices:

$$\mathbf{P}'_W = \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_W \end{bmatrix} = \begin{bmatrix} R^T & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P}_C \end{bmatrix} = H_{CW} \mathbf{P}'_C \Leftrightarrow \mathbf{P}_W = R^T \mathbf{P}_C + \mathbf{T}$$

$$\mathbf{P}'_C = H_{CW}^{-1} \mathbf{P}'_W = H_{WC} \mathbf{P}'_W \quad H_{CW} = \begin{bmatrix} R_{00} & R_{10} & R_{20} & T_x \\ R_{01} & R_{11} & R_{21} & T_y \\ R_{02} & R_{12} & R_{22} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H_{WC} = \begin{bmatrix} R & -R\mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \Leftrightarrow H_{WC} H_{CW} = I \quad I - \text{identity matrix}$$

$$\Leftrightarrow \mathbf{P}_C = R(\mathbf{P}_W - \mathbf{T})$$

H_{CW} : camera to world coordinate transformation matrix

COMS30030 Stereo Lec 1

5

Stereo Coordinate Systems

$\mathbf{P}'_L = H_{WL} \mathbf{P}'_W$

$\mathbf{P}'_R = H_{WR} \mathbf{P}'_W$

$\mathbf{P}'_L = H_{WL} H_{WR}^{-1} \mathbf{P}'_R$

$\mathbf{P}'_L = H_{RL} \mathbf{P}'_R$

$H_{RL} = \begin{bmatrix} R^T & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix}$

$\mathbf{P}_L = R^T \mathbf{P}_R + \mathbf{T}$

$\mathbf{P}_R = R(\mathbf{P}_L - \mathbf{T})$

left camera coordinate system

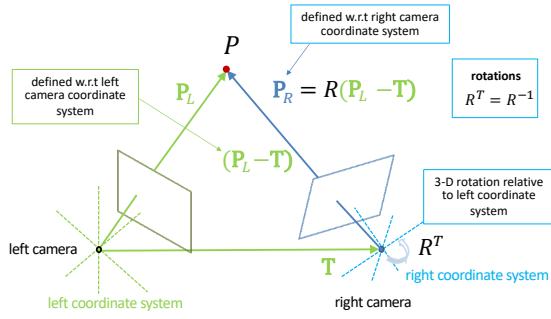
right camera coordinate system

R defines rotation to be applied to right camera coordinate system to align it with left coordinate system

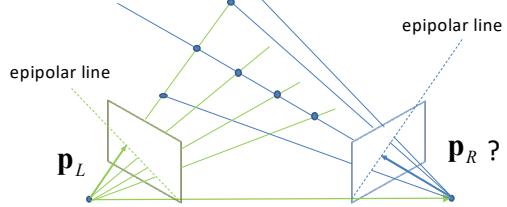
COMS30030 Stereo Lec 1

6

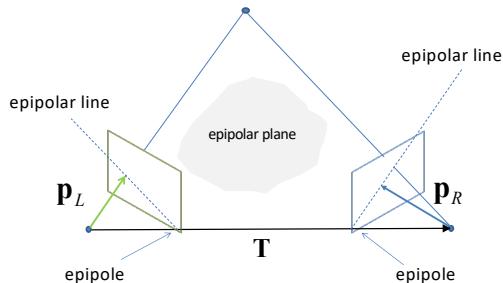
General Two-View Stereo



Epipolar Lines



Epipolar Planes



Epipolar Geometry

- Epipolar geometry defines relationship between two stereo views
- For known viewpoints:
 - it constrains matches to lie along epipolar lines
- For unknown viewpoints:
 - given matching points
 - it enables estimation of viewpoints

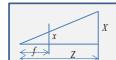
COMS30030 IPCV Stereo Lec 2

10

Epipolar Geometry - Maths

Rigid transformation between cameras:

$$\mathbf{P}_R = R(\mathbf{P}_L - \mathbf{T})$$



Perspective projection:

$$\mathbf{P}_L = \begin{bmatrix} X_L \\ Y_L \\ Z_L \end{bmatrix} \quad \mathbf{p}_L = \begin{bmatrix} x_L \\ y_L \\ f \end{bmatrix} = \frac{f\mathbf{P}_L}{Z_L} \quad \mathbf{p}_R = \begin{bmatrix} x_R \\ y_R \\ f \end{bmatrix} = \frac{f\mathbf{P}_R}{Z_R}$$

COMS30030 IPCV Stereo Lec 2

11

Epipolar Geometry - Maths

w.r.t. left camera

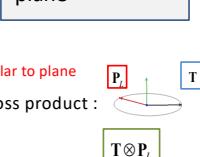
$$\mathbf{P}_L = (\mathbf{P}_L - \mathbf{T}) + \mathbf{T}$$

in plane

$$(\mathbf{P}_L - \mathbf{T})^T (\mathbf{T} \otimes \mathbf{P}_L) = 0$$

dot product = 0 for perpendicular vectors

Vectors \mathbf{P}_L , \mathbf{T} and $\mathbf{P}_L - \mathbf{T}$ all lie in epipolar plane



COMS30030 IPCV Stereo Lec 2

12

11

Epipolar Geometry - Maths

$$(\mathbf{P}_L - \mathbf{T})^T (\mathbf{T} \otimes \mathbf{P}_L) = 0$$

$$(\mathbf{T} \otimes \mathbf{P}_L) = S \mathbf{P}_L$$

$$S = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

w.r.t. right camera

$$\mathbf{P}_R = R(\mathbf{P}_L - \mathbf{T})$$

$$R^T = R^{-1} \quad \text{Rotation matrix}$$

$$R^T \mathbf{P}_R = (\mathbf{P}_L - \mathbf{T})$$

$$\mathbf{P}_R^T R = (\mathbf{P}_L - \mathbf{T})^T$$

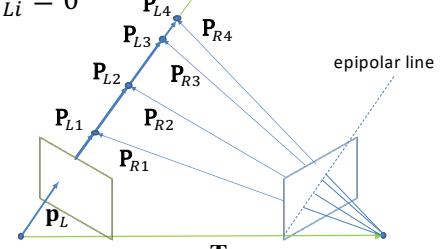
$$\mathbf{P}_R^T R S \mathbf{P}_L = 0$$

COMS30030 IPCV Stereo Lec 2

13

Epipolar Line Constraint

$$\mathbf{P}_{Ri}^T R S \mathbf{P}_{Li} = 0$$



COMS30030 IPCV Stereo Lec 2

14

The Essential Matrix

$$\mathbf{P}_R^T R S \mathbf{P}_L = 0 \iff \mathbf{P}_R^T E \mathbf{P}_L = 0$$

$E = RS \iff$ the essential matrix

$$\mathbf{p}_L = \frac{f\mathbf{P}_L}{Z_L} \quad \mathbf{p}_R = \frac{f\mathbf{P}_R}{Z_R}$$

$$\mathbf{p}_L = \frac{Z_L \mathbf{p}_L}{f} \quad \mathbf{p}_R = \frac{Z_R \mathbf{p}_R}{f}$$

$$\Rightarrow \frac{Z_R}{f} \mathbf{p}_R^T E \frac{Z_L}{f} \mathbf{p}_L = 0$$

$$\mathbf{p}_R^T E \mathbf{p}_L = 0$$

COMS30030 IPCV Stereo Lec 2

15

COMS30030
Image Processing and Computer Vision

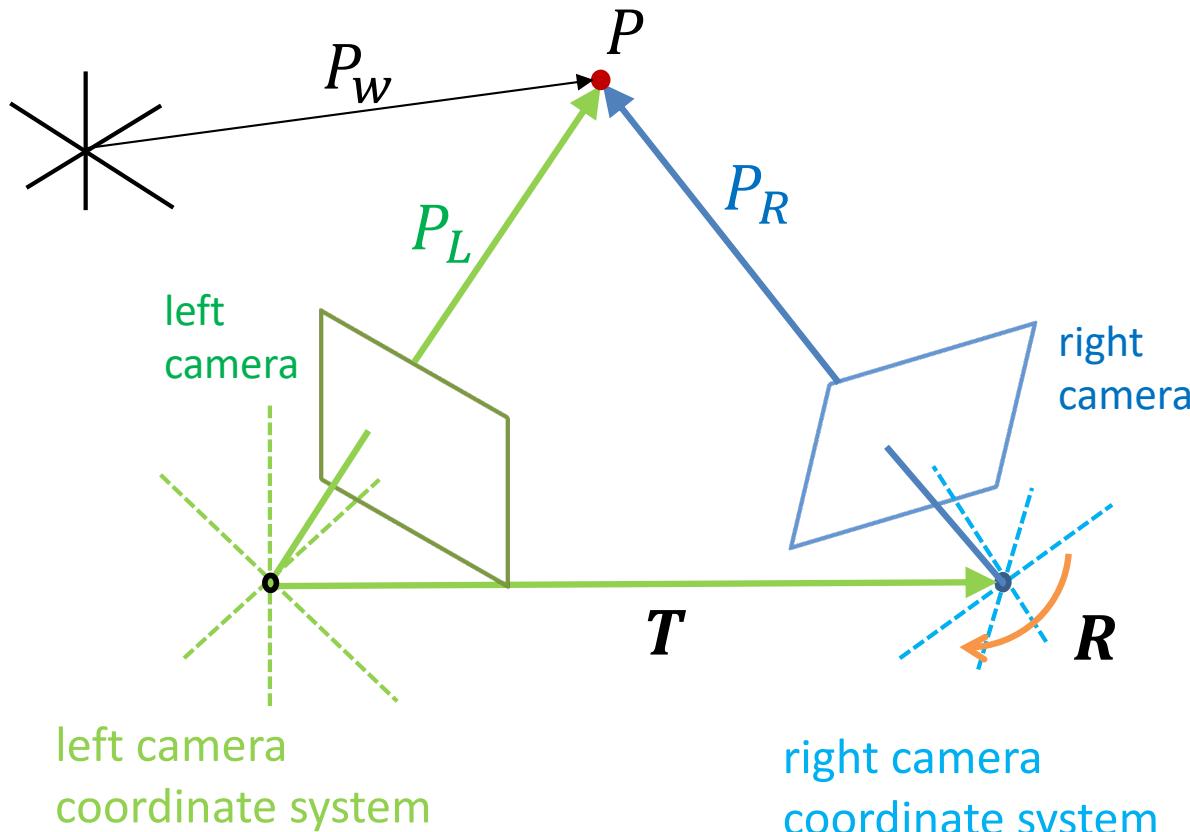
Stereo – 3-D Reconstruction

Andrew Calway

andrew@cs.bris.ac.uk

Stereo Coordinate Systems

previous
lecture



R defines rotation to be applied to right camera coordinate system to align it with left coordinate system

$$P'_L = H_{WL}P'_W$$

$$P'_R = H_{WR}P'_W$$

$$P'_L = H_{WL}H_{WR}^{-1}P'_R$$

$$P'_L = H_{RL}P'_R$$

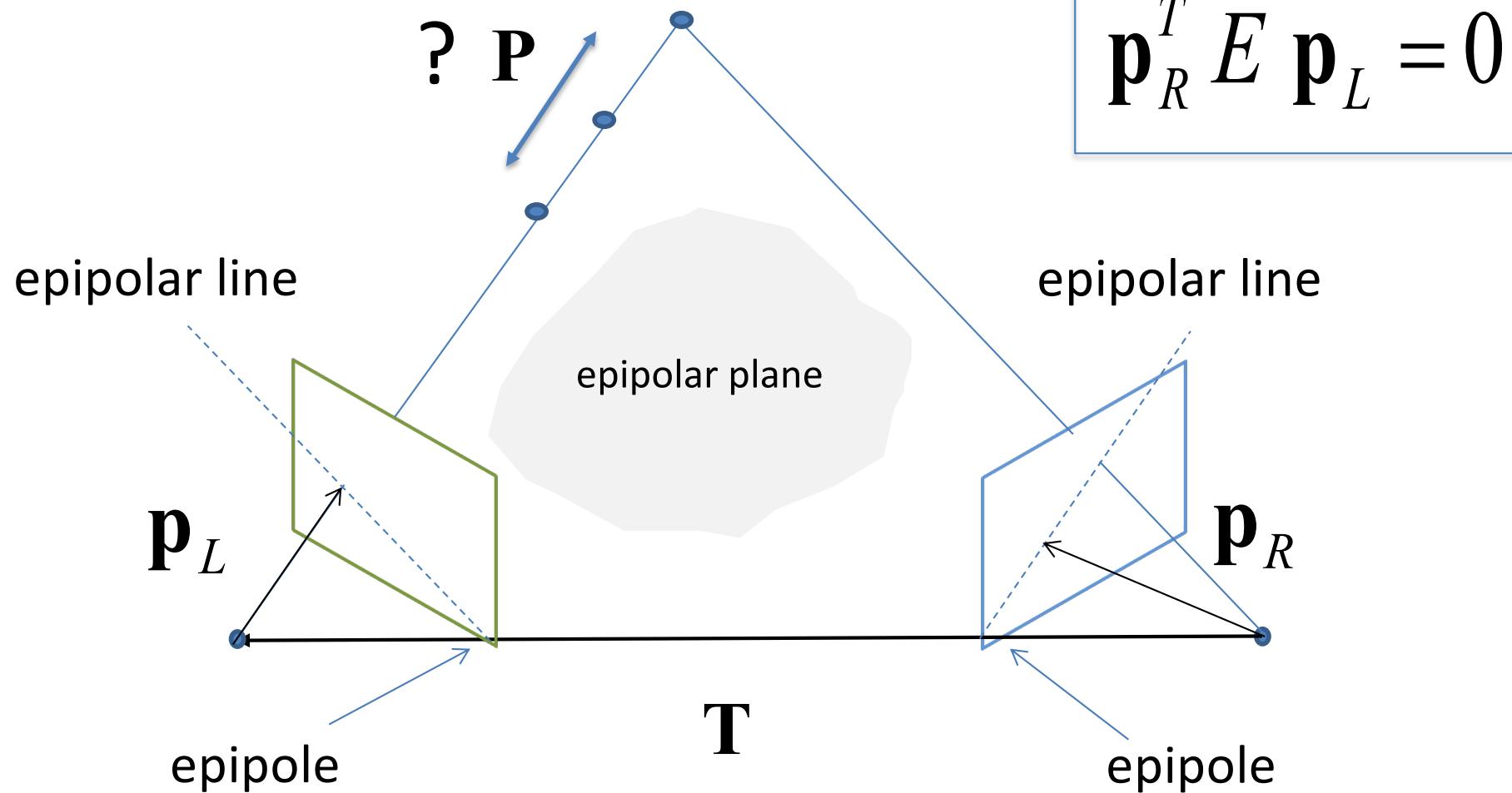
$$H_{RL} = \begin{bmatrix} R^T & T \\ \mathbf{0} & 1 \end{bmatrix}$$

$$P_L = R^T P_R + T$$

$$P_R = R(P_L - T)$$

Epipolar Geometry

previous
lecture



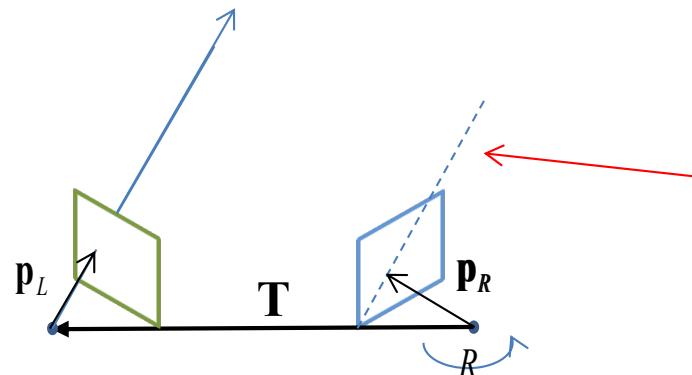
Epipolar Lines

$$\mathbf{p}_R^T E \mathbf{p}_L = 0$$



Let $\mathbf{u}_L = E \mathbf{p}_L = \begin{bmatrix} u_{L1} \\ u_{L2} \\ u_{L3} \end{bmatrix}$

$$\mathbf{p}_R^T E \mathbf{p}_L = \mathbf{p}_R^T \mathbf{u}_L = x_R u_{L1} + y_R u_{L2} + f u_{L3} = 0$$



Equation of epipolar line in
right image

Image Points and Pixels

- Pixel values represent light intensity within small region of image plane, e.g. of size $s_x \times s_y$

- Pixel coordinates: (\hat{x}, \hat{y})

- Image coordinates:

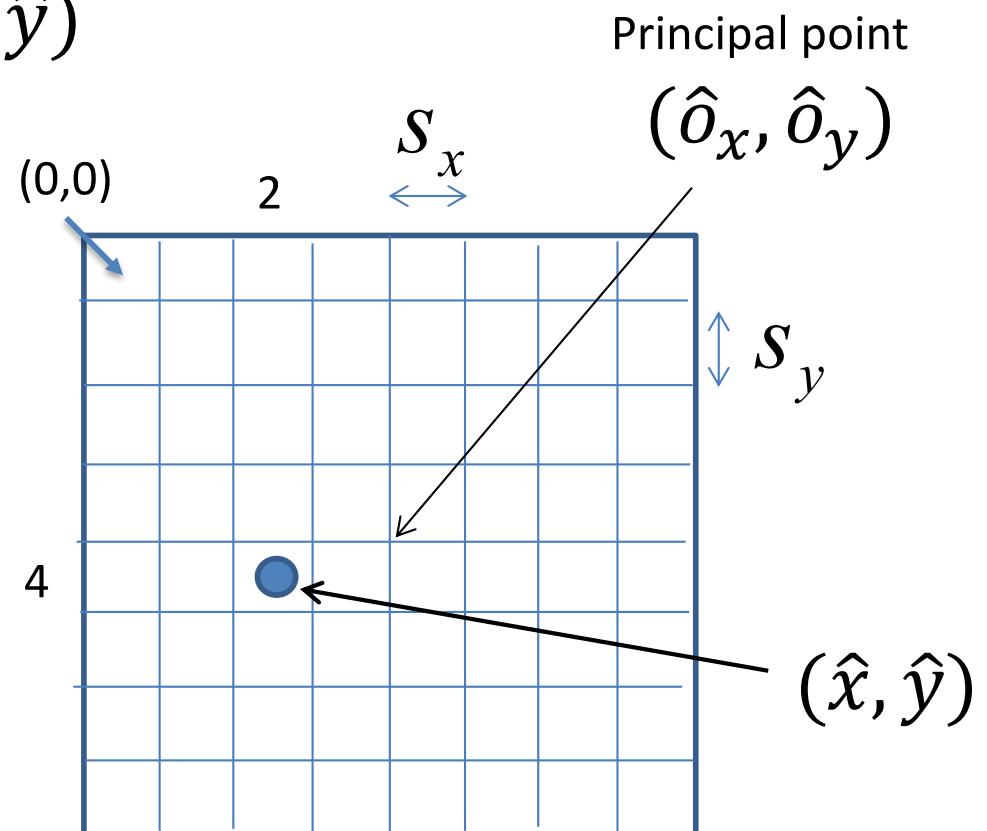
$$x = s_x(\hat{x} - \hat{o}_x)$$

$$y = s_y(\hat{y} - \hat{o}_y)$$

Example: $s_x = s_y = 2$

$$x = 2(2 - 3.5) = -3$$

$$y = 2(4 - 3.5) = 1$$



Fundamental Matrix

$$\begin{aligned}x &= s_x(\hat{x} - \hat{o}_x) \\y &= s_y(\hat{y} - \hat{o}_y)\end{aligned}\quad \Rightarrow \quad \mathbf{p}_L = \begin{bmatrix}x_L \\ y_L \\ f\end{bmatrix} = M_L \begin{bmatrix}\hat{x}_L \\ \hat{y}_L \\ f\end{bmatrix} = M_L \hat{\mathbf{p}}_L$$

$$\mathbf{p}_R^T E \mathbf{p}_L = 0 \quad \Rightarrow \quad \hat{\mathbf{p}}_R^T M_R^T E M_L \hat{\mathbf{p}}_L = 0$$

$$\Rightarrow \quad \hat{\mathbf{p}}_R^T F \hat{\mathbf{p}}_L = 0 \quad \quad F = M_R^T E M_L$$

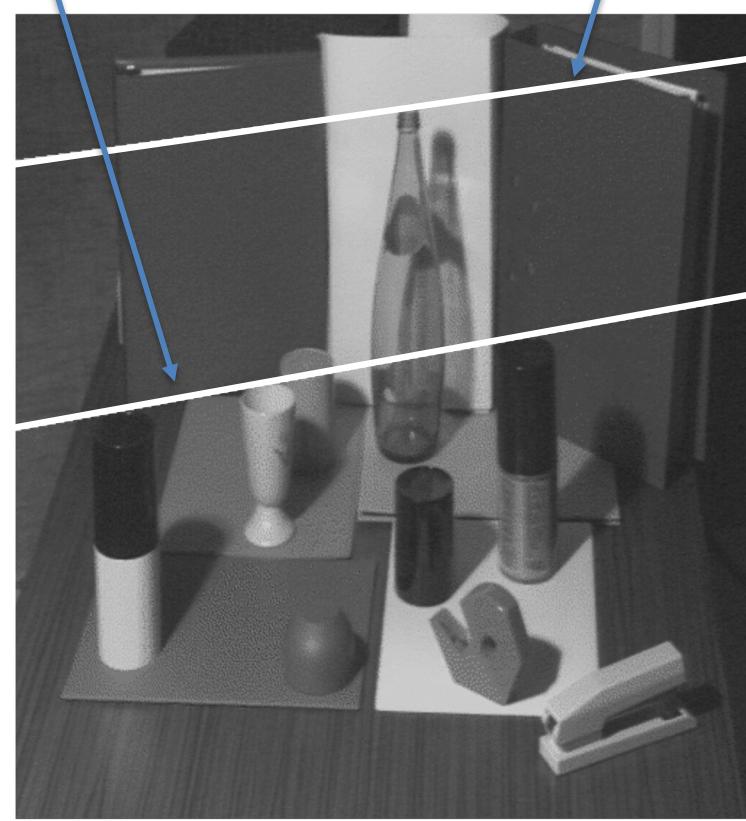
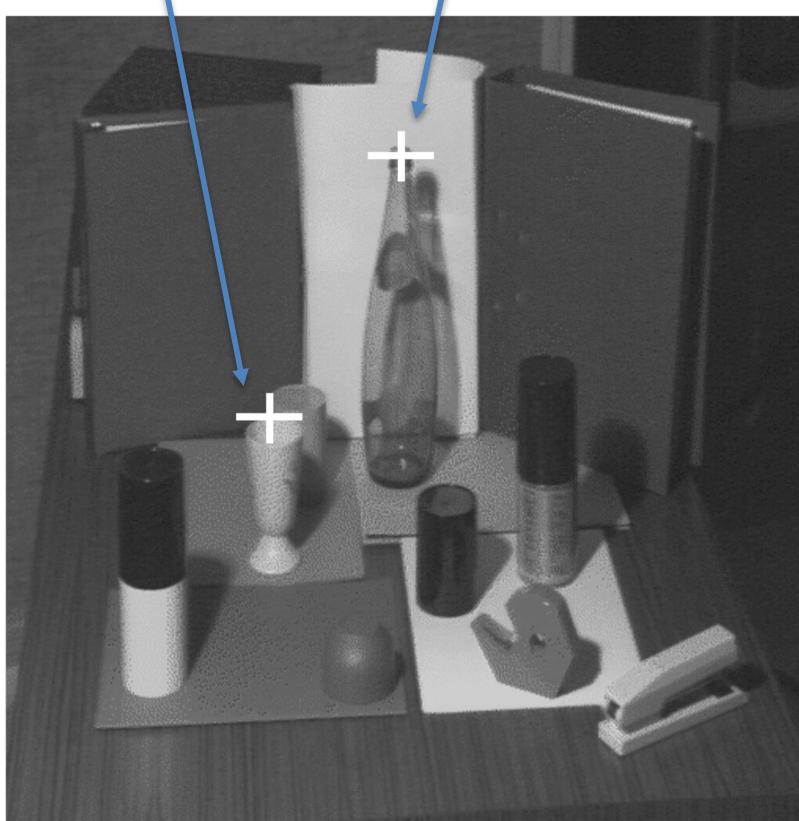
The fundamental matrix

Epipolar Lines - Example

 \hat{p}_{L1} \hat{p}_{L2}

$$\hat{\mathbf{p}}_{R1}^T F \hat{\mathbf{p}}_{L1} = 0$$

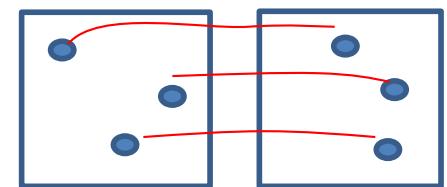
$$\hat{\mathbf{p}}_{R2}^T F \hat{\mathbf{p}}_{L2} = 0$$



F from Correspondences

- Given set of correspondences, $i = 1 \dots N$, we can also estimate the fundamental matrix :

$$\hat{\mathbf{p}}_{Ri}^T F \hat{\mathbf{p}}_{Li} = 0 \quad i = 1 \dots N$$



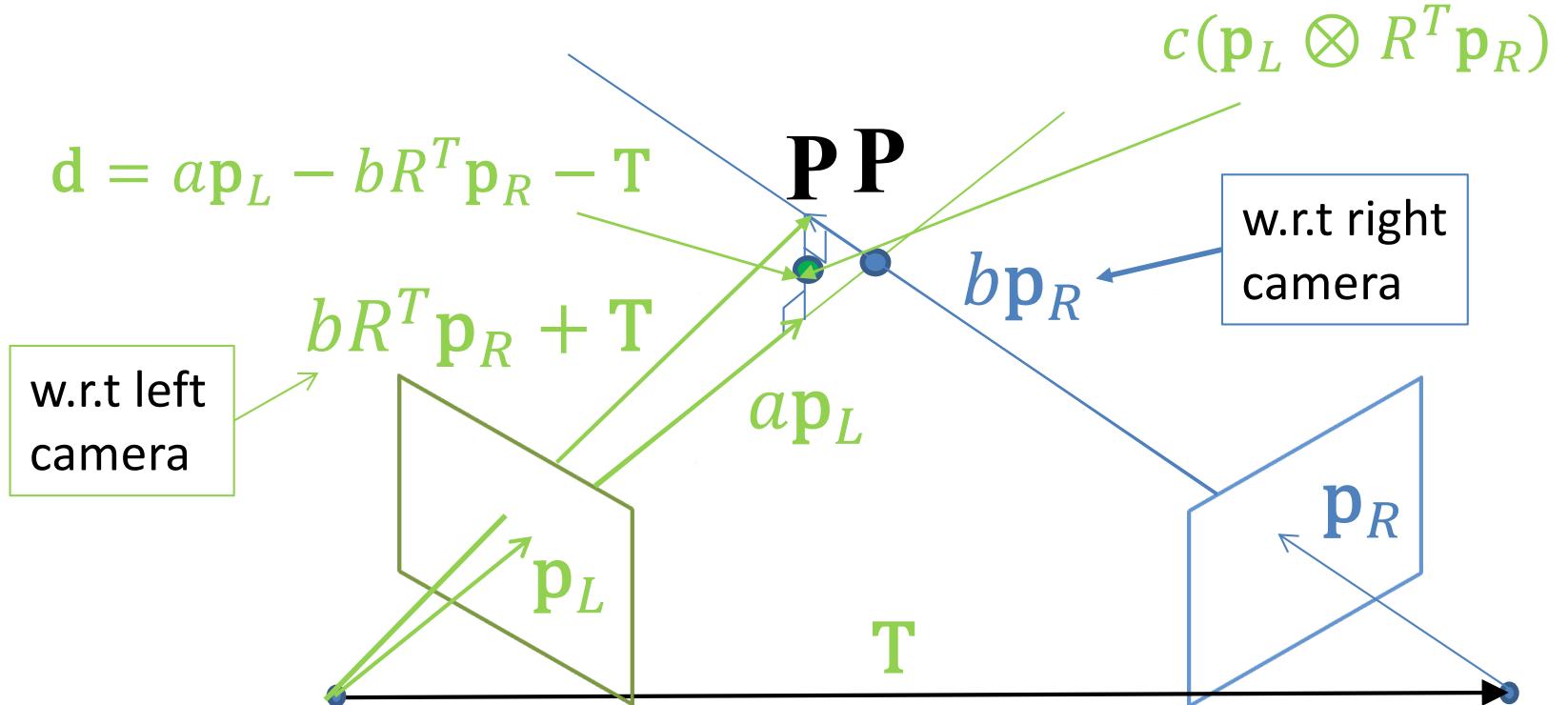
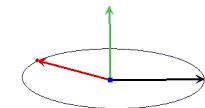
$$\xrightarrow{} A\mathbf{v} = 0$$

$N \times 9$ matrix defined by correspondence vectors $\hat{\mathbf{p}}_{Li}$ and $\hat{\mathbf{p}}_{Ri}$

Components of F

Solve for \mathbf{v} using Singular Value Decomposition

3-D Reconstruction



→ find a, b, c s.t: $ap_L - bR^T p_R - T - c(p_L \otimes R^T p_R) = 0$

3-D Reconstruction

find a, b, c s.t: $a \mathbf{p}_L - b R^T \mathbf{p}_R - \mathbf{T} - c (\mathbf{p}_L \otimes R^T \mathbf{p}_R) = 0$

Given **corresponding points**, we know : $\mathbf{p}_L, \mathbf{p}_R$

Given **calibrated views**, we know : R, \mathbf{T}

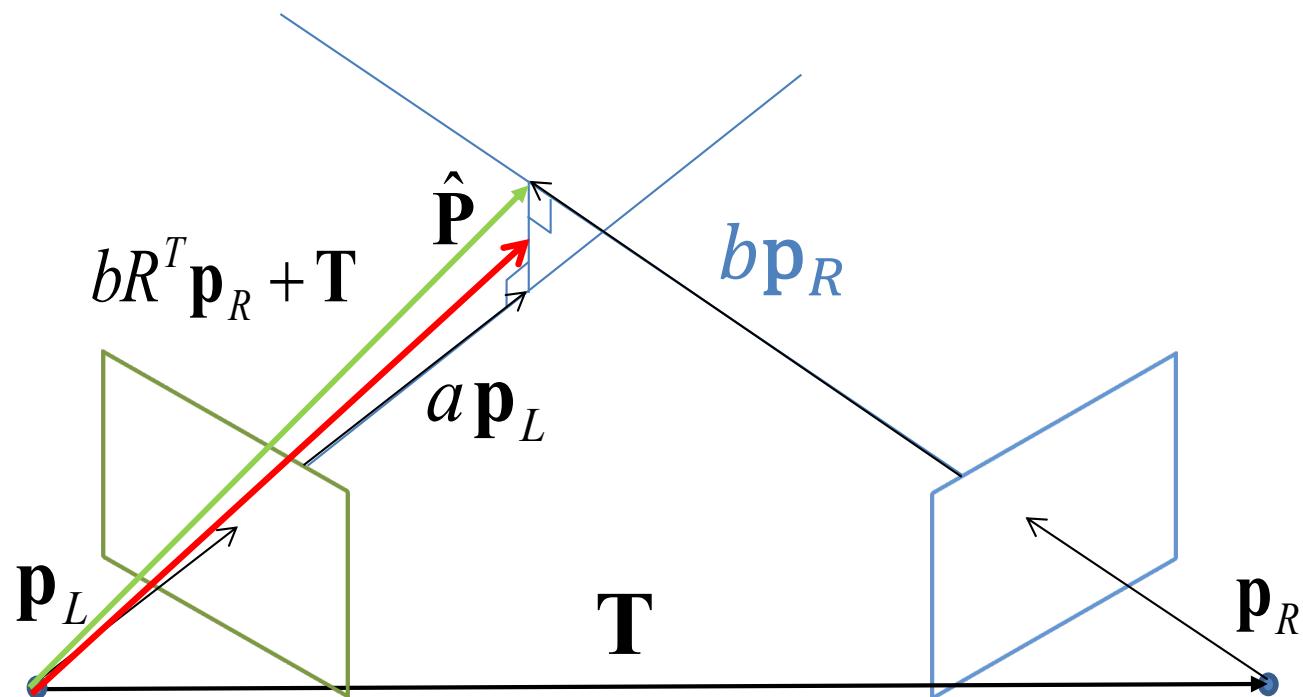
$$a \begin{bmatrix} \bullet \\ \mathbf{p}_L \\ \bullet \end{bmatrix}_{3 \times 1} - b \begin{bmatrix} R^T \mathbf{p}_R \end{bmatrix}_{3 \times 1} - c \begin{bmatrix} \mathbf{p}_L \otimes R^T \mathbf{p}_R \end{bmatrix}_{3 \times 1} = \begin{bmatrix} \mathbf{T} \end{bmatrix}_{3 \times 1}$$

3-D Reconstruction

$$a \begin{bmatrix} \bullet \\ \mathbf{p}_L \\ \bullet \end{bmatrix} - b \begin{bmatrix} R^T \mathbf{p}_R \end{bmatrix}_{3 \times 1} - c \begin{bmatrix} \mathbf{p}_L \otimes R^T \mathbf{p}_R \end{bmatrix}_{3 \times 1} = \begin{bmatrix} \mathbf{T} \end{bmatrix}_{3 \times 1}$$

$$\xrightarrow{\hspace{1cm}} H \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{T} \quad \xrightarrow{\hspace{1cm}} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = H^{-1} \mathbf{T}$$

3-D Reconstruction



$$\hat{\mathbf{P}} = (a \mathbf{p}_L + b R^T \mathbf{p}_R + T) / 2$$

COMS30030
Image Processing and Computer Vision
Stereo – 3-D Reconstruction
Andrew Calway
andrew@cs.bris.ac.uk

COMS30030 IPCV Stereo 3

1

Stereo Coordinate Systems

[previous lecture](#)

$$\begin{aligned}P'_L &= H_{WL}P'_W \\P'_R &= H_{WR}P'_W \\P'_L &= H_{WL}H_{WR}^{-1}P'_R \\P'_L &= H_{RL}P'_R \\H_{RL} &= \begin{bmatrix} R^T & T \\ \mathbf{0} & 1 \end{bmatrix} \\P_L &= R^T P_R + T \\P_R &= R(P_L - T)\end{aligned}$$

COMS30121 IPCV Stereo I

2

Epipolar Geometry

[previous lecture](#)

$$\mathbf{p}_R^T E \mathbf{p}_L = 0$$

epipole

epipole

epipolar line

epipolar plane

epipole

epipole

COMS30030 IPCV Stereo 3

3

Epipolar Lines

$\mathbf{p}_R^T E \mathbf{p}_L = 0$

Let $\mathbf{u}_L = E \mathbf{p}_L = \begin{bmatrix} u_{L1} \\ u_{L2} \\ u_{L3} \end{bmatrix}$

$$\mathbf{p}_R^T E \mathbf{p}_L = \mathbf{p}_R^T \mathbf{u}_L = x_R u_{L1} + y_R u_{L2} + f u_{L3} = 0$$

Equation of epipolar line in right image

COMS30030 IPCV Stereo 3

4

Image Points and Pixels

- Pixel values represent light intensity within small region of image plane, e.g. of size $s_x \times s_y$
- Pixel coordinates: (\hat{x}, \hat{y})
- Image coordinates: (\hat{x}, \hat{y})

$x = s_x(\hat{x} - \hat{o}_x)$

$y = s_y(\hat{y} - \hat{o}_y)$

Example: $s_x = s_y = 2$
 $x = 2(2 - 3.5) = -3$
 $y = 2(4 - 3.5) = 1$

Principal point (\hat{o}_x, \hat{o}_y)

(\hat{x}, \hat{y})

$(0,0)$ 2 s_x (\hat{o}_x, \hat{o}_y)

4 s_y

COMS30030 IPCV Stereo 3

5

Fundamental Matrix

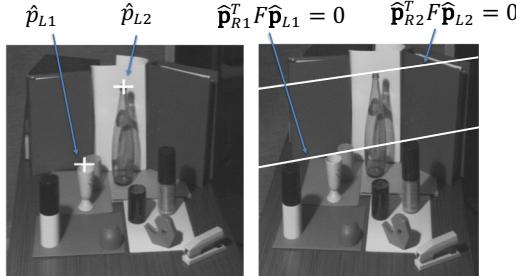
$$\begin{aligned}x = s_x(\hat{x} - \hat{o}_x) &\Rightarrow \mathbf{p}_L = \begin{bmatrix} x \\ y \\ f \end{bmatrix} = M_L \begin{bmatrix} \hat{x}_L \\ \hat{y}_L \\ f \end{bmatrix} = M_L \hat{\mathbf{p}}_L \\y = s_y(\hat{y} - \hat{o}_y) &\Rightarrow \mathbf{p}_R = \begin{bmatrix} x \\ y \\ f \end{bmatrix} = M_R \begin{bmatrix} \hat{x}_R \\ \hat{y}_R \\ f \end{bmatrix} = M_R \hat{\mathbf{p}}_R \\ \mathbf{p}_R^T E \mathbf{p}_L = 0 &\Rightarrow \hat{\mathbf{p}}_R^T M_R^T E M_L \hat{\mathbf{p}}_L = 0 \\ &\Rightarrow \hat{\mathbf{p}}_R^T F \hat{\mathbf{p}}_L = 0 \quad F = M_R^T E M_L\end{aligned}$$

The fundamental matrix

COMS30030 IPCV Stereo 3

6

Epipolar Lines - Example



COMS30030 IPCV Stereo 3

7

F from Correspondences

- Given set of correspondences, $i=1\dots N$, we can also estimate the fundamental matrix :

$$\hat{p}_{Ri}^T F \hat{p}_{Li} = 0 \quad i = 1..N$$

$$\Rightarrow A\mathbf{v} = 0$$

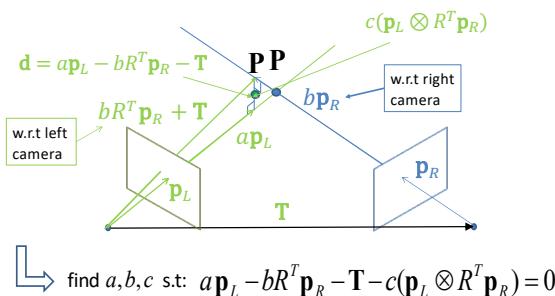
$N \times 9$ matrix defined by correspondence vectors \hat{p}_{Li} and \hat{p}_{Ri}

Solve for \mathbf{v} using Singular Value Decomposition

COMS30030 IPCV Stereo 3

8

3-D Reconstruction



COMS30030 IPCV Stereo 3

9

3-D Reconstruction

$$\text{find } a, b, c \text{ s.t: } a p_L - b R^T p_R - T - c (p_L \otimes R^T p_R) = 0$$

Given corresponding points, we know: p_L, p_R

Given calibrated views, we know: R, T

$$a \begin{bmatrix} \bullet \\ p_L \\ \bullet \\ 3 \times 1 \end{bmatrix} - b \begin{bmatrix} R^T p_R \\ 3 \times 1 \end{bmatrix} - c \begin{bmatrix} p_L \otimes R^T p_R \\ 3 \times 1 \end{bmatrix} = \begin{bmatrix} T \\ 3 \times 1 \end{bmatrix}$$

COMS30030 IPCV Stereo 3

10

3-D Reconstruction

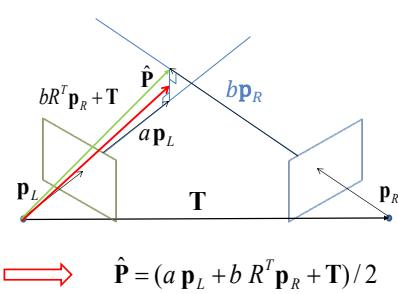
$$a \begin{bmatrix} \bullet \\ p_L \\ \bullet \\ 3 \times 1 \end{bmatrix} - b \begin{bmatrix} R^T p_R \\ 3 \times 1 \end{bmatrix} - c \begin{bmatrix} p_L \otimes R^T p_R \\ 3 \times 1 \end{bmatrix} = \begin{bmatrix} T \\ 3 \times 1 \end{bmatrix}$$

$$\Rightarrow H \begin{bmatrix} a \\ b \\ c \end{bmatrix} = T \quad \Rightarrow \quad \begin{bmatrix} a \\ b \\ c \end{bmatrix} = H^{-1} T$$

COMS30030 IPCV Stereo 3

11

3-D Reconstruction



COMS30030 IPCV Stereo 3

12

11

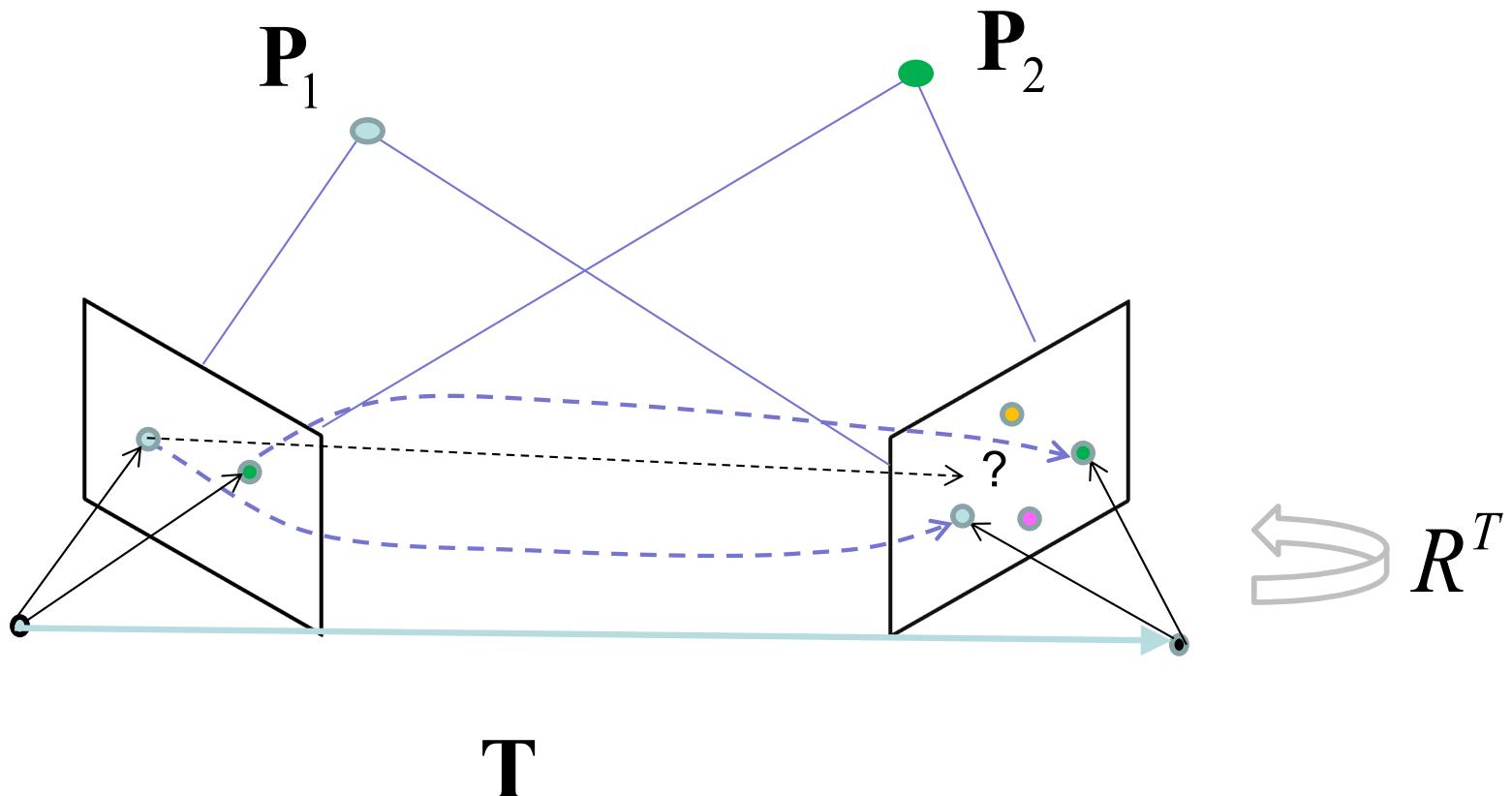
COMS30030
Image Processing and Computer Vision

Stereo – Correspondence Matching

Andrew Calway

andrew@cs.bris.ac.uk

Stereo Correspondence

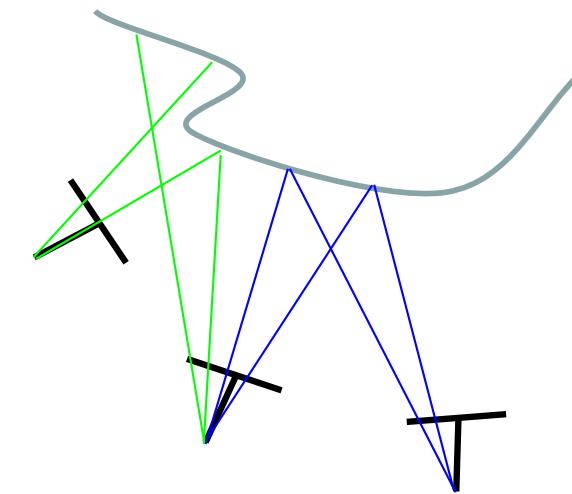
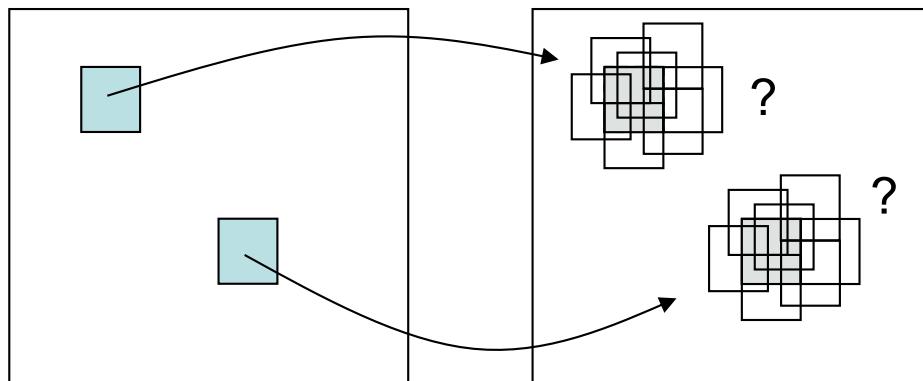


Stereo Matching is Hard



Region-Based Methods

- Compare pixel values within regions in two views.
- For region in left image, compute similarity with regions of same size in right image.
- Corresponding point - centre of most similar region



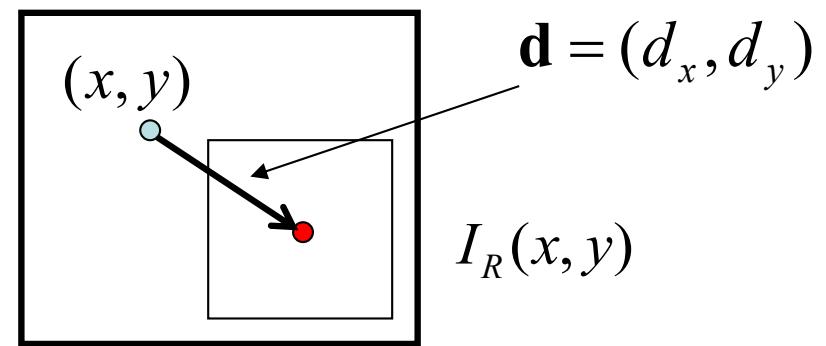
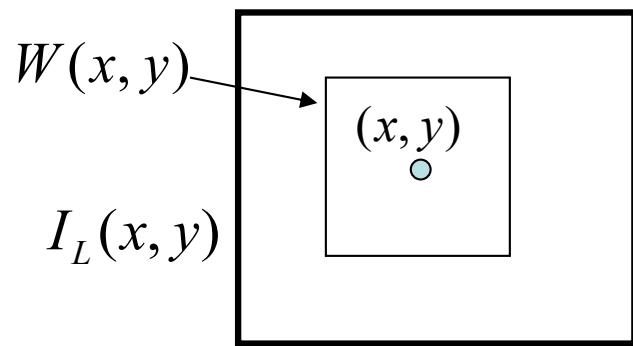
Region Matching

- Stereo image pair: $I_L(x, y)$ and $I_R(x, y)$
- For each pixel, find disparity $\mathbf{d} = (d_x, d_y)$ which minimises (or maximises) cost function

$$c(\mathbf{d}) = \sum_{(i,j) \in W(x,y)} s[I_L(i,j), I_R(i + d_x, j + d_y)]$$

$W(x, y) \rightarrow$ window of pixels around (x, y)

similarity measure



Similarity Measures

- **Sum of squared differences:** $s(u, v) = (u - v)^2$

- **Similar pixel count:**

$$s(u, v) = \begin{cases} 1 & \text{if } |u - v| < T \\ 0 & \text{else} \end{cases}$$

- **Normalised cross correlation:**

$$s(u, v) = (u - \bar{u})(v - \bar{v}) / N_u N_v$$

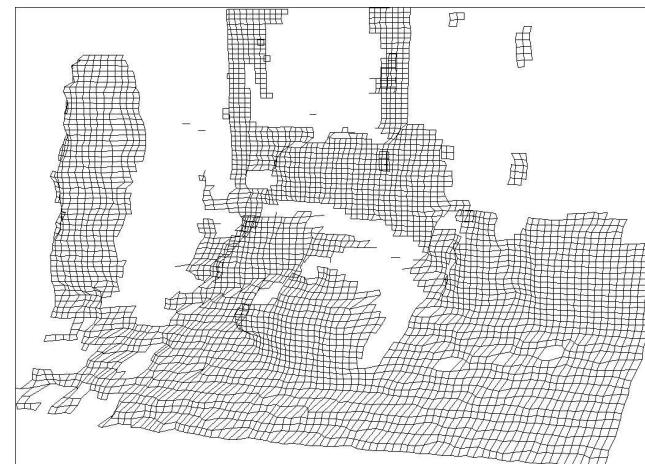
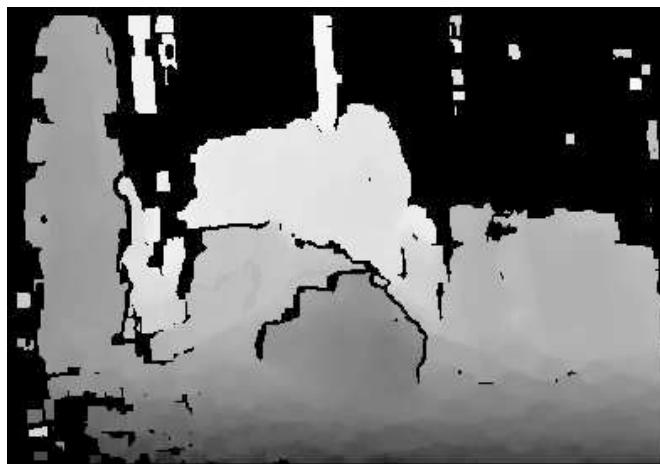
$$\bar{u} = \frac{1}{|W|} \sum_{u \in W} u$$

mean

$$N_u = \sqrt{\sum_{u \in W} (u - \bar{u})^2}$$

Std deviation

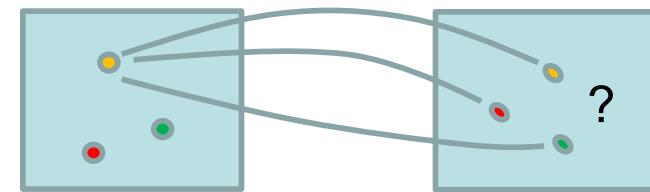
Example



Real time correlation-based stereo, Faugeras et al, 1993

Feature-Based Methods

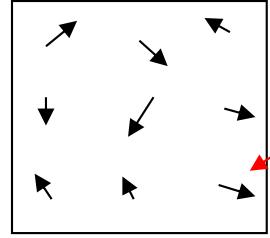
- Restrict search to sparse set of features
 - reduces mis-matches caused by texture-less regions
- Find salient (distinct) points in each view and match points by comparing pixels or image descriptors in local regions about each point
- Examples
 - Harris corner detector (salient points)
 - Scale-Invariant Feature Transform (SIFT)



Harris Corner Detector

- Detects salient (distinct, interesting) image points
- Covariance of spatial gradient vectors within region W

$$A = \sum_{x,y \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

W  (I_x, I_y) spatial gradient

$I_x \equiv I_x(x, y)$

- Eigenvalues λ_1 and λ_2 of A indicate ‘spread’ of gradients in region, e.g. 2 high values \rightarrow ‘busy’ region.
- Example saliency metric: $sal = \lambda_1 \lambda_2 / (\lambda_1 + \lambda_2)$
- Properties:
 - if eigenvalues both large $\rightarrow sal$ large
 - if either eigenvalue small $\rightarrow sal$ small

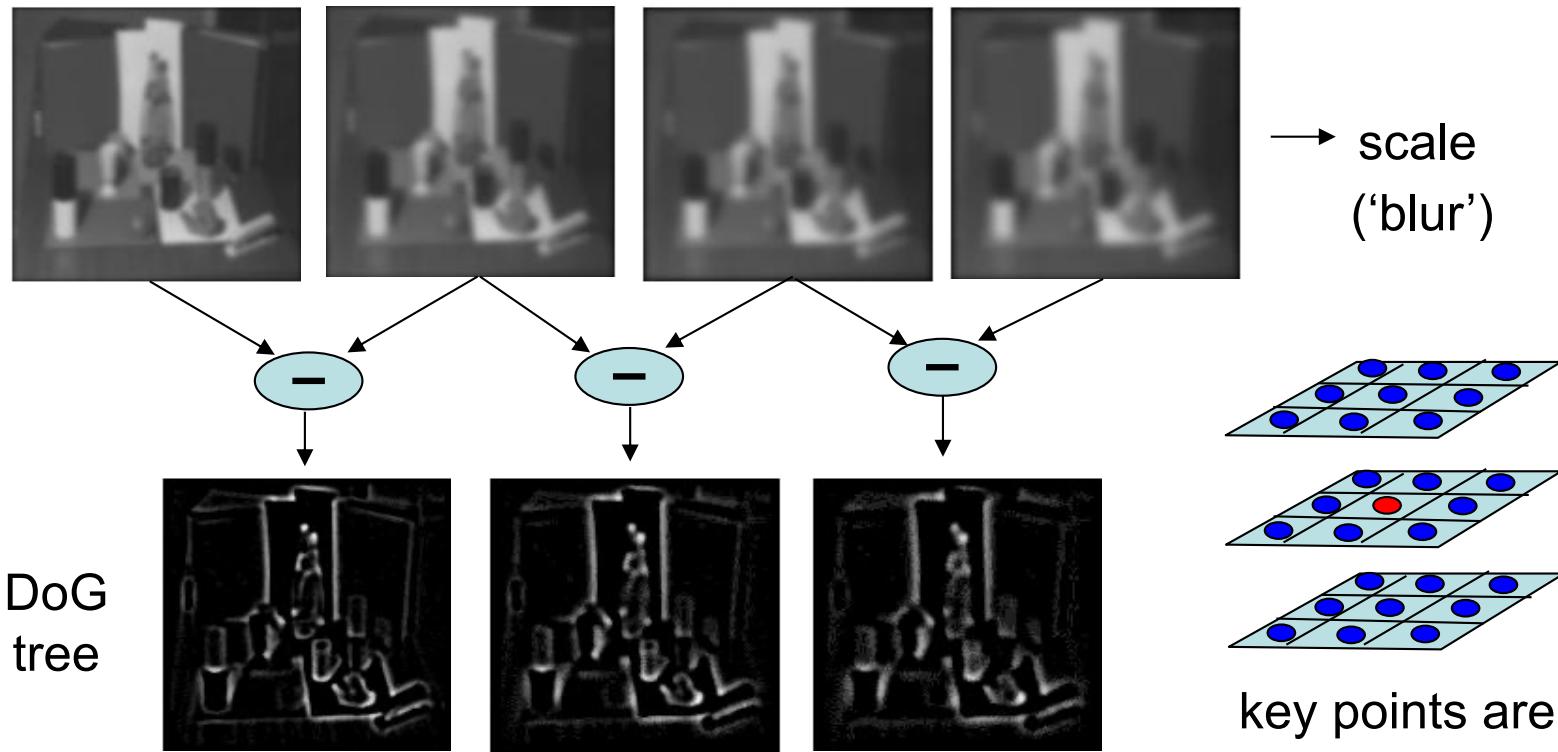
SIFT Matching

- **Two main elements:**
 - scale invariant detection of salient (key) points
 - matching by highly distinct local descriptors
- **Key point detection:**
 - extrema (max or min) in difference of Gaussian blurred versions of image → Difference-of-Gaussians (DoG) tree
 - points imaged at different resolutions appear at different levels of DoG tree → scale invariance
- **Spatial gradient descriptors:**
 - built from histograms of spatial gradients in local neighbourhood
 - invariant to rotation and perspective warp (almost)

Difference of Gaussians

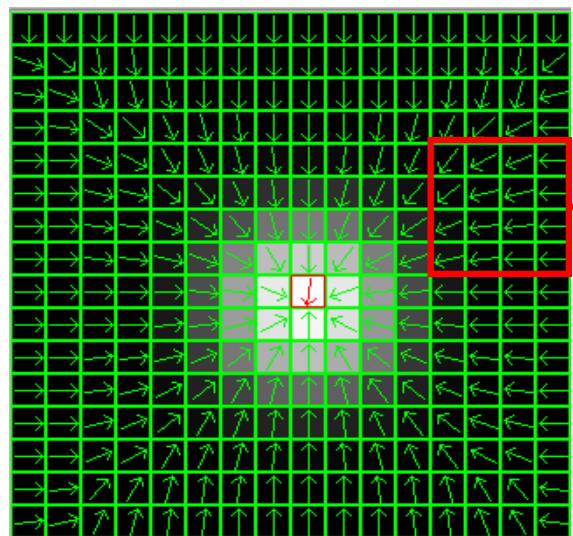
Most extrema present in both views possibly at different levels – **scale invariance**

Gaussian tree

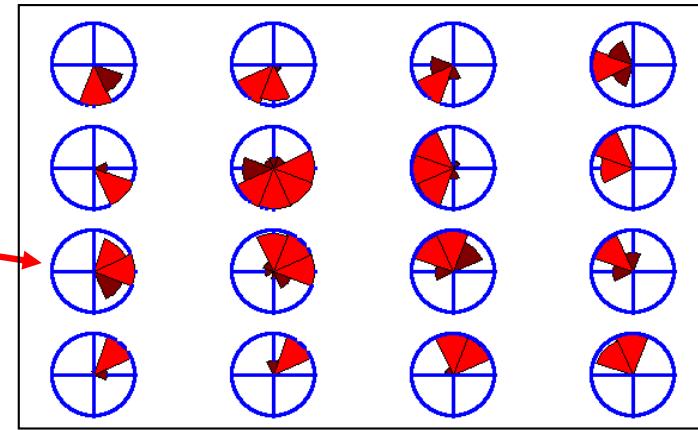


key points are
extrema in DoG tree

Spatial Gradient Descriptor



16x16 gradient field



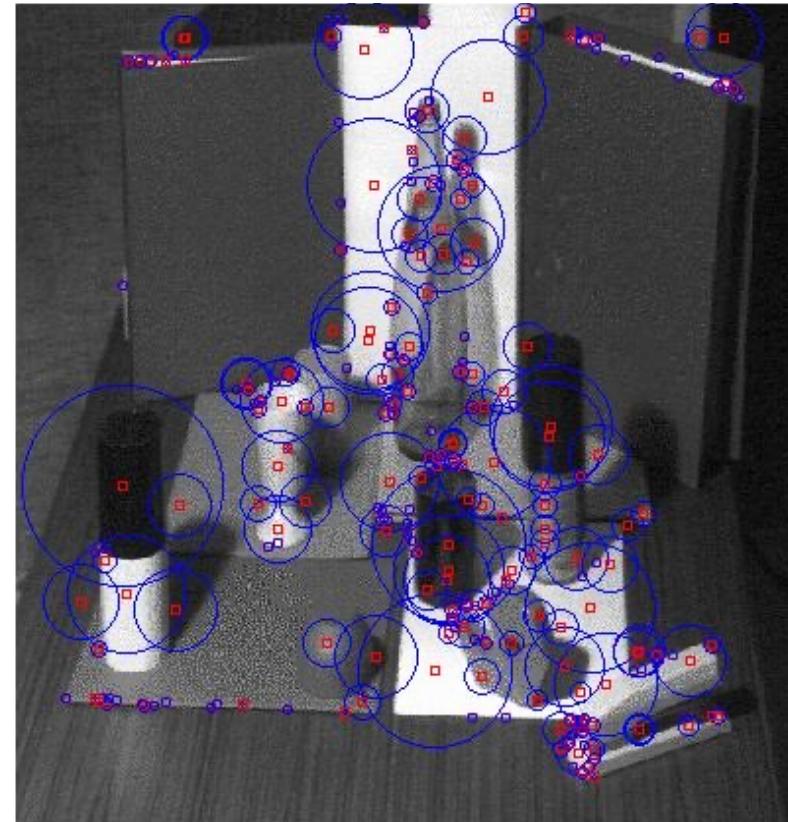
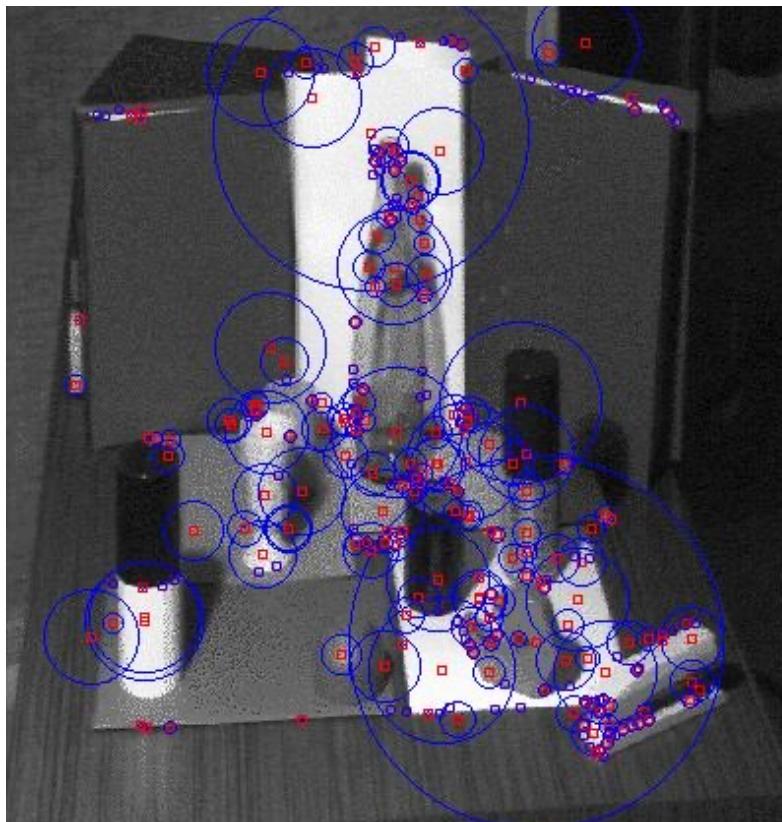
Orientation histograms (8 bins) over
4x4 blocks → 4x4 matrix of histograms,
normalised → 128-D feature vector.

Histograms built w.r.t dominant
orientation in region – **rotation
invariance** (almost)

Descriptor matching based on
Euclidean distance between 128-D
feature vectors

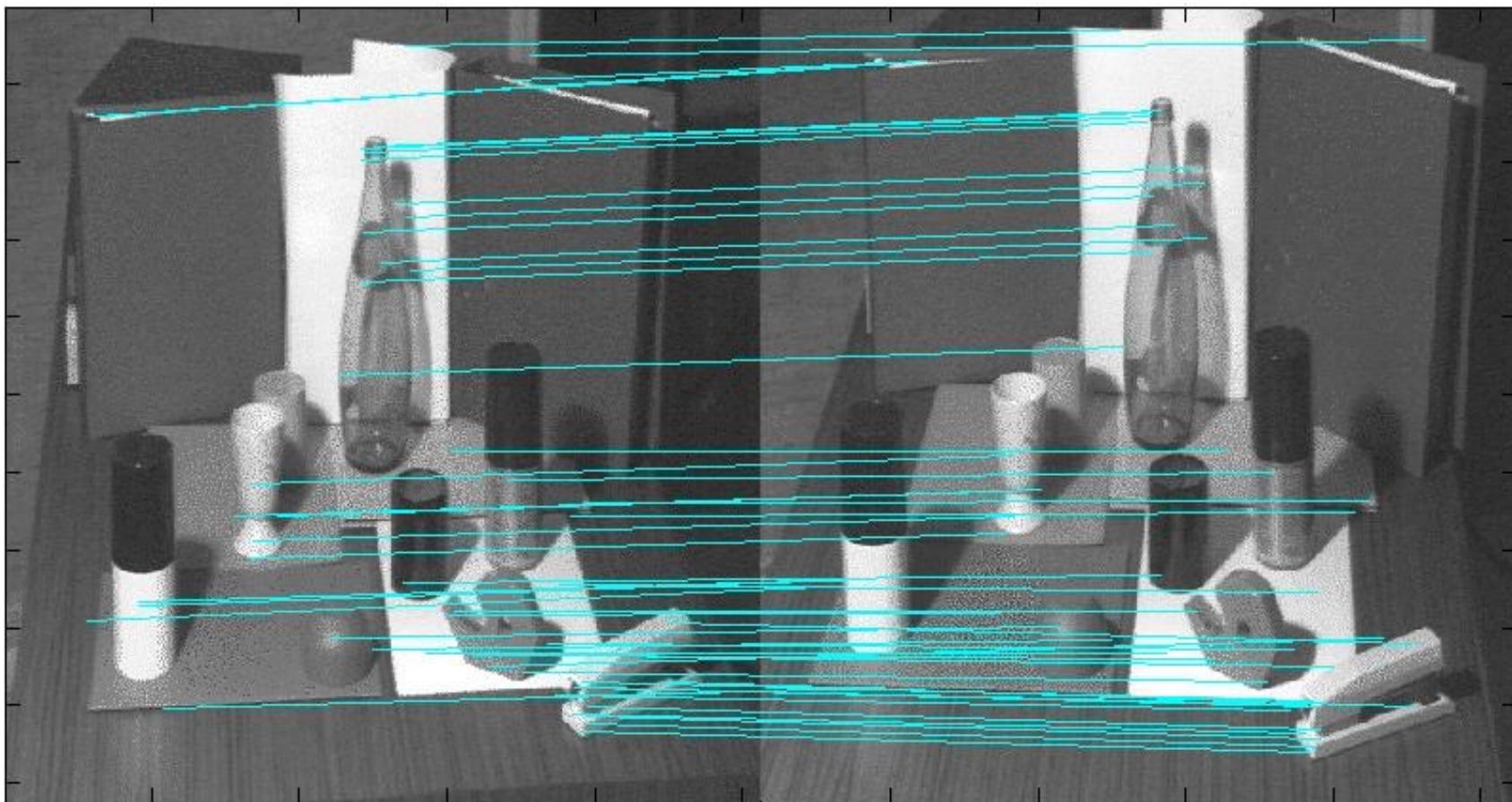
Distinctive Image Features from Scale-Invariant Keypoints,
David G. Lowe, International Journal of Computer Vision, 2004.

SIFT Example – Selected Key Points



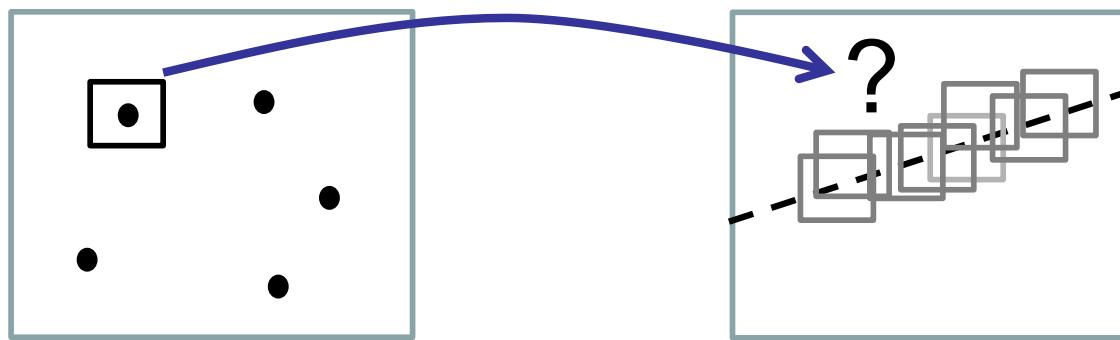
Each key point shown with circle indicating scale

SIFT Example – Matched Key Points



Where to look - Calibrated

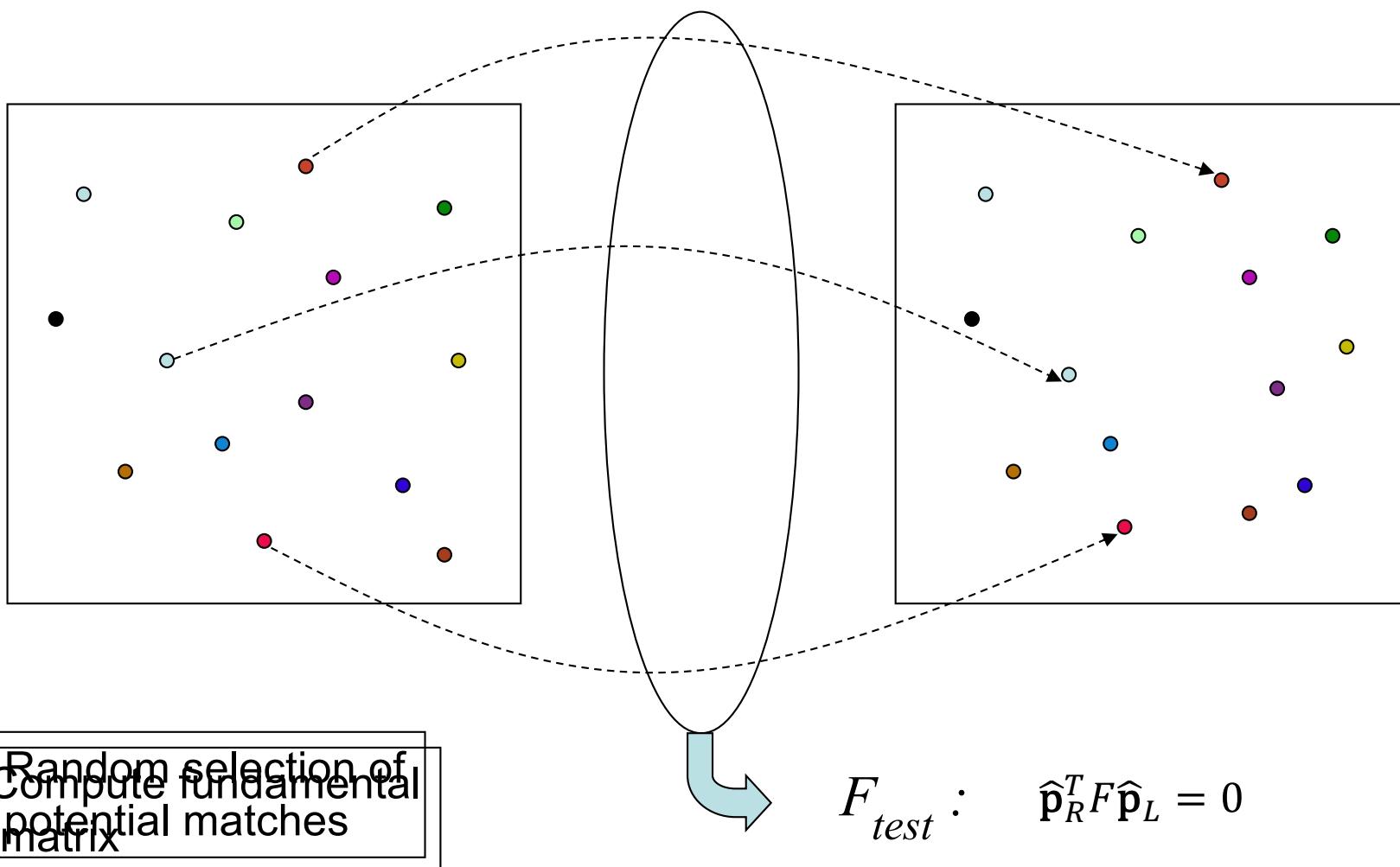
- For calibrated stereo set up, corresponding points lie on epipolar lines
- Hence, given point in left image, search for point in right image along band about epipolar line:
- Increases speed, reduces mismatches



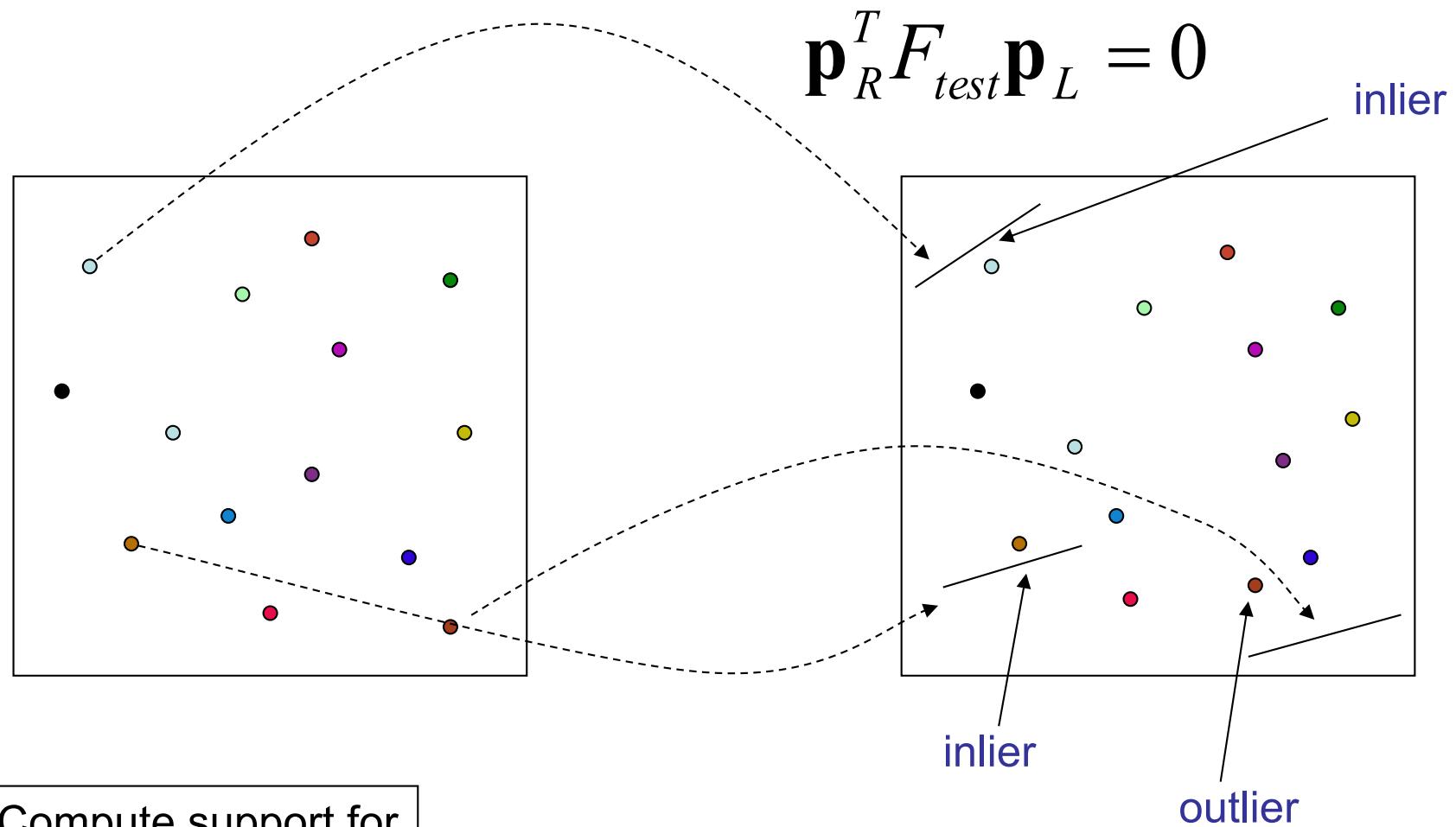
Where to look - Uncalibrated

- When geometry unknown, can only match points using pixel values – region or feature based approach.
- Often leads to mismatches amongst true matches, known as **outliers** and **inliers**.
- We know that inliers will be related by epipolar constraint equation $\hat{\mathbf{p}}_R^T F \hat{\mathbf{p}}_L = 0$
- We can use **RAN**dom **SA**mple **C**onsensus to sort out:
 - select subset of matches at random (minimum 4)
 - compute fundamental matrix F from subset (lecture 8, slide 7)
 - assess support for F amongst other correspondences
 - repeat until best F found

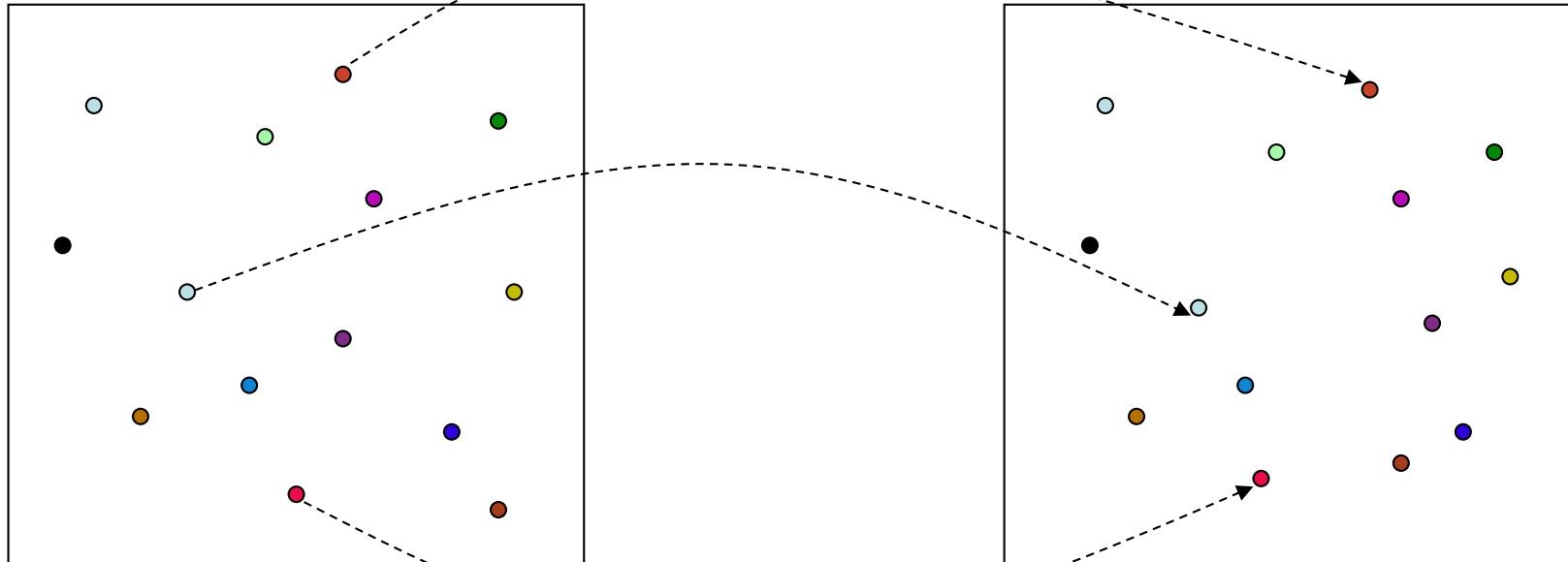
RANSAC Matching



RANSAC Matching



RANSAC Matching

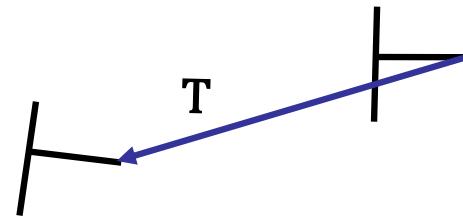
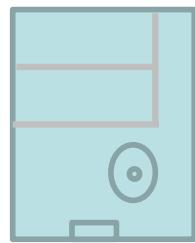


1. Random selection of potential matches

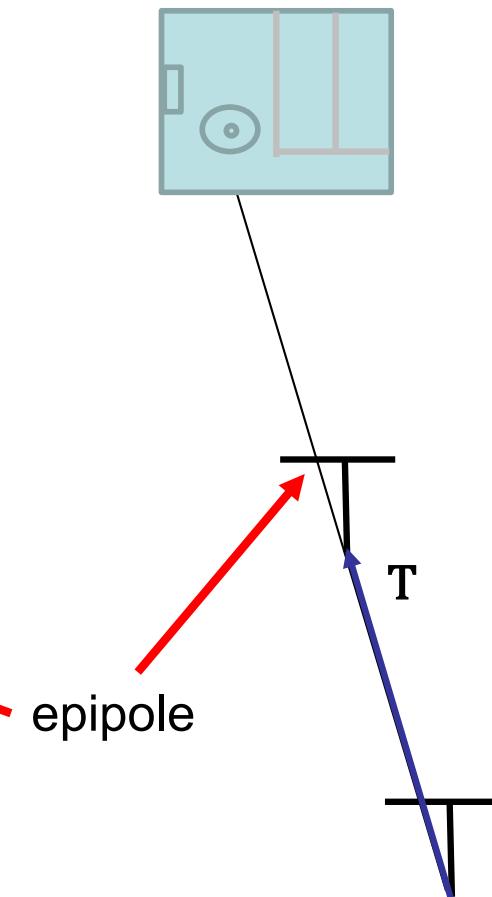
..... and so on

F with most support defines outliers and inliers → most likely correspondences

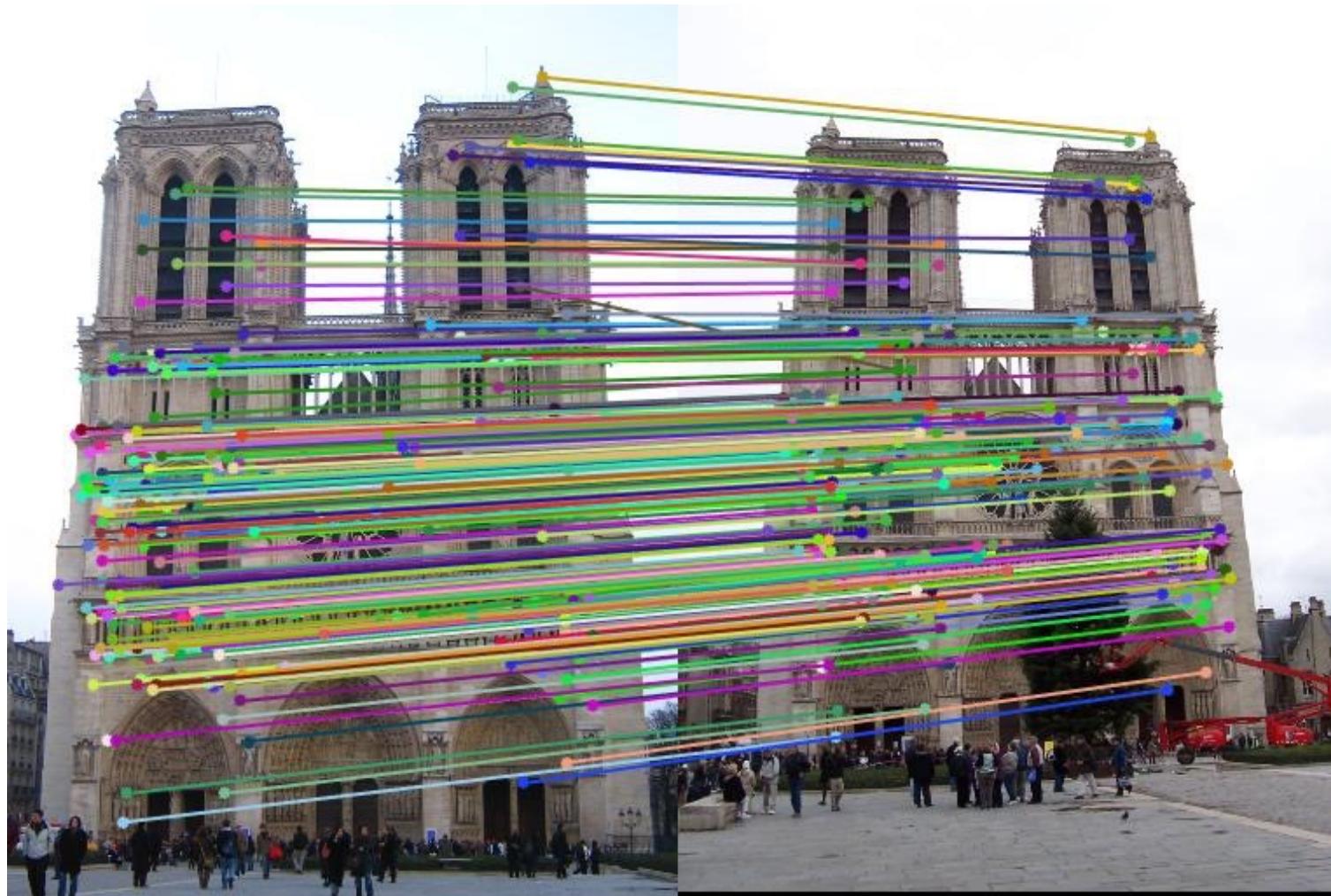
SIFT Matching – Keypoint Matches



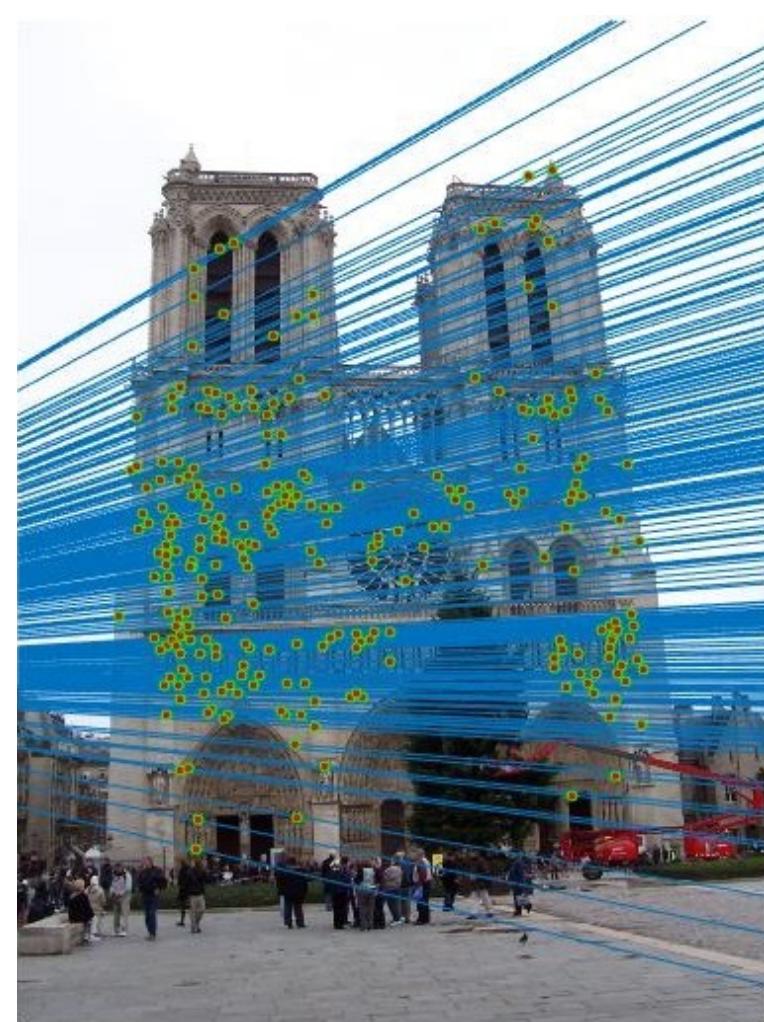
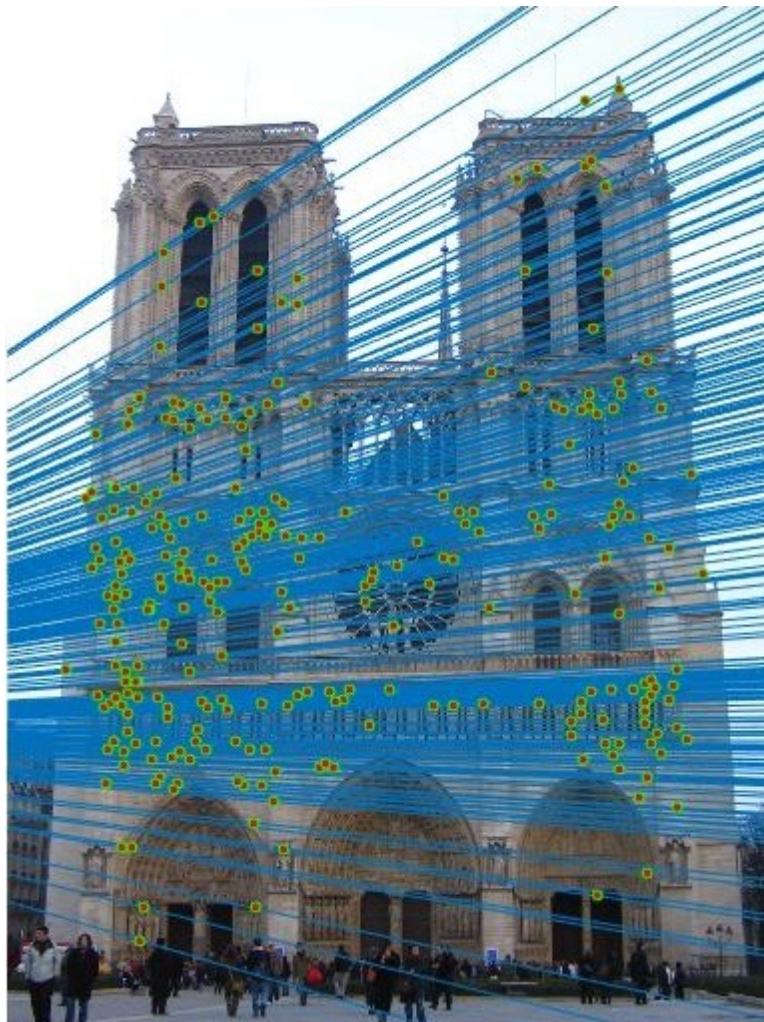
SIFT Matching – Epipolar Lines



SIFT Matching – Keypoint Matches



SIFT Matching – Epipolar Lines



COMS30030
Image Processing and Computer Vision

Stereo – Correspondence Matching

Andrew Calway
andrew@cs.bris.ac.uk

COMS30030 IPCV Stereo 4

1

1

Stereo Correspondence

COMS30030 IPCV Stereo 4

2

2

Stereo Matching is Hard

COMS30030 Stereo Lec 4

3

Region-Based Methods

- Compare pixel values within regions in two views.
- For region in left image, compute similarity with regions of same size in right image.
- Corresponding point - centre of most similar region

COMS30030 IPCV Stereo 4

4

4

Region Matching

- Stereo image pair: $I_L(x, y)$ and $I_R(x, y)$
- For each pixel, find disparity $\mathbf{d} = (d_x, d_y)$ which minimises (or maximises) cost function

$$c(\mathbf{d}) = \sum_{(i,j) \in W(x,y)} s[I_L(i,j), I_R(i + d_x, j + d_y)]$$

$W(x,y)$ \Rightarrow window of pixels around (x,y)

$I_L(x,y)$ $I_R(x,y)$

5

5

Similarity Measures

- Sum of squared differences:** $s(u, v) = (u - v)^2$
- Similar pixel count:**

$$s(u, v) = \begin{cases} 1 & \text{if } |u - v| < T \\ 0 & \text{else} \end{cases}$$

- Normalised cross correlation:**

$$s(u, v) = (u - \bar{u})(v - \bar{v}) / N_u N_v$$

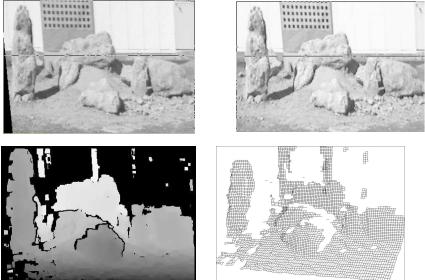
$$\bar{u} = \frac{1}{|W|} \sum_{u \in W} u \quad \text{mean} \quad N_u = \sqrt{\sum_{u \in W} (u - \bar{u})^2} \quad \text{Std deviation}$$

COMS30030 IPCV Stereo 4

6

6

Example



Real time correlation-based stereo, Faugeras et al, 1993

COMS30030 IPCV Stereo 4

7

Feature-Based Methods

- Restrict search to sparse set of features
 - reduces mis-matches caused by texture-less regions
- Find salient (distinct) points in each view and match points by comparing pixels or image descriptors in local regions about each point
 -
- Examples
 - Harris corner detector (salient points)
 - Scale-Invariant Feature Transform (SIFT)



COMS30030 IPCV Stereo 4

8

Harris Corner Detector

- Detects salient (distinct, interesting) image points
- Covariance of spatial gradient vectors within region W

$$A = \sum_{x,y \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad W \quad \begin{matrix} \nearrow & \searrow \\ \downarrow & \uparrow \\ \times & \times \end{matrix} \quad (I_x, I_y) \quad \text{spatial gradient}$$

$$I_x \equiv I_x(x, y)$$
- Eigenvalues λ_1 and λ_2 of A indicate 'spread' of gradients in region, e.g. 2 high values \rightarrow 'busy' region.
- Example saliency metric: $sal = \lambda_1 \lambda_2 / (\lambda_1 + \lambda_2)$
- Properties:
 - if eigenvalues both large $\rightarrow sal$ large
 - if either eigenvalue small $\rightarrow sal$ small

COMS30030 IPCV Stereo 4

9

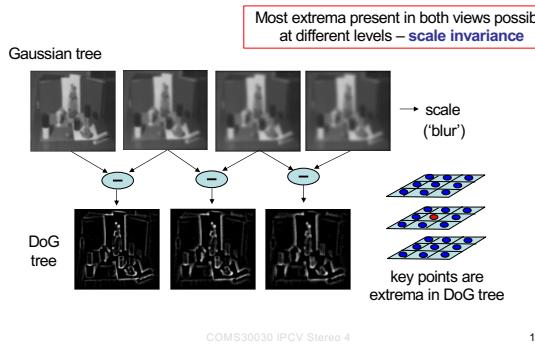
SIFT Matching

- **Two main elements:**
 - scale invariant detection of salient (key) points
 - matching by highly distinct local descriptors
- **Key point detection:**
 - extrema (max or min) in difference of Gaussian blurred versions of image \rightarrow Difference-of-Gaussians (DoG) tree
 - points imaged at different resolutions appear at different levels of DoG tree \rightarrow scale invariance
- **Spatial gradient descriptors:**
 - built from histograms of spatial gradients in local neighbourhood
 - invariant to rotation and perspective warp (almost)

COMS30030 IPCV Stereo 4

10

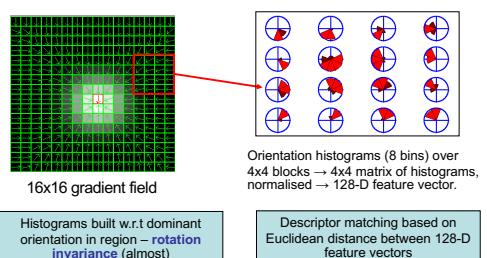
Difference of Gaussians



COMS30030 IPCV Stereo 4

11

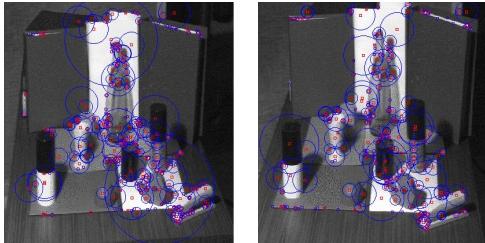
Spatial Gradient Descriptor



COMS30030 IPCV Stereo 4

12

SIFT Example – Selected Key Points

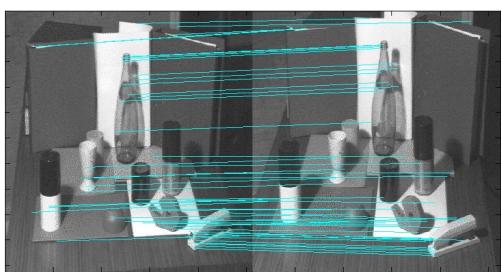


Each key point shown with circle indicating scale

COMS30030 IPCV Stereo 4

13

SIFT Example – Matched Key Points



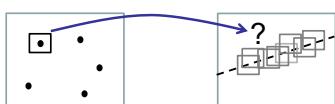
COMS30030 IPCV Stereo 4

14

13

Where to look - Calibrated

- For calibrated stereo set up, corresponding points lie on epipolar lines
- Hence, given point in left image, search for point in right image along band about epipolar line:
- Increases speed, reduces mismatches



COMS30030 IPCV Stereo 4

15

Where to look - Uncalibrated

- When geometry unknown, can only match points using pixel values – region or feature based approach.
- Often leads to mismatches amongst true matches, known as **outliers** and **inliers**.
- We know that inliers will be related by epipolar constraint equation $\hat{\mathbf{p}}_R^T F \hat{\mathbf{p}}_L = 0$
- We can use **RANSAC** to sort out:
 - select subset of matches at random (minimum 4)
 - compute fundamental matrix F from subset (lecture 8, slide 7)
 - assess support for F amongst other correspondences
 - repeat until best F found

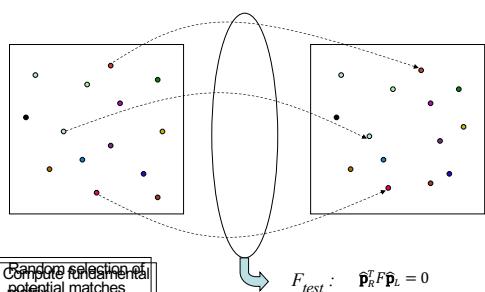
COMS30030 IPCV Stereo 4

16

15

16

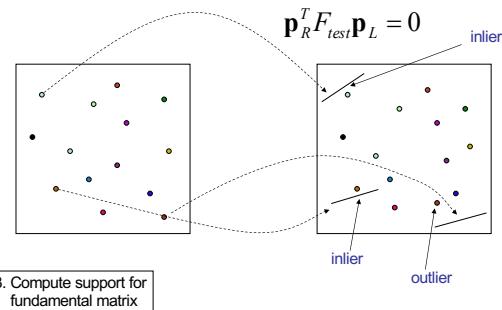
RANSAC Matching



COMS30030 IPCV Stereo 4

17

RANSAC Matching

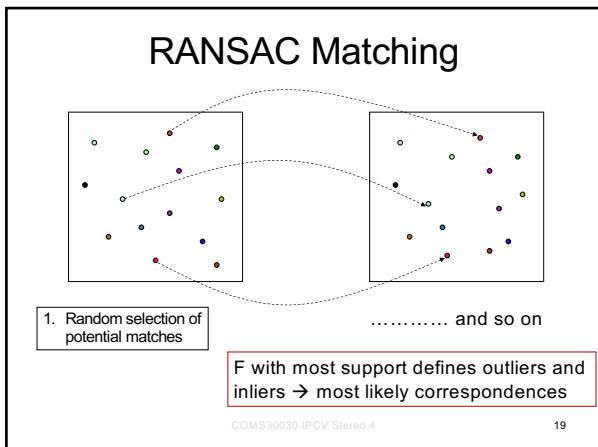


COMS30030 IPCV Stereo 4

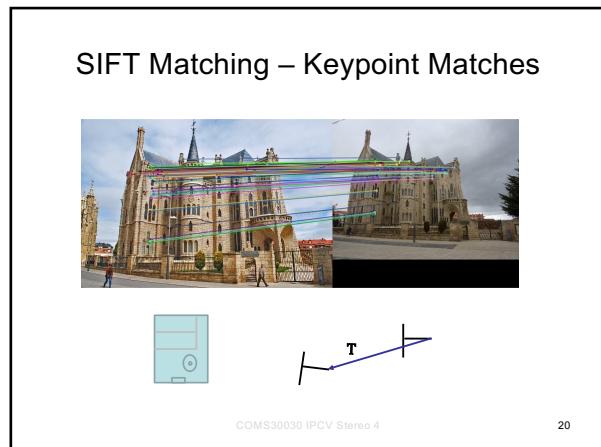
18

17

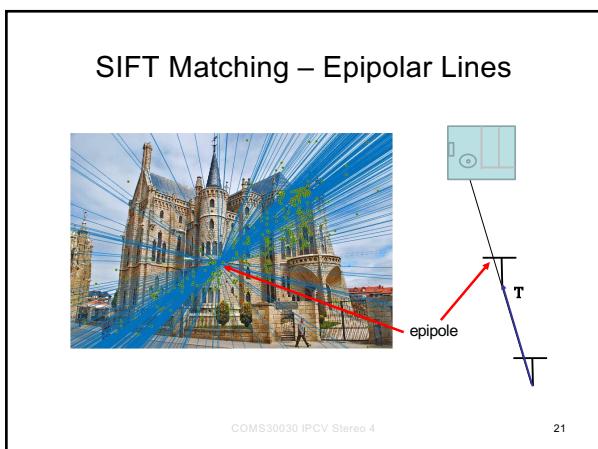
18



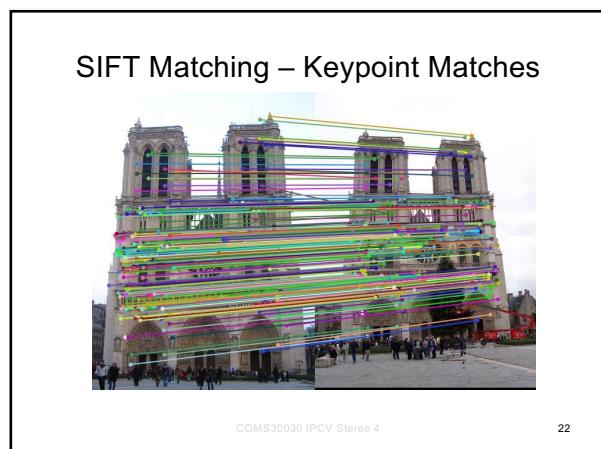
19



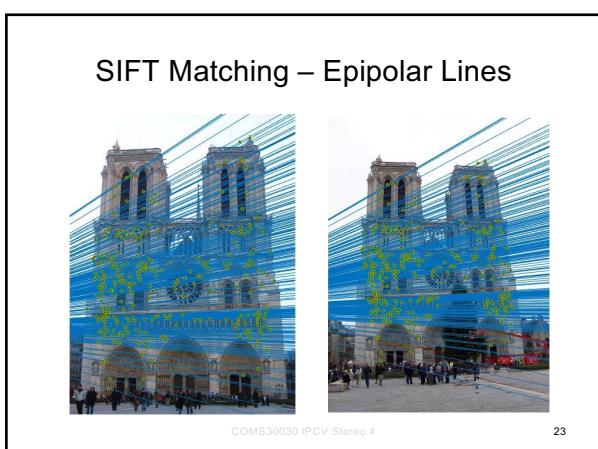
20



21



22



23

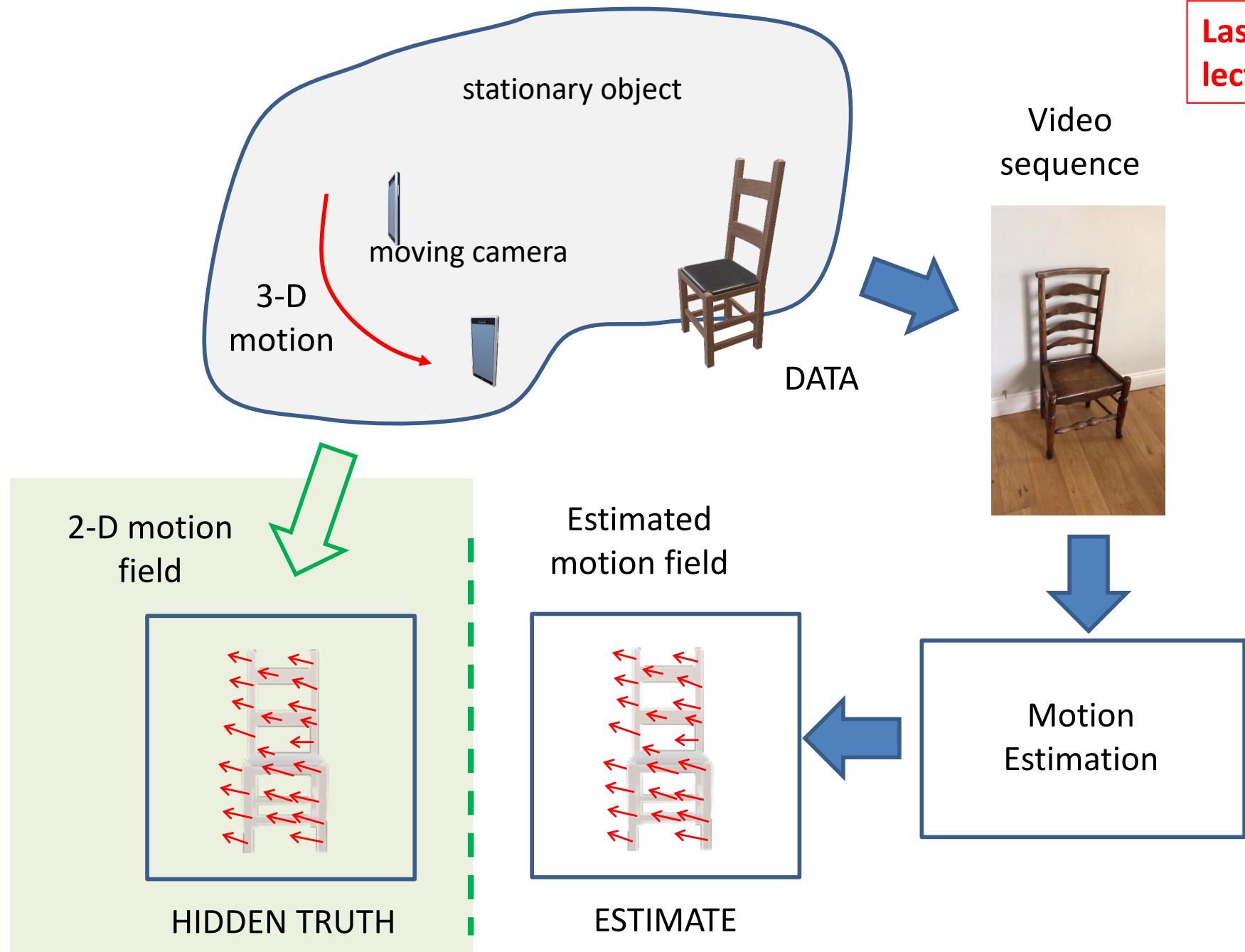
COMS30030
Image Processing and Computer Vision

Motion Estimation

Andrew Calway

andrew@cs.bris.ac.uk

Last
lecture



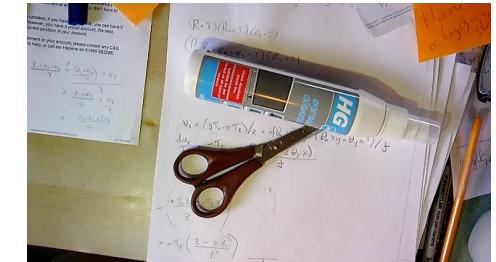
Motion Estimation

- The estimation of the 2-D motion field from frames in an image sequence
- Using spatial and temporal variation of pixel values
- **BUT**- relationship between variation in pixel values – known as **apparent motion** or **optical flow** – and the true motion is not straightforward.

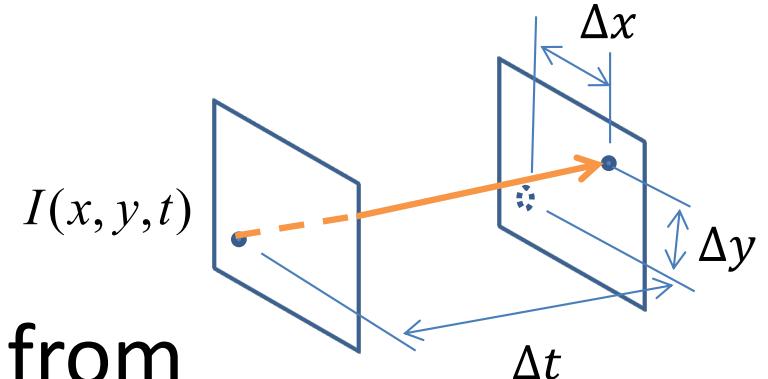


Apparent versus True Motion

- Apparent motion or **optical flow** - perceived motion in video sequence caused by changes in pixel values.
- Relationship with true 2-D motion field not always straightforward.
- Extreme cases:
 - non-zero apparent motion for zero motion field, e.g. static scene, moving light source
 - zero apparent motion for non-zero motion field, e.g. constant colour sphere rotating in diffuse lighting
- Sometimes not possible to determine 2-D motion field without additional constraints or assumptions.



Optical Flow



- Assume optical flow results from **brightness constancy constraint**
 - ‘*a moving pixel retains its value between frames*’
- For continuous video $I(x, y, t)$ (grey level)

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t)$$

HoT → 0 for tiny
 $\Delta x, \Delta y, \Delta t$

- Using Taylor’s expansion:

→ zero

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\delta I}{\delta x} \Delta x + \frac{\delta I}{\delta y} \Delta y + \frac{\delta I}{\delta t} \Delta t + \dots$$

Optical Flow Equation

- For $I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t)$

$$\frac{\delta I}{\delta x} \Delta x + \frac{\delta I}{\delta y} \Delta y + \frac{\delta I}{\delta t} \Delta t = 0$$

- Dividing throughout by Δt

$$\frac{\delta I}{\delta x} \frac{\Delta x}{\Delta t} + \frac{\delta I}{\delta y} \frac{\Delta y}{\Delta t} + \frac{\delta I}{\delta t} = 0$$

- For $\Delta x, \Delta y, \Delta t \rightarrow 0$

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

Optical Flow
Equation
(OFE)

Optical Flow Equation (OFE)

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

$\frac{dx}{dt}, \frac{dy}{dt}$ Rate of change of x, y with time
 ➡ optical flow field $\mathbf{u} = (u_x, u_y)$

$\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}, \frac{\delta I}{\delta t}$ Rate of change of I with x, y, t
 ➡ spatial & temporal gradients (I_x, I_y, I_t)

**Optical flow
equation**

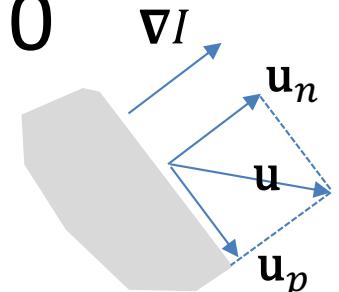
$$I_x u_x + I_y u_y + I_t = 0$$

Normal Flow

- OFE for $\mathbf{u} = (u_x, u_y)$ and $\nabla I = (I_x, I_y)$

$$I_x u_x + I_y u_y + I_t = 0 \Rightarrow \nabla I \cdot \mathbf{u} + I_t = 0$$

$$\nabla I \cdot \mathbf{u} = I_x u_x + I_y u_y \quad \text{dot product}$$



- OFE alone not sufficient to estimate motion
 - one equation in two unknowns
 - Only estimate **normal flow** \mathbf{u}_n
- $$\nabla I \cdot \mathbf{u} + I_t = \nabla I \cdot \mathbf{u}_n + I_t = 0$$
- ⇒ $\|\mathbf{u}_n\| = -I_t / \|\nabla I\| \quad \angle \mathbf{u}_n = \angle \nabla I$

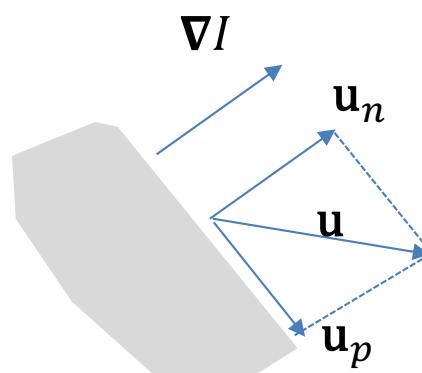
$$\begin{aligned}\mathbf{u} &= \mathbf{u}_p + \mathbf{u}_n \\ \mathbf{u}_p \cdot \mathbf{u}_n &= 0 \\ \nabla I \cdot \mathbf{u}_p &= 0\end{aligned}$$

Normal Flow

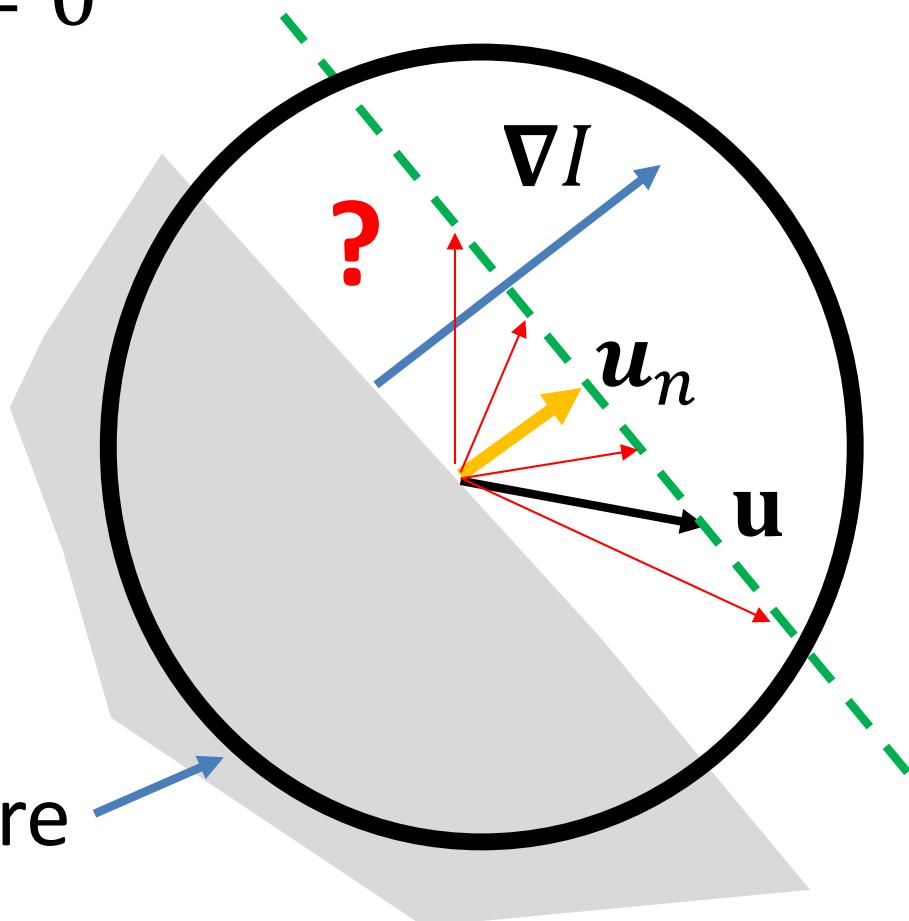
$$\nabla I \cdot \mathbf{u} + I_t = \nabla I \cdot \mathbf{u}_n + I_t = 0$$

$$\Rightarrow \|\mathbf{u}_n\| = -I_t / \|\nabla I\|$$

$$\angle \mathbf{u}_n = \angle \nabla I$$

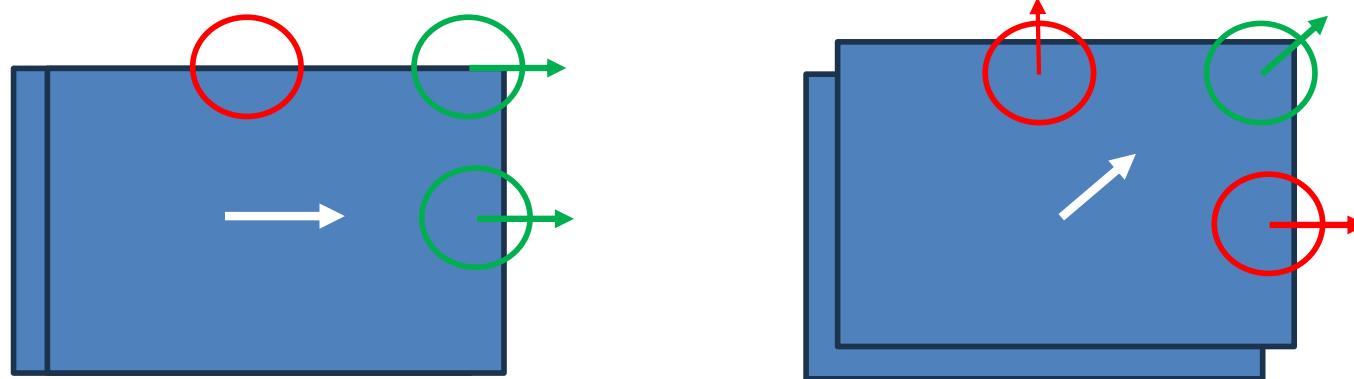


aperture



Aperture Problem

With single gradient direction in window (aperture), observed motion is different from true motion as we can only observe motion parallel to the gradient:



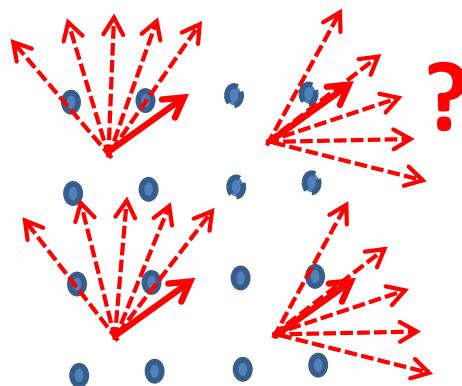
Hence: Good motion estimation depends on having sufficient variation in spatial gradient within regions.

Constraining the OFE

$$I_x u_x + I_y u_y + I_t = 0$$

OFE is under constrained – can only estimate normal flow

Need to add extra constraint(s)



Example : assume parametric form of motion field in regions

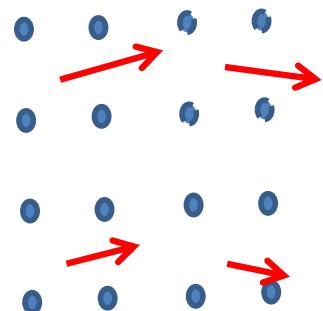
Example : constant velocity

Constraining the OFE

$$I_x u_x + I_y u_y + I_t = 0$$

OFE is under constrained – can only estimate normal flow

Need to add extra constraint(s)



Example : assume parametric form of motion field in regions

Example : linear in x and y , e.g.

$$u_x = ax + by + c$$
$$u_y = dx + ey + f$$

Constant Velocity Model

For a region, find the velocity $\mathbf{u} = (u_x, u_y)$ which minimises :

$$\varepsilon(u_x, u_y) = \sum_{region} (I_x u_x + I_y u_y + I_t)^2$$

Solution: take derivatives w.r.t u_x and u_y , set to zero, and solve for u_x and u_y .

OFE $\rightarrow 0$

NB: same $\mathbf{u} = (u_x, u_y)$ over whole region \rightarrow solution

Lucas and Kanade Algorithm

Find velocity $\mathbf{u} = (u_x, u_y)$ which minimises :

$$\varepsilon(u_x, u_y) = \sum_R (I_x u_x + I_y u_y + I_t)^2$$

Partial derivatives w.r.t u_x and u_y , set to zero, solve for u_x and u_y :

$$\frac{\partial \varepsilon}{\partial u_x} = 2 \sum_R (I_x u_x + I_y u_y + I_t) I_x = 0 \quad \Rightarrow \quad \sum_R (I_x^2 u_x + I_x I_y u_y + I_x I_t) = 0$$

$$\frac{\partial \varepsilon}{\partial u_y} = 2 \sum_R (I_x u_x + I_y u_y + I_t) I_y = 0 \quad \Rightarrow \quad \sum_R (I_x I_y u_x + I_y^2 u_y + I_y I_t) = 0$$

Lucas and Kanade Algorithm

Hence, solve for $\mathbf{u} = (u_x, u_y)$ given that :

$$u_x \sum_R I_x^2 + u_y \sum_R I_x I_y = - \sum_R I_t I_x \quad \Rightarrow \quad A\mathbf{u} = \mathbf{b}$$

$$u_x \sum_R I_x I_y + u_y \sum_R I_y^2 = - \sum_R I_t I_y$$

$$\Rightarrow \boxed{\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = A^{-1}\mathbf{b}}$$

$$A = \sum_R \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad \mathbf{b} = - \sum_R \begin{bmatrix} -I_t I_x \\ -I_t I_y \end{bmatrix}$$

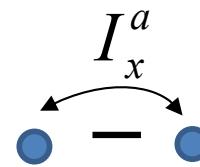
Spatial & Temporal Gradients

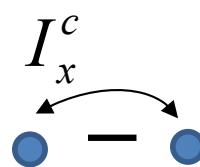
Approximate gradients using differences, e.g.

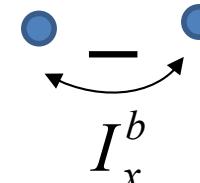
$$I_x = \delta I / \delta x \approx I(x + 1, y, t) - I(x, y, t)$$

i.e. assume $\delta x = 1$

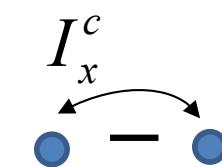
Or use averaging to reduce noise, e.g.

$$I_x^a$$


$$I_x^c$$


$$I_x^b$$


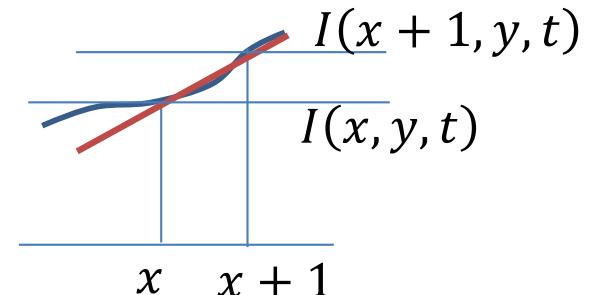
Frame t

$$I_x^d$$


Frame $t + 1$

$$I_x \approx (I_x^a + I_x^b + I_x^c + I_x^d) / 4$$

Rate of change
of I with x



L & K Algorithm

$I^1 = \text{video frame at time } t$

$I^2 = \text{video frame at time } t + 1$

For each pixel x, y in I^1

$A = 0 ; \mathbf{b} = 0 ;$

For each pixel in region Λ about x, y

$$(I_x, I_y, I_t) = \text{CompGrads}(I^1, I^2) ;$$

$$A' = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} ;$$

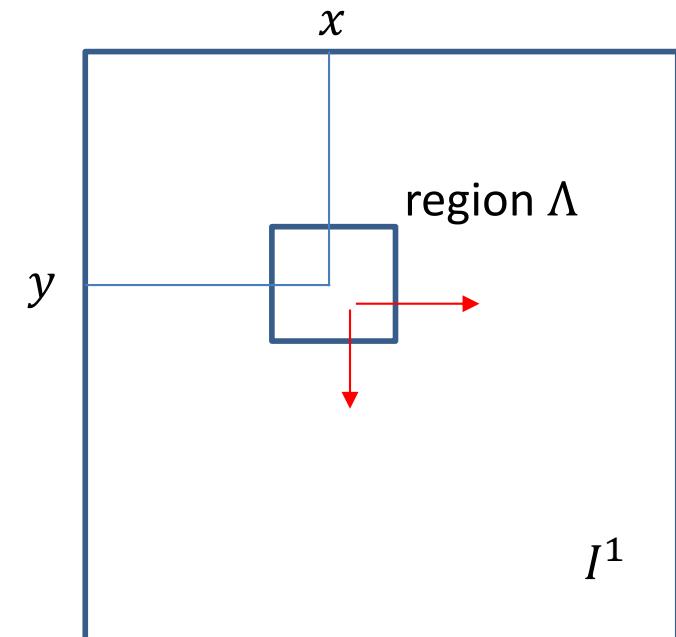
$$\mathbf{b}' = \begin{bmatrix} -I_t I_x \\ -I_t I_y \end{bmatrix} ;$$

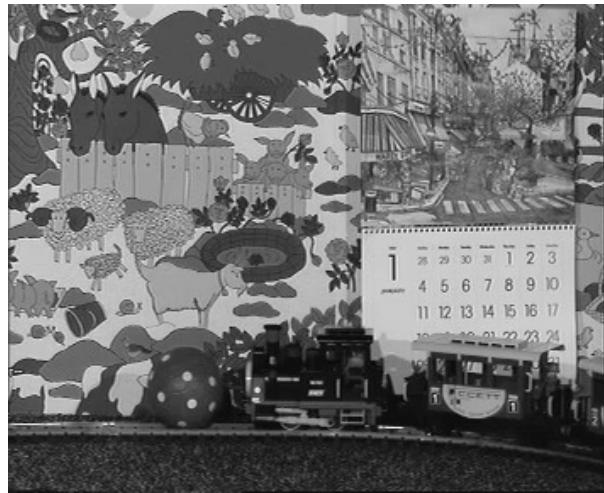
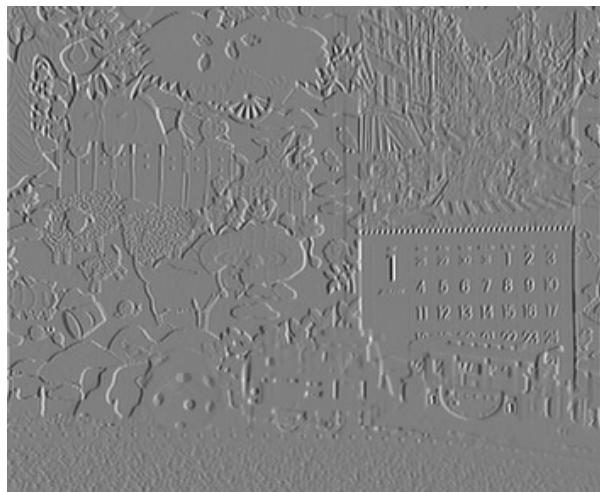
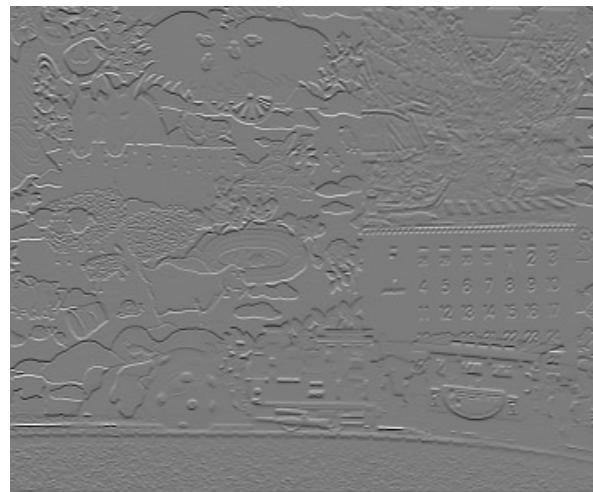
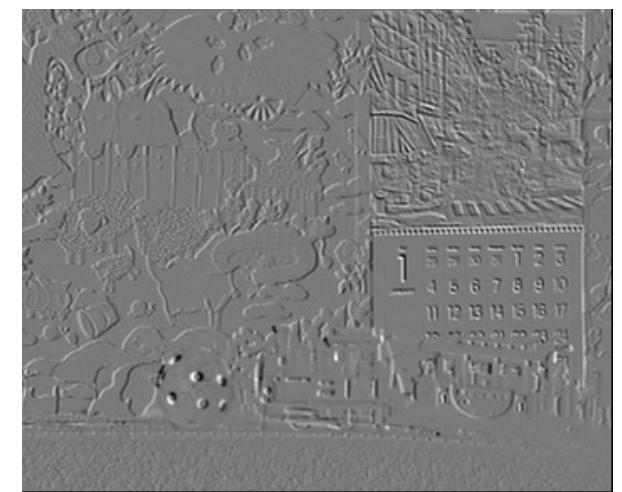
$$A \rightarrow A + A' ; \quad \mathbf{b} \rightarrow \mathbf{b} + \mathbf{b}' ;$$

End;

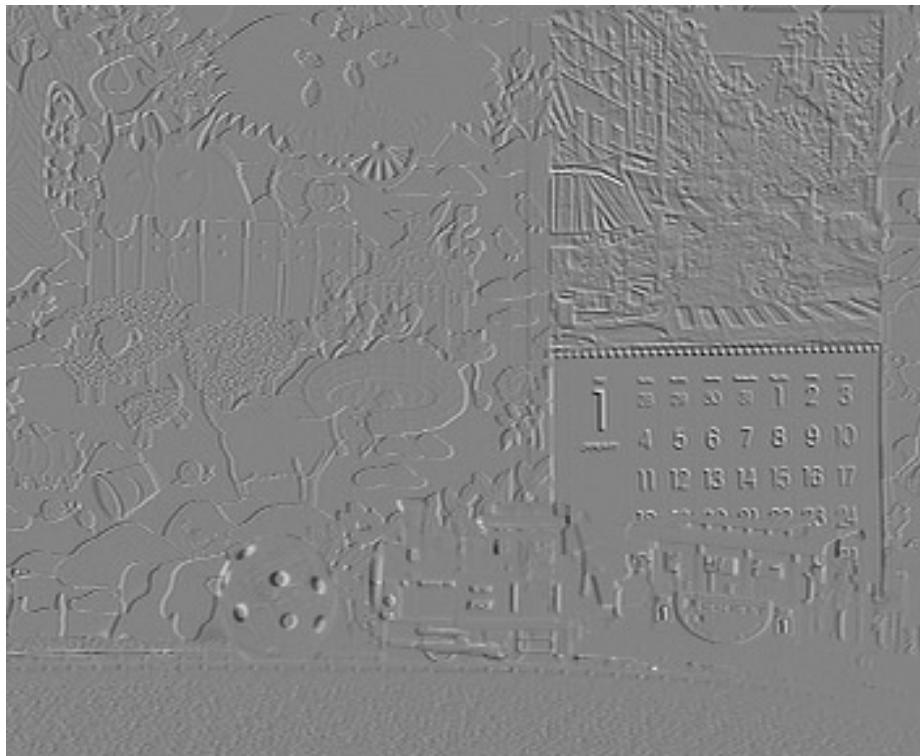
$$\mathbf{u}(x, y) = A^{-1} \mathbf{b}$$

End;

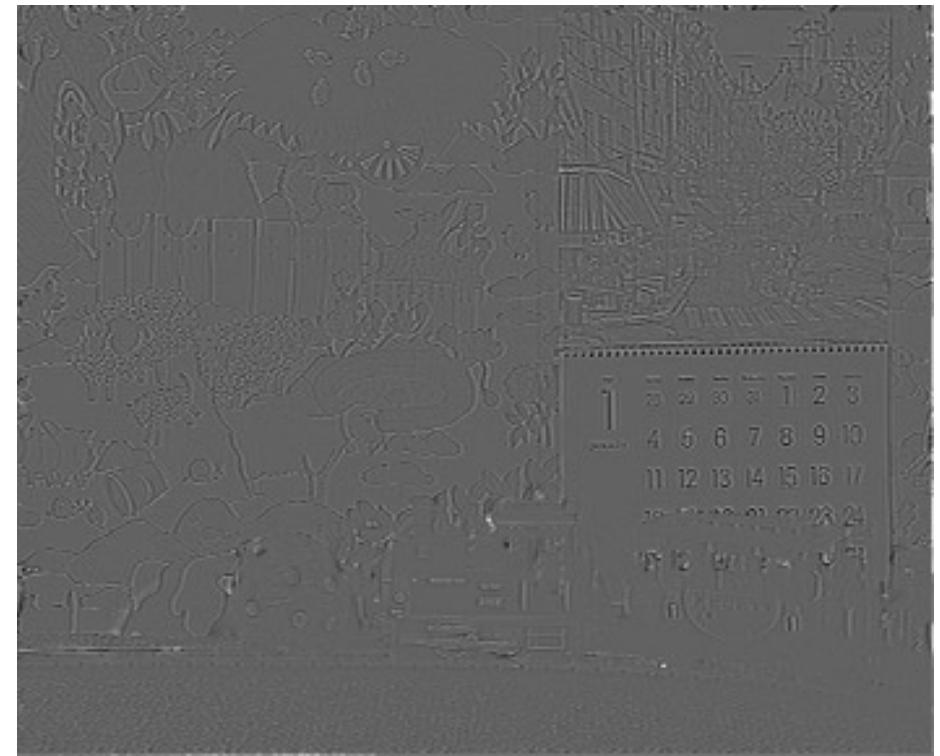


 $I(x, y, t)$  u_x  u_y  I_x  I_y  I_t

Frame Difference

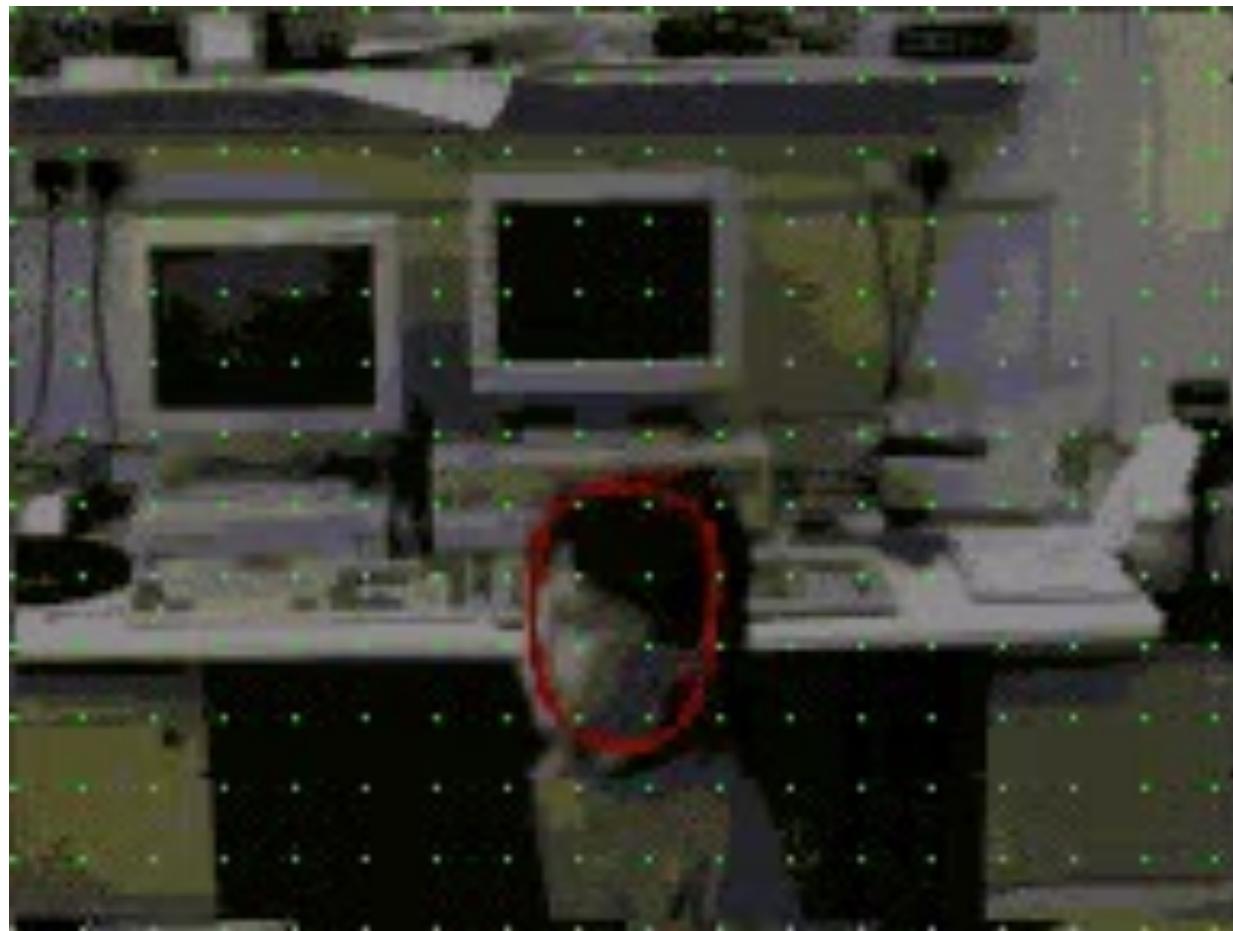


frame difference

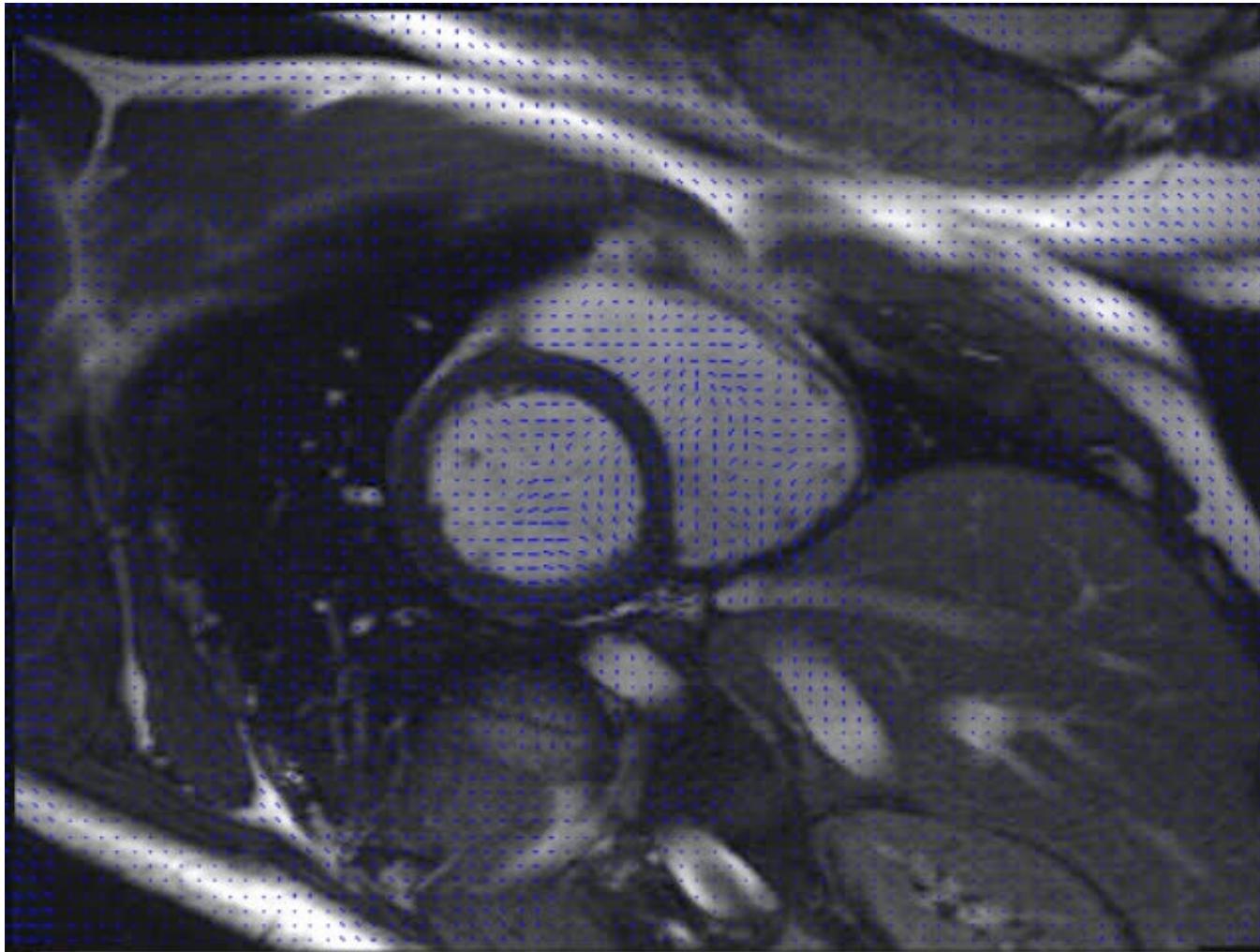


motion compensated frame difference

Motion Estimation - Example

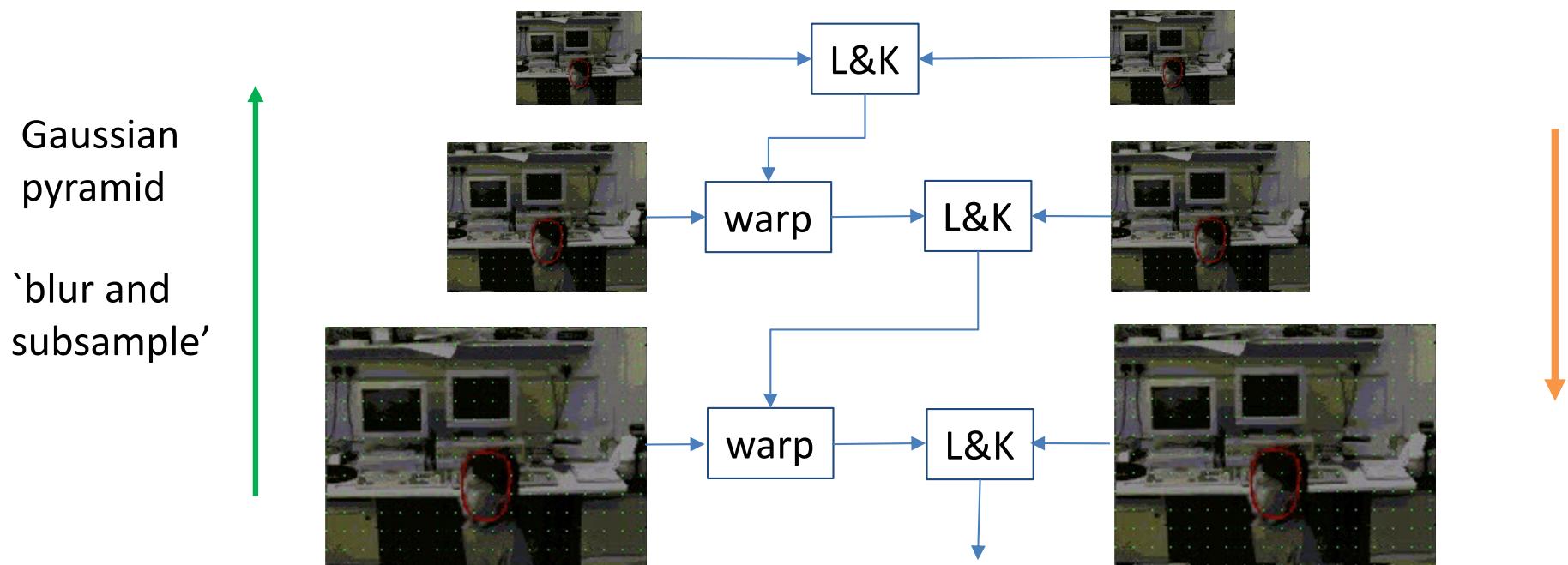


Motion Estimation - Example

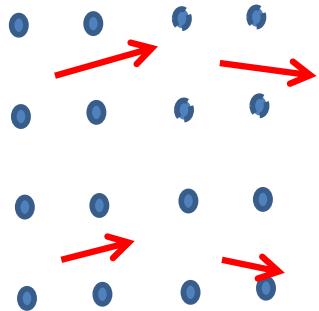


Multiresolution L & K

- To deal with large motions, implement L&K over multiple resolutions – result at lower resolutions used to ‘warp’ higher resolution images prior to estimation.



Affine Motion Model



$$\begin{aligned}u_x &= ax + by + c \\u_y &= dx + ey + f\end{aligned}$$

$$\mathbf{u} = A \mathbf{p}$$

$$\mathbf{p}^T = (a, b, c, d, e, f)$$

Models translation, scaling, rotation and shear

$$\Rightarrow \mathbf{p}^T A^T \nabla I + I_t = 0 \quad \text{affine OFE} \quad \Rightarrow \hat{\mathbf{p}} = M^{-1} \mathbf{b}$$

$$M = \sum_{region} A^T \nabla I \nabla I^T A \quad b = - \sum_{region} I_t (A^T \nabla I)$$

Horn-Schunk Algorithm

- Alternative to L&K which seeks to find optimal motion field with smooth variation in motion vectors
- Algorithm aims to find the motion field $\nu = (\nu_x, \nu_y)$ which minimises following energy functional

$$E = \iint \left[(I_x u_x + I_y u_y + I_t)^2 + \sigma^2 (\|\nabla u_x\|^2 + \|\nabla u_y\|^2) \right] dx dy$$

Diagram illustrating the Horn-Schunk algorithm's energy functional:

- The energy functional E consists of two main terms:
 - The first term is $(I_x u_x + I_y u_y + I_t)^2$, highlighted by a red bracket.
 - The second term is $\sigma^2 (\|\nabla u_x\|^2 + \|\nabla u_y\|^2)$, highlighted by a red bracket.
- A red arrow points from the first term to a box labeled "OFE $\rightarrow 0$ ".
- A blue arrow points from the second term to a box labeled "weighting factor".
- A blue arrow points from the second term to the text "Rate of change of $\nu \rightarrow 0$ ".
- A blue arrow points from the text "Rate of change of $\nu \rightarrow 0$ " to the text "smooth motion field".

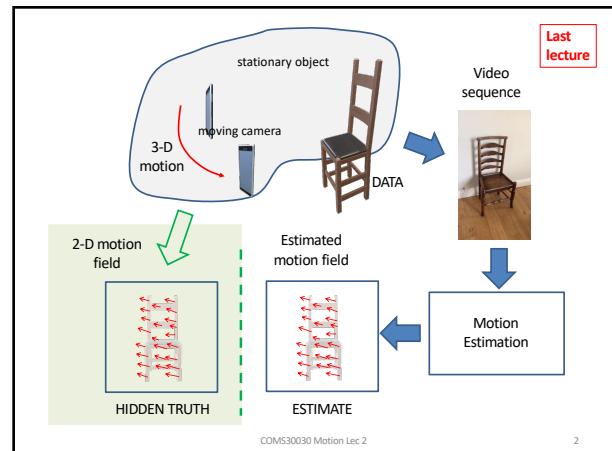
COMS30030
Image Processing and Computer Vision

Motion Estimation

Andrew Calway
andrew@cs.bris.ac.uk

COMS30030 Motion Lec 2

1



2

Motion Estimation

- The estimation of the 2-D motion field from frames in an image sequence
- Using spatial and temporal variation of pixel values
- BUT**- relationship between variation in pixel values – known as **apparent motion** or **optical flow** – and the true motion is not straightforward.

COMS30030 Motion Lec 2

3

Apparent versus True Motion

- Apparent motion or **optical flow** - perceived motion in video sequence caused by changes in pixel values.
- Relationship with true 2-D motion field not always straightforward.
- Extreme cases:
 - non-zero apparent motion for zero motion field, e.g. static scene, moving light source
 - zero apparent motion for non-zero motion field, e.g. constant colour sphere rotating in diffuse lighting
- Sometimes not possible to determine 2-D motion field without additional constraints or assumptions.

COMS30030 Motion Lec 2

4

Optical Flow

- Assume optical flow results from **brightness constancy constraint**
– *a moving pixel retains its value between frames*
- For continuous video $I(x, y, t)$ (grey level)
 $I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t)$
- Using Taylor's expansion:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\delta I}{\delta x} \Delta x + \frac{\delta I}{\delta y} \Delta y + \frac{\delta I}{\delta t} \Delta t + \dots$$

HoT → 0 for tiny $\Delta x, \Delta y, \Delta t$

COMS30030 Motion Lec 2

5

Optical Flow Equation

- For $I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t)$

$$\frac{\delta I}{\delta x} \Delta x + \frac{\delta I}{\delta y} \Delta y + \frac{\delta I}{\delta t} \Delta t = 0$$
- Dividing throughout by Δt

$$\frac{\delta I}{\delta x} \frac{\Delta x}{\Delta t} + \frac{\delta I}{\delta y} \frac{\Delta y}{\Delta t} + \frac{\delta I}{\delta t} = 0$$
- For $\Delta x, \Delta y, \Delta t \rightarrow 0$

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

Optical Flow Equation (OFE)

COMS30030 Motion Lec 2

6

Optical Flow Equation (OFE)

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

$\frac{dx}{dt}, \frac{dy}{dt}$ Rate of change of x, y with time
 \Rightarrow optical flow field $\mathbf{u} = (u_x, u_y)$

$\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}, \frac{\delta I}{\delta t}$ Rate of change of I with x, y, t
 \Rightarrow spatial & temporal gradients (I_x, I_y, I_t)

Optical flow equation

$$I_x u_x + I_y u_y + I_t = 0$$

COMS30030 Motion Lec 2

7

Normal Flow

- OFE for $\mathbf{u} = (u_x, u_y)$ and $\nabla I = (I_x, I_y)$

$$I_x u_x + I_y u_y + I_t = 0 \Rightarrow \nabla I \cdot \mathbf{u} + I_t = 0$$

$\nabla I \cdot \mathbf{u} = I_x u_x + I_y u_y$ dot product



- OFE alone not sufficient to estimate motion

– one equation in two unknowns

- Only estimate normal flow \mathbf{u}_n

$$\nabla I \cdot \mathbf{u} + I_t = \nabla I \cdot \mathbf{u}_n + I_t = 0$$

$$\Rightarrow \|\mathbf{u}_n\| = -I_t / \|\nabla I\| \quad \angle \mathbf{u}_n = \angle \nabla I$$

COMS30030 Motion Lec 2

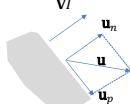
8

Normal Flow

$$\nabla I \cdot \mathbf{u} + I_t = \nabla I \cdot \mathbf{u}_n + I_t = 0$$

$$\Rightarrow \|\mathbf{u}_n\| = -I_t / \|\nabla I\|$$

$$\angle \mathbf{u}_n = \angle \nabla I$$



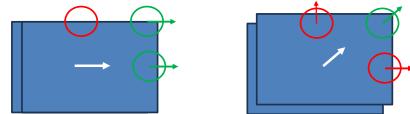
aperture

COMS30030 Motion Lec 2

9

Aperture Problem

With single gradient direction in window (aperture), observed motion is different from true motion as we can only observe motion parallel to the gradient:



Hence: Good motion estimation depends on having sufficient variation in spatial gradient within regions.

COMS30030 Motion Lec 2

10

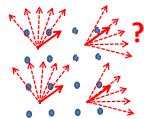
Constraining the OFE

$$I_x u_x + I_y u_y + I_t = 0$$

OFE is under constrained – can only estimate normal flow

Need to add extra constraint(s)

Example : assume parametric form of motion field in regions



Example : constant velocity

COMS30030 Motion Lec 2

11

Constraining the OFE

$$I_x u_x + I_y u_y + I_t = 0$$

OFE is under constrained – can only estimate normal flow

Need to add extra constraint(s)

Example : assume parametric form of motion field in regions



Example : linear in x and y , e.g.

$$\begin{aligned} u_x &= ax + by + c \\ u_y &= dx + ey + f \end{aligned}$$

COMS30030 Motion Lec 2

12

11

Constant Velocity Model

For a region, find the velocity $\mathbf{u} = (u_x, u_y)$ which minimises :

$$\varepsilon(u_x, u_y) = \sum_{\text{region}} (I_x u_x + I_y u_y + I_t)^2$$

Solution: take derivatives w.r.t u_x and u_y , set to zero, and solve for u_x and u_y .

OFE → 0

NB: same $\mathbf{u} = (u_x, u_y)$ over whole region → solution

COMS30030 Motion Lec 2

13

Lucas and Kanade Algorithm

Find velocity $\mathbf{u} = (u_x, u_y)$ which minimises :

$$\varepsilon(u_x, u_y) = \sum_R (I_x u_x + I_y u_y + I_t)^2$$

Partial derivatives w.r.t u_x and u_y , set to zero, solve for u_x and u_y :

$$\frac{\partial \varepsilon}{\partial u_x} = 2 \sum_R (I_x u_x + I_y u_y + I_t) I_x = 0 \Rightarrow \sum_R (I_x^2 u_x + I_x I_y u_y + I_x I_t) = 0$$

$$\frac{\partial \varepsilon}{\partial u_y} = 2 \sum_R (I_x u_x + I_y u_y + I_t) I_y = 0 \Rightarrow \sum_R (I_x I_y u_x + I_y^2 u_y + I_y I_t) = 0$$

COMS30030 Motion Lec 2

14

Lucas and Kanade Algorithm

Hence, solve for $\mathbf{u} = (u_x, u_y)$ given that :

$$\begin{aligned} u_x \sum_R I_x^2 + u_y \sum_R I_x I_y &= - \sum_R I_t I_x \quad \Rightarrow \quad A\mathbf{u} = \mathbf{b} \\ u_x \sum_R I_x I_y + u_y \sum_R I_y^2 &= - \sum_R I_t I_y \end{aligned}$$

$$\Rightarrow \mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = A^{-1} \mathbf{b}$$

$$A = \sum_R \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad \mathbf{b} = - \sum_R \begin{bmatrix} -I_t I_x \\ -I_t I_y \end{bmatrix}$$

COMS30030 Motion Lec 2

15

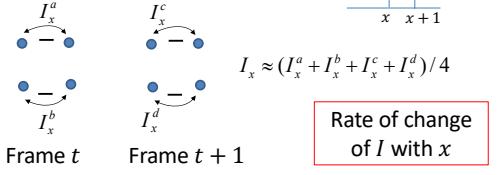
Spatial & Temporal Gradients

Approximate gradients using differences, e.g.

$$I_x = \delta I / \delta x \approx I(x+1, y, t) - I(x, y, t)$$

i.e. assume $\delta x = 1$

Or use averaging to reduce noise, e.g.



Rate of change of I with x

COMS30030 Motion Lec 2

16

L & K Algorithm

I^1 = video frame at time t

I^2 = video frame at time $t+1$

For each pixel x, y in I^1

$$A = 0; \mathbf{b} = 0;$$

For each pixel in region Λ about x, y

$$(I_x, I_y, I_t) = \text{CompGrads}(I^1, I^2);$$

$$A' = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix};$$

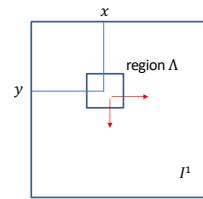
$$\mathbf{b}' = \begin{bmatrix} -I_t I_x \\ -I_t I_y \end{bmatrix};$$

$$A \rightarrow A + A'; \quad \mathbf{b} \rightarrow \mathbf{b} + \mathbf{b}';$$

End;

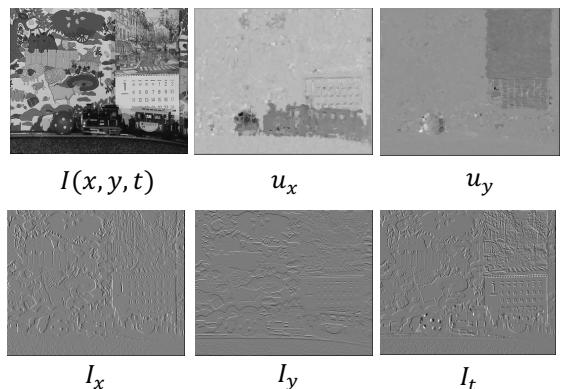
$$\mathbf{u}(x, y) = A^{-1} \mathbf{b}$$

End;



COMS30030 Motion Lec 2

17



COMS30030 Motion Lec 2

18

COMS30030 - Image Processing and Computer Vision



Lecture 01

Intro + Image Acquisition & Representation

Majid Mirmehdi | majid@cs.bris.ac.uk

Staff on the unit



Prof. Majid Mirmehdi



Prof. Andrew Calway
UNIT DIRECTOR

Lectures: Mondays 2pm – PHYS BLDG G12 MOTT
Thursdays 11am – FRY BLDG LG02

Labs: IF YOU ARE ON COMS30087 (20CP): Thursdays 3-5pm – QUEENS BLDG 1.80

Delivery and Assessment: (Minor) COMS30081

The unit aims at providing a first theoretical introduction to classical computational vision: theory, techniques, and algorithms.

Lectures introduce one or more topics and identify key theories, challenges and applications. Students are expected to follow with self study based on problem sheets, revision and further reading.

Problem sheets will be made available for self-study to help in understanding concepts.

Some **past exam papers** will be made available.

ASSESSMENT: 100% Exam



"Well, if the test is multiple choice
I choose not to take it."

Delivery and Assessment: (Major) COMS30087

Lectures and Labs to give you a practical intro to classical computational vision to help you experience implementing such algorithms.

Implementations will be via OpenCV, which is open-source and freely available for most platforms and languages.

Choose to work on your platform in a language you are most fluent in (at your own risk!); we will support the QUEENS 1.80 Lab setup and the Python interface of OpenCV.

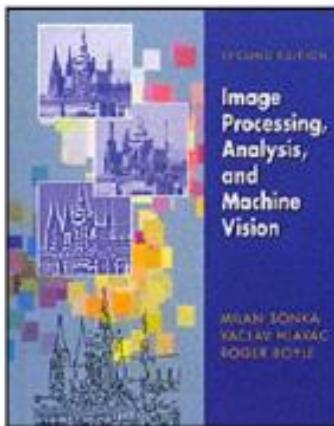
You will work individually during the labs, including parts that will be used for your coursework. You will submit both code and a report for the CW.

**ASSESSMENT: 70% Coursework +
30% Mid-Term Exam**

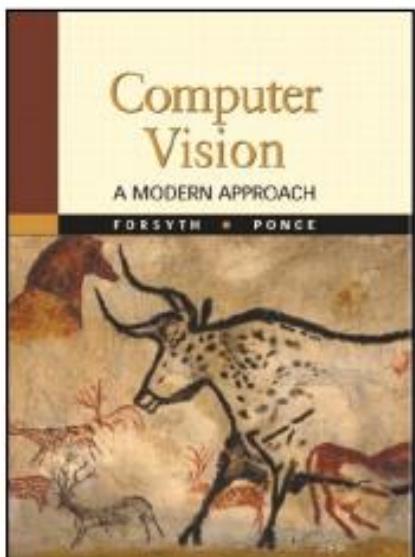
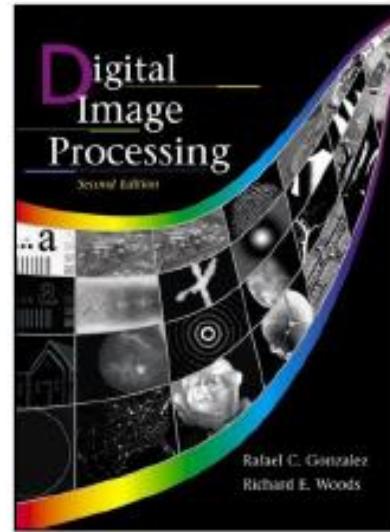


Some Suggestions for General Reading

Image Processing
Analysis and
Machine Vision
by Sonka, Boyle and Hlavac

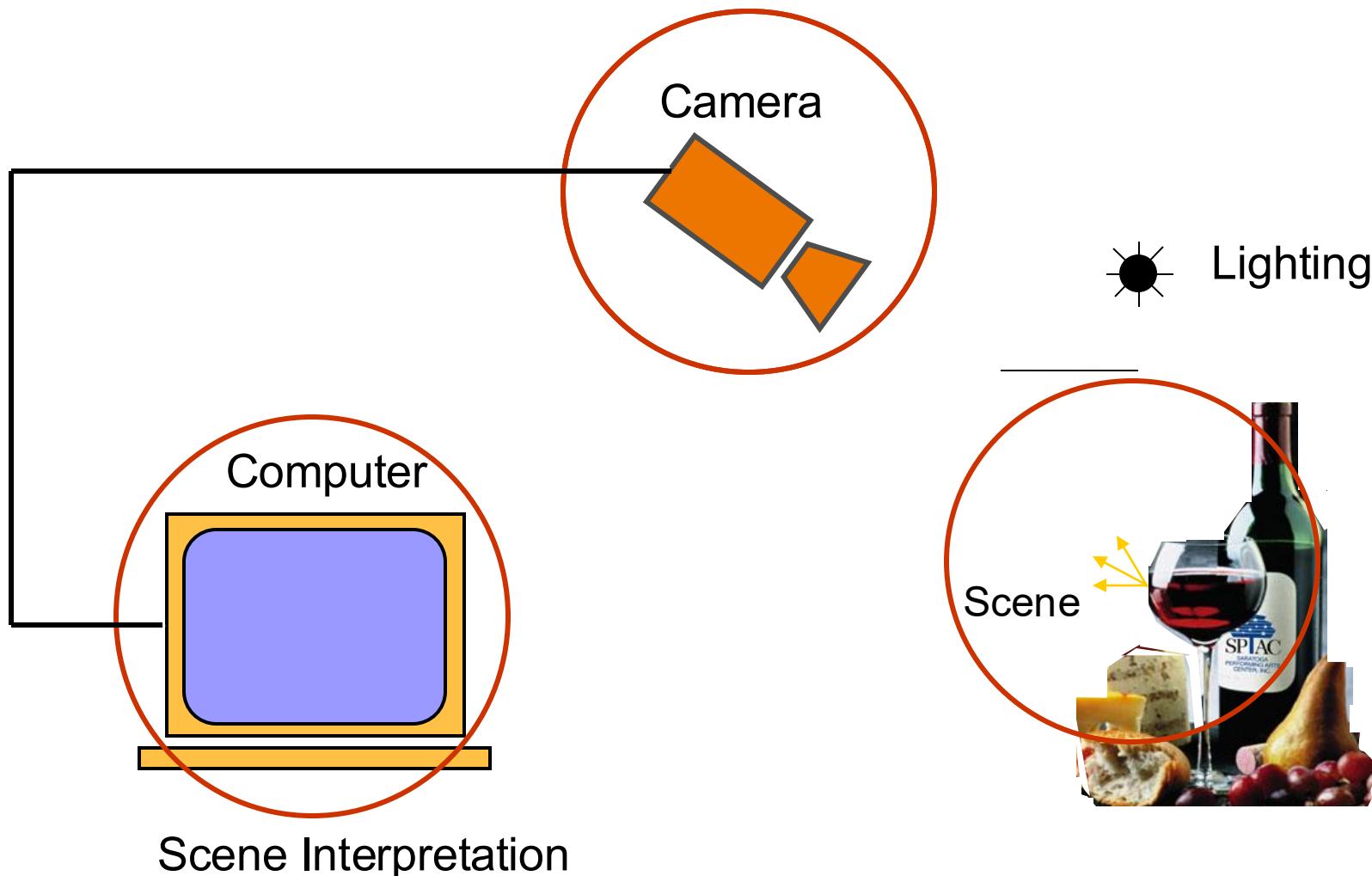


Digital
Image
Processing
by Gonzalez and Woods



Computer Vision:
A Modern Approach
by Forsyth and Ponce

Components of an imaging system



Source: Srinivasa Narasimhan

Images or Image Frames



A single image



**A bunch of image frames
that make up a video**

What is Image Processing?



How to Brighten Images or Videos

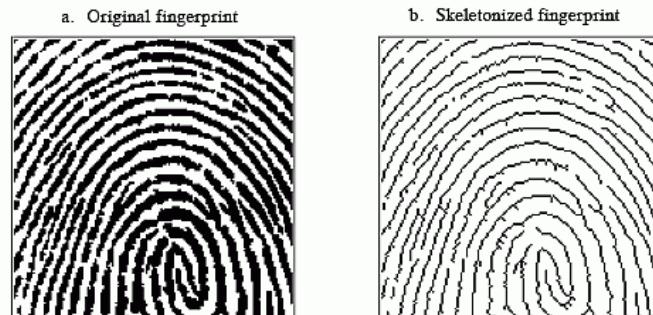


Image Processing... is the digital manipulation of an image to enhance it or extract some useful information from it for further processing.

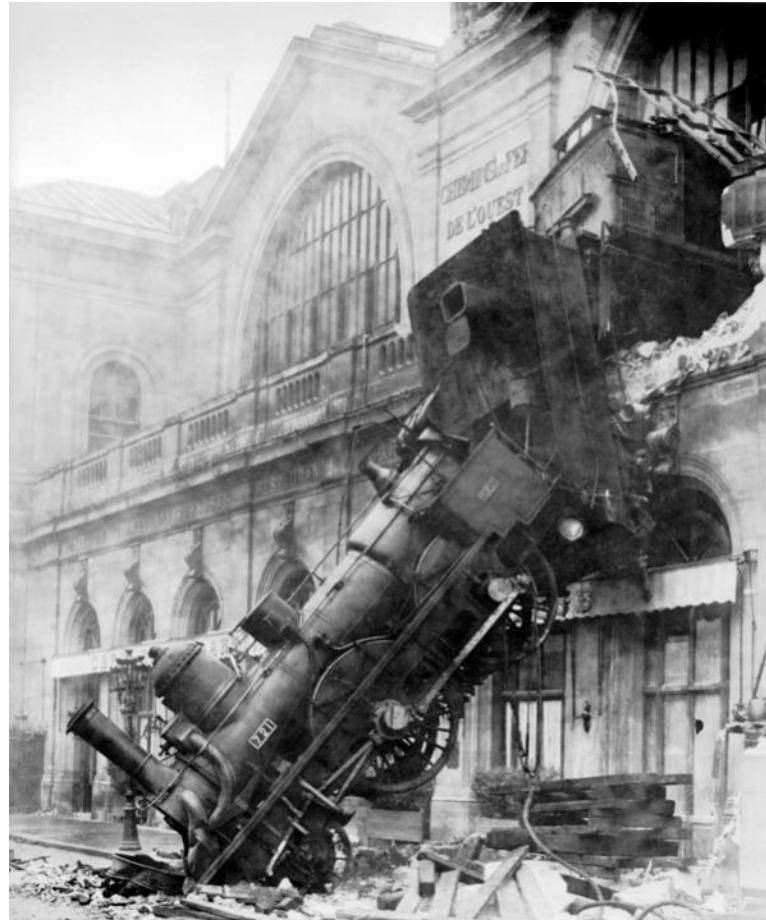
Source image from: <https://www.dspguide.com/ch25/4.htm>

What is Computer Vision?



Computer Vision ... attempts to bridge the semantic gap between pixels and their meaning.

What is Computer Vision?



The goal is to perceive the story in the scene!

Computer Vision is hard!

- Vision is an amazing feature of natural intelligence
 - More human brain devoted to vision than anything else

What we see



151	121	1	93	165	204	14	214	28	235	29	142	142	75	22	189	111	28	6	5	
62	67	17	234	27	1	221	37	189	141	137	168	41	286	100	70	219	127	114	191	
28	168	155	113	178	228	25	130	139	221	285	154	226	14	89	86	242	67	203	15	
236	136	158	230	10	5	165	17	30	155	247	47	128	123	253	229	181	251	232	28	
174	148	93	70	95	186	151	10	160	214	68	75	24	99	93	63	215	222	102	180	
103	126	58	16	138	136	98	202	42	233	206	246	85	183	215	3	62	64	77	216	
235	103	52	37	94	184	173	86	223	113	126	80	165	149	196	75	186	60	179	193	
212	15	179	139	48	232	194	46	174	37	44	253	164	253	14	216	175	30	46	254	
119	81	241	172	95	178	29	210	22	194	137	23	33	283	241	21	144	63	244	188	
129	19	33	253	229	5	152	233	52	44	32	214	142	121	249	189	99	232	183	71	
88	200	194	185	140	200	223	190	164	182	45	36	152	27	198	137	61	1	237	247	
113	16	228	215	143	184	247	29	97	283	1	14	241	70	2	30	151	67	169	285	
9	210	102	246	75	9	158	184	184	129	32	80	182	32	99	169	91	166	73	214	
124	52	76	148	249	187	65	216	187	181	186	219	9	283	209	240	40	249	119	122	
6	251	52	288	46	65	185	38	77	240	177	252	38	283	119	0	217	139	139	157	
150	194	28	206	148	197	288	28	74	93	154	145	49	251	158	185	235	23	230	156	
33	183	248	153	188	285	146	180	254	218	157	168	223	60	247	118	5	180	16	286	
130	53	128	212	61	226	281	110	140	183	162	208	195	246	148	138	54	191	139	79	
165	246	22	182	151	213	40	138	8	93	17	233	85	159	166	24	49	40	160	97	
152	251	101	238	23	162	70	238	75	24	84	242	247	144	283	3	19	24	198	88	
187	105	152	83	167	98	125	180	136	121	67	67	185	98	123	106	168	105	127	153	
139	197	55	289	28	124	288	288	104	40	37	113	214	252	283	80	146	211	7	16	
123	19	144	223	62	253	202	108	47	242	142	241	66	86	214	133	146	253	189	200	
220	144	31	16	136	12										189	54	144	56	59	163

What a computer sees

What is Computer Vision?

Pixels

Features

Models

Meaning

Computer Vision ...
... concerns the study of the theory, engineering and application of artificial systems that extract semantic information from images or other structured, multidimensional data.



Latest Developments: Human Pose/Motion Modelling



Per-frame
estimation



Side View

<https://shubham-goel.github.io/4dhumans/>

$\frac{1}{2}$

- Acquisition and Representation ← Majid
- Image Transforms ← Majid
- Edges and Hough Transforms ← Majid
- Segmentation ← Majid
- Object Detection ← Majid and Amir
- $\frac{1}{2}$ {
 - Motion Analysis ← Andrew
 - Stereo Vision ← Andrew

So what does the human visual system see?

**Or what should an automatic
computer vision system see?**



Detection: are there cars?



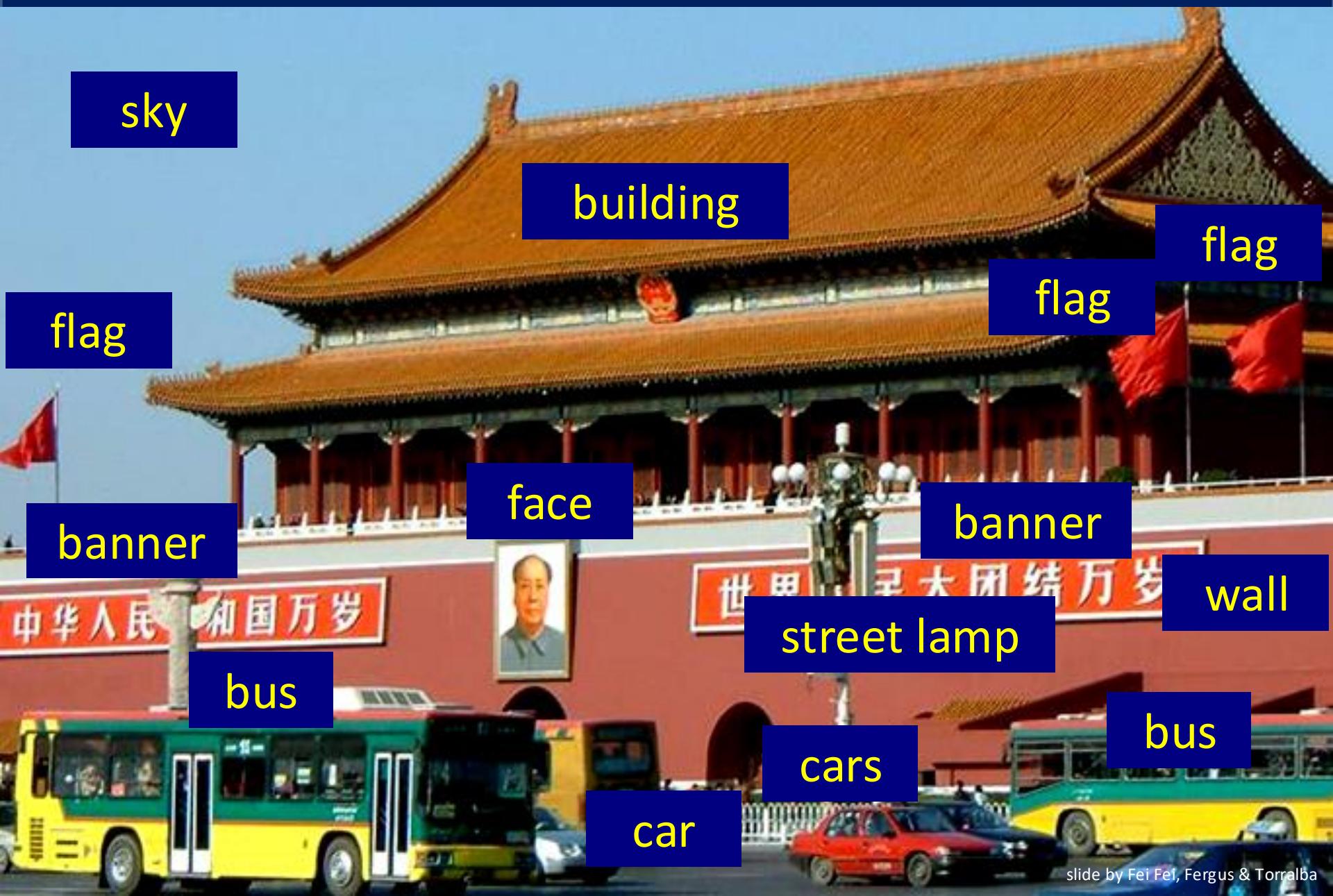
Verification: is that a bus?



Identification: is that a picture of Mao?

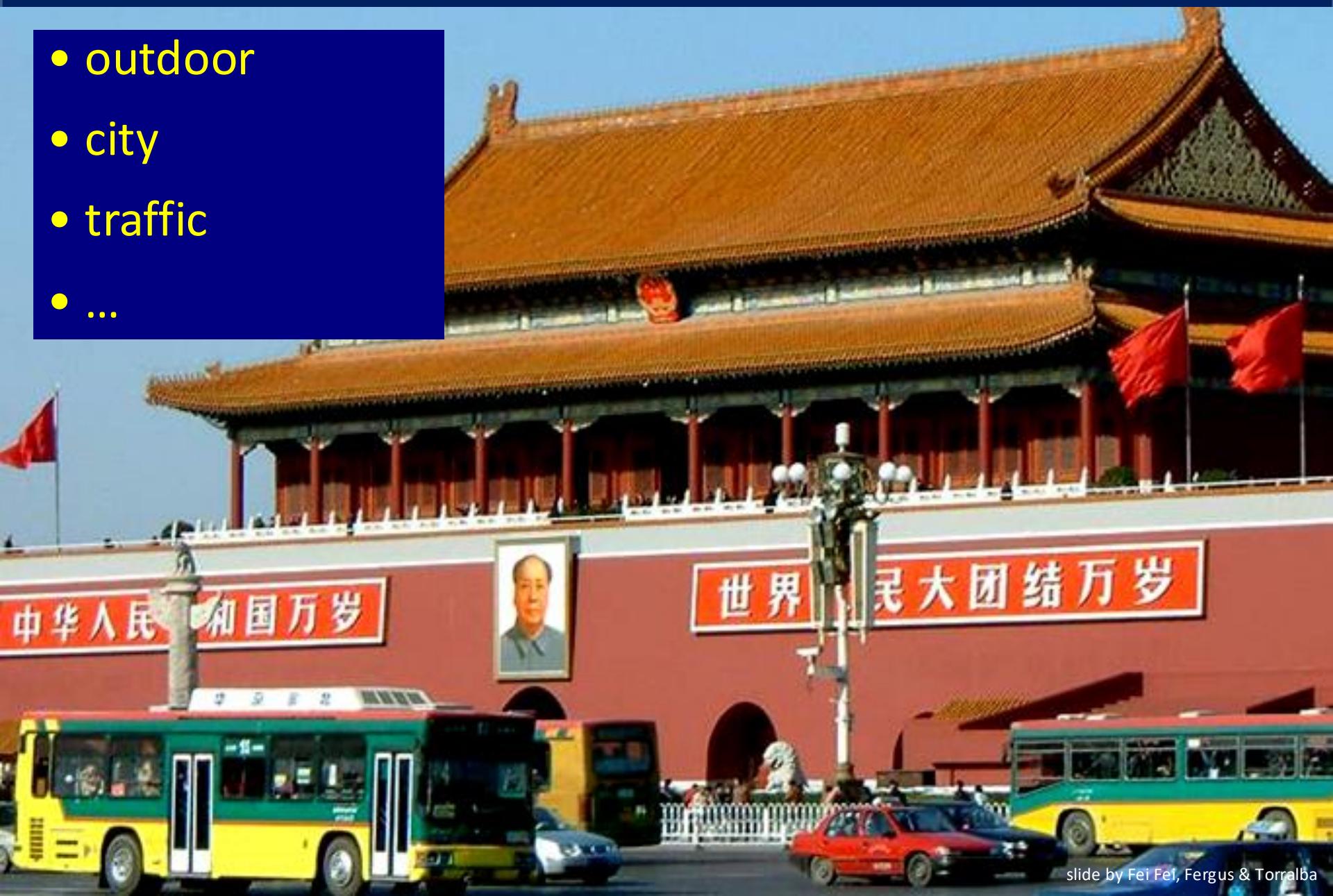


Object categorization



Scene and context categorization

- outdoor
- city
- traffic
- ...



Rough 3D layout, depth ordering



Challenges

So what are the challenges faced by a *computer vision system*?

Challenge 1: view-point variation



Michelangelo 1475-1564

original Challenge slides by Efros, Ullman, and others

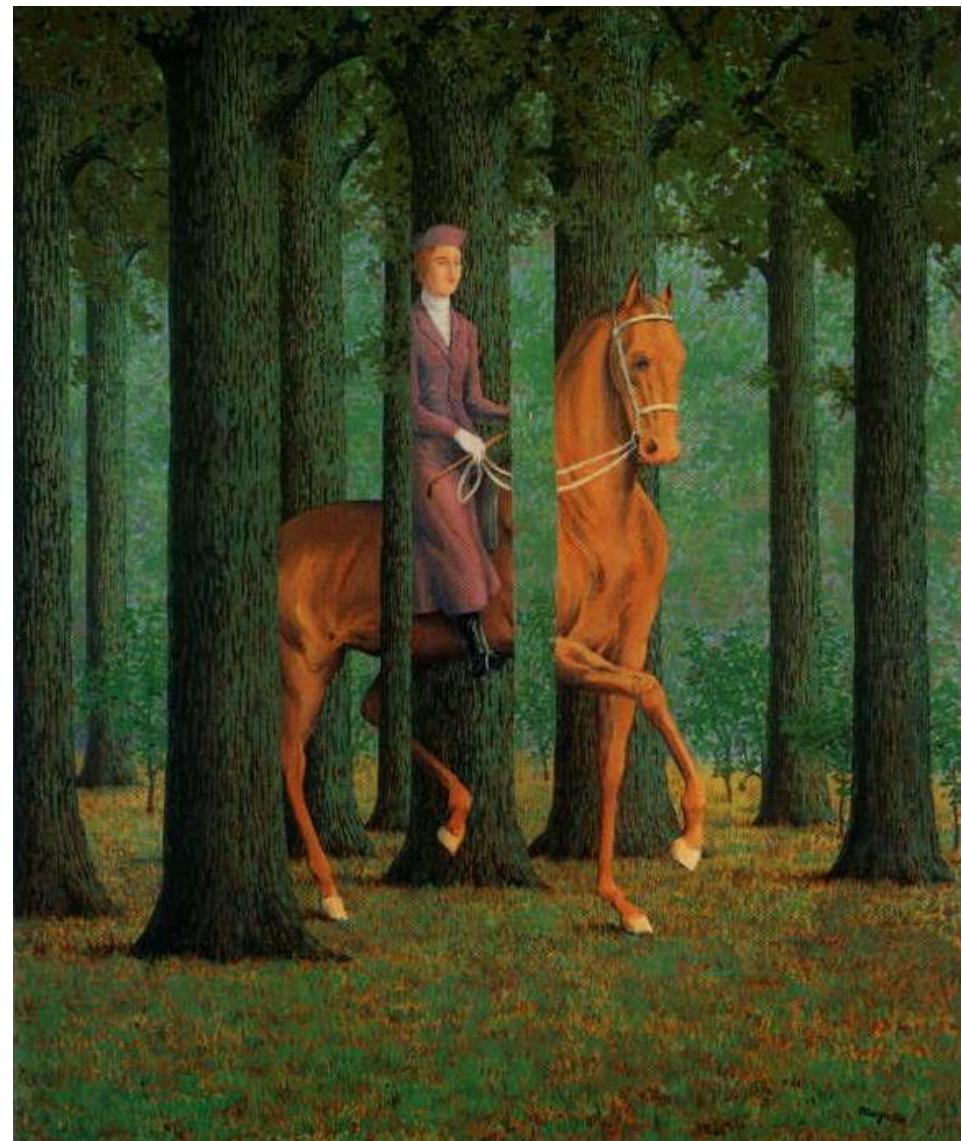
Challenge 2: illumination



original Challenge slides by Efros, Ullman, and others

Challenge 3: occlusion

Rene Magritte, 1957



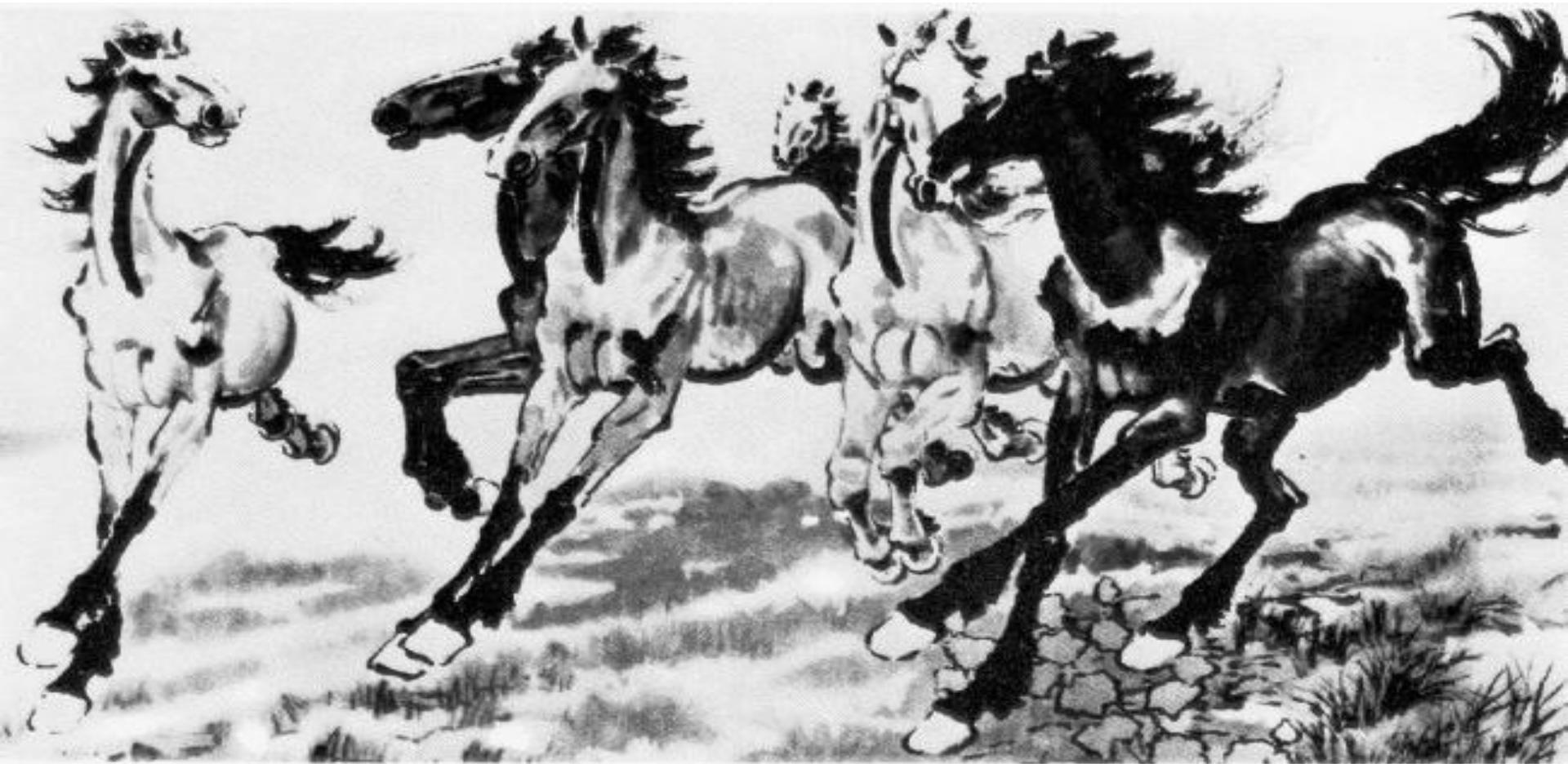
original Challenge slides by Efros, Ullman, and others

Challenge 4: scale



original Challenge slides by Efros, Ullman, and others

Challenge 5: deformation



Xu, Beihong 1943

original Challenge slides by Efros, Ullman, and others

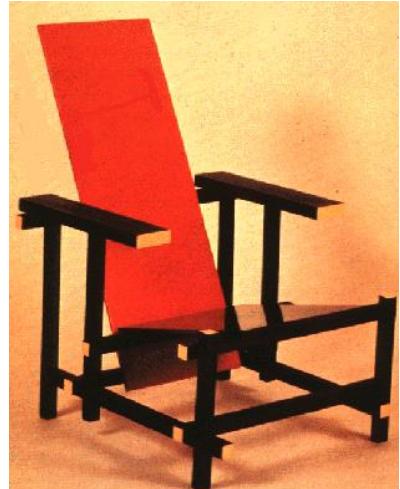
Challenge 6: background clutter



Klimt, 1913

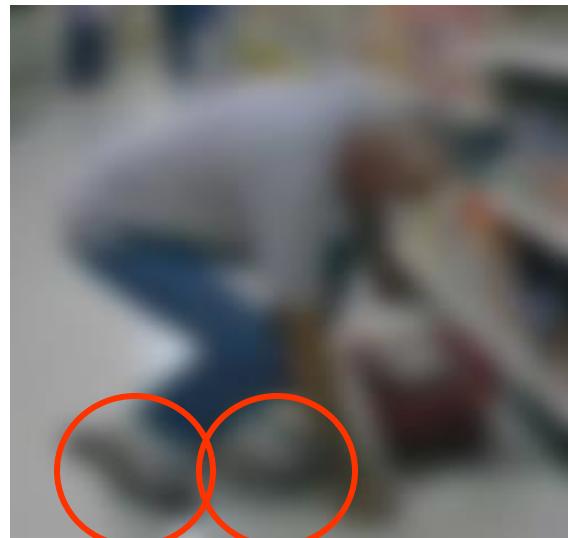
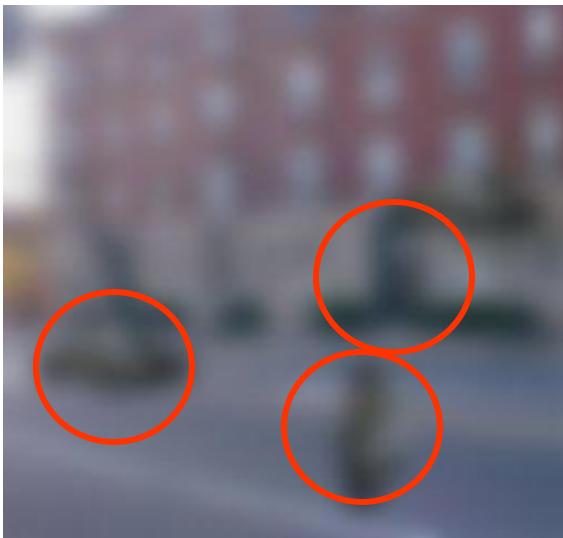
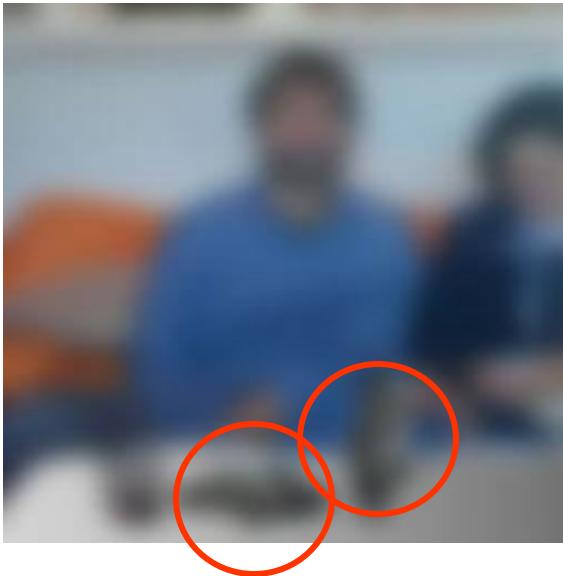
original Challenge slides by Efros, Ullman, and others

Challenge 7: object intra-class variation



original Challenge slides by Efros, Ullman, and others

Challenge 8: local ambiguity



original Challenge slides by Efros, Ullman, and others

The Basics of Image Acquisition and Representation

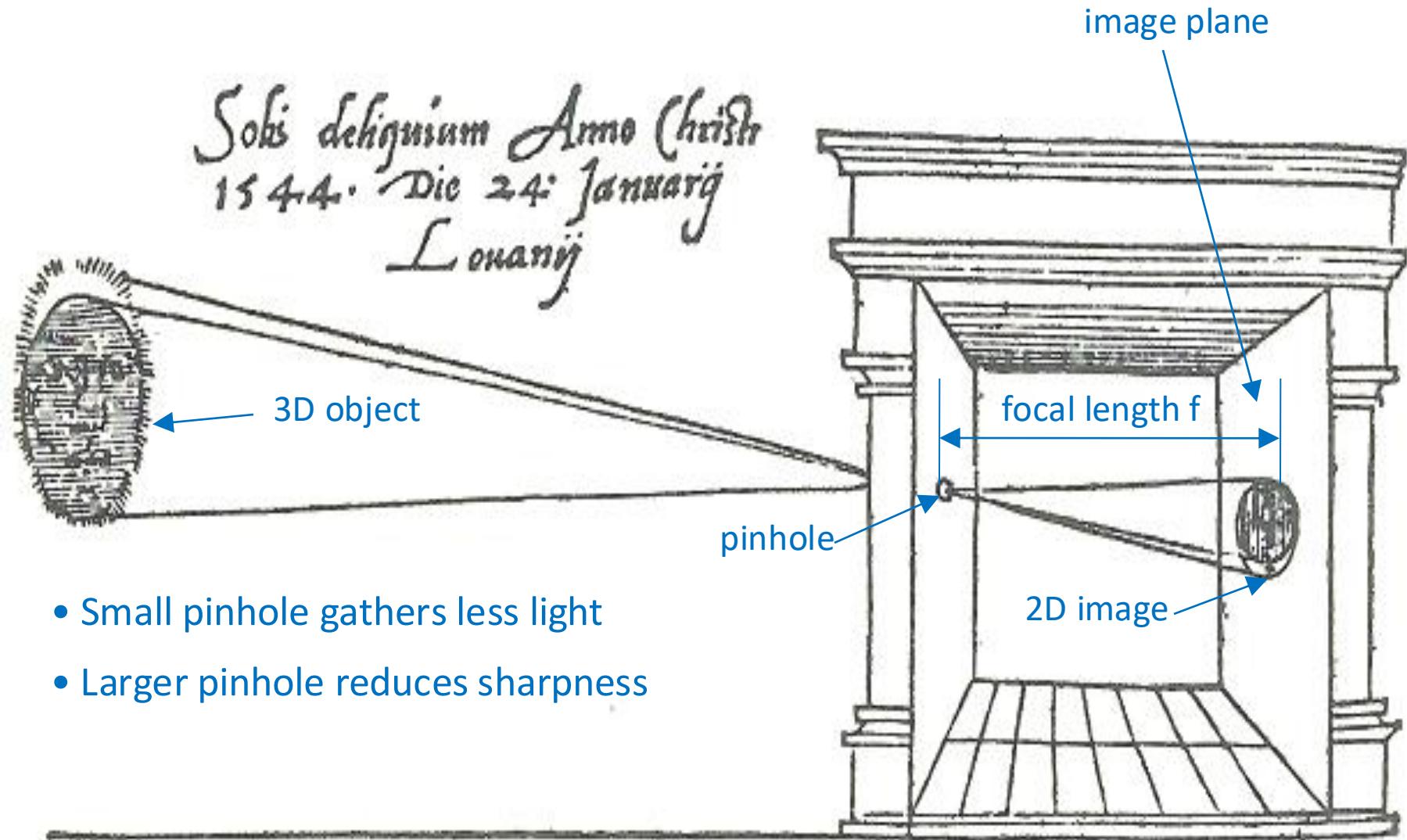
Images as Sensory Data

- How are images acquired?
- Which processes influence digital image formation?

Images as Structured Data

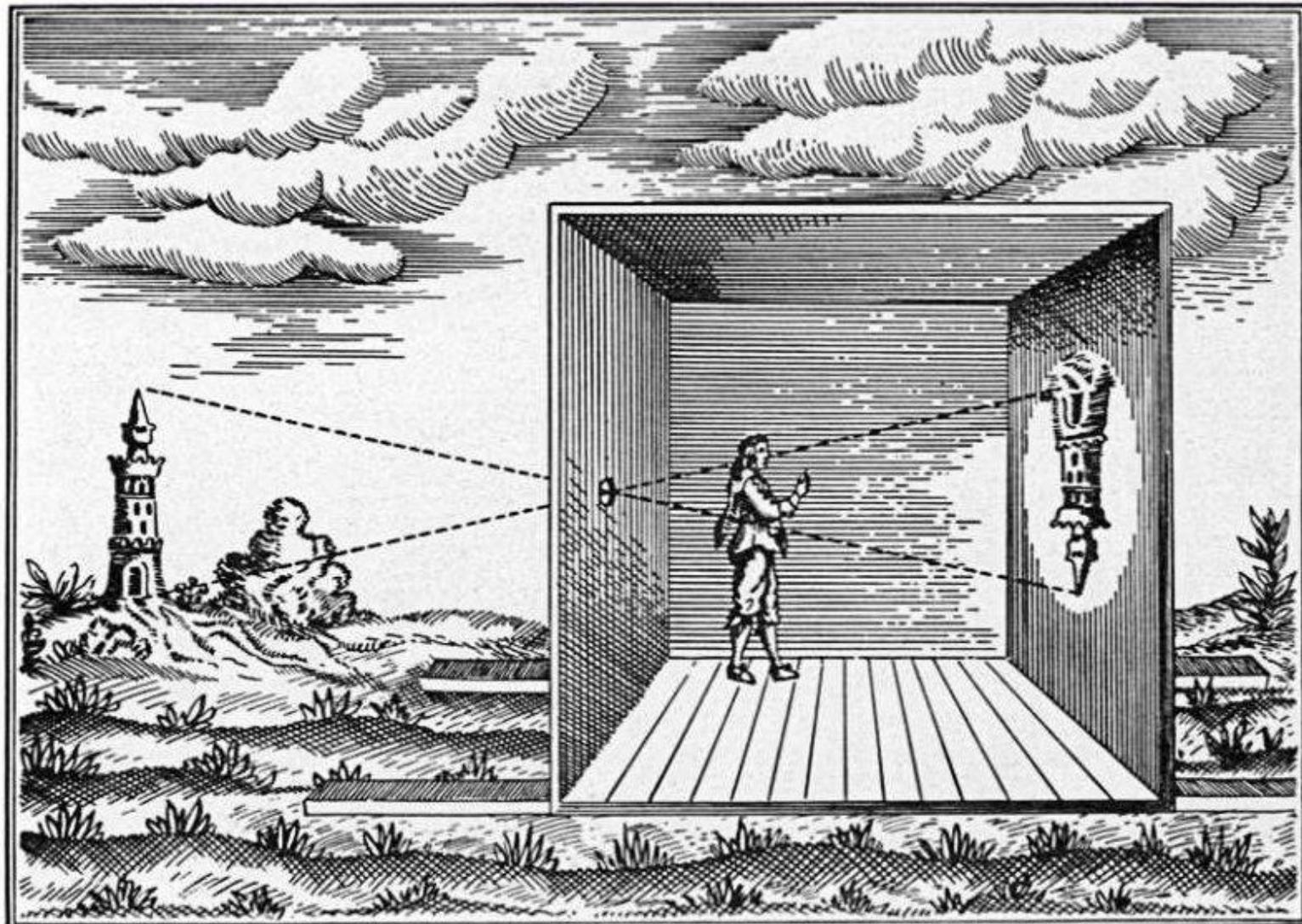
- How can digital images be represented?

The Camera Obscura (Pinhole Camera)

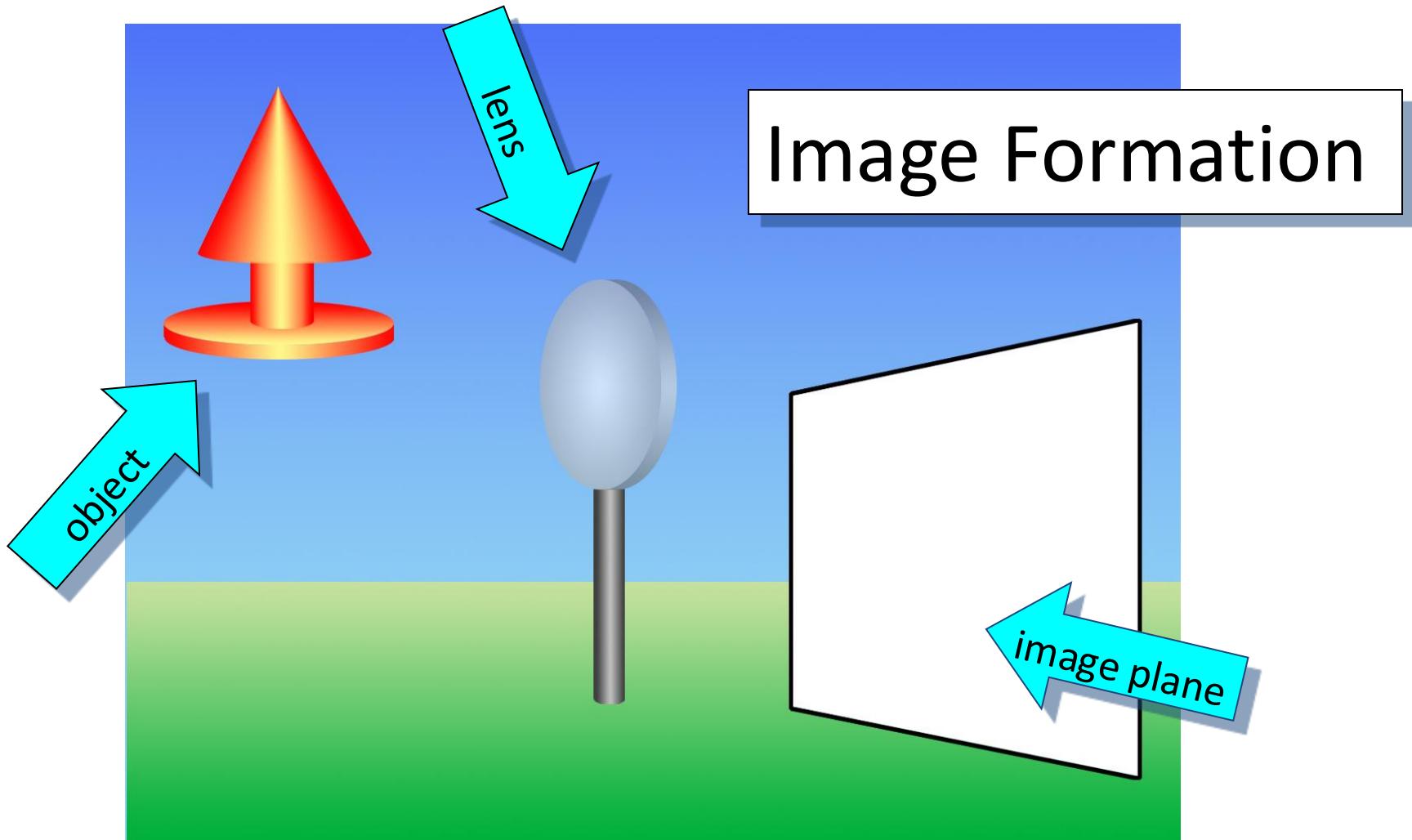


First published picture of camera obscura in Gemma Frisius' 1545 book *De Radio Astronomica et Geometrica*

The Camera Obscura (Pinhole Camera)

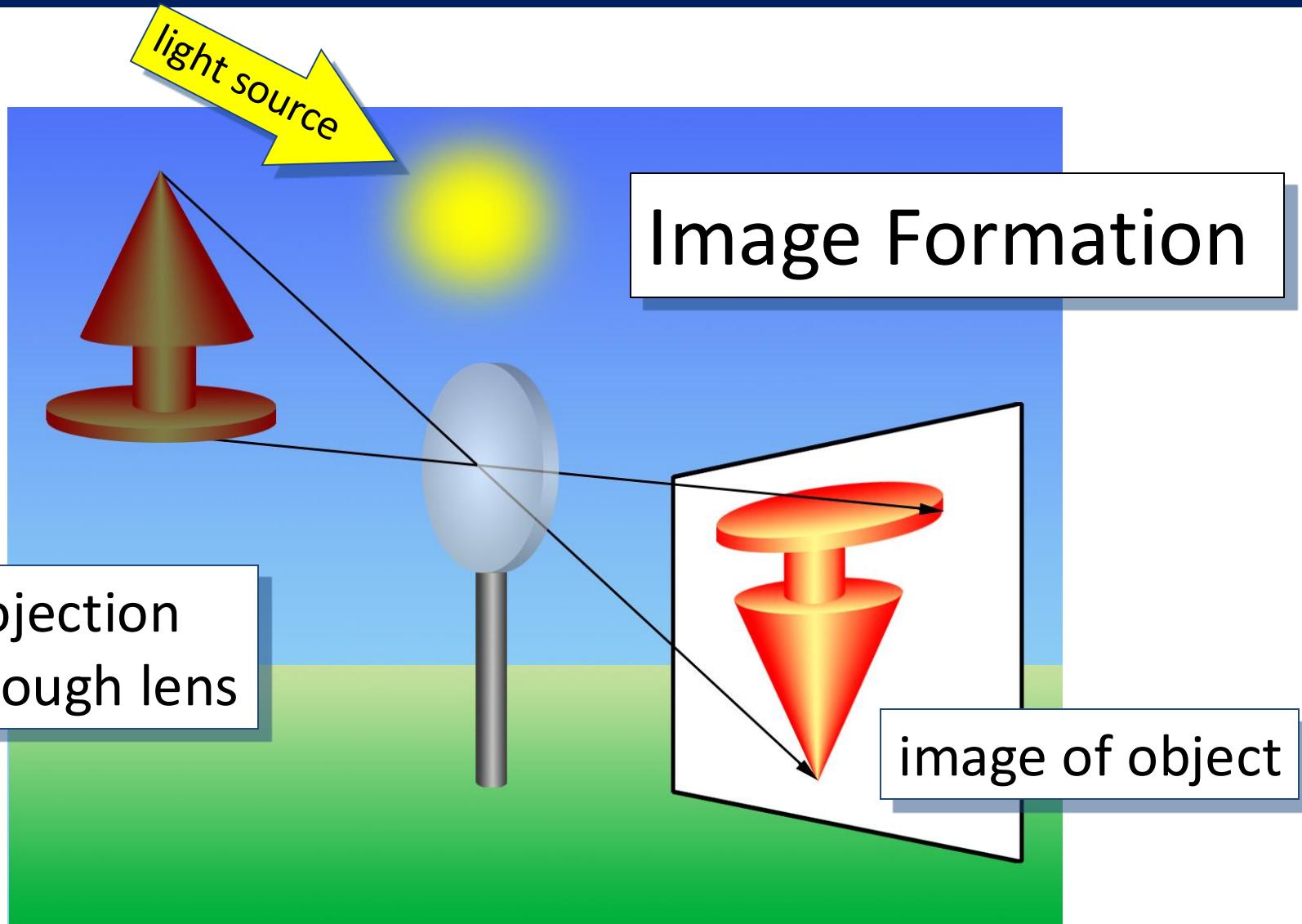


Digital Image Acquisition



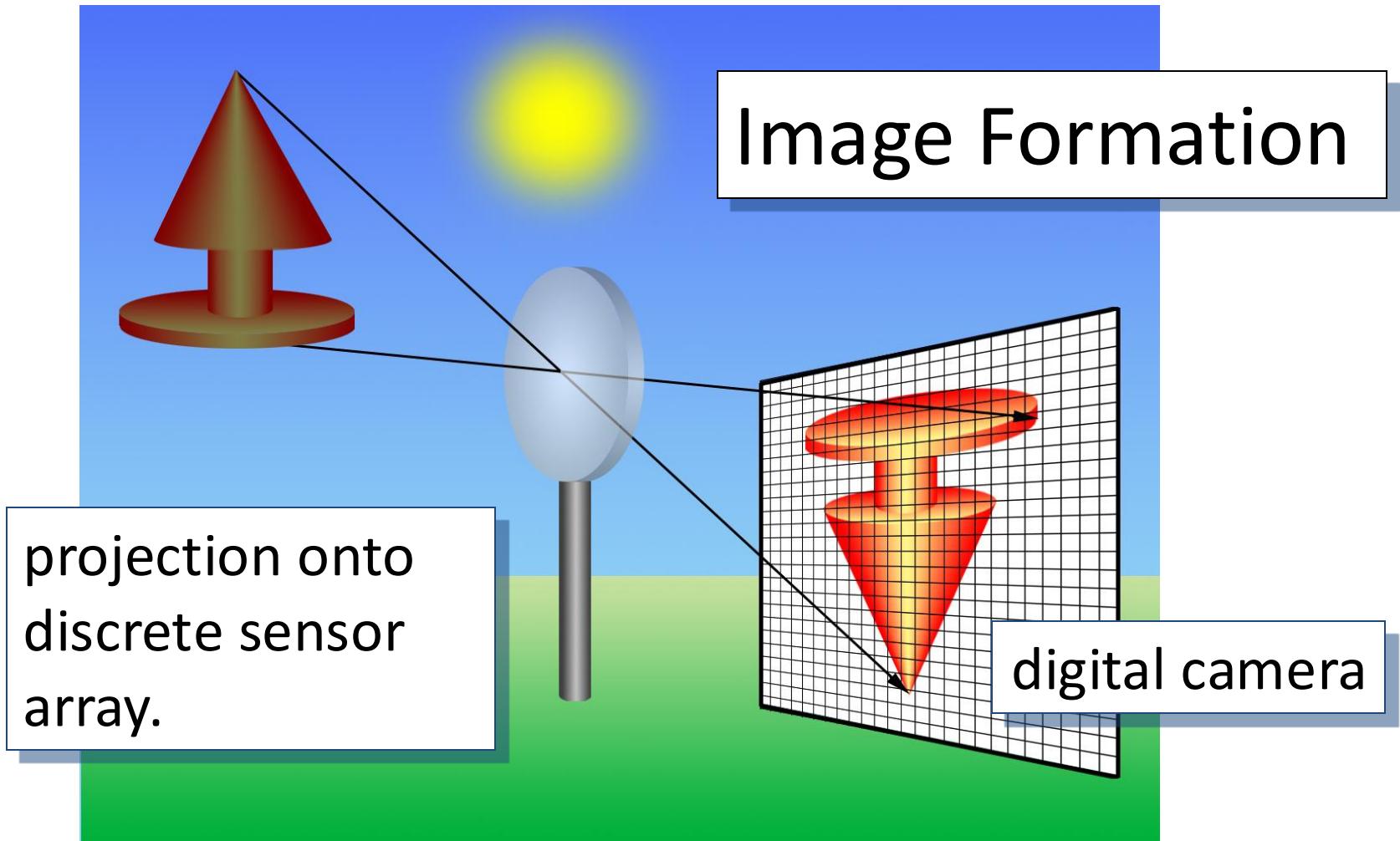
Slide by R A Peters

Digital Image Acquisition



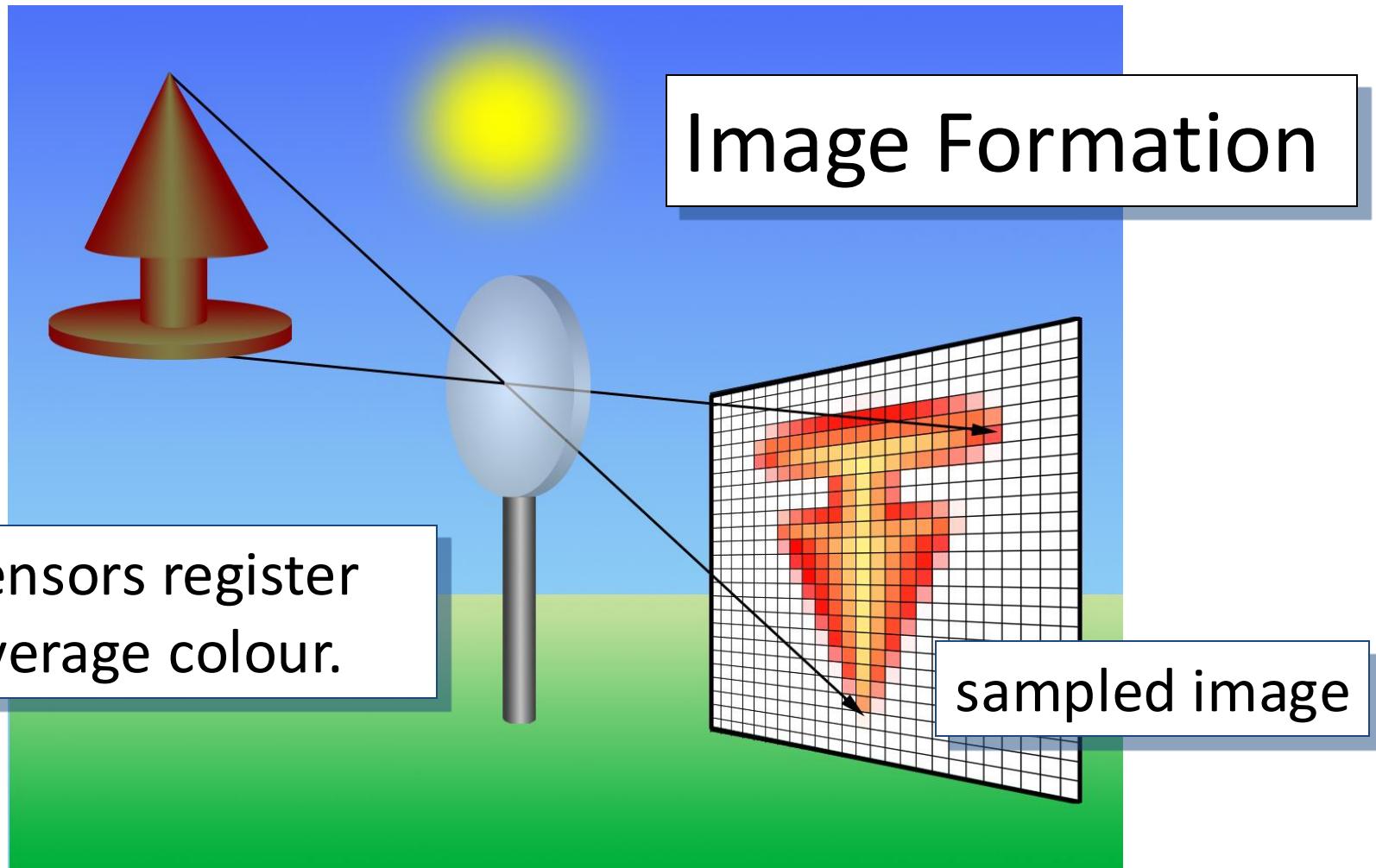
Slide by R A Peters

Digital Image Acquisition



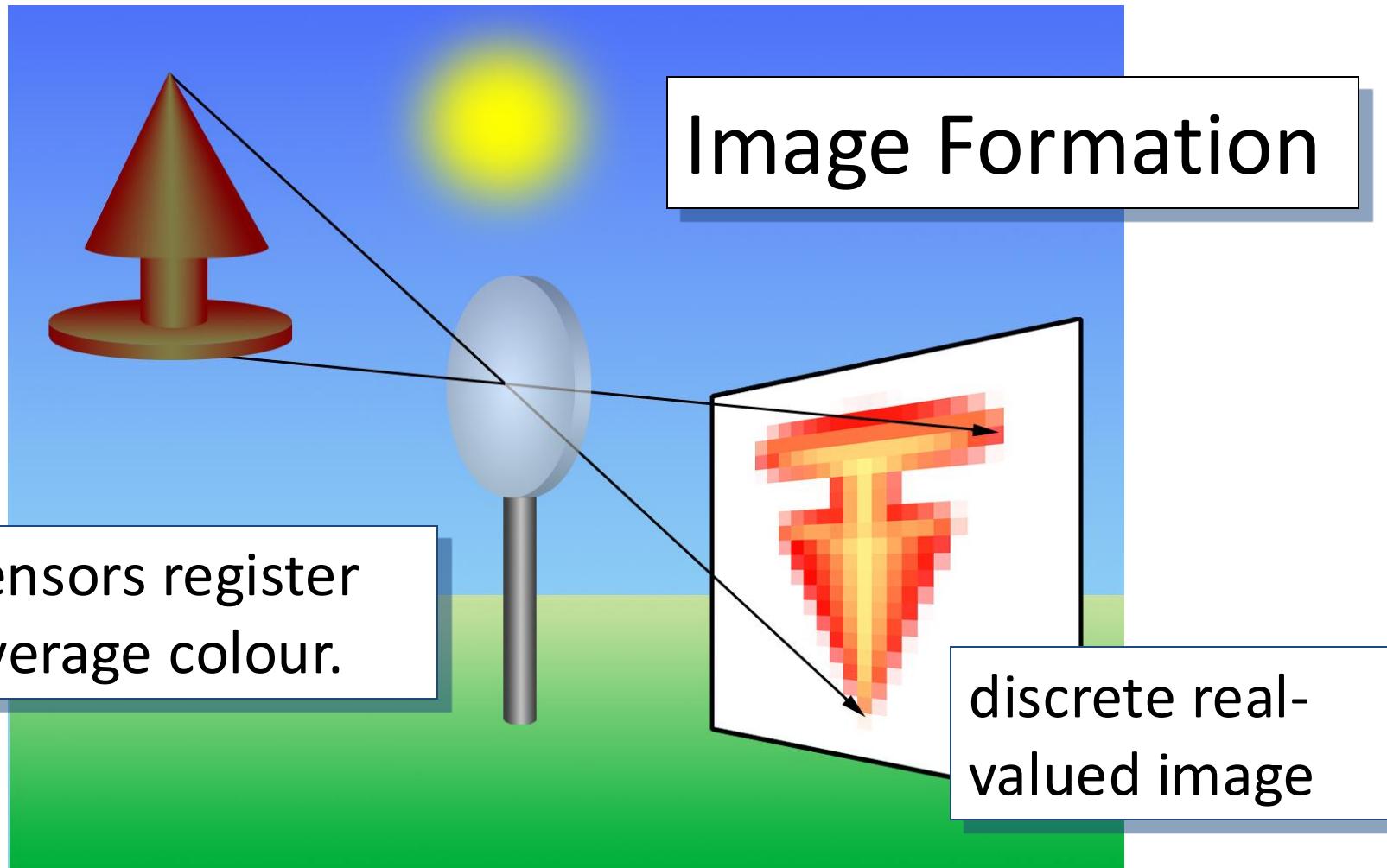
Slide by R A Peters

Digital Image Acquisition



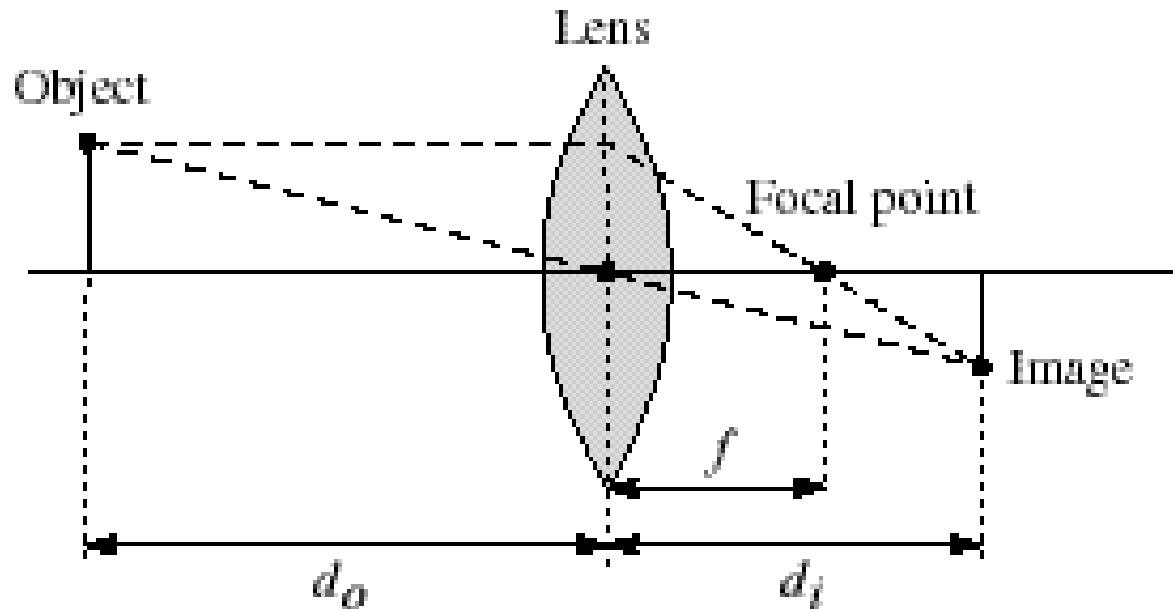
Slide by R A Peters

Digital Image Acquisition



Slide by R A Peters

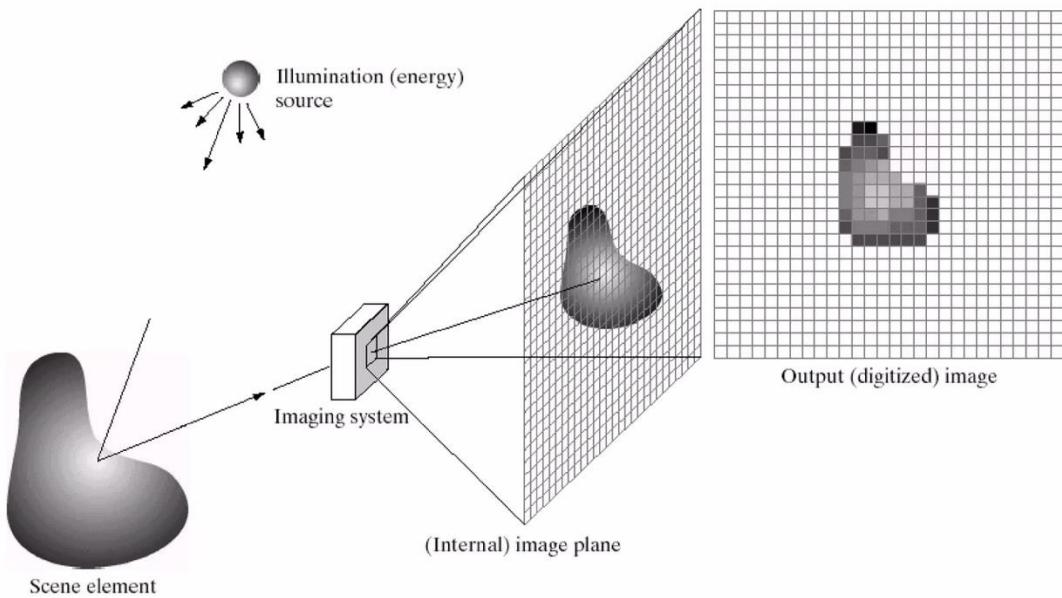
Lens



Any object point satisfying the thin lens equation is in focus:

$$\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$$

Sampling & Quantisation to obtain an Image



a c d e
b

FIGURE 2.15 An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

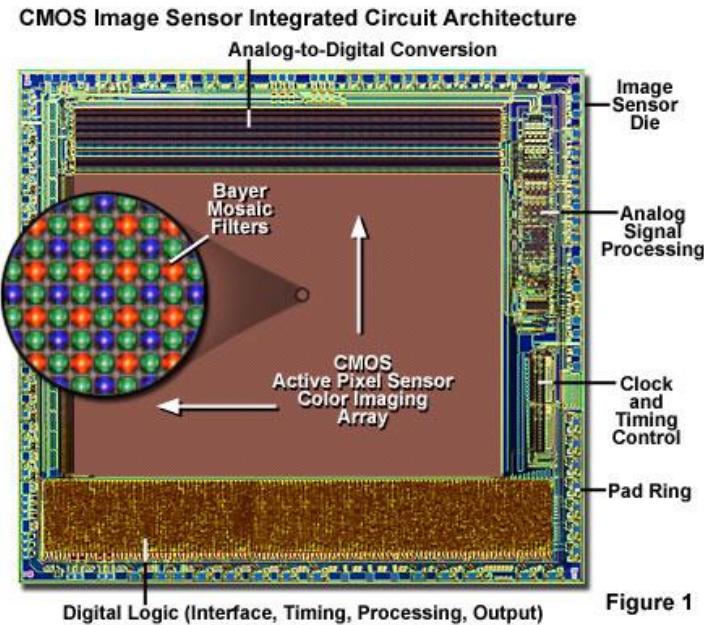
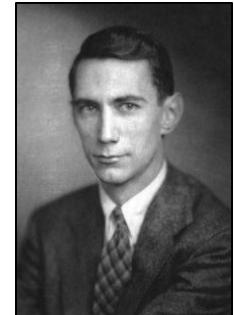


Figure 1

Pixel value corresponds to the electrical charges activated by light photons hitting the sensor array

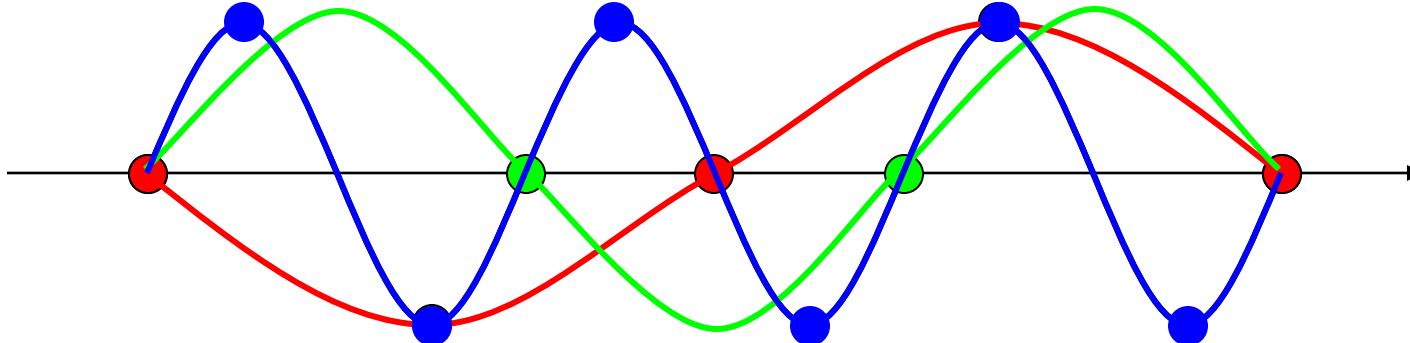
Shannon's Sampling Theorem

“An analogue signal containing components up to some maximum frequency u may be completely reconstructed by regularly spread samples, provided the sampling rate is above $2u$ samples per second.”

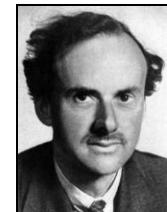
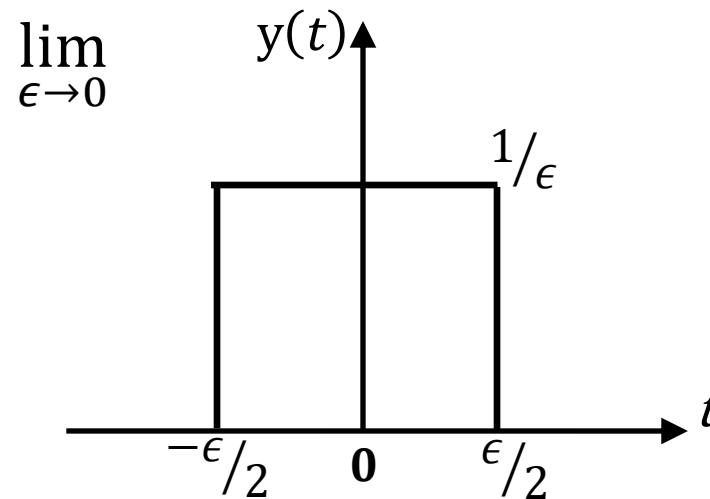
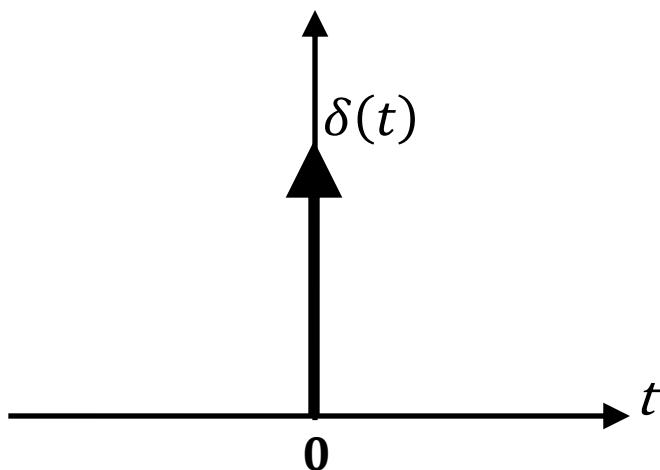


Claude Shannon

Also referred to as the Shannon-Nyquist criterion:
Sampling **must** be performed above twice the highest (spatial) frequency of the signal to be lossless.



Modelling a Spatial Brightness Pulse - Dirac Delta-Function



Paul Dirac

Definition:

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

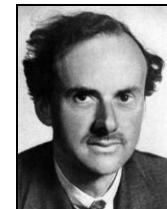
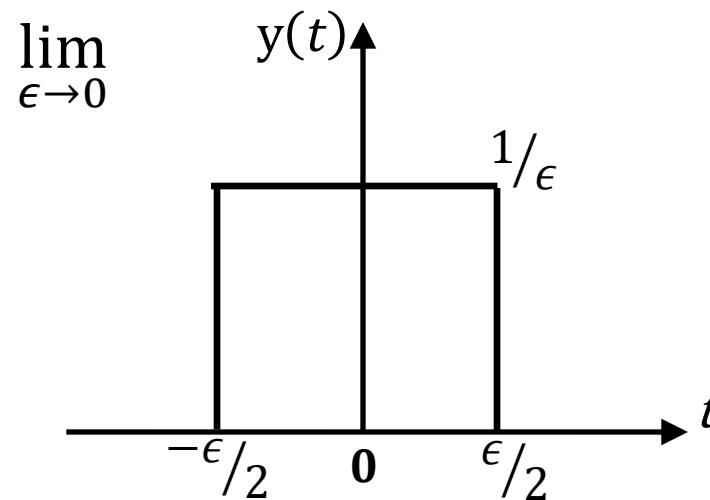
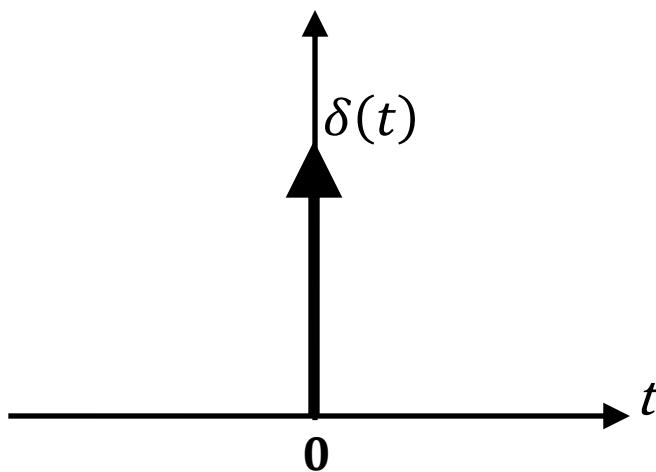


Intuitively:

$$\delta(t) = \lim_{\epsilon \rightarrow 0} [y_\epsilon(t)]$$



Modelling a Spatial Brightness Pulse - Dirac Delta-Function



Paul Dirac

Definition:

$$\int_{-\infty}^{\infty} \delta(t) dt = 1$$

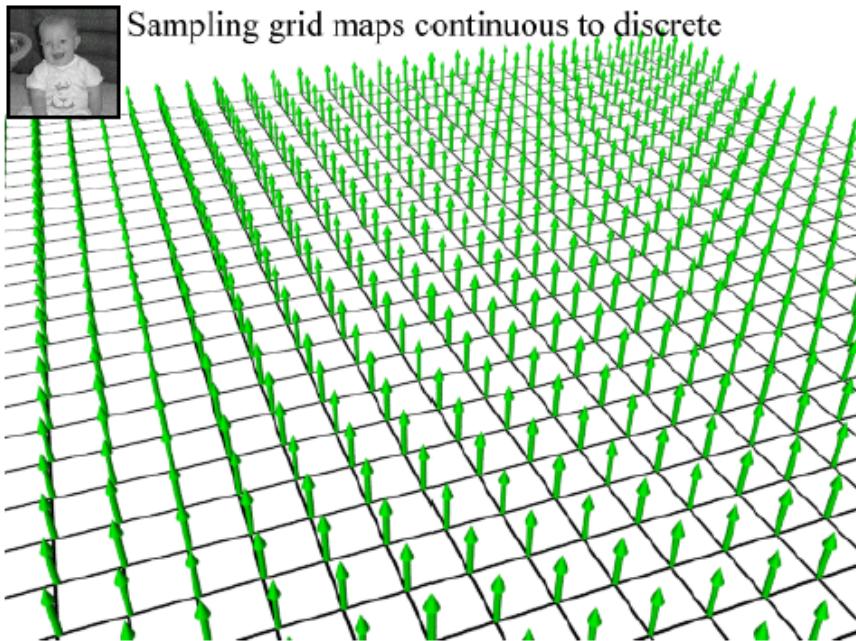
Intuitively:

$$\delta(t) = \lim_{\epsilon \rightarrow 0} [y_\epsilon(t)]$$

Sifting Property:

$$\int_{-\infty}^{\infty} f(t)\delta(t)dt = f(0) \longrightarrow \int_{-\infty}^{\infty} f(t)\delta(t - \alpha)dt = f(\alpha)$$

Sampling in 2D To Obtain An Image



Sampling grid maps continuous to discrete

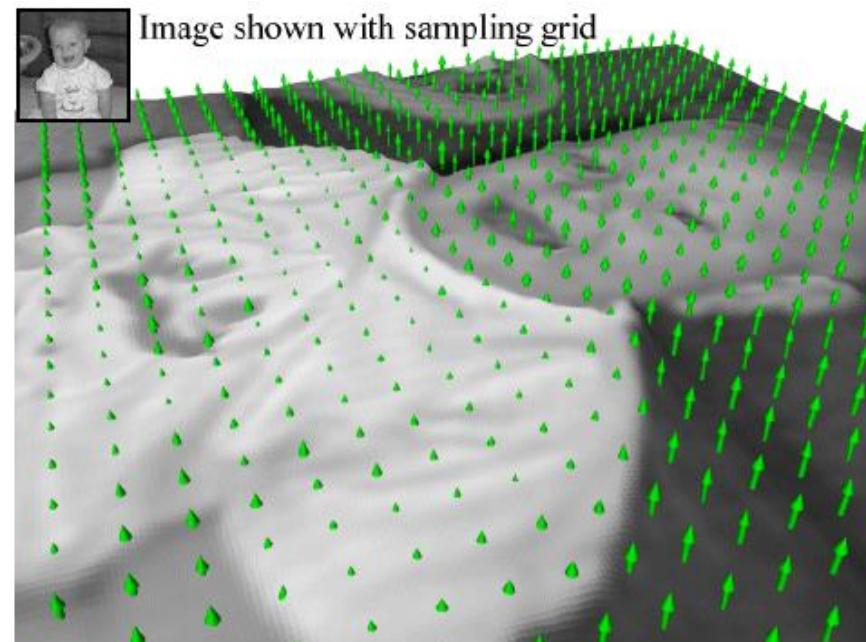
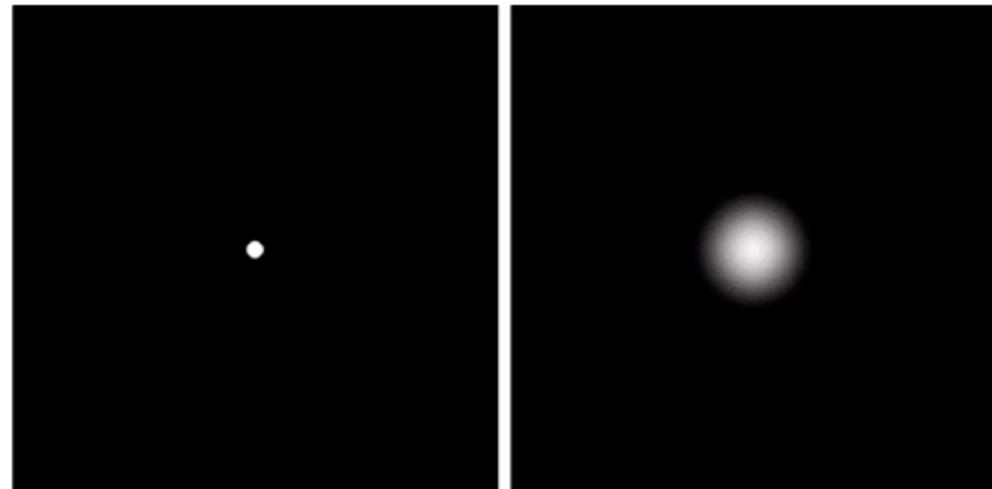


Image shown with sampling grid

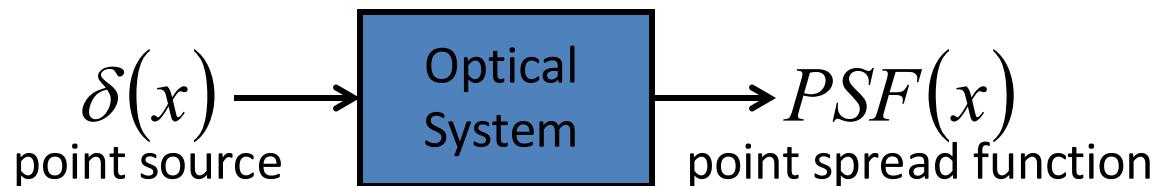
The sifting property can be used to express a 2D ‘image function’ as a linear combination of 2D Dirac pulses located at points (a,b) that cover the whole image plane:

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(a,b) \delta(a-x, b-y) da db = f(x,y)$$

The Point Spread Function

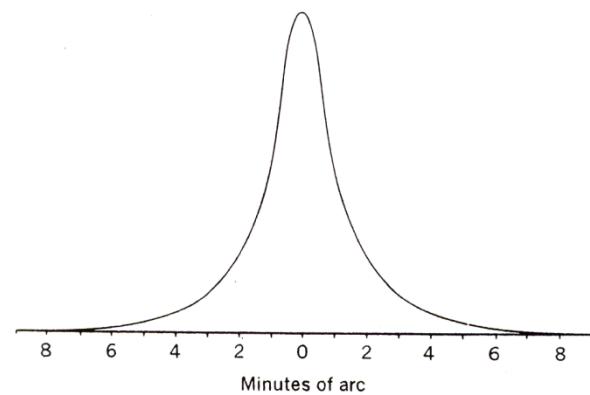


- Ideally, the optical system should be mapping point information to points again.
- However, optical systems are never ideal.



- Superposition Principle:
An image is the sum of the PSF of all its points.

- Point spread function of Human Eyes



PSF Example

$$g(x, y) = f(x, y) * h(x, y)$$

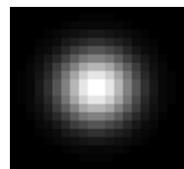
$f(x, y)$ Original



Blurry outcome $g(x, y)$



$h(x, y)$

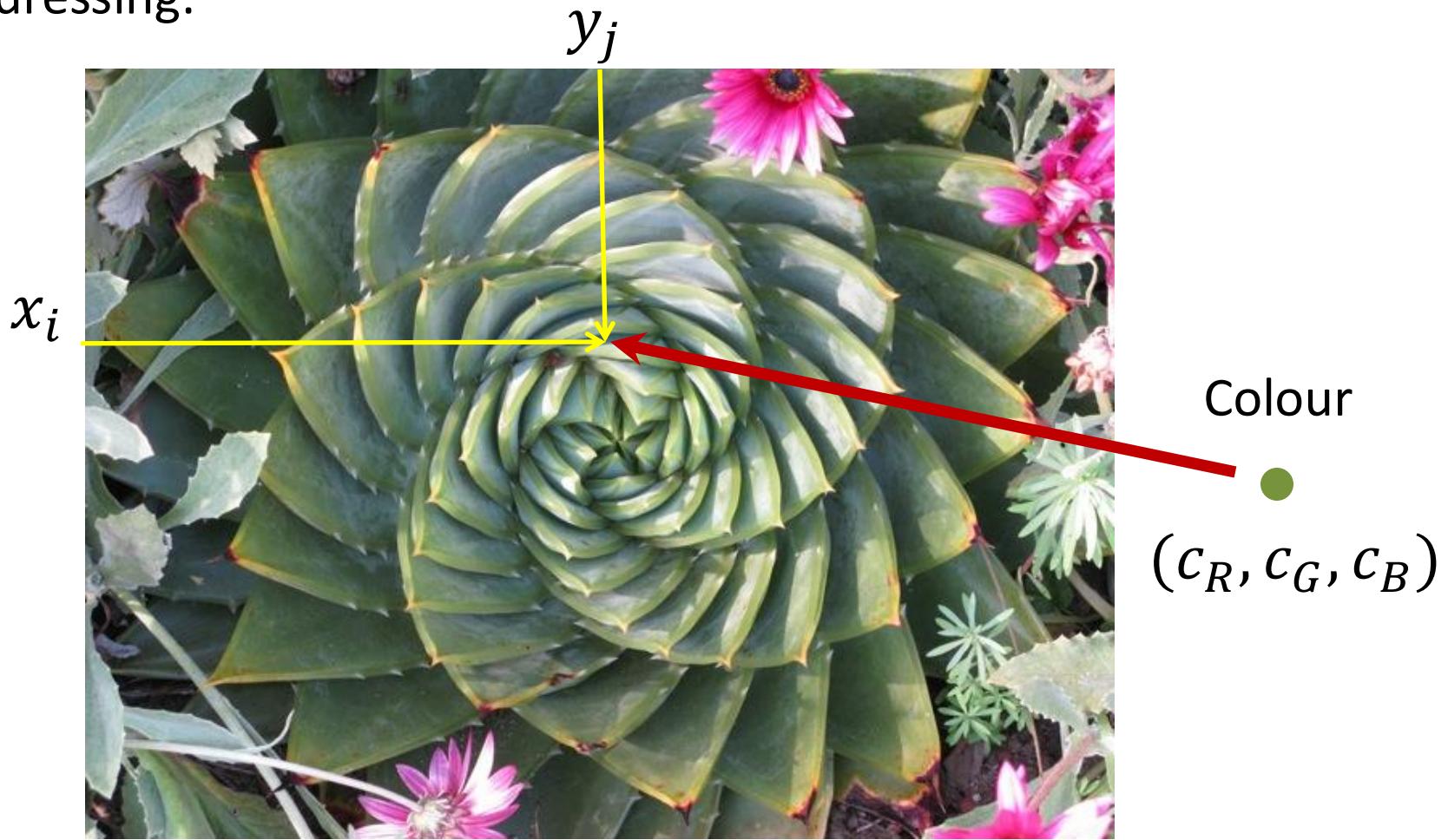


$h(x, y)$ is the PSF of the imaging device.

Adapted from a slide by A. Zisserman

How to model an image?

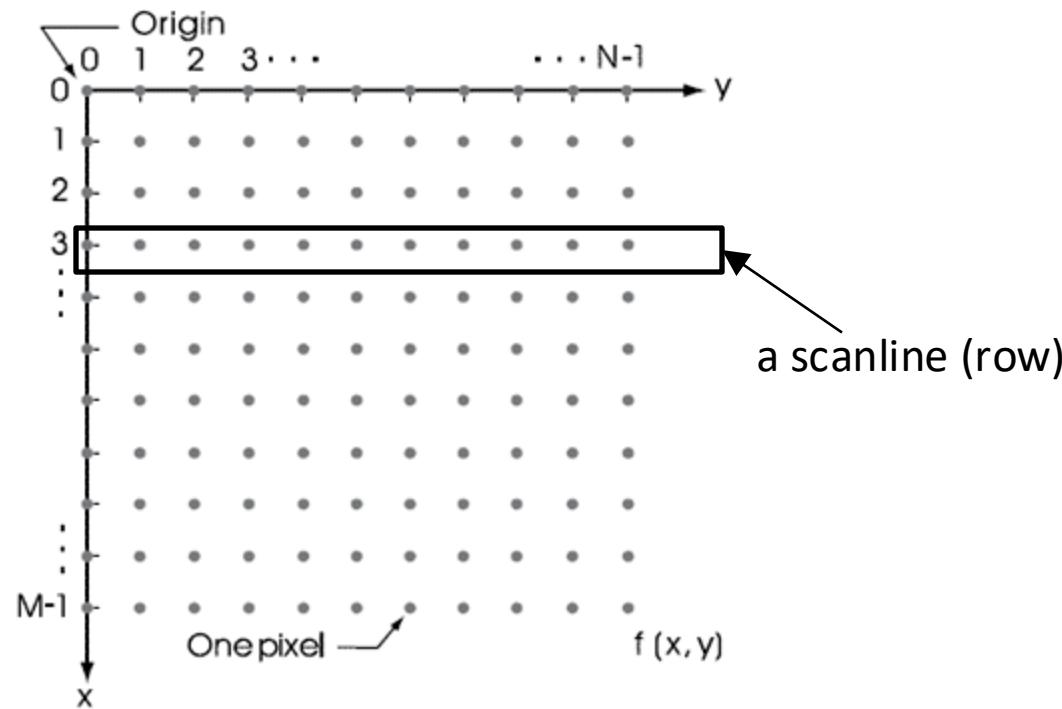
Addressing:



Storage: each colour value is 8-bits, hence $3 \times 8 = 24$ bits per pixel

How to represent a digital image?

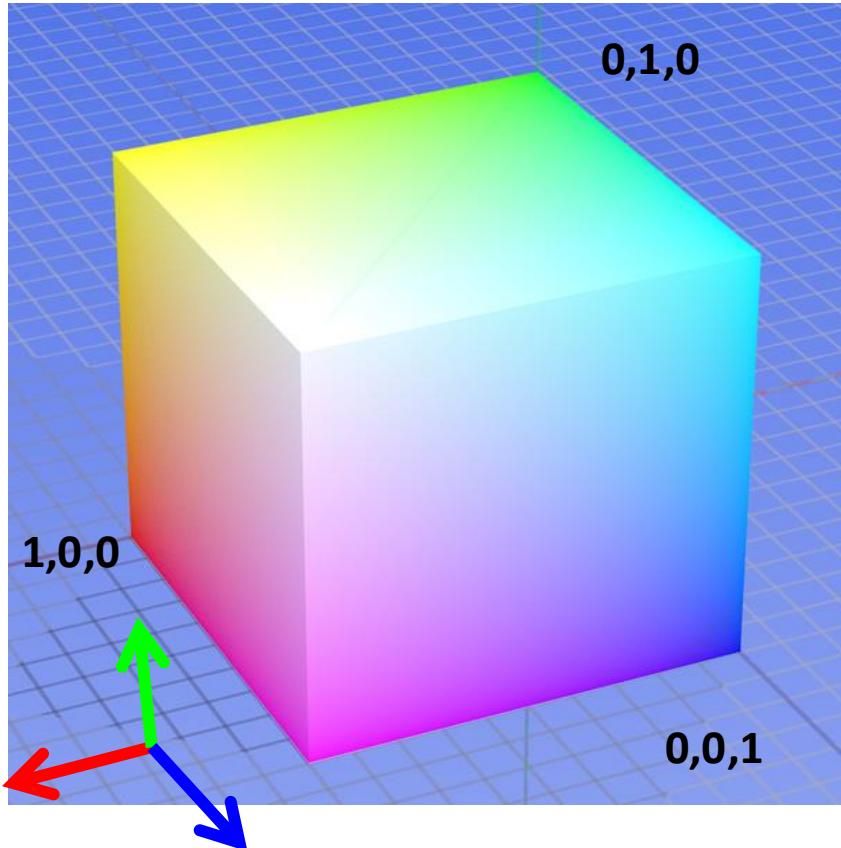
$$f[x, y] = \begin{bmatrix} f[0, 0] & f[0, 1] & \dots & f[0, N - 1] \\ f[1, 0] & f[1, 1] & \dots & f[1, N - 1] \\ \vdots & \vdots & \ddots & \vdots \\ f[M - 1, 0] & f[M - 1, 1] & \dots & f[M - 1, N - 1] \end{bmatrix}$$



Colour spaces: RGB



Default colour space



Some drawbacks

- Strongly correlated channels
- Non-perceptual



R
 $(G=0, B=0)$



G
 $(R=0, B=0)$



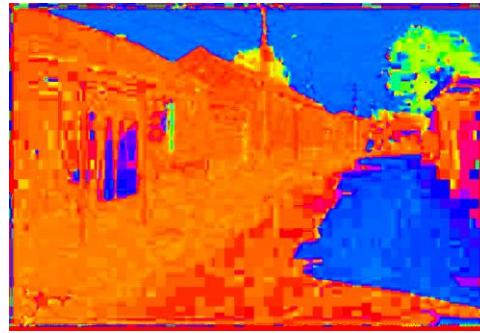
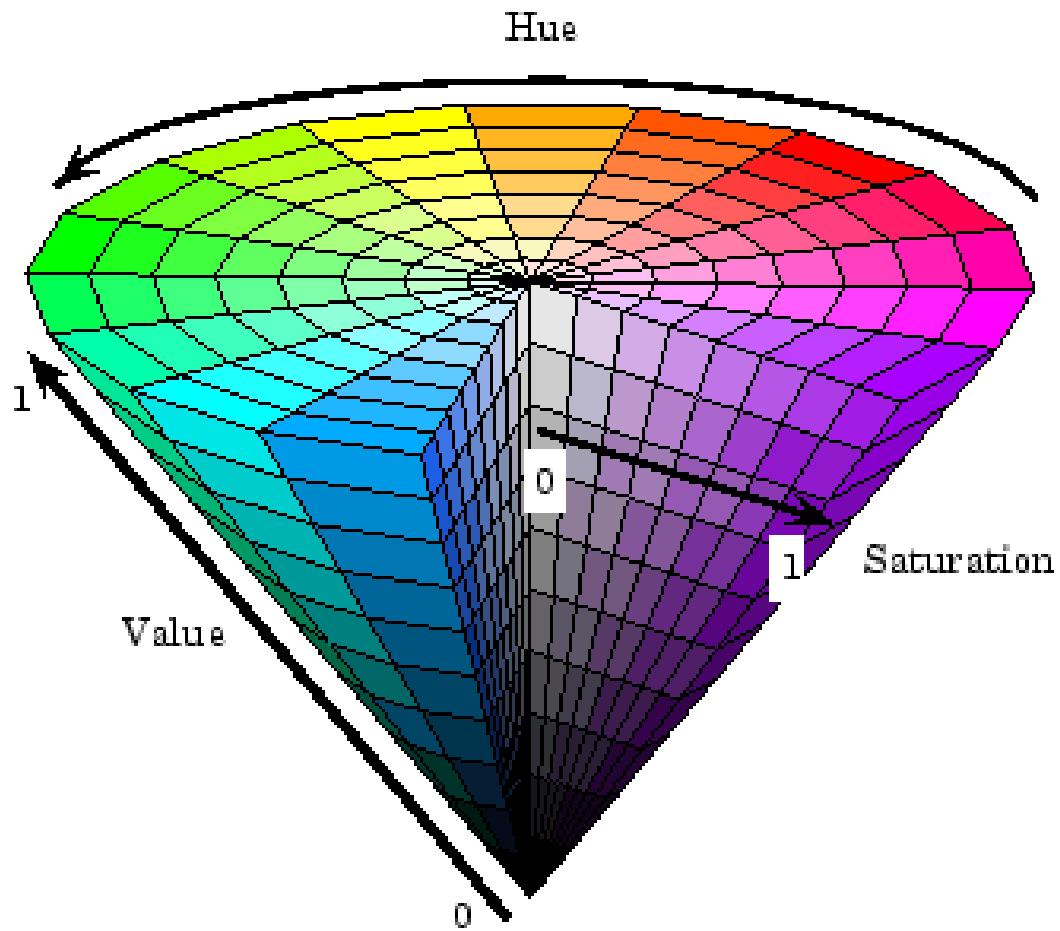
B
 $(R=0, G=0)$

Image from: http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png

Colour spaces: HSV



Intuitive color space



H
($S=1, V=1$)



S
($H=1, V=1$)

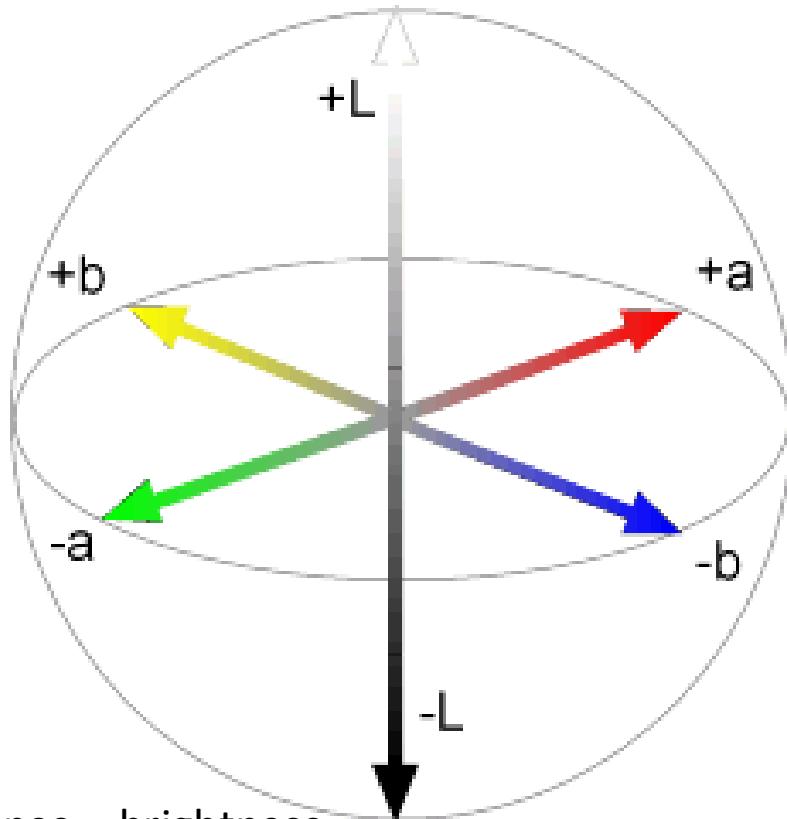


V
($H=1, S=0$)

Colour spaces: CIE L*a*b*



“Perceptually uniform” colour space



Luminance = brightness

Chrominance = colour

Can use Euclidean distances to represent colour differences!



L
($a=0, b=0$)



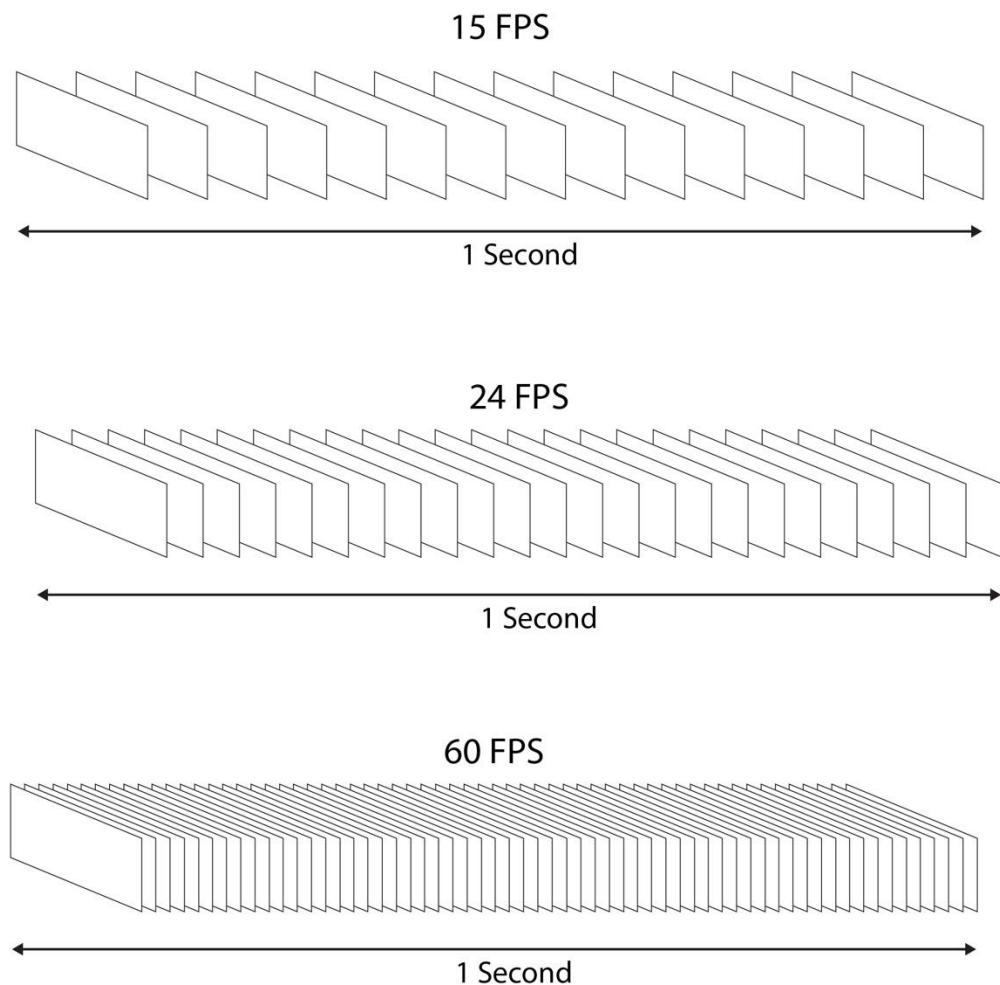
a
($L=65, b=0$)



b
($L=65, a=0$)

How to represent a video?

$$f[x, y, t] = (R, G, B)$$



From quora.com

Summary: Spatial Sampling in Practice

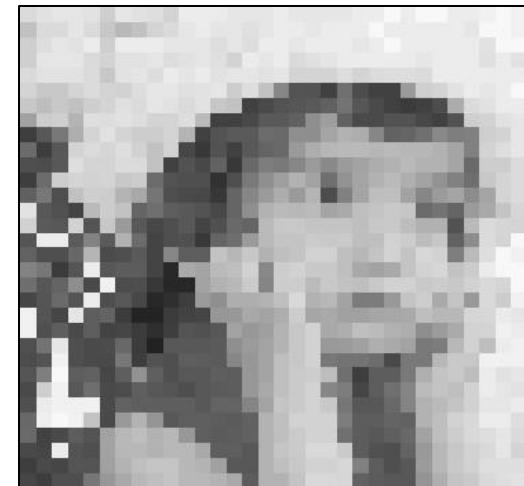
The effect of very sparse sampling ... is often ALIASING



256 x256



64x64



32x32

Anti-aliasing can be achieved by removing spatial frequencies above a critical limit (so-called Shannon-Nyquist Limit).

Summary: Quantization of the Image Function

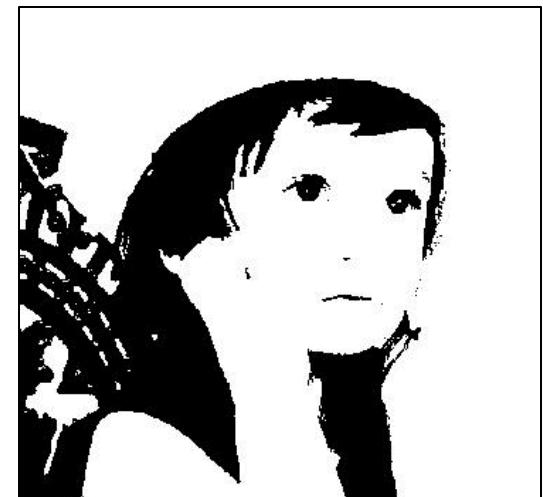
Representing a continuously varying single channel image function $f(x, y)$ with a discrete one using quantization levels:



16 levels



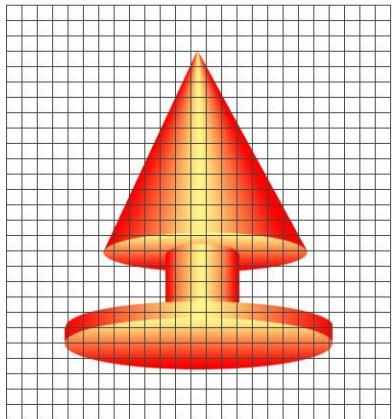
6 levels



2 levels

Summary: Sampling and Quantization

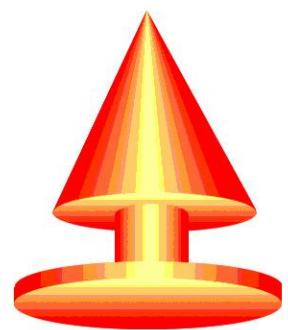
pixel grid



real image



sampled



quantized

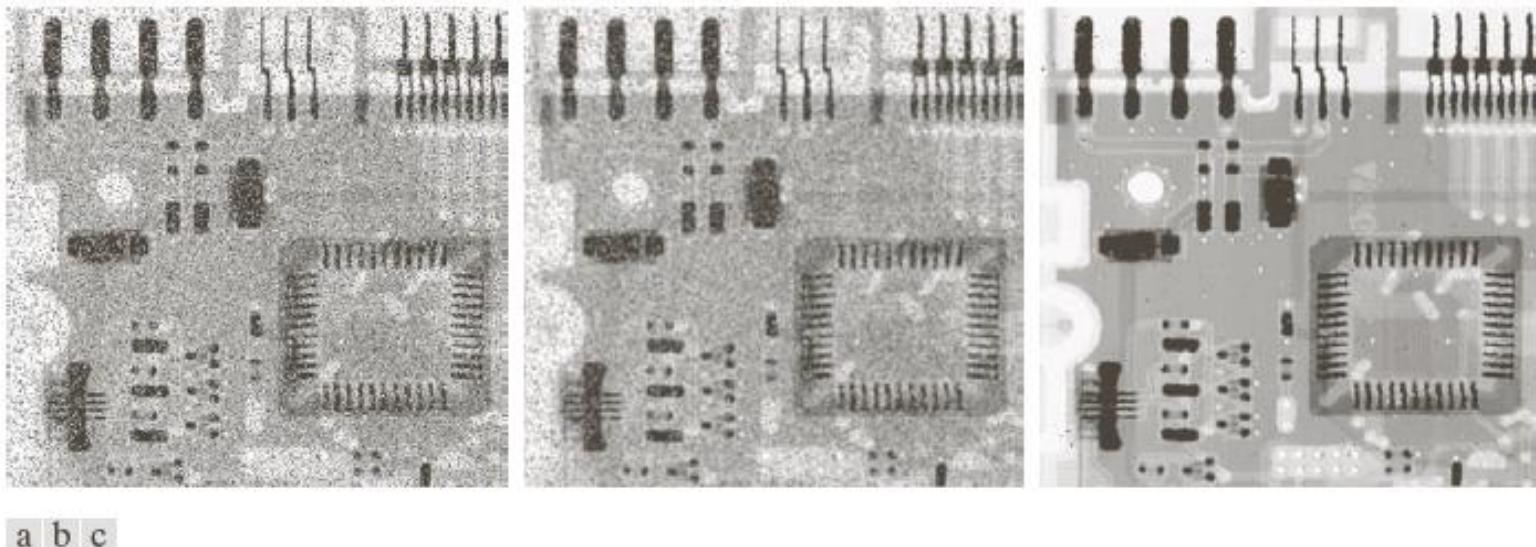


**sampled &
quantized**

Slide by R A Peters

Next Lecture

Filtering



a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

COMS30030 - Image Processing and Computer Vision

Lecture 02

Filtering Images



Majid Mirmehdi | majid@cs.bris.ac.uk

Image Transforms

- **Image Transform** → a *new representation* of the input data (e.g. by re-encoding it in another [parameter] space, e.g. Fourier, Hough, Wavelet, Haar, etc.)
- Image Transforms are classic processing techniques, used in filtering, compression, feature extraction, and much more



Sharpening



Convolution

Convolution = A mathematical operation on two functions (f and g) that produces a third function ($h=f*g$), representing how the shape of one is modified by the other.

(Wikipedia)

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t - x)g(x) \, dt$$

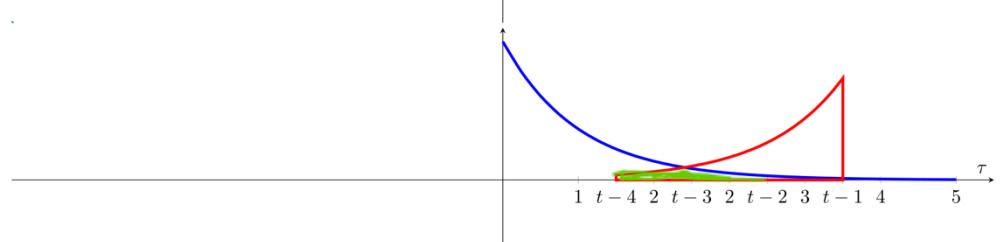
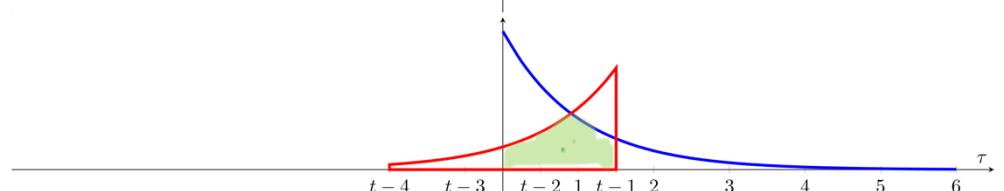
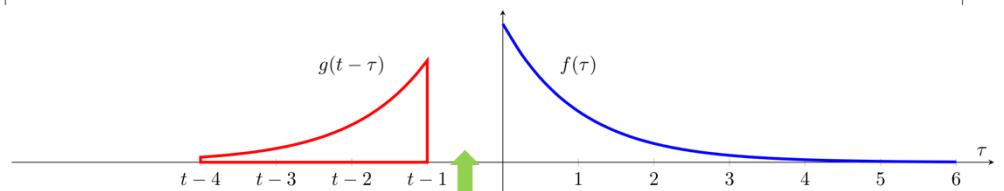
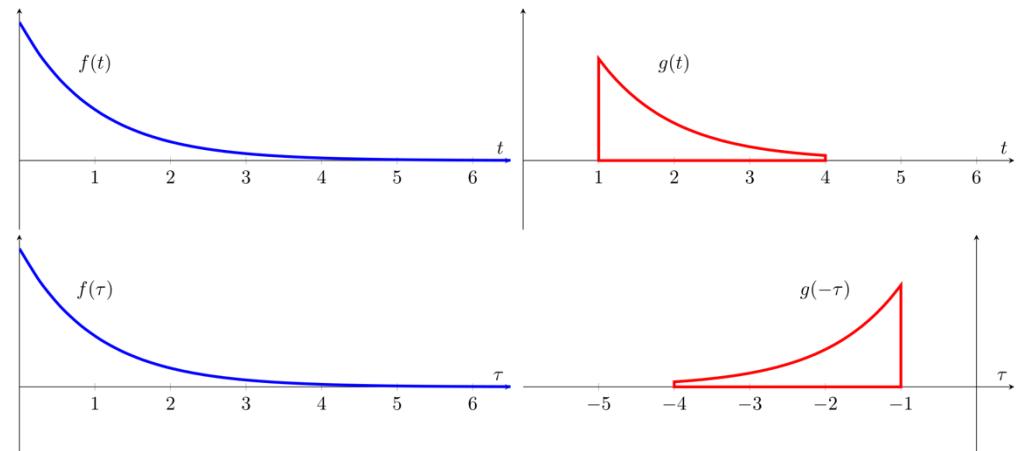
Kernel
Signal

- Used for filtering images in the spatial domain
- Application in other parameter spaces, e.g. frequency domain
- Fundamental to Convolutional Neural Nets for deep Learning
- much, much more

An Intuitive look at Convolution

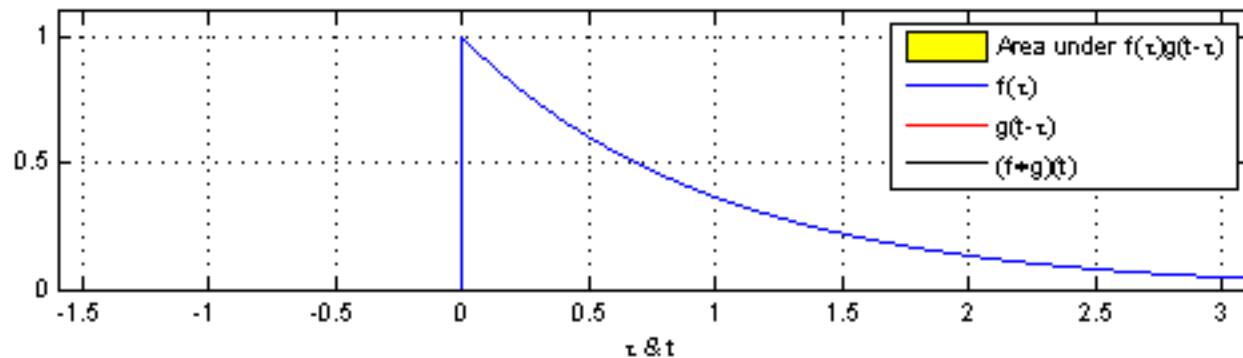
$$f * g = \int_{-\infty}^{\infty} f(t - \tau)g(\tau) \, dt$$

$$f * g = \int_{-\infty}^{\infty} f(\tau)g(t - \tau) \, dt$$

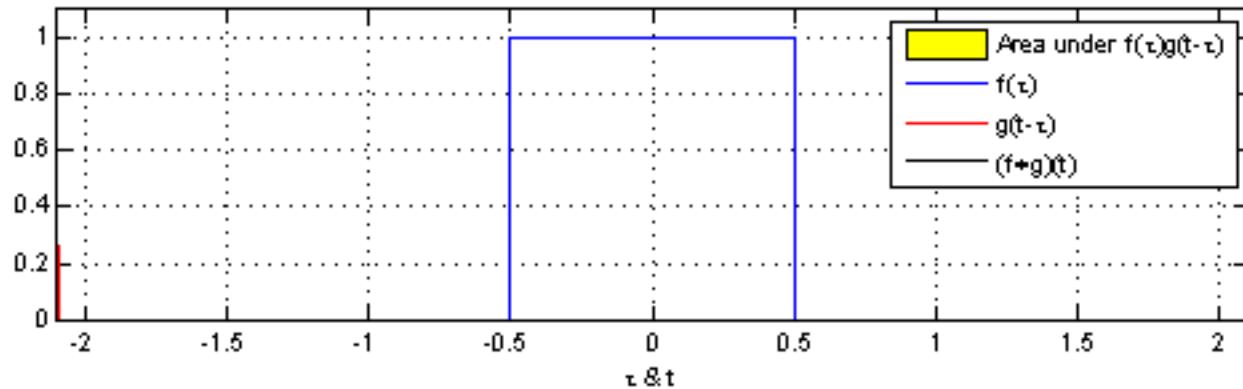


By Krishnavedala

An Intuitive look at Convolution



$$f * g = \int_{-\infty}^{\infty} f(t - \tau)g(\tau) \, dt$$



Autocorrelation

Animations Source: Brian Amberg

1D Discrete Convolution

$$g(x) = h(x) * f(x)$$

➤ f is the signal, h is the convolution filter

➤ h has an origin



Example 1D kernel

➤ Normalization factor (sum of the absolute values of the filter) is also part of the filter!

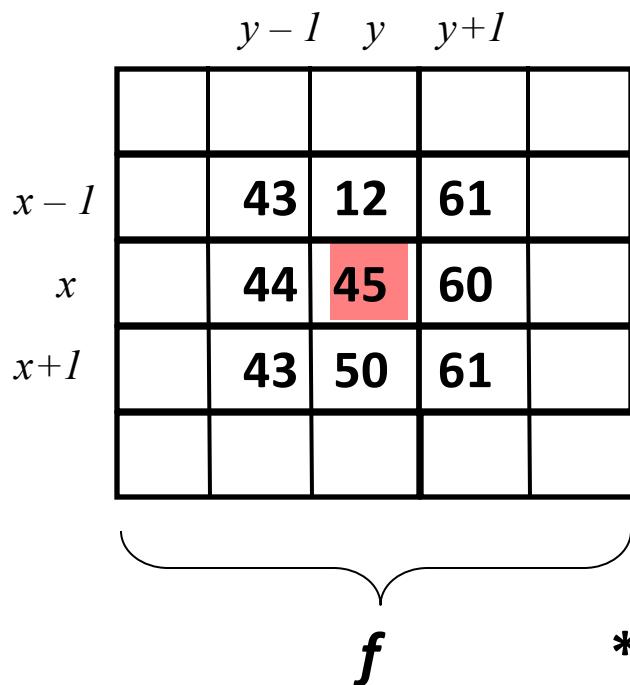
➤ The discrete version of convolution is defined as:

$$g(x) = \sum_{m=-s}^s f(x-m)h(m) \quad \text{for } s \geq 1$$

2D Discrete Convolution

- The discrete version of 2D convolution is defined as:

$$g(x, y) = \sum_m \sum_n f(x - m, y - n)h(m, n)$$



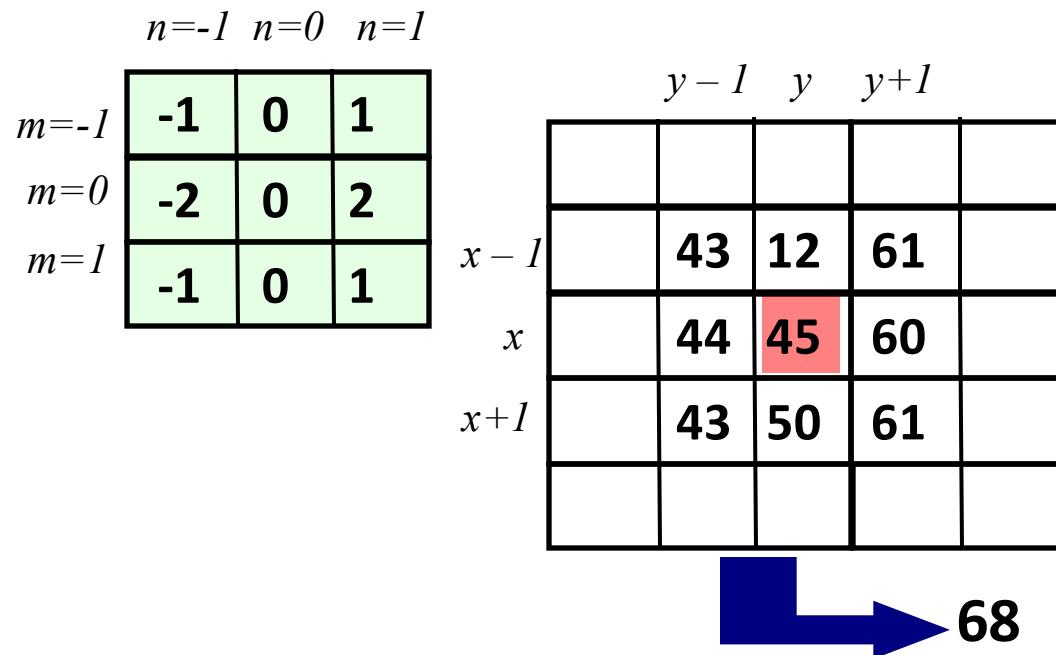
$$\begin{matrix} & n=-1 & n=0 & n=1 \\ m=-1 & -1 & 0 & 1 \\ m=0 & -2 & 0 & 2 \\ m=1 & -1 & 0 & 1 \end{matrix}$$

$$\begin{aligned} & f(x+1, y+1)h(-1, -1) \\ & + f(x+1, y)h(-1, 0) \\ & + f(x+1, y-1)h(-1, 1) \\ & + f(x, y+1)h(0, -1) \\ & + f(x, y)h(0, 0) \\ & + f(x, y-1)h(0, 1) \\ & + f(x-1, y+1)h(1, -1) \\ & + f(x-1, y)h(1, 0) \\ & + f(x-1, y-1)h(1, 1) \end{aligned}$$

2D Discrete Correlation

- The discrete version of 2D correlation is defined as:

$$g(x, y) = \sum_m \sum_n f(x + m, y + n)h(m, n)$$



Correlation \equiv Convolution
when kernel is symmetric
under 180° rotation, e.g.

1	2	1
1	2	1
1	2	1

Spatial Low/High Pass Filtering

Low Pass

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



High Pass

$$h = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



$$f * h$$

Properties of Convolution

- Commutative:

$$f * h = h * f$$

- Associative

$$(f * g) * h = f * (g * h)$$

- Distributes over addition

$$f * (g + h) = (f * g) + (f * h)$$

- Scalars factor out

$$af * g = f * ag = a(f * g)$$

- Identity:

Given unit impulse $e = [\dots, 0, 0, 1, 0, 0, \dots]$ then $f * e = f$

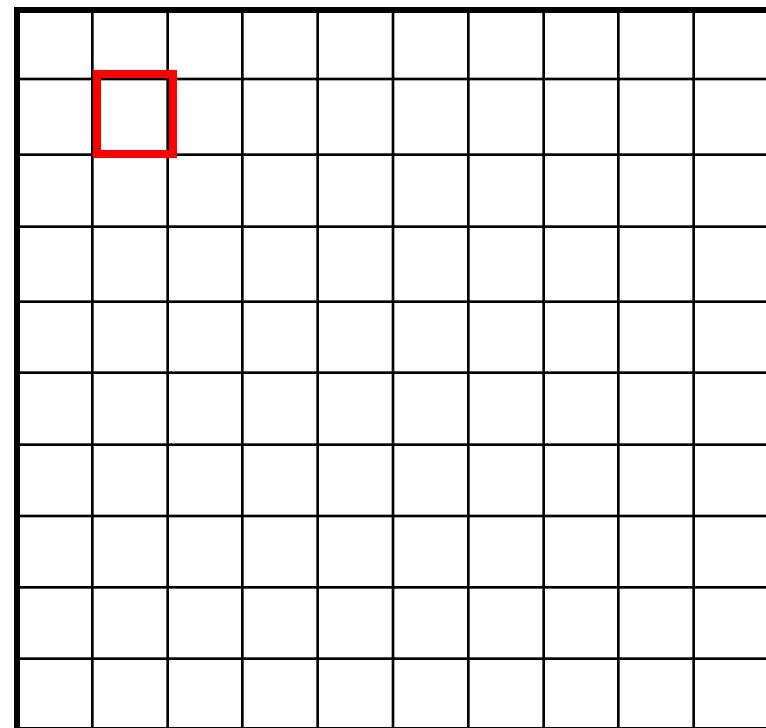
Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$g(x, y)$



$$g(x, y) = \sum_m \sum_n f(x + m, y + n) h(m, n)$$

Credit: S. Seitz

Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$g(x, y)$

0	10									

$$g(x, y) = \sum_m \sum_n f(x + m, y + n)h(m, n)$$

Credit: S. Seitz

Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	90	0
0	0	0	90	90	90	90	90	90	0
0	0	0	90	90	90	90	90	90	0
0	0	0	90	0	90	90	90	90	0
0	0	0	90	90	90	90	90	90	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x, y)$

			0	10	20				

$$g(x, y) = \sum_m \sum_n f(x + m, y + n) h(m, n)$$

Credit: S. Seitz

Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$g(x, y)$

			0	10	20	30				

$$g(x, y) = \sum_m \sum_n f(x + m, y + n)h(m, n)$$

Credit: S. Seitz

Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x, y)$

$$g(x, y) = \sum_m \sum_n f(x + m, y + n) h(m, n)$$

Credit: S. Seitz

Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$g(x, y)$

			0	10	20	30	30			

$$g(x, y) = \sum_m \sum_n f(x + m, y + n) h(m, n)$$

Credit: S. Seitz

Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	90	0	90	90	90	0	0	0
0	0	0	90	90	90	90	90	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

$g(x, y)$

	0	10	20	30	30					

$$g(x, y) = \sum_m \sum_n f(x + m, y + n)h(m, n)$$

Credit: S. Seitz

Image filtering example

$$h(m, n) = \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$f(x, y)$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$g(x, y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$$g(x, y) = \sum_m \sum_n f(x + m, y + n) h(m, n)$$

Credit: S. Seitz

Normalising the Convolution Result!

The weights will affect pixel values and the result could fall outside the 0-255 range → Normalise them such that they sum to 1.

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Smooth

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

3x3 Gaussian Blur

$$\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Unsharp Masking

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

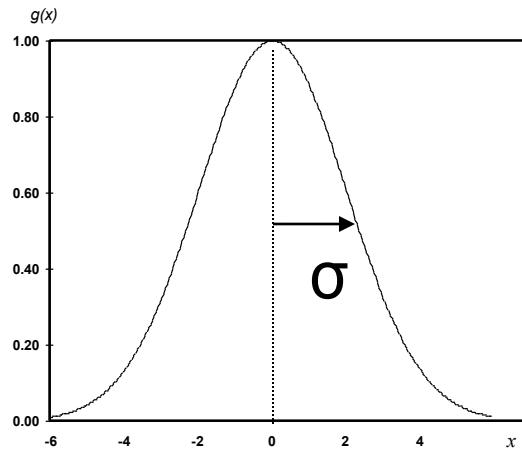
Sharpen

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

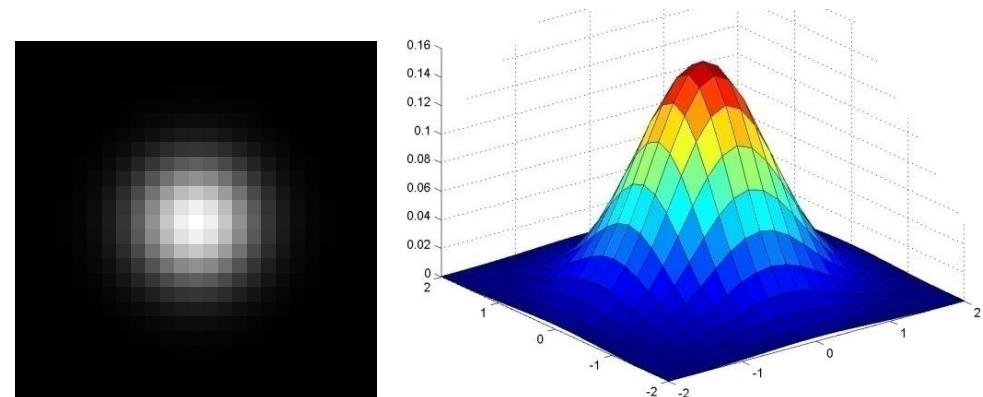
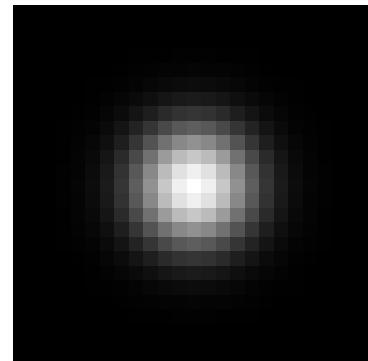
Basic Edge Detector

Gaussian Filter Kernel

The *Gaussian* filter is very commonly used in image processing. The parameter σ determines the width of the filter and hence the amount of smoothing.

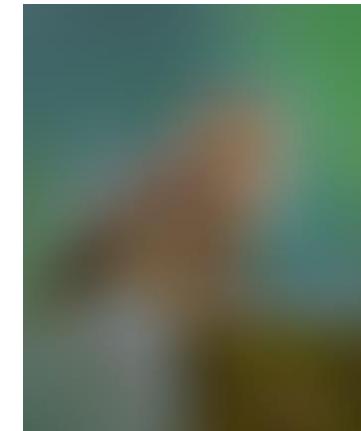
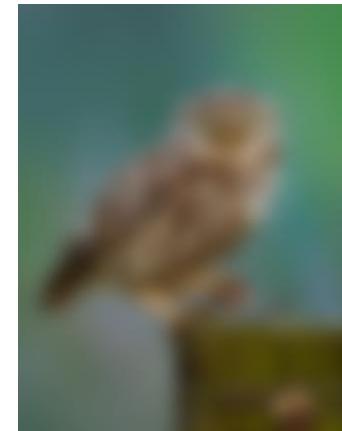


$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

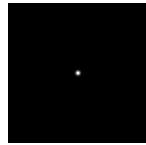


$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

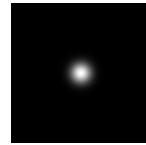
Applying the Gaussian filter



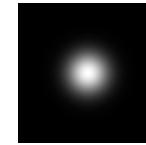
Original



$\sigma = 1$



$\sigma = 5$



$\sigma = 10$

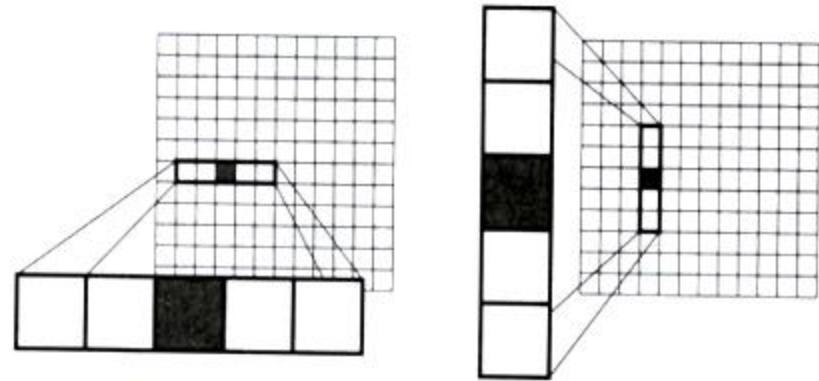


$\sigma = 30$

Source: Noah Snavely

Efficient Gaussian Filtering

Filtering with a 2D Gaussian can be achieved faster using two 1D Gaussian filters! This is because the linear Gaussian kernel is *separable*.



$$K = \frac{1}{256} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix} = \frac{1}{16} \begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 1 \end{pmatrix} x \frac{1}{16} (1 \quad 4 \quad 6 \quad 4 \quad 1)$$

Filtering an $m \times m$ image with an $n \times n$ kernel is $\mathcal{O}(m^2n^2)$ complexity, but with a separable kernel, it would be $\mathcal{O}(m^2n)$.
→ a significant reduction in computations!

Sources from fiveko.com and E.G.M. Petrakis

Separability of the Gaussian Filter

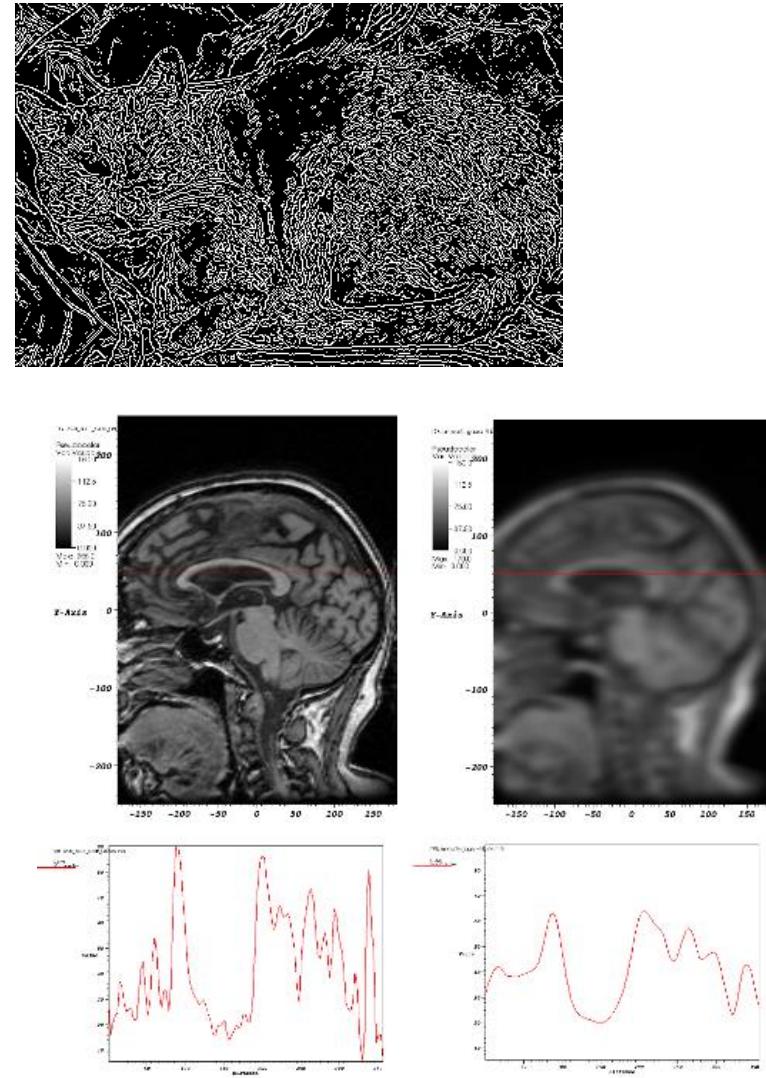
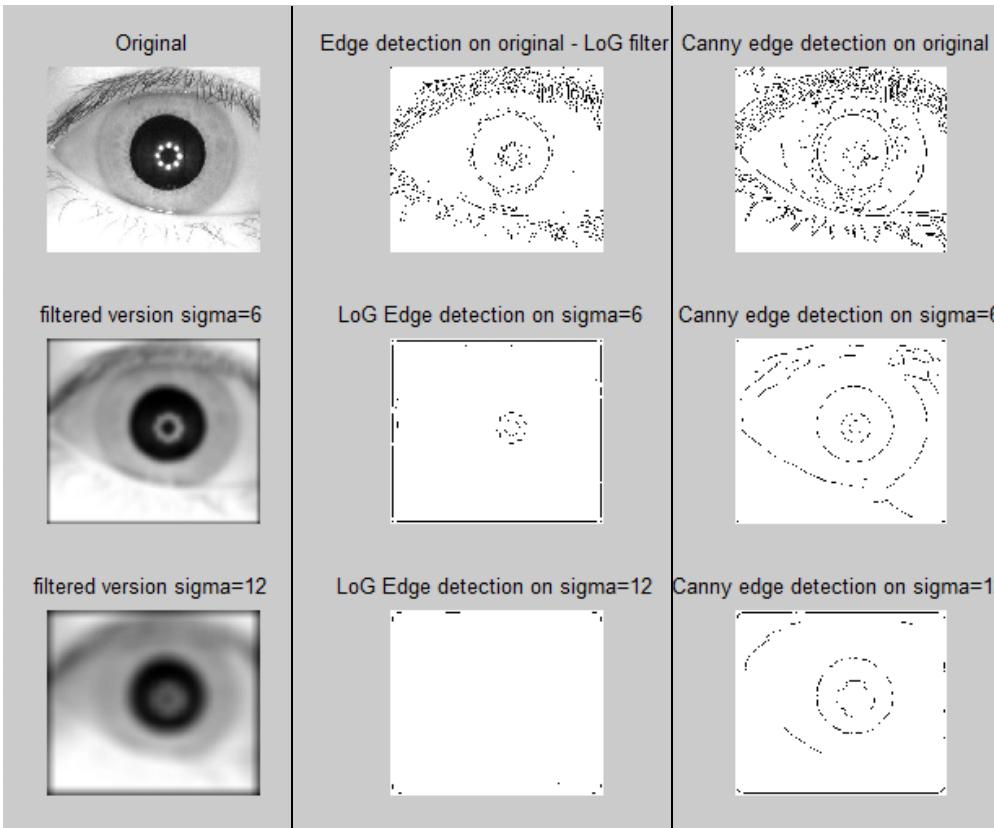
Separability means that a 2D convolution can be reduced to the product of two 1D convolutions - one on the rows and one on the columns:

$$\begin{aligned} g(x, y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \right) \\ &= g(x) g(y) \end{aligned}$$

Sources from fiveko.com and E.G.M. Petrakis

Example benefit of the Gaussian Filter

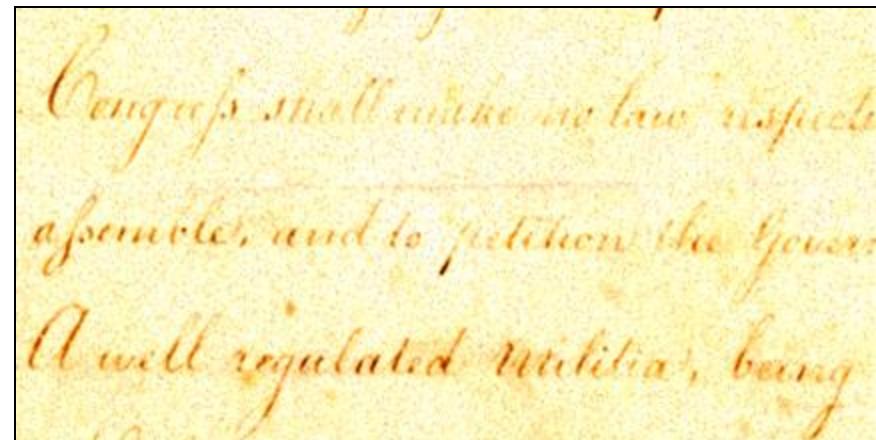
When we smooth an image, the easier the next stage of processing, in this case edge detection



Noise Removal: The Median Filter

- Returns the median value of the pixels in a neighborhood
- Relationship to a uniform blurring filter?
- Is non-linear

original



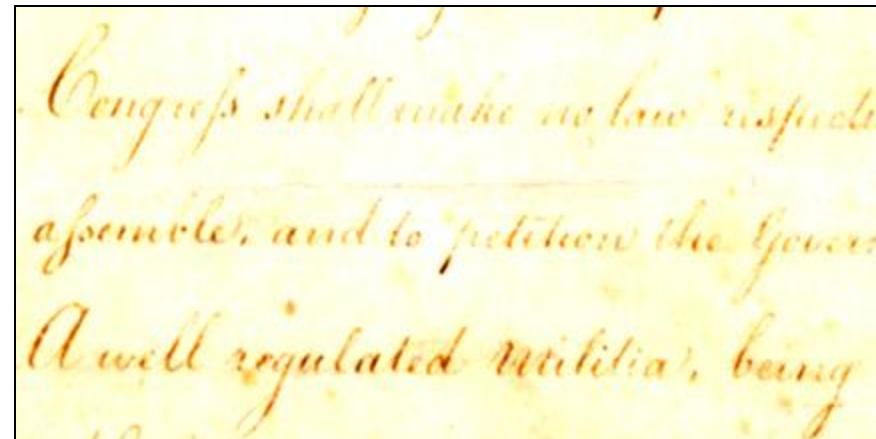
123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

Neighbourhood values:

115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

median filtered



Slide adapted from Richard Peters

!!

Noise Removal: The Median Filter

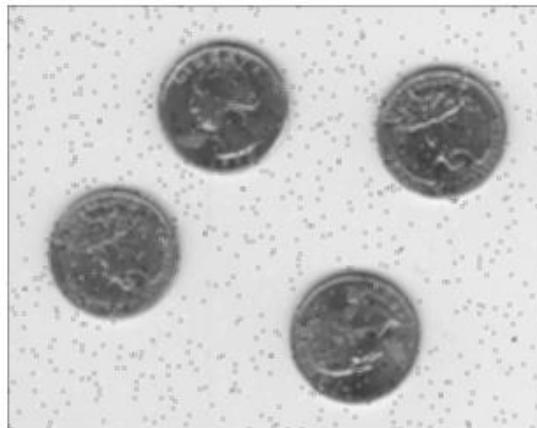
Original



salt & pepper
noise added



3x3
averaging
filter



3x3
median
filter



Noise Removal: The Median Filter

original



corrupted (*i.e.* $p=5\%$ that
a bit is flipped)



median filtered

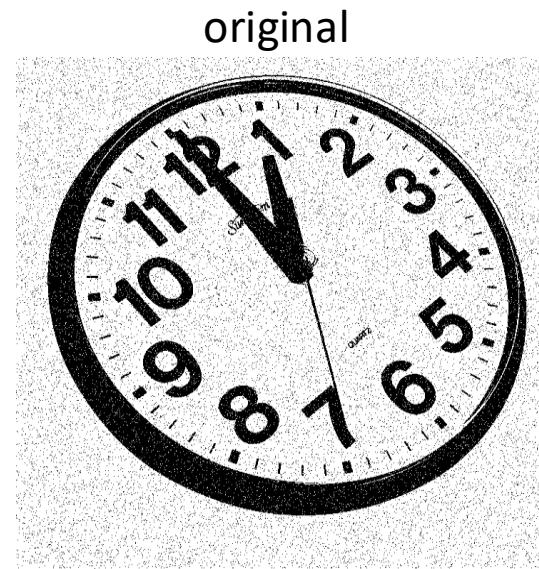


Slide adapted from Richard Peters

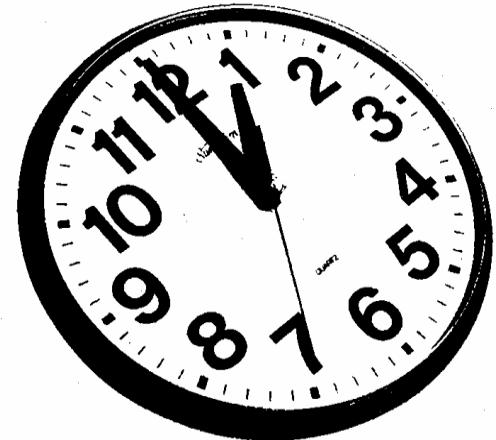
!!

Median Filter: the algorithm

1. Let I be a monochrome image.
2. Let Z define a neighborhood of arbitrary shape.
3. At each pixel location, $\mathbf{p}=(x,y)$, in I ...
4. ... select the n pixels in the Z -neighborhood of \mathbf{p} ,
5. ... sort the n pixels in the neighborhood of \mathbf{p} by value into a list $L(j)$ for $j = 1, \dots, n$.
6. The output value at \mathbf{p} is $L(m)$, where $m = \lfloor n/2 \rfloor + 1$.



median filtered



Slide by Richard Alan Peters II

Sharpening Revisited

$$h(m, n)$$

0	-1	0
-1	4	-1
0	-1	0

Sharpening spatial filters seek to highlight fine detail

- Remove blurring from images
- Highlight edges

Sharpening filters are based on *spatial differentiation*.



Sharpening

What does blurring take away?



Let's add it back:

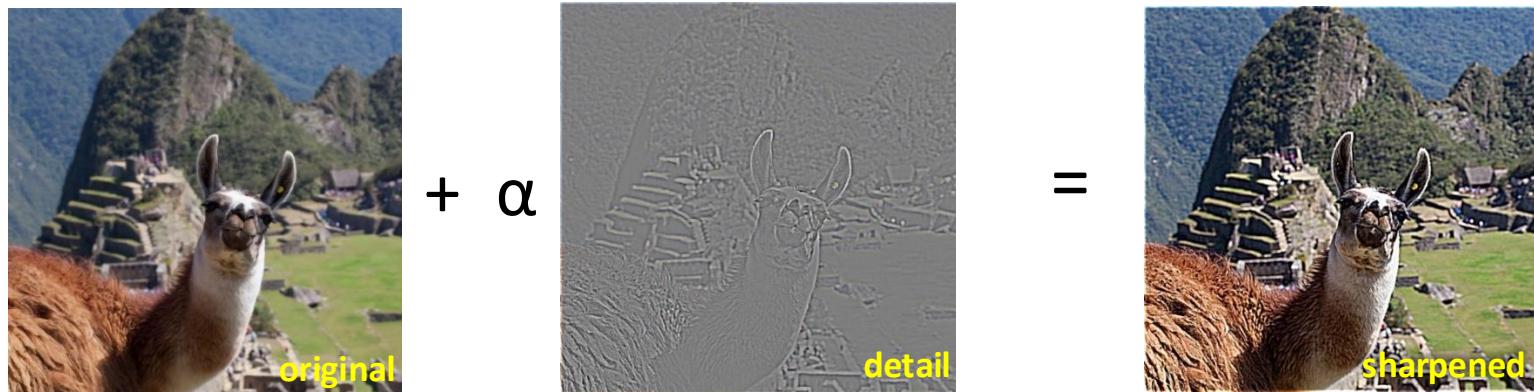


Photo credit: <https://www.flickr.com/photos/geezaweezer/16089096376/>

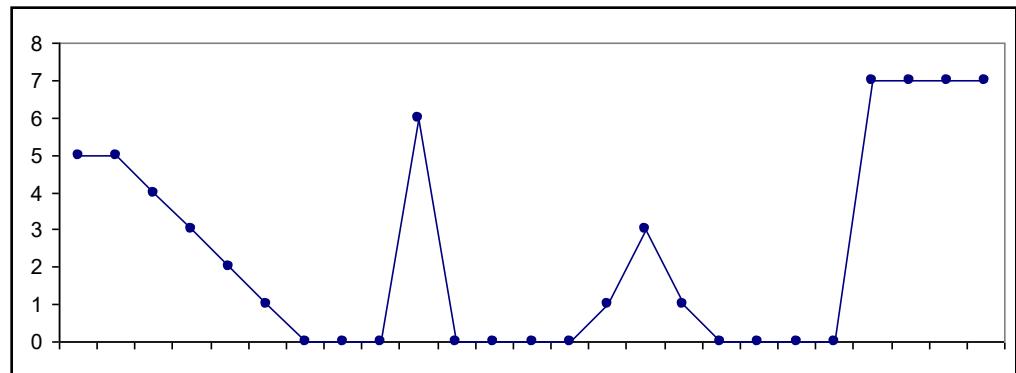
1st Derivative

The 1st derivative of a function is:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

→ it's the difference between subsequent values and measures the rate of change of the function.

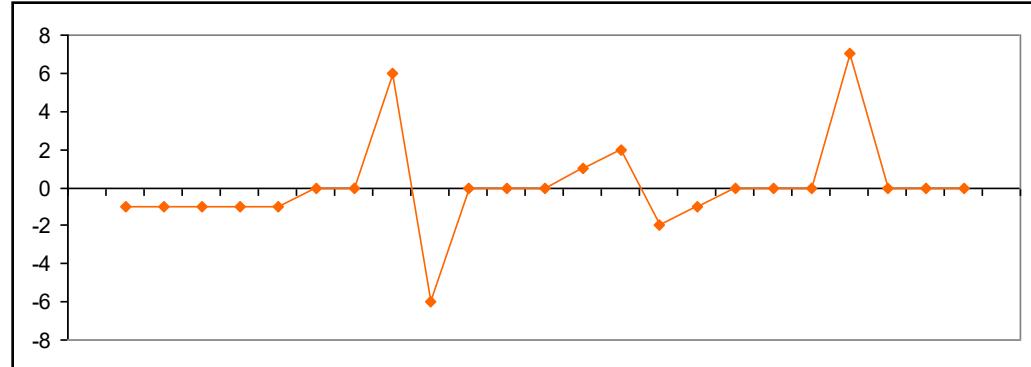
$f(x)$



5 5 4 3 2 1 0 0 0 6 0 0 0 0 1 3 1 0 0 0 0 7 7 7 7

0-1-1-1-1 0 0 6-6 0 0 0 1 2-2-1 0 0 0 7 0 0 0

$$\frac{\partial f}{\partial x}$$



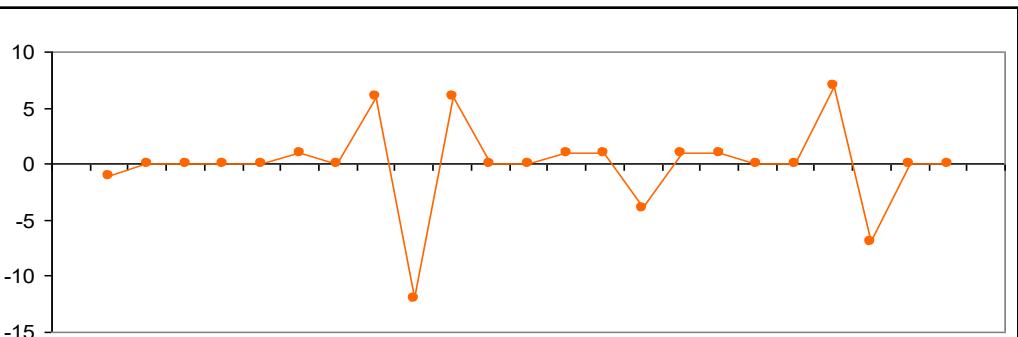
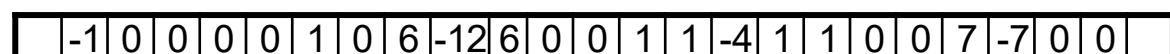
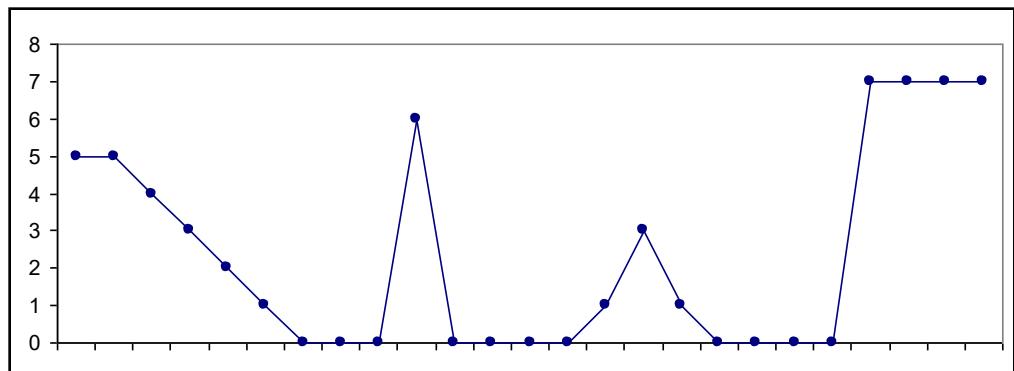
2nd Derivative

The 2nd derivative of a function is:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

→ Takes into account the values both before and after the current value.

$f(x)$



$$\frac{\partial^2 f}{\partial x^2}$$

Using Second Derivatives For Image Enhancement

Now in 2-dimensions for images:

The 2nd derivative is more useful for image sharpening than the 1st derivative

- ✓ Stronger response to fine detail

The *Laplacian Filter*:

- ✓ Isotropic
- ✓ One of the simplest filters for sharpening

0	1	0
1	-4	1
0	1	0

The Laplacian

The Laplacian is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

where the partial 1st order derivative in the x direction is:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

and in the y direction: $\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$

So, the Laplacian can be:

$$\begin{aligned}\nabla^2 f = & [f(x+1, y) + f(x-1, y) \\ & + f(x, y+1) + f(x, y-1)] - 4f(x, y)\end{aligned}$$

0	1	0
1	-4	1
0	1	0

About the Laplacian Operator

As it is a derivative operator:

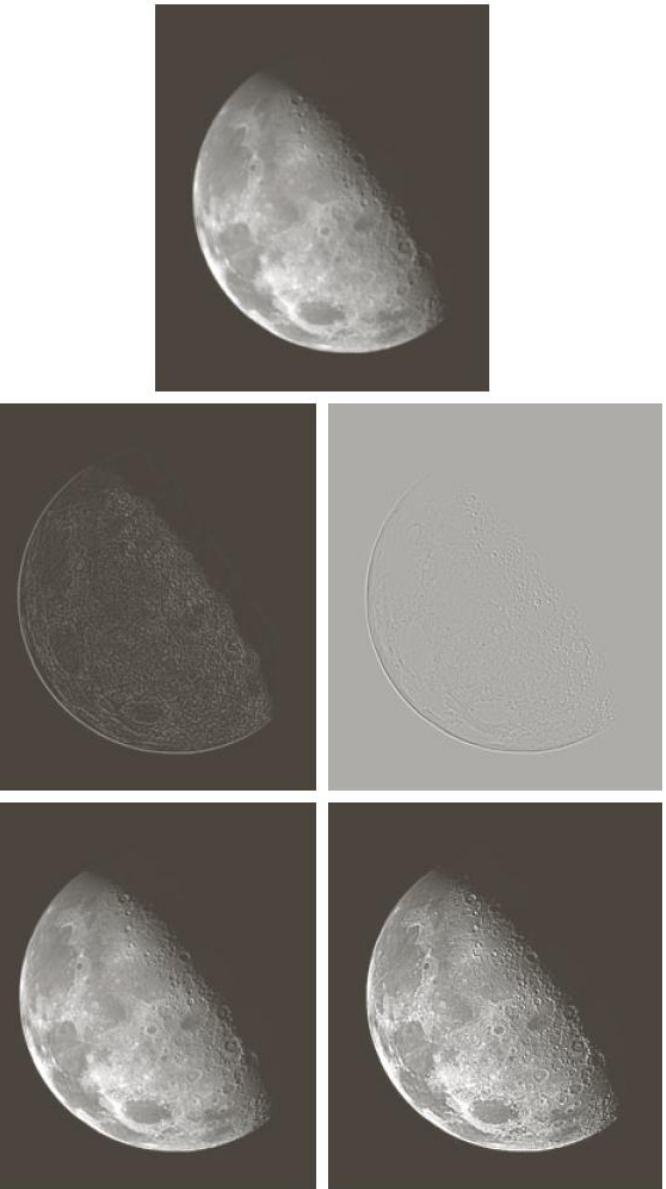
- it highlights gray-level discontinuities in an image.
- it de-emphasizes regions with slowly varying gray-levels.
- Very sensitive to noise as taking derivatives increases noise!

0	1	0
1	-4	1
0	1	0
0	-1	0
-1	4	-1
0	-1	0

1	1	1
1	-8	1
1	1	1
-1	-1	-1
-1	8	-1
-1	-1	-1

a
b c
d e

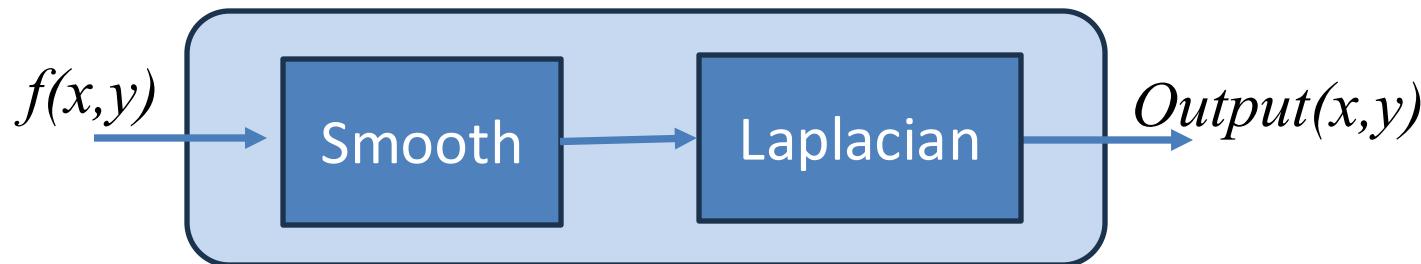
(a) Blurred image of the North Pole.
(b) Laplacian without scaling and (c) with scaling, (d) and (e) Image sharpening using two different masks.



LoG: Laplacian of Gaussian

Because the Laplacian is noise sensitive, it is always combined with a smoothing operation ☺

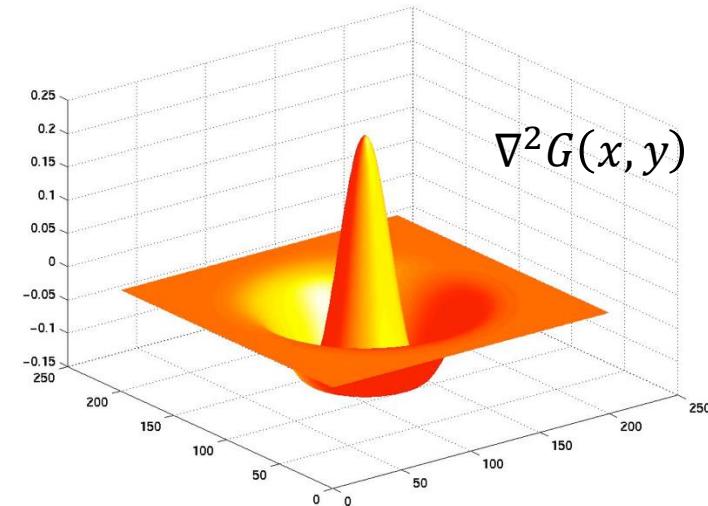
$$Output(x, y) = \nabla^2(f(x, y) * G(x, y))$$



$$\nabla^2(f(x, y) * G(x, y)) = \nabla^2 G(x, y) * f(x, y)$$

Laplacian of a
Gaussian-filtered
image

Laplacian of
Gaussian of a
filtered image



Summary: Image Filtering

- Images are often corrupted by random variations in intensity, illumination, or have poor contrast and cannot be used directly
- *Filtering* allows us to achieve:
 - *Enhancement*: improves contrast
 - *Smoothing*: removes noise
 - *Template matching*: detects known patterns
 - Feature Extraction: provides clues about objects etc., for further analysis
 - Many other uses...

Adapted from E.G.M. Petrakis

COMS30030 - Image Processing and Computer Vision



Lecture 03

Frequency Domain & Transforms

Majid Mirmehdi | majid@cs.bris.ac.uk

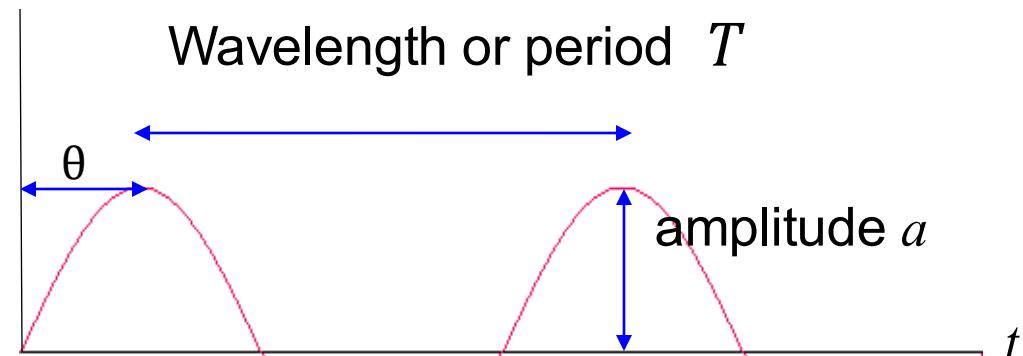
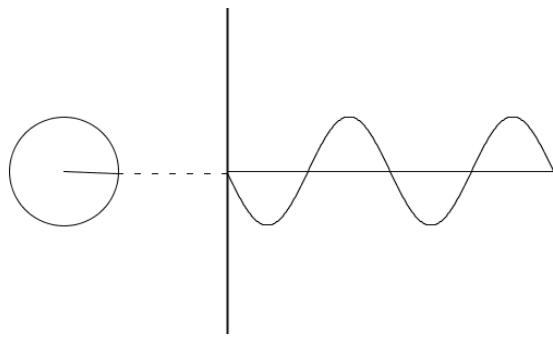
Signals as Functions

Frequency - allows us to characterise signals:

- Repeats over regular intervals with Frequency $\nu = \frac{1}{T}$ cycles/sec (Hz)
- Amplitude a (peak value)
- the Phase θ (shift in degrees)

Example: sine function

$$f(t) = a \sin 2\pi \nu t$$



Fourier's Theorem

$$f(x) = \int a_n \cos(nx) + b_n \sin(nx) \delta n$$

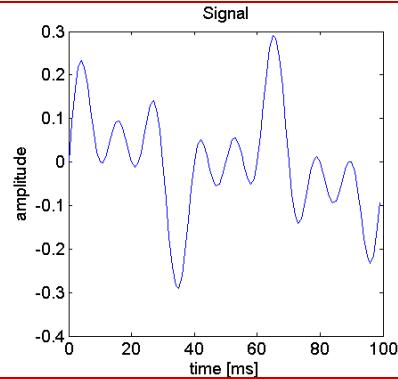


Jean-Baptiste Joseph Fourier

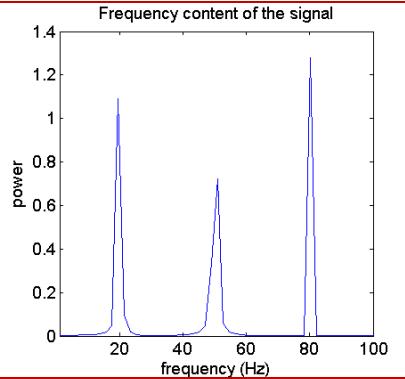
- The sines and cosines are the **Basis Functions** of this representation. a_n and b_n are the **Fourier Coefficients**.
- The sinusoids are harmonically related: each one's frequency is an integer multiple of the fundamental frequency of the input signal.

Intuition I: Simple 1D example

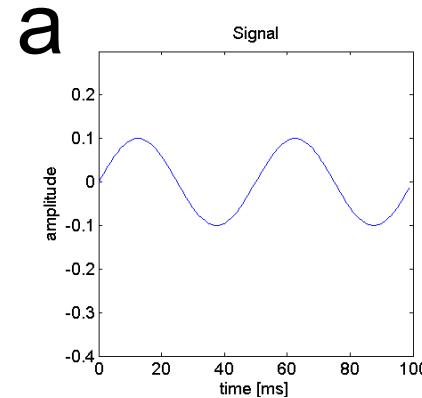
$$d = a + b + c$$



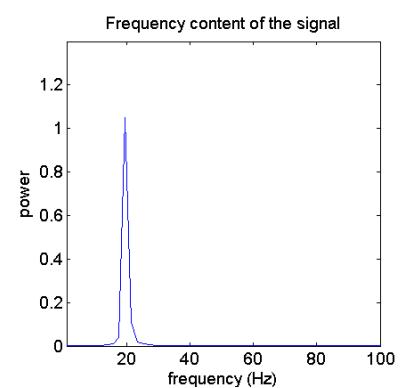
time domain



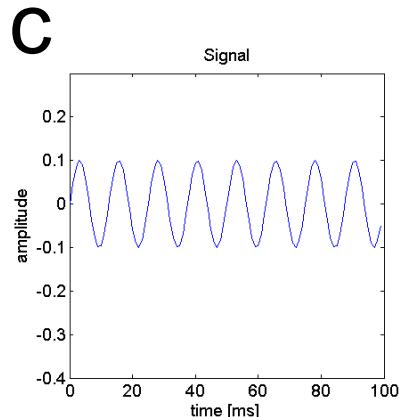
frequency domain



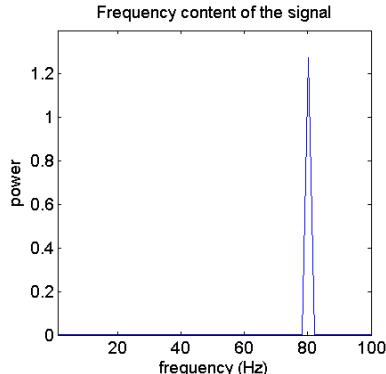
time domain



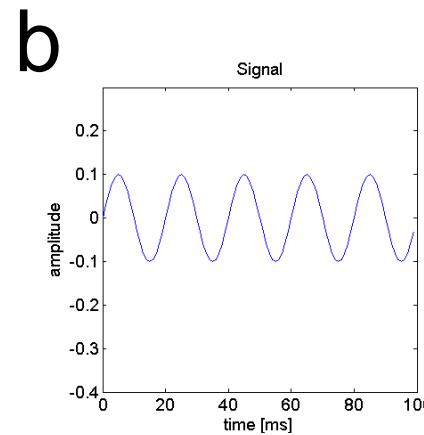
frequency domain



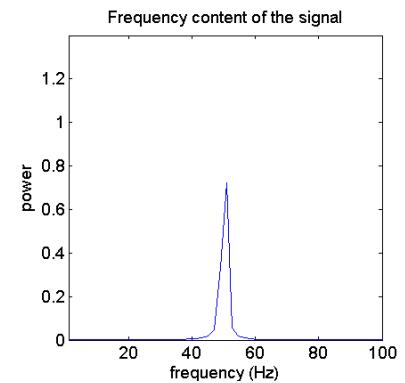
time domain



frequency domain



time domain



frequency domain

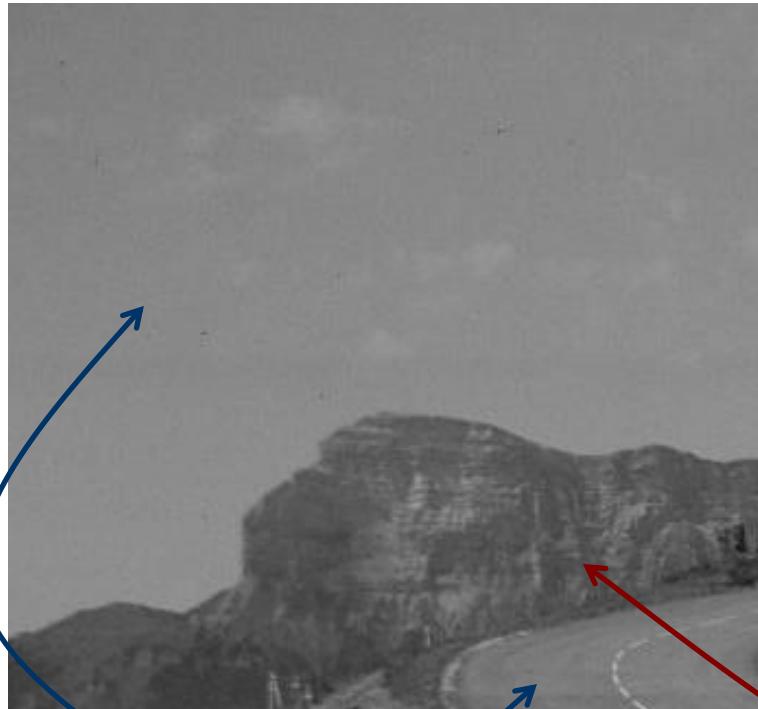
Intuition II: Simple 1D example



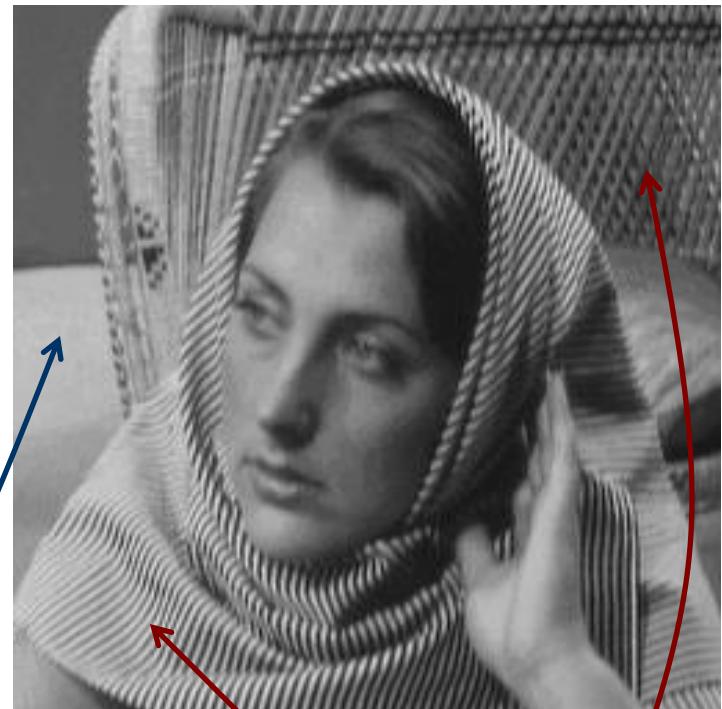
Animation by Lucas V Barbosa

Intuition III: Concept of Frequency in Images

Rate of change of intensity along the two dimensions



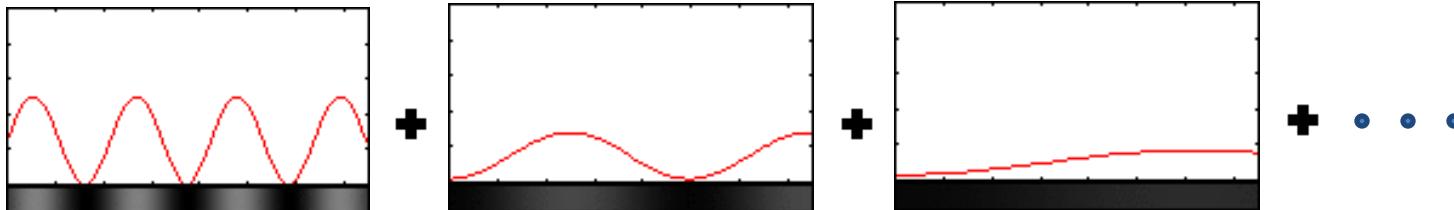
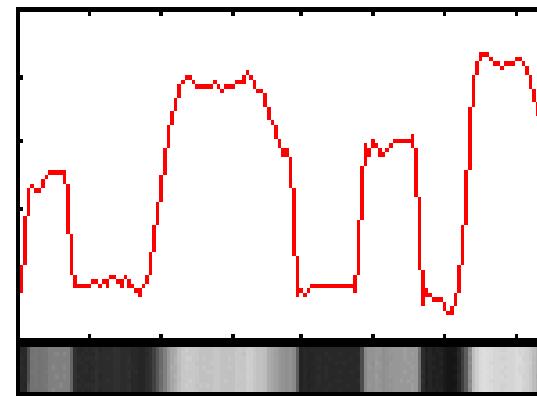
Slowly changing → low frequency



Rapidly changing → high frequency

Intuition IV: Images as waves!?

Take a single row or column of pixels from an image → a 1D signal



From ImageNagik

2D Fourier Transform: Continuous Form

- The Fourier Transform of a continuous function of two variables $f(x,y)$ is:

$$F(u, v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-i2\pi(ux+vy)} dx dy$$

- Conversely, given $F(u, v)$, we can obtain $f(x, y)$ by means of the *inverse Fourier Transform*:

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{i2\pi(ux+vy)} du dv$$

These two equations are also known as the Fourier Transform Pair.
Note, they constitute a lossless representation of data.

2D Fourier Transform: Discrete Form

- The FT of a discrete function of two variables, $f(x,y)$, is:

$$F(u, v) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi(\frac{ux+vy}{N})}$$

image kernels (probing functions)

for $u, v = 0, 1, 2, \dots, N - 1$.

- Conversely, given $F(u,v)$, we can obtain $f(x,y)$ by means of the *inverse FT*:

$$f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) e^{i2\pi(\frac{ux+vy}{N})}$$

for $x, y = 0, 1, 2, \dots, N - 1$.

These two equations are also known as the Fourier Transform Pair.

Note, they constitute a lossless representation of data.

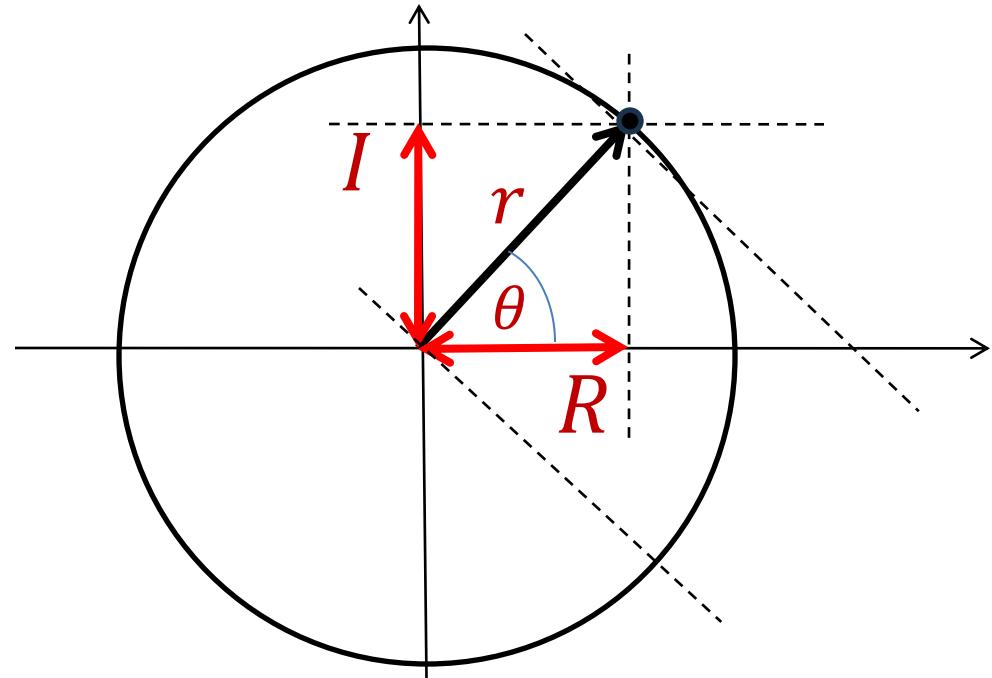
Euler's Formula

$$e^{i2\pi(\frac{ux+vy}{N})}$$



$$e^{i\theta} = \cos \theta + i \sin \theta$$

The kernel is associated with a complex number (r, θ) in polar coordinates or $R(u, v), I(u, v)$ in standard complex notation.



2D Fourier Transforms

- Euler's Formula:

$$e^{i\theta} = \cos \theta + i \sin \theta$$

- Thus, each term of the Fourier Transform is composed of the sum of all values of the image function $f(x,y)$ multiplied by a particular kernel at a particular frequency and orientation specified by (u,v) :

$$F(u,v) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \left[\cos\left(\frac{2\pi(ux + vy)}{N}\right) - i \sin\left(\frac{2\pi(ux + vy)}{N}\right) \right]$$

for $u, v = 0, 1, 2, \dots, N-1$.

All kernels together form a new orthogonal basis for our image.

2D Fourier Transforms

- Euler's Formula:

$$e^{i\theta} = \cos \theta + i \sin \theta$$

- Thus, each term of the Fourier Transform is composed of the sum of all values of the image function $f(x,y)$ multiplied by a particular kernel at a particular frequency and orientation specified by (u,v) :

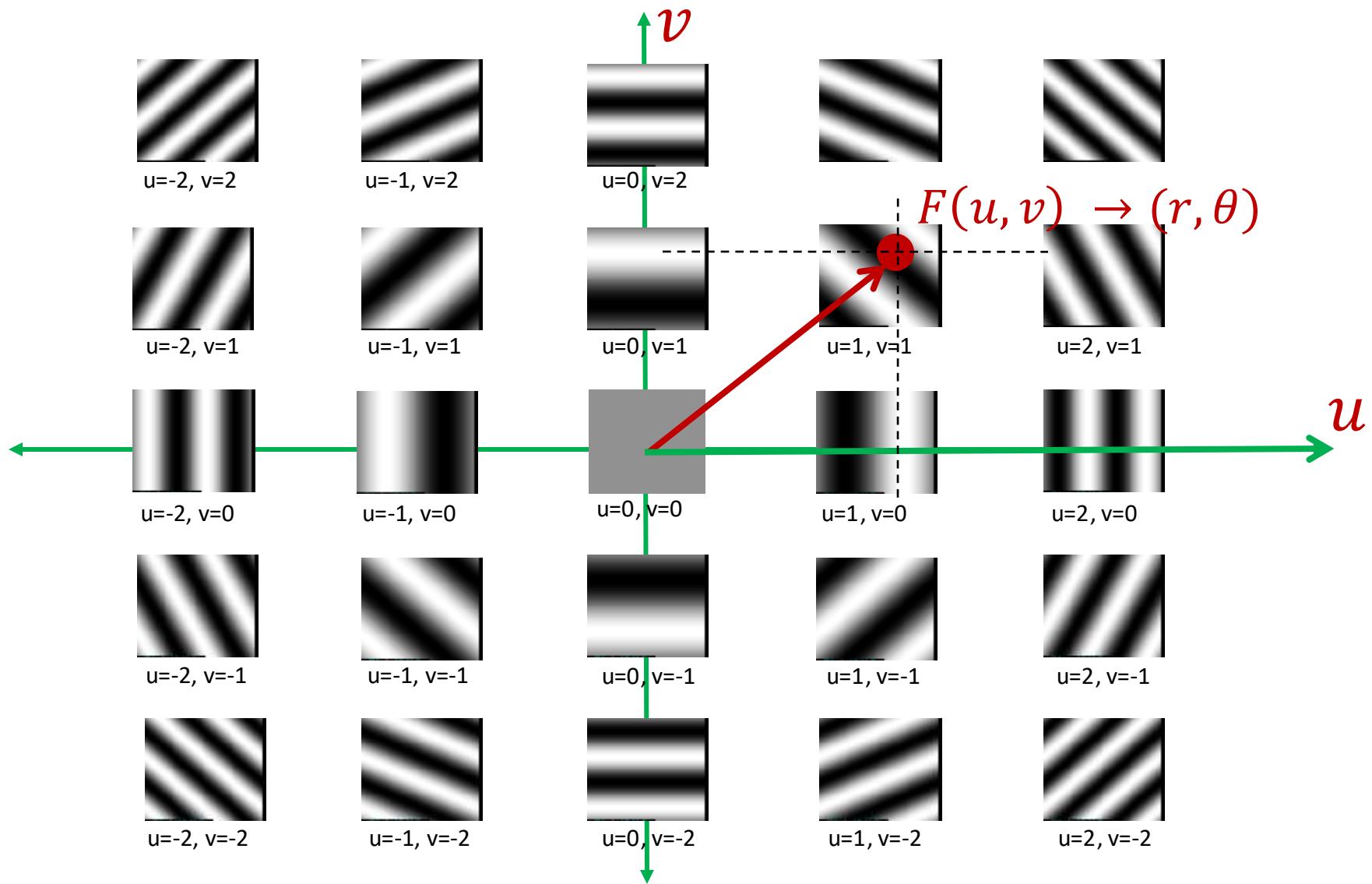
$$F(u, v) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \left[\cos\left(\frac{2\pi(ux + vy)}{N}\right) - i \sin\left(\frac{2\pi(ux + vy)}{N}\right) \right]$$

for $u, v = 0, 1, 2, \dots, N - 1$.

The slowest varying frequency component, i.e.
when $u=0, v=0 \rightarrow$ average image graylevel

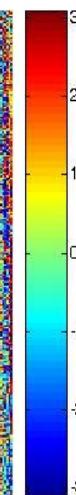
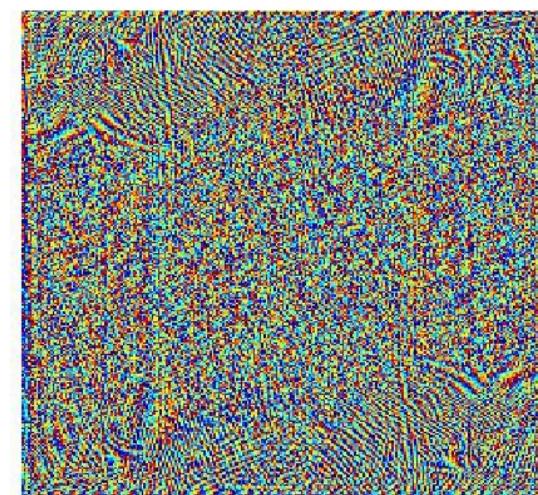
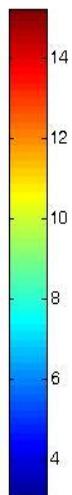
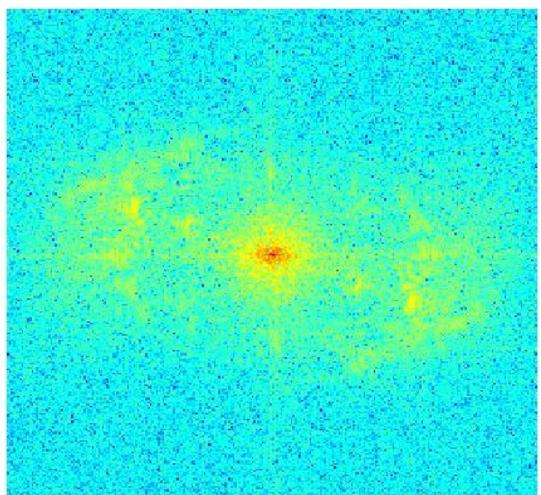
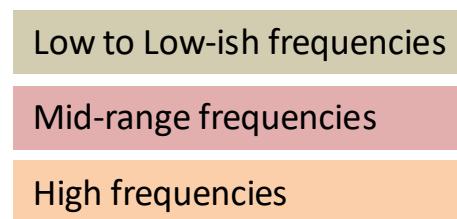
All kernels together form a new orthogonal basis for our image.

'Fabric' of the 2D Fourier Space (as kernels)



Power Spectrum and Phase Spectrum

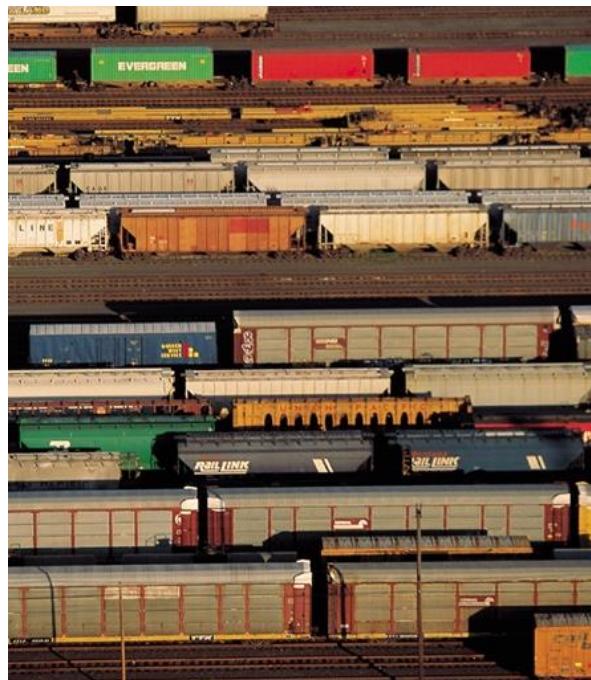
$f(x, y)$



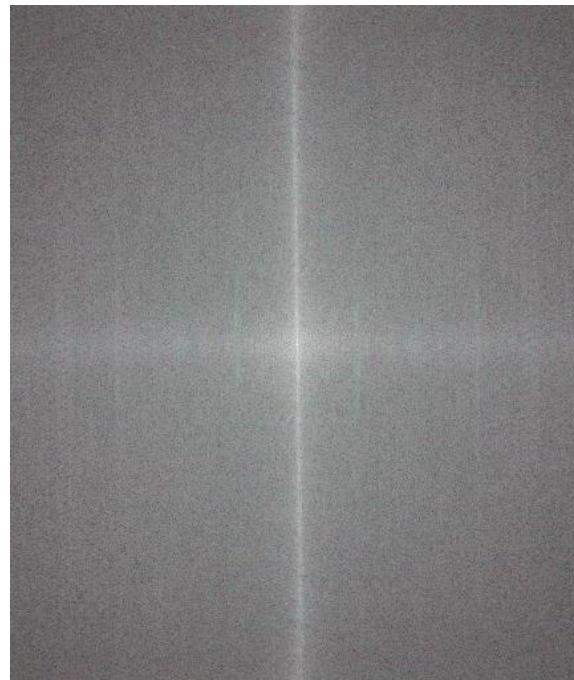
$$|F(u, v)| = \sqrt{R^2(u, v) + I^2(u, v)}$$

$$\theta(u, v) = \tan^{-1} [I(u, v)/R(u, v)]$$

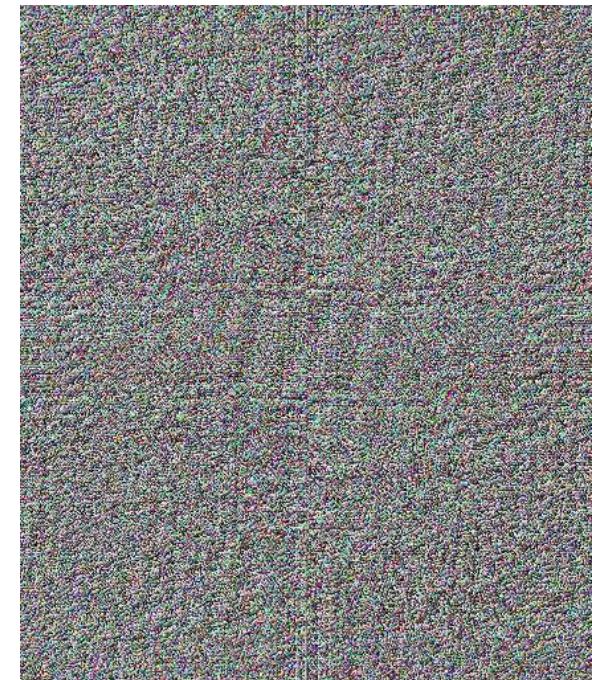
Example: Relating Frequencies to Images



Magnitude



$$|F(u, v)|$$



$$\angle F(I)$$

The Frequency Domain

- $F(u,v)$ is a complex number and has real and imaginary parts:

$$F(u,v) = R(u,v) + iI(u,v)$$

- Magnitudes
(forming the Magnitude Spectrum):

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)}$$

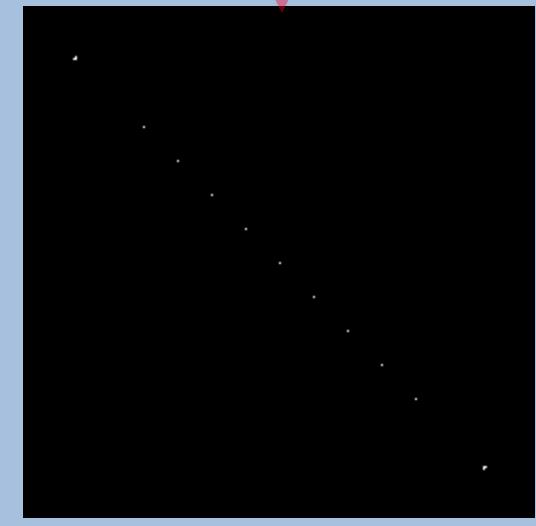
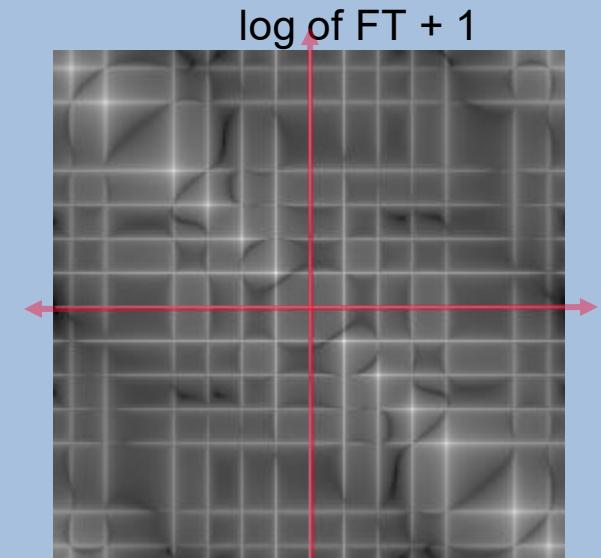
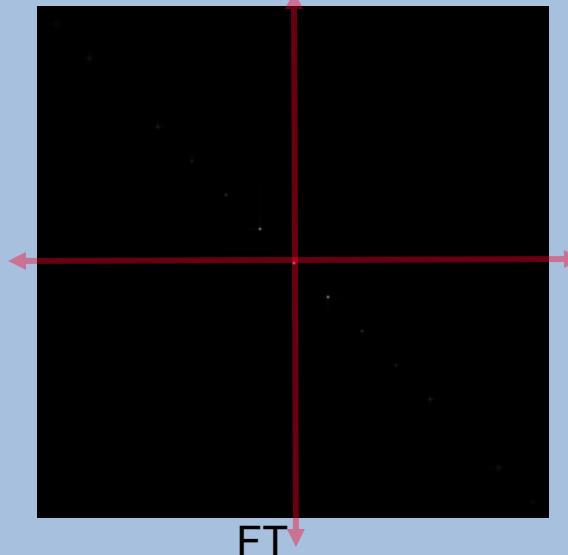
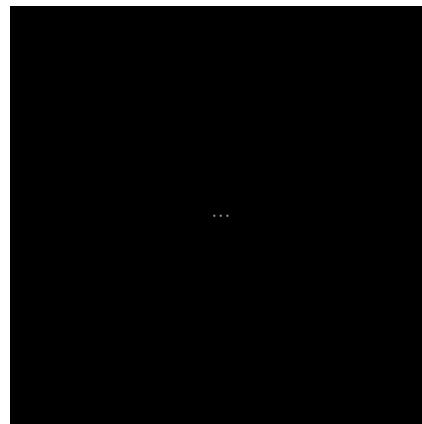
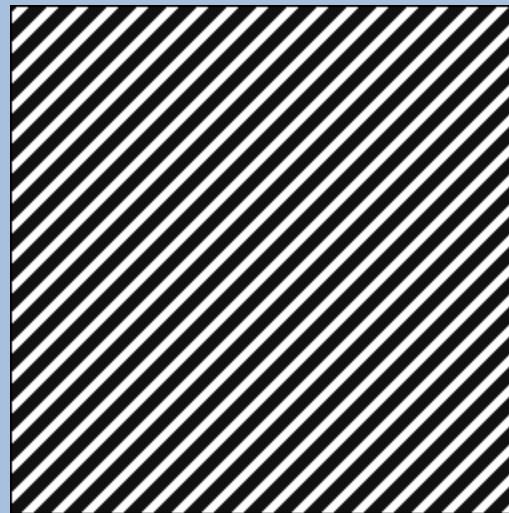
- Phase Angles
(forming the Phase Spectrum):

$$\theta(u,v) = \tan^{-1} [I(u,v)/R(u,v)]$$

- Expressing $F(u,v)$ in polar coordinates (r, θ) :

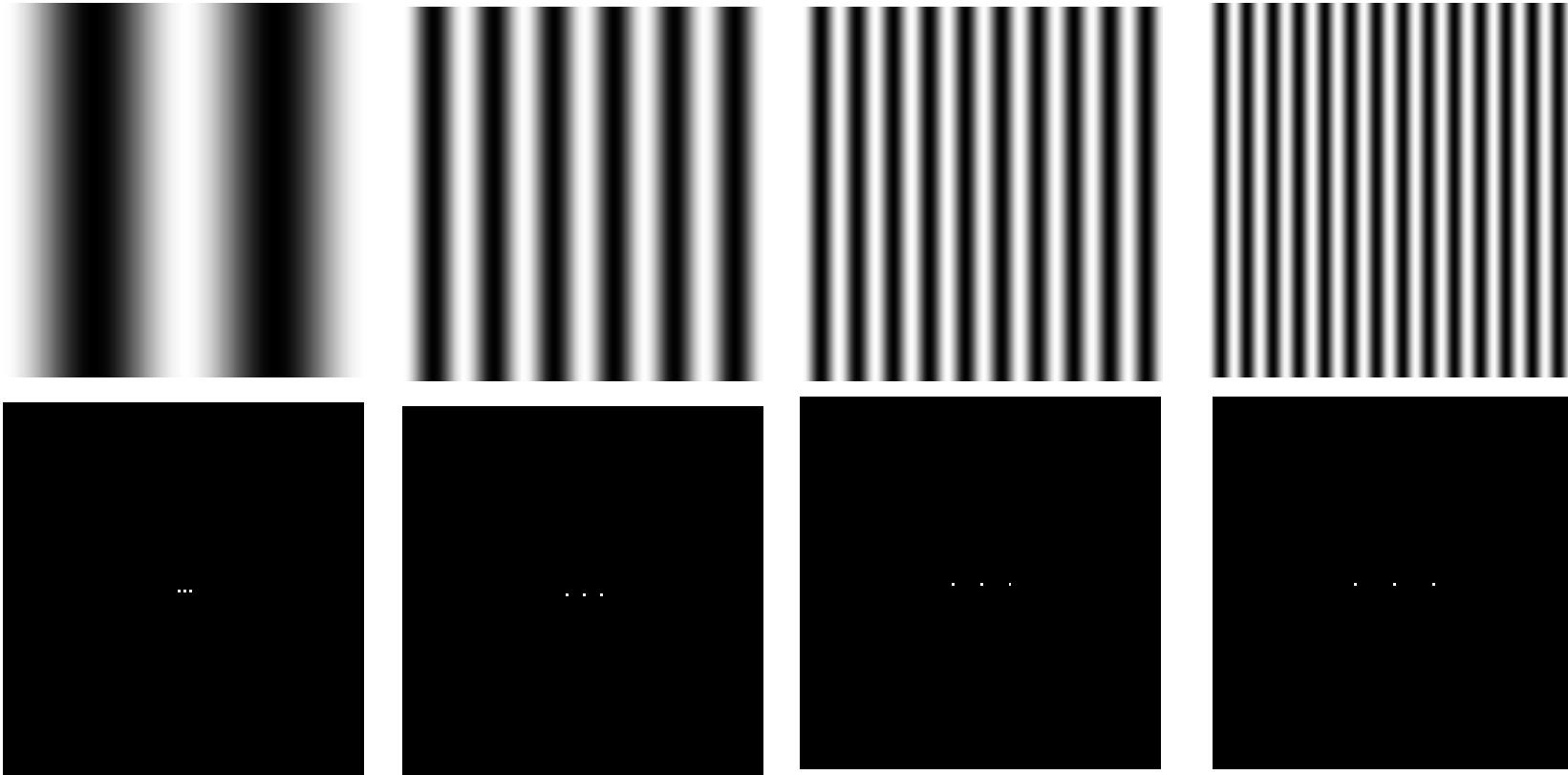
$$F(u,v) = |F(u,v)|e^{i\theta(u,v)} = re^{i\theta}$$

Spatial Domain \longleftrightarrow Frequency Domain



Thresholded log of FT+1

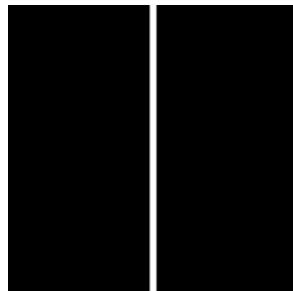
Spatial Domain \longleftrightarrow Frequency Domain



Perpendicular relationship

Ideal edge and line structures have a concentration along a line passing through the origin in the frequency domain and in a **direction**

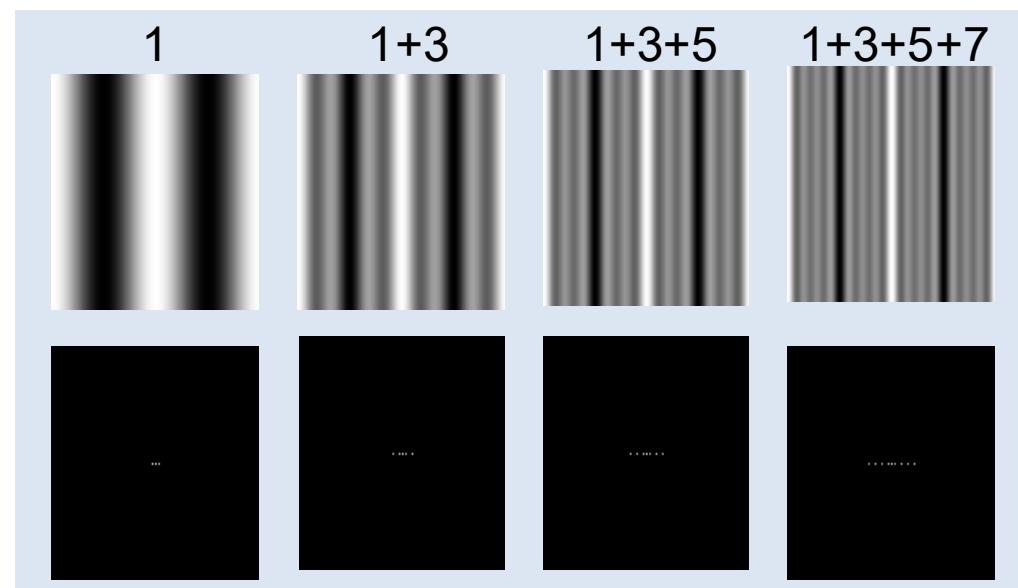
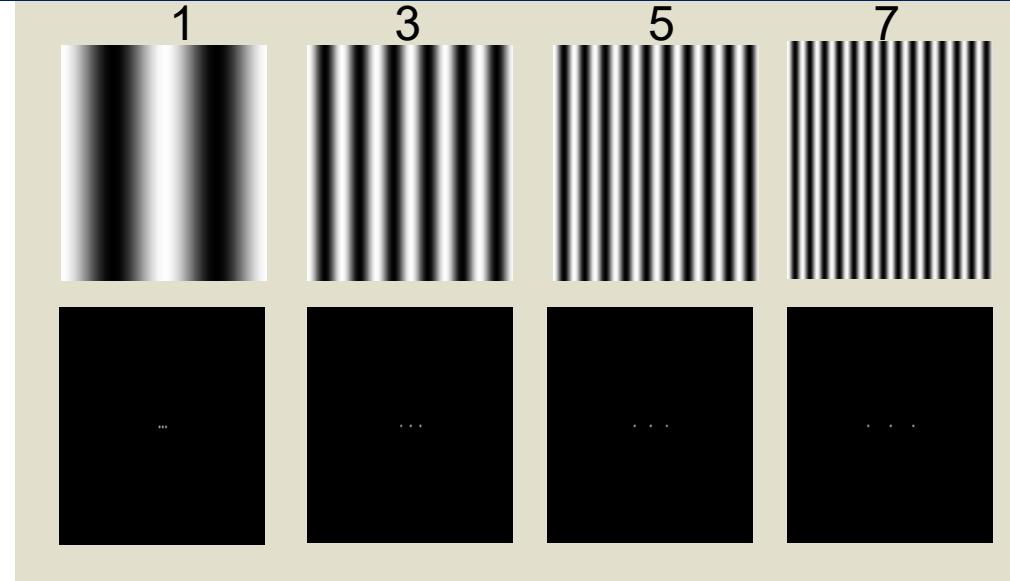
perpendicular to their orientation - they are 'constructed' by adding together all 2D sinusoidal waves that 'travel' perpendicular to the edge or line.



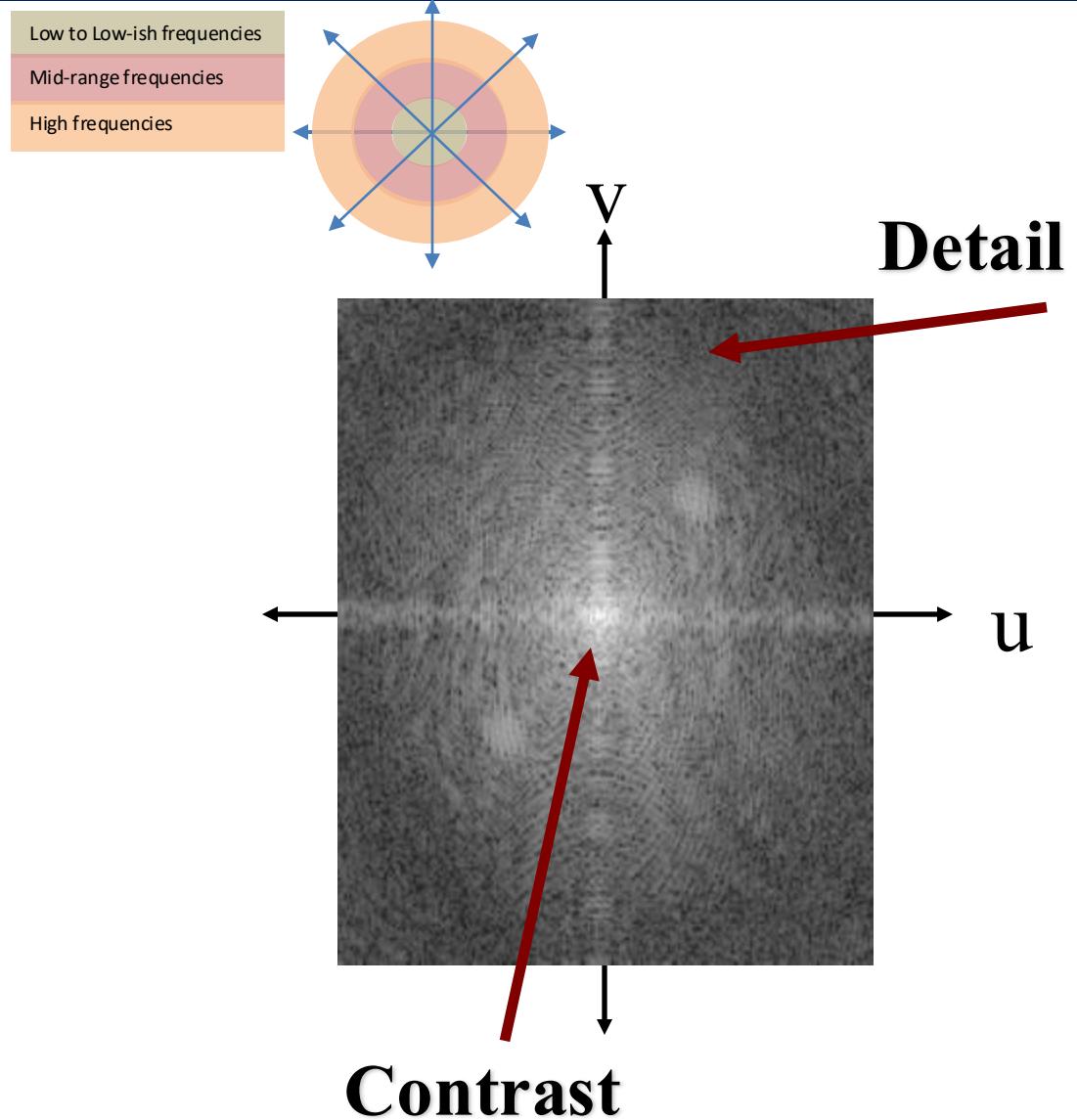
Image



FT

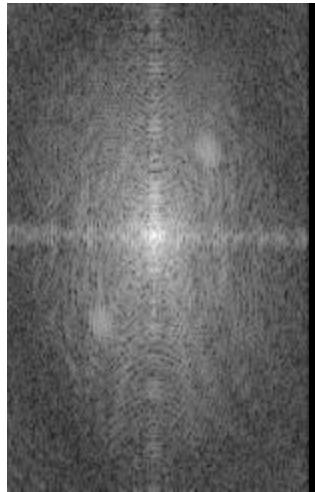


Relating Frequencies to Images

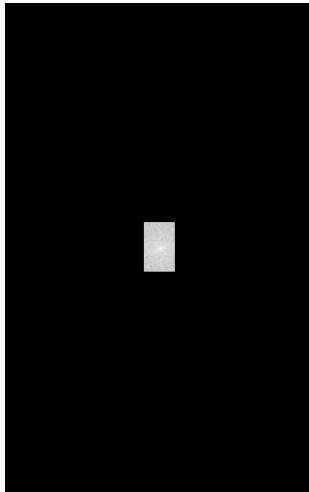


Example: Relating Frequencies to Images

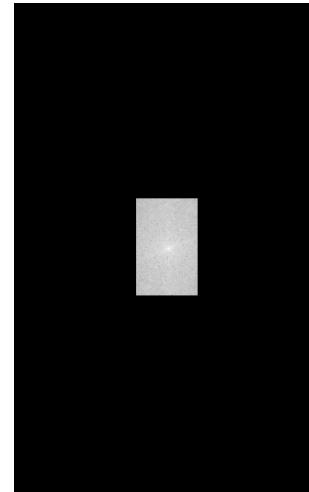
Fourier Space



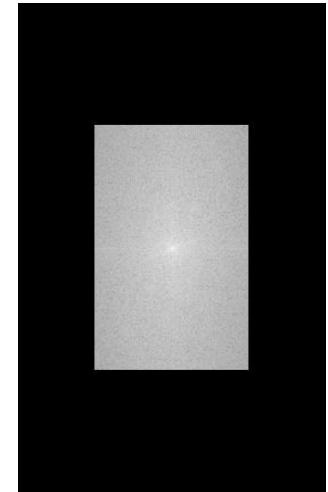
5%



10%

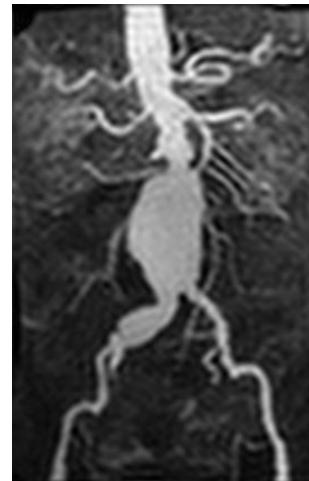
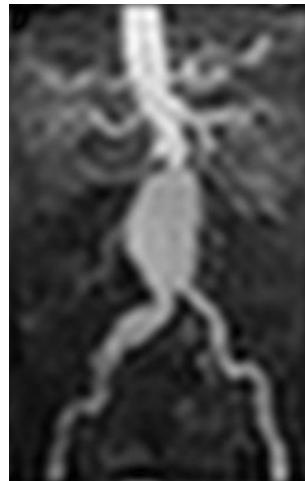
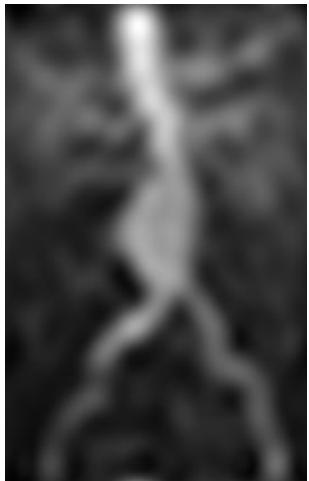


20%



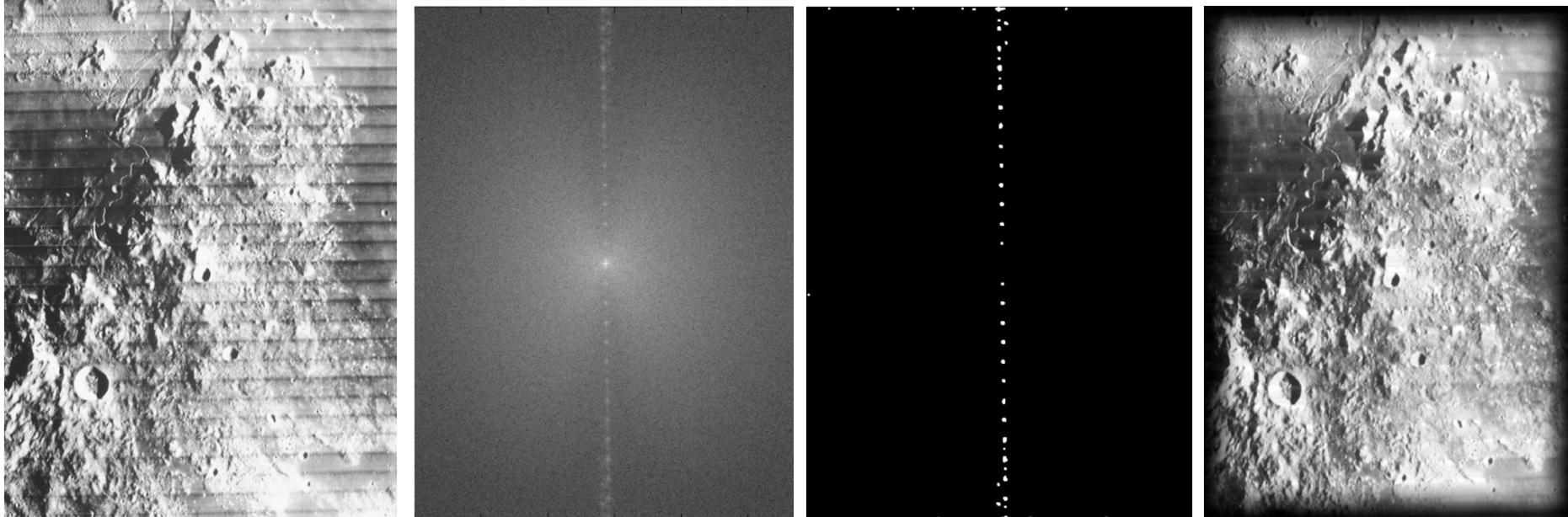
50%

Inverse Transform
back to image Space



Example: Manipulating the FS

Lunar orbital image (1966)



$$|F(u, v)|$$

Mask used to
remove peaks

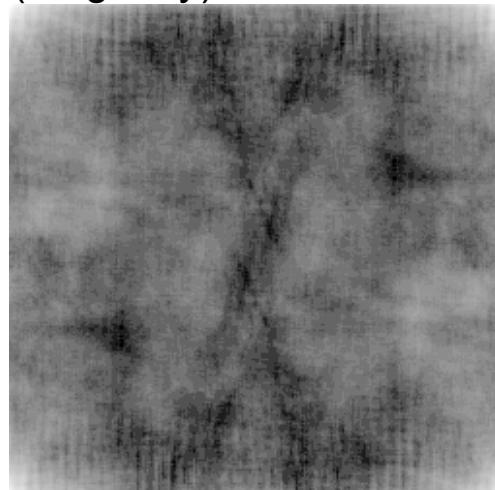
$$\text{iFFT}(F(u, v))$$

Slide by A. Zisserman

Importance of Phase



ifft(mag only)



ifft(phase only)



ifft(mag(Peter) and Phase(Andrew))

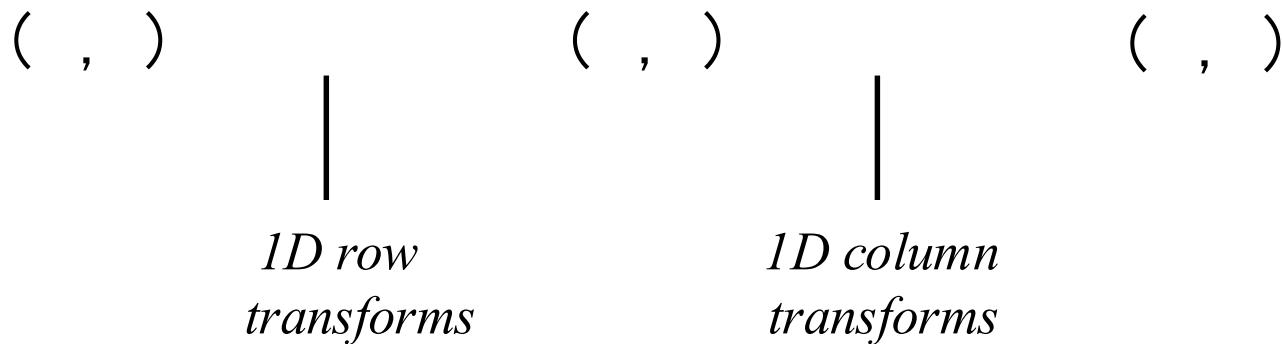


ifft(mag(Andrew) and Phase(Peter))

Separability

- Important property of the FT: *Separability*
 - If a 2D transform is separable, the result can be found by successive application of two 1D transforms. This is a principle aspect of the Fast Fourier Transform (FFT).

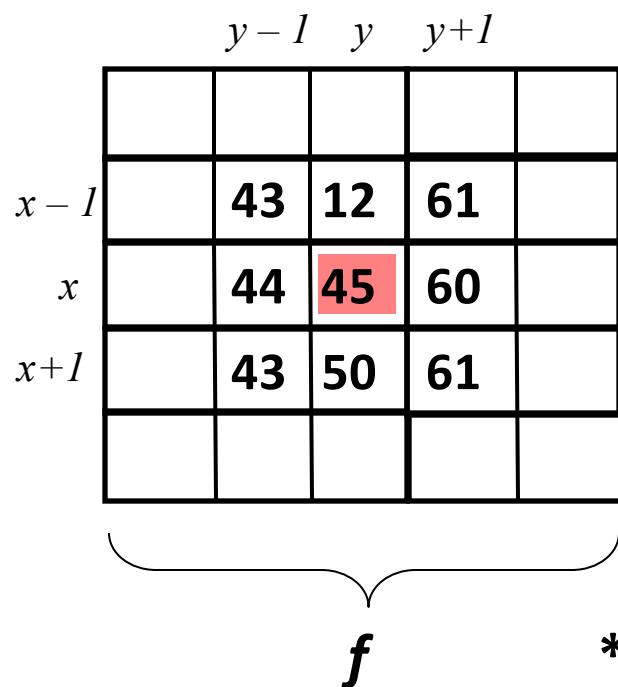
$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} F(x, v) e^{\frac{-j2\pi ux}{N}} \text{ where } F(x, v) = \frac{1}{N} \sum_{y=0}^{N-1} f(x, y) e^{\frac{-j2\pi vy}{N}}$$



Recall: 2D Discrete Convolution

- The discrete version of 2D convolution is defined as:

$$g(x, y) = \sum_m \sum_n f(x - m, y - n)h(m, n)$$



$$\begin{aligned} & f(x+1, y+1)h(-1, -1) \\ & + f(x+1, y)h(-1, 0) \\ & + f(x+1, y-1)h(-1, 1) \\ & + f(x, y+1)h(0, -1) \\ & + f(x, y)h(0, 0) \\ & + f(x, y-1)h(0, 1) \\ & + f(x-1, y+1)h(1, -1) \\ & + f(x-1, y)h(1, 0) \\ & + f(x-1, y-1)h(1, 1) \end{aligned}$$

Convolution in the Spatial/Frequency Domain

Convolution Theorem:

Convolution in spatial domain
is equivalent to
multiplication in frequency domain
(and vice versa)

$$h = f * g \quad \text{implies} \quad H = FG$$

$$h = fg \quad \text{implies} \quad H = F * G$$

Deriving the Convolution Theorem

NOT EXAMINABLE

$$h(x) = f(x) * g(x) = \sum_y f(x - y)g(y)$$

$$H(u) = \sum_x \left(\sum_y f(x - y)g(y) \right) e^{(-iux2\pi/N)}$$

$$H(u) = \sum_y g(y) \left(\sum_x f(x - y) e^{(-iux2\pi/N)} \right)$$

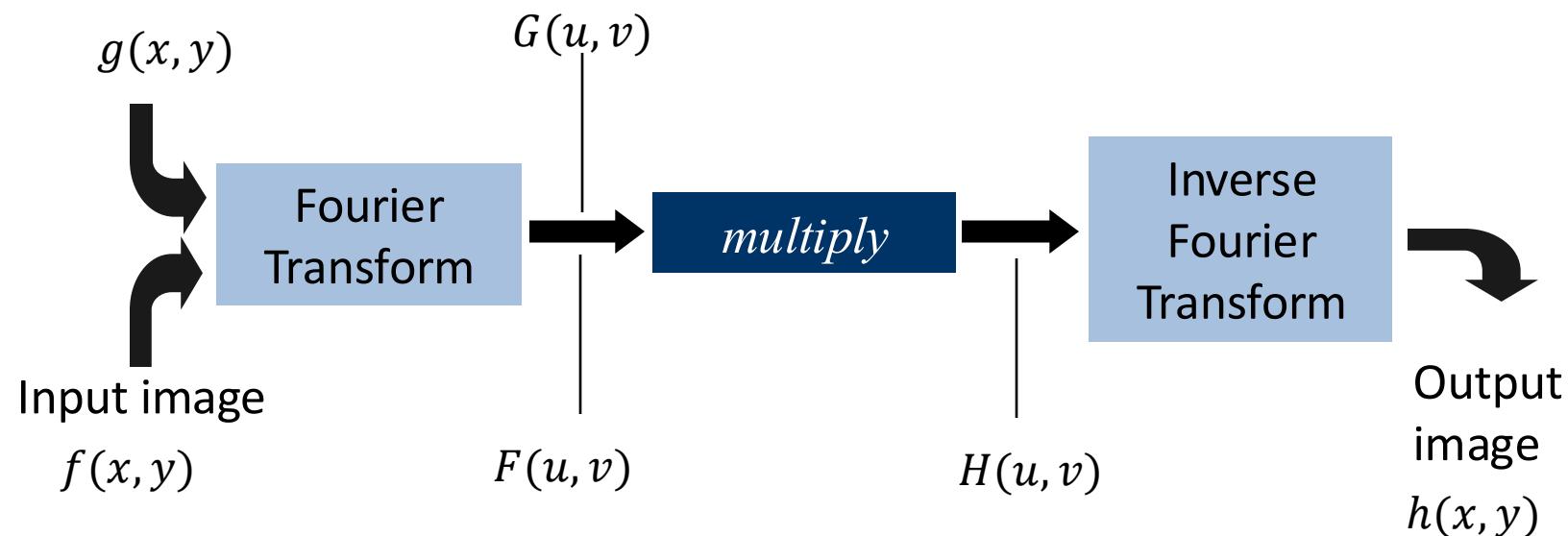
$$H(u) = \sum_y g(y) \left(F(u) e^{(-iuy2\pi/N)} \right)$$

$$H(u) = \sum_y g(y) e^{(-iuy2\pi/N)} F(u) = G(u) \cdot F(u) = F(u) \cdot G(u)$$

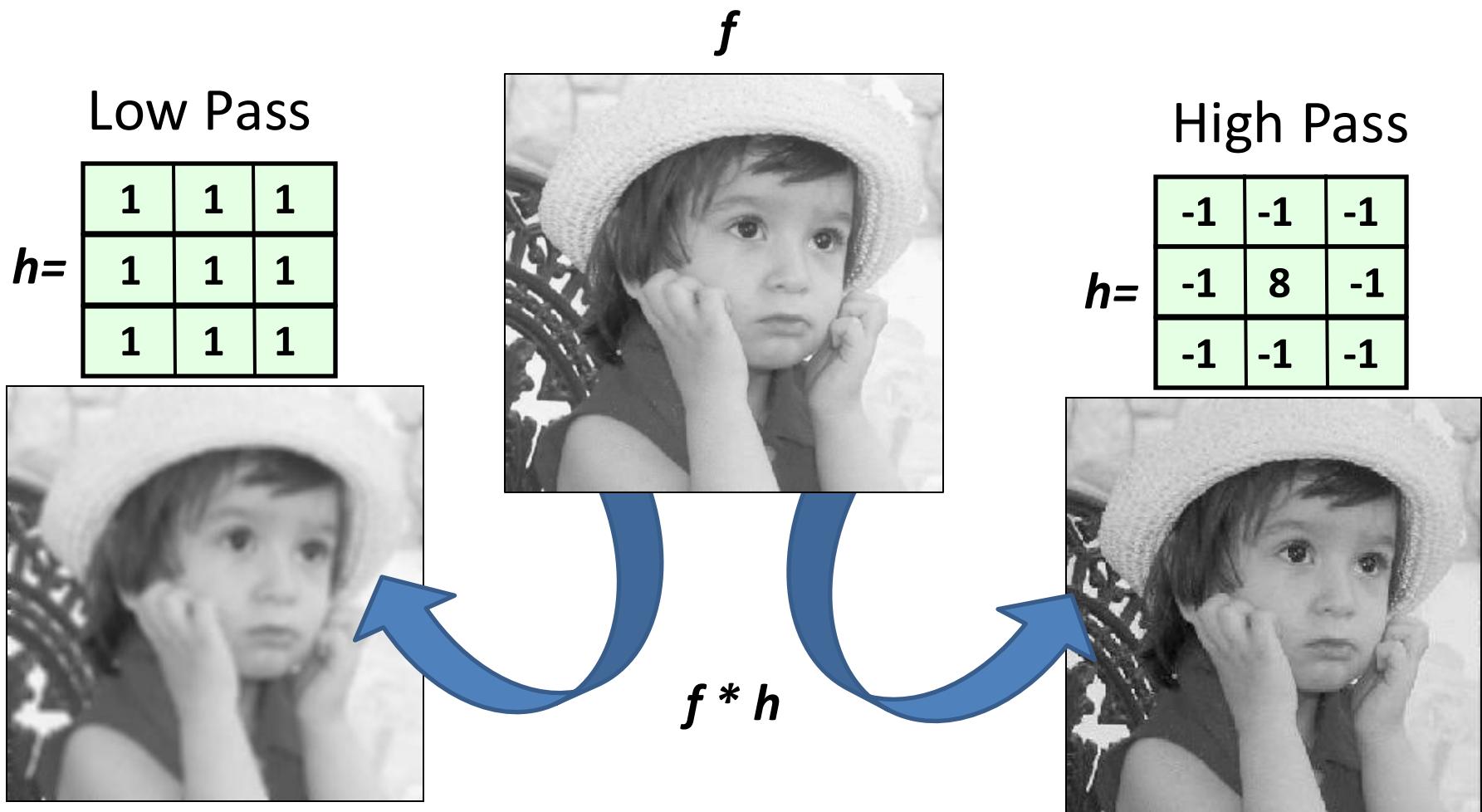
Fast Filtering using the Convolution Theorem

$$1D: H(u) = F(u)G(u)$$

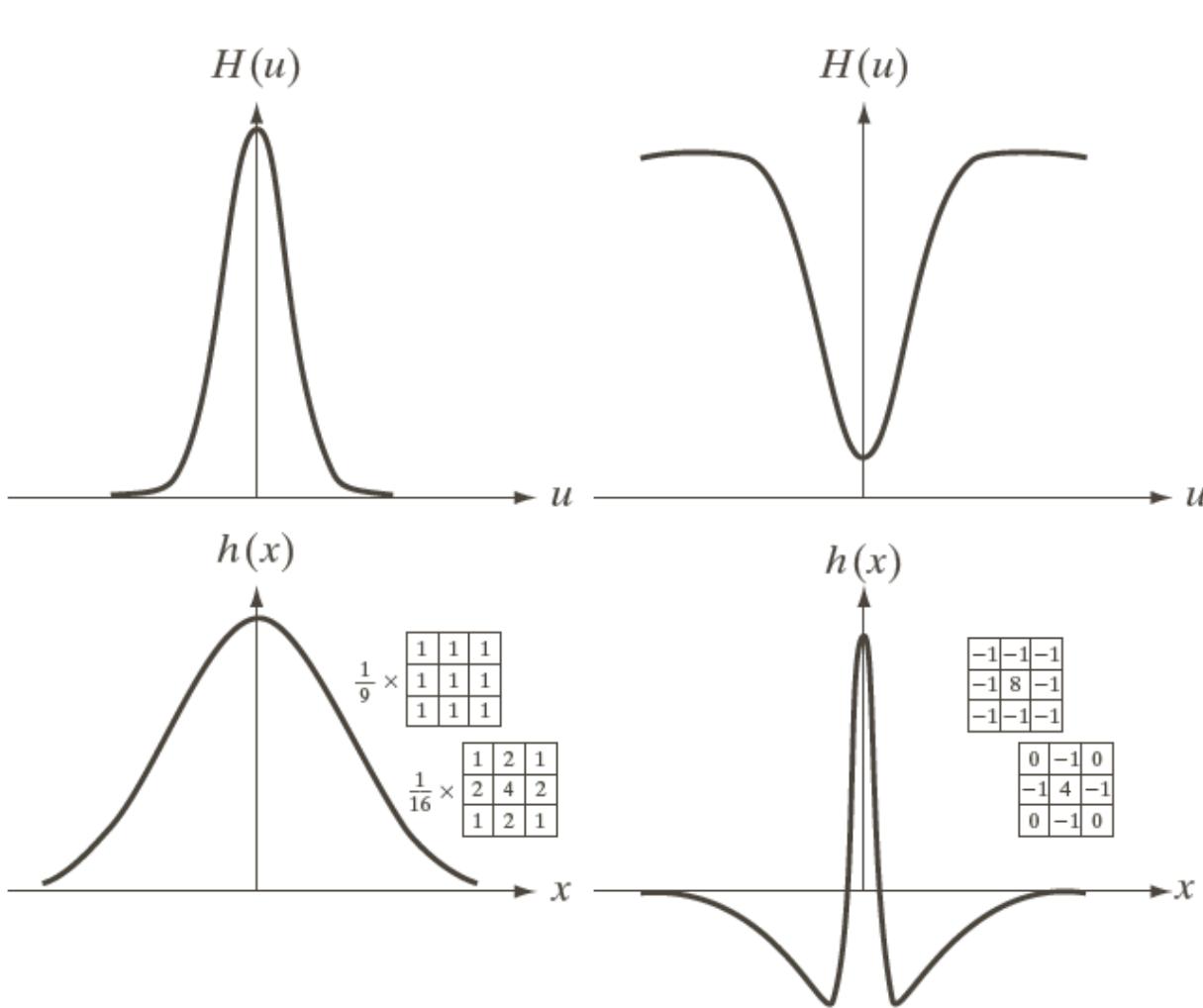
$$2D: H(u, v) = F(u, v)G(u, v)$$



Recall: Spatial Low/High Pass Filtering



Frequency Domain Low/High Pass Filtering



a	c
b	d

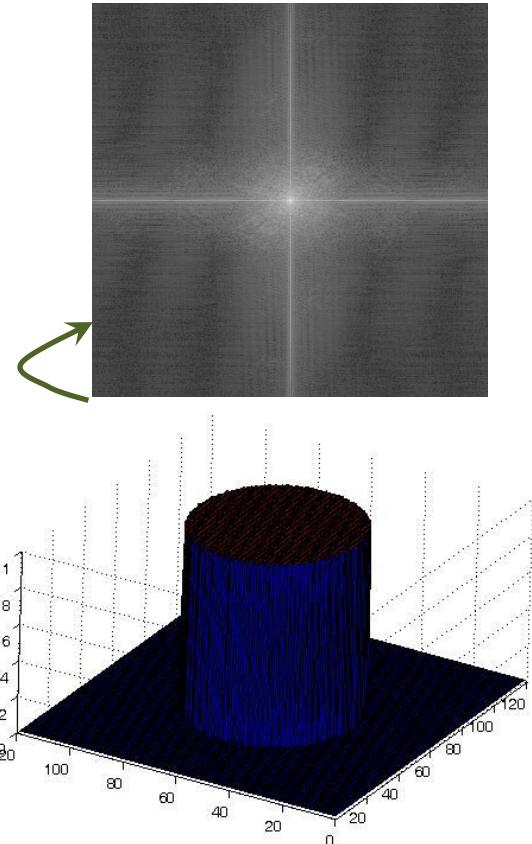
- (a) A 1-D Gaussian lowpass filter in the frequency domain.
(b) Spatial lowpass filter corresponding to (a).
(c) Gaussian highpass filter in the frequency domain.
(d) Spatial highpass filter

Ideal Low Pass Filter

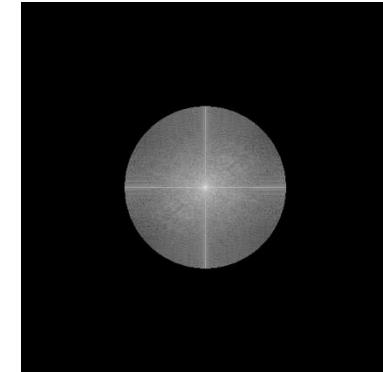
- 1D: turning the “treble” down on audio equipment!
- 2D: smooth image



Apply to freq. domain



$$H(u, v) = \begin{cases} 1 & r(u, v) \leq r_0 \\ 0 & r(u, v) > r_0 \end{cases} \quad r(u, v) = \sqrt{u^2 + v^2}, \quad r_0 \text{ is the filter radius}$$

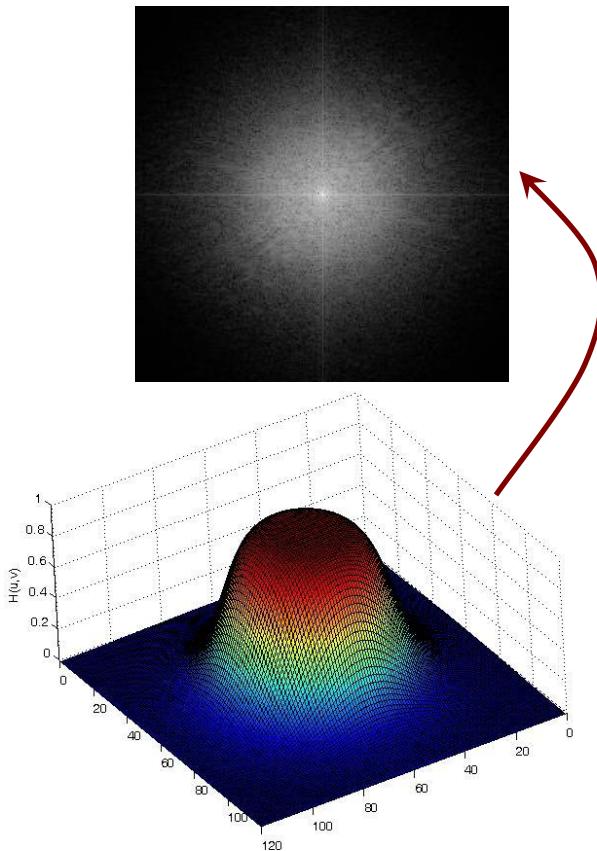


Butterworth's Low Pass Filter

Input image



After applying to freq. domain



After filtering



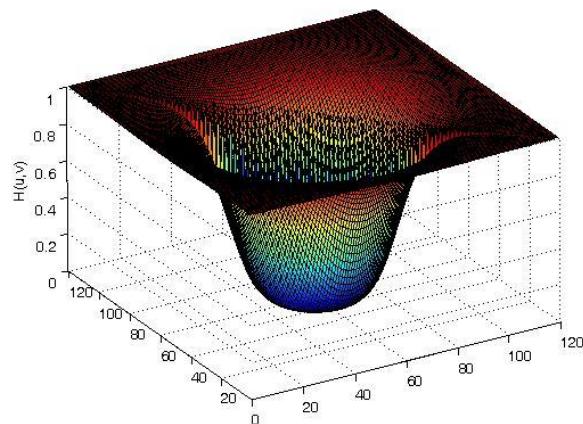
$$H(u, v) = \frac{1}{1 + [r(u, v) / r_0]^{2n}} \quad \text{of order } n$$

Butterworth's High Pass Filter

Input image



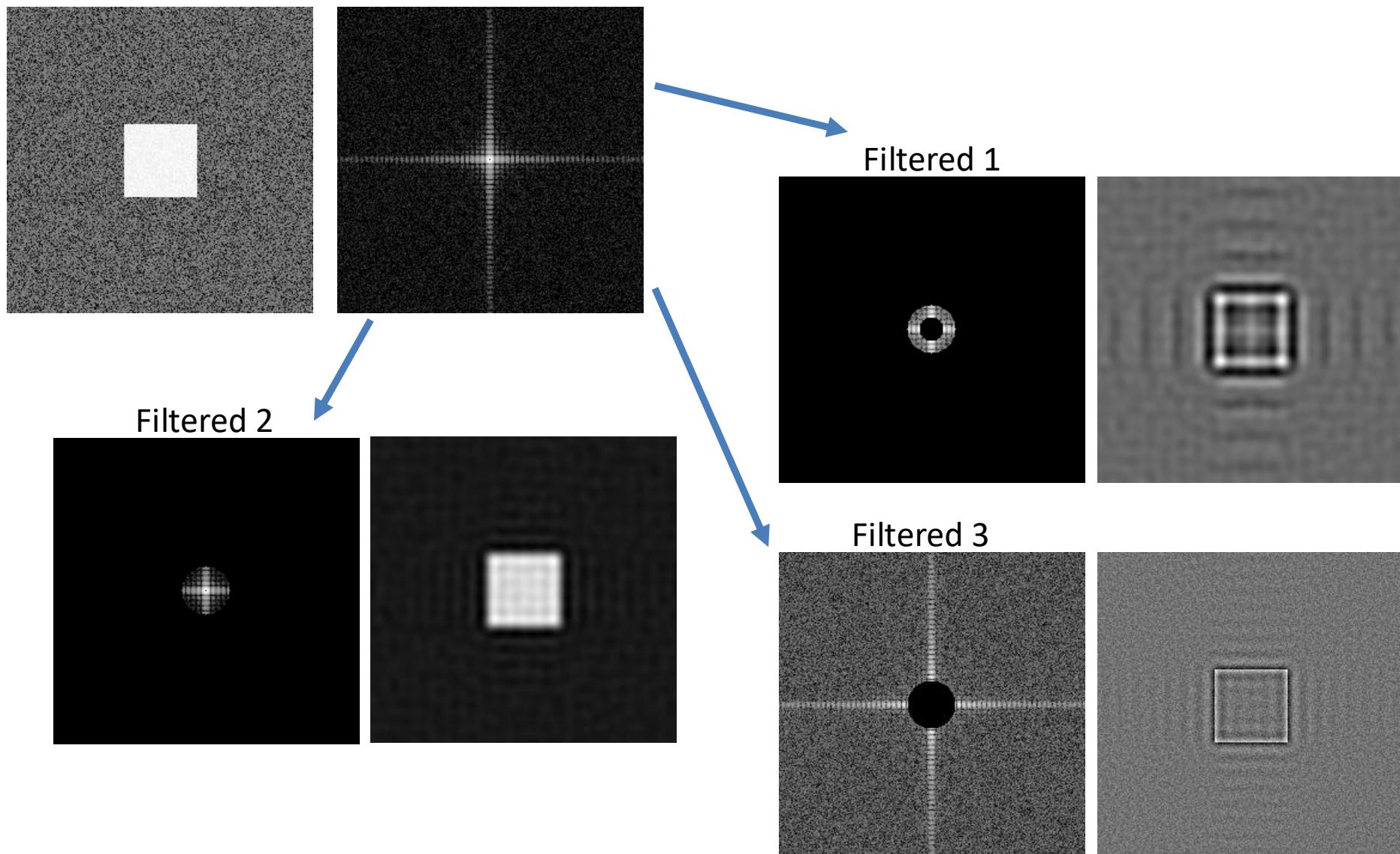
After filtering



Order of $n=3$

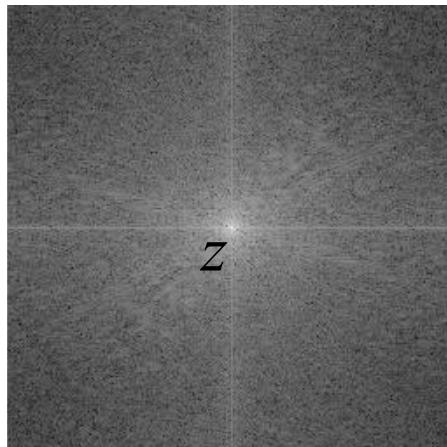
$$H(u,v) = \frac{1}{1 + [r_0 / r(u,v)]^{2n}} \quad \text{of order } n$$

Other Custom/Example filters



Other Custom/Example filters

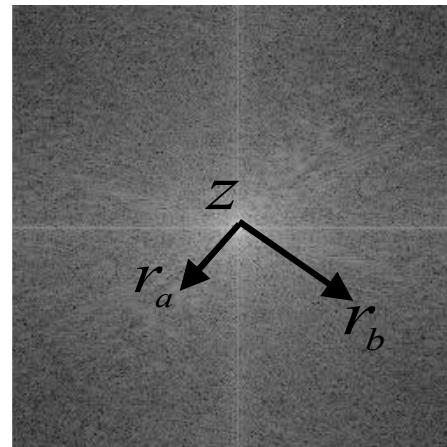
- Fourier space, with origin at $z=(u=0, v=0)$.



$$a \leq u \leq b$$

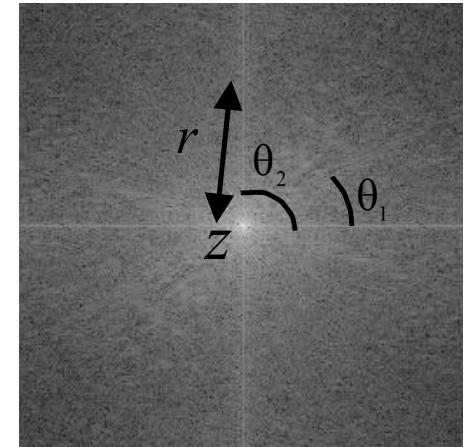
$$c \leq v \leq d$$

box



$$\begin{aligned} -r_b &\leq u \leq r_b \\ \pm\sqrt{r_a^2 - u^2} &\leq v \leq \pm\sqrt{r_b^2 - u^2} \end{aligned}$$

ring

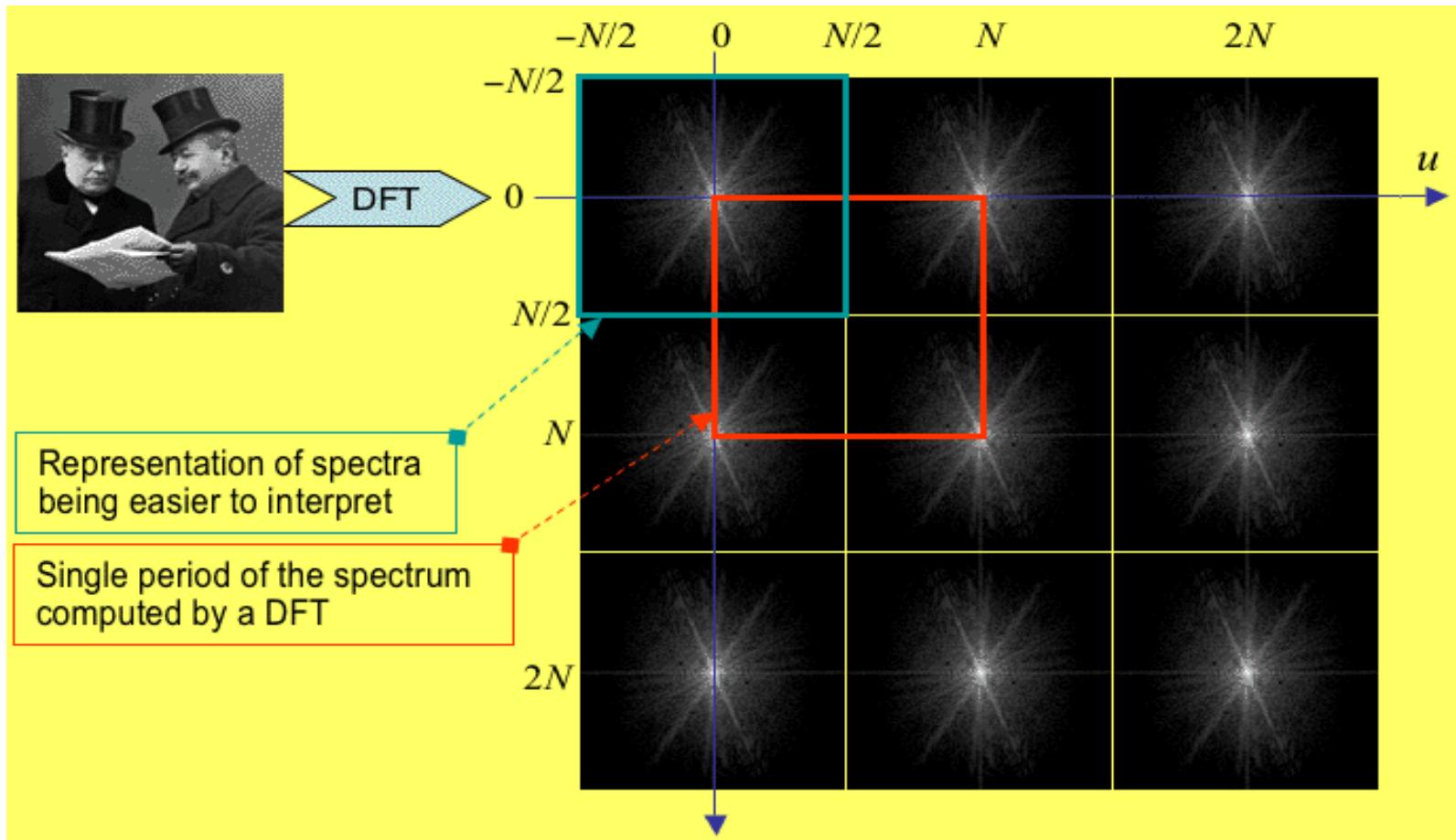


$$u^2 + v^2 = r^2$$

$$\theta_1 \leq \tan^{-1} \frac{v}{u} \leq \theta_2$$

sector

Periodic Spectrum

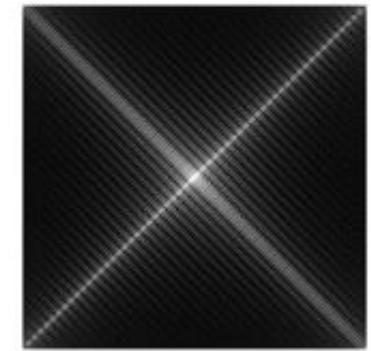
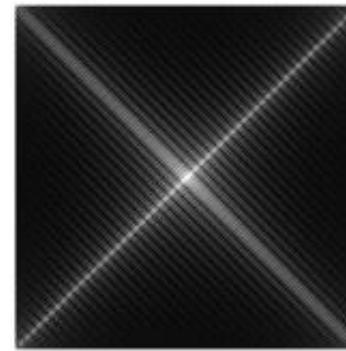
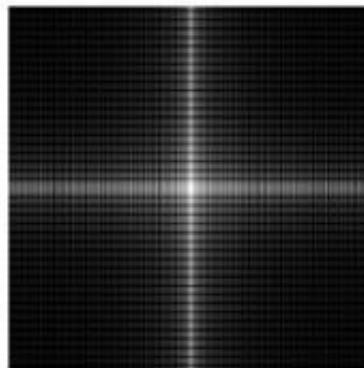


Self-Study: Effects of Rotation/Translation Illustrated

Translation or shift in Spatial Domain

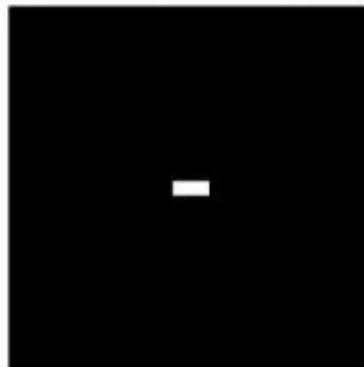
Spatial shifts result in a linear phase change in the frequency domain, but no change in the magnitude spectrum. Hence, the magnitude spectrum of a line or dot, for example, looks the same wherever it is in the image.

FT



Rotation in Spatial Domain

Rotation of an image in the spatial domain results in a corresponding rotation in the Fourier domain.



Image



Rotation



Rotation & translation

Self-Study: Summary of Filter Definitions

Notation notice! Using D instead of r

Lowpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$	$H(u, v) = \frac{1}{1 + [D(u, v)/D_0]^{2n}}$	$H(u, v) = e^{-D^2(u,v)/2D_0^2}$

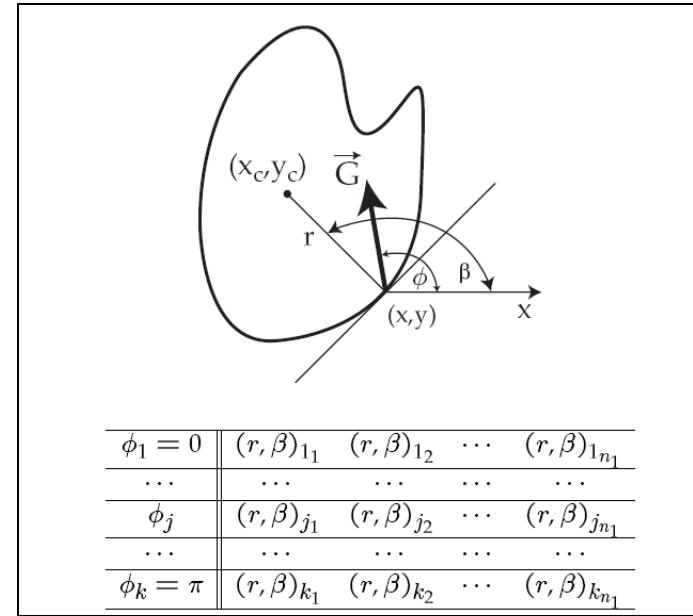
Highpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u, v) = \begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$	$H(u, v) = \frac{1}{1 + [D_0/D(u, v)]^{2n}}$	$H(u, v) = 1 - e^{-D^2(u,v)/2D_0^2}$

Next Lecture

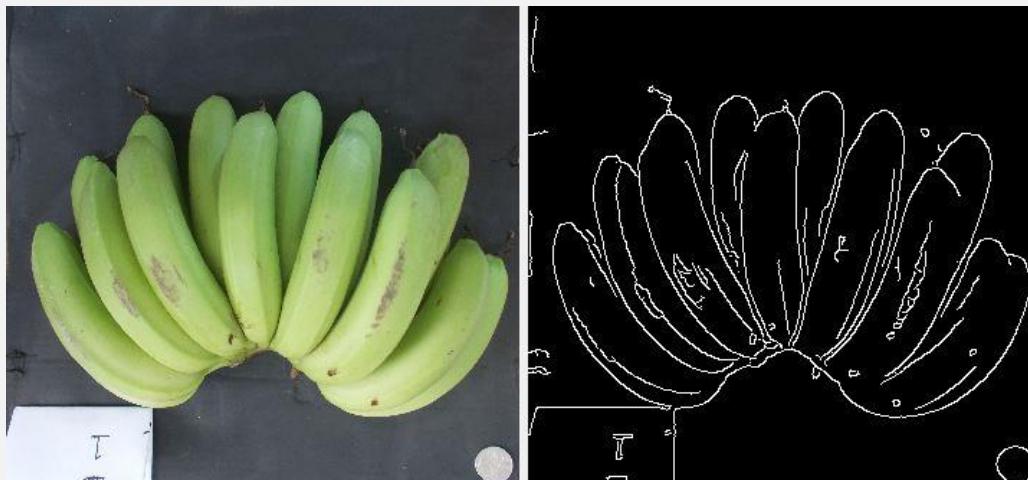


Edge Detection



Hough Transform

COMS30030 - Image Processing and Computer Vision



Lecture 04

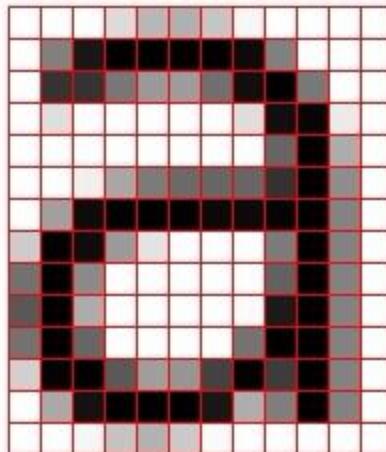
Edge & Shape Detection

Majid Mirmehdi | majid@cs.bris.ac.uk

Beyond the Matrix

- Images are matrices of numbers
- Shapes are more informative than individual pixel values

0.2235	0.1294	Blue	0.4196
0.5804	0.2902	0.0627	0.2902 0.466
0.5804	0.0627	0.0627	0.2235 0.2588
0.5176	0.1922	0.0627	Green 0.1922 0.2588 0.2588
0.5176	0.1294	0.1608	0.1294 0.1294 0.2588 0.2588
0.5176	0.1608	0.0627	0.1608 0.1922 0.2588 0.2588
0.5490	0.2235	0.5490	Red 0.7412 0.7765 0.7765 902
0.5490	0.3882	0.5176	0.5804 0.5804 0.7765 0.7765 196
0.490	0.2588	0.2902	0.2588 0.2235 0.4824 0.2235
0.2235	0.1608	0.2588	0.2588 0.1608 0.2588
0.2588	0.1608	0.2588	0.2588 0.2588 0.2588



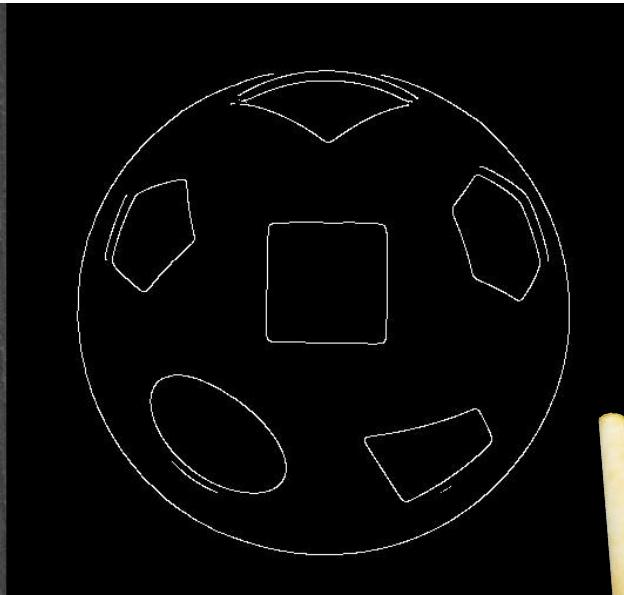
=

1	0	1	0	1	0	0	9	0	6	0	6	0	6	1	0	1	0	1	0	1	0		
1	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	1	0	1	0	
1	0	0	2	0	2	0	5	0	6	0	6	0	5	0	0	0	0	0	5	1	0	1	
1	0	0	9	1	0	1	0	1	0	1	0	1	0	0	9	0	0	0	0	0	9	1	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	5	0	0	0	0	0	5	1	0
1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	1	0	1	0	0	0	5	1	0
1	0	1	0	1	0	0	5	0	5	0	5	0	5	0	4	0	0	0	0	0	5	1	0
1	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	1	0
0	9	0	0	0	0	6	1	0	1	0	1	0	0	5	0	0	0	0	0	5	1	0	1
0	5	0	0	0	6	1	0	1	0	1	0	1	0	0	5	0	0	0	0	0	5	1	0
0	5	0	0	7	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	5	1	0
0	6	0	0	6	1	0	1	0	1	0	1	0	0	5	0	0	0	0	0	5	1	0	1
0	9	0	1	0	0	6	0	7	0	7	0	5	0	0	0	5	0	0	0	0	5	1	0
1	0	0	7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	1	0

Source: <https://www.datasciencecentral.com/profiles/blogs/image-classification-with-hsv-color-model-processing>

What are edges?

- Edges highlight the contour of shapes
- They can be used to identify objects



Edge map

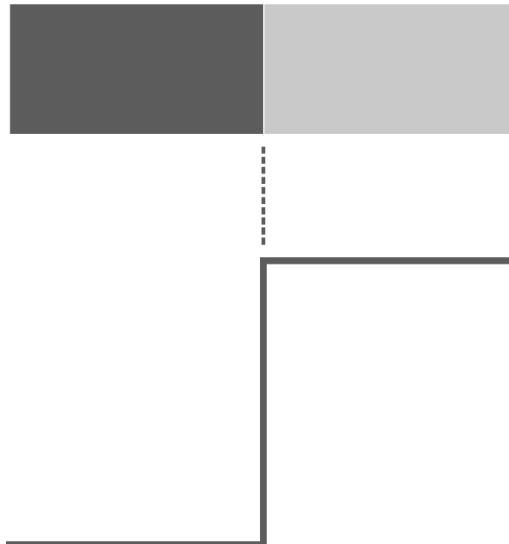


Source: <http://www.walrusvision.wordpress/introduction-to-edge-detection/>

What are edges?

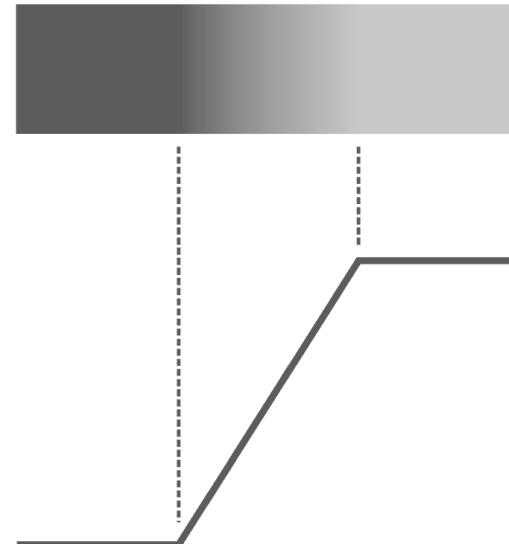
- Edges are those places in an image that correspond to object boundaries.
- Edges are pixels where image brightness changes (relatively) abruptly.

Model of an ideal digital edge



Gray level profile of a horizontal line through the image

Model of a ramp digital edge



Gray level profile of a horizontal line through the image

Why detect Edges?

- **Edges**: Sharp changes of image brightness
- **Sources**: Object boundaries, patterns, shadows, etc.
- For segmentation: finding object boundaries



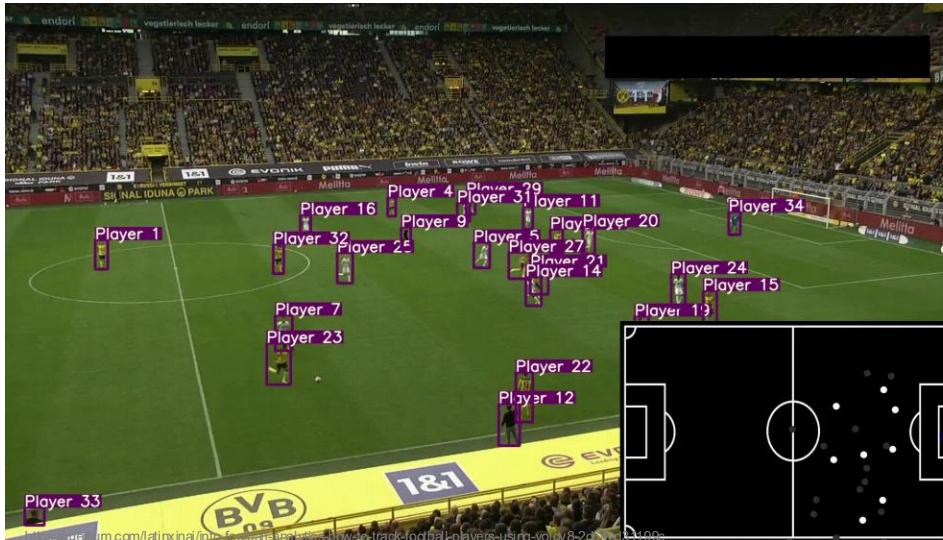
Why detect Edges?

- **Edges**: Sharp changes of image brightness
- **Sources**: Object boundaries, patterns, shadows, etc.
- For segmentation: finding object boundaries
- For recognition: extracting patterns



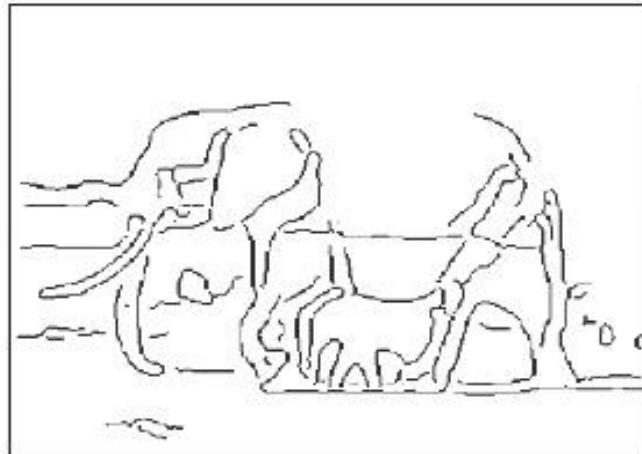
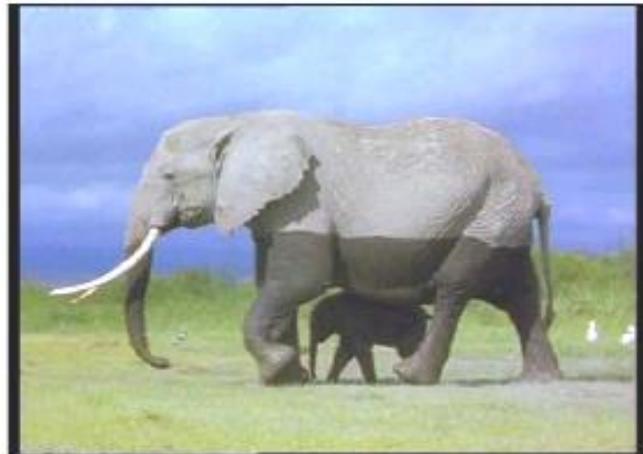
Why detect Edges?

- **Edges**: Sharp changes of image brightness
- **Sources**: Object boundaries, patterns, shadows, etc.
- For segmentation: finding object boundaries
- For recognition: extracting patterns
- For motion analysis: reliable tracking regions

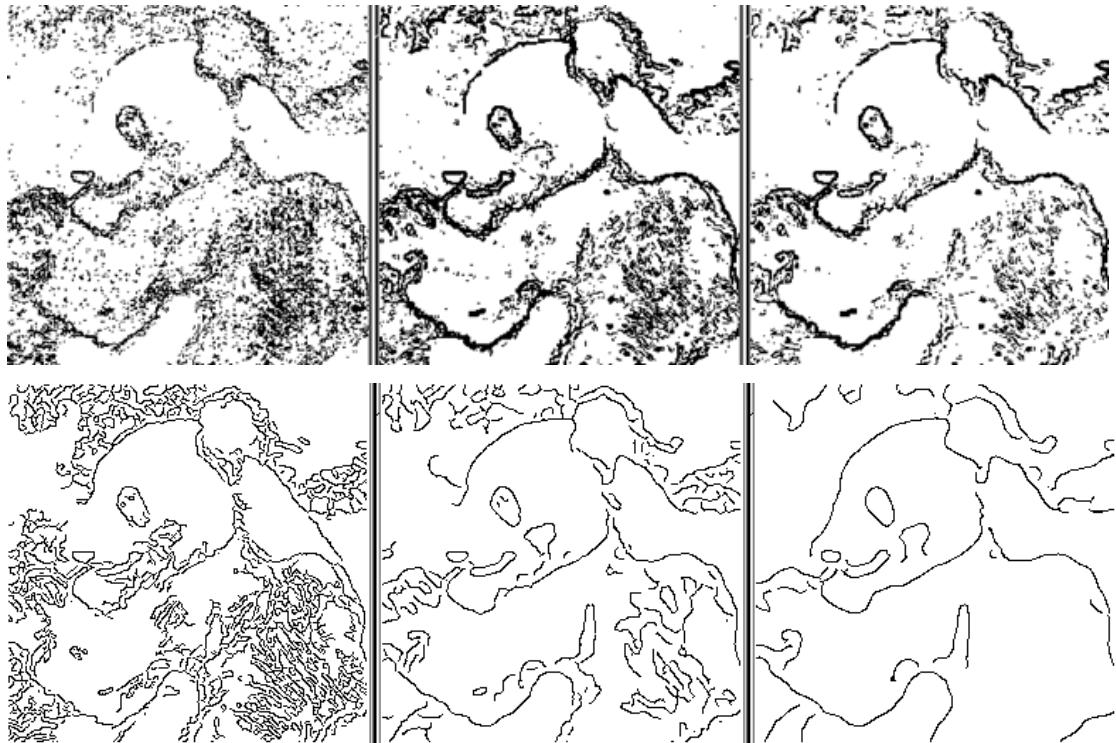


Meaningful edges vs. nuisance edges

- **Edges:** Sharp changes of image brightness

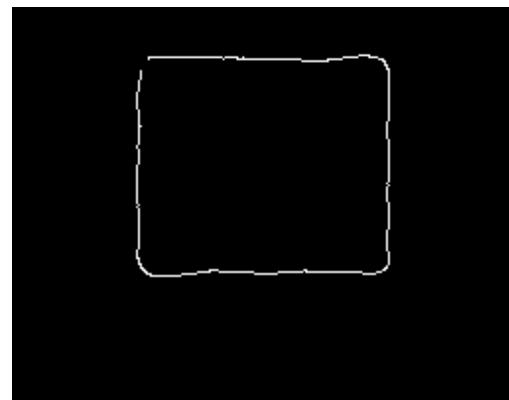
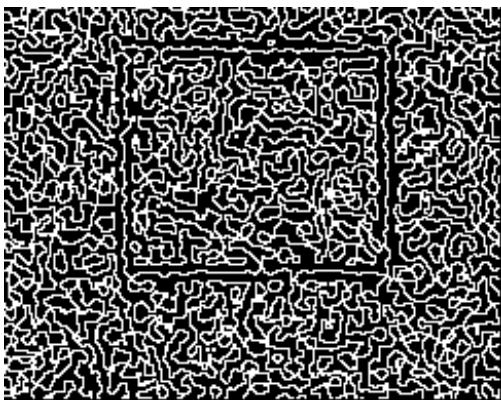
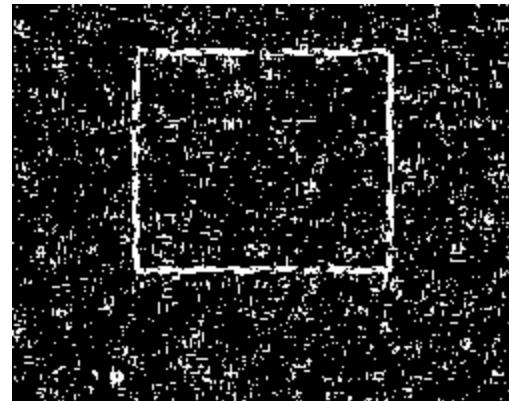
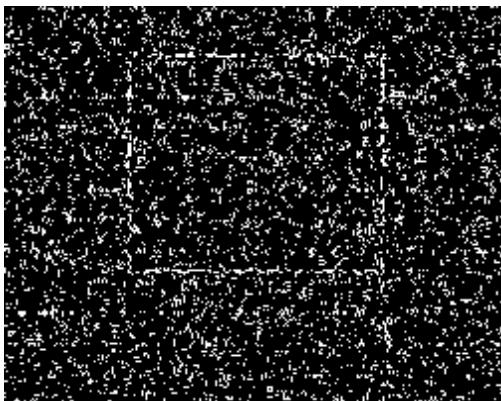
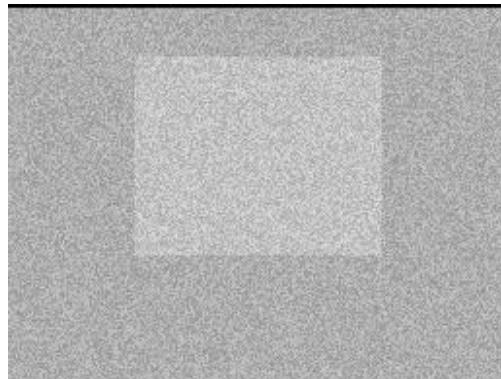


Difficult edges



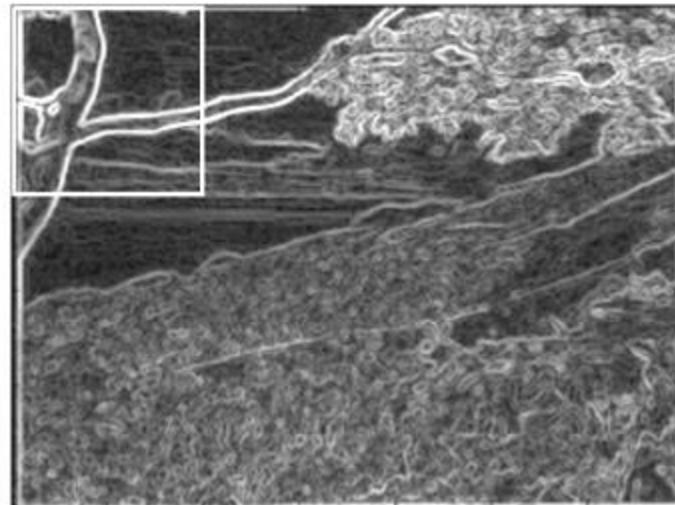
Very difficult edges

Different techniques
work for different cases.



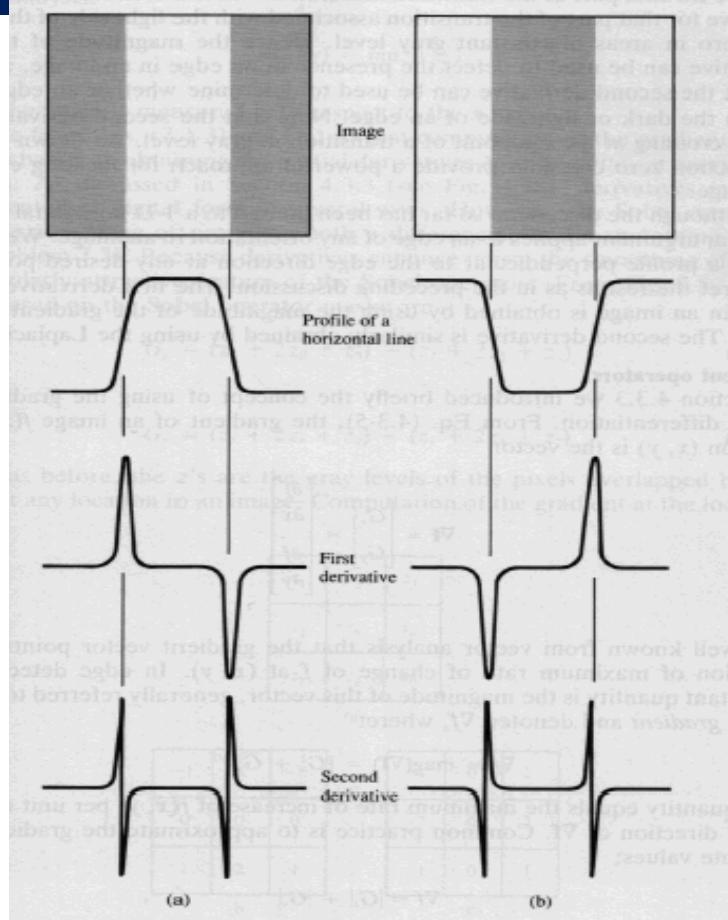
Edge Detection Strategy

- **Recognition Strategy:** determine a '**measure of change**' in a pixel's neighbourhood
- First derivatives in 2D space → image gradient



Edge Detection Strategy

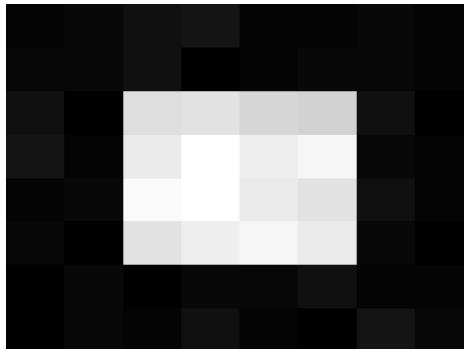
- Magnitude of the 1st derivative can be used to detect the presence of an edge
- Zero-crossing of the 2nd derivative at the midpoint of a transition in gray level, which provides a powerful approach for locating the edge
- The sign of the 2nd derivative usually used to determine whether an edge pixel lies on the dark or light side of an edge



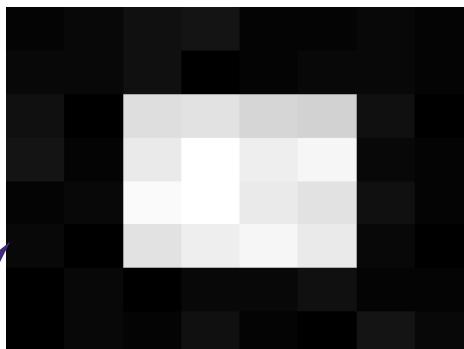
Edge Detection Strategy

Horizontal differencing detects **vertical** edges

Difference
→



Difference
↓

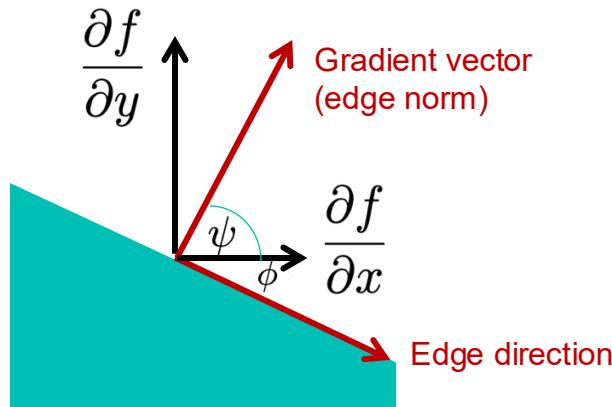


Vertical differencing detects **Horizontal** edges

The Image Gradient

A **vector** variable

- Direction ψ of the maximum growth of the function
- Magnitude $|\nabla f(x, y)|$ of the growth
- Perpendicular to the edge direction ϕ

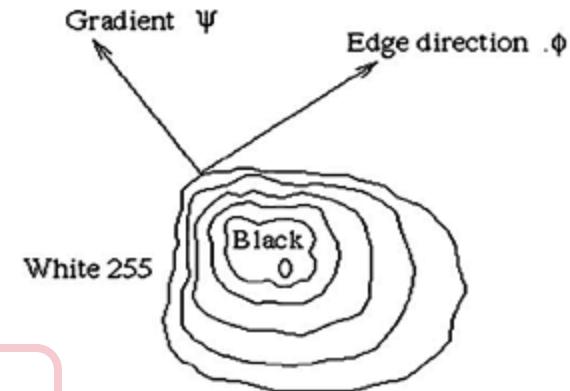


$$\nabla f(x, y) = \frac{\partial f}{\partial x} \hat{x} + \frac{\partial f}{\partial y} \hat{y}$$

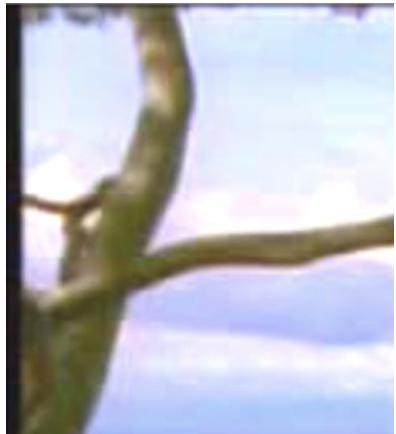
$$|\nabla f(x, y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

$$\psi = \arctan \left(\frac{\partial f / \partial y}{\partial f / \partial x} \right)$$

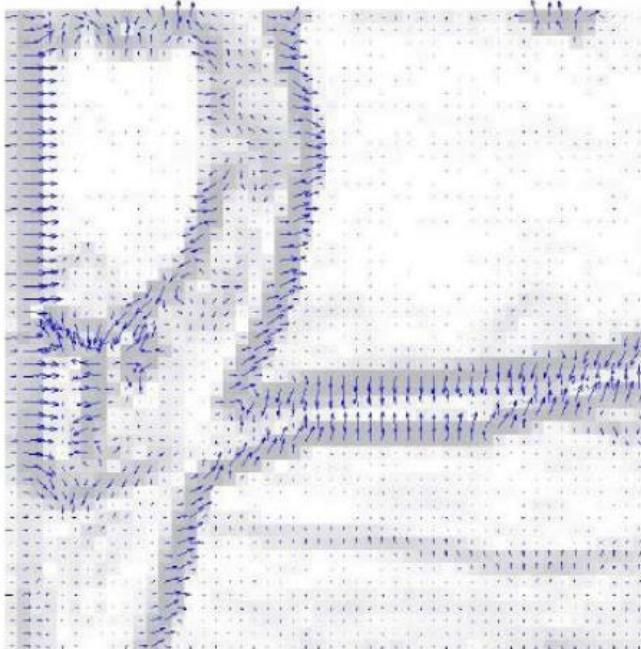
$$\phi = \psi - \frac{\pi}{2}$$



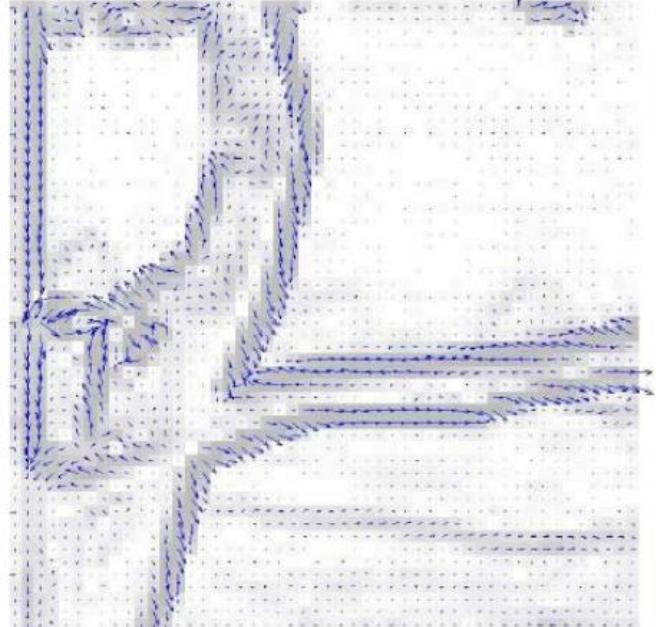
Example: Gradient & Edge Vectors



Gradient vectors



Edge vectors

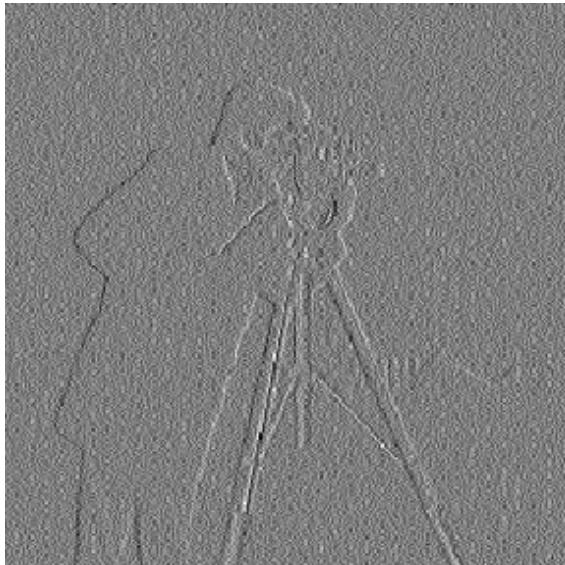


Simple derivatives too noisy

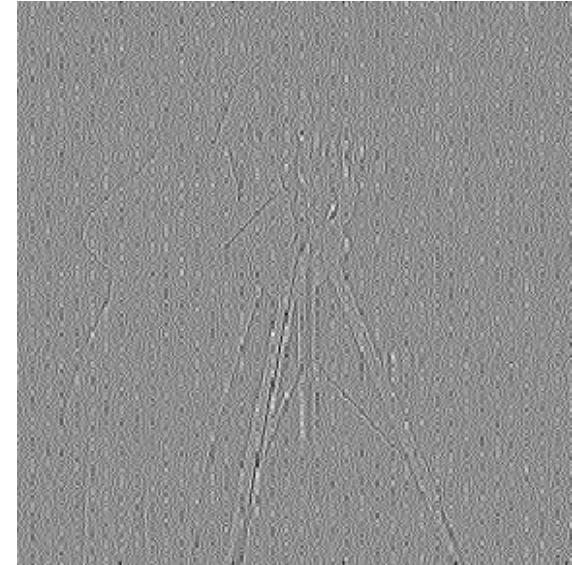
- Straight differencing of pixels is sensitive to noise



Noisy cameraman



First derivative

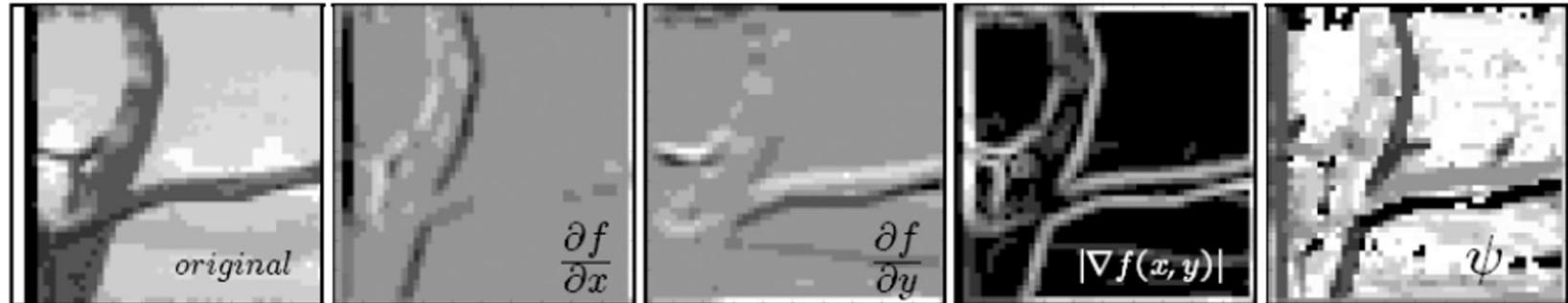


Second derivative

Gradient Extraction via Filtering

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad h_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$grad(f) = |\nabla f(x, y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad \psi = \arctan\left(\frac{\partial f / \partial y}{\partial f / \partial x}\right)$$

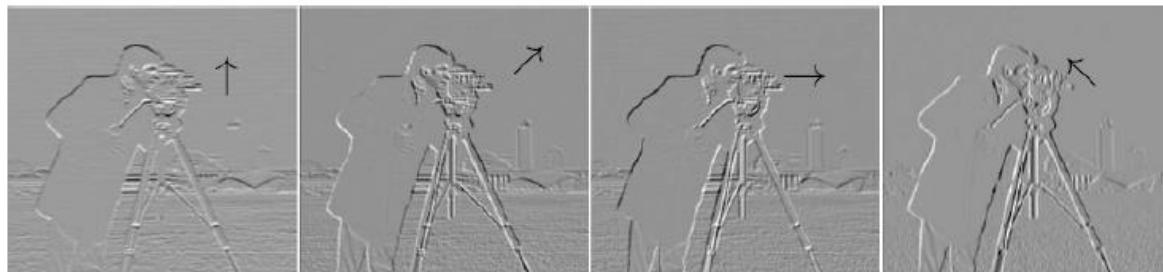
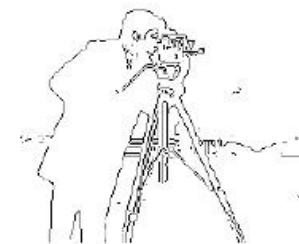


Prewitt Operator

- Central difference $\frac{\partial f}{\partial x} \approx \frac{f(x+1) - f(x-1)}{2}$
- Mask $[-1 \ 0 \ 1]$ is very sensitive to noise
- For 3x3 mask, ∇f can be estimated **in 8 directions**



$$h_{hor} = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_{dia} = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \dots$$



Sobel Operator

- As Prewitt, Sobel relies on central differences
- Greater weight to the central pixels

$$h_{hor} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, h_{ver} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Can be approx. as derivative of a Gaussian
- First Gaussian smoothing, then derivation

$$\frac{\partial}{\partial x}(I * G) = I * \frac{\partial G}{\partial x}$$



Note, the filters are named by the way they seek out gradients, e.g. horizontal and vertical gradients to detect vertical and horizontal edges, respectively.

Sobel Operator



(a) original image

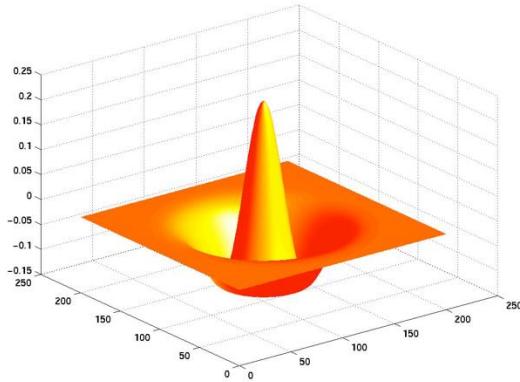


(b) Sobel edge magnitude



(c) thresholded magnitude

Laplacian of a Gaussian Edge Detector



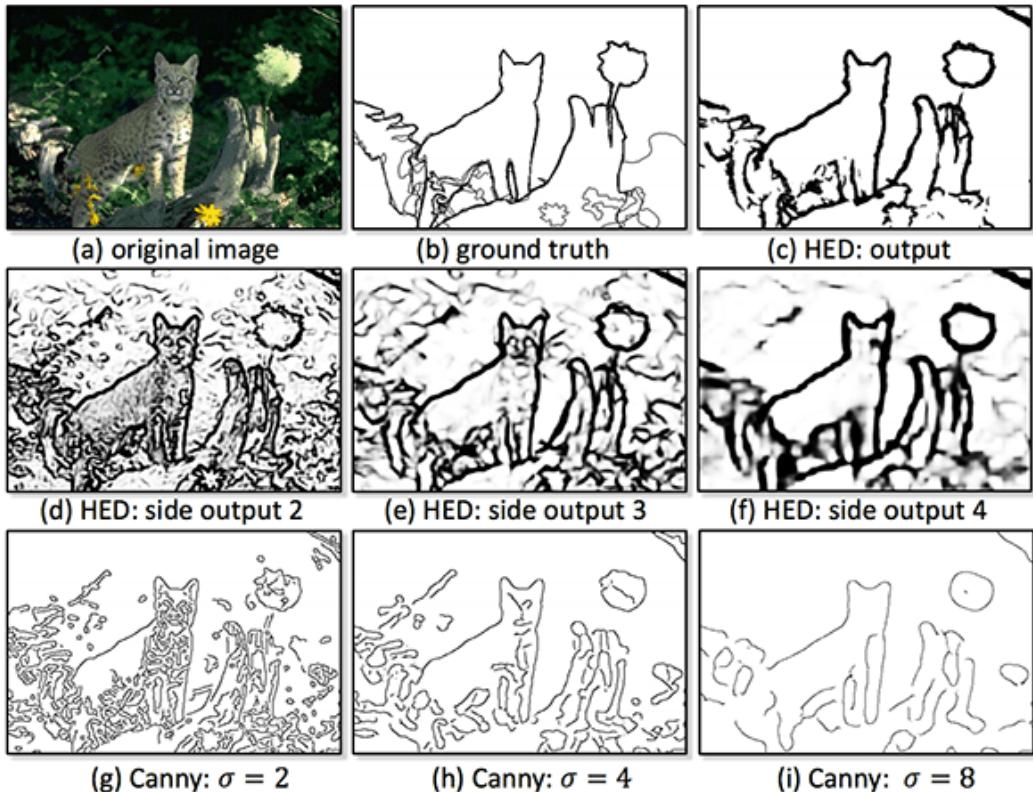
0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0



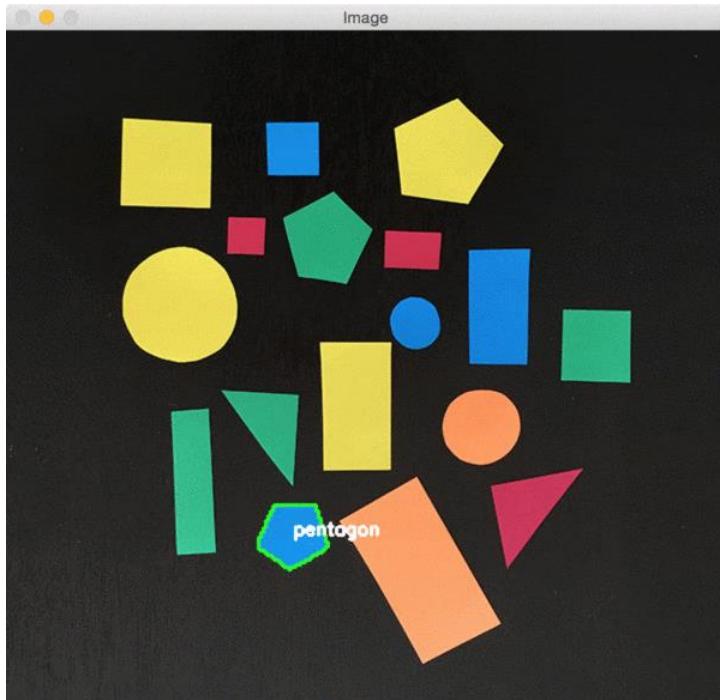
There are many edge detection methods...

Most, if not all, edge detection methods operate with one or more parameters/thresholds.

Is there ever a perfect edge map...



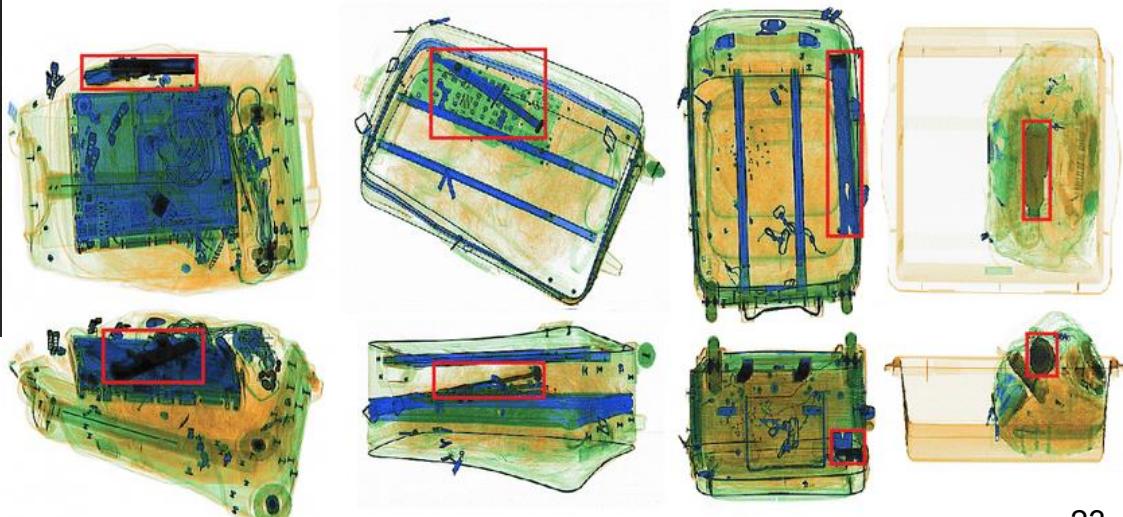
Shape Detection



<https://byimage.com/2016/02/09/binary-shape-detection/>



<https://www.industry.com/journal/article/0202214032223>



https://www.researchgate.net/figure/Example-scans-of-bags-in-false-color-containing-a-firearm-handgun-sharp-knife-blunt_fg1_337944556

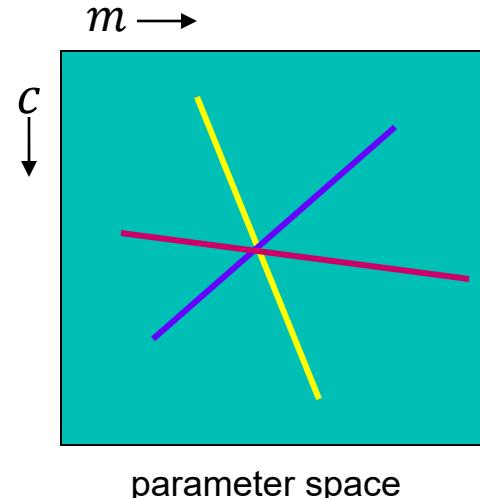
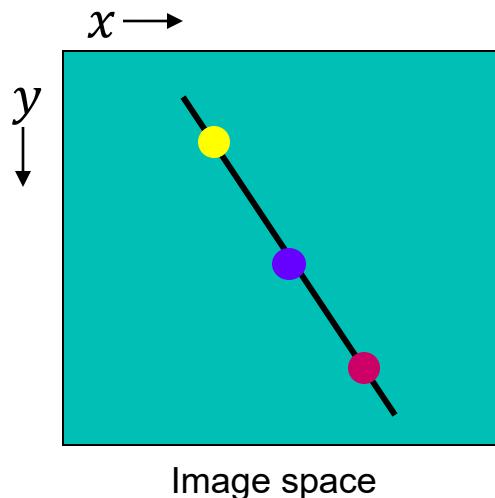
Line Detection via the Hough Transform

In image space, a line is points (x, y) with gradient m , intercept c

$$y = mx + c$$

In parameter space, a line is points (m, c) , with gradient $-x$, intercept y

$$c = -xm + y$$



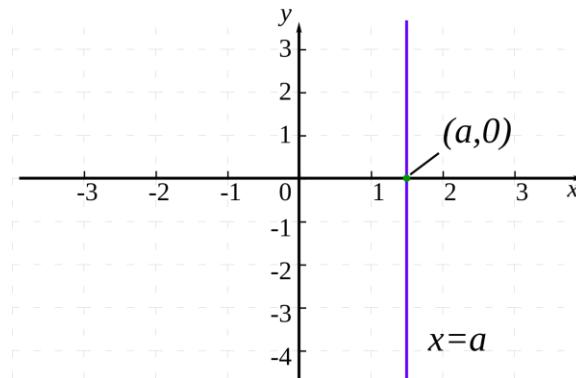
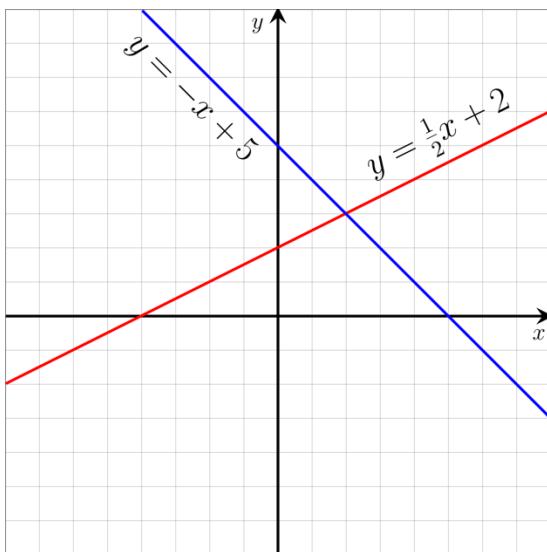
Accumulate votes!
The coordinates of
the peak are the
parameters m, c of
the line!

Image space

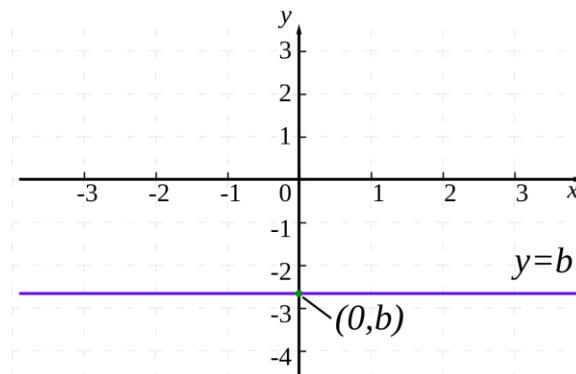
parameter space

Line Representation

$$y = mx + c$$



No y intercept!
slope is infinite!



Line Representation

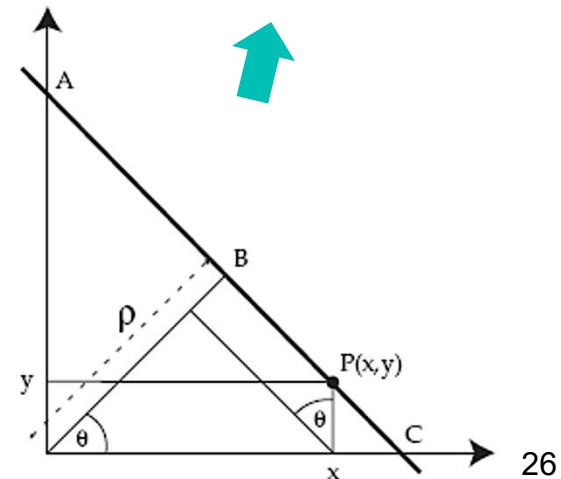
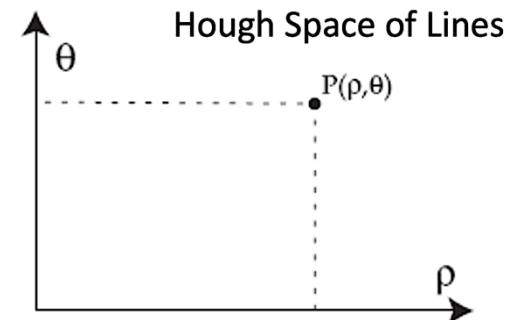
A straight line in 2D space can also be described in its polar form:

$$f(x, y, \rho, \theta) = x \cos \theta + y \sin \theta - \rho = 0$$

It can be represented in the 2D parameter space by point (ρ, θ)

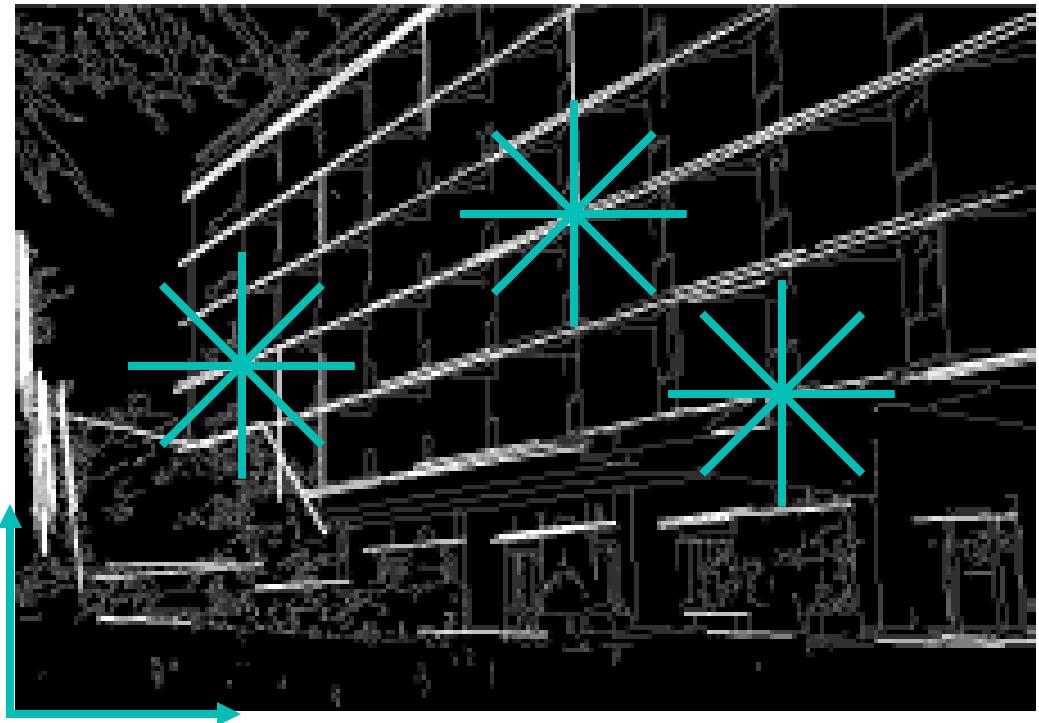
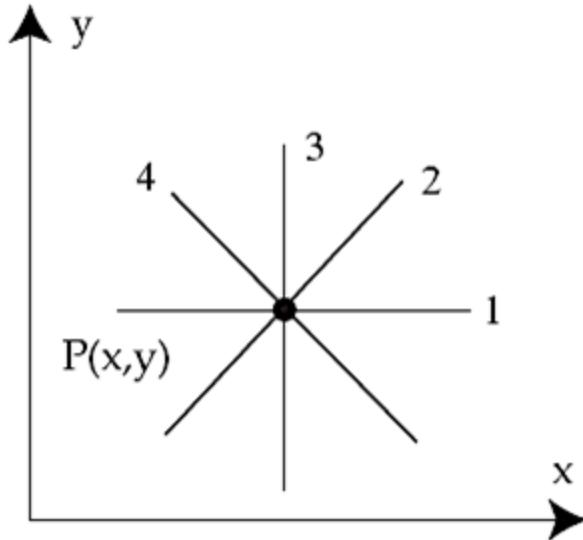
ρ = distance between the straight line and the origin,

θ = angle between the distance vector and the positive x -direction.



Line Representation

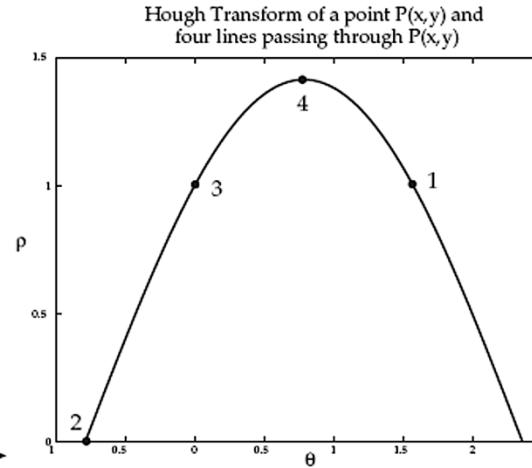
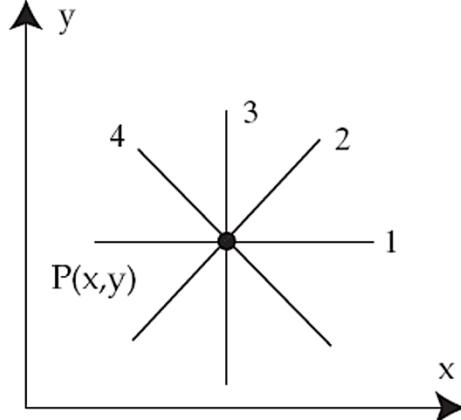
$$f(x, y, \rho, \theta) = x \cos \theta + y \sin \theta - \rho = 0$$



The Hough Space

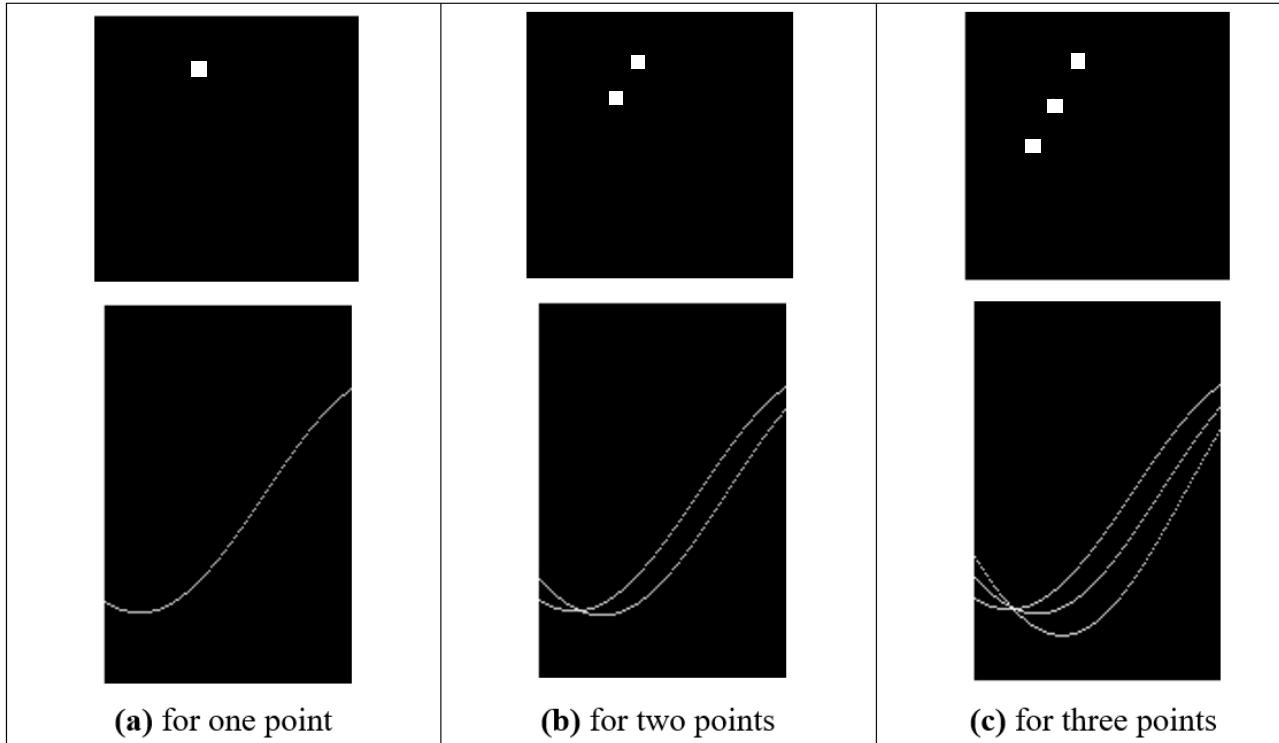
A **point** (x, y) in the image space is transformed into a **sinusoidal curve** in the parameter space. A **point** (ρ, θ) on this sinusoidal curve represents a **straight line** passing through the **point** (x, y) in the image space.

	point 1	point 2	point 3	point 4
θ	$\pi/2 = 1.571$	$-\pi/4 = -0.785$	0	$\pi/4 = 0.785$
ρ	1	0	1	1.4142



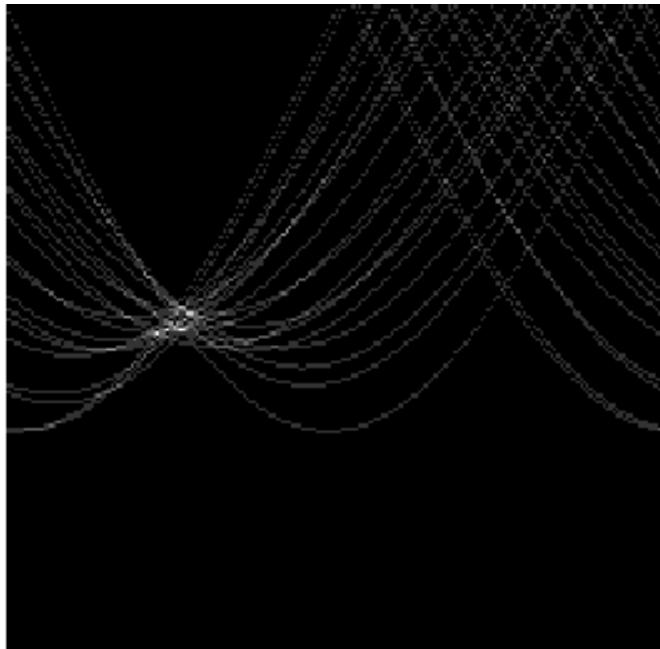
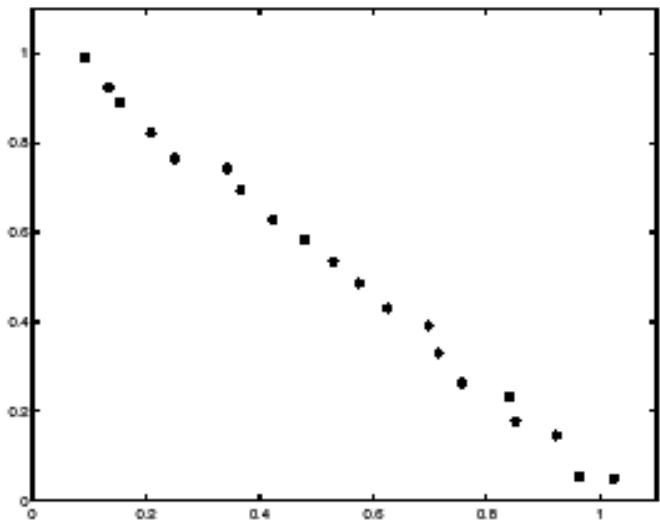
Line Representation

$$f(x, y, \rho, \theta) = x \cos \theta + y \sin \theta - \rho = 0$$



Line Representation

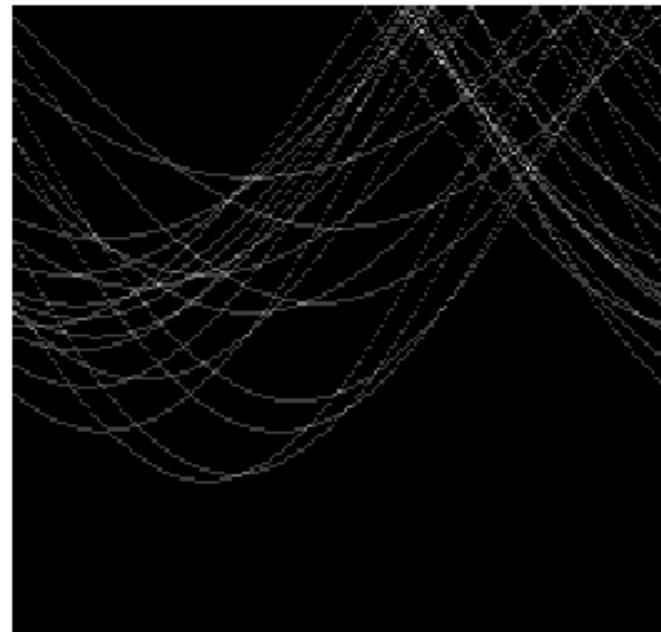
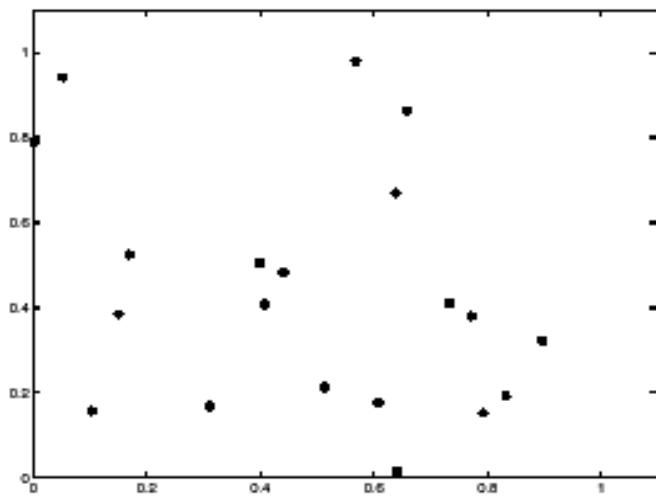
$$f(x, y, \rho, \theta) = x \cos \theta + y \sin \theta - \rho = 0$$



Peak gets fuzzy and hard to locate

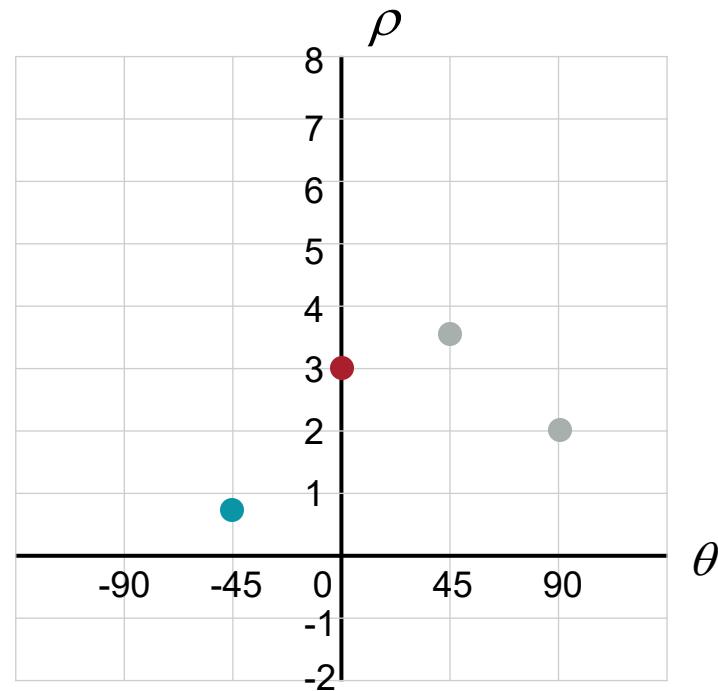
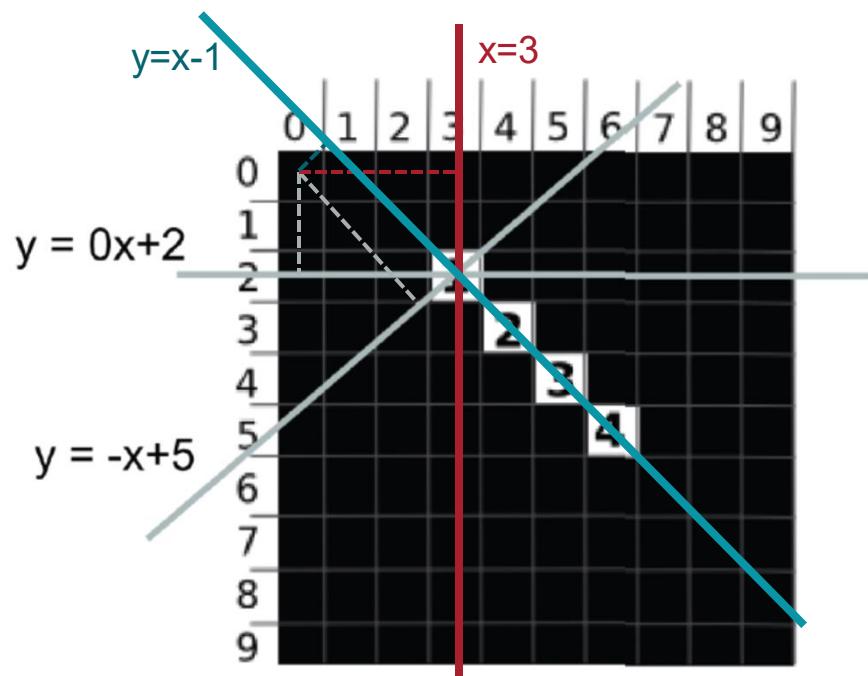
Line Representation

$$f(x, y, \rho, \theta) = x \cos \theta + y \sin \theta - \rho = 0$$

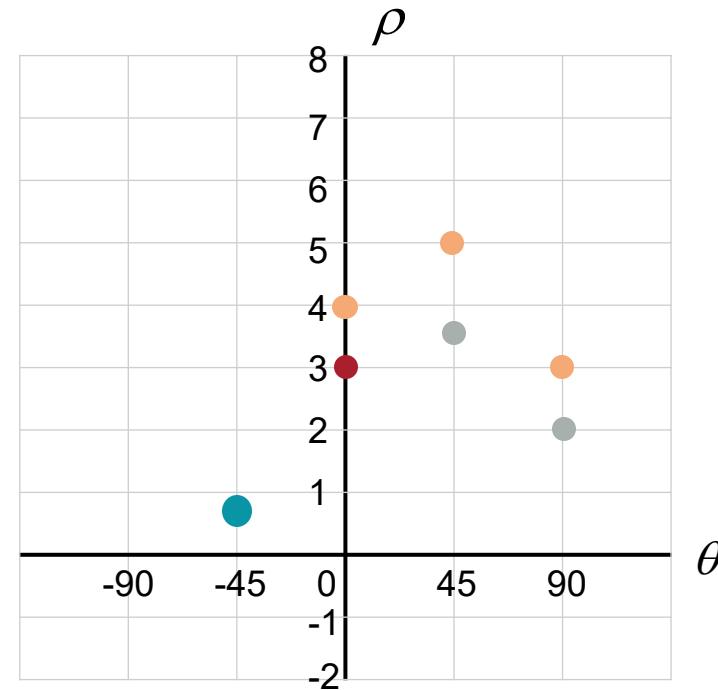
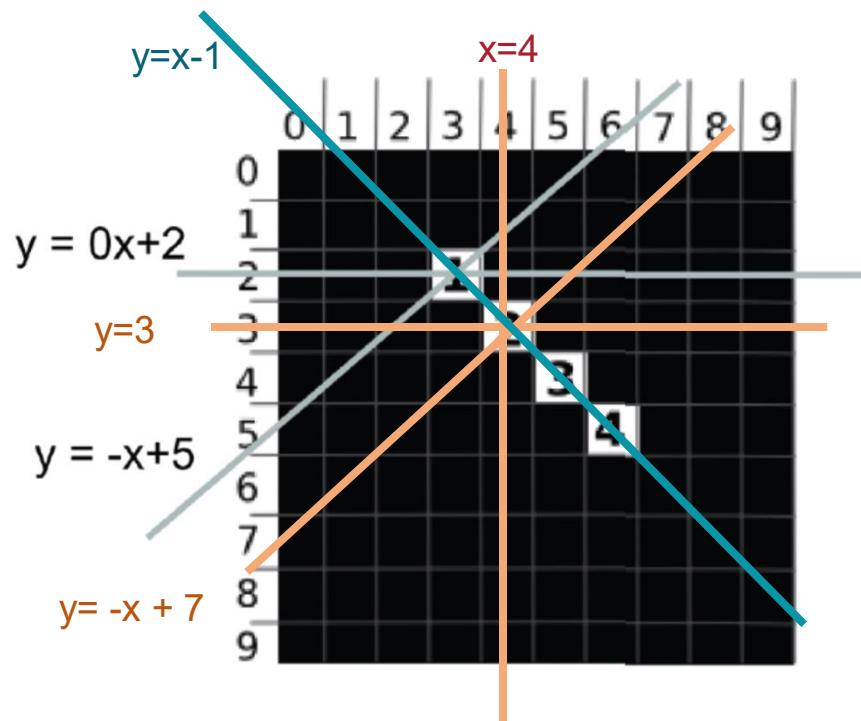


Noise can lead to spurious peaks in the accumulator array

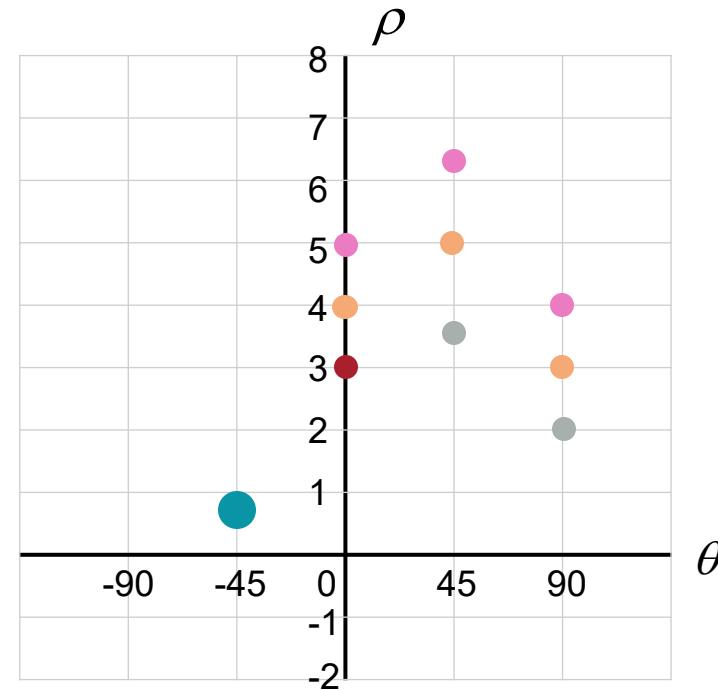
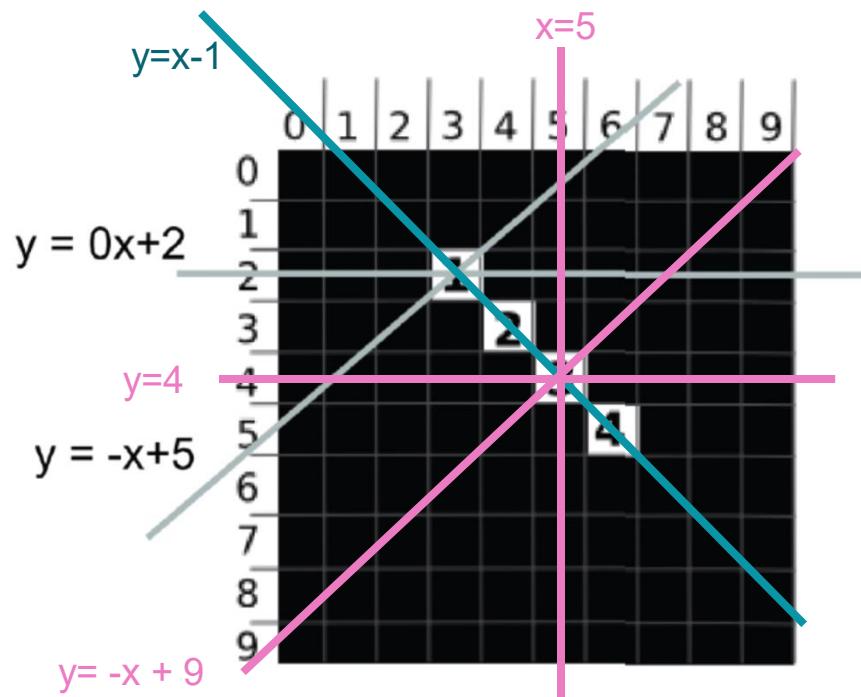
Building the Hough Space



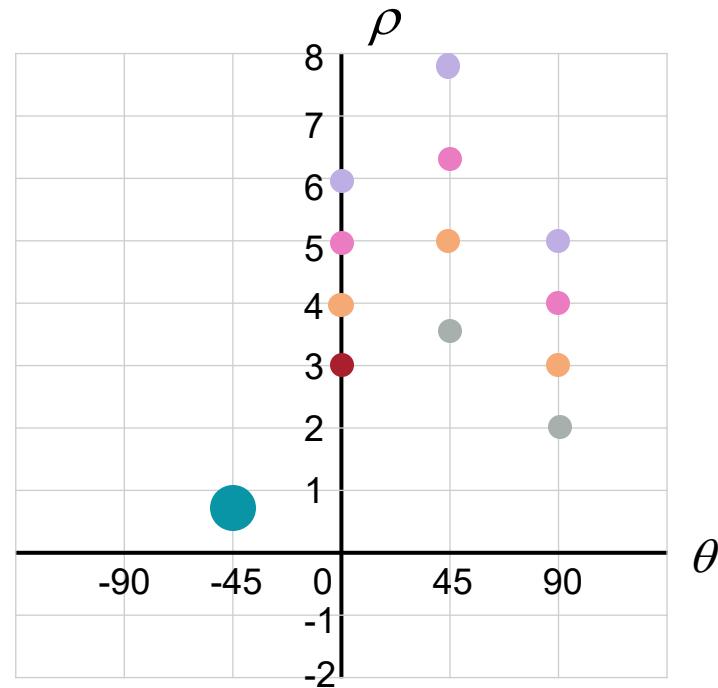
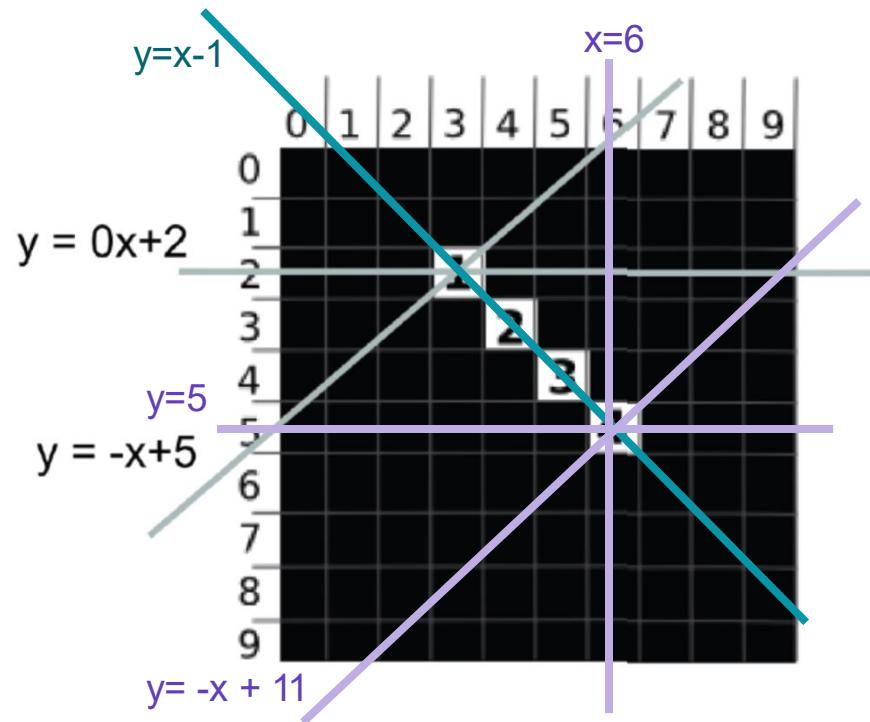
Building the Hough Space



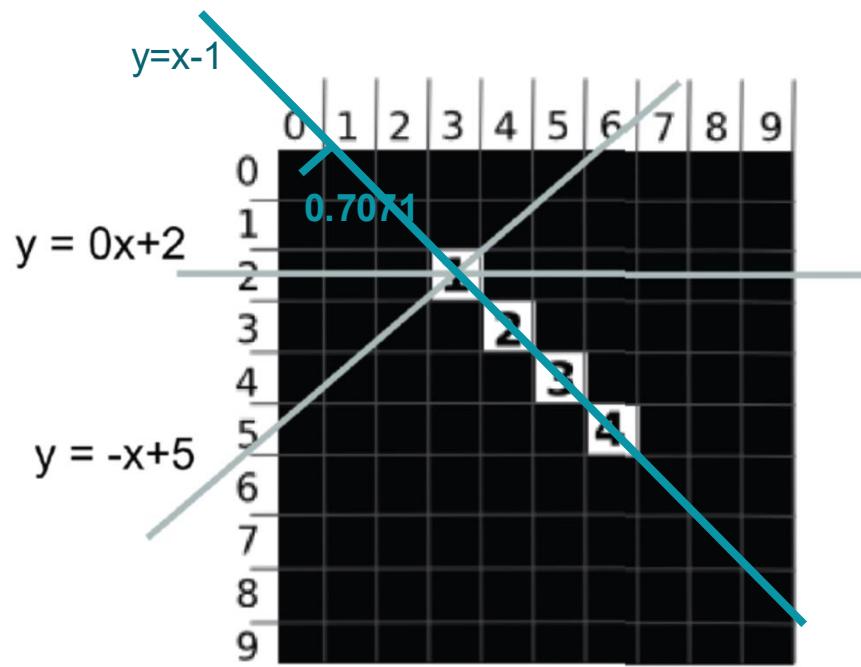
Building the Hough Space



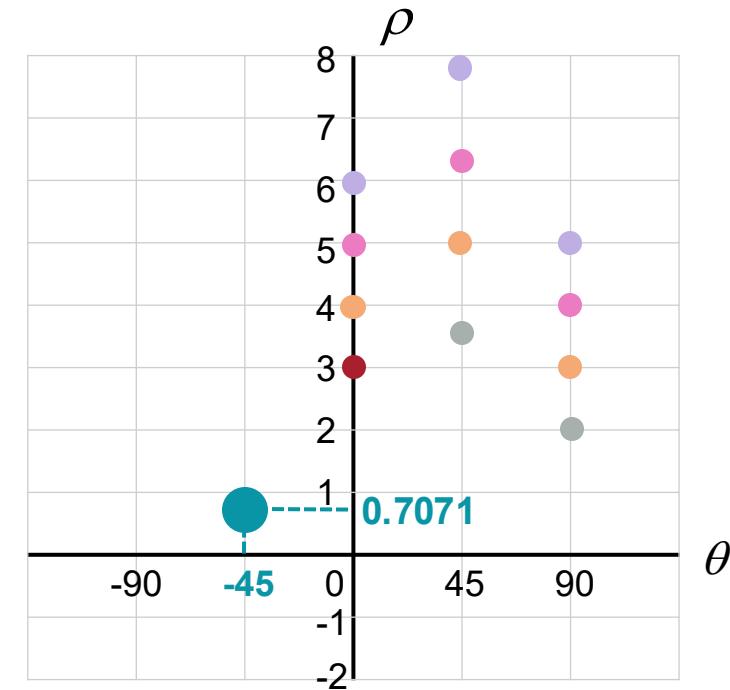
Building the Hough Space



Building the Hough Space



$$\rho = x \cos \theta + y \sin \theta$$



$$0.7071 = x * 0.7071 + y * 0.7071$$

Line Detection Algorithm

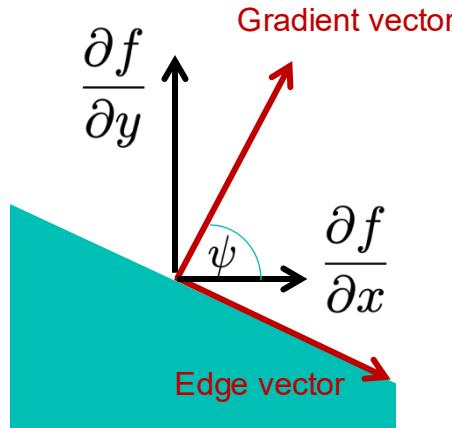
1. Make available an $n = 2$ dimensional array $H(\rho, \theta)$ for the parameter space;
2. Find the gradient image: $G(x, y) = |G(x, y)|\angle G(x, y);$
3. For any pixel satisfying $|G(x, y)| > T_s$, increment all elements on the curve $\rho = x \cos \theta + y \sin \theta$ in the parameter space represented by the H array:

$$\forall \theta \quad | \quad \rho = x \cos \theta + y \sin \theta$$

$$H(\rho, \theta) = H(\rho, \theta) + 1;$$

4. In the parameter space, any element $H(\rho, \theta) > T_h$ represents a straight line detected in the image.

Line Detection using Gradient Information



1. Make $n = 2$ dimensional array $H(\rho, \theta)$
2. Find the gradient image: $G(x, y) = |G(x, y)|\angle G(x, y);$
3. For any pixel satisfying $|G(x, y)| > T_s,$

$$\forall \theta \quad | \quad \angle G(x, y) - \Delta\theta \leq \theta \leq \angle G(x, y) + \Delta\theta$$

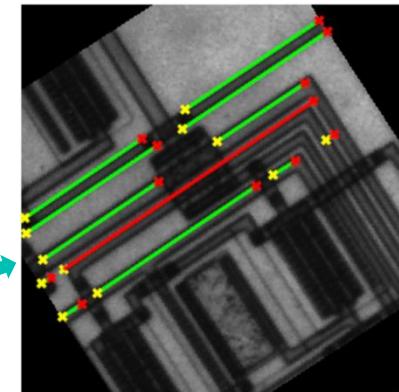
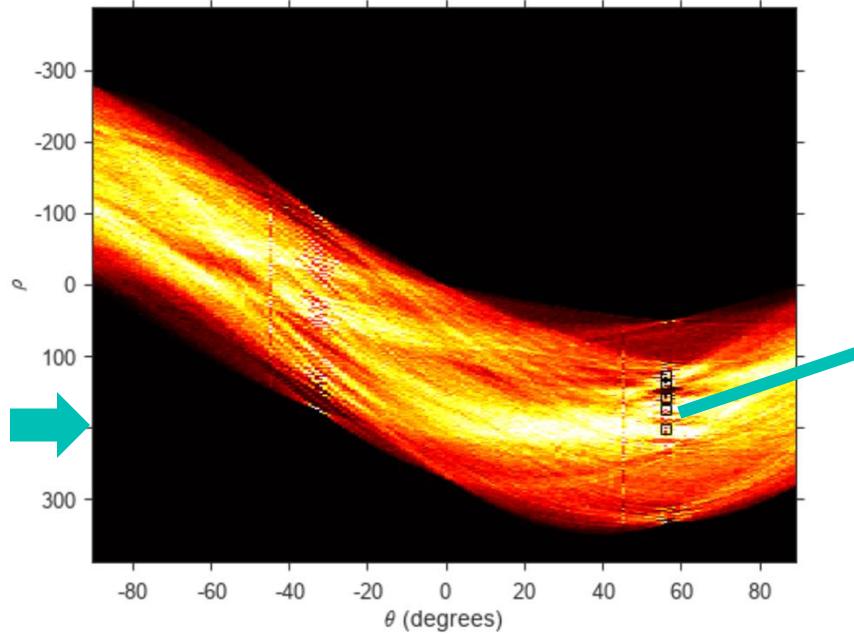
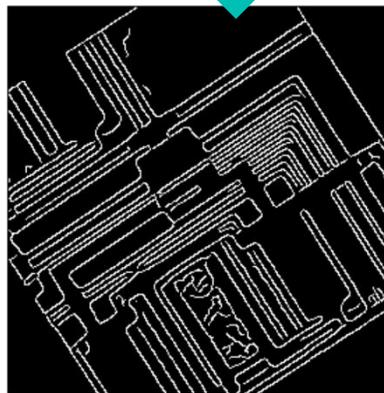
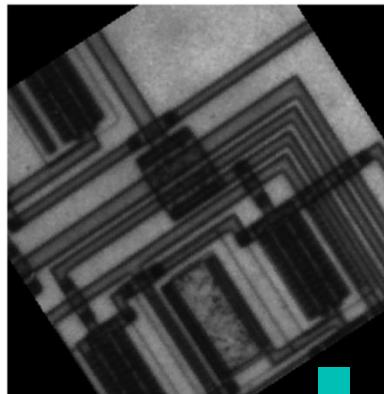
$$\rho = x \cos \theta + y \sin \theta$$

$$H(\rho, \theta) = H(\rho, \theta) + 1;$$

where $\Delta\theta$ defines a small range in θ to allow some room for error in $\angle G$.

- . Any element $H(\rho, \theta) > T_h$ represents a straight line

Line Detection Example



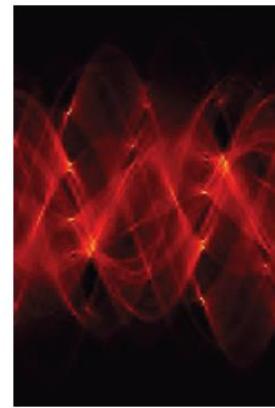
Line Detection Example



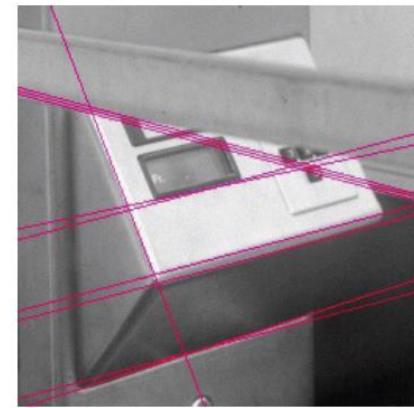
Original



Edges



Parameter
Space



Hough Lines

Circle Detection Algorithm

1. For any pixel satisfying $|G(x, y)| > T_s$, increment all elements satisfying the two simultaneous equations

$$\forall r, \quad \begin{cases} x_0 = x \pm r \cos \angle G \\ y_0 = y \pm r \sin \angle G \end{cases}$$

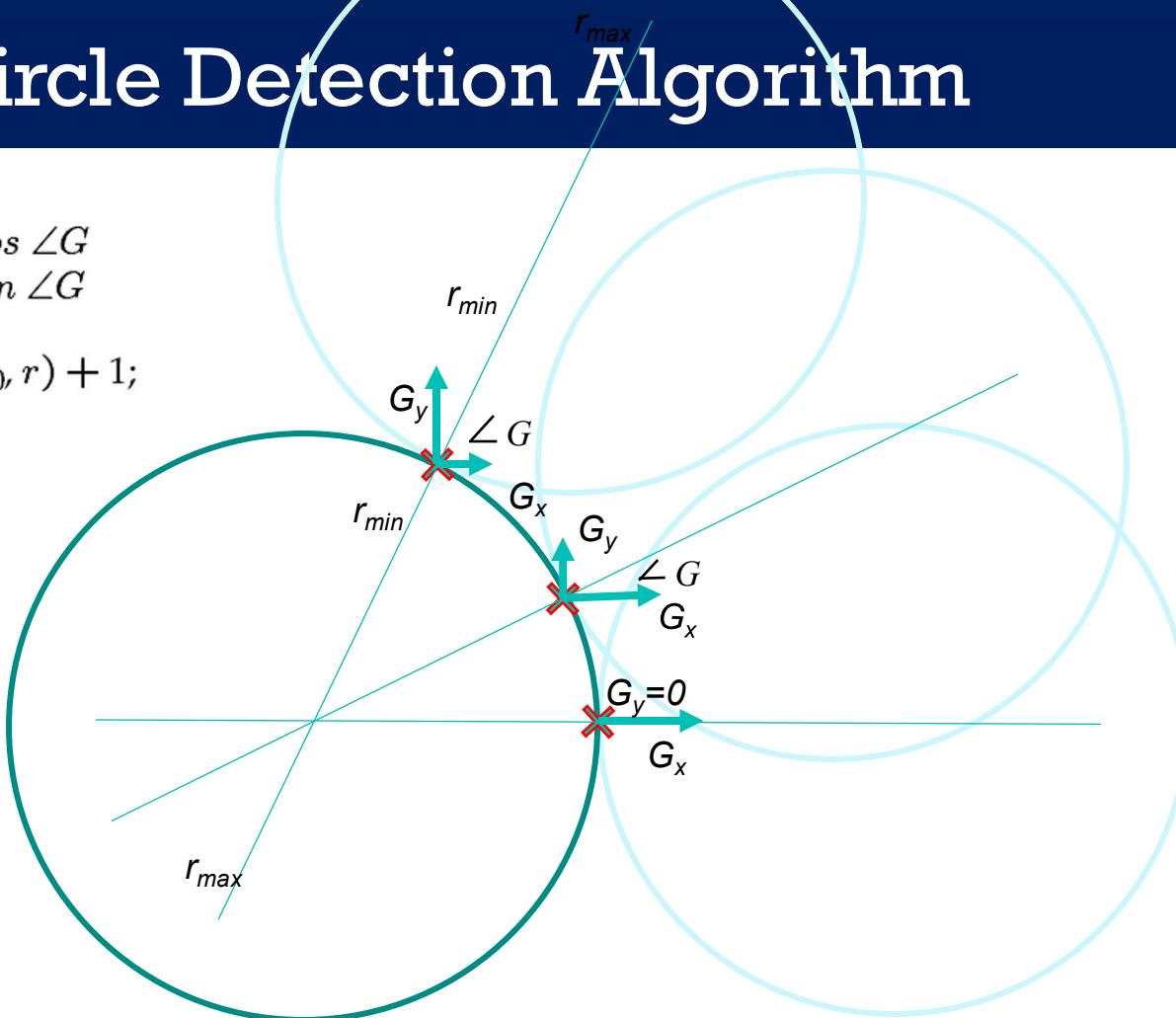
$$H(x_0, y_0, r) = H(x_0, y_0, r) + 1;$$

2. In the parameter space, any element $H(x_0, y_0, r) > T_h$ represents a circle with radius r located at (x_0, y_0) in the image.

Circle Detection Algorithm

$$\forall r, \begin{cases} x_0 = x \pm r \cos \angle G \\ y_0 = y \pm r \sin \angle G \end{cases}$$

$$H(x_0, y_0, r) = H(x_0, y_0, r) + 1;$$

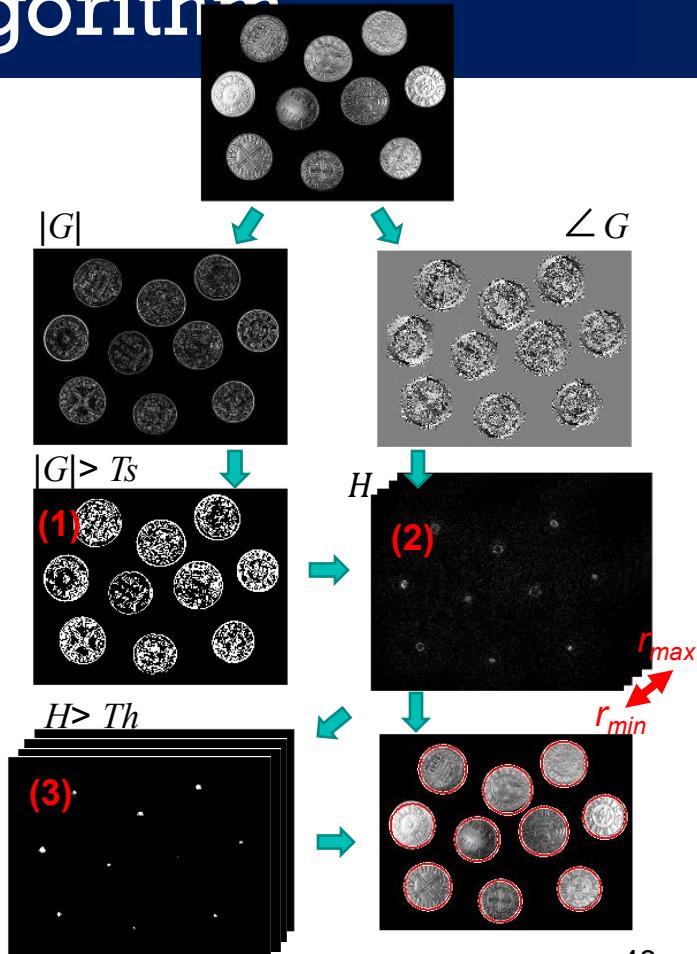


Circle Detection Algorithm

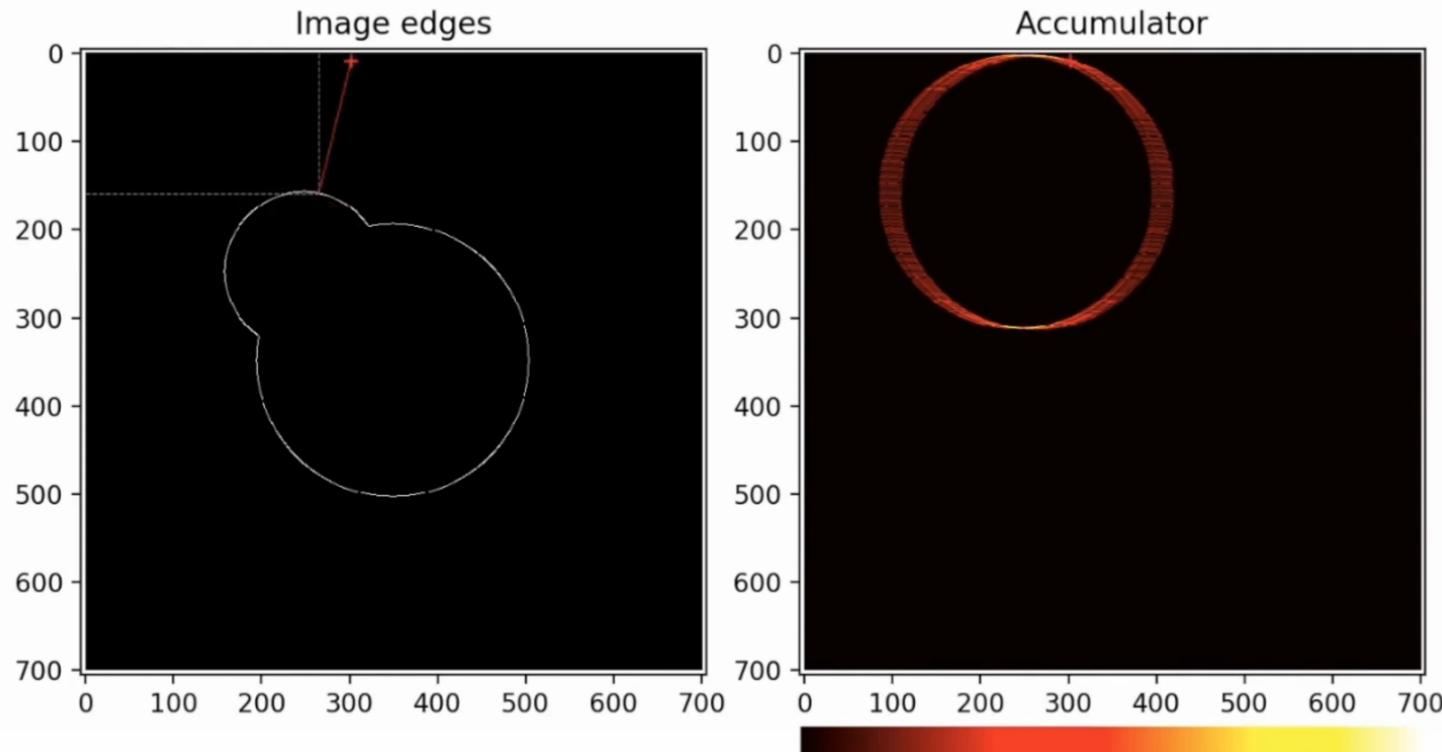
- (1) 1. For any pixel satisfying $|G(x, y)| > T_s$, increment all elements satisfying the two simultaneous equations

$$(2) \quad \forall r, \quad \begin{cases} x_0 = x \pm r \cos \angle G \\ y_0 = y \pm r \sin \angle G \end{cases}$$
$$H(x_0, y_0, r) = H(x_0, y_0, r) + 1;$$

- (3) 2. In the parameter space, any element $H(x_0, y_0, r) > T_h$ represents a circle with radius r located at (x_0, y_0) in the image.

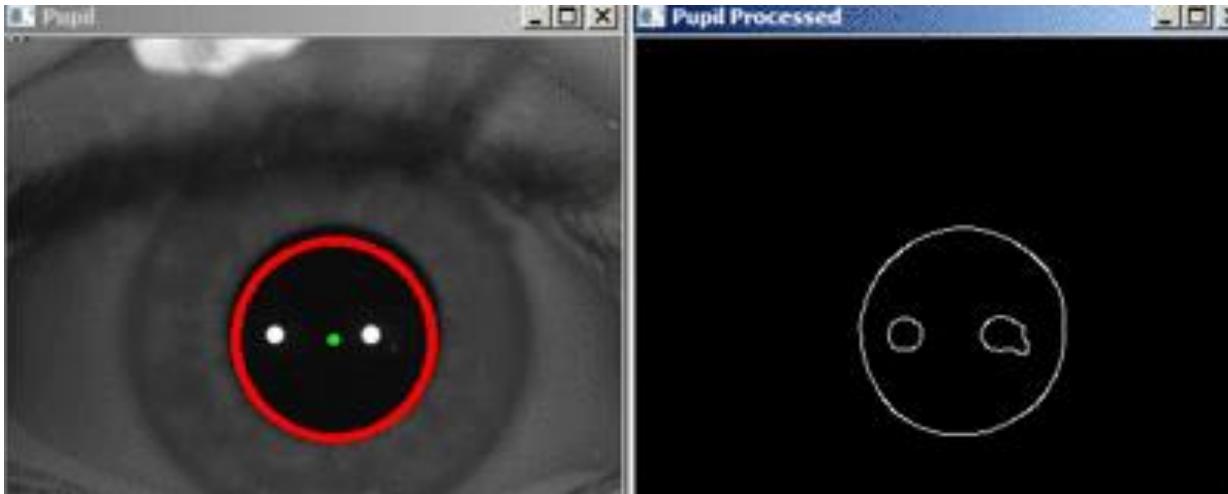


Circle Detection Algorithm



Summary: Shape Recognition via Hough Transform

- Image space is transformed into a **parameter space**
- **Voting procedure** is carried out in the parameter space.
- Object candidates are obtained as **local maxima**.



Source: <https://stackoverflow.com/questions/22274030/detect-objects-similar-to-circles/22270172>

Next Lecture

Image Segmentation



COMS30030 - Image Processing and Computer Vision



Lecture 05
**Segmentation -
The Basics**

Majid Mirmehdi
majid@cs.bris.ac.uk

Examples of Image Segmentation

- **Image Segmentation ...**

... is the process of spatial subsectioning of a (digital) image into multiple partitions of pixels (i.e. segments or regions) according to given criteria.

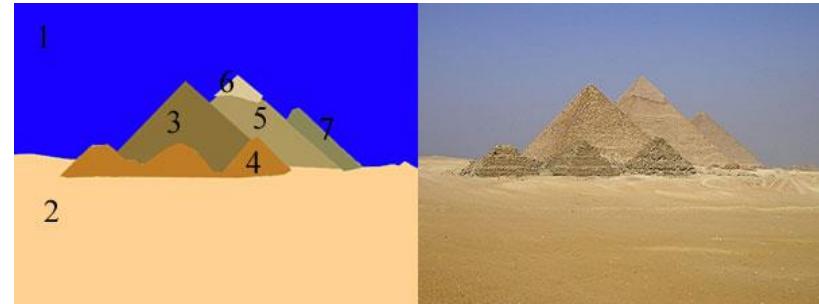


Example: segmentation of an image into locally coherent regions

Motivation: Why Segment Images?

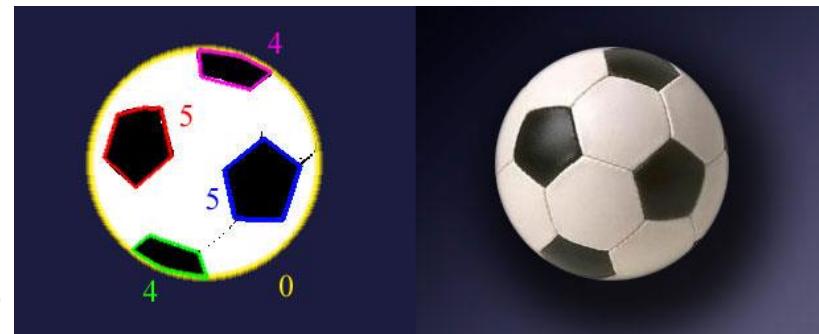
- Image Simplification

- an image may contain millions of pixels but only a few regions



- Higher-level Object Description

- regions tend to belong to the same class of object
- regions may provide object properties (e.g. shape, colour, ...)

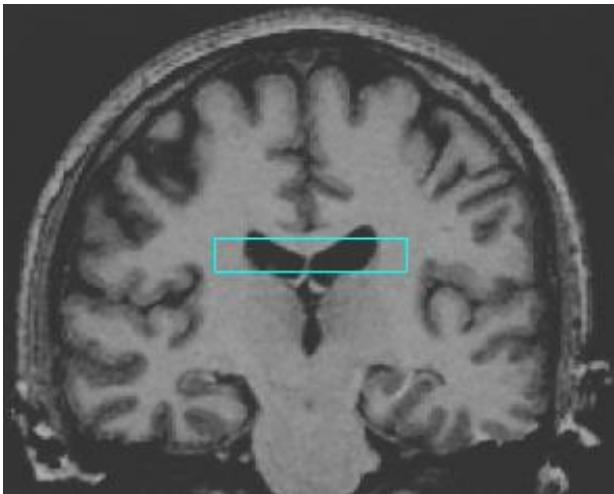


- Input for Content Classifiers

- region descriptions can be input data for higher level classifiers, e.g. Bayesian Classifiers or Neural Networks.



Why Segment Images?



Examples from <https://medium.com/cogitotech> and Alberto Pretto

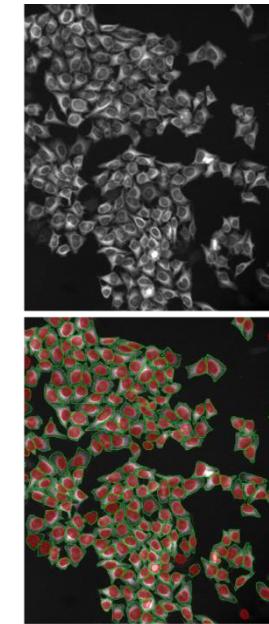
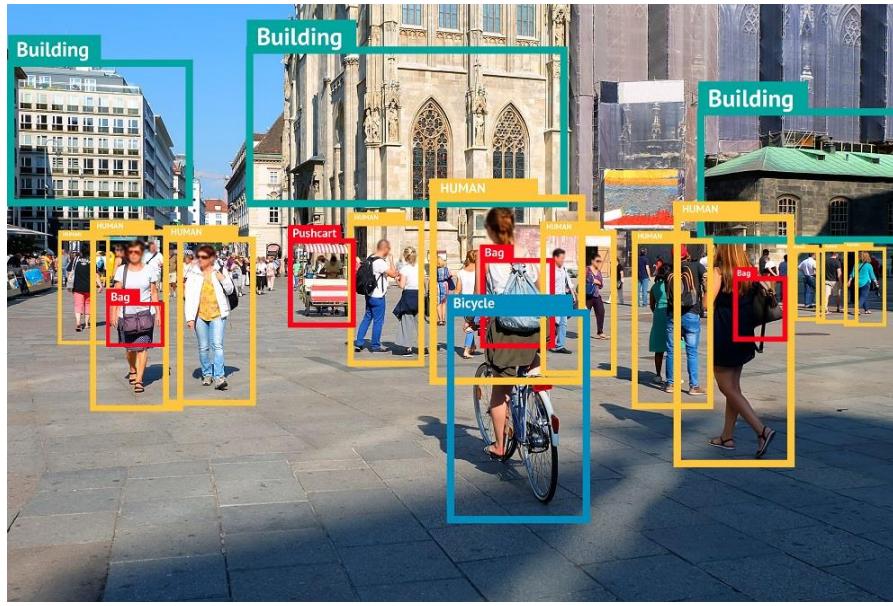
Grouping Pixels

Goals:

- Gather pixels/features that belong together
- Obtain an intermediate representation that compactly describes key image (video) parts

Top-down vs. bottom-up segmentation

- Top-down: pixels belong together because they are from the same object
- Bottom-up: pixels belong together because they look similar



Hard to measure success: what is interesting depends on the application.

Example of Over-Segmentation

Original image



Over-segmentation



Over-segmentation: pixels belonging to the same region [object] are classified as belonging to different regions [objects]

Example of Under-Segmentation

Original image



Under-segmentation



Under-segmentation: pixels belonging to different regions [objects] are classified as belonging to the same region [object]

So many segmentation methods...

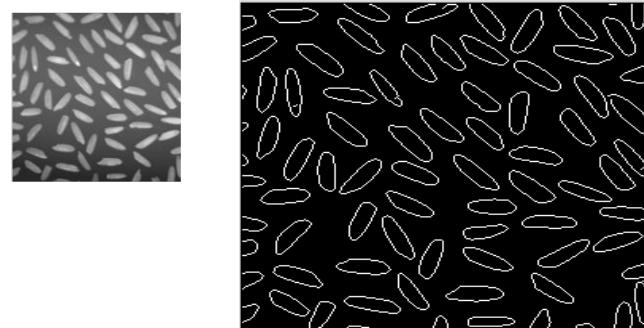
Thresholding Methods

- pixels are categorized based on intensity
- only useful when sufficient contrast exists



Edge-based Methods

- region boundaries are constructed from edgemaps



Region-based Methods

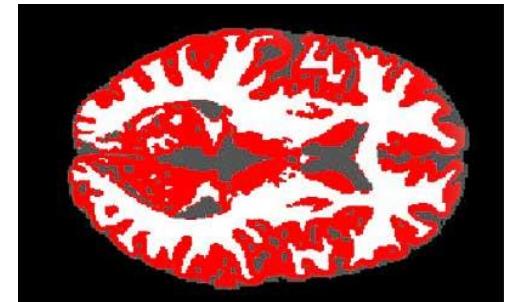
- region growing from seed pixels
- **region splitting and merging for efficient spatial encoding**



So many segmentation methods...

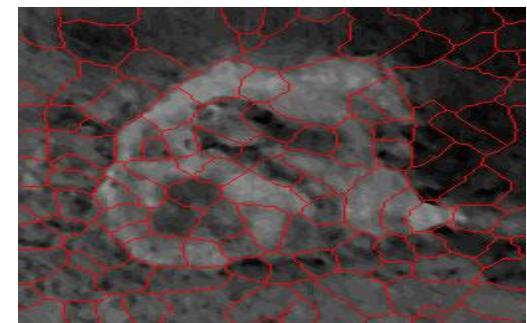
Clustering and Statistical Methods

- global, often histogram based image partitioning, e.g. **K-means**, Gaussian Mixture Model



Topographic Methods (out of scope in this unit)

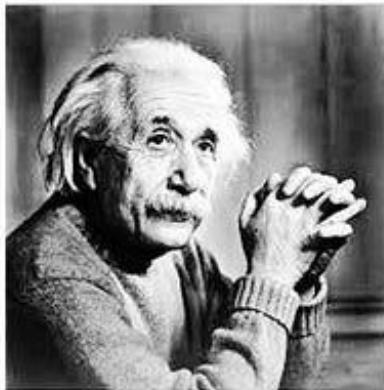
- stepwise simplifications that take spatially wider (topographical) image configurations into account e.g. watershed transform, variational based methods



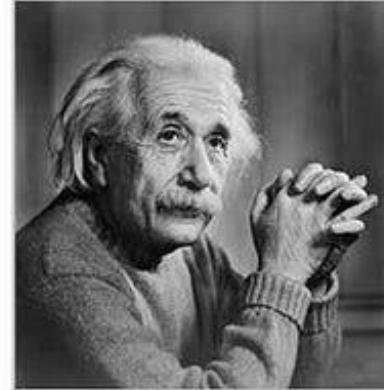
...and many more...

Image Histogram

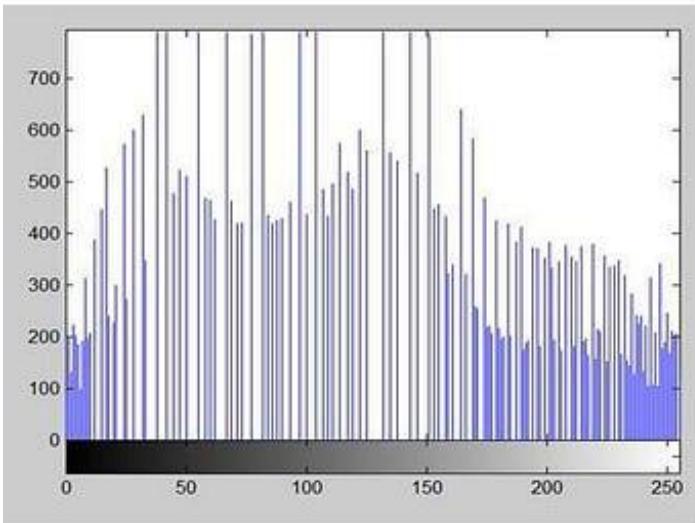
Brighter



Darker



Histogram



Histogram

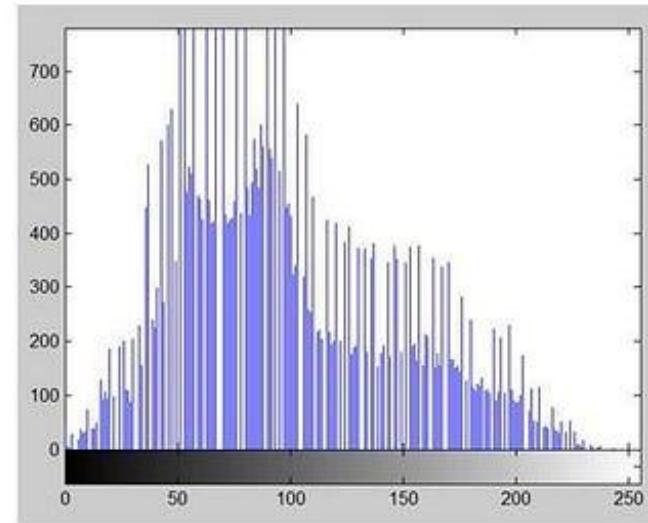
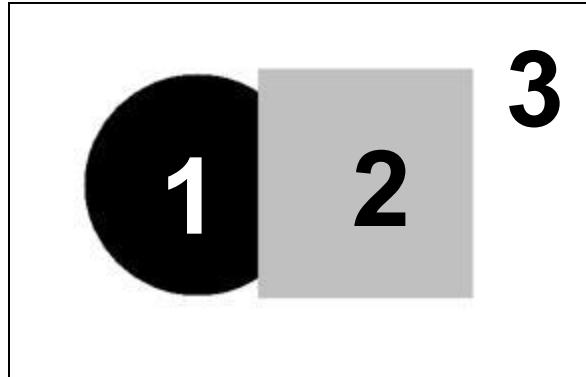
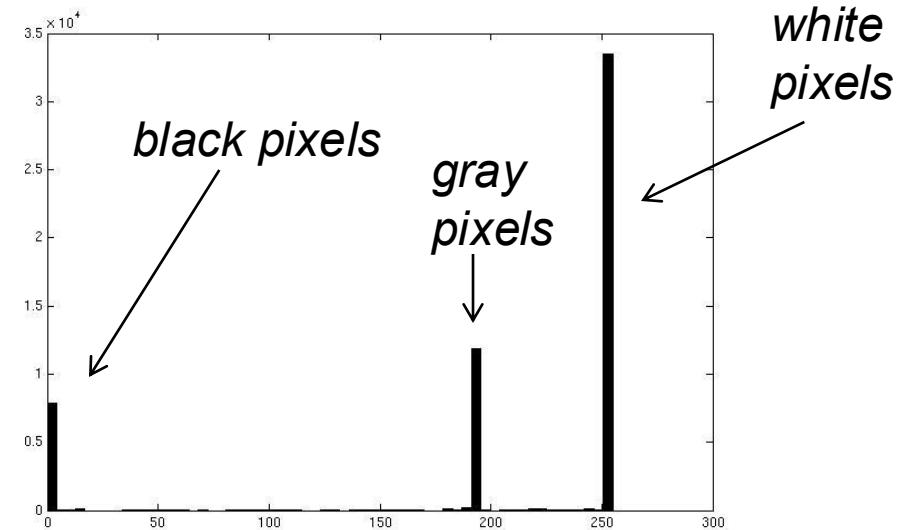


Image segmentation: toy example



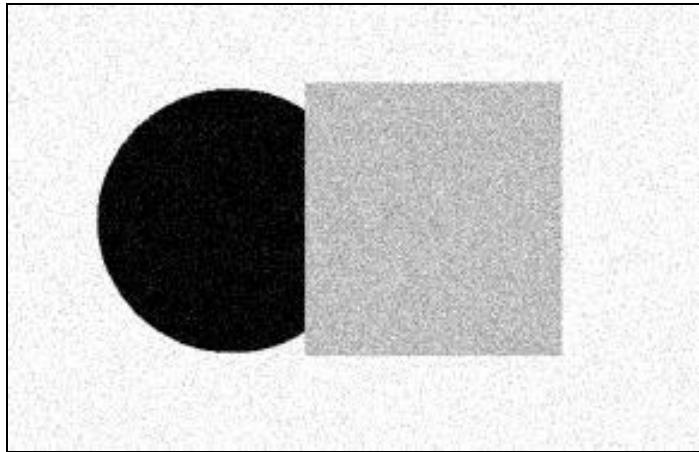
input image



- The intensities define the three groups.
- We could label every pixel in the image according to which of these primary intensities it is.
 - i.e., *segment* the image based on the intensity feature.

Simple thresholding not enough!

What if the image isn't quite so simple?



input image

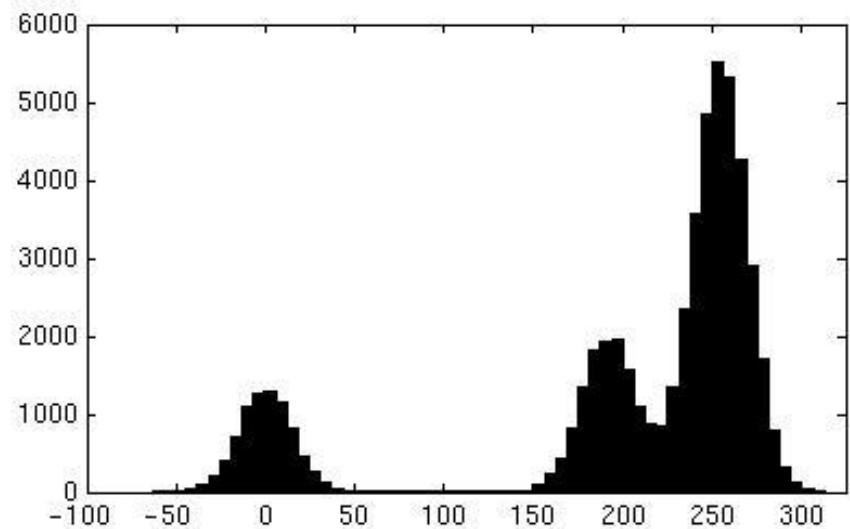
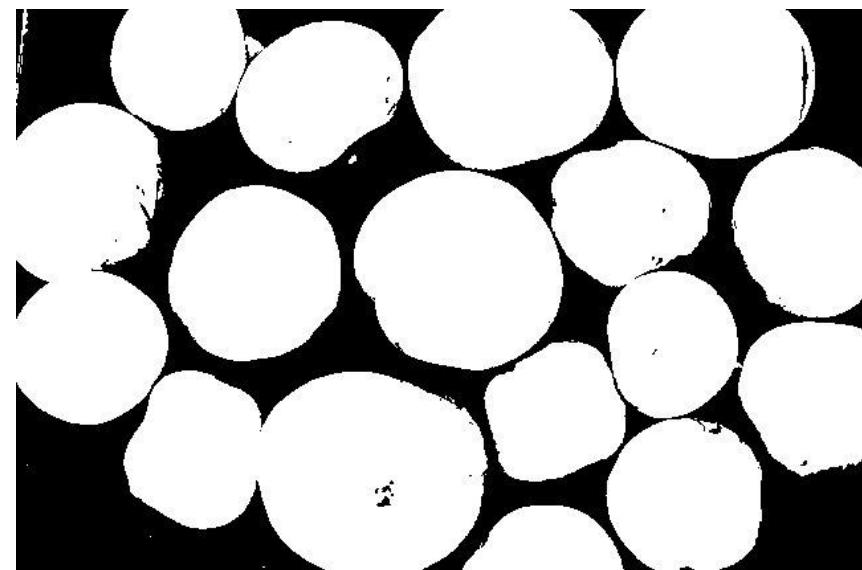
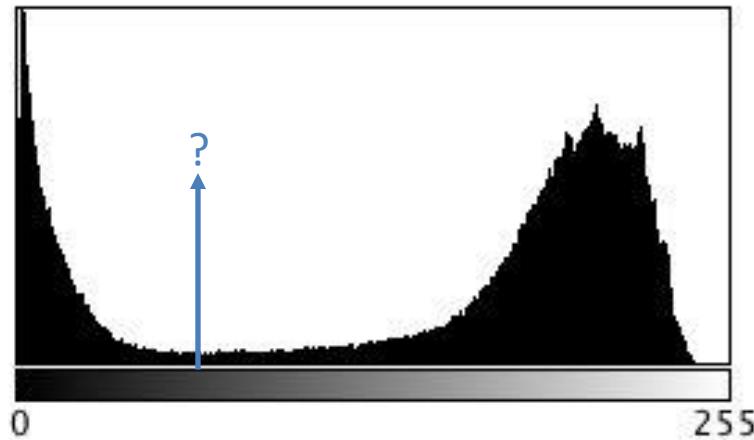


Image Segmentation

Perfect segmentation is difficult to achieve:

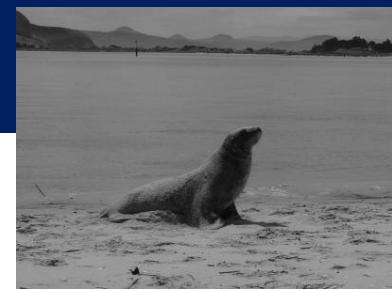
- a pixel may straddle the “real” boundary of objects such that it partially belongs to two or more objects
- effects of noise, non-uniform illumination, occlusions etc. give rise to the problem of *over-segmentation* and *under-segmentation*



Images from craftofcoding.wordpress.com

Thresholding Example

- If the image contains a dark object on a light background
 - choose a threshold value, T
 - for each pixel
 - if the brightness at that pixel is less than T , it is a pixel of interest
 - otherwise it is part of the background
- The value of the threshold is very important
 - if too high \rightarrow background pixels classified as foreground
 - If too low \rightarrow foreground pixels classified as background



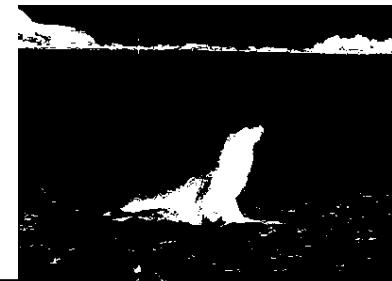
$T = 128$



$T = 96$



$T = 64$

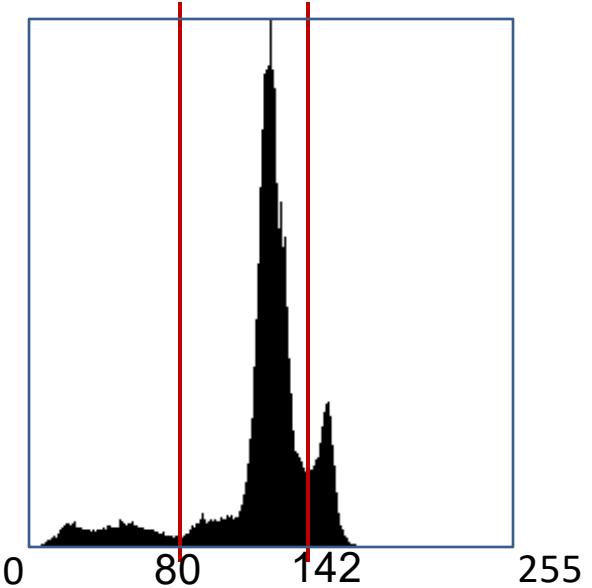


Using Histograms to Stipulate Regions

Maybe apply multiple thresholds?

The seal image shows three regions

- one below $T_1 = 80$
- one above $T_2 = 142$
- one between the two thresholds



Iterative Threshold Selection Algorithm

1. Select an initial estimate for the threshold T

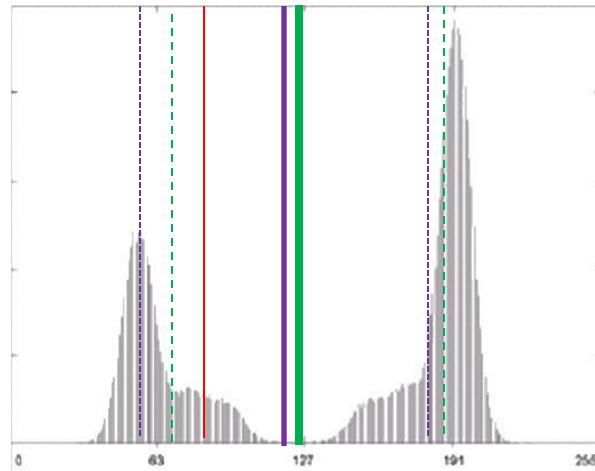
2. Segment the image using T .

This will produce two groups of pixels: G_1 , consisting of all pixels with grey levels $> T$ and G_2 , consisting of pixels with grey values $\leq T$.

3. Compute the average grey level values m_1 and m_2 for the pixels in regions G_1 and G_2 .

4. Compute a new threshold value: $T = (m_1 + m_2)/2$

5. Repeat steps (2.) through (4.) until convergence



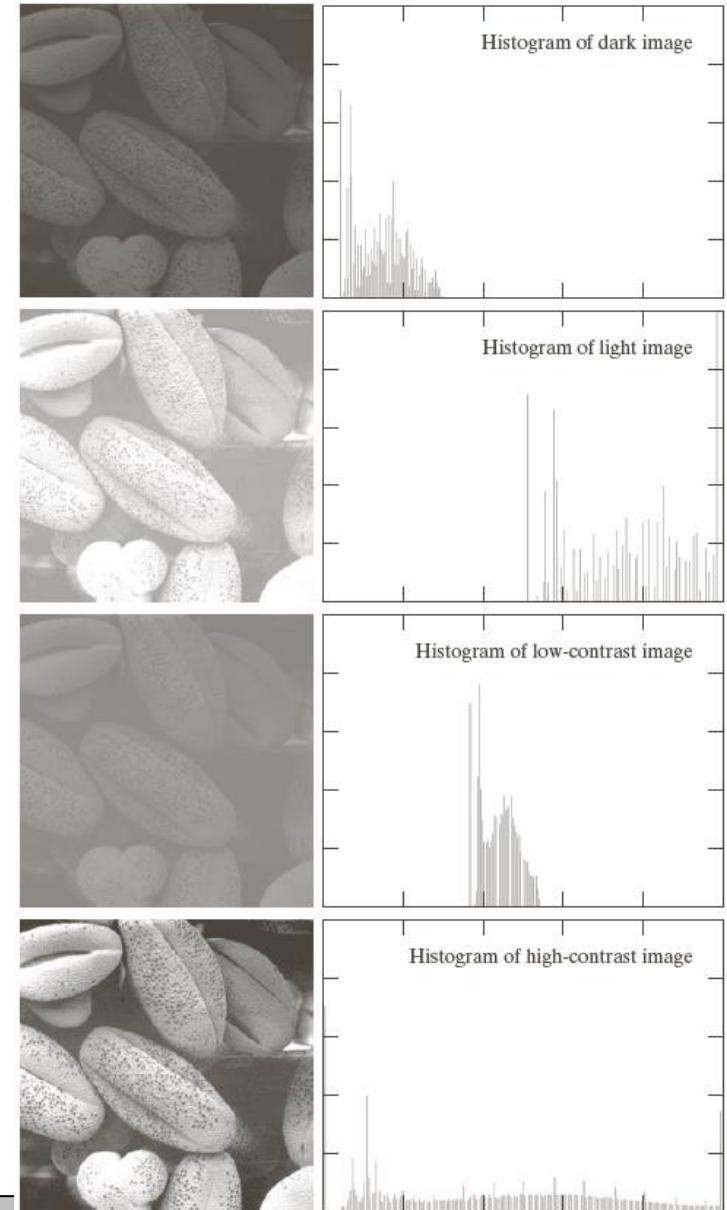
- initial estimate
- - - average values (round 1)
- - - average values (round 2)
- threshold after round 1
- threshold after round 2



Thresholding for Segmentation

Not always a good solution!

Four problematic image types: dark, light, low contrast, and high contrast.

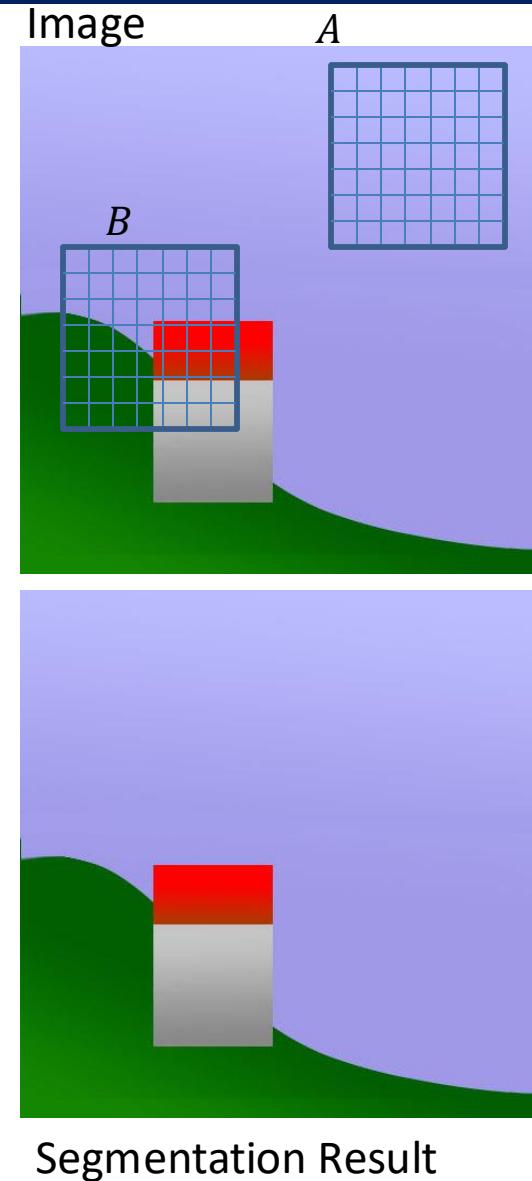


Split & Merge Segmentation – Divide & Conquer

Homogeneity function H

$$H(\text{Region } A) = 1 \quad (\text{homogeneous})$$

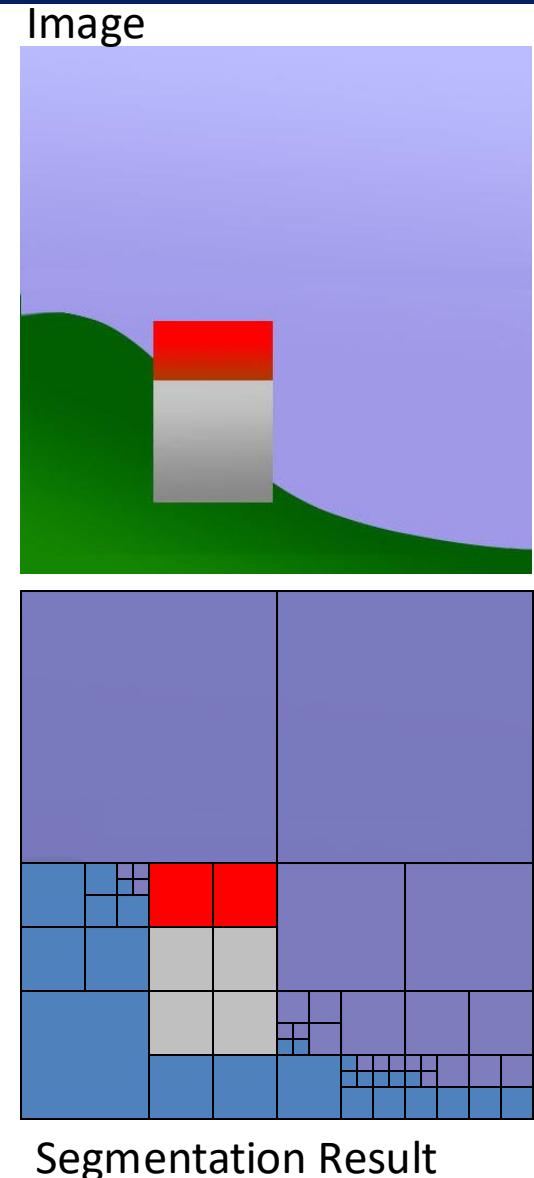
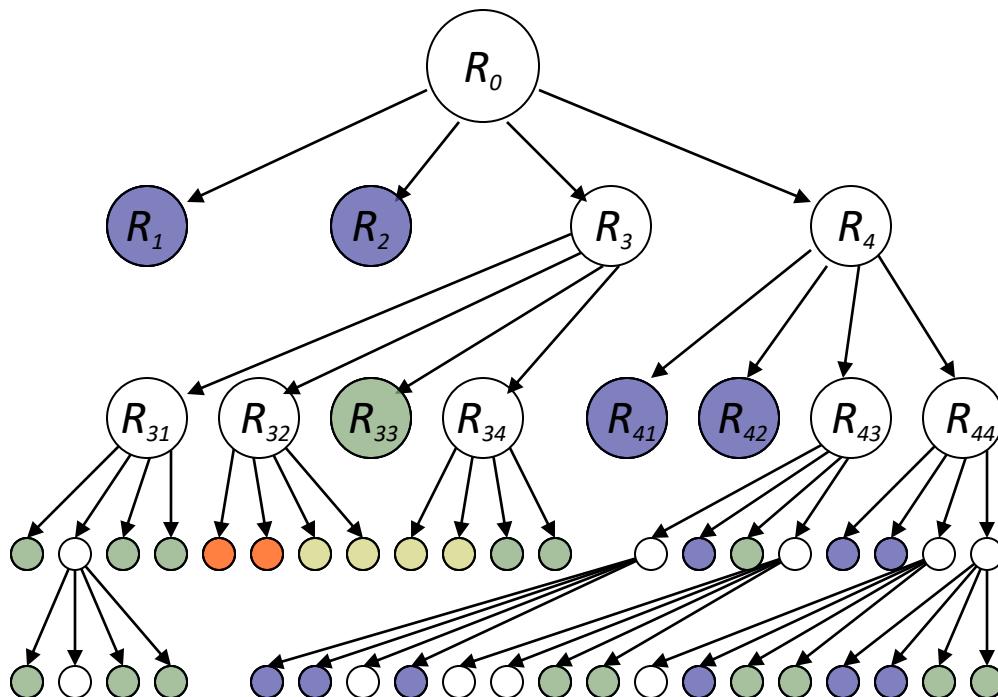
$$H(\text{Region } B) = 0 \quad (\text{inhomogeneous})$$



Segmentation Result

Split & Merge Segmentation – Divide & Conquer

1. Start with R_0 that represents the entire image
2. If $H(R_i) = 0$ (inhomogeneous) then
{split area into 4 blocks (quadtree splitting) and process each area with step (2.)}
3. Merge all subregions that pairwise satisfy
 $H(R_i \cup R_j) = 1$ (homogenous)



Split & Merge – Summary

Conceptual Summary:

- Iteratively decompose an image into regions of a maximally sized selected shape (e.g. rectangle) that do not satisfy a homogeneity condition. (split step)
- Then merge regions that together satisfy a homogeneity condition. (merge step)

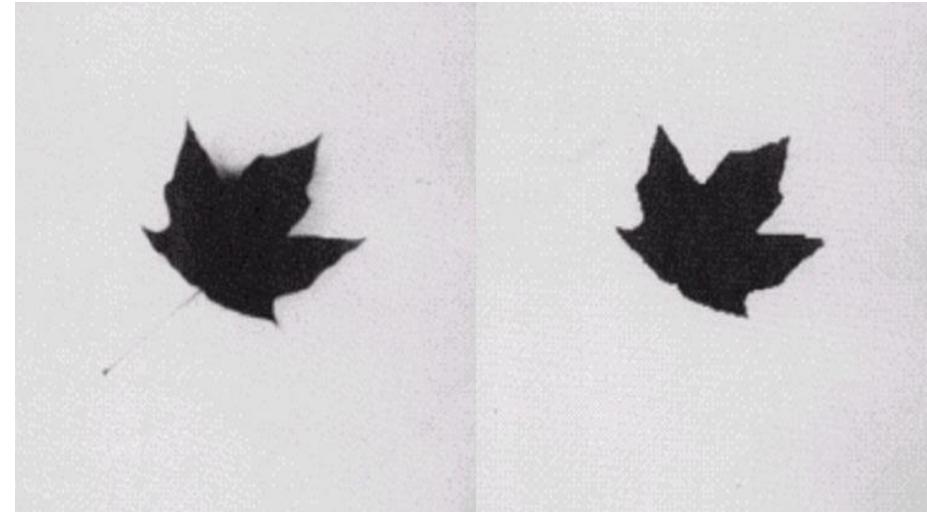
Some Comments:

- Using quadtrees, the results of split and merge tend to be *blocky*.
- Can have an adaptive homogeneity condition that, for instance, changes depending on the region size.

Example H

- $H(R_i)=1$ if at least 80% of the pixels in R_i have the property $|z_j - m_i| < 2\sigma_i$ where z_j is the grey level of the j^{th} pixel in R_i , m_i is the mean grey level of the region and σ_i is the standard deviation of the grey levels in R_i

- If $H(R_i)=1$ then set all the pixels in R_i to value m_i



Original

Result

Split & Merge – Bristol Video Scene Segmentation

Original Video



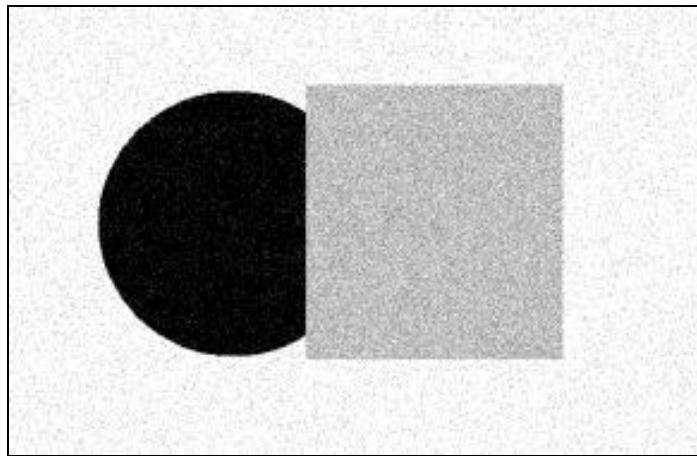
Segmentation Result



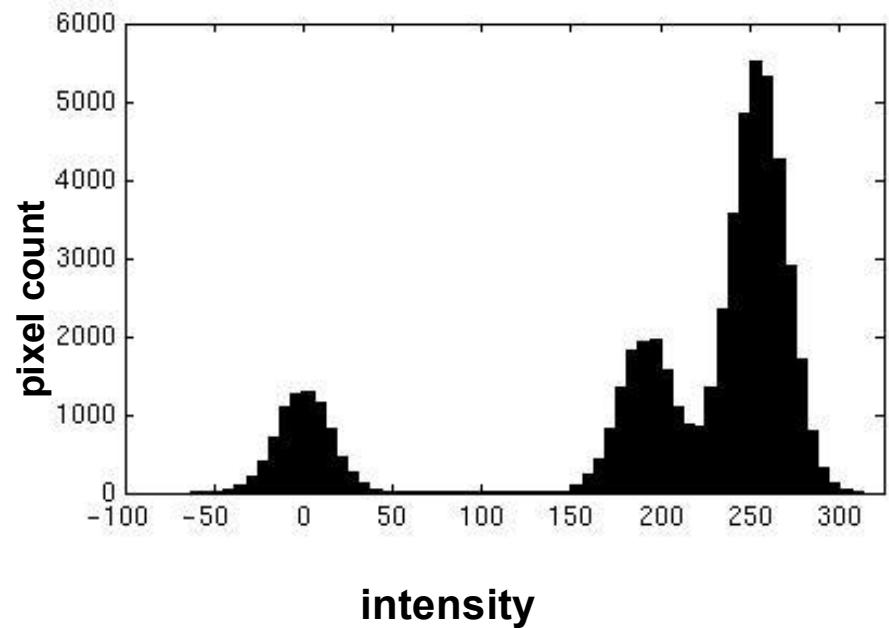
- Images are segmented using a Split-And-Merge technique.
(Note the blocky nature of the regions!)
- Regions are then labelled by a Neural Network to associate the segments with semantics (colouration).
- This project dates back to around 27 years ago!

Image segmentation: toy example

What if the image isn't quite so simple?

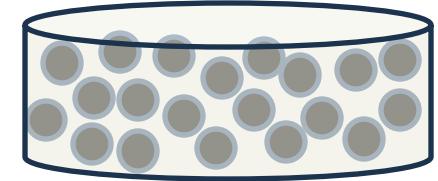


input image

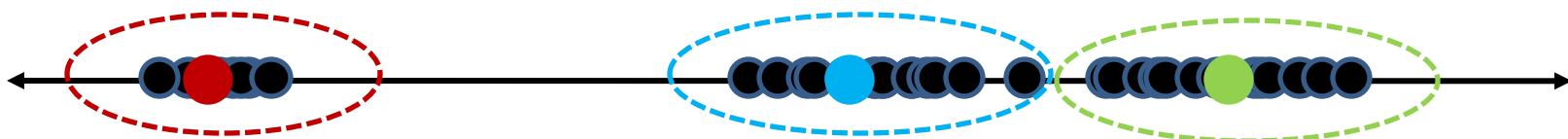


One answer is: use **clustering**...

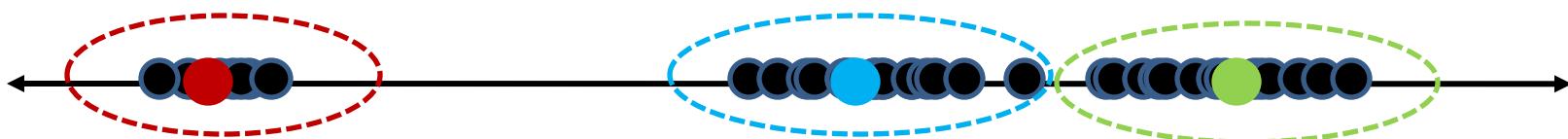
Clustering dilemma



If we knew the **cluster centers**, we could allocate points to groups by assigning each to its closest center.

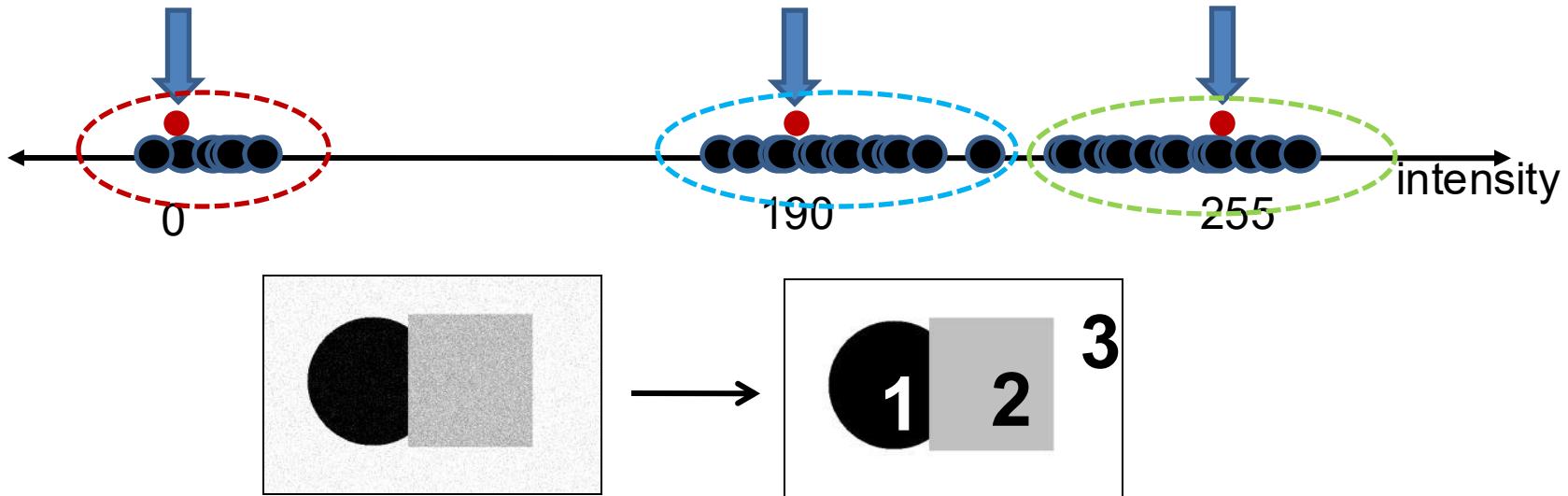


If we knew the **group memberships**, we could get the centers by computing the mean per group.



A “chicken and egg” problem!

Image segmentation: toy example



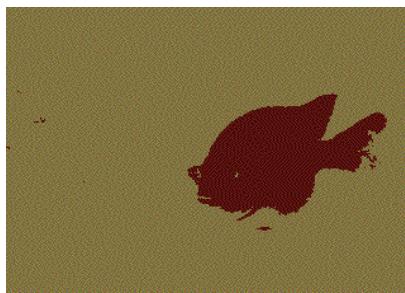
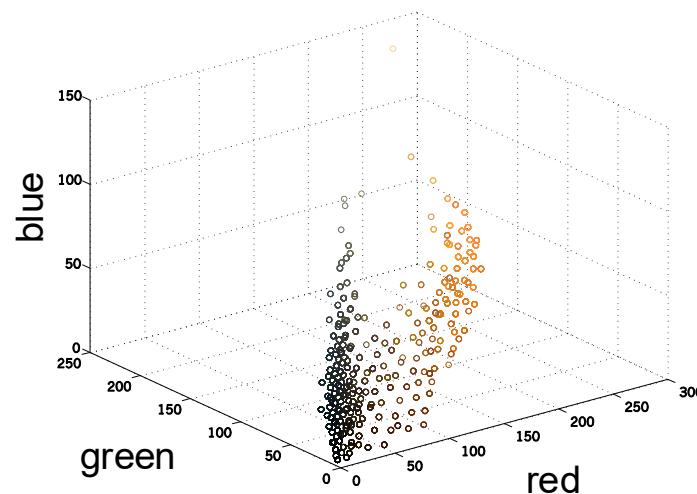
- Goal: choose three “centres” as the **representative intensities**, and label every pixel according to which of these centres it is nearest to.
- Best cluster centres are those that minimize SSD between all points and their nearest cluster centre μ_j

$$\Theta(\text{clusters}, \text{data}) = \sum_{j \in \text{clusters}} \left[\sum_{i \in j^{\text{th}} \text{cluster}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \right]$$

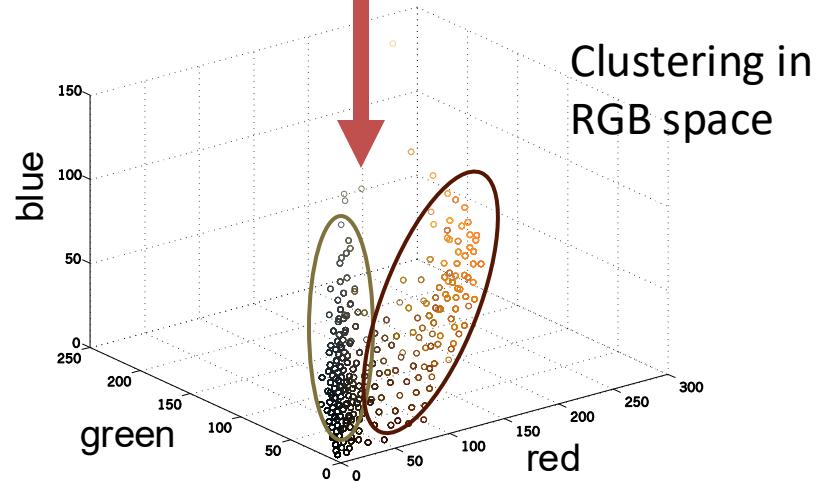
Clustering for image segmentation



map to 3D
RGB space



map back to
pixel space



K-means clustering – theoretical view

- It minimises the following objective function:

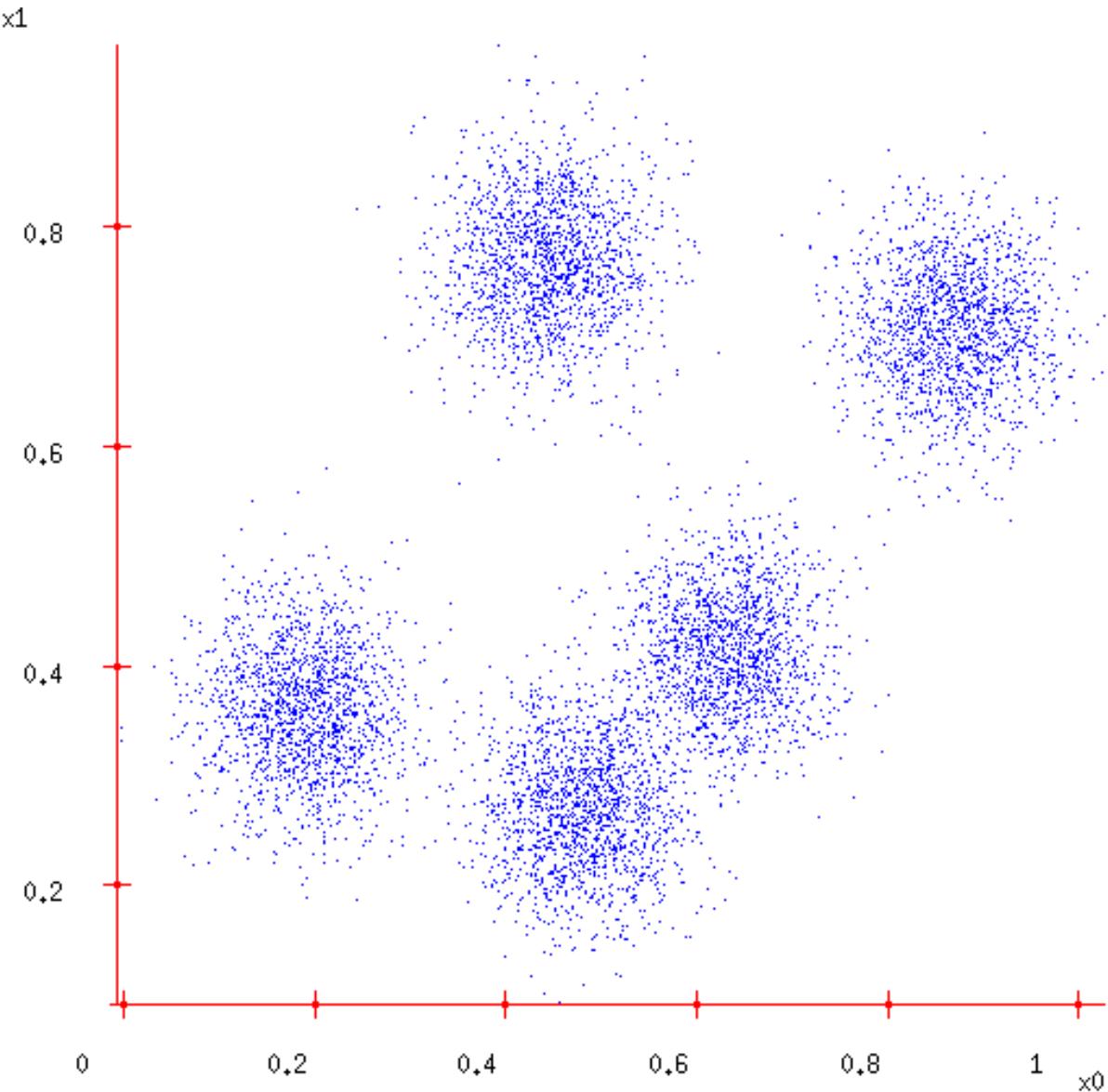
$$\Theta(clusters, data) = \sum_{j \in clusters} \left[\sum_{i \in j^{th} cluster} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2 \right]$$

- An iterative clustering algorithm
 - Pick K random points as cluster centres (means)
 - Iterate:
 - Assign data instances to closest mean
 - Assign each mean to the average of its assigned points
 - Stop when no point's assignment changes



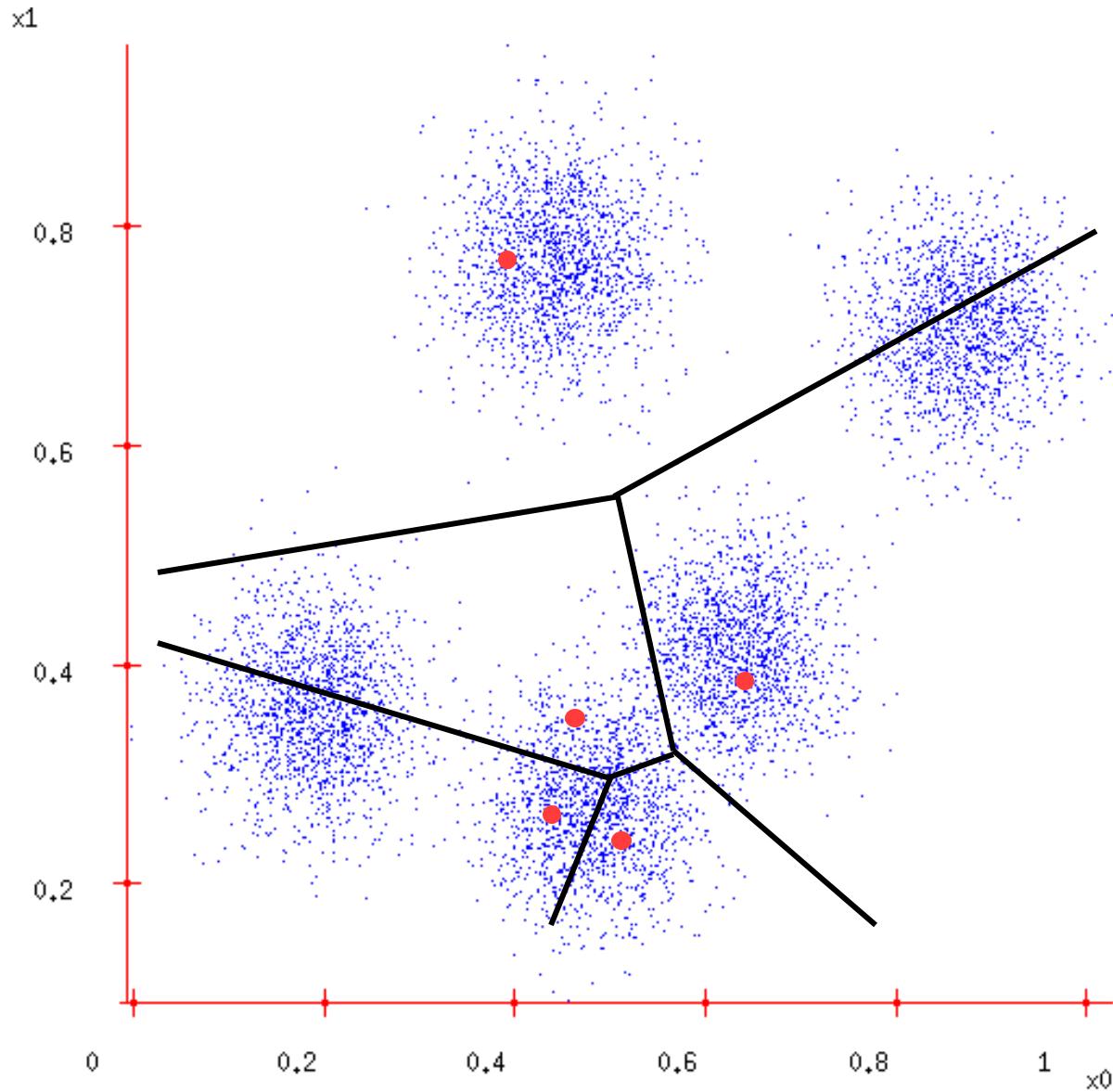
K -means clustering

1. Ask user how many clusters they'd like (e.g., $K=5$)



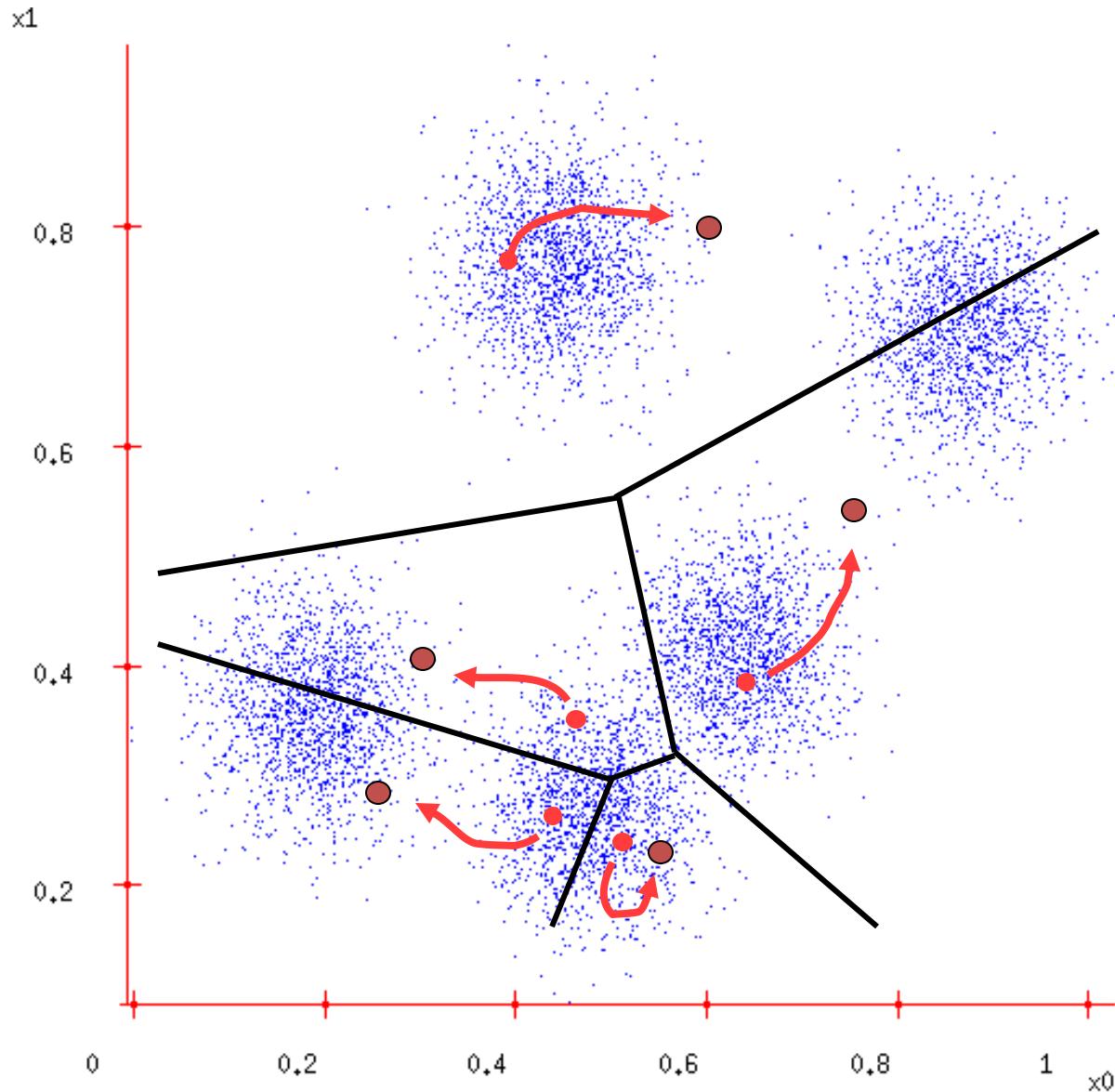
K-means clustering

1. Ask user how many clusters they'd like (e.g., $K=5$)
2. Randomly guess K cluster centre locations ($\mu_1 \dots \mu_K$)
3. Each datapoint finds out which centre it's closest to (thus each centre "owns" a set of datapoints)



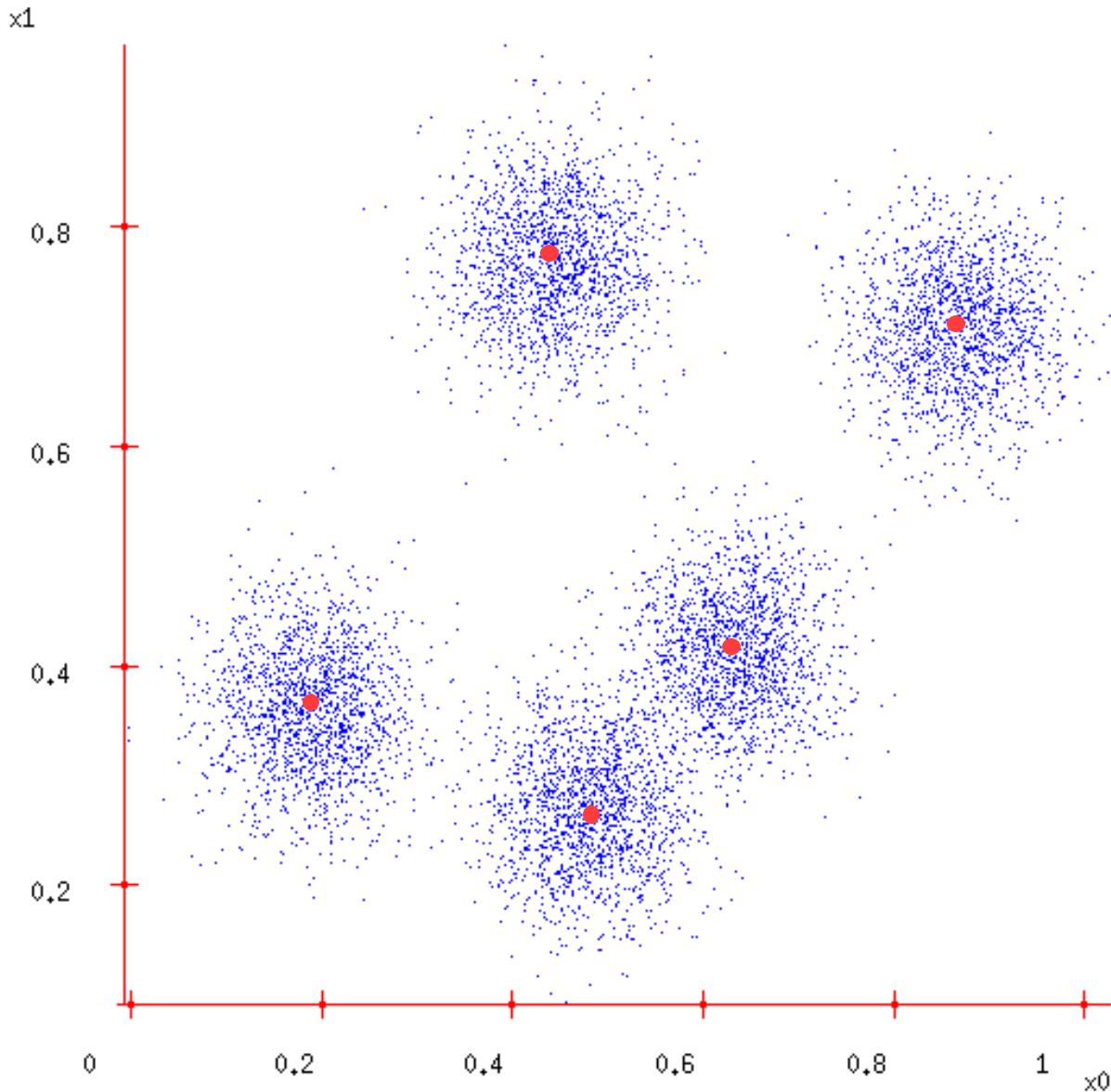
K-means clustering

1. Ask user how many clusters they'd like (e.g., $K=5$)
2. Randomly guess K cluster centre locations ($\mu_1 \dots \mu_K$)
3. Each datapoint finds out which centre it's closest to
4. Each centre finds the centroid of the points it owns...
5. ...and jumps there

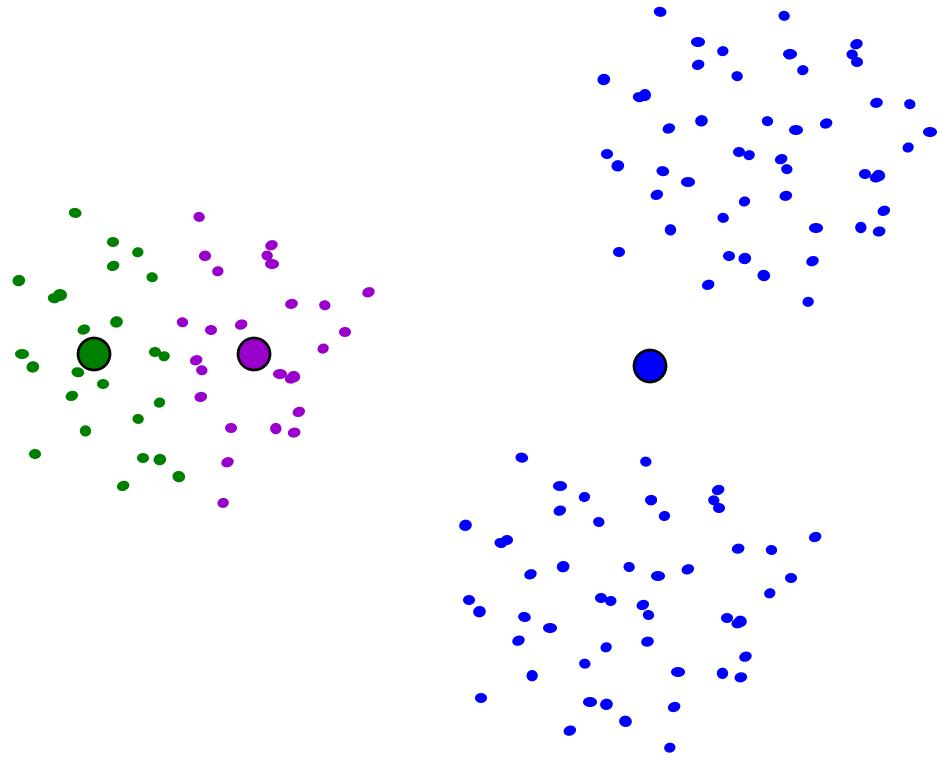


K-means clustering

1. Ask user how many clusters they'd like (e.g., $K=5$)
2. Randomly guess K cluster centre locations ($\mu_1 \dots \mu_K$)
3. Each datapoint finds out which centre it's closest to
4. Each centre finds the centroid of the points it owns...
5. ...and jumps there
6. Repeat from 3 until terminated!



K-means gone wrong!



Reflection on the K-means Algorithm

- **What does it do?**
 - K -means attempts to find a configuration $\mu_1 \dots \mu_K$ that minimises within-cluster scatter: total squared distance between point x_i and centroid μ_j in j^{th} cluster:
$$\sum_i \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$
 - This is equivalent to maximising the between-cluster scatter (total squared distance between each cluster centroid and the global centroid of all points)
- **Does it work?**
 1. The algorithm terminates.
 2. It finds a local optimum from which no further improvement is possible by making local changes.
 3. It does not necessarily find a global optimum.

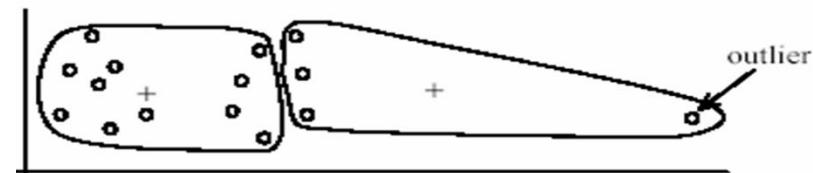
K-means Pros and Cons

Pros

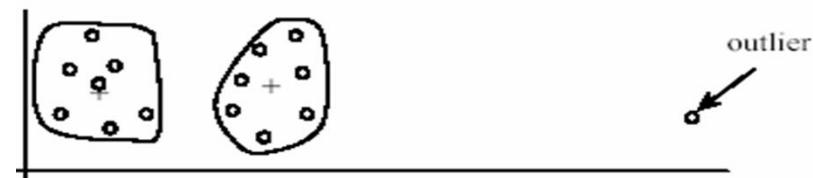
- Simple, fast to compute
- Converges to local minimum of within-cluster squared error

Cons/issues

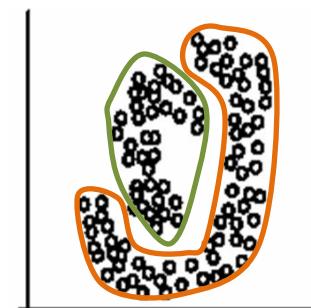
- Setting K ?
- Sensitive to initial centres
- Sensitive to outliers
- Detects spherical clusters



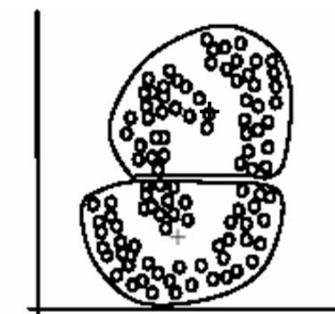
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters



(B): k -means clusters

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on
intensity similarity

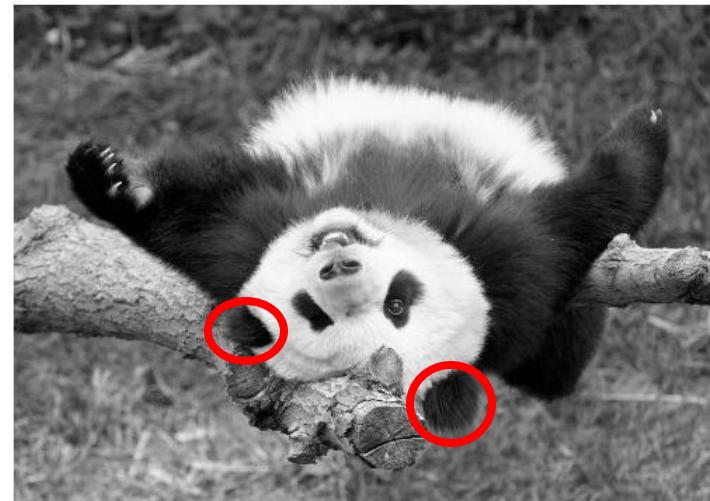
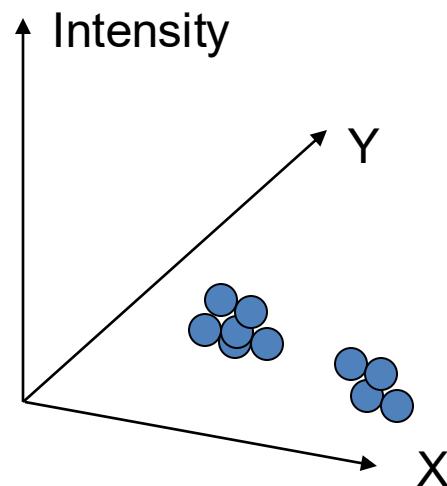


Feature space: intensity value (1D)

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **intensity**
AND position similarity

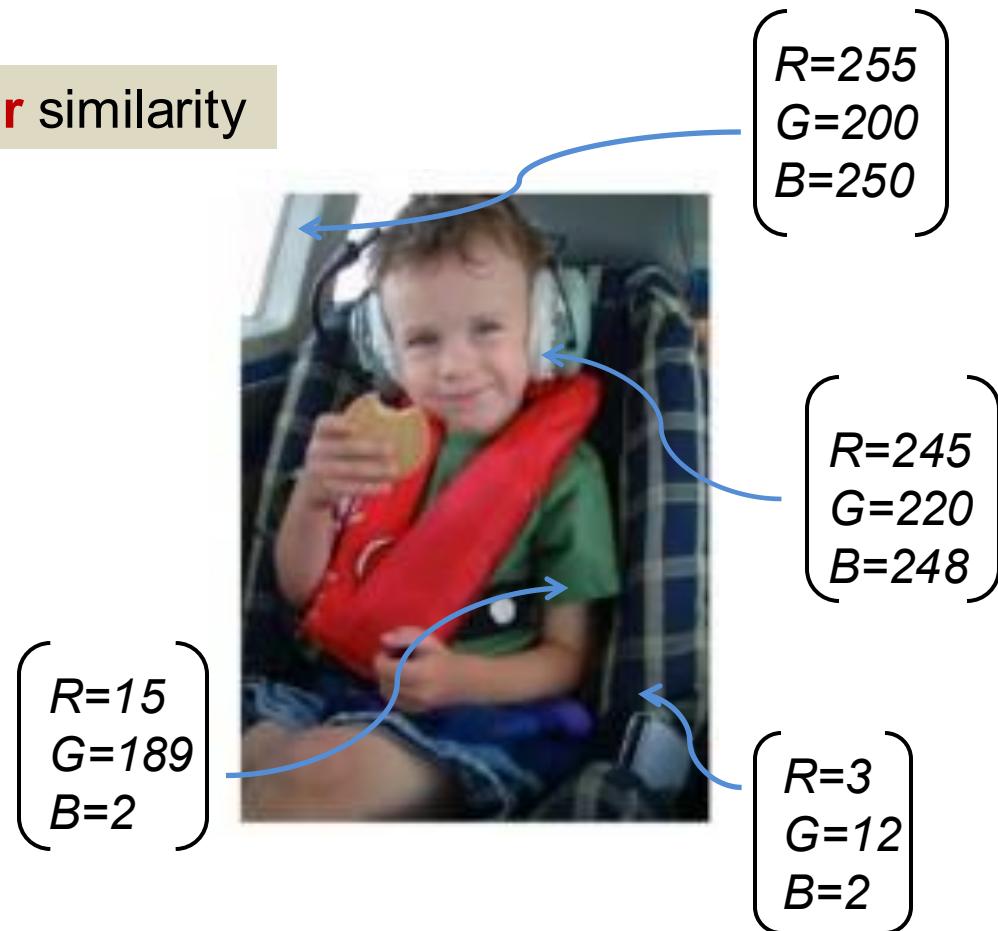
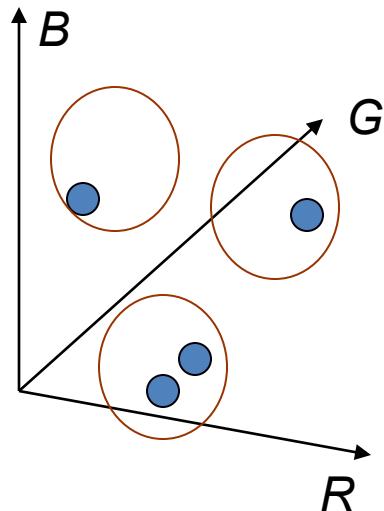


Both regions are black, but if we also include **position (x,y)**, then we could group the two into distinct segments; so encode both similarity & proximity.

Segmentation as clustering

Depending on what we choose as the *feature space*, we can group pixels in different ways.

Grouping pixels based on **colour** similarity



Feature space: intensity values (3D)

Slide inspired from Kristen Grauman

K-means Colour Segmentation

Original image



$K = 2$



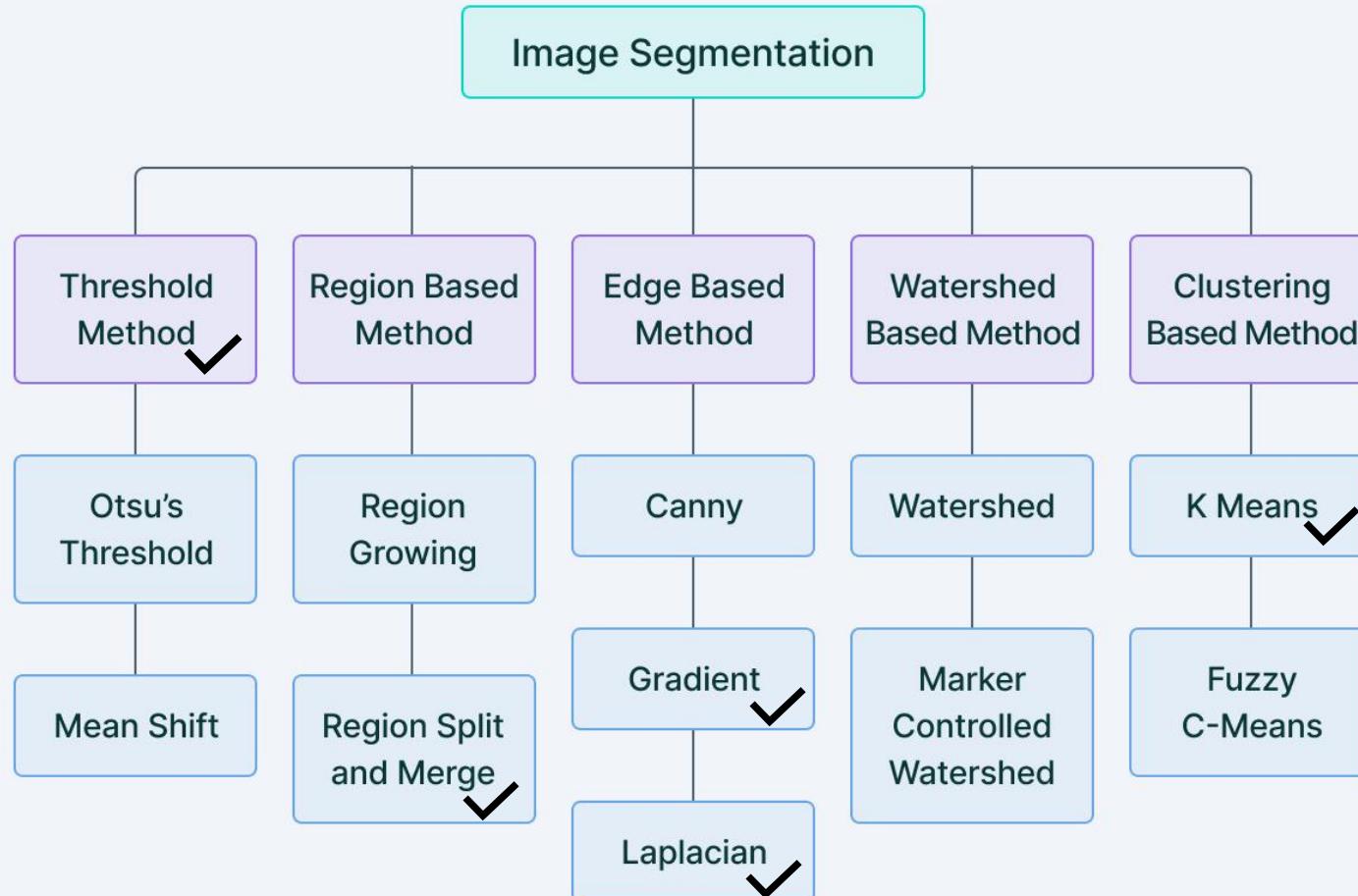
$K = 3$



$K = 10$



Summary: Image Segmentation



V7 Labs

Latest Segmentation Research

Semantic Segmentation vs. Instance Segmentation vs. Panoptic Segmentation



(a) Image



(b) Semantic Segmentation



(c) Instance Segmentation



(d) Panoptic Segmentation

V7 Labs

Next two lectures

Object Detection



Lecture 06
**Object
Detection**

Majid Mirmehdi | majid@cs.bris.ac.uk

What is ‘Object Detection’?

- Object detection aims at bridging the ‘semantic gap’ between...
 - given pixel values, *and*
 - meaningful objects (grouping of pixels + classification of groups)

Image regions need to be found and assigned with **semantic labels** from a space of object classes



What is ‘Object Detection’?

Why do classical shape detection and segmentation on their own rarely work for real-world object detection?

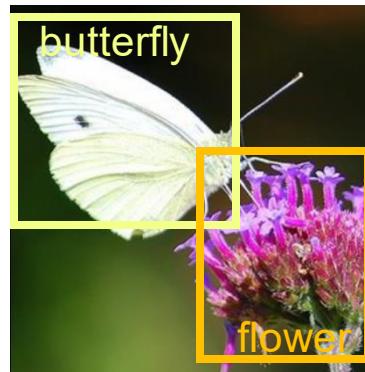
- high intra-class variance
- low inter-class variance
- classes are rarely well defined



- change of illumination, scale, pose, deformation, occlusion...

Terminology

Classification → butterfly



Multiple
object
detection



object detection =
Classification + localisation



Semantic Segmentation



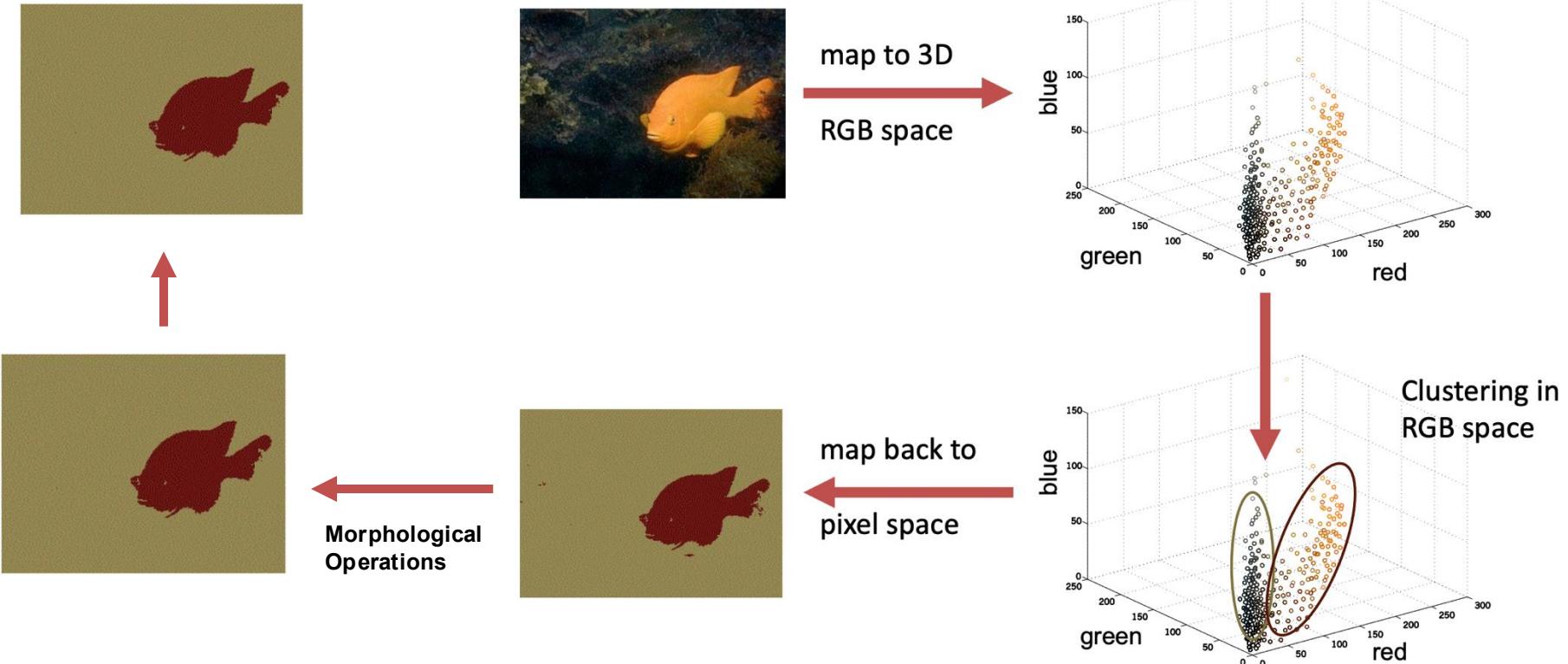
Panoptic Segmentation

Object Detection Techniques

- **Line and circle detection:** Techniques like the Hough Transform can be used to detect lines and circles in an image, which can indirectly help locate objects with specific geometric shapes.
- **Colour-based detection:** In some cases, objects can be detected based on their colour properties. This is especially useful when objects have distinct and consistent colors.
- **Template matching:** Sliding a template over the input image and finding regions where the template best matches the local image content.
- **Classifiers with sliding window detectors:** Applying image classification on overlapped patches in the image.
- **Deep learning-based object detectors:** Object detector automatically learns image features required for detection tasks, and instance segmentation.

(out of scope in this unit)

Colour-based Detection



Morphological operations

What are they used for?

- Binary images (although version for greylevel images also exists)
- Can be used for **post-processing** segmentation results, e.g. noise filtering, enhancing object structure, ...
- Quantitative description of objects (area, perimeter, etc.)

Core techniques

Erosion

Dilation

Opening

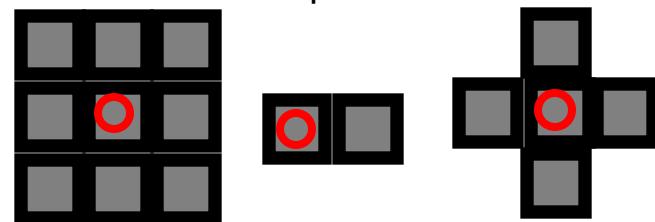
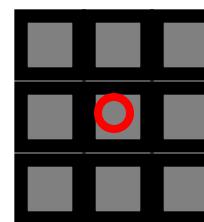
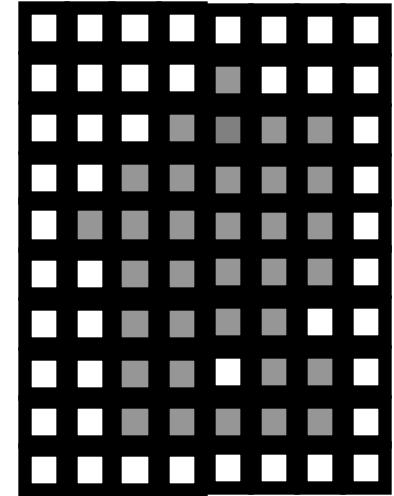
Closing

Morphological operations

Two sets:

- Image
- Morphological ***kernel*** (or *structuring element*)

- Dilation (D)
 - Union of the **kernel** with the **image** set.
 - Increases resulting area.
- Erosion (E)
 - Intersection of the **kernel** with the **image** set.
 - Decreases resulting area.



Example ***kernels***

Dilation

Morphological dilation ‘ \oplus ’ combines two sets using vector of set elements

$$X \oplus B = \{p \in Z^2 \mid p = x + b, x \in X, b \in B\}$$

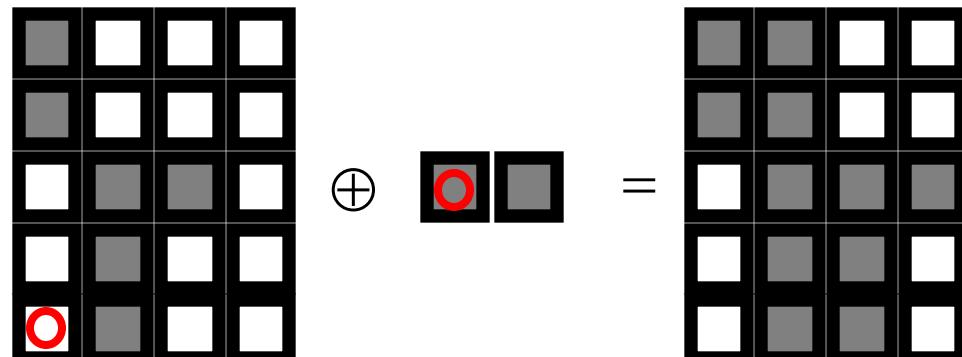
Commutative: $X \oplus B = B \oplus X$

The dilation $X + B$ is the point set of all possible vector additions of pairs of elements, one from each of the sets X and B

Associative: $X \oplus (B \oplus D) = (X \oplus B) \oplus D$

Invariant of translation: $X_h \oplus B = (X \oplus B)_h$

Is an increasing transformation: If $X \subseteq Y$ then $X \oplus B \subseteq Y \oplus B$



Erosion

Morphological erosion ' \ominus ' combines two sets using vector subtraction of set elements and is a dual operator of dilation

$$X \ominus B = \{p \in \mathbb{Z}^2 \mid \forall b \in B, p + b \in X\}$$

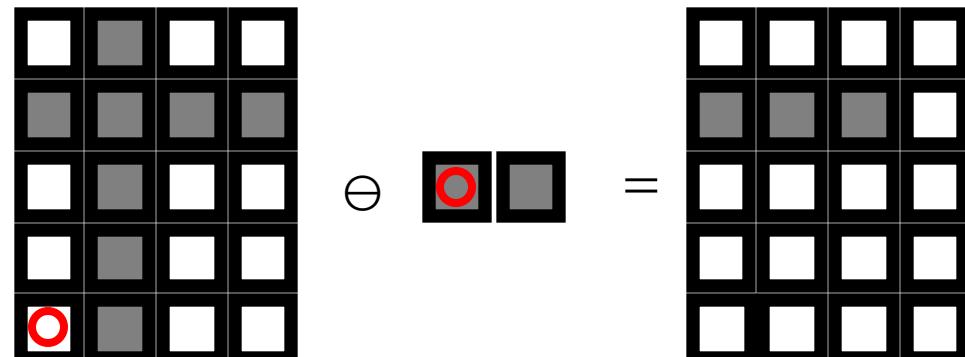
Not Commutative: $X \ominus B \neq B \ominus X$

Every point p from the image is tested; the result of the erosion is given by those points p for which all possible $p + b$ are in X .

Not associative: $X \ominus (B \ominus D) \neq (X \ominus B) \ominus D$

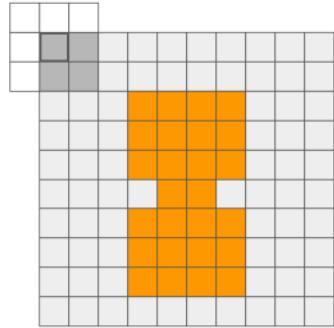
Invariant to translation: $X_h \ominus B = (X \ominus B)_h$ and $X \ominus B_h = (X \ominus B)_{-h}$

Is an increasing transformation: If $X \subseteq Y$ then $X \ominus B \subseteq Y \ominus B$

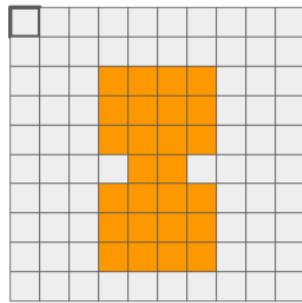


Dilation and Erosion examples

Dilation

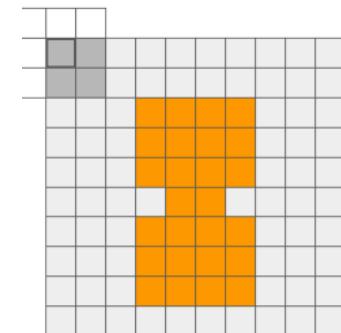


A =binary image

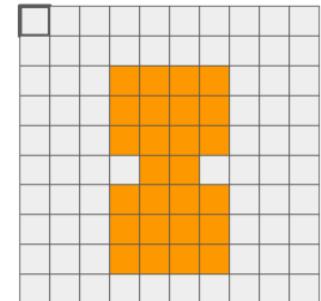


$B =$

1	1	1
1	1	1
1	1	1



A =binary image



Erosion

Examples

Original image



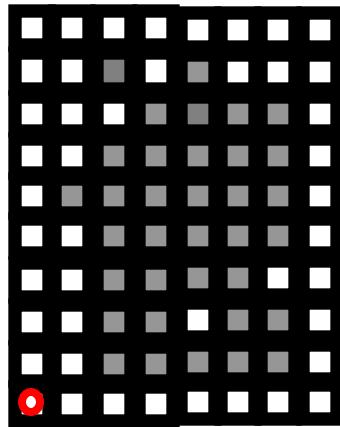
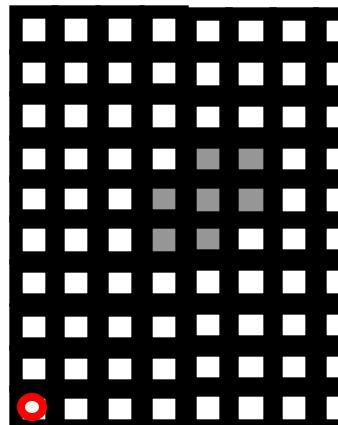
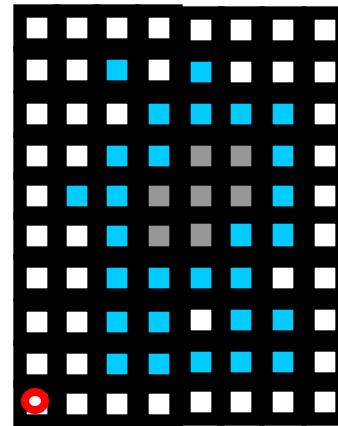
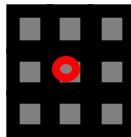
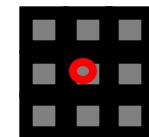
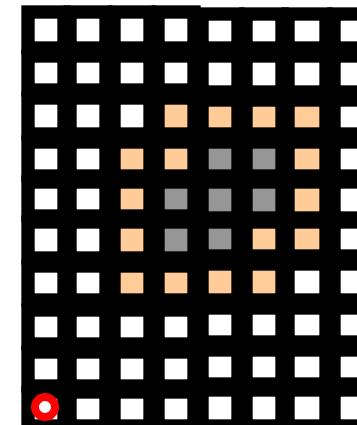
Eroded image



Dilated image

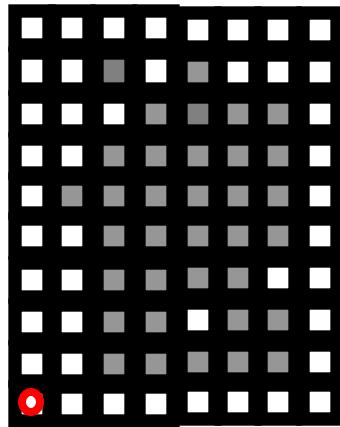
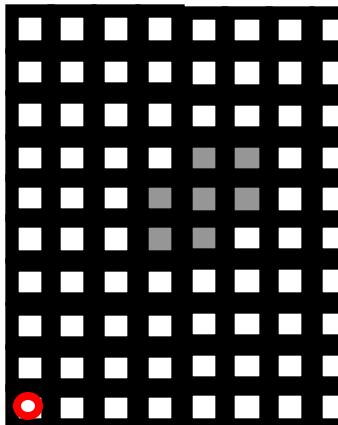
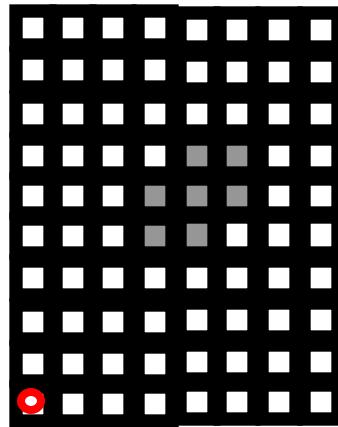
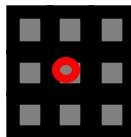
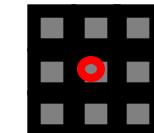
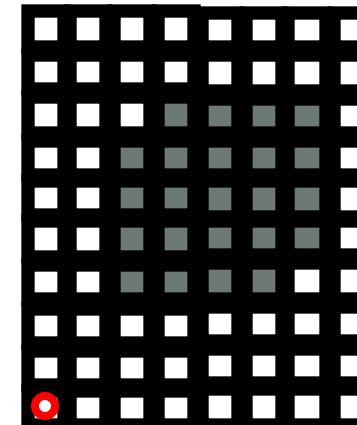


Opening: Erosion followed by Dilation

 \ominus  \oplus  $=$ 

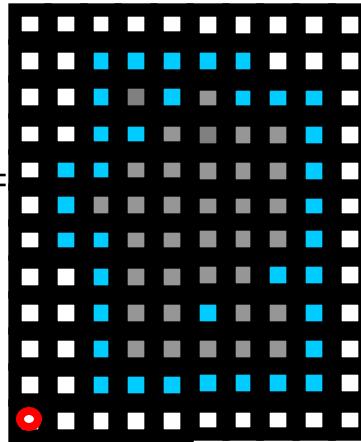
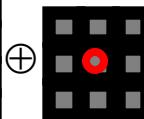
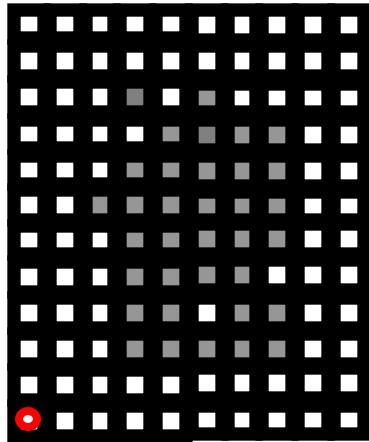
$$X \circ B = (X \ominus B) \oplus B$$

Opening: Erosion followed by Dilation

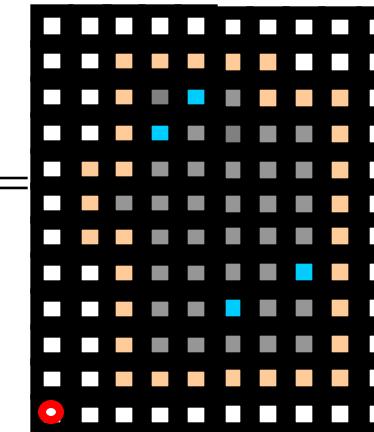
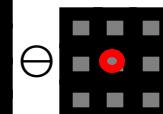
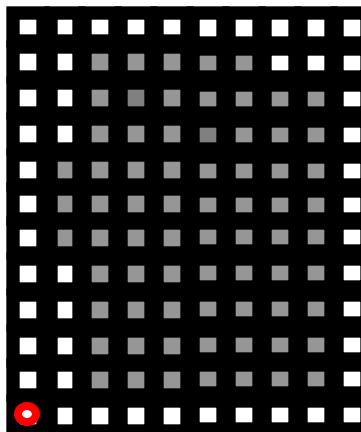
 \ominus  \oplus  $=$ 

$$X \circ B = (X \ominus B) \oplus B$$

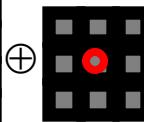
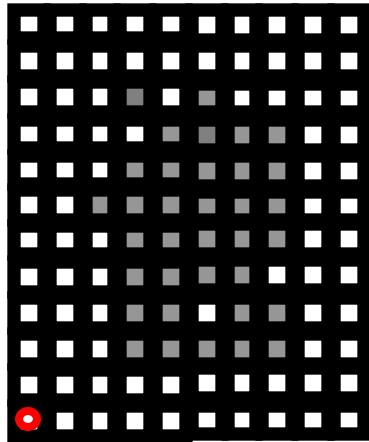
Closing: Dilation followed by Erosion



$$X \bullet B = (X \oplus B) \ominus B$$

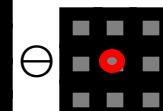
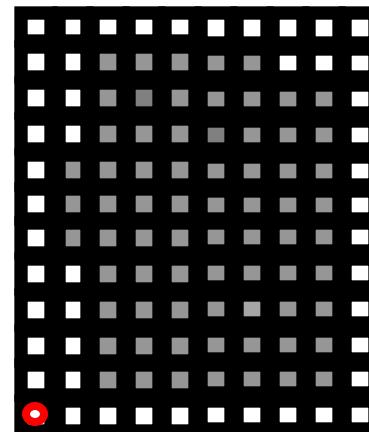


Closing: Dilation followed by Erosion



$$X \oplus B =$$
The result of applying dilation to the input image X using the kernel B. The red dot has been expanded to a 3x3 cluster of cyan pixels, while all other pixels remain black.

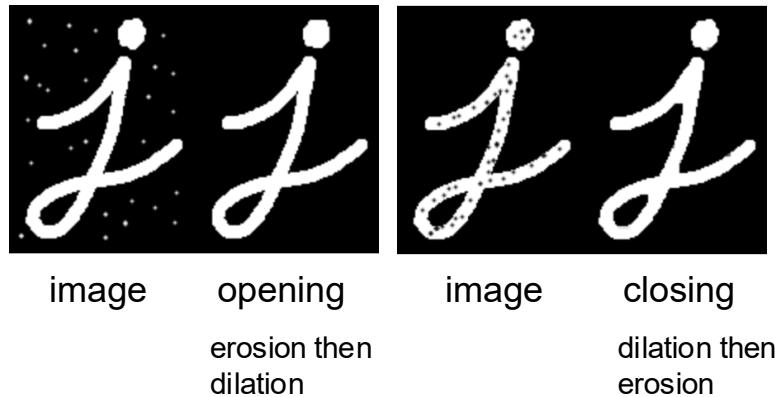
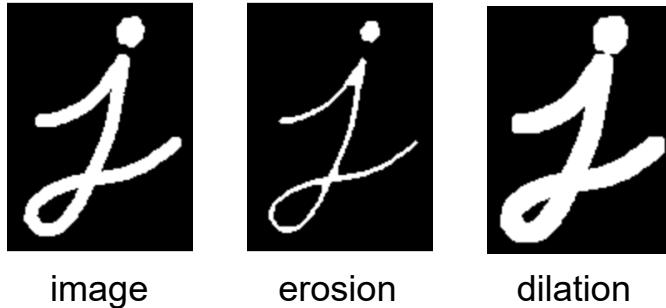
$$X \bullet B = (X \oplus B) \ominus B$$



$$(X \oplus B) \ominus B =$$
The result of applying closing to the input image X using the kernel B. The red dot has been removed, leaving a single black pixel at the bottom-left corner.



Examples



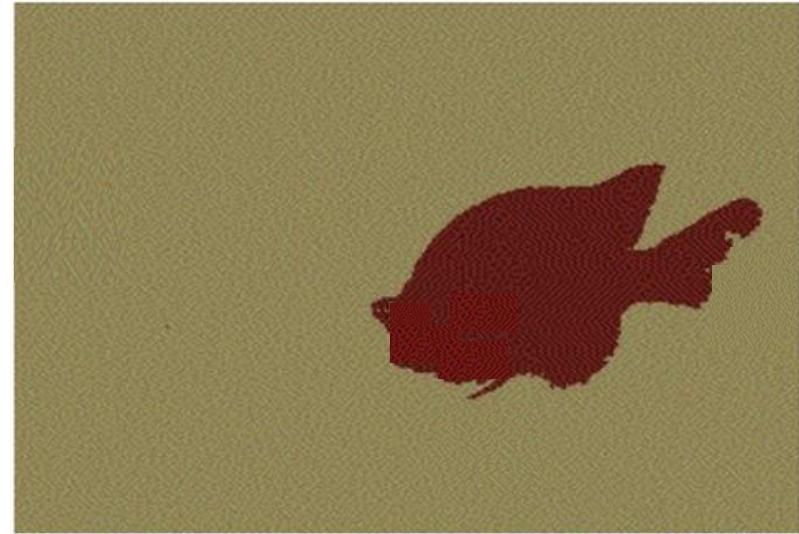
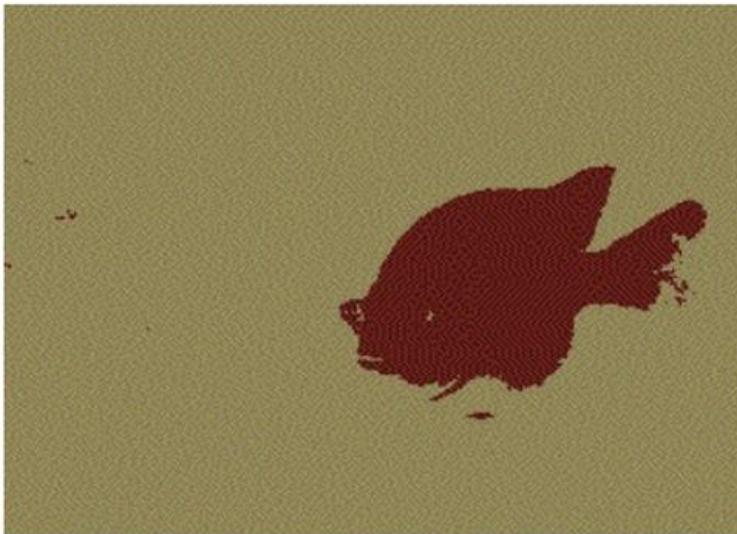
$$\begin{array}{c}
 \begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 0 & 0 & 0 & 1 & 1 & 1 \\ \hline
 0 & 1 & 1 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 \end{array} \oplus \begin{array}{|c|c|c|} \hline
 0 & 0 & 0 \\ \hline
 0 & 1 & 0 \\ \hline
 0 & 1 & 1 \\ \hline
 \end{array} = \begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 1 & 1 & 0 \\ \hline
 0 & 0 & 1 & 1 & 1 & 0 \\ \hline
 1 & 1 & 1 & 1 & 1 & 1 \\ \hline
 0 & 1 & 1 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 1 & 1 & 0 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 \end{array} \\
 A \qquad \qquad \qquad B_1 \qquad \qquad \qquad A' \\
 \end{array}$$

(a) Dilation operator $A \oplus B_1 = A'$

$$\begin{array}{c}
 \begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 0 & 0 & 0 & 1 & 1 & 0 \\ \hline
 0 & 1 & 1 & 1 & 1 & 1 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 \end{array} \ominus \begin{array}{|c|c|c|} \hline
 0 & 1 & 0 \\ \hline
 1 & 1 & 1 \\ \hline
 0 & 1 & 0 \\ \hline
 \end{array} = \begin{array}{|c|c|c|c|c|c|} \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 0 & 0 & 0 & 0 & 1 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
 \end{array} \\
 A \qquad \qquad \qquad B_2 \qquad \qquad \qquad A'' \\
 \end{array}$$

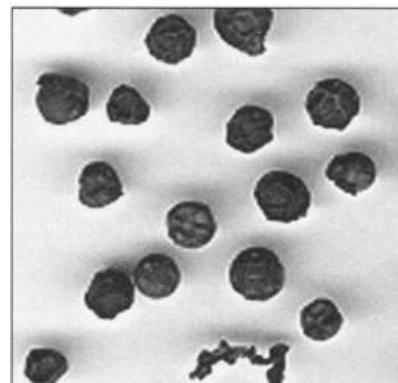
(b) Erosion operator $A \ominus B_2 = A''$

Example of Opening



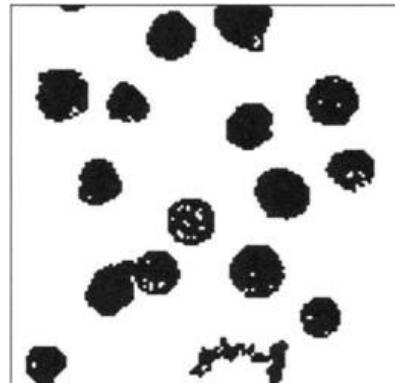
Example of Closing

(a) Image of peppercorns



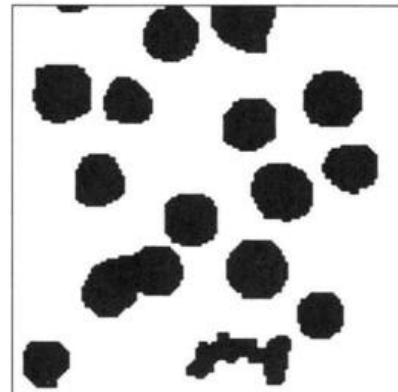
(a)

(b) Thresholded



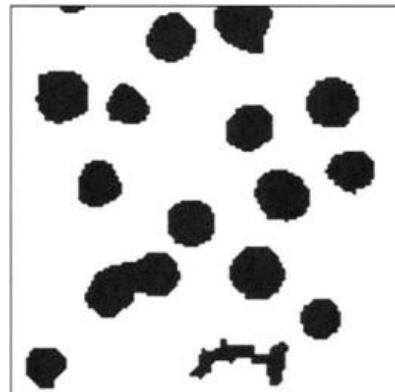
(b)

(c) 3x3 dilation...



(c)

(d) ...then 3x3 erosion



(d)

Example of Edge Detection!



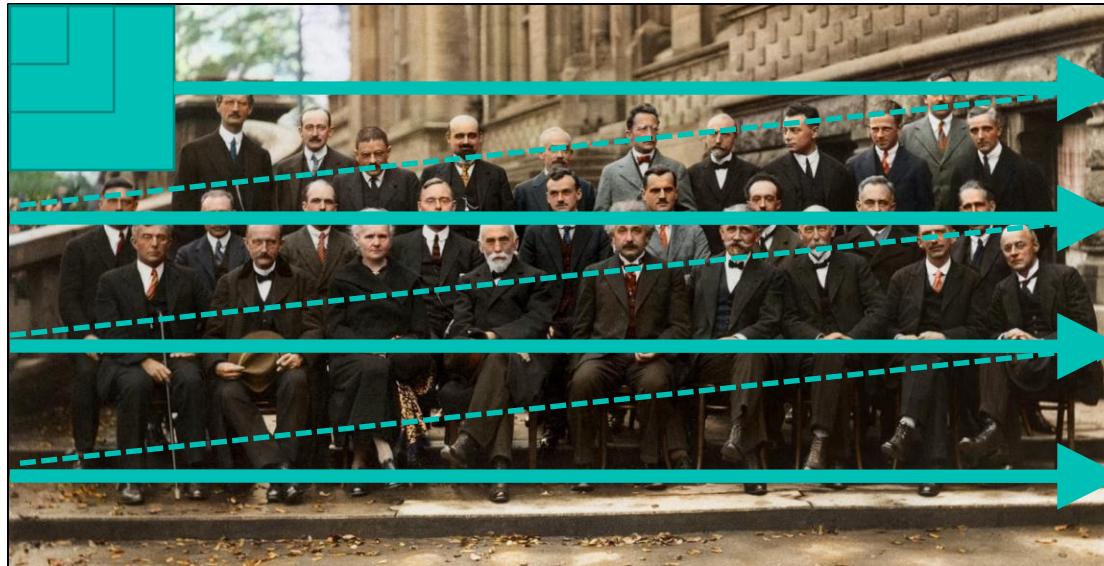
Erosion as isotropic shrink.



Contours obtained by subtraction of an eroded image from the original.

Sliding Window Detectors

- Image is tested for object presence window-by-window
- The window is ‘slided’ and ‘scaled’ throughout the image



- Each resulting window is judged w.r.t. an object model giving a response indicating object presence or absence

Template Matching

- Find the best **similarity** (or the lowest **difference**) or within the defined threshold



- Maximum

$$\text{correlation: } \frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \mu_y}{\sigma_y} \right) \left(\frac{\hat{y}_i - \mu_{\hat{y}}}{\sigma_{\hat{y}}} \right)$$

mean
std

- Minimum

$$\text{mean absolute error: } \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

$$\text{mean square error : } \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

total number of pixels in the template box

Template Matching

- Find the best **similarity** (or the lowest **difference**) or within the defined threshold



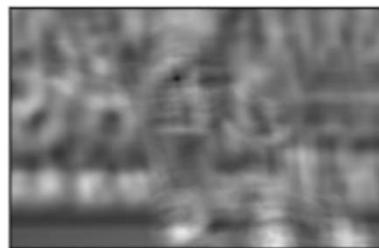
- correlation: $\frac{1}{n} \sum_{i=1}^n \left(\frac{y_i - \mu_y}{\sigma_y} \right) \left(\frac{\hat{y}_i - \mu_{\hat{y}}}{\sigma_{\hat{y}}} \right)$



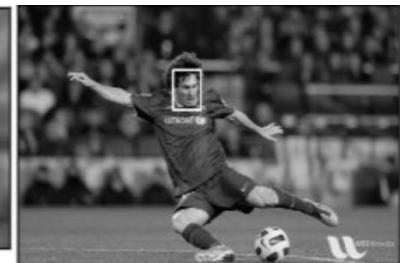
Similarity map



- mean absolute error: $\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$
- mean square error : $\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$



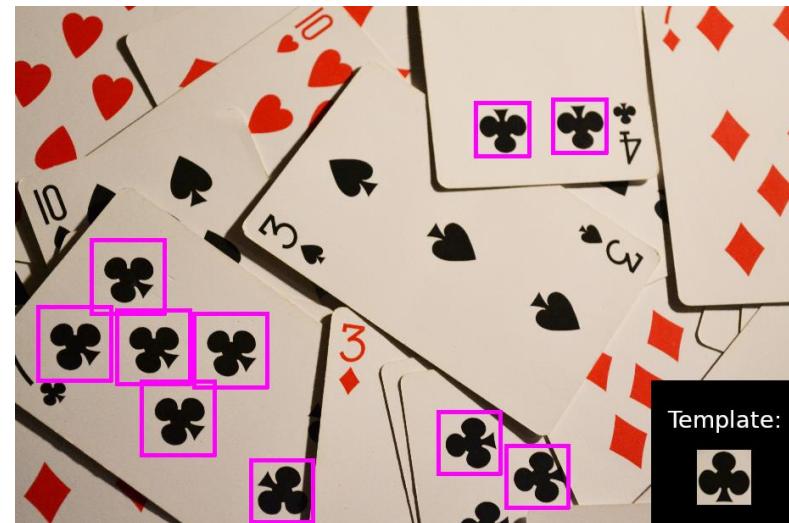
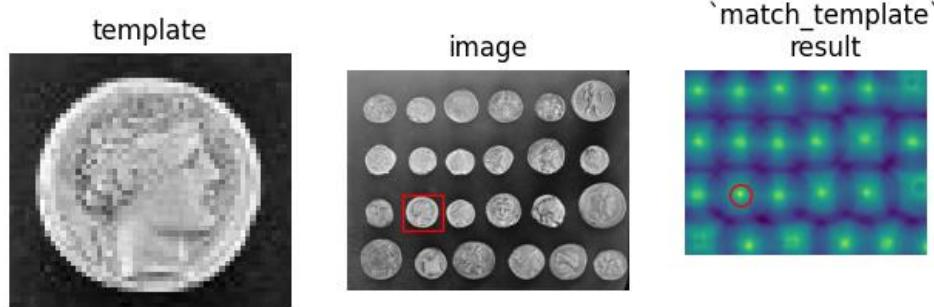
error map



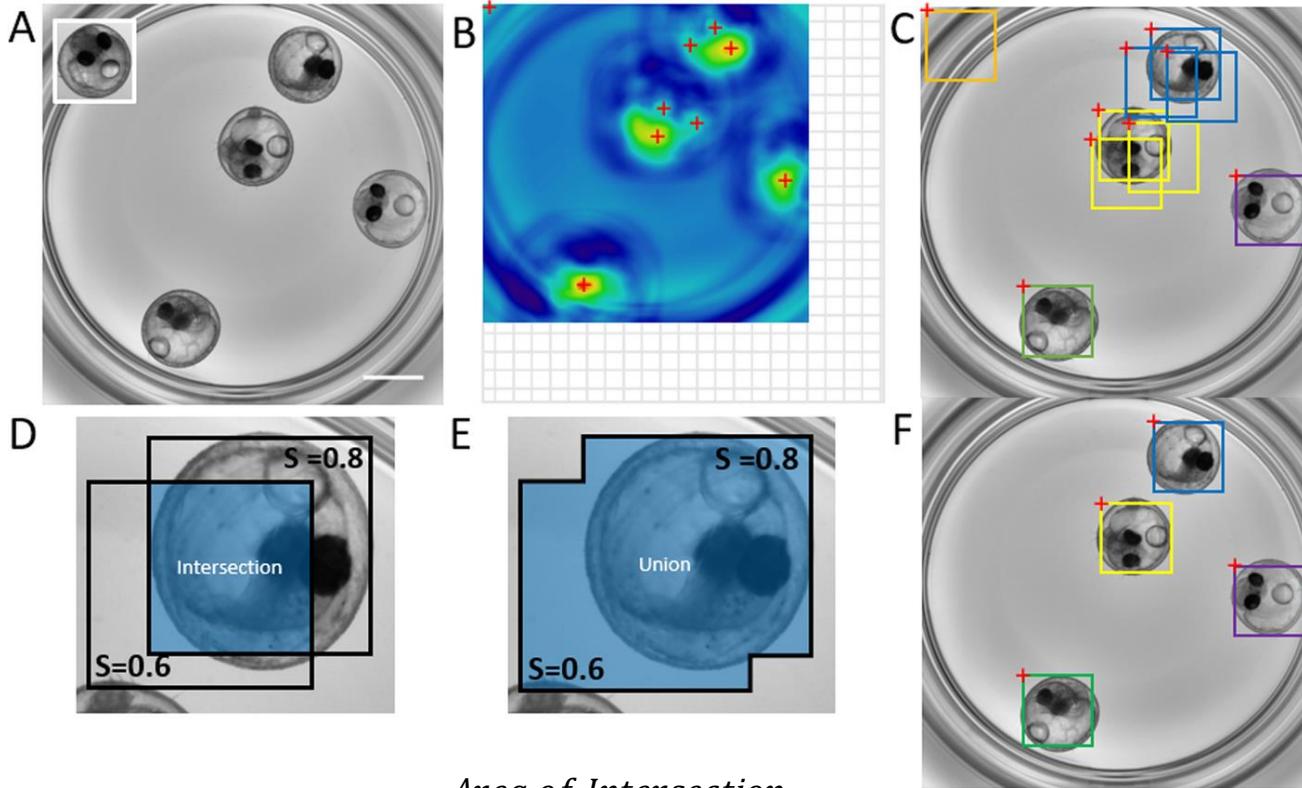
Template Matching can be expensive...

- Template image size: 53×48
- Source image size: 177×236
- Assumption: template image is inside the source image.
- Correlation (search) matrix size: 124×188
- Computation count: $124 \times 188 \times 53 \times 48 = 59,305,728$

Template Matching examples



Template Matching examples

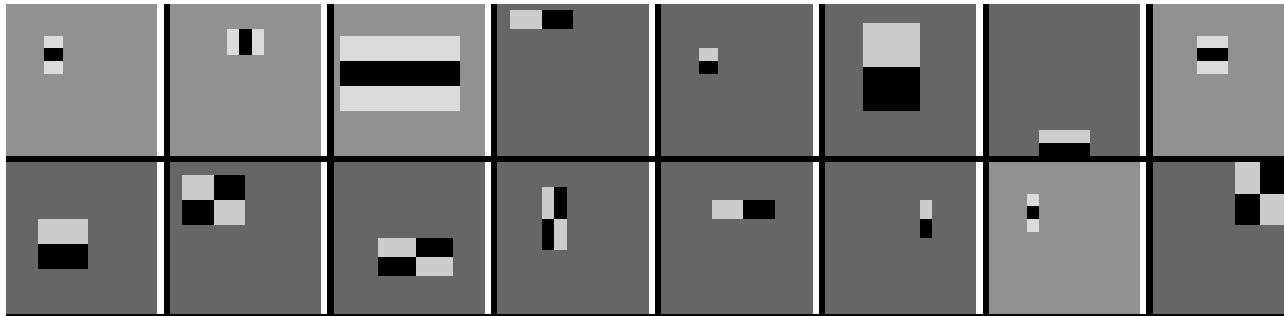


$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

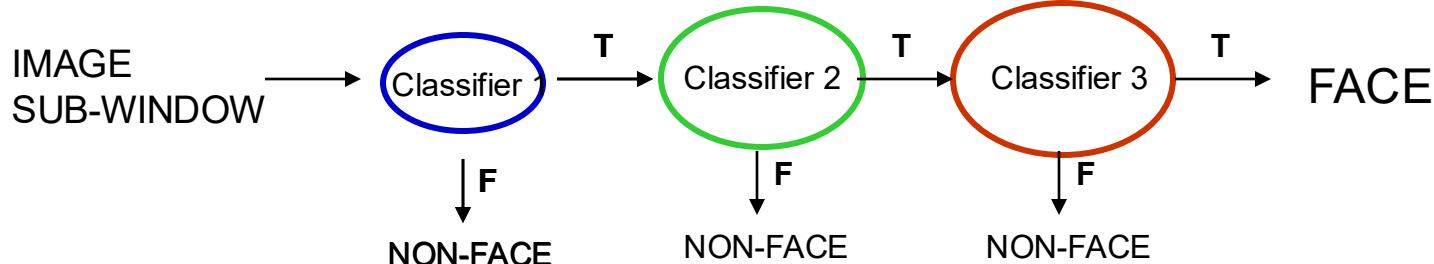
Same object (IoU closer to 1) or distinct objects that are close to each other (IoU closer to 0).

Viola-Jones: Another Sliding Window Approach

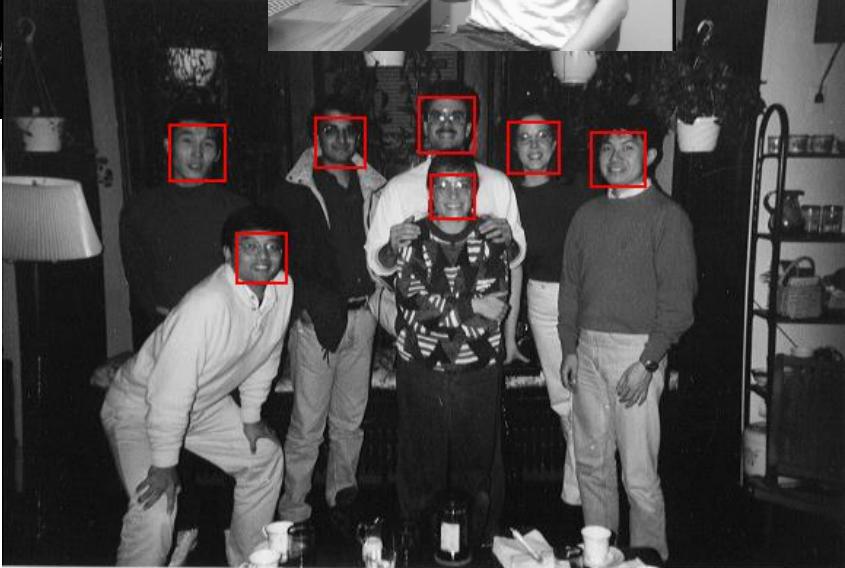
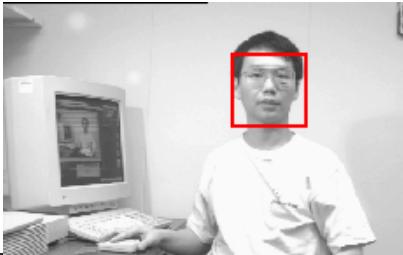
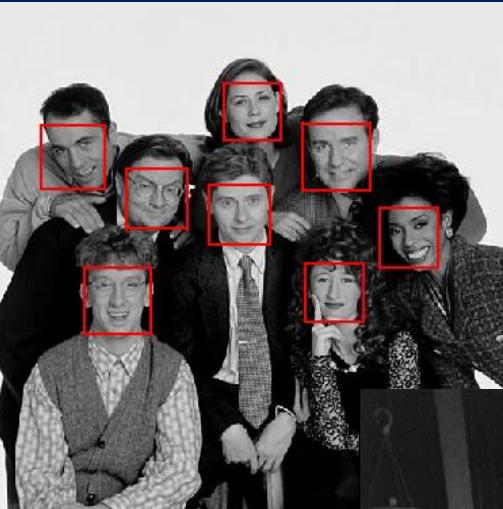
Hand-crafted weak features, but computationally efficient, calculated in sliding windows...



Construct a cascade of classifiers, which can reject most of the negative examples at early stages of processing, thereby significantly reducing computation time.



Viola-Jones: Another Sliding Window Approach



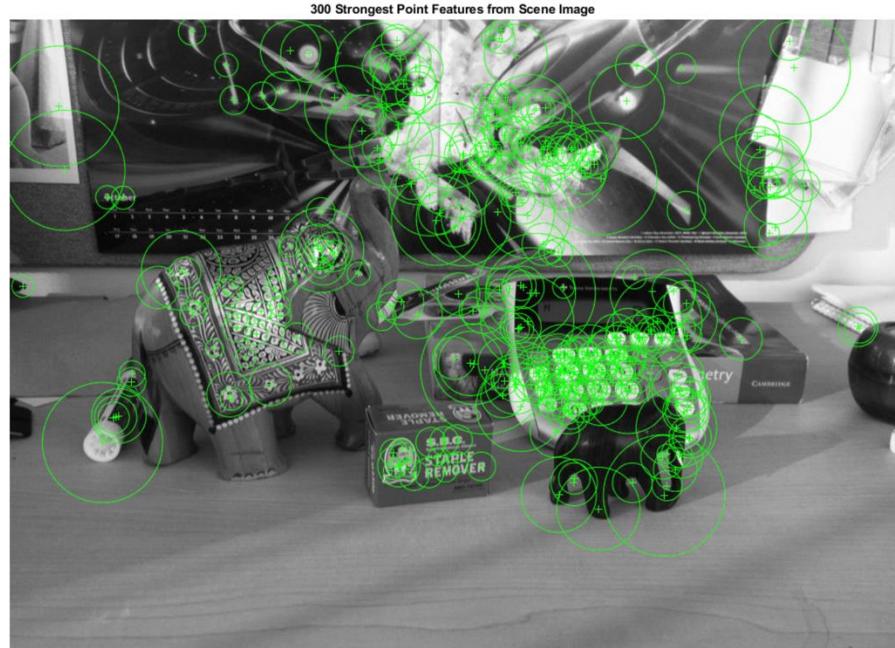
https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework

Point Feature Matching



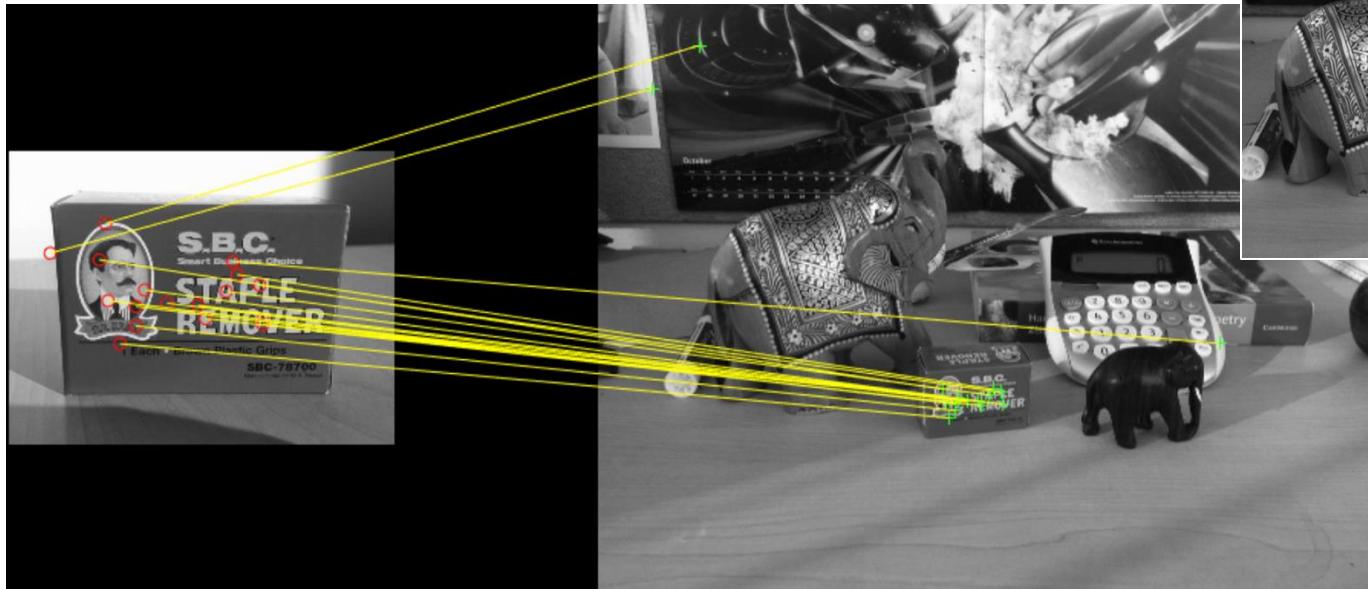
Point Feature Matching

- Harris corner detector
- Scale-Invariant Feature Transform (SIFT)
- Speeded Up Robust Features (SURF)



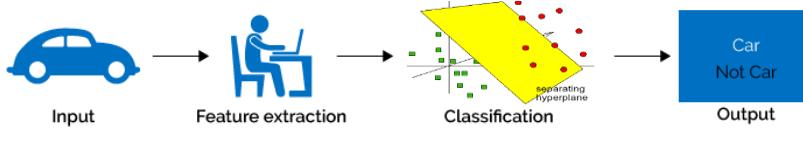
Point Feature Matching

- Rank feature similarities
- Random sample consensus (RANSAC) algorithm

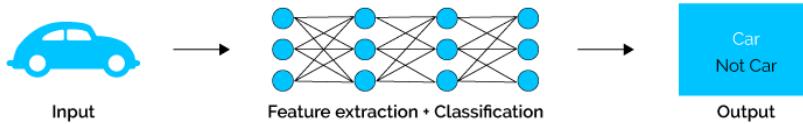


Next year: Deep learning-based object detectors

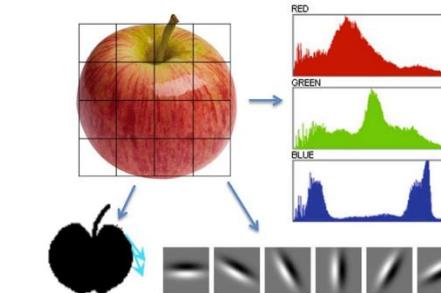
Traditional Machine Learning



Deep Learning

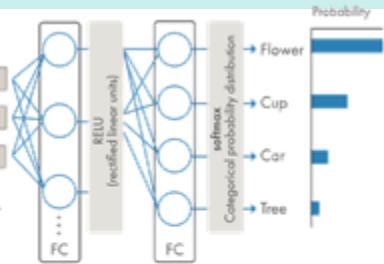
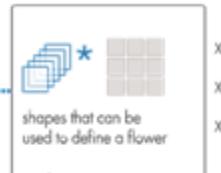
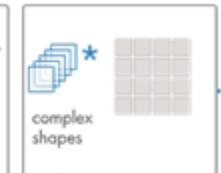
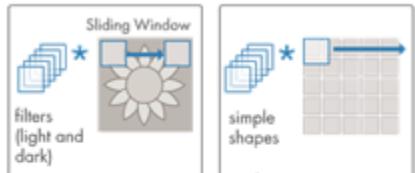
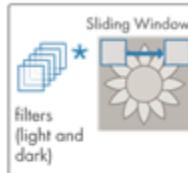
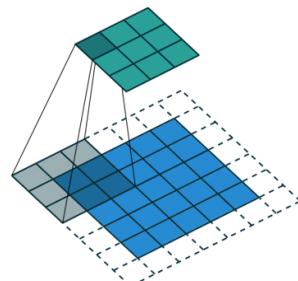


Feature extraction



$X_{apple} = \{\text{mean}_{\text{red}}, \text{variance}_{\text{red}}, \text{mean}_{\text{green}}, \text{variance}_{\text{green}}, \text{mean}_{\text{blue}}, \text{variance}_{\text{blue}}, \text{orientation}, \text{solidity}, \text{texture}, \dots\}$

Copyright © 2014 Victor Lavrenko



Every feature map output is
the result of applying a filter
to the image

The new feature map is the
next input

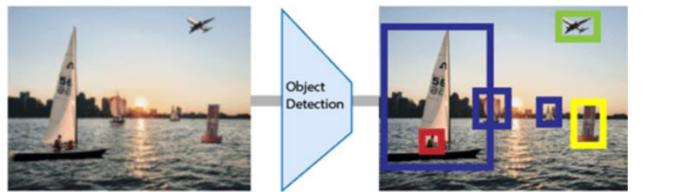
ACTIVATIONS OF THE NETWORK
AT A PARTICULAR LEVEL

Convolutional Neural Network (CNN)

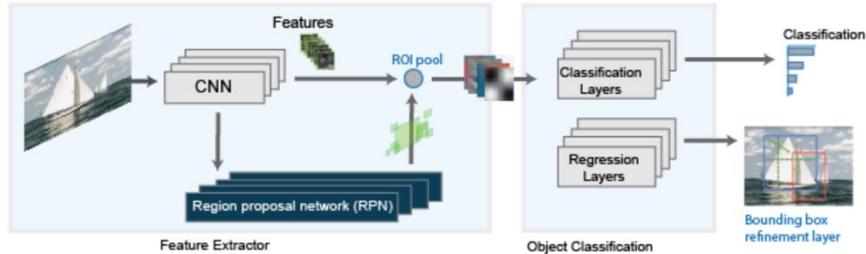
<https://uk.mathworks.com/discovery/deep-learning.html>

Next year: Deep learning-based object detectors

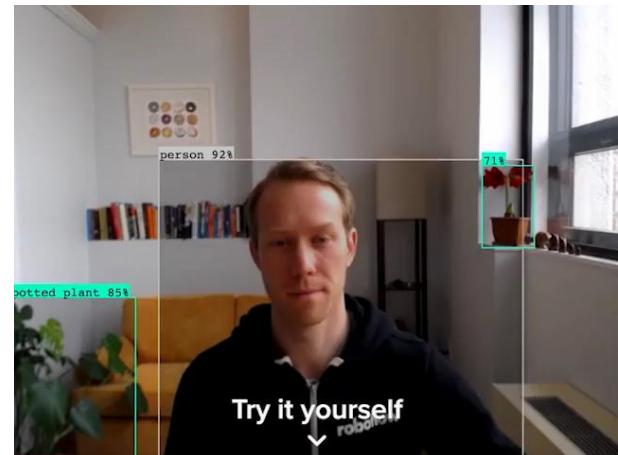
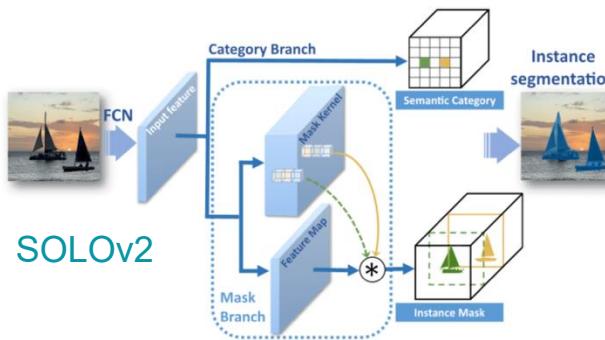
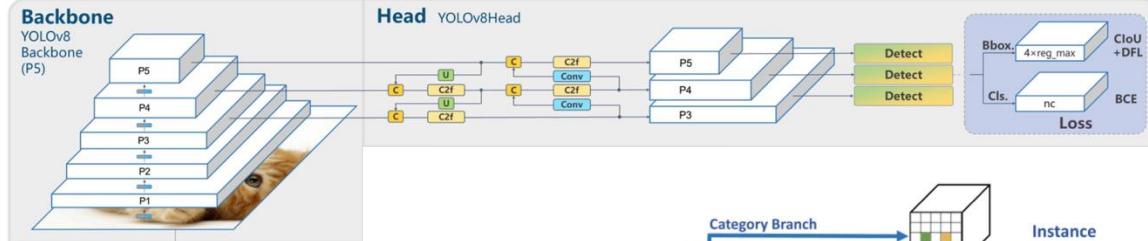
Objective



Faster-RCNN



YOLOv8 [YOLOv5](#), [YOLOv6](#), [YOLOX](#)



<https://yolov8.com/>

Next Lecture

Object Detection:
Viola-Jones Detector

Object Detection

Viola-Jones Detector

Lecture presented by
Amirhossein Dadashzadeh



Object Detection Techniques

- ✓ • **Line and circle detection:** Techniques like the Hough Transform can be used to detect lines and circles in an image, which can indirectly help locate objects with specific geometric shapes.
- ✓ • **Colour-based detection:** In some cases, objects can be detected based on their colour properties. This is especially useful when objects have distinct and consistent colors.
- ✓ • **Template matching:** Using sliding a template over the input image and finding regions where the template best matches the local image content.
- ✓ • **Classifiers with sliding window detectors:** Applying image classification on overlapped patches in the image.
- ✗ • **Deep learning-based object detectors:** Object detector automatically learns image features required for detection tasks, and instance segmentation.

(out of scope in this unit)

Classifiers with sliding window detectors

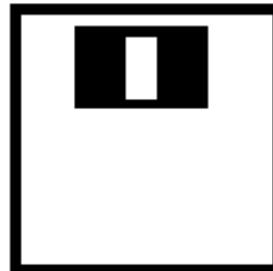
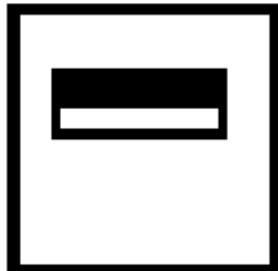
- Example Algorithm: **Viola & Jones' Real-time Method**
 - Sliding Window Detectors
 - Haar-like Features
 - Feature Extraction and Integral Images
 - Weak Classifiers
 - Boosting and Classifier Evaluation
 - Cascades of Boosted Classifiers



*Best description of full details available in
consolidated paper by
Viola and Jones, International Journal of Computer
Vision, 2004*

Haar-like Features

Viola & Jones' (2001)



feature = sum of white pixels – sum of black pixels

filter 1

-1	-1	1	1
-1	-1	1	1
-1	-1	1	1
-1	-1	1	1

hard edge
feature1 = 2040

0	0	255	255
0	0	255	255
0	0	255	255
0	0	255	255
0	0	255	255

soft edge
feature1 = 1245

20	120	220	230
25	125	205	225
25	105	220	234
24	110	215	250

no edge
feature1 = 17

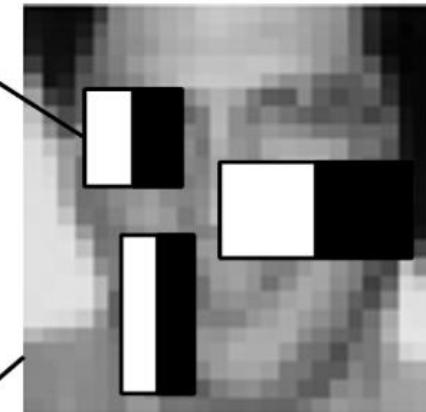
218	230	220	230
200	230	205	225
220	234	220	244
210	250	215	250

Haar-like Features

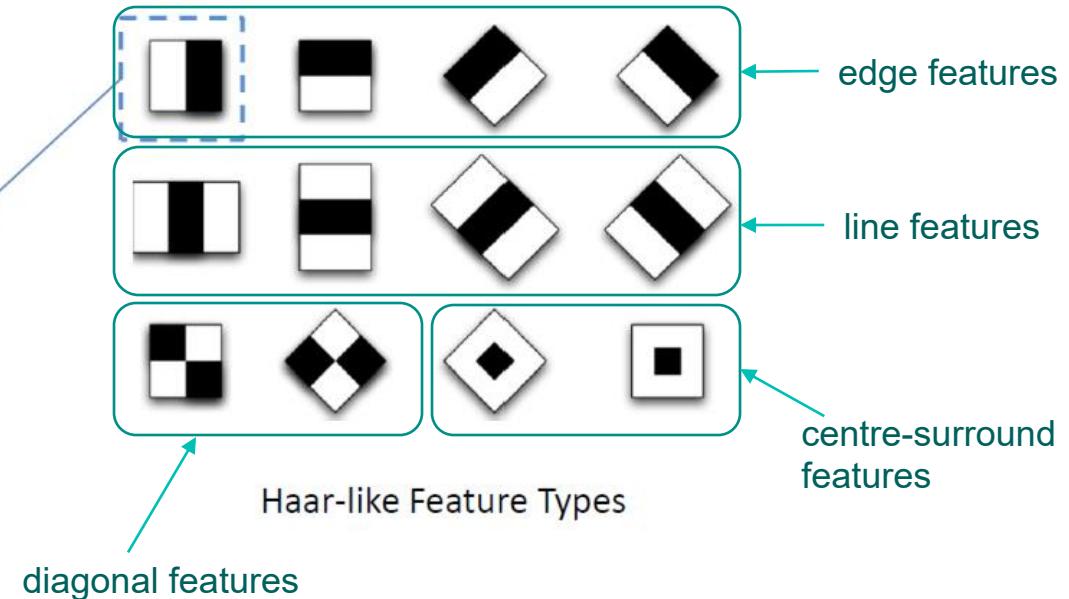
instance
response

$$h_j(x_i)$$

sample
image
 x_i



Examples of Instances of
1 Feature Type



Integral Images

I Image

	0	1	2	3	4	5	6	7
0	1	1	1	2	3	1	2	1
1	1	2	0	0	0	3	1	1
2	1	1	1	1	1	2	3	1
3	1	1	1	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1

II Integral Image

	0	1	2	3	4	5	6	7
0	1	2	3	5	8	9	11	12
1	2	5	6	8	11	15	18	20
2	3	7	9	12	16	22	28	31
3	4	9	12	15	19	25	31	34
4	4	9	12	15	19	25	31	34
5	4	9	12	15	19	25	32	36

\sum

(IMAGE INTEGRATION)

$$\text{II}(-1, y) = 0;$$

$$\text{II}(x, y) = \text{II}(x - 1, y) + A(x, y);$$

$$A(x, -1) = 0;$$

$$A(x, y) = A(x, y - 1) + \text{I}(x, y).$$

$$\begin{aligned} x=0, y=0: A(0,0) &= A(0,-1) + \text{I}(0,0) = 0 + 1 = 1 \\ \text{II}(0,0) &= \text{II}(-1,0) + A(0,0) = 0 + 1 = 1 \end{aligned}$$

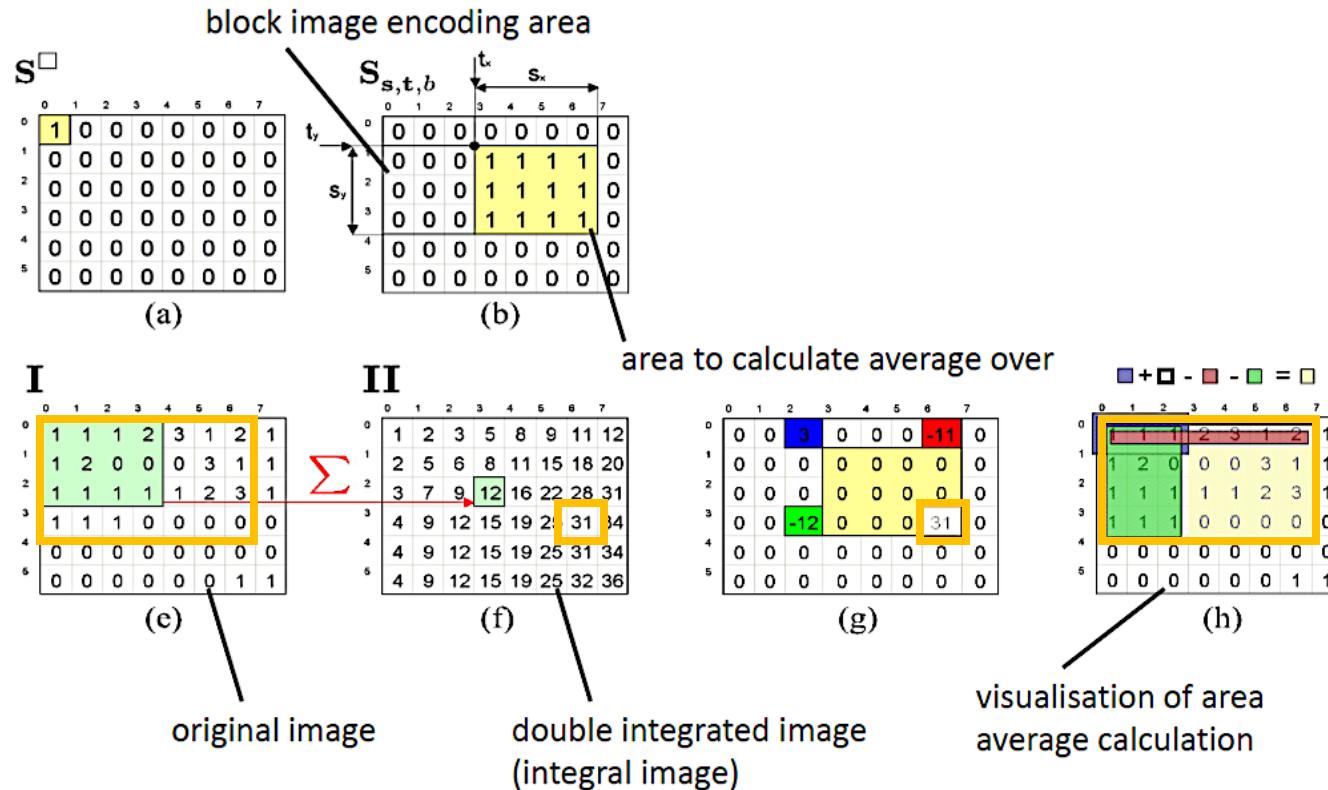
$$\begin{aligned} x=0, y=1: A(0,1) &= A(0,0) + \text{I}(0,1) = 1 + 1 = 2 \\ \text{II}(0,1) &= \text{II}(-1,1) + A(0,1) = 0 + 2 = 2 \end{aligned}$$

$$\begin{aligned} x=1, y=0: A(1,0) &= A(1,-1) + \text{I}(1,0) = 0 + 1 = 1 \\ \text{II}(1,0) &= \text{II}(0,0) + A(1,0) = 1 + 1 = 2 \end{aligned}$$

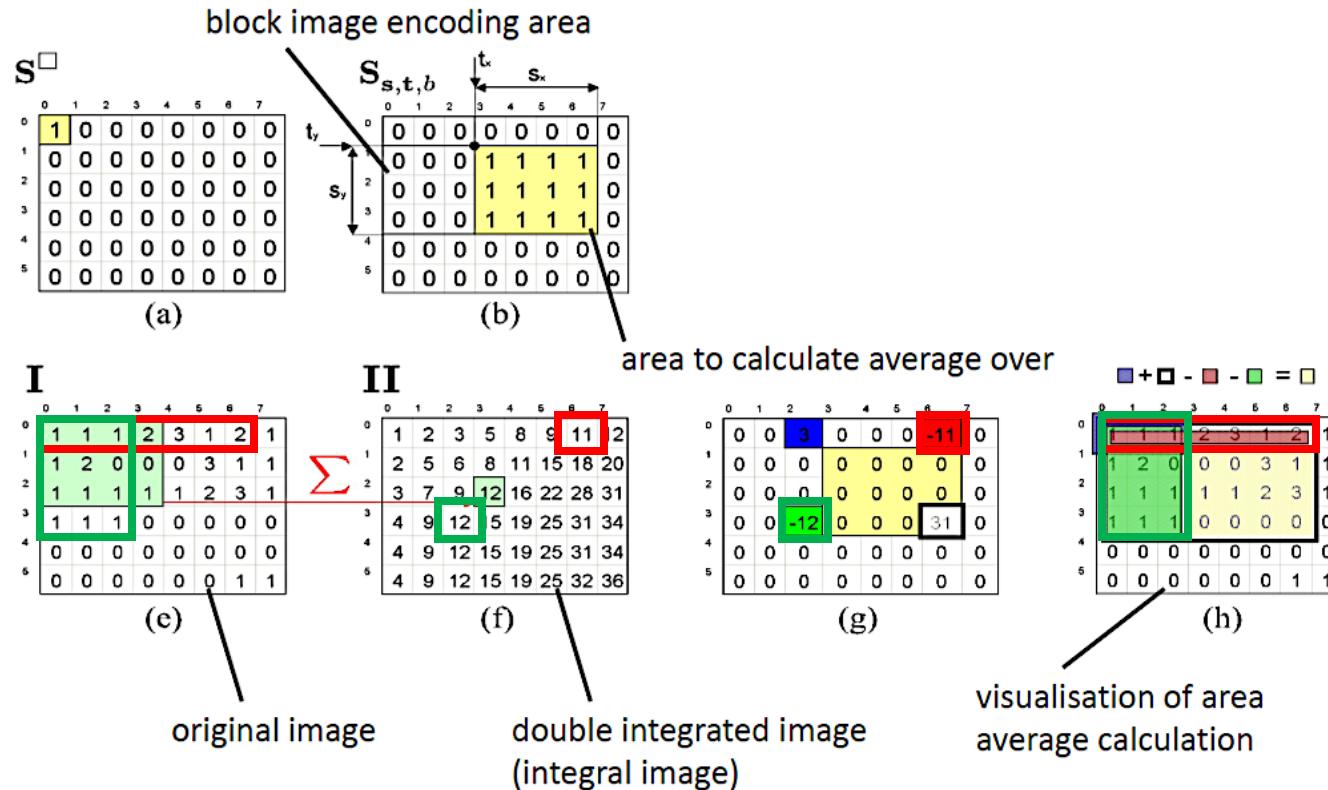
$$\begin{aligned} x=1, y=1: A(1,1) &= A(1,0) + \text{I}(1,1) = 1 + 2 = 3 \\ \text{II}(1,1) &= \text{II}(0,1) + A(1,1) = 2 + 3 = 5 \end{aligned}$$

...

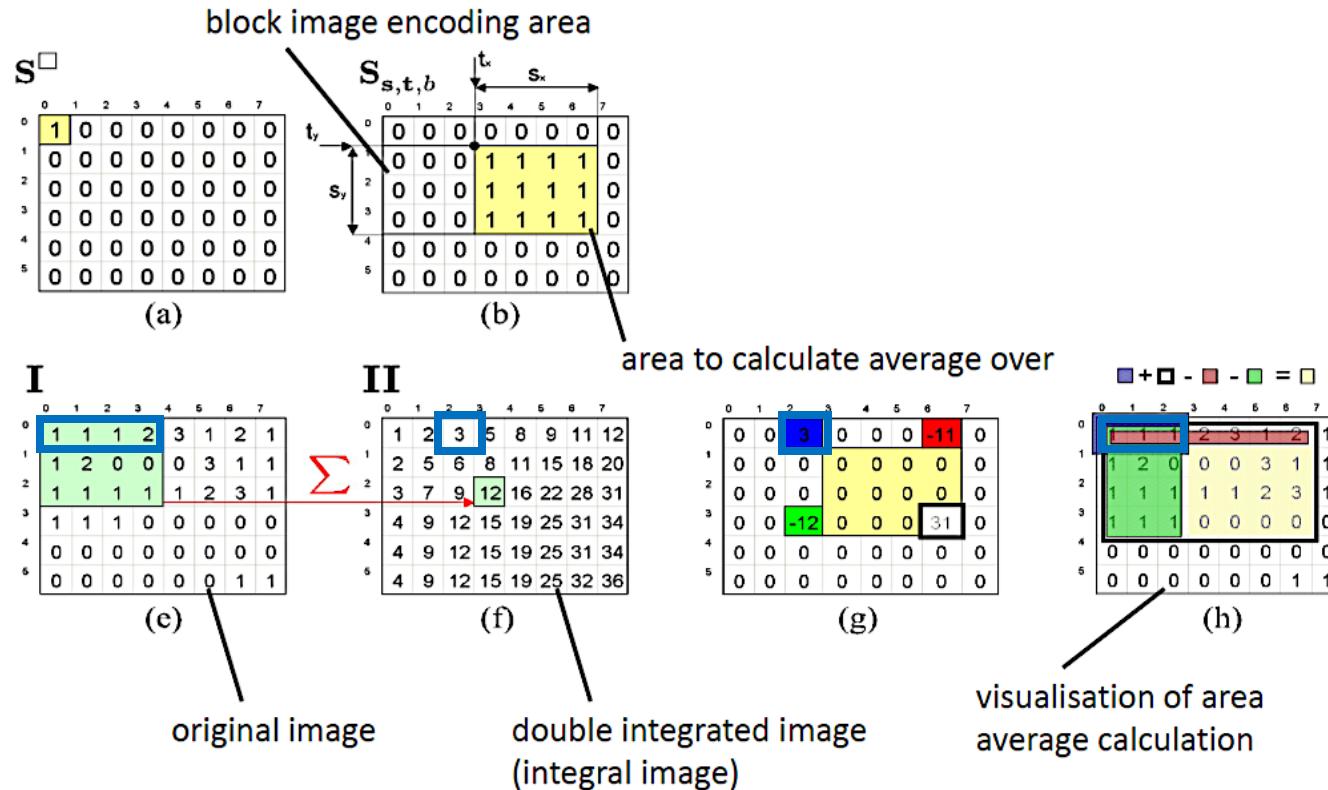
Calculating the Avg Pixel Value of Large Block Fast



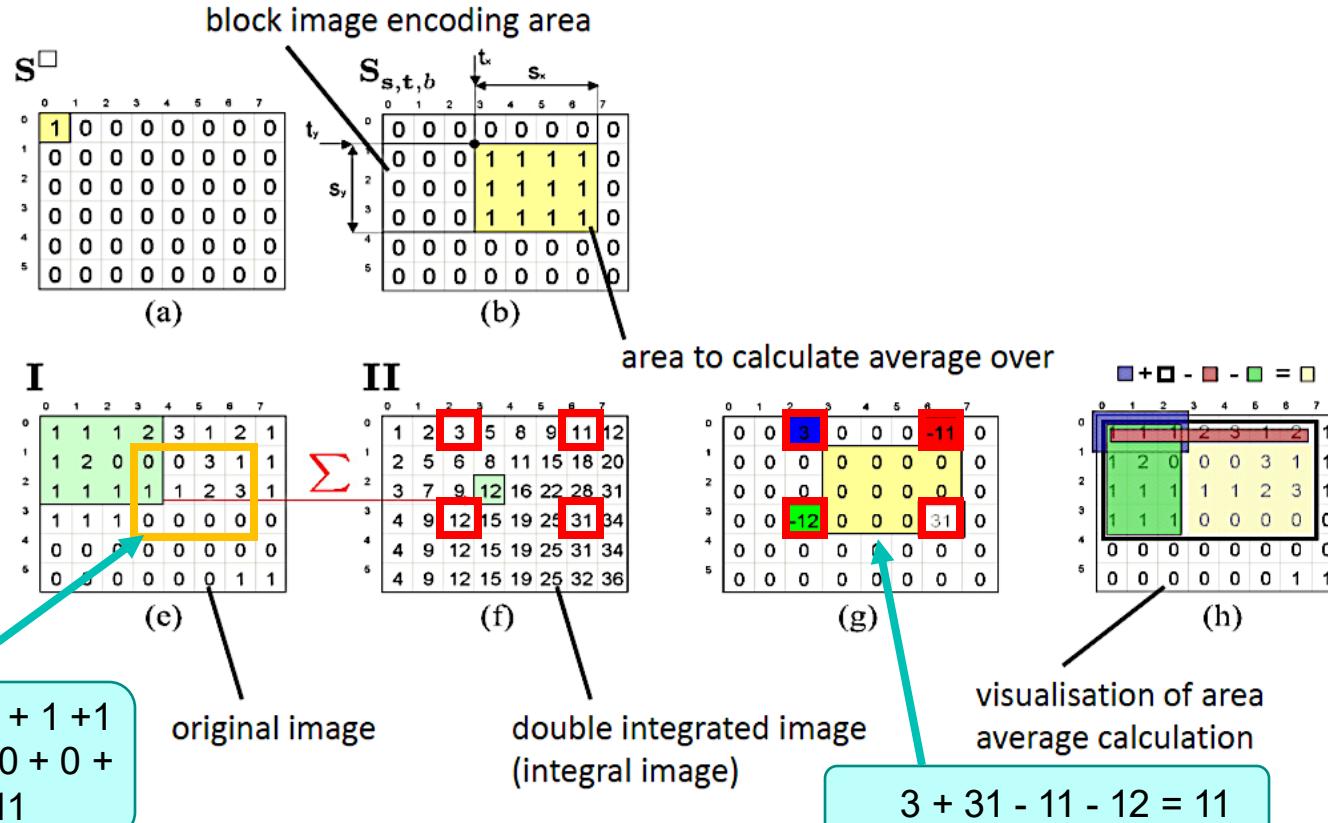
Calculating the Avg Pixel Value of Large Block Fast



Calculating the Avg Pixel Value of Large Block Fast



Calculating the Avg Pixel Value of Large Block Fast



Feature Extraction

	x									
	0	1	2	3	4	5	6	7	8	9
0	4	1	1	0	10	0	1	0	0	5
1	3	5	7	1	4	1	8	9	4	0
2	6	4	5	0	1	0	7	8	3	2
3	2	3	4	5	1	1	6	3	4	5
4	9	9	10	1	5	3	1	4	5	4
5	2	2	1	7	4	0	2	1	5	6
6	8	1	1	4	2	3	2	3	1	0
7	0	1	7	0	3	5	6	3	4	1
8	1	5	6	3	5	9	10	4	2	0
9	1	8	3	4	6	3	6	3	3	5

Image

-5-0-1-4-5-1-10-1-5-1-7-4-1-4-2-7-0-
3+0+7+8+1+6+3+3+1+4+0+2+1+3+2+
3+5+6+3 = -3

	x									
	0	1	2	3	4	5	6	7	8	9
0	4	5	6	6	16	16	17	17	17	22
1	7	13	21	22	36	37	46	55	59	64
2	13	23	36	37	52	53	69	86	93	100
3	15	28	45	51	67	69	91	111	122	134
4	24	46	73	80	101	106	129	153	169	185
5	26	50	78	92	117	122	147	172	193	215
6	34	59	88	106	133	141	168	196	218	240
7	34	60	96	114	144	157	190	221	247	270
8	35	66	108	129	164	186	229	264	292	315
9	36	75	120	145	186	211	260	298	329	357

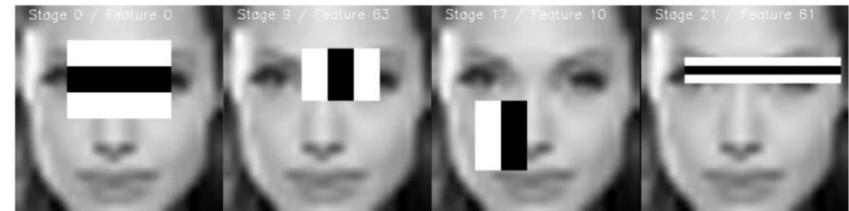
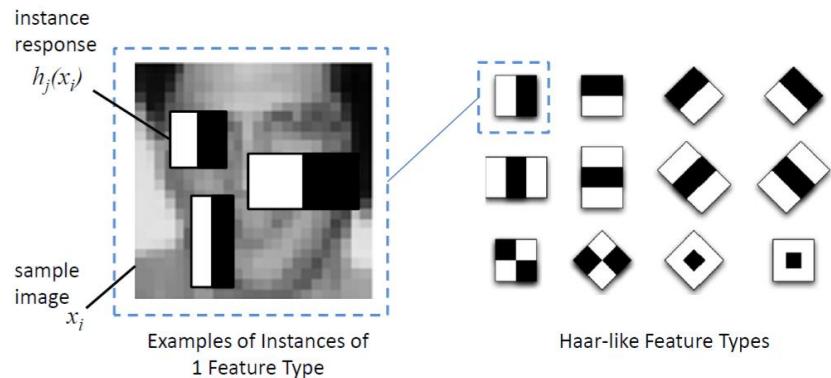
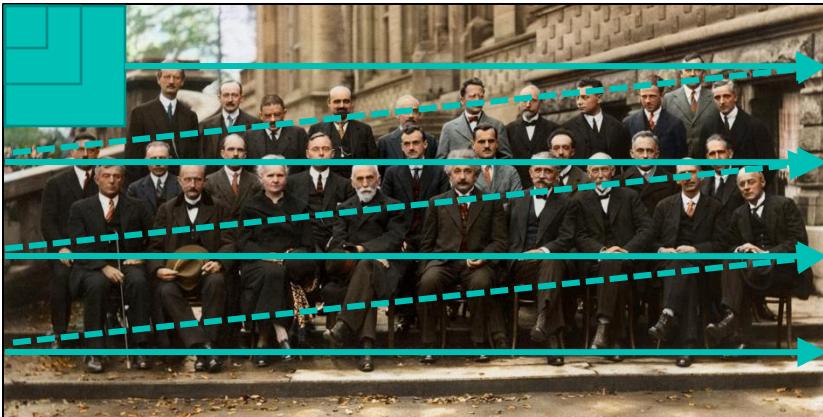
Integral Image

-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1
-1	-1	-1	1	1	1

Haar filter

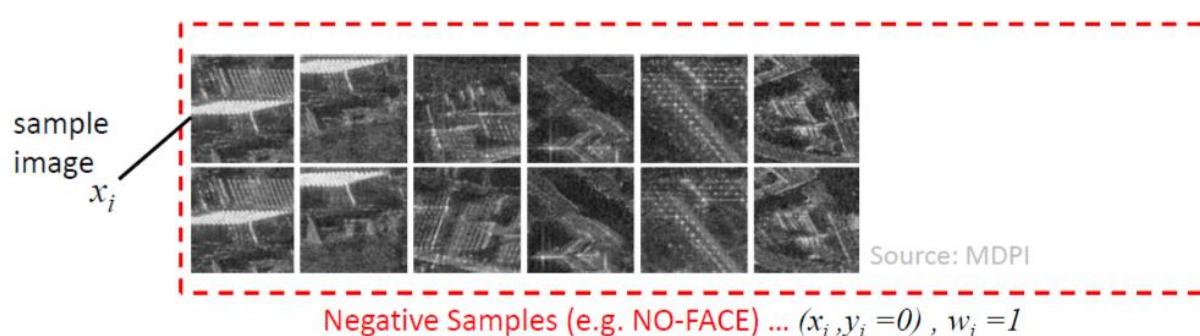
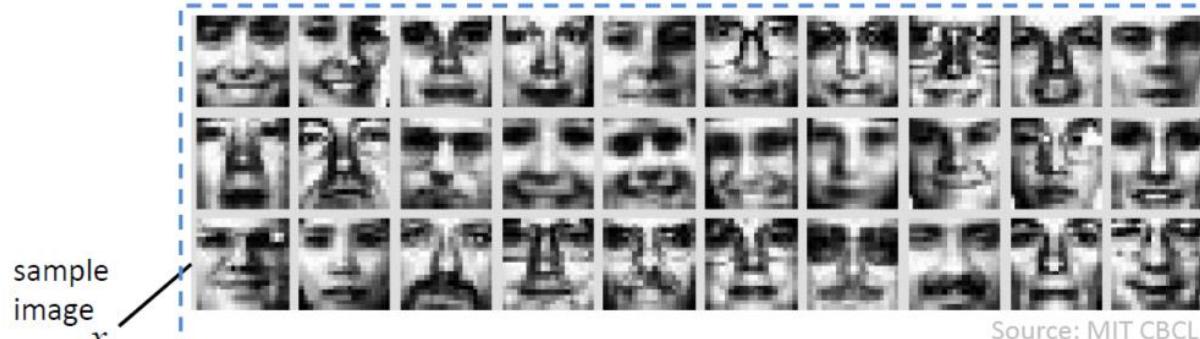
$$-(144+13-36-60) + (221+36-55-144) = -3$$

Viola & Jones' Real-time Method

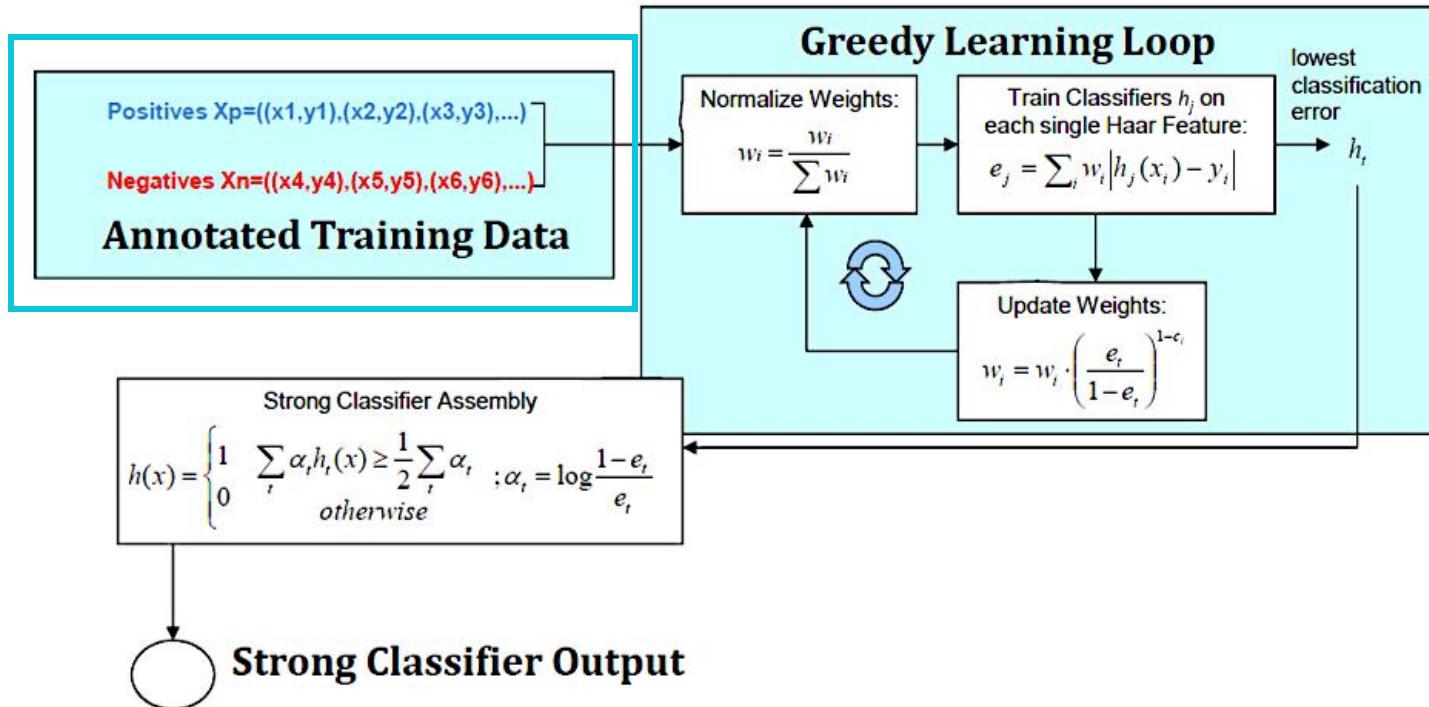


https://medium.com/@Andrew_D./computer-vision-viola-jones-object-detection-d2a609527b7c

Training

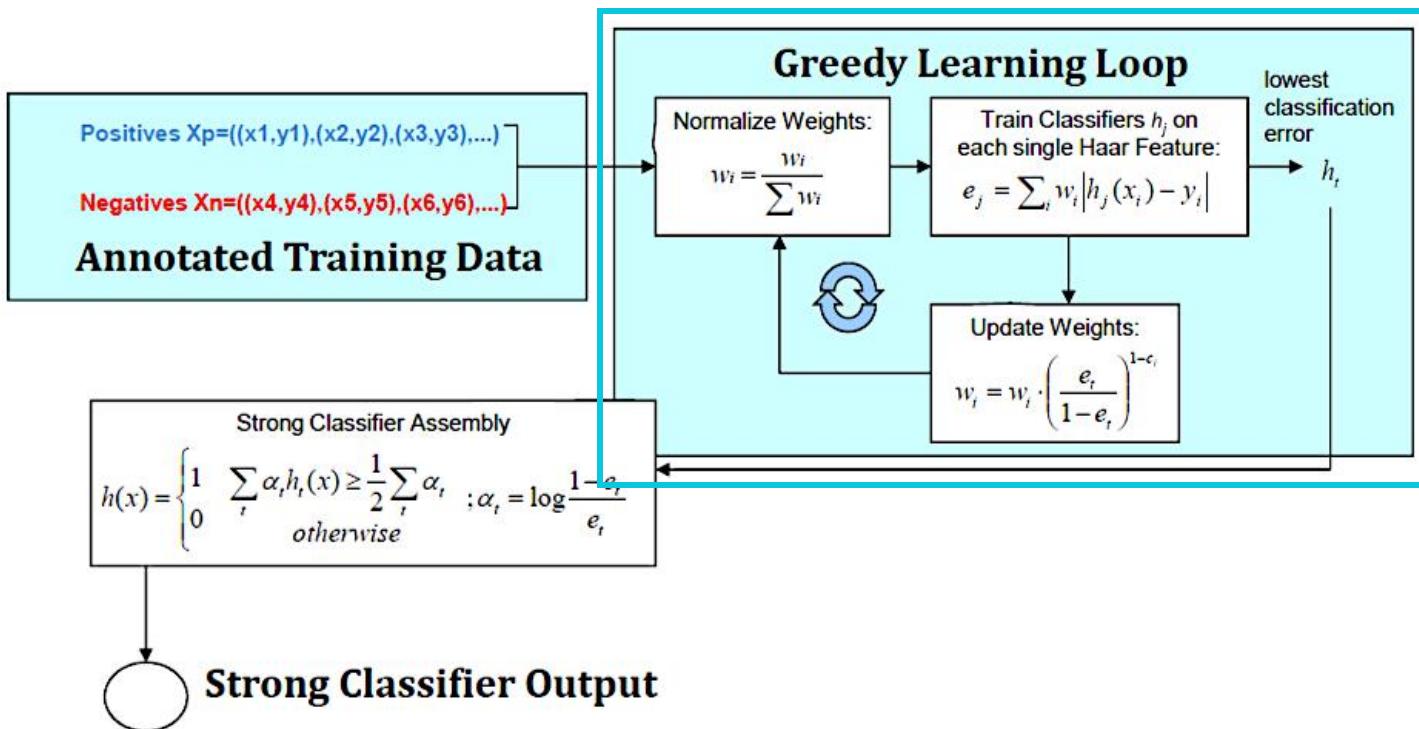


AdaBoost Classifier



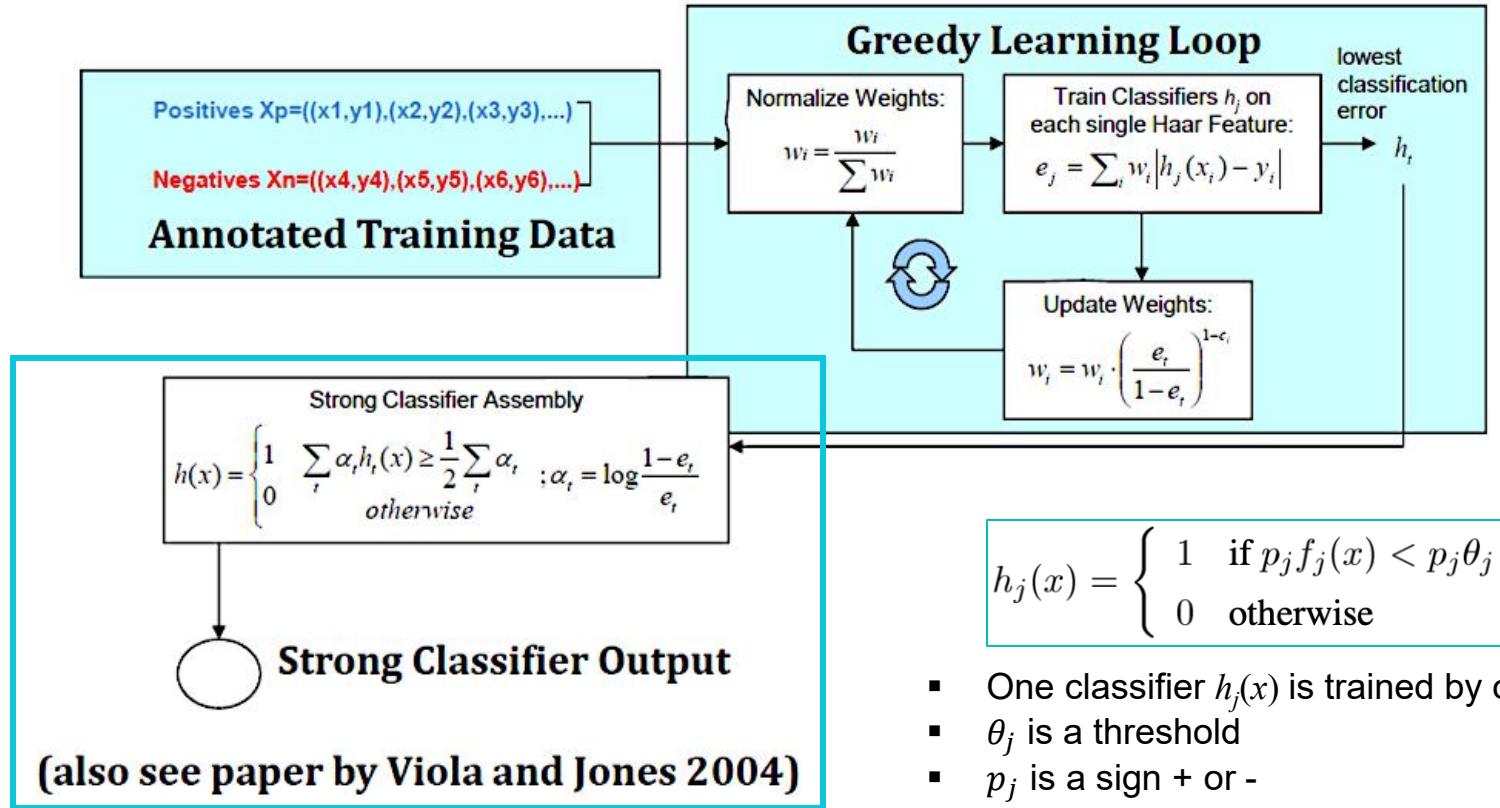
(also see paper by Viola and Jones 2004)

AdaBoost Classifier

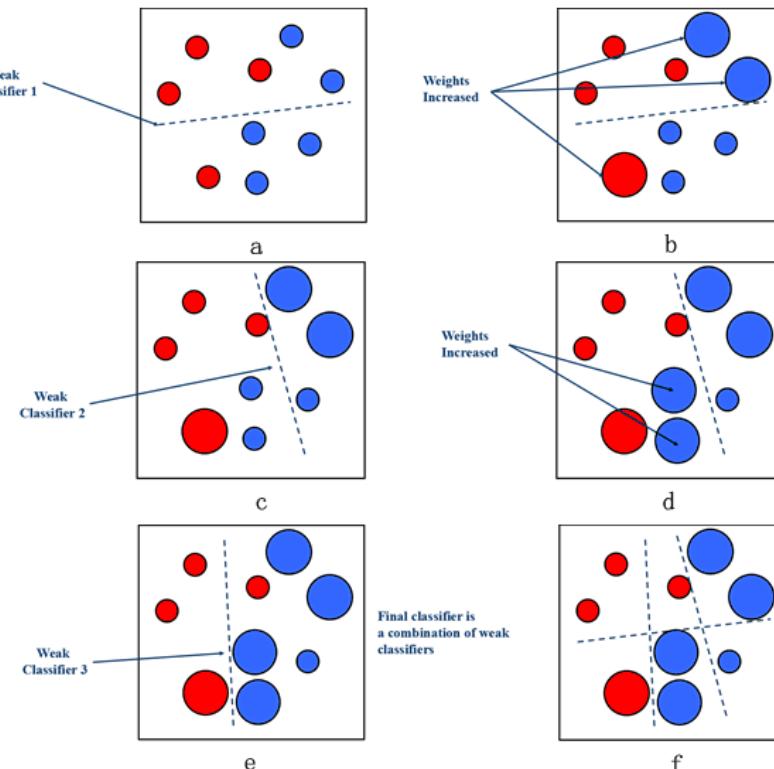
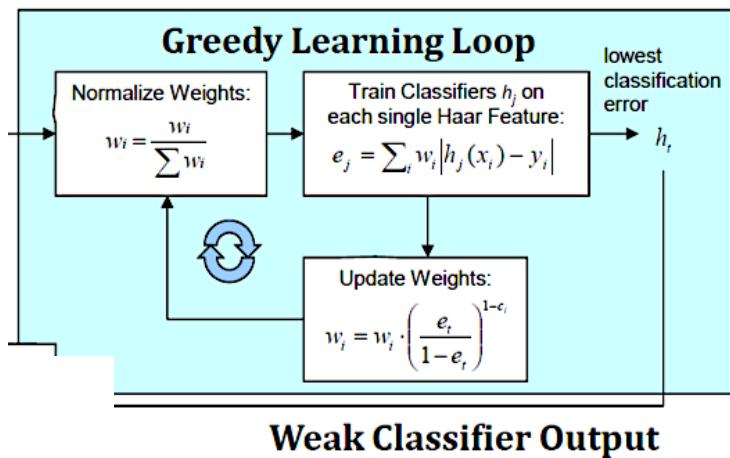


(also see paper by Viola and Jones 2004)

AdaBoost Classifier



AdaBoost Classifier



Adaboost Algorithm

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

- For $t = 1, \dots, T$:

- Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that w_t is a probability distribution.

- For each feature, j , train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t , $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
- Choose the classifier, h_t , with the lowest error ϵ_t .
- Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

Adaboost Algorithm

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha_t = \log \frac{1}{\beta_t}$

- For $t = 1, \dots, T$:

- Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that w_t is a probability distribution.

- For each feature, j , train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t , $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
- Choose the classifier, h_t , with the lowest error ϵ_t .
- Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

Adaboost Algorithm

- Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.



- For $t = 1, \dots, T$:

- Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that w_t is a probability distribution.

- For each feature, j , train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t , $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
- Choose the classifier, h_t , with the lowest error ϵ_t .
- Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.

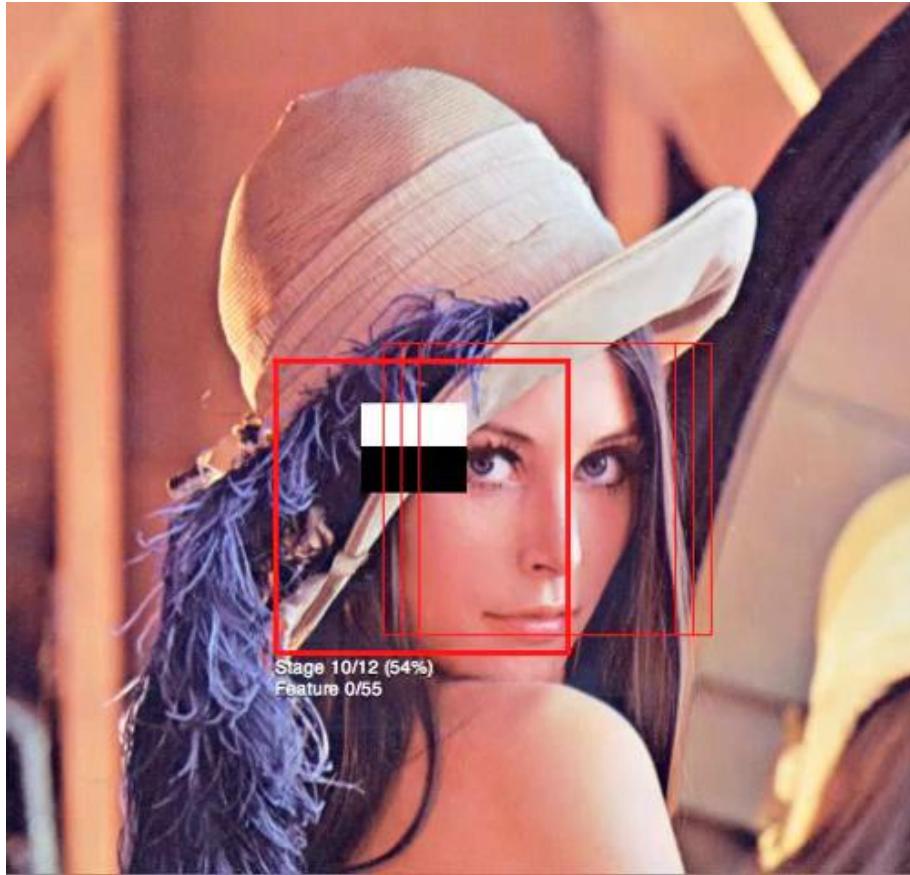
- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$



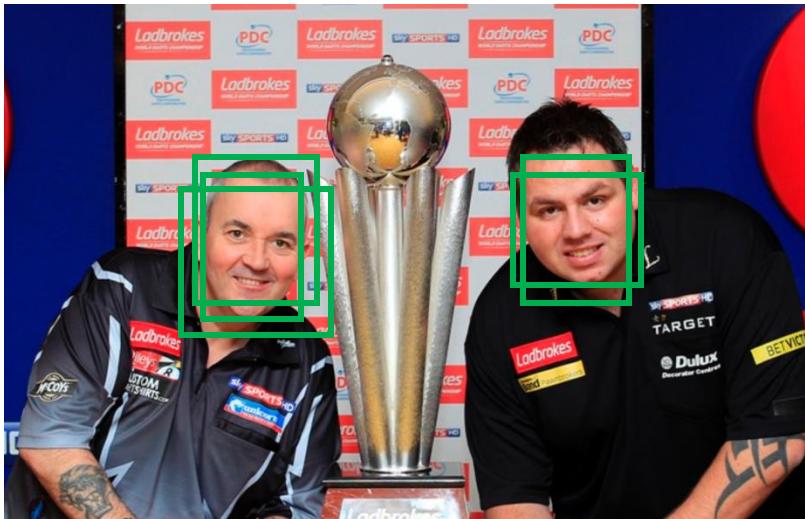
$$\text{where } \alpha_t = \log \frac{1}{\beta_t}$$

Visualisation



Non-Maximum Suppression (NMS)

- A post-processing step to remove redundant detections.



NMS



Non-Maximum Suppression (NMS)

- A post-processing step to remove redundant detections.

Algorithm 1 Non-Maximum Suppression Algorithm

Require: Set of predicted bounding boxes B , confidence scores S , IoU threshold τ , confidence threshold T

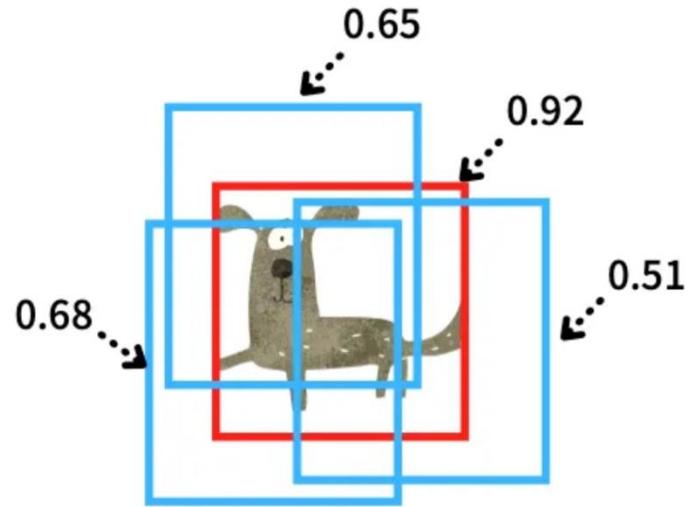
Ensure: Set of filtered bounding boxes F

```
1:  $F \leftarrow \emptyset$ 
2: Filter the boxes:  $B \leftarrow \{b \in B \mid S(b) \geq T\}$ 
3: Sort the boxes  $B$  by their confidence scores in descending order
4: while  $B \neq \emptyset$  do
5:   Select the box  $b$  with the highest confidence score
6:   Add  $b$  to the set of final boxes  $F$ :  $F \leftarrow F \cup \{b\}$ 
7:   Remove  $b$  from the set of boxes  $B$ :  $B \leftarrow B - \{b\}$ 
8:   for all remaining boxes  $r$  in  $B$  do
9:     Calculate the IoU between  $b$  and  $r$ :  $iou \leftarrow IoU(b, r)$ 
10:    if  $iou \geq \tau$  then
11:      Remove  $r$  from the set of boxes  $B$ :  $B \leftarrow B - \{r\}$ 
12:    end if
13:   end for
14: end while
```



<https://browse.arxiv.org/pdf/2304.00501.pdf>

Non-Maximum Suppression (NMS)



1. take the largest probability box

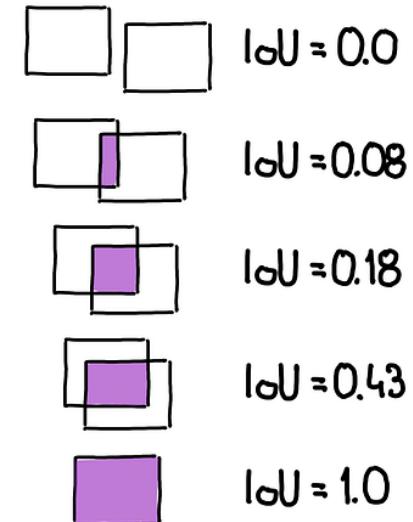
Intersection over

Union

INTERSECTION

$$\text{IoU} = \frac{\text{INTERSECTION}}{\text{UNION}}$$

UNION



2. remove others with IoU score < threshold value.

Non-Maximum Suppression (NMS)

1. Filter Bounding Boxes by Probability:

2. Sort Bounding Boxes by Probability:

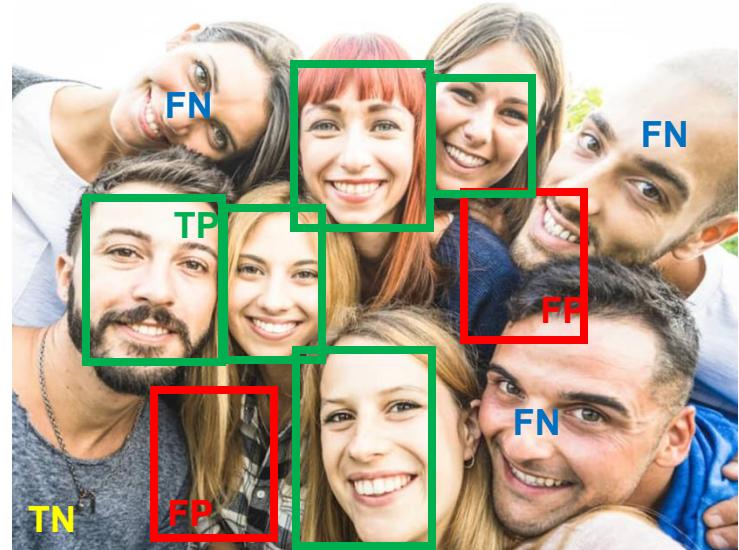
3. Non-Maximum Suppression Loop:



Performance Considerations

		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$

<https://medium.com/@m.virk1/classification-metrics-65b79bfdd776>



<https://www.quickanddirtytips.com/articles/people-or-persons/>

Performance Considerations

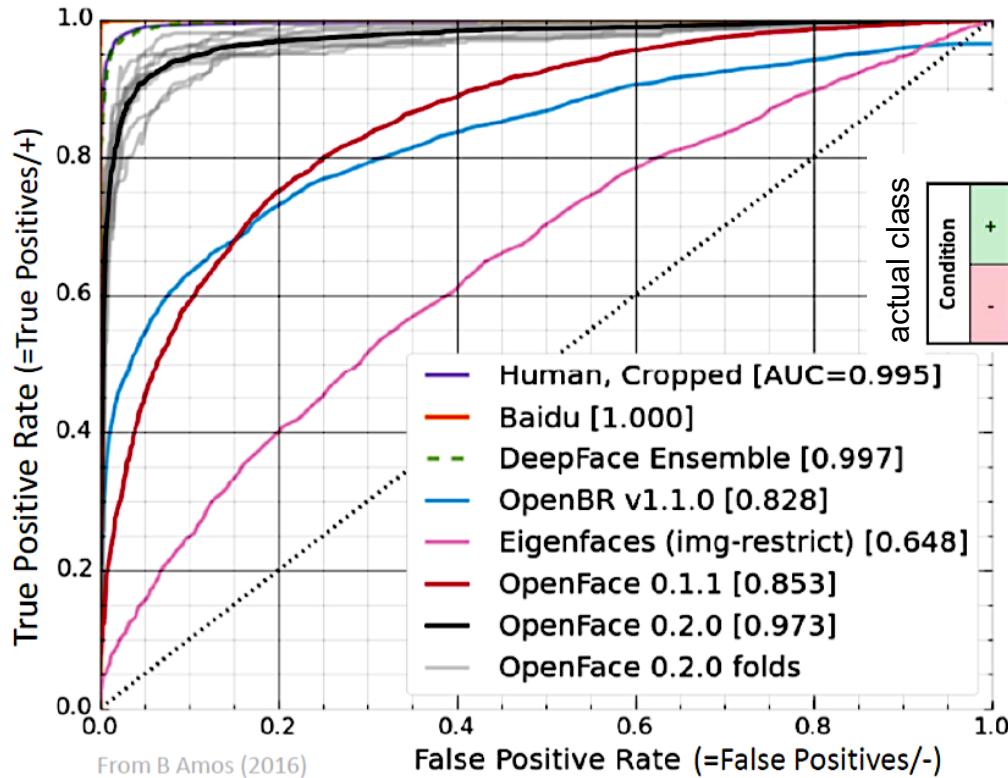
		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
	Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$	

F-score

$$F_1 = 2 \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2\text{tp}}{2\text{tp} + \text{fp} + \text{fn}}$$

<https://medium.com/@m.virk1/classification-metrics-65b79bfdd776>

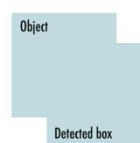
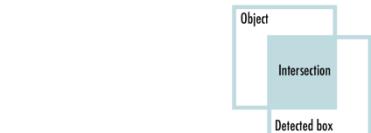
Receiver Operating Characteristic (ROC) Curve



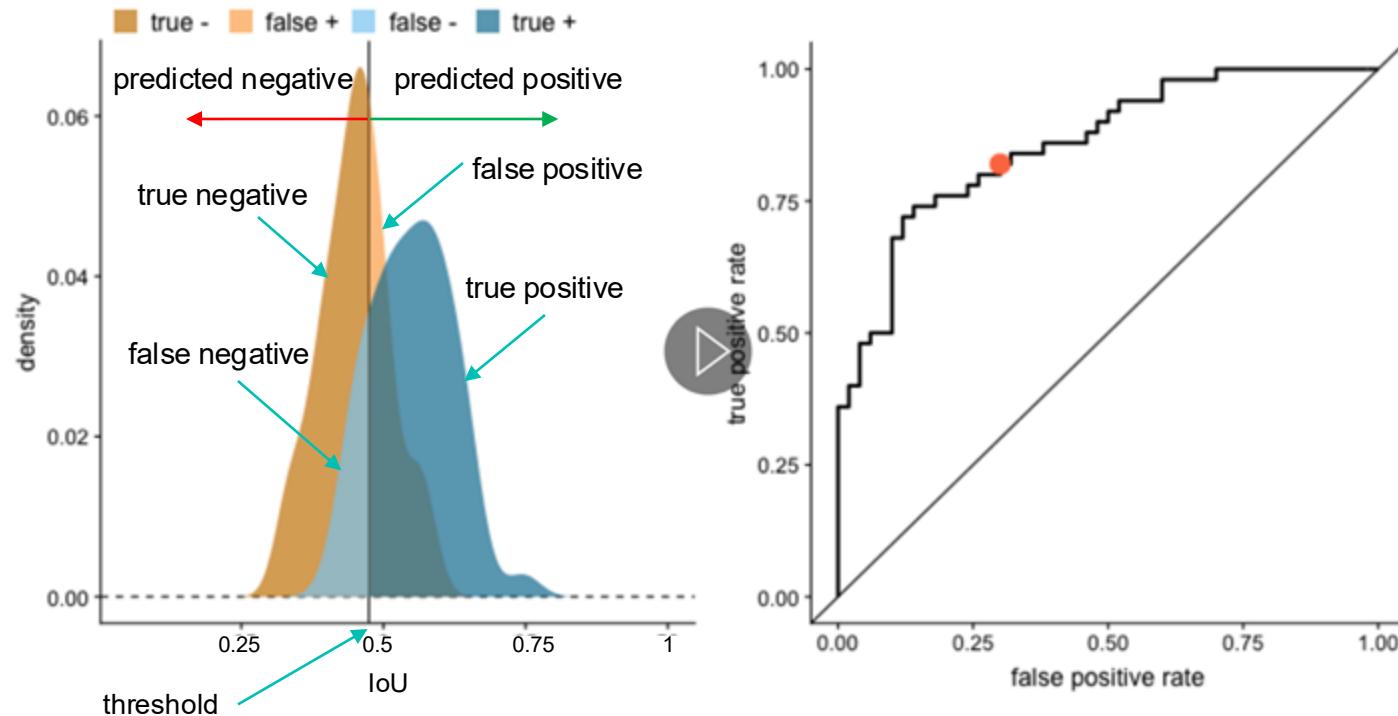
predicted class	
Classification	
Condition	
Positive	Negative
+	True Positive False Negative
-	False Positive True Negative

From R Wingate (2016)

Intersection over Union



Receiver Operating Characteristic (ROC) Curve

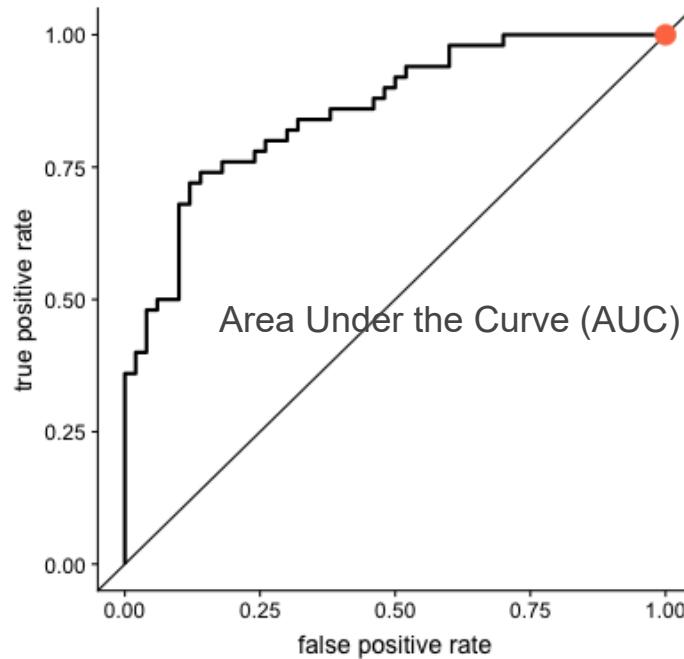
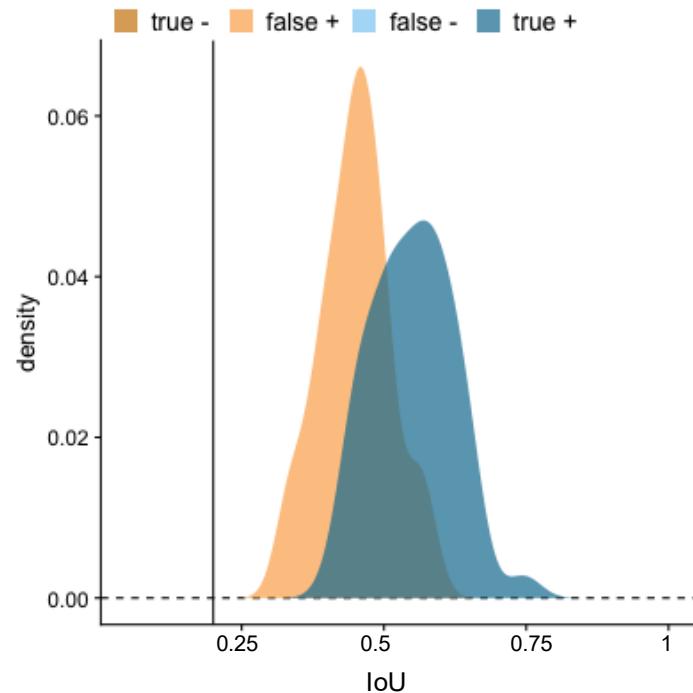


blue: actual positives (e.g. faces)

yellow: actual negatives (e.g. background)

<https://paulvanderlaken.com/2019/08/16/roc-auc-precision-and-recall-visually-explained/>

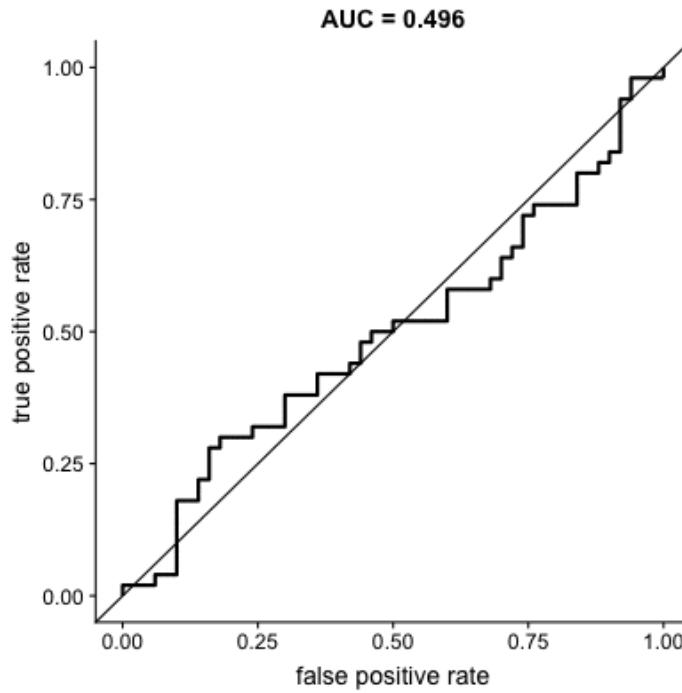
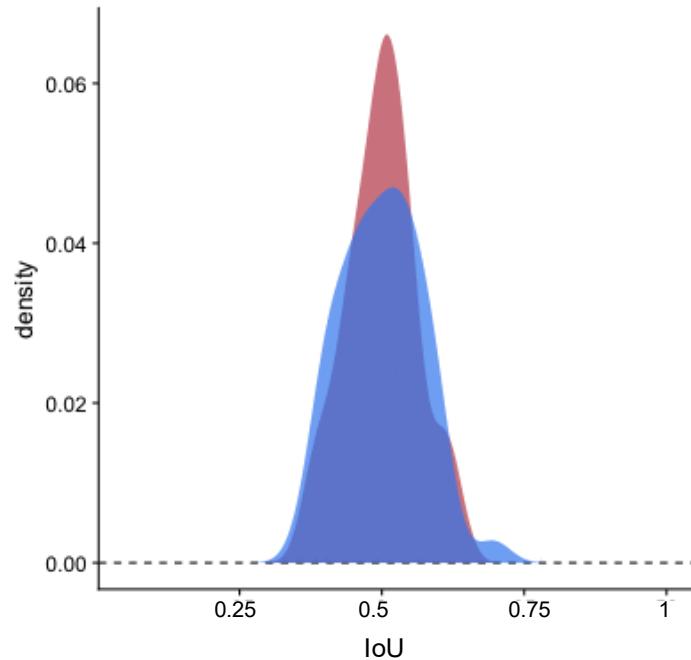
Receiver Operating Characteristic (ROC) Curve



blue: actual positives (e.g. faces)
yellow: actual negatives (e.g. background)

<https://paulvanderlaken.com/2019/08/16/roc-auc-precision-and-recall-visually-explained/>

Receiver Operating Characteristic (ROC) Curve



Next

Stereo : Epipolar Geometry

