Stereo vision

Javier González Jiménez

Reference material:

- *Computer Vision: Algorithms and Applications*. Richard Szeliski. Springer. 2010.

http://szeliski.org/Book

Some slides courtesy of:
- S. Seitz (Washinton Univ.)
- D. Murray (Oxford Univ.)

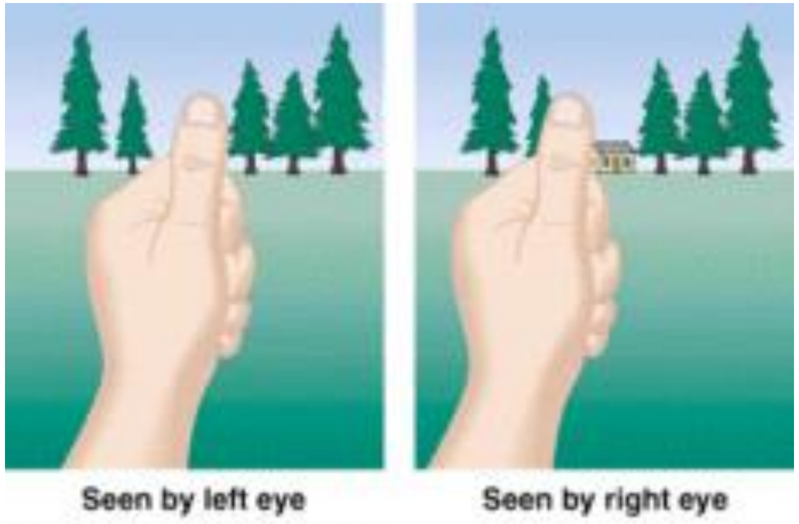
Content

1. Introduction
2. Triangulation
3. Constraints for correspondence
4. Epipolar geometry
5. Reconstruction from stereo

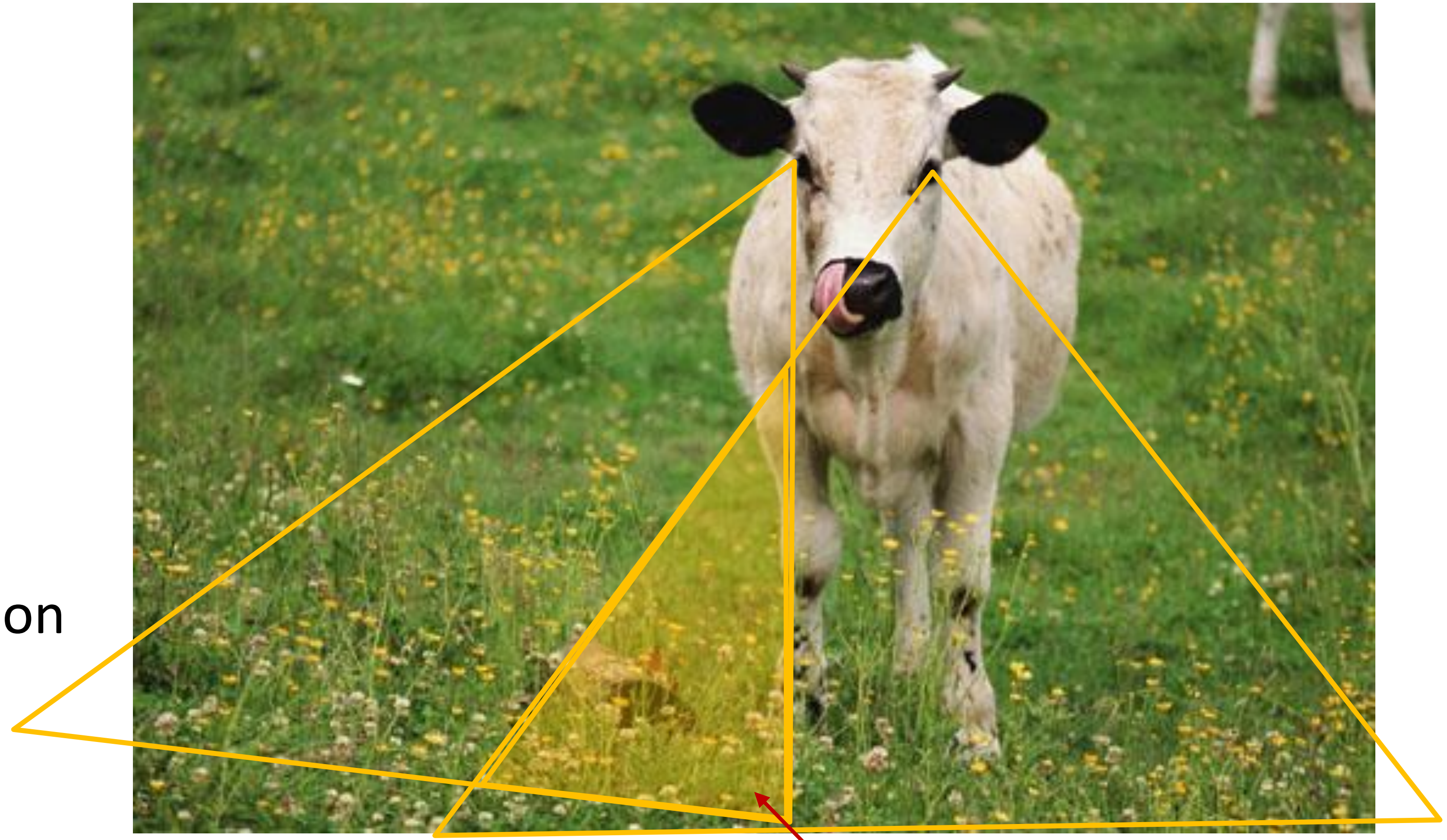
Appendix: RANSAC

1. Introduction

- **Objective**: recover the 3D info from images
- **3D information**: shape, size and location of objects in space
- Humans (and most animals) get 3D info through the "combination" of two images: Stereo vision
- **Principle of Stereo Vision**: Objects project on different locations in the two images: the closer the object the more different (separated) are their projections



Stereo vision in animals

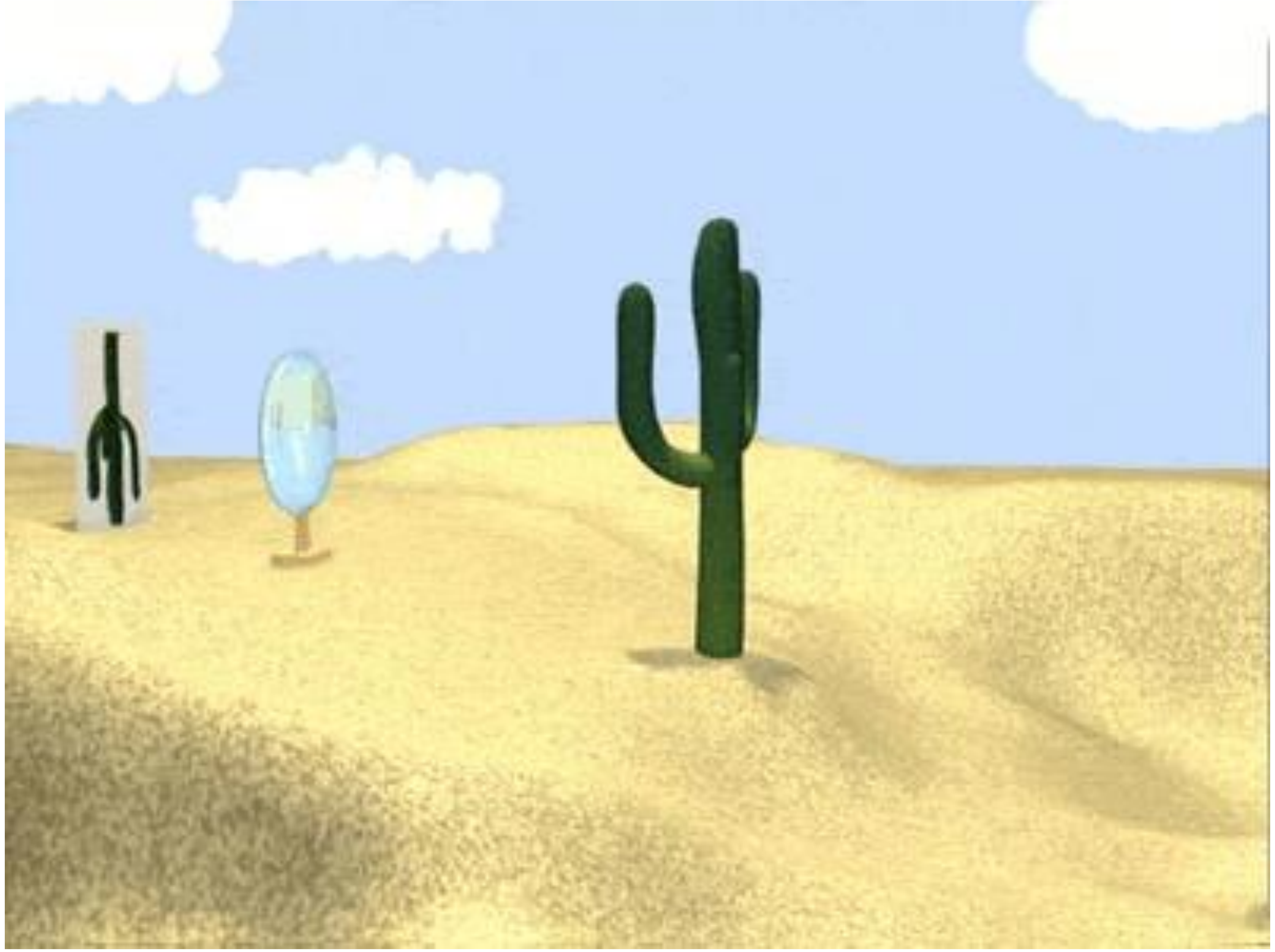


Panoramic vision of ~300º

Only in this overlaped volumen it can do stereo (~30º)

1. Introduction

One image does not suffice to infer depth

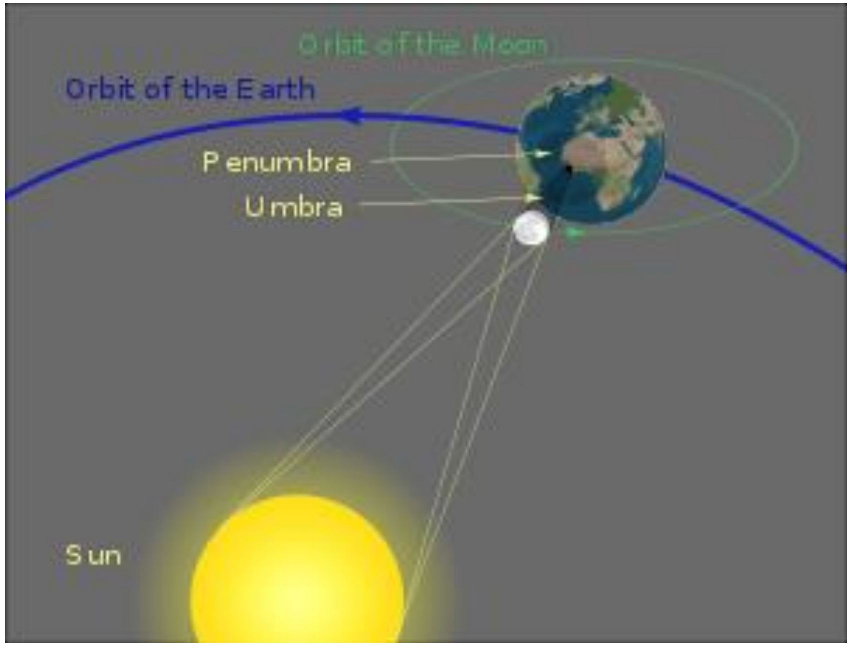


The ratio depth(Z)/size(X) remains constant in the image

1. Introduction

One image does not suffice to infer depth

Total Solar Eclipse

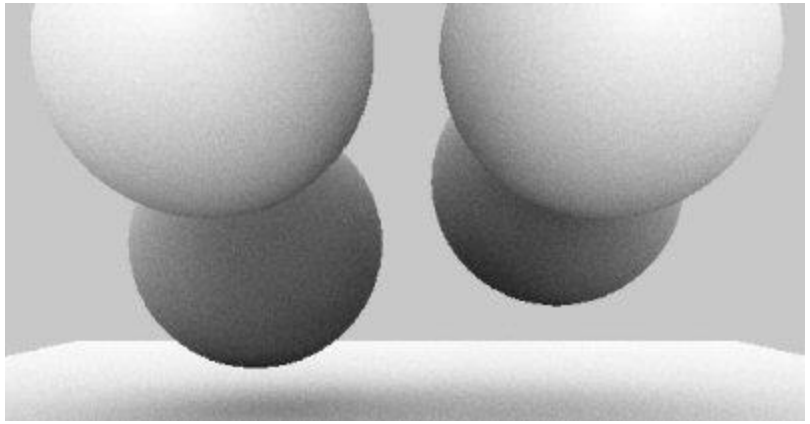


Sun's diameter is about 400 times greater - but the **sun** is also about 400 times further

1. Introduction

3D visual cues

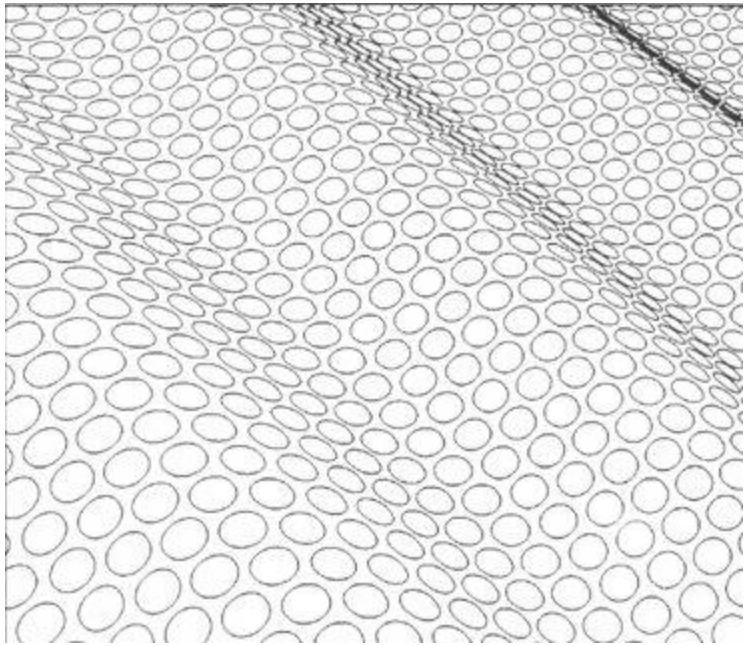
- Shading

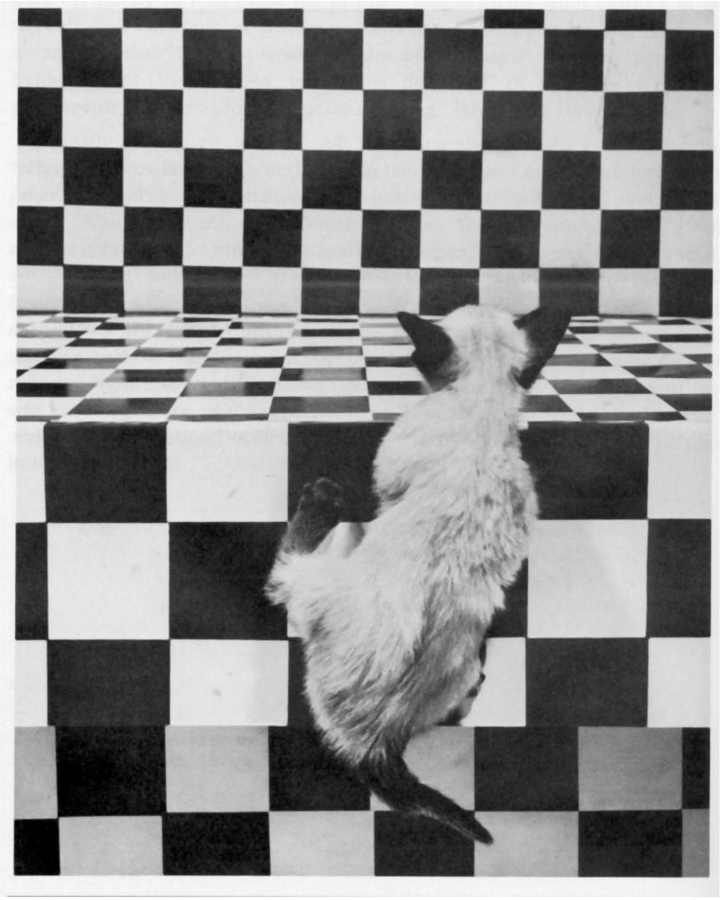


1. Introduction

3D visual cues

- Shading
- Texture





1. Introduction

But, one image does give 3D visual cues with the help from

- Shading
- Texture
- Focus

1. Introduction

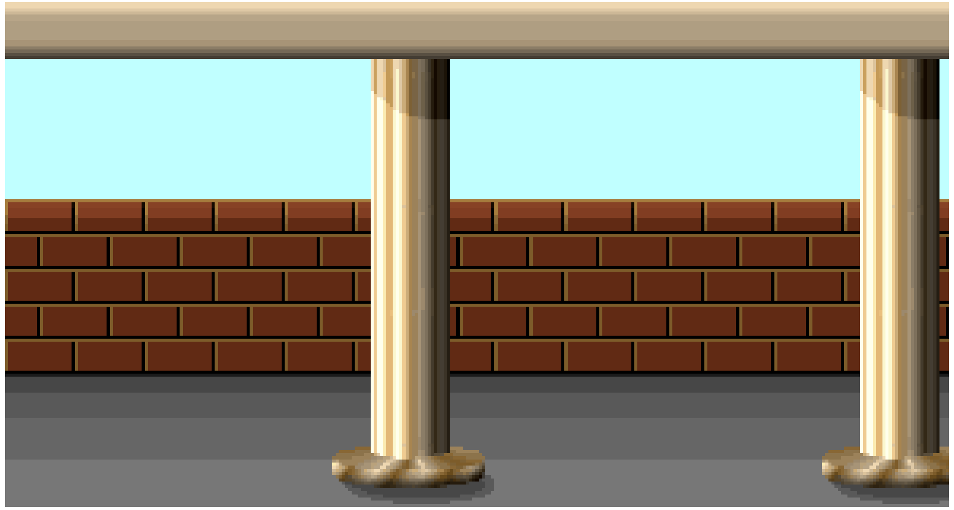
3D visual cues

- Shading
- Texture
- Focus
- Perspective

1. Introduction

3D visual cues

- Shading
- Texture
- Focus
- Perspective
- Motion & Optical flow

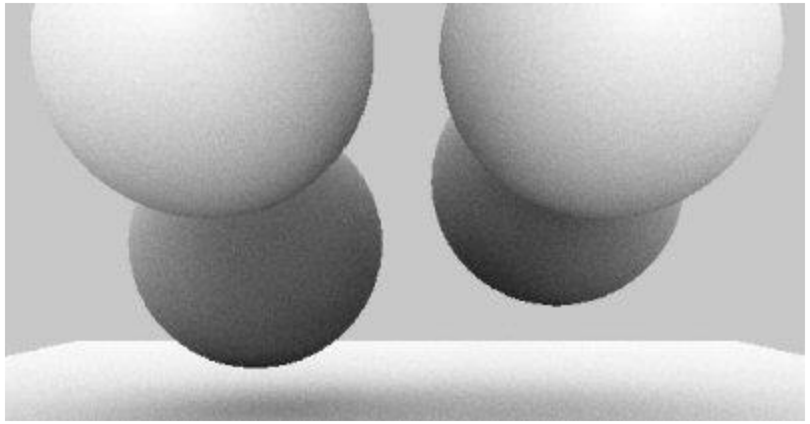


Nearer objects move faster that far away ones

1. Introduction

3D visual cues

- Shading
- Texture
- Focus
- Perspective
- Motion & Optical flow
- Occlusion

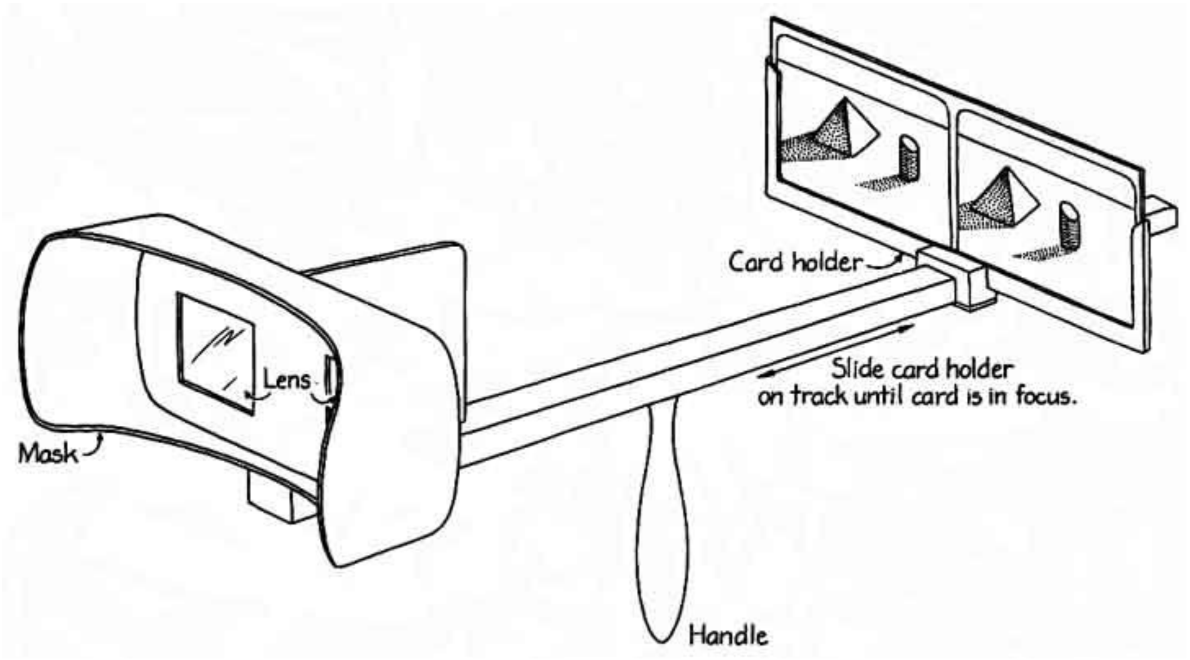


All these techniques are called:

Shape/Depth From X [X = shading, texture, focus, motion, ...]

1. Introduction

Holmes Stereoscope (procursor of Stereo Displays)



Designed by Oliver Wendell Holmes in 1861

1. Introduction

Stereo displays

- The left and right images are displayed as red and blue
- Very popular in the 50's

Anaglyph Red-Cyan Glasses

1. Introduction

Modern technology:

- Two projections of polarized movies
- Polarized glasses to see each image with an eye

Polarized Glasses (passive)

Active Shutter Glasses

3D camera

1. Introduction

Virtual reality glasses

Samsung GR VR

1. Introduction

Scene reconstruction does not require two cameras

... but two different views of the scene

Possibilities:

- StereoVision system

  Two cameras

  Images taken simultaneously

  Relative pose of the cameras known

- Structure-from-Motion (SFM)

  Camera from different unknown, unordered locations (typically with large baseline)

  Offline global optimization for the relative pose and structure → *Bundle Adjustment*

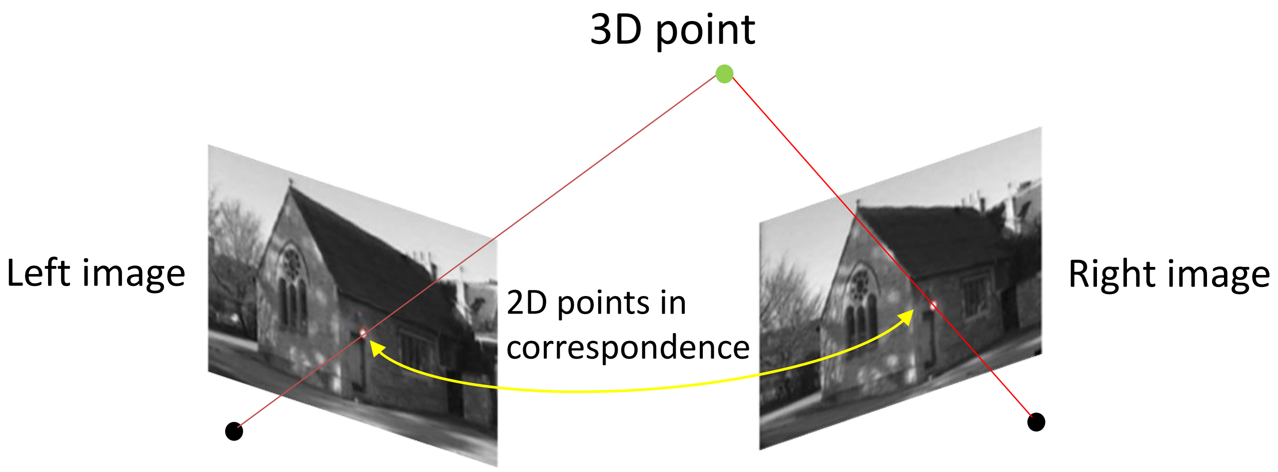
  Cameras can be different, i.e. different matriz $\mathbf{K}$

- MonoVisual SLAM (Simultaneous Localization And Mapping)

  Alike SFM but sequentially, small baseline and in real time

1. Introduction

At first, stereo vision seems a straightforward problem: intersection of two lines



In practice, though, it's not that simple. Two main problems:

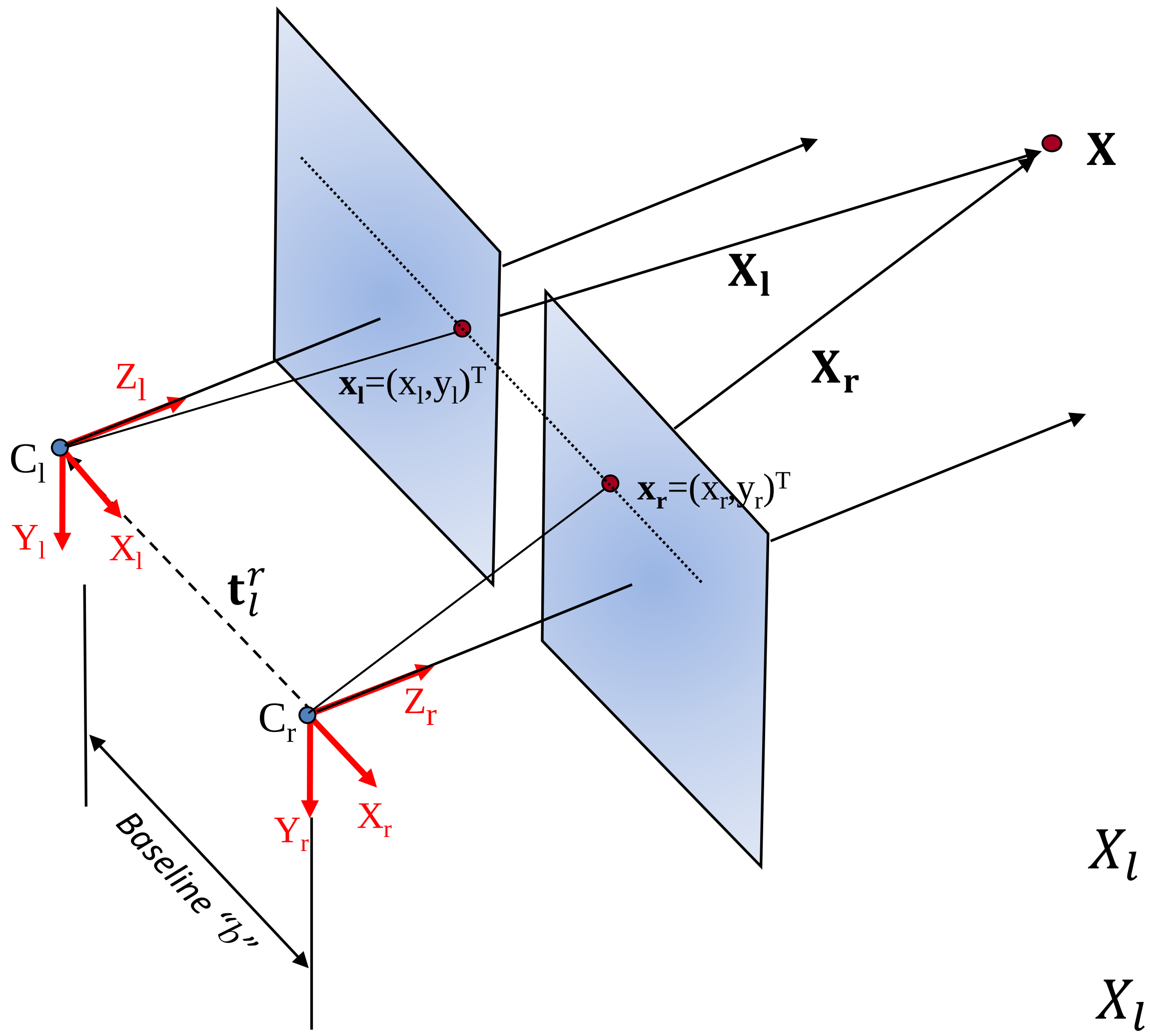
- Feature matching: Detect features (point, segments,...) in one image and their correspondences in the other
- Geometric Triangulation: Compute depth given the features in correspondence

Both issues require perfect knowledge of the relative pose between the cameras, given by the Epipolar Geometry

2. Triangulation

Ideal configuration:

$$X_l = X_r + b \quad Z_l = Z_r = Z$$
$$Y_l = Y_r = Y \quad y_l = y_r = y$$



Point in the **Left camera**:

$$\mathbf{X_l} = \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix} = \begin{bmatrix} X_l \\ Y \\ Z \end{bmatrix} = \frac{Z}{f}\begin{bmatrix} x_l \\ y \\ f \end{bmatrix}$$

$$\boxed{X_l = \frac{Z}{f} x_l}$$

$$\boxed{Y = \frac{Z}{f} y}$$

Point in the **Right camera**:

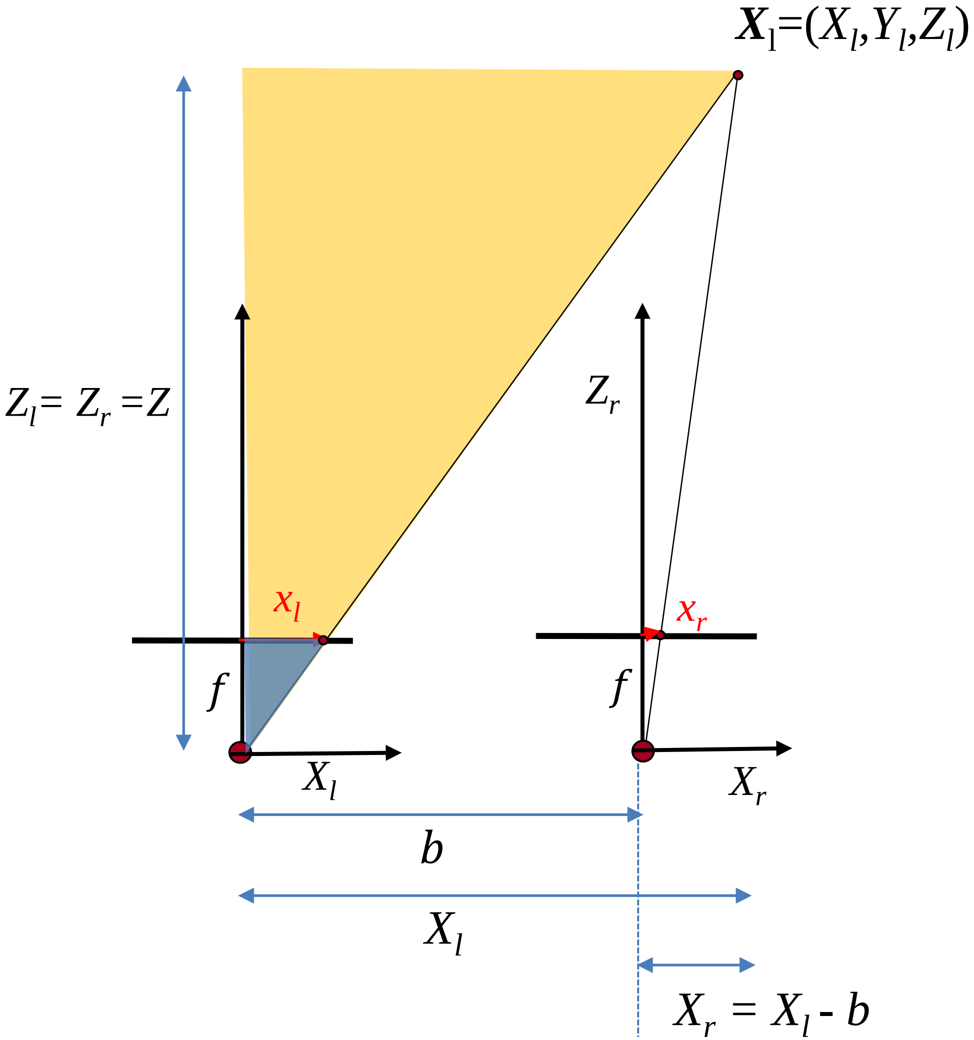
$$\mathbf{X_r} = \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = \begin{bmatrix} X_l - b \\ Y \\ Z \end{bmatrix} = \frac{Z}{f}\begin{bmatrix} x_r \\ y \\ f \end{bmatrix} \quad X_l - b = \frac{Z}{f} x_r$$

$$\left.\begin{aligned} X_l &= \frac{Z}{f} x_l \\ X_l - b &= \frac{Z}{f} x_r \end{aligned}\right\} \quad \boxed{Z = \frac{bf}{(x_l - x_r)} = \frac{bf}{d}}$$

disparity

Another way to solve Z in the ideal configuration:

- we can work with triangulation in a plane
- principle of similar triangles applies



Similar triangles:

$$\frac{x_l}{f} = \frac{X_l}{Z}$$

$$\frac{x_r}{f} = \frac{X_r}{Z} = \frac{X_l - b}{Z}$$

Eliminating X_l

$$\frac{b}{Z} = \frac{x_r - x_l}{f} \rightarrow Z = b\frac{f}{d}$$

Disparity: $d>=0$

Y axis: $$\frac{y_l}{f} = \frac{y_r}{f} = \frac{Y}{Z}$$

$$\boxed{\boldsymbol{X}_l = \begin{bmatrix} b\dfrac{x_l}{d} & b\dfrac{y_l}{d} & b\dfrac{f}{d} \end{bmatrix}^T = \frac{b}{d}\begin{bmatrix} x_l & y_l & f \end{bmatrix}^T}$$

Ideal configuration: $\boxed{\mathbf{X}_l = \left[b\frac{x_l}{d} \quad b\frac{y_l}{d} \quad b\frac{f}{d} \right]^T = \frac{b}{d}\left[x_l \quad y_l \quad f \right]^T}$

Disparity d and coordinates (x_l, y_l) are in meters (measured in the sensor)!

Both images are related by an AFFINE transformation

Computer image matrix (units in pixels)

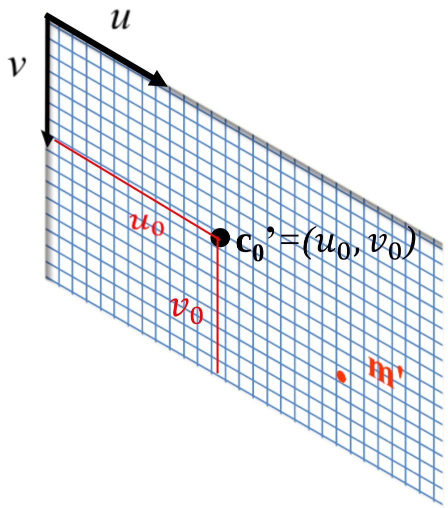
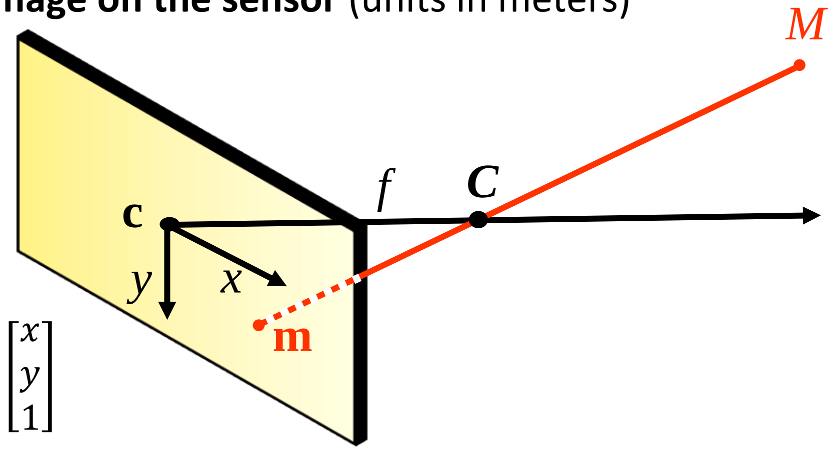


Image on the sensor (units in meters)



k_x, k_y in pixels/mm

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & u_0 \\ 0 & k_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$u = xk_x + u_0 \quad \Rightarrow \quad x = \frac{1}{k_x}(u - u_0)$$

$$v = yk_y + v_0 \quad \Rightarrow \quad y = \frac{1}{k_y}(v - v_0)$$

Disparity in mm

Disparity in pixels

$$d = x_l - x_r = \frac{1}{k_x}(u_l - u_r) = \frac{1}{k_x} d_i$$

If $k_x = k_y = k$

Focal length in pixels

$$\mathrm{X}_l = \frac{k_x b}{d_i}\left[\frac{1}{k_x}(u_l - u_0) \quad \frac{1}{k_y}(v_l - v_0) \quad f \right]^T = \frac{b}{d_i}\left[(u_l - u_0) \quad (v_l - v_0) \quad kf \right]^T$$

2. Triangulation

What is the accuracy of the reconstructed 3D points?

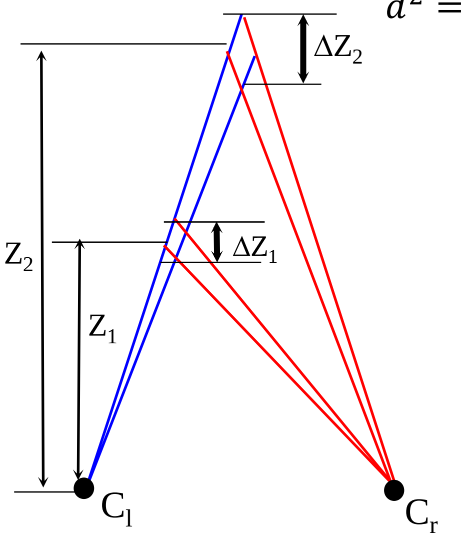
$$Z = b\frac{f}{d}$$

Error in the detection of image points → error in disparity → error in depth (Z)

How error in disparity propagates to Z?:

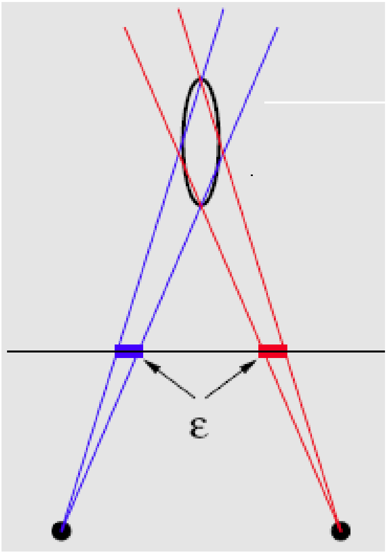
$$Z = b\frac{f}{d} \longrightarrow \frac{\Delta Z}{\Delta d} = -\frac{bf}{d^2} \longrightarrow \Delta Z = -\frac{bf}{d^2}\Delta d = -\frac{Z^2}{fb}\Delta d$$

$$d^2 = \left(b\frac{f}{Z}\right)^2$$



Given a disparity error Δd the error in Z (ΔZ) has a square growth (Z^2)

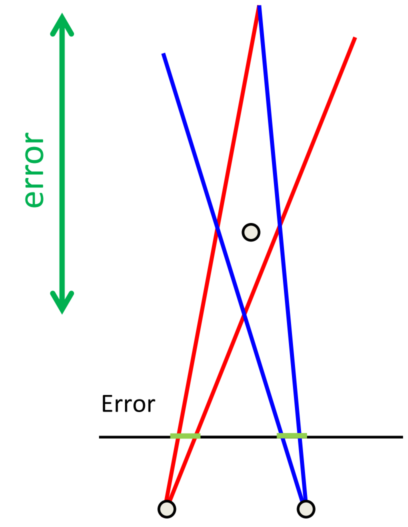
More accuracy for closer objects



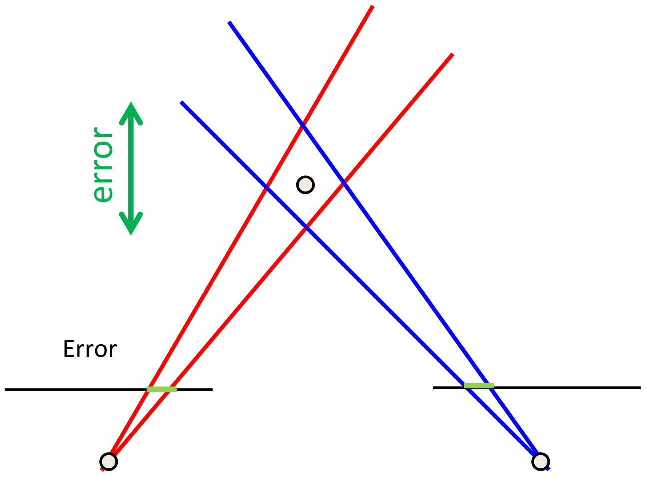
assuming same error ε for each image point

2. Triangulation

To increase accuracy of 3D points → separate the cameras (increase the baseline “b”)

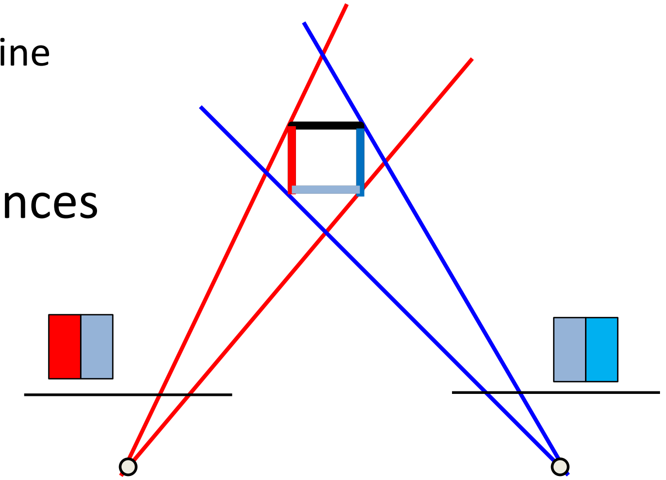


Small baseline



large baseline

But more difficult (or impossible) to find correspondences



2. Triangulation

[All points in homogeneous coordinates]

General configuration

Known: $\mathbf{P}_{\mathrm{l}}$, $\mathbf{P}_{\mathrm{r}}$
Assumptions: **P and x error-free**

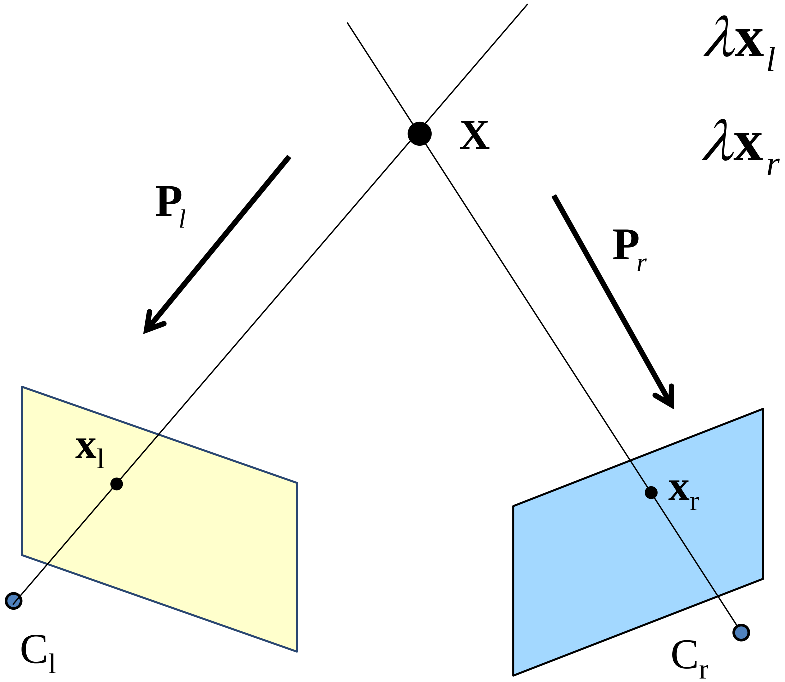
Projection lines do intersect at a 3D point

World coordinate system in the left camera **R = I**, **t = 0**

$$\lambda \mathbf{x}_l = \mathbf{P}_l \mathbf{X} = \mathbf{K}_l [\mathbf{I}|\mathbf{0}] \mathbf{X}_l$$
$$\lambda \mathbf{x}_r = \mathbf{P}_r \mathbf{X}_r = \mathbf{K}_r [\mathbf{R}|\mathbf{t}] \mathbf{X}_l$$

A system of equations where $\mathbf{X}_l$ is the unknown

$$\lambda \mathbf{x}_l = \mathbf{P}_l \mathbf{X}_l$$



$$\lambda \mathbf{x}_r = \mathbf{P}_r \mathbf{X}_r$$

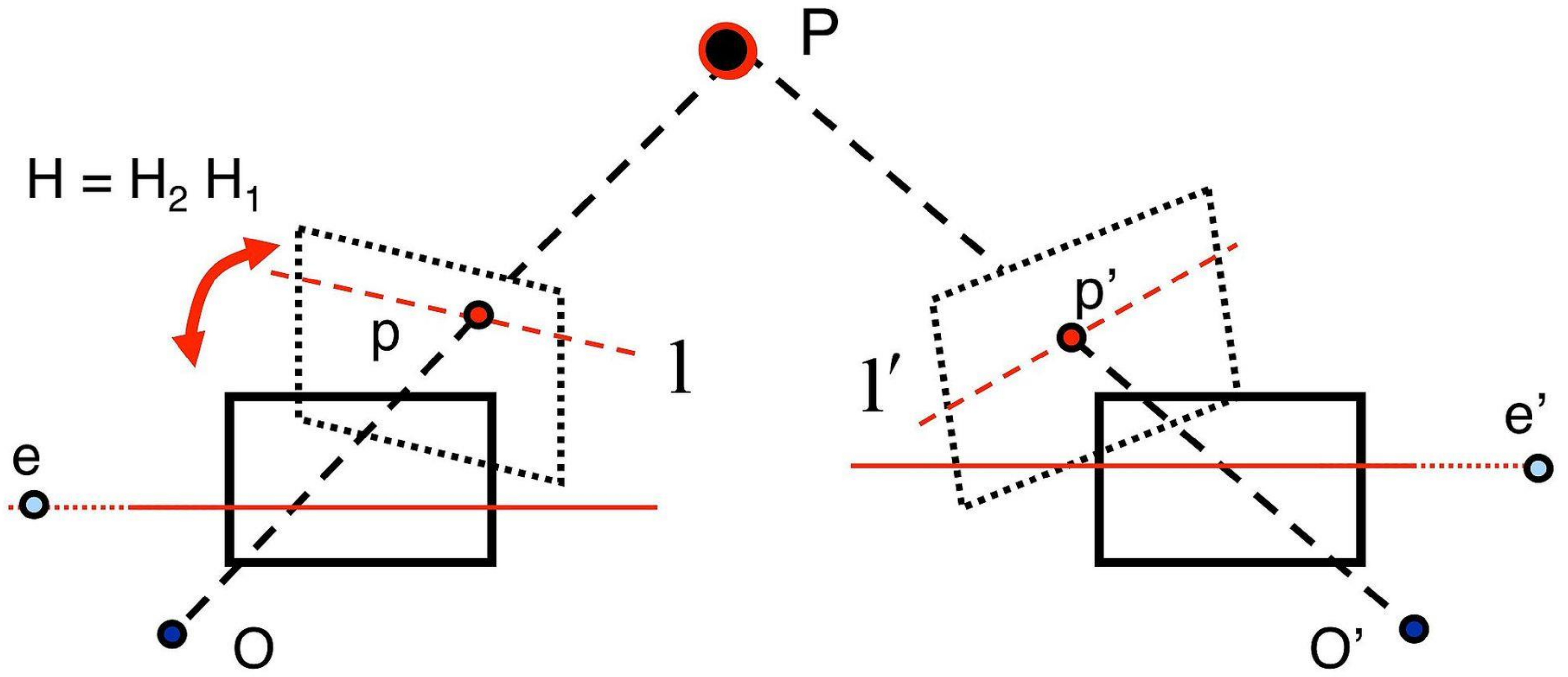
2. Triangulation

If cameras are not aligned → Do a rectification of the images

Stereo rectification:

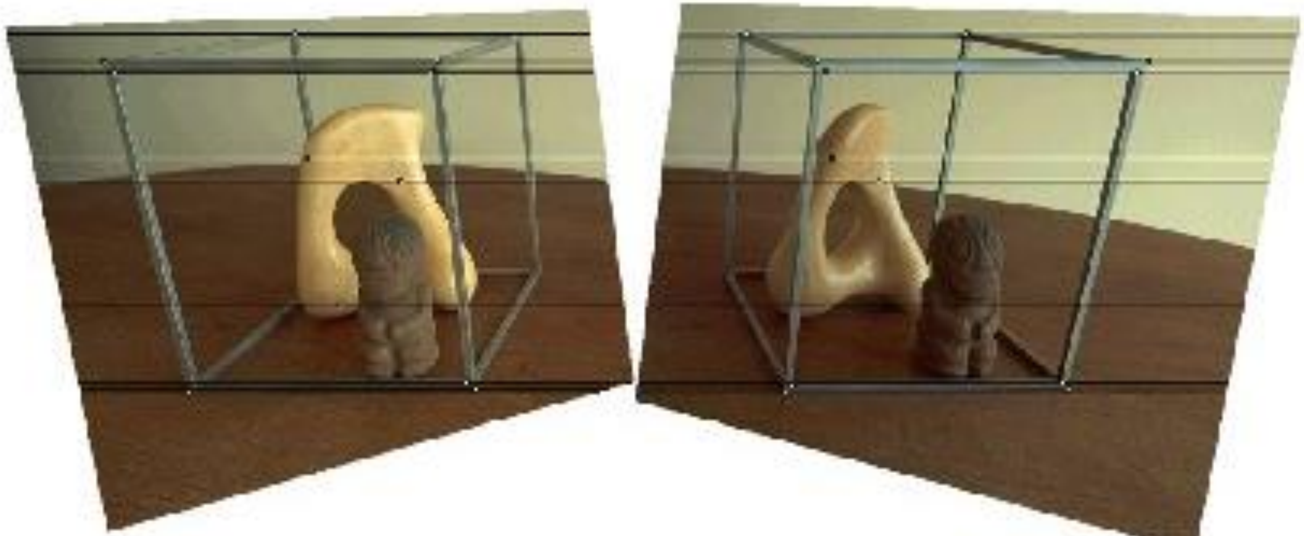
projects the images on a common plane such the epipolar lines l, l’ are horizontal in both images and at the same height

2D Homography



Epipolar lines will be explained next!

Stereo rectification (Example):



Rectified images

3. Constraints for Correspondence

Most common

We have to decide:

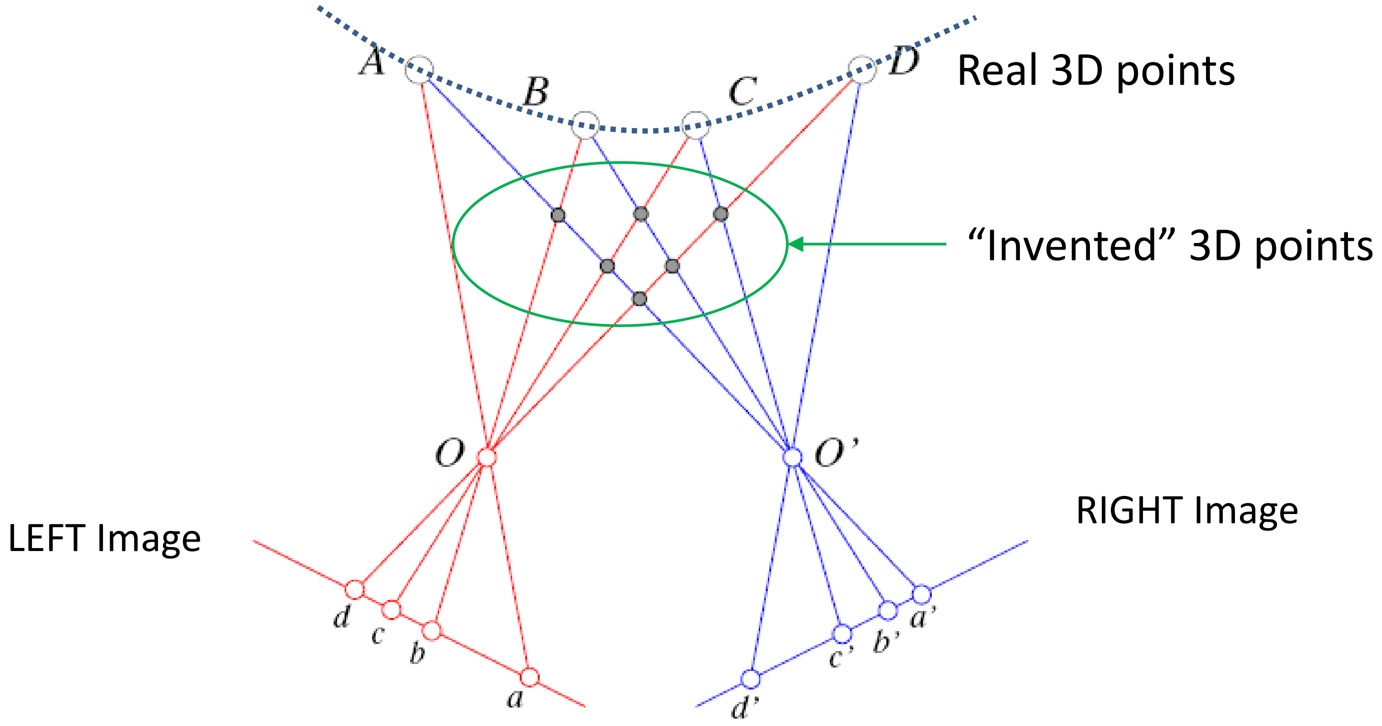
- Which feature to match: all the pixels, keypoints, edgels, segments, regions
- A robust descriptor to solve for matches

Cordoba city from satellite, at two different days and viewpoints

3. Constraints for Correspondence

Correspondence Problem = Data Association Problem

- Given a feature in the LEFT image, find its correspondence/match in the RIGHT image



If the correspondence is wrong the 3D point does not exist in the real world

3. Constraints for Correspondence

Problem: Given a feature in the LEFT image find its correspondence/match in the RIGHT one

An efficient and robust solution requires Constraints

Cameras' configuration

Epipolar geometry

Constraints from the environment (proposed by Marr&Poggio, 1981):

Max-Min disparity (limits)

Continuity of the surfaces

Uniqueness

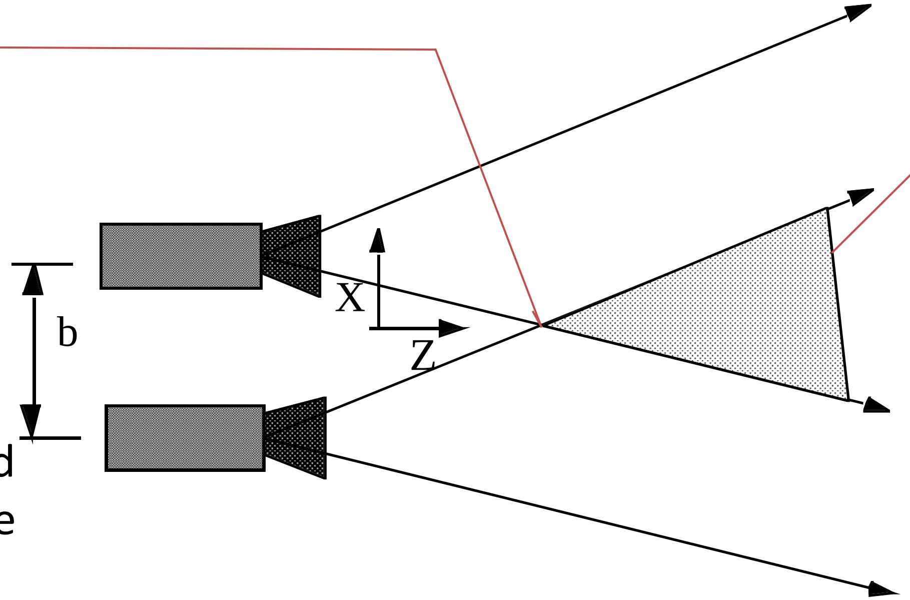
Ordering

3. Constraints for Correspondence

Constraints from the environment: Max-Min disparity allowed

Minimum Z
⇩
Maximum disparity

Maximum Z
⇩
Minimum disparity



The baseline and the focal length (f) determine the overlap in the field-of-view and thus the **maximun disparity** we can have for a pixel.
This is more a camera configuration constraint!

The maximum distance of the environment (e.g. a room) and the camera resolution determine the **minimum disparity.**

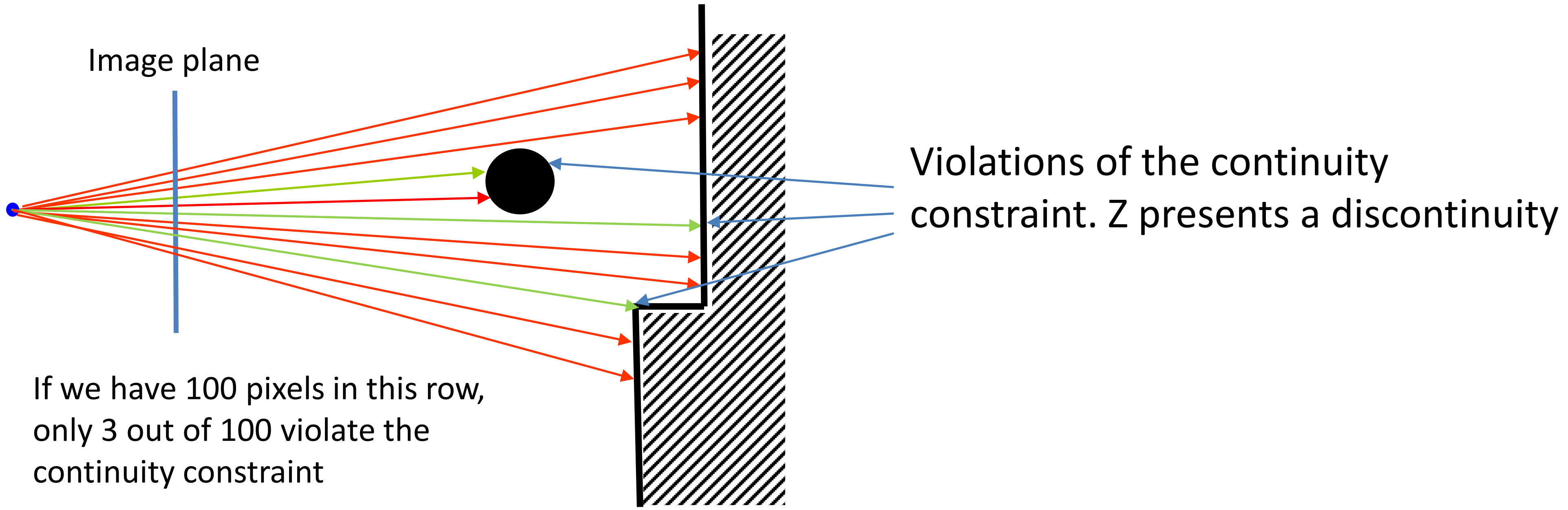
- Disparity = 0 → point at infinity
- Disparity (d_i) = 1 → $Z = \frac{b}{d_i} kf = bkf$

$$k_x = k_y = k \; (in \; \frac{pixels}{meters})$$

Constraints from the environment: Continuity of the surfaces

- Typically, surfaces in the real world are continuous → depth changes smoothly
- This fact is only violated at the occlusion borders (surfaces from different objects)

There is a **high probability that contiguos pixels have similar depth** (and therefore, disparity)



If we have 100 pixels in this row, only 3 out of 100 violate the continuity constraint

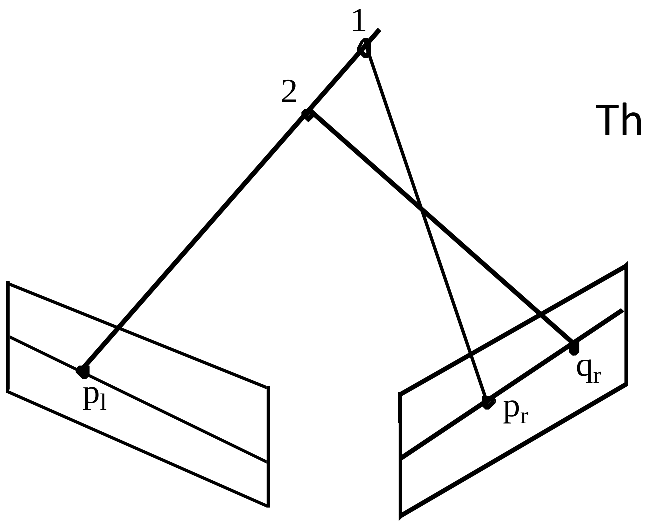
3. Constraints for Correspondence

Constraints from the environment: Uniqueness

Each pixel of the image is the **projection of only one point** in 3D

⬇

A pixel will have only one pixel in correspondence in the other image

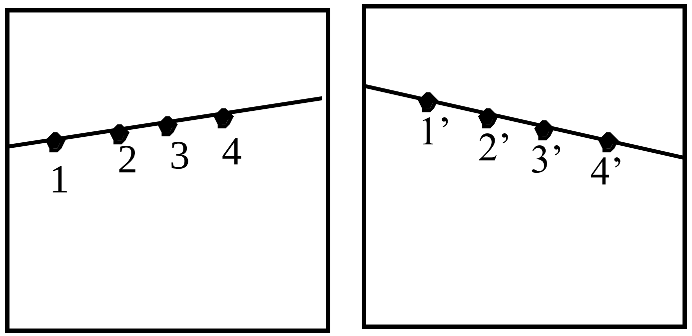


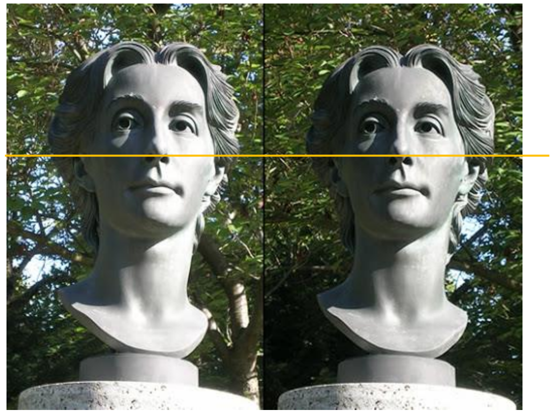
This is not allowed

3. Constraints for Correspondence

Constraints from the environment: Ordering

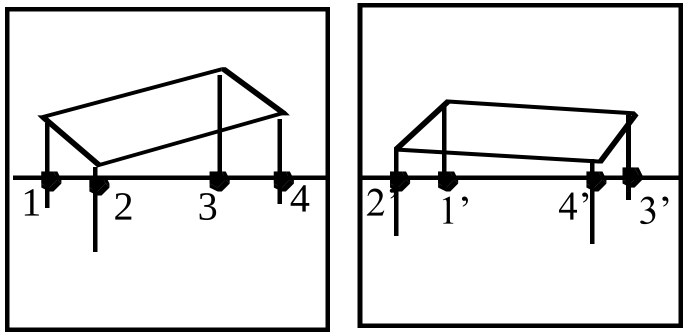
Points along conjugate epipolar lines (explained next) follow the same order





www.bke.org

This is not always true ….

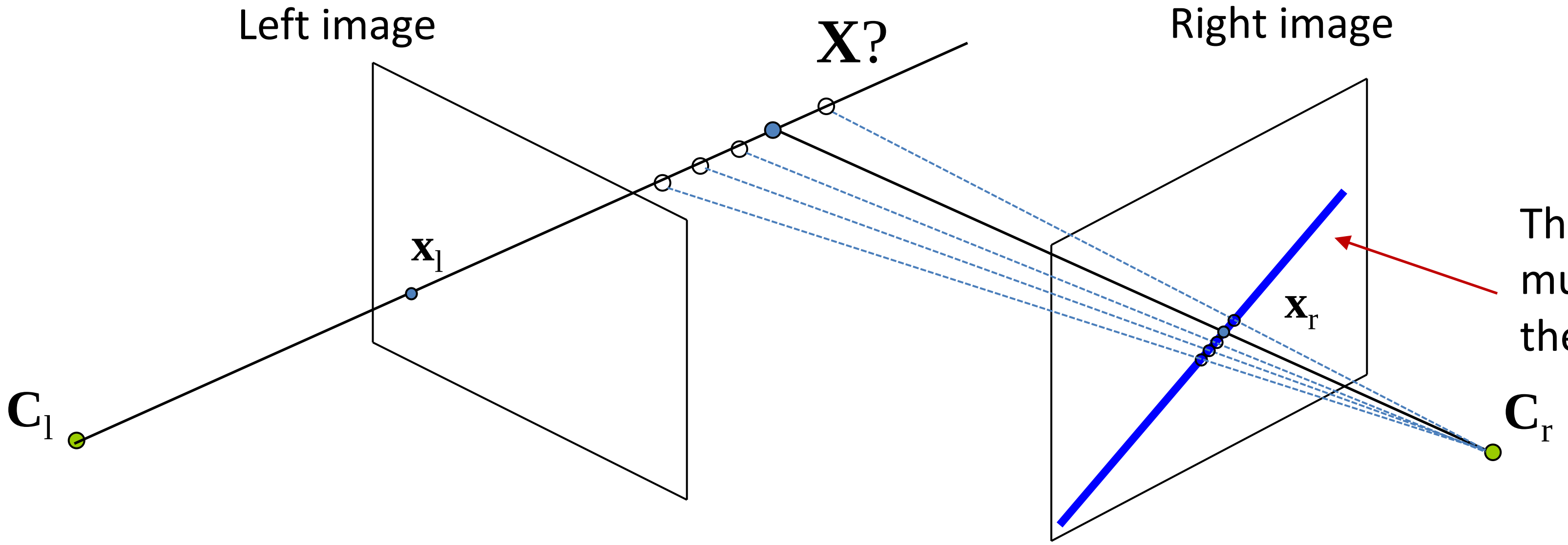


Violation of the ordering constraint

3. Constraints for Correspondence

Epipolar constraint

The correspondence in the right image of any point $\mathbf{x}_l$ must be in the projection of the line through $\mathbf{x}_l$: this projection is the epipolar line of $\mathbf{x}_l$

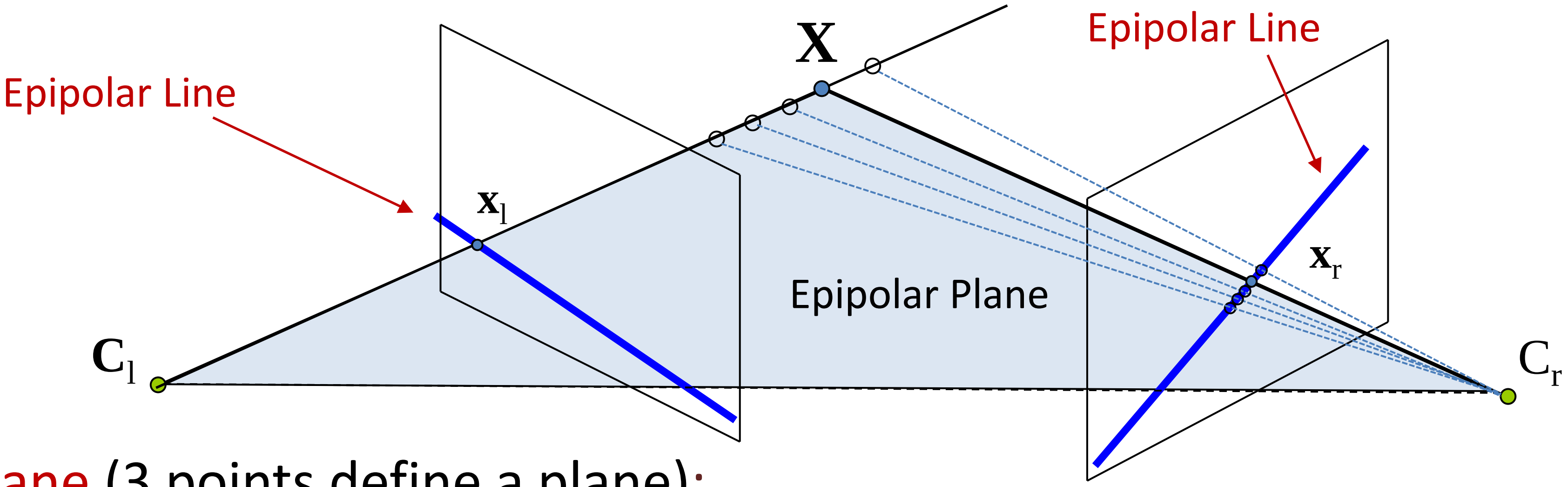


The correspondent point of $\mathbf{x}_l$ must be in this line. This is the Epipolar Line of $\mathbf{x}_l$.

Correspondent epipolar lines

4. Epipolar Geometry

Epipolar geometry is the set of geometric contraints between 2 views of a scene, forced by the cameras' relative pose



Epipolar plane (3 points define a plane):

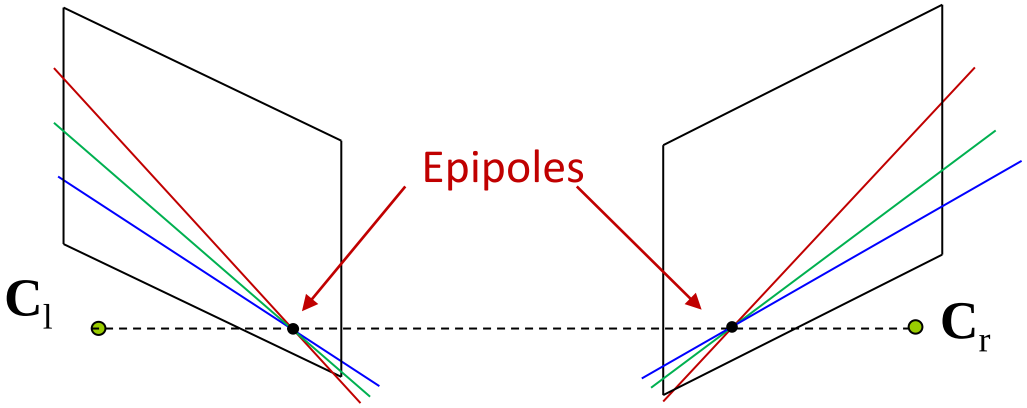
- Given by the optical center: $\mathbf{C}_l$, $\mathbf{C}_r$, and any 3D point $\mathbf{X}$ (or any of its projections $\mathbf{x}_l$ or $\mathbf{x}_r$).
- Intersects the images at the **conjugate epipolar lines** of the StereoVision System.

Given a **two-camera configuration (represented by R,t),** for each 3D point there is a unique epipolar plane and, consequently, a unique pair of epipolar lines.

Epipole:

- Projection of the **optical center (focal point)** of a camera on the other image plane (usually outside the boundary of the image)

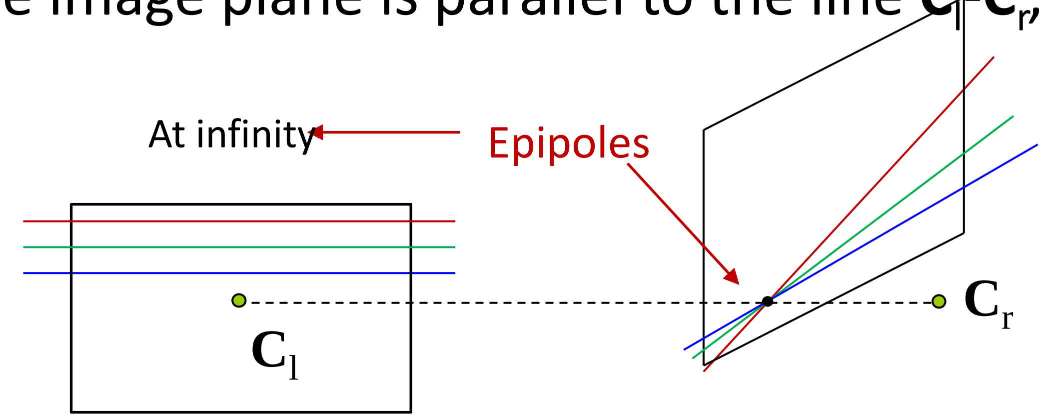
- All the epipolar lines of an image intersect at the epipole



Why? Two ways of seing this:

1) The epipolar lines are projections of rays, in the other image, that depart from the optical center
2) All the epipolar planes rotate around the line $\mathbf{C}_l$-$\mathbf{C}_r$. This line is in all the epipolar planes. The epipoles are in all the epipolar planes.

- If one image plane is parallel to the line $\mathbf{C}_l$-$\mathbf{C}_r$, the epipole in this plane is at infinity



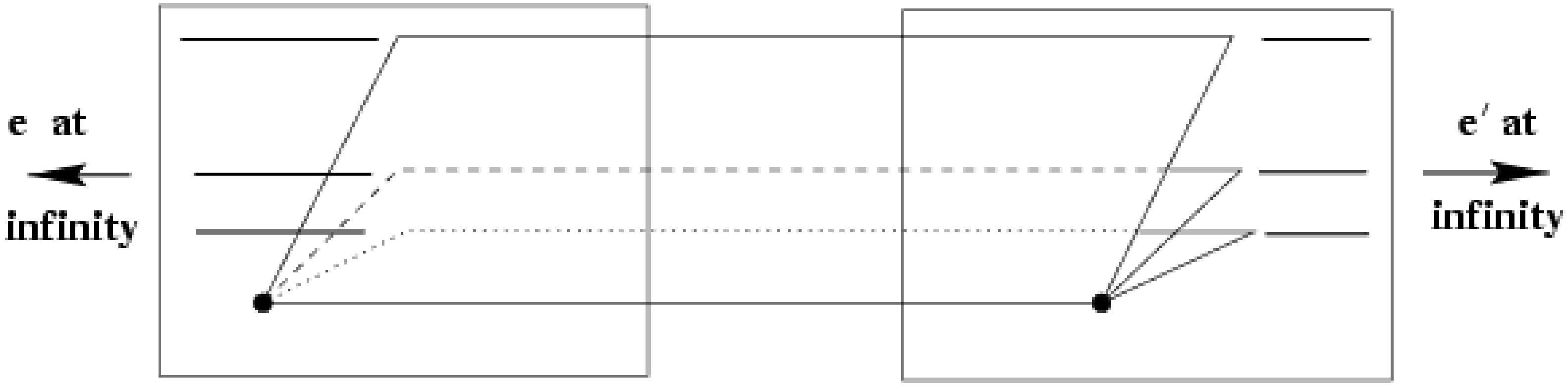
Knowing where the epipoles are gives us information about the relative pose of the cameras

$$\lambda \tilde{\mathbf{e}}_r = \mathbf{t}$$

$$\lambda \tilde{\mathbf{e}}_l = -\mathbf{R}^T \mathbf{t}$$

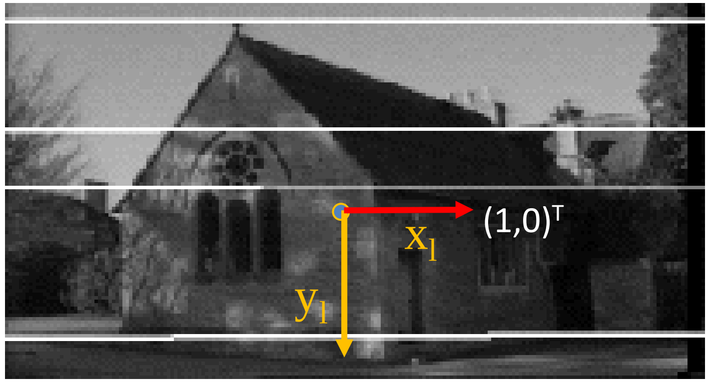
Epipole

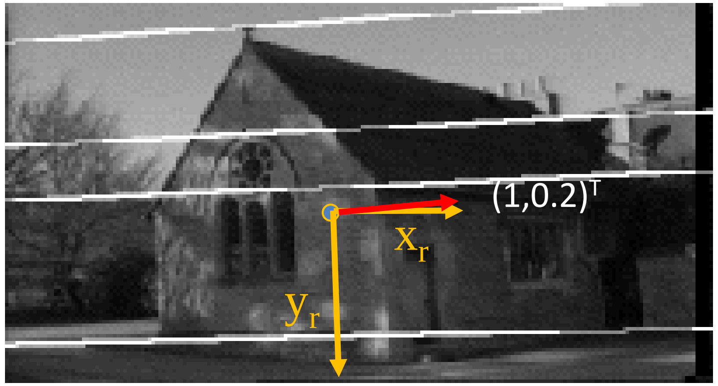
- If the two image planes are parallel to the line $\mathbf{C}_l$-$\mathbf{C}_r$ → the two epipoles are at infinity
- If, besides, the epipolar lines are at the same height, we have the ideal configuration



epipole at $\tilde{\mathbf{e}}_l = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

This means that the point is at infinity, i.e. it's a direction





epipole at $\tilde{\mathbf{e}}_r = \begin{bmatrix} 1 \\ 0.2 \\ 0 \end{bmatrix}$

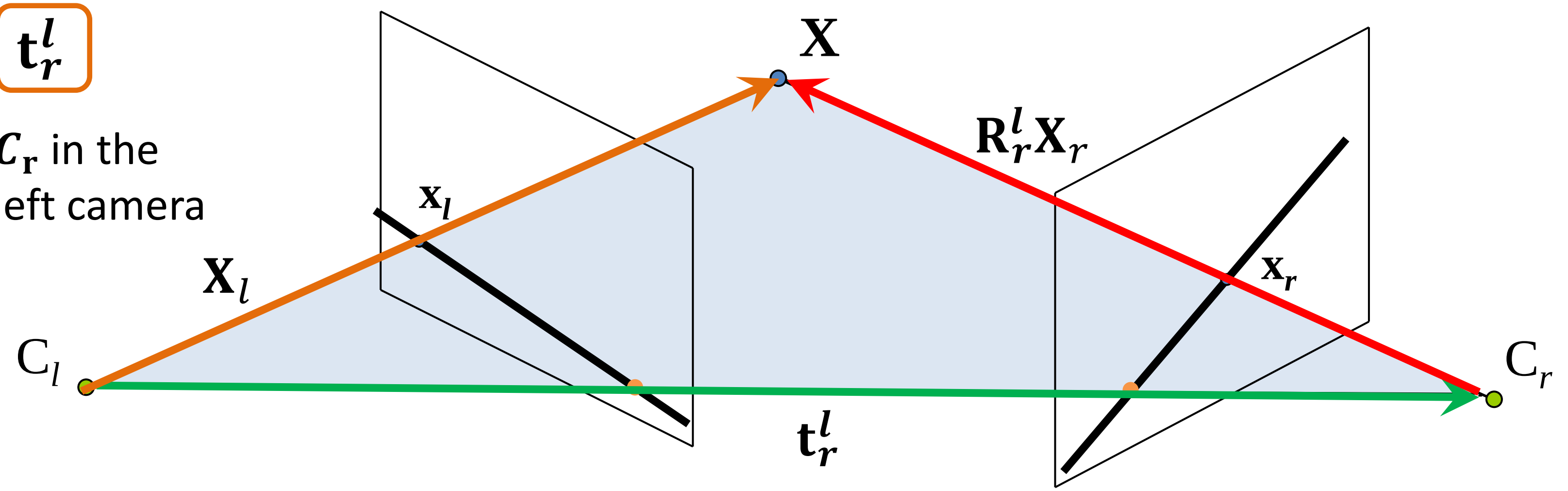
infinity

The epipolar constraint for **Calibrated** camera is given by the **Essential matrix**

$\mathbf{X_l} = \boxed{\mathbf{R}_r^l \mathbf{X_r}} + \boxed{\mathbf{t}_r^l}$

$\mathbf{X_r}$ in the left camera

$\boldsymbol{C}_\mathbf{r}$ in the left camera



Vectors $\mathbf{X_l}$, $\mathbf{t}_r^l$, and $\mathbf{R}_r^l \mathbf{X_r}$ are **coplanar** → their triple product is zero (a.[b × c]= 0)

$$\mathbf{X_l} \,.\, [\mathbf{t} \times (\mathbf{R}\mathbf{X_r})] = \mathbf{0} \longrightarrow \tilde{\mathbf{x}}_l^{\mathrm{T}} [\underbrace{\boxed{\mathbf{t} \times (\mathbf{R}}}_{[\mathbf{t}]_\times \mathbf{R}} \tilde{\mathbf{x}}_r)] = \mathbf{0}$$

$\mathbf{X_r} = \lambda_r \tilde{\mathbf{x}}_r$

$\mathbf{X_l} = \lambda_l \tilde{\mathbf{x}}_l$

$$\boxed{\tilde{\mathbf{x}}_l{}^{\mathbf{T}} \mathbf{E} \tilde{\mathbf{x}}_r = \mathbf{0} \quad \text{with } \mathbf{E} = [\mathbf{t}]_\times \mathbf{R}}$$

$$[\mathbf{t}]_\mathrm{x} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \quad \text{rank}([\mathbf{t}]_\mathrm{x})=2$$

Essential Matrix
(Longuet-Higgins, 1981)

Coordinates $\tilde{\mathbf{x}}_r, \tilde{\mathbf{x}}_l$ are in meters (measured in the sensor)! → The camera needs to be calibrated ($\mathbf{K}$ known) to obtain $\tilde{\mathbf{x}}_r, \tilde{\mathbf{x}}_l$

The epipolar constraint for **Uncalibrated** cameras is given by the **Fundamental matrix**

Essential Matrix: $\tilde{\mathbf{x}}_l{}^{\mathrm{T}}\mathbf{E}\tilde{\mathbf{x}}_r = \mathbf{0}$ with $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$

Calibrated coordinates (in meters) and uncalibrated pixels (in image coordinates) are related by the camera matrix $\mathbf{K}$:

$$\tilde{\mathbf{x}}_r{}' = \mathbf{K}\tilde{\mathbf{x}}_r$$
$$\tilde{\mathbf{x}}_l{}' = \mathbf{K}\tilde{\mathbf{x}}_l$$

assuming identical intrinsic parameters for the two cameras ($\mathbf{K_l} = \mathbf{K_r} = \mathbf{K}$)

$$\tilde{\mathbf{x}}_l{}^{\mathrm{T}}\mathbf{E}\tilde{\mathbf{x}}_r = \tilde{\mathbf{x}}_l{}^{\mathrm{T}}[\mathbf{t}]_\times\mathbf{R}\tilde{\mathbf{x}}_r = \left(\mathbf{K}^{-1}\,\tilde{\mathbf{x}}_l{}'\right)^{\mathrm{T}}\,[\mathbf{t}]_\times\mathbf{R}\mathbf{K}^{-1}\tilde{\mathbf{x}}_r' = \tilde{\mathbf{x}}_l{}'^{\mathrm{T}}\boxed{\mathbf{K}^{-\mathrm{T}}[\mathbf{t}]_\times\mathbf{R}\mathbf{K}^{-1}}\tilde{\mathbf{x}}_r' = \tilde{\mathbf{x}}_l{}'^{\mathrm{T}}\mathbf{F}\tilde{\mathbf{x}}_r' = 0$$

Fundamental Matrix: $\tilde{\mathbf{x}}_l{}'^{\mathrm{T}}\mathbf{F}\tilde{\mathbf{x}}_r' = 0$ with $\mathbf{F} = \mathbf{K}^{-\mathrm{T}}[\mathbf{t}]_\times\mathbf{R}\mathbf{K}^{-1} = \mathbf{K}^{-\mathrm{T}}\mathbf{E}\mathbf{K}^{-1}$

coordinates $\mathbf{x_l}', \mathbf{x_r}'$ are in pixels (measured in the image)

We can compute $\mathbf{F}$ from 8 pairs of points in both images (no need to know $\mathbf{K}$)

FUNDAMENTAL MATRIX F

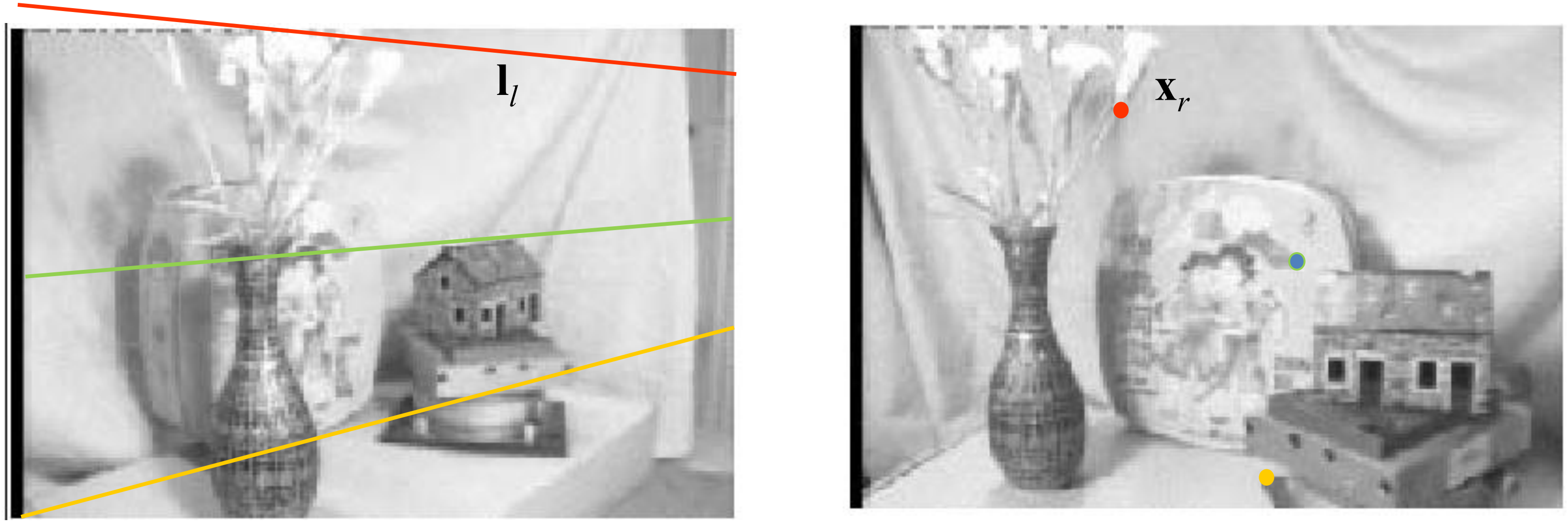
- 3x3 singular matrix (rank 2) that relates elements of both images:

Points to points: $\tilde{\mathbf{x}}_l{'}^{\mathbf{T}}\mathbf{F}\tilde{\mathbf{x}}_r' = 0$

Points and epipolar lines $\mathbf{l}_l' = \mathbf{F}\tilde{\mathbf{x}}_r'$ $\quad \mathbf{l}_r' = \mathbf{F}^T\tilde{\mathbf{x}}_l{'}$

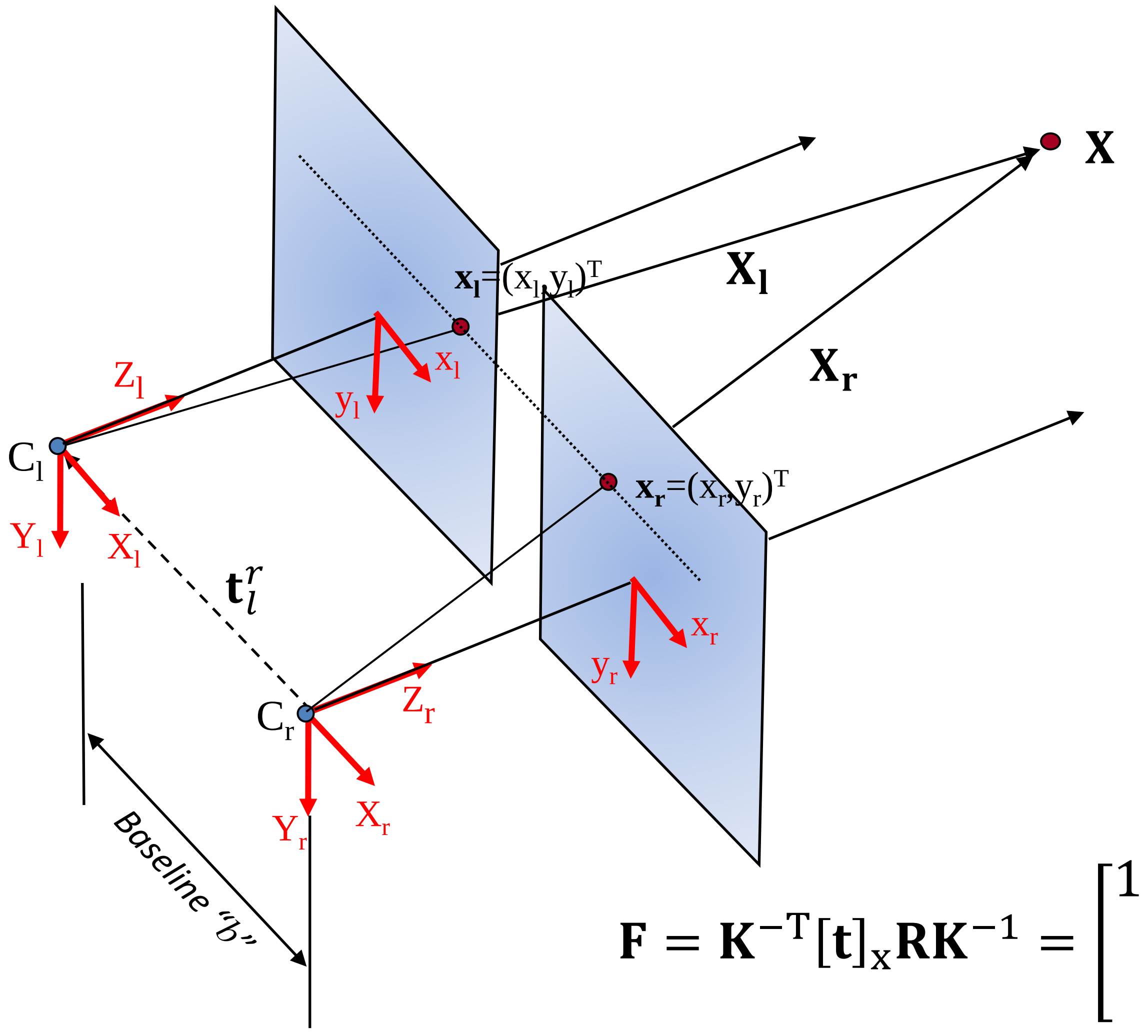
Epipoles: $\mathbf{F}\tilde{\mathbf{e}}_r' = \mathbf{0}$ $\quad \mathbf{F}^T\tilde{\mathbf{e}}_l = \mathbf{0}$

Epipolar lines of the points in the right image



- $\mathbf{F}$ has 7 d.o.f. due to the ambiguity of scale and to its rank= 2 (det($\mathbf{F}$)=0)
- $\mathbf{F}$ can be inferred from point matches (**eight-point algorithm**)

Example: Ideal configuration



$$\lambda\tilde{\mathbf{x}}_l{}' = \mathbf{P}_l\mathbf{X}_l = \mathbf{K}_l[\mathbf{I}|\mathbf{0}]\mathbf{X}_l = \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{X}_l$$

$$\lambda\tilde{\mathbf{x}}_l{}' = \mathbf{P}_r\mathbf{X}_r = \mathbf{K}_r[\mathbf{R}_l^r|\mathbf{t}_l^r]\mathbf{X}_l = \mathbf{K}[\mathbf{I}|\mathbf{t}]\mathbf{X}_l$$

Prime vectors mean coordinates in pixel!
Vectors in homogeneous coordinates!

$$\mathbf{K} = \mathbf{K}_l = \mathbf{K}_r = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{t}_l^r = \begin{bmatrix} -b \\ 0 \\ 0 \end{bmatrix}$$

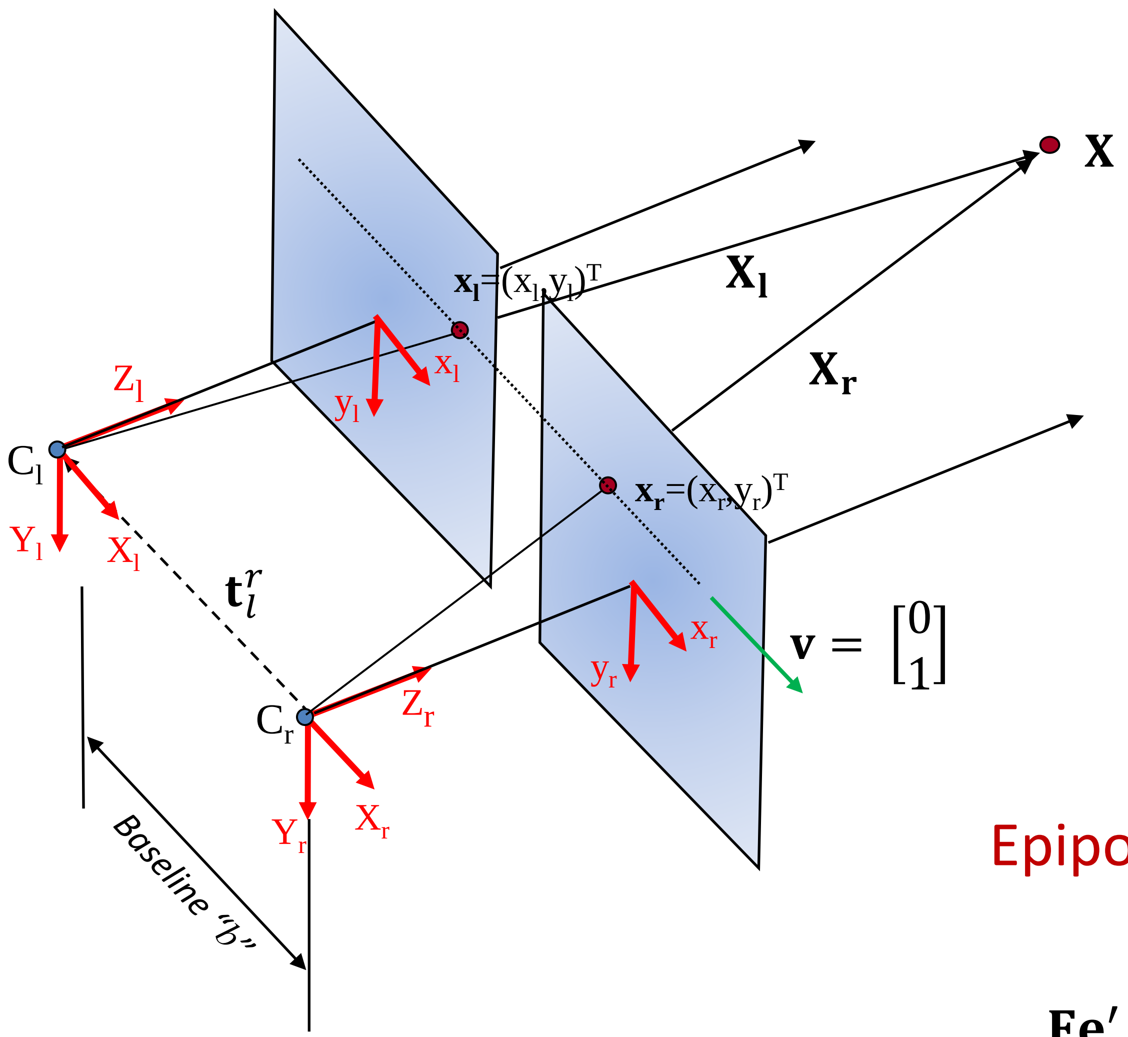
Notice, according to this K the image coordinate system has the origin in image cernter, not the upper right corner!

$$[\mathbf{t}]_x = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix}$$

$$\mathbf{F} = \mathbf{K}^{-T}[\mathbf{t}]_x\mathbf{R}\mathbf{K}^{-1} = \begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b \\ 0 & -b & 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1/f & 0 & 0 \\ 0 & 1/f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & b/f \\ 0 & -b/f & 0 \end{bmatrix} = b/f\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Example: Ideal configuration



$$\mathbf{F} = \mathbf{K}^{-\mathrm{T}}[\mathbf{t}]_{\mathrm{x}}\mathbf{R}\mathbf{K}^{-1} = b/f \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Epipolar line for $[x_r, y_r]^T$

Direction vector

$$\mathbf{l}'_l = \mathbf{F}\tilde{\mathbf{x}}'_r = b/f \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ 1 \end{bmatrix} = b/f \begin{bmatrix} \boxed{\begin{matrix} 0 \\ 1 \end{matrix}} \\ \boxed{-y_r} \end{bmatrix}$$

Distance to the origin

Line equation: $\mathrm{A}x + \mathrm{B}y + C = [A \quad B \quad C] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = l^T \tilde{p} = 0$

Epipolar line: $[0 \quad 1 \quad -y_r] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0 \Rightarrow y = y_r$

Epipoles:

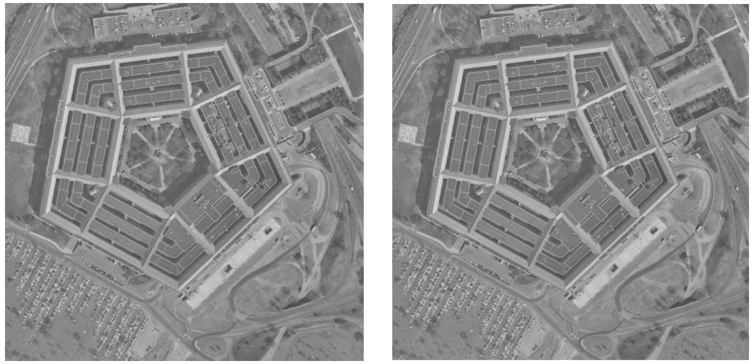
$$\mathbf{F}\mathbf{e}'_r = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} e_{xr} \\ e_{yr} \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ w \\ -e_{yr} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \mathbf{e}'_r = \begin{bmatrix} e_{xr} \\ 0 \\ 0 \end{bmatrix}$$

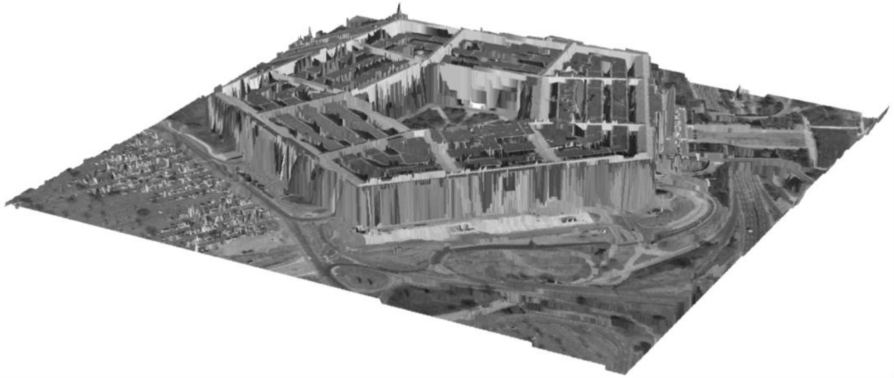
Epipole at infinity along the axis x

5. Reconstruction from stereo

Dense stereo → All the pixels are tried to be matched (**no need of feature detection**)

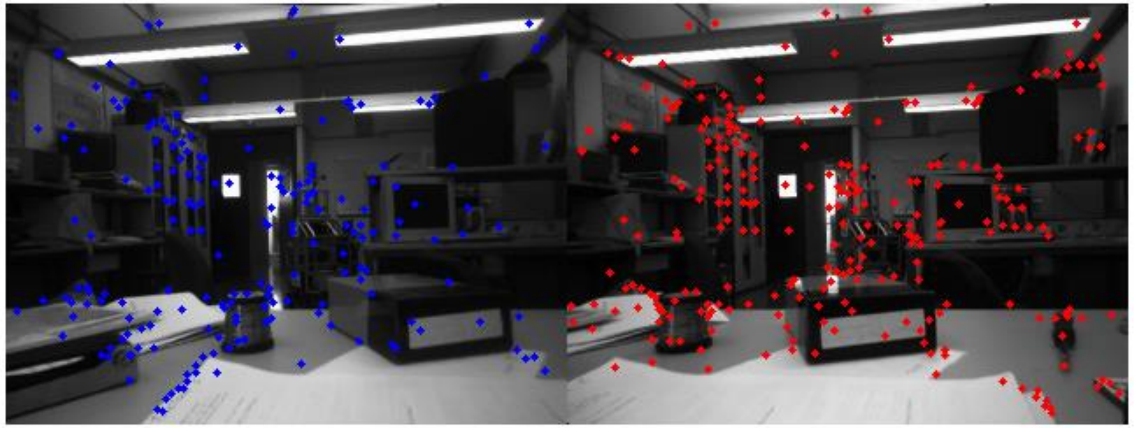
→ Provides a **dense depth map**

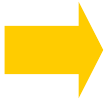




Based on distinctive features (sparse stereo) → keypoints, segments, regions are detected and put in correspondence

→ sparse depth map



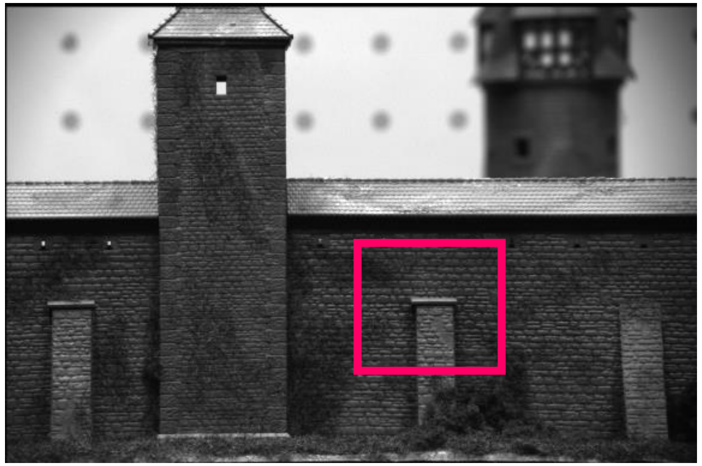


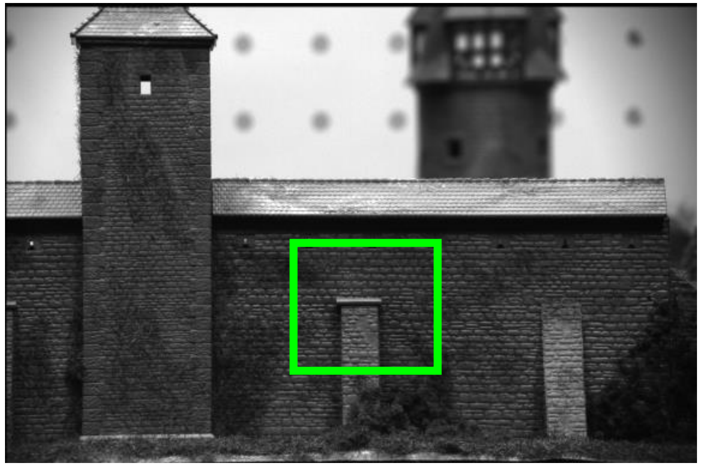
Depth available only at these features (points in this case)

5. Reconstruction from stereo

We need a similarity measure to find good matches between the images

Patch around the candidate correspondence points





Criterium 1: Minimize the **Sum of squared differences** (*SSD*):

$$SSD = \sum_{[i,j]\in R} (f(i,j) - g(i,j))^2 = \sum_{[i,j]\in R} f^2 + \sum_{[i,j]\in R} g^2 - 2 \sum_{[i,j]\in R} fg$$

Criterium 2: Maximize the **Cross-correlation:** $C_{fg} = \sum_{[i,j]\in R} f(i,j)g(i,j)$

Cross-correlation can be made invariant to contrast and brightness : **NCC**

Normalized cross correlation (NCC) between two images I_r and I_l:

$$NCC(I_l, I_r) = \frac{1}{K}\sum(I_l - \bar{I_l})(I_r - \bar{I_r})$$

NCC along a row of the right image I_r for different disparities τ :

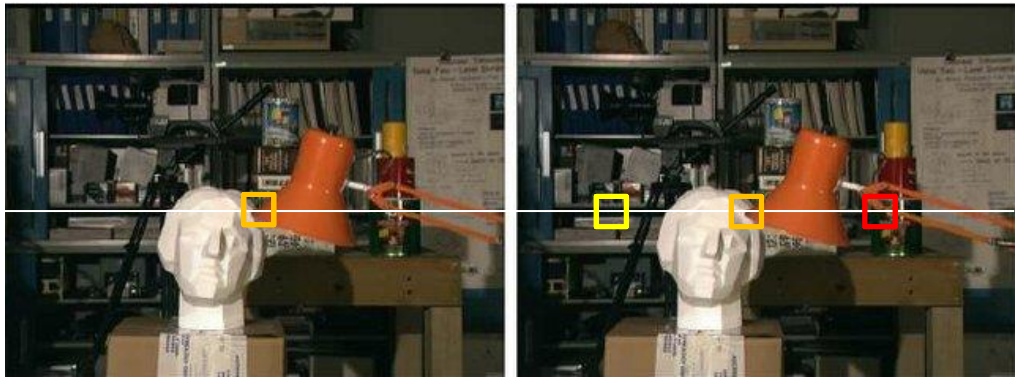
$$NCC(d) = \frac{1}{K}\sum_{u_1=-N}^{N}\sum_{v_1=-P}^{P}\underbrace{\left(I_l(u_1+u_0, v_1+v_0) - \bar{I}_l(u_0,v_0)\right)}_{\text{Left image window}}\underbrace{\left(I_r(u_1+u_0+d, v_1+v_0) - \bar{I}_r(u_0+d,v_0)\right)}_{\text{Right image window moved } \tau \text{ to the right}}$$

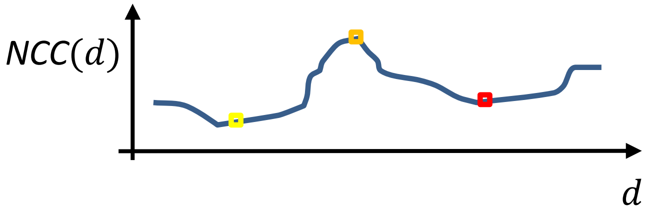
disparity

$$K = (2P+1)(2N+1)\sigma_l(u_0,v_0)\sigma_r(u_0+d,v_0)$$

Computed just once

Computed at each position on the sliding window



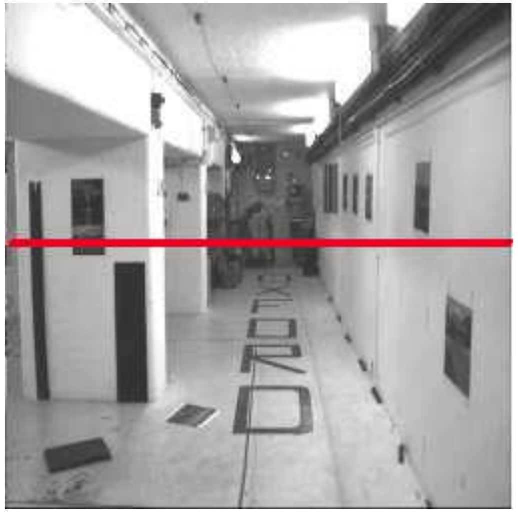


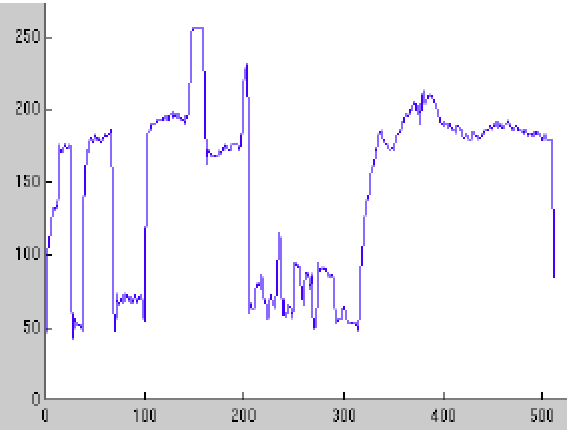
Mean: $\bar{I}_l(u_0,v_0) = \dfrac{1}{(2P+1)(2N+1)}\displaystyle\sum_{u_1=-N}^{N}\sum_{v_1=-P}^{P} I_l(u_1+u_0, v_1+v_0)$

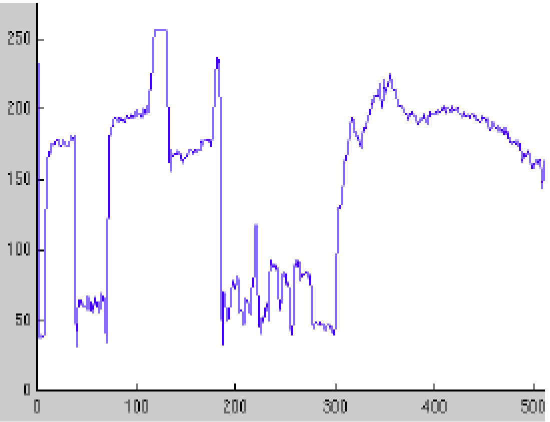
Variance: $\sigma^2{}_l(u_0,v_0) = \frac{1}{(2P+1)(2N+1)}\sum_{u_1=-N}^{N}\sum_{v_1=-P}^{P}\left(I_l(u_1+u_0, v_1+v_0) - \bar{I}_l(u_0,v_0)\right)^2$

Correlation

Small baseline between cameras→ intensity profiles/correlation windows very similar



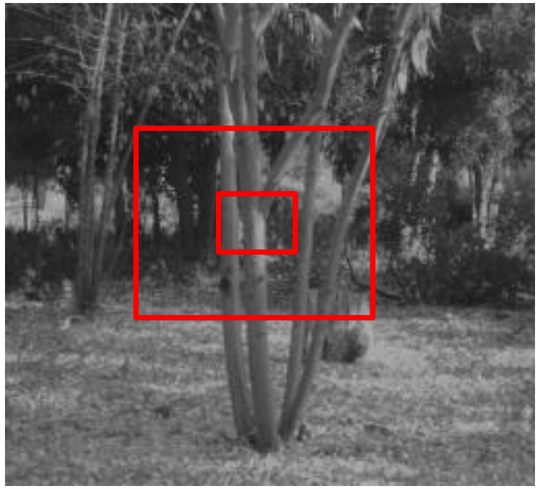




For a good match, the disparities of the pixels in the window must be the same

Cortesy of D. Murray (Oxford University)

Correlation Effect of the window size w on the NCC

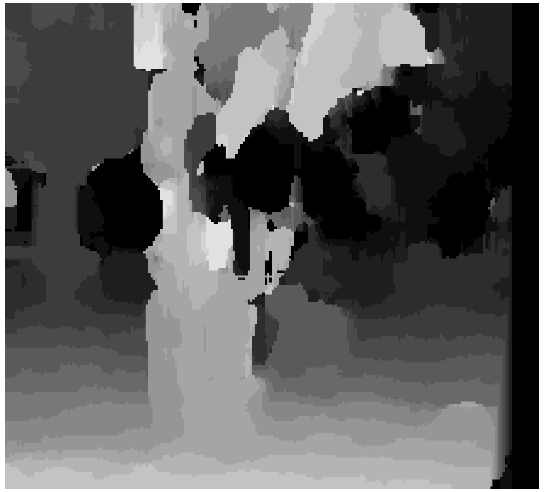


W = 3

Small window

- Detailed 3D map
- but noisy

W = 20

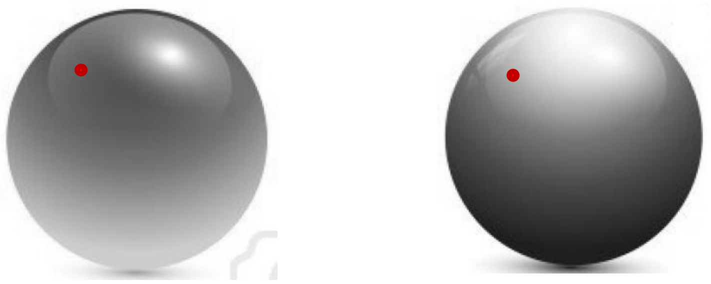


Large window

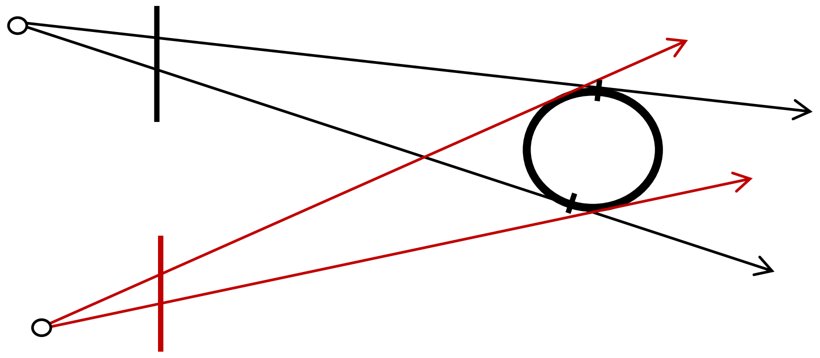
- Less noisy
- Corse matches because different disparities of the pixels in the window
- Close pixels give similar disparities → Coarser 3D map

Limitations of Correlation

The image of a 3D point may have different intensity because of different view-point, then NCC does not work well



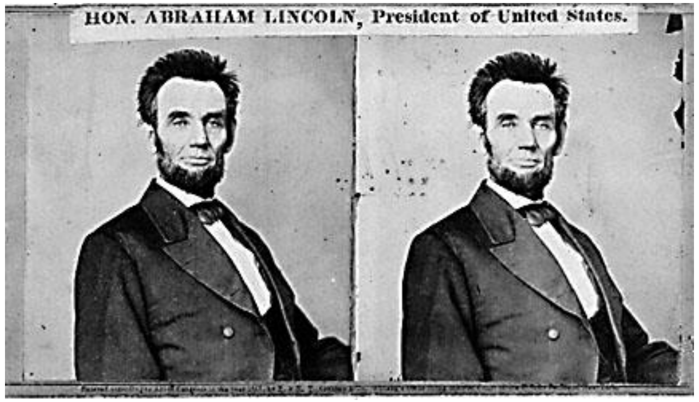
Wrong correspondences even if NCC = 1 : Same projections but corresponding to different 3D points



For example, a circular column

Correlation alone may be not enough to select the correct match

Two examples:



Texture-less surfaces

Occlusion, repetition

We need:

To apply constraints (uniqueness, continuity, ordering)

Some context information: Independent matches, without taking into account other candidate matches, do not give good results

Cortesy of D. Murray (Oxford University)

A global optimization approach for correspondence

Idea: solve correspondence not only using NCC for the keypoints but all the pixels and applying the contraints

- Build a graph with all the posible matches between keypoints in epipolar lines of both images
- Find a minimum cost path from the upper left corner to the botton right one

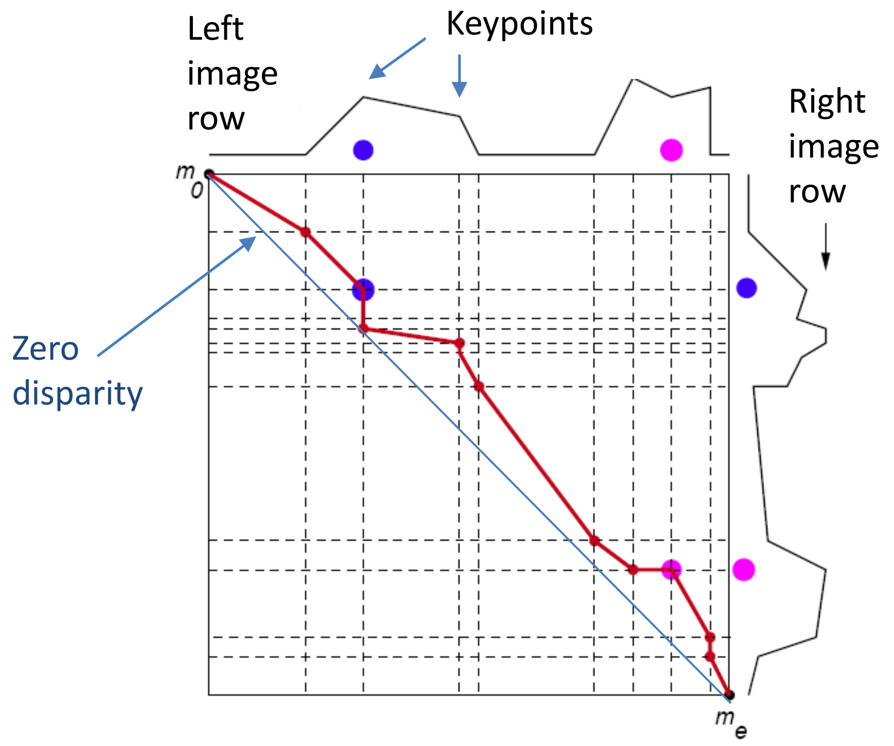
Cost associated to arcs and nodes:

$$C(m_0, m_e) = \sum_{i=0}^{e} f(m_i) + \sum_{i=0}^{e-1} g(m_i, m_{i+1})$$

Cost of the node m_i given by the NCC

Arc cost

The optimal path is computed by a Dijkstra's algorithm

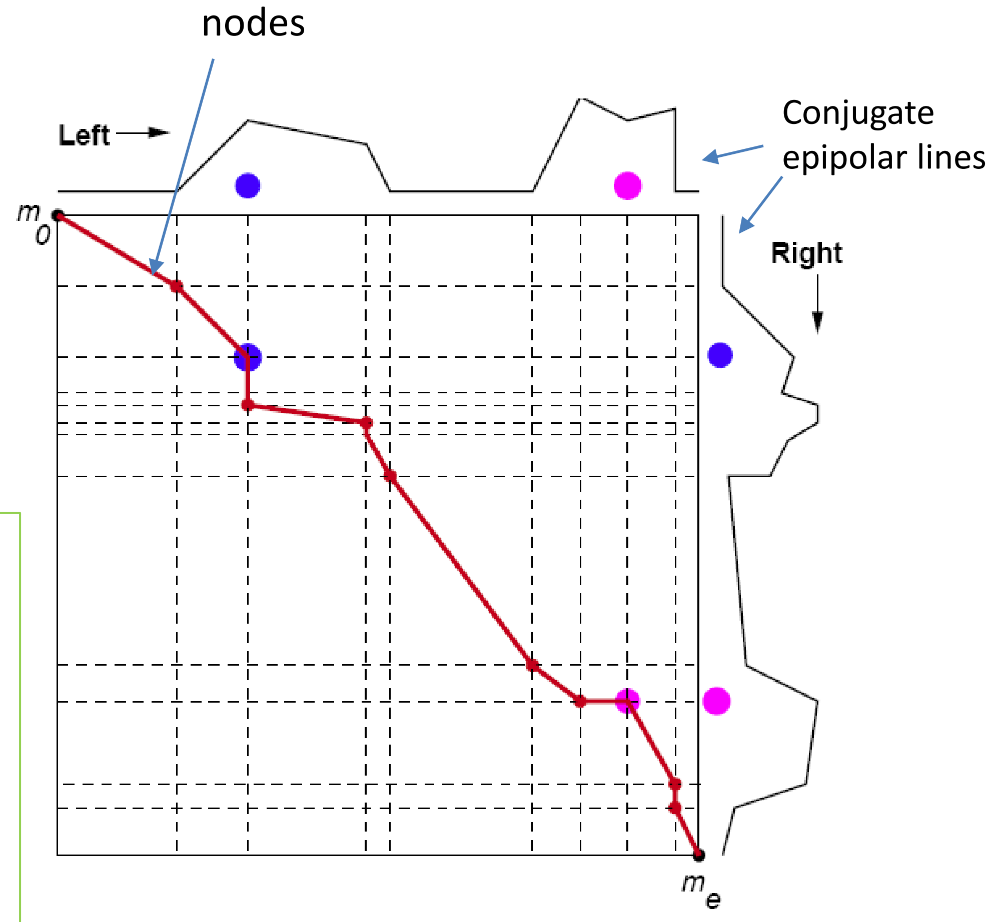


Courtesy of D. Murray (Oxford University)

A global optimization approach for correspondence:

Nodes:

- Nodes arise from the intersection of distintive image keypoints along conjugates epipolar lines
- Some matches are more likely than others because they entail lower cost (i.e. higher NCC)



Arcs: Not meeting the constraint increases the cost of the arcs

- Ordering: arcs in the direction south-east
- Uniqueness: neither horizontal nor vertical arcs allowed
- Completeness: the more nodes visited, the better
- Figural continuity: any path must be similar to the neighbor epipolar lines (above and below)

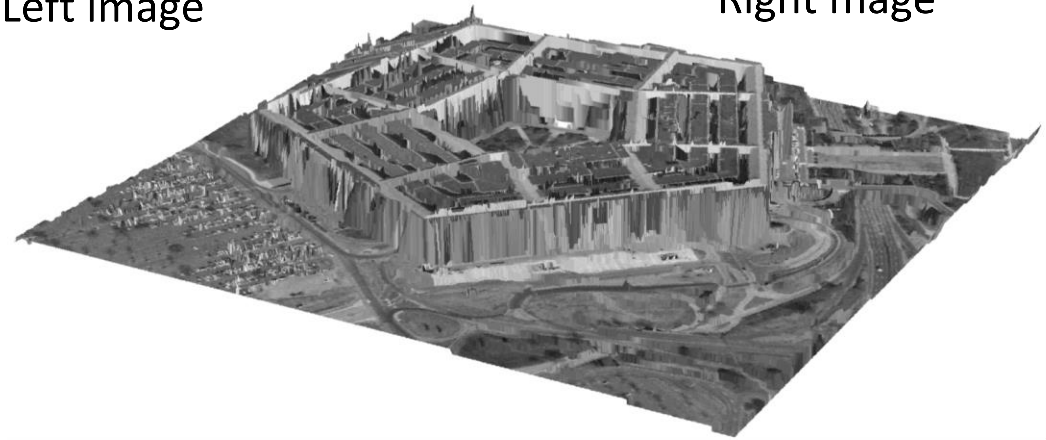
Cortesy of D. Murray (Oxford University)

A global optimization approach for correspondence:

Example: Correlation + Dynamic programming

Left image

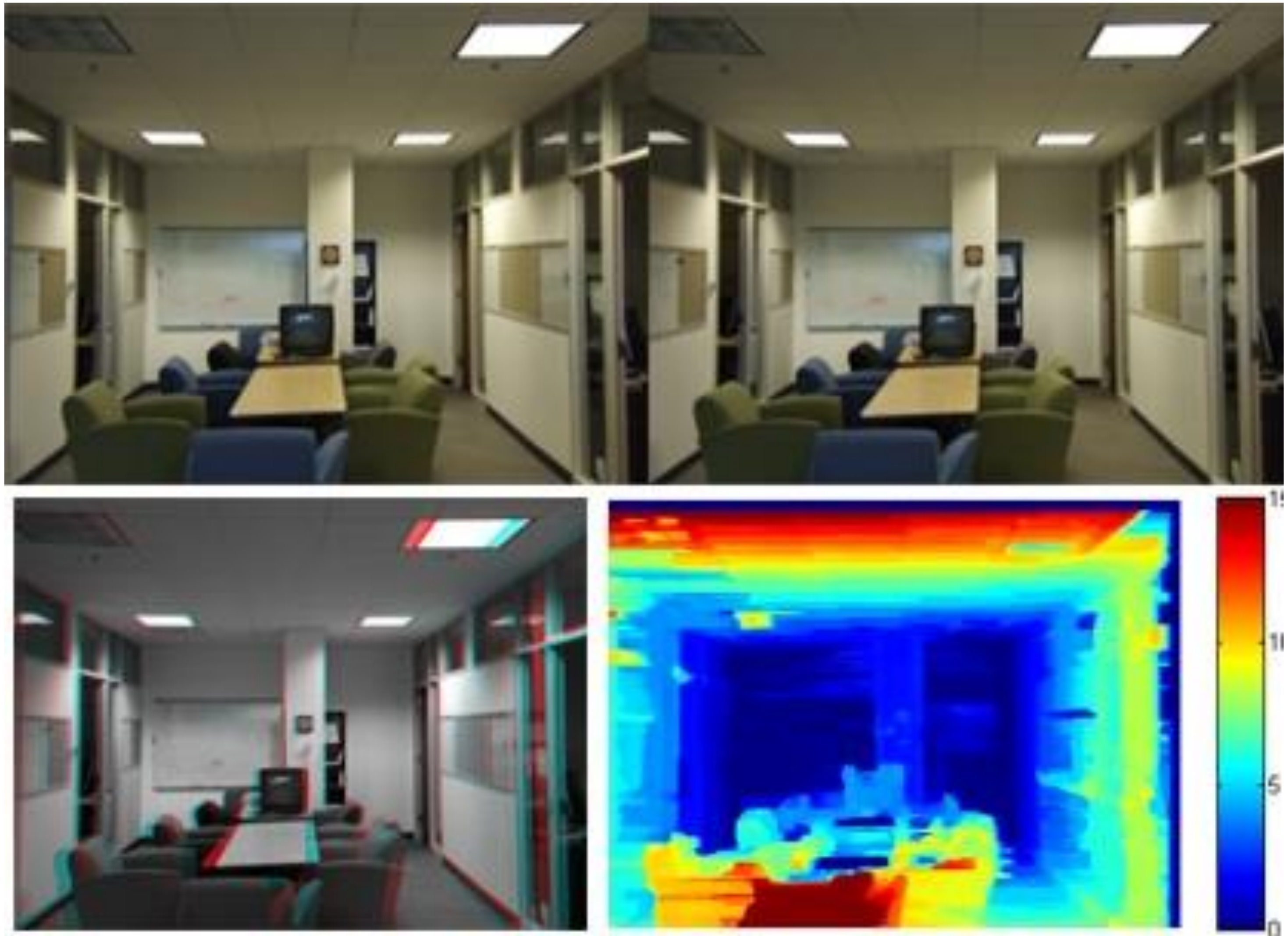
Right mage



For the latest and greatest: http://vision.middlebury.edu/stereo/

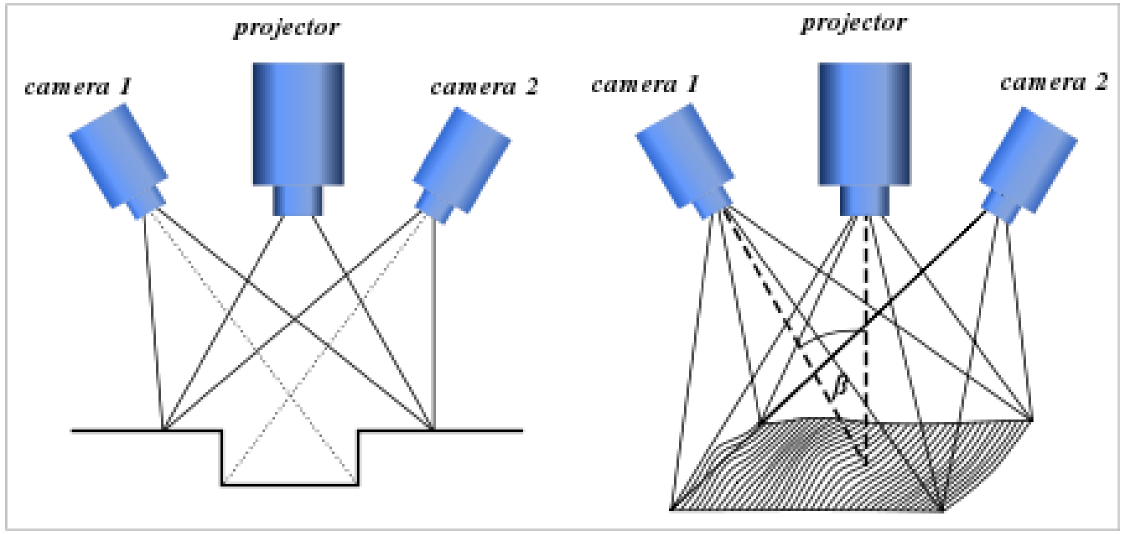
A global optimization approach for correspondence:

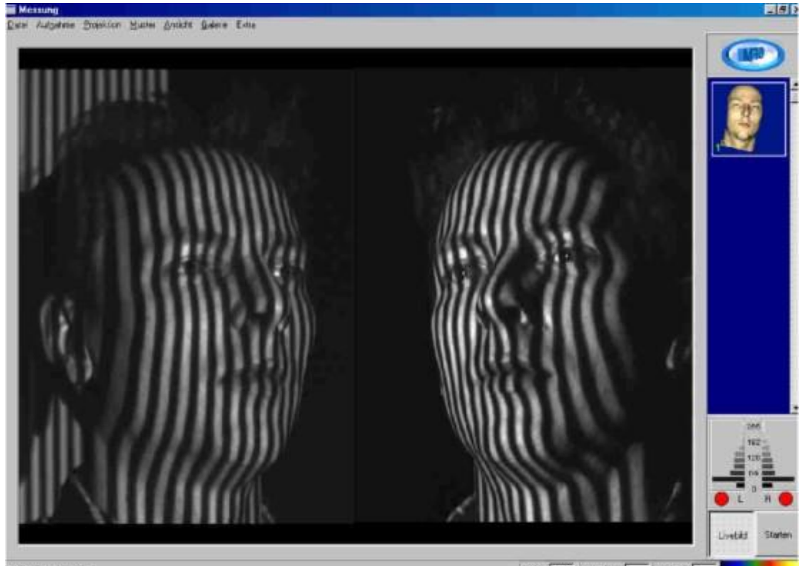
Example:



5. Reconstruction from stereo

A solution for texture-less zones (more invasive, used in CAD)



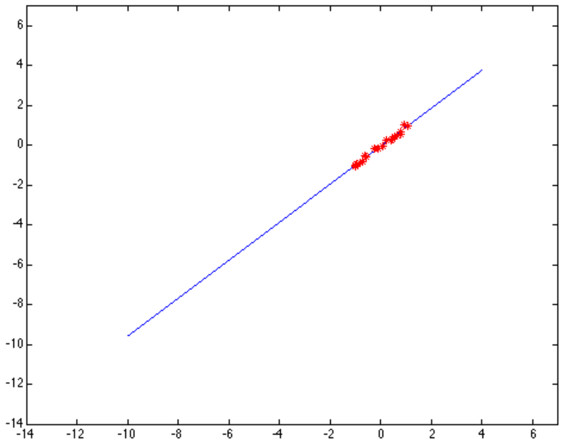


Appendix: RANSAC algorithm

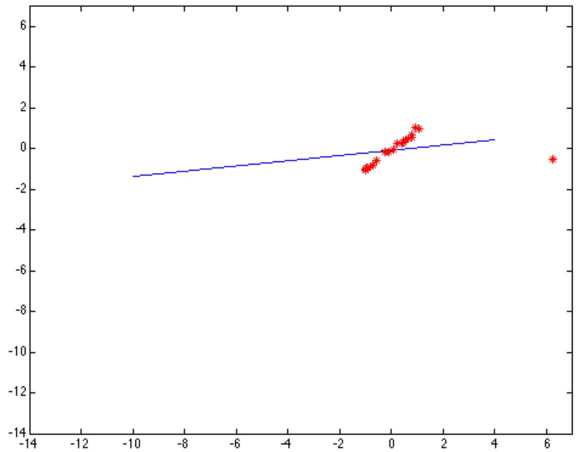
RANSAC: **Ran**dom **Sa**mple **C**onsensus

Iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers.

EXAMPLE: Least squares fit to the red points:



No outliers



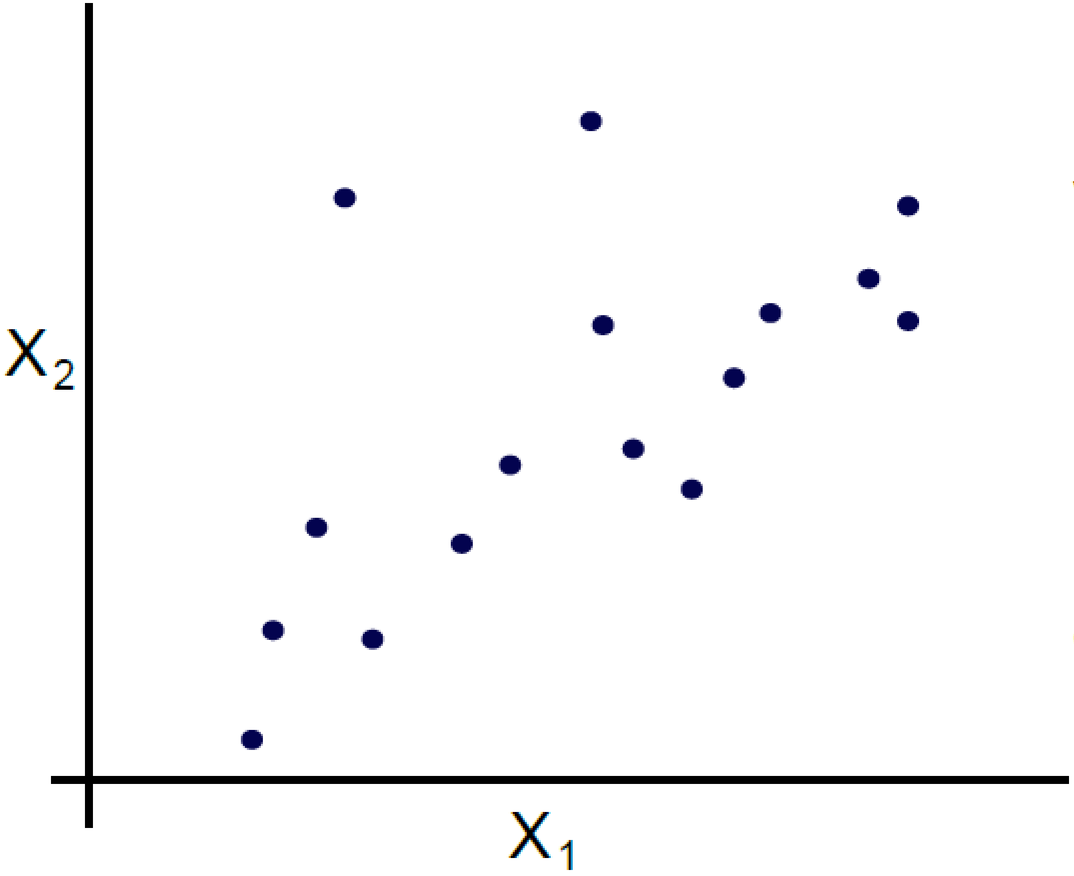
One outlier

Problem: squared error heavily penalizes outliers

Appendix: RANSAC algorithm

$X=\{x_i\}$: 2D Data points
$\theta = \{\rho, \alpha\}$: Line parameters

initialise count of number of points fit

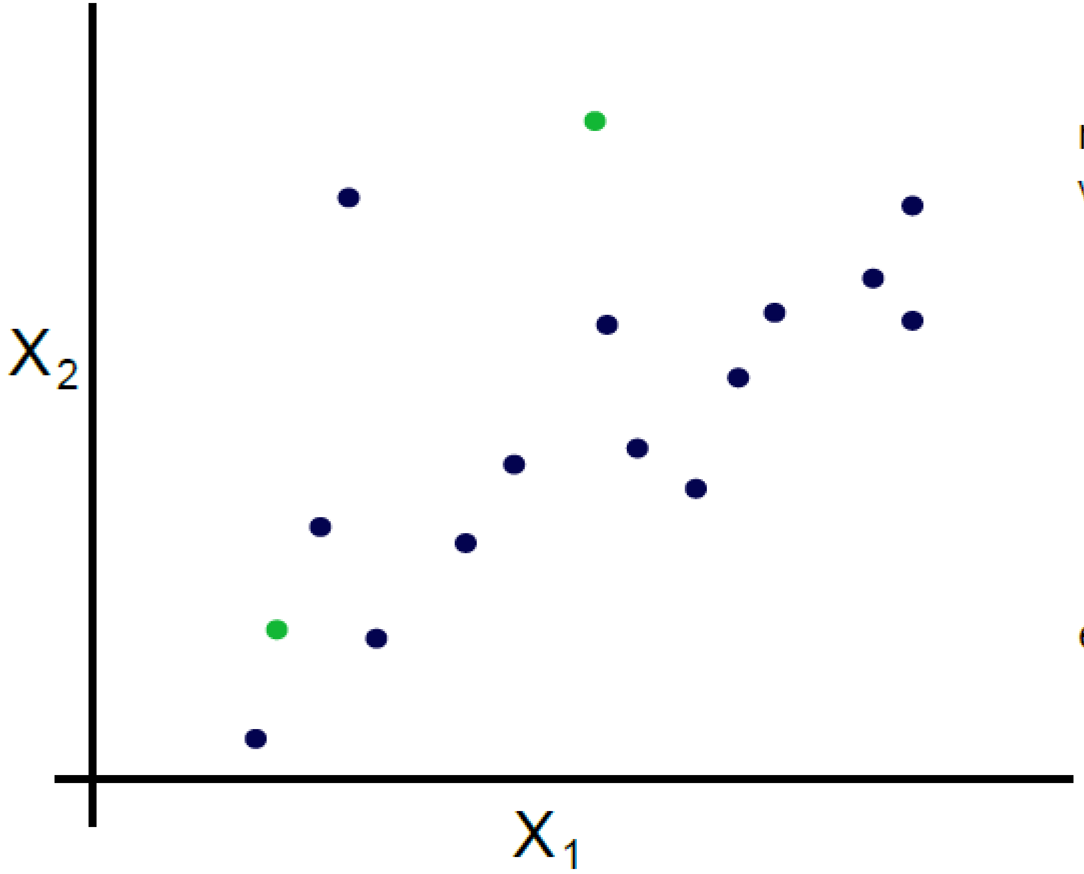



```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
       θ = θ_i
       nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

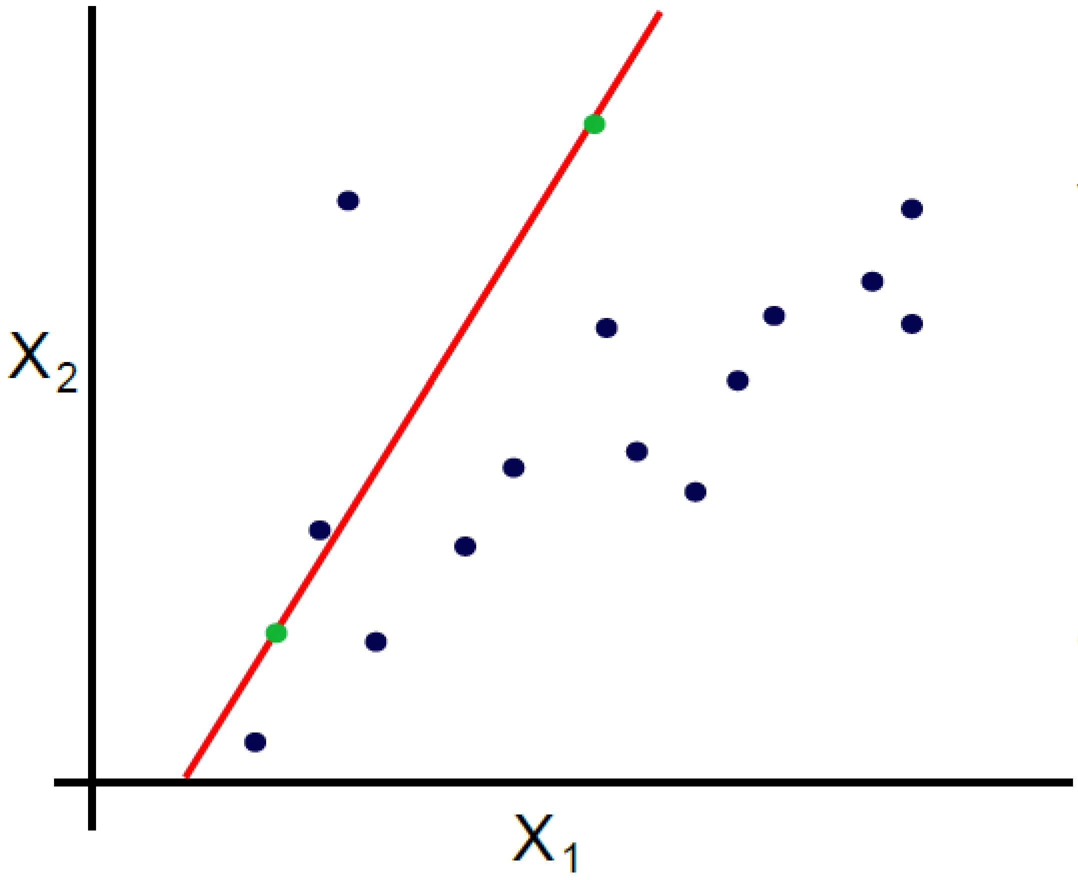
pick a subset of K points randomly (here K=2)




```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
        θ = θ_i
        nBest = nInliers
   endif
endwhile
```

Appendix: RANSAC algorithm

fit parameters to subset S

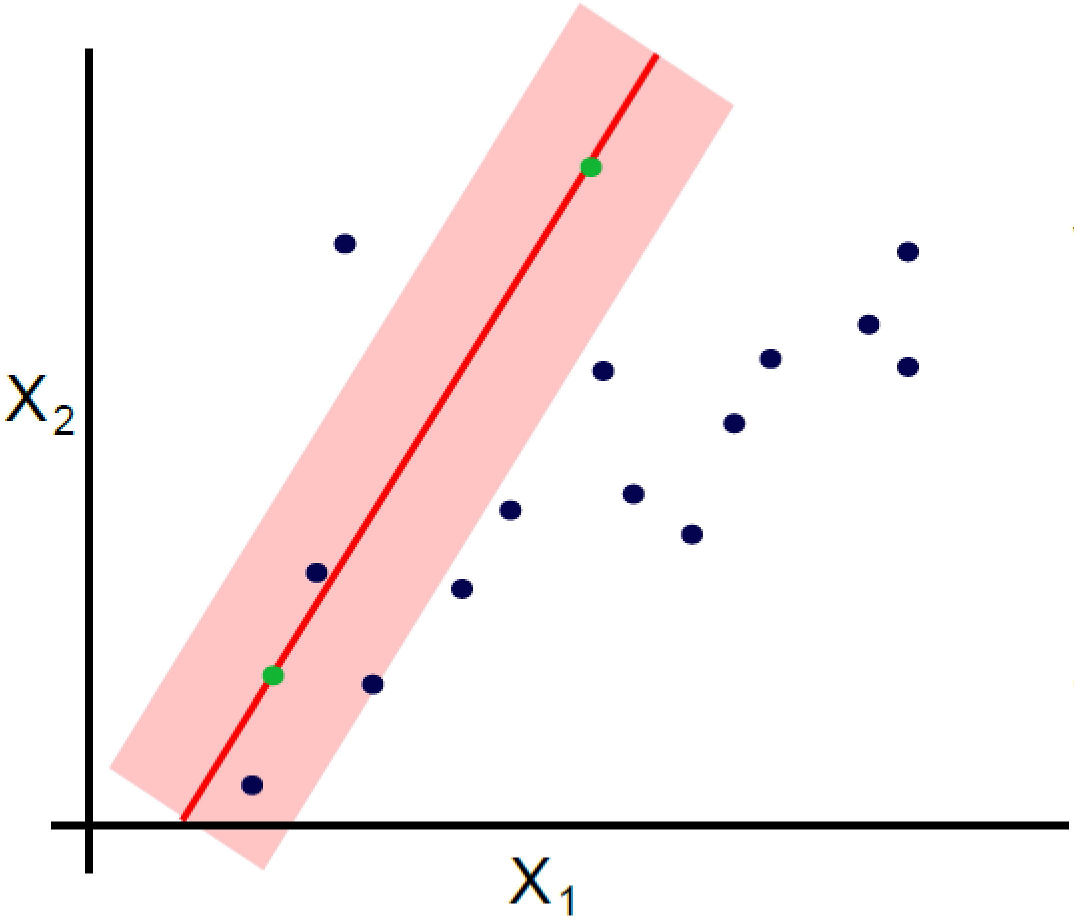



```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
       θ = θ_i
       nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

count number of points which are inliers

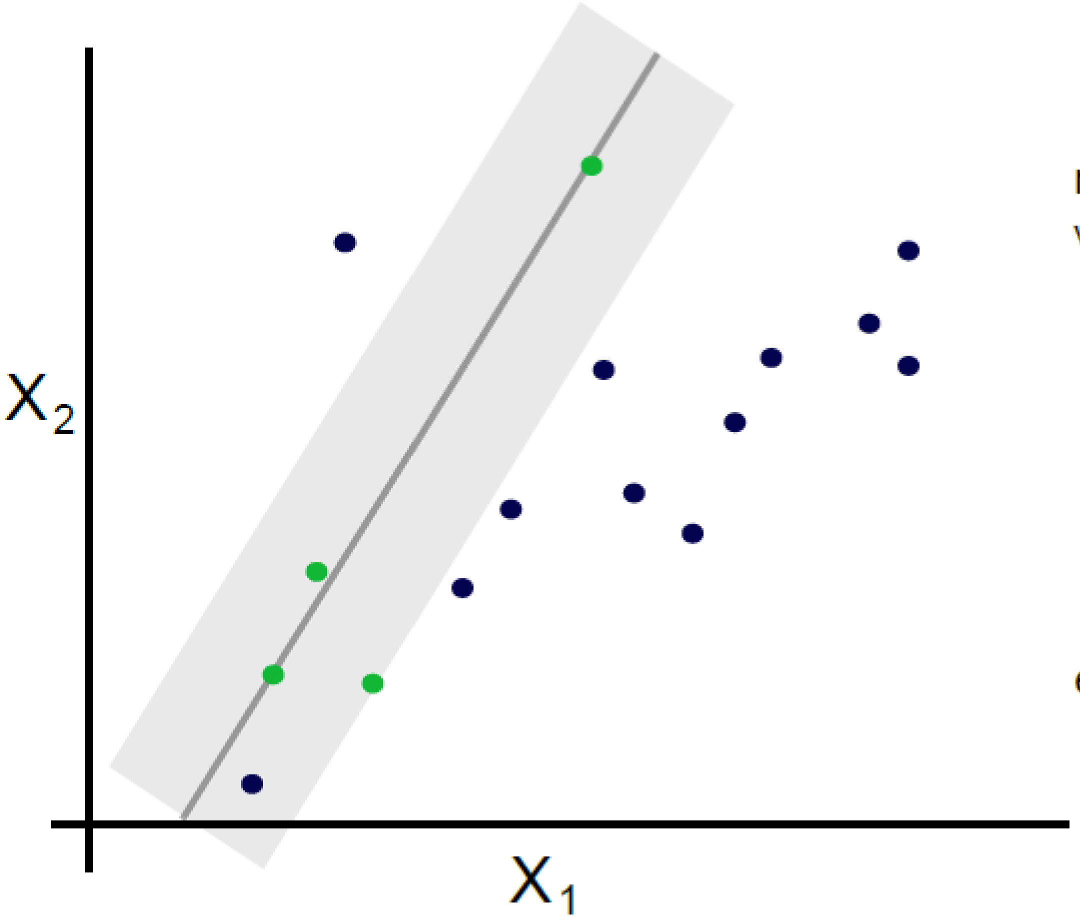



```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
      θ = θ_i
      nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

if number of inliers is larger than before, save parameters & nInliers

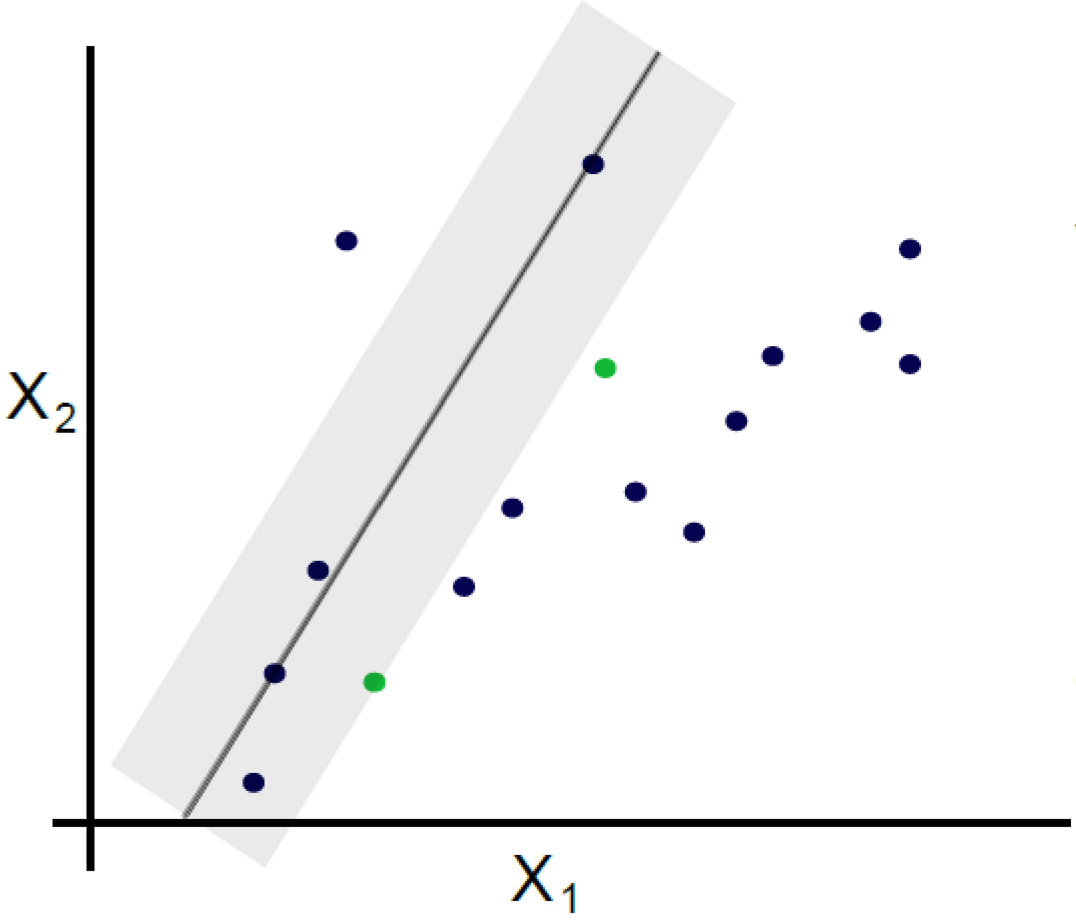



```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
      θ = θ_i
      nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

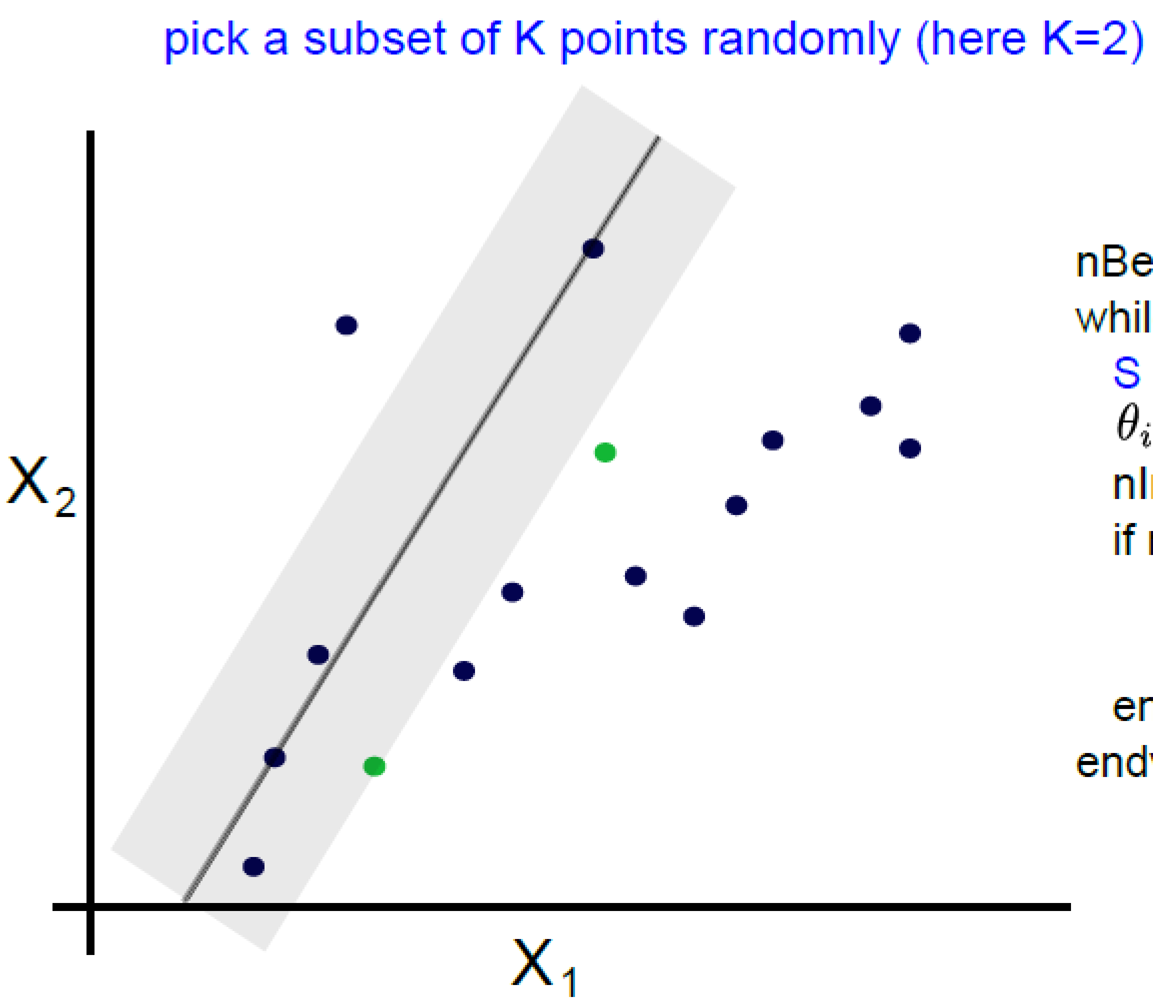
pick a subset of K points randomly (here K=2)



nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X, θ_i)
   if nInliers>nBest
      $\theta = \theta_i$
      nBest = nInliers
   endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

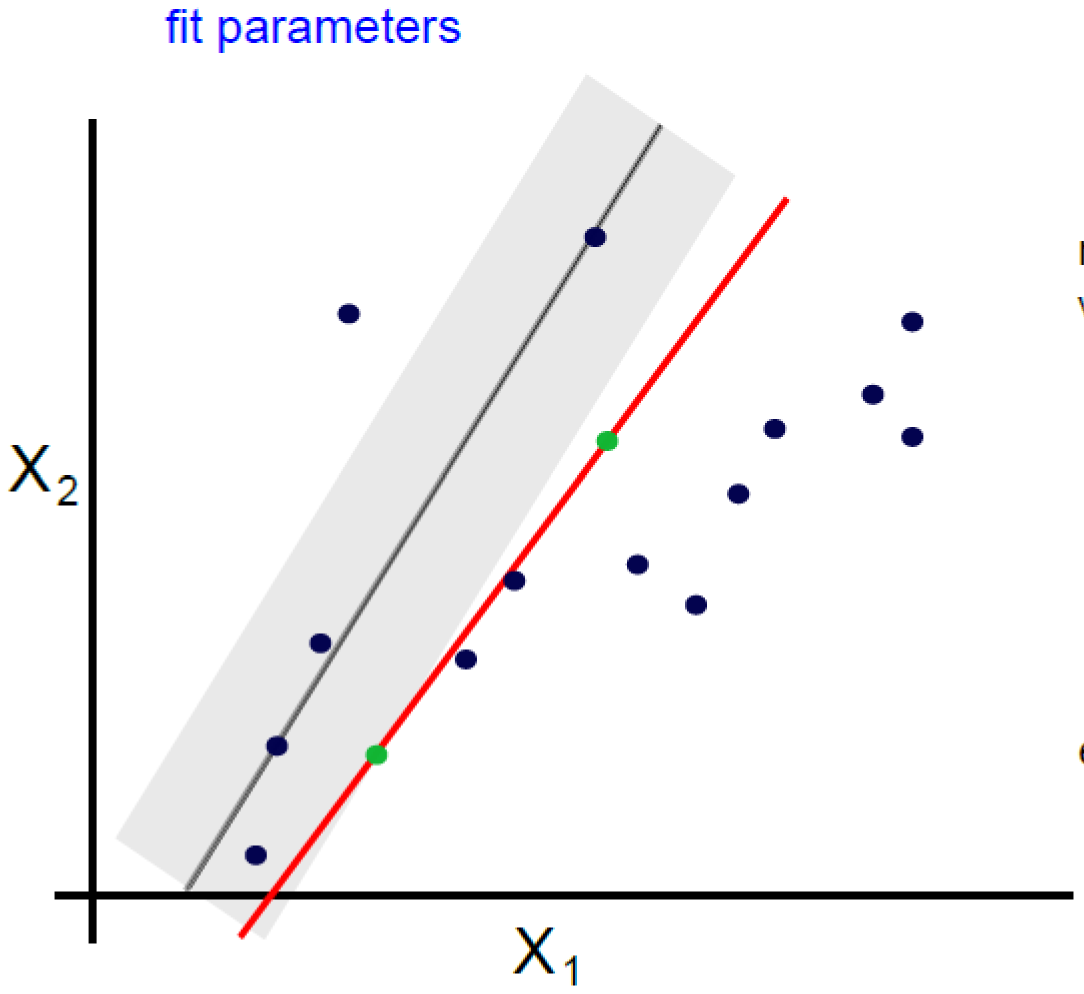
Appendix: RANSAC algorithm




```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
      θ = θ_i
      nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

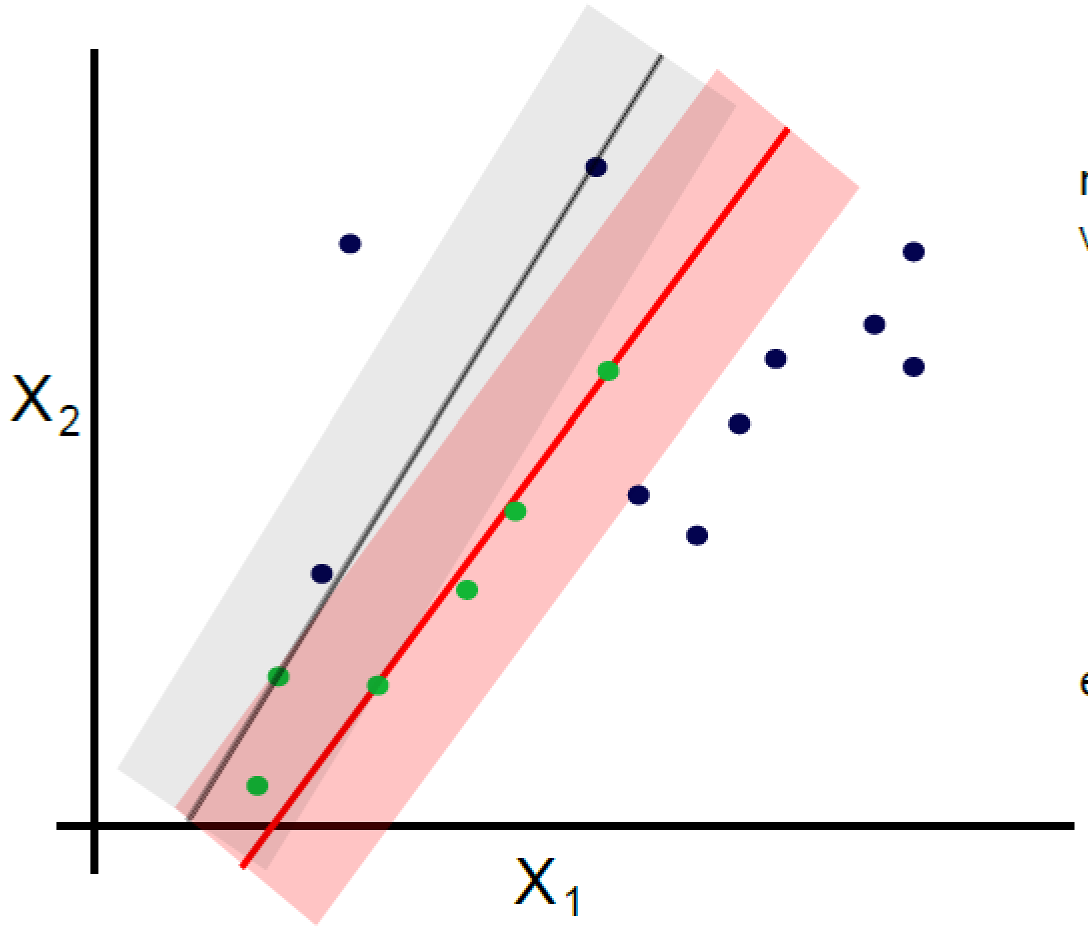



```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
      θ = θ_i
      nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

count number if inliers, here nInliers = 6

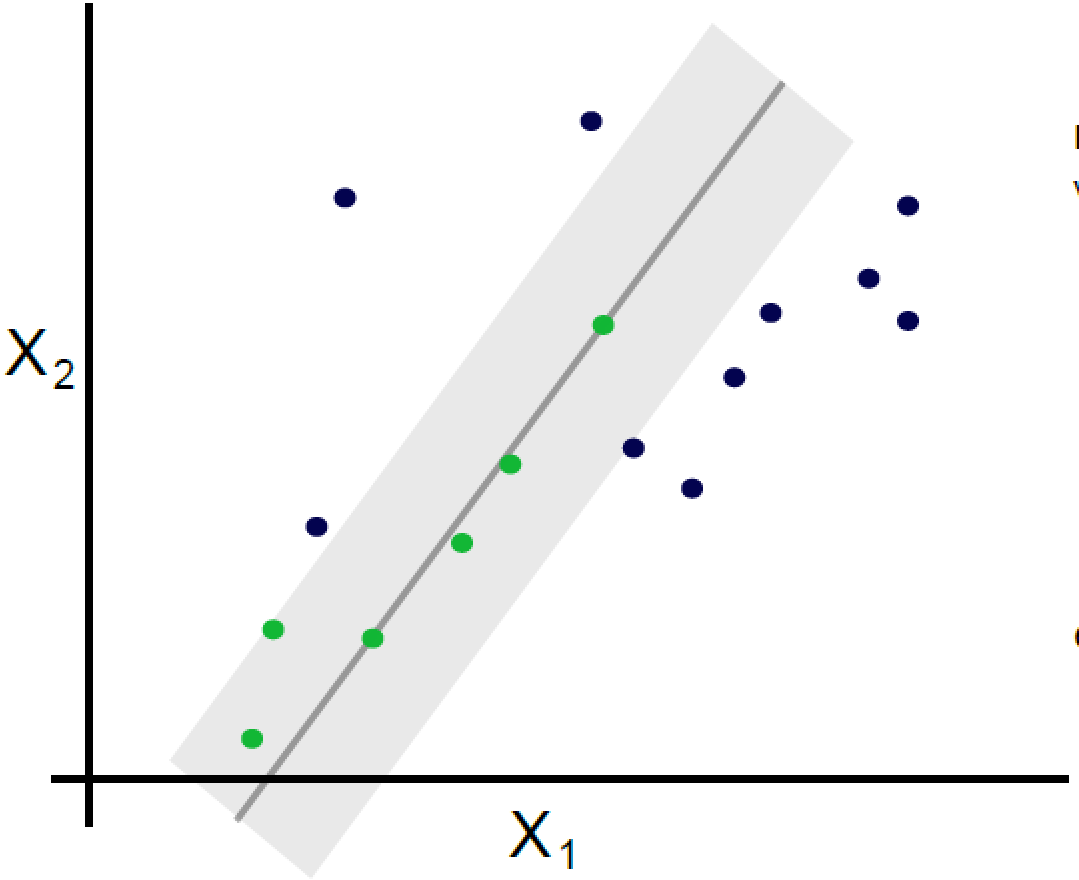



```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
        θ = θ_i
        nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

if number of inliers larger than best, update parameters etc.

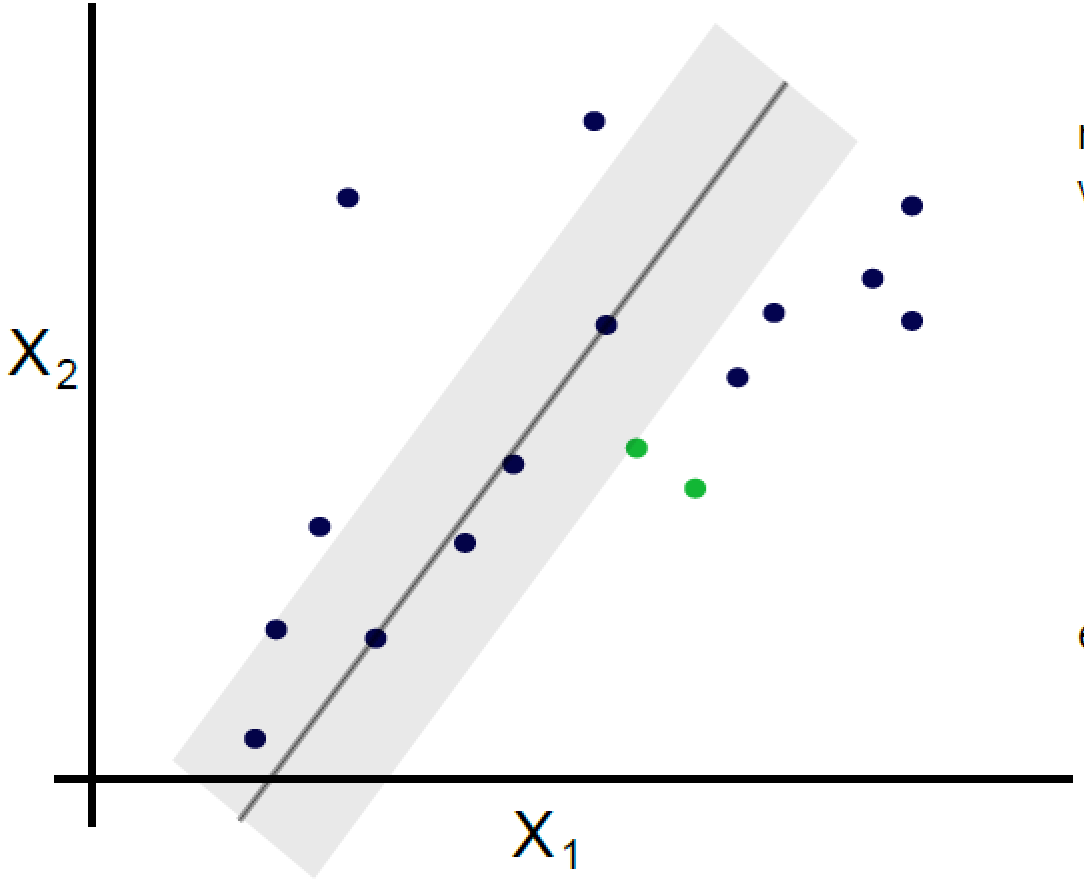


nBest = 0
while i<nIterations
  S = selectRandSubset(X)
  θ_i = fitParams(S)
  nInliers = countInliers(X,θ_i)
  if nInliers>nBest
    $\theta = \theta_i$
    nBest = nInliers
  endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

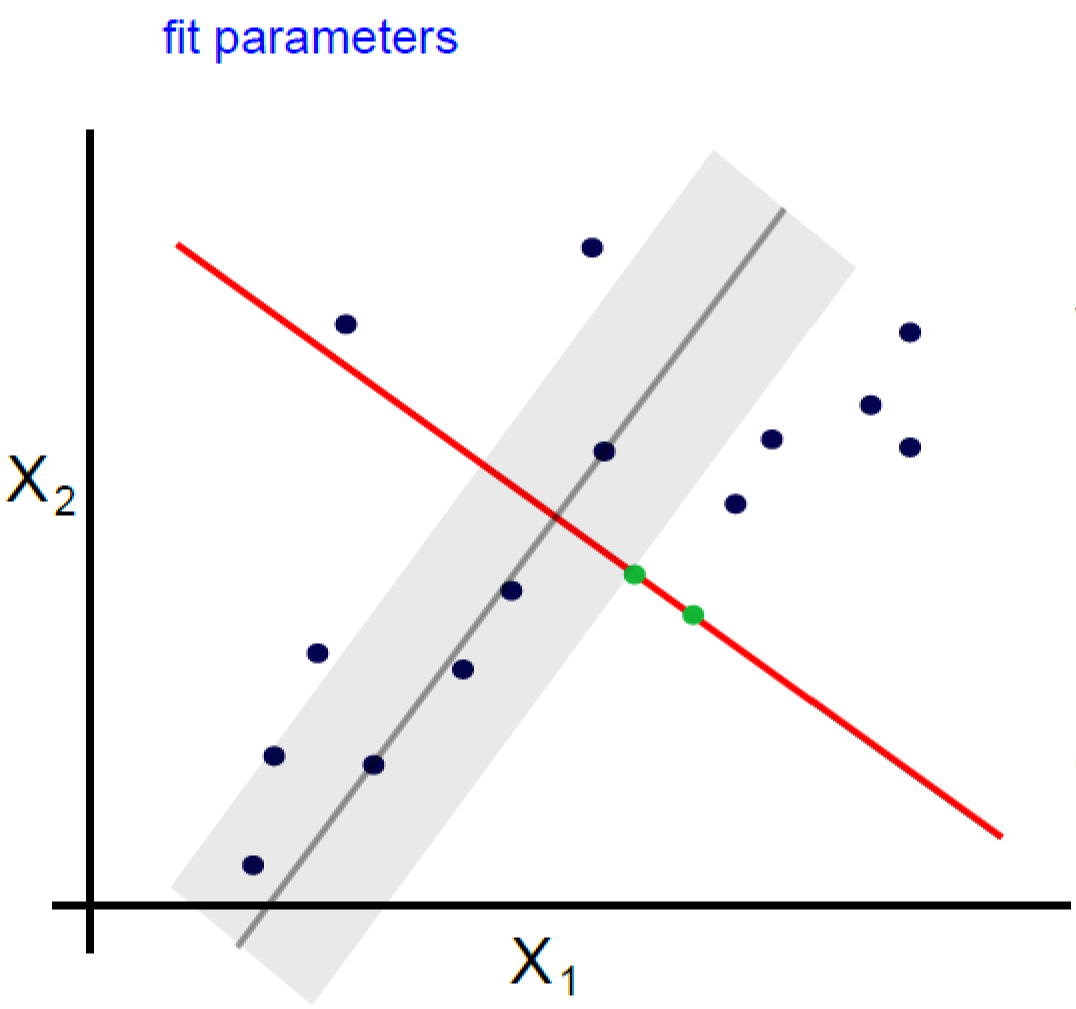
pick a subset of K points randomly (here K=2)




```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
       θ = θ_i
       nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

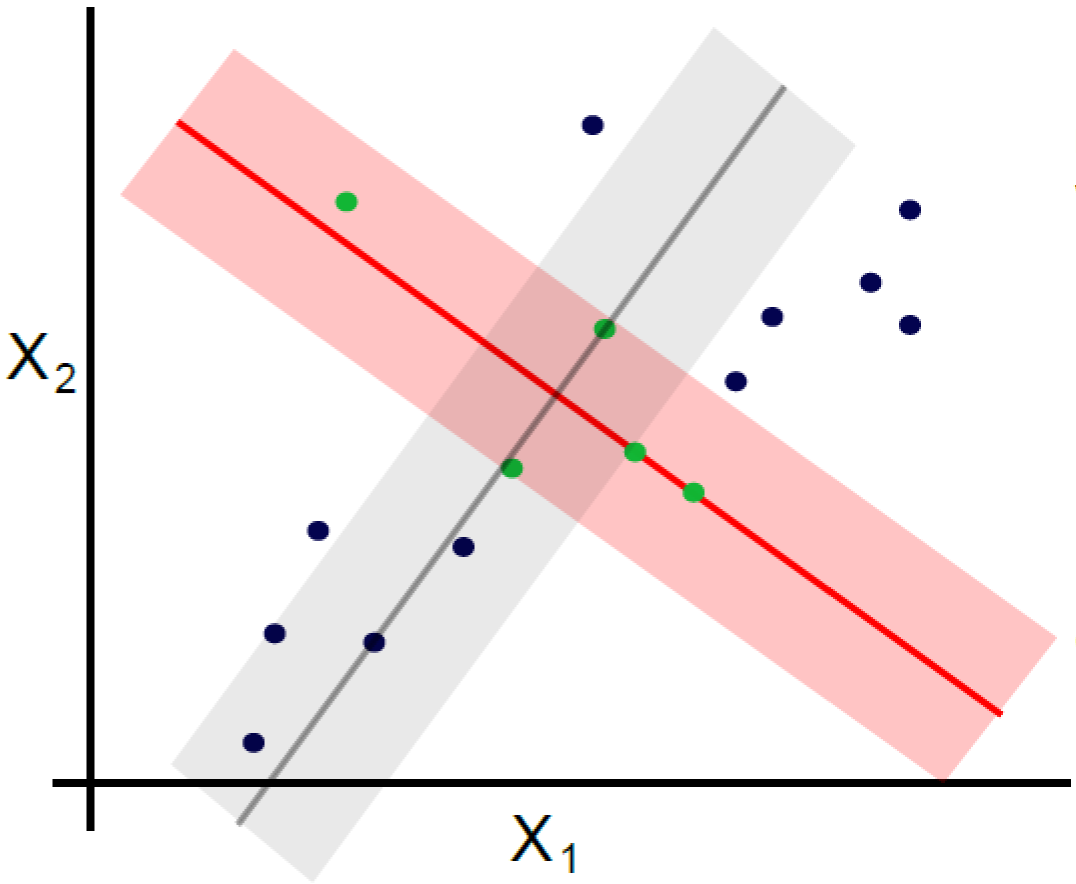


nBest = 0
while i<nIterations
  S = selectRandSubset(X)
  θ_i = fitParams(S)
  nInliers = countInliers(X, θ_i)
  if nInliers>nBest
    $\theta = \theta_i$
    nBest = nInliers
  endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

count inliers

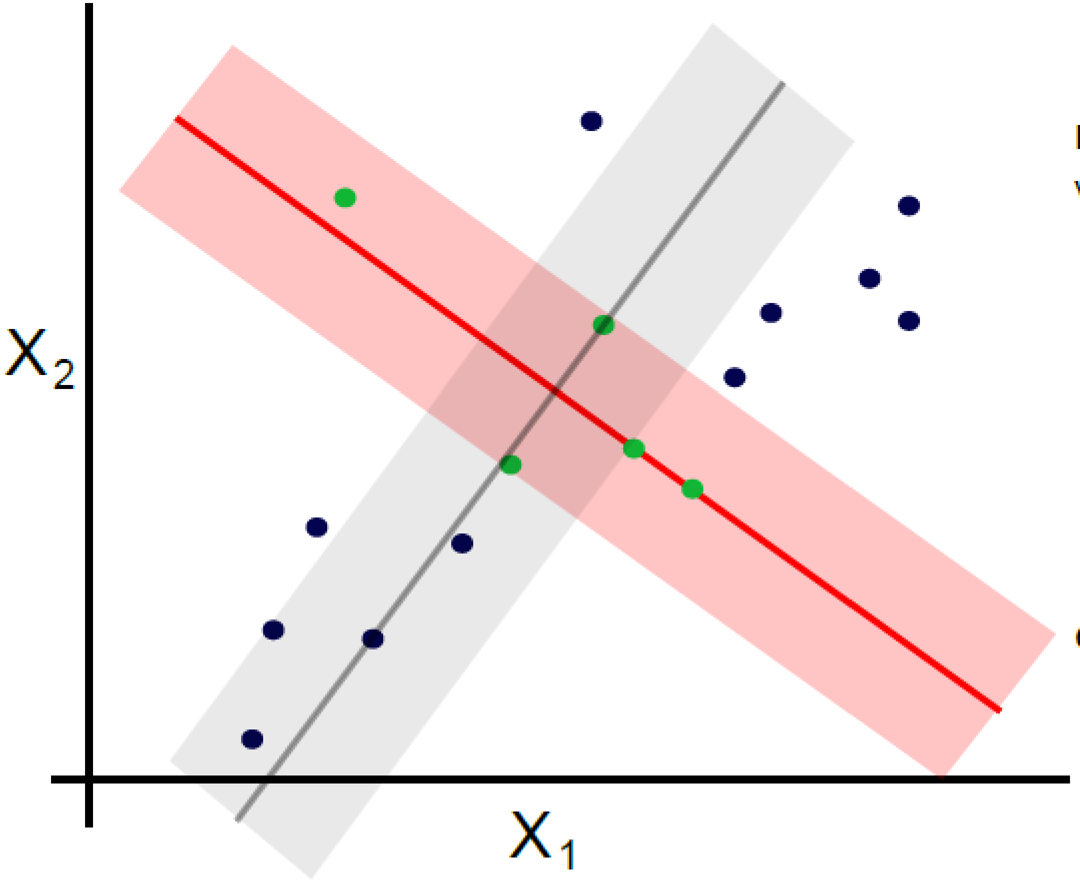


nBest = 0
while i<nIterations
    S = selectRandSubset(X)
    θ_i = fitParams(S)
    nInliers = countInliers(X,θ_i)
    if nInliers>nBest
        $\theta = \theta_i$
        nBest = nInliers
    endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

no need to update as not an improvement

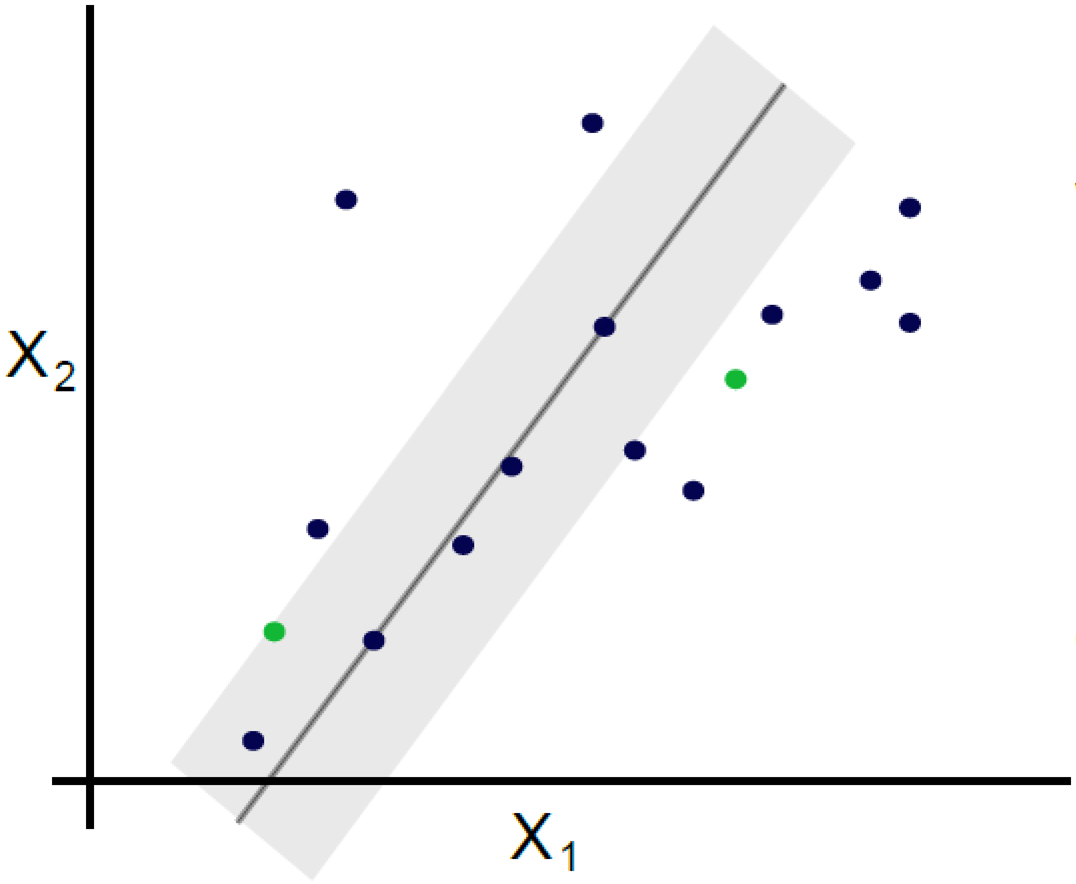



```
nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θi = fitParams(S)
   nInliers = countInliers(X,θi)
   if nInliers>nBest
       θ = θi
       nBest = nInliers
   endif
endwhile
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

pick a subset of K points randomly (here K=2)

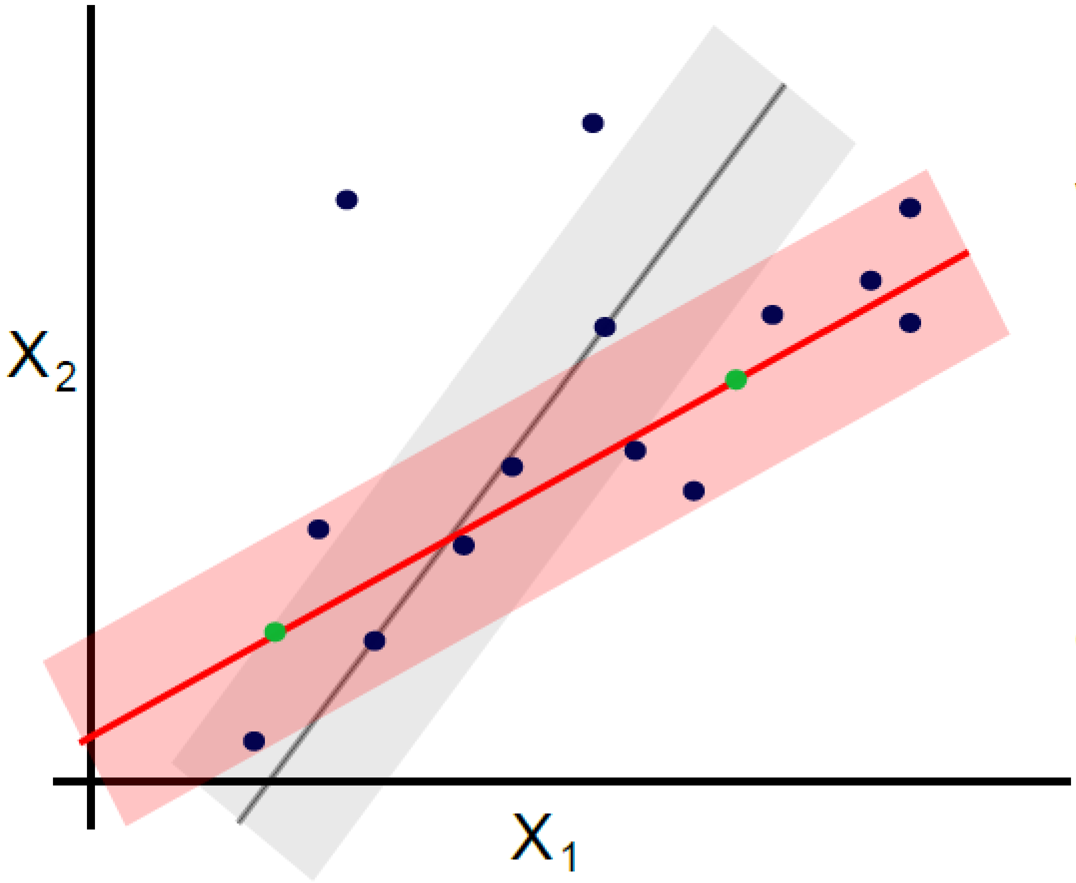


nBest = 0
while i<nIterations
  S = selectRandSubset(X)
  θ_i = fitParams(S)
  nInliers = countInliers(X, θ_i)
  if nInliers>nBest
    $\theta = \theta_i$
    nBest = nInliers
  endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

fit parameters

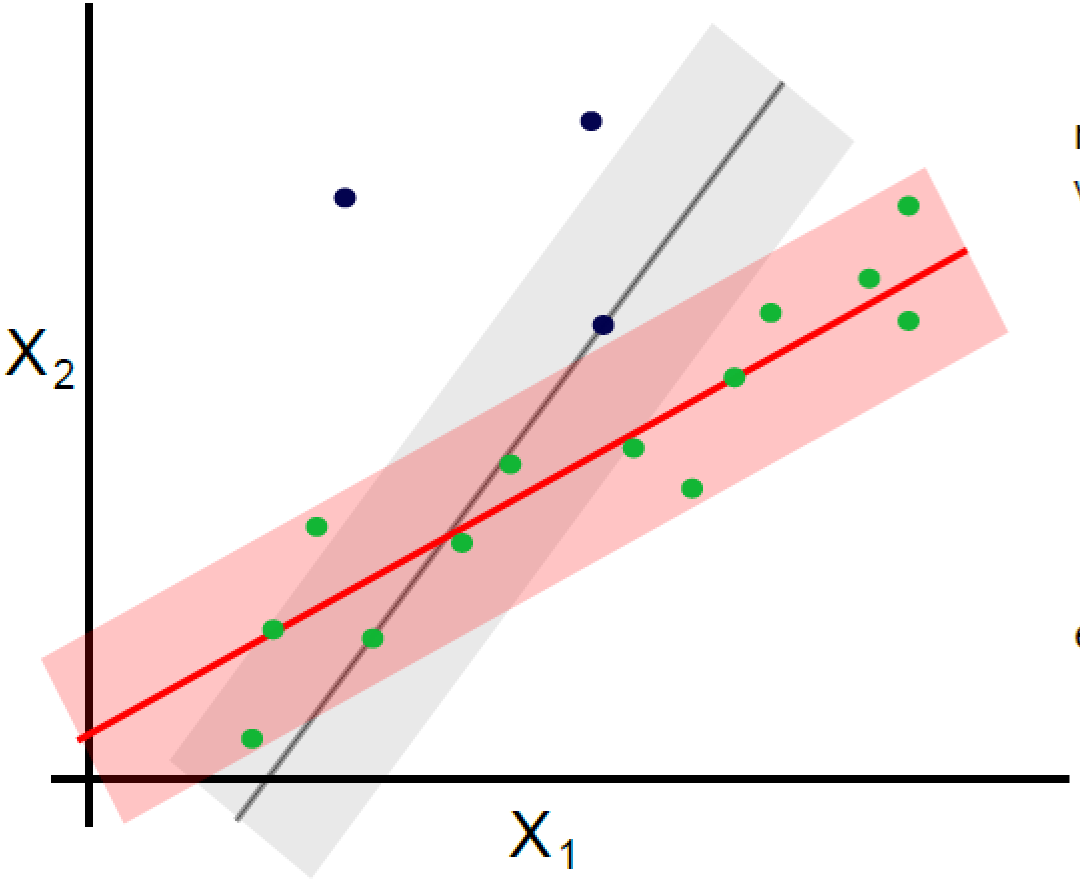


nBest = 0
while i<nIterations
   S = selectRandSubset(X)
   θ_i = fitParams(S)
   nInliers = countInliers(X,θ_i)
   if nInliers>nBest
      $\theta = \theta_i$
      nBest = nInliers
   endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

count number of inliers

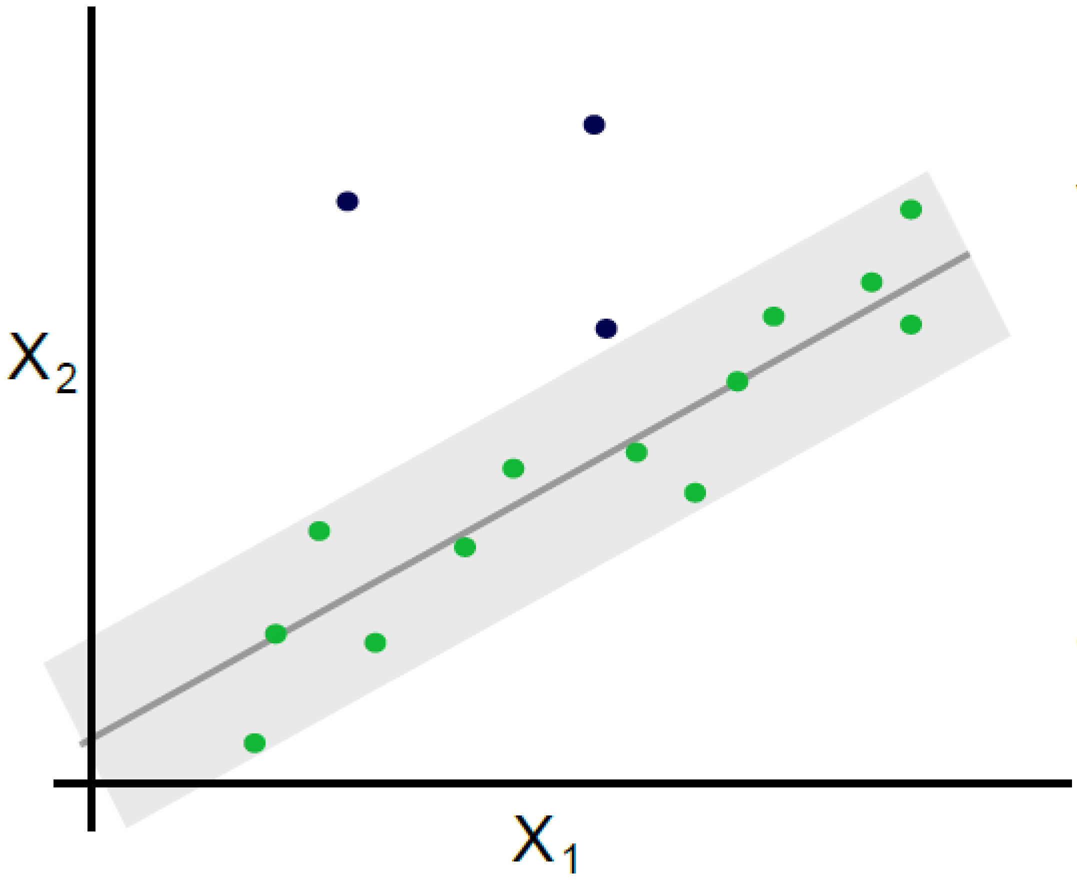


nBest = 0
while i<nIterations
  S = selectRandSubset(X)
  θ_i = fitParams(S)
  nInliers = countInliers(X,θ_i)
  if nInliers>nBest
    $\theta = \theta_i$
    nBest = nInliers
  endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

update stored parameters



nBest = 0
while i<nIterations
  S = selectRandSubset(X)
  θ_i = fitParams(S)
  nInliers = countInliers(X, θ_i)
  if nInliers>nBest
    $\theta = \theta_i$
    nBest = nInliers
  endif
endwhile

Courtesy of Richard E. Turner (Unv. Of Cambridge)

Appendix: RANSAC algorithm

RANSAC for estimating the fundamental matrix

```
F = eye(3,3)                                ▷ fundamental matrix initialised to identity
nBest = 0
for int i = 0; i < nIterations; i++ do
    P8 = SelectRandomSubset(P)              ▷ select subset of points (e.g. 8)
    Fi = ComputeHomography(P8)              ▷ compute F from subset
    nInliers = ComputeInliers(Fi)           ▷ compute no. of consistent points
    if nInliers > nBest then
        F = Fi
        nBest = nInliers
    end if
end for
```

Courtesy of Richard E. Turner (Unv. Of Cambridge)