

Computer Vision & Machine Learning

Alexandros Iosifidis
@
Department of Electrical and Computer Engineering
Aarhus University

This week

Image formation:

- Projection from 3D to 2D
 - Camera model
 - Homogeneous coordinates
 - Homographies
- Camera calibration
- Photometrics

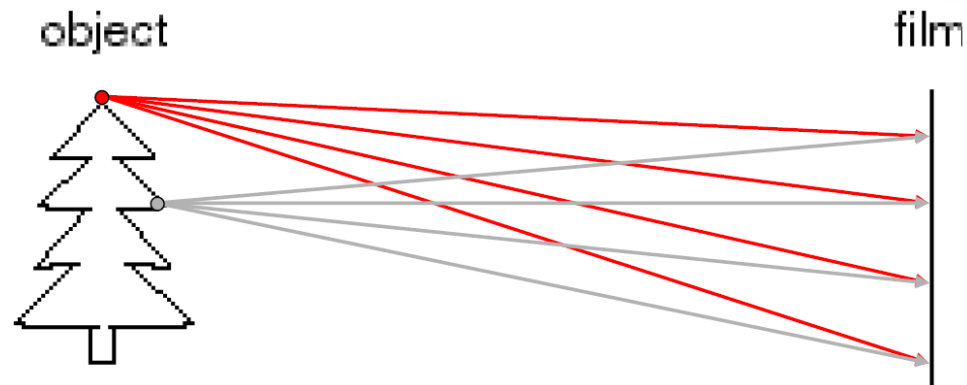
Slides based on:

- Richard Szeliski: CV: Algorithms & Applications
- Mundy & Zisserman: Projective Geometry for
Machine Vision

Camera model

Basic ideas of the camera model:

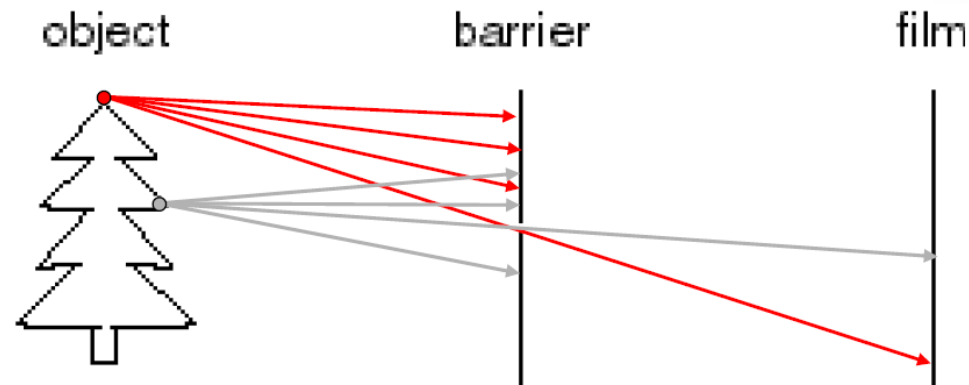
- We place a film at a position
- Light coming from a source of light (e.g. sun) reflects to the object and light rays are directed to the film



Camera model

Basic ideas of the camera model:

- We place a film at a position
- Light coming from a source of light (e.g. sun) reflects to the object and light rays are directed to the film
- We add a barrier between the object and the film
 - It has a small hole that allows few light rays pass
 - All other light rays are blocked

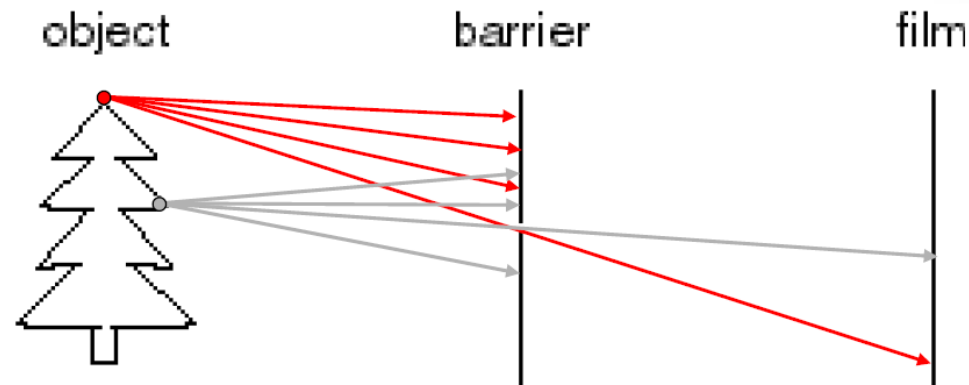


Camera model

Basic ideas of the camera model:

- We place a film at a position
- Light coming from a source of light (e.g. sun) reflects to the object and light rays are directed to the film
- We add a barrier between the object and the film
 - It has a small hole that allows few light rays pass
 - All other light rays are blocked

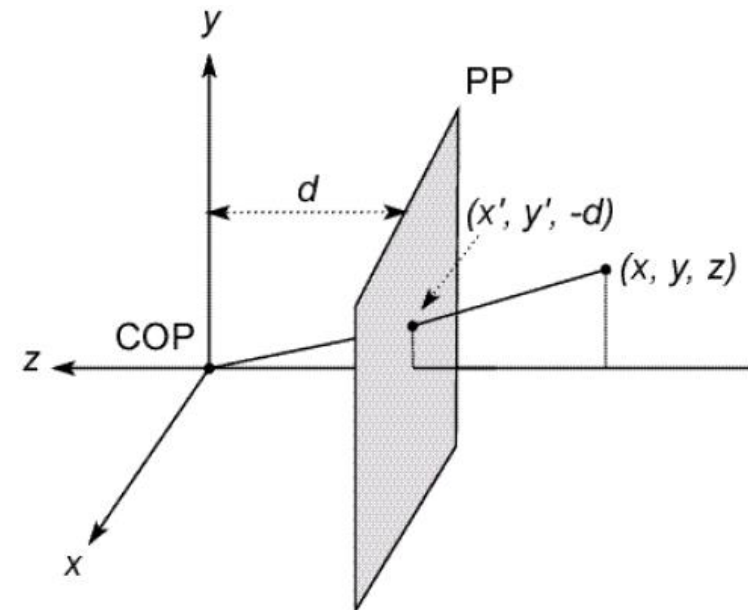
The above process leads to an
inverted image of the object
(in the y axis)



Camera model

We use the above ideas to create the 'pin-hole' camera model:

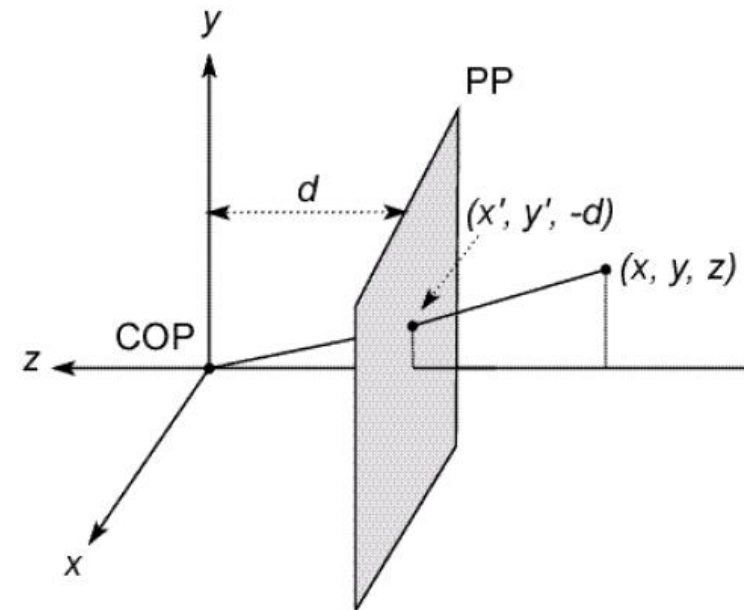
- We put the optical center (Center of Projection – COP) at the origin
- We put the image plane (Projection Plane) in front of the COP
 - Why?



Camera model

We use the above ideas to create the 'pin-hole' camera model:

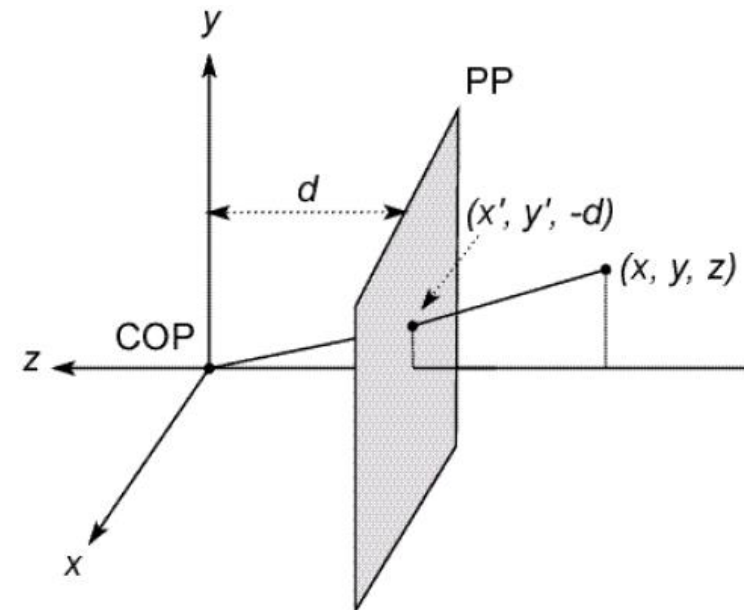
- We put the optical center (Center of Projection – COP) at the origin
- We put the image plane (Projection Plane) in front of the COP
 - In this way objects are not flipped



Camera model

We use the above ideas to create the 'pin-hole' camera model:

- We put the optical center (Center of Projection – COP) at the origin
- We put the image plane (Projection Plane) in front of the COP
 - In this way objects are not flipped
- In order to obtain a right-handed coordinate system, the camera looks towards the negative z-axis



Camera model

We use the above ideas to create the 'pin-hole' camera model:

- We put the optical center (Center of Projection – COP) at the origin
- We put the image plane (Projection Plane) in front of the COP
 - In this way objects are not flipped
- In order to obtain a right-handed coordinate system, the camera looks towards the negative z-axis

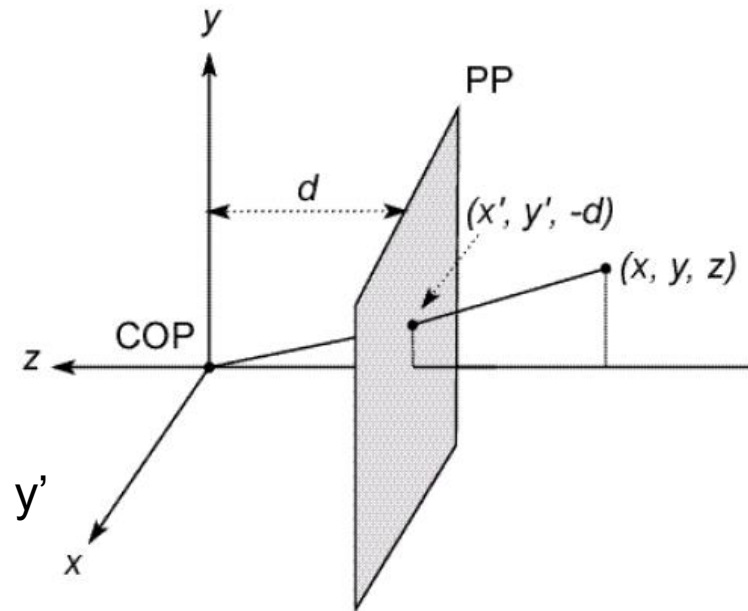
- Projection equation:

- Using similar triangles

$$\frac{AB}{A'B'} = \frac{BC}{B'C'} = \frac{AC}{A'C'}$$

$$(x, y, z) \rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}, -d\right)$$

- The coordinates on the image plane are x' and y'



Homogeneous coordinates

The image coordinates are not linear (division by z)

To transform them to homogeneous, we do the following trick

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous image coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Homogeneous scene coordinates

Going from homogeneous coordinates back to Cartesian coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Perspective projection

Using the homogeneous coordinates we can define a linear mapping (projection)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}\right)$$

divide by the third coordinate

The matrix at the left is the projection matrix. It can be formulated also as a 4x4 matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ -z/d \end{bmatrix} \Rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}\right)$$

divide by the fourth coordinate

Perspective projection

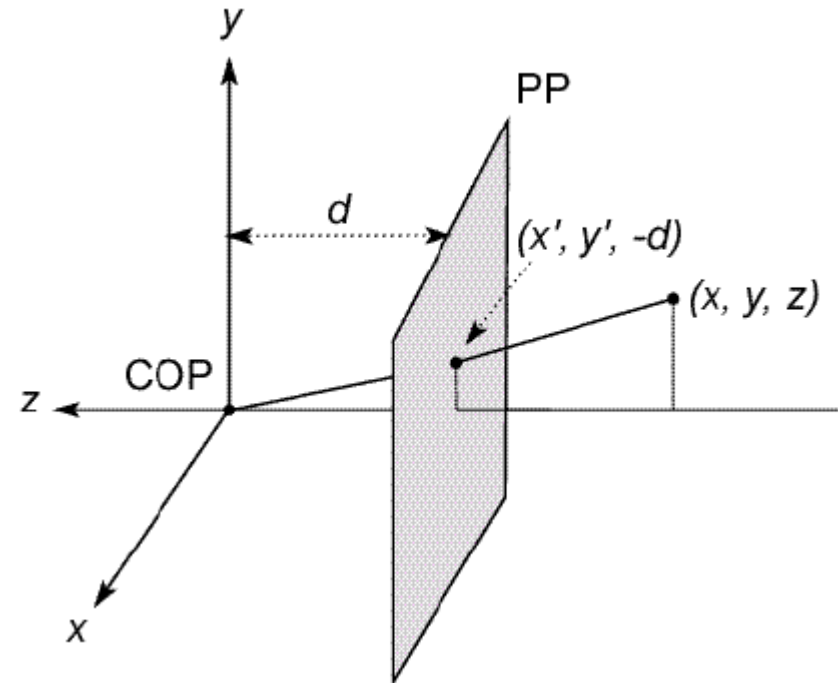
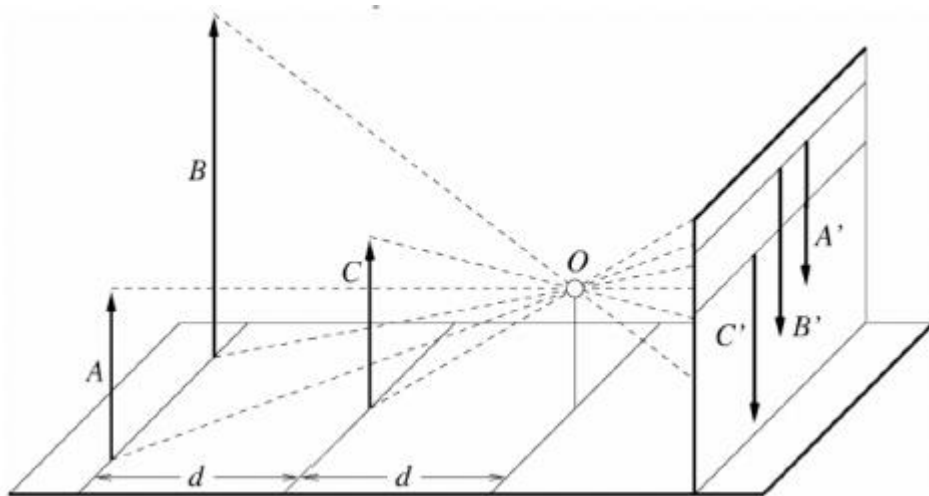
Scaling the projection matrix does not change the transformation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}\right)$$

$$\begin{bmatrix} -d & 0 & 0 & 0 \\ 0 & -d & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} -dx \\ -dy \\ z \end{bmatrix} \Rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}\right)$$

Perspective projection

Objects far from the camera COP appear smaller

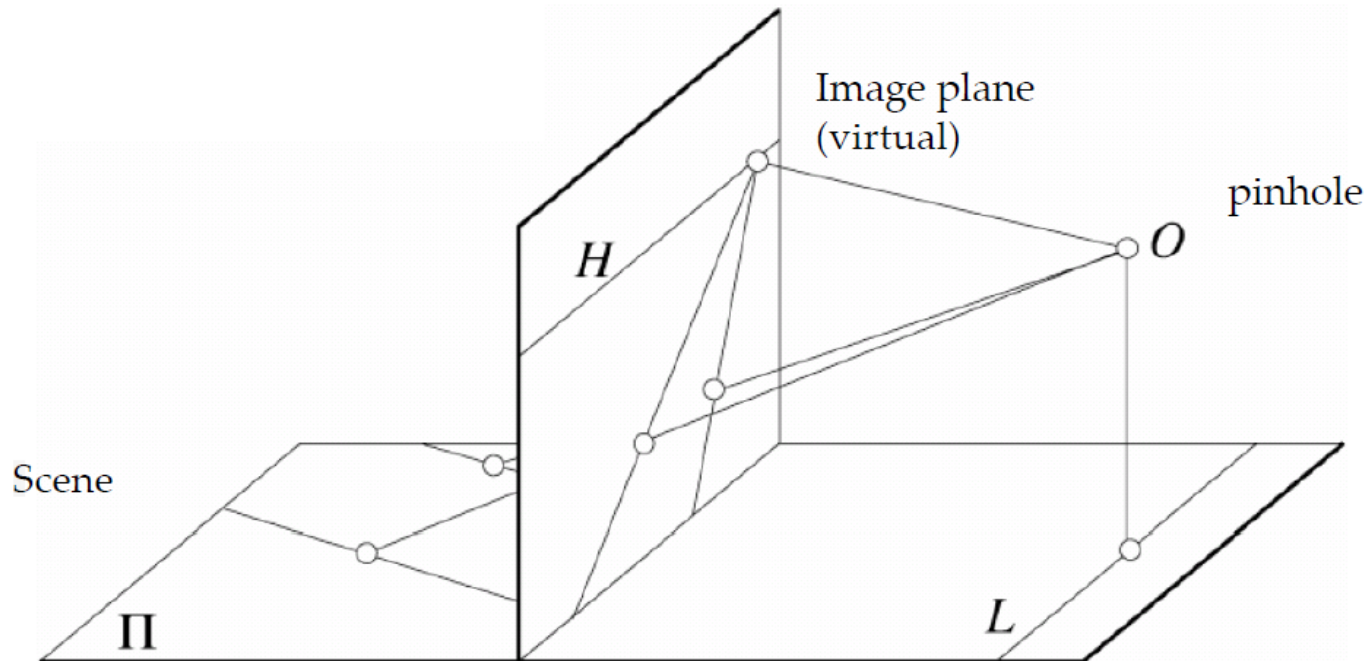


$$(x, y, z) \rightarrow \left(-d\frac{x}{z}, -d\frac{y}{z}\right)$$

Perspective projection

Parallel lines (in the real world) intersect in the image

Converge in image on the horizon line (H)



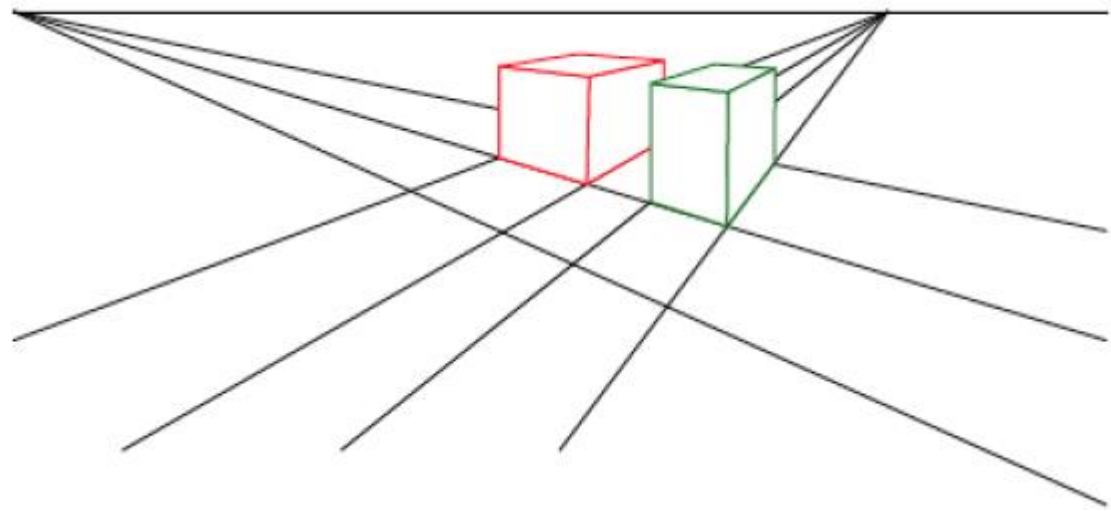
Perspective projection

Each set of parallel lines (direction in the world) meets at a different point in the image plane

- This is the vanishing point for this direction

Sets of parallel lines on the same plane lead to collinear vanishing points

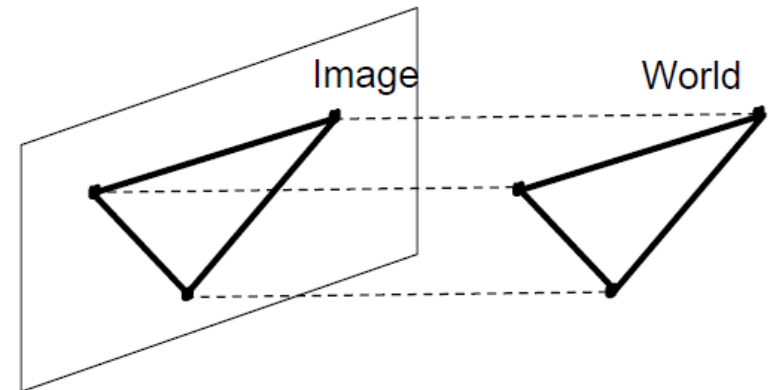
- This is the horizon for that plane



Orthographic projection

A special case of the perspective projection

- It is a good approximation for the cases where the COP of the PP is infinite (when the object is very far from the camera COP)
- It is also called 'parallel projection'



Projection matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Camera parameters

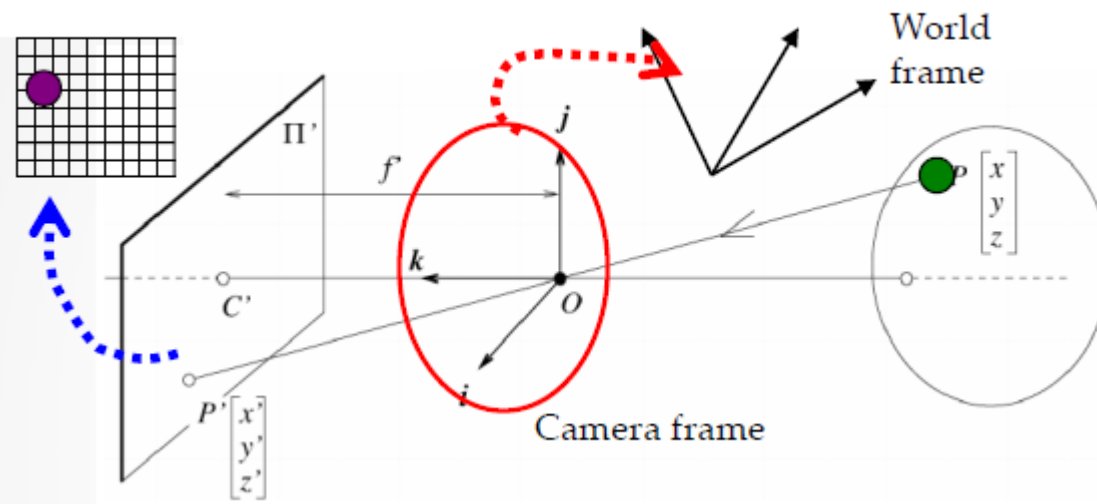
Extrinsic parameters:

Image plane \leftrightarrow World

$$\begin{pmatrix} \text{2D} \\ \text{point} \\ (3 \times 1) \end{pmatrix} = \begin{pmatrix} \text{Camera to} \\ \text{pixel coord.} \\ \text{trans. matrix} \\ (3 \times 3) \end{pmatrix} \begin{pmatrix} \text{Perspective} \\ \text{projection matrix} \\ (3 \times 4) \end{pmatrix} \begin{pmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ (4 \times 4) \end{pmatrix} \begin{pmatrix} \text{3D} \\ \text{point} \\ (4 \times 1) \end{pmatrix}$$

Intrinsic parameters:

Image coordinates relative to camera \leftrightarrow Pixel coordinates



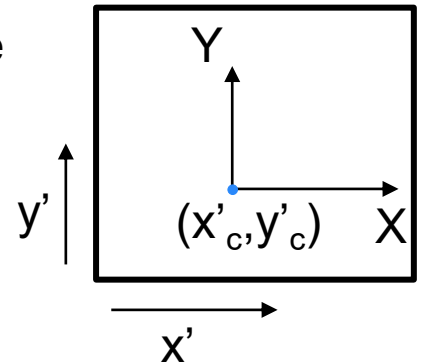
Camera parameters

A camera is described using the following parameters:

- Translation matrix \mathbf{T} , encoding the translation of the optical center from the origin of the world coordinates
- Rotation matrix \mathbf{R} , encoding the rotation of the image plane
- focal length f , principle point (x'_c, y'_c) , pixels size (s_x, s_y)

Intrinsic parameters: f , (x'_c, y'_c) , (s_x, s_y)

Extrinsic parameters: \mathbf{T} , \mathbf{R}



Camera parameters

Projection:

$$\mathbf{X} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M}\mathbf{X}$$

The matrix \mathbf{M} models the effect of all parameters and can be decomposed as:

$$\mathbf{M} = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$$

Intrinsic

Projection

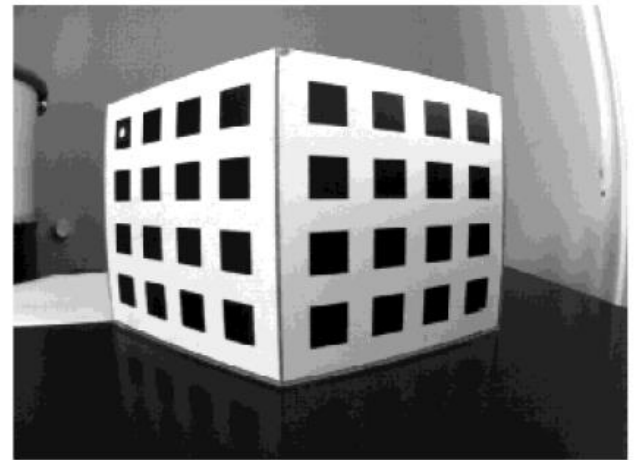
Rotation

Translation

Camera calibration

All these parameters are calculated by applying a process called camera calibration:

- Use a known (and relatively easy) scene
- Determine the positions of a set of points in the scene
- Determine the pixels which depict the same points in the image plane
- Using the correspondences, find the values in the above matrices



The Opti-CAL Calibration Target Image

Projective geometry (applications)

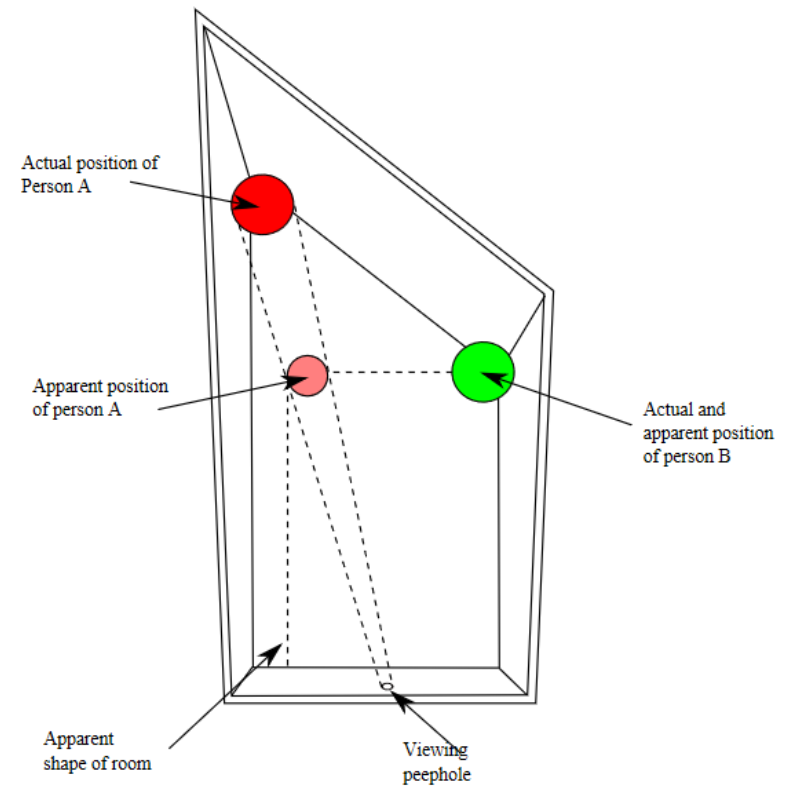
It is used for:

- Drawing
- Measurements
- Mathematics for projection
- Undistorting images
- Focus of expansion
- Camera pose estimation
- Object recognition

Projective geometry (applications)



Projective geometry (applications)

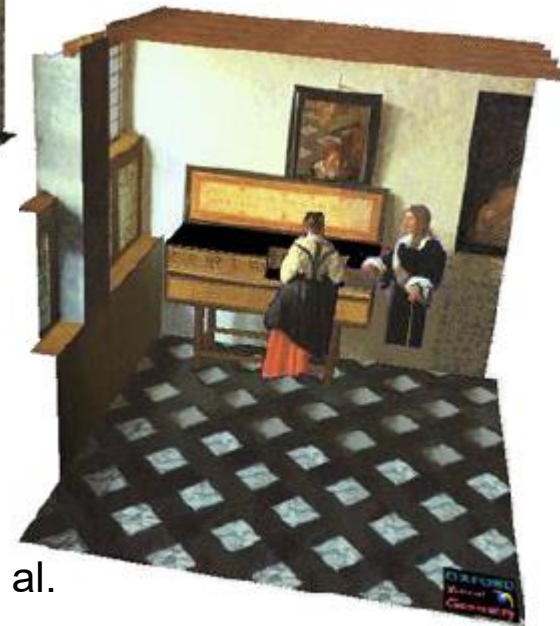
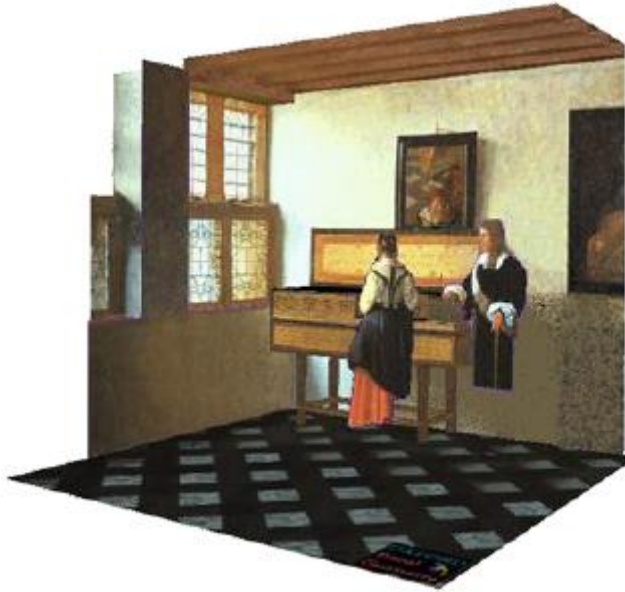


https://en.wikipedia.org/wiki/Ames_room

Projective geometry (applications)



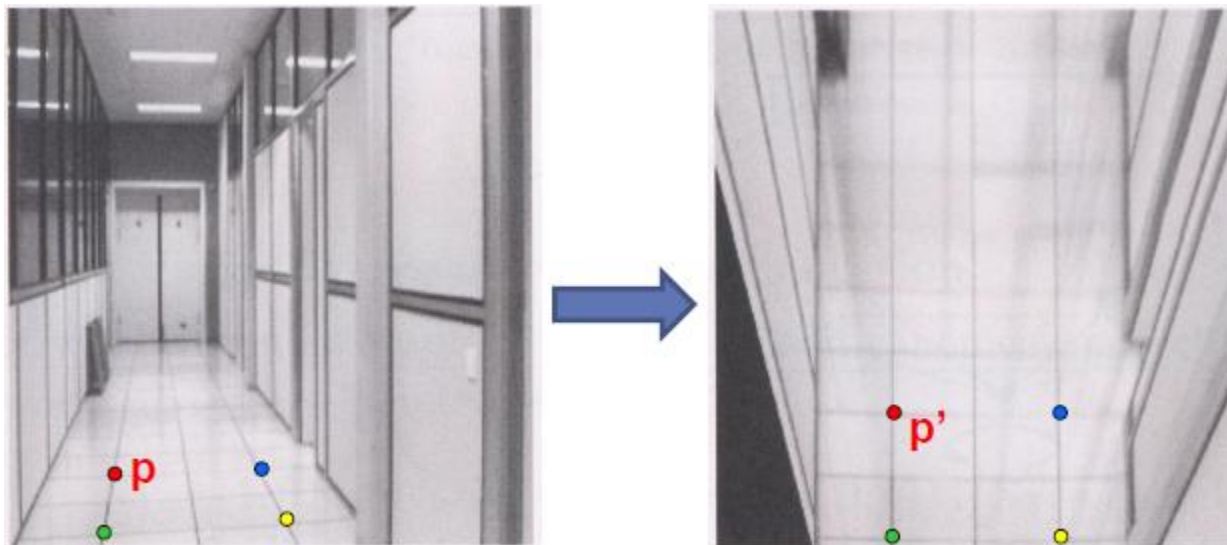
Vermeer's Music Lesson



Reconstructions by Criminisi et al.

Projective geometry (applications)

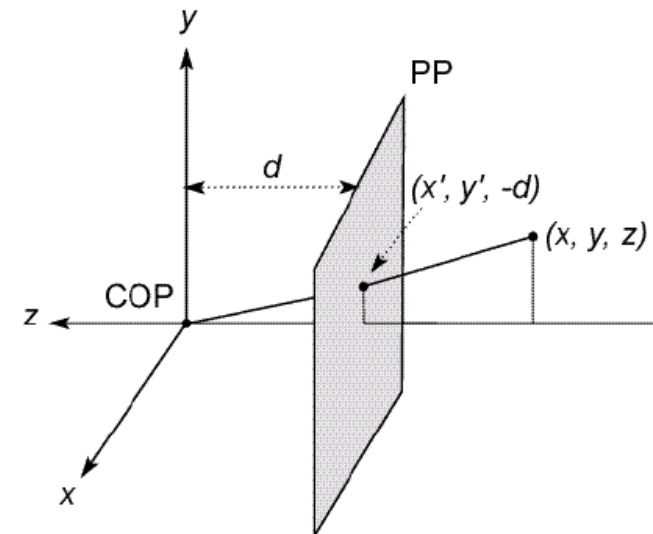
Image-to-image correspondence (reflection)



Projection properties

Degenerate cases:

- Line through the focal point (COP) projects to a point
- Plane through the focal point projects to a line
- Plane vertical to the image plane projects to a part of the image



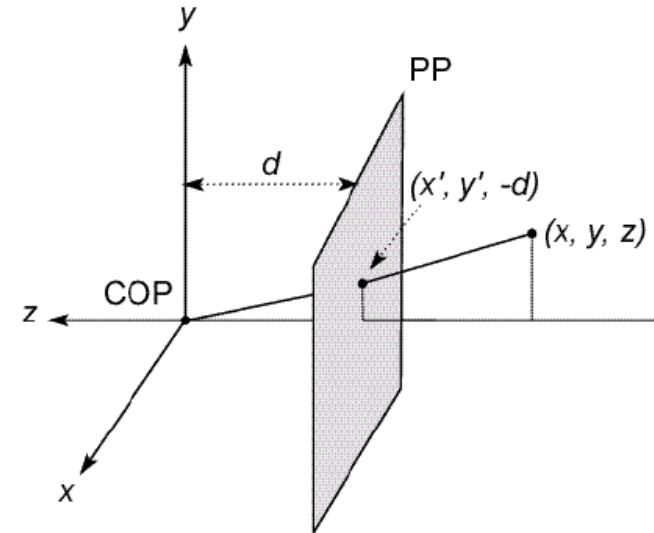
The projective plane

We need homogeneous coordinates to:

- represent points at infinity, homographies, perspective projection, multi-view relationships

Geometric intuition:

- a point in the image plane is equivalent to a ray in the world



The projective plane

We need homogeneous coordinates to:

- represent points at infinity, homographies, perspective projection, multi-view relationships

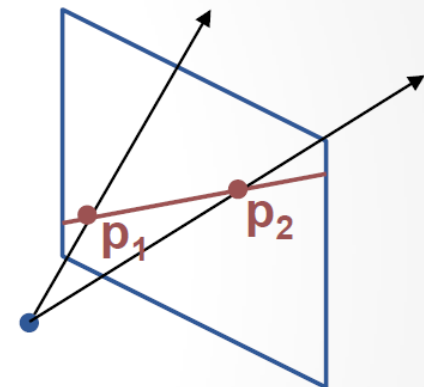
Geometric intuition:

- a point in the image plane is equivalent to a ray in the world
- a line in the image plane corresponds to a plane of rays passing through the COP

A plane is described by: $ax + by + cz = 0$

By describing the line using a homogeneous vector $\mathbf{l} = [a, b, c]^T$
and $\mathbf{p} = [x, y, z]^T$, we have $0 = \mathbf{p}^T \mathbf{l}$

Thus, lines \mathbf{l} and $k\mathbf{l}$ are equivalent for all values of k



Point and line duality

A line $\mathbf{l} = [a, b, c]^T$ is a homogeneous vector

The line \mathbf{l} is orthogonal to every point \mathbf{p} on the plane $\mathbf{p}^T \mathbf{l} = 0$

The line \mathbf{l} spanned by \mathbf{p}_1 and \mathbf{p}_2 is:

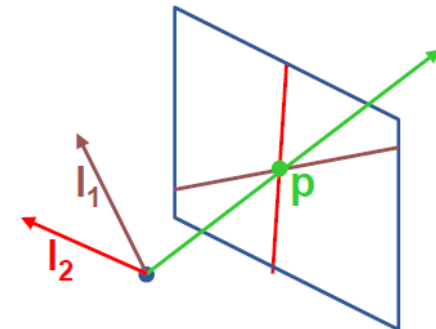
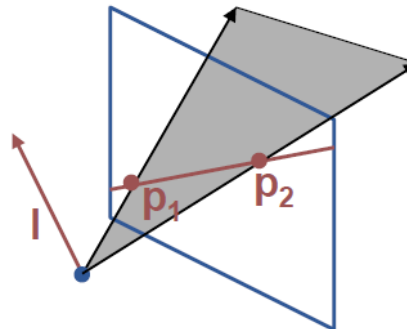
- orthogonal to \mathbf{p}_1 and $\mathbf{p}_2 \rightarrow \mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- \mathbf{l} is the plane normal

More on Point-line duality
Demo

The intersection of two lines \mathbf{l}_1 and \mathbf{l}_2 is:

- \mathbf{p} is orthogonal to \mathbf{l}_1 and $\mathbf{l}_2 \rightarrow \mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2$

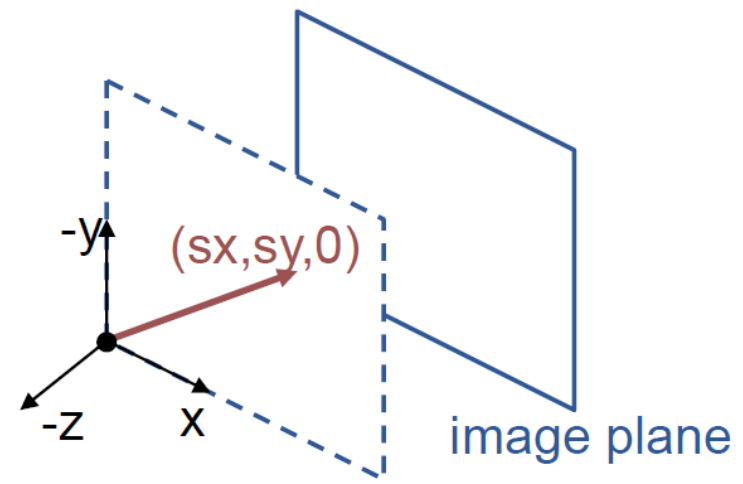
Points and lines are dual in projective space



Points at infinity

Point at infinity (“ideal point”)

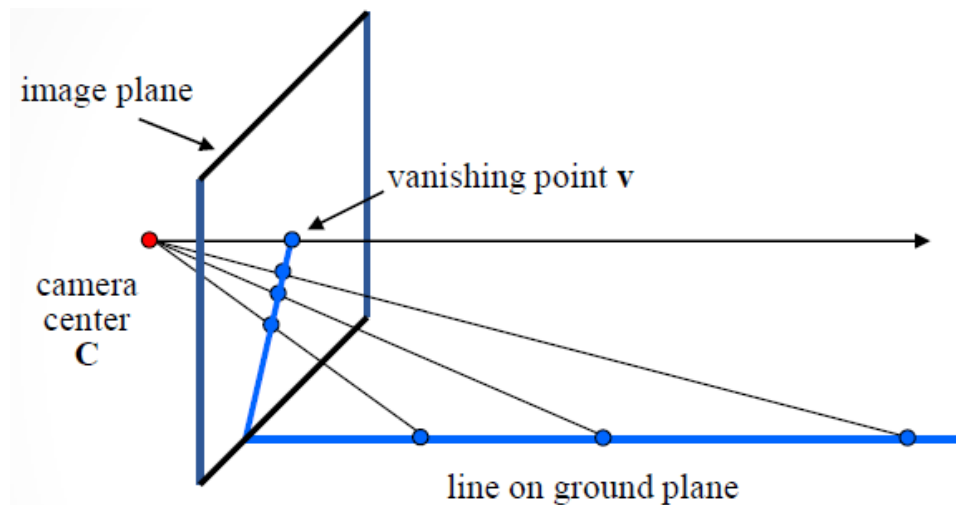
- $\mathbf{p} \sim (x, y, 0)$ – parallel to image plane
- It has infinite image coordinates.
- Why? Because when we convert from homogeneous to Cartesian coordinates, we get $(x/0, y/0)$, which can be interpreted as a point with infinite coordinates



Vanishing points

Vanishing point:

- projection of a point at infinity



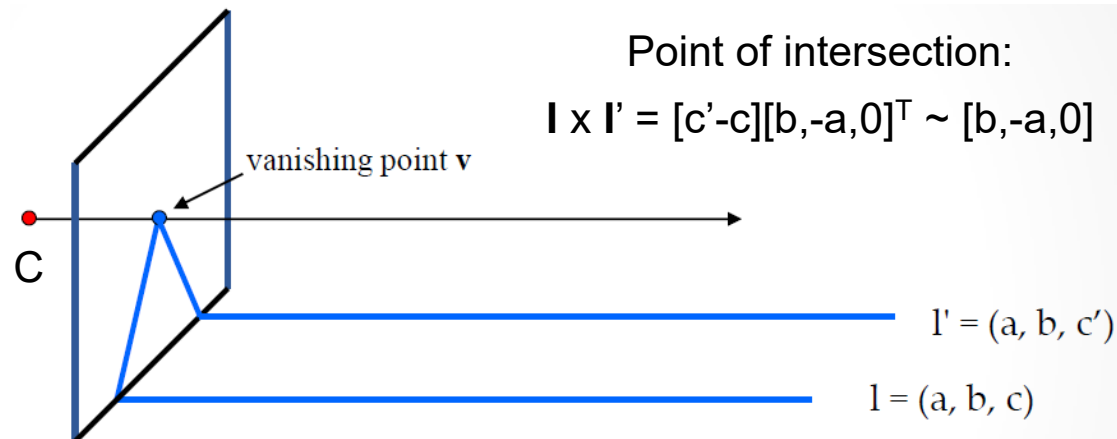
Vanishing points

Vanishing point:

- projection of a point at infinity

Properties

- Any two parallel lines have the same vanishing point \mathbf{v}
- The ray from \mathbf{C} through \mathbf{v} is parallel to the lines
- An image may have more than one vanishing points
- In fact every pixel is a potential vanishing point



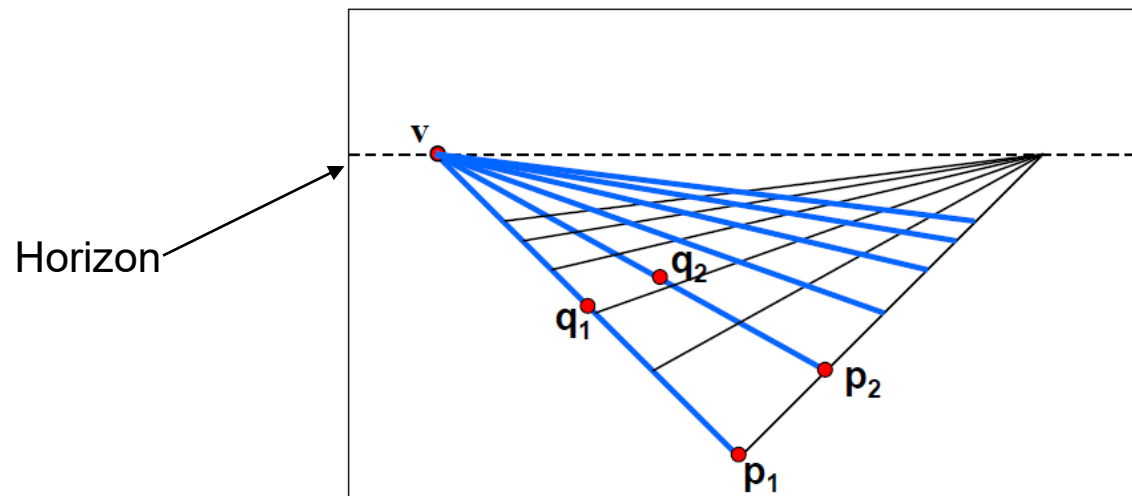
Computing vanishing points from lines

Intersect p_1q_1 and p_2q_2

$$v = (p_1 \times q_1) \times (p_2 \times q_2)$$

In order get a better solution:

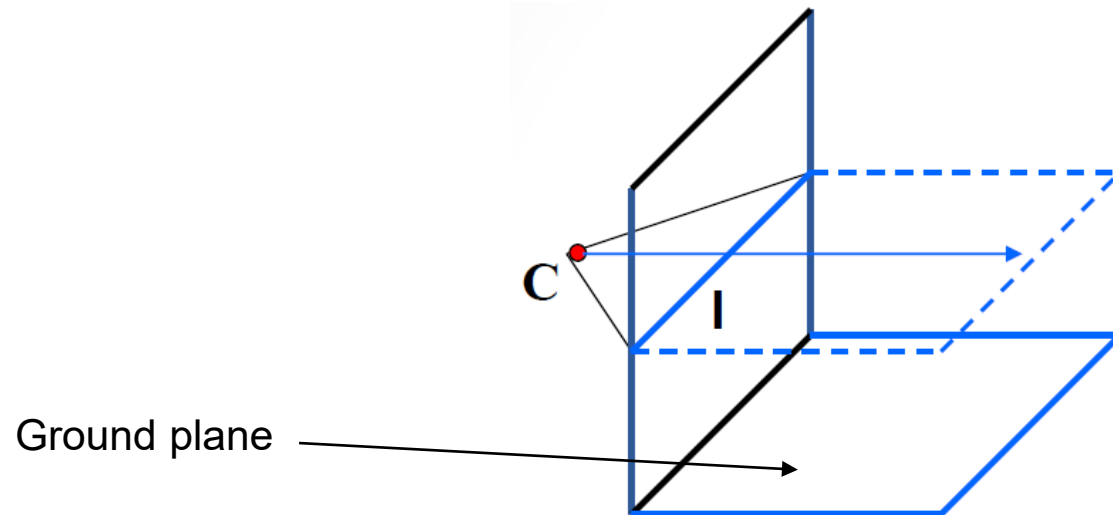
- We use more than two lines
- We compute the closest point of intersection (least squares version)



Computing the horizon

Properties

- \mathbf{l} is intersection of horizontal plane through \mathbf{C} with image plane
- Compute \mathbf{l} from two sets of parallel lines on ground plane
- All points at same height as \mathbf{C} project to \mathbf{l}
 - points higher than \mathbf{C} project above \mathbf{l}
- Provides way of comparing height of objects in the scene



Computing the horizon

Is the parachuter higher or lower than the person taking this picture?



Computing the horizon

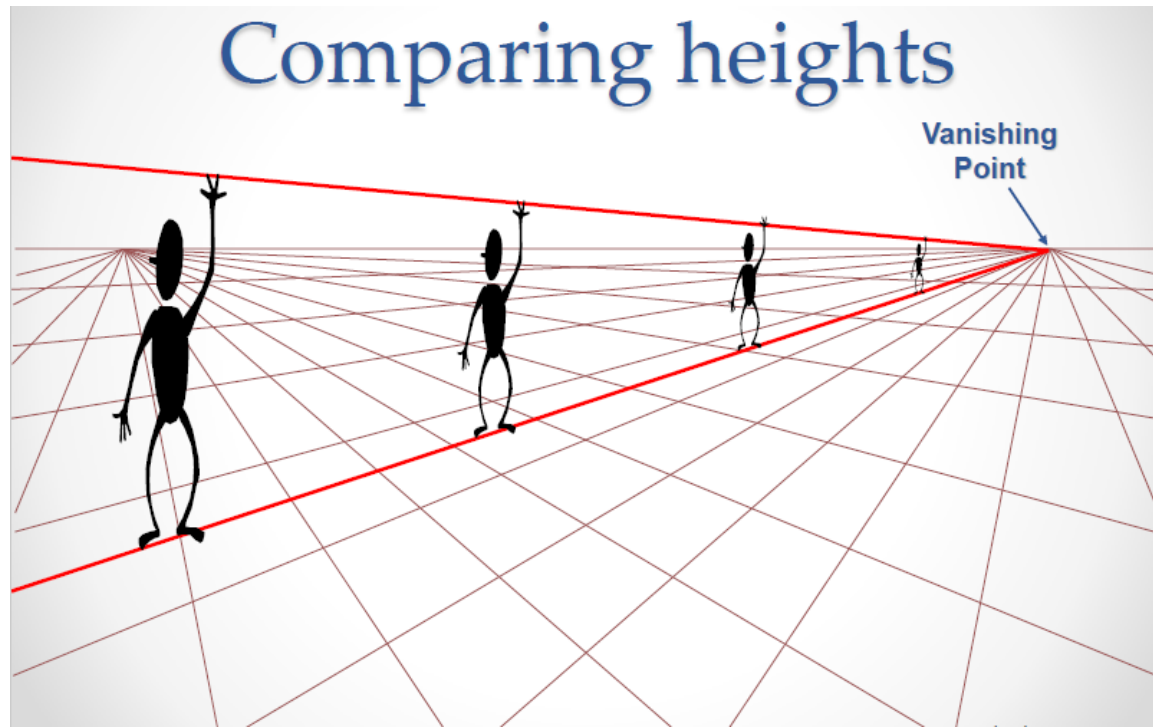
Is the parachuter higher or lower than the person taking this picture?

Lower, since he is below the horizon



Use of vanishing points

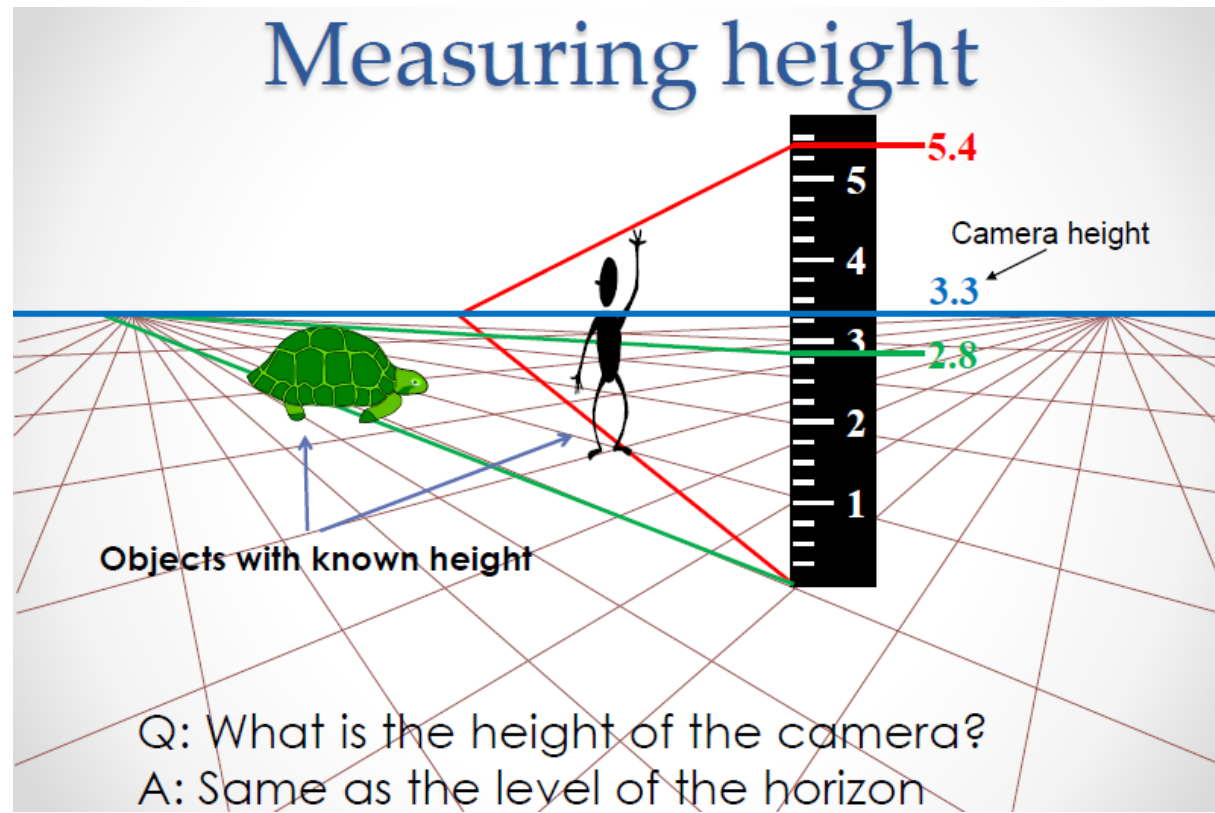
Can be used to identify objects of the same size



Use of horizon

Can be used to compute

- the size of objects in the scene
- the height of the camera



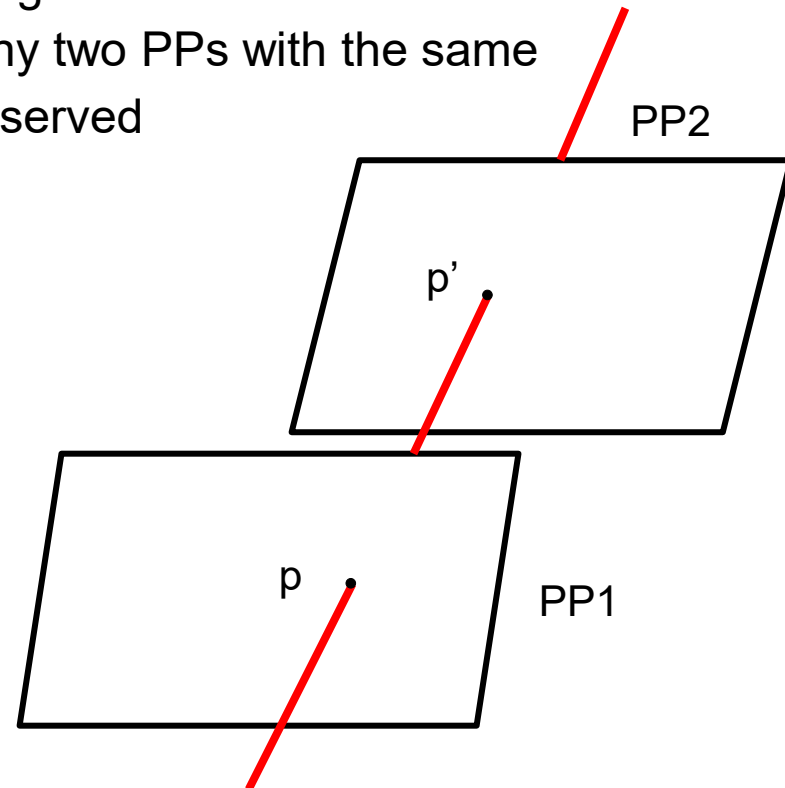
Homography

How to relate two images from the same camera center:

- We think of it as a 2D image warp from one image to another
- A projective transform is a mapping between any two PPs with the same center of projection, since straight lines are preserved

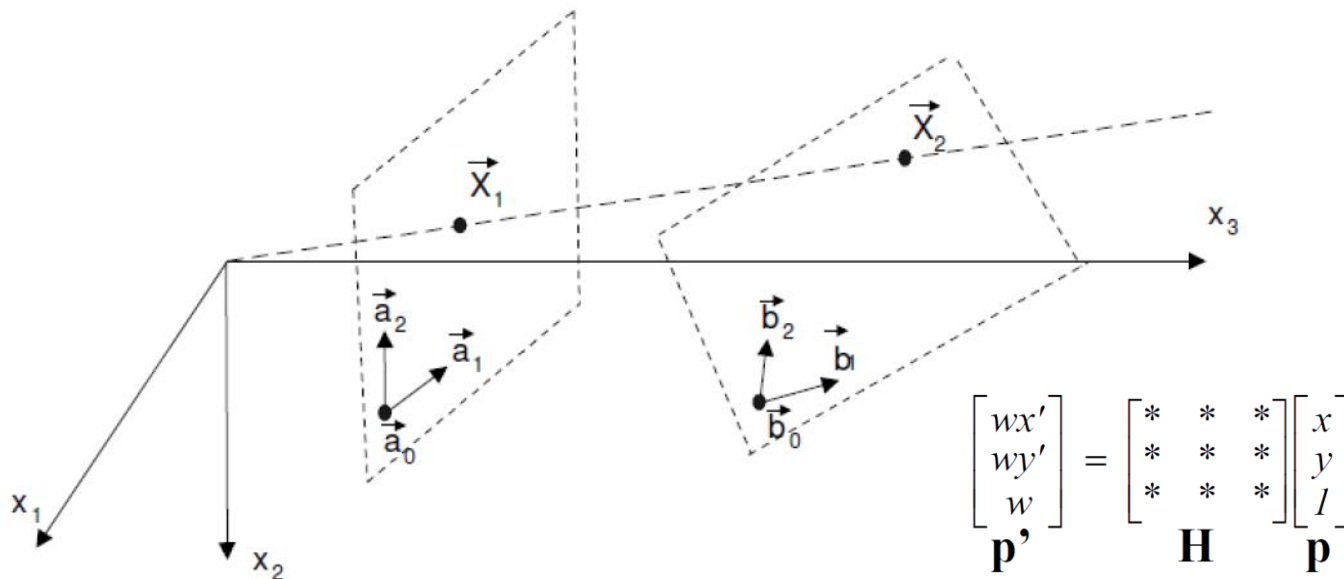
This mapping is called Homography:

$$\begin{bmatrix} wx' \\ wy' \\ w \\ \mathbf{p}' \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ \mathbf{H} \end{bmatrix} \begin{bmatrix} x \\ y \\ l \\ \mathbf{p} \end{bmatrix}$$



Homography

Mapping points between planes through a common center of projection

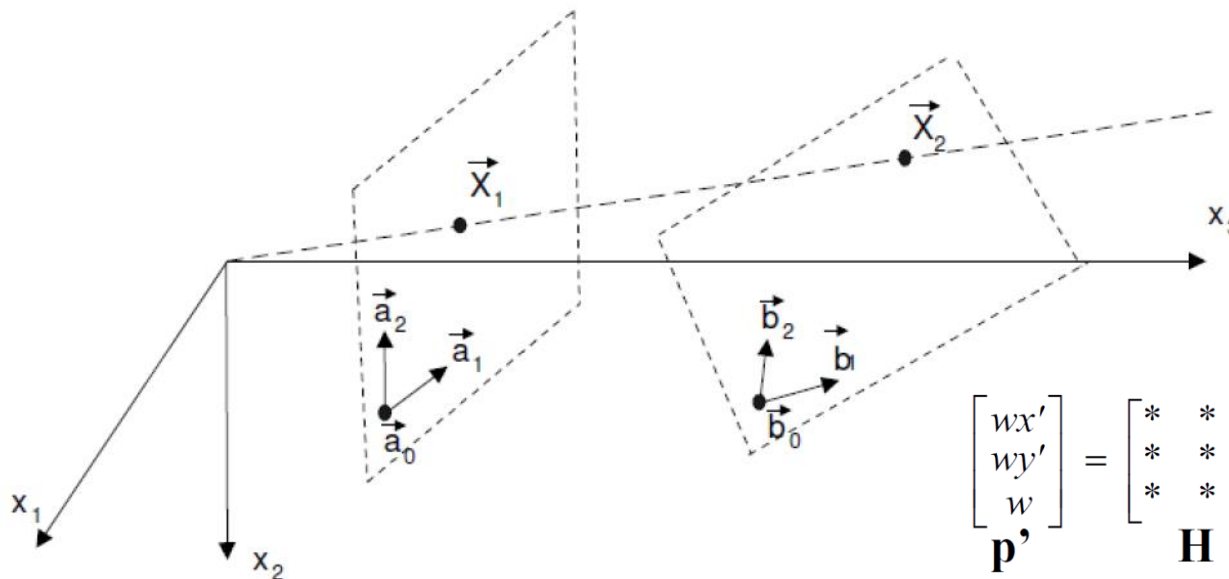


Homography

Mapping points between planes through a common center of projection

- The first plane is defined by the point \mathbf{b}_0 and two linearly independent vectors \mathbf{b}_1 and \mathbf{b}_2 contained in the plane
- We consider a point \mathbf{X}_2 . Since \mathbf{b}_1 , \mathbf{b}_2 and \mathbf{X}_2 belong to the plane

$$\vec{X}_2 = q_1 \vec{b}_1 + q_2 \vec{b}_2 + \vec{b}_0 = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ 1 \end{bmatrix} \equiv B \vec{q}$$



$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$\mathbf{p}' \quad \mathbf{H} \quad \mathbf{p}$

Homography

Mapping points between planes through a common center of projection

- The first plane is defined by the point \mathbf{b}_0 and two linearly independent vectors \mathbf{b}_1 and \mathbf{b}_2 contained in the plane
- We consider a point \mathbf{X}_2 . Since \mathbf{b}_1 , \mathbf{b}_2 and \mathbf{X}_2 belong to the plane

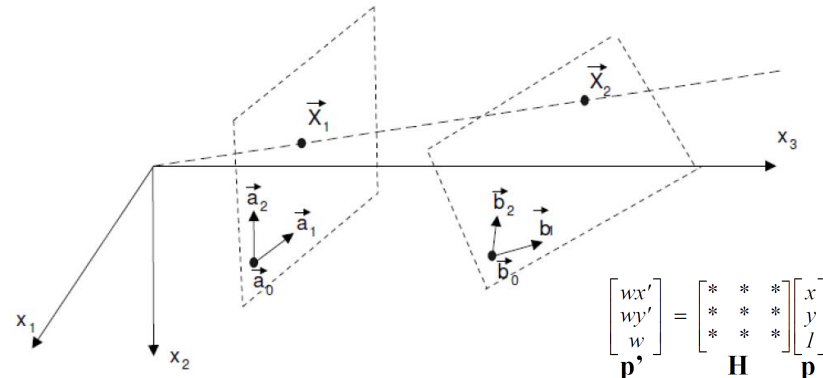
$$\overrightarrow{X_2} = q_1 \overrightarrow{b_1} + q_2 \overrightarrow{b_2} + \overrightarrow{b_0} = \begin{bmatrix} \overrightarrow{b_1} & \overrightarrow{b_2} & \overrightarrow{b_0} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ 1 \end{bmatrix} \equiv B \vec{q}$$

- The same applies for the first plane

$$\overrightarrow{X_1} = p_1 \overrightarrow{a_1} + p_2 \overrightarrow{a_2} + \overrightarrow{a_0} = \begin{bmatrix} \overrightarrow{a_1} & \overrightarrow{a_2} & \overrightarrow{a_0} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ 1 \end{bmatrix} \equiv A \vec{p}$$

- Thus $\vec{p} = \alpha(\vec{q}) A^{-1} B \vec{q}$
and X_2 maps to X_1 under the perspective
transformation $\overrightarrow{X_1} = \alpha(\vec{q}) \overrightarrow{X_2}$

- The role of $\alpha(\mathbf{q})$ is to scale the vector \mathbf{p}
such that the third coordinate is equal to 1

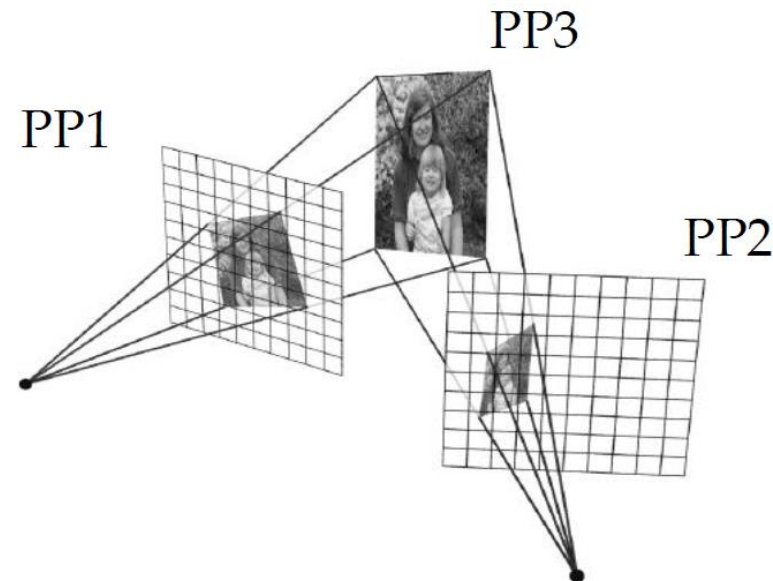


Planar scene

Here PP3 is a projection plane of both PP1 and PP2:

- Thus, mapping points from PP1 to PP2 is also homography

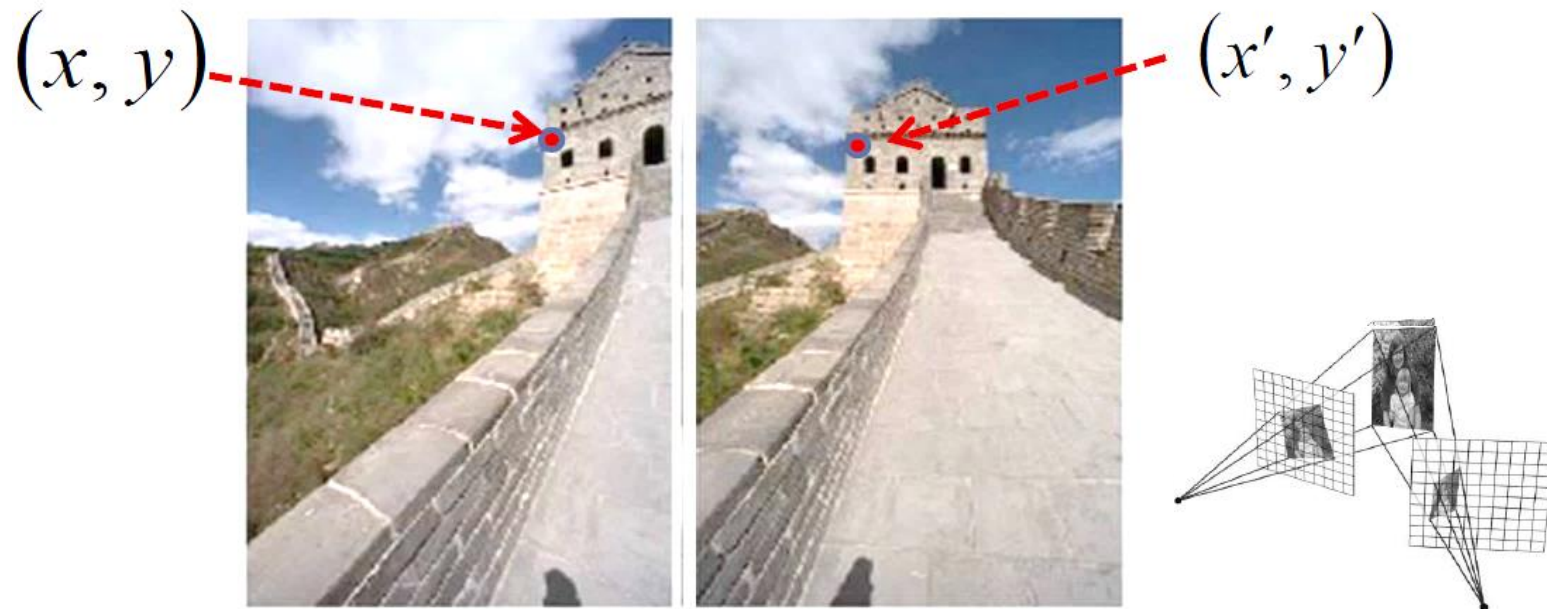
$$\begin{bmatrix} wx' \\ wy' \\ w \\ \mathbf{p}' \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \\ \mathbf{H} \end{bmatrix} \begin{bmatrix} x \\ y \\ l \\ \mathbf{p} \end{bmatrix}$$



Homography (example)

Two images of the same object (different centers of projection).

- The marked point lies in the same plane in the world.
- Therefore the mapping from (x, y) to (x', y') is a homography.

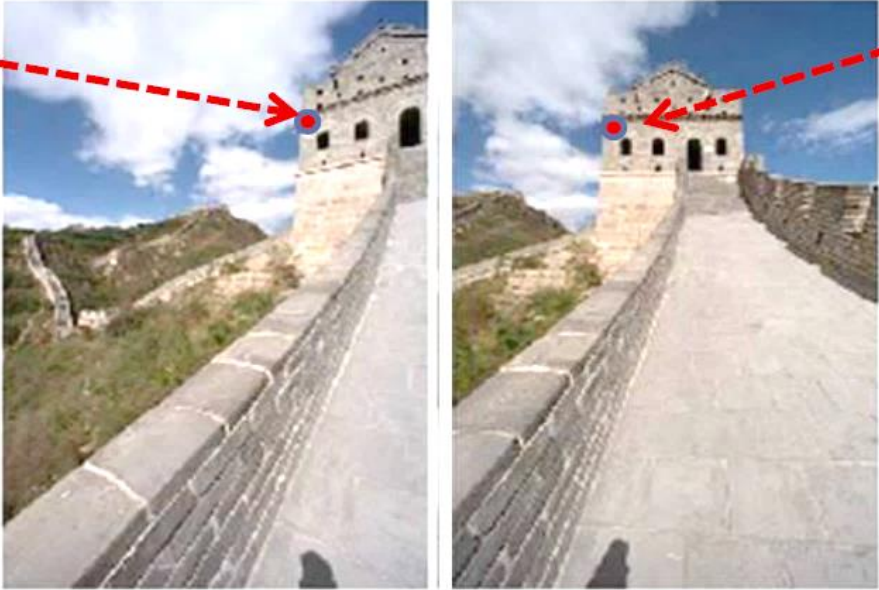


Homography (example)

Two images of the same object (different centers of projection).

- The marked point lies in the same plane in the world.
- Therefore the mapping from (x, y) to (x', y') is a homography.
- Compute $\mathbf{p}' = \mathbf{H} \mathbf{p}$
- Convert \mathbf{p}' from homogeneous to image coordinates

(x, y)



$\begin{pmatrix} wx'/w & wy'/w \end{pmatrix} = (x', y')$

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

\mathbf{p}'
 \mathbf{H}
 \mathbf{p}

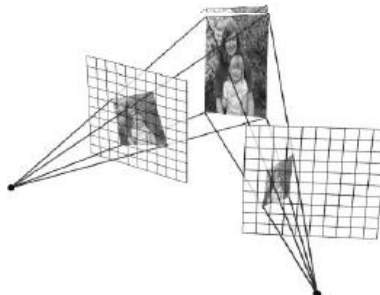
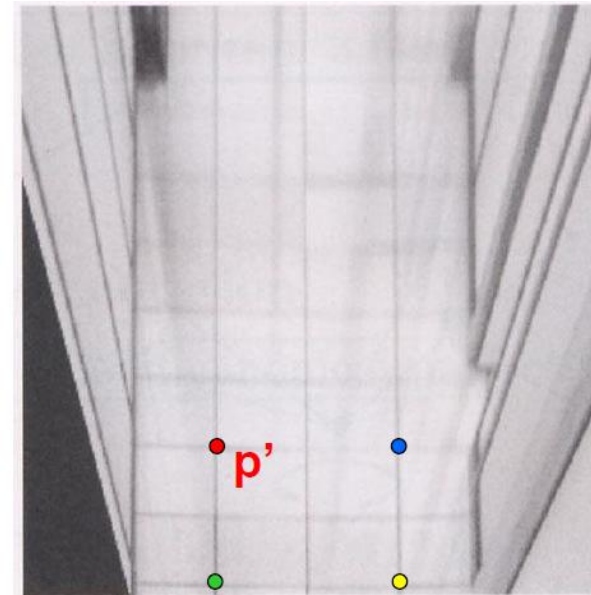
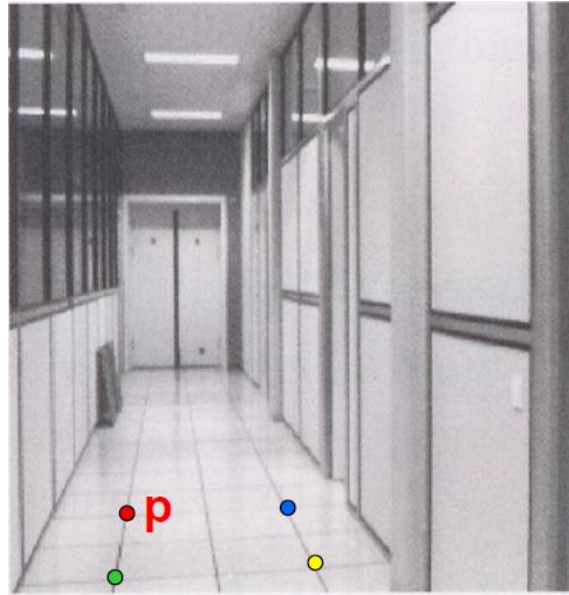


Image rectification

To unwarp (rectify) an image

- solve for homography \mathbf{H} given \mathbf{p} and \mathbf{p}'
- solve equations of the form: $w\mathbf{p}' = \mathbf{H} \mathbf{p}$
 - linear in unknowns: w and coefficients of \mathbf{H}
 - \mathbf{H} is defined up to an arbitrary scale factor
 - how many points are necessary to solve for \mathbf{H} ?



Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

It is a least squares problem:

- Since \mathbf{h} is only defined up to scale, solve for unit vector \mathbf{h}
- Solution: \mathbf{h} is the eigenvector of the matrix $\mathbf{A}^T \mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$x'_i = \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$y'_i = \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}}$$

$$x'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{00}x_i + h_{01}y_i + h_{02}$$

$$y'_i(h_{20}x_i + h_{21}y_i + h_{22}) = h_{10}x_i + h_{11}y_i + h_{12}$$

Outliers might affect the result!

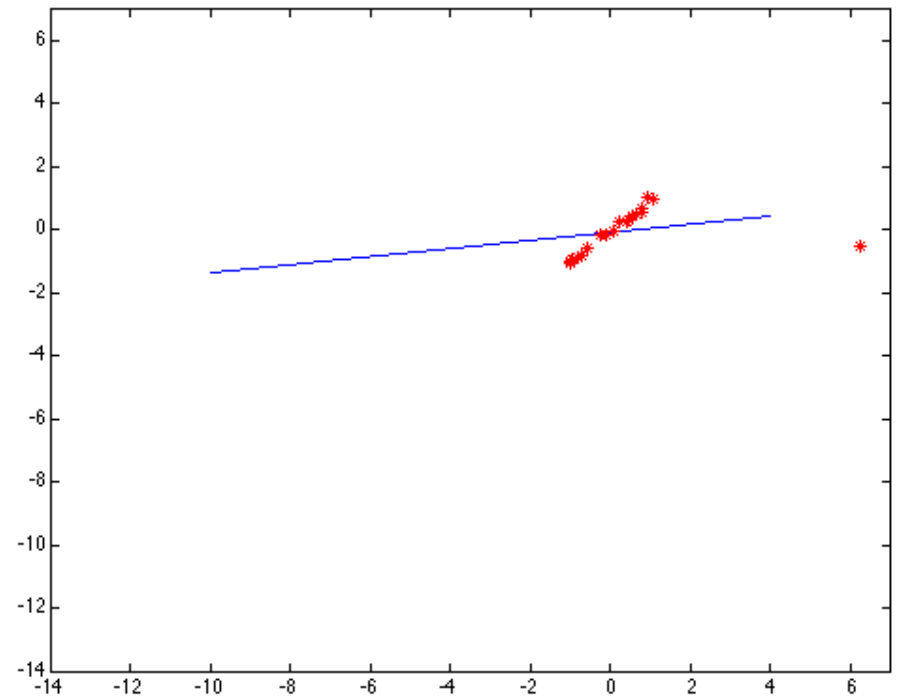
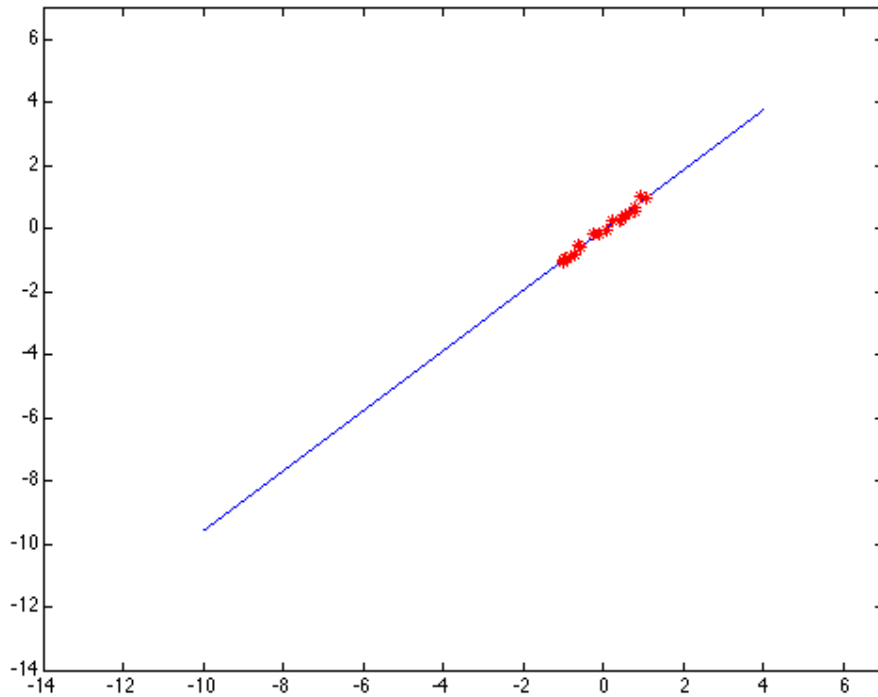


It is a least squares problem:

- Since \mathbf{h} is only defined up to scale, solve for unit vector \mathbf{h}
- Solution: \mathbf{h} is the eigenvector of the matrix $\mathbf{A}^T \mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solving for homographies



RANSAC

RANdom Sample Consensus (RANSAC)

- we want to avoid the impact of outliers, so let's look for “inliers”, and use those only.

Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

RANSAC

RANSAC loop:

1. Randomly select a *seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

Keep the transformation with the largest number of inliers

RANSAC example

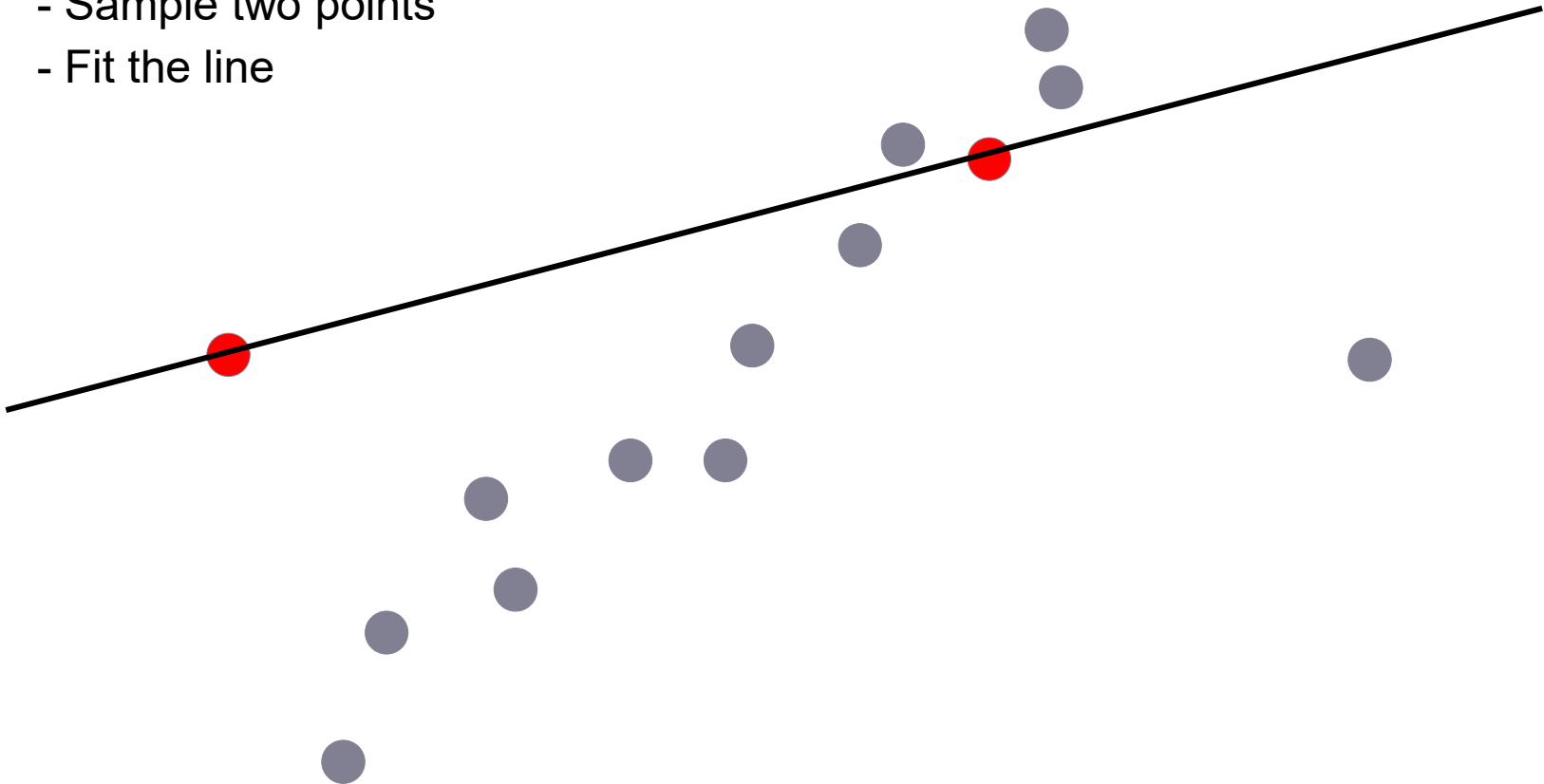
Estimate the best line



RANSAC example

Estimate the best line:

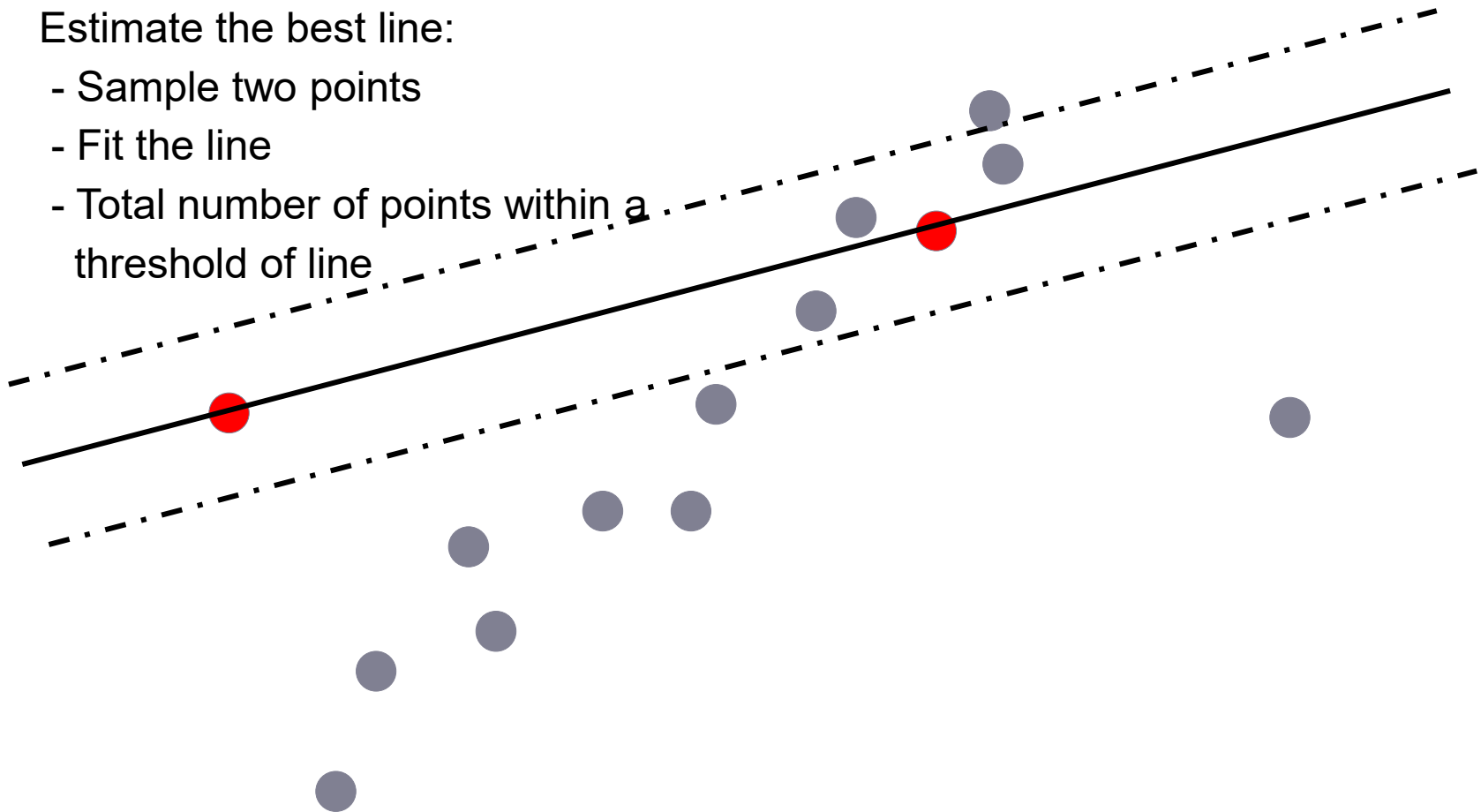
- Sample two points
- Fit the line



RANSAC example

Estimate the best line:

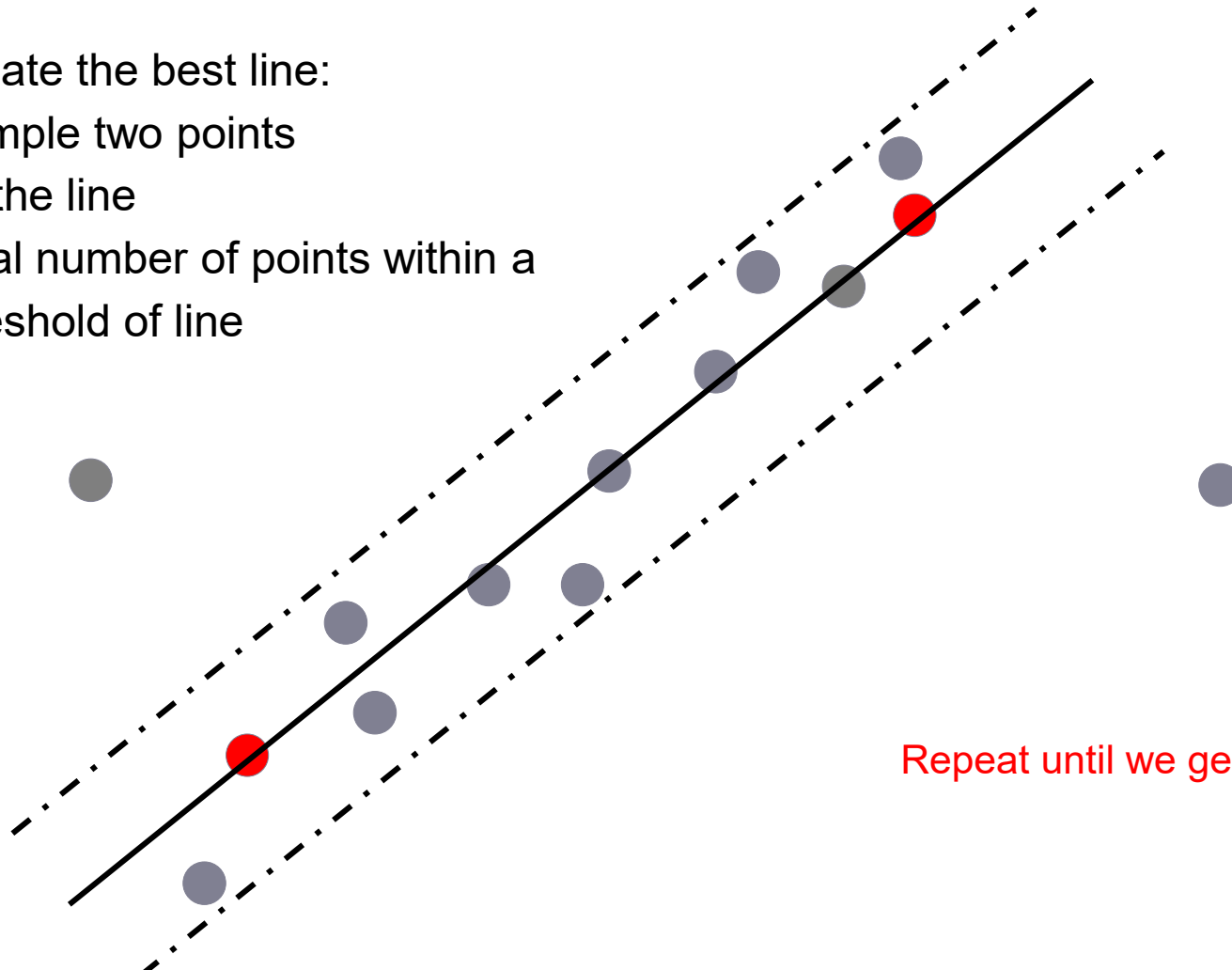
- Sample two points
- Fit the line
- Total number of points within a threshold of line



RANSAC example

Estimate the best line:

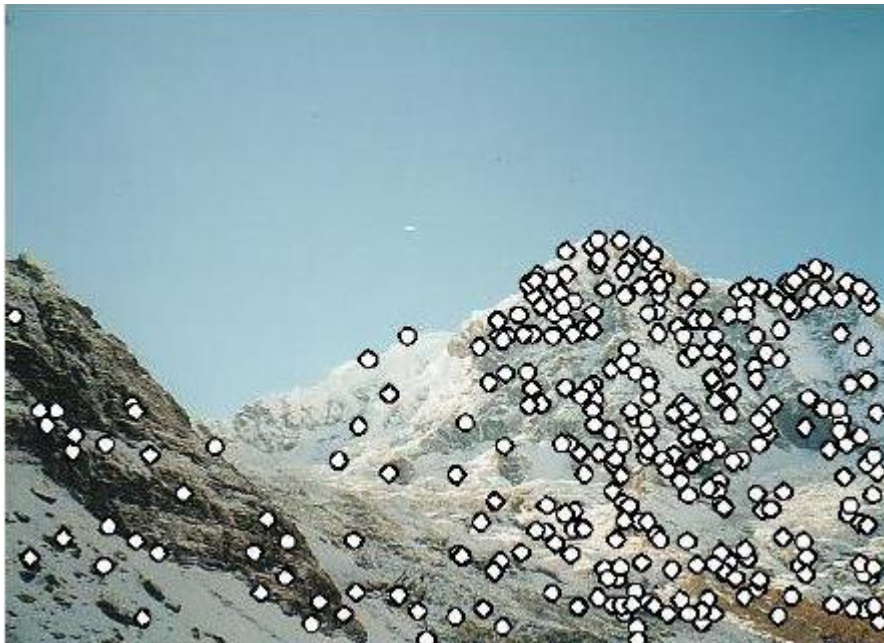
- Sample two points
- Fit the line
- Total number of points within a threshold of line



Repeat until we get a good result

Feature based alignment

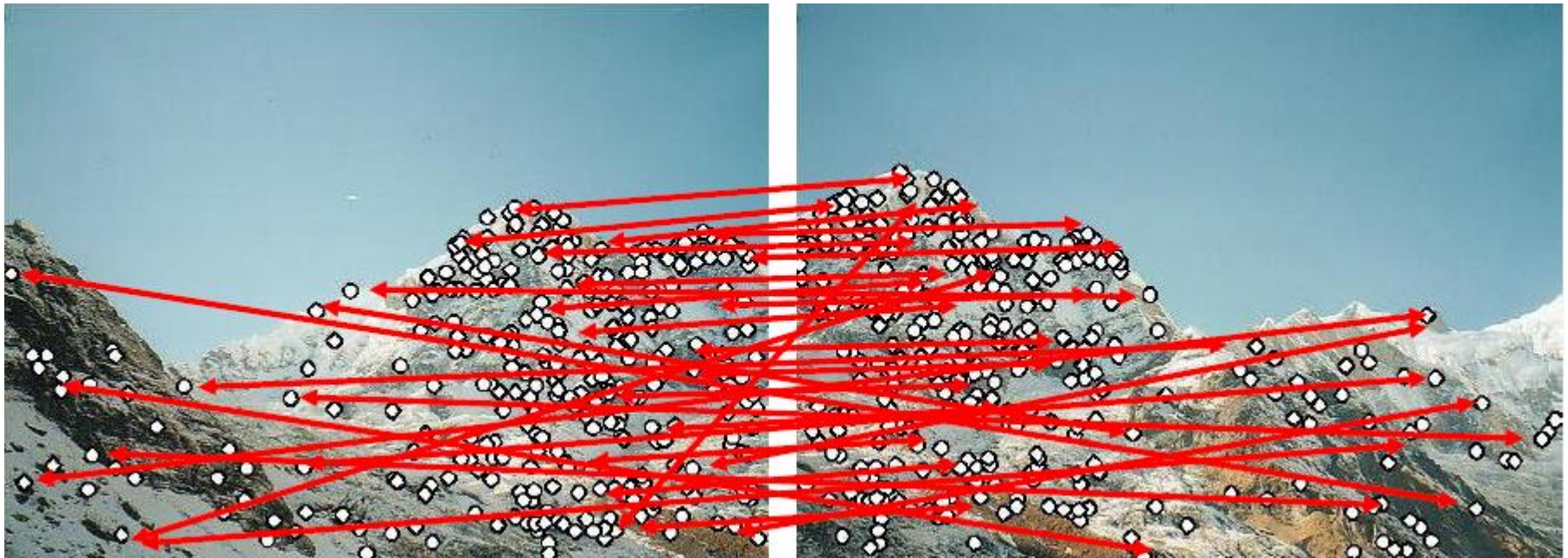
Extract features



Feature based alignment

Extract features

Computer point matches



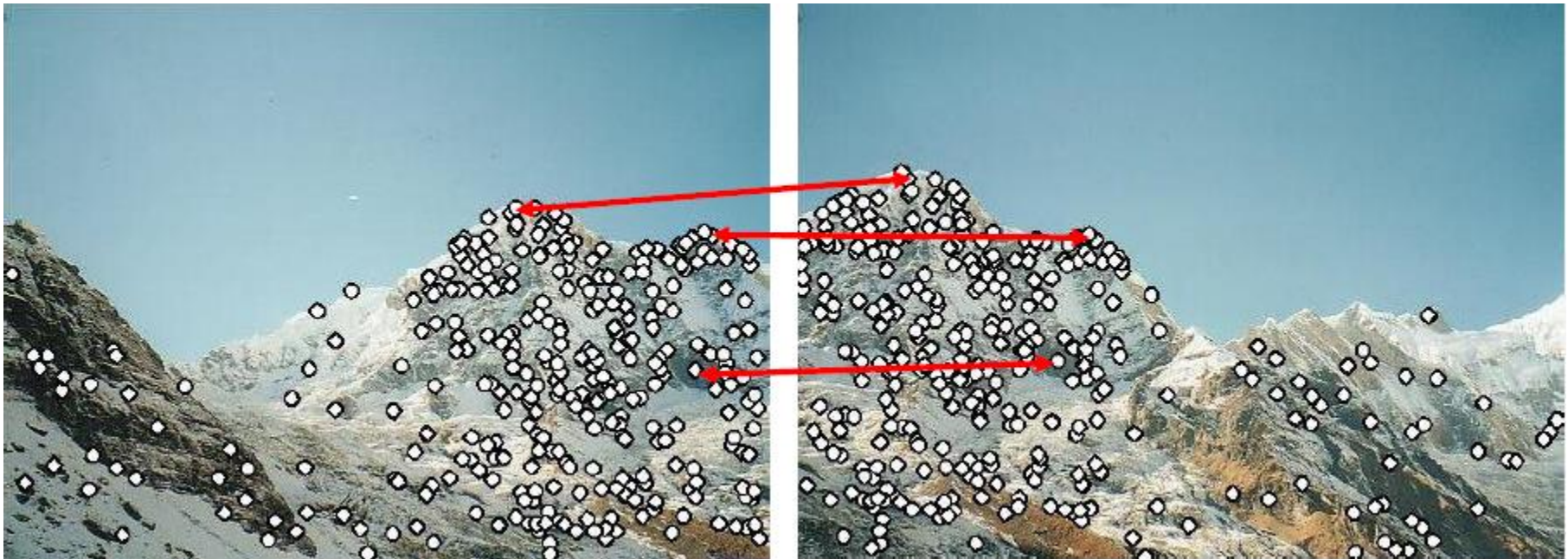
Feature based alignment

Extract features

Computer point matches

RANSAC loop:

- Hypothesize transformation T (small group of matches related to T)



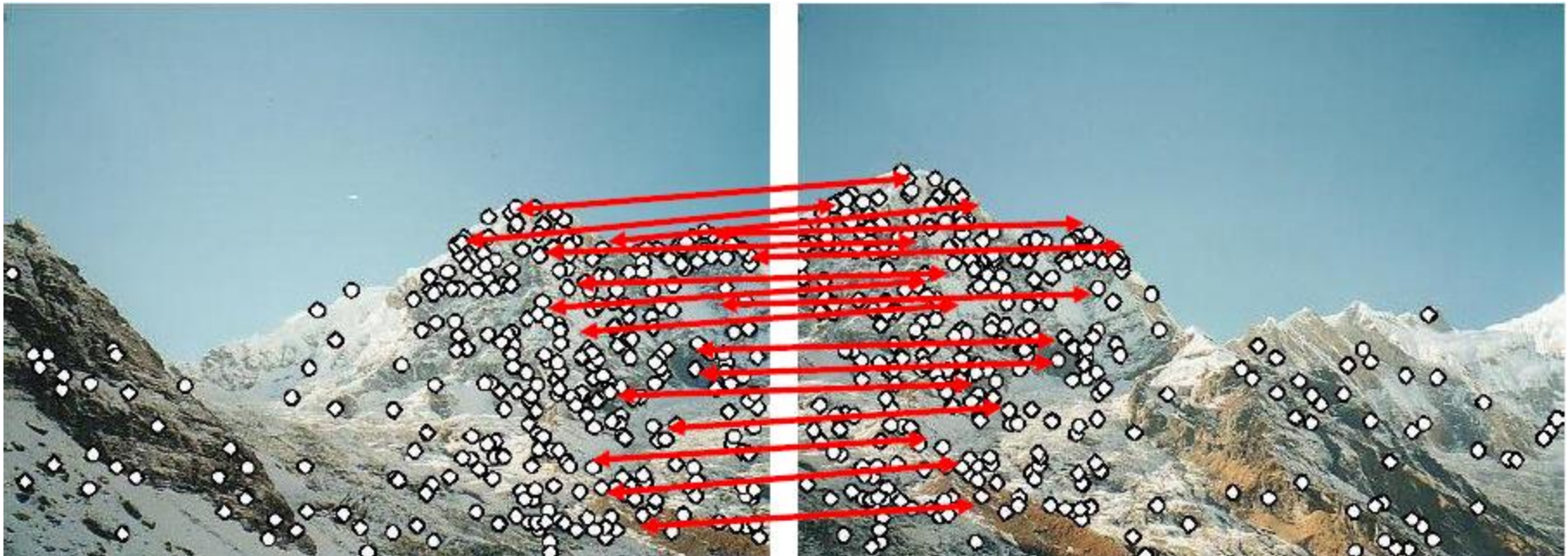
Feature based alignment

Extract features

Computer point matches

RANSAC loop:

- Hypothesize transformation T (small group of matches related to T)
- Verify the transformation (search for other matches consistent to T)



Feature based alignment

Extract features

Computer point matches

RANSAC loop:

- Hypothesize transformation T (small group of matches related to T)
- Verify the transformation (search for other matches consistent to T)



Camera calibration

Goal is to estimate the camera parameters:

- Version 1: solve for the projection matrix

$$\mathbf{X} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{P}\mathbf{X}$$

- Version 2: solve for camera parameters separately:
 - Intrinsic parameters (focal length, principle point, pixel size)
 - extrinsic parameters (rotation angles, translation)
 - radial distortion

Camera calibration

Projection:

$$\mathbf{X} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M}\mathbf{X}$$

The matrix \mathbf{M} models the effect of all parameters and can be decomposed as:

$$\mathbf{M} = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$$

Intrinsic

Projection

Rotation

Translation

Camera calibration

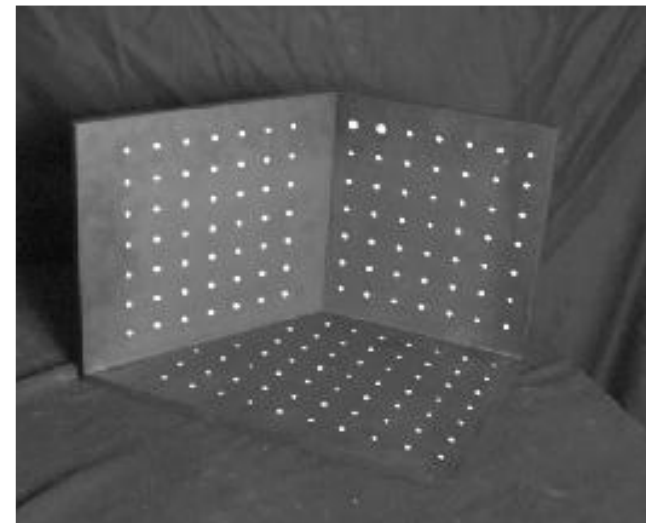
Calibration using a reference object:

- Place a known object in the scene
- Identify the location of points in the scene in the image plane (correspondences)
- Computer mapping from scene to image

Issues:

- must know the geometry accurately
- must know 3D to 2D correspondences

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$



Direct linear calibration

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}$$

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + m_{23}) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Direct linear calibration

Can solve for m_{ij} by linear least squares

- we use the eigenvector trick used in homographies

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_i X_i & -u_i Y_i & -u_i Z_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_i X_i & -v_i Y_i & -v_i Z_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Direct linear calibration

How to get from

$$\mathbf{X} = \begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{M}\mathbf{X}$$

to

$$\mathbf{M} = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3 \times 3} & \mathbf{0}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3 \times 3} & \mathbf{T}_{3 \times 1} \\ \mathbf{0}_{1 \times 3} & 1 \end{bmatrix}$$

Intrinsic

Projection

Rotation

Translation

We apply QR decomposition

$$\mathbf{M} = \mathbf{R}\mathbf{Q}$$

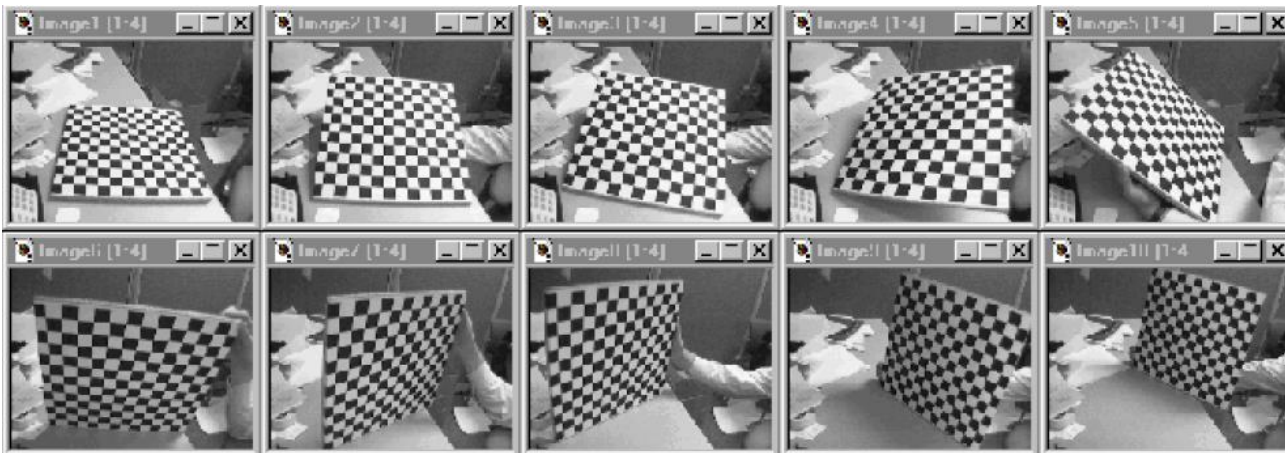
(upper triangular)
intrinsic

(orthogonal)
projection+rotation+translation

Multi-plane calibration (example)

Advantage

- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
 - Intel's OpenCV library: <http://www.intel.com/research/mrl/research/opencv/>
 - Matlab version by Jean-Yves Bouget:
http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
 - Zhengyou Zhang's web site: <http://research.microsoft.com/~zhang/Calib/>



Camera calibration

Goal is to estimate the camera parameters:

$$\mathbf{X} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi} \mathbf{X}$$

Observations

- The columns of the camera matrix $\mathbf{\Pi}$ have a geometric meaning as particular points in the image.
- Columns 1, 2, and 3 are the images (=vanishing points!) of the world coordinates axes X , Y , and Z .
- The fourth column is the image of the world origin.
- Let's see why...

Camera calibration

$$\mathbf{\Pi} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} = [\boldsymbol{\pi}_1 \quad \boldsymbol{\pi}_2 \quad \boldsymbol{\pi}_3 \quad \boldsymbol{\pi}_4]$$

$\boldsymbol{\pi}_1 = \mathbf{\Pi} [1 \ 0 \ 0 \ 0]^T = \mathbf{v}_x$ (vanishing point of the X-axis)

$\boldsymbol{\pi}_2 = \mathbf{\Pi} [0 \ 1 \ 0 \ 0]^T = \mathbf{v}_y$ (vanishing point of the Y-axis)

$\boldsymbol{\pi}_3 = \mathbf{\Pi} [0 \ 0 \ 1 \ 0]^T = \mathbf{v}_z$ (vanishing point of the Z-axis)

$\boldsymbol{\pi}_4 = \mathbf{\Pi} [0 \ 0 \ 0 \ 1]^T =$ projection of the world origin

Thus, we know a scaled version of $\mathbf{\Pi}$

$$\mathbf{\Pi} = [a \mathbf{v}_x \quad b \mathbf{v}_y \quad c \mathbf{v}_z \quad d \mathbf{o}]$$

We need to define (a,b,c,d)

Camera calibration

We assume that the camera:

- has square pixels
- has image plane parallel to the sensor plane
- has the principal point in the center of the image

Given these assumptions, we can define some of the matrices

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/f \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t'_1 \\ r_{21} & r_{22} & r_{23} & t'_2 \\ r_{31}/f & r_{32}/f & r_{33}/f & t'_3/f \end{bmatrix}$$

$$= \begin{bmatrix} a \mathbf{v}_X & b \mathbf{v}_Y & c \mathbf{v}_Z & d \mathbf{o} \end{bmatrix}$$

Rotated translation vector

$$\begin{bmatrix} t'_1 \\ t'_2 \\ t'_3 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

Camera calibration

Solving for f

$$\begin{bmatrix} v_{x1} & v_{y1} & v_{z1} & o_1 \\ v_{x2} & v_{y2} & v_{z2} & o_2 \\ v_{x3} & v_{y3} & v_{z3} & o_3 \\ V_X & V_Y & V_Z & 0 \end{bmatrix} = \begin{bmatrix} r_{11}/a & r_{12}/b & r_{13}/c & t'_1/d \\ r_{21}/a & r_{22}/b & r_{23}/c & t'_2/d \\ r_{31}/af & r_{32}/bf & r_{33}/cf & t'_3/df \end{bmatrix}$$

$$\begin{bmatrix} v_{x1} & v_{y1} & v_{z1} & o_1 \\ v_{x2} & v_{y2} & v_{z2} & o_2 \\ f v_{x3} & f v_{y3} & f v_{z3} & f o_3 \end{bmatrix} = \begin{bmatrix} r_{11}/a & r_{12}/b & r_{13}/c & t'_1/d \\ r_{21}/a & r_{22}/b & r_{23}/c & t'_2/d \\ r_{31}/a & r_{32}/b & r_{33}/c & t'_3/d \end{bmatrix}$$

orthogonal vectors

orthogonal vectors

$$v_{x1}v_{y1} + v_{x2}v_{y2} + f^2v_{x3}v_{y3} = 0 \quad \longrightarrow \quad f = \sqrt{\frac{v_{x1}v_{y1} + v_{x2}v_{y2}}{-v_{x3}v_{y3}}}$$

Camera calibration

Solving for a, b and c

$$\begin{bmatrix} \left\| \begin{matrix} v_{x1}/f \\ v_{x2}/f \\ v_{x3} \end{matrix} \right\| & v_{y1}/f & v_{z1}/f & o_1/f \\ v_{y2}/f & v_{y3} & v_{z2}/f & o_2/f \\ v_{z3} & o_3 \end{bmatrix} = \begin{bmatrix} \left\| \begin{matrix} r_{11}/a \\ r_{21}/a \\ r_{31}/a \end{matrix} \right\| & r_{12}/b & r_{13}/c & t'_1/d \\ r_{22}/b & r_{23}/c & t'_2/d \\ r_{32}/b & r_{33}/c & t'_3/d \end{bmatrix}$$

Norm = 1/a Norm = 1/a

- Divide the first two rows by f, now that it is known
- Now just find the norms of the first three columns
- Once we know a, b, and c, that also determines **R**

Camera calibration

Solving for d : we need a reference point in the scene

Image coordinates $\rightarrow \mathbf{X} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \mathbf{\Pi} \mathbf{X} \leftarrow \text{Known scene point}$

where

$$\mathbf{\Pi} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1/f \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

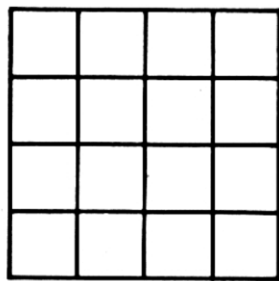
$$= \begin{bmatrix} a \mathbf{v}_X & b \mathbf{v}_Y & c \mathbf{v}_Z & d \mathbf{o} \end{bmatrix}$$

with one 3D to 2D point correspondence \rightarrow 3 equations and one unknown ($=d$)

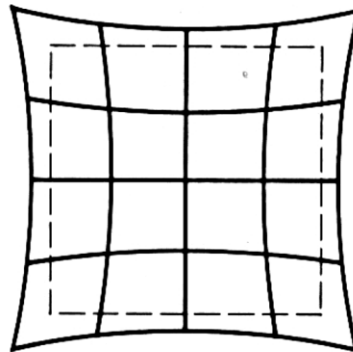
Distortion

Radial distortion of the image

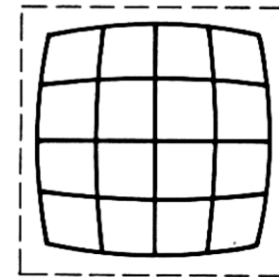
- Caused by imperfect lenses
- Deviations are most noticeable for rays that pass through the edge of the lens



No distortion



Pin cushion



Barrel

Distortion

Correction example



Helmut Dersch

Modeling distortion

To model lens distortion:

- Project $(\hat{x}, \hat{y}, \hat{z})$ to normalized image coordinates

$$x'_n = \hat{x} / \hat{z}$$

$$y'_n = \hat{y} / \hat{z}$$

$$r^2 = x_n'^2 + y_n'^2$$

- Apply radial distortion

$$x'_d = x'_n (1 + \kappa_1 r^2 + \kappa_2 r^4)$$

$$y'_d = y'_n (1 + \kappa_1 r^2 + \kappa_2 r^4)$$

- Apply focal length and translate with image center

$$x' = f x'_d + x_c$$

$$y' = f y'_d + y_c$$

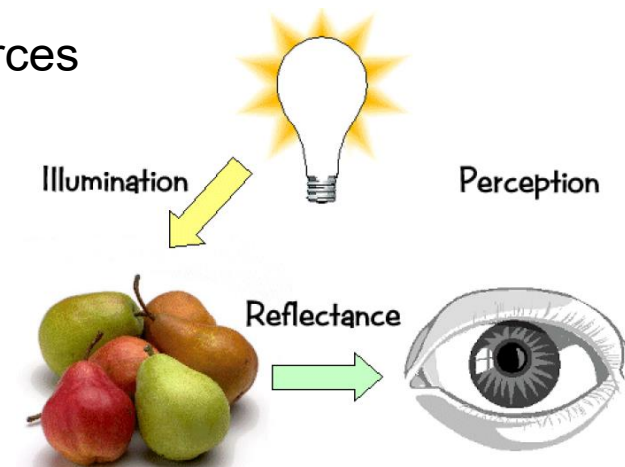
Photometrics

Basic types

- point source
- directional source
 - a point source that is infinitely far away
- area source
 - a union of point sources

More generally

- a light field can describe *any* distribution of light sources



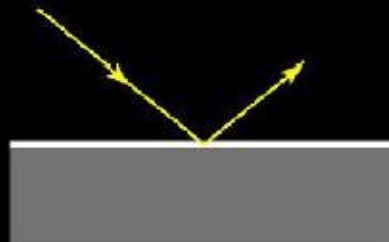
Photometrics

from Steve Marschner

Materials



conductor



conductor plus
microgeometry

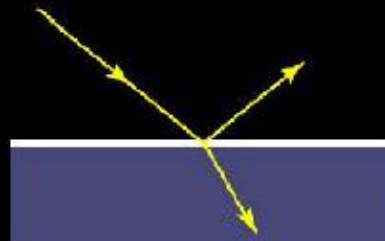


Photometrics

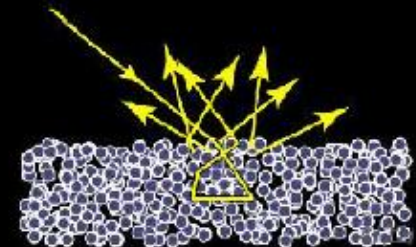
from Steve Marschner



insulator



insulator plus
microgeometry



Interaction of light and matter

What happens when a light ray hits a point on an object?

Interaction of light and matter

What happens when a light ray hits a point on an object?

- Some of the light gets absorbed (converted to other forms of energy (e.g., heat))
- Some gets transmitted through the object (possibly bent, through “refraction”)
- Some gets reflected (it could be reflected in multiple directions at once)

Interaction of light and matter

What happens when a light ray hits a point on an object?

- Some of the light gets absorbed (converted to other forms of energy (e.g., heat))
- Some gets transmitted through the object (possibly bent, through “refraction”)
- Some gets reflected (it could be reflected in multiple directions at once)

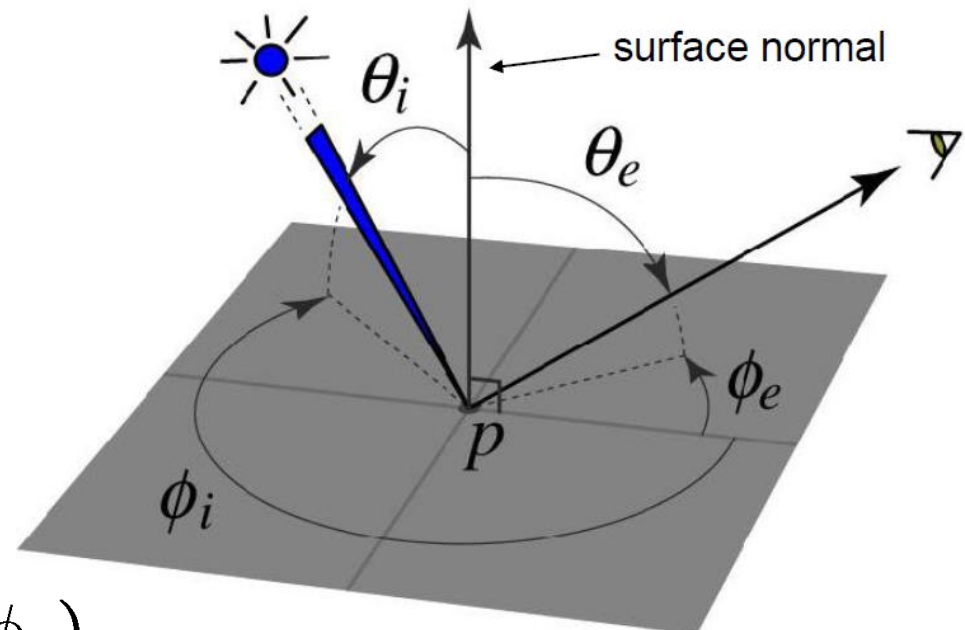
Let's consider the case of reflection in detail:

- In the most general case, a single incoming ray could be reflected in all directions.
- How can we describe the amount of light reflected in each direction?

Bidirectional Reflection Distribution Function

The Bidirectional Reflection Distribution Function (BRDF):

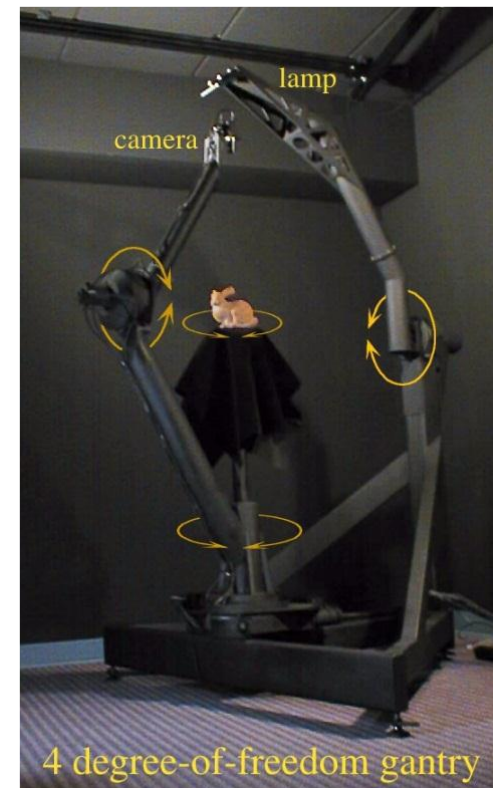
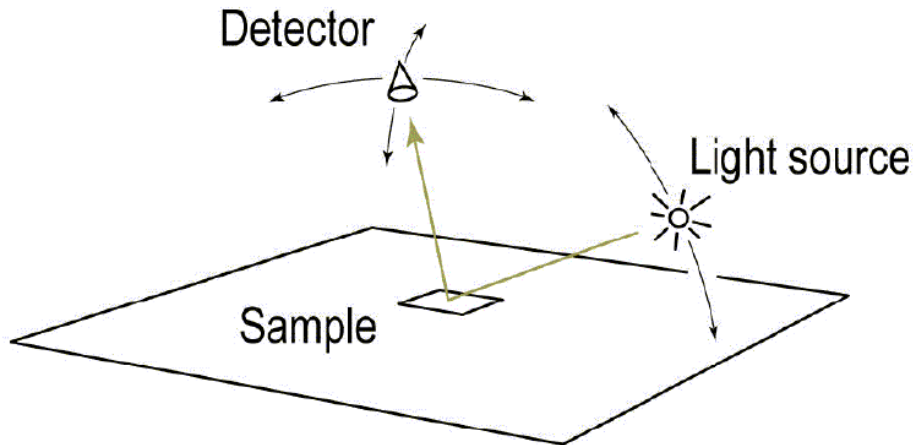
- Given an incoming ray (θ_i, ϕ_i) and outgoing ray (θ_e, ϕ_e) what proportion of the incoming light is reflected along outgoing ray?



$$\rho(\theta_i, \phi_i, \theta_e, \phi_e)$$

Measuring the BRDF

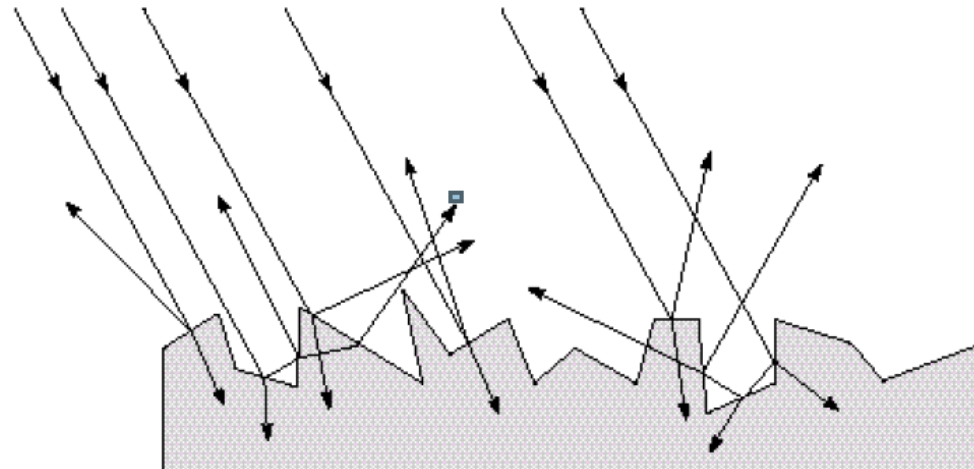
Device for capturing the BRDF by moving a camera + light source
Need careful control of illumination and environment



Diffuse reflection

Diffuse reflection

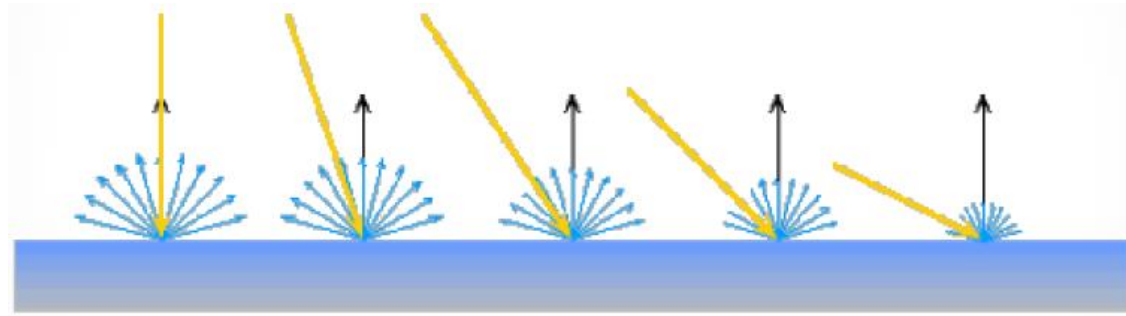
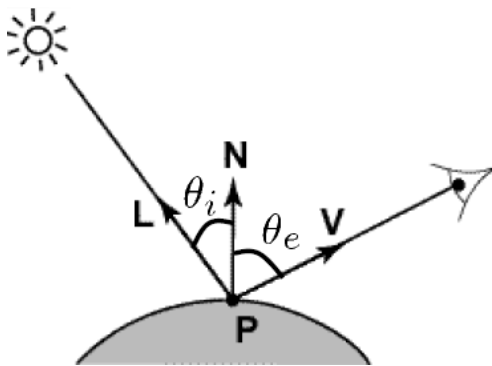
- Dull, matte surfaces like chalk or latex paint
- Microfacets scatter incoming light randomly
- Effect is that light is reflected equally in all directions



Diffuse reflection

Diffuse reflection governed by Lambert's law

- Viewed brightness does not depend on viewing direction
- Brightness does depend on direction of illumination
- This is the model most often used in computer vision



Diffuse reflection

Diffuse reflection governed by Lambert's law

- Viewed brightness does not depend on viewing direction
- Brightness does depend on direction of illumination
- This is the model most often used in computer vision

Lambert's Law: $I_e = k_d \mathbf{N} \cdot \mathbf{L} I_i$

\mathbf{L} , \mathbf{N} , \mathbf{V} are unit vectors

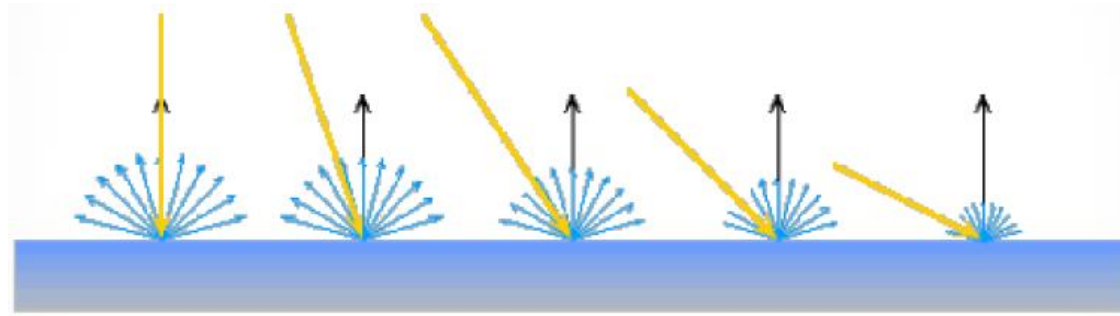
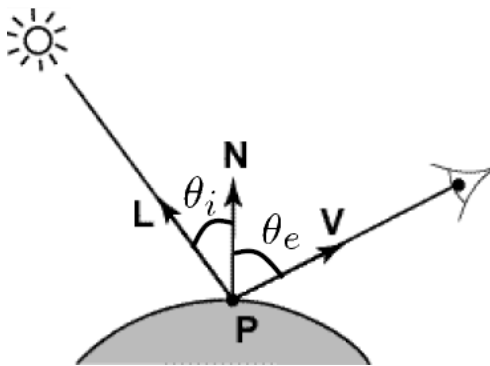
I_e is the outgoing radiance

I_i is the incoming radiance

k_d is called albedo

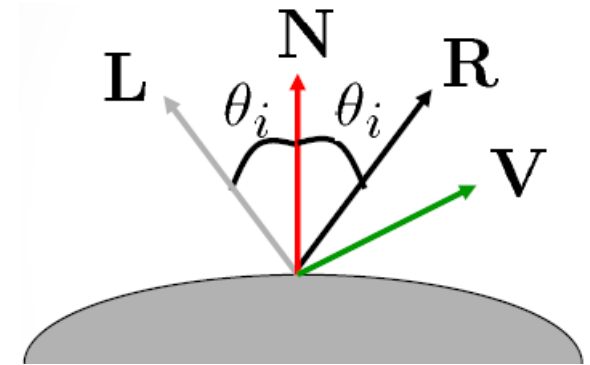
BRDF for **Lambertian surface**

$$\rho(\theta_i, \phi_i, \theta_e, \phi_e) = k_d \cos(\theta_i)$$



Specular reflection

For a perfect mirror, light is reflected about **N**

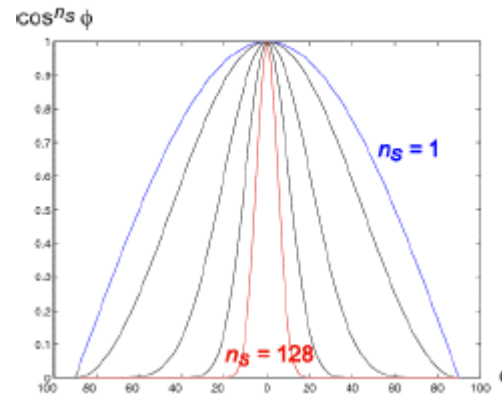


$$I_e = \begin{cases} I_i & \text{if } \mathbf{V} = \mathbf{R} \\ 0 & \text{otherwise} \end{cases}$$

Near-perfect mirrors have a highlight around **R**

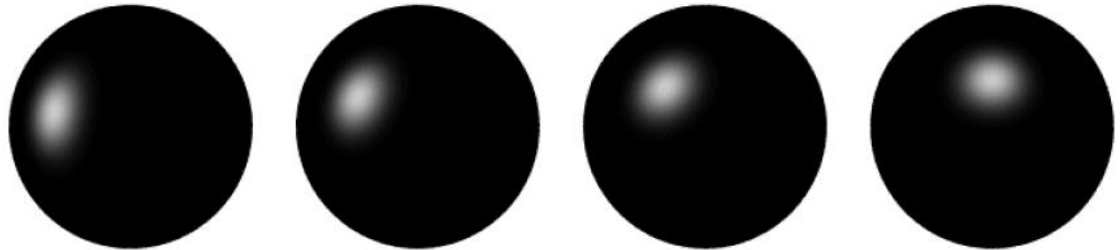
Model:

$$I_e = k_s (\mathbf{V} \cdot \mathbf{R})^{n_s} I_i$$



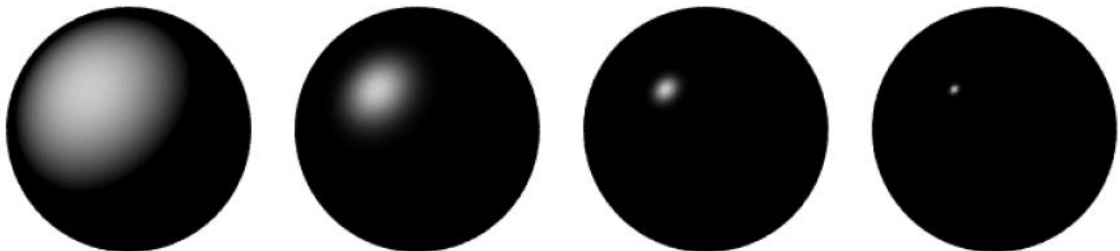
Specular reflection

Moving light source



To detect where a light source is → place a chrome sphere in the scene

Changing n_s



Phong illumination model

Approximate reflectance using three terms:

- Diffusion term
- Specular term
- Ambient term (to compensate for inter-reflected light)

\mathbf{L} , \mathbf{N} , \mathbf{V} are unit vectors

I_e is the outgoing radiance

I_i is the incoming radiance

I_a is the ambient light

k_a is the ambient light reflectance factor

$(x)_+ = \max(x, 0)$

$$I_e = k_a I_a + I_i \left[k_d (\mathbf{N} \cdot \mathbf{L})_+ + k_s (\mathbf{V} \cdot \mathbf{R})_+^{n_s} \right]$$

