Projective Geometry

CSE 6367: Computer Vision

Instructor: William J. Beksi

Introduction

- We'll study the geometry of projective transformations of the plane

Introduction

- We'll study the geometry of projective transformations of the plane
- These transformations model the geometric distortion which arises when a plane is imaged by a perspective camera

Introduction

- We'll study the geometry of projective transformations of the plane
- These transformations model the geometric distortion which arises when a plane is imaged by a perspective camera
- Under perspective imaging, certain geometric properties are preserved (e.g. collinearity) while others are not (e.g. parallel lines)

Introduction

- We'll study the geometry of projective transformations of the plane
- These transformations model the geometric distortion which arises when a plane is imaged by a perspective camera
- Under perspective imaging, certain geometric properties are preserved (e.g. collinearity) while others are not (e.g. parallel lines)
- **Projective geometry** models this imaging and also provides a mathematical representation appropriate for computations

Projective Geometry

- Projective geometry arises in several visual computing domains, in particular computer vision modeling and computer graphics

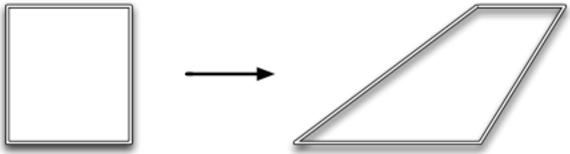
Projective Geometry

- Projective geometry arises in several visual computing domains, in particular computer vision modeling and computer graphics
- A fundamental aspect is the fact that objects at infinity can be represented and manipulated with projective geometry (this is in contrast to Euclidean geometry)

Projective Geometry

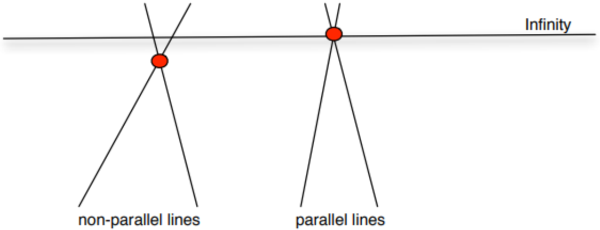
- Projective geometry arises in several visual computing domains, in particular computer vision modeling and computer graphics
- A fundamental aspect is the fact that objects at infinity can be represented and manipulated with projective geometry (this is in contrast to Euclidean geometry)
- This allows perspective deformations to be represented as projective transformations

Perspective Deformation



- An example of perspective deformation or 2D projective transformation

Line Intersections



- Line intersections in a projective space

Euclidean vs. Projective Geometry

- Euclidean geometry can be difficult to use in algorithms, particularly in non-generic situations (e.g. two parallel lines never intersect) that must be identified

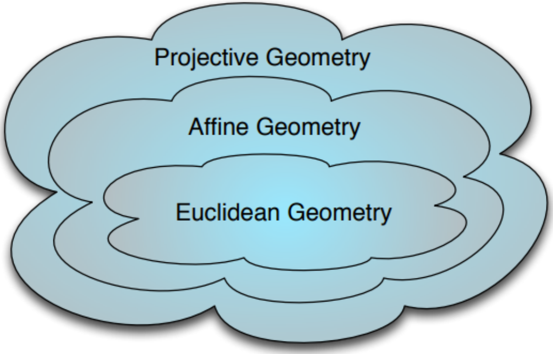
Euclidean vs. Projective Geometry

- Euclidean geometry can be difficult to use in algorithms, particularly in non-generic situations (e.g. two parallel lines never intersect) that must be identified
- In contrast, projective geometry generalizes several definitions and properties (e.g. two lines always intersect)

Euclidean vs. Projective Geometry

- Euclidean geometry can be difficult to use in algorithms, particularly in non-generic situations (e.g. two parallel lines never intersect) that must be identified
- In contrast, projective geometry generalizes several definitions and properties (e.g. two lines always intersect)
- It allows us to represent any transformation that preserves coincidence relationships in a matrix form (e.g. perspective projections) which is easier to use in computer programs

Geometry Hierarchy



Homogeneous Representation of Lines

- A line in the plane is represented by $ax + by + c = 0$ where different choices of a, b, and c give rise to different lines

Homogeneous Representation of Lines

- A line in the plane is represented by $ax + by + c = 0$ where different choices of a, b, and c give rise to different lines
- We represent a line by the vector $[a, b, c]^T$

Homogeneous Representation of Lines

- The correspondence between lines and vectors is not one-to-one since $ax + by + c = 0$ and $(ka)x + (kb)y + (kc) = 0$ are the same for any nonzero constant k

Homogeneous Representation of Lines

- The correspondence between lines and vectors is not one-to-one since $ax + by + c = 0$ and $(ka)x + (kb)y + (kc) = 0$ are the same for any nonzero constant k
- Thus, the vectors $[a, b, c]^T$ and $k[a, b, c]^T$ represent the same line for any nonzero k

Homogeneous Representation of Lines

- The correspondence between lines and vectors is not one-to-one since $ax + by + c = 0$ and $(ka)x + (kb)y + (kc) = 0$ are the same for any nonzero constant k
- Thus, the vectors $[a, b, c]^T$ and $k[a, b, c]^T$ represent the same line for any nonzero k
- In fact, two such vectors related by an overall scaling are considered as being equivalent

Homogeneous Representation of Lines

- An equivalence class of vectors under this equivalence relationship is known as a **homogeneous** vector

Homogeneous Representation of Lines

- An equivalence class of vectors under this equivalence relationship is known as a **homogeneous** vector
- Any particular vector $[a, b, c]^T$ is representative of the equivalence class

Homogeneous Representation of Lines

- An equivalence class of vectors under this equivalence relationship is known as a **homogeneous** vector
- Any particular vector $[a, b, c]^T$ is representative of the equivalence class
- The set of all equivalence classes of vectors in $\mathbb{R}^3 - [0, 0, 0]^T$ forms the **projective space** $\mathbb{P}^2$

Homogeneous Representation of Points

- A point $\mathbf{x} = [x, y]^T$ lies on the line $\mathbf{l} = [a, b, c]^T$ iff $ax + by + c = 0$

Homogeneous Representation of Points

- A point $\mathbf{x} = [x, y]^T$ lies on the line $\mathbf{l} = [a, b, c]^T$ iff $ax + by + c = 0$
- This can be written in terms of an inner product of vectors representing the point as $[x, y, 1][a, b, c]^T = [x, y, 1]\mathbf{l} = 0$

Homogeneous Representation of Points

- A point $\mathbf{x} = [x, y]^T$ lies on the line $\mathbf{l} = [a, b, c]^T$ iff $ax + by + c = 0$
- This can be written in terms of an inner product of vectors representing the point as $[x, y, 1][a, b, c]^T = [x, y, 1]\mathbf{l} = 0$
- An arbitrary homogeneous vector representative of a point is of the form $\mathbf{x} = [x_1, x_2, x_3]^T$ which represents the point $[x_1/x_3, x_2/x_3]^T$ in $\mathbb{R}^2$

Points on Lines

- The point $\mathbf{x}$ lies on the line $\mathbf{l}$ iff $\mathbf{x}^T\mathbf{l} = 0$

Points on Lines

- The point $\mathbf{x}$ lies on the line $\mathbf{l}$ iff $\mathbf{x}^T\mathbf{l} = 0$
- Note that the expression $\mathbf{x}^T\mathbf{l} = \mathbf{l}^T\mathbf{x} = \mathbf{x}.\mathbf{l}$ is just the inner or scalar product of the two vectors $\mathbf{l}$ and $\mathbf{x}$

Points on Lines

- The point $\mathbf{x}$ lies on the line $\mathbf{l}$ iff $\mathbf{x}^T\mathbf{l} = 0$
- Note that the expression $\mathbf{x}^T\mathbf{l} = \mathbf{l}^T\mathbf{x} = \mathbf{x}.\mathbf{l}$ is just the inner or scalar product of the two vectors $\mathbf{l}$ and $\mathbf{x}$
- We distinguish between the **homogeneous coordinates** $\mathbf{x} = [x_1, x_2, x_3]^T$ of a point which is a 3-vector, and the **inhomogeneous coordinates** $[x, y]^T$ which is a 2-vector

Intersection of Lines

- Given two lines $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}' = [a', b', c']^T$, we can find their intersection by defining the vector $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$

Intersection of Lines

- Given two lines $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}' = [a', b', c']^T$, we can find their intersection by defining the vector $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$
- From the triple scalar product identity $\mathbf{l}.(\mathbf{l} \times \mathbf{l}') = \mathbf{l}'.(\mathbf{l} \times \mathbf{l}') = 0$, we see that $\mathbf{l}^T\mathbf{x} = \mathbf{l}'^T\mathbf{x} = 0$

Intersection of Lines

- Given two lines $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}' = [a', b', c']^T$, we can find their intersection by defining the vector $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$
- From the triple scalar product identity $\mathbf{l}.(\mathbf{l} \times \mathbf{l}') = \mathbf{l}'.(\mathbf{l} \times \mathbf{l}') = 0$, we see that $\mathbf{l}^T\mathbf{x} = \mathbf{l}'^T\mathbf{x} = 0$
- Thus, the intersection of two lines $\mathbf{l}$ and $\mathbf{l}'$ is the point $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$

Example: Determining the Intersection of Lines

- Consider the problem of determining the intersection of the lines $x = 1$ and $y = 1$

Example: Determining the Intersection of Lines

- Consider the problem of determining the intersection of the lines $x = 1$ and $y = 1$
- The line $x = 1$ is equivalent to $-1x + 1 = 0$ and has the homogeneous representation $\mathbf{l} = [-1, 0, 1]^T$ while the line $y = 1$ is equivalent to $-1y + 1 = 0$ and has the homogeneous representation $\mathbf{l}' = [0, -1, 1]^T$

Example: Determining the Intersection of Lines

- Consider the problem of determining the intersection of the lines $x = 1$ and $y = 1$
- The line $x = 1$ is equivalent to $-1x + 1 = 0$ and has the homogeneous representation $\mathbf{l} = [-1, 0, 1]^T$ while the line $y = 1$ is equivalent to $-1y + 1 = 0$ and has the homogeneous representation $\mathbf{l}' = [0, -1, 1]^T$
- Therefore, the intersection point is

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ -1 & 0 & 1 \\ 0 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

which is the inhomogeneous point $[1, 1]^T$ as required

Intersection of Parallel Lines

- Consider two lines $ax + by + c = 0$ and $ax + by + c' = 0$ represented by the vectors $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}' = [a, b, c']^T$.

Intersection of Parallel Lines

- Consider two lines $ax + by + c = 0$ and $ax + by + c' = 0$ represented by the vectors $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}' = [a, b, c']^T$.
- The intersection is $\mathbf{l} \times \mathbf{l}' = (c' - c)[b, -a, 0]^T$ and ignoring the scale factor $(c' - c)$ this is the point $[b, -a, 0]^T$

Intersection of Parallel Lines

- Consider two lines $ax + by + c = 0$ and $ax + by + c' = 0$ represented by the vectors $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}' = [a, b, c']^T$.
- The intersection is $\mathbf{l} \times \mathbf{l}' = (c' - c)[b, -a, 0]^T$ and ignoring the scale factor $(c' - c)$ this is the point $[b, -a, 0]^T$
- If we attempt to find the inhomogeneous representation we get $[b/0, -a/0]^T$ which makes no sense

Intersection of Parallel Lines

- Consider two lines $ax + by + c = 0$ and $ax + by + c' = 0$ represented by the vectors $\mathbf{l} = [a, b, c]^T$ and $\mathbf{l}' = [a, b, c']^T$.
- The intersection is $\mathbf{l} \times \mathbf{l}' = (c' - c)[b, -a, 0]^T$ and ignoring the scale factor $(c' - c)$ this is the point $[b, -a, 0]^T$
- If we attempt to find the inhomogeneous representation we get $[b/0, -a/0]^T$ which makes no sense
- In general, points with homogeneous coordinates $[x, y, 0]^T$ do not correspond to any finite point in $\mathbb{R}^2$ which agrees with the idea that parallel lines meet at infinity

Ideal Points and the Line at Infinity

- Homogeneous vectors $\mathbf{x} = [x_1, x_2, x_3]^T$ such that $x_3 \neq 0$ correspond to finite points in $\mathbb{R}^2$

Ideal Points and the Line at Infinity

- Homogeneous vectors $\mathbf{x} = [x_1, x_2, x_3]^T$ such that $x_3 \neq 0$ correspond to finite points in $\mathbb{R}^2$
- One may augment $\mathbb{R}^2$ by adding points with last coordinate $x_3 = 0$

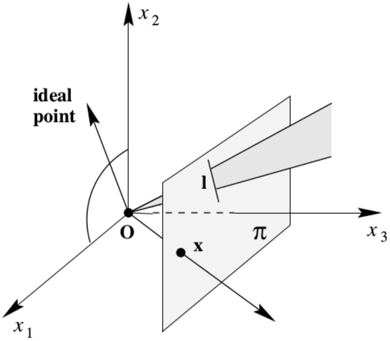
Ideal Points and the Line at Infinity

- Homogeneous vectors $\mathbf{x} = [x_1, x_2, x_3]^T$ such that $x_3 \neq 0$ correspond to finite points in $\mathbb{R}^2$
- One may augment $\mathbb{R}^2$ by adding points with last coordinate $x_3 = 0$
- These points are known as **ideal points** or **points at infinity**

Ideal Points and the Line at Infinity

- Homogeneous vectors $\mathbf{x} = [x_1, x_2, x_3]^T$ such that $x_3 \neq 0$ correspond to finite points in $\mathbb{R}^2$
- One may augment $\mathbb{R}^2$ by adding points with last coordinate $x_3 = 0$
- These points are known as **ideal points** or **points at infinity**
- The set of all ideal points $[x_1, x_2, 0]^T$ lies on a single line, the **line at infinity**, denoted by $\mathbf{l}_\infty = [0, 0, 1]^T$

A Model of the Projective Plane



- Points and lines of $\mathbb{P}^2$ are represented by rays and planes, respectively, through the origin in $\mathbb{R}^3$
- Lines lying in the x_1x_2-plane represent ideal points and the x_1x_2-plane represents $\mathbf{l}_\infty$

Conics

- A **conic** is a curve described by a second-degree equation in the plane, the equation in inhomogeneous coordinates is $ax^2 + bxy + cy^2 + dx + ey + f = 0$

Conics

- A **conic** is a curve described by a second-degree equation in the plane, the equation in inhomogeneous coordinates is $ax^2 + bxy + cy^2 + dx + ey + f = 0$
- Homogenizing this by the replacements $x \rightarrow x_1/x_3$, $y \rightarrow x_2/x_3$ gives

$$ax_1^2 + bx_1x_2 + cx_2^2 + dx_1x_3 + ex_2x_3 + fx_3^2 = 0$$

or in matrix form

$$\mathbf{x}^T C \mathbf{x} = 0$$

where the conic coefficient matrix C is given by

$$\begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$$

Five Points Determine a Conic

- Each point x_i places one constraint on the conic coefficients since if the conic passes through $[x_i, y_i]$ then

$$ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f = 0$$

which can be written as

$$\begin{bmatrix} x_i^2 & x_iy_i & y_i^2 & x_i & y_i & 1 \end{bmatrix} \mathbf{c} = 0$$

where $\mathbf{c} = [a, b, c, d, e, f]^T$ is the conic C

Five Points Determine a Conic

- Each point x_i places one constraint on the conic coefficients since if the conic passes through $[x_i, y_i]$ then

$$ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f = 0$$

which can be written as

$$\begin{bmatrix} x_i^2 & x_iy_i & y_i^2 & x_i & y_i & 1 \end{bmatrix} \mathbf{c} = 0$$

where $\mathbf{c} = [a, b, c, d, e, f]^T$ is the conic C

- Stacking the constraints from five points we obtain

$$\begin{bmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ x_3^2 & x_3y_3 & y_3^2 & x_3 & y_3 & 1 \\ x_4^2 & x_4y_4 & y_4^2 & x_4 & y_4 & 1 \\ x_5^2 & x_5y_5 & y_5^2 & x_5 & y_5 & 1 \end{bmatrix} \mathbf{c} = 0$$

Tangent Lines to Conics

- The line $\mathbf{l}$ tangent to C at a point $\mathbf{x}$ on C is given by $\mathbf{l} = C\mathbf{x}$

  Proof: The line $\mathbf{l} = C\mathbf{x}$ passes through $\mathbf{x}$, since $\mathbf{l}^T\mathbf{x} = \mathbf{x}^T C\mathbf{x} = 0$. If $\mathbf{l}$ has one-point contact with the conic, then it is tangent, and we are done. Otherwise suppose that $\mathbf{l}$ meets the conic in another point $\mathbf{y}$. Then $\mathbf{y}^T C\mathbf{y} = 0$ and $\mathbf{x}^T C\mathbf{y} = \mathbf{l}^T\mathbf{y} = 0$. From this it follows that $[\mathbf{x} + \alpha\mathbf{y}]^T C[\mathbf{x} + \alpha\mathbf{y}] = 0$ for all α, which means that the whole line $\mathbf{l} = C\mathbf{x}$ joining $\mathbf{x}$ and $\mathbf{y}$ lies on the conic C, which is therefore degenerate.

Dual Conics

- The conic C is more properly termed a **point** conic

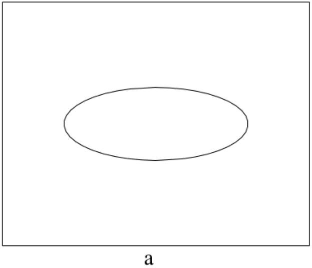
Dual Conics

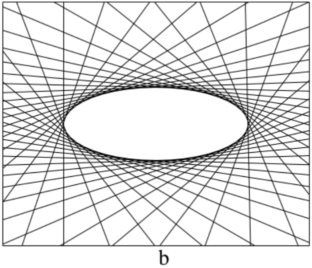
- The conic C is more properly termed a **point** conic
- There is also a conic which defines an equation on lines called the **dual** (or **line**) conic which we denote by C^*

Dual Conics

- The conic C is more properly termed a **point** conic
- There is also a conic which defines an equation on lines called the **dual** (or **line**) conic which we denote by C^*
- A line $\mathbf{l}$ tangent to the conic C satisfies $\mathbf{l}^T C^* \mathbf{l} = 0$

Point and Line Conics





- (a) Points $\mathbf{x}$ satisfying $\mathbf{x}^T C\mathbf{x} = 0$ lie on a point conic; (b) Lines $\mathbf{l}$ satisfying $\mathbf{l}^T C\mathbf{l} = 0$ are tangent to the point conic C

Degenerate Conics

- If the matrix C is not full rank, then the conic is termed degenerate

Degenerate Conics

- If the matrix C is not full rank, then the conic is termed degenerate
- Degenerate point conics include two lines (rank 2), and a repeated line (rank 1)

Degenerate Conics

- If the matrix C is not full rank, then the conic is termed degenerate
- Degenerate point conics include two lines (rank 2), and a repeated line (rank 1)
- Degenerate line conics include two points (rank 2), and a repeated point (rank 1)

Degenerate Conics

- If the matrix C is not full rank, then the conic is termed degenerate
- Degenerate point conics include two lines (rank 2), and a repeated line (rank 1)
- Degenerate line conics include two points (rank 2), and a repeated point (rank 1)
- Example: The line conic $C^* = \mathbf{x}\mathbf{y}^T + \mathbf{y}\mathbf{x}^T$ has rank 2 and consists of lines passing through either of the two points $\mathbf{x}$ and $\mathbf{y}$

Projectivity

- In 2D projective geometry we the study the properties of the projective plane $\mathbb{P}^2$ that are invariant under a group of transformations known as **projectivities**

Projectivity

- In 2D projective geometry we the study the properties of the projective plane $\mathbb{P}^2$ that are invariant under a group of transformations known as **projectivities**
- A projectivity is an invertible mapping h from $\mathbb{P}^2$ to itself such that three points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ lie on the same line iff $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$, and $h(\mathbf{x}_3)$ do

Projectivity

- In 2D projective geometry we the study the properties of the projective plane $\mathbb{P}^2$ that are invariant under a group of transformations known as **projectivities**
- A projectivity is an invertible mapping h from $\mathbb{P}^2$ to itself such that three points $\mathbf{x}_1$, $\mathbf{x}_2$, and $\mathbf{x}_3$ lie on the same line iff $h(\mathbf{x}_1)$, $h(\mathbf{x}_2)$, and $h(\mathbf{x}_3)$ do
- A mapping $h : \mathbb{P}^2 \rightarrow \mathbb{P}^2$ is a projectivity iff there exists a non-singular 3×3 matrix H such that for any point in $\mathbb{P}^2$ represented by a vector $\mathbf{x}$ it is true that $h(\mathbf{x}) = H\mathbf{x}$

Projective Transformation

- A planar projective transformation is a linear transformation on homogeneous 3-vectors by a non-singular 3×3 matrix

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

or more compactly, $\mathbf{x}' = H\mathbf{x}$

Projective Transformation

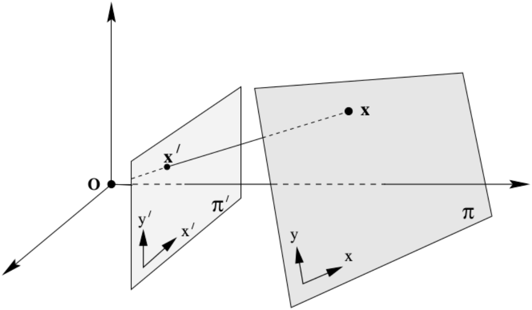
- A planar projective transformation is a linear transformation on homogeneous 3-vectors by a non-singular 3×3 matrix

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

  or more compactly, $\mathbf{x}' = H\mathbf{x}$

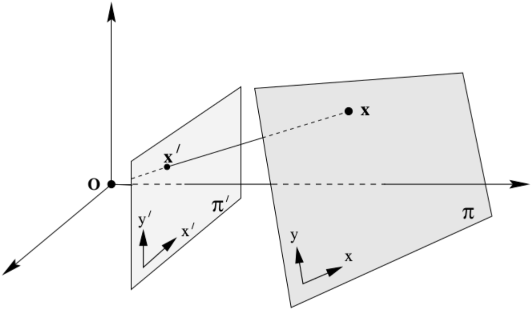
- Note that H may be changed by multiplication by an arbitrary non-zero scale factor without altering the projective transformation

Mappings Between Planes



- Projection along rays through a common point defines a mapping from one plane to another

Mappings Between Planes



- Projection along rays through a common point defines a mapping from one plane to another
- If a coordinate system is defined in each plane and points are represented in homogeneous coordinates, then the **central projection** mapping may be expressed by $\mathbf{x}' = H\mathbf{x}$

Example: Removing Projective Distortion


a


b

- (a) The original image with perspective distortion - the lines of the windows clearly converge at a finite point; (b) Synthesized frontal orthogonal view of the front wall

Example: Removing Projective Distortion

- We begin by letting the inhomogeneous coordinates of a pair of matching points $\mathbf{x}$ and $\mathbf{x}'$ in the world and image plane be $[x, y]$ and $[x', y']$ respectively

Example: Removing Projective Distortion

- We begin by letting the inhomogeneous coordinates of a pair of matching points $\mathbf{x}$ and $\mathbf{x}'$ in the world and image plane be $[x, y]$ and $[x', y']$ respectively
- The projective transformation can be written in inhomogeneous form as

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

Example: Removing Projective Distortion

- We begin by letting the inhomogeneous coordinates of a pair of matching points $\mathbf{x}$ and $\mathbf{x}'$ in the world and image plane be $[x, y]$ and $[x', y']$ respectively
- The projective transformation can be written in inhomogeneous form as

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

- Each point correspondence generates two equations for the elements of H

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13}$$
$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23}$$

Example: Removing Projective Distortion

- These equations are *linear* in the elements of H

Example: Removing Projective Distortion

- These equations are *linear* in the elements of H
- Four point correspondences lead to eight such linear equations in the entries of H, which are sufficient to solve for H

Example: Removing Projective Distortion

- These equations are *linear* in the elements of H
- Four point correspondences lead to eight such linear equations in the entries of H, which are sufficient to solve for H
- The only restriction is that the four points must be in "general position" (i.e. no three points are collinear)

Example: Removing Projective Distortion

- These equations are *linear* in the elements of H
- Four point correspondences lead to eight such linear equations in the entries of H, which are sufficient to solve for H
- The only restriction is that the four points must be in "general position" (i.e. no three points are collinear)
- The inverse of the transformation H is then applied to the whole image to undo the effect of perspective distortion on the selected plane

Hierarchy of Transformations

- Projective transformations form a group of invertible $n \times n$ matrices called the **projective linear group**, or $PL(n)$ (in the case of projective transformations of the plane $n = 3$)

Hierarchy of Transformations

- Projective transformations form a group of invertible $n \times n$ matrices called the **projective linear group**, or $PL(n)$ (in the case of projective transformations of the plane $n = 3$)
- Important subgroups of $PL(3)$ include the **affine group** and the **Euclidean group**

Hierarchy of Transformations

- Projective transformations form a group of invertible $n \times n$ matrices called the **projective linear group**, or $PL(n)$ (in the case of projective transformations of the plane $n = 3$)
- Important subgroups of $PL(3)$ include the **affine group** and the **Euclidean group**
- Transformations can also be described in terms of **invariants**, i.e. those elements or quantities that are preserved by a particular transformation

Isometries

- **Isometries** are transformations of the plane $\mathbb{R}^2$ that preserve Euclidean distance and are represented as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \epsilon \cos\theta & -\sin\theta & t_x \\ \epsilon \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

where $\epsilon = \pm 1$

Isometries

- **Isometries** are transformations of the plane $\mathbb{R}^2$ that preserve Euclidean distance and are represented as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \epsilon \cos\theta & -\sin\theta & t_x \\ \epsilon \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  where $\epsilon = \pm 1$
- Euclidean transformations model the motion of a rigid object and are the most important isometries in practice

Isometries

- **Isometries** are transformations of the plane $\mathbb{R}^2$ that preserve Euclidean distance and are represented as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \epsilon \cos\theta & -\sin\theta & t_x \\ \epsilon \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

  where $\epsilon = \pm 1$
- Euclidean transformations model the motion of a rigid object and are the most important isometries in practice
- A planar Euclidean transformation can be written more concisely in block form as

$$\mathbf{x}' = H_E \mathbf{x} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

Similarity Transformations

- A **similarity transformation** is an isometry composed with an isotropic scaling and has the following matrix representation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or more concisely in block form

$$\mathbf{x}' = H_S\mathbf{x} = \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

where the scalar s represents the isotropic scaling

Affine Transformations

- An **affine transformation** is a non-singular linear transformation followed by a translation, and has the matrix representation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

or in block form

$$\mathbf{x}' = H_A \mathbf{x} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x}$$

Affine Transformations

- A helpful way to understand the geometric effects of the linear component A of an affine transformation is as the composition of two fundamental transformations, namely rotations and non-isotropic scalings

Affine Transformations

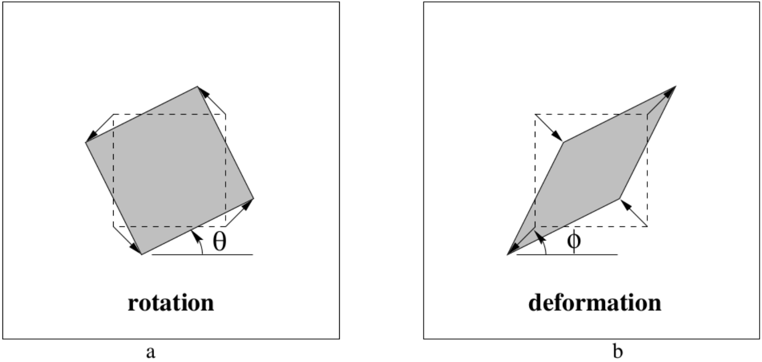
- A helpful way to understand the geometric effects of the linear component A of an affine transformation is as the composition of two fundamental transformations, namely rotations and non-isotropic scalings
- The affine matrix A can always be decomposed as

$$A = R(\theta)R(-\phi)DR(\phi)$$

where $R(\theta)$ and $R(\phi)$ are rotations by θ and ϕ respectively, and D is a diagonal matrix

$$D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Affine Transformation Distortions



- (a) Rotation by $R(\theta)$; (b) A deformation $R(-\phi)DR(\phi)$

Decomposition of a Projective Transformations

- A projective transformation can be decomposed into a chain of transformations, where each matrix represents a transformation higher in the hierarchy than the previous one

$$H = H_S H_A H_P = \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathbf{v}^T & v \end{bmatrix} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix}$$

with A a non-singular matrix given by $A = sRK + \mathbf{t}\mathbf{v}^T$, and K an upper-triangular matrix normalized as $\det(K) = 1$

Decomposition of a Projective Transformations

- A projective transformation can be decomposed into a chain of transformations, where each matrix represents a transformation higher in the hierarchy than the previous one

$$H = H_S H_A H_P = \begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} I & 0 \\ \mathbf{v}^T & v \end{bmatrix} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix}$$

  with A a non-singular matrix given by $A = sRK + \mathbf{t}\mathbf{v}^T$, and K an upper-triangular matrix normalized as $\det(K) = 1$

- This decomposition is valid provided that $v \neq 0$, and is unique if s is chosen positive

The Projective Geometry of 1D

- A point x on a line in $\mathbb{P}^1$ is represented by homogeneous coordinates $[x_1, x_2]^T$, and a point for which $x_2 = 0$ is an ideal point of the line

The Projective Geometry of 1D

- A point x on a line in $\mathbb{P}^1$ is represented by homogeneous coordinates $[x_1, x_2]^T$, and a point for which $x_2 = 0$ is an ideal point of the line
- We use the notation $\bar{\mathbf{x}}$ to represent the 2-vector $[x_1, x_2]^T$

The Projective Geometry of 1D

- A point x on a line in $\mathbb{P}^1$ is represented by homogeneous coordinates $[x_1, x_2]^T$, and a point for which $x_2 = 0$ is an ideal point of the line
- We use the notation $\bar{\mathbf{x}}$ to represent the 2-vector $[x_1, x_2]^T$
- A projective transformation of a line is represented by a 2×2 homogeneous matrix

$$\bar{\mathbf{x}}' = H_{2\times 2}\bar{\mathbf{x}}$$

and can be determined from three corresponding points

Cross Ratio

- The **cross ratio** is the basic projective invariant of $\mathbb{P}^1$

Cross Ratio

- The **cross ratio** is the basic projective invariant of $\mathbb{P}^1$
- Given 4 points $\bar{\mathbf{x}}_i$, the cross ratio is defined as

$$\mathrm{Cross}(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \bar{\mathbf{x}}_3, \bar{\mathbf{x}}_4) = \frac{|\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2||\bar{\mathbf{x}}_3\bar{\mathbf{x}}_4|}{|\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2||\bar{\mathbf{x}}_3\bar{\mathbf{x}}_4|}$$

where

$$|\bar{\mathbf{x}}_i\bar{\mathbf{x}}_j| = \det \begin{bmatrix} x_{i1} & x_{j1} \\ x_{i2} & x_{j2} \end{bmatrix}$$

Cross Ratio

- The **cross ratio** is the basic projective invariant of $\mathbb{P}^1$
- Given 4 points $\bar{\mathbf{x}}_i$, the cross ratio is defined as

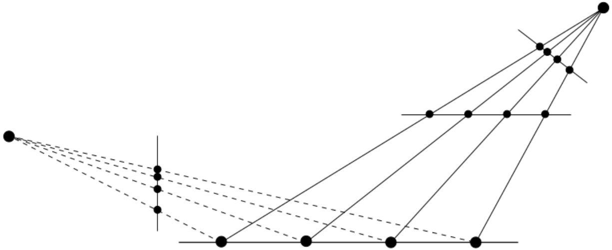
$$\text{Cross}(\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \bar{\mathbf{x}}_3, \bar{\mathbf{x}}_4) = \frac{|\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2||\bar{\mathbf{x}}_3\bar{\mathbf{x}}_4|}{|\bar{\mathbf{x}}_1\bar{\mathbf{x}}_2||\bar{\mathbf{x}}_3\bar{\mathbf{x}}_4|}$$

where

$$|\bar{\mathbf{x}}_i\bar{\mathbf{x}}_j| = \det\begin{bmatrix} x_{i1} & x_{j1} \\ x_{i2} & x_{j2} \end{bmatrix}$$

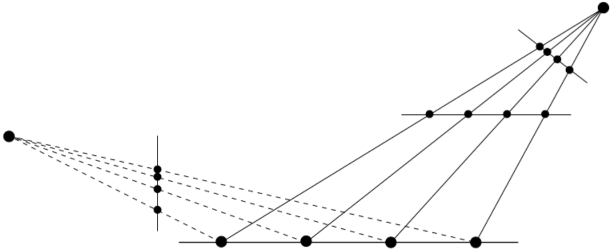
- Under a projective transformation of the plane, a 1D projective transformation is induced on any line in the plane

Projective Transformations Between Lines



- Each set of four collinear points is related to the others by a line-to-line projectivity

Projective Transformations Between Lines



- Each set of four collinear points is related to the others by a line-to-line projectivity
- Since the cross ratio is an invariant under a projectivity, the cross ratio has the same value for all the sets shown

Recovering Affine Properties from Images

- Once the imaged line at infinity is identified in an image plane, it is then possible to make affine measurements on the original plane (e.g. lines may be identified as parallel on the original plane if the imaged lines intersect on the imaged $\mathbf{l}_\infty$)

Recovering Affine Properties from Images

- Once the imaged line at infinity is identified in an image plane, it is then possible to make affine measurements on the original plane (e.g. lines may be identified as parallel on the original plane if the imaged lines intersect on the imaged $\mathbf{l}_\infty$)
- To do this, we simply transform the identified $\mathbf{l}_\infty$ to its canonical position of $\mathbf{l}_\infty = [0, 0, 1]^T$, then the (projective) matrix that achieves this transformation can be applied to every point in the image in order to affinely rectify the image

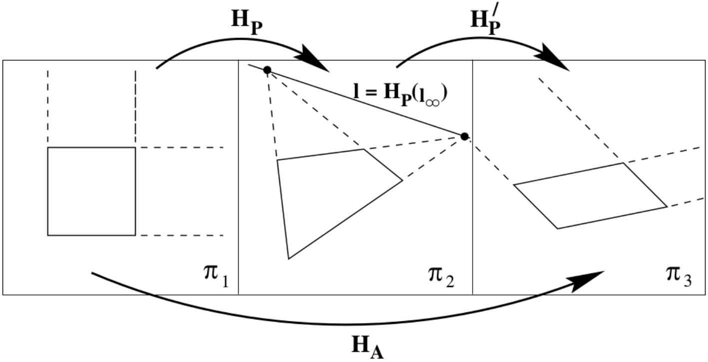
Recovering Affine Properties from Images

- If the imaged line at infinity is the line $\mathbf{l} = [l_1, l_2, l_3]^T$, then provided $l_3 \neq 0$ a suitable projective point transformation which will map $\mathbf{l}$ back to $\mathbf{l}_\infty = [0, 0, 1]^T$ is

$$H = H_A \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix}$$

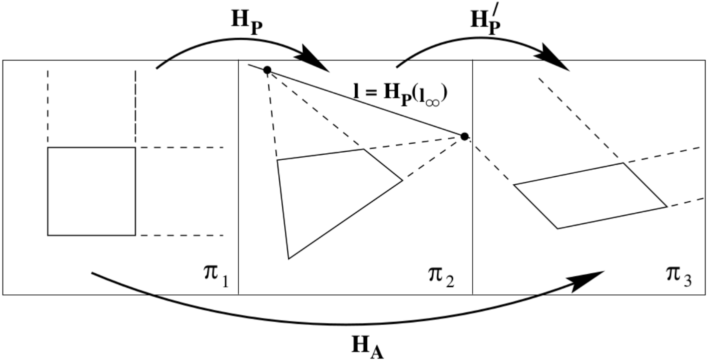
where H_A is any affine transformation (the last row of H is $\mathbf{l}^T$)

Affine Rectification



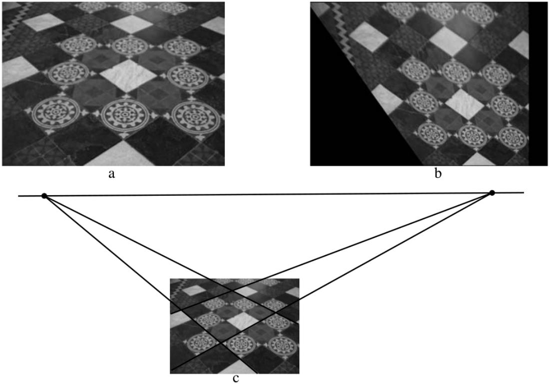
- A projective transformation maps $\mathbf{l}_\infty$ from $[0, 0, 1]^T$ on π_1 to a finite line $\mathbf{l}$ on π_2

Affine Rectification



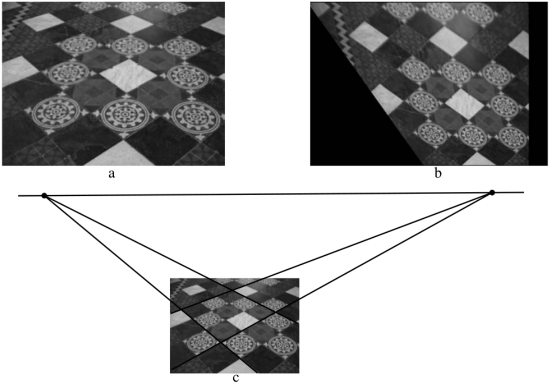
- A projective transformation maps $\mathbf{l}_\infty$ from $[0, 0, 1]^T$ on π_1 to a finite line $\mathbf{l}$ on π_2
- If this transformation is constructed such that $\mathbf{l}$ is mapped back to $[0, 0, 1]^T$ then the transformation between the first and third planes must be an affinity

Affine Rectification via the Vanishing Line



- The vanishing line of the plane imaged in (a) is computed (c) from the intersection of two sets of imaged parallel lines

Affine Rectification via the Vanishing Line



- The vanishing line of the plane imaged in (a) is computed (c) from the intersection of two sets of imaged parallel lines
- The image is then projectively warped to produce the affinely rectified image (b)

Circular Points

- Under any similarity transform there are two points on $\mathbf{l}_\infty$ which are fixed called the **circular points** (or **absolute points**) $\mathbf{I}$,$\mathbf{J}$, with canonical coordinates

$$\mathbf{I} = \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} \qquad \mathbf{J} = \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix}$$

where H_A is any affine transformation (the last row of H is $\mathbf{l}^T$)

Circular Points

- Under any similarity transform there are two points on $\mathbf{l}_\infty$ which are fixed called the **circular points** (or **absolute points**) $\mathbf{I}$,$\mathbf{J}$, with canonical coordinates

$$\mathbf{I} = \begin{bmatrix} 1 \\ i \\ 0 \end{bmatrix} \qquad \mathbf{J} = \begin{bmatrix} 1 \\ -i \\ 0 \end{bmatrix}$$

  where H_A is any affine transformation (the last row of H is $\mathbf{l}^T$)

- The circular points are a pair of complex conjugate ideal points

Circular Points

- The circular points, $\mathbf{I}$,$\mathbf{J}$, are fixed points under the projective transformation H iff H is a similarity

Circular Points

- The circular points, $\mathbf{I}$,$\mathbf{J}$, are fixed points under the projective transformation H iff H is a similarity
- The name ‘circular points’ arises because every circle intersects $\mathbf{l}_\infty$ at the circular points

The Conic Dual to the Circular Points

- The conic

$$C^*_\infty = \mathbf{IJ}^T + \mathbf{JI}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

  is the dual to the circular points

The Conic Dual to the Circular Points

- The conic

$$C^*_\infty = \mathbf{I}\mathbf{J}^T + \mathbf{J}\mathbf{I}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

  is the dual to the circular points

- C^*_∞ is a degenerate (rank 2) line conic which consists of the two circular points

The Conic Dual to the Circular Points

- The conic

$$C^*_\infty = \mathbf{I}\mathbf{J}^T + \mathbf{J}\mathbf{I}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

  is the dual to the circular points

- C^*_∞ is a degenerate (rank 2) line conic which consists of the two circular points

- C^*_∞ is fixed under similarity transforms in an analogous fashion to the fixed properties of circular points

Angles on the Projective Plane

- In Euclidean geometry, the angle between two lines is computed from the dot product of their normals

Angles on the Projective Plane

- In Euclidean geometry, the angle between two lines is computed from the dot product of their normals
- For lines $\mathbf{l} = [l_1, l_2, l_3]^T$ and $\mathbf{m} = [m_1, m_2, m_3]^T$ with normals parallel to $[l_1, l_2]^T, [m_1, m_2]^T$ respectively, the angle is

$$\cos\theta = \frac{l_1 m_1 + l_2 m_2}{\sqrt{(l_1^2 + l_2^2)(m_1^2 + m_2^2)}}$$

Angles on the Projective Plane

- In Euclidean geometry, the angle between two lines is computed from the dot product of their normals
- For lines $\mathbf{l} = [l_1, l_2, l_3]^T$ and $\mathbf{m} = [m_1, m_2, m_3]^T$ with normals parallel to $[l_1, l_2]^T, [m_1, m_2]^T$ respectively, the angle is

$$\cos\theta = \frac{l_1 m_1 + l_2 m_2}{\sqrt{(l_1^2 + l_2^2)(m_1^2 + m_2^2)}}$$

- The problem with this expression is that the first two components if $\mathbf{l}$ and $\mathbf{m}$ do not have well defined transformation properties under projective transformations

Angles on the Projective Plane

- However, an analogous expression which is invariant to projective transformations is

$$\cos\theta = \frac{\mathbf{l}^T C_\infty^* \mathbf{m}}{\sqrt{(\mathbf{l}^T C_\infty^* \mathbf{l})(\mathbf{m}^T C_\infty^* \mathbf{m})}}$$

where C_∞^* is the conic dual to the circular points

Angles on the Projective Plane

- However, an analogous expression which is invariant to projective transformations is

$$\cos\theta = \frac{\mathbf{l}^T C^*_\infty \mathbf{m}}{\sqrt{(\mathbf{l}^T C^*_\infty \mathbf{l})(\mathbf{m}^T C^*_\infty \mathbf{m})}}$$

  where C^*_∞ is the conic dual to the circular points

- Thus, once C^*_∞ is identified on the projective plane then Euclidean angles may be measured

Angles on the Projective Plane

- However, an analogous expression which is invariant to projective transformations is

$$\cos\theta = \frac{\mathbf{l}^T C_\infty^* \mathbf{m}}{\sqrt{(\mathbf{l}^T C_\infty^* \mathbf{l})(\mathbf{m}^T C_\infty^* \mathbf{m})}}$$

  where C_∞^* is the conic dual to the circular points

- Thus, once C_∞^* is identified on the projective plane then Euclidean angles may be measured

- Additionally, lines $\mathbf{l}$ and $\mathbf{m}$ are orthogonal if $\mathbf{l}^T C_\infty^* \mathbf{m} = 0$

Recovery of Metric Properties from Images

- Metric properties can be recovered from an image of a plane by transforming the circular points to their canonical position

Recovery of Metric Properties from Images

- Metric properties can be recovered from an image of a plane by transforming the circular points to their canonical position
- Suppose the circular points are identified in an image and the image is then rectified by a projective transformation H that maps the imaged circular points to their canonical position $[1, \pm i, 0]^T$ on $\mathbf{l}_\infty$

Recovery of Metric Properties from Images

- Metric properties can be recovered from an image of a plane by transforming the circular points to their canonical position
- Suppose the circular points are identified in an image and the image is then rectified by a projective transformation H that maps the imaged circular points to their canonical position $[1, \pm i, 0]^T$ on $\mathbf{l}_\infty$
- Then, the transformation between the world plane and the rectified image is a similarity since it is projective and the circular points are fixed

Metric Rectification Using C_∞^*

- C_∞^* neatly packages all the information required for a metric rectification

Metric Rectification Using C_∞^*

- C_∞^* neatly packages all the information required for a metric rectification
- It enables both the projective and affine components of a projective transformation to be determined, leaving only similarity distortions

Metric Rectification Using C^*_∞

- C^*_∞ neatly packages all the information required for a metric rectification
- It enables both the projective and affine components of a projective transformation to be determined, leaving only similarity distortions
- If the point transformation is $\mathbf{x}' = H\mathbf{x}$, where the x-coordinate frame is Euclidean and $\mathbf{x}'$ projective, C^*_∞ transforms to $C^*_\infty = HC^*H^T$

Metric Rectification Using C^*_∞

- Using the decomposition chain for H (slide 36)

$$
\begin{aligned}
C^{*\prime}_\infty &= (H_P H_A H_S) C^*_\infty (H_P H_A H_S)^T = (H_P H_A)(H_S C^*_\infty H_S^T)(H_A^T H_P^T) \\
&= (H_P H_A) C^*_\infty (H_A^T H_P^T) \\
&= \begin{bmatrix} KK^T & KK^T\mathbf{v} \\ \mathbf{v}^T KK^T & \mathbf{v}^T KK^T \mathbf{v} \end{bmatrix}
\end{aligned}
$$

Metric Rectification Using C_∞^*

- Using the decomposition chain for H (slide 36)

$$
\begin{aligned}
C_\infty^{*'} &= (H_P H_A H_S) C_\infty^* (H_P H_A H_S)^T = (H_P H_A)(H_S C_\infty^* H_S^T)(H_A^T H_P^T) \\
&= (H_P H_A) C_\infty^* (H_A^T H_P^T) \\
&= \begin{bmatrix} KK^T & KK^T\mathbf{v} \\ \mathbf{v}^T KK^T & \mathbf{v}^T KK^T \mathbf{v} \end{bmatrix}
\end{aligned}
$$

- Once C_∞^* is identified on the projective plane then projective distortion may be rectified up to a similarity

Example: Metric Rectification

- Given the perspective image of the plane, suppose lines $\mathbf{l}$ and $\mathbf{m}$ are images of orthogonal lines on the world plane

Example: Metric Rectification

- Given the perspective image of the plane, suppose lines $\mathbf{l}$ and $\mathbf{m}$ are images of orthogonal lines on the world plane
- Then, from $C^*_\infty \mathbf{m} = 0$, and in a similar manner to constraining a conic to contain a point (slide 19), this provides a linear constraint on the elements of C^*_∞:

$$[l_1 m_1, (l_1 m_2 + l_2 m_1)/2, l_2 m_2, (l_1 m_3 + l_3 m_1)/2, (l_2 m_3 + l_3 m_2)/2, l_3 m_3)\mathbf{c} = 0$$

where $\mathbf{c} = [a, b, c, d, e, f]^T$ is the conic matrix of C^*_∞ written as a 6-vector

Example: Metric Rectification

- Given the perspective image of the plane, suppose lines $\mathbf{l}$ and $\mathbf{m}$ are images of orthogonal lines on the world plane
- Then, from $C^*_\infty \mathbf{m} = 0$, and in a similar manner to constraining a conic to contain a point (slide 19), this provides a linear constraint on the elements of C^*_∞:

$$[l_1 m_1, (l_1 m_2 + l_2 m_1)/2, l_2 m_2, (l_1 m_3 + l_3 m_1)/2, (l_2 m_3 + l_3 m_2)/2, l_3 m_3)\mathbf{c} = 0$$

where $\mathbf{c} = [a, b, c, d, e, f]^T$ is the conic matrix of C^*_∞ written as a 6-vector

- Five such constraints can be stacked to form a 5×6 matrix, and $\mathbf{c}$, and thus C^*_∞ is obtained as the null vector

Metric Rectification via Orthogonal Lines

a

b

- (a) C^*_∞ is determined on the perspectively imaged plane using the five orthogonal line pairs shown

Metric Rectification via Orthogonal Lines

a

b

- (a) C^*_∞ is determined on the perspectively imaged plane using the five orthogonal line pairs shown
- (b) C^*_∞ determines the circular points and the projective transformation necessary to metrically rectify the image

Projective Geometry and Transformations of 3D

- Many of the properties and entities of projective 3-space, or $\mathbb{P}^3$, are straightforward generalizations of those in $\mathbb{P}^2$

Projective Geometry and Transformations of 3D

- Many of the properties and entities of projective 3-space, or $\mathbb{P}^3$, are straightforward generalizations of those in $\mathbb{P}^2$
- For example, in $\mathbb{P}^3$ Euclidean 3-space is augmented with a set of ideal points which are on a *plane* at infinity, π_∞ (this is the analogue of $\mathbf{l}_\infty$ in $\mathbb{P}^2$)

Projective Geometry and Transformations of 3D

- Many of the properties and entities of projective 3-space, or $\mathbb{P}^3$, are straightforward generalizations of those in $\mathbb{P}^2$
- For example, in $\mathbb{P}^3$ Euclidean 3-space is augmented with a set of ideal points which are on a *plane* at infinity, π_∞ (this is the analogue of $\mathbf{l}_\infty$ in $\mathbb{P}^2$)
- Parallel lines, and now parallel *planes*, intersect on π_∞

Points and Projective Transforms

- A point $\mathbf{X}$ in 3-space is represented in homogeneous coordinates as a 4-vector

Points and Projective Transforms

- A point $\mathbf{X}$ in 3-space is represented in homogeneous coordinates as a 4-vector
- Specifically, the homogeneous vector $\mathbf{X} = [X_1, X_2, X_3, X_4]^T$ with $X_4 \neq 0$ represents the point $[X, Y, Z]^T$ of $\mathbb{R}^3$ with inhomogeneous coordinates

$$X = X_1/X_4, \quad Y = X_2/X_4 \quad Z = X_3/X_4$$

Points and Projective Transforms

- A point $\mathbf{X}$ in 3-space is represented in homogeneous coordinates as a 4-vector
- Specifically, the homogeneous vector $\mathbf{X} = [X_1, X_2, X_3, X_4]^T$ with $X_4 \neq 0$ represents the point $[X, Y, Z]^T$ of $\mathbb{R}^3$ with inhomogeneous coordinates

$$X = X_1/X_4, \quad Y = X_2/X_4 \quad Z = X_3/X_4$$

- A projective transform acting on $\mathbb{P}^3$ is a linear transformation on homogeneous 4-vectors represented by a non-singular 4×4 matrix: $\mathbf{X}' = H\mathbf{X}$

Planes

- A plane in 3-space may be written as

$$\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$$

Planes

- A plane in 3-space may be written as

$$\pi_1 X + \pi_2 Y + \pi_3 Z + \pi_4 = 0$$

- The homogeneous representation of the plane is the 4-vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \pi_3, \pi_4]^T$, and homogenizing by the replacements $X \to X_1/X_4, Y \to X_2/X_4, Z \to X_3/X_4$ gives

$$\pi_1 X_1 + \pi_2 X_2 + \pi_3 X_3 + \pi_4 X_4 = 0$$

or more concisely

$$\boldsymbol{\pi}^T \mathbf{X} = 0$$

which expresses that the point $\mathbf{X}$ is on the plane $\boldsymbol{\pi}$

Three Points Define a Plane

- Suppose three points $\mathbf{X}_i$ are incident with the plane $\boldsymbol{\pi}$, then $\boldsymbol{\pi}^T\mathbf{X}_i = 0$, $i = 1, \ldots, 3$ and stacking these equations into a matrix gives

$$\begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \mathbf{X}_3^T \end{bmatrix} \boldsymbol{\pi} = 0$$

Three Points Define a Plane

- Suppose three points $\mathbf{X}_i$ are incident with the plane $\boldsymbol{\pi}$, then $\boldsymbol{\pi}^T\mathbf{X}_i = 0,\ i = 1, \ldots, 3$ and stacking these equations into a matrix gives

$$\begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \mathbf{X}_3^T \end{bmatrix} \boldsymbol{\pi} = 0$$

- Since the three points $\mathbf{X}_1, \mathbf{X}_2$, and $\mathbf{X}_3$ in general position are linearly independent it follows that the 3×4 matrix composed of the points as rows has rank 3

Three Points Define a Plane

- Suppose three points $\mathbf{X}_i$ are incident with the plane $\boldsymbol{\pi}$, then $\boldsymbol{\pi}^T \mathbf{X}_i = 0$, $i = 1, \ldots, 3$ and stacking these equations into a matrix gives

$$\begin{bmatrix} \mathbf{X}_1^T \\ \mathbf{X}_2^T \\ \mathbf{X}_3^T \end{bmatrix} \boldsymbol{\pi} = 0$$

- Since the three points $\mathbf{X}_1, \mathbf{X}_2$, and $\mathbf{X}_3$ in general position are linearly independent it follows that the 3×4 matrix composed of the points as rows has rank 3

- The plane $\boldsymbol{\pi}$ is obtained uniquely (up to scale) as the 1D (right) null space

Three Points Define a Plane

- We can obtain an expression for $\boldsymbol{\pi}$ in $\mathbb{P}^3$ using the properties of determinants and minors

Three Points Define a Plane

- We can obtain an expression for $\boldsymbol{\pi}$ in $\mathbb{P}^3$ using the properties of determinants and minors
- Starting from the matrix $M = [\mathbf{X}, \mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3]$, $\det(M) = 0$ when $\mathbf{X}$ lies on $\boldsymbol{\pi}$ since the point $\mathbf{X}$ is expressible as a linear combination of the points $\mathbf{X}_i, i = 1, \ldots, 3$

Three Points Define a Plane

- We can obtain an expression for $\boldsymbol{\pi}$ in $\mathbb{P}^3$ using the properties of determinants and minors
- Starting from the matrix $M = [\mathbf{X}, \mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3]$, $\det(M) = 0$ when $\mathbf{X}$ lies on $\boldsymbol{\pi}$ since the point $\mathbf{X}$ is expressible as a linear combination of the points $\mathbf{X}_i, i = 1, \ldots, 3$
- Expanding the determinant about the column $\mathbf{X}$ we obtain

$$\det(M) = X_1 D_{234} - X_2 D_{134} + X_3 D_{124} - X_4 D_{123}$$

where D_{jkl} is the determinant formed from the jkl rows of $[\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3]$ and thus $\boldsymbol{\pi} = [D_{234}, -D_{134}, D_{124} - D_{123}]^T$

Example: Computing the Plane π

- Suppose the three points defining the plane are

$$\mathbf{X}_1 = \begin{bmatrix} \tilde{\mathbf{X}}_1 \\ 1 \end{bmatrix} \quad \mathbf{X}_2 = \begin{bmatrix} \tilde{\mathbf{X}}_2 \\ 1 \end{bmatrix} \quad \mathbf{X}_3 = \begin{bmatrix} \tilde{\mathbf{X}}_3 \\ 1 \end{bmatrix}$$

where $\tilde{\mathbf{X}} = [X, Y, Z]^T$, then

$$D_{234} = \begin{vmatrix} Y_1 & Y_2 & Y_3 \\ Z_1 & Z_2 & Z_3 \\ 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} Y_1 - Y_3 & Y_2 - Y_3 & Y_3 \\ Z_1 - Z_3 & Z_2 - Z_3 & Z_3 \\ 0 & 0 & 1 \end{vmatrix} = \left((\tilde{\mathbf{X}}_1 - \tilde{\mathbf{X}}_3) \times (\tilde{\mathbf{X}}_2 - \tilde{\mathbf{X}}_3) \right)_1$$

and similarly for the other components, giving

$$\boldsymbol{\pi} = \begin{pmatrix} (\tilde{\mathbf{X}}_1 - \tilde{\mathbf{X}}_3) \times (\tilde{\mathbf{X}}_2 - \tilde{\mathbf{X}}_3) \\ -\tilde{\mathbf{X}}_3^T (\tilde{\mathbf{X}}_1 \times \tilde{\mathbf{X}}_2) \end{pmatrix}$$

Three Planes Define a Point

- The intersection of three planes $\boldsymbol{\pi}_i$ can be computed as the (right) null space of the 3×4 matrix composed of the planes as rows

$$\begin{bmatrix} \pi_1^T \\ \pi_2^T \\ \pi_3^T \end{bmatrix} \mathbf{X} = \mathbf{0}$$

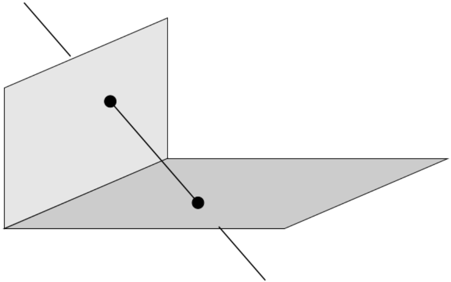
Three Planes Define a Point

- The intersection of three planes $\boldsymbol{\pi}_i$ can be computed as the (right) null space of the 3×4 matrix composed of the planes as rows

$$\begin{bmatrix} \boldsymbol{\pi}_1^T \\ \boldsymbol{\pi}_2^T \\ \boldsymbol{\pi}_3^T \end{bmatrix} \mathbf{X} = \mathbf{0}$$

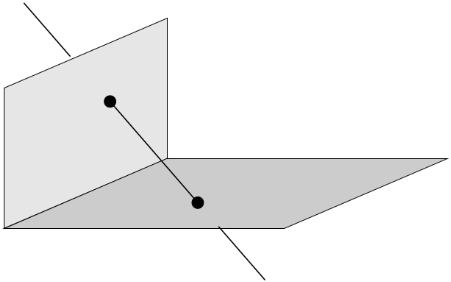
- A direct solution for $\mathbf{X}$ can be found by computing the determinants of 3×3 matrices

Lines



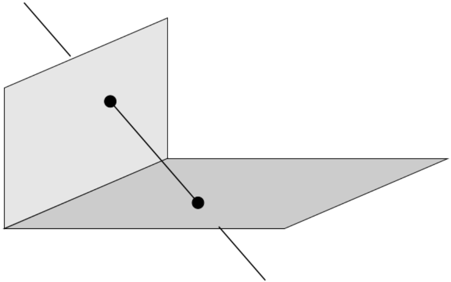
- A line is defined by the **join** or the intersection of two planes

Lines



- A line is defined by the **join** or the intersection of two planes
- Lines are awkward to represent in 3-space since a natural representation for an object with 4 degrees of freedom would be a homogeneous 5-vector

Lines



- A line is defined by the **join** or the intersection of two planes
- Lines are awkward to represent in 3-space since a natural representation for an object with 4 degrees of freedom would be a homogeneous 5-vector
- To overcome this, a number of line representations have been proposed

Plücker Matrices

- A line can be represented by a 4×4 skew-symmetric homogeneous matrix known as a **Plücker matrix**

Plücker Matrices

- A line can be represented by a 4×4 skew-symmetric homogeneous matrix known as a **Plücker matrix**
- In particular the line joining two points $\mathbf{A}, \mathbf{B}$ is represented by the Plücker matrix L with elements

$$l_{ij} = A_i B_j - B_i A_j$$

or equivalently in vector notation as

$$L = \mathbf{A}\mathbf{B}^T - \mathbf{B}\mathbf{A}^T$$

Example: Computing L as a Plücker Matrix

- The X-axis is represented as

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

where the points **A** and **B** are the origin and ideal point in the X-direction respectively

The Dual Representation L^*

- The dual Plücker representation L^* is obtained for a line formed by the intersection of two planes $\mathbf{P}, \mathbf{Q}$

$$L^* = \mathbf{P}\mathbf{Q}^T - \mathbf{Q}\mathbf{P}^T$$

and has similar properties to L

The Dual Representation L^*

- The dual Plücker representation L^* is obtained for a line formed by the intersection of two planes $\mathbf{P}, \mathbf{Q}$

  $$L^* = \mathbf{P}\mathbf{Q}^T - \mathbf{Q}\mathbf{P}^T$$

  and has similar properties to L

- Under the point transformation $\mathbf{X}' = H\mathbf{X}$, L^* transforms as $L^{*'} = H^{-T} L H^{-1}$

The Dual Representation L^*

- The dual Plücker representation L^* is obtained for a line formed by the intersection of two planes $\mathbf{P}, \mathbf{Q}$

$$L^* = \mathbf{P}\mathbf{Q}^T - \mathbf{Q}\mathbf{P}^T$$

  and has similar properties to L

- Under the point transformation $\mathbf{X}' = H\mathbf{X}$, L^* transforms as $L^{*'} = H^{-T} L H^{-1}$

- L^* can be obtained directly from L by a simple rewrite rule

$$l_{12} : l_{13} : l_{14} : l_{23} : l_{42} : l_{34} = l^*_{34} : l^*_{42} : l^*_{23} : l^*_{14} : l^*_{13} : l^*_{12}$$

Plücker Line Coordinates

- The **Plücker line coordinates** are the six non-zero elements of the 4×4 skew-symmetric L, namely

$$\mathcal{L} = \{l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}\}$$

Plücker Line Coordinates

- The **Plücker line coordinates** are the six non-zero elements of the 4×4 skew-symmetric L, namely

$$\mathcal{L} = \{l_{12}, l_{13}, l_{14}, l_{23}, l_{42}, l_{34}\}$$

- It follows from evaluating $\det(L) = 0$ that the coordinates satisfy the equation

$$l_{12}l_{34} + l_{13}l_{42} + l_{14}l_{23} = 0$$

and a 6-vector $\mathcal{L}$ corresponds to a line in 3-space only if this equation is satisfied

Plücker Line Coordinates

- Suppose two lines $\mathcal{L}, \hat{\mathcal{L}}$ are the joins of the points $\mathbf{A}, \mathbf{B}$ and $\hat{\mathbf{A}}, \hat{\mathbf{B}}$ respectively, then the lines intersect iff the four points are coplanar

Plücker Line Coordinates

- Suppose two lines $\mathcal{L}, \hat{\mathcal{L}}$ are the joins of the points $\mathbf{A}, \mathbf{B}$ and $\hat{\mathbf{A}}, \hat{\mathbf{B}}$ respectively, then the lines intersect iff the four points are coplanar
- A necessary and sufficient condition for this is that $\det[\mathbf{A}, \mathbf{B}, \hat{\mathbf{A}}, \hat{\mathbf{B}}] = 0$

Plücker Line Coordinates

- Suppose two lines $\mathcal{L}, \hat{\mathcal{L}}$ are the joins of the points $\mathbf{A}, \mathbf{B}$ and $\hat{\mathbf{A}}, \hat{\mathbf{B}}$ respectively, then the lines intersect iff the four points are coplanar

- A necessary and sufficient condition for this is that $\det[\mathbf{A}, \mathbf{B}, \hat{\mathbf{A}}, \hat{\mathbf{B}}] = 0$

- The determinant expands as

$$
\begin{aligned}
\det[\mathbf{A}, \mathbf{B}, \hat{\mathbf{A}}, \hat{\mathbf{B}}] &= l_{12}\hat{l}_{34} + \hat{l}_{12}l_{34} + l_{13}\hat{l}_{42} + \hat{l}_{13}l_{42} + l_{14}\hat{l}_{23} + \hat{l}_{14}l_{23} \\
&= (\mathcal{L}|\hat{\mathcal{L}})
\end{aligned}
$$

therefore two lines $\mathcal{L}$ and $\hat{\mathcal{L}}$ are coplanar (and thus intersect) iff $(\mathcal{L}|\hat{\mathcal{L}}) = 0$

The Plane at Infinity

- The plane at infinity has the canonical position $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$ in affine 3-space

The Plane at Infinity

- The plane at infinity has the canonical position $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$ in affine 3-space
- It contains the directions $\mathbf{D} = [X_1, X_2, X_3, 0]^T$ and enables the identification of the following properties:

The Plane at Infinity

- The plane at infinity has the canonical position $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$ in affine 3-space
- It contains the directions $\mathbf{D} = [X_1, X_2, X_3, 0]^T$ and enables the identification of the following properties:
  - Two planes are parallel iff their line of intersection is on $\boldsymbol{\pi}_\infty$

The Plane at Infinity

- The plane at infinity has the canonical position $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$ in affine 3-space
- It contains the directions $\mathbf{D} = [X_1, X_2, X_3, 0]^T$ and enables the identification of the following properties:
  - Two planes are parallel iff their line of intersection is on $\boldsymbol{\pi}_\infty$
  - A line is parallel to another line, or to a plane, if the point of intersection is on $\boldsymbol{\pi}_\infty$

The Plane at Infinity

- The plane at infinity has the canonical position $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$ in affine 3-space
- It contains the directions $\mathbf{D} = [X_1, X_2, X_3, 0]^T$ and enables the identification of the following properties:
  - Two planes are parallel iff their line of intersection is on $\boldsymbol{\pi}_\infty$
  - A line is parallel to another line, or to a plane, if the point of intersection is on $\boldsymbol{\pi}_\infty$
- $\boldsymbol{\pi}_\infty$ is a fixed plane under the projective transformation H iff H is an affinity

Estimation Problems

- There are a number of estimations problems that we would like to consider including:

Estimation Problems

- There are a number of estimations problems that we would like to consider including:
  - **2D homography** - Given a set of image points $\mathbf{x}_i$ and $\mathbf{x}_i'$ in $\mathbb{P}^2$, compute the projective transformation that takes each $\mathbf{x}_i$ to $\mathbf{x}_i'$

Estimation Problems

- There are a number of estimations problems that we would like to consider including:
  - **2D homography** - Given a set of image points $\mathbf{x}_i$ and $\mathbf{x}'_i$ in $\mathbb{P}^2$, compute the projective transformation that takes each $\mathbf{x}_i$ to $\mathbf{x}'_i$
  - **3D to 2D camera projection** - Given a set of points $\mathbf{X}_i$ in 3D and a set of corresponding points $\mathbf{x}_i$ in an image, find the 3D to 2D projective mapping that maps $\mathbf{X}_i$ to $\mathbf{x}_i$

Direct Linear Transformation (DLT) Algorithm

- Consider a set of point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ between two images where our problem is to compute a 3×3 matrix H such that $H\mathbf{x}_i = \mathbf{x}'_i$ for each i

Direct Linear Transformation (DLT) Algorithm

- Consider a set of point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ between two images where our problem is to compute a 3×3 matrix H such that $H\mathbf{x}_i = \mathbf{x}'_i$ for each i
- The equation $H\mathbf{x}_i = \mathbf{x}'_i$ may be expressed in terms of the vector cross product as $\mathbf{x}'_i \times H\mathbf{x}_i = \mathbf{0}$ (this form will enable a simple linear solution for H to be derived)

Derivation of the DLT Algorithm

- If the j-th row of the matrix H is denoted by $\mathbf{h}^{jT}$, then we can write

$$H\mathbf{x}_i = \begin{bmatrix} \mathbf{h}^{1T}\mathbf{x}_i \\ \mathbf{h}^{2T}\mathbf{x}_i \\ \mathbf{h}^{3T}\mathbf{x}_i \end{bmatrix}$$

Derivation of the DLT Algorithm

- If the j-th row of the matrix H is denoted by $\mathbf{h}^{jT}$, then we can write

$$H\mathbf{x}_i = \begin{bmatrix} \mathbf{h}^{1T}\mathbf{x}_i \\ \mathbf{h}^{2T}\mathbf{x}_i \\ \mathbf{h}^{3T}\mathbf{x}_i \end{bmatrix}$$

- Writing $\mathbf{x}'_i = [x'_i, y'_i, w'_i]^T$, the cross product may then be given explicitly as

$$\mathbf{x}'_i \times H\mathbf{x}_i = \begin{bmatrix} y'_i\mathbf{h}^{3T}\mathbf{x}_i - w'_i\mathbf{h}^{2T}\mathbf{x}_i \\ w'_i\mathbf{h}^{1T}\mathbf{x}_i - x'_i\mathbf{h}^{3T}\mathbf{x}_i \\ x'_i\mathbf{h}^{2T}\mathbf{x}_i - y'_i\mathbf{h}^{1T}\mathbf{x}_i \end{bmatrix}$$

Derivation of the DLT Algorithm

- Since $\mathbf{h}^{jT}\mathbf{x}_i = \mathbf{x}_i^T\mathbf{h}^j$ for $j = 1, \ldots, 3$, this gives a set of three equations in the entries of H, which may be written in the form

$$\begin{bmatrix} \mathbf{0}^T & -w_i'\mathbf{x}_i^T & y_i'\mathbf{x}_i^T \\ w_i'\mathbf{x}_i^T & \mathbf{0}^T & -x_i'\mathbf{x}_i^T \\ -y_i'\mathbf{x}_i^T & x_i'\mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0} \quad (4.1)$$

Derivation of the DLT Algorithm

- Since $\mathbf{h}^{jT}\mathbf{x}_i = \mathbf{x}_i^T\mathbf{h}^j$ for $j = 1, \ldots, 3$, this gives a set of three equations in the entries of H, which may be written in the form

$$\begin{bmatrix} \mathbf{0}^T & -w_i'\mathbf{x}_i^T & y_i'\mathbf{x}_i^T \\ w_i'\mathbf{x}_i^T & \mathbf{0}^T & -x_i'\mathbf{x}_i^T \\ -y_i'\mathbf{x}_i^T & x_i'\mathbf{x}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0} \quad (4.1)$$

- These equations have the form $A_i\mathbf{h} = \mathbf{0}$, where A_i is a 3×9 matrix, and $\mathbf{h}$ is a 9-vector made up of the entries of H

$$\mathbf{h} = \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix}, \quad H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad (4.2)$$

with h_i the ith element of H

Remarks on the Derivation of the DLT Algorithm

- The equation $A_i\mathbf{h} = \mathbf{0}$ is an equation *linear* in the unknown $\mathbf{h}$ while the matrix elements of A_i are quadratic in the known coordinates of the points

Remarks on the Derivation of the DLT Algorithm

- The equation $A_i\mathbf{h} = \mathbf{0}$ is an equation *linear* in the unknown $\mathbf{h}$ while the matrix elements of A_i are quadratic in the known coordinates of the points
- Although there are three equations, only two of them are linearly independent thus the set of equations becomes

$$\begin{bmatrix} \mathbf{0}^T & -w_i'\mathbf{x}_i^T & y_i'\mathbf{x}_i^T \\ w_i'\mathbf{x}_i^T & \mathbf{0}^T & -x_i'\mathbf{x}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{bmatrix} = \mathbf{0}$$

which will be written as

$$A_i\mathbf{h} = \mathbf{0}$$

DLT Algorithm: Solving for H

- Given a set of four point correspondences, we obtain a set of equations $A\mathbf{h} = \mathbf{0}$ where $\mathbf{h}$ is a vector of unknown entries of H

DLT Algorithm: Solving for H

- Given a set of four point correspondences, we obtain a set of equations $A\mathbf{h} = \mathbf{0}$ where $\mathbf{h}$ is a vector of unknown entries of H
- We seek a non-zero solution $\mathbf{h}$ (the obvious solution $\mathbf{h} = 0$ is of no interest to us)

DLT Algorithm: Solving for H

- Given a set of four point correspondences, we obtain a set of equations $A\mathbf{h} = \mathbf{0}$ where $\mathbf{h}$ is a vector of unknown entries of H
- We seek a non-zero solution $\mathbf{h}$ (the obvious solution $\mathbf{h} = 0$ is of no interest to us)
- A solution $\mathbf{h}$ giving the required H can only be determined up to scale (a scale may be arbitrarily chosen for $\mathbf{h}$ e.g. $||\mathbf{h}|| = 1$)

DLT Algorithm: Overdetermined Solution

- If more than four point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ are given, then the set of equations $A\mathbf{h} = \mathbf{0}$ is overdetermined

DLT Algorithm: Overdetermined Solution

- If more than four point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ are given, then the set of equations $A\mathbf{h} = \mathbf{0}$ is overdetermined
- If the position of the points is exact, then A will still have rank 8, a 1D null space, and there is an exact solution for $\mathbf{h}$

DLT Algorithm: Overdetermined Solution

- If more than four point correspondences $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ are given, then the set of equations $A\mathbf{h} = \mathbf{0}$ is overdetermined
- If the position of the points is exact, then A will still have rank 8, a 1D null space, and there is an exact solution for $\mathbf{h}$
- However, this will not be the case if the measurement of the image coordinates is inexact (i.e. noise) - there will not be an exact solution to $A\mathbf{h} = \mathbf{0}$ (apart from the zero solution)

DLT Algorithm: Overdetermined Solution

- Instead of demanding an exact solution, one attempts to find an approximate solution, e.g. a vector $\mathbf{h}$ that minimizes a suitable cost function

DLT Algorithm: Overdetermined Solution

- Instead of demanding an exact solution, one attempts to find an approximate solution, e.g. a vector $\mathbf{h}$ that minimizes a suitable cost function
- One way to do this is to minimize the norm $||A\mathbf{h}||$ subject to the constraint $||\mathbf{h}|| = 1$

DLT Algorithm: Overdetermined Solution

- Instead of demanding an exact solution, one attempts to find an approximate solution, e.g. a vector **h** that minimizes a suitable cost function
- One way to do this is to minimize the norm $||A\mathbf{h}||$ subject to the constraint $||\mathbf{h}|| = 1$
- The solution is the (unit) eigenvector of $A^T A$ with least eigenvalue, or equivalently the unit singular vector corresponding to the smallest singular value of A

The Basic DLT Algorithm for H

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$, determine the 2D homography matrix $\mathtt{H}$ such that $\mathbf{x}'_i = \mathtt{H}\mathbf{x}_i$.

Algorithm

(i) For each correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ compute the matrix $\mathtt{A}_i$ from (4.1). Only the first two rows need be used in general.

(ii) Assemble the n 2×9 matrices $\mathtt{A}_i$ into a single $2n \times 9$ matrix $\mathtt{A}$.

(iii) Obtain the SVD of $\mathtt{A}$ (section A4.4(*p*585)). The unit singular vector corresponding to the smallest singular value is the solution $\mathbf{h}$. Specifically, if $\mathtt{A} = \mathtt{UDV}^\mathsf{T}$ with $\mathtt{D}$ diagonal with positive diagonal entries, arranged in descending order down the diagonal, then $\mathbf{h}$ is the last column of $\mathtt{V}$.

(iv) The matrix $\mathtt{H}$ is determined from $\mathbf{h}$ as in (4.2).

Example: Solving for H using MATLAB

- We want to use the DLT algorithm to solve the projective transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Example: Solving for H using MATLAB

- We want to use the DLT algorithm to solve the projective transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- How do we deal with α?

Example: Solving for H using MATLAB

- We want to use the DLT algorithm to solve the projective transform

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- How do we deal with α?
  - Remove the scale factor and put into linear form

Example: Solving for H using MATLAB

- Multiply out

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \alpha \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

to get

$$\begin{aligned} x' &= \alpha(h_1 x + h_2 y + h_3) \\ y' &= \alpha(h_4 x + h_5 y + h_6) \\ 1 &= \alpha(h_7 x + h_8 y + h_9) \end{aligned}$$

Example: Solving for H using MATLAB

- Divide out the unknown scale factor

$$x'(h_7x + h_8y + h_9) = (h_1x + h_2y + h_3)$$
$$y'(h_7x + h_8y + h_9) = (h_4x + h_5y + h_6)$$

Example: Solving for H using MATLAB

- Divide out the unknown scale factor

$$
\begin{aligned}
x'(h_7 x + h_8 y + h_9) &= (h_1 x + h_2 y + h_3) \\
y'(h_7 x + h_8 y + h_9) &= (h_4 x + h_5 y + h_6)
\end{aligned}
$$

- Rearrange terms

$$
\begin{aligned}
h_7 x x' + h_8 y x' + h_9 x' - h_1 x - h_2 y - h_3 &= 0 \\
h_7 x y' + h_8 y y' + h_9 y' - h_4 x - h_5 y - h_6 &= 0
\end{aligned}
$$

Example: Solving for H using MATLAB

- In matrix form we have

$$A_i\mathbf{h} = \mathbf{0}$$

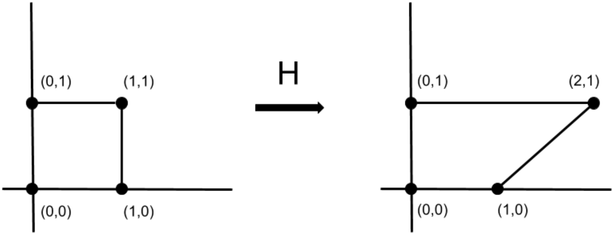
where

$$A_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$$

and

$$\mathbf{h} = \begin{bmatrix} h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_9 \end{bmatrix}^T$$

Example: Solving for H using MATLAB



- Consider the following point correspondences:

```
x = [0 1 0 1 1.01];
y = [0 0 1 1 0.99];
x_p = [0 1 0 2 2.01];
y_p = [0 0 1 1 1.01];
```

Example: Solving for H using MATLAB

- First, assemble the n 2×9 matrices A_i into a single $2n \times 9$ matrix A:

```
n = 5;
A = zeros(2*n,9)
for i=1:n
  A(2*i-1,:) = [-x(i),-y(i),-1,0,0,0,x(i)*x_p(i),x_p(i)*y(i),x_p(i)];
  A(2*i,:)   = [0,0,0,-x(i),-y(i),-1,x(i)*y_p(i),y_p(i)*y(i),y_p(i)];
end
```

Example: Solving for H using MATLAB

- Then, obtain the SVD of A:

```
[U,S,V] = svd(A)
```

Example: Solving for H using MATLAB

- Then, obtain the SVD of A:

```
[U,S,V] = svd(A)
```

- The singular vector corresponding to the smallest singular value is the solution **h**:

```
h = V(:,9)
```

Example: Solving for H using MATLAB

- Then, obtain the SVD of A:

```
[U,S,V] = svd(A)
```

- The singular vector corresponding to the smallest singular value is the solution **h**:

```
h = V(:,9)
```

- The matrix H is determined from **h**:

```
H = reshape(h,3,3)
```

Statistical Cost Functions

- To determine H for overdetermined solutions we can minimize a cost function (e.g. the DLT algorithm minimizes the norm $||A\mathbf{h}||$)

Statistical Cost Functions

- To determine H for overdetermined solutions we can minimize a cost function (e.g. the DLT algorithm minimizes the norm $||A\mathbf{h}||$)
- In order to obtain a best (optimal) estimate of H it is necessary to have a model for the measurement error (noise)

Statistical Cost Functions

- We will assume that in the absence of measurement error the true points exactly satisfy a homography, i.e. $\bar{\mathbf{x}}'_i = H\bar{\mathbf{x}}_i$

Statistical Cost Functions

- We will assume that in the absence of measurement error the true points exactly satisfy a homography, i.e. $\bar{\mathbf{x}}'_i = H\bar{\mathbf{x}}_i$
- A common assumption is that the image coordinate measurement obey a Gaussian (or normal) probability distribution

Statistical Cost Functions

- We will assume that in the absence of measurement error the true points exactly satisfy a homography, i.e. $\bar{\mathbf{x}}_i' = H\bar{\mathbf{x}}_i$
- A common assumption is that the image coordinate measurement obey a Gaussian (or normal) probability distribution
- Note that this assumption is not justified in general, and takes no account of the presence of outliers (grossly erroneous measurements) in the measured data

Statistical Cost Functions

- We will assume that in the absence of measurement error the true points exactly satisfy a homography, i.e. $\bar{\mathbf{x}}'_i = H\bar{\mathbf{x}}_i$
- A common assumption is that the image coordinate measurement obey a Gaussian (or normal) probability distribution
- Note that this assumption is not justified in general, and takes no account of the presence of outliers (grossly erroneous measurements) in the measured data
- Once outliers have been removed, the assumption of a Gaussian error model becomes more tenable

Statistical Cost Functions

- We will assume that the noise is Gaussian on each image coordinate with zero mean and uniform standard deviation σ

Statistical Cost Functions

- We will assume that the noise is Gaussian on each image coordinate with zero mean and uniform standard deviation σ
- This means that $x = \bar{x} + \delta x$, with δx obeying a Gaussian distribution with variance σ^2

Statistical Cost Functions

- We will assume that the noise is Gaussian on each image coordinate with zero mean and uniform standard deviation σ
- This means that $x = \bar{x} + \delta x$, with δx obeying a Gaussian distribution with variance σ^2
- Furthermore if we assume that the noise on each measurement is independent, then, if the true point is $\bar{\mathbf{x}}$, the **probability density function** (PDF) of each measured point $\mathbf{x}$ is

$$Pr(\mathbf{x}) = \left(\frac{1}{2\pi\sigma^2}\right) e^{-d(\mathbf{x},\bar{\mathbf{x}})^2/(2\sigma^2)}$$

Error in One Image

- First, we'll consider the case where the errors are only in the second image

Error in One Image

- First, we'll consider the case where the errors are only in the second image
- The probability of obtaining the set of correspondences $\{\bar{\mathbf{x}}_i \leftrightarrow \mathbf{x}'_i\}$ is simply the product of their individual PDFs, since the errors on each point are assumed independent

Error in One Image

- First, we'll consider the case where the errors are only in the second image
- The probability of obtaining the set of correspondences $\{\bar{\mathbf{x}}_i \leftrightarrow \mathbf{x}'_i\}$ is simply the product of their individual PDFs, since the errors on each point are assumed independent
- Then the PDF of the noise-perturbed data is

$$Pr(\{\mathbf{x}'_i\}|H) = \prod_i \left(\frac{1}{2\pi\sigma^2}\right) e^{-d(\mathbf{x}'_i, H\bar{\mathbf{x}}_i)^2/(2\sigma^2)}$$

where the symbol $Pr(\{\mathbf{x}'_i|H\})$ is to be interpreted as meaning the probability of obtaining the measurements $\{\mathbf{x}'_i\}$ given that the true homography is H

Error in One Image

- The *log-likelihood* of the set of correspondences is

$$\log Pr(\{\mathbf{x}_i'\}|H) = -\frac{1}{2\sigma^2}\sum_i d(\mathbf{x}_i', H\bar{\mathbf{x}}_i)^2 + \text{ constant}$$

Error in One Image

- The *log-likelihood* of the set of correspondences is

$$\log Pr(\{\mathbf{x}'_i\}|H) = -\frac{1}{2\sigma^2}\sum_i d(\mathbf{x}'_i, H\bar{\mathbf{x}}_i)^2 + \text{ constant}$$

- The **Maximum Likelihood** estimate (MLE) of the homography, $\hat{H}$, maximizes this log-likelihood, i.e. it minimizes

$$\sum_i d(\mathbf{x}'_i, H\bar{\mathbf{x}}_i)^2$$

Error in Both Images

- Similar to an error in one image, if the true correspondences are $\{\bar{\mathbf{x}}_i \leftrightarrow H\bar{\mathbf{x}}_i = \bar{\mathbf{x}}_\mathbf{i}'\}$, then the PDF of the noise-perturbed data is

$$Pr(\{\mathbf{x}_i, \mathbf{x}_i'\}|H, \{\bar{\mathbf{x}}_i\}) = \prod_i \left(\frac{1}{2\pi\sigma^2}\right) e^{-(d(\mathbf{x}_i,\bar{\mathbf{x}}_i)^2 + d(\mathbf{x}_i', H\bar{\mathbf{x}}_i)^2)/(2\sigma^2)}$$

Error in Both Images

- Similar to an error in one image, if the true correspondences are $\{\bar{\mathbf{x}}_i \leftrightarrow H\bar{\mathbf{x}}_i = \bar{\mathbf{x}}_\mathbf{i}'\}$, then the PDF of the noise-perturbed data is
$$Pr(\{\mathbf{x}_i, \mathbf{x}_i'\}|H, \{\bar{\mathbf{x}}_i\}) = \prod_i \left(\frac{1}{2\pi\sigma^2}\right) e^{-(d(\mathbf{x}_i,\bar{\mathbf{x}}_i)^2 + d(\mathbf{x}_i', H\bar{\mathbf{x}}_i)^2)/(2\sigma^2)}$$
- The additional complication is that we have to seek "corrected" image measurements that play the role of the true measurements $H\bar{\mathbf{x}}$

Error in Both Images

- Similar to an error in one image, if the true correspondences are $\{\bar{\mathbf{x}}_i \leftrightarrow H\bar{\mathbf{x}}_i = \bar{\mathbf{x}}_\mathbf{i}'\}$, then the PDF of the noise-perturbed data is

$$Pr(\{\mathbf{x}_i, \mathbf{x}_i'\}|H, \{\bar{\mathbf{x}}_i\}) = \prod_i \left(\frac{1}{2\pi\sigma^2}\right) e^{-(d(\mathbf{x}_i, \bar{\mathbf{x}}_i)^2 + d(\mathbf{x}_i', H\bar{\mathbf{x}}_i)^2)/(2\sigma^2)}$$

- The additional complication is that we have to seek "corrected" image measurements that play the role of the true measurements $H\bar{\mathbf{x}}$
- Thus the ML estimate of $H \; and \; \{\mathbf{x}_i \leftrightarrow \mathbf{x}_\mathbf{i}'\}$, is $\hat{H}$ and $\{\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}_\mathbf{i}'\}$ that minimize

$$\sum_i d(\mathbf{x}_i, \hat{\mathbf{x}}_i)^2 + d(\mathbf{x}_i', \hat{\mathbf{x}}_i')^2$$

with $\hat{\mathbf{x}}_i' = \hat{H}\hat{\mathbf{x}}_i$

Mahalanobis Distance

- In the general Gaussian case, one may assume a vector of measurements $\mathbf{X}$ satisfying a Gaussian distribution function with covariance matrix Σ

Mahalanobis Distance

- In the general Gaussian case, one may assume a vector of measurements $\mathbf{X}$ satisfying a Gaussian distribution function with covariance matrix Σ
- Maximizing the log-likelihood is then equivalent to minimizing the Mahalanobis distance

$$||\mathbf{X} - \bar{\mathbf{X}}||_{\Sigma}^{2} = (\mathbf{X} - \bar{\mathbf{X}})^{T}\Sigma^{-1}(\mathbf{X} - \bar{\mathbf{X}})$$

Mahalanobis Distance

- In the case where there is error in each image, but assuming that errors in one image are independent of the error in the other image, the appropriate cost function is

$$||\mathbf{X} - \bar{\mathbf{X}}||^2_{\Sigma} + ||\mathbf{X}' - \bar{\mathbf{X}}'||^2_{\Sigma}$$

where Σ and Σ' are the covariance matrices of the measurements in the two images

Mahalanobis Distance

- Finally, if we assume that the errors for all the points $\mathbf{x}_i$ and $\mathbf{x}'_i$ are independent, with individual covariance matrices Σ_i and Σ'_i respectively, then we have

$$\sum ||\mathbf{x}_i - \bar{\mathbf{x}}_i||^2_{\Sigma_i} + \sum ||\mathbf{x}'_i - \bar{\mathbf{x}}'_i||^2_{\Sigma'_i}$$

Mahalanobis Distance

- Finally, if we assume that the errors for all the points $\mathbf{x}_i$ and $\mathbf{x}'_i$ are independent, with individual covariance matrices Σ_i and Σ'_i respectively, then we have

$$\sum ||\mathbf{x}_i - \bar{\mathbf{x}}_i||^2_{\Sigma_i} + \sum ||\mathbf{x}'_i - \bar{\mathbf{x}}'_i||^2_{\Sigma'_i}$$

- This equation allows for the incorporation of the type of anisotropic covariance matrices that arise from point locations computed as the intersection of two non-perpendicular lines

Mahalanobis Distance

- Finally, if we assume that the errors for all the points $\mathbf{x}_i$ and $\mathbf{x}'_i$ are independent, with individual covariance matrices Σ_i and Σ'_i respectively, then we have

$$\sum ||\mathbf{x}_i - \bar{\mathbf{x}}_i||^2_{\Sigma_i} + \sum ||\mathbf{x}'_i - \bar{\mathbf{x}}'_i||^2_{\Sigma'_i}$$

- This equation allows for the incorporation of the type of anisotropic covariance matrices that arise from point locations computed as the intersection of two non-perpendicular lines

- In the case where the points are known exactly in one of the two images, errors being confined to the other image, one of two summation terms disappears

Robust Estimation

- Up to this point we've assumed that given a set of correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$ the only source of error is in the measurement of the point's position, which follows a Gaussian distribution

Robust Estimation

- Up to this point we've assumed that given a set of correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$ the only source of error is in the measurement of the point's position, which follows a Gaussian distribution
- In many practical situations this assumption is not valid because points are mismatched

Robust Estimation

- Up to this point we've assumed that given a set of correspondences $\{\mathbf{x}_i \leftrightarrow \mathbf{x}'_i\}$ the only source of error is in the measurement of the point's position, which follows a Gaussian distribution
- In many practical situations this assumption is not valid because points are mismatched
- The mismatched points are *outliers* to the Gaussian error distribution

Robust Estimation

- These outliers can severely disturb the estimated homography and consequently should be identified

Robust Estimation

- These outliers can severely disturb the estimated homography and consequently should be identified
- The goal then is to determine a set of *inliers* from the presented “correspondences” so that the homography can then be estimated in an optimal manner

Robust Estimation

- These outliers can severely disturb the estimated homography and consequently should be identified
- The goal then is to determine a set of *inliers* from the presented "correspondences" so that the homography can then be estimated in an optimal manner
- This is **robust estimation** since the estimation is robust (tolerant) to outliers (measurements following a different, and possibly unmodelled, error distribution)

RANSAC

- Consider the following problem: Given a set of 2D data points, find the line which minimizes the sum of the squared perpendicular distances subject to the condition that none of the valid points deviates from this line by more than t units

RANSAC

- Consider the following problem: Given a set of 2D data points, find the line which minimizes the sum of the squared perpendicular distances subject to the condition that none of the valid points deviates from this line by more than t units
- This is actually two problems: A line fit to the data; and a classification of the data into inliers (valid points) and outliers

RANSAC

- Consider the following problem: Given a set of 2D data points, find the line which minimizes the sum of the squared perpendicular distances subject to the condition that none of the valid points deviates from this line by more than t units
- This is actually two problems: A line fit to the data; and a classification of the data into inliers (valid points) and outliers
- The measurement t is set according to the measurement noise (e.g. $t = 3\sigma$)

RANSAC

- There are many types of robust algorithms and which one to use depends to some extent on the proportion of outliers

RANSAC

- There are many types of robust algorithms and which one to use depends to some extent on the proportion of outliers
- A very successful robust estimator is the **RANdom SAmple Consensus** (RANSAC) algorithm of Fischler and Bolles

RANSAC

- There are many types of robust algorithms and which one to use depends to some extent on the proportion of outliers
- A very successful robust estimator is the **RANdom SAmple Consensus** (RANSAC) algorithm of Fischler and Bolles
- The RANSAC algorithm is able to cope with a large proportion of outliers

RANSAC

- The idea of the RANSAC algorithm is simple: Two of the points are selected randomly; these points define a line and the *support* for this line is measured by the number of points that lie within a distance threshold

RANSAC

- The idea of the RANSAC algorithm is simple: Two of the points are selected randomly; these points define a line and the *support* for this line is measured by the number of points that lie within a distance threshold
- This random selection is repeated a number of times and the line with the most support is deemed the robust fit

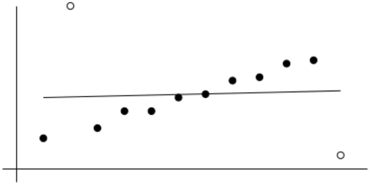
RANSAC

- The idea of the RANSAC algorithm is simple: Two of the points are selected randomly; these points define a line and the *support* for this line is measured by the number of points that lie within a distance threshold
- This random selection is repeated a number of times and the line with the most support is deemed the robust fit
- The points within the threshold distance are the inliers (and constitute the *consensus* set)

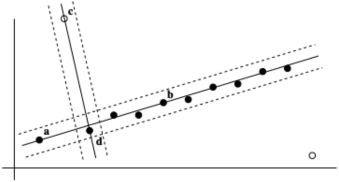
RANSAC

- The idea of the RANSAC algorithm is simple: Two of the points are selected randomly; these points define a line and the *support* for this line is measured by the number of points that lie within a distance threshold
- This random selection is repeated a number of times and the line with the most support is deemed the robust fit
- The points within the threshold distance are the inliers (and constitute the *consensus* set)
- The intuition is that if one of the points is an outlier, then the line will not gain much support

Robust Line Estimation



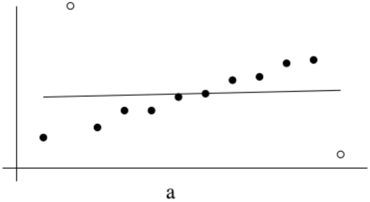
a

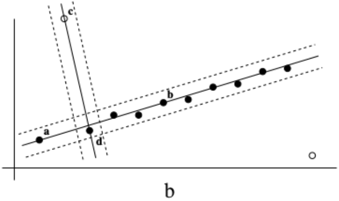


b

- The solid points are inliers, the open points outliers

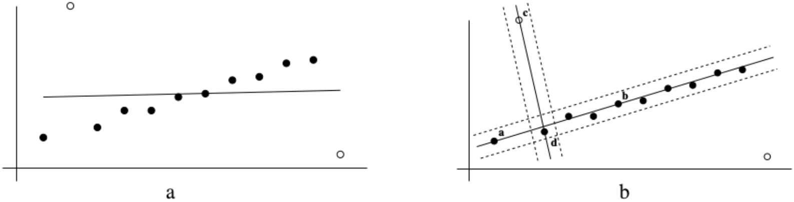
Robust Line Estimation





- The solid points are inliers, the open points outliers
- (a) A last squares fit to the point data is severely affected by outliers

Robust Line Estimation



- The solid points are inliers, the open points outliers
- (a) A last squares fit to the point data is severely affected by outliers
- (b) Using RANSAC, the support for lines through randomly selected point pairs is measured by the number of points within a threshold distance of the lines, the dotted lines indicate the threshold distance

RANSAC and Model Fitting

- More generally, we wish to fit a *model*, in this case a line, to data, and the random *sample* consists of a minimal subset of the data, in this case two points, sufficient to determine the model

RANSAC and Model Fitting

- More generally, we wish to fit a *model*, in this case a line, to data, and the random *sample* consists of a minimal subset of the data, in this case two points, sufficient to determine the model
- If the model is a planar homography, and the data a set of 2D point correspondences, then the minimal subset consists of four correspondences

RANSAC Algorithm

Objective

Robust fit of a model to a data set S which contains outliers.

Algorithm

(i) Randomly select a sample of s data points from S and instantiate the model from this subset.
(ii) Determine the set of data points S_i which are within a distance threshold t of the model. The set S_i is the consensus set of the sample and defines the inliers of S.
(iii) If the size of S_i (the number of inliers) is greater than some threshold T, re-estimate the model using all the points in S_i and terminate.
(iv) If the size of S_i is less than T, select a new subset and repeat the above.
(v) After N trials the largest consensus set S_i is selected, and the model is re-estimated using all the points in the subset S_i.

- A minimum of s data points are required to instantiate the free parameters of the model

Example: Fitting a line to set of 2D points using MATLAB

- Load and plot a set of noisy 2D points

```
load 'pointsForLineFitting.mat';
plot(points(:,1), points(:,2), '*');
hold on
```

Example: Fitting a line to set of 2D points using MATLAB

- Fit a line using linear least squares

```
modelLeastSquares = polyfit(points(:,1), points(:,2), 1);
x = [min(points(:,1)), max(points(:,1))];
y = modelLeastSquares(1)*x + modelLeastSquares(2);
plot(x, y, 'r-');
```

Example: Fitting a line to set of 2D points using MATLAB

- Fit a line to the points using the RANSAC algorithm

```
sampleSize = 2;
maxDistance = 2;
fitLineFcn = @(points) polyfit(points(:,1), points(:,2), 1);
evalLineFcn = ...
 @(model,points) sum((points(:,2) -
  polyval(model,points(:,1))).^2,2);
[modelRANSAC,inlierIdx] = ransac(points, fitLineFcn, ...
 sampleSize, maxDistance);
```

Example: Fitting a line to set of 2D points using MATLAB

- Refit a line to the inliers

```
modelInliers = polyfit(points(inlierIdx, 1), ...
 points(inlierIdx, 2), 1);
```

Example: Fitting a line to set of 2D points using MATLAB

- Display the line

```
inlierPts = points(inlierIdx, :);
x = [min(inlierPts(:,1)); max(inlierPts(:,1))];
y = modelInliers(1)*x + modelInliers(2);
plot(x, y, ‘g-’);
legend(‘Noisy Points’, ‘Least Squares Fit’, ‘Robust Fit’);
hold off
```

How to Choose the Distance Threshold?

- We would like to choose the distance threshold, t, such that with a probability of α the point is an inlier

How to Choose the Distance Threshold?

- We would like to choose the distance threshold, t, such that with a probability of α the point is an inlier
- This calculation requires the probability distribution for the distance of an inlier from the model, in practice this the threshold is chosen empirically

How to Choose the Distance Threshold?

- However, if it is assumed that the measurement error is Gaussian with zero mean and standard deviation σ, then a value for t may be computed

How to Choose the Distance Threshold?

- However, if it is assumed that the measurement error is Gaussian with zero mean and standard deviation σ, then a value for t may be computed
- In this case the square of the point distance, $d_\perp^2$, is a sum of squared Gaussian variables and follows a χ_m^2 distribution with m degrees of freedom

$$\begin{array}{rl} \text{inlier} & d_\perp^2 < t^2 \\ \text{outlier} & d_\perp^2 \geq t^2 \end{array}$$

with $t^2 = F_m^{-1}(\alpha)\sigma^2$ where $F_m(k^2) = \int_0^{k^2} \chi_m^2(\xi)d\xi$ and α is usually chosen as 0.95

How Many Samples?

- It is often computationally infeasible and unnecessary to try every possible sample

How Many Samples?

- It is often computationally infeasible and unnecessary to try every possible sample
- Instead, the number of samples N is chosen sufficiently high to ensure with a probability, p, that at least one of the random samples of s points is free from outliers (usually p is chosen at 0.99)

How Many Samples?

- It is often computationally infeasible and unnecessary to try every possible sample
- Instead, the number of samples N is chosen sufficiently high to ensure with a probability, p, that at least one of the random samples of s points is free from outliers (usually p is chosen at 0.99)
- Suppose w is the probability that any selected data point is an inlier ($\epsilon = 1 - w$ is the probability that it is an outlier), then at least N selections of s points are required, where $(1 - w^s)^N = 1 - p$, so that

$$N = \log(1 - p) / \log(1 - (1 - \epsilon)^s)$$

How Large is an Acceptable Consensus Set?

- A rule of thumb is to terminate if the size of the consensus set is similar to the number of inliers believed to be in the data set, given the assumed proportion of outliers

How Large is an Acceptable Consensus Set?

- A rule of thumb is to terminate if the size of the consensus set is similar to the number of inliers believed to be in the data set, given the assumed proportion of outliers
- For example, given n data points $T = (1 - \epsilon)n$ and a conservative estimate of ϵ may be 0.2

Determining the Number of Samples Adaptively

- It is often the case that ϵ, the fraction of data consisting of outliers, is unknown

Determining the Number of Samples Adaptively

- It is often the case that ϵ, the fraction of data consisting of outliers, is unknown
- In such cases the algorithm is initialized using a worst case estimate of ϵ, and this estimate can then be updated as larger consistent sets are found

Determining the Number of Samples Adaptively

- It is often the case that ϵ, the fraction of data consisting of outliers, is unknown
- In such cases the algorithm is initialized using a worst case estimate of ϵ, and this estimate can then be updated as larger consistent sets are found
- For example, if the worst case guess is $\epsilon = 0.5$ and a consensus set with 80% of the data is found as inliers, then the updated estimate is $\epsilon = 0.2$

Determining the Number of Samples Adaptively

- $N = \infty$, sample_count$= 0$.
- While $N >$ sample_count Repeat
  - Choose a sample and count the number of inliers.
  - Set $\epsilon = 1 -$ (number of inliers)/(total number of points)
  - Set N from ϵ and (4.18) with $p = 0.99$.
  - Increment the sample_count by 1.
- Terminate.

- An adaptive algorithm for determining the number of RANSAC samples

Robust Maximum Likelihood Estimation

- The RANSAC algorithm partitions the data set into inliers (the largest consensus set) and outliers (the rest of the data set) and also delivers an estimate of the model M_0 computed from the minimal set with greatest support

Robust Maximum Likelihood Estimation

- The RANSAC algorithm partitions the data set into inliers (the largest consensus set) and outliers (the rest of the data set) and also delivers an estimate of the model M_0 computed from the minimal set with greatest support
- The final step of the algorithm is to re-estimate the model using all inliers which should be optimal and will involve minimizing a ML cost function

Robust Maximum Likelihood Estimation

- The RANSAC algorithm partitions the data set into inliers (the largest consensus set) and outliers (the rest of the data set) and also delivers an estimate of the model M_0 computed from the minimal set with greatest support
- The final step of the algorithm is to re-estimate the model using all inliers which should be optimal and will involve minimizing a ML cost function
- In the case of a line, ML estimation is equivalent to orthogonal regression (a closed form solution is available) however in general the estimation involves iterative minimization and M_0 provides the starting point

Summary

- The main geometric ideas and notation to understand upcoming material has been provided

Summary

- The main geometric ideas and notation to understand upcoming material has been provided
- The problem of estimating 2D projective transformations has been presented