CORK INSTITUTE OF TECHNOLOGY
INSTITIÚID TEICNEOLAÍOCHTA CHORCAÍ

Computing@CIT

Machine Vision

Lecture 7: Projective Geometry

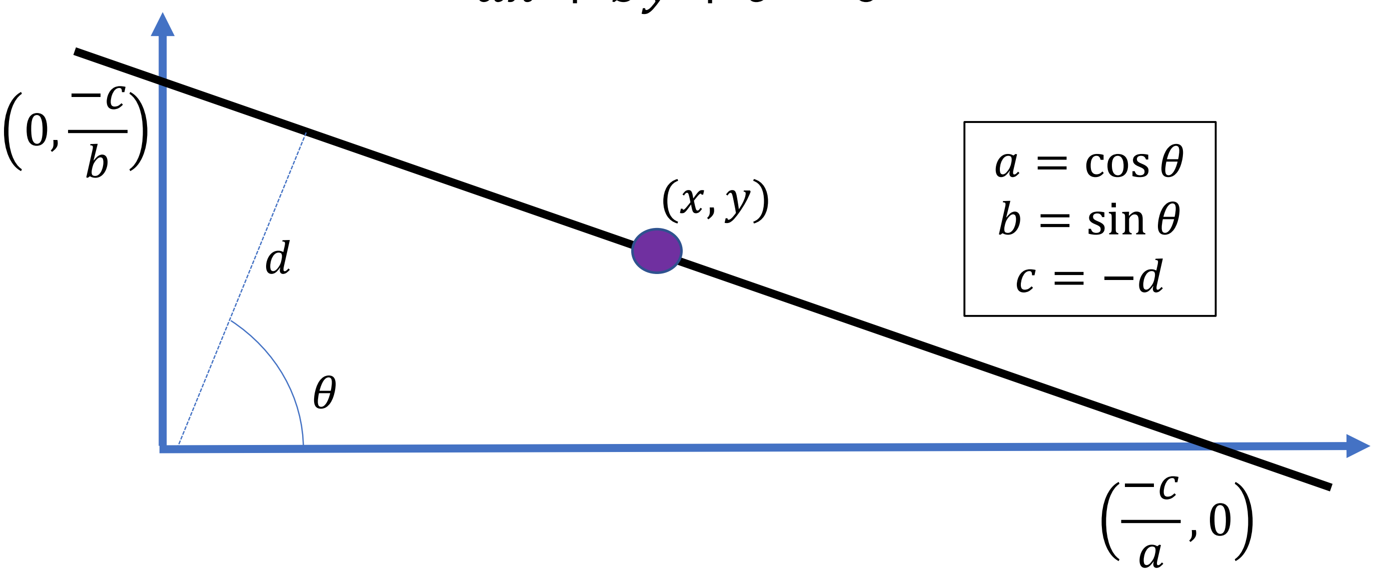
Why projective geometry?

- One of the main goals of machine vision is to make accurate measurements of 3d scene geometry using images
- However, when taking a picture of a 3d scene the geometry is distorted:
  - Parallel lines can intersect (vanishing points)
  - Points at infinity become finite, potentially crossing through the image (horizon)

Homogeneous coordinates

- To implement any algorithms on the geometry of images we need to represent the geometry algebraically
- A point (x, y) is on a line (a, b, c), if the following equation holds

$$ax + by + c = 0$$



Homogeneous coordinates

- To implement any algorithms on the geometry of images we need to represent the geometry algebraically
- A point (x, y) is on a line (a, b, c), if the following equation holds

$$ax + by + c = 0$$

- We can multiply the line parameters with any constant $\lambda \neq 0$ without changing the line equation

$$(\lambda a)x + (\lambda b)y + (\lambda c) = 0$$

- Therefore, (a, b, c) and $\lambda(a, b, c)$ represent the same line

Homogeneous coordinates

- The line equation $ax + by + c = 0$ can be written more compactly as scalar product

$$\boldsymbol{l}^T \boldsymbol{x} = 0$$

- Of the two vectors

$$\boldsymbol{l} = (a, b, c)^T$$
$$\boldsymbol{x} = (x, y, 1)^T$$

- We already observed that $\lambda \boldsymbol{l}$ and $\boldsymbol{l}$ represent the same line
- Very similar we can also conclude that the point represented by $\boldsymbol{x}$ is the same as the point represented by

$$\lambda \boldsymbol{x} = (\lambda x, \lambda y, \lambda)^T$$

Homogeneous coordinates

- A 2d point in **homogeneous coordinates** is represented by a homogeneous 3-vector

$$\boldsymbol{x} = (x_1, x_2, x_3)^T$$

- A 2d line in homogeneous coordinates is also represented by a homogeneous 3-vector

$$\boldsymbol{l} = (l_1, l_2, l_3)^T$$

- A **homogeneous vector** $\boldsymbol{x} \in \mathbb{P}^2$ is the equivalence class of vectors in $\mathbb{R}^3 \ - \{(0,0,0)^T\}$ under the equivalence relationship $\boldsymbol{x} \equiv \lambda \boldsymbol{x}$, i.e. multiplying a homogeneous vector with a non-zero constant does change the representation but not the geometric object

Homogeneous coordinates

- A point in Euclidean coordinates $(x, y) \in \mathbb{R}^2$ is converted into a point in homogeneous coordinates by simply appending a “1” at the end

$$\boldsymbol{x} = (x, y, 1)$$

- A point in homogeneous coordinates $(x_1, x_2, x_3) \in \mathbb{P}^2$ is converted back into Euclidean coordinated by dividing by the last component

$$\boldsymbol{x} = \left(\frac{x_1}{x_3}, \frac{x_2}{x_3}, 1\right)$$

- Note, that not all homogeneous vectors can be transformed into Euclidean vectors ($x_3 = 0$)

Homogeneous coordinates

- Similar, a line in Euclidean angle-distance representation

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

- Is equivalent to

$$\boldsymbol{l} = (\cos\theta, \sin\theta, -d)$$

- And vice-versa the homogeneous line (l_1, l_2, l_3) can be converted back to angle-distance representation by dividing by the norm of the first two components (because $\cos^2\theta + \sin^2\theta = 1$)

$$\boldsymbol{l} = \left(\frac{l_1}{\sqrt{l_1^2 + l_2^2}}, \frac{l_2}{\sqrt{l_1^2 + l_2^2}}, \frac{l_3}{\sqrt{l_1^2 + l_2^2}}\right)$$

- Note again, that not all lines homogeneous lines can be transformed into Euclidean representation ($\boldsymbol{l} = (0,0,1)^T$)

Intersection of lines

- The point $\boldsymbol{x}$ is in the intersection of the two lines $\boldsymbol{l_1}$ and $\boldsymbol{l_2}$ if

$$\boldsymbol{l_1^T x} = 0$$
$$\boldsymbol{l_2^T x} = 0$$

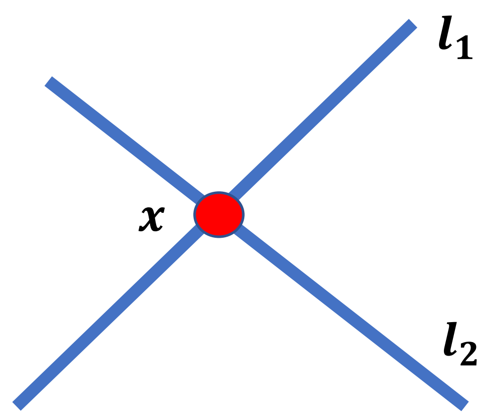
- We know that the triple products

$$\boldsymbol{l_1^T}(\boldsymbol{l_1} \times \boldsymbol{l_2}) = 0$$
$$\boldsymbol{l_2^T}(\boldsymbol{l_1} \times \boldsymbol{l_2}) = 0$$

- Therefore we can conclude that the intersection of two lines in homogeneous coordinates is simply the cross product

$$\boldsymbol{x} = \boldsymbol{l_1} \times \boldsymbol{l_2}$$

```
x=np.cross(l1,l2)
```



Duality

- The incidence relation between points and lines is symmetric, i.e.

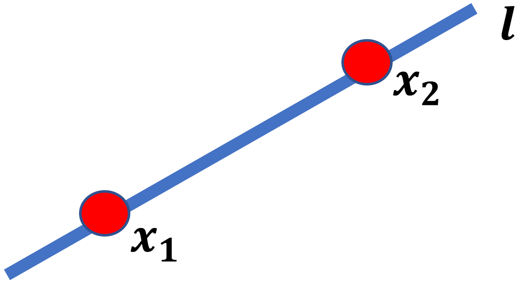
$$\boldsymbol{l}^T \boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{l} = 0$$

- Points and lines are dual entities in $\mathbb{P}^2$

- This **duality** means the that the line simultaneously going through two points can be calculated using the cross product again

$$\boldsymbol{l} = \boldsymbol{x_1} \times \boldsymbol{x_2}$$

```
l=np.cross(x1,x2)
```



Points at infinity

- What happens if we intersect two parallel lines?

$$\begin{pmatrix} \cos\theta \\ \sin\theta \\ -d_1 \end{pmatrix} \times \begin{pmatrix} \cos\theta \\ \sin\theta \\ -d_2 \end{pmatrix} = (\cos\theta - \sin\theta) \begin{pmatrix} d_1 - d_2 \\ d_2 - d_1 \\ 0 \end{pmatrix}$$

- The resulting point has $x_3 = 0$, i.e. there is no corresponding point in Euclidean space in this case

- We call these points **ideal points** or **points at infinity**

Line at infinity

- Now what happens if we calculate the line joining two of these points at infinity?

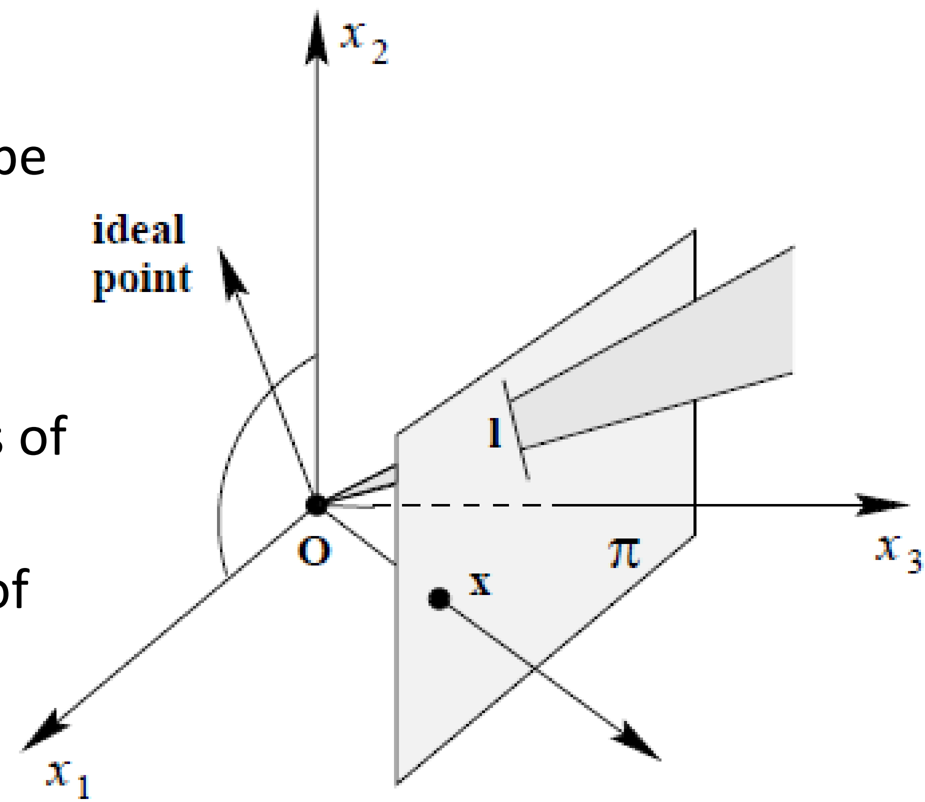
$$\begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} \times \begin{pmatrix} y_1 \\ y_2 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ x_1 y_2 - x_2 y_1 \end{pmatrix}$$

- The resulting line is $\boldsymbol{l}_\infty = (0,0,1)^T$, for which we already saw that there is no corresponding Euclidean representation
- We call $\boldsymbol{l}_\infty$ the **line at infinity** because all points at infinity are on this one line
- A big advantage of projective spaces is that elements at infinity (e.g. vanishing points, horizon) can be seamlessly represented

Geometric intuition

- All points not at infinity have $x_3 \neq 0$, therefore we can always normalise such homogeneous vectors so that the last component is $x_3 = 1$

- This means, Euclidean space can be visualised as the plane $x_3 = 1$ in the 3-d space into which $\mathbb{P}^2$ is embedded
- Euclidean points are intersections of lines with this plane
- Euclidean lines are intersections of planes with this plane



Normalisation

- Projective entities are only defined up to scale, which means the expression

$$\boldsymbol{x} = \boldsymbol{y}$$

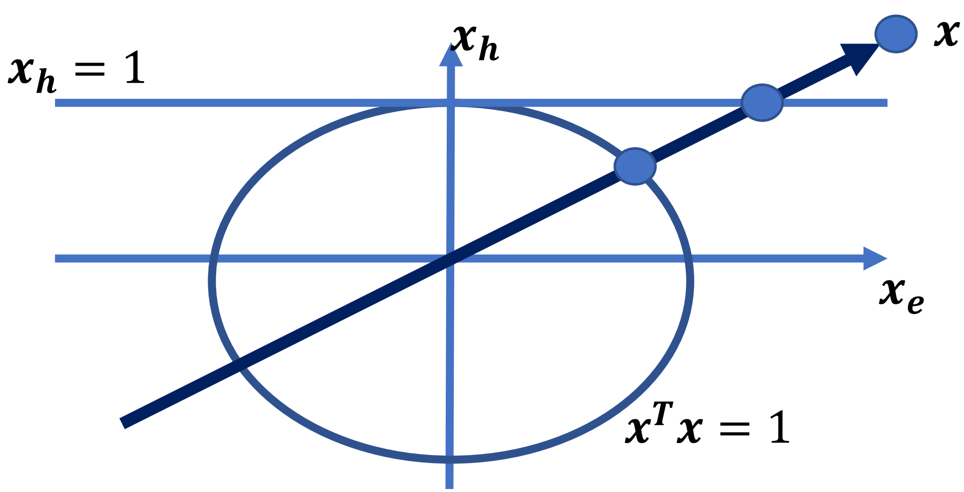
- can mean either of two things
  - It can be an assignment, i.e. the homogeneous vector $\boldsymbol{x}$ is assigned the values of the homogeneous vector y
  - It can be a test for equality, in which case it is short for

$$\exists \lambda \neq 0 : \boldsymbol{x} = \lambda \boldsymbol{y}$$

- It is important to keep this in mind and not confuse equality of homogeneous entities with equality of the representation vectors

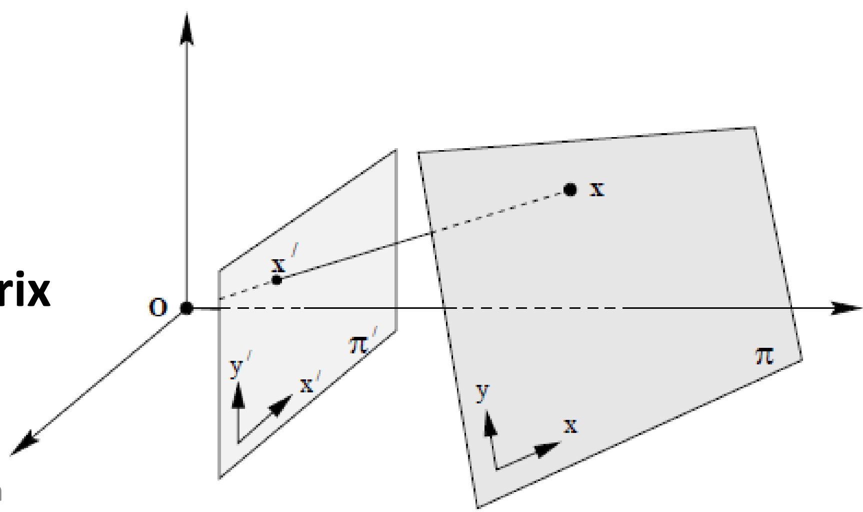
Normalisation

- If we want to make the representation vectors equal, we can normalise the representation
- Geometrically, a homogeneous entity $\boldsymbol{x} = (\boldsymbol{x}_{\boldsymbol{e}}, x_h)$ can be considered as the line (excluding **0**) connecting the origin with the representation vector in the embedding space
- We have already seen that dividing by the homogeneous component we can move the representation vector to the Euclidean plane $x_h = 1$ (**Euclidean normalisation**)
- We can also normalise the representation to the unit sphere by dividing by the length of the representation vector (**Spherical normalisation**)



Projective transformations

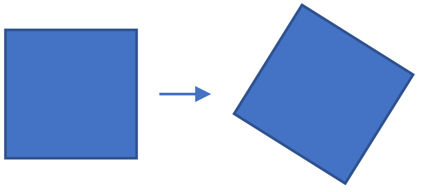
- A **homography** is an invertible mapping $h: \mathbb{P}^2 \rightarrow \mathbb{P}^2$ that preserves co-linearity of points, i.e. if x_1, x_2, x_3 are on the same line, then so are the transformed points $h(x_1), h(x_2), h(x_3)$
- A homography can always be represented by a non-singular 3×3 matrix $\boldsymbol{H}$ and applied to a homogeneous vector as matrix-vector product
$$\boldsymbol{x}' = \boldsymbol{H}\boldsymbol{x}$$
- Note, that this transformation (despite being a matrix-vector product) is **not linear** because it operates on homogeneous vectors
- Because of the homogeneity, the equation can be multiplied by a non-zero factor, and all matrices $\boldsymbol{H} \equiv \lambda \boldsymbol{H}$ are again equivalent
- Therefore also $\boldsymbol{H}$ is a **homogeneous matrix**



Isometries

- Transformations that preserve Euclidean distances are called **isometries**
- Every isometry can be written as homogeneous equation (i.e. up to scale) as follows (with $\epsilon = \pm 1$ to allow for non-orientation preserving rotation matrices)

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



- Or short

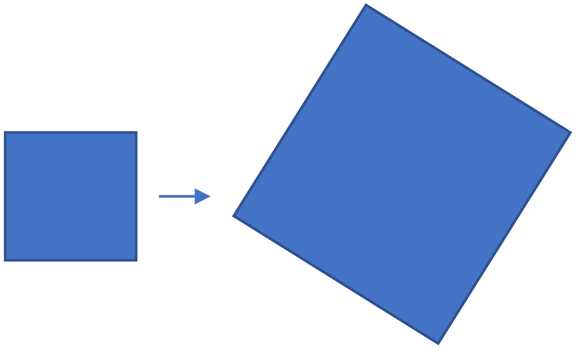
$$\boldsymbol{x}' = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} \boldsymbol{I}_3 & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \boldsymbol{x}$$

- Note, that unlike with regular linear algebra in Euclidean space we are able to <u>easily represent translation</u> in a matrix-vector product
- Also note, that we can concatenate these operations as matrix-matrix products of a rotation followed by a translation

Similarity transformations

- Transformations that preserve angles are called **similarities**
- They are isometries composed with an isotropic scaling, i.e. they do not only rotate and translate objects but also make them bigger or smaller
- They can be written as homogeneous equations

$$\boldsymbol{x}' = \begin{pmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \boldsymbol{x}$$



- Similarity transformations preserve “shape”, but not size

Affine transformations

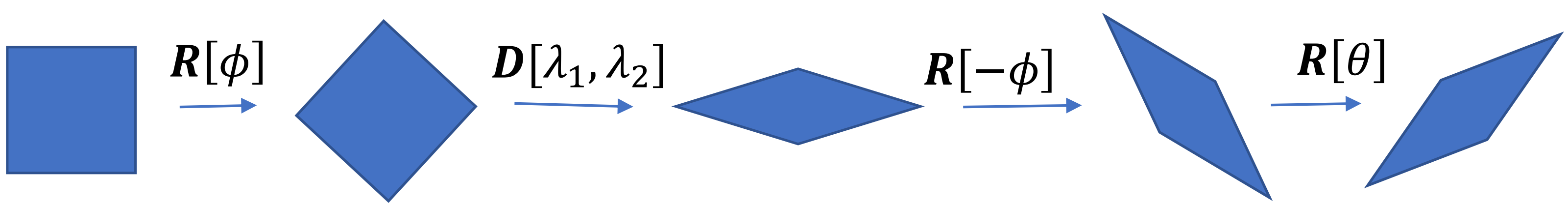
- A non-singular linear transformation followed by a translation is called **affinity**
- They can be written as homogeneous equations

$$\boldsymbol{x}' = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \boldsymbol{x}$$

- To understand what this class of transformations does it is helpful to look at the singular value decomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T = (\boldsymbol{U}\boldsymbol{V}^T)(\boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T) = \boldsymbol{R}[\theta]\left(\boldsymbol{R}[-\phi]\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}\boldsymbol{R}[\phi]\right)$$

- Which means the shape is rotated by ϕ, “squeezed” along the axes by the singular values λ_1 and λ_2, rotated back, and then finally rotated by θ



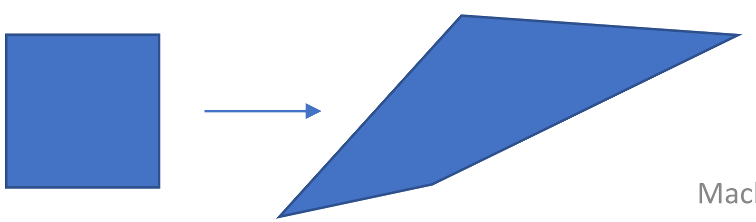
- Affinities preserve elements at infinity, i.e. parallel lines are mapped onto parallel lines, and ratios of areas

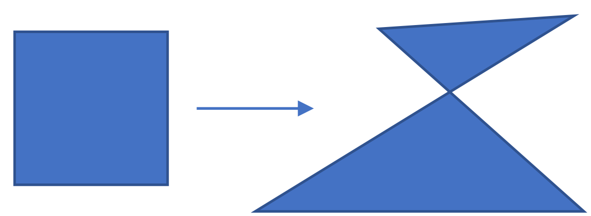
Projective transformations

- The most generic homography is

$$\boldsymbol{x}' = \boldsymbol{H}\boldsymbol{x} = \begin{pmatrix} \boldsymbol{A} & \boldsymbol{t} \\ \boldsymbol{v}^T & v \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{K} & \boldsymbol{0} \\ \boldsymbol{0}^T & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{v}^T & v \end{pmatrix} \boldsymbol{x}$$

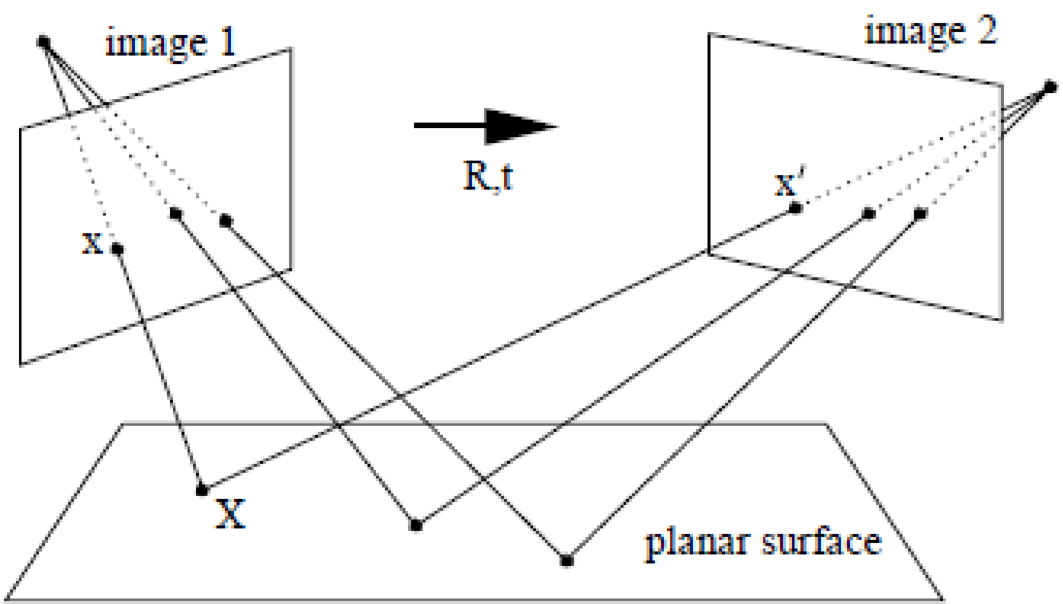
- The projective distortion $(\boldsymbol{v}^T, v)$ will move elements from infinity into Euclidean space and vice versa
- It preserves incidence relations between points and lines and therefore co-linearity; it also preserves the **cross-ratio**, i.e. the ratio of ratios, of distances
- However, it might break the Euclidean topology, because parallel lines can now intersect

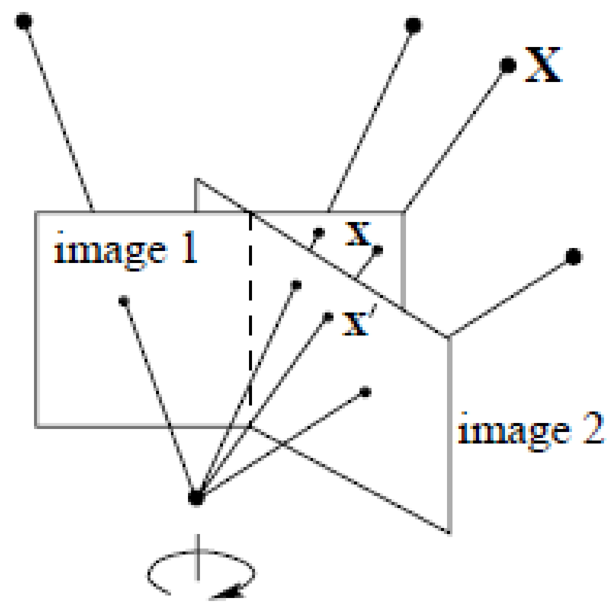




Projective transformations

- Two images of a planar surface are related by a homography
- Two rotated images are also related by a homography, because (as we will see later) in this case every pixel corresponds to a point on the plane at infinity





Example homography

Example homography

Estimating 2d homographies

- A 2d homography has 8 degrees of freedom, hence it is uniquely determined by 4 point correspondences between two images $(\boldsymbol{x}_i' \leftrightarrow \boldsymbol{x}_i)$
- For each of these correspondences the homogeneous equation $\boldsymbol{x_i}' = \boldsymbol{H}\boldsymbol{x_i}$ must hold
- However, it is only valid up to scale, i.e. the equality in a homogeneous equation only means that the two vectors are co-linear in $\mathbb{R}^3$
- We can test co-linearity in $\mathbb{R}^3$ using the cross product

$$\exists \lambda \neq 0 \colon \boldsymbol{x}_i' = \lambda \boldsymbol{H}\boldsymbol{x}_i \Leftrightarrow \boldsymbol{x}_i' \times \boldsymbol{H}\boldsymbol{x}_i = \boldsymbol{0}$$

Skew symmetric matrix

- The cross product $\boldsymbol{x} \times \boldsymbol{y}$ can also be expressed as matric-vector product $\boldsymbol{S}[\boldsymbol{x}]\boldsymbol{y}$ using the skew-symmetric matrix

$$\boldsymbol{S}[\boldsymbol{x}] = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$$

- Therefore we can rewrite the conditions as

$$\boldsymbol{S}[\boldsymbol{x}_i']\boldsymbol{H}\boldsymbol{x}_i = \boldsymbol{0}$$

Kronecker product

- The matrix $\boldsymbol{H}$ is in the middle of this equation

$$\boldsymbol{S}[\boldsymbol{x}_i']\boldsymbol{H}\boldsymbol{x}_i = \boldsymbol{0}$$

- To extract the $\boldsymbol{H}$ from the middle, the Kronecker-product is very useful
- It enables the re-writing a product of three matrices as follows (the symbol $\otimes$ is NOT the convolution in this context):

$$vec[\boldsymbol{ABC}] = \left(\boldsymbol{C}^{\boldsymbol{T}} \otimes \boldsymbol{A}\right) vec[\boldsymbol{B}]$$

Kronecker product

- The matrix $\boldsymbol{H}$ is in the middle of this equation

$$\boldsymbol{S}[\boldsymbol{x}_i']\boldsymbol{H}\boldsymbol{x}_i = \boldsymbol{0}$$

- We can therefore rewrite it as follows

$$\boldsymbol{A}_i\, vec[\boldsymbol{H}] = \left(\boldsymbol{x}_i^T \otimes \boldsymbol{S}[\boldsymbol{x}_i{}']\right) vec[\boldsymbol{H}] = \boldsymbol{0}$$

```
A=np.kron(x1.T, [[0,-x2[2],x2[1]],
                  [x2[2],0,-x2[0]],
                  [-x2[1],x2[0],0]])
```

- Note, that only two out of these three equations are linearly independent, so we can choose the first two rows of each $\boldsymbol{A}_i$

The DLT algorithm

- We can now stack these equations for all four (or more) points and get the homogeneous equation system

$$\boldsymbol{A}\, vec[\boldsymbol{H}] = \begin{pmatrix} \boldsymbol{A_1} \\ \vdots \\ \boldsymbol{A_n} \end{pmatrix} vec[\boldsymbol{H}] = \boldsymbol{0}$$

- The solution is the null-space of the singular matrix $\boldsymbol{A}$, which can be obtained as the singular vector corresponding to the smallest singular value

```
U,S,V = np.linalg.svd(A)
H = V[8,:].reshape(3,3).T
```

Application: Image Panoramas

Points in 3d

- The concept of projective spaces can be transferred into higher dimensions
- A 3d $\boldsymbol{X} \in \mathbb{P}^3$ in homogeneous coordinates is analogous to the 2d case a 4d vector

$$\boldsymbol{X} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix}$$

- Again, points with $X_4 = 0$ are at infinity, and for all other points Euclidean normalisation yields the 3d coordinate

$$(X, Y, Z) = \left(\frac{X_1}{X_4}, \frac{X_2}{X_4}, \frac{X_3}{X_4} \right)$$

Planes in 3d

- The dual entity to the 3d point is the plane
- A point $\boldsymbol{X}$ is on a plane $\boldsymbol{A}$ if

$$\boldsymbol{A}^T\boldsymbol{X} = 0$$

- Three points define a plane in 3d, which then has to be the null-space of the matrix of stacked point vectors

$$\begin{pmatrix} \boldsymbol{X}_1^T \\ \boldsymbol{X}_2^T \\ \boldsymbol{X}_3^T \end{pmatrix} \boldsymbol{A} = 0$$

- Because of duality the same is also true for the 3d point at the intersection of 3 planes
- The plane at infinity is $\boldsymbol{A}_\infty = (0,0,0,1)^T$

Lines in 3d

- In 3d space there is another linear element, the line
- A line is the connection of two 3d points $\boldsymbol{X} = (\boldsymbol{X_0}, X_h)^T$ and $\boldsymbol{Y} = (\boldsymbol{Y_0}, Y_h)^T$
- To represent a 3d line we use a 6-vector with 4 degrees of freedom, i.e. 2 constraints **(Plücker coordinates**)

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L_h} \\ \boldsymbol{L_0} \end{pmatrix} = \begin{pmatrix} X_h \boldsymbol{Y_0} - Y_h \boldsymbol{X_0} \\ \boldsymbol{X_0} \times \boldsymbol{Y_0} \end{pmatrix}$$

- This representation is homogeneous (1 constraint) and the two 3-vector components are orthogonal ($\boldsymbol{L_h^T L_0} = 0$; 2nd constraint)

- In accordance with the other definitions, lines at infinity are $\boldsymbol{L_h} = \boldsymbol{0}$

Homographies in 3d

- In analogy to the 2d case, a 3d homography is a non-singular homogeneous 4×4 matrix that transforms 3d points as
$$\boldsymbol{X}' = \boldsymbol{H}\boldsymbol{X}$$

- While the 2d homography had 8 degrees of freedom ($\boldsymbol{H}_{2d} \in \mathbb{P}^8$), the 3d homography has 15 degrees of freedom ($\boldsymbol{H}_{3d} \in \mathbb{P}^{15}$)
- Again, a hierarchy of transformations can be defined with the most useful one the Euclidean motion in 3d space with 6dof (3 translation + 3 rotation parameters)
$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix}$$
- And the 3d similarity (7dof), which adds an addition scalar scale parameter
$$\boldsymbol{H} = \begin{pmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{pmatrix}$$

Rotations in 3d

- Rotations in 3d have 3 degrees of freedom, classically represented by the rotation angles (ω, ϕ, κ) around the three coordinate axis

$$R_1[\omega] = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{pmatrix}$$

$$R_2[\phi] = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix}$$

$$R_3[\kappa] = \begin{pmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- A full 3d rotation is then $R[\omega, \phi, \kappa] = R_3[\kappa]R_2[\phi]R_1[\omega]$
- Rotations in 3d are not commutative, therefore ordering matters
- Also the directions are not well standardised, so my advice with regards to this angle representation: **Avoid at all cost!**

Rotations in 3d

- A better representation uses the rotation axis $\boldsymbol{r}$ and the rotation angle α

$$\boldsymbol{R}[\boldsymbol{r}, \alpha] = \cos\alpha\,\boldsymbol{I} + (1 - \cos\alpha)\boldsymbol{D}[\boldsymbol{r}] + \sin\alpha\,\boldsymbol{S}[\boldsymbol{r}]$$

- Where $\boldsymbol{S}[\boldsymbol{r}]$ is again the skew symmetric matrix defined earlier and $\boldsymbol{D}[\boldsymbol{r}] = \boldsymbol{r}\boldsymbol{r}^T$ is the outer product matrix

- Note that this definition requires the rotation axis to be spherically normalised, i.e. $\boldsymbol{r}^T\boldsymbol{r} = 1$

Quaternions

- A quaternion $\boldsymbol{q} = (q_0, q_1, q_2, q_3)^T$ is a 4-vector comprising a real part q_0 and an imaginary part $\boldsymbol{q_i} = (q_1, q_2, q_3)^T$
- As a mathematical concept it is a generalisation of complex numbers, but we only use it to represent rotations
- Due to the 3/1 split of components it is very related to homogeneous 3d points
- The quaternion rotation matrix is defined as

$$\boldsymbol{R}[\boldsymbol{q}] = \frac{1}{\boldsymbol{q}^T\boldsymbol{q}}\Big(\big(q_0^2 - \boldsymbol{q}_i^T\boldsymbol{q}_i\big)\boldsymbol{I} + 2\boldsymbol{D}[\boldsymbol{q_i}] + 2q_0\boldsymbol{S}[\boldsymbol{q_i}]\Big)$$

- Note the spherical normalisation, which mean we can consider rotation quaternions as homogeneous entities

Rodrigues representation

- If we want to work with 3 parameters instead of the homogeneous quaternion (4 parameters + 1 constraint) we can use the Rodrigues representation

$$\boldsymbol{m} = (a, b, c)$$

- which corresponds to the quaternion $\boldsymbol{q} = \left(1, \frac{a}{2}, \frac{b}{2}, \frac{c}{2}\right)$ and therefore defines the following rotation

$$\boldsymbol{R}[\boldsymbol{m}] = \frac{1}{4 + \boldsymbol{m}^T \boldsymbol{m}} \left((4 - \boldsymbol{m}^T \boldsymbol{m}) \boldsymbol{I} + 2 \boldsymbol{D}[\boldsymbol{m}] + 4 \boldsymbol{S}[\boldsymbol{m}] \right)$$

Useful properties of rotations

- A rotation matrix is always orthogonal

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$$

- Or equivalent, if $\boldsymbol{R}$ is the forward rotation then $\boldsymbol{R}^T = \boldsymbol{R}^{-1}$ is the backward rotation

- A small rotation can be approximated with the skew-symmetric matrix corresponding to the cross-product

$$\boldsymbol{R}[\boldsymbol{r}, d\alpha] \approx \boldsymbol{I} + d\alpha\boldsymbol{S}[\boldsymbol{r}]$$

Projective mapping from 3d to 2d

- A camera performs a projective mapping of 3d points $\boldsymbol{X}$ to 2d points $\boldsymbol{x}$
- This can be represented by a homogeneous 4×3 projection matrix $\boldsymbol{P} \in \mathbb{P}^{11}$

$$\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{X}$$

- In theory the DLT algorithm can be used to determine this matrix from 3d-2d correspondences, using

$$(\boldsymbol{X}^T \otimes \boldsymbol{S}[\boldsymbol{x}'])vec[\boldsymbol{P}] = \boldsymbol{0}$$

Thank you for your attention!