



Università
Ca' Foscari
Venezia

Computer Vision

Geometric primitives and
transformations

Filippo Bergamasco (filippo.bergamasco@unive.it)

<http://www.dais.unive.it/~bergamasco>

DAIS, Ca' Foscari University of Venice

Academic year 2017/2018



Università
Ca' Foscari
Venezia

Geometric primitives

Geometric primitives form the basic building blocks used to describe 2D and 3D shapes.

We will study the basic geometric primitives (points, lines, conics) and the transformations that can be defined between them

The framework of projective geometry allow us to describe such transformations in a powerful generic way.



Università
Ca' Foscari
Venezia

Points

Points lying on an Euclidean 2D plane (like the image plane) are usually described as vectors:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$$

This is a common way to reason about points but has some limitations. For example, we cannot describe points at infinity...

...Other alternatives are possible!



Università
Ca' Foscari
Venezia

2D Projective space

Since the imaging apparatus usually behaves like a pinhole camera model, many of the transformations that may happen can be described as projective transformations.

> This offer a general and powerful way to work with points, lines and conics

The **2D projective space** is simply defined as:

$$\mathbb{P}^2 = \mathbb{R}^3 - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$



Università
Ca' Foscari
Venezia

Homogeneous coordinates

A point in a 2D Euclidean plane can be described in homogeneous coordinates (2D projective space) as follows:

$$\mathbf{x} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} \in \mathbb{P}^2, \quad w \in \mathbb{R} - \{0\}$$

This implies that:

- There are infinitely many ways to describe a point \mathbf{x}
- points in euclidean space are represented by all the equivalence classes of three dimensional vectors where two elements are in the same class if they differ by a non zero scale factor.



Università
Ca' Foscari
Venezia

Homogeneous coordinates

From Euclidean to homogeneous:

Just add 1 to the last component

$$\begin{bmatrix} x \\ y \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

From homogeneous to Euclidean:

Divide by the last component (if not zero)

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x/w \\ y/w \end{bmatrix}$$



Università
Ca' Foscari
Venezia

Homogeneous coordinates

All the points

$$\begin{bmatrix} x \\ y \\ 0 \end{bmatrix} \in \mathbb{P}^2$$

are called **ideal points** or **points at infinity** and do not have an equivalent inhomogeneous representation (in Euclidean plane).



Università
Ca' Foscari
Venezia

2D Lines

A line in the Euclidean plane can be described as the locus of points $p=(x,y)$ so that:

$$ax + by + c = 0$$

In projective space, the same line can be represented as:

$$\begin{bmatrix} X & Y & w \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$



Università
Ca' Foscari
Venezia

2D Lines

$$\begin{bmatrix} X & Y & w \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

This represent each
point on the line

This represent the line
itself

in P^2 points and lines are described in the same way!

This leads to a simple expression to find the
intersection \mathbf{x} of two lines \mathbf{u} and \mathbf{u}' using the cross
product:

$$\mathbf{x} = \mathbf{u} \times \mathbf{u}'$$



2D Lines intersection

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \times \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

What happens if the lines are parallel?

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \times \begin{bmatrix} a \\ b \\ c' \end{bmatrix} = \begin{bmatrix} bc' - cb \\ ca - ac' \\ 0 \end{bmatrix}$$



Università
Ca' Foscari
Venezia

Line and points

Similarly, the line l joining two points p_1 and p_2 can be expressed as:

$$l = p_1 \times p_2$$



Università
Ca' Foscari
Venezia

2D Lines

We can normalize a line

$$\mathbf{l} = \begin{bmatrix} n_1 \\ n_2 \\ d \end{bmatrix}$$

So that $\sqrt{n_1^2 + n_2^2} = 1$

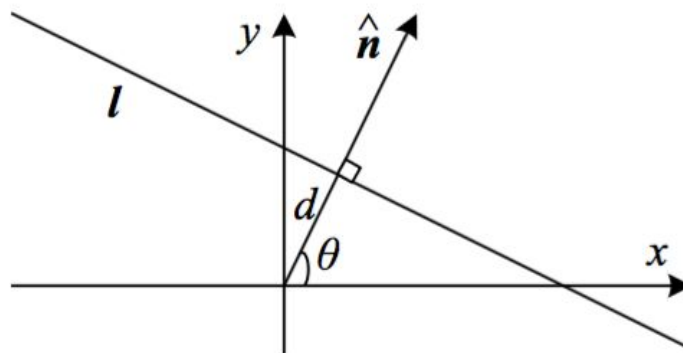


Università
Ca' Foscari
Venezia

2D Lines

With a normalized line,

- The vector $(n_1 \ n_2)^T$ is the line normal and d is the line distance to the origin



- The point-line distance d from a line \mathbf{l} and a point \mathbf{x} can be computed as

$$d = |\mathbf{l}^T \mathbf{x}|$$



Università
Ca' Foscari
Venezia

Conics

Conics that can be represented by second degree polynomials in the form $ax^2+bx+cy^2+dx+ey+f=0$

In homogeneous coordinates the conic can be represented by a matrix

$$\mathbf{Q} = \begin{bmatrix} a & b & d \\ b & c & f \\ d & f & g \end{bmatrix}$$

Such that, for each point $\mathbf{x}=(x \ y \ 1)^T$, it holds that

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



Conics

Depending on the parameters, we can obtain different conics:

- Circles
- Ellipses
- Parabolae
- Hyperbolae

$$\mathbf{Q} = \begin{bmatrix} a & b & d \\ b & c & f \\ d & f & g \end{bmatrix}$$

Any projectivity transforming the projective space P^2 in which the conic lie will result in a (possible different) type of conic.

Ellipses present in a scene are interesting geometric primitives since they remain ellipses after being projected into the image plane.



Università
Ca' Foscari
Venezia

2D projectivities

A planar projective transformation is a linear transformation in P^2 that can be represented by any non-singular 3x3 matrix \mathbf{H} :

$$\begin{bmatrix} X' \\ Y' \\ W' \end{bmatrix} = \mathbf{H}\mathbf{x} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

Any projective transformation **transform lines in lines** and **preserve the incidence**.



Università
Ca' Foscari
Venezia

2D Translation

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

Dof: 2

Preserves: Orientation



Università
Ca' Foscari
Venezia

2D Rotation

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

Dof: 1

Preserves: Lengths



Università
Ca' Foscari
Venezia

2D Rigid motion

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \mathbf{TR} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

Combination of a rotation \mathbf{R} and a translation \mathbf{T}

Dof: 3

Preserves: Lengths



Università
Ca' Foscari
Venezia

2D Similarity

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

Where s is a non-zero scale factor.

Rotation+Scale+Transl.

Dof: 4

Preserves: Angles between lines



Università
Ca' Foscari
Venezia

2D Affine transformation

$$\begin{bmatrix} X' \\ Y' \\ W \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

Dof: 6

Preserves: Parallellism of lines



Università
Ca' Foscari
Venezia

2D Projective transformation

$$\begin{bmatrix} X' \\ Y' \\ W' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ W \end{bmatrix}$$

Projective transformation is also called **homography**.
Since we work in P^2 , **H** is defined up to scale






Dof: 8 (not 9! Because any scale define the same transformation)

Preserves: Straight lines



Università
Ca' Foscari
Venezia

Transformations table

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \mathbf{I} & & \mathbf{t} \end{bmatrix}_{2 \times 3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \mathbf{R} & & \mathbf{t} \end{bmatrix}_{2 \times 3}$	3	lengths	
similarity	$\begin{bmatrix} s\mathbf{R} & & \mathbf{t} \end{bmatrix}_{2 \times 3}$	4	angles	
affine	$\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$	8	straight lines	



Università
Ca' Foscari
Venezia

Spatial transformations

The projective transformations discussed so far can be applied to the image domain to **transform the geometry of the image plane**

Given an image $f(\mathbf{x}) = f(x, y)$

And a transformation function $s : \mathbb{R}^2 \rightarrow \mathbb{R}^2$

A spatial transformation changes the image as following:

$$g(\mathbf{x}) = f(s(\mathbf{x}))$$

NOTE: Different from intensity transformations: $f(\mathbf{x}) = h(f(\mathbf{x}))$

Spatial transformations

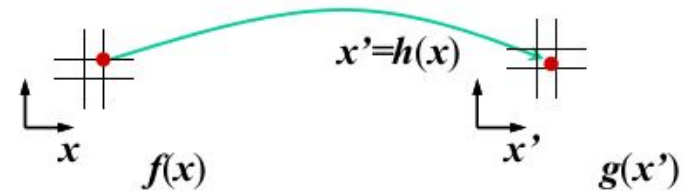
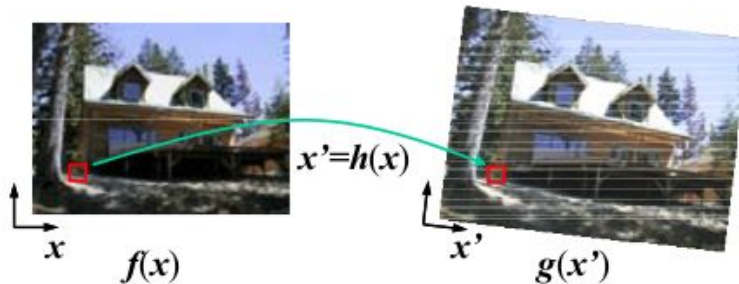
Example: image rotation

$$s(\mathbf{x}) = s(x, y) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Forward warp

How to perform the transformation? One simple way is via forward warping:



A pixel $f(\mathbf{x})$ is copied to its corresponding location $\mathbf{x}' = s(\mathbf{x})$ in image $g(\mathbf{x}')$.



Università
Ca' Foscari
Venezia

Forward warp

Problem: x' usually has a non-integer value, and $g(x')$ is not defined in that case

Possible solution1:

Round the value and copy the pixel there (Create cracks and holes)

Possible solution2:

Distribute the value among its n -neighbours in a weighted fashion (Cause aliasing and blur)



Università
Ca' Foscari
Venezia

Forward warp



Inverse warp

A preferable solution is to use inverse warping:



each pixel in the destination image $g(\mathbf{x}')$ is sampled from the original image at $\mathbf{x} = s^{-1}(\mathbf{x}')$



Università
Ca' Foscari
Venezia

Inverse warp

Advantages:

- No holes since $s^{-1}(\mathbf{x}')$ is defined for all values of \mathbf{x}' (in all the $g(\mathbf{x}')$ domain)

Problems:

- The transformation function must be invertible (not really a problem since our transformation matrix is non-singular)
- Point sampling may still occur at non integer locations
 - This is a well studied problem: **Image interpolation**



Università
Ca' Foscari
Venezia

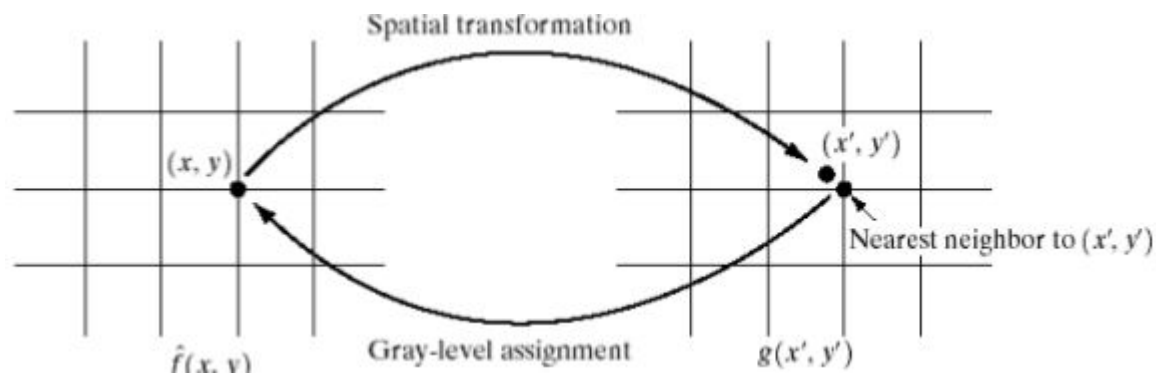
Nearest-neighbour interpolation

Interpolation: estimate the values using the information from nearby samples

Nearest-neighbour:

Use the image value at the closest integer location

$$g(\mathbf{x}') = f(\text{round}(s^{-1}(\mathbf{x}')))$$





Università
Ca' Foscari
Venezia

Nearest-neighbour interpolation

NN-interpolation is very fast but creates artefacts (blocks) especially if the transformation changes the scale (ie.zooming)





Università
Ca' Foscari
Venezia

Bilinear interpolation

Use the four points around $s^{-1}(\mathbf{x}')$ to get a better estimation:

$$g(\mathbf{x}') = \alpha f(x', y') + \beta f(x' + 1, y') + \gamma f(x', y' + 1) + \delta f(x' + 1, y' + 1)$$

$$(x' \ y') = \text{floor}(s^{-1}(\mathbf{x}'))$$

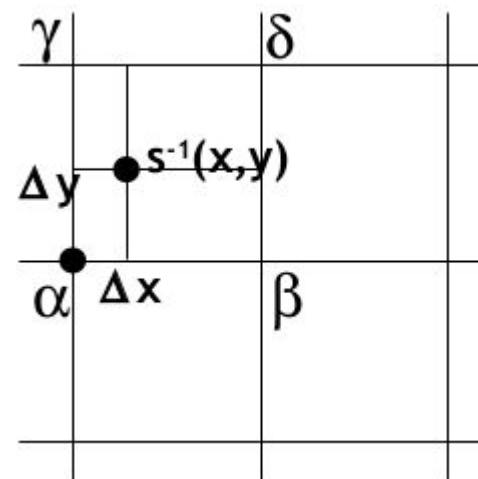
$$(\Delta x \ \Delta y) = s^{-1}(\mathbf{x}') - \mathbf{x}'$$

$$\alpha = (1 - \Delta x)(1 - \Delta y)$$

$$\beta = \Delta x(1 - \Delta y)$$

$$\gamma = (1 - \Delta x)\Delta y$$

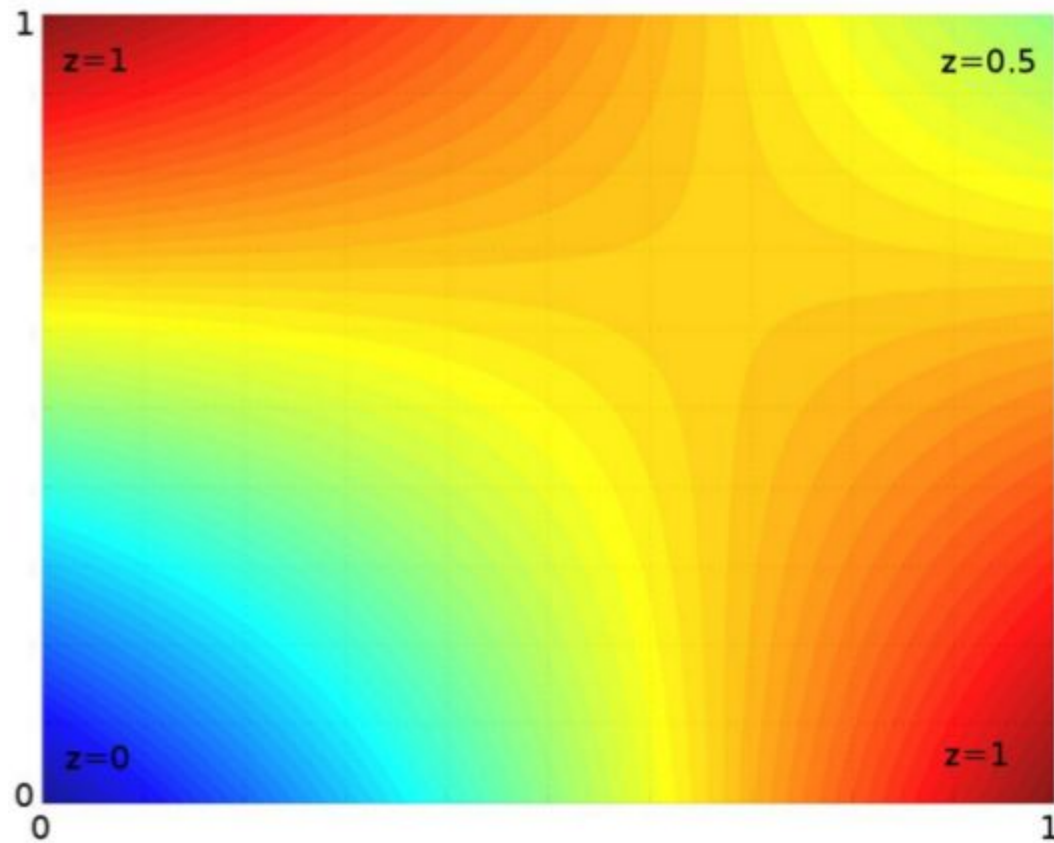
$$\delta = \Delta x\Delta y$$





Università
Ca' Foscari
Venezia

Bilinear interpolation





Università
Ca' Foscari
Venezia

Bilinear interpolation



NN-interpolation



Bilinear
interpolation