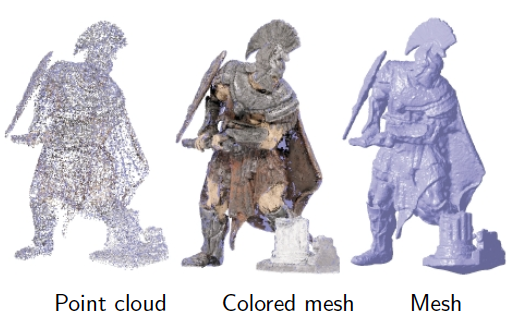
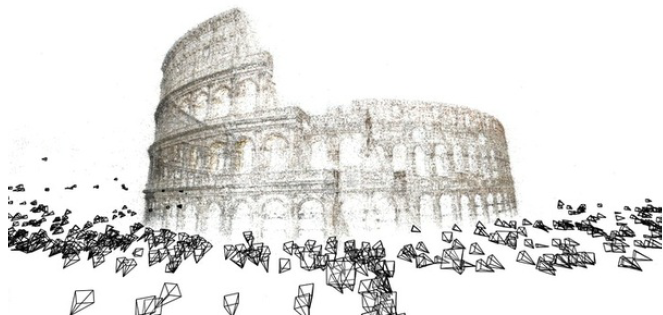
**2D projective geometry**

* **Multi-view systems:**
  + Dynamic scene + multiple fixed cameras:
    - Bullet time effect 🡪 we can pause the scene and have different perspective for the same time.
    - Motion capture (the main goal is to capture motion without use any marker).
  + Static scene + moving camera
    - Object scanning
    - Augmented reality
    - Match moving
  + Hybrids
    - Moving robots with two or more cameras
    - 3D cinema cameras
    - Google’s street-view car
  + Active lighting systems (3D sensors):
    - Time-of-flight
    - Structured light
* **3D reconstruction from multiple views:**
  + Types of reconstruction:
    - Calibrated case 🡪 multi-view stereo, shape from X
    - Non-calibrated case 🡪 structure from motion, simultaneous location and mapping (SLAM)
  + Input data 🡪 images from different view, video sequence, images from databases, data from 3D sensors…
  + Representation of the 3D 🡪 depth maps, voxels (like pixels but in 3D), meshes, point clouds (3D position of the reconstructed point)…





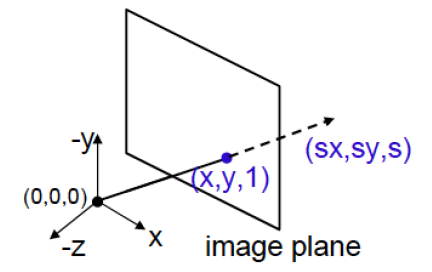
*Point cloud from multiple camera location*

**

*Depth map*

**Projective geometry:**

* 3D world are transformed into images with a projective transformation (linear transformation).
* **Transformation:**
  + Geometric properties not preserved: length, angles, distance ratio
  + Straight lines (except for lens distortion) or cross ratio (ratio of ratio of lengths) are preserved.
* **Properties:**
  + Linear transformation
  + Intersection between two lines, and the line that passes through two points are linear operations.
  + Points at infinity have natural expression (points at infinity not exist in Euclidean space).
* **2D projective geometry** 🡪 remove the projective distortion of flat objects and build mosaics.
* **Point in an image** 🡪 line in the projective space. The point depend on the distance of the plane (Z coordinate).
  + Point (0,0,0) is not in the projective space



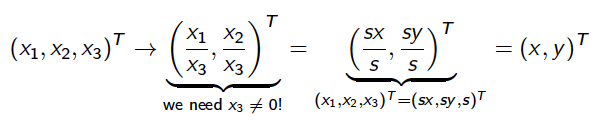
* **Projective space** 🡪 homogeneous coordinates (3 components), as the plane can change the distance and all the points in the line are the same, two points that varies on a factor are the same.



* **Coordinates change:**
  + From Euclidean to homogeneous 🡪 add a new component as 1.

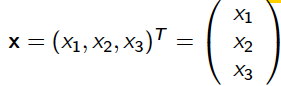


* + From homogeneous to Euclidean 🡪 divide all the components by the third one and remove the third coordinate.





*Euclidean coordinates*



*Homogeneous coordinates*



*Given two points we can define an equivalence relationship*

* **Points at infinity:**
  + If a point in homogeneous coordinate have the third coordinate equal to 0, have no representation in the Euclidean coorinates.
  + Also called ideal points.
* **Representation of a line:**
  + Given a point (x,y)T, a line that passes through that point is:
    - ax+by+c=0 🡪 line ecuation
    - Decomposing into point and line
      * x=(x,y,1)T
      * l=(a,b,c)T
  + Line equation can be written as:



*If a point accomplish the equation for a line, belongs to the line*

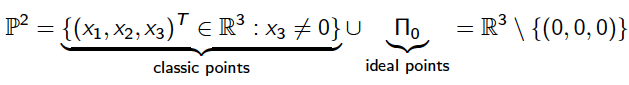
**

*Equivalent lines, the same as for a points*

* **Plane of the points at infinity:**

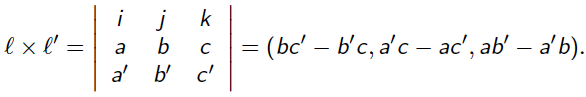


* **Projective space:**
  + Set of points formed with the classic points (the ones are not at infinity, are on images) and the ideal points.



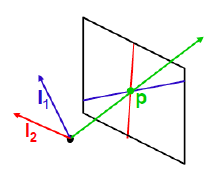
* **Intersection of two lines:**
  + The intersection of two lines is a point.
  + Computed as a cross product of the lines:





*Cross product*

* + Cross product result is a vector perpendicular to both, in this case is the lines are on the image the result is perpendicular to the image, so is a point on the projective space.



* + If are parallel lines:
    - Parallel lines have the same slope.
    - The intersection (vanishing point) is a point at infinity.



*Because a’b-ab’ is 0*

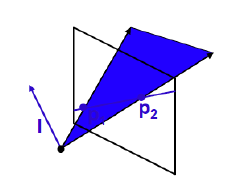
**

*Is (c-c’)(-d,a,0), as (c-c’) is a scalar the point is the same*

*c≠c’ because lines are parallel but not the same*

* **Line that joints two points:**
  + The line that passes through two points is the cross product of the coordinates of the points.
  + Both points are lines in the projective space. The result when we compute the cross product is a line perpendicular to both lines, so in this case is a line that is contained in the image plane.





*Line that joints 2 points*

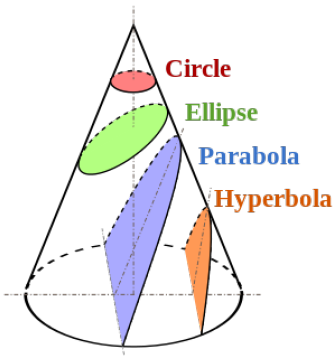
\*Points and lines are dual in the projective space. Exist a single line that joints two points and exist a single point that intersect two lines.

* **The line at infinity:**
  + This line joint points at infinity.
  + This line is 🡪 (0,0,1)T or another proportional.
  + The dot (scalar) product with any point at infinity (vanish point) is equal to 0.
  + The intersection between the line at infinity and another line give a point at infinity (vanish point) 🡪 x = (b,-a,0)T
* **Conics:**
  + Curves formed by the intersection of a cone with a plane at different angles (conic sections).
  + There are second order equations.



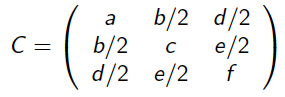
*According to the value of the constants we obtain different curves*

* + Curves that generates the intersection:
    - Circles
    - Ellipses
    - Parabolas
    - Hyperbolas



* + Points that belongs to the conic follow the next equation:





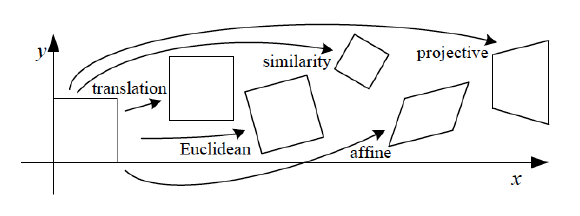
*This is a symmetric matrix*

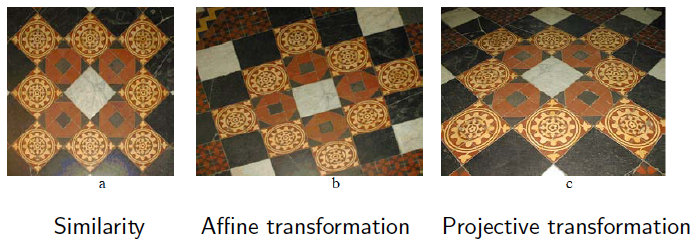
* + Two conics are equivalent if:



**Planar (2D) transformations:**

* Transformations that can be applied to images.





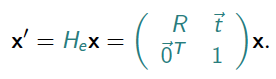
*In the affine transformations parallelisms are conserves and also the angles,*

*but not the lengths.*

*In the projective transformations, just straight lines and ratio of ratio are kept.*

**Isometry:**

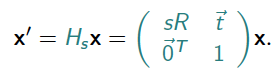
* Result of a translation and rotation of a point.
* Matrix He defines the translation and modifying the parameters we can achieve different results.
* **Degrees of freedom** 🡪 parameters that we can vary: 3
  + Rotation angle
  + 2 parameters for the translation (in both directions).
* This transformation keep the Euclidean norm (the modulus).
* **Invariant** 🡪 lengths, angles, parallelism…



*R is a matrix, not a scalar value*

**Similarity:**

* Transformation where there is a rotation, translation and scaling of the point.
* Matrix Hs defines the transformation and is the same as the isometry but adding a scale factor to the first two coordinates of the homogeneous coordinates of a point.

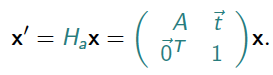


*‘s’ is a scalar value, ‘R’ a matrix*

* **Degrees of freedom**: 4
  + Rotation angle
  + 2 parameters for the translation
  + Scaling factor
* **Invariant** 🡪 angles, ration of length (division of two lengths), ratio of two areas.

**Affine:**

* Transformation that deforms the image.
* Matrix Ha defines the transformation.



* **A** 🡪 non-singular 2x2 matrix



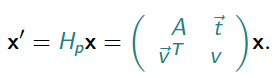
*‘A’ can be decomposed by a SVD into a product of*

*rotation matrix and diagonal matrix*

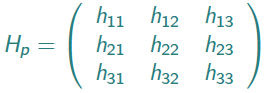
* **Degrees of freedom**: 6
  + 2 rotation angles
  + 2 scaling factor
  + 2 translation coefficients
* **Invariants** 🡪 parallel lines, ratio of parallel lengths, ratio of two areas, line at infinity (the points at infinity still at infinity, not near).
* Angles are not kept, for example transforming a square into a rhombus.

**Projective transformation:**

* Transformation that happens when we project a 3D scene into a 2D plane.
* Matrix Hp defines the transformation.



* Hp 🡪 non-singular 3x3 matrix.
  + Can be invertible.



* **Degrees of freedom**: 8
  + Matrix have 9 elements but there is one multiplicative factor.
* **Invariants**:
  + Concurrency 🡪 when lines intersect a point, when the transformation is applied continue intersecting that point.
  + Collinearity 🡪 if a line joint two points, after the transformation, those points will be joint by a line, so lines are kept.
  + Order of contact 🡪 the order of contact determines for how many derivative two functions are equal for a certain point.
    - 0 order 🡪 intersection of two lines
    - 1 order 🡪 tangency point
    - 2 order 🡪 tangency between a curve and oscillating circle.
  + Cross ratio 🡪 ratio of ratios

**Homographies:**

* Relates two images to transform one of them to the view of the other.
* This two images:
  + Are from the same plane in the 3D scene: flat object or plane in the 3D world.
  + Taken with a camera rotating about its center.
  + Taken from the same static camera varying the focal (zoom).
  + The whole scene is far away from the camera (depth of the objects in relation to the distance camera-object is not important.
* **Apply an homography:**