

Introduction to

Simultaneous Localization and Mapping

Spring Semester 2019

CS284

Course Convener:

Prof Laurent Kneip

lkneip@shanghaitech.edu.cn

Disclaimer



- Some of the lecturing material is naturally taken from publically available online material from other groups. Sources include:
 - Autonomous Systems Lab ETH Zurich
 - Research School of Engineering, ANU
 - Informatik Department, Uni Freiburg
 - MIT
 - ...
- By using the present material you confirm that
 - You use it only internally and for the purpose of education
 - You are aware that the lecture material may originate from other sources, even if explicit reference is occasionally missing

Where are we at?

- Today: Homogeneous coordinates & Transformations

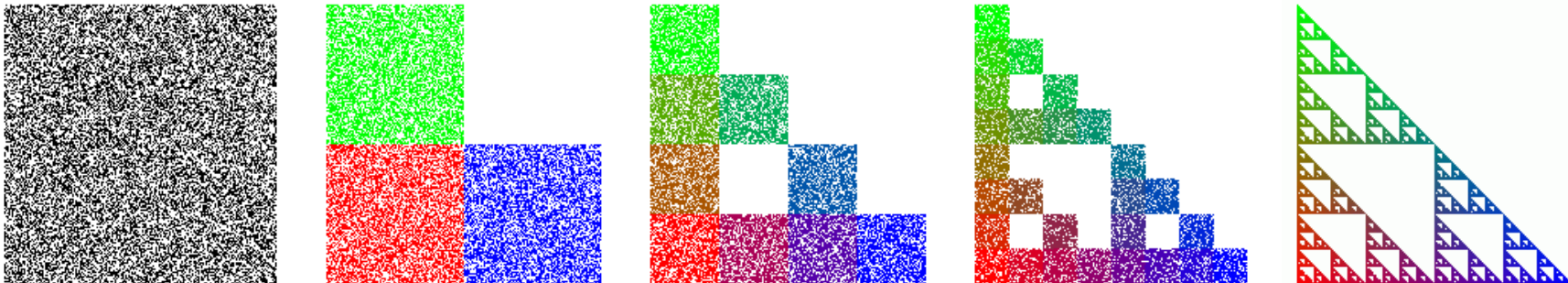
Week	Tuesday course	Thursday course
18th February	Introduction (what is SLAM, curriculum, course structure (homeworks, projects), course webpage)	Homogeneous coordinates, Transformations, etc.
25th February	Filtering methods (GF, PF)	Filtering methods (GF,PF)
4th March	Pose-only SLAM, Graph SLAM	Introduction to the projects
11th March	Camera as a sensor, camera calibration	Introduction into Visual SLAM, feature extraction, tracking, and matching
18th March	Feature extraction, tracking, and matching	Case study: MonoSLAM (why filter?)
25th March	Bootstrapping: The Relative Pose Problem	Full visual odometry pipeline, Non-linear optimization, Bundle adjustment
1st April	Non-linear optimization, Bundle adjustment	Place recognition, loop closure
8th April	Place recognition, loop closure, the absolute pose problem	Case study: ORB SLAM
15th April	Dense Tracking and Mapping (DTAM), Photometric vs geometric errors, semi-dense opt.	RGBD SLAM, Iterative closest points (Laser-point cloud registration), TSDF
22nd April	SLAM with Stereo and Multi-Perspective cameras	Midterm exam
29th April	Visual-Inertial SLAM	Project discussions/buffer
6th May	Dynamic and Multi-body SLAM	Project discussions/buffer
13th May	Semantic SLAM	Project Presentations

What is a Transformation?

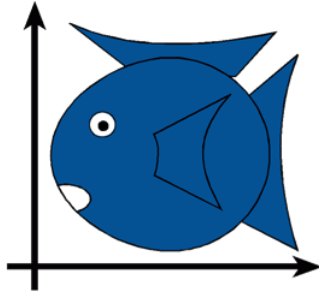
- 2D example:
 - Maps points (x, y) in one coordinate system to points (x', y') in another coordinate system

$$x' = ax + by + c$$

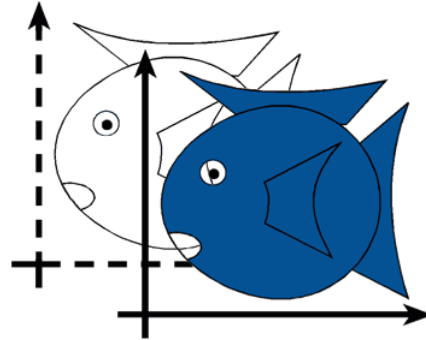
$$y' = dx + ey + f$$



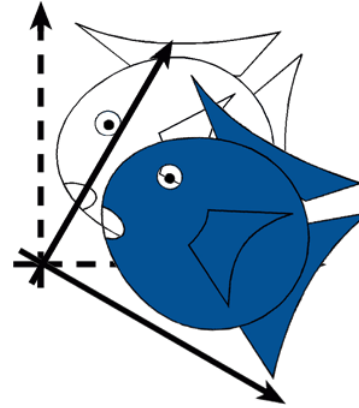
Simple Transformations



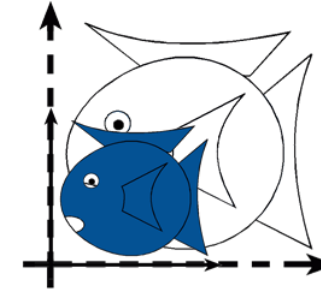
Identity



Translation



Rotation



Isotropic
(Uniform)
Scaling

- Can be combined!
- Invertible?
 - Yes, except scale = 0

Why do we need Transformations?



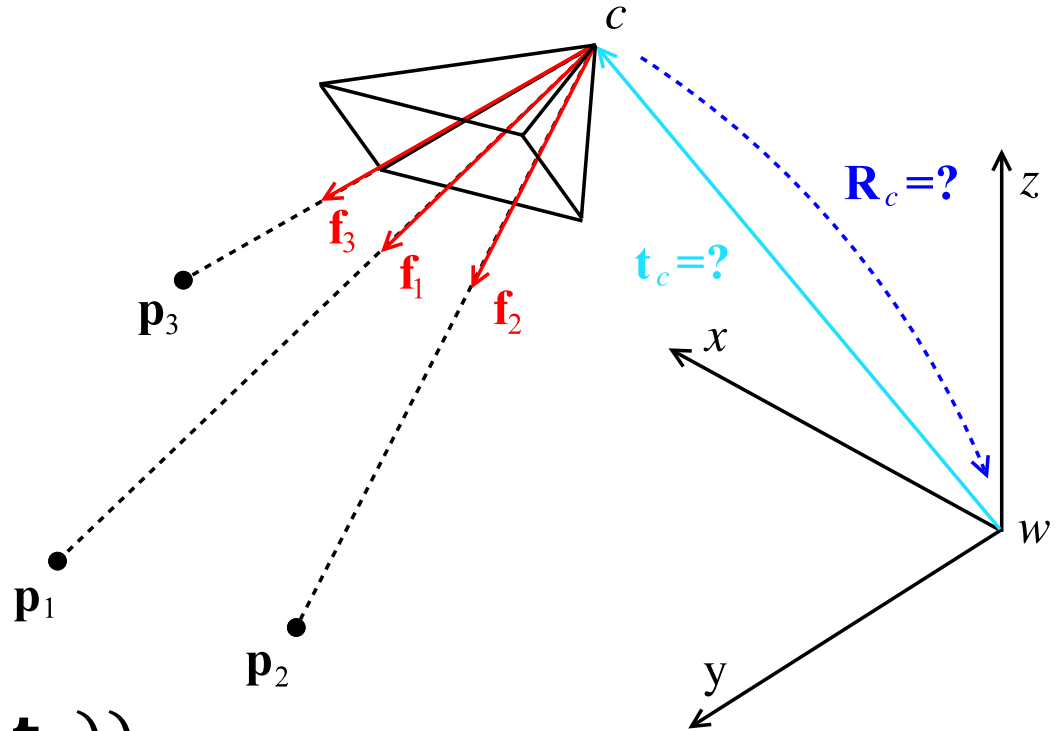
- Position objects in a scene (modelling)
- Change the shape of objects
- Model the position and orientation of a sensor/camera
- Projection for (virtual) cameras
- ...

Why do we need Transformations?

- Expresses Camera pose!

- Example: Absolute Pose
 - Objective:

$$\operatorname{argmax}_{\mathbf{t}_c, \mathbf{R}_c} \sum_i \mathbf{f}_i^t \cdot (\mathbf{R}_c^T \cdot (\mathbf{p}_i - \mathbf{t}_c))$$

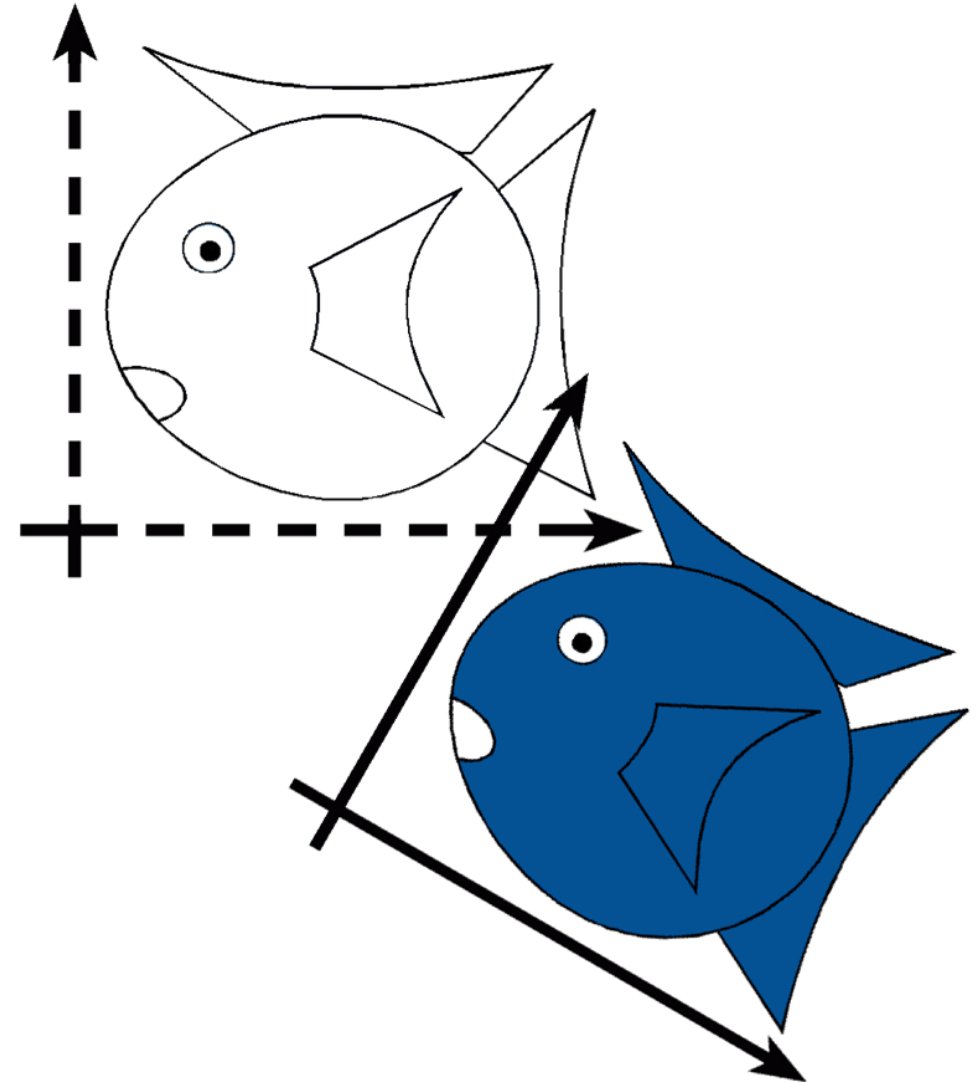


Classes of transformations

Rigid Body / Euclidean Transforms

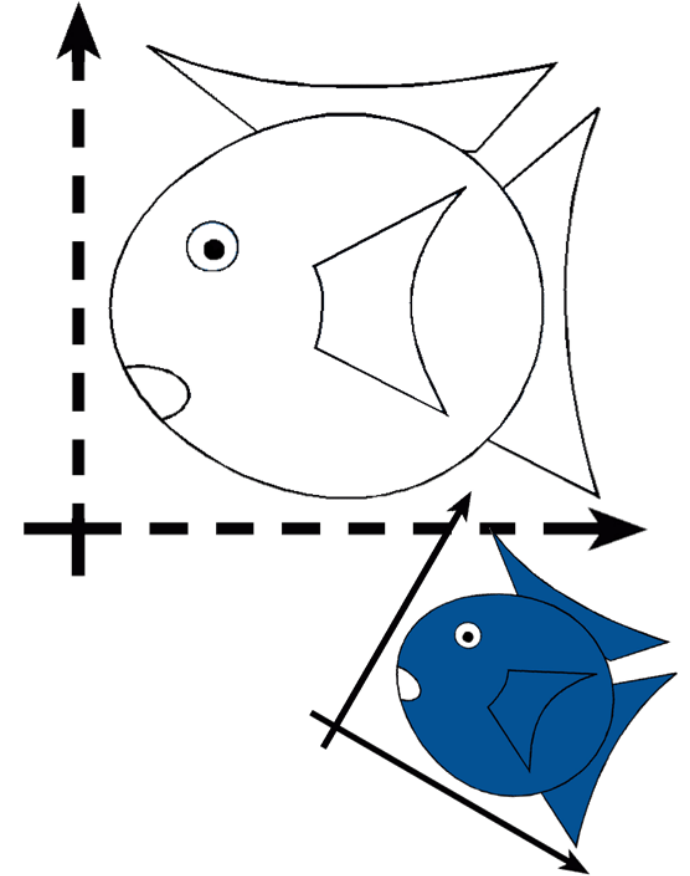
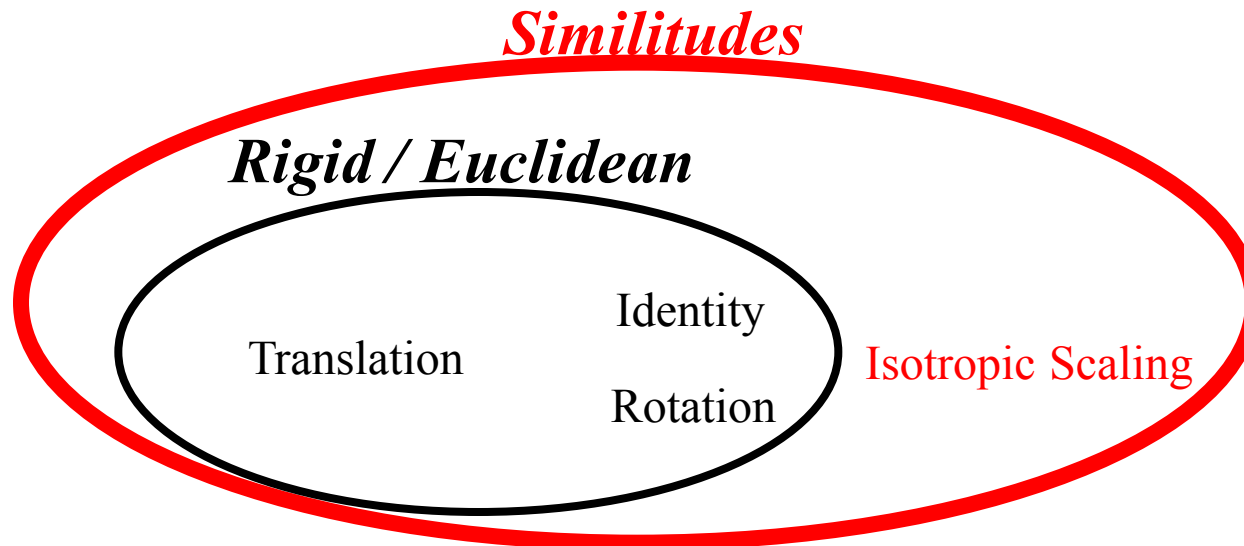
- Preserves distances
- Preserves angles

Rigid / Euclidean



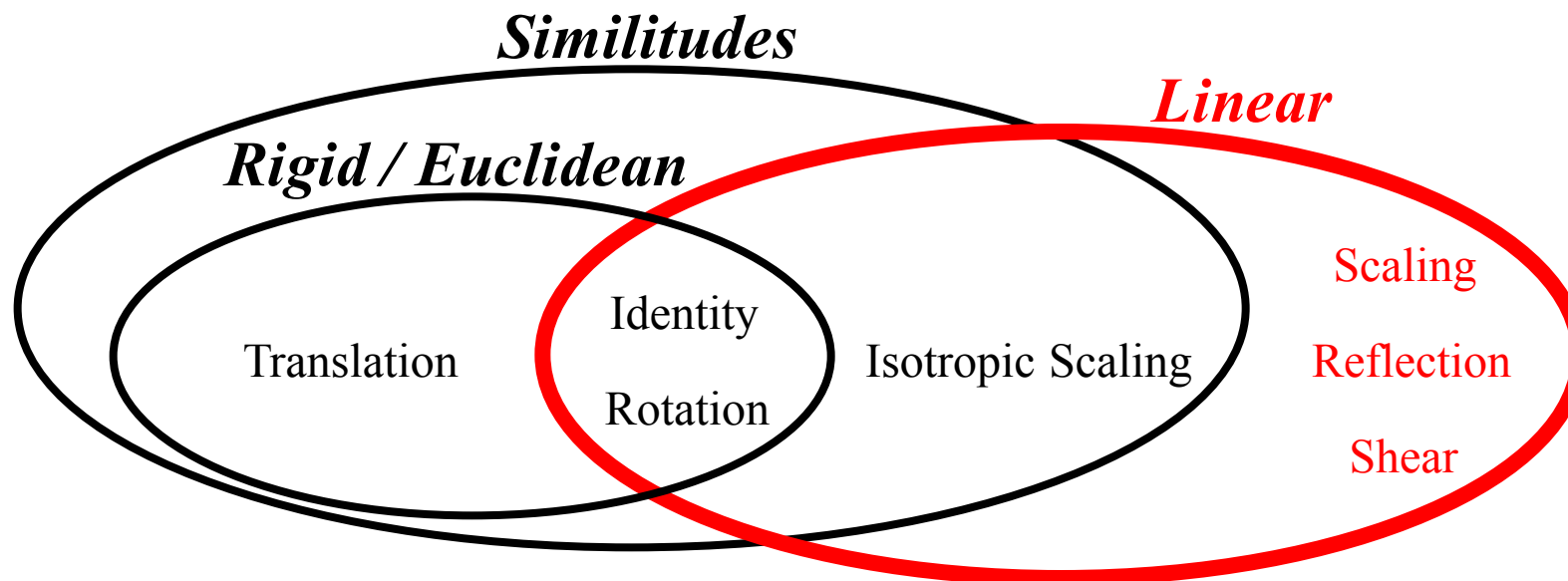
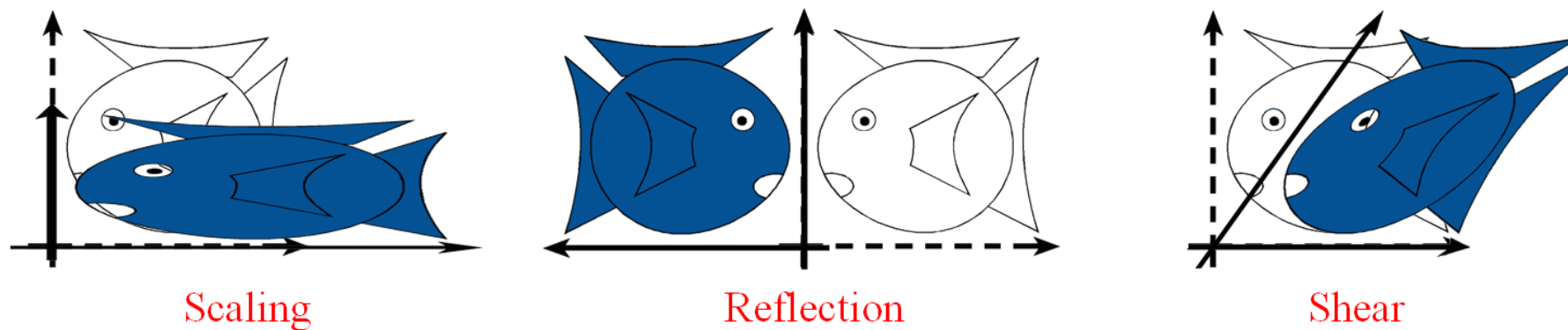
Similitudes/Similarity Transforms

- Preserves angles



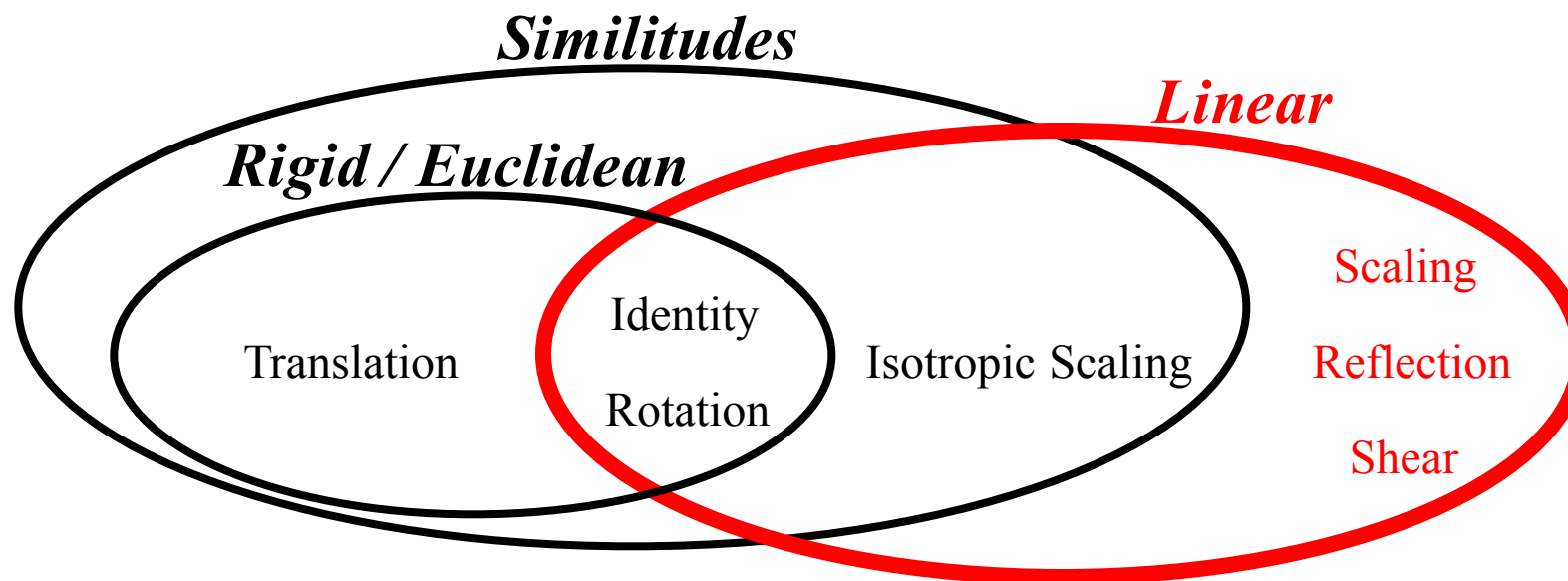
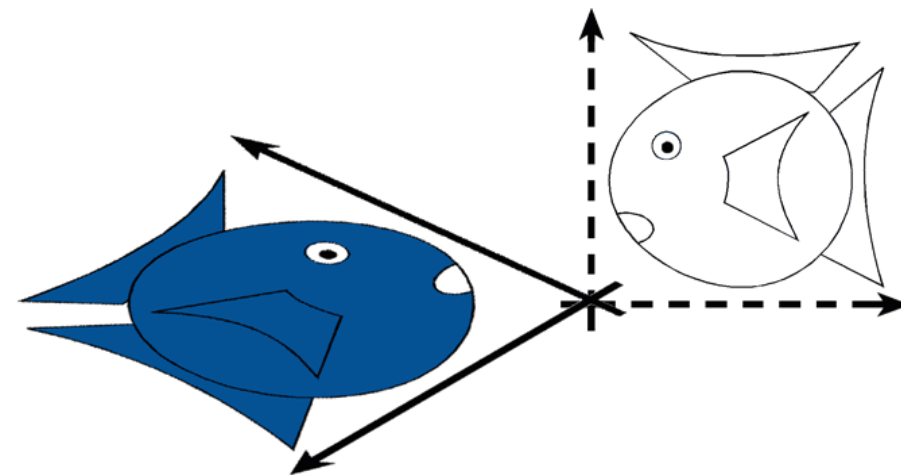
Linear Transformations

- Preserves parallelism



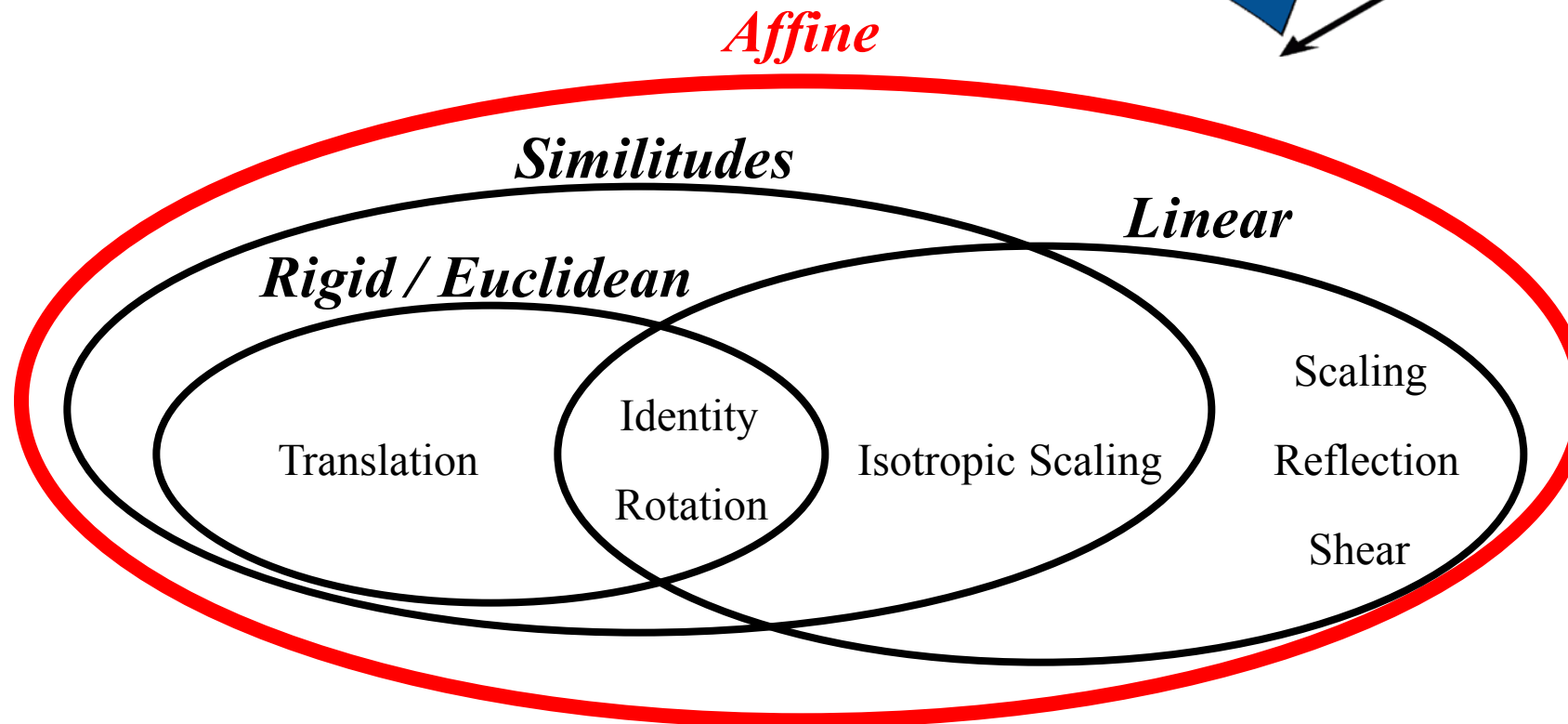
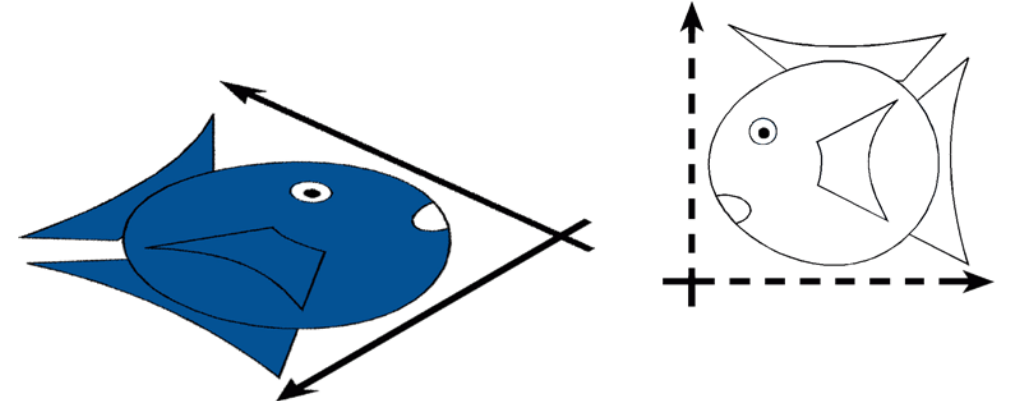
Linear Transformations

- $L(p + q) = L(p) + L(q)$
- $L(ap) = a L(p)$



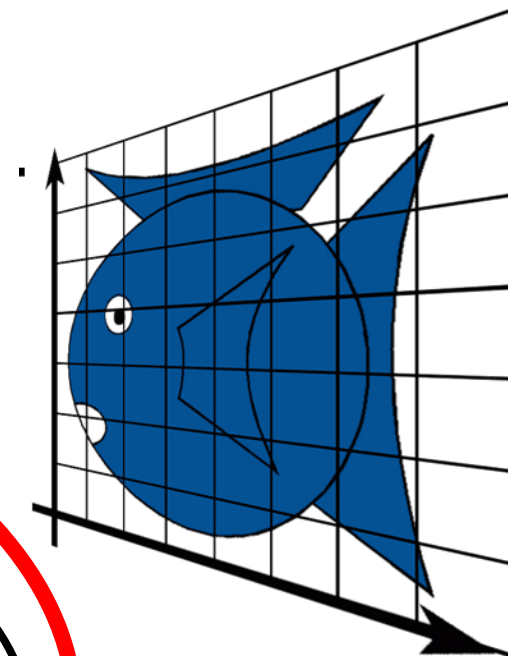
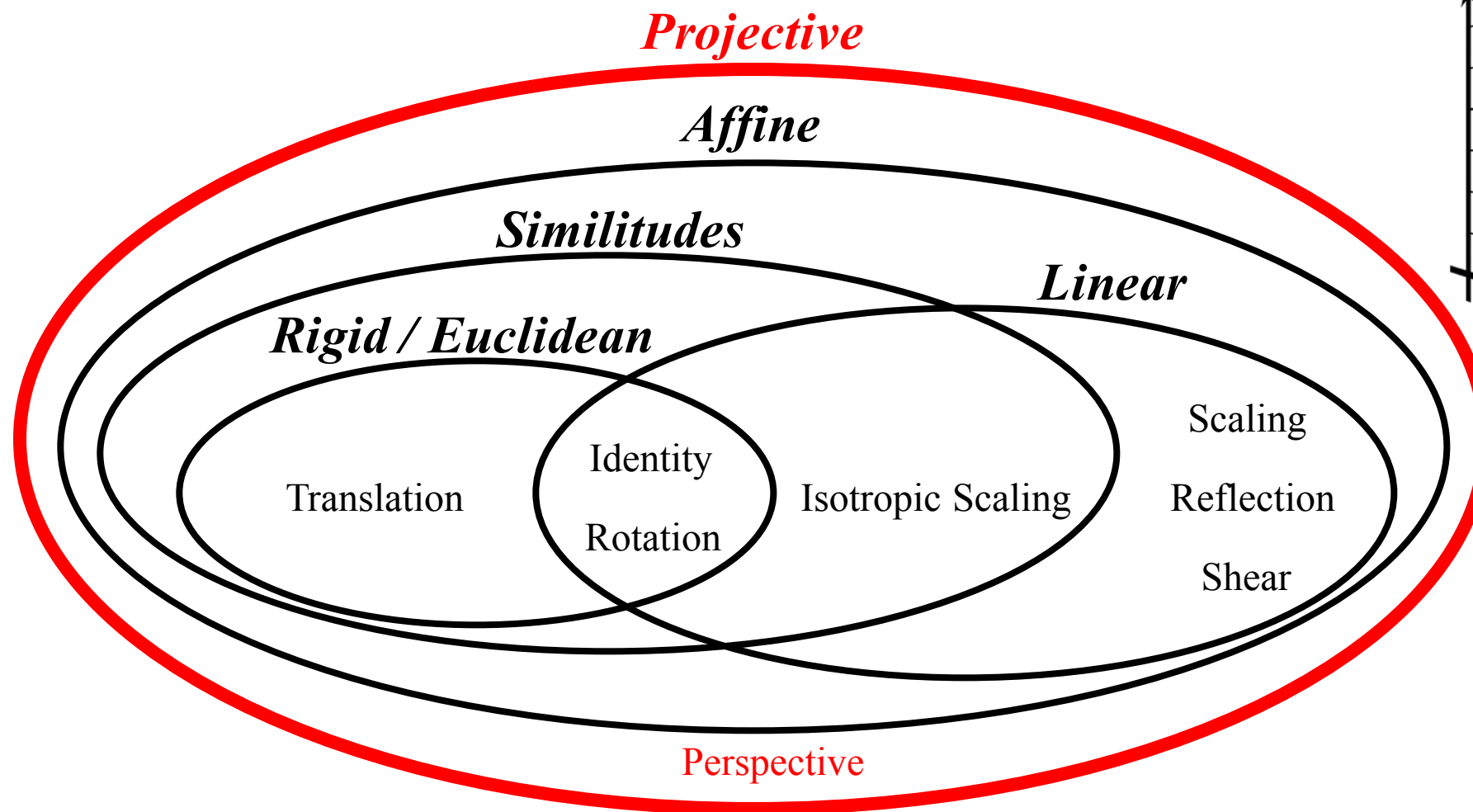
Affine transformations

- Preserves parallelism



Projective Transformations

- Preserves colinearity



General Free-Form Transformation

- Does not preserve lines
- Not as pervasive, computationally more involved
- Won't be treated until much later in this course

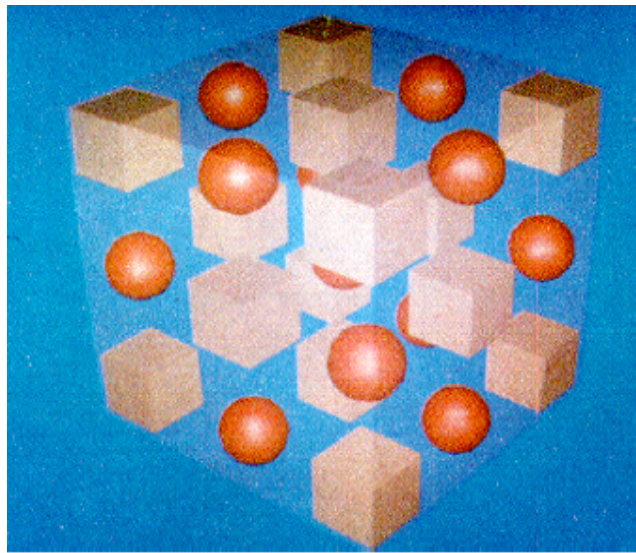


Fig 1. Undeformed Plastic

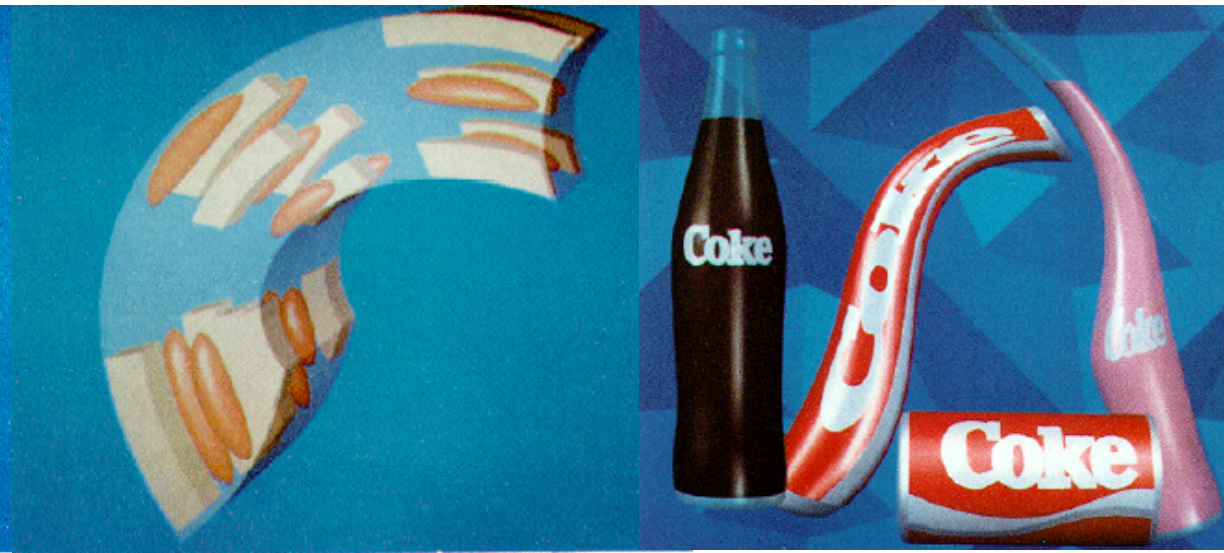


Fig 2. Deformed Plastic

Representing transformations

How are Transformations represented?



MPL
Mobile Perception Lab



$$x' = ax + by + c$$

$$y' = dx + ey + f$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$p' = Mp + t$$

Homogeneous coordinates

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

Affine formulation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c \\ f \end{bmatrix}$$

$$p' = Mp + t$$

Homogeneous formulation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$p' = Mp$$

Homogeneous coordinates in 3D

$$x' = ax + by + cz + d$$

$$y' = ex + fy + gz + h$$

$$z' = ix + jy + kz + l$$

Affine formulation

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a & b & c \\ e & f & g \\ i & j & k \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} d \\ h \\ l \end{bmatrix}$$

$$p' = Mp + t$$

Homogeneous formulation

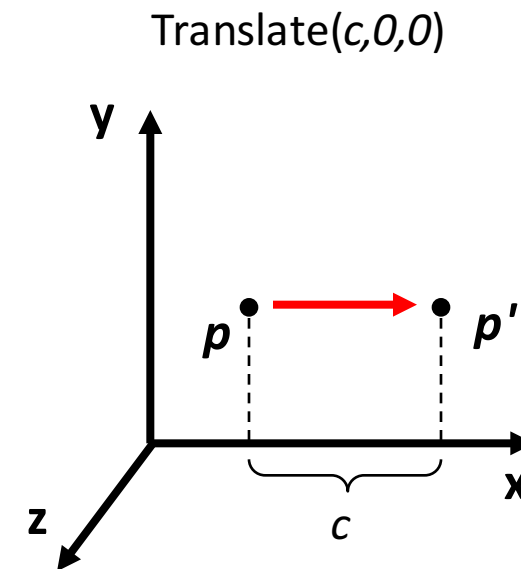
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$p' = Mp$$

Translation

- Why bother with the extra dimension?
 - Because now translations can be encoded in the matrix!

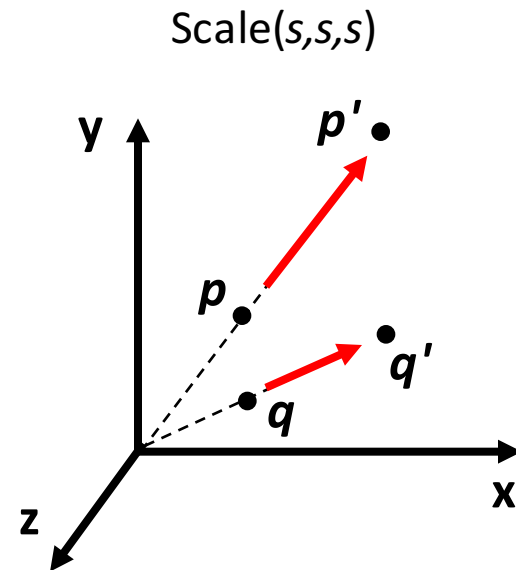
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Scaling

- Isotropic (uniform) scaling: $s_x = s_y = s_z$

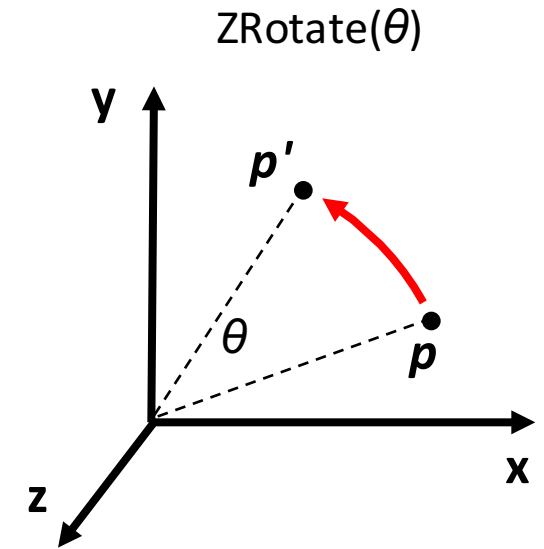
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Rotation

- About z axis

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



Homogeneous coordinates

- Add an extra dimension
 - in 2D, we use 3 x 3 matrices
 - In 3D, we use 4 x 4 matrices
- Each point has an extra value, w

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

$$p' = M p$$

Homogeneous visualization

- Divide by w to normalize (homogenize)
- $w = 0$? *Point at infinity (direction)*

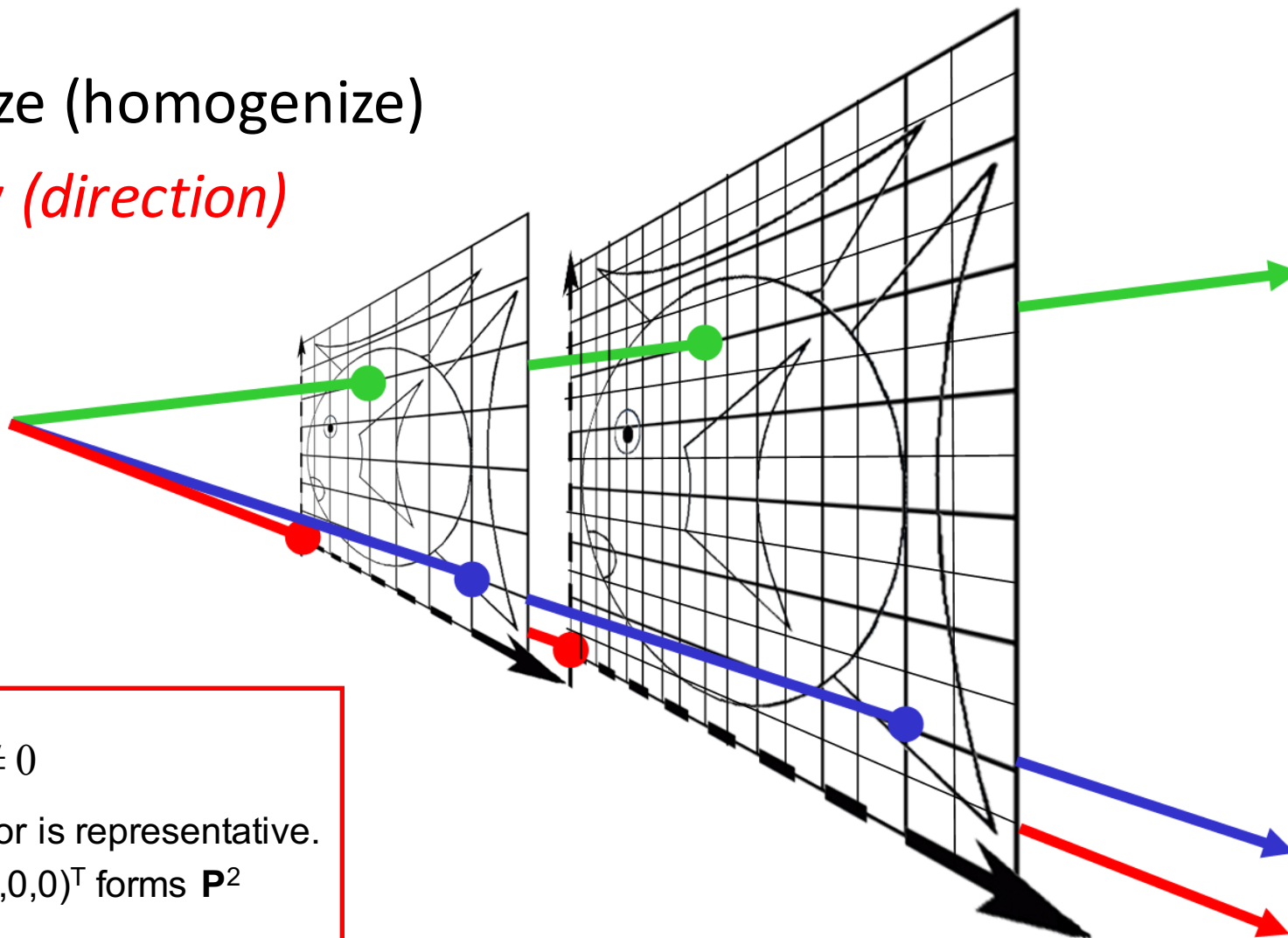
$$(0, 0, 1) = (0, 0, 2) = \dots$$

$$(7, 1, 1) = (14, 2, 2) = \dots$$

$$(4, 5, 1) = (8, 10, 2) = \dots$$

$$(x, y, 1)^T \sim k(x, y, 1)^T, \forall k \neq 0$$

Equivalence class of vectors, any vector is representative.
Set of all equivalence classes in $\mathbf{R}^3 - (0,0,0)^T$ forms \mathbf{P}^2



Homogeneous coordinates

- Most of the time $w = 1$, and we can ignore it

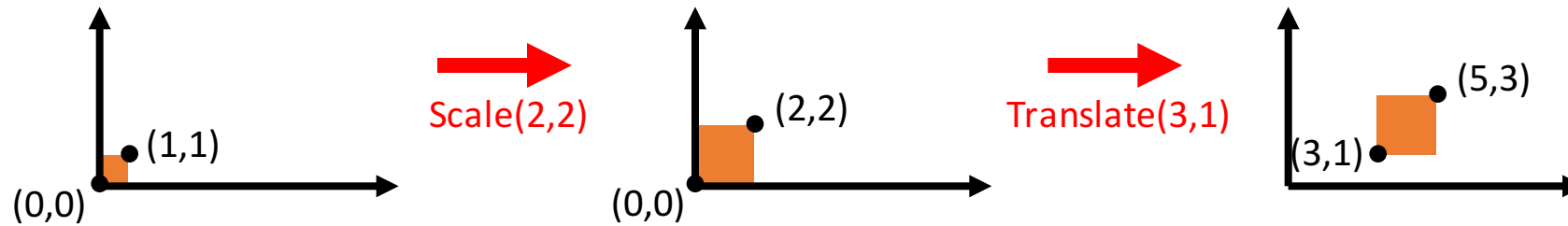
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- If we multiply a homogeneous coordinate by an *affine matrix*, w is unchanged

Combining transformations

How are Transformations combined?

Scale then Translate



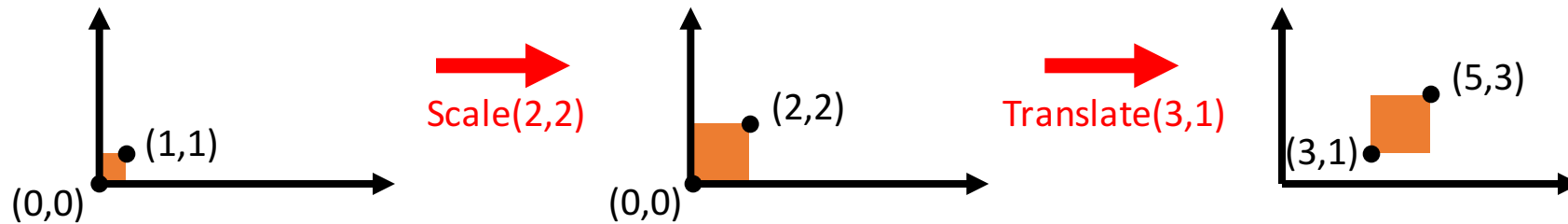
Use matrix multiplication: $p' = T (S p) = TS p$

$$TS = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

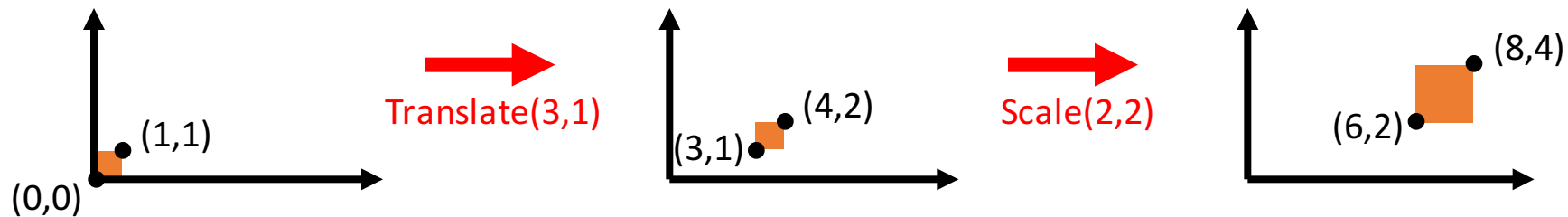
Caution: matrix multiplication is NOT commutative!

Non-commutative composition!

Scale then Translate: $p' = T (S p) = TS p$



Translate then Scale: $p' = S (T p) = ST p$



Non-commutative composition!

Scale then Translate: $p' = T (S p) = TS p$

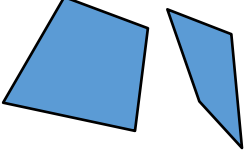
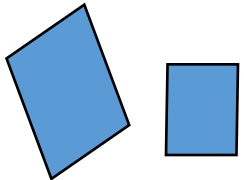
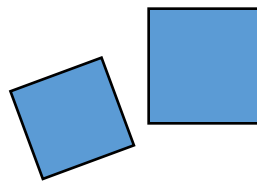
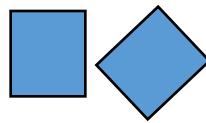
$$TS = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Translate then Scale: $p' = S (T p) = ST p$

$$ST = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

Important example 1: 2D projective transformation

Hierarchy of 2D Transformations

Projective 8dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$	<p>transformed squares</p> 	<p>invariants</p> <p>Concurrency, collinearity, order of contact (intersection, tangency, inflection, etc.), cross ratio</p>
Affine 6dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		<p>Parallelism, ratio of areas, ratio of lengths on parallel lines (e.g midpoints), linear combinations of vectors (centroids).</p> <p>The line at infinity l_∞</p>
Similarity 4dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		<p>Ratios of lengths, angles.</p> <p>The circular points I, J</p>
Euclidean 3dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$		<p>lengths, areas.</p>

projectivity=collineation=projective transformation=homography

Definition:

A *projectivity* is an invertible mapping h from P^2 to itself such that three points x_1, x_2, x_3 lie on the same line if and only if $h(x_1), h(x_2), h(x_3)$ do.

Theorem:

A mapping $h: P^2 \rightarrow P^2$ is a projectivity if and only if there exist a non-singular 3×3 matrix \mathbf{H} such that for any point in P^2 represented by a vector x it is true that $h(x) = \mathbf{H}x$

Definition: Projective transformation

$$\begin{matrix} x'_1 \\ x'_2 \\ x'_3 \end{matrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \quad \text{or} \quad x' = \mathbf{H}x$$

8DOF!

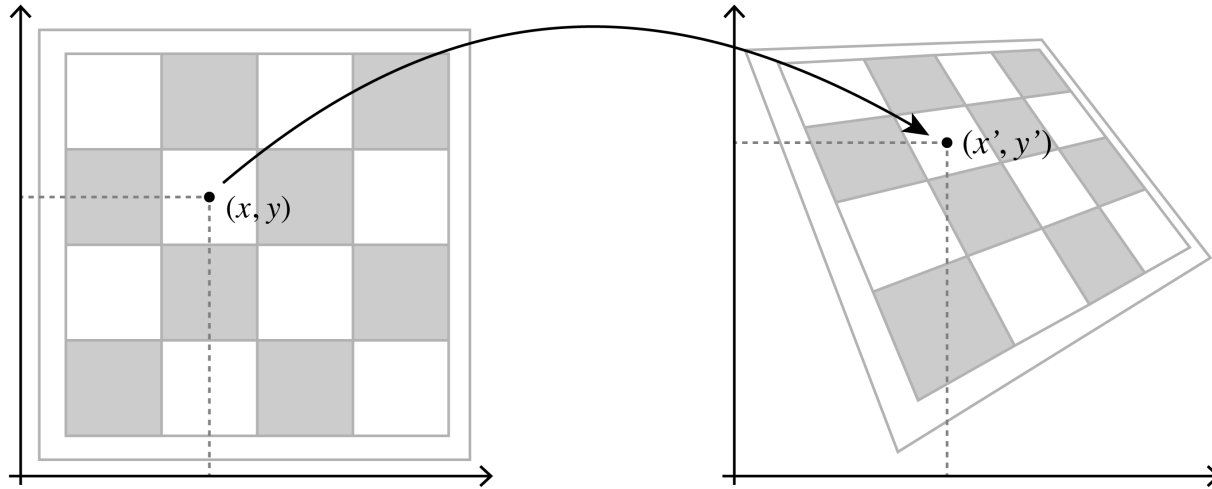
Where does it occur?

- Pure rotation of a camera
 - Panorama stitching!



Where does it occur?

- Full change of camera pose but planar scene
 - Aerial cartography!

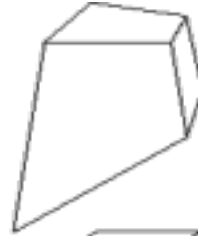


Important example 2: 3D rigid body transformation

Hierarchy of 3D Transformations

Projective
15dof

$$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$$



Intersection and tangency

Affine
12dof

$$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$$



Parallellism of planes,
Volume ratios, centroids,
The plane at infinity π_∞

Similarity
7dof

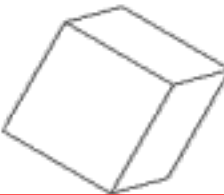
$$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$$



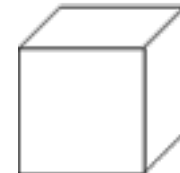
Angles, ratios of length
The absolute conic Ω_∞

Euclidean
6dof

$$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$$



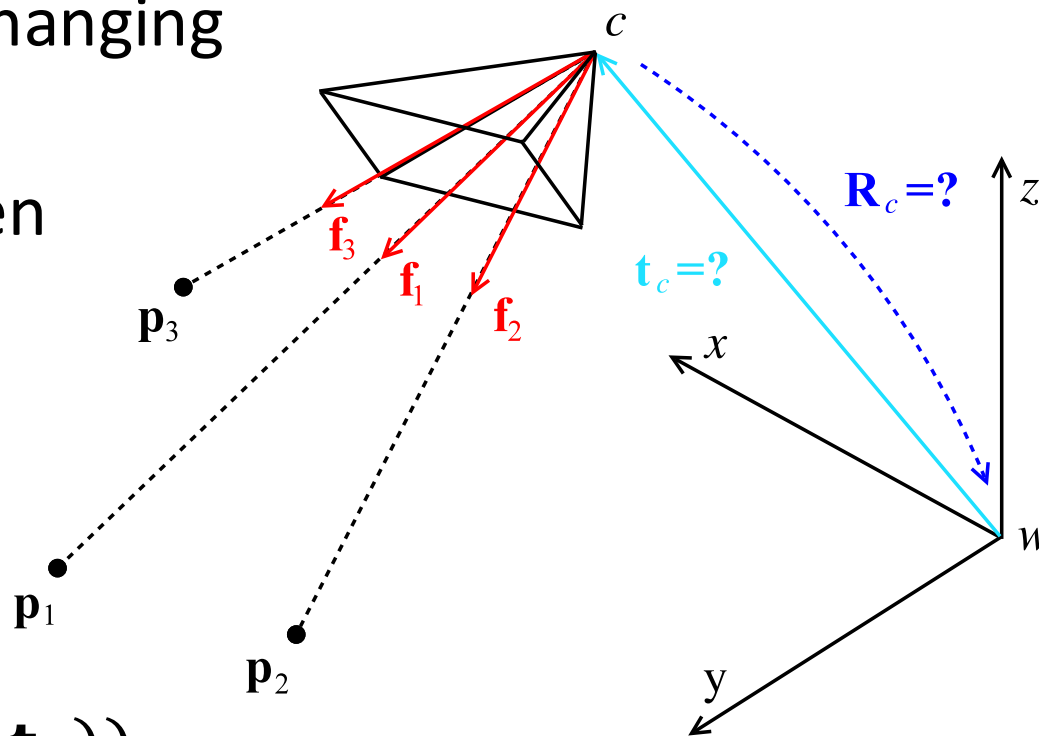
Volume



Back to our example

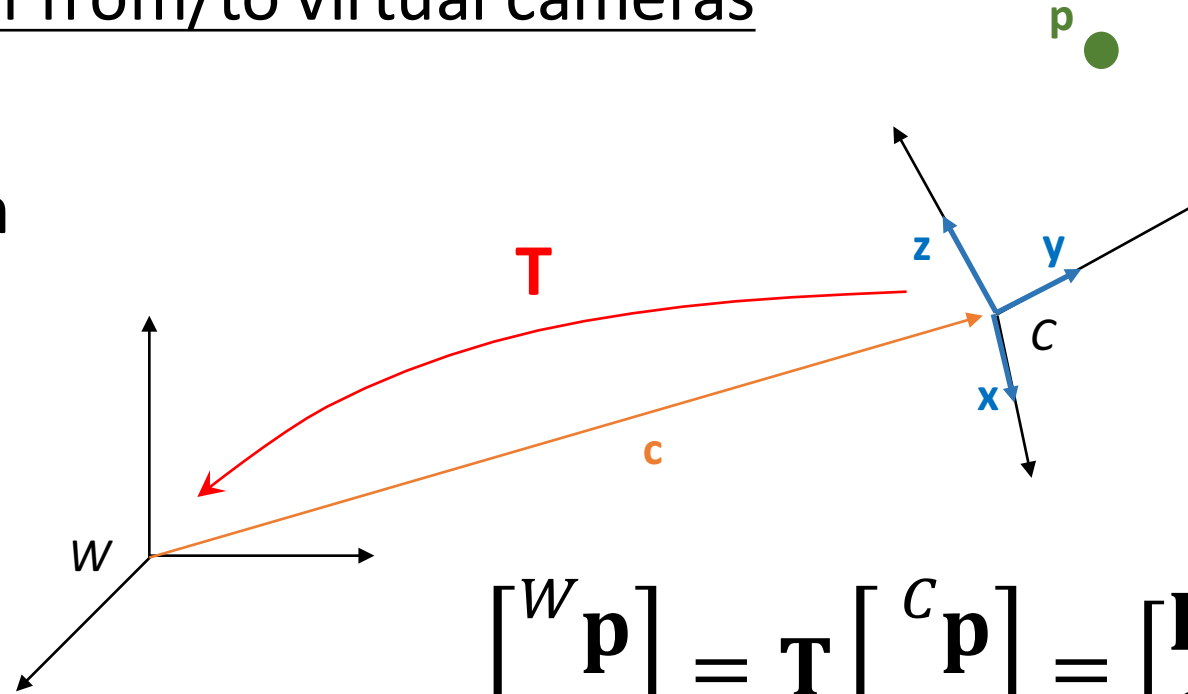
- 3D Euclidean transformations for changing reference frames
- → Describe transformation between world and camera frame
- Example: Absolute Pose
 - Objective:

$$\operatorname{argmax}_{\mathbf{t}_c, \mathbf{R}_c} \sum_i \mathbf{f}_i^t \cdot (\mathbf{R}_c^T \cdot (\mathbf{p}_i - \mathbf{t}_c))$$



3D rigid body transformation

- Describe transformation from/to virtual cameras
- Describe **camera pose**
- Used in pose estimation
- Used in global mapping



$$\begin{bmatrix} {}^w\mathbf{p} \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} {}^c\mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^c\mathbf{p} \\ 1 \end{bmatrix}$$

$$\mathbf{t} = {}^w\mathbf{c}$$

$$\mathbf{R} = \begin{bmatrix} {}^w\mathbf{x} & {}^w\mathbf{y} & {}^w\mathbf{z} \end{bmatrix} {}^w\mathbf{c}$$

Inverse Transformation

- Easy to calculate

$$\begin{bmatrix} {}^W\mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^C\mathbf{p} \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} {}^C\mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{Q} & \mathbf{v} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^W\mathbf{p} \\ 1 \end{bmatrix}$$

with

$$\begin{aligned} \mathbf{Q} &= \mathbf{R}^T \\ \mathbf{v} &= -\mathbf{R}^T \mathbf{t} \end{aligned}$$

Representations of Rotations



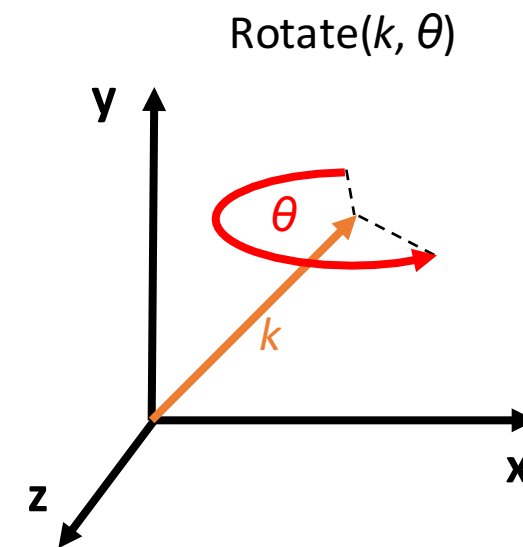
- Rotation matrix
- Axis-angle
- Euler
- Quaternion
- Cayley
- ...

Axis-angle

- About (k_x, k_y, k_z) , a unit vector on an arbitrary axis (Euler axis-angle, Rodrigues)

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} k_x k_x (1-c) + c & k_z k_x (1-c) - k_z s & k_x k_z (1-c) + k_y s & 0 \\ k_y k_x (1-c) + k_z s & k_z k_x (1-c) + c & k_y k_z (1-c) - k_x s & 0 \\ k_z k_x (1-c) - k_y s & k_z k_x (1-c) - k_x s & k_z k_z (1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where $c = \cos \theta$ & $s = \sin \theta$

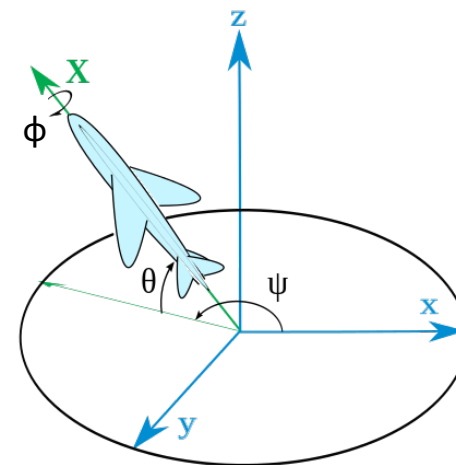


Euler rotations

- Basic idea:
 - A rotation matrix is decomposed into a sequence of 3 rotation matrices

$$\mathbf{R} = \mathbf{R}_1 \mathbf{R}_2 \mathbf{R}_3$$

- Each rotation is about a different axis
- There are 12 possibilities to choose the axes
- Example: Tait-Bryan convention: $\mathbf{R} = \mathbf{R}_x(\phi) \mathbf{R}_y(\theta) \mathbf{R}_z(\psi)$



Roll	Pitch	Yaw
$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}$	$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$	$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Quaternion

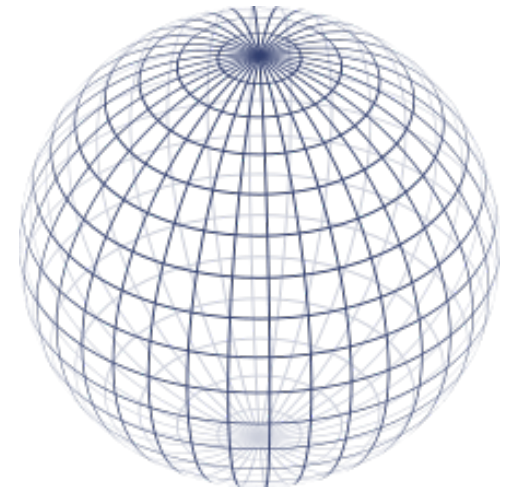
- $\mathbf{q} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix}$ such that $q_x^2 + q_y^2 + q_z^2 + q_w^2 = 1$

- Relationship to Euler-axis representations:

- Advantages:

- Compact representation
- Continuous variation over the unit sphere
→ eliminates discontinuous jumps present in other parameterizations such as Euler-angles or Euler-axis
- Quaternion to rotation matrix transformation is polynomial
- Simple to compose product of two rotations

$$\begin{cases} q_x = k_x \sin\left(\frac{\theta}{2}\right) \\ q_y = k_y \sin\left(\frac{\theta}{2}\right) \\ q_z = k_z \sin\left(\frac{\theta}{2}\right) \\ q_w = \cos\left(\frac{\theta}{2}\right) \end{cases}$$



Cayley parameters

- $\mathbf{c} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

- From Cayley parameters to the rotation matrix:

$$\mathbf{R} = \frac{1}{1 + x^2 + y^2 + z^2} \begin{bmatrix} 1 + x^2 - y^2 - z^2 & 2xy - 2z & 2y + 2xz \\ 2xy + 2z & 1 - x^2 + y^2 - z^2 & 2yz - 2x \\ 2xz - 2y & 2x + 2yz & 1 - x^2 - y^2 + z^2 \end{bmatrix}$$

- Advantage:
 - No trigonometric functions
 - A polynomial form of the original parameters

Notations



- Q: How many elements are required for a rotation?
- Q: How many elements are required for a translation?