Contents

Section II. MODELING

Chapter 3. Coordinates and Transformations

In almost all fields of science and engineering, it is essential to identify and manipulate mathematical representatic
exception. Intelligent robots build a "mental model" of themselves and the world as they perceive their environmer
the past and predicting the future. In an abstract sense, these models are simply a collection of numbers and labe
of a robot engineer to correctly associate numbers with meaning, and correspondingly, meanings with numbers. (I
some concrete examples will be offered soon enough!)

Two of the most important mathematical representations are vectors and matrices from linear algebra. Vectors are
two or three dimensions of space, but can also represent other quantities like sensor measurements. Matrices are
either through an action, or even through a change in how those numbers are interpreted. We will be using them l
almost every subject of robotics. Hence, they must be mastered to get anywhere beyond a superficial understand

This chapter discusses how vectors and matrices are used in robotics to represent 2D and 3D positions, direction
It is assumed that all students will have taken a course in linear algebra and can refresh themselves on basic defi
matrix notation, conventions, and basic identities that will be used throughout this book.

1 Vectors and coordinates

Vectors extend concepts that are familiar to us from working with real numbers $\mathbb{R}$ to other spaces of interest. They
numbers that have a common meaning like position or direction, or readings from a signal taken at a given time. T
which helps us wrap our heads around more difficult concepts.

Most often, n-dimensional Euclidean spaces $\mathbb{R}^n$ is used, in which a vector is simply a tuple of n real numbers. Th
way that vectors are conceived of by engineers and computer scientists, and that is certainly how they are stored
definition" of vectors. However, it is often important to realize that these numbers are just an interpretation of a mo
physical meaning -- and the numbers will change depending on their manner of interpretation, such as a chosen f
operations in 2D and 3D, and follow it with a discussion about the importance of separating meaning from represe

1.1 2D coordinate frames

In the "layman's definition", an n-dimensional *vector* $\mathbf{x}$ is a tuple of real numbers $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. For n
only temporarily to help distinguish between vectors and real numbers. In the future, the boldface will typically be

A 2D position P is represented by a 2-element vector $\mathbf{p} = (p_x, p_y)$ that gives its coordinates relative to axis dire
axes cross, called the origin (Fig. 1). We will also represent vectors in column vector form:

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$$

for use in matrix-vector products. Both parenthetical and column vector notations are equivalent and interchangea

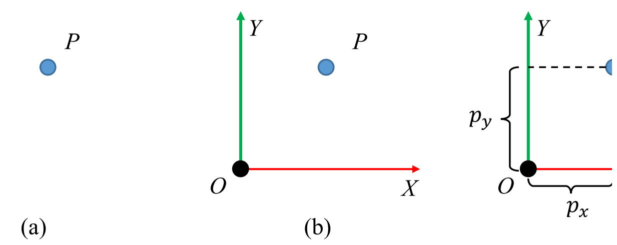


Figure 1. A point P in the plane (a) has no numerical representation until we define a reference coordinate fr
coordinate axes X and Y. Its coordinates $\mathbf{p} = (p_x, p_y)$ are respectively the extents of P al

The items O, X, and Y define the *coordinate frame* in which the coordinates are interpreted. Here O is an arbitra
directions with Y rotated 90° counter-clockwise from X. Note that in isolation, a vector of coordinates does not d
coordinates *in reference to a certain coordinate frame*. The frame will often be left implicit, or spoken of as the refe

Contents

1.2 3D coordinate frames

The situation in 3D space is similar, except that we represent a 3D position P with a 3-element vector $\mathbf{p} = (p_x, p$
Y, and Z and offset from an origin O in 3D space where the axes cross. The parenthetical notation is equivalent

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}.$$

In 3D the coordinate frame consists of the origin O and the mutually orthogonal axes X, Y, and Z. In this book w
which the axes can be envisioned in the layout of the first three fingers of the right hand, suitably arranged at 90°
axis corresponds to the index finger, and Z axis corresponds to the middle finger.

1.3 Directional quantities

Vectors are also used to represent *directional quantities*, such as a displacement, direction, or derivative. A *displa*
gives the amount that would need to be moved in both the X and Y direction to move from P to Q, where $\mathbf{q}$ give
frame. It has both a direction and a magnitude. In contrast, a *direction* does not have magnitude, and is a unit vec

$$\frac{\mathbf{q}-\mathbf{p}}{\|\mathbf{q}-\mathbf{p}\|}.$$

In 2D, a direction can also be given as an angle $\theta \in [0, 2\pi)$ rad, with the convention that the angle measures the
corresponding unit vector is $(\cos(\theta), \sin(\theta))$.

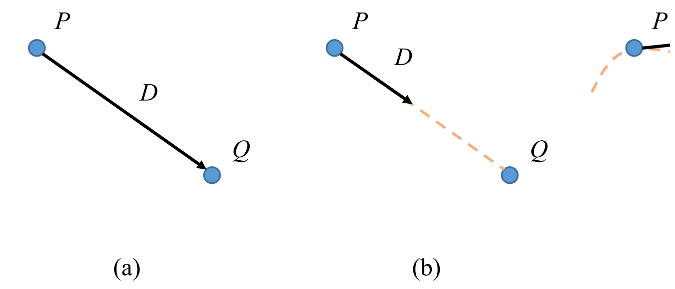


Figure 2. Directional quantities arise from displacements (a), directions (b), and de

A *derivative* is an infinitesimal displacement. If the position $P(t)$ is a function of t, then its derivative $\mathbf{p}'(t)$ is a ve

The major difference between directional and position quantities is that *coordinates of directional quantities do no*
coordinates of both positions and directions are affected by the choice of coordinate axes.

Contents

1.4 Geometric operations

The coordinates of a point $\mathbf{p}$ after translation by a displacement $\mathbf{d}$ can be computed by vector addition $\mathbf{p}+\mathbf{d}$. In specified by the equation

$$\mathbf{x}(u)=(1-u)\mathbf{p}+u\mathbf{q}$$

for $u \in \mathbb{R}$. This equation starts at $\mathbf{x}(0)=\mathbf{p}$ at $u=0$, and ends at $\mathbf{x}(1)=\mathbf{q}$ at $u=1$. Extrapolation can be ob

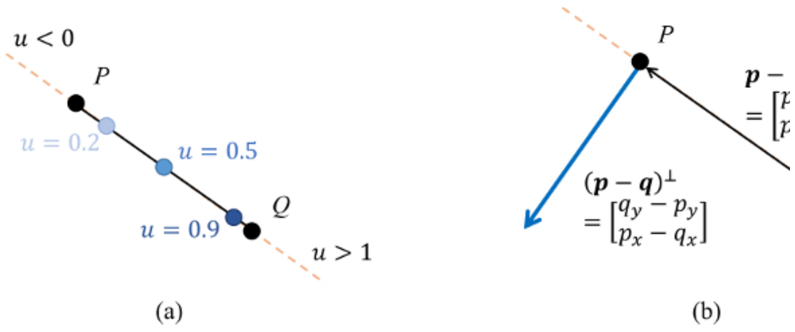

$u<0$
P
$u=0.2$
$u=0.5$
Q
$u=0.9$
$u>1$
(a)
P
$\mathbf{p}-$
$(\mathbf{p}-\mathbf{q})^\perp$
$=\begin{bmatrix} q_y-p_y \\ p_x-q_x \end{bmatrix}$
(b)

Figure 4. The line passing through points P and Q can be modeled as a parametric interpola

The line through $\mathbf{p}$ and $\mathbf{q}$ can be obtained by sweeping the above interpolation / extrapolation formula across the and $\mathbf{q}$ is obtained by sweeping u across the range $[0,1]$.

An other useful definition of a line in 2D uses a point on the line and an orthogonal direction. We define $\mathbf{p}^\perp=(-$ which has the same magnitude but is rotated 90° clockwise. The line through the origin passing through $\mathbf{p}$ can be equation $\mathbf{x}\cdot\mathbf{p}^\perp=0$. Similarly, the line through points P and Q can be expressed as the equation

$$\mathbf{x}\cdot(\mathbf{p}-\mathbf{q})^\perp=\mathbf{p}\cdot(\mathbf{p}-\mathbf{q})^\perp$$

Another expression of lines is the following:

$$\mathbf{x}\cdot\mathbf{n}=c$$

Where $\mathbf{n}$ is orthogonal to the direction of the line and $c=\mathbf{p}\cdot\mathbf{n}$ for any point $\mathbf{p}$ on the line. (Fig. 4.b.)

This definition is known as the *plane equation*, which generalizes lines in 2D to planes in 3D and hyperplanes in h dimensions in an n-dimensional space, which we call a generalized plane. A unique representation for a generaliz orthogonal to the plane known as the *normal direction* and b is a nonnegative *offset* that determines the distance

2 Transformations

Transformations are functions that map n-D vectors to other n-D vectors: $T:\mathbb{R}^n\rightarrow\mathbb{R}^n$. They can represent ge or action, as well as changes of coordinates, which are caused by changes of interpretation. Many common spatia and scaling are represented by matrix / vector operations. Changes of coordinate frames are also matrix / vector stored and operated on ubiquitously in robotics.

2.1 Linear transformations

A *linear transformation* is one that can be represented as a matrix operation

$$\mathbf{y}=A\mathbf{x}$$

where $\mathbf{x}\in\mathbb{R}^m$, $\mathbf{y}\in\mathbb{R}^n$, and A is an $m\times n$ matrix. In particular, in 2D, this operation takes the form

$$\begin{bmatrix} q_x \\ q_y \end{bmatrix}=\begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} p_x \\ p_y \end{bmatrix}=\begin{bmatrix} ap_x+bp_y \\ cp_x+dp_y \end{bmatrix}.$$

In 3D, a linear transformation takes the form

$$\begin{bmatrix} q_x \\ q_y \\ q_z \end{bmatrix}=\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\cdot\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}=\begin{bmatrix} a_{11}p_x+a_{12}p_y+a \\ a_{21}p_x+a_{22}p_y+a \\ a_{31}p_x+a_{32}p_y+a \end{bmatrix}$$

2.2 Rotations in 2D

Rotations about the origin by angle θ can be defined as linear transformations. Consider two reference frames wit
Y, and the post-rotation axes X' and Y'. Depicting X' on top of X and Y as a line emanating from the origin, a
has coordinates $\mathbf{x}' = (\cos\theta, \sin\theta)$. It is a bit more involved, but not much, to determine that Y' has coordinates

Now consider that along with the coordinate frames, a point P was rotated to P'. We will derive how to determine
frame. Notice that P' still has coordinates (p_x, p_y) relative to the post-rotation frame X', Y', since distances do
Specifically, P' is obtained by walking p_x units from the origin in the direction of X', and then p_y units in the dire
coordinates in the original reference frame, we can use the fact that the coordinates of X' and Y' are known:

$$\mathbf{p}' = p_x\mathbf{x}' + p_y\mathbf{y}' = \begin{bmatrix} p_x\cos\theta - p_y\sin\theta \\ p_x\sin\theta + p_y\cos\theta \end{bmatrix}.$$

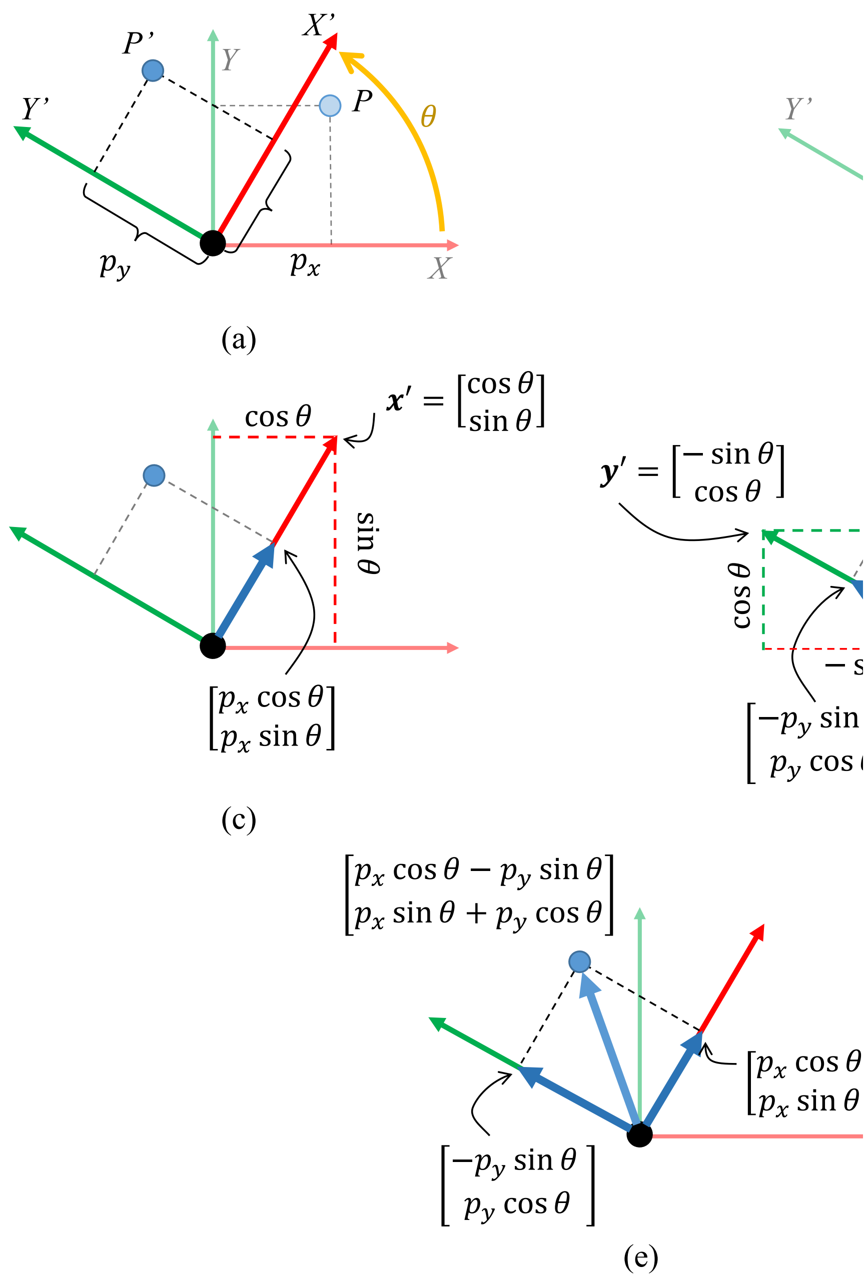

P'
Y
X'
θ
P
Y'
Y'
p_y
p_x
X
(a)
cos θ
$x' = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix}$
sin θ
$y' = \begin{bmatrix}-\sin\theta\\ \cos\theta\end{bmatrix}$
cos θ
$\begin{bmatrix}p_x\cos\theta\\ p_x\sin\theta\end{bmatrix}$
(c)
$\begin{bmatrix}p_x\cos\theta - p_y\sin\theta\\ p_x\sin\theta + p_y\cos\theta\end{bmatrix}$
$\begin{bmatrix}p_x\cos\theta\\ p_x\sin\theta\end{bmatrix}$
$\begin{bmatrix}-p_y\sin\theta\\ p_y\cos\theta\end{bmatrix}$
(e)

Figure 5. Rotating at an angle θ about the origin to achieve a new point P' (a). To calculate the coordinat
transformed axes X' and Y' (c,d) and then use the parallelogram

A more compact and convenient way of writing this is with a matrix equation

$$\mathbf{p}' = R(\theta)\mathbf{p}$$

with the rotation matrix given by

$$R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

There are several useful properties of such matrices:

1. The matrix composition $R(\theta_1)R(\theta_2) = R(\theta_1+\theta_2)$ gives the rotation matrix for the sum of the angles.
2. The determinant $\det(R(\theta)) = \cos^2\theta + \sin^2\theta = 1$ for all θ.
3. Due to the identities $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$, the operation of rotating about $-\theta$ is equ
$$R(-\theta) = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = R(\theta)^T.$$
4. Moreover, the transpose is equivalent to the matrix inverse:
$$R(-\theta) = R(\theta)^T = R(\theta)^{-1}.$$
In other words, rotation matrices are orthogonal.
5. Vector norms are invariant under rotation:
$$\|R(\theta)\mathbf{x}\| = \|\mathbf{x}\|.$$
6. The rotation matrix is only dependent on the argument's value modulo 2π.

The space of rotations is known as the *special orthogonal group* $SO(2)$. The reason why it is called the special o
2×2 matrices with positive determinant, while there do exist other orthogonal matrices with determinant -1.

Property 4 implies that it is more proper to consider rotation matrices as only representing instantaneous *orientati*
For example, if a motor has spun 720°, the matrix representation is indistinguishable from the 0 rotation. In certai

2.3 Rotations in 3D

In 3D, rotations can also be defined as linear transformations, although parameterizing them is not as simple as ir
rotation representations will discussed in further detail, but for now let us describe some of their properties.

A rotation in 3D can be represented by a matrix equation

$$\mathbf{p}' = R\mathbf{p}$$

with R a 3×3 rotation matrix.

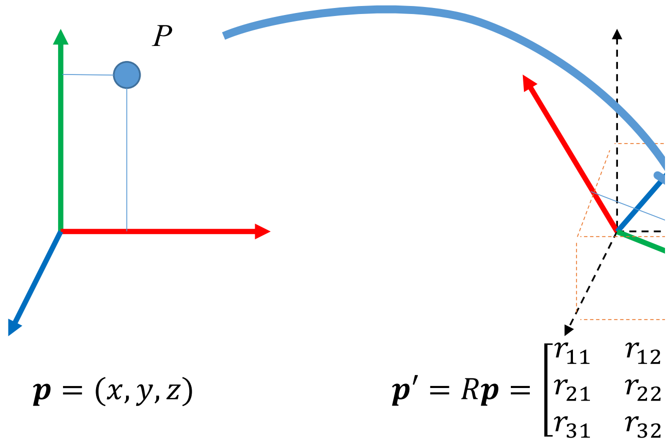

P
$\boldsymbol{p} = (x, y, z)$
$\boldsymbol{p}' = R\boldsymbol{p} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \\ r_{31} & r_{32} \end{bmatrix}$

Figure 6. After a rotation, the coordinates of the transformed point (relative to the original axes) are determinec
R.

Like in 2D, rotation matrices satisfy several properties:

1. The composition of two rotation matrices $R_1 R_2$ is also a rotation matrix.
2. In general, however, $R_1 R_2 \neq R_2 R_1$ unless the two are rotations about a shared axis.
3. The determinant $\det(R) = 1$ for all rotation matrices.
4. The transpose of a rotation matrix is its inverse: $R^T = R^{-1}$, or $RR^T = R^T R = I$.
5. Vector norms are invariant under rotation.

The space of 3D rotations is known as the *special orthogonal group* $SO(3)$.

Another interpretation of rotation matrices is to interpret each of the 3 columns as the coordinates of each of the c
coordinate frame rotates by matrix R about the origin, the entries of the rotation matrix can be interpreted as:

$$R = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix}$$

where (x_x, x_y, x_z) give the coordinates of the new X axis in the old frame, (y_x, y_y, y_z) gives the coordinates of
coordinates of the new Z axis.

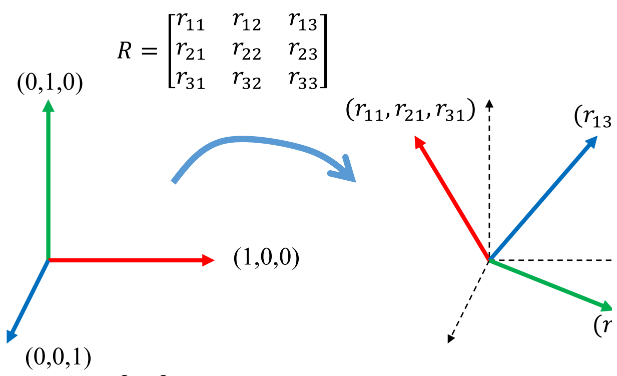

$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$
(0,1,0)
(1,0,0)
(0,0,1)
(r_{11}, r_{21}, r_{31})
$(r_{13}$
$(r$

Figure 7. A 3D rotation is encoded by a 3×3 matrix whose columns give the coordinates of the rot

This interpretation is useful when determining the coordinates of a rotated point in the original reference frame: if t
such that $P - O = p_x X + p_y Y + p_z Z$, then the new coordinates of P relative to the old reference frame are g

Contents

2.4 Scaling

Axis-aligned scaling in 2D about the origin can be represented as a linear transform with matrix

$$S(s_x, s_y) = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

where s_x is the scaling about the X direction and s_y is the scaling about the Y direction. If $s_x = s_y$ this is know

This definition can be generalized to n-D space using an n-D scaling vector $\mathbf{s}$ which determines the scaling in ea matrix:

$$S(\mathbf{s}) = diag(\mathbf{s}).$$

2.5 Compositions of linear transformations

When performing two linear transformations one after another, the results are determined via matrix multiplication transformations with matrices A and B, respectively. When performing the operation of T_2 first to obtain $\mathbf{y} = T_2$ ultimate result is:

$$\mathbf{z} = T_1(T_2(\mathbf{x}))$$

where it should be noted that T_1 appears first in the equation even though it is performed after T_2. Expanding this

$$\mathbf{z} = AB\mathbf{x}$$

holding for all values of $\mathbf{x}$. As a result, the function composition $T_1 \circ T_2$ is also a linear transformation with matrix

Using composition we can derive other useful transformations, like scaling not aligned to an axis. Suppose we wis $\mathbf{v}$, where $\mathbf{v}$ is a unit vector. We can think of this as first rotating by an angle θ so that the X axis is aligned with $\mathbf{v}$, rotating back to the original coordinate frame:

$$A = R(-\theta)S(s,1)R(\theta).$$

It so happens that

$$R(\theta) = \begin{bmatrix} v_x & v_y \\ -v_y & v_x \end{bmatrix}$$

performs the rotation that we desire. As a result we can compute:

$$\begin{aligned} A &= \begin{bmatrix} v_x & -v_y \\ v_y & v_x \end{bmatrix} \begin{bmatrix} s & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_x & v_y \\ -v_y & v_x \end{bmatrix} = \begin{bmatrix} v_x & -v_y \\ v_y & v_x \end{bmatrix} \begin{bmatrix} sv_x \\ -v_y \end{bmatrix} \\ &= \begin{bmatrix} sv_x^2 + v_y^2 & sv_x v_y - v_y v_x \\ sv_x v_y - v_x v_y & sv_y^2 + v_x^2 \end{bmatrix} = I + (s-1) \begin{bmatrix} v_x^2 \\ v_x v_y \end{bmatrix} \end{aligned}$$

Note that due to the non-symmetricity of matrix multiplication, order of transformation matters: a rotation followed followed by a rotation. However, with a little inspection, we can derive the following symmetric compositions:

- As a consequence of $R(\theta_1)R(\theta_2) = R(\theta_1 + \theta_2)$, rotation by angle θ_1 followed by angle θ_2 is symmetric: does not generally hold in 3D!)
- A rotation and a uniform scaling.
- Two axis-aligned scalings.

2.6 Rigid transformations

Rigid transformations in two dimensions have two properties:

1. The distance between two points do not change after being transformed.
2. In 2D, the orientation and area of any triangle does not change, and in 3D, the orientation and volume of any

The form of all rigid transforms is a rotation R followed by an arbitrary translation $\mathbf{t}$:

$$T(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$$

Which can be thought of applying a *rotation about the origin first*, and then a *translation second*. Proving that all ri exercise.

It is also possible to interpret rigid transforms as rotation about an arbitrary point. Letting the *center of rotation* be translating a point so that $\mathbf{c}$ is the origin, then rotating about the origin by some matrix R, and then translating bac

$$T(\mathbf{x}) = R(\mathbf{x} - \mathbf{c}) + \mathbf{c}.$$

The parenthetical term is the translation to $\mathbf{c}$ as the origin, the multiplication by R is the rotation about the new ori the original origin. These two representations are related by $\mathbf{t} = \mathbf{c} - R\mathbf{c}$, and $\mathbf{c} = (I - R)^{-1}\mathbf{t}$.

The set of rigid transformations is called the *special Euclidean group* $SE(2)$ in 2D, and $SE(3)$ in 3D. Repeated a rigid transformation. Given two rigid transforms

$$T_1(\mathbf{x}) = R_1\mathbf{x} + \mathbf{t_1}$$

and

$$T_2(\mathbf{x}) = R_2\mathbf{x} + \mathbf{t_2}$$

then the composite transform $T_1 \circ T_2$ is the operation of performing T_2 first, then T_1. If we let $\mathbf{y} = T_2(\mathbf{x})$, and $\mathbf{z}$

$$\mathbf{z} = T_1(T_2(\mathbf{x})) = R_1(R_2\mathbf{x} + \mathbf{t_2}) + \mathbf{t_1}.$$

By the distributive property of matrix multiplication, we have

$$\mathbf{z} = R_1 R_2 \mathbf{x} + R_1\mathbf{t_2} + \mathbf{t_1}.$$

This is simply a rigid transform with rotation matrix $R_1 R_2$ and translation by $(R_1\mathbf{t_2} + \mathbf{t_1})$.

2.7 Inverse transformations

Not all transformations have inverses, but rotations, translations, rigid transformations, and many linear transform rotation matrix is simply its transpose. Translations are inverted by translating in the negative direction. Linear tran invertible, with the inverse transformation $A^{-1}\mathbf{x}$.

Rigid transformations are also invertible, and their inverse is also a rigid transformation:

$$T^{-1}_{(R,\mathbf{t})} = T_{(R^T, -R^T\mathbf{t})}$$

where $T_{(R,\mathbf{t})}(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$. Proof of this equation will be left as an exercise.

2.8 Rigid movement

Rigid transformations are used to represent movement of rigid bodies in space. If, in 2D the origin of a body move rotates by angle $R = R(\theta)$, then the transformation that *converts positional coordinates from the new coordinate* $T_p(\mathbf{x}) = R\mathbf{x} + \mathbf{t}$. In other words, if $\mathbf{x}$ gives the coordinates of a position P that is attached to the body, then afte the original body's frame. However, the transformation of directional coordinates will simply be a rotation and igno gives the coordinates of a directional quantity V that is attached to the body (such as the direction of a line attach relative to the original coordinate frame (as in other directional quantities, these are interpreted ignoring the origin

2.9 Representation of coordinate frames and coordinate transforms

Coordinate frames, as well as conversions between them, are interpreted as rigid transformations.

Any 2D coordinate frame F with origin O and axes X and Y may be represented by the coordinates of O, X, ar
coordinates $\mathbf{t}$, and X and Y have (directional) coordinates $\mathbf{x} = (x_1, x_2)$ and $\mathbf{y} = (y_1, y_2)$ relative to W, then t
coordinates in the frame F can be calculated by the rigid transform

$$\mathbf{p}_W = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix} \mathbf{p} + \mathbf{t}.$$

Hence, by storing $R = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}$ and $\mathbf{t}$ we can perform this calculation for any point in F. (We remark that the
orthogonal and Y is 90° ccw from X.) The information stored for a 3D coordinate frame is similarly a rotation mat
known as the *coordinate transform* $A \to W$ with A the *source* frame and W the *target* frame . We can also perfo
by applying the inverse transform.

Changes of coordinate frames can also be represented in terms of rigid transforms. Suppose A and B are two co
respect to the world frame by a rotation matrix R_A and translation $\mathbf{t}_A$, and B is represented by R_B and $\mathbf{t}_B$. Ther
to A, we can determine P's coordinates relative to $\mathbf{p}^B$ in two steps. First, we calculate its world coordinates:

$$\mathbf{p}^W = T_A(\mathbf{p}^A) = R_A \mathbf{p}^A + \mathbf{t}_A$$

And then we perform the inverse of B coordinates to world coordinate to obtain its coordinates with respect to B:

$$\mathbf{p}^B = T_B^{-1}(\mathbf{p}^W) = R_B^T(\mathbf{p}^W - \mathbf{t}_B).$$

This transform can be calculated for all points by the composition of the transform from $A \to W$ and then $W \to$

$$\mathbf{p}^B = T_B^{-1}(T_A(\mathbf{p}^A)) = R_B^T R_A \mathbf{p}^A + R_B^T(\mathbf{t}_A - \mathbf{t}_B).$$

2.10 Homogeneous coordinate representations

Homogeneous coordinates gives a convenient representation of rigid transforms as linear transforms on an expar
distinction between positional and directional quantities. The idea is to augment every point with an additional *hon*
0 if it is directional. This operation is denoted with the hat operator $\hat{\cdot}$.

For 2D points and directions, we have

$$\hat{\mathbf{p}} = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix}$$
$$\hat{\mathbf{v}} = \begin{bmatrix} v_x \\ v_y \\ 0 \end{bmatrix}$$

Then, a 2D rigid transformation of both positions and directions is represented uniformly by a 3×3 matrix multipl

$$\hat{T}(\hat{\mathbf{x}}) = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{x}}$$

where the original transformation $T(\mathbf{x}) = R(\theta)\mathbf{x} + \mathbf{t}$ is a rotation about angle θ followed by a translation of vect

In 3D, the hat operator adds a 4th coordinate: For 2D points and directions, we have

$$\hat{\mathbf{p}} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$
$$\hat{\mathbf{v}} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$$

and a 3D transform is represented by a 4×4 matrix multiplication:

$$\hat{T}(\hat{\mathbf{x}}) = \begin{bmatrix} & & & t_x \\ & R & & t_y \\ & & & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \hat{\mathbf{x}}$$

Note that when applied to homogeneous positions, the rigid transform is applied to the first two coordinates of the
(since the dot product of a position representation with the last row of the matrix is 1). Also, when applied to homo
first two coordinates of the vector, since the third 0 coordinate nullifies the effect of the third column. The homoene
position representation with the last row of the matrix is 0.

The nice thing about this representation is that transform application is a matrix-vector multiply, transform compos
inversion is a matrix inversion. This makes it much easier to write out complex transformations. For example, cons
frame A to frame B that we described above. Rather than writing out the operator expression $T_B^{-1}(T_A(\mathbf{p}^A))$, us
of matrix-matrix and matrix-vector multiplies:

$$\hat{p}^B = \hat{T}_B^{-1} \cdot \hat{T}_A \cdot \hat{p}^A$$

.

3 Working with coordinate representations of positions and directi

In small 2D examples we can often treat coordinates casually, with our minds quickly filling in the gaps of meaning
3D and beyond, this type of casual thinking runs the risk of making the robot walk through a wall! It is unlikely that
or Numpy in Python) will help you guard against performing invalid operations, since you will be working with vect
properly convert between reference coordinate systems.

This section will once again discuss the importance of separating the coordinate representation of an object from
common pitfalls when working with coordinates, and tools to help reduce the risk of making calculation errors.

3.1 An illustrative example

Let us start with a running example where a point P is drawn on a sheet of graph paper G. Some physical positic
axes X and Y which are attached to the graph paper. If we were to draw the straight line from O to P, suppose t
direction and p_y units in the Y direction. If we were to move the paper (say, a few inches to the left and rotated 25
since they were attached to the paper, the distances between them did not change. Nor did the 90° angle betwee
closest positions on the X or Y axes. Hence there are many physical quantities that have not changed, even thro

Let us also suppose that just before moving the paper, we have erased the arrow, and now we need to draw it aga
between O and P when the line was originally drawn, we can simply reproduce that arm movement (you know...
crossing of the axes, so we would simply need to find the new physical location of position P to end the arrow. Lu
given O, X, and Y. The procedure is simple: start at O, then walk p_x squares in the new X direction, and finally
say that (p_x, p_y) are the *coordinates* of the position P, relative to origin O and axes X and Y.

More formally, the coordinate frame defined by the physical position O and directions X and Y defines a one-to-c
paper and the vector of coordinates $(p_x, p_y) \in \mathbb{R}^2$. It would also be desirable to be able to say that $P = O + p_x$
understanding of how to "arrive at" P. However, we have only defined addition and scalar multiplication of vectors
and Y are not themselves vectors, they are elements of thought with a physical meaning).

Luckily, we can do this more formally if we define O, X, and Y with reference so some other *privileged coordinat*
let $\mathbf{p}^W$, $\mathbf{o}^W$, $\mathbf{x}^W$, and $\mathbf{y}^W$ be the coordinates of P, O, X, and Y with respect to W, then we indeed have the re
$\mathbf{p}^W = \mathbf{o}^W + p_x \mathbf{x}^W + p_y \mathbf{y}^W$. Moreover, $\mathbf{x}^W \cdot (\mathbf{p}^W - \mathbf{o}^W) = p^x$, and $\mathbf{y}^W \cdot (\mathbf{p}^W - \mathbf{o}^W) = p^y$. An important
choice of world frame. Hence we can conclude that it is reasonable to casually state that $P = O + p_x X + p_y Y$,
without regards to choosing a particular value of W.

3.2 Directional quantities

Next, let us illustrate how the coordinates of directional quantities transform. As an illustration, let us go back in tin
consider the line segment $\overline{PQ}$. Originally, $\overline{PQ}$ was drawn by moving a pen in a straight line from P to Q, with a
moved the pen d_x cells in the X direction and d_y cells in the Y direction, with $(d_x, d_y) = (q_x - p_x, q_y - p_y)$.

Once the graph paper has moved, and we wish to draw $\overline{PQ}$ again in the paper's new location, but suppose that r
remembered the values of (d_x, d_y). To reproduce the movement starting at P, the stroke in the new displacemer
direction and d_y units in the new Y direction.

Notice that *the origin has dropped* when performing this exercise, because the choice of origin when defining $(p_x$
difference. Using the same world coordinates trick as performed above, we can also say in casual shorthand that
$d_y = D \cdot Y$.

3.3 Associating physical meaning to operations on coordinates

The main distinction between positions and directions is that the choice of origin does not affect how directions ar
choice of coordinate axes. Hence, when performing changes of coordinates, the interpretation of position quantitie
directional quantities.

Positions and directional quantities also transform *into each other* when various operations are performed on then
displacement is produced. When a displacement is added to a position, another position is obtained. Displacemer
a scalar. However, it is meaningless to multiply a position by a scalar, or to add two position together, *even though*
representations. (In cases where these latter two operations would seem to make sense, it is more appropriate to
position.) Directions transform in much the same way that displacements do, but they do not scale or add.

In summary, the following mathematical operations on coordinates correspond to meaningful physical operations.
of the equations) may transform the type of object being represented. Here, P indicates a position, D indicates a d
scalar.

Valid operations	Meaning
P - P = D	Calculating the displacement between points
P + D = P	Displacing a point by a displacement
D + D = D	Calculating the displacement caused by two displacem
c D = D	Scaling a displacement
D/‖D‖ = V if D ≠ 0	Finding the direction of a displacement
c V = D	Scaling a direction to obtain a displacement

Here, it is assumed that all items share the same coordinate frame. Operations on items not sharing the same coo
following types of operations:

Invalid operations	Meaning
c P	Scaling a point
P + P	Adding two points
V + V	Adding two directions
P + V	Adding a direction to a point

Finally, dot products and norms are meaningful only when performed on directional quantities.

3.4 Working with coordinates

When working with different coordinate representations, it is easy to make mistakes in calculations. Below are sor

Mistaken coordinate conventions include:

1. Assuming coordinates are given in a different frame.
2. Incorrectly assuming a matrix represents its inverse.
3. Incorrectly assuming a matrix represents its transpose (row-major vs column-major).
4. Assuming a coordinate frame is represented with respect to an incorrect base frame.
5. Applying a coordinate transform to a quantity not represented with respect to the source frame.
6. Operating on the wrong units (e.g., a frame's translation is expressed in *m* while the point's coordinates are e

To guard against these mistakes, it is common practice to indicate frames of reference with letters and annotate v superscript/subscript convention used by this book is popular amongst many authors. In this convention, we indica in its superscript, e.g. $\mathbf{p}^A$ denotes a point expressed in frame A. A coordinate transform from frame A to frame way:

> F in *superscript*: Coordinates should be interpreted as "relative to F".
>
> F in *subscript*: Coordinates should be interpreted as "describing the transform of frame F".

In this convention, *superscripts and subscripts must cancel out for the equation to make sense*. Specifically, the fo

1. When applying a transform to a coordinate, the subscript of the transform's source frame must match the poi $\mathbf{p}^B = T_A^B \mathbf{p}^A$ is allowed but $T_B^A \mathbf{p}_B$ is illegal.
2. When composing two transforms, the source frame of the second frame must match the destination of the firs $T_A^B T_B^C$ is illegal.
3. When inverting a transform, the order of frames swaps: e.g. $(T_A^B)^{-1} = T_B^A$.
4. When adding or subtracting coordinates, their superscripts must match! For example, $\mathbf{a}^W - \mathbf{b}^W$ is the displa frame, but $\mathbf{a}^W - \mathbf{b}^C$ is an illegal operation.

By convention, a coordinate without a superscript is assumed to be associated with the world frame W: $\mathbf{p} \equiv \mathbf{p}^W$ to a transform from A to W: $T_A \equiv T_A^W$.

Matrix expressions are similar to standard expressions regarding real numbers in that addition and subtraction are inverses give an approximation of division. But, this similarity leads to common pitfalls when manipulating matrix e should be safeguarded against:

1. Performing transformations out of order, or swapping the arguments of a matrix product (products are not cor
2. Propagating transposes or inverses into a matrix product without swapping the order of arguments.
3. Assuming that a matrix is invertible (or worse, assuming a non-square matrix is invertible).
4. Performing operations on matrices of incompatible size.

Contents

3.5 Coordinate management

It is essential when working with coordinates to understand 1) which coordinate frame is being used, 2) what unit coordinates represent a position or directional quantity? Failure to do so is a frequent source of errors. When we b substantial bookkeeping and notation. Moreover, in a production robot, dozens or hundreds of coordinate convent

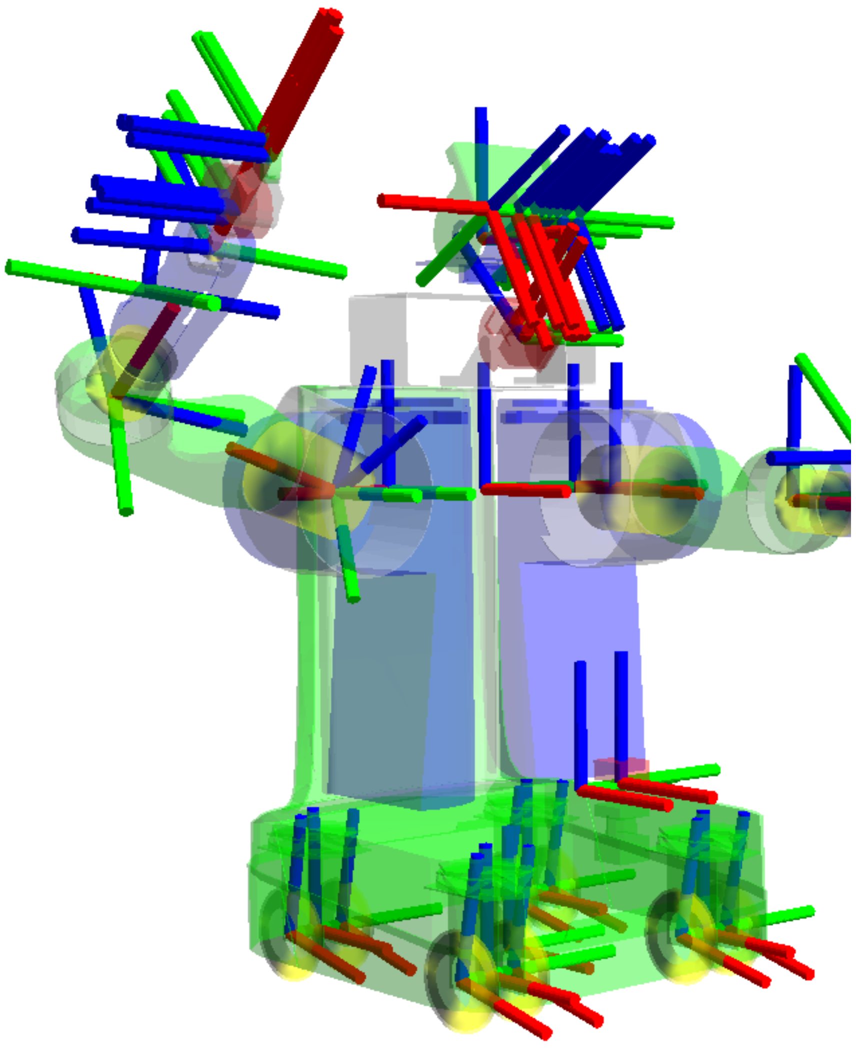
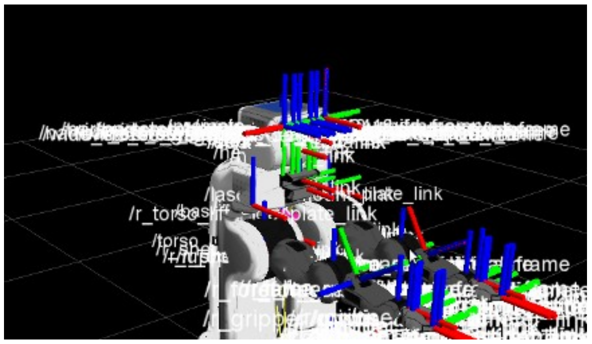


Figure 8. The Willow Garage PR2 robot has dozens of coordinate frames for its var

As a result, some middleware systems come with coordinate management systems. These systems attach an ann directional quantity consists of a vector $\mathbf{p}$ stored alongside its annotation A_p. The annotation may include whethe an identifier of its coordinate frame. The system will also store a list of coordinate frames, represented in coordina the "world frame"). Such systems will allow users to query the coordinates of points and directions in arbitrary fran matrices. With these systems, it becomes somewhat more convenient to maintain and manipulate a large number include the `tf` module in ROS (Fig. 9) and the `coordinates` module in Klamp't.



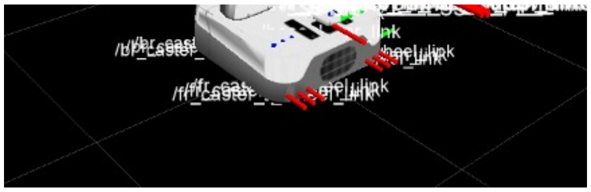


Figure 9. Coordinate management functionality is provided by the Robot Operating System (ROS) Tf module

The convenience of coordinate management systems comes at the cost of typically more verbose code and the le

4 Summary

Key takeaways:

- Coordinates are numerical representations of geometric concepts, like points, directions, frames of reference
- Points, directions, and displacements in n-dimensional space are represented by n-dimensional vectors, whi matrices.
- Rigid transformations consist of a rotation followed by a translation. They represent both rigid body movemen
- Homogeneous coordinates represent rigid transforms using matrix multiplication in an $n+1$ dimensional spa
- When working with coordinates it is easy to make mistakes. Having clear assumptions, clear notation and/or the risk of error.

Contents

5 Exercises

1. Calculate the matrices representing $+90°$, $+180°$, and $-90°$ 2D rotations.

- Prove that $R(\theta)^{-1} = R(-\theta)$ using linear algebra and geometric identities.
- Prove that $R(\theta_1)R(\theta_2) = R(\theta_1 + \theta_2)$ using geometric identities.
- Give examples of 3D rotation matrices for which $R_1R_2 \neq R_2R_1$.
- Prove that vector length is preserved under rotation $\|R(\theta)\mathbf{x}\| = \|\mathbf{x}\|$. Use this fact to prove that distance is
- Prove that all length-preserving transforms in 2D are either rotations or rotations followed by a mirroring trans

$$T(\mathbf{x}) = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x}.$$

- Produce the transform (i.e., rotation matrix R and translation $\mathbf{t}$) that rotates any point $\mathbf{x}$ by $45°$ about the poi

  What is the general form of a rotation of angle θ about a center point $\mathbf{c}$?
- A mobile robot has a coordinate convention that X is forward and Y is to the left. It has an sweeping laser se all angles around the robot using a spinning mirror. The laser sensor gives perfect depth readings, and is atta at the point $(0.7, 0.05)$ in the robot's X, Y frame and oriented so that the $0°$ reading is pointing in the robot' at 1 m distance at the $15°$ reading. At time $t + 1$, the robot moves 0.5 m forward, 0.1 m to the left, and 20° cc move, at which angle will the sensor detect the obstacle, and at what distance reading?
- Suppose a 3D camera produces points (x, y, z) in a coordinate frame that has $+x$ pointing to the right of the forward. Is the camera given with a right-handed or a left handed coordinate system?

  Suppose we wish to find the transformation for points detected by the camera so that that the camera origin i frame, its forward direction points in the world's X direction, up in the image points in the Z direction, and lef transformation for converting camera points to world points.
- Prove (30), i.e., that $T^{-1}_{(R,\mathbf{t})} = T_{(R^T, -R^T\mathbf{t})}$.
- The midpoint between positions P and Q is given by the equation $M = 0.5 * (P + Q)$. But this expressio that addition and scalar multiplication with positions are not meaningful operations. How can one make sense operations?
- Implement a 2D coordinate management system, in your programming language of choice, where each posit should define an API to create positions and frames, and to retrieve coordinates of any position in any given f The manager should maintain a list of named coordinate frames, each of which is represented in its coordina special frame "world"). Define a `Point` data structure that stores two elements
  - `coords` : the point's coordinates, expressed in terms of its reference frame.
  - `frame` : the name of its reference frame.

  Do the same for a `Directional` data structure. Now, implement methods `Point.to(frame)` and `Direct` `Directional` , respectively, whose reference frames are now the given frame, and whose coordinates are e
- Extend the coordinate management system of exercise 12 to implement geometric operations, enforcing that frame. Ensure that `Point - Point = Directional` , `Point + Directional = Directional` , `Directi` `Directional * scalar = Directional` .

  This tutorial (https://www.tutorialsteacher.com/python/magic-methods-in-python) for operator overriding in Pyt
- Extend the coordinate management system of exercise 12 to also include **units** that should be specified to th upon construction. Implement support for `'m'` , `'mm'` , `'cm'` and `'km'` arguments indicating meters, milli

Contents

In [1]:
```
#Skeleton code for exercise 12

class Frame:
    #TODO: what here?
    pass

#a dictionary from strings to Frame objects, to be used in Point.to and Directi
named_frames = dict()

class Point:
    def __init__(self,coords,frame):
        self.coords = coords
        self.frame = frame
    def to(self,newframe):
        """Returns a Point expressing the same physical point in space, but rep
        #TODO: what here?
        return Point(self.coords,newframe)

class Directional:
    def __init__(self,coords,frame):
        self.coords = coords
        self.frame = frame
    def to(self,newframe):
        """Returns a Directional expressing the same physical direction in spac
        #what here?
        return Point(self.coords,newframe)
```

In [ ]: