Robotics: Principles and Practice

Module 4: Robot Manipulators

Lecture 2: Object pose specification with homogenous transformations and vectors & quaternions

David Vernon
Carnegie Mellon University Africa

www.vernon.eu

Recall, we have developed a system where we can

specify the position and orientation of coordinate reference frames anywhere

with respect to each other

w.r.t. station frame of reference

or

with respect to a given base frame

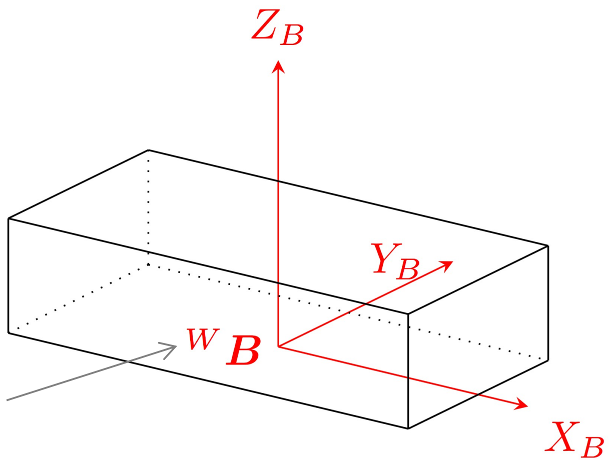
w.r.t. fixed world frame of reference

This, in itself, is not much use since the world you and I know does not have too many coordinate reference frames in it

What we really require is a way of identifying the pose of objects

Position and Orientation:
Six degrees of freedom

The trick, and it is no more than a trick, is to attach a coordinate frame to an object, *i.e.* symbolically glue an XYZ frame into an object simply by defining it to be there

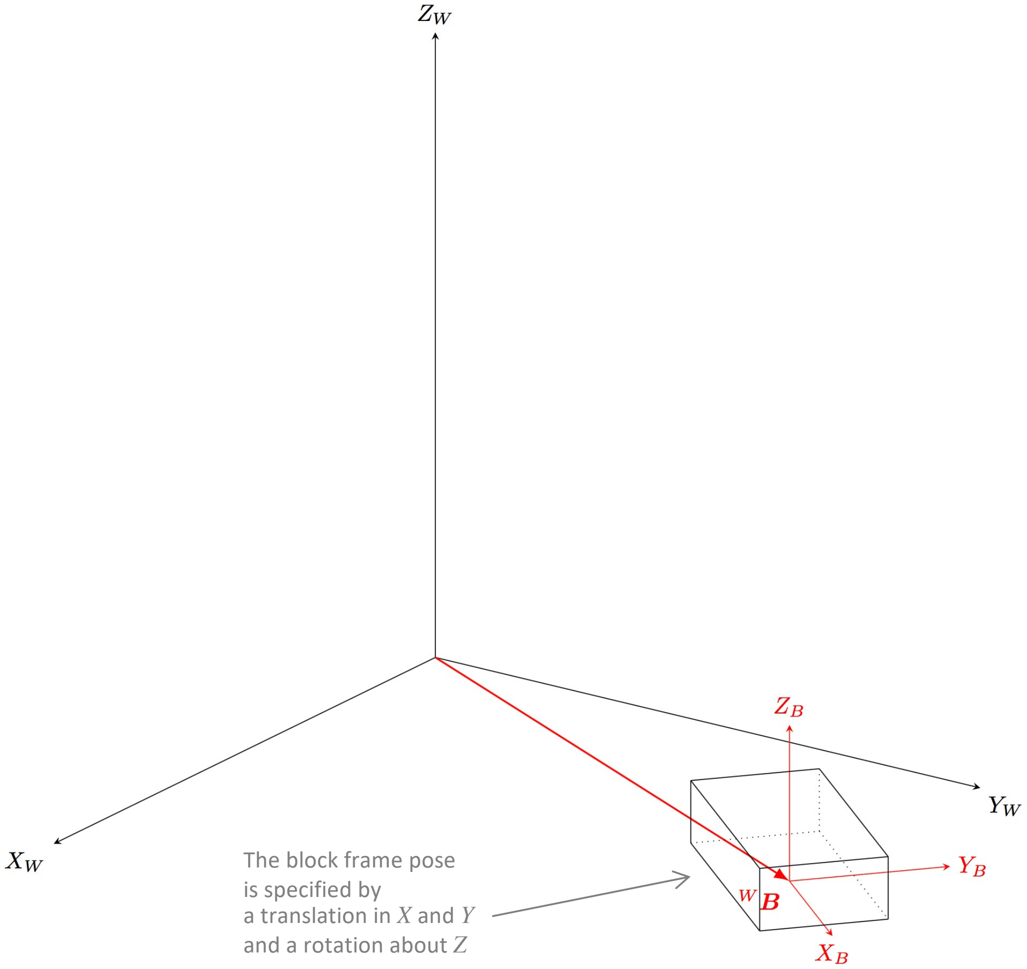


The block frame is defined with respect to the world frame of reference

- As we rotate and translate the coordinate frame, so we rotate and translate objects

- We can arbitrarily position and orient a coordinate frame – and an object – by specifying the required translations and rotations

- Thus, we specify the pose of an object by specifying its associated coordinate frame (homogeneous transformation)

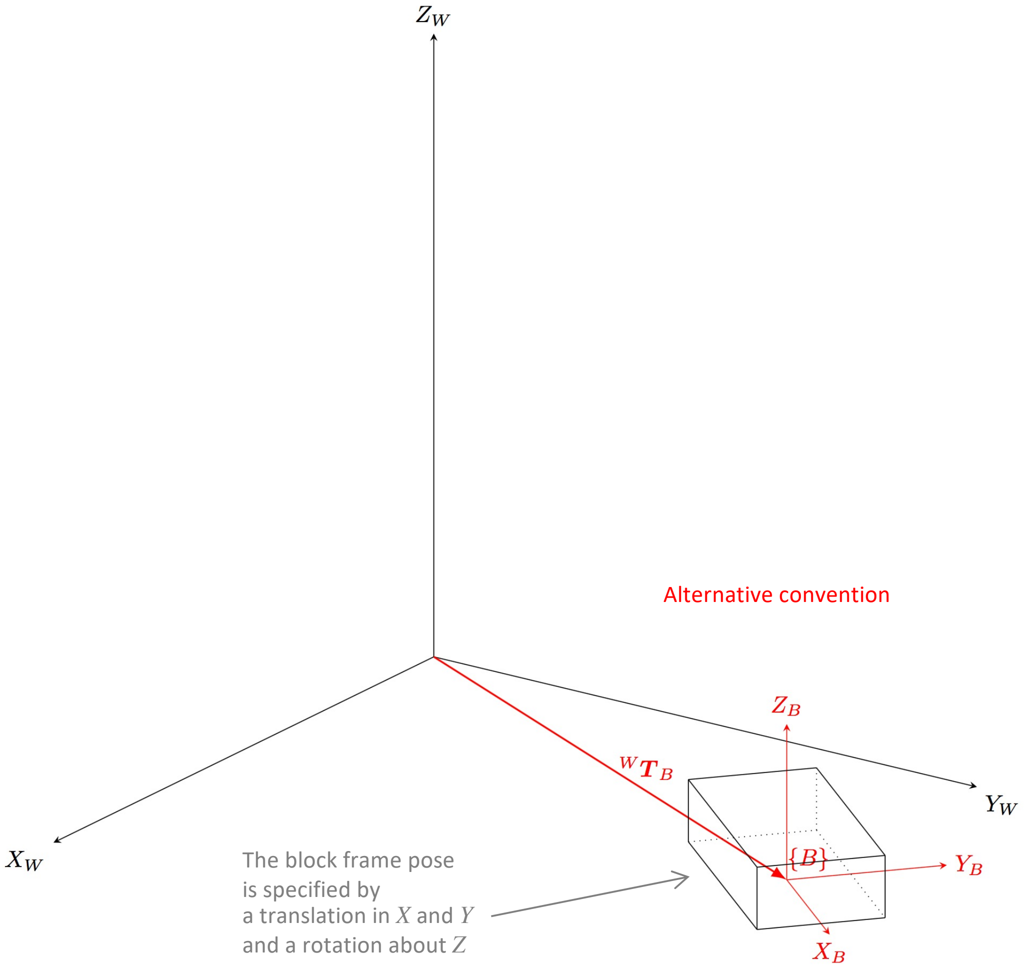


The block frame pose is specified by a translation in X and Y and a rotation about Z

- As we rotate and translate the coordinate frame, so we rotate and translate objects

- We can arbitrarily position and orient a coordinate frame – and an object – by specifying the required translations and rotations

- Thus, we specify the pose of an object by specifying its associated coordinate frame (homogeneous transformation)



Alternative convention

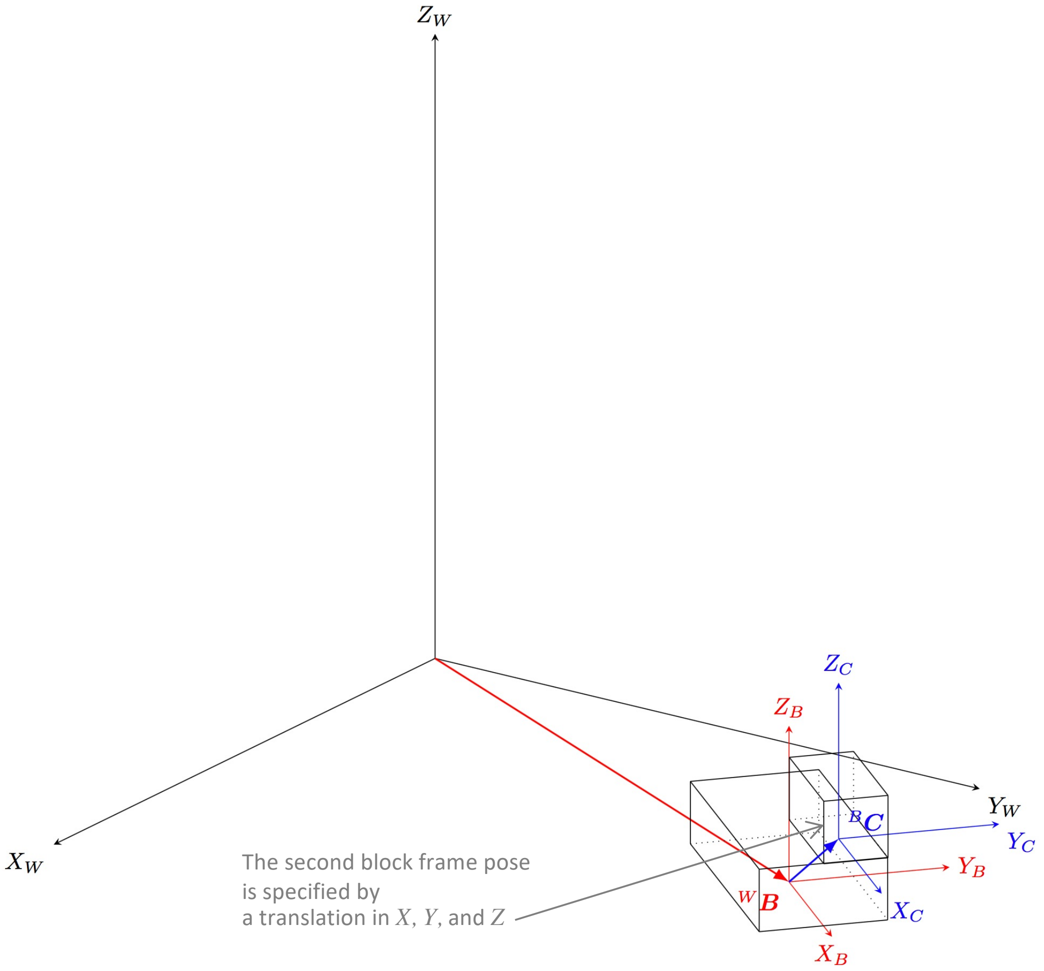
The block frame pose is specified by a translation in X and Y and a rotation about Z

- We can arbitrarily position and orient one object, i.e. its pose, with respect to another object

- How? By specifying the required translations and rotations of its associated coordinate frame (homogeneous transformation)

i.e. ${}^{B}C = \mathrm{Trans}(x, y, z)$

These values represent translations along the X_B, Y_B, Z_B axes;
the values of the translations depend on the dimensions of the objects



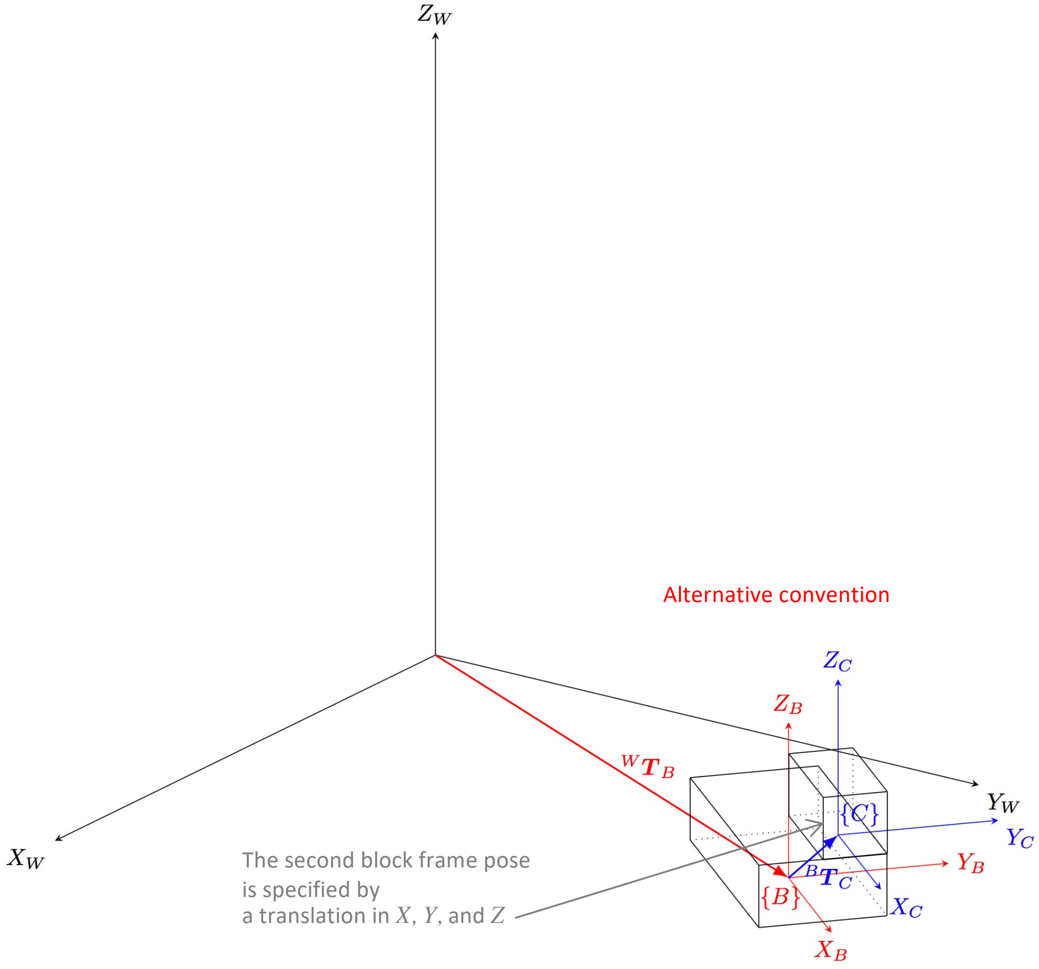
The second block frame pose
is specified by
a translation in X, Y, and Z

- We can arbitrarily position and orient one object, i.e. its pose, with respect to another object

- How? By specifying the required translations and rotations of its associated coordinate frame (homogeneous transformation)

i.e. $^{B}C = \mathrm{Trans}(x, y, z)$

These values represent translations along the X_B, Y_B, Z_B axes;
the values of the translations depend on the dimensions of the objects



Alternative convention

The second block frame pose
is specified by
a translation in X, Y, and Z

Specifying Pose in ROS

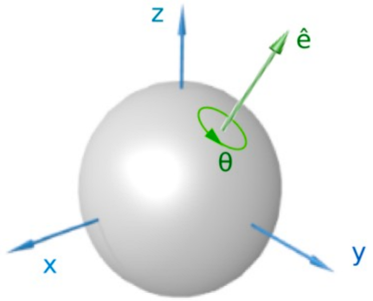
- We will use homogeneous transformations to specify a frame of reference, for end-effector and object pose

- ROS uses a different (but entirely equivalent) approach
  - Specify the origin of the frame as a 3-D vector
  - Specify the orientation of the frame as a quaternion: a single rotation about some (appropriate) axis

Specifying Pose in ROS

- Euler's rotation theorem states that any displacement of a rigid body (in 3D space), such that a point on the rigid body remains fixed,

  is equivalent to a single rotation θ about some axis that runs through the fixed point



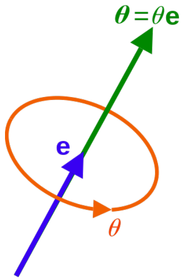
https://en.wikipedia.org/wiki/Euler%27s_rotation_theorem

- The axis of rotation is known as an Euler axis, typically represented by a unit vector **ê**

See: https://en.wikipedia.org/wiki/Rotation_formalisms_in_three_dimensions

Specifying Pose in ROS

- The product by $\theta\,\hat{\mathbf{e}}$ is known as an axis-angle



https://en.wikipedia.org/wiki/Axis–angle_representation

- Quaternions are a simple way to encode this axis–angle representation of a rotation in four numbers

Specifying Pose in ROS

- Quaternions are hypercomplex numbers

$$q = w + \overbrace{x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k}}^{\text{Vector part (imaginary part)}}$$

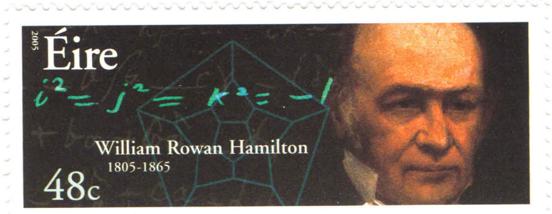
Scalar part (real part)

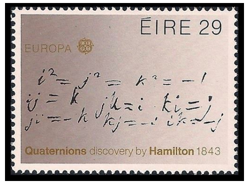
w, x, y, and z are real numbers

quaternion units

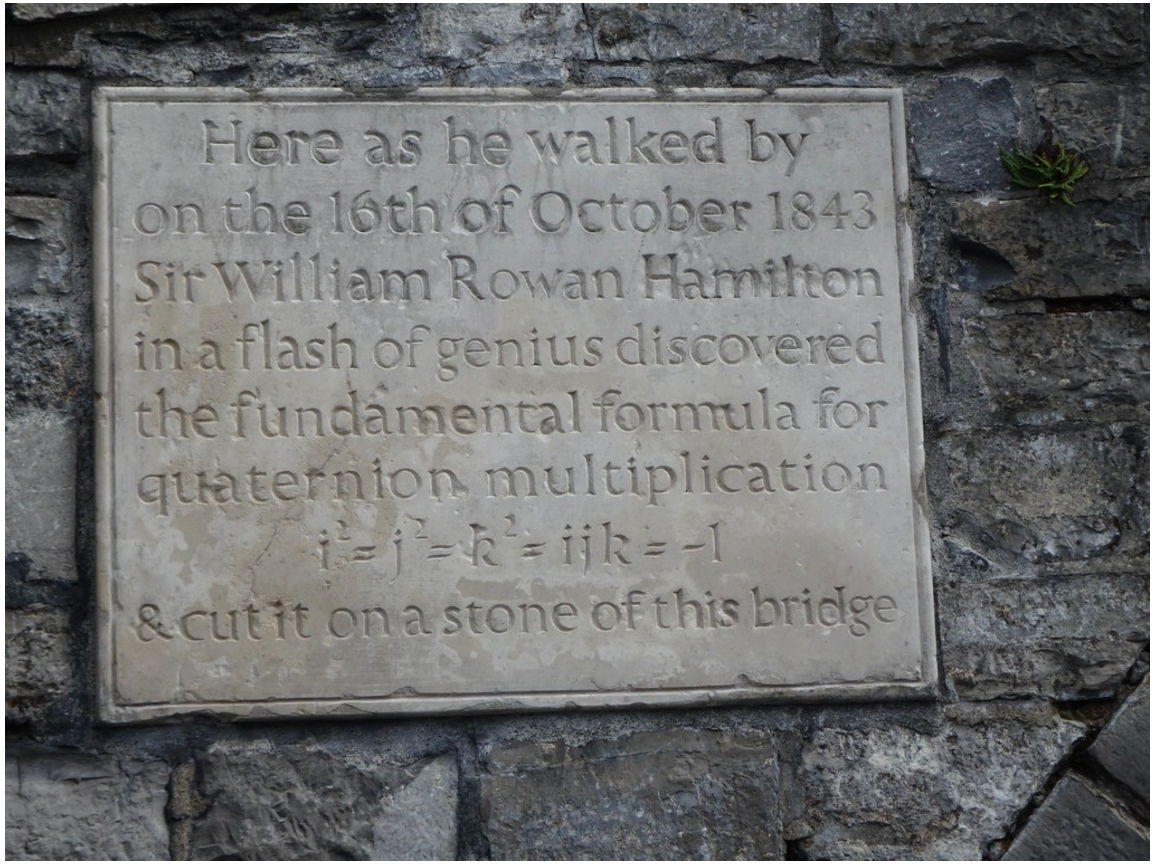
$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i}\,\mathbf{j}\,\mathbf{k} = -1$$

- Discovered by Irish mathematician William Rowan Hamilton in 1843





Specifying Pose in ROS


Here as he walked by
on the 16th of October 1843
Sir William Rowan Hamilton
in a flash of genius discovered
the fundamental formula for
quaternion multiplication
$i^2 = j^2 = k^2 = ijk = -1$
& cut it on a stone of this bridge

Broom Bridge in Dublin, Ireland

https://www.flickr.com/photos/infomatique/44408785822

Specifying Pose in ROS

- A rotation of θ about the Euler axis $\hat{\mathbf{e}} = e_x\,\mathbf{i} + e_y\,\mathbf{j} + e_z\,\mathbf{k}$ is given by

$$\begin{aligned} q &= w + x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k} \\ &= \cos(\theta/2) + e_x \sin(\theta/2)\,\mathbf{i} + e_y \sin(\theta/2)\,\mathbf{j} + e_z \sin(\theta/2)\,\mathbf{k} \end{aligned}$$

NB: half the angle

- Equivalently

$$q = \begin{pmatrix} w \\ x \\ y \\ z \end{pmatrix}$$

Specifying Pose in ROS

In ROS, we write it slightly differently

$$q = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \end{array}\right\} \leftarrow \text{Axis part} \\ \leftarrow \text{Angle part} \end{array}$$

Specifying Pose in ROS

A rotation of -90° about the Z axis would be specified in quaternion notation as

$$q = \begin{bmatrix} 0.7071 \\ 0.0 \\ 0.0 \\ -0.7071 \end{bmatrix}$$

$$q = \begin{bmatrix} 0.0 \\ 0.0 \\ -0.7071 \\ 0.7071 \end{bmatrix} \longleftarrow \text{ROS}$$

Recommended Reading

D. Vernon, Machine Vision – Automated Visual Inspection and Robot Vision, Prentice Hall International, 1991. Chapter 8.

http://vernon.eu/publications/91_Vernon_Machine_Vision.pdf

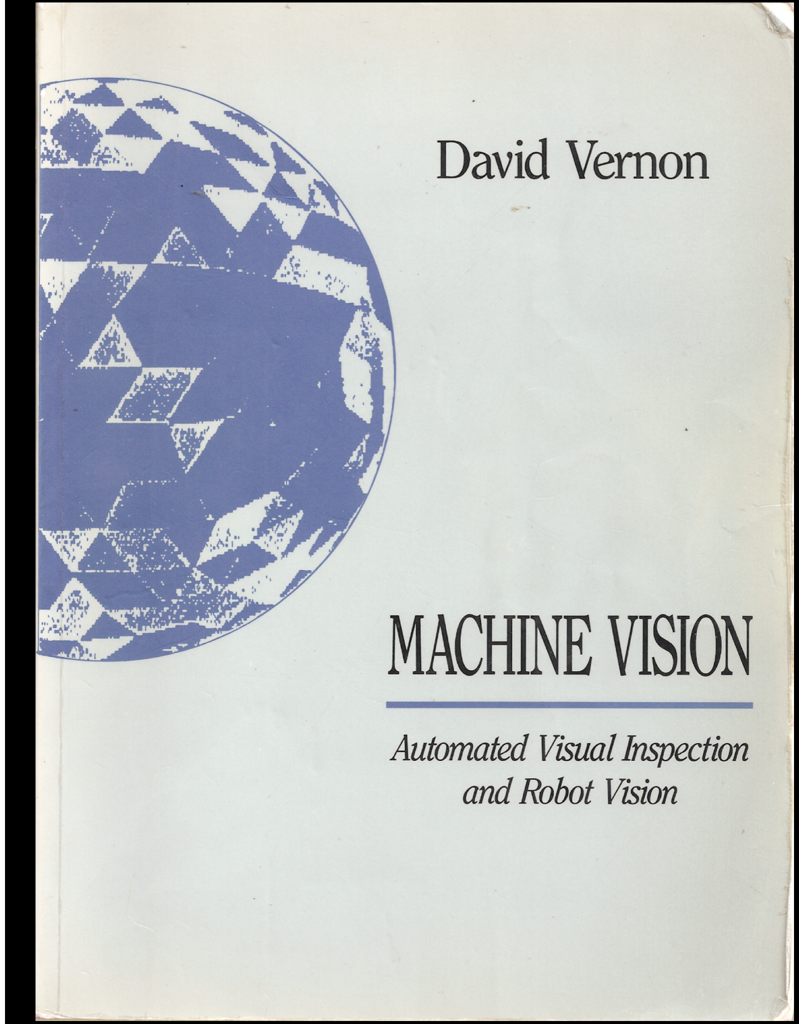
Similar material to that presented in this lecture.

R. P. Paul, Robot Manipulators – Mathematics, Programming, and Control, MIT Press, 1981. Chapter 1.

https://books.google.rw/books?id=UzZ3LAYqvRkC&printsec=frontcover&source=gbs_ViewAPI&redir_esc=y#v=onepage&q&f=false

Similar material to that presented in this lecture but complete comprehensive treatment.

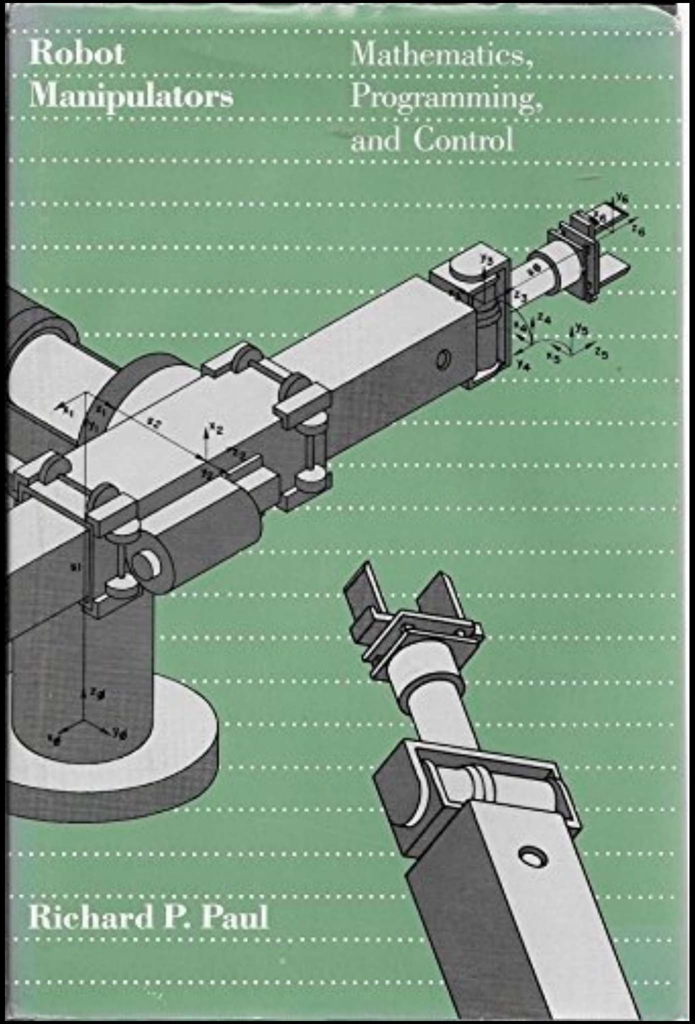
P. Corke, Robotics, Vision and Control, 2nd Edition, Springer, 2017.
Comprehensive contemporary treatment; highly recommended.



David Vernon

MACHINE VISION

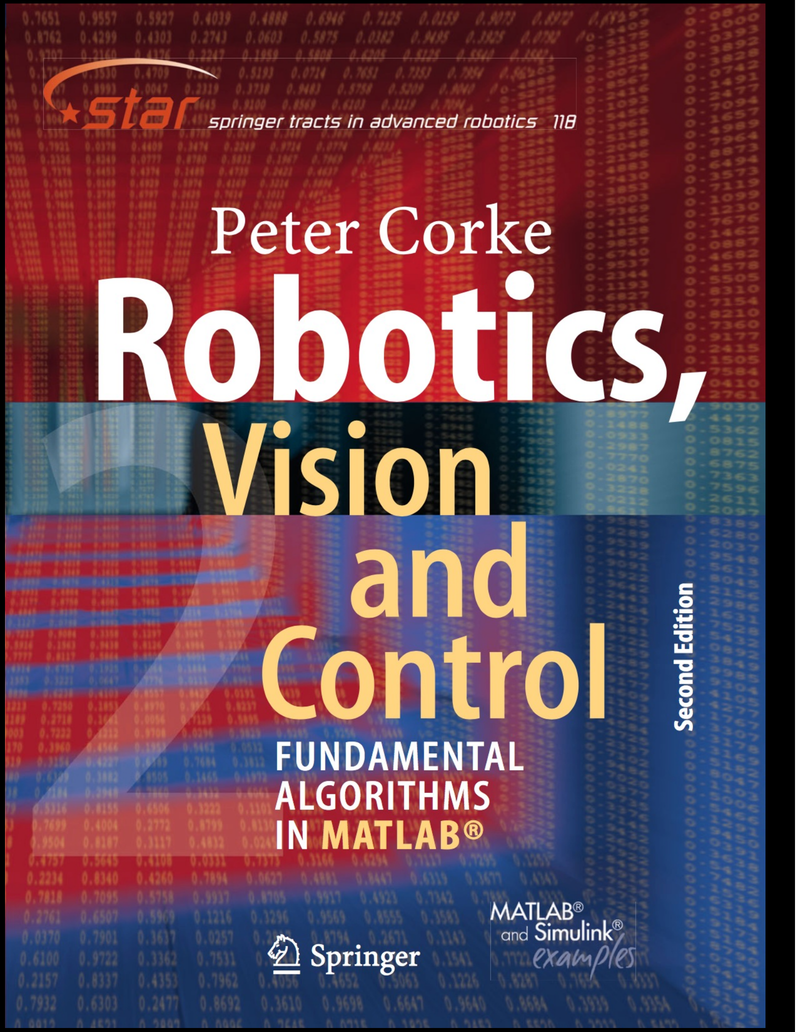
Automated Visual Inspection and Robot Vision



Robot Manipulators

Mathematics, Programming, and Control

Richard P. Paul



star springer tracts in advanced robotics 118

Peter Corke

Robotics, Vision and Control

FUNDAMENTAL ALGORITHMS IN MATLAB®

Second Edition

Springer

MATLAB® and Simulink® examples