Robotics I

Instructor's notes on robot kinematics

Wyatt S. Newman

I Introduction

For a robot to perform a useful task, it must achieve some pose which places its hand or tool in a location appropriate for influencing its environment in the desired manner. For example, if the robot is to pick up an object, then the robot must place its gripper in a position coincident with the object. A robot achieves a desirable pose by utilizing its sensors, actuators and computer controls. For humans, the dominant sense used for achieving successful object acquisition is vision. For robots, the most common sensing means used is electronic angle sensors on each movable joint (e.g., encoders or resolvers).

For a computer to direct a robot to a desirable pose, its program must incorporate geometric information. Coordinates of the desired object must be known. Also, coordinates of the hand (or tool) must be computable in terms of the available sensors (typically joint sensors). The computations which relate joint-sensor measurements to hand position and orientation is known as *kinematics*.

In this two-week module, we will introduce a formal technique for unambiguously defining the position and orientation of objects, and for computing the position and orientation of a robot's hand or tool based on joint-sensor measurements. The process of deriving hand coordinates from joint sensors is relatively friendly. It is always solvable for any arbitrary "chain" of movable "joints" connecting multiple rigid "links". Computing hand coordinates from joint angles is called *forward kinematics*.

The complementary process to forward kinemantics is *inverse kinematics*. With inverse kinematics, one starts with a specification of the *desired* hand coordinates, then computes the joint angles which would produce that desired pose. Inverse kinematics is necessary for defining the required angles for each motor. This information feeds into a control system, which is responsible for driving the power amplifiers to enforce the commanded motor angles. The process of computing inverse kinematics is considerably more complex than forward kinematics. Solutions are not guaranteed to exist. Multiple distinct solutions or entire classes of solutions are also possible. Computing inverse kinematics requires solving a system of simultaneous nonlinear algebraic equations. The present notes present only forward kinematics.

II Defining Coordinate Frames

When we (as humans) refer to objects in our environment, we typically appeal to visual attributes and to relative position. It is uncommon to quote specific coordinates. For a computer-controlled robot, however, coordinates are the most natural (and usually necessary) specification of object location. To refer to object coordinates, we must define a "world" coordinate reference frame. Imagine, for example, that you are in a factory, and one corner of the room is used to define a reference coordinate system. The z axis might be defined vertically up from the floor, lying along the corner line between the two walls. The x axis and the y axis could run along the floor in the corner lines between the floor and respective walls. The three axes meet in the corner point. There is another restriction: the three axes must form a right-hand triad.

With respect to our world coordinate frame, we can define the position of objects within our environment.

For example, consider a block in the room. We could paint a small reference dot on the block, and then refer to the coordinates, (x, y, z), of that point with respect to the world reference frame. We will typically express such points in the format of a 3-row, single-column *vector*. We can refer to this vector of coordinates abstractly; I will choose the symbol $\mathbf{p}$ to represent this body-fixed point.

If our object were a mere point, such a vector-position description would be adequate. However, if we care about *orientation*, then we need to specify more information. For example, we may care that some surface of the block be placed parallel to and coincident with a table.

To specify orientation, we can label yet another set of coordinate axis on our object of interest. Imagine that we painted, e.g., an x and y axis on our object. The two axis must be orthogonal (90 degree angle between them), and they must intersect at the reference point we defined on the body. Having defined x and y "body-fixed" axes, the z axis follows uniquely from the definition of a right-hand coordinate system.

With our reference definitions, we can now express both the position and orientation of our object with respect to the world frame. Here is one way to do so. Let us define 3 unit-length vectors lying along the body-frame x, y and z axes, respectively. The tail of each vector is coincident with the point $\mathbf{p}$. The tip of each vector is located at a point in space which is expressable in terms of coordinates with respect to the world reference frame. I will define these 3 points in space in terms of 3 vectors of coordinates as $\mathbf{n}'$, $\mathbf{t}'$ and $\mathbf{b}'$, respectively. Invoking these definitions, we can define three vectors $\mathbf{n}$, $\mathbf{t}$ and $\mathbf{b}$, in terms of vector subtractions as $\mathbf{n} = \mathbf{n}' - \mathbf{p}$, $\mathbf{t} = \mathbf{t}' - \mathbf{p}$ and $\mathbf{b} = \mathbf{b}' - \mathbf{p}$.

One way to interpret these vectors $\mathbf{n}$, $\mathbf{t}$, $\mathbf{b}$, is to picture three unit vectors extending from the origin of the world reference frame, where these vectors are parallel to the body-fixed x, y and z axes, respectively. That is, the vectors $\mathbf{n}$, $\mathbf{t}$, and $\mathbf{b}$ define the *orientation* of the object of interest, independent of the body's *position*.

It will be convenient to express the orientation as a collection of the three direction vectors, $\mathbf{n}$, $\mathbf{t}$, $\mathbf{b}$. We can construct a 3-by-3 matrix composed of these vectors, which we will label R, as follows:

$$R = [\mathbf{n}\mathbf{t}\mathbf{b}] \tag{1}$$

That is,

$$R = \begin{bmatrix} n_x & t_x & b_x \\ n_y & t_y & b_y \\ n_z & t_z & b_z \end{bmatrix} \tag{2}$$

By specifying the nine component values of R, we can uniquely identify the orientation of a coordinate frame, and thus the body which is associated with that coordinate frame.

III Rotation Matrices

The matrix R may seem to have been defined arbitrarily. In fact, the choices made in defining R result in a surprisingly large number of interpretations and uses. Since R is composed of elements of perpendicular vectors of unit length, R has following properties. Since each column vector of R is perpendicular to the other two vectors,

$$\begin{aligned} \mathbf{n}^T\mathbf{t} &= 0 \\ \mathbf{t}^T\mathbf{b} &= 0 \\ \mathbf{b}^T\mathbf{n} &= 0 \end{aligned} \tag{3}$$

Since each column vector of R is of unit length,

$$|\mathbf{n}| = \sqrt{n_x^2 + n_y^2 + n_z^2} = 1$$

$$|\mathbf{t}| = \sqrt{t_x^2 + t_y^2 + t_z^2} = 1 \qquad (4)$$
$$|\mathbf{b}| = \sqrt{b_x^2 + b_y^2 + b_z^2} = 1$$

Thus, the elements of R cannot be chosen arbitrarily. Rather, of the 9 elements, there are only 6 "degrees of freedom" available. After abiding by the 6 constraints of equations 3 and 4, there are only 3 "free" choices available. Thus, we may infer that specification of orientation involves prescribing 3 parameters. Altogether, specification of position and orientation of a rigid body requires 6 parameters: three for position and three for orientation. Thus, the space in which we manipulate objects may be thought of as 6-dimensional.

Here is another useful property of the R matrix: it is *orthonormal*. That is:

$$R^T R = R R^T = I \qquad (5)$$

where R^T is "R-transpose", the operation of rearranging elements of a matrix by exchanging them symetrically about the diagonal, i.e. $R^T_{ij} = R_{ji}$. The matrix I is the identity matrix, consisting of ones along the diagonal and zeroes elsewhere. The matrix product $C = AB$ is computed as $C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$, where matrix A has n columns and matrix B has n rows. (C_{ij} refers to the element of matrix C located in the i'th row and j'th column).

Since $R^T R = I$, we may define $R^{-1} = R^T$. In general, the *inverse* of a matrix is another matrix which, when used to premultiply the first matrix, produces the identity matrix. A matrix inverse is usually tedious to compute. Orthogonal matrices, such as our orientation matrix R, have particularly easy-to-compute inverses.

Although we seemingly defined R recklessly, it is, in fact, closely related to common rotation matrices. Consider the following thought experiment. Our object is initially aligned with the world reference frame. The respective origins are coincident and the respective axes are colinear. Now, we take our object and rotate it about the z axis by an angle θ_z. (Such rotations are "signed". A positive rotation about the z axis is counterclockwise with respect to a "clock", where the z axis is pointing outwards from the face of the reference "clock"). Upon such rotation, the object's body-fixed coordinate frame is no longer lined up with the world frame (only the z axes remain colinear). The coordinates of the tip of our $\mathbf{n}$ axis of the rotated body-fixed frame are:

$$\begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} = \begin{bmatrix} \cos(\theta_z) \\ \sin(\theta_z) \\ 0 \end{bmatrix} \qquad (6)$$

The unit vector lying along the rotated $\mathbf{t}$ axis would have tip coordinates of:

$$\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = \begin{bmatrix} -\sin(\theta_z) \\ \cos(\theta_z) \\ 0 \end{bmatrix} \qquad (7)$$

and the $\mathbf{b}$ axis would remain unaffected:

$$\begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \qquad (8)$$

We can construct the matrix R composed of the vectors $R = [\mathbf{n}\mathbf{t}\mathbf{b}]$ to get:

$$R = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix} = R_z(\theta_z) \qquad (9)$$

For shorthand, we define the matrix R resulting from a pure rotation about z by angle θ_z as $R_z(\theta_z)$.

Similarly, if we were to repeat our experiment for a rotation about axis x by an angle θ_x, our resulting orientation would be:

$$R_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \tag{10}$$

A rotation about axis y by an amount θ_y would produce orientation:

$$R_y(\theta_y) = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \tag{11}$$

Given our observations above, we may now think of the matrix R as more than a collection of vector-tip coordinates. We may interpret R as an *operator*, which produces a change in orientation. Consider, for example, our original pose, aligned with the world coordinate frame. We can express the orientation of our body as the identity matrix. Now, consider premultiplying this matrix by the matrix $R_x(\psi)$. The result of this matrix multiply is, identically, $R_x(\psi)$, which corresponds to the new representation of orientation of our body, as rotated about the x axis by angle ψ.

We can cascade such rotation operations. Suppose the initial orientation of our object is described by the matrix R_0. Now, imagine that we rotate this body about its own x axis (**n**) by angle ψ. The resulting orientation of the body will be R', where $R' = R_x(\psi)R_0$. (The matrix R' will be 3x3, resulting from the product of the 3x3 matrices R_x and R_0). After having rotated our object by ψ about its own x axis, we next rotate it by angle θ about its *own* y axis.

There is a subtle but important distinction here. Our second rotation is to be performed about the *body-fixed* y axis, (**t**), which (after our initial rotation) is not the same as the world reference-frame y axis. Our successive rotations are not rotations about any axis aligned with the world reference frame. Rather, they are rotations about body-fixed axes.

The orientation after the second rotation is R'', where $R'' = R_y(\theta)[R_x(\psi)R_0]$. Now, consider yet a third rotation, $R_z(\phi)$. The resulting orientation, R''', after our three successive rotations is:

$$R''' = R_z(\phi)\{R_y(\theta)[R_x(\psi)R_0]\} = [R_z(\phi)R_y(\theta)R_x(\psi)]R_0 \tag{12}$$

Rearrangement of the order of operations above is permissible, since matrix multiplication has the property of algebraic ***associativity***. That is, $A(BC) = (AB)C$. Since matrices associate, we may reorder the sequence of operations to define the product:

$$R_{\phi,\theta,\psi} = R_z(\phi)R_y(\theta)R_x(\psi) \tag{13}$$

The matrix $R_{\phi,\theta,\psi}$ consists of nine elements and may be interpreted as a description of a body's orientation. We can also interpret $R_{\phi,\theta,\psi}$ as an operator which produces successive rotations of a body. Thus, the R-matrix representation of a body can be thought of as a description of orientation, or equivalently as a description of a physical process which would produce the body's current orientation.

The matrix $R_{\phi,\theta,\psi}$ is fully defined by three parameters: ϕ, θ and ψ. Thus, we see that our original specification of orientation in terms of nine parameters with six constraints can, indeed, be specified more compactly in terms of only three independent parameters. This particular choice of three parameters is referred to as "roll, pitch and yaw" coordinates.

The roll-pitch-yaw notation is particularly convenient for vehicles (e.g., ships, aircraft, spacecraft). Typically, the body-z axis is defined as pointing from the tail of the vehicle towards the front of the vehicle. A "roll" is a rotation about the body's z axis, e.g. as a "barrel-roll" performed by an airplane, or a ship's roll (rocking about its keel) in waves. The vehicle's y axis is typically defined as pointing (e.g.) perpendicular from the z axis towards the side or the wing (ordinarily horizontal, perependicular to gravity). A rotation about the body's y axis is the "pitch". In an aircraft, the pitch produces a climb or a dive. The vehicle's x axis

complete's a right-handed coordinate frame. It is typically defined as pointing perpendicular to the z axis towards the top or roof of the vehicle (thus ordinarily pointing "up"). Rotations about the vehicle's x axis is called a "yaw". Yaw is ordinarily produced by the rudder (or tail), typically controlling heading in a horizontal plane. The roll-pitch-yaw definitions should help to clarify rotations with respect to body-defined frames vs rotations with respect to a world reference frame.

Invoking our interpretation of rotation matrices, we could always define the orientation of a body by referring to a sequence of rotations performed from some initial orientation. The history of rotations could be thousands of motions, yet we could define the resulting rotation with nine components within a single rotation matrix. The matrix could be computed by performing the sequential matrix multiply of all of the preceding rotation operations. Further, it is always possible to express any rotation matrix as an equivalent roll, pitch and yaw. Thus, although a body's orientation might be the result of millions of sequential rotations, we can express the equivalent change in orientation in at most *three* rotations of roll, pitch and yaw.

The associative property of matrix multiplication is highly useful. Unfortunately, matrix multiplication does not enjoy the property of *commutivity*. That is, $AB \neq BA$. We can visualize this in a thought experiment with rotations. Consider a car with a body-fixed frame defined as above (z pointing horizontally from trunk to front, x pointing vertically from the road through the roof and y pointing horizontally from the centerline towards the right door). We will perform a rotation of 90 degrees about z followed by a rotation of 90 degrees about y. After the first rotation, the car is resting on its side on its right door(s). The car's y axis is now oriented vertically, pointing down. A rotation about the y axis at this point changes the direction which the car is "headed", but the car remains resting on its right door. This order of operations corresponds to the matrix multiplication $R_z(\pi/2)R_y(\pi/2)$.

Now consider reversing the order of operations. We first rotate the vehicle about its y axis by $\pi/2$. The resulting pose has the car pointing upright, resting on its tail-lights. The new orientation of the body's z axis is pointing straight up. Now consider a rotation about the body's z axis by $\pi/2$. After this rotation, the car is still pointing upwards, resting on its tail-lights. This sequence of rotations corresponds to $R_y(\pi/2)R_z(\pi/2)$. Clearly, the two scenarious produce quite different orientations. Thus, $R_z R_y \neq R_y R_z$. By this counterexample, we have proven that rotation matrices (in fact, matrices in general) do not commute.

We have seen that the R matrix may be interpreted as both a specification of orientation, as well as an operator which produces arbitrary rotations. We next consider a third interpretation of the R matrix in the context of coordinate transformations.

IV Coordinate Transformations

Coordinate transformations are essential for controlling robots. We may know, for example, the position of a desired object with respect to a world reference frame. We may also know the position of the base of a robot arm with respect to that same reference frame. What we would need in order to control the robot to acquire the object is a computation of the object's location with respect to the robot's location.

For the position component, (x, y and z), such computation is relatively simple. Let us assume there is a coordinate frame associated with the base of the robot, a coordinate frame associated with the desired object, and a world reference frame. An observer standing at the world reference frame observes the object body at location $\mathbf{p}_{b/0}$ (we use the /0 to denote that the coordinate measurements are performed by an observer using the world, or "0", reference frame). We could also denote the orientation of the object's body as $R_{b/0}$. Similarly, we can specify the location and orientation of the robot's base coordinate frame as $\mathbf{p}_{r/0}$ and $R_{r/0}$, respectively. We can compute the displacement of the object of interest from the robot's base as the vector difference $\mathbf{p}_{b/0} - \mathbf{p}_{r/0}$.

Here is a subtle issue. We can define a point in space in terms of a vector. When we refer to this vector abstractly, without filling in numerical values for its components and without including specification of a measurement reference frame, the vector still has meaning. For example, we could observe the location of

the sun and define this "point" in space as $\mathbf{p}_{Sun}$. An observer on Earth and an observer on Mars could agree about the existence and location of the Sun, and both observers could refer to this point using the same name "$\mathbf{p}_{Sun}$". Both observers would be referring to exactly the same location. However, when it becomes necessary to specify numerical values for the definition of $\mathbf{p}_{Sun}$, the two observers (using their own personal coordinate reference frames) will obtain entirely different numbers for $\mathbf{p}_{Sun}$. As we get more specific, it becomes necessary to distinguish $\mathbf{p}_{Sun/Earth}$, the coordinates of the Sun as specified by the observer on Earth, from $\mathbf{p}_{Sun/Mars}$, the coordinates of the Sun as specified by the observer on Mars.

Although our two hypothetical observers use two different coordinate frames, their independent measurements are reconcilable. Suppose we knew the the position and orientation of the Mars reference frame, as measured with respect to the Earth's reference frame, call it $\mathbf{p}_{Mars/Earth}$ and $R_{Mars/Earth}$. We can define the components of $R_{Mars/Earth}$ as:

$$R_{Mars/Earth} = \left[\mathbf{n}_{M/E}\mathbf{t}_{M/E}\mathbf{b}_{M/E}\right] = \begin{bmatrix} n_{x,M/E} & t_{x,M/E} & b_{x,M/E} \\ n_{y,M/E} & t_{y,M/E} & b_{y,M/E} \\ n_{z,M/E} & t_{z,M/E} & b_{z,M/E} \end{bmatrix} \tag{14}$$

The above is nothing new. We can certainly define the orientation of the "body-fixed" frame of any object (in this case, Mars) per the rotation matrices defined previously. The vectors $\mathbf{n}_{M/E}$, $\mathbf{t}_{M/E}$, and $\mathbf{b}_{M/E}$ are the direction vectors of the x, y and z axes used by the Martian observer.

If the Martian measurement of the position of the sun is:

$$\mathbf{p}_{Sun/Mars} = \begin{bmatrix} p_{x,S/M} \\ p_{y,S/M} \\ p_{z,S/M} \end{bmatrix} \tag{15}$$

then we can interpret this as the Martian observer saying: "proceed along my $\mathbf{n}$ axis a distance $p_{x,S/M}$, then turn 90 degrees and proceed in the direction of my $\mathbf{t}$ axis a distance $p_{y,S/M}$. Turn again and proceed in the directon of my $\mathbf{b}$ axis a distance $p_{z,S/M}$, and you will be at the center of the Sun." Mathematically, this corresponds to the motion:

$$\mathbf{p}_{Sun/Mars} = p_{x,S/M}\mathbf{n}_{Mars} + p_{y,S/M}\mathbf{t}_{Mars} + p_{z,S/M}\mathbf{b}_{Mars} = R_{Mars}\mathbf{p}_{Sun/Mars} \tag{16}$$

Our three sequential steps are expressed above in terms of a single matrix-vector multiplication.

Our observer on Earth could get to the same location (the sun) by first proceeding to Mars (conceptually), then following the directions from Mars. That is, a valid description of how to get from the Earth to the sun, $\mathbf{p}_{Sun/Earth}$, is:

$$\begin{aligned} \mathbf{p}_{Sun/Earth} &= \mathbf{p}_{Mars/Earth} + \\ &\quad p_{x,S/M}\,\mathbf{n}_{Mars/Earth} + p_{y,S/M}\,\mathbf{t}_{Mars/Earth} + p_{z,S/M}\,\mathbf{b}_{Mars/Earth} \\ &= \mathbf{p}_{Mars/Earth} + R_{Mars/Earth}\,\mathbf{p}_{Sun/Mars} \end{aligned} \tag{17}$$

We can generalize from the above example. If we are given the coordinates of a point with respect to a reference frame A, call it $\mathbf{p}_{/A}$, and if we are given the position and orientation of reference frame A with respect to reference frame B, $\mathbf{p}_{A/B}$ and $R_{A/B}$, respectively, then we can compute the coordinates of the named point with respect to reference frame B, $\mathbf{p}_{/B}$, as follows:

$$\mathbf{p}_{/B} = \mathbf{p}_{A/B} + R_{A/B}\mathbf{p}_{/A} \tag{18}$$

We thus see yet another interpretation and use for rotation matrices. They can be used to perform coordinate transformations, expressing measurements taken in one reference frame to consistent values in another reference frame.

The coordinate transform expressed by equation 18 can be applied successively to perform multiple, sequential transformations. For example, if we know the position of the sun with respect to Pluto, and we knew

the position and orientation of Pluto's reference frame with respect to Neptune, and we knew the position and orientation of Neptune's reference frame with respect to Uranus, ..., and so on eventually to Earth, we could compute the position of the sun with respect to Earth's reference frame.

Successive coordinate-frame computations are particularly useful for robots. It is convenient to define a coordinate frame associated with each rigid link of a robot, then define transformations between coordinate frames of links sequentially connected via motorized joints. In fact, such coordinate-frame transformations are so common, it is useful to define transformation matrices which are particularly well suited for computer implementation. These matrices are called *Homogeneous Transformations*.

V Homogeneous Transformations

We saw in the preceding section that coordinate transformations can be accomplished using a matrix-vector multiply and a vector addition. If we wanted to transform coordinates successively through frames E, D, C, B and A, the computation could be written as:

$$\mathbf{p}_{/A} = \mathbf{p}_{B/A} + R_{B/A}\left\{\mathbf{p}_{C/B} + R_{C/B}\left[\mathbf{p}_{D/C} + R_{D/C}\left(\mathbf{p}_{E/D} + R_{E/D}\mathbf{p}_{/E}\right)\right]\right\} \tag{19}$$

Clearly, successive transformations get cumbersome quickly. To ease successive transformations, we define a homogeneous transformation matrix from from i to frame $i-1$, denoted as $A_{i/i-1}$ or as A_i^{i-1} as follows:

$$A_{i/i-1} = A_i^{i-1} = \left[\begin{array}{c|c} R_{i/i-1} & \mathbf{p}_{i/i-1} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \tag{20}$$

In equation 20, we construct a 4x4 matrix by assembling smaller blocks within the array. A 3x3 matrix (the rotation matrix $R_{i/i-1}$) defines the nine elements of $A_{i/i-1}$ in the upper left corner. The vector $\mathbf{p}_{i/i-1}$ defines the first, second and third rows of the fourth column of $A_{i/i-1}$. The fourth row of $A_{i/i-1}$ is *always* defined as [0 0 0 1]. It may not be obvious why it should be useful to carry along the "excess baggage" of the fourth row. This definition, though, permits us to express successive transformations as pure multiplies.

To take advantage of our new 4x4 homogeneous transform matrix, we also augment definitions of vectors to make them dimensionally consistent. We may define the 4x1 vector $\mathbf{P}$ from the 3x1 vector $\mathbf{p}$ according to:

$$\mathbf{P} = \left[\begin{array}{c} \mathbf{p} \\ 1 \end{array}\right] \tag{21}$$

That is, we convert the 3x1 vector $\mathbf{p}$ into the 4x1 vector $\mathbf{P}$ by defining the fourth row to be unity.

We can now illustrate the utility of our definitions, e.g. using the example of Eqn 19. Equation 19 can be written more simply using homogeneous coordinates as:

$$\mathbf{P}_{/A} = A_B^A\, A_C^B\, A_D^C\, A_E^D\, \mathbf{P}_{/E} \tag{22}$$

Following the definition of Eqn 20 and the rules of matrix multiplication, Eqn 22 performs the same computation as Eqn 19. To obtain the desired result ($\mathbf{p}_{/A}$), we extract the top three elements of the vector $\mathbf{P}_{/A}$. (The fourth element of $\mathbf{P}_{/A}$ will always be unity as a result of the specified computations).

In general, we write successive coordinate frame transformations compactly. For a point $\mathbf{p}_{/n}$ expressed in frame n, we can define the 4x1 vector $\mathbf{P}_{/n}$. We can state the sequence of transformations, from the frame n through sequential frames $n-1$, $n-2$, ..., to frame 0, to compute the point $\mathbf{P}_{/0}$, expressed with respect to the 0 frame. This computation can be expressed compactly as:

$$\mathbf{P}_{/0} = \prod_{i=1}^{n} A_i^{i-1}\mathbf{P}_{/n} = A_1^0 A_2^1 \cdots A_n^{n-1}\mathbf{P}_{/n} \tag{23}$$

Note that the matrix product $\prod_{i=1}^{n} A_i^{i-1}$ is, itself, a 4x4 matrix. Further, this matrix has its fourth row equal to [0 0 0 1], just as any other A-matrix. We can define this product as the matrix A_n^0:

$$A_n^0 = \prod_{i=1}^{n} A_i^{i-1} \tag{24}$$

The matrix A_n^0 contains the 3x3 rotation matrix R_n^0 in its upper left corner. The fourth column of A_n^0 includes the 3x1 vector $\mathbf{p}_{n/0}$, a vector which points from the origin of the 0-frame to the origin of the n-frame.

Unfortunately, the inverse of an A matrix, (A^{-1}, where $A^{-1}A = AA^{-1} = I$) is not as easy to compute as R^{-1}. A matrices are not orthonormal. However, since A matrices include R matrices, inverses of A matrices are still much easier to compute than for 4x4 matrices in general. An efficient formula for computing A^{-1} is:

$$A^{-1} = \left[\begin{array}{c|c} R & \mathbf{p} \\ \hline 0\ 0\ 0 & 1 \end{array}\right]^{-1} = \left[\begin{array}{c|c} R^T & -R^T\mathbf{p} \\ \hline 0\ 0\ 0 & 1 \end{array}\right] \tag{25}$$

We next introduce a method for defining individual coordinate frames on each movable link of a robot arm. We will then relate these frames through homogeneous computations, resulting in a complete forward-kinematic solution for an arbitrary robot arm.

VI Kinematic Modeling of Robot Arms: links and joints

Given our efficient method for expressing and computing coordinate transformations, our next task is to prescribe an unambiguous means of defining coordinate frames for robots. A commonly accepted approach is due to Denavit and Hartenberg, which produces a surprisingly general, compact, orderly and rigorous description of kinematics. Their scheme produces a set of describing parameters, referred to as the D-H parameters.

Figure 1 shows an abstraction of a general robot arm design. The arm consists of rigid *links* connected serially via rotational *joints*. We will define a body-fixed coordinate frame for each such link, from the floor on out to the robot's hand. If we can compute the coordinate transformation between each pair of successive link frames, then we can easily compute the net transformation from ground to the gripper. Such a computation would tell us where the robot's hand is in space. With this information, we can control the robot to perform desired tasks.

With sufficient generality, we can model a robot arm as an *open kinematic chain*. The "chain" consists of joints and links. We start by defining our "ground" (typically the floor to which the robot is bolted) as link number zero. All parts of the robot which are immovable with respect to ground are also defined as part of ground. The link number 1 is connected to link number 0 through a joint, which we define as joint 1. Most commonly, a robot joint consists of an electric motor, a transmission, some type of electronic angle sensor on the motor or the transmission, support bearings, and an output shaft. Joint 1 induces a motion of link 1. Since all other links are chained downstream of link 1, a rotation of joint 1 also induces motion of all other links. Link 1 is uniquely identifiable, though, since arbitrary motions of any combination of all other joints (except joint 1) will have *no* influence on the motion of link 1. All parts of the robot arm which move solely as a function of joint-1 motion comprise link 1.

Link 2 is connected to link 1 via joint 2. If we freeze joint 1, then the motion of link 2 depends solely on the rotation of joint 2. Although links 2 through n are all influenced by motions of joint 2, link 2 is *only* influenced by motions of joints 1 and 2; rotations of joints 1 through n have no influence on link 2. Similarly, we can define sequential joints 1 through n and links 1 through n, as illustrated in Fig 1.

In general, link $i-1$ and link i share the common joint number i. Link i is bounded by joint i (connecting

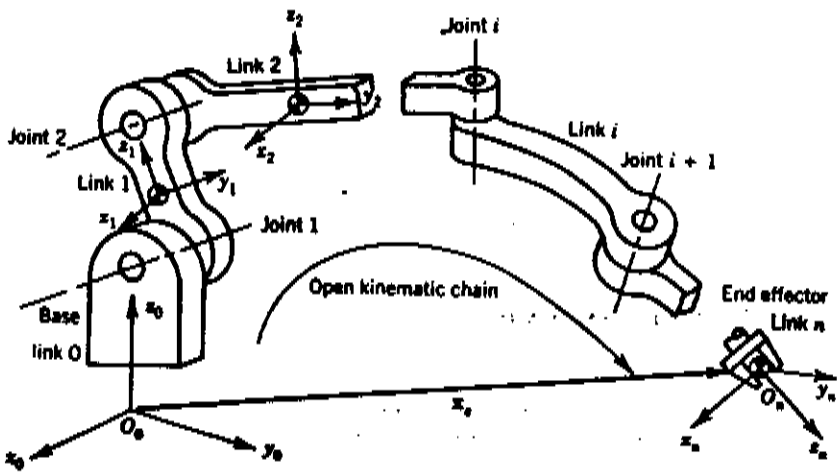

Figure 1: open kinematic chain

it to link $i-1$) and joint $i+1$ (which connects link i to link $i+1$), as illustrated by the abstract example of Fig 2. The final link, link n, includes the tool or gripper (generically called the *end effector*).

In this brief treatment, we will consider only *revolute* joints, which consist of a single-axis rotation (like a hinge). Although this is the most common type of robot joint, other types of joints are possible. Higher-order joints include multiple "degrees of freedom". An example is our shoulder joint, which is a ball and socket, capable of rotating in any of three orthogonal directions. Such higher-order joints may be modeled as a collection of single-degree-of-freedom hinge-type joints, so the present simplification is more applicable than it may seem. Another type of joint which is often used in robot designs is the *prismatic* joint. This type of joint produces a telescoping-type of motion (a pure translation). Prismatic joints do not occur in nature, but prismatic joints are not uncommon in machine designs. Prismatic joints can also be modeled using the techniques presented below. For brevity, however, prismatic joints will not be discussed here further.

VII Link i D-H parameters a_i and α_i

Let us now consider an arbitrary link, link i, which is bounded by joints i and $i+1$, as in Fig 2. Joint i is closer in the chain to ground than is joint $i+1$. We may refer to joint i as the *proximal* joint (closer to ground) and joint $i+1$ as the *distal* joint (more distant in the chain from ground). We wish to define a coordinate system on link i. In fact, the true physical shape of link i is irrelevant. The only important feature of link i is how joint axis i relates to joint axis $i+1$. This information (for each link) would be sufficient to compute the position and orientation of the end effector.

By invoking some clever definitions and observations, We may describe the important features of link i with just two numbers: the link length, a_i, and the link twist, α_i. This simplification is striking. Consider that we are trying to define the relative position and orientation of two joints in space (joints i and $i+1$). Defining the position and orientation of an object with respect to a reference frame requires 6 parameters. Dramatic simplifications for link and joint definitions in particular can be made, permitting specification in just 2 parameters.

Joints i and $i+i$ may be abstracted as two lines corresponding to respective joint axes. (See Fig 3) Each

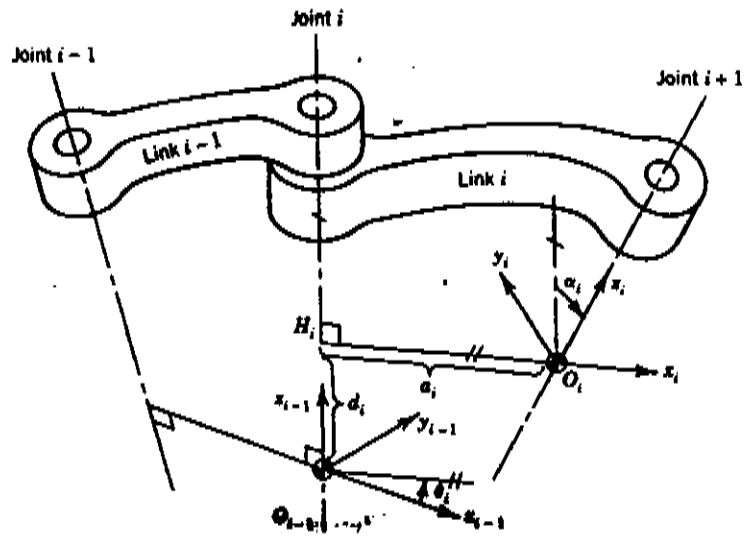

Figure 2: Denavit-Hartenberg Notation

joint axis defines a unique axis of rotation. We thus focus our attention on how to describe the relative position and orientation of two lines in space (two sequential joint axes). One property which relates two lines in space is the minimum distance between them. The minimum distance is a unique, unambiguous number (which may be zero, but never negative). We assign this unique number to the parameter name "a_i" for link number i (which connects joint axes i and $i+1$). Parameter a_i is labelled in Fig 3.

Another property relating two lines is the common perpendicular. There are 3 cases we must consider. First, consider two lines in space which are not parallel and which do not intersect. The minimum distance between such lines is identifiable as the distance between a unique point on the first line and a unique point on the second line. We may construct a third line which contains these two points. This third line will, necessarily, have the property that it is perpendicular to both of the original two lines. We therefore call this line the *common normal*. We will use the common normal to help uniquely define a body-fixed coordinate frame associated with link i.

Having defined the common normal line, we can define a vector direction along this line, and define it to be the x axis of the i'th link coordinate frame, $\mathbf{x}_i$. We define the direction of the $\mathbf{x}_i$ vector as pointing away from joint i towards joint $i+1$.

We may also define a z axis for link i. By convention, $\mathbf{z}_i$ is defined to be colinear with joint axis $i+1$. The difference in numbering (axis i along joint $i+1$) is a confusing nuisance. Nonetheless, that is the convention. We are free to choose either of the possible directions of $\mathbf{z}_i$ which satisfy colinearity with joint axis $i+1$. This choice will influence what we will later define as positive and negative joint rotations.

Note that our chosen $\mathbf{x}_i$ and $\mathbf{z}_i$ intersect and are orthogonal, as must be the case with any valid coordinate-frame construction. This is hardly accidental. Our definition or $\mathbf{x}_i$ followed from the definition of the common normal, which must always intersect axes i and $i+1$ at right angles. Having defined $\mathbf{x}_i$ and $\mathbf{z}_i$, we have also defined the origin of our i'th coordinate frame, O_i (located at the intersection of $\mathbf{x}_i$ and $\mathbf{z}_i$). Note that this origin lies on the *distal* joint axis (joint $i+1$) of link i. Also, having defined $\mathbf{x}_i$ and $\mathbf{z}_i$, the third axis, y_i, follows uniquely as the third vector of a right-hand triad. These three vectors define a coordinate frame for link i, as illustrated in Fig 3. Thus we see that by merely considering two arbitrary lines in space, we can define a corresponding unique coordinate system. (For intersecting and for parallel lines, the preceding definitions have some additional considerations, as discussed further below).

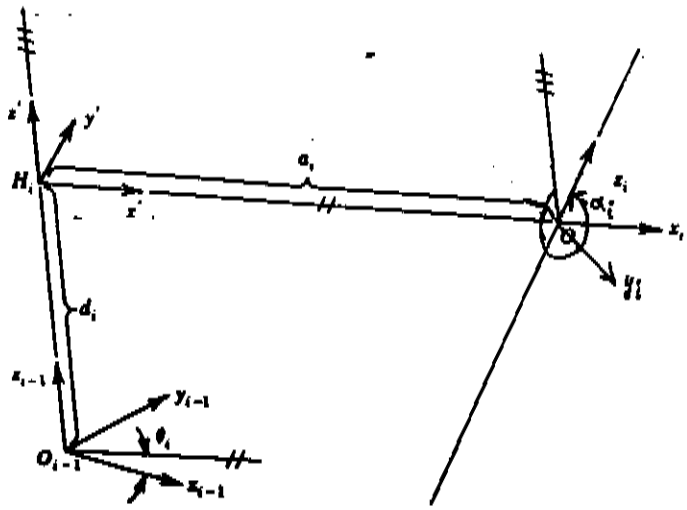

Figure 3: successive coordinate frames

We have used the common normal to define the link parameter a_i. We can invoke this unique common normal again to define a link twist, α_i. Understanding α_i takes some careful visualization, which is introduced below by construction.

Imagine that we start with three lines in the shape of an "H"; a common normal (the cross-bar of the H) joins two parallel lines (the uprights of the H), and all three lines lie in a plane. The distance between the "left" leg of the H (joint axis i) and the "right" leg of the H (joint axis $i+1$) is a_i. We define $\mathbf{z}_{i-1}$ as pointing "up" along axis i, and $\mathbf{z}_i$ (lying along axis $i+1$) also pointing "up" (parallel to $\mathbf{z}_{i-1}$). The cross-bar of the H defines the $\mathbf{x}_i$ axis, pointing in the direction from $\mathbf{z}_{i-1}$ towards $\mathbf{z}_i$. We define this state ($\mathbf{z}_{i-1}$ parallel to $\mathbf{z}_i$) as having a twist of $\alpha_i = 0$.

Now, we may imagine "twisting" z_i (the right leg of the H) such that it no longer lies in the original plane of the H. In so doing, we maintain $\mathbf{z}_i$ perpendicular to the common normal. That is, we rotate $\mathbf{z}_i$ about $\mathbf{x}_i$. The magnitude of rotation of $\mathbf{z}_i$ about $\mathbf{x}_i$ is α_i. We define plus and minus α_i according to our definition of signed rotation. In our example, the left leg of the H is joint axis i (colinear with $\mathbf{z}_{i-1}$), the right leg is joint axis $i+i$ (colinear with $\mathbf{z}_i$), and the $\mathbf{x}_i$ axis points from left to right. A positive rotation of $\mathbf{z}_i$ thus corresponds to the top of the right leg of the H (the arrowhead of vector $\mathbf{z}_i$) moving out of the page, and the bottom of the right leg of the H moving into the page. The angle α is indicated in Fig 3

Having defined the rotation α, we now state a crucial observation. We may define the relative position and orientation of *any* two lines by specifying two parameters: a (the length of the common normal) and α (the relative twist angle about the common normal). It may seem counterintuitive that a single angle α is sufficient to describe the relative orientation of lines in 3-D. To see that this is possible, consider the following. We first define our $\mathbf{z}_{i-1}$, $\mathbf{z}_i$ and $\mathbf{x}_i$ axes according to the preceding definitions. Next, construct an additional vector $\mathbf{z}''$ with the following properties. Vector $\mathbf{z}''$ must be perpendicular to $\mathbf{x}_i$, parallel to $\mathbf{z}_{i-1}$, and must pass through the origin of coordinate frame i (the intersection of $\mathbf{x}_i$ and $\mathbf{z}_i$). This new vector $\mathbf{z}''$ forms a planar "H" with $\mathbf{z}_{i-1}$ and x_i. If we rotate vector $\mathbf{z}''$ about the axis $\mathbf{x}_i$, it is possible to make it colinear with $\mathbf{z}_i$. The twist angle which produces such colinear alignment defines the twist between $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$, which defines the value of α_i.

It may not be obvious that $\mathbf{z}''$ can be made colinear with vector $\mathbf{z}_i$ by invoking a pure rotation α_i about $\mathbf{x}_i$. To see this, recall that both $\mathbf{z}''$ and $\mathbf{z}_i$ are perpendicular to $\mathbf{x}_i$, and that $\mathbf{z}''$ and $\mathbf{z}_i$ intersect. Thus, $\mathbf{z}''$

and $\mathbf{z}_i$ both lie in the same plane perpendicular to $\mathbf{x}_i$. By rotating $\mathbf{z}''$ about $\mathbf{x}_i$ by angle α, it is possible to make $\mathbf{z}''$ line up with *any* line passing through O_i and lying in the plane perpendicular to $\mathbf{x}_i$. It is therefore necessarily true that there does exist some twist angle α_i of $\mathbf{z}''$ about $\mathbf{x}_i$ which makes $\mathbf{z}''$ colinear with $\mathbf{z}_i$. We thus conclude that we can uniquely describe the relative position and orientation of two lines in space with two parameters: a_i and α_i.

We have restricted consideration so far to non-intersecting and non-parallel lines. In the case of parallel lines, a_i and α_i are both well defined. The distance between the two lines is a_i. The twist is either 0 or π, depending on whether the corresponding $\mathbf{z}$ axes are defined as parallel or anti-parallel, respectively. There is an ambiguity regarding the $\mathbf{x}_i$ axis, though. For parallel lines, the common normal is not unique. Rather, like the rungs of a ladder, there are an infinite number of choices for $\mathbf{x}_i$ which are all valid common normals. In this case, the modeler has the option of choosing the location of O_i anywhere along $\mathbf{z}_i$.

The final special case is when two lines intersect. In this case, the minimum distance is $a_i = 0$. We can still define an $\mathbf{x}_i$ axis perpendicular to both $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$. Since $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$ intersect, they are coplanar. The vector $\mathbf{x}_i$ is perendicular to this plane, and passes through origin O_i, which is located at the intersection of $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$. There is a modeler's choice, though, for the direction of $\mathbf{x}_i$ ("up" or "down" with respect to the plane of $\mathbf{z}_{i-1}$ and $\mathbf{z}_i$). Once the directions of $\mathbf{z}_{i-1}$, $\mathbf{z}_i$ and $\mathbf{x}_i$ are chosen, α_i follows as a special (simpler) case of the preceding definition of α_i. The angle of rotation from $\mathbf{z}_{i-1}$ to $\mathbf{z}_i$ about $\mathbf{x}_i$ is α_i.

We thus conclude that we can specify precisely how to construct a unique coordinate system (with some freedom in special cases), $\mathbf{x}_i$, $\mathbf{y}_i$, $\mathbf{z}_i$ at origin O_i associated with an arbitrary link i bounded by joints i and $i+1$. For rigid links, the parameters a_i and α_i fully describe the relationship between successive joints. Both of these parameters are static quantities. They are properties of the joint design, and their values never change during robot motion.

We next consider the relationship between successive links sharing a common joint. This consideration produces two more D-H parameters, including one which describes the variable rotation of a robot joint.

VIII Joint i D-H parameters θ_i and d_i

We now introduce the last two Denavit-Hartenberg parameters, joint angle θ_i and distance d_i. With respect to Fig 2, we note that origin O_i is defined as the intersection of x_i with $\mathbf{z}_i$. Similarly, we may define the origin O_{i-1} as the intersection of $\mathbf{z}_{i-1}$ with $\mathbf{x}_{i-1}$ (as illustrated). It is also useful to define the point H_i at the intersection of $\mathbf{x}_i$ and $\mathbf{z}_{i-1}$. Both O_{i-1} and H_i lie on the $\mathbf{z}_{i-1}$ axis. The distance from O_{i-1} to H_i along $\mathbf{z}_{i-1}$ is defined as the D-H parameter d_i. Note that although a_i must always be nonnegative, the parameter d_i is measured along $\mathbf{z}_{i-1}$ as a signed distance, as defined by the chosen direction of vector $\mathbf{z}_{i-1}$. In Fig 2, the parameter d_i is positive for this example.

Our last parameter, θ_i, is the only D-H parameter which (for revolute joints) changes its value as the robot moves. θ_i is the measure of rotation of joint motor i. As we control motor i, link i rotates about joint i (vector $\mathbf{z}_{i-1}$). As motor i rotates, i.e. as θ_i changes, frame i (attached to link i) changes its position and orientation in space. (For prismatic joints, the parameter d_i changes its length as joint i displaces H_i relative to O_{i-1}). We may define a "home" angle of $\theta_i = 0$ for the special pose of $\mathbf{x}_{i-1}$ parallel to $\mathbf{x}_i$. As with the definition of α_i, it may not be obvious that we can guarantee that an angle θ_i exists such that $\mathbf{x}_i$ is parallel to $\mathbf{x}_{i-1}$. Recall, though, that $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$ are both perpendicular to joint $\mathbf{z}_{i-1}$. For vizualization, we may define a vector $\mathbf{x}''$ parallel to $\mathbf{x}_{i-1}$ passing through point H_i. Since $\mathbf{x}''$ intersects H_i, $\mathbf{x}''$ also intersects $\mathbf{x}_i$. Together, $\mathbf{x}''$ and $\mathbf{x}_i$ define a plane perpendicular to $\mathbf{z}_{i-1}$. As link i rotates about axis $\mathbf{z}_{i-1}$ (due to motion of motor i), the vector $\mathbf{x}_i$ remains in the same plane. Since $\mathbf{x}_i$ can achieve any rotation within this plane, there is some rotation which corresponds to $\mathbf{x}_i$ lying coincident (colinear) with $\mathbf{x}''$. We may define this unique pose to correspond to $\theta_i = 0$.

Having defined the "home" angle of θ_i, we can define all other rotations of link i as the angle from $\mathbf{x}''$ to $\mathbf{x}_i$, as measured in a signed sense with respect to the rotation axis $\mathbf{z}_{i-1}$. We have now established how to define

a unique coordinate frame on each link of a robot, and we have defined a compact set of four parameters, a_i, α_i, d_i, and θ_i, which relate to the two successive frames. We will describe how to convert D-H parameters into an equivalent homogeneous transformation A matrix shortly. First, we conclude our modeling with consideration of two important special cases: link 0 and link n.

IX Special cases: joints 0 and n

The preceding discussion describes how to model an arbitrary link connecting two joints, and an arbitrary joint connecting two links. The first and the last links, however, are connected to only one joint each. It is possible to adapt our definitions of D-H parameters, though, to include these links in a consistent manner.

For the case of link 0, we choose to define coordinate frame 0 with $\mathbf{z}_o$ colinear with the first joint axis. Again, the choice of vector direction ("up" or "down") is the modelers choice; this choice defines positive rotation for θ_1. For the base frame (frame 0), we are free to choose origin O_o anywhere along $\mathbf{z}_o$. Having chosen O_o, we still have some freedom in defining $\mathbf{x}_o$. We must choose $\mathbf{x}_o$ to pass through origin O_o, and $\mathbf{x}_o$ must be chosen perpendicular to $\mathbf{z}_o$. These constraints still permit us a continuum of choices for $\mathbf{x}_o$ lying in a plane perpendicular to $\mathbf{z}_o$. The (arbitrary) choice of a particular eligible $\mathbf{x}_o$ defines the home angle for θ_1. Having chosen $\mathbf{z}_o$ and $\mathbf{x}_o$, vector $\mathbf{y}_o$ follows by definition of a right-hand coordinate system. With these definitions, we have established a stationary frame (frame 0) which is not affected by any joint motions. We can construct the D-H parameters for link 1, as described above, based on these definitions for frame 0.

We have similar modeling freedom for the final link, link n. Link n is bounded by joint n on the proximal side and by an end-effector on its distal end. It is usually convenient to define a final coordinate system (frame n) with its origin at a useful location on the gripper or tool. For example, O_n is chosen to lie between the fingers of the gripper in Fig 1. To make such a definition consistent with D-H parameter definitions (to treat link n like any other link), we must require that $\mathbf{x}_n$ pass through O_n, and that $\mathbf{x}_n$ be perpendicular to $\mathbf{z}_{n-1}$. This is always possible to do, even if O_n happens to lie on $\mathbf{z}_{n-1}$. After this choice, we now have some freedom in selecting $\mathbf{z}_n$. Vector $\mathbf{z}_n$ must pass through origin n, O_n, and it must be perpendicular to axis $\mathbf{x}_n$. These restrictions confine the options for $\mathbf{z}_n$ to a plane perpendicular to $\mathbf{x}_n$. The particular choice of $\mathbf{z}_n$ defines the "twist" of the last link, α_n.

This completes our modeling rules for open kinematic chains.

X Homogeneous Transforms and D-H Parameters

We have now introduced two significant bodies of definitions and derivations: coordinate transformations and kinematic modelling. We can combine these two topics to obtain a general solution for robot forward kinematics. The purpose to this section is to derive the result that A_i^{i-1}, the transformation from the link-i coordinate frame to the link-$i-1$ coordinate frame, can be expressed in terms of Denavit-Hartenberg parameters. It will be shown that the solution is:

$$A_i^{i-1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{26}$$

To prove the above result, it is convenient to define a temporary intermediate frame, frame *int*. This intermediate frame is is labelled in Fig 3 with axes $\mathbf{x}'$, $\mathbf{y}'$ and $\mathbf{z}'$ intersecting at point H_i. The $\mathbf{x}'$ axis is colinear with the $\mathbf{x}_i$ axis and the $\mathbf{z}'$ axis is colinear with the $\mathbf{z}_{i-1}$ axis. The vector from the origin of the intermediate frame to O_i (the origin of frame i), expressed with respect to the intermediate coordinate frame,

is

$$\mathbf{p}_{i/int} = \begin{bmatrix} a_i \\ 0 \\ 0 \end{bmatrix} \tag{27}$$

The orientation of frame i with respect to the intermediate frame can be expressed in terms of the tip coordinates of the $\mathbf{x}_i$, $\mathbf{y}_i$ and $\mathbf{z}_i$ vectors, as measured with respect to the intermediate frame. The $\mathbf{x}_i$ axis is coincident with the $\mathbf{x}'$ axis, but the $\mathbf{z}_i$ and $\mathbf{y}_i$ axes are rotated by angle α_i with respect to the intermediate frame about axis $\mathbf{x}'$. Thus, we can write the relative orientation matrix as $R_x(\alpha)$, or

$$R_{i/int} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix} \tag{28}$$

Assembling the 4x4 homogeneous transform matrix from the elements R and $\mathbf{p}$, we obtain the transformation from frame i to the intermediate frame:

$$A_i^{int} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{29}$$

Next, we define the transformation which relates the intermediate frame to the $i-1$ frame. In this case, the vector $\mathbf{z}'$ is colinear with vector $\mathbf{z}_{i-1}$. With respect to frame $i-1$, the vector from O_{i-1} (the origin of the $i-1$ frame) to H_i (the origin of the intermediate frame) is:

$$\mathbf{p}_{int/i-1} = \begin{bmatrix} 0 \\ 0 \\ d_i \end{bmatrix} \tag{30}$$

The orientation of the intermediate frame with respect to the $i-1$ frame is a pure z rotation of magnitude $R_z(\theta_i)$, or:

$$R_{int/i-1} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{31}$$

We can now assemble the A-matrix 4x4 transformation from the intermediate frame to the $i-1$ frame, using the R and p components:

$$A_i^{int} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{32}$$

Finally, we obtain the transformation from frame i to frame $i-1$ by cascading the transformation from i to the intermediate frame, and from the intermediate frame to frame $i-1$:

$$A_i^{i-1} = A_{int}^{i-1} A_i^{int} \tag{33}$$

Substituting Eqn 29 and Eqn 32 into Eqn 33 and performing the matrix multiplication, we obtain our desired result of equation 26.

XI Forward Kinematics

We now have all the tools we need to compute forward kinematics for an arbitrary robot arm.

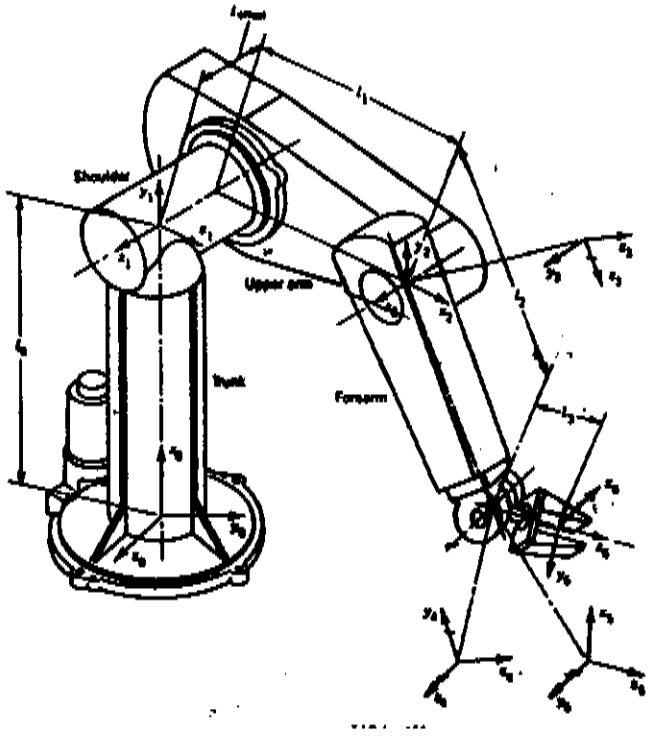

Figure 4: PUMA 600 robot

As an example, consider the arm of Fig 4. This robot is a popular 6 degree-of-freedom (six-jointed) robot with all revolute joints. Joint axis 1 is perpendicular to joint axis 2, and they intersect. Joint axis 3 is parallel to joint axis 2. Joint axis 4 (which is a forearm rotation) is perpendicular to joint axis 3, and these axes intersect. Joint axis 5 (a wrist bend) is perpendicular to joint axis 4, and axes 4 and 5 intersect. Joint axis 6 is a final wrist twist. This axis is perpendicular to axis 5, and axes 4, 5 and 6 intersect in a point (which is a common, desirable configuration referred to as a "spherical wrist").

The D-H parameters corresponding to this robot, with the optional frame choices illustrated, can be expressed in tabular form as:

L	α	a	d	θ	notes
1	$\pi/2$	0	l_0	θ_1	choice of O_1 yields nonzero d_1
2	0	l_1	$-l_{offset}$	θ_2	parallel joints; d_2 depends on chosen O_2
3	$\pi/2$	0	0	θ_3	O_3 coincident with O_2
4	$-\pi/2$	0	l_2	θ_4	axes intersect; chosen x_4 yields $\alpha_4 < 0$
5	$\pi/2$	0	0	θ_5	O_5 coincident with O_4
6	0	0	l_3	θ_6	z_6 colinear with z_5

From the above table, we can read off all of the D-H parameters from link 1 to link 6. For each link, we can define A_i^{i-1}, as per Eqn 26. Note that the fifth column of our D-H table contains only variable names (θ_i). All other columns contain constants. The angle values θ_i are the only variables in our system. These angles change as the robot's joints are driven. We cannot complete our transformation calculations until these joint variables have been measured.

At any instant in time, the robot's control computer can read the sensors on each of the robot's joints to determine each θ_i. With such numerical data, each transformation matrix A_i^{i-1} can be computed numerically. The product of these matrices describes the position and orientation of the robot's n'th frame (the hand frame) with respect to a ground frame (the 0 frame). To compute the hand frame, we measure all θ_i, then invoke equation 26 for each link, then multiply the resulting matrices as:

$$A_6^0 = A_1^0 A_2^1 A_3^2 A_4^3 A_5^4 A_6^5 \tag{34}$$

The matrix A_6^0 completely specifies the position and orientation of the hand frame. This is the result that we desired for forward kinematics.

Computation of forward kinematics for our PUMA 600 example is straightforward, if tedious. Evaluation of Eqn 34 requires evaluation of 6 sines and 6 cosines, 92 floating-point multiplies, and 60 additions. This is easily performed by unexceptional computers within less than 1 millisecond, though a computation by hand is unenjoyable and error prone. The forward kinematic computation can be made more efficient by recognizing that (for the PUMA 600), at least 7 of the 16 terms in each A_i^{i-1} matrix are zero, and one term is unity. If the code (or human) exploits this fact, the number of multiplications and additions is halved. "Sparse" A-matrices (matrices with lots of zero terms) are common for robots which have twists (α_i) of zero or $\pi/2$. Such designs are the most common, as they are easiest to construct and easiest to analyze. Note from Eqn 26 that α_i enters only as the argument of sines and cosines. For α's of 0 or $\pi/2$, half of these terms are zero.

While our 6-dof example is the most common case among industrial robots, it is too tedious for illustrative purposes. To get a sense of how to use and interpret equation 26, we now consider the simplified, two-link planar arm of Fig 5. In this figure, the $\mathbf{z}_0$, $\mathbf{z}_1$, and $\mathbf{z}_2$ axes are always perpendicular to (out of) the page.

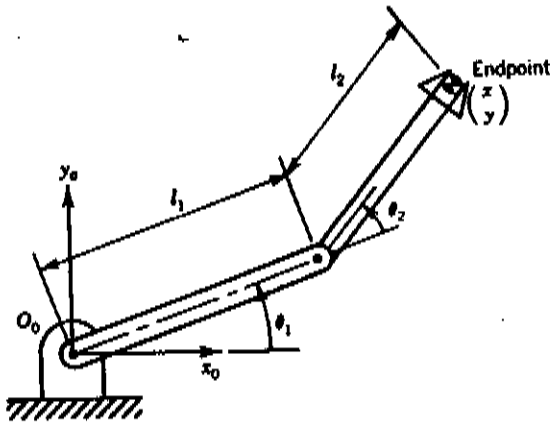

Figure 5: Two-link planar arm

We may define the origin of our 2-frame, O_2, to be located between the gripper fingers. This choice of O_2 and $\mathbf{z}_2$ defines $\mathbf{x}_2$ and $\mathbf{y}_2$.

The D-H parameters for this simple arm are:

L	α	a	d	θ	notes
1	0	l_1	0	θ_1	$\mathbf{z}_0$ parallel to $\mathbf{z}_1$ yields $\alpha_1 = 0$
2	0	l_2	0	θ_2	$\mathbf{z}_2$ chosen parallel to $\mathbf{z}_1$ yields $\alpha_2 = 0$

Our corresponding A matrices (from Eqn 26) are:

$$A_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & l_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & l_1\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{35}$$

and

$$A_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & l_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & l_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{36}$$

Our resulting hand frame is the product of A_2^1 and A_1^0, as expressed in equations 36 and 35, respectively. For compactness, we use the notation C_i to mean $\cos(\theta_i)$ and S_i to mean $\sin(\theta_i)$ to obtain:

$$\begin{aligned} A_2^0 &= A_1^0 A_2^1 \\ &= \begin{bmatrix} C_1C_2 - S_1S_2 & -C_1S_2 - S_1C_2 & 0 & l_2(C_1C_2 - S_1S_2) + l_1C_1 \\ S_1C_2 + C_1S_2 & -S_1S_2 + C_1C_2 & 0 & l_2(S_1C_2 + C_1S_2) + l_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{37}$$

We may simplify Eqn 37 further by invoking double-angle trigonometric identities and defining C_{12} and S_{12} as:

$$S_{12} = \sin(\theta_1 + \theta_2) = \sin(\theta_1)\cos(\theta_2) + \cos(\theta_1)\sin(\theta_2) \tag{38}$$

and

$$C_{12} = \cos(\theta_1 + \theta_2) = \cos(\theta_1)\cos(\theta_2) - \sin(\theta_1)\sin(\theta_2) \tag{39}$$

from which we can simplify Eqn 37 to:

$$A_2^0 = \begin{bmatrix} C_{12} & -S_{12} & 0 & l_2C_{12} + l_1C_1 \\ S_{12} & C_{12} & 0 & l_2S_{12} + l_1S_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

Equation 40 is easily interpreted. We see from the fourth column that the position of the hand is:

$$\mathbf{p}_{2/0} = \begin{bmatrix} l_2C_{12} + l_1C_1 \\ l_2S_{12} + l_1S_1 \\ 0 \end{bmatrix} \tag{41}$$

We see from the above that the z coordinate of the hand frame is always 0. This makes sense, since the planar arm cannot lift its gripper out of the plane of the page. The x coordinate of the gripper is $x = l_2C_{12} + l_1C_1$. This term is comprised of two contributions. The term l_1C_1 is the projection of the elbow position (origin O_1) onto the $\mathbf{x}_0$ axis. The term l_2C_{12} is the horizontal (x) contribution of link two, from the elbow (0_1) to the hand (O_2), which is easily visualized from Fig 5. Together, these two terms express the perpendicular projection of the hand position onto the $\mathbf{x}_0$ axis. The y coordinate of the gripper is $l_2S_{12} + l_1S_1$. These two terms are the analogous contributions of link 2 and link 1, respectively, to the y-position of the gripper projected onto the $\mathbf{y}_0$ axis.

The upper-left 3x3 matrix of A_2^0 defines the orientation of the gripper. We see that $\mathbf{b}_{2/0}$ is always parallel to the $\mathbf{z}_0$ axis. This makes sense, as no motion of joints 1 or 2 can cause $\mathbf{z}_2$ to rotation away from perpendicular to the plane. The axes $\mathbf{x}_2$ and $\mathbf{y}_2$ correspond to a rotation about the $\mathbf{z}_0$ axis by an angle of $\theta_1 + \theta_2$. Both the link 1 and link 2 joint motions contribute to a net z rotation of the hand frame.

XII Conclusion

We have introduced the fundamental mathematics required to inform a robot where a task is located, and how to compute where the end-effector of a robot is located. Defining positions and orientations of objects in space requires specification of 6 parameters: three positions and three angles. We can express orientation in

terms of a convenient 3x3 rotation matrix of 9 components. Although this matrix contains more terms than necessary for specifying orientation, it is nonetheless a useful representation. R matrices not only express orientation, but can also be treated as operators and can be utilized in coordinate transformations.

We found it convenient to construct A matrices from R matrices and vectors. By inserting our information into a 4x4 matrix, our notation is simplified considerably.

In modeling robots, we defined procedures which permit kinematic specification of an arm with only four scalar parameters per link, and unambiguous definition of body-fixed coordinate frames for each link. We further showed that such a specification can be readily introduced into our previously defined A matrices, enabling coordinate transformations between successive joint frames. Cascading successive coordinate transformations, we obtain a single 4x4 transformation relating the base-frame of the robot to the coordinate frame attached to its n'th link. Such an expression can be interpreted immediately to obtain the coordinates and the orientation of the robot's hand frame. Such computation constitutes forward kinematics.

With the collection of mathematics presented here, we can specify a goal pose for a robot precisely, and we can rapidly compute the position and orientation of a robot's hand or tool. With this set of formulae and definitions, we are prepared to compose algorithms which perform computer control of a robot's joints.