Three Dimensional SCARA Control Planning

Miles Higman

March 2024

1 Introduction

Accurate and precise motion along a path is necessary for many robotics applications, such as manufacturing and 3D printing. This paper extends the work done in *Motion Planning of SCARA Robots for Trajectory Tracking* by Giovanni Incerti [2] into three dimensions with a SCARA robot that has an eccentric kinematic profile.

2 The Arm

The arm we will examine in this paper is similar to a conventional SCARA arm but on its side, with a motor rotating the assembly in the xy plane.

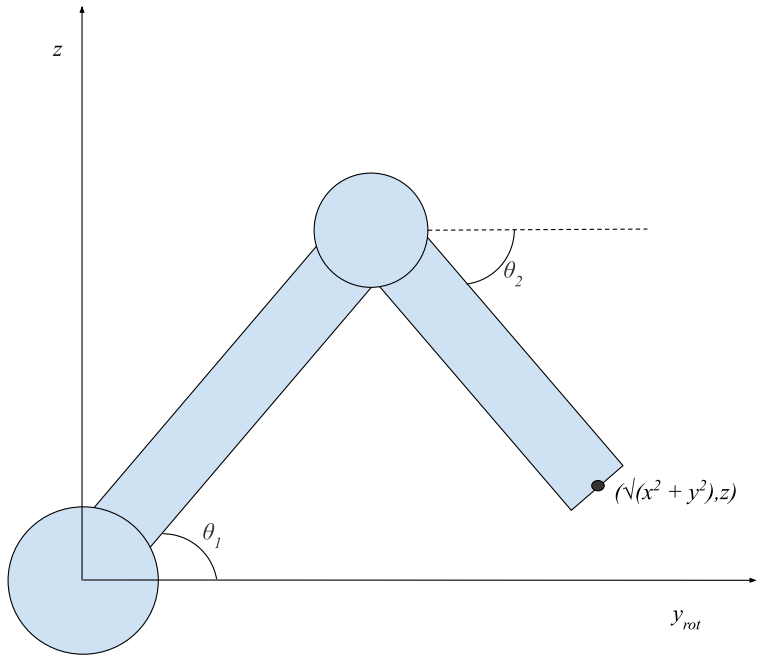

Figure 1: Side View of the robot

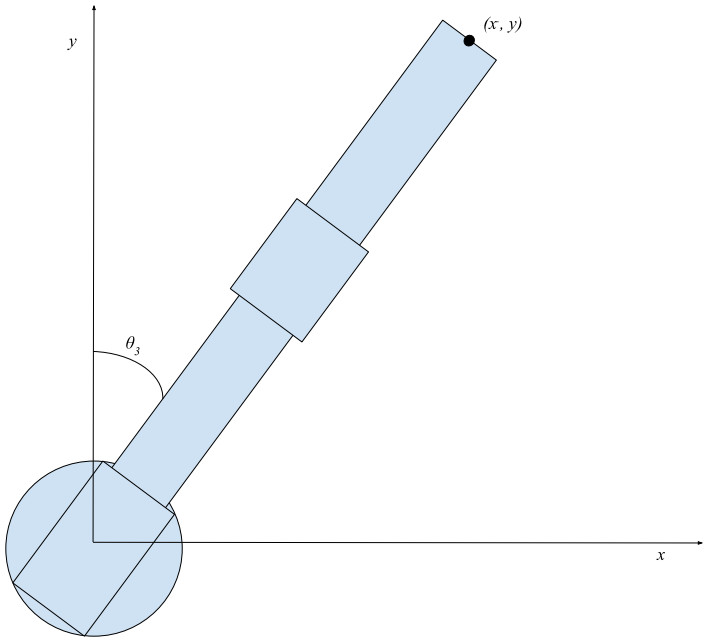

Figure 2: Top View of the robot

As shown above, the end effector is given by (x, y, z). The coordinates of the end effector depend on θ_1, θ_2, and θ_3, where...

• θ_1 is the angle between the first part (L_1) of the arm and the xy plane.

• θ_2 is the angle between a plane parallel to the xy plane and the second part of the arm (L_2).

• θ_3 is the angle between the y-axis and the plane the robot arm inhabits (basically the heading of the arm).

3 Inverse Kinematics

Since the robot arm we are using has a side profile similar to a conventional SCARA arm, the law of cosines and Carnot's theorem apply, providing the equations:

$$\begin{cases} \theta_{2a} = \pm \arccos\left(\dfrac{x^2 + y^2 + z^2 - L_1^2 - L_2^2}{2L_1L_2}\right) \\ \theta_1 = \arctan 2\left(z, \sqrt{x^2 + y^2}\right) - \arctan 2\left(L_2 \sin(\theta_{2a}), L_1 + L_2 \cos(\theta_{2a})\right) \end{cases} \tag{1}$$

However, it is essential to note that θ_{2a} is measured as the angle between the direction θ_1 and the direction of L_2. To keep our relationships between $(\theta_1, \theta_2, \theta_3)$ and (x, y, z) simpler, we will define θ_2 as the angle between a line parallel to the xy plane and the robot arm, meaning that:

$$\theta_2 = \theta_{2a} + \theta_1 \tag{2}$$

θ_3 is pretty intuitive as, viewing the arm through the z axis, it is easy to see that θ_3 is the heading of the vector the robot arm makes in the xy plane. So:

$$\theta_3 = \arctan 2\,(x, y) \tag{3}$$

4 Motion and the Jacobian

Now that position can be found, it is crucial to characterize other parts of the motion. To start, the direct kinematics of the arm are as follows:

$$\begin{cases} x = \sin(\theta_3)\,(L_2\cos(\theta_2) + L_1\cos(\theta_1)) = f(\theta_1, \theta_2, \theta_3) \\ y = \cos(\theta_3)\,(L_2\cos(\theta_2) + L_1\cos(\theta_1)) = g(\theta_1, \theta_2, \theta_3) \\ z = L_1\sin(\theta_1) + L_2\sin(\theta_2) = h(\theta_1, \theta_2) \end{cases} \tag{4}$$

Velocity and acceleration can be computed from these equations. [2]

Differentiating these three equations, we find that:

$$\begin{cases} \dot{x} = \dfrac{\partial f}{\partial \theta_1}\dot{\theta_1} + \dfrac{\partial f}{\partial \theta_2}\dot{\theta_2} + \dfrac{\partial f}{\partial \theta_3}\dot{\theta_3} \\ \dot{y} = \dfrac{\partial g}{\partial \theta_1}\dot{\theta_1} + \dfrac{\partial g}{\partial \theta_2}\dot{\theta_2} + \dfrac{\partial g}{\partial \theta_3}\dot{\theta_3} \\ \dot{z} = \dfrac{\partial h}{\partial \theta_1}\dot{\theta_1} + \dfrac{\partial h}{\partial \theta_2}\dot{\theta_2} + \dfrac{\partial h}{\partial \theta_3}\dot{\theta_3} \end{cases} \tag{5}$$

Now, if we define:

$$\dot{\vec{q}} = \begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \\ \dot{\theta_3} \end{bmatrix} \qquad \dot{\vec{p}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{6}$$

Then, we can define the Jacobian, $\mathbf{J}$, as follows:

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f}{\partial \theta_1} & \dfrac{\partial f}{\partial \theta_2} & \dfrac{\partial f}{\partial \theta_3} \\ \dfrac{\partial g}{\partial \theta_1} & \dfrac{\partial g}{\partial \theta_2} & \dfrac{\partial g}{\partial \theta_3} \\ \dfrac{\partial h}{\partial \theta_1} & \dfrac{\partial h}{\partial \theta_2} & \dfrac{\partial h}{\partial \theta_3} \end{bmatrix} \tag{7}$$

Finally, with some simple linear algebra and calculus, we can state the following:

$$\dot{\vec{p}} = \mathbf{J}\dot{\vec{q}} \qquad \dot{\vec{q}} = \mathbf{J^{-1}}\dot{\vec{p}} \tag{8}$$

$$\ddot{\vec{p}} = \mathbf{J}\ddot{\vec{q}} + \dot{\mathbf{J}}\dot{\vec{q}} \qquad \ddot{\vec{q}} = \mathbf{J^{-1}}(\ddot{\vec{p}} - \dot{\mathbf{J}}\dot{\vec{q}}) \tag{9}$$

5 Defining a 3D Path

When talking about precise motion along a path, it is vital to qualify the path, which we will require to be C_2 continuous across the entire path for accuracy reasons. Here are multiple paths that can be used effectively:

5.1 Vector Form

The first and most intuitive path definition we can use would be a simple function in 3D space, which would be some type of parametric curve with initial conditions being vectors, like so:

$$\vec{s}(t) = \langle x(t), y(t), z(t) \rangle \tag{10}$$

To use an acceleration profile, it is essential to know how to find the arc length. Assuming the known and desired points are known, t is constrained to $t_{start} \leq t \leq t_{end}$, with t_{start} and t_{end} easily be solved with some type of system of equations. With a parametric curve, finding arc length is relatively straightforward:

$$\int_{t_{start}}^{t_{end}} ||\dot{\vec{s}}(t)|| dt \tag{11}$$

Also, if t_{end} is defined as $t_{current}$, it is easy to find the total distance traveled at a current time.

5.2 Polar Form

Another way to define a path would be to use a 3d polar equation like so:

$$\vec{s}(t) = \langle \rho(t), \phi(t), \theta(t) \rangle \tag{12}$$

With the vector showing the radius and both angles defining the direction with respect to time.

It is crucial to clarify how to find the total distance along a polar curve. This process is made easy by knowing these equalities:

$$\begin{cases} x = \rho \cos\phi \cos\theta \\ y = \rho \sin\phi \cos\theta \\ z = \rho \sin\phi \end{cases} \tag{13}$$

This equation [1] gives the arc length of the polar curve to be used in the robot motion.

$$\int_{t_{start}}^{t_{end}} \sqrt{\dot{\rho}^2 + \rho^2 \cos^2(\theta)\dot{\phi}^2 + \rho^2\dot{\theta}^2} dt \tag{14}$$

5.3 Extrapolated Bézier

Another way to define a C_2 continuous path would be to use a cubic (or higher order) Bézier and extrapolate past $t = 1$. However, this limits some local control of the path, with finding the perfect, probably higher-order path being computationally intensive.

Finding the arc length of a cubic or higher order Bézier is not impossible, but there is no reasonably concise representation. The best way to get the arc length is to compute it using a simple approximation of the integral (with Δt being the step size of the approximation) like this:

$$\int_0^{t_{end}} ||\dot{\vec{B}}(t)|| dt \approx \sum_{i=0}^{n} ||\dot{\vec{B}}(i\Delta t)|| \Delta t \text{ where } n = \lfloor \frac{t_{end}}{\Delta t} \rfloor \tag{15}$$

5.4 Splines

Finally, the most controllable path definition would be using splines. Splines can be made from all the functions listed above. However, there are pros and cons to each type, which I will mention here:

• A C_2 continuous B spline made from multiple cubic vector equations is probably the best general solution as local control is maintained along with continuity. However, some preprocessing requirements exist to find the control points that allow the spline to pass through all desired points.

• Another method to make a spline would be to use a B spline containing a series of polar equations. The result would also be C_2 continuous as the polar vectors (and, by extension, the rectangular vectors) between intersections would be equal. This method would be a better alternative to the first method if rotational velocity/acceleration or direction needed to be easily accessible throughout the motion.

• Finally, making a C_2 continuous path with a Bézier spline is also possible. However, this isn't recommended, as local control is lost around intersection points.

Arc length for a spline is relatively straightforward. The arc length of a spline is the sum of the arc length for each composite function. For example, the arc length would be if we have n number of composite vector functions, denoted as $C_n(t)$, starting at t_{n-1}, and ending at t_n.

$$\sum_{i=0}^{n} \int_{t_{i-1}}^{t_i} ||\dot{\vec{C}}_i(t)||dt \tag{16}$$

6 Integration with an Acceleration Profile

Now that a path is defined, discussing its integration with an acceleration profile is essential.

In the example presented in this paper, I am using a modified trapezoidal profile that satisfies the condition

$$\int_0^{t_{end1}} \int a(t)dt = L \tag{17}$$

where L is the arc length of the given path.

For our example, let's consider a path that is C_2 continuous depending on a parameter t. The path starts at t_{start2} and ends at t_{end2}. It's essential to consider that we need a time t such that the double integral of the profile is equal to the current arc length. To ensure that we get the correct time to substitute into the acceleration profile, we can set up a proportion like this:

$$\frac{t_{c1}}{L_{part}} = \frac{t_{end1}}{L_{whole}} \quad => \quad t_{c1} = \frac{t_{end1} L_{part}}{L_{whole}} \tag{18}$$

where t_{c1} is the current time in the acceleration profile. t_{end1} and L_{whole} are already known, but finding L_{part} needs some clarification. L_{part} is simply the arc length of the path at $t = t_{c2}$, where t_{c2} is the parameter that is put into the path at a given time. The method to calculate this depends on the type of path being used, but it usually follows a certain form:

$$\int_{t_{start}}^{t_{c2}} ||\dot{\vec{S}}(t)||dt \text{ where } S(t) \text{ is a vector function defining the path.} \tag{19}$$

Now that all the variables to find t_{c1} are clearly defined, substituting t_{c1} into $a(t)$, $v(t)$, and $s(t)$ yields the magnitude of the acceleration, velocity, and position vector respectively.

To get the correct position, velocity, and acceleration vectors, simply scale the vector by the desired magnitude divided by the magnitude of the vector. To clarify, given a vector function $S_i = \langle x(t), y(t), z(t) \rangle$, the resultant vectors would be:

$$\begin{cases} \dot{\vec{S}}_r(t_{c2}) = \dfrac{v(t_{c1})}{||\dot{\vec{S}}_i(t_{c2})||} \dot{\vec{S}}_i(t_{c2}) \\ \ddot{\vec{S}}_r(t_{c2}) = \dfrac{a(t_{c1})}{||\ddot{\vec{S}}_i(t_{c2})||} \ddot{\vec{S}}_i(t_{c2}) \end{cases} \tag{20}$$

7 Conclusion

This flexible motion algorithm can allow for precise motion in three dimensions. When combined with other control methods, like PID control of velocity or acceleration and a direction-tracking algorithm for an end effector,

A Finding the Jacobians

When finding $\mathbf{J}$, $\mathbf{J}^{-1}$, and $\dot{\mathbf{J}}$, it is essential to consider the relative complexity of the generic formulas and the computation necessary to use them. To display each matrix concisely, some of the trig functions will be abbreviated like so:

$$S_n = \sin(\theta_n) \qquad C_n = \cos(\theta_n) \tag{21}$$

Now, $\mathbf{J}$ is as follows:

$$\mathbf{J} = \begin{bmatrix} -L_1 S_3 S_1 & -L_2 S_3 S_2 & C_3(L_1 C_1 + L_2 C_2) \\ -L_1 C_3 S_1 & -L_2 C_3 S_2 & -S_3(L_1 C_1 + L_2 C_2) \\ L_1 C_1 & L_2 C_2 & 0 \end{bmatrix} \tag{22}$$

However, $\mathbf{J}^{-1}$ is a very complex matrix which can be computed numerically. $\mathbf{J}^{-1}$ will not be listed here, although it is not exceedingly difficult to calculate for yourself.

However, $\dot{\mathbf{J}}$ will be listed here despite its complexity, as it is important to compute more accurate derivatives than can be provided by a more numerical approach. Using the multivariate chain rule, it is easy to compute $\dot{\mathbf{J}}$:

$$\dot{\mathbf{J}} = \begin{bmatrix} -L_1 C_3 S_1 \dot{\theta_3} - L_1 S_3 C_1 \dot{\theta_1} & -L_2 C_3 S_2 \dot{\theta_3} - L_2 S_3 C_2 \dot{\theta_2} & -L_1 C_3 S_1 \dot{\theta_1} - L_2 C_3 S_2 \dot{\theta_2} - S_3(L_1 C_1 + L_2 C_2)\dot{\theta_3} \\ L_1 S_3 S_1 \dot{\theta_3} - L_1 C_3 C_1 \dot{\theta_1} & L_2 S_3 S_2 \dot{\theta_3} - L_1 C_3 C_2 \dot{\theta_2} & L_1 S_3 S_1 \dot{\theta_1} + L_2 S_3 S_2 \dot{\theta_2} - C_3(L_1 C_1 + L_2 C_2)\dot{\theta_3} \\ -L_1 S_1 \dot{\theta_1} & -L_2 S_2 \dot{\theta_2} & 0 \end{bmatrix} \tag{23}$$

References

[1] Christian Blatter. Length of curve in 3d spherical coordinate. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/328340 (version: 2013-03-12).

[2] Giovanni Incerti. Motion planning of scara robots for trajectory tracking. *World Academy of Science, Engineering and Technology*, pages 61–70, 2015.