2-Jointed Arm Control System for FRC

Thomas Anderson, FRC 997

January 2023

1 Introduction

Robotic arm systems of two coplanar revolute joints are commonly implemented in both industry and education. However, the required control systems can be more complex than control systems for other types of coplanar 2-dof systems (revolute-prismatic, prismatic-prismatic) and require intentional design to meet the requirements.

2 Kinematics

2.1 Forward Kinematics

Forward kinematics in this case refer to the determination of the position in space of the arm segments and end effector from given information on the angle of each joint.

For arm segments

$$[S_a, S_b],$$

such that S_a is connected to the shoulder joint of the arm, and S_b is connected to the end effector, of positive lengths

$$[L_a, L_b],$$

and joint angles

$$[\theta_a, \theta_b],$$

such that θ_a cooresponds to the shoulder joint angle, and θ_b to the elbow joint angle (from S_a), the position of the tip of S_a is given by

$$\begin{bmatrix} x_a \\ y_a \end{bmatrix} = \begin{bmatrix} L_a cos(\theta_a) \\ L_a sin(\theta_a) \end{bmatrix},$$

and the position of the tip of S_b (the end effector) is given by

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \begin{bmatrix} L_a cos(\theta_a) + L_b cos(\theta_a + \theta_b) \\ L_a sin(\theta_b) + L_b sin(\theta_a + \theta_b) \end{bmatrix}.$$

2.2 Inverse Kinematics

Inverse kinematics refer to the derivation of a solution to arm joint angles given a desired end effector position.

The position inverse kinematics of a 2-jointed coplanar arm can be geometrically solved. [Lynch and Park, 2017].

Recalling arm segments $[S_a, S_b]$ from section 2.1 of the same characteristics, no solution exists for the goal point

$$\begin{bmatrix} x_g \\ y_g \end{bmatrix},$$

if

$$L_a + L_b < \sqrt{x_g^2 + y_g^2},$$

or

$$|L_a - L_b| > \sqrt{x_g^2 + y_g^2}.$$

In solvable cases, let

$$\gamma = \begin{pmatrix} tan^{-1}(y_g/x_g), \text{ if } x_g > 0 \\ tan^{-1}(y_g/x_g) + \pi, \text{ if } x_g < 0 \\ \pi/2, \text{ if } x_g = 0 \text{ and } y_g > 0 \\ -\pi/2, \text{ if } x_g = 0 \text{ and } y_g < 0 \end{pmatrix} [1]$$

$$\alpha = cos^{-1}\left(\frac{x_g^2 + y_g^2 + L_a^2 - L_b^2}{2L_a\sqrt{x_g^2 + y_g^2}}\right)$$

$$\beta = cos^{-1}\left(\frac{L_a^2 + L_b^2 - x_g^2 - y_g^2}{2L_aL_b}\right)$$

If x_g and y_g both equal 0, the system is at a singularity and infinitely many solutions exist (assuming that the point $[0, 0]$ is reachable by the arm).

Multiple solutions to the configuration of the arm can be found at all points which do not lie on the edge of the feasible space.

The solutions can be given as:

$$\begin{bmatrix} \theta_a \\ \theta_b \end{bmatrix}_r = \begin{bmatrix} \gamma - \alpha \\ \pi - \beta \end{bmatrix}$$

$$\begin{bmatrix} \theta_a \\ \theta_b \end{bmatrix}_l = \begin{bmatrix} \gamma + \alpha \\ \beta - \pi \end{bmatrix}$$

Geometric solutions are sufficient to determine the position of the arm, but for more advanced planning and movement time-derivatives of position are useful. These can be found with an intermediary Jacobian.

As per the inverse kinematics of the system, let

$$\vec{E} = \begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} L_a cos(\theta_a) + L_b cos(\theta_a + \theta_b) \\ L_a sin(\theta_a) + L_b sin(\theta_a + \theta_b) \end{bmatrix},$$

[1] Also known as the *atan2* function.

and

$$\vec{\theta} = \begin{bmatrix} \theta_a \\ \theta_b \end{bmatrix}.$$

It follows that

$$J_{\vec{E},\vec{\theta}} = \begin{bmatrix} \frac{\partial}{\partial \theta_a} x_g & \frac{\partial}{\partial \theta_b} x_g \\ \frac{\partial}{\partial \theta_a} y_g & \frac{\partial}{\partial \theta_b} y_g \end{bmatrix}$$

$$J_{\vec{E},\vec{\theta}} = \begin{bmatrix} \frac{\partial}{\partial \theta_a}(L_a cos(\theta_a) + L_b cos(\theta_a + \theta_b)) & \frac{\partial}{\partial \theta_b}(L_a cos(\theta_a) + L_b cos(\theta_a + \theta_b)) \\ \frac{\partial}{\partial \theta_a}(L_a sin(\theta_a) + L_b sin(\theta_a + \theta_b)) & \frac{\partial}{\partial \theta_b}(L_a sin(\theta_a) + L_b sin(\theta_a + \theta_b)) \end{bmatrix}$$

$$J_{\vec{E},\vec{\theta}} = \begin{bmatrix} -L_a sin(\theta_a) - L_b sin(\theta_a + \theta_b) & -L_b sin(\theta_a + \theta_b) \\ L_a cos(\theta_a) + L_b cos(\theta_a + \theta_b) & L_b cos(\theta_a + \theta_b) \end{bmatrix}$$

With our Jacobian, we can write the the relationship between the time-derivatives of joint angles and end effector position as

$$\frac{d}{dt}\vec{E} = \frac{d}{dt}\vec{\theta} J_{\vec{E},\vec{\theta}}$$

$$\frac{d}{dt}\vec{E} J^{-1}_{\vec{E},\vec{\theta}} = \frac{d}{dt}\vec{\theta}$$

3 Planning

3.1 Configuration Space

Configuration space is a 2-dimensional space with axes of the joint angles of each rotational joint. It is distinct from the Cartesian space containing the position of the end effector, but it is possible to convert between the two by using forward and inverse kinematics.

The configuration space for a double-jointed arm is non-Euclidean, as the angles of each joint can "wrap", or be coterminal with other angles. This makes our configuration space actually the surface of a torus, where traveling too far in one direction brings you to the other side of the graph in that dimension.

Configuration spaces are advantageous as they can better represent information about impossible or unwanted states so a path-planning algorithm can avoid them.

3.2 Space Traversal

A configuration space with defined obstacles still presents challenges in avoiding these obstacles. Many pathfinding algorithms exist, such as RRT, which randomly explores the space, A*, which biases towards the target, many of which have better performance or results, but Dijkstra's is very simple to implement and still performs well. [Dijkstra, 1959].

The hardware computing resources available on the robot during competition are powerful, but performance can still be an issue with dozens of processes simultaneously running at 50 hz. Because Dijkstra's graph traversal algorithm

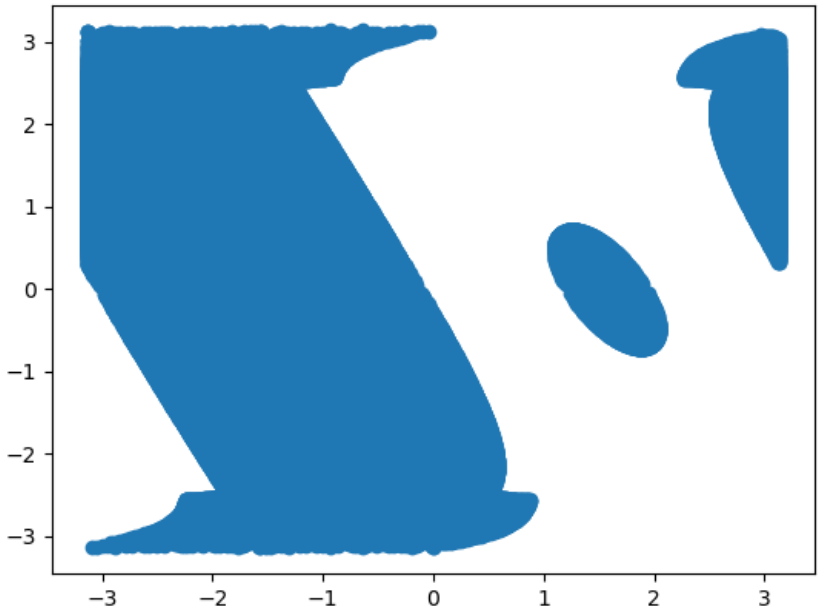


Figure 1: Sample configuration space for a two-jointed arm. White areas are valid states for the arm joints to occupy, according to parameters on prohibited spatial areas. Source code to generate this figure can be found at https://github.com/TandeJ/Config-Space-Finder.

is able to produce optimal paths and be run in real-time (although it requires pre-defined nodes and node connections), it is a natural fit for this application.

The algorithm works to minimize the sum of a cost function of moving from one point to another, over the course of a path, or sequence of moves between points. It does this over all points, not just those which are goals, and shortens all paths whenever it can.

The least-costly (in this case, lowest $\sum(\Delta\theta_a + \Delta\theta_b)$) path from the initial node to the goal node is followed. It is important to note that the trajectory for each degree of freedom should be viewed as a separate function. As there are an infinite number of possible initial and goal nodes, which depend on inverse kinematic solutions and physical state, the algorithm almost always does not begin at a pre-defined point. However, defining these initial and goal states is very possible at runtime.

The baselined system uses linear trajectories between points. However, there is interest into using n-th order Bézier splines, or parametric polynomial curves defined by control points, to make $\frac{d}{dt}\theta_a$ and $\frac{d}{dt}\theta_b$ continuous. There are un-

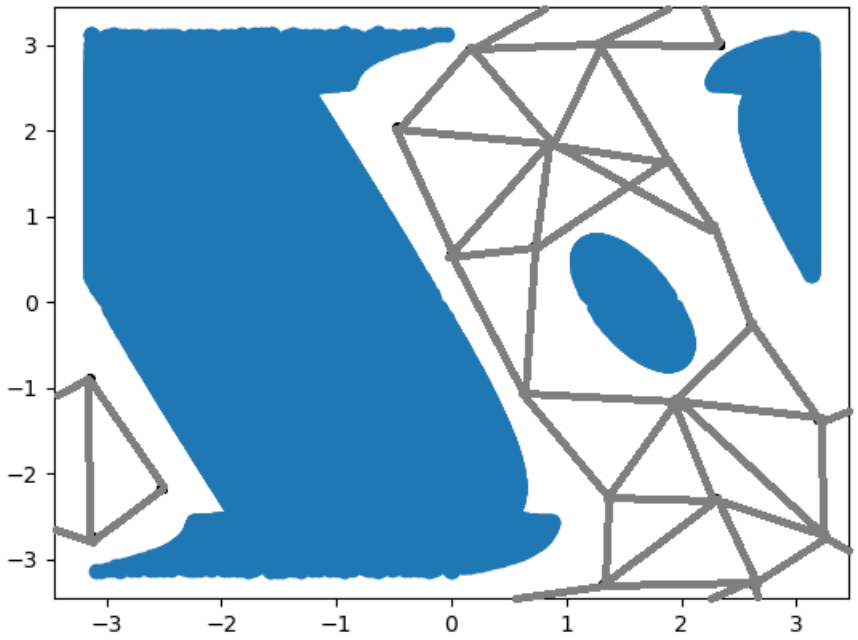


Figure 2: The sample configuration space from figure 1, with possible nodes and node connections added.

resolved difficulties ensuring that these Bézier splines cannot enter prohibited parts of the configuration space, and as such linear trajectories will be used, at least initially. Additionally, simple acceleration and deceleration constraints could easily be placed upon each degree of freedom, as they can effectively be considered independent of each other. This maintains the shape of connections and keeps them inside of prohibited spaces.

4 Control

Control of each joint is given to both a feedback controller, which can respond to disturbances, and a feedforward controller, which adds control effort per an *a priori* physics model.

4.1 Feedforward Control

Feedforward control uses modeled system dynamics to determine the control input that, theoretically, drives the system to respond as intended. We use

a feedforward model developed by Pedersen for a massed double-jointed arm, which does not account for factors like gearbox friction, which is generally accounted for by feedback control. [Pedersen, 2023]

Given arm segments $[S_a, S_b]$, with lengths $[L_a, L_b]$, masses $[m_a, m_b]$, distances from their centers of mass to their lowest joints $[r_a, r_b]$, and moments about their center of mass $[I_a, I_b]$, the system dynamics can be written as

$$M(\theta)\ddot{\theta} + C(\dot{\theta}, \theta)\dot{\theta} + \tau_g(\theta) = \tau$$

where the inertia matrix M is

$$M(\theta) = \begin{bmatrix} m_a r_a^2 + m_b(L_a^2 + r_b^2) + I_a + I_b + 2m_b L_a r_b cos(\theta_b) & m_b r_b^2 + I_b + m_b L_a r_b cos(\theta_b) \\ m_b r_b^2 + I_b + m_b L_a r_b cos(\theta_b) & m_b r_b^2 + I_b \end{bmatrix},$$

the coriolis and centrifugal force matrix C is

$$C(\dot{\theta}, \theta) = \begin{bmatrix} -m_b L_a r_b \dot{\theta_b} & -m_b L_a r_b sin(\theta_b)(\dot{\theta_a} + \dot{\theta_b}) \\ m_b L_a r_b sin(\theta_b)\dot{\theta_a} & 0 \end{bmatrix},$$

the gravity torque matrix τ_g is

$$\tau_g = \begin{bmatrix} gcos(\theta_a)(m_a r_a + m_b L_a) + m_b r_b g cos(\theta_a + \theta_b) \\ m_b r_b g cos(\theta_a + \theta_b) \end{bmatrix},$$

and the resultant joint torque matrix τ is

$$\tau = \begin{bmatrix} \tau_a \\ \tau_b \end{bmatrix}$$

However, torque is not a directly controllable quantity with our motor controllers, and cannot be summed with the voltage outputs of a feedback controller.

We can approximate a secondary feedforward model for direct-current permanent magnet motors with a gear reduction of G (G:1), empirical stall torque τ_s maximum input voltage V, and input voltage U, the torque τ output by each motor can be given by

$$\tau = \tau_s G \frac{U}{V}.$$

The feedforward input voltage for torque τ can be independently written as

$$U = \frac{\tau V}{\tau_s G}.$$

4.2 Feedback Control

Feedback control addresses unmodelable dynamics on the arm, such as a collision with a field element, friction, back-EMF, and so on. We will use simple Proportional-Integral-Derivative controllers independently for each arm joint, with setpoints defined by the trajectory generated in section 3.2, with kinodynamic constraints applied, unless some serious inadequacy in the PID controller is found. State-space LQR controllers could be considered in that case.

$$u(t) = k_p e(t) + k_i \int e(t)dt + k_d \frac{de}{dt}$$

A PID controller, properly tuned, is generally able to respond to large errors, constant small errors, and fast movements that could result in future error. PID controllers can be hard to tune without feedforward, especially for complex dynamics like our arm system, but with feedforward they can be very straightforward.

5 State Estimation

5.1 Dynamics Model

A prediction of where the arm should be can be obtained with a predefined mathematical simulation of the system, much like a feedforward model, which can convert from current states and control inputs to future states. This can be used to inform what "reasonable" state estimates can be, but should not be solely relied upon as they cannot compensate for unmodeled dynamics.

5.2 Sensors

Sensors, while prone to electromechanical randomness ("noise"), are important to determination of system state as they report unmodeled disturbances to the physics model, which could otherwise easily compound over time to create a wildly inaccurate prediction.

5.2.1 External Potentiometers

Potentiometers connected to the axis of rotation of the arm return analog signals to the main computer encoding absolute position of the arm segment relative to the mounting of the potentiometer. They retain their accumulation and position through power cycles, and thus can be relied on for a reliable estimate of actual arm state.

5.2.2 Integrated Encoders

The brushless motors used for the mechanical arm powertrain necessarily require rotary encoders for basic functionality. These encoders are conveniently also exposed to an end user through an API. They are not incremental encoders, in that their position is not tracked from a known point, and additionally not absolute as they lose any accumulation through a power cycle.

Unfortunately, these motors are situated at the opposite end of a large gear reduction from the actual arm system, and may be subject to a large amount of gear lash. However, as another source of measurement, they are still useful.

5.3 Kalman Filtering

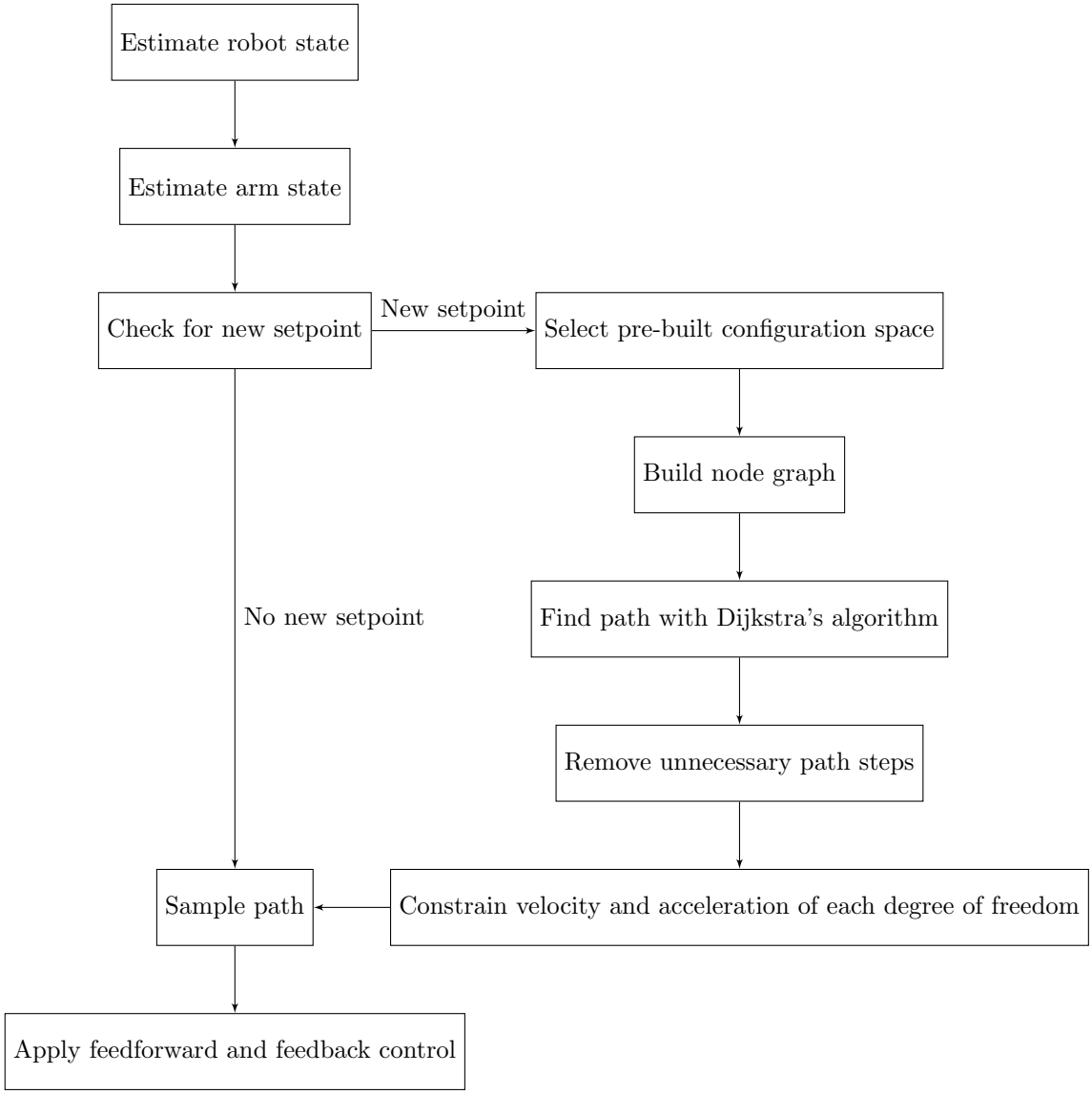
The Kalman filter is the final piece that allows us to use all of these disparate sources of state information, of varying accuracy and precision, to find one unified state estimate.

The Kalman filter exploits the Central Limit Theorem of statistics (that, as the number of independent sources of noise goes to infinity, the more Gaussian the resulting distribution is) to make estimates about the ground-truth state of a system better than any individual measurement. [Veness, 2017] Each source of data is given standard deviations of noise, and the Kalman filter runs looping estimation cycles in sequence *predict* and *update*. The filter **retroactively modifies** the weight it gives to each source of data to minimize the error between the *predicted* measurement and the *actual* measurement.

By using a Kalman filter, with sources of measurements including physics models and multiple sensors, a highly accurate estimate of arm state can be derived with little to no phase lag.

6 System Integration

6.1 Operation Sequence



6.2 Robot State Machine

The physical robot design this control system aims to address needs to avoid entering undefined or unwanted states. For instance, the floor intake of the robot needs to be retracted before scoring out the front to avoid violating rules

on extension from multiple sides of a robot.

The high-level robot code for controlling most if not all subsystems of the robot will be a form of a finite state machine, in which there are a number of predefined states the robot can enter where different code is executed, determined transitions between these states, and a mechanism to traverse these states to a desired state (which ensures the robot never enters a disallowed state).

6.3 Robot-Driver Interaction

Generally, due to the complexity and unconventional response dynamics of the system, control of the arm is handled in software, rather than by driver, using the control scheme enumerated previously.

To respond to the unpredictability of an FRC match, we also provide a method for a driver to control the velocity of the end effector in Cartesian space using the Jacobian derived in section 2.2 with a joystick.

In the notional driver control scheme, the driver or operator pushes a button cooresponding to a setpoint to update the desired robot state machine state to one where the arm reaches that setpoint, or to switch to the velocity control mode.

7 Conclusion

The control of a double-jointed robotic arm is certainly subject to many considerations and problems to be solved. However, as shown by its extensive use in both the "real world" and in education, the problems are not unsolvable by any measure, and many of them are easier than they appear.

Regardless, it takes a substantial amount of effort to convert what's written here into an actual control system on an actual robot.

References

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematlk*, 1:269–271.

[Lynch and Park, 2017] Lynch, K. M. and Park, F. C. (2017). *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press.

[Pedersen, 2023] Pedersen, R. (2023). Two jointed arm dynamics. `"https://www.chiefdelphi.com/uploads/short-url/pfucQonJecNeM7gvH57SpOOgPyR.pdf"`.

[Veness, 2017] Veness, T. (2017). *Controls Engineering in the FIRST Robotics Competition*. n.p.