

Coordination Tailoring

J. Randolph Andrews, President
Douloi Automation, Inc.
Campbell, CA

Introduction

The phrase “One size fits all” often produces suspicion. How can a single solution accommodate the special requirements of my problem? General purpose motion controllers are useful when dealing with generally encountered problems. However in the same way that a quality suit needs tailoring to fit well, so also do motion controllers for certain applications. This article discusses "coordination tailoring" through an example glue dispensing application. In the course of solving this application both mechanism and controller attributes will be discussed which help simplify the problem and contribute towards the solution.

Example Glue Gun Application

A scara-style robot is equipped with a glue gun. The mechanism has a shoulder joint, an upperarm link, an elbow joint, and a forearm link. The robot deposits glue in a pattern described by an XY coordinate file. A top-view “stick” model of the robot with a rectangular glue pattern is shown in Figure 1.

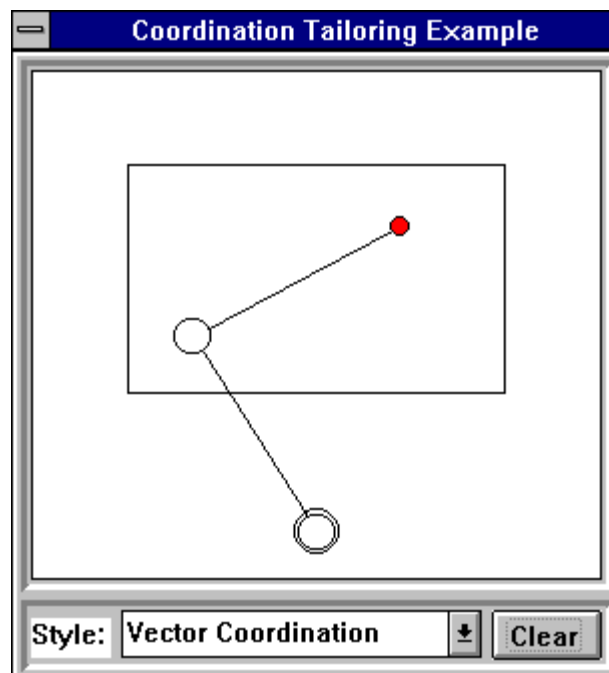


Figure 1. "Stick" model of Scara Robot

The shoulder joint is in the bottom center with the upperarm and forearm extending upward in a "lefty" configuration. The smaller dark circle represents the glue gun tip.

The length of the upper arm, measured from the shoulder axis to the elbow axis, is the same as the length of the forearm, measured from the elbow axis to the glue gun tip. The gun is raised and lowered by a linear Z axis on the end of the forearm link. There is no “wrist” rotation in this example. The glue gun is turned on and off through an output bit. For glue-bead uniformity it is important for the robot to perform the motion at a constant XY velocity.

"Off-the-Rack" Vector Coordination

One motion control approach is to calculate joint angles for the various XY coordinates in the path file and perform “vector coordinated” motion to these points. Vector coordination involves having all axis in a group begin motion, accelerate, slew, decel, and stop at the same time regardless of the move distances for each axis. For Cartesian mechanisms vector coordination produces straight line movement. However having the robot perform vector coordination between these mapped XY coordinates produces a rectangle with warped sides as shown in Figure 2.

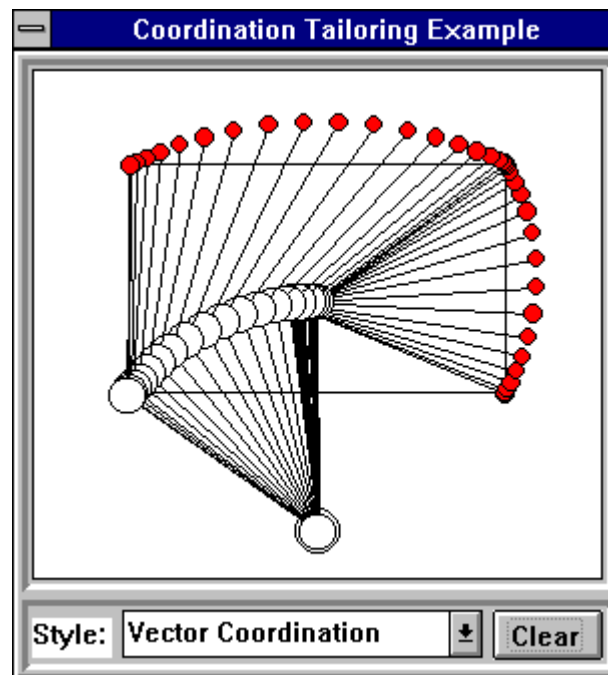


Figure 2. Warped Motion

Even if additional interpolation points were provided to reduce this warping, there would still be an XY speed problem. The glue gun speed at a far radial reach would be higher than the speed in a tucked position for the same shoulder joint rotation. Vector coordination does not fit this problem and some kinematic tailoring is required.

Kinematic Coordination Tailoring

The problem is fundamentally an XY problem. Through kinematic techniques the motion will be tailored so that the application developer solving this problem can pretend that the robot has an XY structure. The issues related to rotary joint movement will be concealed inside the motion system.

Creating the Static Geometry Connection

The first step in making the rotary robot appear Cartesian is to understand the mathematical connection between Cartesian space and robot joint angles. A convenient Cartesian origin is at the shoulder axis with the X axis to the right. The shoulder and joint angles are defined with respect to the X axis. Angles of 0 for both would position the robot to be fully extended to the right with both links on the X axis. The elbow angle is relative to the X axis and is independent of the shoulder angle. Other mechanical designs may have different angle definitions and joint angle coupling. Calculating joint angles from a Cartesian point becomes solving the triangle shown in figure 3.

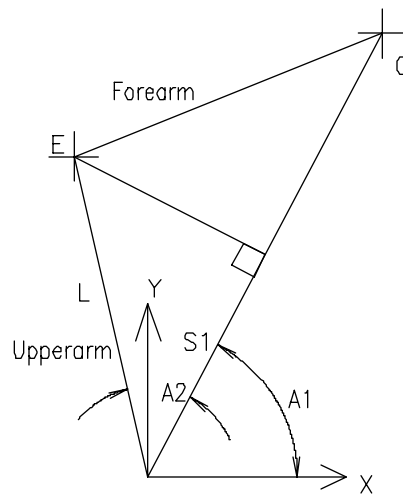


Figure 3. Cartesian To Joint Triangle

“C” is the Cartesian location. Angle A1 is the angle of the vector C. The length of S1 is half of the magnitude of C due to link symmetry. L is the link length. A2 is the arccos of S1/L. The Shoulder angle is A1+A2. The elbow angle is A1 - A2. One way to perform this calculation is shown below using the Object Pascal language in Douloi’s Servo Application Workbench.

```
Procedure CartesianToJoint(C:T2SingleVector;var J:T2SingleVector);  
  
  var S1,A1,A2:double;  
  
  begin  
    S1:=C.Magnitude/2;  
    A1:=C.Angle;  
  
    FInit;  
    PushDouble(S1);  
    PushLongint(UpperarmLength);  
    ArcCos;  
    RadiansToDegrees;  
    A2:=PopDouble;
```

```

J.Init(
    (A1+A2)*ShoulderCountsPerDegree,
    (A1-A2)*ElbowCountsPerDegree);
end;

```

The T2SingleVectors used in the calculation are 2 dimensional floating point vectors convenient for doing math in Cartesian and Polar formats. Some of the math is performed with floating point math hardware, the floating point coprocessor of a 486 chip. These instructions are used like a “Reverse polish” calculator, pushing arguments onto a “stack” and then performing operations.

The calculation time for this routine was measured on a 486 DX/2 and found to be 180 microseconds, far below the default controller sample rate of 1000 microseconds.

There are times when the Cartesian position of the machine needs to be found based on the joint angles. This involves representing the robot links as vectors and “walking” the vector chain out to the endpoint as shown below:

```

Procedure JointToCartesian(J:T2SingleVector;var C:T2SingleVector);

var Upperarm:T2SingleVector;
var Forearm:T2SingleVector;

begin
Upperarm.InitPolar(UpperarmLength,J.x/ShoulderCountsPerDegree);
Forearm.InitPolar(ForearmLength,J.y/ElbowCountsPerDegree);
C:=UpperArm;
C.Add(Forearm);
end;

```

Creating the Dynamic Motion Connection

Having a “mapping” for Cartesian geometry to joint geometry is important but is not enough to manage dynamic Cartesian movement. Some form of "trajectory generator" is required. The following method sends commands to a “virtual” Cartesian robot. A “virtual” robot is like a simulator. It responds to all of the normal motion commands available but has no actual motors attached. The outputs of the simulator are the commanded positions of the individual axis in the group. These commanded positions are available every sample and can be fed into the Cartesian to joint calculations. This produces, every sample, the joint positions corresponding to the desired Cartesian movement. This process is illustrated in figure 4.

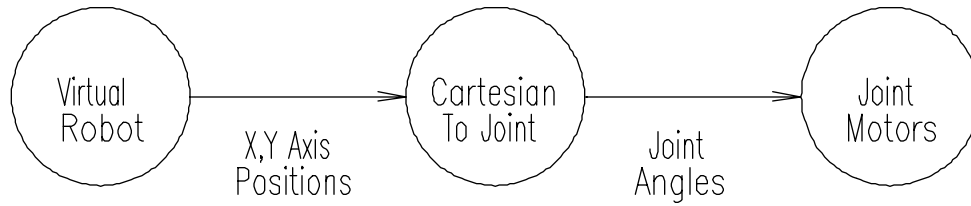


Figure 4. Virtual Robot Driving Joints through Kinematics

A virtual robot can be created in the following manner:

```
var GlueGun:T2Axis;
```

This 2 axis "glue gun" is declared like a variable and responds to motion commands such as SetSpeed, MoveTo, MoveBy etc. This virtual robot acts as a "surrogate" or "stand-in" for the missing mechanical Cartesian machine, performing the operations that a Cartesian machine would have performed if it was present. By having the real-time positions flow through the CartesianToJoint calculation the robot "follows" the simulator producing Cartesian motion.

Movement of this "robot simulator" is connected to the actual joint motors by performing the CartesianToJoint calculation every controller sample period and assigning the commanded positions for the real motors. This is done with the following routine:

```

Procedure PerformKinematics;

  var J,C:T2SingleVector;

  begin
  C.Init(
    GlueGun.XAxis.CommandedPosition,
    GlueGun.YAxis.CommandedPosition);
  CartesianToJoint(C,J);
  Shoulder.SetCommandedPosition(J.XLongint);
  Elbow.SetCommandedPosition(J.YLongint);
  end;

```

This routine is initially "engaged" through the following routine.

```

procedure EngageKinematics;

  var C,J:T2SingleVector;

  begin
  J.Init(Shoulder.CommandedPosition,Elbow.CommandedPosition);
  JointToCartesian(J,C);
  GlueGun.SetCommandedPosition(C.XLongint,C.YLongint);
  ScheduleTask(TaskAddr(PerformKinematics),1);
  end;

```

This routine uses the current joint angles of the robot to determine the Cartesian location of the glue gun. The "virtual" robot GlueGun is then assigned to that location so as to be "in sync" with the physical mechanism. Then the kinematic calculations are scheduled to run every millisecond on an ongoing basis.

If normal joint movement is to be performed, the kinematics can be disengaged with the following routine:

```
Procedure DisengageKinematics;  
begin  
  AbortTask(TaskAddr(PerformKinematics));  
end;
```

Trying it all on

With kinematic coordination tailoring the previously warped rectangle becomes straight as shown in Figure 5.

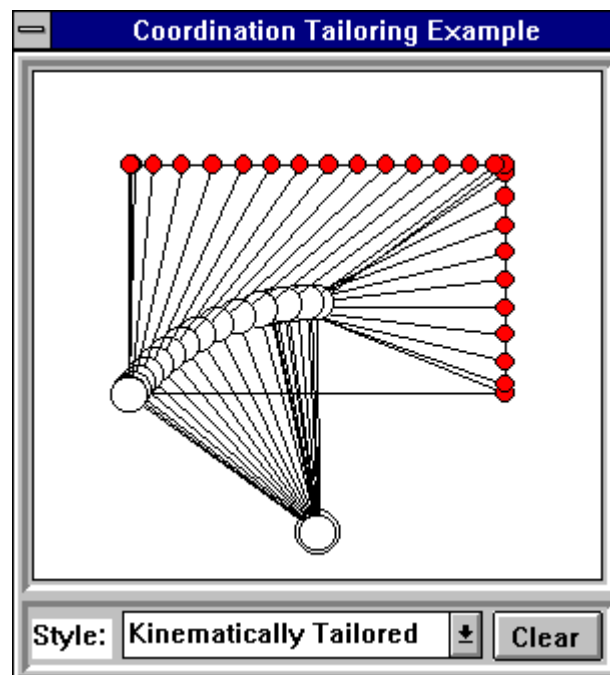


Figure 5. Straight Line Motion

Now that a Cartesian robot is "available", the glue dispensing problem becomes much simpler. One example solution is the following.

```

Procedure PerformGlueOperation(var DataFilename:string);

var PointFile:TFile;
var FirstPoint:boolean;
var X,Y:longint;

begin
PointFile.Assign(DataFilename);
PointFile.Reset;
FirstPoint:=true;
ZAxis.MoveTo(ClearancePlane);
EngageKinematics;
while not PointFile.EndOfFile do
begin
PointFile.Readln(X,Y);
GlueGun.MoveTo(X,Y);
if FirstPoint then
begin
ZAxis.MoveTo(GluePlane);
SetOutputBit(GlueControl,on);
FirstPoint:=false;
end;
end;
SetOutputBit(GlueControl,off);
ZAxis.MoveTo(ClearancePlane);
GlueGun.MoveToVector(ParkPosition);
DisengageKinematics;
PointFile.Close;
end;

```

The PointFile object is used to read information from the computer. The ZAxis moves out of the way, above the work. The kinematics is engaged and the glue gun moved to each point in the file. After the first move, the ZAxis descends to the glue plane and the glue gun is turned on. After moving to the last point the glue gun is turned off, the ZAxis raises the glue gun, and the robot is parked. The GlueGun could also have been directed in continuous curved motion if arc information had been part of the data provided.

What helps keep things simple?

Equal Link Lengths

Having the Upperarm link length the same as the Forearm link length simplified the math. Was having these lengths be the same cheating? Yes it was! Every opportunity should be taken to “cheat” so as to make the problem simpler as long as important functionality or flexibility is not lost. Minimizing automation deployment time and expense is critical. No bonus points are given for solving hard problems which could have been simple problems.

As an engineer involved with the control responsibility for a project it is important to keep other engineers informed of the consequences of their decisions. An “either way is fine” decision on the part of a mechanism designer could become a “trivial versus difficult” issue for the person working on the control

system. The reverse is also true. It may be much simpler to calculate some trigonometry in real-time than to make 2 joint axis intersect, for example. Good inter-discipline communication is necessary for simple, timely solutions.

Simplified Kinematic Treatment

Normally kinematic equations are represented in a matrix format which is quite versatile and able to describe mechanisms of most any shape. This 2 dimensional problem did not require this more sophisticated representation. Here a simplification occurred by using “right-sized” vector expressions that could be manipulated in polar and Cartesian formats. Do not undervalue simplicity.

Floating Point Math Hardware

Modern microprocessors are remarkable having features such as a 32 bit data bus, internal cache, and 80 bit floating point hardware. Particularly in applications involving real-time trigonometry there is no reason not to benefit from this type of technology.

High-speed Application Programs

Without high speed application programs, the calculations for the kinematics may have spanned many controller sample periods. In response to this problem interpolation equations are often used to give the axis a place to go while the calculation is in progress. This requires additional programming effort, adds complexity and reduces profiler bandwidth. Being able to perform the entire equation each millisecond helps keep profiling simple.

Multithreading Real-Time Environment

This coordination tailored solution involves an ongoing calculation from Cartesian space to joint angles. This approach requires a multithreading, hard-real-time environment to properly work. The entire control system must not be “tied up” while the robot is performing a straight line move. The safety task, operator interface, and other communication activities all must be sustained while the kinematic equations are operating.

Expressive Language

Using names like “ForearmLength”, “Elbow”, and "GlueGun" makes the program much simpler to read and understand. The ability to create, fix, and tailor software is critically dependent on the developer’s ability to understand what the code is doing. Comments are helpful, but the best correspondence between what a program says it does and what it actually does is obtained through expressive techniques that are self-documenting.

Summary

This example has shown that when “one size fits all” motion coordination cannot handle the job, a coordination tailored solution can. The best approach for solving advanced problems is to keep things as simple as possible. When sophisticated approaches are required, such as real-time kinematic calculations, having suitable hardware and software tools that support tailoring can make a difficult job easy to do. The coordination tailoring example described in this article is available on disk from the author.

About the Author

Mr. J. Randolph Andrews is the President of Douloi Automation in Campbell, California. Prior to establishing Douloi in 1991, Mr. Andrews spent four years with Galil Motion Control and four years at Hewlett Packard's corporate research laboratory in the Applied Physics Research Center as well as the Manufacturing Research Center. Mr. Andrews attended the Massachusetts Institute of Technology where he received his B.S.M.E. and B.S.E.E. in 1981, and his M.S.M.E. in 1983. He is a four time recipient of the first place award in the Mechanical Engineering Department's DeFlorez Design Competition at MIT.