Finding Error Corrective Matrix

Joseph Leung

May 17, 2017

Abstract

The purpose of this paper is demonstrate a method of force control for a manipulator to complete a given task with a set initial and final configuration known. Modeling our contact as a spring mass damper, we should be able to synthesize a matrix to generalize this motion from initial to final configuration. The reason this is different is because our motion involves decreasing the errors of the task at hand. This paper is a guide to form this matrix that results in error-corrective motion for a task.

1 Problem Motivation

When a robot contacts rigid objects, there are forces imposed on the manipulator, which help us define how the manipulator moves in space. When moving rigid objects with just position control, we may not have the exact position of the object just because sensors have noise and there is uncertainty in how we determine the position of the object in space.

The robot should be consistently able to get from the an initial position to a desired final configuration. Though there are already ways for the robot to do this, we are introducing a new way to look at how we do this. Normally, we have a compliance matrix that is consistent, which means that there are bounded forces. This does however not mean that it is error corrective. As the name implies, we want a matrix that reduces the error when heading to a final configuration. When forming these matrices, we also need to account for the different geometries of the objects and space the manipulator is moving in.

Past research in fine motion planning has already been done by Mason, Lozano-Perez, and Taylor [2] [3] [4], where there is a planning algorithm that searches through the geometry of the nominal path. There is also error reduction and recovery that stops when the present configuration is recognized as a failure. When such failures are recognized the motion will bring the manipulator back to a position where the original nominal motion plan will work. With this paper we plan to continue on if there is an error and keep moving to reduced error toward the end goal. The reason we are doing this is so that we don't have to do any correction and just input a generalized compliant and error corrective matrix that will account for all possible configurations/errors.

Our plan will succeed despite there being errors because it is these errors that forces arise, which are used to move in an error corrective way given a task. It must be noted that the assembly speeds

are low enough that inertial effects of the robot and work piece are small compared to frictional effects.

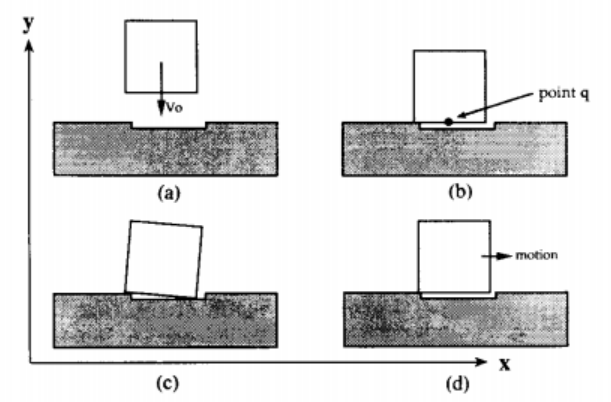

Figure 1: This shows an example of how we have a task and how it moves in space and sets up our example of moving a block into a dent: (a) This shows the block moving a nominal velocity V_o (b) Point q is the point where we consider our reference point is where our coordinate frame is (c) The block is resting on the lip of the dent and the bottom of the dent (d) This shows the motion possible to get from the lip of the dent to the final configuration. This horizontal movement reduces the error.

2 Problem Statement

The way we are modeling the manipulator and how contact forces result in motion is using a spring-mass damper system, where X_0 is the initial nominal position, which is proportional to an applied force.

$$X - X_0 = CF \tag{1}$$

We note that in a 3-D case there are 6 degrees of freedom, 3 for translation and 3 around rotational axis. In vector form X and X_0 are 6 by 1 vectors, F is a 6 by 1 vector representing forces and torques, and C is a 6 x 6 matrix. Using equation 1, we can then change C such that by knowing the force, results in a motion we desire. The control law implemented is following the motion of a spring-mass damper after force is applied to the system:

$$V = V_0 + AF \tag{2}$$

Observing equation 2, we note that in the absence of any force there will be a nominal velocity followed by the manipulator. A is also a 6 x 6 matrix, which when multiplied by F will add to the velocity V. V is the motion of the manipulator. We will call A the accommodation matrix, which is the the inverse of a damping matrix. The A matrix is what is changing our value of V as V_0 is a constant, which will be an arbitrary velocity heading straight down, and F represents the forces sensed.

In this paper, we will be referencing a planar example to help guide us through deriving and formulating of the equations to form the accommodation matrix as seen in fig 1.

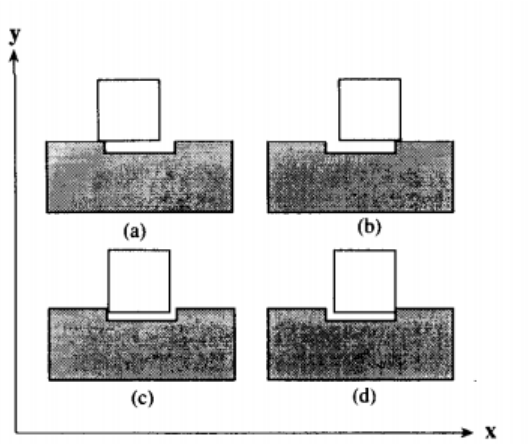

Figure 2: These are the four configurations we are considering for the task: (a)shows the block at the left side lip of the dent (b) Shows the block at the right side lip of the dent (c) Shows the block inside the left side of the dent (d) Shows the block inside the right side of the dent

Note that we are only considering a finite set of possible contact configurations shown in fig 2. If given more information about the geometry of the model then we would be able to define more configurations such as the block not hitting the dent entirely. For our contact configurations, we assume negligible orientational errors and translational errors limited to 2 mm.

Now that we have set up the model of how the manipulator moves, the environment, and possible configurations, we have to specify how to determine the A matrix, so we can control the movement of the manipulator. The A matrix has two major properties that the manipulator must exhibit: "Bounded Forces" and "Error Reduction".

Property 1: The Bounded forces property requires that the A matrix must be such that the contact forces that arise from mating parts must be bounded. For example if my hand were a manipulator with a box in it and I push it down into a surface, there will be a normal force pushing back up. I must make sure that the manipulator does not try to keep pushing until forces try to approach infinity. This property is to prevent damage to the workpiece and/or the manipulator.

In order to do this, let us consider the block in fig 2(a) pushed down on a surface. To avoid unbounded forces we must have a velocity component in the +y component. Essentially we don't want to push into the normal of the surface that the workpiece is on.

Property 2: The Error Reduction Property is to be expressed in the A matrix such the displacement $X - X_0$ is reduced.

Expressing Properties Mathematically:
When we look at the inner product AF we know the result should match a 6 x 1 velocity vector or for planar cases a 3 x 1 velocity vector. Since we know this, doing a dot product with a unit vector and AF should be able to get a result +, 0, or -, telling us how the manipulator will move.

$$\hat{u_i} * Af_i = t_i \; i = 1...m \tag{3}$$

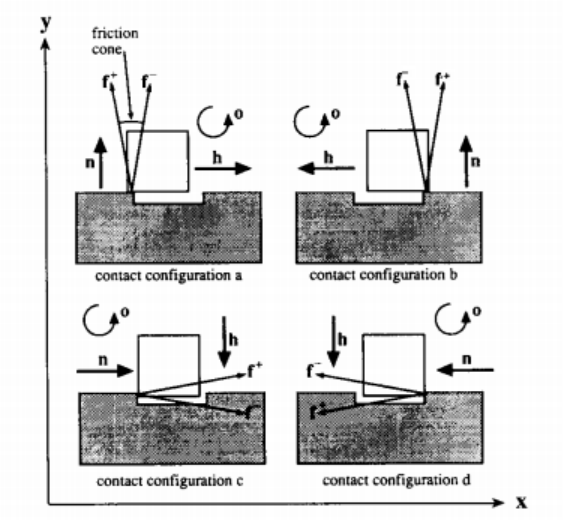

Figure 3: This figure shows the four configuration and the direction of the unit vectors that help us figure out how the block should move toward the end configuration (a)First configuration (b) Second Configuration (c) Third Configuration (d) Fourth Configuration

Forces arise from errors or being not in the final configuration, but we do not actually need the magnitude of the forces but rather the direction of the forces, which is the f in eq 3. The forces we consider come from the friction cone, and we observe the direction in the different configurations when setting up our problem. f^+ of the friction cone is noted to be on the side opposite the motion heading towards $\hat{h}$ as seen in fig 3.

The scalar t_i represents inverse stiffness. Determining how the friction cone will work we first need to know the coefficient of friction as the $atan(\mu) = \gamma$ for the angle of the friction cone. This gives us the bounds of the forces/friction cone. What the friction cone tells us that the friction force though unknown must lie in the cone, so by taking into account the maximum and minimum of the friction cone in the planar example we would be able to take into account any force in the cone.

Coming back to property 1, we now can express it in mathematical terms. The bounded forces says we should not be pushing the object into a surface. This means that there should be no velocity in the direction into the surface which can be shown in the following equation:

$$\hat{n} \cdot V = 0 \tag{4}$$

Next we express the F as a product of its magnitude and direction: $|F|\hat{f}$ and now combine (4) with (2) and expand it while replacing F. We end up with:

$$|F| = -\frac{\hat{n} \cdot V_0}{\hat{n} \cdot Af} \tag{5}$$

The denominator of (5) represents the stiffness of the manipulator to interactions with a particular surface. Since we don't want the magnitude of the force to be negative or 0, it should be positive. If the magnitude were 0 then it would break property 1, therefore the "bounded forces" property is represented by:

$$\textbf{Property 1: } \text{"}Bounded\ forces\text{"} \implies n \cdot AF \approx t_1 \tag{6}$$

This value of t_1 is a user selected value because all we care about is that t_1 is positive.

Now we will go onto property 2. To express it in a similar form as (6), first we must define a unit vector "home" $\hat{h}$ where "home" is a unit vector that points in a direction toward the end configuration, reducing (1). There are also orthogonal vectors to both $\hat{h}$ and $\hat{n}$, which we need to prevent the motion of so that the point on the surface does not move in an unnecessary direction. In the planar case, we have the z axis as the orthogonal vector. We can now express Property 2 as:

$$\hat{h} \cdot V > 0 \tag{7}$$

and

$$\hat{o_i} \cdot V = 0 \tag{8}$$

Equation (7) tells us we want our motion to be in the direction of home and that there be no motion along the orthogonal direction, i.e no rotation. Let us choose that $\hat{h} \cdot V \approx |V_0|$ and combine it with (2) to get:

$$\hat{h} \cdot Af = (\hat{h} \cdot V_0 - |V_0|)\frac{\hat{n} \cdot AF}{\hat{n} \cdot V_0} \tag{9}$$

Since we want (7) to be true we also want (9) to be positive in order for it to to be error reducing. Therefore the motion of going toward $\hat{h}$, the home unit vector, should be positive meaning (9) can be re-expressed as:

$$\textbf{Property 2: } \text{"}Error\ Reduction\text{"} \implies \hat{h} \cdot AF = t_2 \tag{10}$$

and the condition to prevent motion along the orthogonal direction is:

$$\hat{o_i} \cdot AF = 0 \tag{11}$$

The main equations that we need to pay attention to that sum up property 1 and 2 are: (6),(10), and (11).

When there are more than one points of contact, we change eq (2) to be $V = V_0 + AF_1 + AF_2$ and exchange each F with $F_i = |F_i|\hat{f}_i$. One thing about f_i is that when we look at fig 3, the point that is on the surface is not point q. There is a moment at point q due to the friction forces at the point that must be considered.

3 Solution Approach

Okay so now that we have all the properties represented mathematically, the next step is to form the accommodation matrix. All our conditions have been reduced to the form :

$$\hat{u_i} * Af_i = t_i \tag{12}$$

We have three equations that represent our two properties. Also we must remember that f_i is the force to be expected, and we consider both f^+ and f^-, which means we have 6 conditions for each configuration in the planar case. In total, since we have 4 configurations, we have 24 equations. To know the direction of the force itself, we must prescribe the value of the coefficient of friction.

Forming the conditions so they match equation (12), we can look at each configuration and determine what $\hat{u}$ is for each. Stated before, we said that the A matrix was a 6 x 6 for the 3D case and 3 x 3 for the 2D case. Let us look at the planar case with A as a 3 x 3 matrix. In the A matrix there are $n = 9$ real value variables. (The A matrix acts as a type of k value/ see if I want to add this)Here we need to string it out as a n x 1 vector.

$$A = \begin{pmatrix} A_{xx} & A_{xy} & A_{x\theta} \\ A_{yx} & A_{yy} & A_{y\theta} \\ A_{\theta x} & A_{\theta y} & A_{\theta\theta} \end{pmatrix} \Rightarrow A^{so} \begin{pmatrix} A_{xx} \\ A_{xy} \\ A_{x\theta} \\ A_{yx} \\ A_{yy} \\ A_{y\theta} \\ A_{\theta x} \\ A_{\theta y} \\ A_{\theta\theta} \end{pmatrix} \tag{13}$$

We do this because our construction of the form $\hat{u} * Af = t$ shown explicitly is:

$$(u_x, u_y, u_\theta) \cdot \begin{pmatrix} A_{xx} & A_{xy} & A_{x\theta} \\ A_{yx} & A_{yy} & A_{y\theta} \\ A_{\theta x} & A_{\theta y} & A_{\theta\theta} \end{pmatrix} \begin{pmatrix} f_x \\ f_y \\ f_\theta \end{pmatrix} = t \tag{14}$$

and we want it to be written as

$$(u_x f_x, u_x f_y, u_x f_\theta, u_y f_x, u_y f_y, u_y f_\theta, u_\theta f_x, u_\theta f_y, u_\theta f_\theta) \cdot A^{so} = t \tag{15}$$

After using equation (15) for all the conditions and configurations, we end up with a huge m by n matrix , where m is the number of equations.

$$\begin{pmatrix} u_x^1 f_x^1, u_x^1 f_y^1, u_x^1 f_\theta^1, u_y^1 f_x^1, u_y^1 f_y^1, u_y^1 f_\theta^1, u_\theta^1 f_x^1, u_\theta^1 f_y^1, u_\theta^1 f_\theta^1 \\ \cdot \\ \cdot \\ \cdot \\ u_x^m f_x^m, u_x^m f_y^m, u_x^m f_\theta^m, u_y^m f_x^m, u_y^m f_y^m, u_y^m f_\theta^m, u_\theta^m f_x^m, u_\theta^m f_y^m, u_\theta^m f_\theta^m \end{pmatrix} \cdot A^{so} = \begin{pmatrix} t^1 \\ \cdot \\ \cdot \\ t^m \end{pmatrix} \tag{16}$$

We reduce this equation to the form of:

$$G \cdot A^{so} = t \tag{17}$$

where G is the m x n matrix. It is noted that G, the huge matrix, means there are more equations than number of variables. It means that there is generally no A^{so} that satisfies (17) exactly. We can still find though a solution that most closely matches the values of t on the right hand side of the equation. The way we find A^{so} is by forming a n-by-m pseudo inverse G^*:

$$A^{so} = G^* t \tag{18}$$

After finding A^{so} plug it into equation (17) to see if the values match the positives, and 0's that were part of the user defined values. For this example all the equations may not be satisfied.

$$t^{\sim} = GA^{so} = GG^* t \tag{19}$$

If $t^{\sim}$ does not match t in sign we can still verify component by component if A still posses the "bounded forces" and "error reduction" property by computing V=V_o +AF for every contact configuration.

4 Results

Let us show that we can do this given the example from fig 2. Our task is to put a block into a dent and there are four given configurations. First we need some givens which is that the block is a 1 in x 1 in block. The point we mark as our origin and where our reference frame is, is at point q as seen in fig 1. Our unit vectors are shown in fig 3. The coefficient of friction that we took was .25 which is what determines the angle of the friction cone. Now we can start filling in our m x n matrix and t_i values so that we can form G.

The way I grouped my 6 conditions per configurations is not unique because I arbitrarily chose to group my normal, home, and orthogonal in pairs with f^+ and f^-. In this example the way I set it up was:

$$\hat{n}_i \cdot Af^+ = t_i$$
$$\hat{n}_i \cdot Af^- = t_i$$
$$\hat{h}_i \cdot Af^+ = t_i$$
$$\hat{h}_i \cdot Af^- = t_i$$
$$\hat{o}_i \cdot Af^+ = t_i$$
$$\hat{o}_i \cdot Af^- = t_i$$

Remember that we have to have the form shown in equation (15) in order to solve for the A matrix which can be done by forming a 3 x 3 matrix and stringing it out horizontally:

$$\begin{pmatrix} u_x \\ u_y \\ u_\theta \end{pmatrix} \cdot (f_x, f_y, f_\theta) = \begin{pmatrix} u_x f_x & u_x f_y & u_x f_\theta \\ u_y f_x & u_y f_y & u_y f_\theta \\ u_\theta f_x & u_\theta f_y & u_\theta f_\theta \end{pmatrix} \tag{20}$$

How to Pick t Values- We know fundamentally that the dot products of the unit vector with AF is a velocity term. This means that if we want the value to be 0, we do not want motion in that direction, and if we want it to be positive, we want it to be along that direction of motion. For the

sake of simplicity, we will choose the possible values of t to be 1 or 0. For our 4 conditions, we have to think fundamentally how do I get to the final configuration given my geometry and conditions. An example is that in the first configuration, we go along the x-axis in the positive direction. If we need to go in the direction defined by our unit vector the value becomes a 1. In the case of the orthogonal vectors we want them to be 0.

It is to be noted for the third and fourth configurations, the value of t for the dot products with the "home" vectors is equal to 0. Since our "bounded forces" already takes into account the block getting to the end configuration and not needing to push into the normal vector pointing in the -y direction and that V_0 is already always telling the block to go downward we do not need to tell the manipulator to go down even more. Therefore these values of t are 0. Also the value for V_0 is 1 initially.

Now that the equations are set, remember to find the pseudo-inverse and find A^{so} using equation (18). The value of the A matrix taken at point Q from following the method is

$$A_q = \begin{pmatrix} .9701 & 0 & -1.9403 \\ 0 & 1.0308 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{21}$$

Putting this result back into the equation we know our result is satisfactory if it matches the 0 and positives that we chose for t. We can also see from our results that there is no rotational component that comes out when solving for V.

Now if we want to specify a point we want to measure torques and where we want to apply a rotation at, we can apply a transformation to the accommodation matrix so that we can do just that. This means it is necessary to choose a coordinate system. Lets use our example where we used point q as shown in fig 1, and we want to choose a point p. For our example point p is point q + $[0; 1; 0]$, which is right above q. Now we apply the transformation to eq (22) to get (23)

$$A_p = \begin{pmatrix} 1 & 0 & q_y - p_y \\ 0 & 1 & p_x - q_x \\ 0 & 0 & 1 \end{pmatrix} A_q \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ q_y - p_y & p_x - q_x & 1 \end{pmatrix} \tag{22}$$

And we get

$$A_q = \begin{pmatrix} 2.91 & 0 & -1.9403 \\ 0 & 1.0308 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{23}$$

Another Example:

There are other examples like placing a block down which is similar to the example presented. As shown in fig 4):

In this example, we have fewer configurations which makes it more likely of getting t values that match when inputting the A matrix back into eq (17). Here we do want rotation so that the block is put down gently, and there is no skidding. This is represented by a dot product preventing motion in that direction, making the t-value = 0. In a sense, we have the same 2 properties represented by

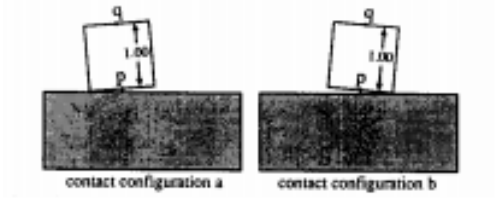

Figure 4: This figure shows two configurations of putting a block down where configuration a needs a clockwise rotation and configuration b needs a counterclockwise rotation.

the three equations (6), (10), and (11), The most important thing is that the values for the normal, home, and orthogonal vectors are different for each example. Note the points in this example are different from example 1; p and q are switched. After inputting new values for the normal vectors for the property equations we get a matrix:

$$A_q = \begin{pmatrix} 2.06 & 0 & -2.06 \\ 0 & .52 & 0 \\ -2.06 & 0 & 2.06 \end{pmatrix} \tag{24}$$

and after applying the transformation shown at equation ((22)) at point q to point p we get

$$A_p = \begin{pmatrix} 0 & 0 & 0 \\ 0 & .52 & 0 \\ 0 & 0 & 2.06 \end{pmatrix} \tag{25}$$

5 Conclusion

I learned the method of finding an accommodation matrix given a setup with configurations and a task. To make sure that my result did end up going toward the final configuration, I was able to use equation (2) to verify that the block would go to the final configuration. This paper is inspired by the work on the RCC wrist but it differs because what we have is not a clear solution path from point A to B. What we have presented is a method to produce a path.

Limitations:

Key limitations of this paper are as follows:

1.)It only accounts for a limited number of configurations and we end up having to use a psuedo-inverse because we have more equations than unknowns.

2.)Another limitation is that there is no real metric of how to measure if we are getting closer to the end configuration based on just the forces alone. Though it is shown that it is possible to use forces that result from error to guide the manipulator to motions that reduces errors.

3.)There is also no way to distinctly create a method for generating the configurations themselves.

4.)Also note this example is in 2 dimensions. When working in 3 dimensions there are many things to consider with a block having line contact with the surface. Also for the 3D problem,

the friction cone is actually a cone and not a triangle like in the planar example, which means it becomes a second-order convex optimization problem.

6 Acknowledgments

I want to thank the author of the paper I based my discussion off of: Micheal A. Peshkin, My professor Nilanjan Chakraborty, and PhD candidate Anirban Sinha for helping me through this process.

7 References

[1]Micheal A. Peshkin "Programmed Compliance for Error Corrective Assembly " IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION. Vol. 6. No. 4 August 1990.

[2]M.A.Erdmann, "On motion planning with uncertainty," M.S. thesis¡ Massachusetts Inst. Tech, Cambridge, 1984.

[3]J.C laTombe, Preprint, 1989.

[4]T.Lozano-Porez, M.T. Mason, and R.H. Taylor, "Automatic synthesis of fine-motion strategies for robots," Int.J.Robotics Res., vol. 3 , no. 1, pp. 3-24, 1984.