Closed-Loop Operation of Actuator Arrays

Jonathan E. Luntz

Mechanical Engineering
University of Michigan
Ann Arbor, MI 48109

William Messner and Howie Choset

Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

An actuator array performs distributed manipulation where an object being transported and manipulated rests on a large number of stationary supporting actuators. The authors have developed a macroscopic actuator array consisting of many motorized wheels. As opposed to a MEMS array, the analysis requires the explicit modeling of the discreteness in the system, including the set of supports, distribution of weight, and generation of traction forces. Using an open-loop wheel velocity field, discreteness causes undesirable behavior such as unstable rotational equilibria, suggesting the use of object feedback. Discrete distributed control algorithms are derived by inverting the dynamics of manipulation. These algorithms reduce the many-input-three-output control problem to a three-input-three-output control problem.

1 Introduction

An actuator array performs distributed manipulation where many small stationary elements (which we call cells) cooperate to manipulate larger objects. Current applications of actuator arrays range from micromechanical systems transporting pieces of silicon wafer to macroscopic arrays of motorized wheels transporting cardboard boxes. In such applications, an object lies on a regular array as it is transported and oriented. Many cells support the object simultaneously, and as it moves, the set of supporting cells changes. While supporting the object, each cell is capable of providing a traction force on it, and the combined action of all the cells supporting the object determines the motion of the object.

Actuator arrays represent a recent development in robotic manipulation. Typical work in actuator arrays addresses the task of bringing an object to a particular position and orientation on the array. Generally, open-loop (passive) modes of operation are used, where the action of the cells is pre-programmed and constant, and a "force field" is established in which the object moves [1, 2]. These analyses were intended for micro-electromechanical (MEMS) scale applications, where the large number of actuators justifies the use of the assumption that the forces from the discrete actuators are approximated by a continuous force field applied over the area of the object. Under this assumption, researchers used potential field theory to predict motions and stable poses of objects.

In this work, we examine a macroscopic actuator array, the Modular Distributed Manipulator System (MDMS). Each cell of the MDMS is made up of a pair of motorized roller wheels which together can apply a directable traction force to objects such as cardboard boxes. The MDMS uses distributed control and therefore is fully programmable. An 18 cell prototype MDMS has been built and tested. On the MDMS, as few as four or six cells support an object such that continuous approximations fail and we must account for discreteness when analyzing the MDMS.

Discreteness causes imprecision in open-loop manipulation, particularly in the orientation of objects. For example, under certain modeling assumptions appropriate to the MDMS, the net moment acting on an object is not a direct function of the object's orientation, and orientation can only be done to cell resolution [6]. In addition, some objects which have stable rotational equilibria on a continuous array have unstable rotational equilibria on a discrete array [5]. Therefore, closed-loop object control is necessary to precisely orient objects. In a closed-loop (active) mode of operation, information about the object's motion is used to update the action of the cells. Such information may be obtained from local sensing at each cell or from some global sensor, such as a vision system. Other

work has been done in closed-loop distributed manipulation [7] using vibrating plates to generate continuous force fields acting on small (point) objects rather than pointwise forces acting on large objects.

In this paper, we derive closed-loop manipulation strategies for positioning and orienting objects on the MDMS. The dynamics of manipulation on the MDMS are summarized in Section 2 paying particular attention to discreteness. In Section 3, we derive closed-loop policies by inverting the dynamics. Section 4 contains simulations of closed-loop manipulation. Concluding remarks are made in Section 5.

2 Dynamics of Manipulation

We will consider the dynamics of an array of cells transporting and rotating an object in the plane while it rests on a single arbitrary set of cells. To compute the traction forces on the object from each wheel, we first compute the supporting (normal) forces and then apply a viscous-type friction law.

Consider an object of weight W, whose center of mass is located at $\vec{X}_{cm} = \begin{bmatrix} x_{cm} & y_{cm} \end{bmatrix}^{\mathrm{T}}$ resting on n cells located at the positions contained in the following matrix.

$$\mathbf{B} = \begin{bmatrix} 1 & \dots & 1 \\ x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix} \tag{1}$$

The object is supported by vertical normal forces $\vec{N} = \begin{bmatrix} N_1 & \dots & N_n \end{bmatrix}$ whose determination requires consideration of equilibrium of the object in both the vertical (z) direction and in rotation about the x and y axes. Equilibrium provides three equations, and since we have n supports, the system is statically indeterminant, and we must consider flexibility in the system.

We assume each support is a linear spring. Physically, this flexibility is either a flexible suspension under each wheel as shown in Figure 1 or, as in the prototype, flexibility in the surface of the bottom of the object. We assume the bottom of the object is flat such that the spring deflections (and hence the normal forces) distribute linearly under the object. The 3 equilibrium equations combined with n instances of this compatibility constraint form a set of $n+3$ equations and $n+3$ unknowns from which we can solve for $\vec{N}$ as a function of object position [6].

$$\vec{N}^{\mathrm{T}} = W\mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}\vec{X}_{cm}\right) \tag{2}$$

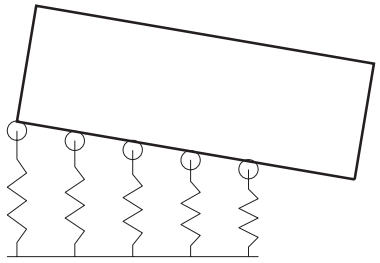


Figure 1: Flexible supports act as springs to distribute the load.

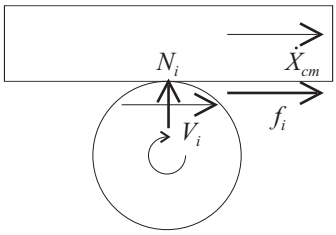


Figure 2: Interaction between wheel and object.

The horizontal forces on the object are derived from the normal forces through the use of a viscous-type friction law (see Figure 2). The horizontal force from each cell $\vec{f}_i$ is proportional to a coefficient of friction μ, that cell's normal force N_i, and the vector difference between the velocity of the wheel and the velocity of the object at the point of the cell.

$$\vec{f}_i = \mu\left(\vec{V}_i - \dot{\vec{X}}_{cm} + \omega\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\left(\vec{X}_i - \vec{X}_{cm}\right)\right)N_i \tag{3}$$

where ω is the rotation speed of the object. This traction force from each cell is summed over all the cells. Define a wheel velocity matrix $\mathbf{V}$ as

$$\mathbf{V} = \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \end{bmatrix} = \begin{bmatrix} V_{1_x} & \dots & V_{n_x} \\ V_{1_y} & \dots & V_{n_y} \end{bmatrix} = \begin{bmatrix} \vec{V}_1 & \dots & \vec{V}_n \end{bmatrix} \tag{4}$$

Summing vectorially, the net horizontal force is

$$\vec{f} = \mu\mathbf{V}\ \vec{N}^{\mathrm{T}} - \mu\dot{\vec{X}}_{cm}W \tag{5}$$

Observe that the net horizontal force is not a function of the object's rotation speed - the terms multiplying ω are identically zero [6]. Furthermore, the second term in this equation is a constant linear damping term. Substituting $\vec{N}$ from Equation 2 yields

$$\vec{f} = \underbrace{\mu W\mathbf{V}\mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{k_s}}\vec{X}_{cm}$$

$$+\underbrace{\mu W \mathbf{V}\mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\begin{bmatrix}1\\0\\0\end{bmatrix}}_{\vec{f}_o} - \mu\dot{\vec{X}}_{cm}W \quad (6)$$

The 2×2 spring constant matrix $\mathbf{k_s}$, and the offset force vector $\vec{f}_o$ are constant while the object rests on a particular set of supports.

We can compute the net torque on the object similarly [6], resulting in the following expression for the net moment acting on the object as a function of position and rotational speed (and set of supports).

$$\tau = \underbrace{\mu W \vec{R}\mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\begin{bmatrix}1\\0\\0\end{bmatrix}}_{\tau_o} + \underbrace{\mu W \vec{R}\mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\begin{bmatrix}0&0\\1&0\\0&1\end{bmatrix}}_{\vec{k}_{s_\tau}}\vec{X}_{cm}$$
$$+\vec{X}_{cm}\times\left(\vec{f}_o + \mathbf{k_s}\vec{X}_{cm}\right) - \mu\omega\left(\vec{\mathcal{X}}\vec{N}^{\mathrm{T}} - W\vec{X}_{cm}^{T}\vec{X}_{cm}\right) \quad (7)$$

where $R_i = \vec{X}_i \times \vec{V}_i$ and $\mathcal{X}_i = \vec{X}_i^{\mathrm{T}}\vec{X}_i = \left(x_i^2 + y_i^2\right)$. The vector $\vec{k}_{s_\tau}$ relates torque to position, and τ_o is a scalar constant torque. The net applied torque and damping are both effectively position dependent. The term multiplying ω is always negative, and hence dissipative (although not constant). Note that while an object rests on a particular set of supports, torque on the object is not a function of orientation. This is important for determining stable orientations.

3 Position and Orientation Feedback

Our strategy to implement position and orientation feedback is to apply velocity fields which reduce the array to a single "virtual" actuator capable of generating a desired force and torque on the object. Since force and torque change as the object moves, we must continuously recompute and adjust the wheel speeds. We assume sensing of the object's position and orientation is done externally, for example, by a vision system. It is desirable to distribute the computation to reduce communication such that each cell need only be aware of the desired net force and torque and its location relative to the center of the object.

3.1 Applying both Force and Torque

We compute velocity fields by inverting the relationship between wheel speeds and net force and torque. As a first attempt, we specify both a force and torque to compute the velocity field. Without loss of generality, we set the origin of the system to lie at the center of the object such that the cells move rather than the object. Also, we ignore the damping terms in Equations 6 and 7. For the purpose of feedback control, we treat the damping as a property of the controlled system rather than the actuator since the damping forces are functions of the objects motion and not the wheel speeds. The expressions for the action of our virtual actuator (i.e. the net force and torque on the object) reduce to:

$$\vec{f} = \vec{f}_o = \mu W \mathbf{V}\mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\begin{bmatrix}0&0\\1&0\\0&1\end{bmatrix}$$
$$\tau = \tau_o = \mu W \vec{R}\mathbf{B}^{\mathrm{T}}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\begin{bmatrix}1\\0\\0\end{bmatrix} \quad (8)$$

We can rewrite these equations in terms of the stacked velocity vector formed by stacking the transposes of the x and y component rows of $\mathbf{V}$.

$$\begin{bmatrix}f_{o_x}\\f_{o_y}\\\tau_o\end{bmatrix} = \mu W \cdot \quad (9)$$
$$\underbrace{\left[\begin{array}{ccc|ccc|ccc}1&0&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0&0\\0&0&0&0&0&0&1&0&0\end{array}\right]\left[\begin{array}{c|c|c}\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\mathbf{B}&0_{3\times n}&0_{3\times n}\\\hline 0_{3\times n}&\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\mathbf{B}&0_{3\times n}\\\hline 0_{3\times n}&0_{3\times n}&\left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\mathbf{B}\end{array}\right]\left[\begin{array}{c|c}I_n&0_n\\\hline 0_n&I_n\\\hline -\mathbf{D_y}&\mathbf{D_x}\end{array}\right]}_{\mathbf{Q}}\begin{bmatrix}\vec{V}_x^{\mathrm{T}}\\\hline \vec{V}_y^{\mathrm{T}}\end{bmatrix}$$

where $\mathbf{D_x}$ and $\mathbf{D_y}$ are diagonal matrices containing the x and y components of the cell positions. We pseudo-invert this underconstrained set of equations to solve for the stacked velocity vector.

$$\begin{bmatrix}\vec{V}_x^{\mathrm{T}}\\\hline \vec{V}_y^{\mathrm{T}}\end{bmatrix} = \frac{1}{\mu W}\mathbf{Q}^{\mathrm{T}}\left(\mathbf{Q}\mathbf{Q}^{\mathrm{T}}\right)^{-1}\begin{bmatrix}f_{o_x}\\f_{o_y}\\\tau_o\end{bmatrix} \quad (10)$$

Unfortunately, it is not possible to algebraically reduce $\mathbf{Q}^{\mathrm{T}}\left(\mathbf{Q}\mathbf{Q}^{\mathrm{T}}\right)^{-1}$ to a more useful form. Each cell's wheel speeds depend on the positions of all cells currently supporting the object. Therefore, while a field computed in such a manner will provide the desired force and torque, its computation cannot be distributed, and a centralized controller must give speed commands to each cell. This is not practical for a large system because of bandwidth limitations.

3.2 Superposition of Fields

Because traction force from each cell varies linearly with wheel speeds, net forces and torque add with the

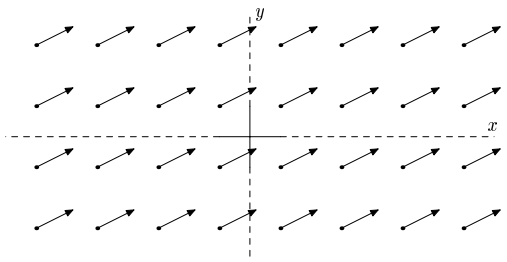


Figure 3: Uniform field which applies an arbitrary net force on an object with no net torque.

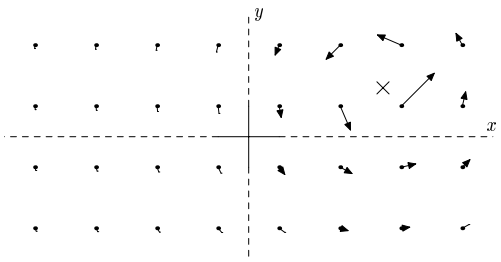


Figure 4: Computed rotational field which applies an arbitrary net torque on an object.

superposition of velocity fields. Therefore, we adopt the strategy of superimposing two fields: a field to apply a force without a torque and a field to apply a torque without a force, where each field operates under distributed control.

To compute a field to apply a force with no torque, we invert the relationship between wheel speeds and net force. By using the Penrose pseudo-inverse, we obtain the velocity field which provides the desired net force with the minimum sum of squares of wheel speeds. This minimizes extraneous motions, and hopefully (with no guarantee) the resulting field will generate no net torque.

The expression for the net force in terms of the stacked velocity vector is

$$\begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix} = \mu W \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c|c} \left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\mathbf{B} & 0_{3\times n} \\ \hline 0_{3\times n} & \left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1}\mathbf{B} \end{array}\right] \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \end{bmatrix} \quad (11)$$

We pseudo-invert this underconstrained set of equations to solve for the stacked velocity vector.

$$\begin{bmatrix} \vec{V}_x^{\mathrm{T}} \\ \vec{V}_y^{\mathrm{T}} \end{bmatrix} = \frac{1}{\mu W} \left[\begin{array}{c|c} \mathbf{B}^{\mathrm{T}} & 0_n \\ \hline 0_n & \mathbf{B}^{\mathrm{T}} \end{array}\right] \left[\begin{array}{c|c} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ \hline 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{array}\right] \begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix} = \frac{1}{\mu W} \left[\begin{array}{c|c} 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ \hline 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{array}\right] \begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix} \quad (12)$$

Figure 3 shows the resulting uniform field. Our methodology ensures that the net force is independent of the set of supports such that we can apply the same net force without knowing which cells the object rests on. The velocities of the cells are decoupled, and are only functions of the net force to be applied. This field easily operates under distributed control, where a centralized controller need only broadcast the desired net force to all the cells.

This uniform field can be shown to apply no torque. While it may seem intuitive that this be true, it is not trivial. If we specified force from each cell rather than wheel velocities, there would be a net torque when the center of mass of the object and the centroid of the supporting cells did not coincide. Since the traction force from each cell depends on the supporting force, however, the distribution of weight ensures that all moments balance. The algebraic proof is omitted.

The same methodology used to compute a field to apply an arbitrary force applies to the application of an arbitrary torque. The expression for the net torque in terms of the stacked velocity vector is

$$\tau_o = \mu W \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1} \mathbf{B} \left[\begin{array}{c|c} -\mathbf{D_y} & \mathbf{D_x} \end{array}\right] \begin{bmatrix} \vec{V}_x^{\mathrm{T}} \\ \vec{V}_y^{\mathrm{T}} \end{bmatrix} \quad (13)$$

Once again, we pseudo-invert this underconstrained set of equations to solve for the stacked velocity vector.

$$\begin{bmatrix} \vec{V}_x^{\mathrm{T}} \\ \vec{V}_y^{\mathrm{T}} \end{bmatrix} = \frac{1}{\mu W} \begin{bmatrix} -\mathbf{D_y} \\ \mathbf{D_x} \end{bmatrix} \left(\begin{bmatrix} -\mathbf{D_y} & \mathbf{D_x} \end{bmatrix} \begin{bmatrix} -\mathbf{D_y} \\ \mathbf{D_x} \end{bmatrix} \right)^{-1} \mathbf{B}^{\mathrm{T}} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tau_o \quad (14)$$

This expression can be algebraically reduced since $\mathbf{D_x}$ and $\mathbf{D_y}$ are diagonal matrices. After some algebra, the wheel velocities decouple and the velocity of each wheel becomes

$$\begin{aligned} V_{x_i} &= \frac{1}{\mu W} \frac{-y_i}{x_i^2 + y_i^2} \tau_o \\ V_{y_i} &= \frac{1}{\mu W} \frac{x_i}{x_i^2 + y_i^2} \tau_o \end{aligned} \quad (15)$$

Figure 4 shows the resulting computed rotational field. The vector formed by each cell's two wheel speeds is perpendicular to its position relative to the center of the object with a magnitude inversely proportional to distance. Again, our methodology ensures that the net torque is independent of the set of supports. This field also operates under distributed control, where a centralized controller need only broadcast the desired torque and current object location to all the cells.

This rotational field, however, has two problems. First, since the wheels speeds are inversely proportional to their distance from the object center, some

wheel speed commands may become arbitrarily large, and the wheel speeds will saturate. In practice, only one cell will be close enough to the object center to saturate, and its contribution will be limited.

The second problem is that a net force is applied by the rotational field. We obtain the expression for the net force by substituting Equations 15 into Equation 11. Considering the x component of Equation 11 (the y component is handled similarly), the expression for the net force is

$$\begin{aligned} f_{o_x} &= \mu W \left[1\,0\,0\right] \left(\mathbf{B}\mathbf{B}^{\mathrm{T}}\right)^{-1} \mathbf{B} \begin{bmatrix} \frac{1}{\mu W}\frac{-y_1}{x_1^2+y_1^2} \\ \vdots \\ \frac{1}{\mu W}\frac{-y_n}{x_n^2+y_n^2} \end{bmatrix} \\ &= \left[1\,0\,0\right] \begin{bmatrix} n & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 \end{bmatrix} \begin{bmatrix} \sum \frac{-y_i}{x_i^2+y_i^2} \\ \sum \frac{-x_i y_i}{x_i^2+y_i^2} \\ \sum \frac{-y_i^2}{x_i^2+y_i^2} \end{bmatrix} \end{aligned} \quad (16)$$

This expression, in general is not equal to zero and generates an extra force which was not intended by this field. This disturbance force ($\vec{f}_d$) is dependent on the application of torque. While it can be shown that there are many positions and orientations at which this disturbance force vanishes [4] and its effect may be small, we desire an alternate method for generating a torque without generating a disturbance force.

3.3 Known Torque Direction Method

To derive a field to generate a torque with no force, we exploit the distribution of weight among the cells to compute wheel velocities in such a way that no net force is applied. Rotational equilibrium of the object about the x and y axes ensures that the normal forces are distributed such that

$$0 = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix} \begin{bmatrix} N_1 \\ \vdots \\ N_n \end{bmatrix} \quad (17)$$

We can represent the net force on the object as

$$\vec{f} = \mu \begin{bmatrix} V_{1_x} & \cdots & V_{n_x} \\ V_{1_y} & \cdots & V_{n_y} \end{bmatrix} \begin{bmatrix} N_1 \\ \vdots \\ N_n \end{bmatrix} \quad (18)$$

These two equations are of the same form with the position matrix replaced by the velocity matrix. If we set the wheel velocity matrix to match the position matrix, the net force will be zero. In addition, we can swap the two rows of the velocity matrix and negate the top row while maintaining zero force to generate the following velocity field.

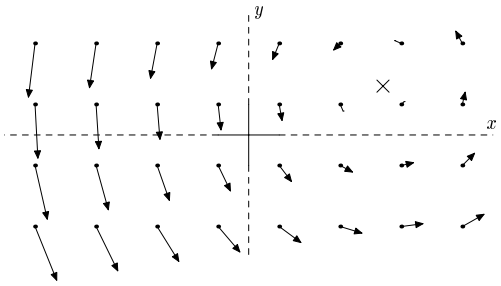


Figure 5: A kinematic rotational field applies a torque of known direction to the object without applying a net force.

$$\begin{aligned} V_{i_x} &= -\alpha y_i \\ V_{i_y} &= \alpha x_i \end{aligned} \quad (19)$$

where α is a constant with which we can scale the field. We refer to this as a kinematic rotational field since the wheel speeds vary linearly with position as they do in rigid body rotation as shown in Figure 5. This field is decoupled and can operate under distributed control. The net torque applied by this field is

$$\tau_l = \sum_i \tau_i = \mu\alpha \sum_i N_i \left(y_i^2 + x_i^2\right) \quad (20)$$

This torque is not constant; it changes with both object position and set of supports. However, because each cell contributes positively to the summation of torques, the *direction* of the torque is known. Therefore, we eliminate the disturbance force generated by the computed rotational field at the expense of uncertainty in the applied torque magnitude. This field is still useful for feedback by scaling α proportionally with the desired torque.

4 Implementation of Feedback

We now implement position and orientation feedback using the uniform field for force actuation and both the computed rotational field and the kinematic rotational field for torque actuation. Both the position and orientation loops use proportional control. Figure 6 shows the structure of the feedback system under the computed rotational field. The linear translational damping and position dependent rotational damping are lumped into the object dynamics. The translational and rotational components of the MDMS are separated to show the effect of the disturbance force

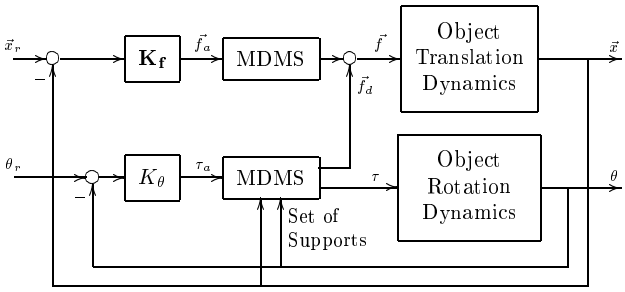


Figure 6: Structure of feedback control system under the computed rotational field. Blocks labeled MDMS represent the effect of the uniform field (top) and a rotational field (bottom). The object dynamics blocks are mass-damper systems. Position feedback loop is coupled to orientation feedback loop by the disturbance force generated by the application of a torque. Only the disturbance force is affected by the set of supports.

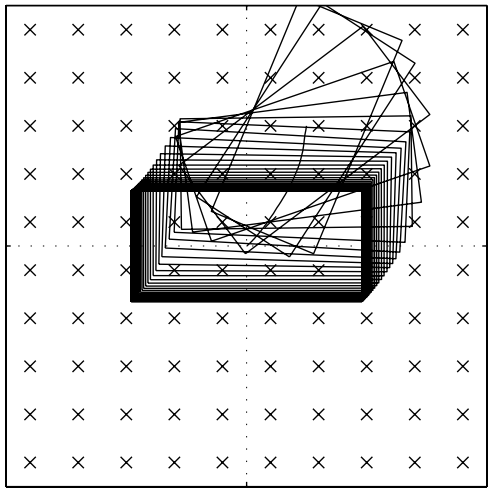


Figure 7: Simulation of the closed-loop system under the computed rotational field. The object is shown as the moving rectangle, with a curve tracing the motion of it's center. Each cell is marked with an 'X'.

generated by the computed rotational field. The feedback structure under the kinematic rotational field is identical, but without the disturbance force.

Asymptotic stability of this feedback system under both types of rotational fields is guaranteed [3]. We have omitted the mathematical proof which employs the circle criterion and bounds on nonlinear damping and disturbance terms. Asymptotic stability of the closed-loop system can be explained intuitively as follows. Except for the position dependent damping, the orientation loop is independent of the position loop. The orientation loop is necessarily asymptotically stable because the nonlinear damping dissipates energy and no rotational energy is added by the rest of the loop. In the case of the kinematic rotational field, the torque gain varies, but is bounded. For constant (or zero) reference input, the object will eventually reach the desired orientation with no steady state error because of the free integrator in the mass-damper plant. When this occurs, the torque command, and therefore the disturbance force, will vanish. The position loop then operates independently and as a pair of time-invariant linear mass-damper systems under proportional control, and will be asymptotically stable.

Evaluation of the closed-loop performance requires that the nonlinear rotational damping and the disturbance force (or the nonlinear application of torque) be bounded. Any bound on these terms can be used to guarantee stability, and while it is easy to come up with some bounds on these terms, it is difficult to come up with bounds to make useful performance guarantees. The computation of useful bounds and their application to performance is the subject of future work.

Figure 7 shows a simulation of the closed-loop system under the computed rotational field. The object reaches equilibrium exactly at both the desired position and orientation. Since this is approximately a set of mass-damper systems, by adjusting the gains, we can adjust the overshoot and settling time of each component of the closed-loop response. The curvature of the path of the center of the object (which otherwise would have been straight) demonstrates the effect of the disturbance force. Figure 8 shows a simulation of the closed-loop system under the kinematic rotational field. In this case, the object moves in a straight line to the equilibrium position, and the rotational motion is greatly unaffected by the torque uncertainty.

5 Conclusions

In this paper we have derived closed-loop control policies for manipulating objects. While more complicated to implement than open-loop policies, these strategies address precision limitations apparent in the open-loop operation of discrete actuator arrays. We derived a class of fields which reduce the array of actuators to a single virtual actuator for which simple feedback methods apply. Because of the design methodology used, these fields eliminate the dependence of the dynamics on the set of supports. Because of the use

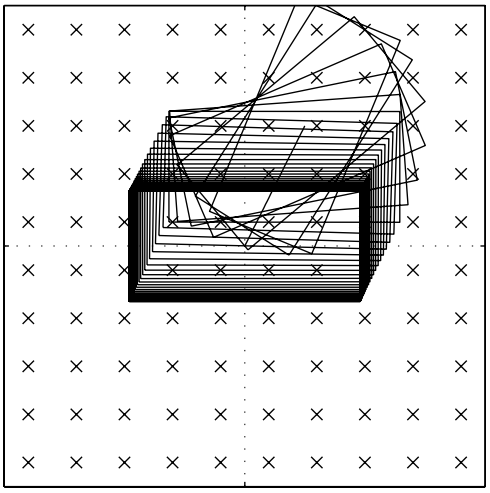


Figure 8: Simulation of the closed-loop system under the kinematic rotational field.

of distributed control, limitations are present in these strategies in the form of coupling from torque to force, or in the form of torque uncertainty which do not affect stability or precision of equilibria, but may affect closed-loop performance.

Our future efforts will be toward performance guarantees for closed-loop manipulation using distributed control. These feedback methods will be extended to include not only manipulation to a single position and orientation, but also path and trajectory following with multiple objects simultaneously.

References

[1] K.F. Böhringer, B.R. Donald, R. Mihailovich, and N.C. MacDonald. A theory of manipulation and control for microfabricated actuator arrays. In *Proceedings, IEEE International Conference on Robotics and Automation*, 1994.

[2] L. Kavraki. Part orientation with programmable vector fields: Two stable equilibria for most parts. In *Proceedings, IEEE International Conference on Robotics and Automation*, 1997.

[3] J.E. Luntz and W. Messner. Closed-loop stability of distributed manipulation. In *Submitted to AACC American Controls Conference*, 2000.

[4] J.E. Luntz, W. Messner, and H. Choset. Discreteness issues of manipulation using actuator arrays. In *Workshop on Distributed Manipulation, IEEE International Conference on Robotics and Automation*, 1999.

[5] J.E. Luntz, W. Messner, and H. Choset. Open loop orientability of objects on actuator arrays. In *Proceedings, IEEE International Conference on Robotics and Automation.*, 1999.

[6] J.E. Luntz, W. Messner, and H. Choset. Velocity field design for parcel manipulation on the modular distributed manipulator system. In *Proceedings, IEEE International Conference on Robotics and Automation.*, 1999.

[7] D. Reznik, E. Moshkoich, and J. Canny. Building a universal planar manipulator. In *Proceedings. Workshop on Distributed Manipulation at the International Conference on Robotics and Automation*, 1999.