

Jonathan E. Luntz

Department of Engineering
University of Michigan
Ann Arbor, MI 48109, USA
jluntz@engin.umich.edu

William Messner Howie Choset

Department of Mechanical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract

Distributed manipulation systems induce motions on objects through the application of many external forces. An actuator array performs distributed manipulation using a planar array of many small stationary elements (which are called cells) that cooperate to manipulate larger objects. Typically, highly dense actuator arrays are modeled as spatially continuous, programmable force fields, although in many implementations a relatively small number of actuators supports an object and continuous assumptions break down. This paper serves two purposes: to present a methodology for modeling and analyzing the dynamics of manipulation on a highly discrete actuator array and to present a methodology for designing manipulation strategies on discrete actuator arrays. This is done in the context of a particular macro-scale actuator array comprising a fixed planar array of motorized wheels. Modeling of the dynamics takes into account several models of the interaction between the actuators and the object, the distribution of the weight of the object among the supports, and the discrete nature of the system. Under certain modeling assumptions, the manipulation dynamics of an object are extremely simple for a given set of supporting cells. An inversion of these piecewise-continuous dynamics generates a fully continuous open-loop manipulation strategy, effectively smoothing out the discontinuities. The authors show that although the resulting manipulation field may stably position and orient any object in the continuous field case, discreteness causes many objects to experience unstable rotational equilibria. Thus, poor orientation precision is a limitation of open-loop manipulation using discrete actuator arrays and motivates the use of feedback. The authors also derive closed-loop manipulation strategies through an inversion of the discrete dynamics that reduce the many-input, three-output distributed control problem to a standard three-input, three-output control problem that operates under

Distributed Manipulation Using Discrete Actuator Arrays

distributed control. In effect, the array of actuators is reduced to a single virtual actuator capable of applying a desired net force and moment on an object. It is proven that even in the presence of dynamic coupling and nonlinearities introduced due to discreteness, these closed-loop strategies are asymptotically stable. Multimedia extensions include a complete simulator and videos of the experimental prototype.

KEY WORDS—distributed manipulation, actuator array, distributed control, parts feeding, programmable force fields

Important Variables

i	Cell index
n	Number of cells supporting object
$\vec{X}_i = [x_i \ y_i]^T$	i th cell position
\mathbf{X}	Cell position matrix ($2 \times n$)
\vec{x}, \vec{y}	x and y cell positions ($1 \times n$)
N_i, \vec{N}	Normal force on i th cell and normal force vector ($1 \times n$)
$\vec{X}_{cm} = [x_{cm} \ y_{cm}]^T$	Position of object's center of mass
ω	Rotation speed of object
W, m	Weight and mass of object
K_s	Cell support spring constant
Δz_i	Vertical deflection of i th cell under load
a, b, c	Parameters defining plane of object bottom
\vec{N}_{abc}	Normal force vector augmented with plane parameters ($n + 3 \times 1$)
\vec{W}	Object weight and weight moments augmented with zeros ($n + 3 \times 1$)

A	Equilibrium and constitutive equation matrix ($n + 3 \times n + 3$)
B	Cell position matrix augmented with the ones vector ($3 \times n$)
$\vec{V}_i = [V_{ix} \ V_{iy}]^T$	Wheel velocities of i th cell
V	Wheel velocity matrix ($2 \times n$)
μ	Coefficient of friction between wheel and object
\vec{f}_i	Traction force from each cell (2×1)
\vec{f}	Net traction force on object (2×1)
k_s	Effective object spring constant matrix (2×2)
\vec{f}_o	Effective offset force on object (2×1)
τ_i, τ	Torque applied on object by i th cell and net torque applied on object
R_i, \vec{R}	First moment of velocity of i th cell and velocity moment vector ($1 \times n$)
$\mathcal{X}_i, \vec{\mathcal{X}}$	Second moment of position of i th cell and second moment of position vector ($1 \times n$)
τ_o	Effective offset torque on object
$\vec{k}_{s\tau}$	Linear torque variation constant (2×1)
f_{ox}, f_{oy}	Components of net offset force
$k_{sxx}, k_{sxy}, k_{syy}, k_{syy}$	Components of effective spring constant matrix
B	Structured cell position matrix ($6 \times 2n$)
D_x, D_y	Cell position diagonal matrices ($n \times n$)
\vec{X}_{cm_e}	Desired equilibrium position
\vec{V}_x, \vec{V}_y	Wheel velocity component vectors ($1 \times n$)
Q	Structured cell positions augmented with diagonal position matrices ($3 \times 2n$)
τ_d, τ_a	Desired and actual torque applied by kinematic rotational field
α	Kinematic rotational field scaling constant
θ	Orientation of object
J	Rotational inertia of object
$b(\vec{X}_{cm}(t), \theta)$	Nonlinear (variable) rotational damping gain
k_1, k_2	Nonlinearity bounding gains
Φ, ϕ	Generic and specific sector nonlinearities
D (k_1, k_2)	Disk used for circle criterion
K_{f_x}, K_{f_y}	Position loop gains
K_θ, K_θ'	Rotational loop gain and scaled gain

1. Introduction

Distributed manipulation systems induce motions on objects through the application of many external forces. The general task is to impart some desired motion to an object, either in space or restricted to a plane, using either a large number of independently controlled actuators acting at many points on the object or by exploiting the dynamics of some continuous medium that is in contact with the manipulated object. Many forms of distributed manipulation systems exist, including those based on vibrating plates, multiple mobile robots, actively controlled arrays of air jets, and planar micromechanical and macromechanical actuator arrays. By nature, such systems generally involve redundant actuation and, hence, provide tremendous manipulation power.

Many of the ideas in distributed manipulation stemmed from work in sensorless manipulation using standard robotic manipulators. Akella et al. (1996) and Erdmann (1995) provided a radical alternative to standard robotic manipulation by significantly simplifying the robot manipulator and developing manipulation algorithms for these low-degree-of-freedom, sensorless, "minimalistic" mechanisms. Goldberg (1993) developed an algorithm that orients to symmetry a part with a sequence of parallel-jaw gripper open, close, and rotation operations without sensor information. Each operation in this sequence is termed a *squeeze*.

Böhringer et al. (1994) applied this type of sensorless manipulation to an array of microelectromechanical actuators that manipulated very small objects. Whereas Goldberg used a single manipulator to orient objects, Böhringer et al. used distributed actuation, where all actuators in the plane apply a force directed toward a "squeeze line" dividing the array. This sensorless "squeeze field" has a similar effect as a gripper squeeze for orienting objects.

Since then, researchers have examined many forms of distributed manipulation, encompassing all ranges of scales. On the macroscopic scale, node lines on transversally vibrating plates act like squeeze fields for open-loop manipulation (Böhringer, Bhatt, and Goldberg 1995). In-plane vibration of plates has been used for closed-loop manipulation of objects using vision systems for feedback (Reznik, Moshkoich, and Canny 1999; Frei et al. 1999). An array of foot-like mechanisms was built to manipulate large, light objects (Yim and Berlin 1999). An array of motorized wheels operating under distributed control manipulates large, heavier objects (Luntz, Messner, and Choset 2000). On a slightly smaller scale, arrays of controllable air jets were used to manipulate paper (Yim and Berlin 1999). An array of small, soft, ciliary, electroactive polymer gel actuators was used to manipulate small plates floating in water (Tadokoro and Takamori 1999). On a near-microscopic scale, both electrostatic vibratory and thermal biomorph microelectromechanical arrays have been used to move very small objects (Böhringer et al. 1994; Suh et al. 1999).

Much of the prior work in distributed manipulation dealt with the parts-feeding problem: manipulating a single object to a single static position and orientation using a static, planar, programmable manipulation “field.” Although this task represents only a small part of the capabilities of a distributed manipulation system, it is the focus of initial study because it provides a basis for more advanced tasks such as manipulating many objects along arbitrary paths. These analyses typically searched for stable equilibria in the configuration space of the manipulated object. A continuous field approximation of microactuated actuator arrays allows the applied forces to be integrated over the object, “lifting” the distributed force field to a net force/moment field acting on the geometric center of the object, where conservative distributed force fields provide conservative lifted force fields (Böhringer, Donald, and MacDonald 1996). This sort of analysis has been used to study a variety of classes of fields, including squeeze fields, providing a large but finite number of equilibria; elliptical potential fields (Böhringer et al. 1999), providing two stable equilibria for most parts; and a combination of a radial and uniform fields (Lamiroux and Kavraki 2000), providing a unique equilibrium for most parts.

This prior work applies to the concept of a general programmable vector field, the implementation of which is a microactuated array of actuators assumed to be a continuous medium. This paper focuses on the use of macro-scale actuator arrays to perform similar tasks. An actuator array performs distributed manipulation using a planar array of many small stationary elements (which are called cells) that cooperate to manipulate larger objects. Each object lies on many supporting cells 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 object’s motion. Through proper coordination, the actuator array induces arbitrary translation and rotation on objects that ride on top of the array. Because actuation is distributed, each of many objects can be manipulated independently, appearing as if each object were carried separately by a mobile vehicle (Fig. 1). The modular distributed manipulator system (MDMS) built by the authors is a macro-scale actuator array comprising an array of motorized wheels. Macro-scale arrays (and micro-scale arrays manipulating very small parts) have relatively few actuators manipulating an object simultaneously, and hence the actuation cannot be assumed continuous. These systems require the explicit modeling of the discrete forces in the system. As we will show in this paper, in cases where a relatively small number of cells support an object, equilibria mechanics of the object do not behave as predicted by continuous modeling assumptions.

The remainder of this section describes the prototype actuator array built by the authors and the operation and application of such systems, as well as some prior work done by the authors. Section 2 examines the dynamics of manipulation

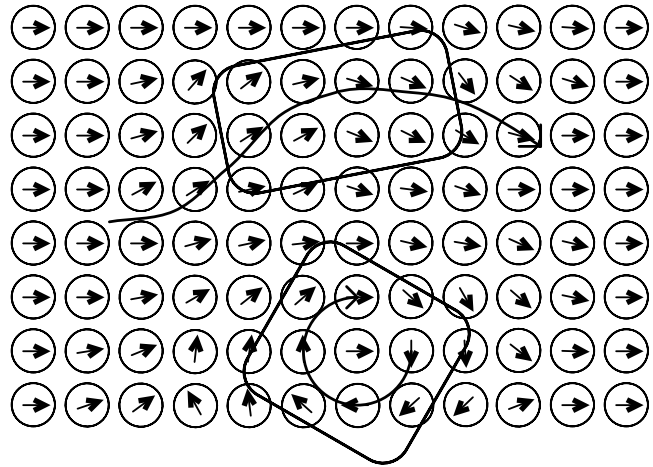


Fig. 1. The modular distributed manipulator system. Several objects can be independently translated and rotated.

of this discrete actuator array accounting for the distribution of load and the specific action of the spatially discrete actuators. We find that even though the system is a combination of many discrete points of contact, under certain modeling assumptions the dynamics of object manipulation simplify to an extremely concise form. Section 3 derives open-loop (sensorless) control strategies based on the dynamic model and analyzes their behavior. We find that certain objects are not orientable on a discrete actuator array that would otherwise be orientable on a continuous actuator array, demonstrating the importance of discrete modeling. The open-loop behavior is verified both through simulation (which exploits the discrete nature of the dynamics to gain efficiency) and through experiment on the prototype. Orientability problems in discrete open-loop manipulation motivate the use of closed-loop strategies that are derived in Section 4, also based on the dynamic model. These strategies reduce the many-input, three-output control problem to a much simpler three-input, three-output control problem and prove the stability of this approach. Concluding remarks are made in Section 5. The multimedia extensions include the complete simulator, implemented in Matlab and Simulink, and video of the MDMS in operation.

1.1. Prototype System

We built a prototype MDMS in which each cell consists of a pair of orthogonally mounted motorized roller wheels (Figs. 2 and 3). These wheels are capable of producing a force perpendicular to their axes while allowing free motion parallel to their axes. The combined action of the two roller wheels effects force in any planar direction to an object resting on top of the array. Each of these cells is connected to a large breadboard-style base (Fig. 4) to create a regular array of manipulators. Currently, we have 18 cells that can be arranged

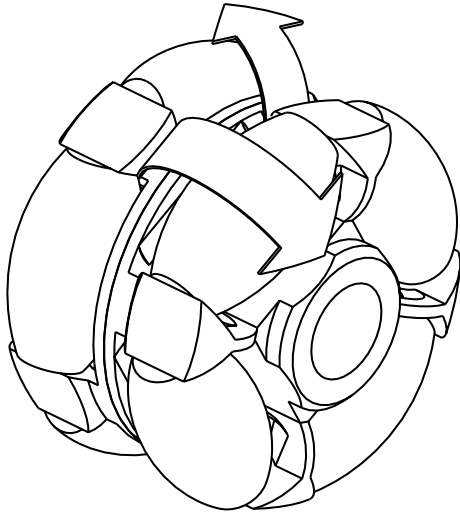


Fig. 2. A roller wheel that allows free motion along its axis of rotation.

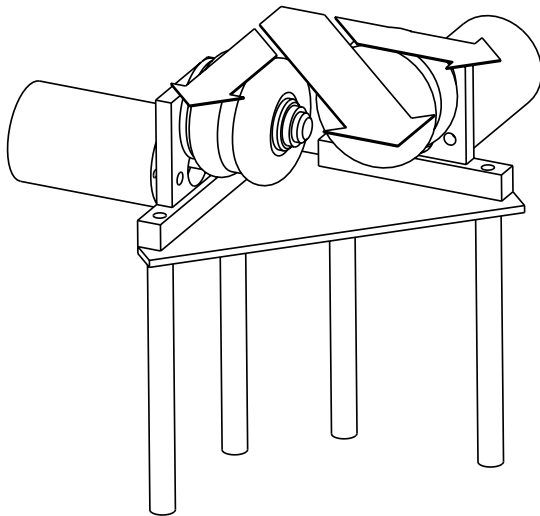


Fig. 3. A prototype cell made up of a motorized pair of roller wheels mounted orthogonally.

arbitrarily in a two-dimensional grid. A photograph of the prototype MDMS is shown in Figure 5 (see Extension 1¹).

Each cell is controlled by an inexpensive single-board computer based on the MC68HC11 microcontroller. We are exploring the use of distributed control on the MDMS because we believe that it would become impractical or impossible for a single centralized controller to control hundreds or possibly thousands of cells. Each cell communicates with its four neighboring cells, allowing messages to be passed along the array. In addition, each cell contains one (or more) binary sensors to detect the presence of an object. Figure 6 shows this control architecture.

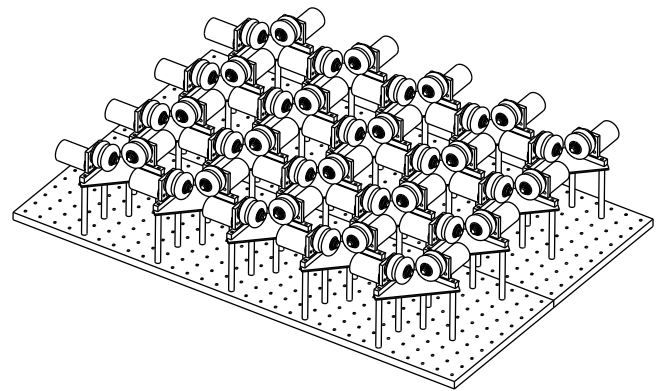


Fig. 4. Two-dimensional array of cells arranged in a grid.

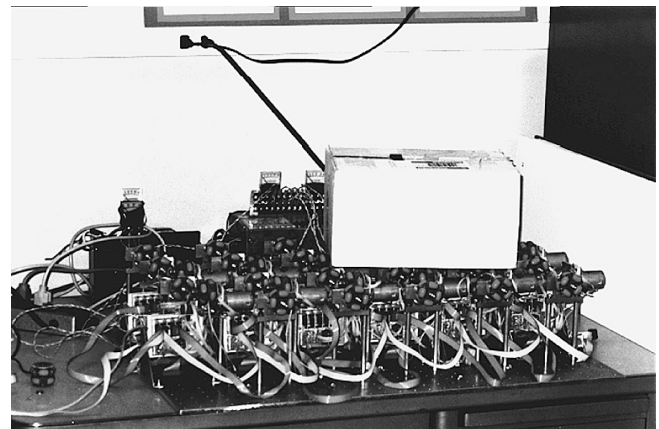


Fig. 5. Current experimental setup. A cardboard box rests on top.

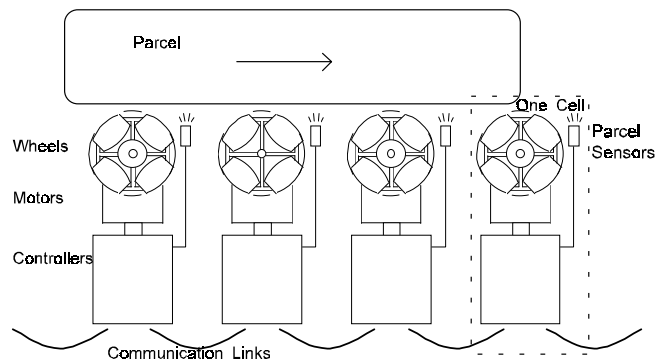


Fig. 6. A few manipulator cells carrying an object. Neighboring cells can communicate.

1. Please see the Index to Multimedia Extensions at the end of this article.

1.2. Operation and Modeling of Actuator Arrays

For many (macro-scale) applications, a dedicated robot or conveyor belt is the simplest and most appropriate solution. There are cases, however, such as reconfigurable manufacturing and complex sorting applications, that can benefit from features such as flexibility, modularity, redundancy, and reconfigurability provided by an actuator array. Actuator arrays may operate in conjunction with traditional robots and conveyor belts to form hybrid systems. For example, in airport baggage handling, long conveyors transport bags over long distances while actuator arrays at conveyor junctions sort and redirect baggage traffic. In flexible manufacturing, the actuator arrays can transport objects between robot workspaces where simple robots are used for object fixturing. The realization of this potential requires an understanding of this sort of distributed manipulation and the ability to generate useful manipulation strategies.

Two modes of operation may be used to manipulate objects with an actuator array: passive (open loop) and active (closed loop). In a passive mode, the action of the cells is pre-programmed and constant, establishing a force field in which the object moves. In an active mode, the array uses information about the object's motion to update the action of the cells. Local sensors at each cell or some global sensor, such as a vision system, provide such information, and in either case, real-time communication either to or between cells is generally necessary. Passive modes are useful for object positioning and orientation tasks with simple maneuvers that do not require extreme precision. It is usually simpler and less expensive to implement an open-loop mode because no feedback or control is necessary. Active modes are generally useful for more precise motions or more complex motions involving multiple objects.

For an open-loop system, it is generally desirable to scale the actuator array so that very large numbers of cells support an object. This provides an approximately continuous medium on which the object can rest, resulting in smoother and more predictable motion. In the limit of very small cells placed closely together, the array does in fact become a continuous medium providing a true force field. If the force field is designed to be conservative, potential field theory analytically predicts the motion of an object (Kavraki 1997). This is a reasonable assumption for a microscopic (microelectromechanical) scale system.

Unfortunately, it is usually impractical to construct a system on a macroscopic scale with enough actuators to justify the continuous medium approximation. The manipulated objects must be supported by relatively smaller numbers of cells. This requires that the discrete nature of an actuator array be taken into account when designing open-loop fields and when predicting the motion of objects. Therefore, for full functionality of the MDMS, and actuator arrays in general, a thorough understanding of the dynamics of object manipulation on a

discrete array is necessary. Such an understanding enables the design of static force and velocity fields and the design of active manipulation strategies to transport and manipulate objects arbitrarily.

1.3. Prior Work by the Authors

The authors first began to model the discrete dynamics of manipulation in one dimension along a single row of cells (Luntz, Messner, and Choset 1997a), where they found that the object experiences piecewise constant linear dynamics as it moves along the array. In this work (and the other work leading up to this paper), the discrete nature of the system was modeled explicitly, taking into account the distribution of weight of the object among the cells and the specific interaction between each wheel and the object.

That work was extended into two dimensions (plus rotation) in a following paper (Luntz, Messner, and Choset 1997b), which is a precursor to the work presented here. Much of the modeling in its final form presented here is presented in another precursor paper (Luntz, Messner, and Choset 1998b), although this paper presents two-dimensional modeling results for several friction modes. The authors presented work on modeling nonlinear friction modes, including mixed stick-slip friction modes (in one dimension), in another paper (Luntz, Messner, and Choset 1998a) along with a rudimentary simulator. The simulator presented here is much more efficient because it takes advantage of the piecewise nature of the dynamics.

In addition to the dynamics of manipulation on the MDMS, the authors have examined other practical and dynamic issues of the MDMS. Further details on the MDMS prototype as well as simple tests of the system's functionality can be found in previous papers by the authors (Luntz and Messner 1995, 1996). The authors have also developed and tested a distributed network and a communications language for the efficient passing of messages between cells (Luntz and Messner 1997). Videos demonstrating the functionality of the MDMS prototype are included in Extensions 5 and 6.

2. Modeling

To generate discrete distributed manipulation strategies (the goal of this paper), we first must have a dynamic model. We compute the net force and moment acting on an object as it moves along the actuator array. On the MDMS, the force and moment are functions of the position of the object, the speed at which the object is moving, the set of cells supporting the object, and the speeds of the wheels in those cells. The basic approach examines the situation in which the object rests on a particular arbitrary set of cells, computes the forces as a function of object position and motion, and recomputes if the set of supporting cells changes. First, we discuss the interaction

between the wheels and the object. Then, we compute the forces acting on the object.

The following are the assumptions used for modeling:

- Each orthogonally oriented pair of wheels acts as a single support.
- Supports act as springs to support the object.
- Vertical and tipping motions of the object are in pseudo-equilibrium.
- The bottom of the object is flat.
- The speed of each wheel is constant.
- Horizontal force between each wheel and the object is due to either rolling or sliding friction.
- The friction is modeled by one of several simple laws discussed later.

To compute the horizontal translation and rotation dynamics of the object, first we use the vertical equilibrium of the object and constitutive relations for the supports to compute the normal forces supporting the object. Then, we apply a friction model between the object and wheels to determine the traction force from each cell. These traction forces are summed over all cells supporting the object, and the net force and torque acting on the object are functions of the object's position and motion. The structure of the mathematics and several convenient mathematical identities simplify the problem considerably.

Notation. Scalar variables will appear in normal math font (e.g., s), vectors will appear in normal math font with an arrow (e.g., \vec{v}), and matrices will appear in bold font (e.g., \mathbf{m}). Subscripts x and y will indicate x and y components, and subscript i will indicate the i th cell. Vectors with x and y components are generally column vectors, and vectors with a component for each cell are generally row vectors. Variables with both x and y components and components for each cell are represented as matrices. For example, \mathbf{V} is a matrix made up of velocity (column) vectors \vec{V}_i for each cell, with components V_{x_i} and V_{y_i} . \mathbf{V} can also be said to be made up of component (row) vectors \vec{V}_x and \vec{V}_y listing the velocity components for all the cells.

2.1. Wheel-Object Interaction

A model of the contact between the wheels and object riding on the array is essential for modeling the dynamics of object manipulation on the MDMS. In the model, each wheel applies a force in the plane to the object through friction between the

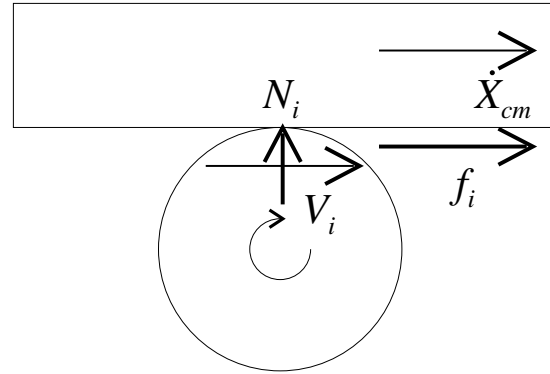


Fig. 7. Interaction between wheel and object. The rotating wheel supports the object and provides a traction force.

wheel and the object (see Fig. 7). There are several possible models of this friction force.

Viscous Friction. The wheel slides on the object. The friction force is proportional to the normal force between the wheel and the object and to the difference between the wheel's speed and the speed (in the direction of the wheel) of the surface of the object at the wheel. The roller wheels used in the prototype system ensure that the friction will only occur in the direction perpendicular to the wheel's axis. The assumption is that the motor is strong enough to maintain its speed regardless of the traction force.

Coulomb Friction. The wheel slides on the object. The friction force is proportional to the normal force between the wheel and the object. The direction of the friction will be the same as that in the viscous case. The assumption is that the motor is strong enough to maintain relative motion between the wheel and the object.

No-Slip Friction. The wheel rolls on the object. The traction force is not a function of the normal force, and there is no relative motion (in the direction perpendicular to the wheel's axis) between the wheel and the object. As the wheels rotate to follow the motion of the object, the motors interact, back-driving each other, and the traction force from each wheel depends on the properties of the motors and the voltage applied to the motors.

We will examine the dynamics of the object mainly under a viscous friction model, because the computation of the normal forces most closely matches the observed qualitative behavior of the prototype system (demonstrated most clearly in the videos in Extension 5). We will discuss the effects of the other friction models, although details of their derivation are omitted. Because the sliding friction modes (viscous and Coulomb) depend on the normal forces at each wheel, we must first compute the normal forces as functions of the object's position. This is the topic of the next section.

2.2. Supporting Forces (normal forces)

Consider the case in which n cells support an object. Approximating the two-wheel support of each cell as a single-point support, we need to solve for n forces supporting the object when the object is vertically in pseudoequilibrium. These n point supports are arranged arbitrarily in the x - y plane, having coordinates as entries of the matrix

$$\mathbf{X} = \begin{bmatrix} \vec{x} \\ \vec{y} \end{bmatrix} = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix} = [\vec{X}_1 \dots \vec{X}_n]. \quad (1)$$

An object whose center of mass is located at $\vec{X}_{cm} = [x_{cm} \ y_{cm}]^T$ resting on n of these cells is supported by vertical normal forces

$$\vec{N} = [N_1 \ \dots \ N_n]. \quad (2)$$

The object must be in equilibrium in the vertical direction and in rotation about the x and y axes. The vertical equilibrium of the object, with weight W , requires that

$$\sum_{i=1}^n N_i = W = [1 \ \dots \ 1] \vec{N}^T. \quad (3)$$

Rotational equilibrium about the x and y axes requires that the moments induced by the normal forces about any point (in this case, the arbitrarily located origin of our coordinate system) sum to the moment about this point induced by the weight of the object. The moment equilibrium about the x and y axes, respectively, requires that

$$\sum_{i=1}^n N_i y_i = \vec{y} \vec{N}^T = W y_{cm}, \quad \text{and} \quad (4)$$

$$\sum_{i=1}^n N_i x_i = \vec{x} \vec{N}^T = W x_{cm}. \quad (5)$$

At this point in the development, there are n unknowns (the elements of \vec{N}) but only three equations ((3), (4), and (5)) from equilibrium. To solve for the remaining $n - 3$ forces, we must consider flexibility in the system. A simple physical model of this flexibility is shown in side view in Figure 8. Each support is assumed to be a spring, with Hooke's law ($N_i = K_s \Delta z_i$) representing the compression of the i th cell under a normal load. Physically, this flexibility is equivalent to a flexible suspension under each wheel, such as the next generation of our hardware array will include. In the current system, an equivalent flexibility is apparent in the bottom of the object, either in the cardboard of a box or a foam pad on which the box rests.

Assuming the bottom of the box is nominally flat, the flexible cells conform to the bottom of the box to distribute the weight. All the cells (under the object) lie in this plane. A representation of the plane of the bottom of the object is

$$z = a'y + b'x + c'. \quad (6)$$

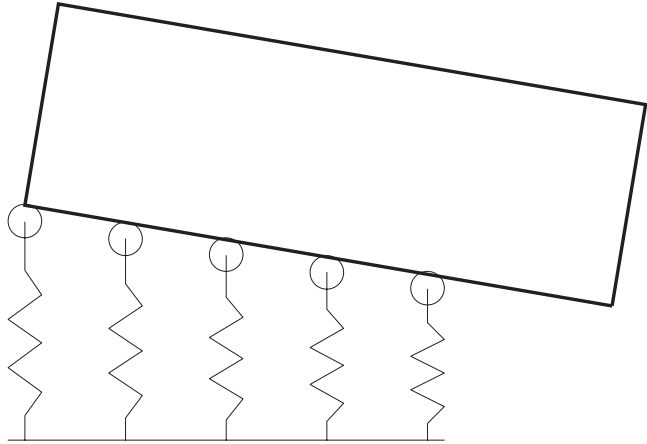


Fig. 8. Flexible supports in the many-cell case.

Each cell will obey the displacement relation

$$\begin{aligned} \Delta z_i = N_i / K_s &= a'y_i + b'x_i + c' \\ \Rightarrow N_i + ay_i + bx_i + c &= 0. \end{aligned} \quad (7)$$

Equations (3), (4), and (5), along with n instances of eq. (7) supply $n + 3$ equations and $n + 3$ unknowns (n N_i 's and three plane parameters, a , b , and c). The matrix form of this system of equations is

$$\underbrace{\begin{bmatrix} 1 & \dots & 0 & | & 1 & x_1 & y_1 \\ \vdots & \ddots & \vdots & | & \vdots & \vdots & \vdots \\ 0 & \dots & 1 & | & 1 & x_n & y_n \\ \hline 1 & \dots & 1 & | & 0 & 0 & 0 \\ x_1 & \dots & x_n & | & 0 & 0 & 0 \\ y_1 & \dots & y_n & | & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} N_1 \\ \vdots \\ N_n \\ c \\ b \\ a \end{bmatrix}}_{\vec{N}_{abc}^T} = \underbrace{\begin{bmatrix} 0 \\ \vdots \\ 0 \\ W \\ Wx_{cm} \\ Wy_{cm} \end{bmatrix}}_{\vec{W}}. \quad (8)$$

The top portion of this equation represents the n instances of the compatibility equation (eq. (7)). The bottom three rows represent the vertical, x -moment, and y -moment equations (eqs. (3), (4), and (5)). Solving this equation for \vec{N}_{abc} (which contains \vec{N} and a , b , and c) results in

$$\vec{N}_{abc}^T = \mathbf{A}^{-1} \vec{W}. \quad (9)$$

Because \mathbf{A} is symmetric and very structured, its inverse is easy to compute. Define the matrix \mathbf{B} as

$$\mathbf{B} = \begin{bmatrix} 1 & \dots & 1 \\ x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix}. \quad (10)$$

The block form of the matrix \mathbf{A} is

$$\mathbf{A} = \left[\begin{array}{c|c} \mathbf{I}_{n \times n} & \mathbf{B}^T \\ \hline \mathbf{B} & \mathbf{0}_{3 \times 3} \end{array} \right]. \quad (11)$$

The inverse of the matrix \mathbf{A} exists if \mathbf{B} has rank 3. \mathbf{B} has rank 3 if and only if all the cells under the object are not colinear. The expression for the inverse is

$$\mathbf{A}^{-1} = \left[\begin{array}{c|c} \mathbf{I}_{n \times n} - \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} & \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \\ \hline (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} & -(\mathbf{B}\mathbf{B}^T)^{-1} \end{array} \right]. \quad (12)$$

(The reader is invited to verify the form of the inverse by multiplying \mathbf{A} by \mathbf{A}^{-1} .)

Because \vec{W} only multiplies nonzero elements into the right side of \mathbf{A}^{-1} , and because only the slopes (a , b , and c) result from the lower portion of \mathbf{A}^{-1} , only the upper right partition of \mathbf{A}^{-1} is used to calculate \vec{N} . The expression for \vec{N} is then

$$\begin{aligned} \vec{N}^T &= \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \begin{bmatrix} W \\ Wx_{cm} \\ Wy_{cm} \end{bmatrix} \\ &= \mathbf{W}\mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \vec{X}_{cm} \right). \end{aligned} \quad (13)$$

Note that there may be cases in which computation results in negative values for one or more of the normal forces in \vec{N} . Because the object cannot actually “pull” on the springs in the support model, the object lifts off these cells, and the other normal forces must be recalculated with the negative supports removed from the set of supports. This occurs, however, only in exceptional cases, and we will ignore this effect from now on (its only effect is to change the set of supports).

2.3. Translational Dynamics

We will now represent the planar dynamics of an object resting on the array as a net planar force and torque acting on the object as a function of the object’s position, linear velocity, and rotational velocity. We will first apply the viscous friction model to derive the planar forces from the supporting forces (see Fig. 7).

Under the viscous friction assumption, the horizontal force from each cell \vec{f}_i is proportional to a coefficient of friction μ , the normal force N_i exerted by the cell, and the vector difference between the velocity of the wheel and the velocity of the object at the point of the cell. This velocity difference is a function of the translational velocity of the object, $\dot{\vec{X}}_{cm}$, the velocity of the wheel, \vec{V}_i , the rotation speed of the object about its center of mass, ω , and the position difference between the cell and the center of mass, $\vec{X}_i - \vec{X}_{cm}$:

$$\vec{f}_i = \mu \left(\vec{V}_i - \dot{\vec{X}}_{cm} + \omega \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\vec{X}_i - \vec{X}_{cm}) \right) N_i. \quad (14)$$

The total horizontal force is the sum of \vec{f}_i over all the cells. To simplify notation, the wheel velocity matrix, \mathbf{V} , is defined as

$$\mathbf{V} = \begin{bmatrix} V_{1x} & V_{2x} & \cdots & V_{nx} \\ V_{1y} & V_{2y} & \cdots & V_{ny} \end{bmatrix}. \quad (15)$$

Inner-product notation eliminates the explicit summation, and the expression for the net horizontal force vector is

$$\begin{aligned} \vec{f} &= \sum_{i=1}^n \vec{f}_i = \mu \left(\mathbf{V} - \dot{\vec{X}}_{cm} [1 \dots 1] \right. \\ &\quad \left. + \omega \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\mathbf{X} - \vec{X}_{cm} [1 \dots 1]) \right) \vec{N}^T \\ &= \mu \mathbf{V} \vec{N}^T - \mu \dot{\vec{X}}_{cm} [1 \dots 1] \vec{N}^T \\ &\quad + \omega \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\mathbf{X} \vec{N}^T - \vec{X}_{cm} [1 \dots 1] \vec{N}^T). \end{aligned} \quad (16)$$

Note that $[1 \dots 1] \vec{N}^T$ is the sum of the normal forces, which is the object weight, W . Therefore, the last two terms in the parentheses are identically zero from eqs. (4) and (5). Hence, the net horizontal force is not a function of the object’s rotation speed and eq. (16) becomes

$$\vec{f} = \mu \mathbf{V} \vec{N}^T - \mu \dot{\vec{X}}_{cm} W. \quad (17)$$

The second term in eq. (17) is a linear damping term, which is always dissipative because weight is positive. The result of substituting \vec{N} from eq. (13) into eq. (17) is

$$\begin{aligned} \vec{f} &= \underbrace{\mu \mathbf{W}\mathbf{V}\mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{k}_s} \vec{X}_{cm} \\ &\quad + \underbrace{\mu \mathbf{W}\mathbf{V}\mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}_{\vec{f}_o} - \mu \dot{\vec{X}}_{cm} W \\ &= \mathbf{k}_s \vec{X}_{cm} + \vec{f}_o - \mu W \dot{\vec{X}}_{cm}, \end{aligned} \quad (18)$$

where \mathbf{k}_s is a constant 2×2 matrix and \vec{f}_o is a constant 2×1 vector. The matrix \mathbf{k}_s is essentially a matrix of spring constants because it specifies force as a linear function of position. The vector \vec{f}_o is an offset force. The translational dynamics of the object are piecewise constant and linear (as the object changes supports). Therefore, it is easy to predict the local translational motion of the object.

2.4. Rotational Dynamics

In two dimensions, the torque each cell applies to the object is the scalar cross product of the position vector of the point of application of the force, \vec{X}_i , relative to the object’s center of mass, \vec{X}_{cm} , and the horizontal force vector from that point, \vec{f}_i . Applying the matrix definition of scalar cross products, and taking \vec{f}_i from eq. (14) (viscous friction), the expression for the torque applied by the i th cell is

$$\begin{aligned}
\tau_i &= \mu (\vec{X}_i - \vec{X}_{cm}) \times \left(\vec{V}_i - \dot{\vec{X}}_{cm} + \omega \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\vec{X}_i - \vec{X}_{cm}) \right) N_i \\
&= \mu \vec{X}_i \times \vec{V}_i N_i - \mu \vec{X}_{cm} \times \vec{V}_i N_i - \mu (\vec{X}_i - \vec{X}_{cm}) \times \dot{\vec{X}}_{cm} N_i \\
&\quad + \omega \mu (\vec{X}_i - \vec{X}_{cm})^T \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} (\vec{X}_i - \vec{X}_{cm}) N_i. \quad (19)
\end{aligned}$$

To simplify notation, define $R_i = \vec{X}_i \times \vec{V}_i$, which defines the components of the vector \vec{R} , and $\mathcal{X}_i = \vec{X}_i^T \vec{X}_i$, which defines the components of the vector $\vec{\mathcal{X}}$. The product of cross-product matrices is $-\mathbf{I}_{2 \times 2}$ (the negative 2×2 identity matrix). In addition, use the fact that $\vec{X}_i^T \vec{X}_{cm} = \vec{X}_{cm}^T \vec{X}_i$, and that reversing the order of a cross product changes sign, to obtain

$$\begin{aligned}
\tau_i &= \mu R_i N_i - \mu \vec{X}_{cm} \times \vec{V}_i N_i + \mu \dot{\vec{X}}_{cm} \times (\vec{X}_i - \vec{X}_{cm}) N_i \\
&\quad - \omega \mu (\vec{X}_i - \vec{X}_{cm})^T (\vec{X}_i - \vec{X}_{cm}) N_i \\
&= \mu R_i N_i - \mu \vec{X}_{cm} \times \vec{V}_i N_i + \mu \dot{\vec{X}}_{cm} \times (\vec{X}_i - \vec{X}_{cm}) N_i \\
&\quad - \omega \mu (\mathcal{X}_i N_i - \vec{X}_{cm}^T \vec{X}_i N_i - \vec{X}_{cm}^T (\vec{X}_i - \vec{X}_{cm}) N_i). \quad (20)
\end{aligned}$$

Note that the term multiplying ω is positive because N_i and the product $(\vec{X}_i - \vec{X}_{cm})^T (\vec{X}_i - \vec{X}_{cm})$ are always positive.

The total torque is the sum of the τ_i over all the cells. Inner-product notation eliminates the explicit summation, and the expression for the total torque is

$$\begin{aligned}
\tau &= \sum \tau_i = \mu \vec{R} \vec{N}^T - \mu \vec{X}_{cm} \times \mathbf{V} \vec{N}^T \\
&\quad + \mu \dot{\vec{X}}_{cm} \times (\mathbf{X} \vec{N}^T - \vec{X}_{cm} [1 \dots 1] \vec{N}^T) \\
&\quad - \omega \mu (\vec{\mathcal{X}} \vec{N}^T - \vec{X}_{cm}^T \mathbf{X} \vec{N}^T \\
&\quad - \vec{X}_{cm}^T (\mathbf{X} \vec{N}^T - \vec{X}_{cm} [1 \dots 1] \vec{N}^T)). \quad (21)
\end{aligned}$$

Because $\mathbf{X} \vec{N}^T = W \vec{X}_{cm}$, and $[1 \dots 1] \vec{N}^T = W$ from object equilibrium (eqs. (3), (4), and (5)), $\mathbf{X} \vec{N}^T - \vec{X}_{cm} [1 \dots 1] \vec{N}^T = 0$ and the expression for torque reduces to

$$\tau = \mu \vec{R} \vec{N}^T - \mu \vec{X}_{cm} \times \mathbf{V} \vec{N}^T - \omega \mu (\vec{\mathcal{X}} \vec{N}^T - W \vec{X}_{cm}^T \vec{X}_{cm}). \quad (22)$$

The last term in this expression is $-\omega$ multiplied by a positive term (from eq. (20)) and, hence, is dissipative. Substituting the expression for normal forces from eq. (13) into eq. (22) results in an expression for the moments acting on the object

as a function of position and rotational speed:

$$\begin{aligned}
\tau &= \underbrace{\mu W \vec{R} \vec{B}^T (\mathbf{B} \mathbf{B}^T)^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}}_{\tau_o} \\
&\quad + \underbrace{\mu W \vec{R} \vec{B}^T (\mathbf{B} \mathbf{B}^T)^{-1} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\vec{k}_{s\tau}} \vec{X}_{cm} \\
&\quad + \vec{X}_{cm} \times (\vec{f}_o + \mathbf{k}_s \vec{X}_{cm}) - \mu \omega (\vec{\mathcal{X}} \vec{N}^T - W \vec{X}_{cm}^T \vec{X}_{cm}) \\
&= \tau_o + \vec{k}_{s\tau} \vec{X}_{cm} - (\vec{f}_o + \mathbf{k}_s \vec{X}_{cm}) \\
&\quad \times \vec{X}_{cm} - \mu \omega (\vec{\mathcal{X}} \vec{N}^T - W \vec{X}_{cm}^T \vec{X}_{cm}), \quad (23)
\end{aligned}$$

where \mathbf{k}_s and \vec{f}_o are the spring and offset constants from the translational dynamics (eq. (18)), $\vec{k}_{s\tau}$ is a 1×2 constant vector relating torque to position, and τ_o is a scalar constant torque. Note that nothing in the derivation involved the orientation of the object other than the set of supports; hence, although the object rests on a particular set of supports, torque on the object is not a function of orientation. This is very important for determining stable orientations of an object.

These expressions for net force and moment are purposely left in a coordinate-dependent form. In Sections 3 and 4, we fix different sets of coordinates and simplify these expressions for the appropriate coordinate system; open-loop manipulation uses an array-fixed coordinate frame whereas closed-loop manipulation uses an object-fixed coordinate frame.

2.5. Other Friction Modes

The previous derivation used only a viscous friction contact model. The derivation changes only slightly for certain other types of friction models. Here, we present the results of these derivations but omit the details.

2.5.1. Coulomb Friction

Assuming full sliding contact exists between the wheels and the object, with Coulomb friction, the magnitude of each component of force from each cell ($f_{i_x}^C$ and $f_{i_y}^C$) is proportional to the normal force at that cell. Note that we are using a superscript C to represent Coulomb friction. The signs of the force components are equal to the signs of the components of the velocity difference between the wheels and the point of the object at the wheel. However, full sliding contact can only occur if the wheels always spin faster than the linear speed of the object at the point of each wheel. Under this condition, the signs of the velocity difference components are always equal to the signs of the wheel velocities. Therefore, we can express the force from each cell under Coulomb friction (similar to eq. (14)) as

$$\vec{f}_i^C = \mu \text{sign}(\vec{V}_i) N_i. \quad (24)$$

Summing this over all the cells supporting the object, the total force is of the form

$$\vec{f}^C = \mathbf{k}_s^C \vec{X}_{cm} + \vec{f}_o^C. \quad (25)$$

This represents the dynamics of a mass-spring system without damping. The spring constants and offset force are different from those obtained from a viscous friction model.

The expression for the net torque under Coulomb friction has the form

$$\tau^C = \tau_o^C + \vec{k}_{s_r}^C \vec{X}_{cm} - (\vec{f}_o^C + \mathbf{k}_s^C \vec{X}_{cm}) \times \vec{X}_{cm}, \quad (26)$$

which is similar to the viscous case but without damping.

The viscous model provides a better representation of the real prototype system than the Coulomb model for two reasons. First, damped oscillations occur in the prototype system even though the damping is not necessarily linear (see the first video in Extension 5). Second, the traction force in the prototype is observed to be dependent on the speed of the wheels, which is not represented by the Coulomb model. For example, the equilibrium position of a cardboard box resting on cells with opposing motion changes with the speeds of the wheels. Of course, because the contact between the wheels is actually a dry contact, the friction is not truly viscous. The velocity dependence of the traction force may come about from the ‘‘bumpiness’’ of the roller wheels, so the contact is not truly sliding. Rather, the contact is actually repeated impacts from the rollers on the wheels. A higher impact frequency provides a stronger average traction force.

2.5.2. No-Slip Friction

If the wheels never slip on the object, the speed of the wheels is equal to the speed of the object, and the object back-drives the wheels. Thus, the dynamics of the motors become important. A previous paper by the authors (Luntz, Messner, and Choset 1998a) derived the no-slip case in one dimension. That derivation employed a first-order model of a motor under proportional feedback speed control to find the following relationship (e.g., in the x -direction) between wheel speed, V_i , reference speed, r_i , and traction force, \vec{f}_i^N :

$$f_{i_x}^N = \underbrace{r_{i_x} \frac{K_m G}{R_m}}_{f_{o_{i_x}}^N} - V_{i_x} \underbrace{\left(B_m + \frac{K_m^2}{R_m} + \frac{K_m G}{R_m} \right)}_{b^N}, \quad (27)$$

where R_m , B_m , K_m , and G are the motor’s coil resistance, damping, torque (and back electromotive force) constant K_m , and proportional feedback gain, respectively. Note that the

wheel’s radius and gear reduction are lumped into the motor constants.

The expression for the total force (rotational dynamics for no-slip contact are not presented in this paper) acting on the object in two dimensions is of the form

$$\vec{f}^N = \vec{f}_o^N - B^N \dot{\vec{X}}_{cm} - \omega b^N \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \left(\sum \vec{X}_i - n \vec{X}_{cm} \right). \quad (28)$$

Note that we use a superscript N to indicate no-slip contact. Under no-slip contact, the object experiences a piecewise constant force with translational damping and an additional force due to rotation on an off-centroid set of supports. This additional term will dissipate as the rotation of the object is damped, so no energy is added to the system by this term. Note that the equivalent of this term in the other contact cases was identically zero; the supporting forces at each cell and the drag forces at each cell due to the object’s rotation were proportional to each other because the moment arms used in determining rotational equilibrium about the x and y axes are the same as the components of the radii of rotation from the center of the object to the cell. (The moment arms used in determining rotational equilibrium about the x and y axes are the same as the components of the radius of rotation from the center of the object to each cell.)

Essentially, in this case, each cell applies a constant force plus damping to the object, which is closer to the general assumptions made by Böhringer and Kavraki for microactuated systems. Under these translational dynamics, there is no unique equilibrium position because the force is not a direct function of the object’s position while the object rests on a particular set of supports. Because of this, with no-slip contact, the equilibria are not precise, and the location at which the object comes to rest depends on the initial state of the object. In the prototype, the wheels actually do slip on the object, and unique equilibria exist (as seen in repeated trials in the first and third videos of Extension 5). Therefore, we employ the viscous friction model rather than the no-slip friction model.

The dynamics provided by the no-slip model are not linear, nor are they as useful for open-loop manipulation as those resulting from the viscous model. Therefore, to ensure that the wheels do slip on the object at constant speed, a proportional plus integral feedback controller is used on the cell’s motors to maintain a constant speed.

Qualitative observations show that the viscous friction model both better predicts the behavior of the prototype system and provides for a mode of operation better suited for open-loop object manipulation than the Coulomb and no-slip contact models. In the following sections, we will make use of the viscous friction model to design useful manipulation strategies in both open and closed loop.

3. Open-Loop Manipulation

Using the model presented in the previous section, consider now the design of open-loop (passive) manipulation strategies. The philosophy behind passive strategies on the MDMS establishes a class of constant velocity fields that manipulate an object to a desired position and orientation for any starting position and orientation without feedback. The field should be independent of the size and shape of the object, although discreteness induces a restriction to a class of objects. This sensorless approach parallels the methods of researchers examining microelectromechanical actuator arrays using continuous approximations.

The design of open-loop fields relies on an inversion of the dynamic relationship developed in Section 2 where specified desired dynamics determine the set of wheel speeds. A set of nine constants describes the mass-spring-damper-like dynamics of an object as it rests on a single set of supports. Although these dynamics are only piecewise constant, the velocity field design process presented here will produce mass-spring-damper behavior with uniform desired properties over the entire array. The goal is that the translational dynamics of the object be such that the piecewise spring, damping, and offset constants are uniformly constant over the entire array (i.e., invariant with respect to the set of supports), in effect smoothing out the discontinuities. In particular, the specification of an equilibrium position and return-spring stiffnesses leads to the design of an open-loop field.

This design process is an example of a general method of designing manipulation fields. The inversion of the dynamics can produce fields other than that designed in this section, including fields useful for closed-loop manipulation derived in Section 4.

3.1. Design of Open-Loop Manipulation Fields

With a viscous friction model, eq. (18) states that the translational forces acting on an object are described by

$$\vec{f} = \mathbf{k}_s \vec{X}_{cm} + \vec{f}_o - \mu W \dot{\vec{X}}_{cm}. \quad (29)$$

This expression consists of six constants with which we can specify the dynamics of the motion of the object: four spring constants in \mathbf{k}_s and two constant offset forces in \vec{f}_o . Designing the field reduces to computing wheel speeds based on the selected values of these six constants. Consider first the four constants from the 2×2 matrix \mathbf{k}_s . The symmetric component of this matrix corresponds to an inward-pointing component of the field, whereas the skew-symmetric component corresponds to a circulatory field about the origin (see Fig. 9). Because this rotational motion is undesirable, designed fields should only use symmetric \mathbf{k}_s . In addition, without loss of generality, the development uses only diagonal \mathbf{k}_s (symmetric but nondiagonal \mathbf{k}_s can be transformed to a diagonal \mathbf{k}_s through a simple change of coordinate system).

Moving an object to a particular position requires the specification of the equilibrium position \vec{X}_{cm_e} and return-spring strengths $k_{s_{xx}}$ and $k_{s_{yy}}$ (the diagonal terms of \mathbf{k}_s). Physically, this is like the mass-spring-damper system shown in Figure 10. At equilibrium, $\vec{f}_o = -\mathbf{k}_s \vec{X}_{cm_e}$, in effect specifying \vec{f}_o . The design problem then becomes the problem of determining wheel velocities \mathbf{V} given their positions (specified in \mathbf{X} and equivalently in \mathbf{B}) and the constants \mathbf{k}_s and \vec{f}_o .

Equation (18) defines the elements in \mathbf{k}_s and \vec{f}_o as

$$\begin{bmatrix} f_{o_x} & k_{s_{xx}} & k_{s_{xy}} \\ f_{o_y} & k_{s_{yx}} & k_{s_{yy}} \end{bmatrix} = \mu W \mathbf{V} \mathbf{B}^T (\mathbf{B} \mathbf{B}^T)^{-1}. \quad (30)$$

Rewrite this equation in terms of the vector (of length $2n$) formed by stacking the transposes of the two rows of \mathbf{V} (\vec{V}_x^T and \vec{V}_y^T). Taking advantage of the symmetry of $\mathbf{B} \mathbf{B}^T$,

$$\begin{bmatrix} f_{o_x} \\ k_{s_{xx}} \\ k_{s_{xy}} \\ f_{o_y} \\ k_{s_{yx}} \\ k_{s_{yy}} \end{bmatrix} = \mu W \underbrace{\begin{bmatrix} (\mathbf{B} \mathbf{B}^T)^{-1} \mathbf{B} & \mathbf{0} \\ \mathbf{0} & (\mathbf{B} \mathbf{B}^T)^{-1} \mathbf{B} \end{bmatrix}}_{\mathcal{B}} \begin{bmatrix} \vec{V}_x^T \\ \vec{V}_y^T \end{bmatrix}. \quad (31)$$

This produces a set of six equations and $2n$ unknowns.

Inverting the previous set of equations produces a solution for the wheel speeds (\vec{V}_x and \vec{V}_y). Because this set of



Fig. 9. A decoupled field (left) with symmetric \mathbf{k}_s and a circulatory field (right) with skew-symmetric \mathbf{k}_s .

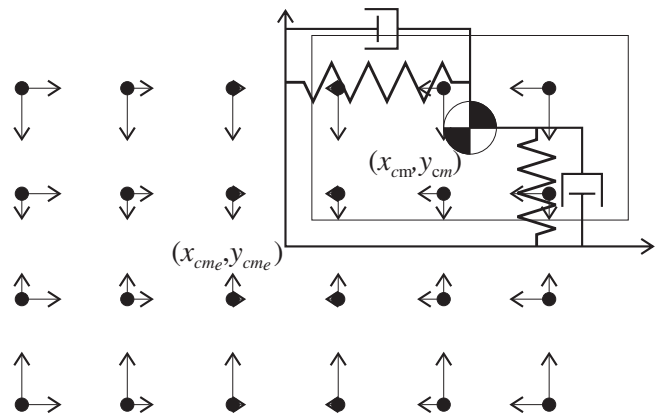


Fig. 10. With $k_{s_{xy}} = k_{s_{yx}} = 0$, the object behaves like a mass-spring-damper system on a discrete elliptic field.

equations is underconstrained, some freedom in the solution exists, requiring further constraints to select one solution from the infinity of possible solutions. The Penrose pseudoinverse accomplishes this by minimizing the sum of the squares of the wheel speeds. For the nonsquare matrix \mathcal{B} , the Penrose pseudoinverse \mathcal{B}^\dagger is defined as $\mathcal{B}^\dagger \equiv \mathcal{B}^T (\mathcal{B}\mathcal{B}^T)^{-1}$. Rewrite the pseudoinverse, \mathcal{B}^\dagger , in terms of \mathbf{B} and use the properties of matrix inverses to obtain

$$\begin{aligned} \mathcal{B}^\dagger &= \mathcal{B}^T (\mathcal{B}\mathcal{B}^T)^{-1} \\ &= \mathcal{B}^T \left(\left[\begin{array}{c|c} (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} & \mathbf{0} \\ \hline \mathbf{0} & (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} \end{array} \right] \right. \\ &\quad \left. \left[\begin{array}{c|c} \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1T} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1T} \end{array} \right] \right)^{-1} \\ &= \mathcal{B}^T \left(\left[\begin{array}{c|c} (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} & \mathbf{0} \\ \hline \mathbf{0} & (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \end{array} \right] \right)^{-1} \\ &= \left[\begin{array}{c|c} \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1T} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1T} \end{array} \right] \\ &\quad \left(\left[\begin{array}{c|c} (\mathbf{B}\mathbf{B}^T)^{-1} & \mathbf{0} \\ \hline \mathbf{0} & (\mathbf{B}\mathbf{B}^T)^{-1} \end{array} \right] \right)^{-1} \\ &= \left[\begin{array}{c|c} \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \end{array} \right] \left[\begin{array}{c|c} \mathbf{B}\mathbf{B}^T & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}\mathbf{B}^T \end{array} \right] \\ &= \left[\begin{array}{c|c} \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B}\mathbf{B}^T & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B}\mathbf{B}^T \end{array} \right] \\ \mathcal{B}^\dagger &= \left[\begin{array}{c|c} \mathbf{B}^T & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}^T \end{array} \right]. \end{aligned} \tag{32}$$

Therefore, given a set of constants determined by the desired equilibrium and spring constants (with $k_{s_{xy}} = k_{s_{yx}} = 0$ to decouple the x and y motions of the object), the solution for the set of wheel speeds that give an object the desired dynamics is

$$\left[\begin{array}{c} \vec{V}_x^T \\ \vec{V}_y^T \end{array} \right] = \frac{1}{\mu W} \left[\begin{array}{c|c} \mathbf{B}^T & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{B}^T \end{array} \right] \left[\begin{array}{c} f_{o_x} \\ k_{s_{xx}} \\ 0 \\ f_{o_y} \\ 0 \\ k_{s_{yy}} \end{array} \right]. \tag{33}$$

Because each row of \mathbf{B}^T contains the vector $[1 \ x_i \ y_i]$, the expression for each cell's wheel speeds is

$$\begin{aligned} V_{x_i} &= \frac{1}{\mu W} (f_{o_x} + k_{s_{xx}} x_i) \\ V_{y_i} &= \frac{1}{\mu W} (f_{o_y} + k_{s_{yy}} y_i), \end{aligned} \tag{34}$$

which is independent of the other cells. In the diagonal \mathbf{k}_s case, $f_{o_x} = -x_{cme} k_{s_{xx}}$ and $f_{o_y} = -y_{cme} k_{s_{yy}}$, which, when substituted in the previous equation, yield

$$\begin{aligned} V_{x_i} &= \frac{k_{s_{xx}}}{\mu W} (x_i - x_{cme}), \\ V_{y_i} &= \frac{k_{s_{yy}}}{\mu W} (y_i - y_{cme}). \end{aligned} \tag{35}$$

This is a field in which the cells point toward the equilibrium (for negative $k_{s_{xx}}$ and $k_{s_{yy}}$), with velocities of each component proportional to the perpendicular distance to the corresponding axis. Figure 10 shows this field with $k_{s_{xx}}, k_{s_{yy}} < 0$ and $(k_{s_{yy}} - k_{s_{xx}}) < 0$.

Because of the methodology used to design this field, the dynamics of an object are invariant with respect to the set of supports. The net translational forces on the object vary smoothly and continuously over the entire array as the supports change even though the forces from each cell may vary discontinuously. This results from the fact that the design process implicitly takes into account the distribution of weight among the supports but does not consider the particular set of supports.

In addition, the field is decoupled, where the wheel speeds of each cell are independent of all the other cells and the set of supports. This allows for the computation of the wheel speeds for each cell using distributed control, where each cell need only know its own position relative to the center of the field.

Note that this field is a discretized version of an elliptic field as described in the continuous medium case by Kavraki (1997). The following expression defines the continuous elliptic field in terms of specific traction force (traction force per unit area) ($\vec{f}_s(x, y)$) as a function of position (x, y) relative to the center of the field:

$$\vec{f}_s(x, y) = (k_x x, k_y y)^T, \tag{36}$$

where the constants $k_y < k_x < 0$ determine the strength of the field. The discrete version of this field, which we just derived, replaces the specific traction force with wheel speeds such that, with $x_{cme} = y_{cme} = 0$, the following expression, equivalent to eq. (35), defines the discrete elliptic field in terms of wheel speeds as a function of cell position for the i th cell:

$$\vec{V}_i = \frac{1}{\mu W} (k_{s_{xx}} x_i, k_{s_{yy}} y_i)^T. \tag{37}$$

The fact that our derived discrete elliptic field is of the same form as the continuous elliptic field is important because Kavraki (1997) showed that the continuous elliptic field not only brings an object to a particular position but also brings it to one of two orientations within symmetry of the object. Therefore, even though the design process presented here considers only the translational dynamics, there is reason to believe that this discrete elliptical field also orients objects.

3.2. Translational Analysis of the Discrete Elliptic Field

Substituting the expression for the discrete elliptic velocity field (eq. (35)) into the translational dynamics of manipulation (eq. (18)) produces an expression for the net forces on the object as a function of the object's position and velocity. Without loss of generality, taking the designed equilibrium position to be at the origin, the dynamics of the object become

$$m\ddot{\vec{X}}_{cm} = \begin{bmatrix} k_{s_{xx}} & 0 \\ 0 & k_{s_{yy}} \end{bmatrix} \vec{X}_{cm} - \mu W \dot{\vec{X}}_{cm}, \quad (38)$$

where m is the mass of the object.

This is precisely the uniform mass-spring-damper dynamics for which we designed the discrete elliptic field. These dynamics are independent of the size and shape of the object and do not change as the supports change (and, hence, do not depend on the orientation of the object). The translational motion of the object is easily predictable, and there will be a unique translation equilibrium at the origin as long as $k_{s_{xx}}, k_{s_{yy}} < 0$.

3.3. Rotational Analysis of the Discrete Elliptic Field

Conversion of the continuous elliptic field to the discrete elliptic field preserves the translational properties but does not preserve the rotational properties. Section 2.4 showed that for an object resting on a single set of supports, the torque is not a function of orientation. Therefore, static velocity fields are incapable of orienting an object more precisely than the range of orientations which keeps the object on a single set of supports. The black rectangle in Figure 11 demonstrates this free range of motion.

Locally, we can only ensure that the object will be in rotational equilibrium when it is in translational equilibrium. Objects are oriented as they change support from one cell to the next. There are then three considerations for the object's orientation: (i) torque is zero at the translational equilibrium (at $\vec{X}_{cm} = 0$); (ii) when the object rotates about its equilibrium position, a change in supports induces a restoring torque; and (iii) given any starting position, the object will eventually reach the desired position and orientation.

3.3.1. Torque Equilibrium

Without loss of generality, we place the origin of the coordinate system at the desired equilibrium. When the object rotates about its translational equilibrium, $\vec{X}_{cm} = 0$ and $\dot{\vec{f}}_o = 0$, and eq. (23) reduces to

$$\tau = \tau_o - \omega \mu \vec{X} \vec{N}^T. \quad (39)$$

Equilibrium requires that the applied torque $\tau_o = 0$. To compute this torque from the definition of τ_o from eq. (23), we first must express the rotational velocity vector \vec{R} in this equation in terms of the elliptic velocity field. By rewriting the

wheel velocity matrix \mathbf{V} as a single-column vector formed by stacking the transposes of the x -component and y -component rows of \mathbf{V} , we obtain the following expression for \vec{R}^T :

$$\vec{R}^T = \begin{bmatrix} -y_1 & x_1 \\ \vdots & \vdots \\ -y_n & x_n \end{bmatrix} \begin{bmatrix} \vec{V}_x^T \\ \vec{V}_y^T \end{bmatrix}. \quad (40)$$

We now substitute the wheel speeds of the discrete elliptic field into this expression to obtain

$$\vec{R}^T = \frac{1}{\mu W} \begin{bmatrix} -\mathbf{D}_y & \mathbf{D}_x \end{bmatrix} \begin{bmatrix} \mathbf{B}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^T \end{bmatrix} \begin{bmatrix} 0 \\ k_{s_{xx}} \\ 0 \\ 0 \\ 0 \\ k_{s_{yy}} \end{bmatrix}. \quad (41)$$

We now substitute this expression into the transpose of the expression for τ_o (because τ_o is a scalar, it is unaffected by the transpose) to obtain

$$\tau_o = [1 \ 0 \ 0] \left(\mathbf{B}\mathbf{B}^T \right)^{-1} \mathbf{B} \begin{bmatrix} -\mathbf{D}_y & \mathbf{D}_x \end{bmatrix} \begin{bmatrix} \mathbf{B}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^T \end{bmatrix} \begin{bmatrix} 0 \\ k_{s_{xx}} \\ 0 \\ 0 \\ 0 \\ k_{s_{yy}} \end{bmatrix}. \quad (42)$$

The terms produced by multiplying \mathbf{B} with other vectors and matrices form sums of the x and y components of all the cell locations. For example,

$$\mathbf{B}\mathbf{B}^T = \begin{bmatrix} n & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 \end{bmatrix}. \quad (43)$$

In terms of these sums, eq. (42) becomes

$$\tau_o = [1 \ 0 \ 0] \begin{bmatrix} n & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum x_i y_i \\ \sum x_i^2 y_i \\ \sum x_i y_i^2 \end{bmatrix} (k_{s_{yy}} - k_{s_{xx}}). \quad (44)$$

In general, this torque is not zero, and there may or may not exist rotational equilibria at any orientation. However, consider the case in which the object is of a particular size and shape at a particular orientation, in which a set of cells on which it rests are arranged symmetrically (mirrored in x and y) about the coordinate axes. This is true for a mirror-symmetric object on a regular array when the axes of symmetry of the object are aligned with the coordinate axes. Under

this condition, any term with odd powers of x_i or y_i in a summation (such as $\sum x_i y_i$, $\sum x_i^2 y_i$, and $\sum x_i y_i^2$) will be identically zero and τ_o becomes identically zero. Therefore, a mirror-symmetric object will be in rotational equilibrium when resting on a mirror-symmetric set of supports at the translational equilibrium.

3.3.2. Rotation about Equilibrium

The symmetric arrangement of cells under the object depends on the object's shape and orientation and the cell positions. Figure 11 shows a rectangular object on a regular array of cells in three orientations. In this figure, we can see that both the symmetrically oriented object (solid black line) and the slightly perturbed object (dashed black line) cover a symmetric set of cells and, therefore, do not feel a torque. However, the object that has rotated enough to change supports (dotted black line) has a set of supports that is symmetric about the origin (radially symmetric) rather than about the coordinate axes and will feel a torque.

If both the object and the cells have mirror symmetry, the set of supports will be radially symmetric about the origin at any orientation. Many of the terms in eq. (14) become zero, and the torque reduces to

$$\tau_o = \frac{1}{n} \sum x_i y_i (k_{s_{yy}} - k_{s_{xx}}). \quad (45)$$

This will be a stabilizing torque for $k_{s_{yy}} < k_{s_{xx}} < 0$ if the object has the property that as it rotates counterclockwise, $\sum x_i y_i > 0$. We call this the positive rotation property. Many objects have this property because more of the object lies in the first and third quadrants covering more first and third quadrant cells with $x_i y_i > 0$. The black rectangle in Figure 11 shows

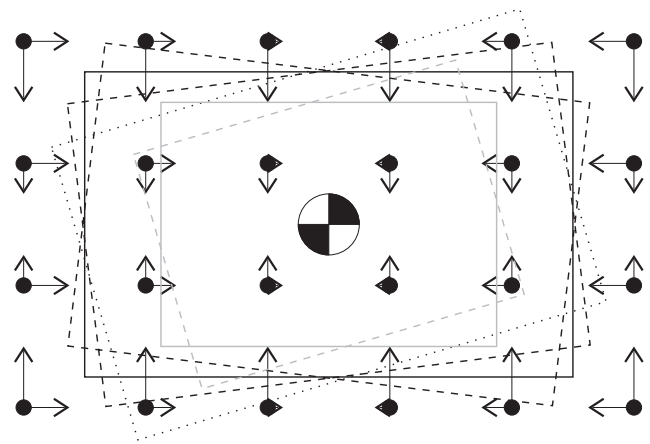


Fig. 11. Rotation of two objects (black and gray) about their equilibrium positions. Black dashed lines represent the free (unorientable) range of motion of the black object. The gray object does not satisfy the positive rotation property. Note that $k_{s_{yy}} < k_{s_{xx}} < 0$.

an object with this property. However, because of the discreteness of the array, some objects (e.g., the gray rectangle in Fig. 11) may have a negative $\sum x_i y_i$ for some counterclockwise rotations. A particular object can be checked for orientability on the array by analyzing it as it rotates about equilibrium to see which cells it covers. For objects with the positive rotation property, there will be a restoring torque for $(k_{s_{yy}} - k_{s_{xx}}) < 0$ with a stable orientation.

The positive rotation property is a property of a particular object on a particular array; a particular object may have the positive rotation property on one array but not on another. In general, when dealing with a particular array only, we consider the positive rotation property to be associated with the object only. To determine whether a particular object has the positive rotation property, we must rotate it around its equilibrium and track the sum $\sum x_i y_i$. This can be done without much difficulty for a particular object to determine its orientability.

The positive rotation property has two scopes: global and local. The global positive rotation property applies when the sum $\sum x_i y_i > 0$ for all positive orientations of the object from just after the first change in supports after 0 degrees until just before the last change in supports before 90 degrees. The global positive rotation property is a sufficient condition to ensure that an object will reach the desired orientation from any starting configuration. The local positive rotation property applies only when the sum $\sum x_i y_i > 0$ at an orientation just after the first change of supports. The local positive rotation property is sufficient for a stable rotational equilibrium but not for convergence to the equilibrium from any starting configuration: other stable equilibria may exist.

3.3.3. Arbitrary Starting Locations

We have not yet discussed torques on the object when its position is not at equilibrium. Because the translational dynamics are independent of orientation and set of supports by design, an object will reach the origin, after which the previous rotational analysis applies, ensuring an object with the positive rotational property (in the global sense) will reach the desired orientation.

3.4. Geometric Analysis of Rotational Stability

Although we can determine whether a particular object has the positive rotation property, it is of more interest to determine a class of objects that have the positive rotation property. Rather than computing orientability for each object, a method of analyzing a class is required that relies on the geometry of the system rather than simply an enumerative process. This is very difficult to do in the global sense because many changes in set of supports occur as an object rotates through 90 degrees, although this is necessary to determine which objects can be brought to a desired orientation from arbitrary starting

positions. We can, however, determine which objects have locally stable rather than globally attractive rotational equilibria. This is a necessary but not sufficient condition for global orientation of objects using open-loop manipulation.

Local rotational stability can be determined analytically for rectangular objects on a class of fields, called symmetric-squeeze-like fields, on regular arrays. More details on this subject can be found in another paper by the authors (Luntz, Messner, and Choset 1999). The result of this analysis (which we summarize here) shows that many objects (including many relatively large objects tending toward the continuous field limit) have locally unstable equilibria. Therefore, open-loop manipulation may often be impractical for precise orientation on discrete actuator arrays.

A squeeze-like field is (casually) defined such that cells in the first and third quadrants apply a clockwise torque about the origin and cells in the second and fourth quadrants apply a counterclockwise torque about the origin. Also, cells in a squeeze-like field push “toward” the coordinate axes. For example, first-quadrant cells cannot push in the positive x -direction or in the positive y -direction. We present a formal definition in another paper (Luntz, Messner, and Choset 1999). In a symmetric field, the positions and actions of the cells are mirrored about the x and y axes. The discretized version of Böhringer et al.’s (1994) squeeze field and the discrete elliptic field are both squeeze-like, as are many other fields.

The intended equilibrium for a rectangular object places its center at the origin and its major axis aligned with the x -axis. This equilibrium is reached by any object on a continuous squeeze-like field and by an object with the positive rotation property on a discrete squeeze-like field.

Because the torque on an object is independent of its orientation while on a particular set of supports, determining local rotational stability for a given object requires one only to examine the first cell transition it makes as it rotates from equilibrium. Due to symmetry, we examine only cell transitions in the first and fourth quadrants and only counterclockwise rotations.

Figure 12 shows the four types of cell transitions an object can make. In a type A transition, a cell is newly covered in the first quadrant. This is stabilizing because for counterclockwise rotations, the newly covered cell applies a clockwise moment. In a type C transition, a cell is newly covered in the fourth quadrant. This is destabilizing because for counterclockwise rotations, the newly covered cell applies a counterclockwise moment. Similarly, type B transitions are stabilizing and type D transitions are destabilizing.

Stability for a given rectangular object is determined analytically by computing which type of transition occurs first as the object rotates, and a stability map can be generated over the space of object width and height. Boundaries of this map can be generated analytically. For example, the set of object dimensions in which a type A transition and a type D transition occur simultaneously forms a boundary. Boundaries

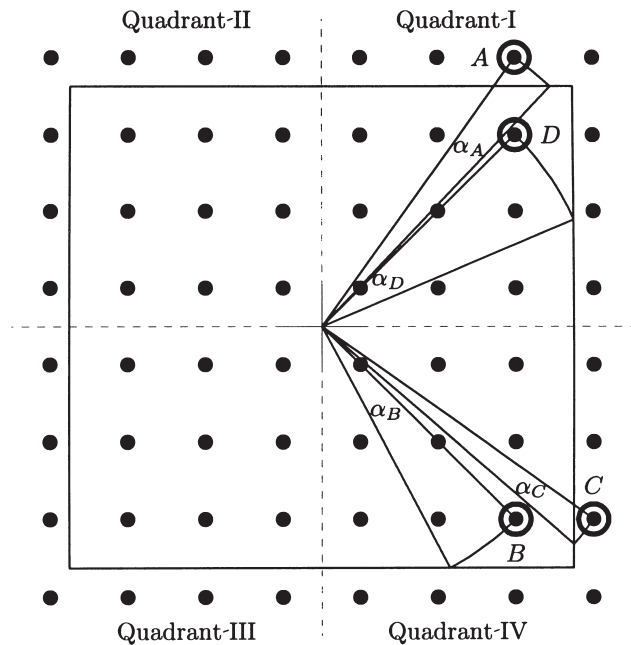


Fig. 12. Four possible first transitions exist. Two are stabilizing (A and B) and two are destabilizing (C and D).

are segments of skewed and rotated ellipses in the object dimension space. Figure 13 shows the map that was generated analytically in METAPOST by intersecting these elliptical boundaries.

Note that even large objects (which in a sense of scale tend toward a continuous array) have locally unstable rotational equilibria. Although an equilibrium may be unstable locally, there will generally be some angle, larger than the unstable transition angle, at which there is a net restoring torque. This angle may be significantly larger than the angle to first transition, which is generally considered “cell resolution.” Objects on the MDMS have been observed to find stable secondary equilibria.

3.5. Open-Loop Simulation

To verify the results from the previous section, we now simulate open-loop manipulation using the discrete elliptic field (and one other field) under the viscous friction model. Under the viscous friction model, while an object rests on a particular set of supports, its dynamics are constant and linear, and simple integration predicts the motion of the object. The following pseudocode represents the simulation of the motion of an object taking advantage of the piecewise dynamics to gain efficiency:

```

Compute which cells lie under the object
Loop forever
  Compute the constants of motion,  $\mathbf{k}_s$ ,  $\vec{f}_o$ , etc.
  based on the set of supports

```

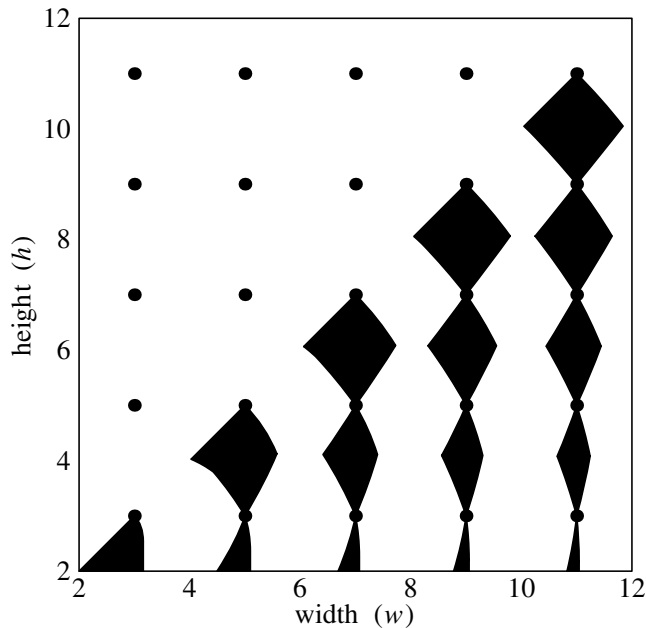


Fig. 13. Rotational stability map in the space of object width and height measured in units of cell spacings. Black regions represent objects with unstable equilibria (i.e., unorientable objects).

```

Loop
  Integrate the motion one time step
  Compute which cells lie under the object
Until supporting cells have changed
End Loop

```

Using this procedure, only simple computations (integration and support checking) are done at every time step, and the more difficult computations (recomputing constants) are done only when the supports change.

Because of its ability to efficiently simulate systems with discrete changes in state (in our case, the set of supports), Simulink is particularly suited for the computational model of the MDMS. This model, along with the simulation data of the results presented here and many other runs, is available in Extension 2. The heart of the efficiency gain in the model lies in the “Check for Change of Supports and Recalculation” block (Fig. 14). The mask vector, containing a list of cells currently supporting the object, is compared to the previous value of the mask (generated by the memory block), and if they are different, the recalculate block is triggered. When triggered, this block generates new values of the constants of motion according to their definitions in eqs. (18) and (23).

The simulator is capable of running test cases on any field with constant wheel velocities because both the cell positions and wheel speeds are inputs to the model. Here, we present

results from two classes of fields on regular square-lattice arrays. The first is the elliptic field, defined in eq. (37).

As mentioned in Section 3.3, this field has the special property that the translational constants of motion, \mathbf{k}_s and \vec{f}_o , are invariant with respect to the set of supports (Luntz, Messner, and Choset 1998b). In particular, \mathbf{k}_s is diagonal (with elements $k_{s,xx}$ and $k_{s,yy}$) and \vec{f}_o is everywhere zero. Therefore, the object acts as a simple mass-spring-damper system centered at the origin over the entire array. This field also tends to orient a rectangular object such that its major axis aligns with the x -axis of the array. In the limit of an infinitely dense array, the object exactly aligns itself with the array (Kavraki 1997).

The results of a typical simulation of an elliptic field are shown in Figure 15. Note the second-order underdamped dynamics in both the x and y directions. Because we selected this object as one having the positive rotation property, it comes to rest in an orientation that is very close to aligned with the array (but not quite).

Figure 16 shows the results of a simulation similar to that in Figure 15. In both these simulations, the field is the same but the dimensions of the object are slightly different, where we selected the second object as one without the positive rotation property. In the second simulation, the translations of the object are the same but the object comes to rest at an orientation well away from zero. This misalignment is a result of the discrete nature of the array and is not an artifact of the simulation. Continuous field theory does not predict this behavior.

A different type of field, called a squeeze field, is used in the simulation shown in Figure 17. This field is defined such that all cells with positive y -coordinate push on the object in the negative y -direction and all cells with negative y -coordinate push in the positive y -direction. Such a field tends to squeeze an object in a manner similar to that of a parallel-jaw gripper. Under such a field, a rectangular object will move to the x -axis and align its major axis with the x -axis (Böhringer et al. 1994). In this simulation, the object moves to the x -axis but comes to rest in a nonzero orientation. Again, this misalignment is due to the discrete nature of the array and is not predicted by continuous field theory. Note that the translational dynamics under the squeeze field are piecewise second order rather than purely second order as they were under the elliptic field.

3.6. Experimental Verification

A similar set of experiments was run on the experimental prototype where discrete elliptic fields manipulated cardboard rectangles to equilibrium near the upper right-hand corner of the array. Figures 18 and 19 show two experimental runs with an orientable object and an unorientable object, respectively. The orientable object is of dimension 3.1 by 2.1 (cell spacings), and the unorientable object is of dimension 2.6 by 2.1. The orientability of these objects is predicted by Figure 13.

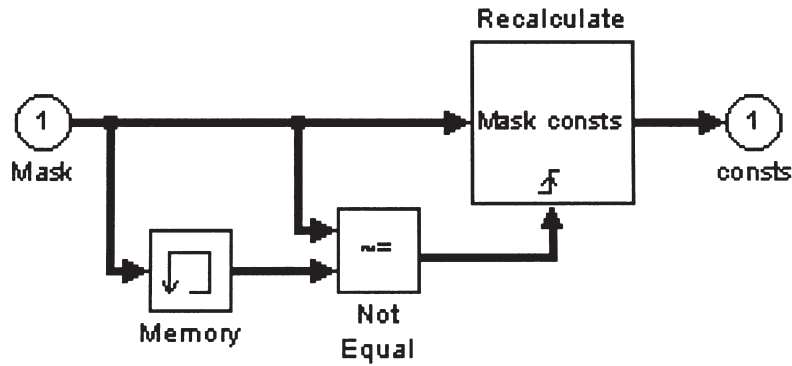


Fig. 14. Efficiency is gained using the “Check for Change of Supports and Recalculation” block. The memory and not-equal blocks check for a change in supports. A change triggers the recalculate block to update the constants of motion.

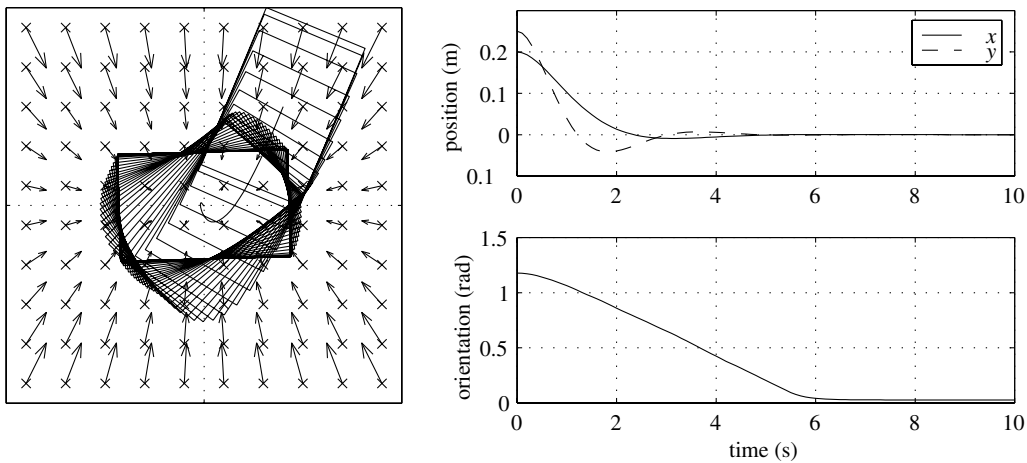


Fig. 15. Simulation of an elliptic-like field: overhead view (left) and a time plot of \vec{X}_{cm} and θ (right). In the left plot, the object starts in the upper right and is transported to the center. The \times 's in this plot represent the positions of the cells, and arrows emanating from the \times 's represent the vector whose components are the wheel speeds of each cell. The curved line traces the path of the center of the object.

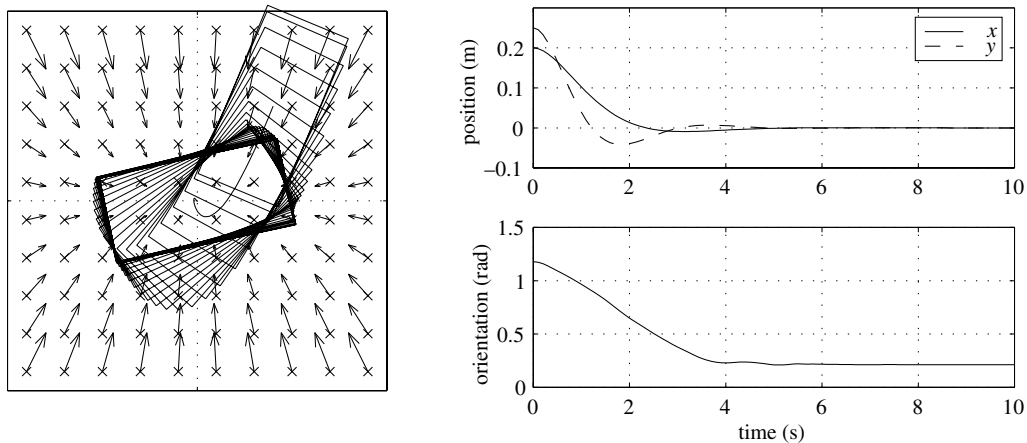


Fig. 16. Simulation of the same field as in Figure 15 with an object of slightly different size: overhead view (left) and a time plot of \vec{X}_{cm} and θ (right).

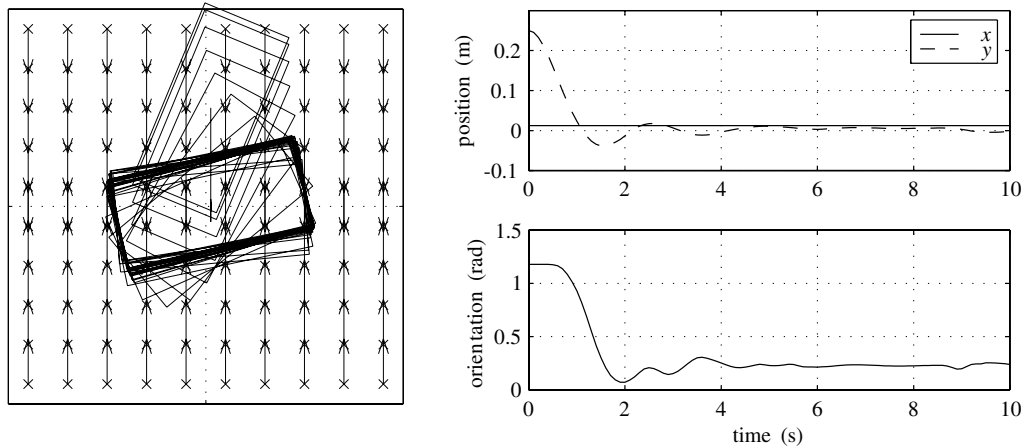


Fig. 17. Simulation of a squeeze-like field: overhead view (left) and a time plot of \vec{X}_{cm} and θ (right).

The four images in Figure 18 show that the orientable object starts on the lower left, moves and rotates toward the equilibrium, overshoots it, and finally settles at the desired position and orientation. This is the same behavior predicted analytically and in simulation. In Figure 19, the unorientable object exhibits similar behavior but settles in a secondary equilibrium orientation away from the desired. The videos from which these images were taken are available in Extension 4.

Increasingly large objects have secondary equilibria closer to the intended equilibrium, and in the limit they are equal. However, for moderately sized arrays, unstable equilibria and the existence of secondary equilibria reduce the precision to which open-loop velocity fields can orient objects. This suggests the use of closed-loop control. Closed-loop fields can provide more precision with fewer cells but increase the complexity of the system. The next section examines the design and operation of closed-loop manipulation fields.

4. Closed-Loop Manipulation

The application of feedback on the position and orientation of an object may avoid the precision limits that exist for positioning and orienting objects on a discrete actuator array using open-loop manipulation. The implementation of feedback requires either an external sensor or local sensors at each cell. An external sensor (such as a PC connected to a vision system) must be capable of broadcasting information to all the cells. This information should be as simple and generic as possible to allow for distributed control; it is not practical for a single computer to broadcast wheel speed commands to each cell individually for large systems. To use local sensing, cells must share sensor information among themselves and deduce the actual position and orientation of the object through a distributed computation.

The feedback strategy is independent of the sensor type. The assumption in this section is that each cell knows the desired position and orientation of the object and the current position and orientation of the object. Each cell adjusts its wheel speeds in response to this information.

This section demonstrates the strategy of implementing position and orientation feedback by computing classes of velocity fields that reduce the entire array to a single “virtual” actuator. This virtual actuator generates a desired net force and torque on the object. The many-input, three-output control problem then reduces to a three-input, three-output control problem with (it is hoped) decoupled dynamics. (In actuality, the Penrose pseudoinverse reduces coupling in a least squares sense, totally decoupling the dynamics only in some cases.)

Because for a given static velocity field force and torque change as the object moves, the velocity field must follow the object. The origin of the field will always be at the center of mass of the object, and in effect the cells move relative to this coordinate system rather than the object. For example, a rotational field (such as that shown in Fig. 22) must move along with the object or it will not remain centered as the object moves and will tend to transport the object along a circular path rather than just rotate it. From the point of view of the object, the field always circulates about the center of the object but the cells’ positions and velocities change.

The computation of closed-loop manipulation fields allows for such “moving” fields by expressing a cell’s wheel speeds as a function of the cell’s position relative to the center of mass of the object. When computing the wheel speeds, only the instantaneous net force and torque on the object matter, and the net force and torque’s dependence on the object’s position is irrelevant: when the object moves from a particular position, the wheel speeds must update, so the positional dependence of the net force and torque for the original field never comes into play.

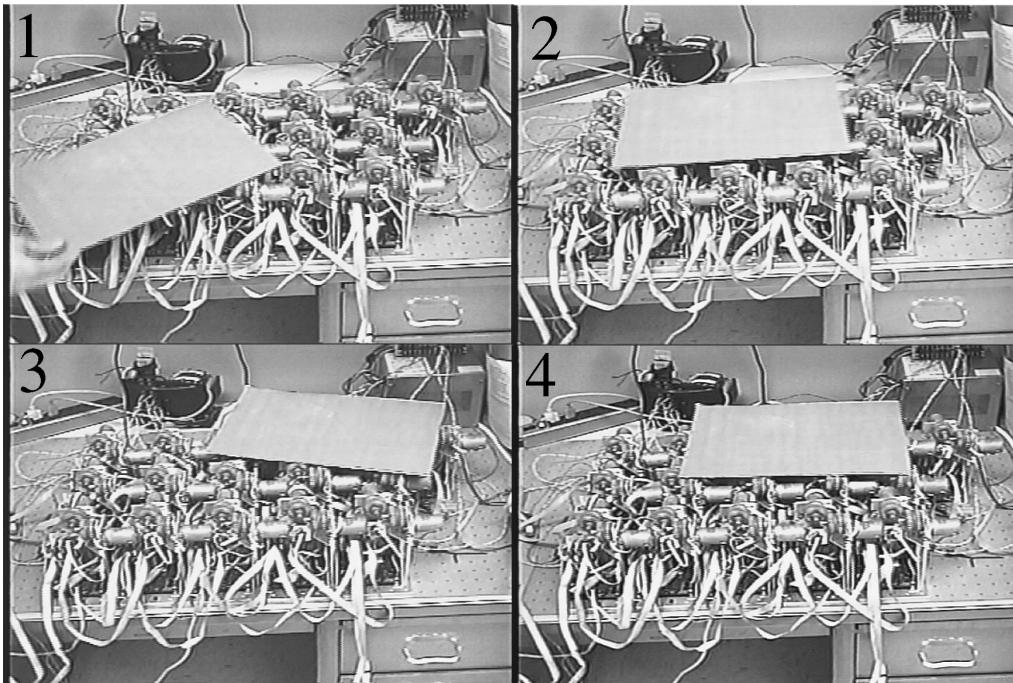


Fig. 18. Experimental time sequence. An object with the positive rotation property reaches the desired horizontal equilibrium on the modular distributed manipulator system array.

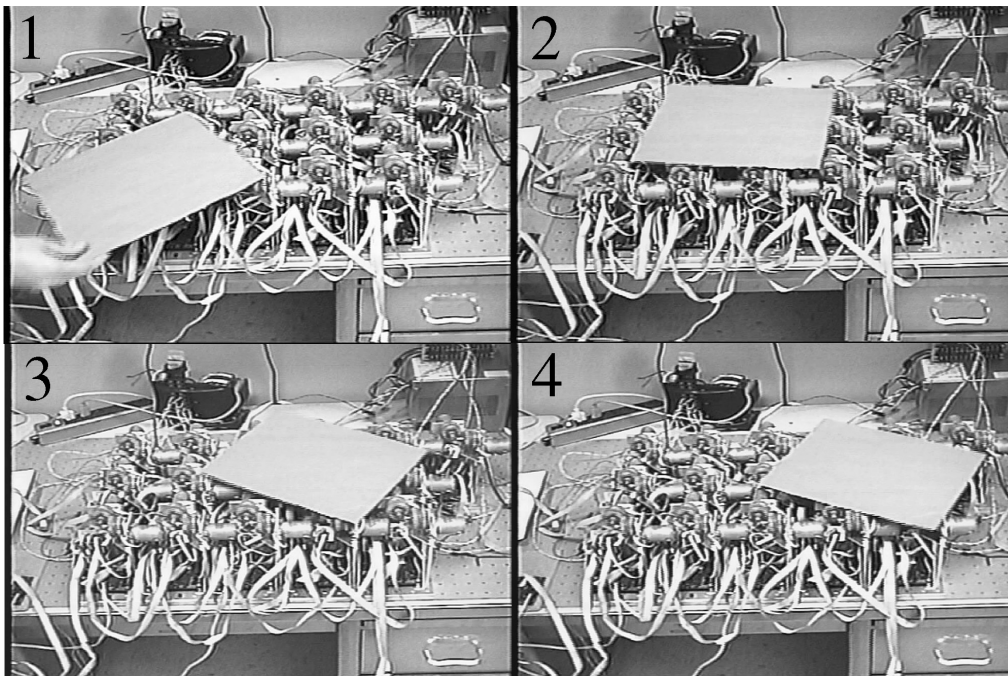


Fig. 19. Experimental time sequence. An object without the positive rotation property deviates from the desired horizontal equilibrium on the modular distributed manipulator system array.

The computation of virtual actuator fields uses the same dynamics inversion process used in Section 3 for open-loop manipulation. In this case, however, because the origin is fixed to the object, the computation generates classes of fields rather than a specific field.

The next two sections demonstrate two strategies for computing classes of virtual actuator fields.

1. **Compute force and torque together.** Solve for a single set of wheel speeds that will apply both a desired force and torque to the object. The force and torque on the object will be independent of the set of supports. Unfortunately, the wheel speeds at each cell will be a function of the set of cells currently supporting the object and the positions of the other cells under the object, making distributed control difficult.
2. **Superposition.** Solve for two fields separately: a field to apply a desired force and a field to apply a desired torque. Because wheel speeds contribute linearly to traction forces, velocity fields contribute linearly to net force and torque. The net force and torque will not depend on the set of supports, and the wheel speeds at each cell will be independent of the set of supports, making distributed control possible. The torque field, however, will result in a small disturbance force on the object, coupling the fields. A second method of computing a torque field can decouple the dynamics at the expense of variations in the magnitude of the applied torque, but not its sign.

4.1. Design of Closed-Loop Fields Computing Force and Torque Together

As a first attempt, this section enforces both a net force and torque to compute the velocity field. Without loss of generality, place the origin of the velocity field at the center of the object such that the cells appear to move rather than the object. This computation ignores the damping terms in eqs. (18) and (23); for the purpose of feedback control, treat the damping as a property of the controlled system (i.e., drag on the object moving across the cells) rather than the actuator. The expressions for the action of the virtual actuator (i.e., the net force and torque on the object) reduce to

$$\begin{aligned} \vec{f} = \vec{f}_o &= \mu W \mathbf{V} \mathbf{B}^T (\mathbf{B} \mathbf{B}^T)^{-1} \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \tau = \tau_o &= \mu W \vec{R} \mathbf{B}^T (\mathbf{B} \mathbf{B}^T)^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}. \end{aligned} \tag{46}$$

Rewriting these equations in terms of the stacked velocity vector as in Section 3,

$$\begin{bmatrix} f_{o_x} \\ f_{o_y} \\ \tau_o \end{bmatrix} = \mu W \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} (\mathbf{B} \mathbf{B}^T)^{-1} \mathbf{B} & 0_{3 \times n} & 0_{3 \times n} \\ 0_{3 \times n} & (\mathbf{B} \mathbf{B}^T)^{-1} \mathbf{B} & 0_{3 \times n} \\ 0_{3 \times n} & 0_{3 \times n} & (\mathbf{B} \mathbf{B}^T)^{-1} \mathbf{B} \end{bmatrix} \begin{bmatrix} I_n & 0_n \\ 0_n & I_n \\ -\mathbf{D}_y & \mathbf{D}_x \end{bmatrix}}_{\mathbf{Q}} \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \end{bmatrix}, \tag{47}$$

where \mathbf{D}_x and \mathbf{D}_y are defined as

$$\mathbf{D}_x = \begin{bmatrix} x_1 & & \\ & \ddots & \\ & & x_n \end{bmatrix} \quad \text{and} \quad \mathbf{D}_y = \begin{bmatrix} y_1 & & \\ & \ddots & \\ & & y_n \end{bmatrix}. \tag{48}$$

Pseudoinverting this underconstrained set of equations to solve for the stacked velocity vector produces the field expressed by

$$\begin{bmatrix} \vec{V}_x \\ \vec{V}_y \end{bmatrix} = \frac{1}{\mu W} \mathbf{Q}^T (\mathbf{Q} \mathbf{Q}^T)^{-1} \begin{bmatrix} f_{o_x} \\ f_{o_y} \\ \tau_o \end{bmatrix}. \tag{49}$$

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

4.2. Design of Closed-Loop Manipulation Fields Using Superposition

For control to be distributed, the wheel speed command at each cell must be independent of the set of supports and the actions of the other cells under the object. For distributed control, each cell would know its position relative to the center of mass of the object and the desired force and torque on the object. Because the top two partition rows of eq. (47) have forms that are different from that of the bottom partition row, the complete set of equations does not pseudoinvert to a decoupled form. Separately, however, the forms of the top partition rows and the bottom partition row do pseudoinvert to decoupled forms. Thus, computing separate fields for force and torque is possible.

Fields superimpose linearly because of the linear form of the viscous friction model (eq. (14)). A linear combination of wheel speeds results in a linear combination of traction forces at each cell and, therefore, a linear combination of net force and torque (which is simply the sum of the force and torque at each cell). Therefore, actuation fields superimpose linearly, and adopting the strategy of superimposing a force field and a torque field where each field operates under distributed control is possible.

4.2.1. Application of Arbitrary Force: Uniform Field

Using the Penrose pseudoinverse, this section computes a velocity field that provides a desired net force with the minimum sum of squares of wheel speeds. This minimizes extraneous motions in one sense, and it is hoped (with no guarantee) that the resulting field will generate no net torque.

Taking only the top two partition rows of eq. (47) yields the following expression for the net force in terms of the stacked velocity vector:

$$\begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix} = \mu W \begin{bmatrix} 1 & 0 & 0 & | & 0 & 0 & 0 \\ 0 & 0 & 0 & | & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} (\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B} & | & 0_{3 \times n} \\ \hline 0_{3 \times n} & | & (\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B} \end{bmatrix} \begin{bmatrix} \vec{v}_x \\ \vec{v}_y \end{bmatrix}. \quad (50)$$

Pseudoinverting this underconstrained set of equations requires the following lemma.

LEMMA 4.1. If an underconstrained set of equations is represented by the equation $\vec{y} = \mathbf{A}\mathbf{B}\vec{x}$, where \mathbf{A} is an $l \times m$ matrix and \mathbf{B} is an $m \times n$ matrix with $n \geq m \geq l$, and \mathbf{A} has rank l and \mathbf{B} has rank m , then $\vec{x} = \mathbf{B}^\dagger \mathbf{A}^\dagger \vec{y}$ is a solution to the original equation. (Note that $\mathbf{B}^\dagger \mathbf{A}^\dagger \vec{y}$ is not necessarily equal to $(\mathbf{A}\mathbf{B})^\dagger \vec{y}$.)

To verify this lemma, simply substitute the solution into the original equation.

To pseudoinvert eq. (50), pseudoinvert each matrix in the equation separately. The pseudoinverse of the first matrix of ones and zeros is

$$\begin{aligned} & \begin{bmatrix} 1 & 0 & 0 & | & 0 & 0 & 0 \\ 0 & 0 & 0 & | & 1 & 0 & 0 \end{bmatrix}^\dagger \\ &= \begin{bmatrix} 1 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ \hline 0 & | & 1 \\ 0 & | & 0 \\ 0 & | & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & | & 0 & 0 & 0 \\ 0 & 0 & 0 & | & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ \hline 0 & | & 1 \\ 0 & | & 0 \\ 0 & | & 0 \end{bmatrix} \right)^{-1} \end{aligned}$$

$$\begin{aligned} &= \begin{bmatrix} 1 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ \hline 0 & | & 1 \\ 0 & | & 0 \\ 0 & | & 0 \end{bmatrix} \left(\begin{bmatrix} 1 & | & 0 \\ 0 & | & 1 \end{bmatrix} \right)^{-1} \\ &= \begin{bmatrix} 1 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ \hline 0 & | & 1 \\ 0 & | & 0 \\ 0 & | & 0 \end{bmatrix}. \quad (51) \end{aligned}$$

Section 3 computed the pseudoinverse of the second matrix in eq. (50) as

$$\begin{bmatrix} (\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B} & | & 0_{3 \times n} \\ \hline 0_{3 \times n} & | & (\mathbf{B}\mathbf{B}^T)^{-1}\mathbf{B} \end{bmatrix}^\dagger = \begin{bmatrix} \mathbf{B}^T & | & 0_n \\ \hline 0_n & | & \mathbf{B}^T \end{bmatrix}. \quad (52)$$

Using these, a solution for the velocity field that applies a desired force to the object is

$$\begin{bmatrix} \vec{v}_x \\ \vec{v}_y \end{bmatrix} = \frac{1}{\mu W} \begin{bmatrix} \mathbf{B}^T & | & 0_n \\ \hline 0_n & | & \mathbf{B}^T \end{bmatrix} \begin{bmatrix} 1 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ \hline 0 & | & 1 \\ 0 & | & 0 \\ 0 & | & 0 \end{bmatrix} \begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix} = \frac{1}{\mu W} \begin{bmatrix} 1 & | & 0 \\ \vdots & | & \vdots \\ 1 & | & 0 \\ \hline 0 & | & 1 \\ \vdots & | & \vdots \\ 0 & | & 1 \end{bmatrix} \begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix}. \quad (53)$$

The resulting field, which we call the uniform field, is uniform with all cells having the same wheel speeds (see Fig. 20). In this case, 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. Note that the derivation ensures that the net force is independent of the set of supports.

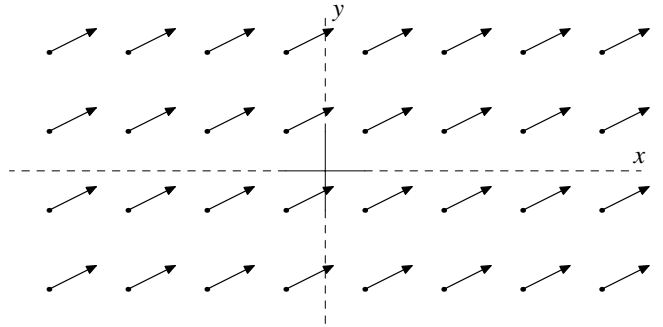


Fig. 20. Uniform field that applies an arbitrary net force on an object with no net torque.

The purpose of the uniform field is to apply a force without applying a torque. Substituting the expression for this field from eq. (53) into the expression for the net force from the bottom partition row of eq. (47) shows the net torque to be

$$\begin{aligned}
 \tau_o &= \mu W [1 \ 0 \ 0] (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} [-\mathbf{D}_y | \mathbf{D}_x] \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \end{bmatrix} \\
 &= [1 \ 0 \ 0] (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} [-\mathbf{D}_y | \mathbf{D}_x] \begin{bmatrix} 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \\ 0 & 1 \end{bmatrix} \begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix} \\
 &= [1 \ 0 \ 0] (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} \begin{bmatrix} -y_1 & x_1 \\ \vdots & \vdots \\ -y_n & x_n \end{bmatrix} \begin{bmatrix} f_{o_x} \\ f_{o_y} \end{bmatrix} \\
 &= [1 \ 0 \ 0] (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} \underbrace{\begin{bmatrix} 1 & x_1 & y_1 \\ \vdots & \vdots & \vdots \\ 1 & x_n & y_n \end{bmatrix}}_{\mathbf{B}^T} \begin{bmatrix} 0 \\ f_{o_y} \\ -f_{o_x} \end{bmatrix} \\
 &= [1 \ 0 \ 0] (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B}\mathbf{B}^T \begin{bmatrix} 0 \\ f_{o_y} \\ -f_{o_x} \end{bmatrix} \\
 &= [1 \ 0 \ 0] \begin{bmatrix} 0 \\ f_{o_y} \\ -f_{o_x} \end{bmatrix} \\
 &= 0.
 \end{aligned} \tag{54}$$

Thus, the uniform field applies no torque to the object. Although it may seem intuitively true, it is not trivial; the distribution of weight ensures that all moments balance. For example, assuming a uniform distribution of weight, the uniform field generates a uniform pointwise application of traction force at each cell that generates a net torque when the center of mass of the object does not coincide with the centroid of the supporting cells. In the actual system, however, the traction force is proportional to the normal force. The moment arms about the x and y axes used to compute the normal forces are the same as the components of the moment arms about the z -axis used to compute the torque applied by each cell such that these factors cancel and the net torque is zero. The cancellation of these moment terms displays itself algebraically in the cancellation of $\mathbf{B}\mathbf{B}^T$ with $(\mathbf{B}\mathbf{B}^T)^{-1}$ leading to eq. (54). The computation generates the uniform field with no rotational component because of the quadratic minimization inherent in the Penrose pseudoinverse.

The uniform field can apply an arbitrary force to an object using distributed control with no coupling to the rotation of the object. Because the field is uniform, the cells need no information other than the desired force; the set of supports,

the current position of the object, and even the location of each cell may be unknown. Of course, the weight of the object and the coefficient of friction affect the net force, although these multiplicative terms may be part of the control law gains in the centralized controller and remain unknown by the cells themselves. The uniform field can be used in any feedback control law to translate the object. The next step is to produce a class of fields to apply a torque to the object, hopefully without applying a force.

4.2.2. Application of Arbitrary Torque: Computed Rotational Field

The same methodology used to compute a field to apply an arbitrary force applies to the computation of a field to apply an arbitrary torque. The expression for the net torque in terms of the stacked velocity vector (from the bottom partition row of eq. (47)) is

$$\tau_o = \mu W [1 \ 0 \ 0] (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} [-\mathbf{D}_y | \mathbf{D}_x] \begin{bmatrix} \vec{V}_x \\ \vec{V}_y \end{bmatrix}. \tag{55}$$

Once again, pseudoinvert this underconstrained set of equations to solve for the stacked velocity vector. Lemma 4.1 applies to the three-matrix case in exactly the same way as it does to the two-matrix case, allowing for the separate pseudoinversion of each of three matrices that form this equation. The pseudoinverse of the first matrix is

$$\begin{aligned}
 [1 \ 0 \ 0]^\dagger &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \left([1 \ 0 \ 0] \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right)^{-1} \\
 &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} [1]^{-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.
 \end{aligned} \tag{56}$$

The pseudoinverse of the second matrix is

$$\begin{aligned}
 ((\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B})^\dagger &= \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} ((\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B}\mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1})^{-1} \\
 &= \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} ((\mathbf{B}\mathbf{B}^T)^{-1})^{-1} \\
 &= \mathbf{B}^T.
 \end{aligned} \tag{57}$$

The pseudoinverse of the third matrix is

$$\begin{aligned}
 [-\mathbf{D}_y | \mathbf{D}_x]^\dagger &= \begin{bmatrix} -\mathbf{D}_y \\ \mathbf{D}_x \end{bmatrix} \left([-\mathbf{D}_y | \mathbf{D}_x] \begin{bmatrix} -\mathbf{D}_y \\ \mathbf{D}_x \end{bmatrix} \right)^{-1} \\
 &= \begin{bmatrix} -\mathbf{D}_y \\ \mathbf{D}_x \end{bmatrix} \left(\begin{bmatrix} x_1^2 + y_1^2 & & \\ & \ddots & \\ & & x_n^2 + y_n^2 \end{bmatrix} \right)^{-1}
 \end{aligned}$$

$$\begin{aligned}
&= \begin{bmatrix} -\mathbf{D}_y \\ \mathbf{D}_x \end{bmatrix} \begin{bmatrix} \frac{1}{x_1^2+y_1^2} & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \frac{1}{x_n^2+y_n^2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{-y_1}{x_1^2+y_1^2} & & & \\ & \ddots & & \\ & & \frac{-y_n}{x_n^2+y_n^2} & \\ \frac{x_1}{x_1^2+y_1^2} & & & \\ & \ddots & & \\ & & \frac{x_n}{x_n^2+y_n^2} & \end{bmatrix}
\end{aligned}$$

Pseudoinverting eq. (55) to solve for the wheel speeds in terms of torque by multiplying the three submatrix pseudoinverses from eqs. (56), (57), and (58) in reverse order produces

$$\begin{aligned}
\begin{bmatrix} \vec{V}_x \\ \vec{V}_y \end{bmatrix} &= \frac{1}{\mu W} \begin{bmatrix} \frac{-y_1}{x_1^2+y_1^2} & & & \\ & \ddots & & \\ & & \frac{-y_n}{x_n^2+y_n^2} & \\ \frac{x_1}{x_1^2+y_1^2} & & & \\ & \ddots & & \\ & & \frac{x_n}{x_n^2+y_n^2} & \end{bmatrix} \mathbf{B}^T \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tau_o \\
&= \frac{1}{\mu W} \begin{bmatrix} \frac{-y_1}{x_1^2+y_1^2} & & & \\ & \ddots & & \\ & & \frac{-y_n}{x_n^2+y_n^2} & \\ \frac{x_1}{x_1^2+y_1^2} & & & \\ & \ddots & & \\ & & \frac{x_n}{x_n^2+y_n^2} & \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \tau_o \\
&= \frac{1}{\mu W} \begin{bmatrix} \frac{-y_1}{x_1^2+y_1^2} \\ \vdots \\ \frac{-y_n}{x_n^2+y_n^2} \\ \frac{x_1}{x_1^2+y_1^2} \\ \vdots \\ \frac{x_n}{x_n^2+y_n^2} \end{bmatrix} \tau_o. \tag{58}
\end{aligned}$$

The resulting rotational field is decoupled, where the separated expression for wheel speeds at each cell is

$$\begin{aligned}
V_{x_i} &= \frac{1}{\mu W} \frac{-y_i}{x_i^2 + y_i^2} \tau_o, \text{ and} \\
V_{y_i} &= \frac{1}{\mu W} \frac{x_i}{x_i^2 + y_i^2} \tau_o. \tag{59}
\end{aligned}$$

Figure 21 shows this rotational field, which we call the computed rotational field because it is computed to apply a

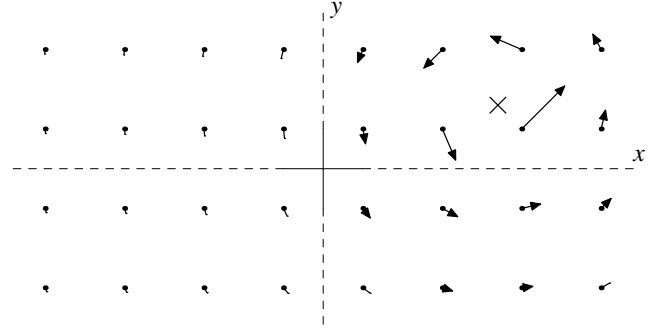


Fig. 21. Computed rotational field that applies an arbitrary net torque on an object.

specified torque. 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, the derivation ensures that the net torque is independent of the set of supports. The computed rotational field also operates under distributed control, where a centralized controller need only broadcast the desired torque and current object location to all the cells. Each cell does need to know its location relative to the object's center of mass, and therefore each cell must locate itself in the array. By passing messages from cell to cell, each cell can determine its location relative to the array in a distributed manner.

The energy minimization of the pseudoinverse distributed the net torque somewhat evenly among the supports. In particular, the torque applied from each cell (ignoring damping terms) is

$$\tau_i = \mu N_i \frac{\tau_o}{\mu W} \begin{bmatrix} \frac{-y_i}{x_i^2+y_i^2} \\ \frac{x_i}{x_i^2+y_i^2} \end{bmatrix} \times \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \frac{N_i}{W} \tau_o \frac{x_i^2 + y_i^2}{x_i^2 + y_i^2} = \frac{N_i}{W} \tau_o. \tag{60}$$

Thus, the fraction of the total torque applied by each cell is equal to the fraction of the weight supported by that cell. This distribution verifies that the net torque is independent of the set of supports as intended by the computation.

The computed rotational field, however, has two drawbacks. First, because the wheel speeds are inversely proportional to their distance from the object center, some wheel speed commands may become arbitrarily large and will saturate. In practice, only one cell will be close enough to the object center to saturate. The fraction of the total torque that will be reduced due to saturation is equal to the fraction of the weight supported by the saturated cell. In general, this inaccuracy will be small if a cell does saturate.

The second problem is that the computed rotational field applies a net force. Substituting eqs. (59) into eq. (50) produces an expression for the net force. Taking only the x -component of eq. (50) (the y -component is handled similarly), the expression for the net force in the x -direction is

$$f_{o_x} = \mu W [1 \ 0 \ 0] (\mathbf{B}\mathbf{B}^T)^{-1} \mathbf{B} \begin{bmatrix} \frac{1}{\mu W} \frac{-y_1}{x_1^2+y_1^2} \\ \vdots \\ \frac{1}{\mu W} \frac{-y_n}{x_n^2+y_n^2} \end{bmatrix} \tau_o$$

$$= [1 \ 0 \ 0] \begin{bmatrix} n & \sum x_i & \sum y_i \\ \sum x_i & \sum x_i^2 & \sum x_i y_i \\ \sum y_i & \sum x_i y_i & \sum y_i^2 \end{bmatrix} \begin{bmatrix} \sum \frac{-y_i}{x_i^2+y_i^2} \\ \sum \frac{-x_i y_i}{x_i^2+y_i^2} \\ \sum \frac{-y_i^2}{x_i^2+y_i^2} \end{bmatrix} \tau_o. \quad (61)$$

This expression, in general, will not be equal to zero and generates an extra force that was not intended by the design of this field. This force can be thought of as a force disturbance (\vec{f}_d) that is dependent on the application of torque.

If the cells under the object are arranged with radial symmetry, however, many of the terms in eq. (61) will be identically zero, and the net force vanishes as follows:

$$f_{o_x} = [1 \ 0 \ 0] \begin{bmatrix} n & 0 & 0 \\ 0 & \sum x_i^2 & \sum x_i y_i \\ 0 & \sum x_i y_i & \sum y_i^2 \end{bmatrix} \begin{bmatrix} 0 \\ \sum \frac{-x_i y_i}{x_i^2+y_i^2} \\ \sum \frac{-y_i^2}{x_i^2+y_i^2} \end{bmatrix} \tau_o = 0. \quad (62)$$

A mirror-symmetric object (such as a rectangle) on a regular square array will cover a radially symmetric set of cells (regardless of orientation) when the center of the object lies exactly on or between cells. Therefore, the disturbance force will vanish on a set of points four times as dense as the cell array. One hopes that the disturbance force does not vary much between these vanishing points and, therefore, does not become too large. However, the disturbance force is difficult to reasonably bound because it varies discontinuously as the supports change. The disturbance force may become large enough between the vanishing points to affect closed-loop performance. Section 4.2.3 examines another method to generate a torque with no disturbance force.

This computed rotational field can apply an arbitrary torque, independent of the set of supports, for orientation control of the object using distributed control. In combination with the uniform force field, there is no coupling from force to torque, so objects can be oriented without concern for the position.

4.2.3. Application of Torque with No Force: Kinematic Rotational Field

This section computes a field that generates a torque with no force by exploiting the distribution of weight among the cells to compute wheel velocities in such a way that no net force is applied. Rotational (tipping) equilibrium of the object about the x and y axes ensures that the normal forces are distributed such that

$$0 = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{bmatrix} \begin{bmatrix} N_1 \\ \vdots \\ N_n \end{bmatrix}. \quad (63)$$

The net force on the object due to viscous friction is

$$\vec{f} = \mu \begin{bmatrix} V_{1_x} & \cdots & V_{n_x} \\ V_{1_y} & \cdots & V_{n_y} \end{bmatrix} \begin{bmatrix} N_1 \\ \vdots \\ N_n \end{bmatrix}. \quad (64)$$

Equations (63) and (64) are of the same form, with the $2 \times n$ wheel velocity matrix in place of the $2 \times n$ cell position matrix. Setting the wheel velocities in eq. (64) proportional to the cell positions in eq. (63) constrains the right sides of the two equations to be equal (within an arbitrary multiplicative constant, α) and the net force to be identically zero. In addition, because the two rows of eq. (64) are linear and independent, swapping the x - and y -component rows of the wheel velocity matrix and negating the top row maintains zero net force. This generates the following rotational velocity field:

$$\begin{aligned} V_{i_x} &= -\tau_d \alpha y_i \\ V_{i_y} &= \tau_d \alpha x_i, \end{aligned} \quad (65)$$

where α is a constant with which to scale the field relative to the desired applied torque, τ_d . In this linear rotational field, the wheel speeds vary linearly with position (see Fig. 22). We refer to this as the kinematic rotational field because the velocity at each wheel matches exactly the pointwise velocity of a rigid body that rotates about the center of the field. This field is decoupled and operates under distributed control.

The net torque applied by the kinematic rotational field is

$$\tau_o = \sum_i \tau_i = \tau_d \mu \alpha \sum_i N_i (y_i^2 + x_i^2). \quad (66)$$

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, the kinematic rotational field eliminates the disturbance force generated by the computed rotational field but sacrifices the ability to apply a torque of known magnitude.

Although the magnitude of applied torque is unknown, the kinematic rotational field is still useful for feedback because it scales with the desired torque. The next section shows that even in the presence of this torque uncertainty, the closed-loop system under the kinematic rotational field will be asymptotically stable, at least for a range of gains. Because the linear rotational field applies no force, the position loop is entirely independent of the orientation loop. Positional stability and performance can be easily determined by evaluating the linear mass-damper system using standard feedback techniques.

The next section combines the uniform (translational) field with the computed and kinematic rotational fields to demonstrate simultaneous position and orientation feedback.

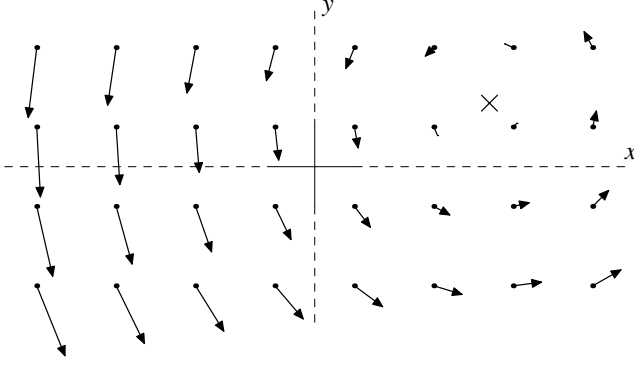


Fig. 22. A kinematic rotational field applies a torque of known direction on the object without applying a net force.

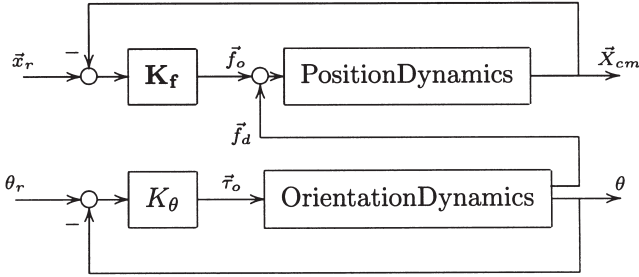


Fig. 23. Feedback control structure. Superimposable fundamental fields allow separate position (top) and orientation (bottom) loops. \bar{f}_d does not appear when using the kinematic rotation field.

4.3. Analysis of Closed-Loop Manipulation

Figure 23 shows the control system implementing two separate proportional control feedback loops for position and orientation based on the previously derived superimposable manipulation fields. The two loops are completely independent except for the disturbance force generated by the orientation loop in the case of the computed rotational field.

4.3.1. Closed-Loop Translation

The position loop has simple linear, second-order, time invariant dynamics. The forces acting on the object are linear: the net force applied by the uniform field is always equal to the desired force, and the linear damping on the object is constant and independent of the set of supports. Therefore, a transfer function from position reference input x_r to the object's position x_{cm} exists. (Actually, an identical pair of transfer functions exist, one for x and one for y .)

Under proportional control with gain K_{f_x} , the force on the object is equal to $K_{f_x}(x_r - x_{cm})$. The damping constant

from eq. (18) is equal to μW . The translational dynamics of the object are

$$\begin{aligned} m\ddot{x}_{cm} &= f_x - b\dot{x}_{cm} \\ m\ddot{x}_{cm} &= K_{f_x}(x_r - x_{cm}) - \mu W\dot{x}_{cm} \\ m\ddot{x}_{cm} + \mu W\dot{x}_{cm} + K_{f_x}x_{cm} &= K_{f_x}x_r. \end{aligned} \quad (67)$$

The transfer function from reference position to object position is

$$\frac{X_{cm}(s)}{X_r(s)} = \frac{K_{f_x}}{ms^2 + \mu Ws + K_{f_x}}. \quad (68)$$

Similarly, the transfer function from the disturbance input, f_{d_x} , to the object's position is

$$\frac{X_{cm}(s)}{F_{d_x}(s)} = \frac{1}{ms^2 + \mu Ws + K_{f_x}}. \quad (69)$$

In the case of the kinematic rotational field, the position loop is independent of the orientation loop and is already stable. In the case of the computed rotational field, the position loop (which is a stable, linear, time invariant system with a disturbance) is asymptotically stable as long as the disturbance force asymptotically approaches zero. Assuming the orientation loop is itself asymptotically stable, the orientation error (and therefore the torque command) asymptotically approaches zero. The disturbance force asymptotically approaches zero because it varies in proportion to the torque command. Therefore, the entire manipulation dynamics are asymptotically stable as long as the orientation loop is itself asymptotically stable.

4.3.2. Closed-Loop Rotation: Computed Rotational Field

The closed-loop rotational dynamics under the computed rotational field have a linear form, with mass, spring, and damping coefficients, but the damping is a function of the set of supports and the object's translational position. The computed rotational field is such that the torque applied is precise (ignoring saturation) and with a proportional gain of K_θ is equal to $K_\theta(\theta_r - \theta)$. The rotational dynamics of the object under the computed rotational field, therefore, are

$$J\ddot{\theta} + b(\bar{X}_{cm}(t), \theta)\dot{\theta} + K_\theta\theta = K_\theta\theta_r, \quad (70)$$

where $b(\bar{X}_{cm}(t), \theta) = \mu\bar{X}\bar{N}^T = \mu \sum (x_i^2 + y_i^2) \bar{N}^T$ from eq. (23).

The damping coefficient $b(\bar{X}_{cm}(t), \theta)$ is not a function of the wheel speeds, but it is a function of the position of the object and the set of supports (and implicitly the orientation). It changes discontinuously, although it is bounded both above and below. This nonlinearity is deterministic, but it is time varying because it depends on the object position (which is independent of orientation).

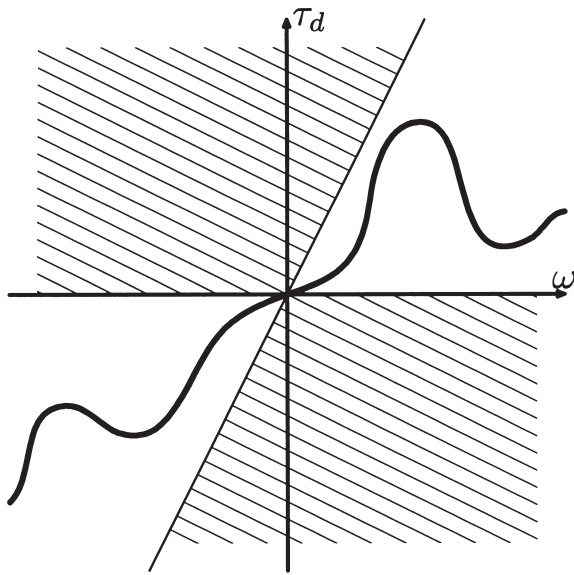


Fig. 24. Rotational damping is a sector nonlinearity where rotation speed (ω) maps to damping torque (τ_d). In this example, the nonlinearity lies in the sector $(0, 2)$.

Although it is difficult to come up with good upper and lower bounds on the damping coefficient, we can base extremely conservative bounds on the fact that the object is of finite size (contained in a circle of radius r_{max}) and that all of the supporting forces are positive. These limits ensure that the damping force is a sector nonlinearity lying in a sector (k_1, k_2) such that $k_1 < b(\vec{X}_{cm}(t), \theta) < k_2$. Figure 24 shows an example of such a sector nonlinearity.

Figure 25 shows the nonlinear orientation feedback loop for the computed rotational field. The state-dependent damping makes the orientation loop truly nonlinear, although it is *instantaneously* linear such that some linear block diagram manipulations (such as passing gains through the nonlinearity and reordering blocks) apply.

The circle criterion (Vidyasagar 1978) applied to this sector nonlinearity shows that this closed-loop system is stable. A sector nonlinearity is defined as a function $\Phi(t, x)$ such that $k_1 x^2 \leq x\Phi(t, x) \leq k_2 x^2$ for all $t \geq 0$. The damping sector nonlinearity is a variable gain, $k_1 \leq b(\vec{X}_{cm}(t), \theta) \leq k_2$, such that $\Phi(t, \theta) = \phi(\vec{X}_{cm}(t), \theta)\theta$. To apply the circle criterion, define the disk $\mathbf{D}(k_1, k_2)$ as the circle whose center lies on the real axis and crosses the real axis at $-\frac{1}{k_1}$ and $-\frac{1}{k_2}$ (see Fig. 26).

The circle criterion states that a system of the form shown in Figure 27 with a linear system $G(s)$ with ρ unstable poles connected in feedback by a sector nonlinearity, $\Phi(t, x)$, will be globally asymptotically stable if the Nyquist plot of the

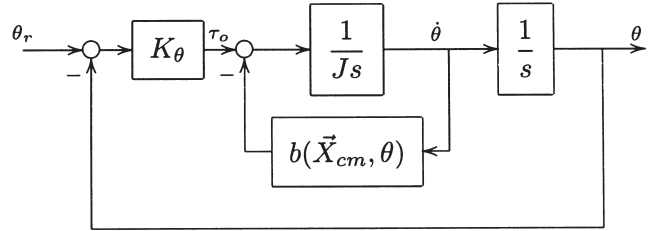


Fig. 25. The orientation loop for the computed rotational field is a second-order system with sector nonlinear damping represented by the variable multiplicative constant $b(\vec{X}_{cm}(t), \theta)$.

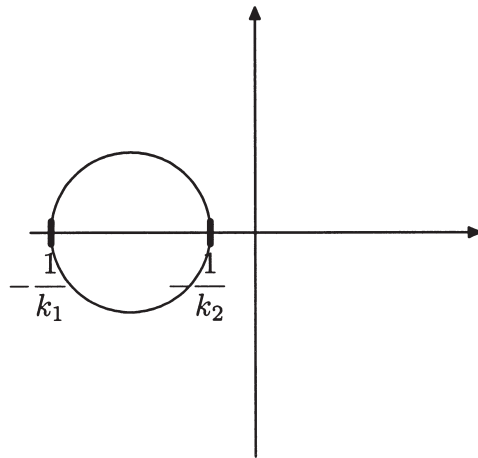


Fig. 26. The disk $\mathbf{D}(k_1, k_2)$ defined by gains k_1 and k_2 used in the application of the circle criterion.

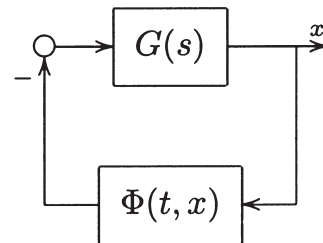


Fig. 27. Standard form of a system to be analyzed by the circle criterion.

forward-path linear system (the plot of $G(s)$ in the complex plane as s varies from zero to infinity along the imaginary axis) encircles the disk $\mathbf{D}(k_1, k_2)$ exactly ρ times counterclockwise.

In the case of the computed rotational field, the orientation loop readily transforms into the standard form of a system to be analyzed by the circle criterion. For a zero reference, the block diagram in Figure 25 is equivalent to the form shown

in Figure 28. It can be shown that the Nyquist plot of the forward path always lies in the right half plane and cannot enter or encircle the disk $D(k_1, k_2)$. Because there are no unstable poles in this forward path, by the circle criterion, the closed-loop orientation dynamics are globally asymptotically stable under the computed rotational field for all values of proportional gain. Because the orientation loop is asymptotically stable, the entire dynamics are stable.

4.3.3. Closed-Loop Rotation: Kinematic Rotational Field

Figure 29 shows the orientation loop using the kinematic rotation field. An additional nonlinearity is introduced from the desired torque to the applied torque where $\tau_a = \tau_d \mu \alpha \sum N_i(x_i^2 + y_i^2)$. Because there are two sector nonlinearities, it is more difficult to manipulate the block diagram to the proper form to apply the circle criterion.

Figure 30 shows the rearranged version of this loop. To rearrange the loop, assume the input is zero and combine the gain K_θ with the integrator block. Because the two nonlinearities differ only by a multiplicative constant, α , factor out α and define $K'_\theta = \alpha K_\theta$. The two nonlinearities (one with the integrator block), now called $\phi(\vec{X}_{cm}, \theta)$, are in parallel. Because $\phi(\vec{X}_{cm}, \theta)$ is a pointwise-linear, state-dependent, time-varying gain, combine the two parallel paths to form the block diagram at the bottom of Figure 30.

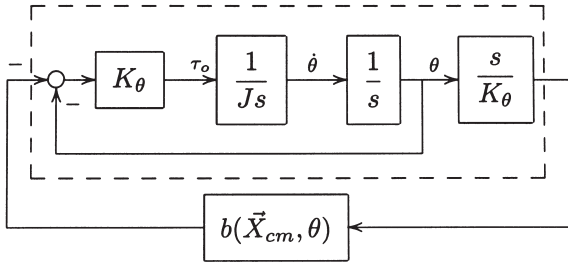


Fig. 28. The rearranged orientation feedback loop under the computed rotational field. The linear part of the system is in the forward path, and the sector nonlinearity $(b(\vec{X}_{cm}, \theta))$ is in the feedback path.

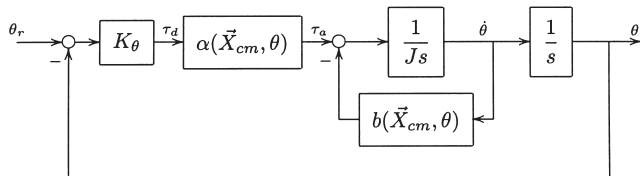


Fig. 29. Using the kinematic rotation field, a second sector nonlinearity, $\alpha(\vec{X}_{cm}, \theta)$, is introduced to the orientation feedback loop.

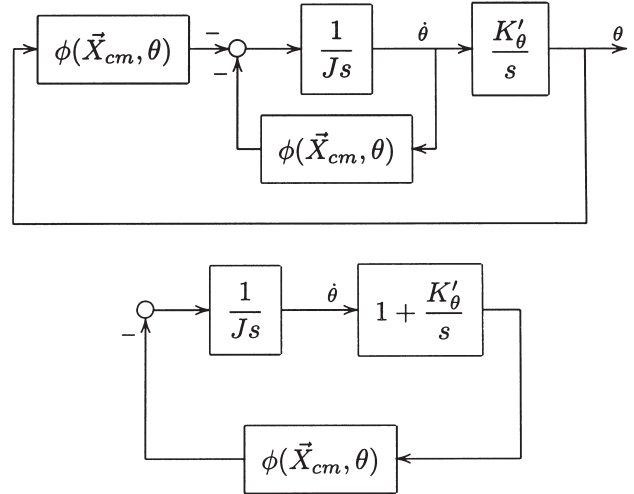


Fig. 30. The block diagram for the orientation feedback loop with the kinematic rotational field can be rearranged to apply the circle criterion. The two nonlinearities are in parallel (top) and, because they have the same form, can be combined (bottom).

The transfer function of the forward path is now

$$G'(s) = \frac{s + K'_\theta}{Js^2}. \quad (71)$$

Because $G'(s)$ has two poles at the origin, the contour must circumvent the origin. Passing the contour to the right of the origin treats the two poles at the origin as unstable. (Alternatively, one could shift the two poles slightly away from the origin by applying appropriate block diagram loop transformations.) Therefore, to satisfy the circle criterion, the Nyquist plot must encircle the disk $\mathbf{D}(k_1, k_2)$ twice.

Figure 31 shows the Nyquist plot of $G'(s)$. The solid portion of the Nyquist plot follows the function $y = \pm \sqrt{\frac{-x}{K'_\theta}}$, where x and y refer to the real and imaginary components of $G(s)$. The dashed portion of the Nyquist plot loops around counterclockwise at infinity. This figure shows that given values of k_1 and k_2 , there is a range of gains K'_θ for which the disk $\mathbf{D}(k_1, k_2)$ is encircled twice. The square root curve shifts inward as the gain is increased, and for a large enough gain, the Nyquist plot enters the disk and closed-loop stability is no longer guaranteed.

Rather than solving the cubic equation to determine the range of gains for which the Nyquist plot does not enter the disk, one can obtain a sufficient condition for asymptotic stability by guaranteeing that the Nyquist plot will not enter the square circumscribing the disk (shown by the dotted square in Fig. 31). The upper right-hand corner of the square is located at

$$-\frac{1}{k_2} + \frac{\frac{1}{k_1} - \frac{1}{k_2}}{2} j = -\frac{1}{k_2} + \frac{k_2 - k_1}{2k_1k_2} j. \quad (72)$$

This point will lie inside the square root portions of the Nyquist plot if the following condition holds:

$$\begin{aligned} \sqrt{\frac{1}{k_2}} > \frac{k_2 - k_1}{2k_1k_2} \\ K'_\theta < \frac{4k_1^2k_2}{(k_2 - k_1)^2}. \end{aligned} \quad (73)$$

Therefore, determining bounds k_1 and k_2 on $\phi(\vec{X}_{cm}, \theta)$ allows for the selection of a gain to guarantee global asymptotic stability of the rotational closed-loop system under the kinematic rotation field. It is interesting to examine limiting cases for these bounds. In the limit where $k_1 = k_2$, the denominator in eq. (73) goes to zero, and the system will be stable at any gain. This is expected because this is simply the linear case. In the limit where $k_2 \gg k_1$, the bound becomes $K'_\theta < \frac{4k_1^2}{k_2}$, which may be a very small number. Therefore, it is important to have good bounds on the nonlinearity to allow for a reasonable range of gains.

We do not currently have good bounds on $\phi(\vec{X}_{cm}, \theta) = \mu \vec{X} \vec{N}^T$, although we know finite bounds exist. The bounds on this nonlinearity are functions of the size and shape of an object. For a given object, one could obtain exact bounds by computationally moving the object over its entire configuration space (which repeats on a regular array).

Alternatively, one could obtain better bounds with a more general, geometric analysis for a class of objects (namely, rectangular objects). This would require an analysis of the discrete sampling of the object's shape over its range of motion. This is quite difficult in itself and is not within the scope of this work. Better bounds, however, would allow for the use of higher gains while guaranteeing stability. Also, the circle criterion itself provides only a sufficient condition for stability. Our conjecture is that the rotational closed-loop system using the kinematic rotation field is stable at all gains.

4.4. Simulation of Closed-Loop Manipulation

Simulating the closed-loop system is more difficult than simulating the open-loop system because the velocity field changes as the object moves. The closed-loop simulator differs from that used in Section 3 in that the constants of motion update at every time step rather than only when the supports change. This added computation reduces the simulation speed significantly, although by computing the net force and torque directly, it is still faster than evaluating the forces at each cell. Note that in the following simulations, the system achieves the desired orientation before the desired translational position. This is a result of the gains chosen and not necessarily a fundamental property of the control system. This model, along with the simulation data of the results presented here, is available in Extension 3.

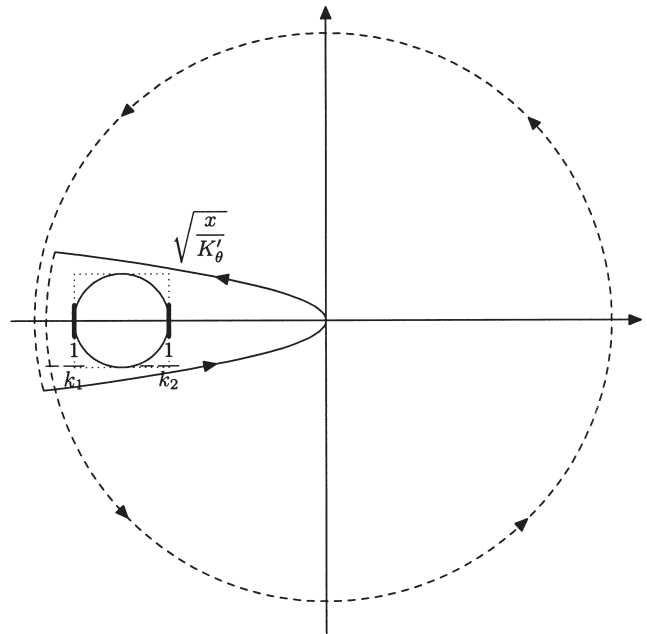


Fig. 31. Application of the circle criterion to the orientation loop using the kinematic rotational field. The circle represents the disk $D(k_1, k_2)$, where k_1 and k_2 provide the bounds on the sector nonlinearity. The solid and dashed paths show the Nyquist plot in the complex plane of the linear forward path. The dashed portion of the path represents an encirclement at infinity. The solid portion of the path is of the form of a square root. Note that $D(k_1, k_2)$ is encircled twice.

Figure 32 shows a simulation of the closed-loop system using the computed rotational field. Unlike the open-loop system, the object reaches equilibrium exactly at both the desired position and orientation. Because 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 generated by the orientation loop.

Figure 33 shows a simulation of the closed-loop system using the kinematic rotational field. Although the rotational dynamics are nearly identical to those obtained from the closed-loop simulation of the computed field (Fig. 32), the translational dynamics differ. In the case of the kinematic rotational field, the object moves in a straight line to the origin compared with the curved path of the computed rotational field simulation. Also note that the rotational motion is largely unaffected by the torque uncertainty.

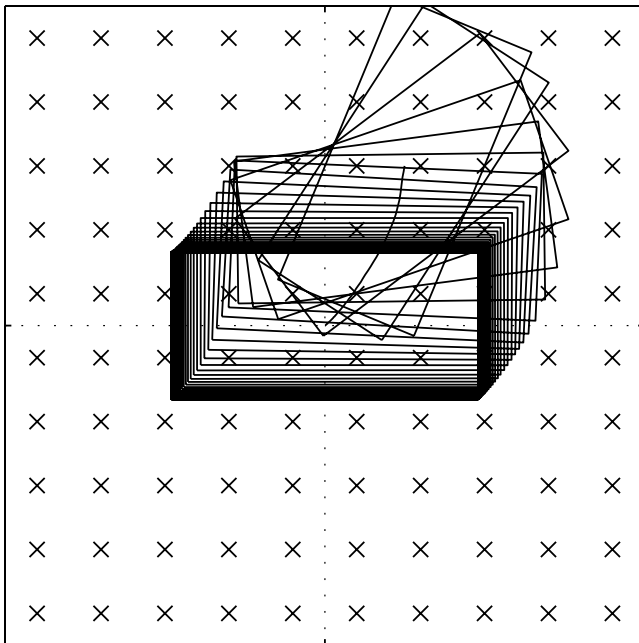


Fig. 32. Simulation of the closed-loop system under the computed rotational field. The object is shown as the moving rectangle, with a curve tracing the motion of its center. Each cell is marked with an \times .

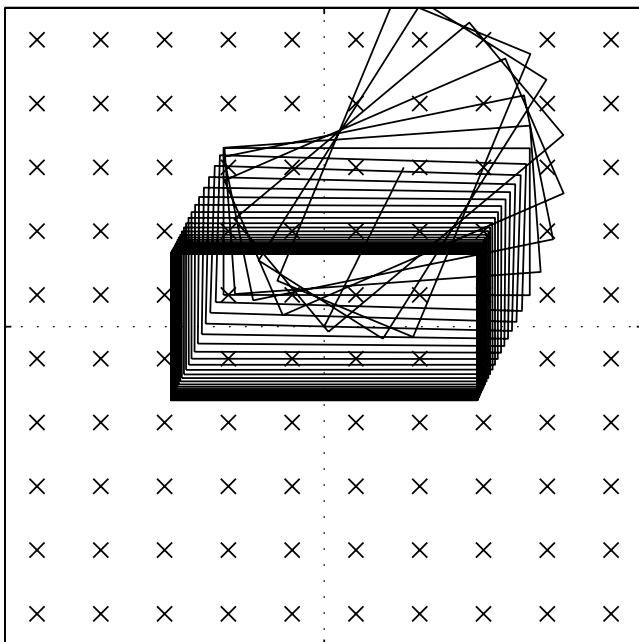


Fig. 33. Simulation of the closed-loop system under the kinematic rotational field. The object is shown as the moving rectangle, with a curve tracing the motion of its center. Each cell is marked with an \times .

5. Discussion

In this paper, we introduced the MDMS, a macro-scale variation of several micro-scale microelectromechanical manipulators with significantly greater control, computation, and communication capabilities. The paper developed the dynamic models of objects transported on discrete actuator arrays under several different contact models. Unlike previous approaches, the models specifically accounted for the finite number of contact points. Depending on the contact model, the resulting dynamics were piecewise linear first order or second order.

The assumptions used in modeling object manipulation apply not only to the MDMS but to any actuator array with discrete, spring-like supports where traction forces vary with supporting forces. This includes arrays of wheel actuators in various configurations and possibly some instances of vibratory and microelectromechanical systems. The discrete model presented here gives a better representation of the dynamics of manipulation for such systems when the spatial density of actuators is fairly low relative to object size.

We then demonstrated a methodology of systematically computing control strategies for object manipulation with discrete actuator arrays. The purpose of these methods is to eliminate many of the problems caused by array discontinuities by forcing the dynamics to be independent of the set of cells currently supporting the object. The design of wheel velocity fields used an inversion of the dynamics of manipulation to produce desired force and torque characteristics. It is possible to invert the dynamics because the dynamics themselves are (piecewise) very simple. Because the computation does not rely on the current set of supports, the resulting dynamics are independent of the set of supports, and the motions of the object are readily predictable.

The computed open-loop field (the discrete elliptic field) provides smooth translational mass-spring-damper dynamics with a single stable translational position. Because the computation of this field used only the inversion of the translational dynamics, only the translational dynamics are smooth. The rotational dynamics resemble those of the continuous version of this field only for objects having the positive rotation property, an analytically computable property (for a class of objects on a class of fields) based on the geometry of the object and the spacing of the array. Only objects with this property are stably orientable on a discrete array using such an open-loop manipulation strategy. Other objects generally find stable rotational equilibria away from the intended orientation. This behavior was demonstrated both in simulation and on the prototype system. These results are not predicted by prior work of other researchers that used continuous field approximations to discrete actuator arrays. Under continuous modeling assumptions, any object is orientable regardless of size and aspect ratio, but this is not so on a discrete array.

Good stable orientation of any object on a discrete actuator array thus requires the use of closed-loop manipulation policies. Although more complicated to implement than open-loop policies, these strategies address the open-loop precision limitations. Inverting the dynamics produced a class of fields that reduce the array of actuators to a single virtual actuator for which simple feedback methods apply. The design methodology used eliminates the dependence of the dynamics on the set of supports and once again smooths out the inherent discontinuities of the discrete array. Because of the use of distributed control, limitations are present in these strategies in the form of coupling from torque to force and in the form of torque uncertainty. These, in combination with nonlinear damping, raised the question of stability of closed-loop manipulation.

Nonlinear control theory showed that these strategies are in fact globally asymptotically stable for either some or all values of proportional gains (depending on which rotational field is used). Simulations verified the usefulness of closed-loop manipulation and demonstrated that the coupling and nonlinearities present in the closed-loop system do not greatly affect performance, at least in nominal cases. Future efforts will be directed toward performance guarantees for closed-loop manipulation using distributed control and in the development of distributed sensing algorithms. 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.

Acknowledgment

This work was supported in part by National Science Foundation (NSF) award number 9872255.

References

- Akella, S., Huang, W., Lynch, K. M., and Mason, M. T. 1996. Sensorless parts feeding with a one joint robot. *Proceedings of the Workshop on the Algorithmic Fundamentals of Robotics*.
- Böhringer, K., Bhatt, V., and Goldberg, K. 1995. Sensorless manipulation using transverse vibrations of a plate. *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Böhringer, K. F., Donald, B. R., Kavraki, L. E., and Lamiroux, F. 1999. Part orientation with one or two stable equilibria using programmable vector fields. *Proceedings of the Workshop on Distributed Manipulation at the International Conference on Robotics and Automation*.
- Böhringer, K. F., Donald, B. R., and MacDonald, N. C. 1996. Upper and lower bounds for programmable vector fields with applications to MEMS and vibratory parts feeders. *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*.
- Böhringer, K. F., Donald, B. R., Mihailovich, R., and MacDonald, N. C. 1994. A theory of manipulation and control for microfabricated actuator arrays. *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Erdmann, M. 1995. An exploration of nonprehensile two-palm manipulation: Planning and execution. *Proceedings of the Seventh International Symposium on Robotics Research*.
- Frei, P. U., Wiesendanger, M., Büchi, R., and Ruf, L. 1999. Simultaneous planar transport of multiple objects on individual trajectories using friction forces. *Proceedings of the Workshop on Distributed Manipulation at the International Conference on Robotics and Automation*.
- Goldberg, K. Y. 1993. Orienting polygonal parts without sensors. *Algorithmica: Special Issue on Computational Robotics* 10:201–225.
- Kavraki, L. 1997. Part orientation with programmable vector fields: Two stable equilibria for most parts. *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Lamiroux, F., and Kavraki, L. 2000. Positioning symmetric and non-symmetric parts using radial and constant force fields. *Workshop on the Algorithmic Foundations of Robotics*.
- Luntz, J., Messner, W., and Choset, H. 1997a. Parcel manipulation and dynamics with a distributed actuator array: The virtual vehicle. *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Luntz, J., Messner, W., and Choset, H. 1997b. Virtual vehicle: Parcel manipulation and dynamics with a distributed actuator array. *Proceedings of SPIE: Sensors and Controls for Advanced Manufacturing, International Symposium on Intelligent Systems and Advanced Manufacturing*, Vol. 3201.
- Luntz, J., Messner, W., and Choset, H. 1998a. Stick-slip operation of the modular distributed manipulator system. *Proceedings of the AACC American Controls Conference*.
- Luntz, J., Messner, W., and Choset, H. 1998b. Velocity field design on the modular distributed manipulator system. *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*.
- Luntz, J. E., and Messner, W. 1995. The virtual vehicle—A materials handling system using highly distributed coordination control. *Proceedings of the IEEE Conference on Control Applications*.
- Luntz, J. E., and Messner, W. 1996. The virtual vehicle: Materials handling using distributed actuation and highly distributed coordination control. *Proceedings of the Japan-USA Symposium on Flexible Automation*.
- Luntz, J. E., and Messner, W. 1997. A distributed control system for flexible materials handling. *IEEE Control Systems Magazine* 17(1):22–28.
- Luntz, J. E., Messner, W., and Choset, H. 1999. Open loop orientability of objects on actuator arrays. *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Luntz, J. E., Messner, W., and Choset, H. 2000. Closed-loop operation of actuator arrays. *Proceedings of the IEEE International Conference on Robotics and Automation*.

- Reznik, D., Moshkoich, E., and Canny, J. 1999. Building a universal planar manipulator. *Proceedings of the Workshop on Distributed Manipulation at the International Conference on Robotics and Automation.*
- Suh, J. W., Darling, R. B., Böhringer, K. F., Donald, B. R., Bates, H., and Kovacs, G.T.A. 1999. Cmos integrated organic ciliary actuator array as a general-purpose micromanipulation tool for small objects. *Proceedings of the Workshop on Distributed Manipulation at the International Conference on Robotics and Automation.*
- Tadokoro, S., and Takamori, T. 1999. Distributed actuation devices using soft gel actuators. *Proceedings of the Workshop on Distributed Manipulation at the International Conference on Robotics and Automation.*
- Vidyasagar, M. 1978. *Nonlinear Systems Analysis.* Englewood Cliffs, NJ: Prentice Hall.
- Yim, M., and Berlin, A. 1999. Contact and non-contact mechanisms for distributed manipulation. *Proceedings of the Workshop on Distributed Manipulation at the International Conference on Robotics and Automation.*

Index to Multimedia Extensions

Extension	Media Type	Description
1	Image	Color photo of modular distributed manipulator system (MDMS) array
2	Code, Data	Simulator and simulation results in Matlab and Simulink of open-loop manipulation
3	Code, Data	Simulator and simulation results in Matlab and Simulink of closed-loop manipulation
4	Video	Videos of an orientable and an unorientable object on the MDMS prototype implementing an elliptic field
5	Video	Video of other open-loop manipulation demos
6	Video	Technology demos on the MDMS prototype implementing distributed sensing, control, and communication

NOTE: The multimedia extensions can be found online by following the hyperlinks from www.ijr.org.