Robot Space Coordinate Representation of Objects in Euclidean Space

Justin Gallagher
Rose-Hulman Institute of Technology

ROBOT SPACE COORDINATE REPRESENTATION OF OBJECTS IN EUCLIDEAN SPACE

Justin Gallagher

MS TR 94-04

June 1994

Department of Mathematics
Rose-Hulman Institute of Technology
Terre Haute, IN 47803

FAX(812) 877-3198 Phone: (812) 877-8391

Robot Space Coordinate Representation of Objects in Euclidean Space

Justin Gallagher

Home: 1921 Steven Ave., Apt. 630
Bedford, IN 47421

School: Rose-Hulman Institute of Technology
Box 673, 5500 Wabash Ave.
Terre Haute, IN 47803

e-mail: gallagjr@rose-hulman.edu

ABSTRACT

Robot motion control strategies generally center around trajectory planning schemes which are point-to-point. This paper explores the problem of planning robot trajectories which sweep an area in a two-link robot's work space. A diffeomorphism which transforms the linear coordinates of Euclidean space to the non-linear angular coordinates which represent the displacements of the joint motors is developed. It is used to determine the distortion of an object's area at different locations in the robot's work space and for different robot link length geometries. Study of such distortions may lead to an optimization scheme by which the placement of the object in the work space and/or choice of a robot link length ratio will lead to enhanced performance of the robot.

1.0 Introduction

1.1 Motivations for this Work

An extensive amount of work has been done in the area of robot manipulator design and control relating to maximizing the performance of the robot by optimizing the path the robot takes between two points on a trajectory.[1] These schemes analyze the kinematics of a particular robot, or a whole class of robots, and define a measure which indicates which robot, among a choice of robots, delivers better performance. This approach can be used in the design stage, when the geometry of the robot is being optimized or in the program-

1. Krzysztof Tchon and Ignacy Deleba, "Definition of a Kinematic Metric for Robot Manipulators," *Journal of Robotic Systems*, 11(3), (1994), p. 211-221.

ming stage, when path trajectories are being defined. The ultimate goal is always to make the most efficient use of the robot's dexterity while it performs its tasks. In the experience of the author, no analytical scheme has taken into account the position, size or orientation of an object being operated on.

Objects are described, located, and oriented in terms of their Euclidean dimensions and coordinates. However, robots operate in terms of their link joint angle coordinates. For example, if it is desired to move a robot from point A to point B, the path can be *described* in terms of Euclidean coordinates, but it must be *performed* by adjusting each of the joints by some angular displacement. This can be considered as the way the robots "perceive" the size and location of objects in their work space.

The term "robot space" refers to the space of all possible joint angles of a robot. These angles must be controlled in order for the robot to place its end effector at any given set of coordinates in Euclidean space. The translation from Euclidean space, in which objects exist, to robot space is non-linear, but can be approximated locally by an affine transformation.The actual "size" of an object, of course, does not change depending upon where it is placed, but the amount of energy and time required to complete any given robot trajectory on or about that object may vary. The effects of inertia and torque required to complete a certain path, with the object located in one place, may offer some advantages over having the object in another place.

Because of the non-linearity of the transformation to robot space, the geometric properties of the object are distorted. The length of vectors is not preserved by such transformations; thus the length, width and area of objects, the conventional measures of an object's size, are no longer appropriate.This distortion is a characteristic of the transformation to robot space, yet it seems that it has not yet been applied to the problem of placement of an object in a robot's work space. The focus of the work in this paper is to determine if it makes a difference in the performance of a robot, how "big" an object is, as perceived by that robot. Since distance and angle (and therefore, area) are defined in Euclidean geometry by the dot product, a generalization of the dot product is needed so that it may be used with various robot space transformations.

1.2 Affine and Euclidean Transformations

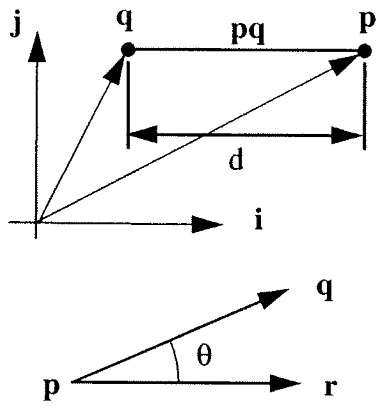
Most robot control problems treat both the robot structures and the objects being manipulated as rigid bodies. For most practical purposes, this is a perfectly reasonable assumption. When this assumption is applicable, motions by the robot and operations on the objects can be described by Euclidean geometry. Euclidean motions on objects include translations, rotations, and reflections. They have in common the property that the lengths of vectors in Euclidean space and the angle between two vectors is preserved.

A square operated on by a Euclidean motion still looks like a square. More relevant to this work is the fact that the area of the square is the same before and after the transformation. The above property is a manifestation of a fundamental property of Euclidean transformations: they preserve the Euclidean dot product. The form of the Euclidean dot product is shown in Eq. 1.

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}^T \bullet \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = a_1 b_1 + a_2 b_2 + a_3 b_3 \qquad \text{(EQ 1)}$$

Before giving the needed generalization of the dot product, it is helpful to show how the dot product defines the concepts of distance and angle in Euclidean geometry. Fig. 1 illustrates these concepts.

FIGURE 1. The Euclidean Dot Product and the Concept of Distance and Angles



$$\mathbf{q} - \mathbf{p} = \mathbf{pq} \qquad d = \sqrt{\mathbf{pq} \bullet \mathbf{pq}}$$

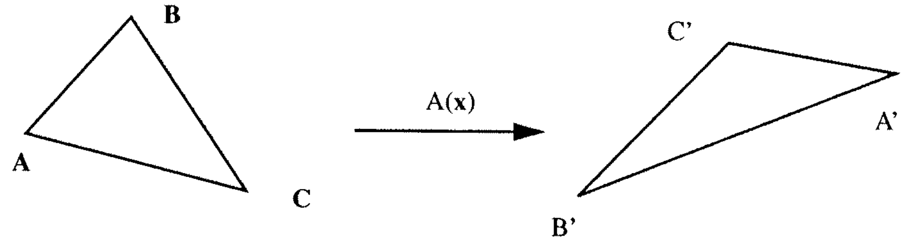
$$\theta = \cos^{-1}\left(\frac{\mathbf{pq} \bullet \mathbf{pr}}{\lVert \mathbf{pq} \rVert \lVert \mathbf{pr} \rVert}\right)$$

Euclidean motions, such as translation, rotation and reflection fall into a subset of a more broadly defined class of transformations called affine transformations. Affine transformations in two-dimensional space can be performed by matrix operations on coordinate pairs. The general form of an affine transformation is given in Eq. 2.

$$A(x) = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad \text{(EQ 2)}$$

Affine transformations are collineations. A collineation is a transformation for which all the points on a line will remain colinear after they are operated on by the transformation. Affine transformations, however, do not preserve the *length* of a vector, nor do they preserve the *angles* between vectors as Euclidean transformations do. This means that the Euclidean dot product is also not preserved. Without the dot product, the concept of the area of an object, and therefore, its “size” has no meaning. The implication in affine geometry is that all triangles are congruent, because there is an affine transformation which will take the first triangle into the second one. Fig. 2 shows two triangles which are congruent in affine geometry.

FIGURE 2. Affine Transformations and Congruent Triangles



By some affine transformation, A(**x**), ABC is congruent to A'B'C'.

All transformations must also have an inverse. This implies that, for the example in Fig. 2, there exists an affine transformation, $A^{-1}(\mathbf{x})$, which will take A'B'C' to ABC. Since the **b** component of any affine transformation represents only a translation, the **A** matrix of the transformation is responsible for any of the rotation, shearing or stretching which occurs. This matrix determines the classification of the motion. If the motion is Euclidean, it preserves the Euclidean dot product and the value of Det[A] is equal to ± 1.

2.0 The Generalized Dot Product

The generalized form of the dot product uses the notion of a positive-definite matrix:

Definition: (Positive-Definite Matrix)

A matrix **M** is positive-definite if, for all $\mathbf{v} \neq 0$,

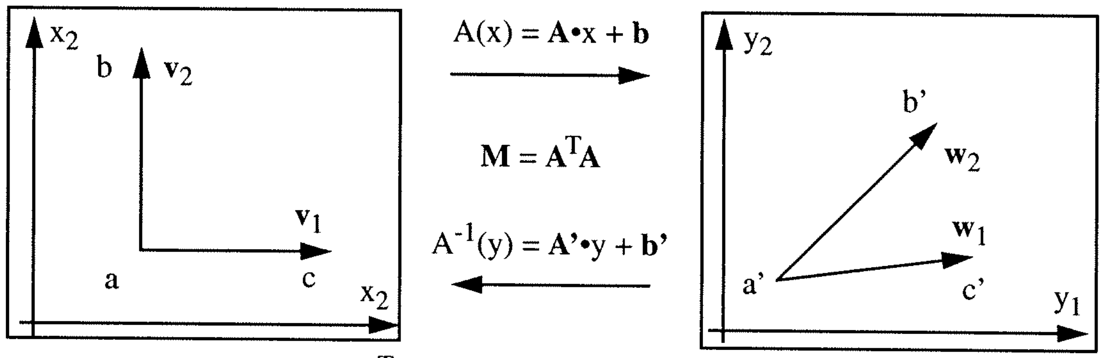
$\mathbf{v}^T \bullet \mathbf{M} \bullet \mathbf{v} > 0$, and

$\mathbf{v}^T \bullet \mathbf{M} \bullet \mathbf{v} = 0$ only if $\mathbf{v} = 0$.

The affine motions discussed in this paper use the positive-definite matrix $\mathbf{M} = \mathbf{A}^T \bullet \mathbf{A}$. For Euclidean motions, $\mathbf{M} = \mathbf{I}$, the identity matrix, which is usually not written. The generalized dot product as defined for the affine motions in Fig. 3 is

$$\mathbf{w}_1^T \bullet \mathbf{M} \bullet \mathbf{w}_2 \qquad \text{(EQ 3)}$$

FIGURE 3. Generalized Dot Product for Affine Transformations



Usual dot product: $\mathbf{v}_1 \bullet \mathbf{v}_2 = \mathbf{v}_1{}^T \bullet \mathbf{I} \bullet \mathbf{v}_2$ Usual dot product: $\mathbf{w}_1 \bullet \mathbf{w}_2 = \mathbf{w}_1{}^T \bullet \mathbf{I} \bullet \mathbf{w}_2$

In general, $v_1 \bullet \mathbf{I} \bullet v_2$ does not equal $w_1 \bullet \mathbf{I} \bullet w_2$. When **M** is defined as $\mathbf{A}^T \bullet \mathbf{A}$, however, the dot product in the untransformed space can be equated to the dot product using the identity matrix in the transformed space. That is,

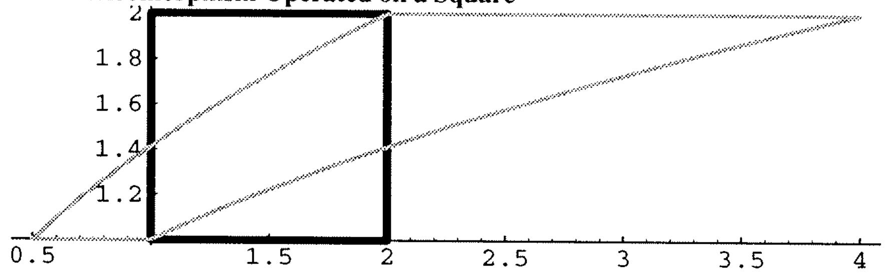
$$\mathbf{v}_1^T \bullet \mathbf{M} \bullet \mathbf{v}_2^T = \mathbf{w}_1^T \bullet \mathbf{I} \bullet \mathbf{w}_2^T \qquad \text{(EQ 4)}$$

The result is that if one uses the generalized dot product on objects in (x_1,x_2) space with the matrix **M** from the transformation to (y_1,y_2) space, the quantities determined (lengths, angles, etc.) will be those of $\mathbf{w}_1$ and $\mathbf{w}_2$, where $\mathbf{w}_1$ and $\mathbf{w}_2$ are $\mathbf{v}_1$ and $\mathbf{v}_2$, operated on by the affine transformation $A(\mathbf{x})$.

3.0 Diffeomorphisms

Diffeomorphisms are a more general class of geometric transformations than affine transformations. Diffeomorphisms are smooth, non-linear transformations for which straight lines may not map to straight lines. An example of such a diffeomorphism is given in Fig. 4.

FIGURE 4. A Diffeomorphism Operated on a Square



$$(y_1, y_2) = \left(\frac{x_1 x_2^2}{2}, x_2\right)$$

The gray "square" which exists in the transformed (y_1,y_2) space is superimposed on the black square which exists in the original (x_1,x_2) space.

Such is the case for the transformation which maps a point (x_1,x_2) in Euclidean space to the ordered pair which correspond to the joint angles (θ_1, θ_2) of a two-link robot which has its end effector at (x_1,x_2): objects have a different "shape" and "size", as perceived by a robot, depending on where they are located in the robot's work space.

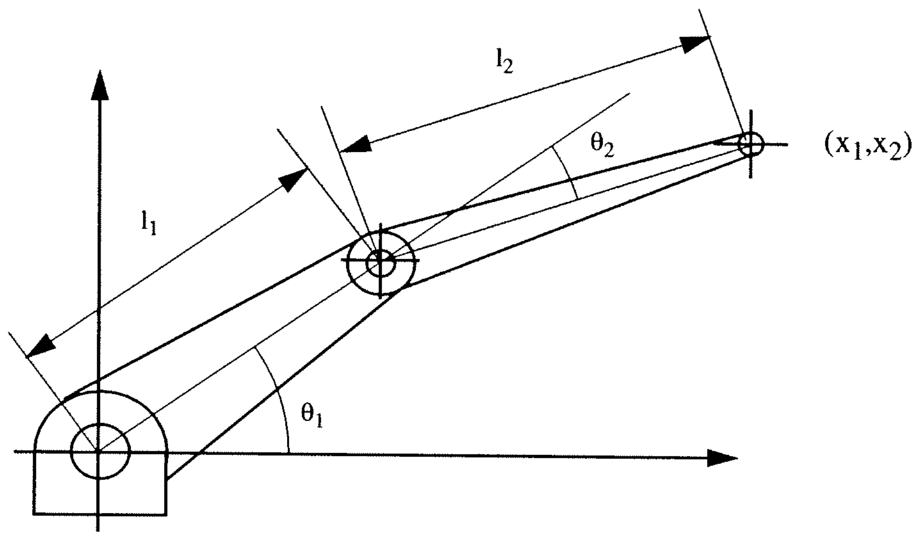
4.0 Transformation to Robot Space

The analysis and discussion in this paper pertain to a robot consisting of two members, or links, connected by rotating joints to a base. It is used to position an end effector (gripper, tool, etc.) at a specified location in a two-dimensional space, (x_1,x_2). None of the work is restricted to two dimensions by the mathematics involved; robots of three or more degrees of freedom may be considered as well. However, the kinematics of such a device become much more complicated. In particular, at many locations in the robot's work space there are multiple robot configurations which will achieve the desired positioning of the end effector. The scope of this paper is restricted to a two-link robot in order to explore the potential advantages of object location on robot performance.

4.1 The Two-Link Robot

Fig. 5 is a diagram of a generic two-link robot and the nomenclature used to describe it. Note that the definition of θ_2 makes θ_2 negative in the position shown.

FIGURE 5. Two-Link Robot and Nomenclature



Because the joint angle displacements are really the variables to be controlled, via voltage signals to the joint motors, it can be said that the robot "perceives" the object in terms of a field of (θ_1, θ_2) pairs.

4.2 Coordinate Transformation

The coordinate transformation from Euclidean space to robot joint angles are given in Eq. 5 and Eq. 6.[1-2]

$$\theta_2(x_1, x_2, l_1, l_2) = \mathrm{ATAN2}\left(\frac{x_1^2 + x_2^2 - l_1^2 - l_2^2}{2l_1l_2}, -\sqrt{1 - \left(\frac{(x_1^2 + x_2^2 - l_1^2 - l_2^2)}{2l_1l_2}\right)^2}\right) \quad \text{(EQ 5)}$$

1. Michael Brady, et.al., *Robot Motion: Planning and Control*, MIT Press, 1984, p. 10.

2. John L. Craig, *Introduction to Robotics: Mechanics and Control*, Addison-Wesley, Reading Massachusetts, 1989, p. 445-446.

$$\theta_1(x_1, x_2, l_1, l_2) = \mathrm{ATAN2}(x_1, x_2) - \mathrm{ATAN2}(l_1 + l_2 \mathrm{COS}\,\theta_2, l_2 \mathrm{SIN}\,\theta_2) \quad \text{(EQ 6)}$$

The appearance of θ_2 in the definition of θ_1 is not an indication of dependence, as either angle can be determined independently. The principle arctangent function is used to force the robot into the correct quadrant and to choose the "elbow up" or "elbow down" configuration (Fig. 5 is in the "elbow up" configuration). For each configuration, these robot space coordinates define the position of the robot end effector uniquely for a given pair of robot link lengths.

4.3 Forward Kinematics

The forward kinematics, that is, the diffeomorphism which transforms the Euclidean coordinates of the end effector to the robot space coordinates of the joint angles is the ordered pair shown in Eq. 7.

$$F(x_1, x_2, l_1, l_2) = (\theta_1(x_1, x_2, l_1, l_2), \theta_2(x_1, x_2, l_1, l_2)) \quad \text{(EQ 7)}$$

4.4 Generalized Dot Product for Diffeomorphisms

For diffeomorphisms, such as $F(x_1,x_2,l_1,l_2)$, the matrix **M** described earlier is a non-linear function of x_1 and x_2. For affine transformations, $\mathbf{M} = \mathbf{A}^T \bullet \mathbf{A}$, as previously stated. For diffeomorphisms, it is necessary to form the matrix **M** from the system Jacobian matrix, which is obtained from a linearization of $F(x_1,x_2,l_1,l_2)$ at some (x_1,x_2) operating point, $\mathbf{x}_0$. See Eq. 8 - Eq. 10.

$$L(\mathbf{x}) = \frac{d(F(\mathbf{x}_0))}{d\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) + F(\mathbf{x}_0) \quad \text{(EQ 8)}$$

$$L(\mathbf{x}) = \frac{d(F(\mathbf{x}_0))}{d\mathbf{x}_0}\mathbf{x} - \frac{d(F(\mathbf{x}_0))}{d\mathbf{x}_0}\mathbf{x}_0 + F(\mathbf{x}_0) \quad \text{(EQ 9)}$$

$$L(\mathbf{x}) = \mathbf{A} \bullet \mathbf{x} + \mathbf{b} \quad \text{(EQ 10)}$$

where $\mathbf{A} = \frac{d(F(\mathbf{x}_0))}{d\mathbf{x}_0}$ and $\mathbf{b} = -\frac{d(F(\mathbf{x}_0))}{d\mathbf{x}_0}\mathbf{x}_0 + F(\mathbf{x}_0)$.

The term $\frac{d(F(\mathbf{x}_0))}{d\mathbf{x}_0}$ is the system Jacobian matrix, **J**. It contains partial derivatives with respect to all the coordinates. For diffeomorphisms, the matrix **M** is $\mathbf{J}^T \bullet \mathbf{J}$. The matrix **M** carries with it all of the dependencies of $F(x_1,x_2,l_1,l_2)$. A full symbolic expression for this matrix fills several pages of *Mathematica* output. See Appendix C for a list of the *Mathematica* functions used.

For every robot geometry, and for every point in the plane, this matrix is uniquely defined. This results in a "matrix field" which describes the degree to which the transformation distorts the object at each point in the plane. Eq. 11 gives an expression by which it is possible to quantify the amount of distortion induced by a particular robot at a particular point in the plane.

$$\sqrt{Det[\mathbf{M}\,(x_1, x_2, l_1, l_2)]} \qquad \text{(EQ 11)}$$

A value of more than one is interpreted to mean that the diffeomorphism has "expanded" the plane at this particular point. Similarly, a value of less than one implies a contraction. The locus of points in the plane at which Eq. 11 is evaluated to 1 represent those places where the transformation has not distorted the plane. It is possible to integrate this expression in one and/or two dimensions and so quantify how whole regions of space are distorted.

4.5 Visualization of Robot Space Transformations

It is helpful to see the effect of transformations by examining a simple figure before and after transformation. See Fig. 6 for an example.

FIGURE 6. Transformation from Euclidean Space to Robot Space

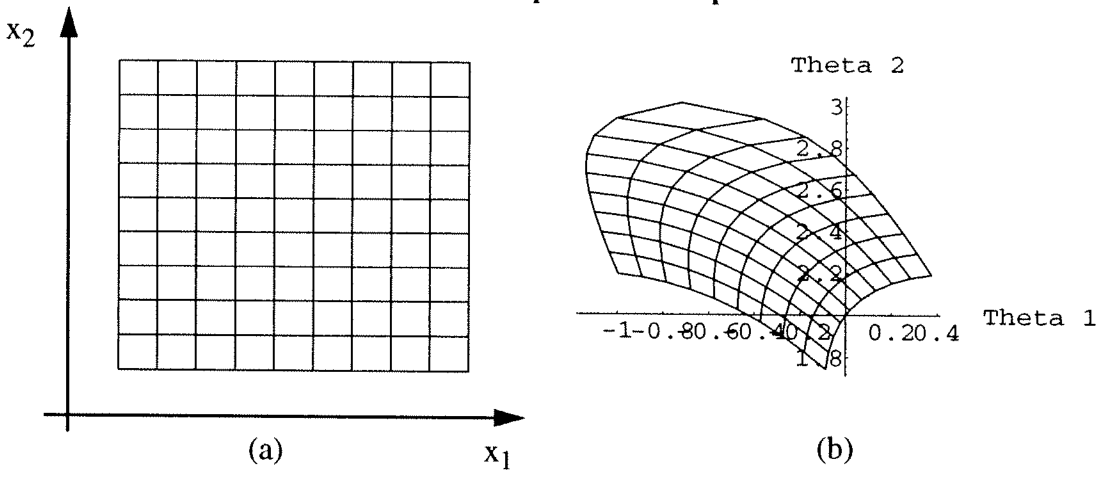


Fig. 6 (a) represents a grid composed of unit squares with the lower, left-hand corner at (1,1) in the plane. It is important to note that the distorted "grid" in Fig. 6 (b) is not the same grid in (a). It is the locus of (θ_1, θ_2) coordinates that the robot must have to place the end effector at the corresponding (x_1,x_2) coordinate in (a). There is a one-to one correspondence, however, at each of the intersections of the grid.

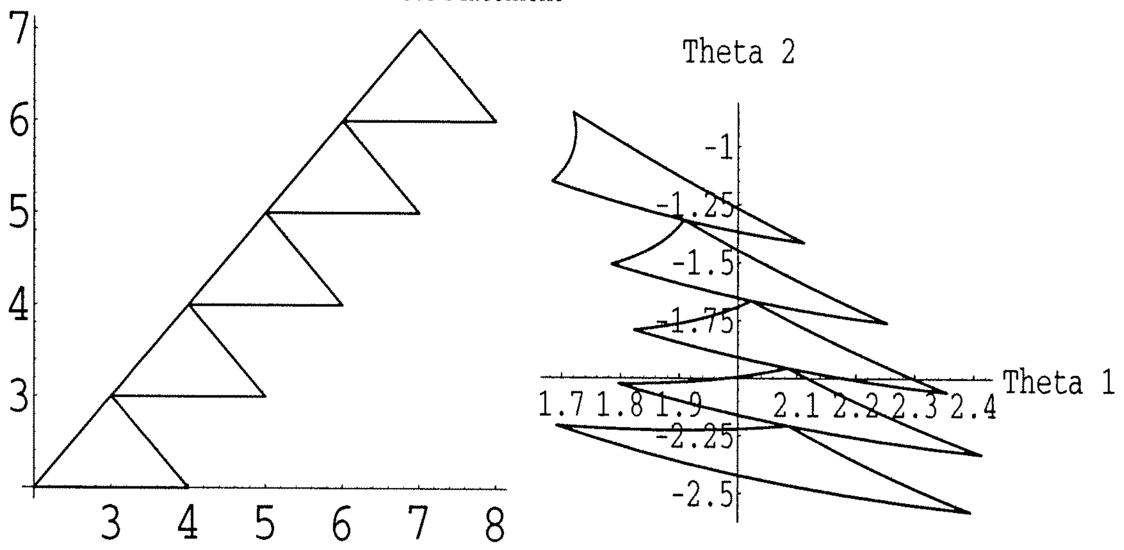
5.0 Effects of Robot Geometry and Object Placement

The primary goal of the work described in this paper is to determine if objects in the robot's work space are distorted more at one location in the robot's work space than at another. The objective is to determine which location of the object in the work space, which robot geometry, or some combination of the two will minimize the area of the object *as perceived by the robot.* For example, if the painting of an object can be accomplished by smaller accumulated joint angle displacements, then the painting process should require less energy, and perhaps, less time to complete.

5.1 Placement of the Object in the Work Space

By plotting a triangle in several different locations in the plane and observing the changes in the (θ_1, θ_2) coordinates at each location, it is observed that the object requires distinctly different motions by a robot of constant link lengths to circumscribe the triangle. See Fig. 7 to observe this effect. This particular robot was defined as having equal link lengths at 4 units each. The magnitude of the link units is important only in that the robot's work space must encompass the entire object (or objects).

FIGURE 7. The Effect of Work Piece Placement



The results of a more detailed analysis appear in Appendix A. Each of the graphics arrays represents a different stage of the analysis, but the upper-left graph, for example, in each represents the same portion of the work space. Discussion of each of the graphs follows.

- **Undistorted Grids**

A 4×4 grid is used as the object to be operated on. The robot is assumed to have its base joint at the origin of the coordinate axes. The location of the object is set at nine different locations, encompassing most of the robot's work space.

- **Distorted Grids**

Just as is shown in Eq. 7, the object is distorted differently in each location. It is also possible to identify a trend as the object is moved away from the base of the robot. The object seems to "shrink" more the farther out in the robot's work space it is.

- **Distortion Matrix Fields**

Values for $\sqrt{\mathrm{Det}\,[\mathbf{M}(l_1, l_2, x_1, x_2)]}$ are computed at each of the intersections of the grid used as the object. Surface maps of these values indicate the variation of the distortion between intersections.

- **Distortion Matrix Contours**

The curvature of the surface plots is difficult to perceive, due to the small differences in magnitude over most of the work space. The contour plots are included to show that the distortion continues to have the effect of "shrinking" the object over most of the work space.

5.2 Robot Geometry

Appendix B consists of a study in which the size and location of the object (grid) are held constant, and the geometry of the robot varies. First, both links are the same length, but the lengths are increased for each of three trials. Secondly, the effects of varying the ratio of the length of the base link to the free link is examined.

- **Equal Link Lengths**

The effect of holding the link lengths equal and varying them for three values: 11, 22, and 33, from left to right., as shown in Table 1.

TABLE 1. Equal Length Ratios

Robot	Link 1	Link 2	Ratio
Robot 1	11	11	1
Robot 2	22	22	1
Robot 3	33	33	1

The grid patterns are accompanied by three-dimensional plots of the degree of distortion, as calculated by Eq. 11, and by contour plots, as in the previous study. The trend clearly shows that the larger the robot is relative to the object the greater the distortion (shrinkage) observed. This is not as obvious as it might seem, because this not a visual observation, but the effect of the highly non-linear transformation to robot space on the dot product used to evaluate the area of an object.

- **Varying Link Length Ratio**

By changing the ratio of L_1/L_2, it is possible to see how the object "looks" to different robots. In this case a region is used which all of the robots are able to reach.The grid used here begins at (5,5) in the plane and extends for 10 units up and 10 units to the right. The robot lengths used are listed in Table 2.

TABLE 2. Varying Link Length Ratios

Robot	Link 1	Link2	Ratio
Robot 1	7	14	1/2
Robot 2	11	11	1
Robot 3	14	7	2

The length dimensions are chosen to allow all three configurations to reach the same work space, while allowing the links to have lengths in integer units.

It is interesting to note that the plot seems to "swerve" to the right for the first plot, become more "square" for the second, and then "swerve" to the left in the third. In addition, the degree to which the plot is distorted is not the same for $L_1/L_2 = 1/2$ as for $L_1/L_2 = 2$. This is interesting; it suggests that the object is distorted differently when the longer of the two links is the base link or the free link. The effect of this difference on robot performance would be important to a robot designer trying to optimize the geometry of the robot.

Note that the surface plot for Robot 1 seems identical to the surface plot for Robot 3. Given that the (θ_1, θ_2) plots look so different, this result is highly unexpected and will require additional analysis to understand.

- **Area Calculations**

Eq. 11 can be integrated numerically over (x_1,x_2) to quantify the degree of distortion induced by the different robot geometries. The results are given in Table 3 for each study.

TABLE 3. Aggregate Area Distortion

Study	Robot	Link Length Ratio	Area "Fraction of Original"
Moving Object	Locations 1 - 3	1 (Length = 11)	0.36729, 0.20614, 0.15025
	Locations 4 - 6		0.20614, 0.16593, 0.13955
	Locations 7- 9		0.15025, 0.13955, 0.13376
Equal Link Length	Robot 1	1 (Length = 11)	0.72377
	Robot 2	1 (Length = 22)	0.29423
	Robot 3	1 (Length = 33)	0.19070
Varying Link Length	Robot 1	1/2 (Length = 7/14)	0.97517
	Robot 2	1 (Length = 11/11)	0.72377
	Robot 3	2 (Length = 14/7)	0.97517

6.0 Conclusions

It is clear that the placement of the object in the robot's work space and the geometry of the robot itself have an impact on the motions the robot must undergo to reach all points on the object. As of now, this work is, for the most part, qualitative, as the numbers can be calculated, but their significance is not yet completely clear.

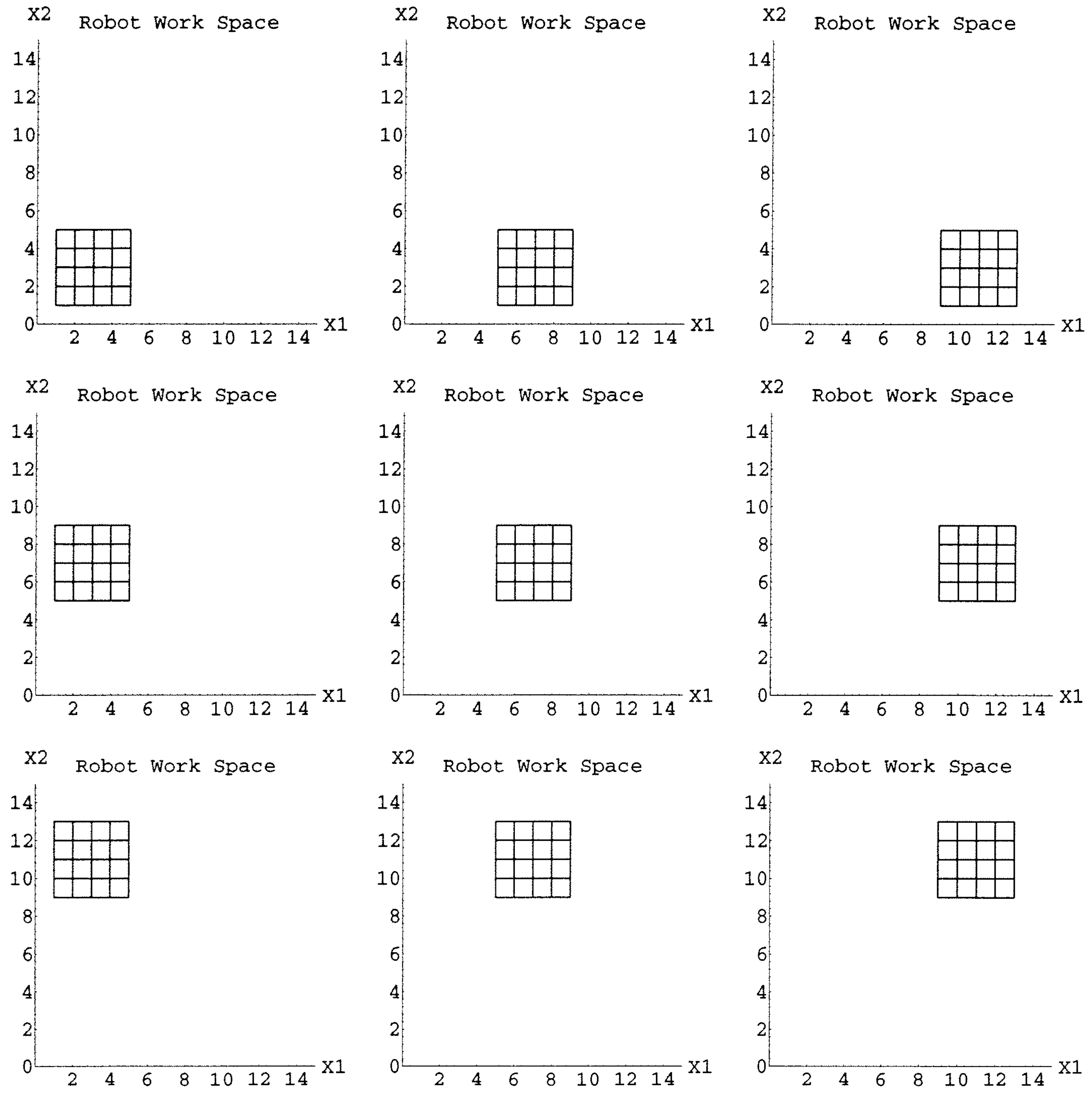
The possibility of developing a routine to optimize robot geometry based on the object size and/or placement is very intriguing. Even better, optimizing robot trajectories by virtue of simply placing the object in the most fortuitous location, and thus maximizing the utilization of the robot seems to be a subject worthy of further study.

The physical meaning of Euclidean properties in a non-Euclidean setting, such as the coordinate transformation to robot space is not completely understood. The scheme presented here, to quantify the area of an object, *as perceived by a robot*, seems to make a degree of intuitive sense, but the actual significance of the magnitude of specific values relative to each other, is as yet poorly understood. Extension of this work to robots with more degrees of freedom represents a significant analytical challenge, but is a necessary step if an optimization scheme is to be developed.

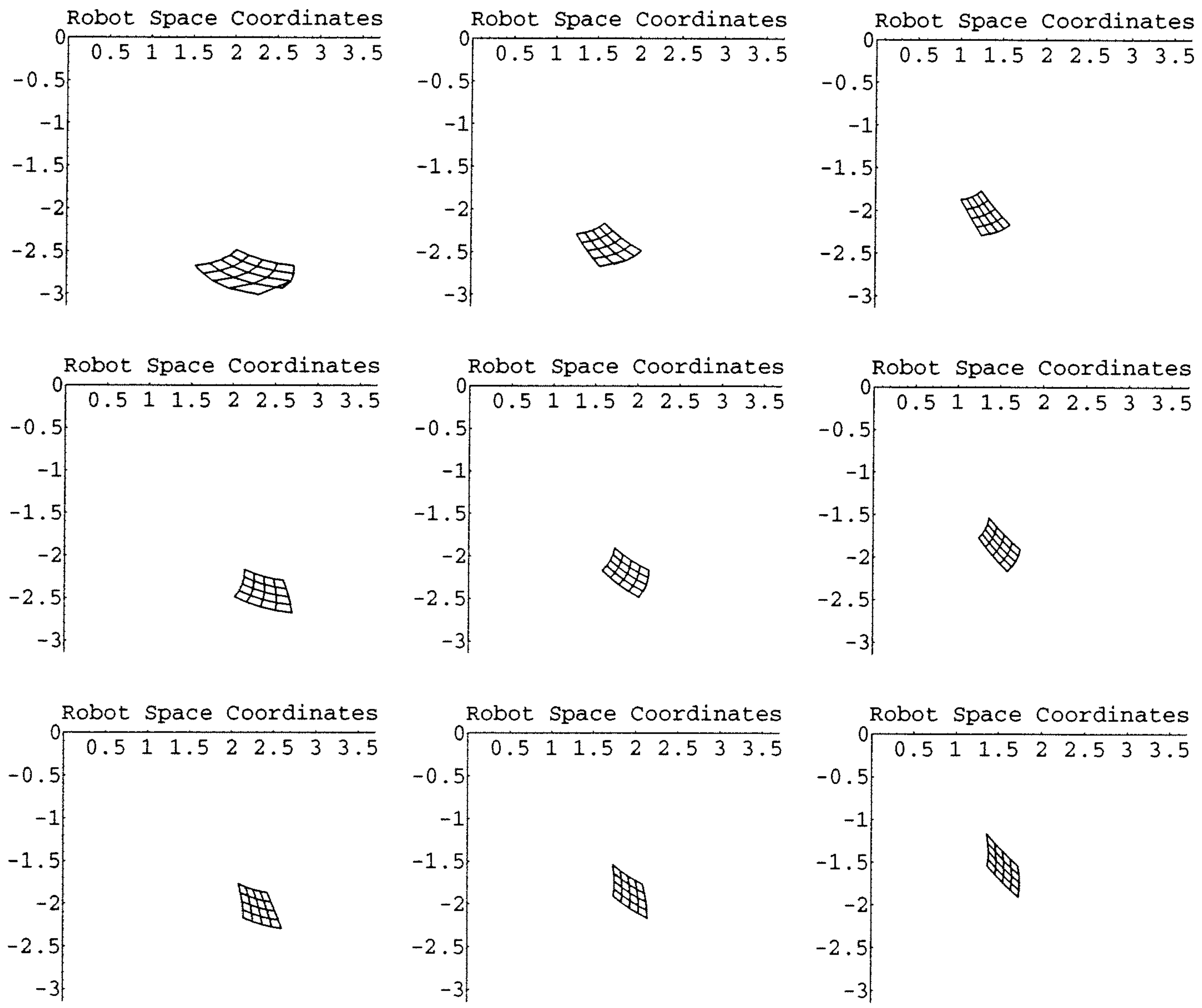
It would seem reasonable that experiments with a robot, fitted with sufficiently sensitive instrumentation are in order. These experiments might indicate whether the location of an object in the robot's work space affects the performance of the robot in a beneficial way. The next step would be to correlate the data collected with the analysis in this report, in an attempt to develop an optimized location for objects in the robot's work space.

Appendix A: Placement of the Object in the Work Space

☐ Undistorted Grids

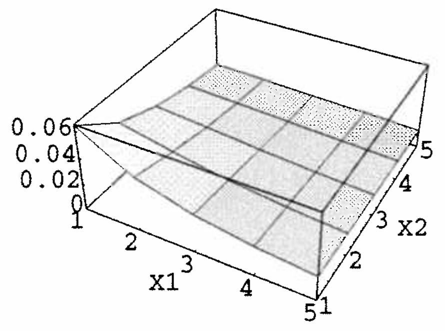


□ Distorted Grids

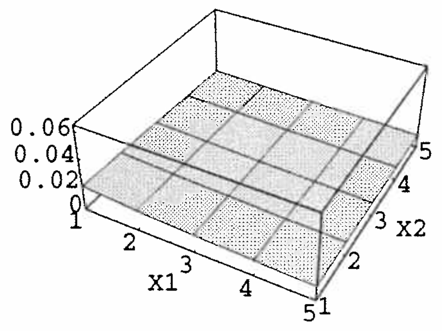


☐ Distortion Matrix Fields

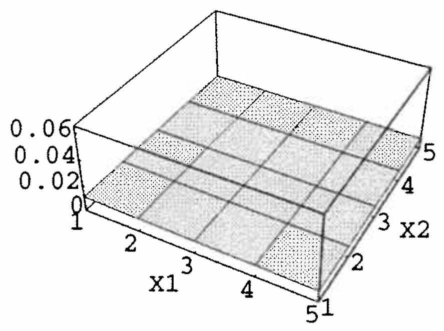
Distortion Matrix Field



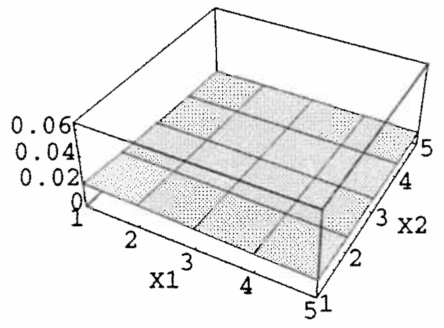
Distortion Matrix Field



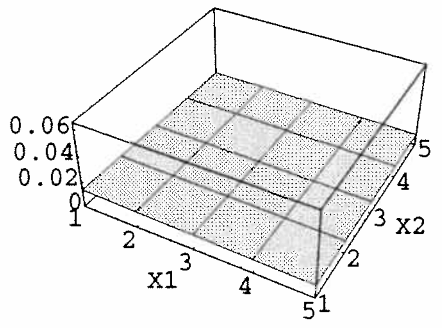
Distortion Matrix Field



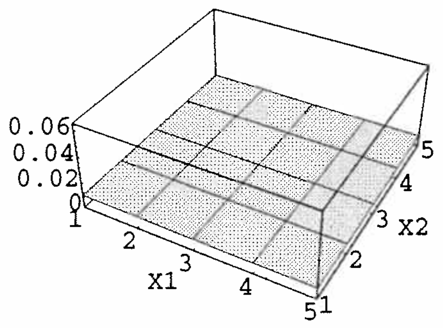
Distortion Matrix Field



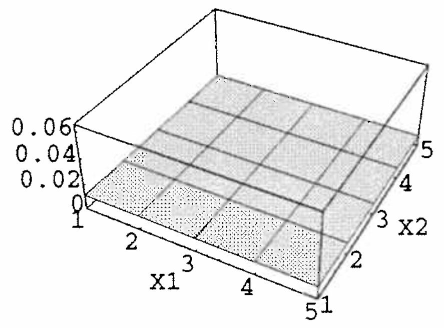
Distortion Matrix Field



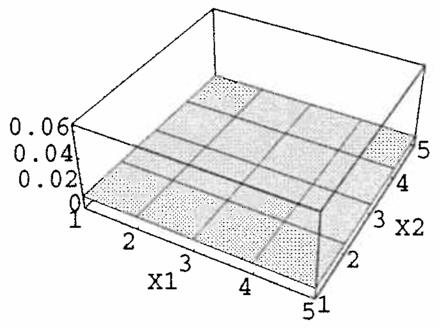
Distortion Matrix Field



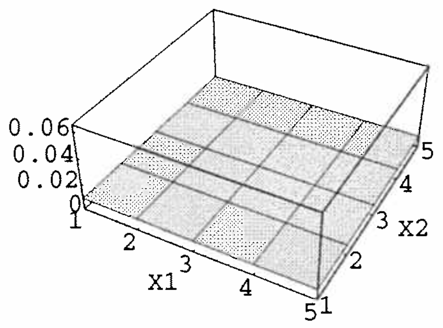
Distortion Matrix Field



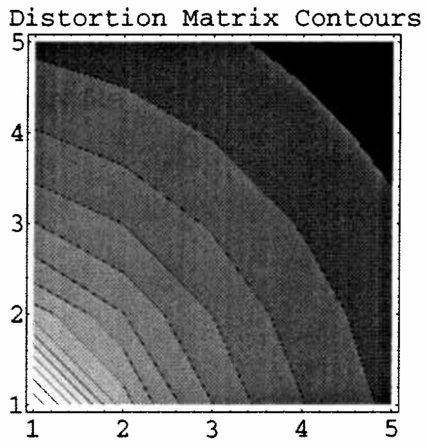
Distortion Matrix Field

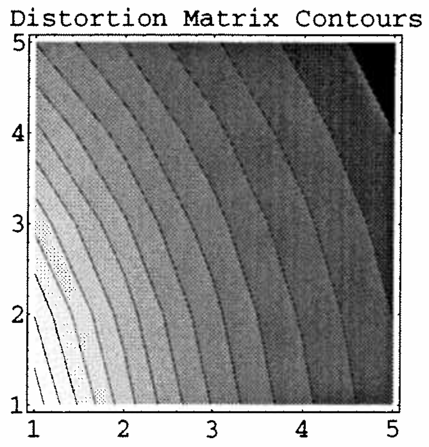


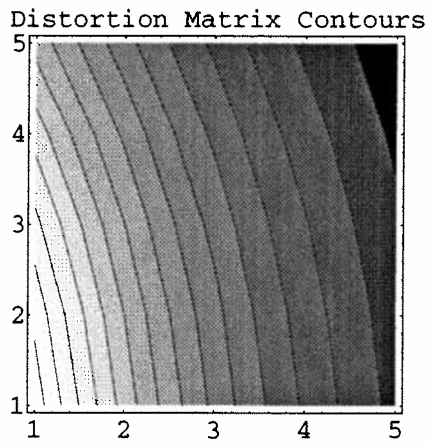
Distortion Matrix Field

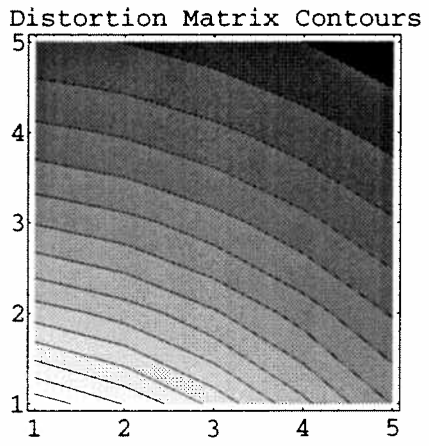


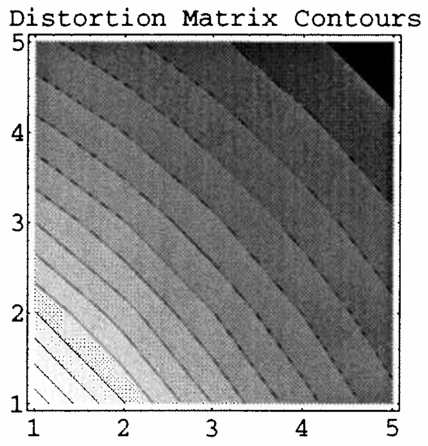
□ Distortion Matrix Contours

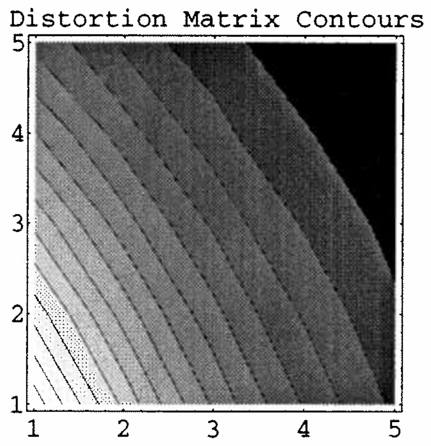


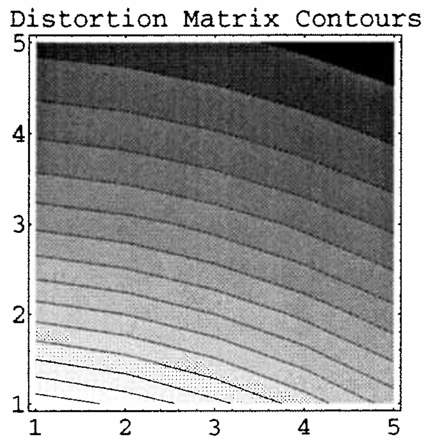


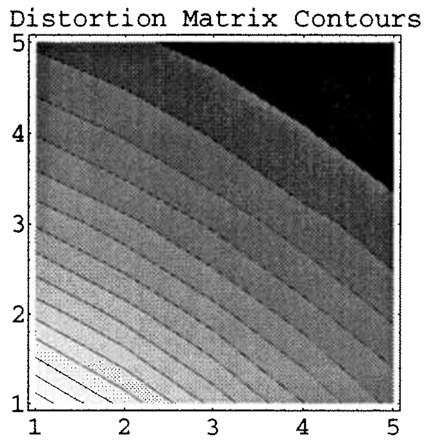


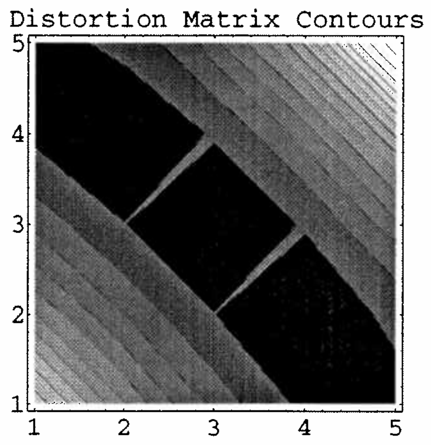






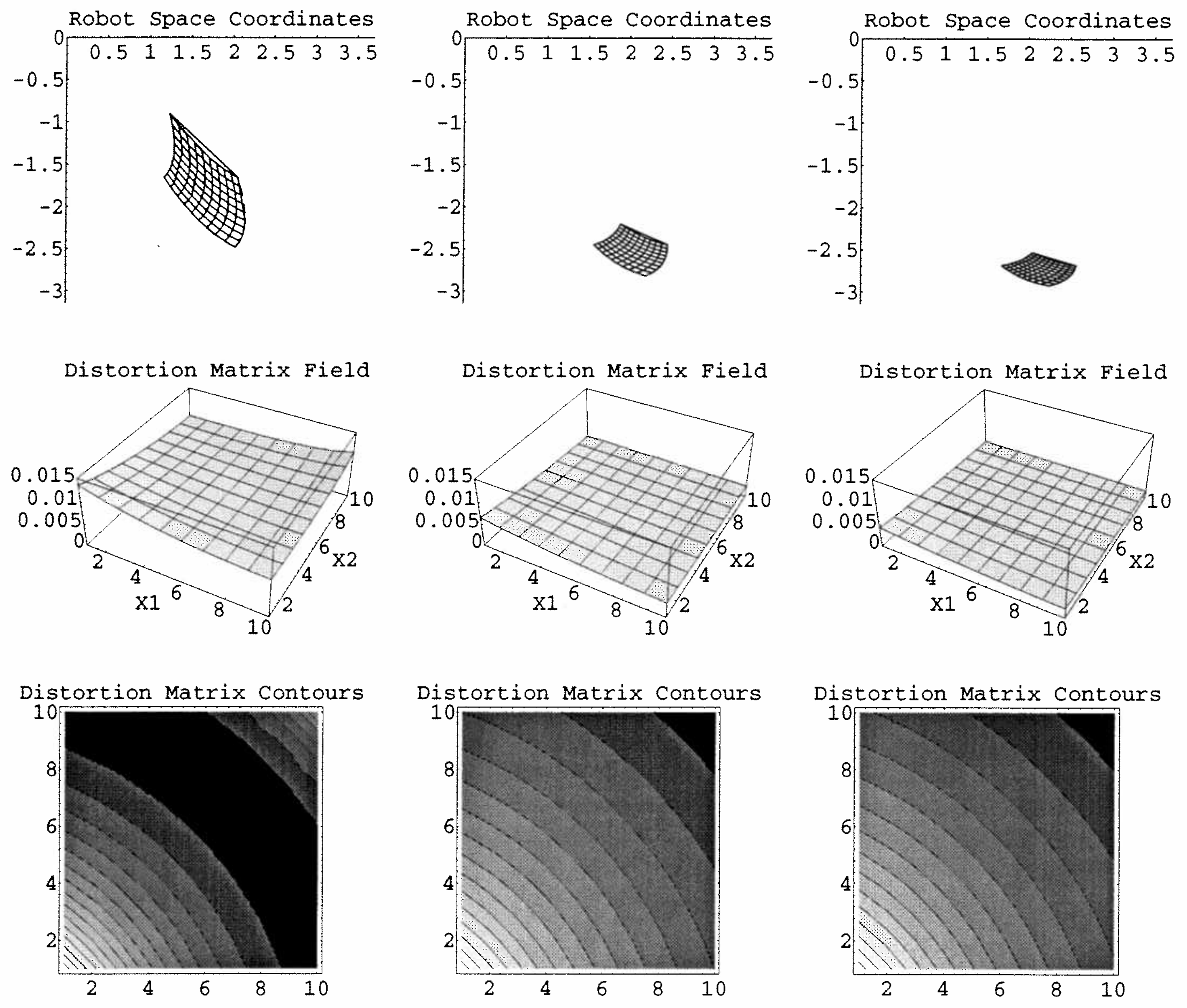




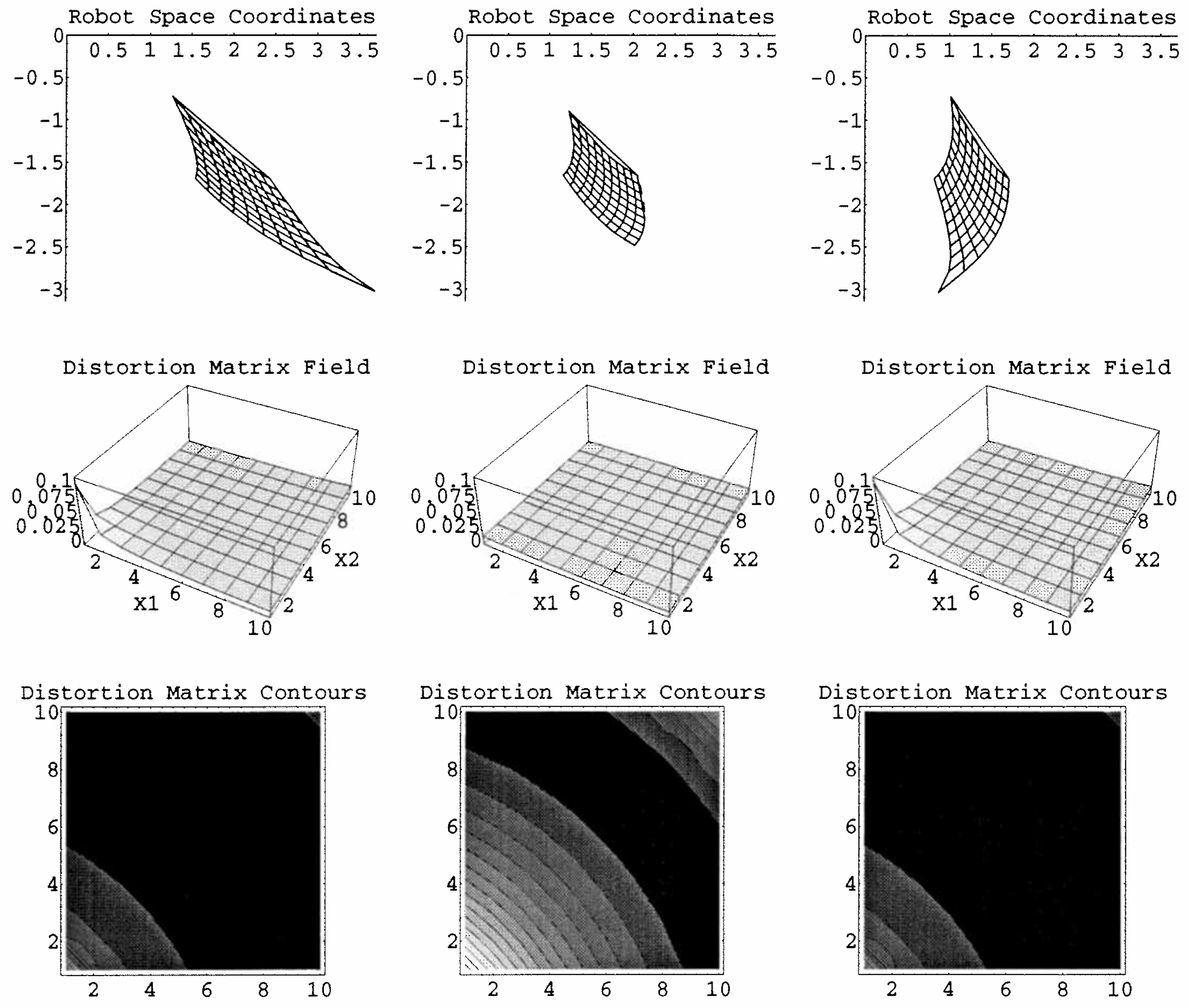


Appendix B: Robot Geometry

■ Graphics from *RobotDistortionEqual.ma*



■ Graphics from *RobotDistortionVarying.ma*



Appendix C: *Mathematica* Programs and Output

Justin Gallagher

Robot Space Coordinate Representation of Objects in Euclidean Space

Initialization and Definitions

▨ Initialization

```
<<Graphics/Graphics.m
```

```
Off[General::spell1,
    General::spell,
    ParametricPlot3D::ppcom,
    ParametricPlot::ppcom]
TRACK
```

▨ Robot Definition

☐ Link Angles

```
a = (x1^2 + x2^2 - l1^2 - l2^2) / (2 l1 l2);
b = -Sqrt[1 - ((x1^2 + x2^2 - l1^2 - l2^2) / (2 l1 l2))^2];
```

```
Theta2[{x1_,x2_},l1_,l2_] = ArcTan[a , b];
```

```
Theta1[{x1_,x2_},l1_,l2_] =
    ArcTan[x1,x2] -
    ArcTan[l1 + l2 (a / Sqrt[a^2 + b^2]),
           l2 Sin[ArcTan[a, b]] ];
```

☐ Link Joint Positions

```
point1[{x1_,x2_},l1_,l2_] =
    l1{ Cos[Theta1[{x1,x2},l1,l2]],
        Sin[Theta1[{x1,x2},l1,l2]]};
```

```
point2[{x1_,x2_},l1_,l2_] =
    l2 {Cos[Theta1[{x1,x2},l1,l2] + Theta2[{x1,x2},l1,l2]],
        Sin[Theta1[{x1,x2},l1,l2] + Theta2[{x1,x2},l1,l2]]}
```

☐ Robot Graphics

```
robot[{x1_,x2_}, l1_, l2_] := Graphics[Line[{
    {0,0},
    point1[{x1,x2},l1,l2],
    point1[{x1,x2},l1,l2] + point2[{x1,x2},l1,l2]
    }], AspectRatio->Automatic]
```

▨ Robot Space Representation of Objects

☐ Robot Space Coordinates (Theta1, Theta2)

```
robotSpace[{x1_,x2_},l1_,l2_] :=
        {Theta1[{x1,x2},l1,l2], Theta2[{x1,x2},l1,l2]}
```

▨ Grid Verticies Generator

Creates an m × n array of coordinate pairs, beginning at the point {x0, y0}.

```
pointsMaker[x0_,y0_,rows_,columns_] :=
    Table[{x0 + i,y0 + j},{j,0,rows},{i,0,columns}]
```

Reverses the order of every other row of coordinate pairs, beginning with the second row.

```
scramble[list_] :=
    Flatten[ Table[
        If[ EvenQ[i], Reverse[list[[i]]], list[[i]] ],
    {i,1,Length[list]} ] ,1]
```

Creates the reverse path of the robot across the grid.

```
RpointsMaker[x0_,y0_,rows_,columns_] :=
    Table[{x0 + columns - j, y0 + rows - i},
          {j,0,columns},{i,0,rows}]
```

Defines path of robot across grid as a function which can be parametrically plotted.

```
gridPath[list_,t1_] := list[[ 1 + Floor[t1] ]] /. t -> t1
```

Plots the grid representing the work space of the robot.

```
gridPlot[list_,t_] :=
    ParametricPlot[gridPath[list,t],{t,0,Length[list]-1},
AspectRatio->Automatic,PlotRange->{{0,15},{0,15}},
AxesLabel->{"X1","X2"},
PlotLabel->"Robot Work Space"];
```

▨ Distortion Analysis

■ Forward Diffeomorphism from Cartesian Space to Robot Space

□ Diffeomorphism

Converts work space coordinates into Robot Space joint angle coordinates.

```
F[{x1_,x2_},l1_,l2_] :=
    {Theta1[{x1,x2},l1,l2],Theta2[{x1,x2},l1,l2]}
```

□ System Jacobian

Precursor to the matrix used to redefine the dot product for Robot Space.

```
J[{x1_,x2_},l1_,l2_] =
    Transpose[{D[F[{x1,x2},l1,l2],x1],
    D[F[{x1,x2},l1,l2],x2]}];

J[{x1,x2},2,2] // Simplify;
```

□ Redefined Dot Product Matrix

Gives meaning to length and angle after the non-linear transformation to Robot Space.

```
M[{x1_,x2_},l1_,l2_] :=
    Block[ {Jn = N[J[{x1,x2},l1,l2]]},
            Transpose[Jn].Jn]
```

■ Area Calculations

Applies the diffeomorphism to the grid path.

```
distort[list_,l1_,l2_] :=
Table[F[list[[i]],l1,l2],{i,Length[list]}];
```

Calculates distortion of the grid area at each of the verties.

```
matrixField[list_,l1_,l2_] :=
    Table[
        Table[
            Sqrt[Det[M[list[[i]][[j]],l1,l2]]],
        {j,1,Length[ list[[1]] ]}],
    {i,1,Length[ list[[1]] ]}];
```

Numericly calculates the degree of distortion over the entire work space.

```
gridArea[l1_,l2_,x0_,y0_,rows_,columns_] :=
    NIntegrate[
        NIntegrate[
            Sqrt[ Det[ M[{x1,x2},l1,l2] ] ],
        {x1,x0,x0+columns} ],
    {x2,y0,y0+rows} ]
```

■ Graphing Tools

Conventional parametric plot for objects in the work space.

```
pPlot[object_,tmax_] := ParametricPlot[object,
    {t, 0, tmax}, PlotRange->All,
    AspectRatio->Automatic
    ];
```

Parametrically plots Robot Space coordinates.

```
thetaGridPlot[list_,t_] :=
ParametricPlot[gridPath[list,t],
    {t, 0, Length[list]-1}, AspectRatio->Automatic,
    PlotRange->{{0,3.7},{0,-Pi}},
    PlotLabel->"Robot Space Coordinates"];
```

Plots the magnitude of Sqrt[Det[M1[{x1,x2}]]] as altitudes over the work space.

```
fieldPlot[list_,height_] := ListPlot3D[list,
    PlotRange->{0,height}, AxesLabel->{"X1","X2",""},
    PlotLabel->"Distortion Matrix Field",
    AspectRatio->Automatic];
```

Contour line representation of the previous graphic.

```
gridContour[list_] :=
    Show[ContourGraphics[list,Contours->15],
    PlotRange->All,AxesLabel->{"X1","X2"},
    PlotLabel->"Distortion Matrix Contours"];
```