2000

Computer-Aided Synthesis of Higher Pairs Via Configuration Space Manupulation

Min-ho Kyung

Elisha Sacks
Purdue University, eps@cs.purdue.edu

Report Number:
00-012

COMPUTER-AIDED SYNTHESIS OF HIGHER PAIRS VIA CONFIGURATION SPACE MANIPULATION

Min-ho Kyung
Elisha Sacks

Computer Science Department
Purdue University
West Lafayette, IN 47907

CSD TR #00-012
August 2000

Computer-aided synthesis of higher pairs via configuration space manipulation

Min-ho Kyung and Elisha Sacks
Computer Science Department
Purdue University
West Lafayette, IN 47907, USA

Abstract

We describe a parametric synthesis algorithm for planar mechanical systems comprised of higher kinematic pairs in which each part translates along a fixed axis or rotates around a fixed point. Kinematic function is computed from the CAD models of the parts and is represented graphically as configuration spaces. The designer uses the mouse to request changes in the configuration spaces. The program computes parameter values that achieve the changes. The computation is iterative: the program repeatedly linearizes the mapping from design parameters to kinematics around the current values, pseudo-inverts the linear mapping, and performs a small parameter modification that moves the system toward the desired kinematics. At each iteration, the program matches the current kinematics against the initial kinematics. If it detects an unintended change, it backs up, adds kinematic constraints that prevent the change, and resumes iteration.

1 Introduction

We describe research in computer-aided synthesis of higher kinematic pairs. Kinematic synthesis is a key step in the mechanical design cycle: an iterative process that starts with a design concept and ends with a detailed design. The designer selects a design concept (a linkage, a ratchet, a Geneva pair), specifies its geometry parametrically (four links, triangular teeth, circular pin), and picks parameter values (link lengths, tooth dimensions, pin radius). The last step is tolerance allocation, typically parametric tolerances for functional features and geometric tolerances for assembly features. Each step has an analysis and a synthesis component. The designer makes changes, derives their impact, and decides whether to advance to the next step or to return to a prior step. The cycle ends when the design meets the specifications.

This paper addresses the parametric design step in the design cycle. The parts are specified in terms of parameters with ranges of allowable values. The Cartesian product of the parameter ranges, called the design space, defines the possible designs. The design task is to select parameter values that assure correct and ideally optimal function. Most design is parametric because most designs are revision of prior designs. Examples of non-parametric design are replacing a linkage by a cam mechanism or changing the number of teeth on a gear.

The traditional method of parametric design is exhaustive manual search of the design space. This is often impractical, especially when there are more than three parameters. The designer must examine many points to assure that a good design has not been overlooked. Each point requires a time consuming analysis. The search is especially difficult when the mechanical function is sensitive to small perturbations in the parameter values, which is common.

The impracticality of exhaustive search has led researchers to pose parametric design as an optimization problem [1]. The design goals are encoded in an objective function that is maximized subject to the design constraints. The challenge is to formulate objective functions, constraints, and optimization algorithms for specific design tasks. The objective function must balance performance, quality, and cost according to the design priorities.

Prior research applies this methodology to mechanical systems with permanent part contacts. The constraints are a fixed set of algebraic equalities, hence nonlinear constrained optimization [2] is applicable. Most research addresses linkage design [3, 4]. The parameters are the configurations of the joint attachment points. The constraints are the joint equations. The designer picks the objective function, for example "minimize deviation from linear motion" or "maximize piston travel." Angeles et al [5, 6] synthesize and optimize cam pairs. The parameters describe the part profiles. The constraints specify the follower configuration and its derivatives at important cam angles. The objective function is the overall deviation of the follower from a prescribed path.

We have developed a synthesis algorithm for planar mechanical systems comprised of higher kinematic pairs. The parts are required to translate along a fixed axis or to rotate around a fixed point. These higher pairs are com-

mon in mechanical design. Gears and cams are used in all types of mechanical systems. Ratchets, indexers, and other specialized pairs are used in low-torque precision mechanisms, such as sewing machines, copiers, cameras, and VCRs. Higher pairs are more versatile than lower pairs because they can realize multiple functions. They are usually cheaper, lighter, more compact, and more robust than actuators. When manufacturing variation and wear are taken into account, lower pairs must be analyzed as higher pairs, as in pin joints with play.

Higher pairs pose unique synthesis problems because they are much more complex than lower pairs. Instead of a few equality constraints, there are many equality and inequality constraints. When two part features (edges or vertices in planar systems; faces too in spatial systems) touch, the part motions are constrained to prevent them from overlapping. When the contact point shifts to another feature, a different set of constraints takes effect. The challenge of higher-pair synthesis is to implement a sequence of contacts that performs the mechanical function, while avoiding undesirable contacts. There are thousands of possible contacts in typical pairs, which leads to a combinatorial explosion of contact sequences.

We address this challenges within our configuration space framework of mechanical design [7]. The designer inputs a parametric model of a mechanical system and specifies initial parameter values. The synthesis program computes and displays configuration spaces for the kinematic pairs. These spaces encode the initial kinematics: feature contacts appear as contact curves and contact changes appear as curve adjacencies. Design objectives are expressed as changes in the contact curve geometry. The designer inputs the objectives with the mouse and the program achieves them by changing the design parameters.

The program monitors updates for unintended kinematic changes. Contact curves that were not selected by the designer will often change shape because they share parameters with the selected curves. A sufficiently large change can cause a pair of disjoint curves to intersect or vice a versa. These events can cause structural changes in the system kinematics, such as jamming. The program detects these changes and modifies the parameter update to prevent them.

Our algorithm builds on prior work by Caine [8] who designs planar part feeders via configuration space manipulation. The kinematic function is represented by a partial part/feeder configuration space. The designer requests a single change in the configuration space and the program changes the feeder geometry accordingly. The models are non-parametric and the modifications are heuristic. Structural tive changes are not addressed.

The main steps in the synthesis algorithm are configuration space computation, parameter updates, and structural change handling. The first step is described elsewhere [9]. We describe the other steps in Sections 3 and 4, after presenting an example in Section 2.

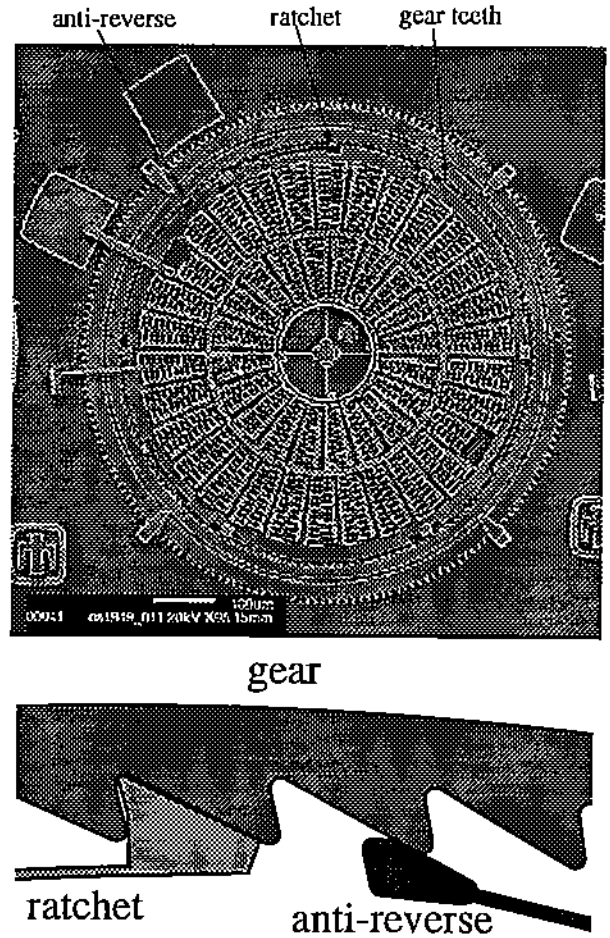

Figure 1: Torsional ratcheting actuator: (top) SEM image courtesy of Sandia National Laboratories Intelligent Micromachine Initiative www.mems.sandia.gov; (bottom) Detail of CAD model.

2 Synthesis example

We illustrate the synthesis algorithm on a MEMS (micro electro-mechanical system) torsional ratcheting actuator from Sandia National Laboratory [10]. The mechanism consists of a drive wheel, three ratchet pawls, a ring gear, and three anti-reverse pawls (Figure 1). The drive wheel is rotated 2.5° counterclockwise by an electro-static comb drive (not shown). The ratchet pawls, which are mounted on the drive wheel with pin joints, engage the inner teeth of the ring gear and rotate it counterclockwise. When the

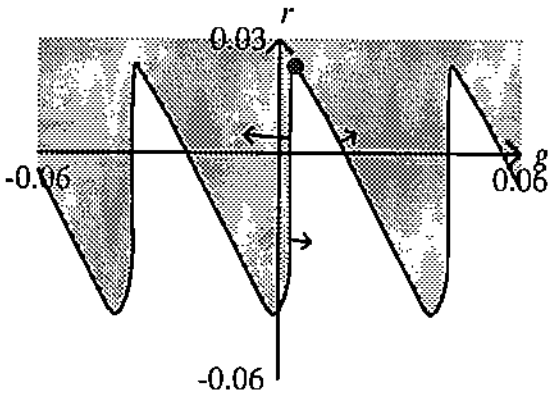

Figure 2: Detail of gear/ratchet configuration space.

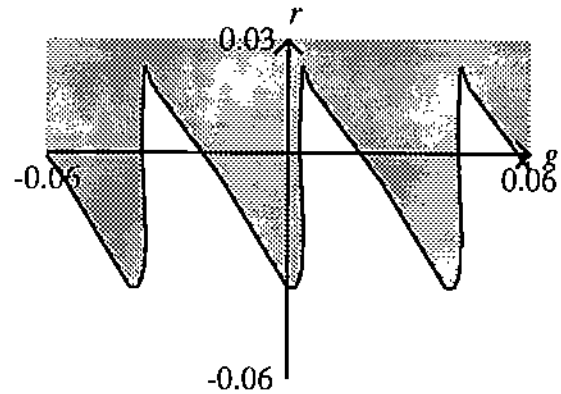

Figure 3: Improved gear/ratchet configuration space.

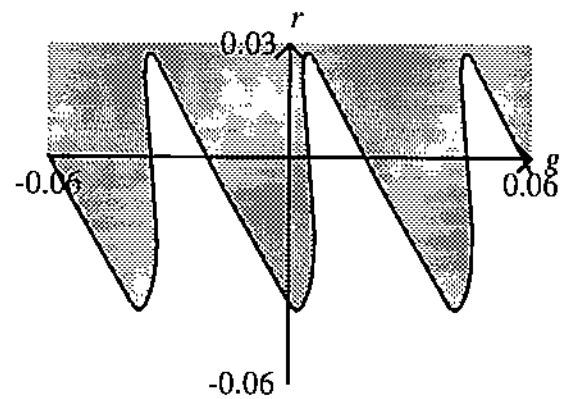

Figure 4: Satisfactory gear/ratchet configuration space.

voltage drops, torsional springs restore the drive wheel to its start orientation, which disengages the ratchet pawls. The anti-reverse pawls prevent the ring gear from rotating clockwise. The cycle repeats rapidly enough that the ring gear rotates with near constant angular velocity. Its outer, involute teeth drive a load via a transmission.

The synthesis tasks are to realize the intended function in the nominal design and to ensure correct function whenever the parameter values fall within specified tolerance limits. There are 35 design parameters, such as the slope of the ratchet teeth, the radii of the ring gear tooth fillets, and the part centers of rotation.

The gear/ratchet configuration space shows the kinematics of the three gear/ratchet pairs, which are identical except for a phase shift (Figure 2). The horizontal axis represents the angle between the drive wheel and the gear. The vertical axis represents the ratchet orientation in the global frame. The configuration space is partitioned into free space where the parts do not touch (white area) and blocked space where they overlap (grey area), separated by contact space where they touch (black curves). The dot marks the displayed configuration in Figure 1 where the ratchet is driving the gear counterclockwise. The vertical contact curve to the left represents the contact between the right side of a gear tooth and the ratchet tip, which prevents the gear from rotating clockwise relative to the driver. If the gear were to rotate clockwise, it would enter blocked space, which is physically impossible. The diagonal contact curve to the right represents the contact between the left side of a tooth and the ratchet back, which allows the gear to disengage the ratchet via counterclockwise rotation. The gear angle and the ratchet angle both increase as the configuration follows the contact curve.

The configuration space reveals a design flaw: the vertical contact curve slopes slightly to the right. This means that the ratchet can rotate counterclockwise, escape the gear, and jump to the next tooth. Friction will prevent this from happening until the driver torque reaches a critical value. Manufacturing variation can exacerbate the problem. This failure mode has been observed intermittently in prototype actuators.

The designer uses the synthesis program to change the slope of the vertical curve in the gear/ratchet configuration space. He specifies new locations for two configurations on the vertical curve and for one configuration on the diagonal curve to the right. The changes are represented by arrows, called draggers, whose tails are the old configurations and whose heads are the new ones. The program computes parameter values that achieve these goals (Figure 3). The designer assigns draggers several more times until the configuration space is satisfactory (Figure 4).

3 Parameter updates

The parameter update algorithm computes design parameter values that achieve specified kinematic changes. The input is a set of draggers. Each dragger consists of a contact curve, a start point p_0 on the curve, and a goal

point $\mathbf{p}_0 + \delta\mathbf{p}$. Contact curves have the form $C(\mathbf{p}, \mathbf{u}) = 0$ where $\mathbf{p}$ is the two configuration space coordinates (g, r in the gear/ratchet) and $\mathbf{u}$ is the vector of design parameters. The start point, $\mathbf{p}_0$, satisfies $C(\mathbf{p}_0, \mathbf{u}_0) = 0$ with $\mathbf{u}_0$ the initial parameter values. The program computes a parameter update $\delta\mathbf{u}$ for which the curve goes through the goal point, $C(\mathbf{p}_0 + \delta\mathbf{p}, \mathbf{u}_0 + \delta\mathbf{u}) = 0$.

The contact equations are solved numerically because a closed-form solution is impractical. The program performs a sequence of small parameter updates governed by the linearized contact equation

$$\frac{\partial C}{\partial \mathbf{p}}(\mathbf{p}_0, \mathbf{u}_0)\delta\mathbf{p} + \frac{\partial C}{\partial \mathbf{u}}(\mathbf{p}_0, \mathbf{u}_0)\delta\mathbf{u} = 0. \qquad (1)$$

There is one equation per dragger. The equations are normally under constrained because a typical number of draggers is less than five, while a typical number of design parameters is twenty. But they are over constrained when the draggers are inconsistent or when there are more draggers than parameters. We compute an exact, minimum-norm solution if possible and a least-squares solution otherwise, using singular value decomposition.

Sacks and Joskowicz [11] developed a preliminary version of this algorithm that updates a single vertical dragger. Gleicher uses a similar technique, which he calls differential constraint satisfaction, for interactive computer graphics [12].

The design parameters are updated from $\mathbf{u}_0$ to $\mathbf{u}_0 + \delta\mathbf{u}$ and the new configuration space is computed. If every dragger head lies on its new contact curve to a tolerance, the computation ends successfully. Otherwise, each dragger tail is updated to the first intersection point between the dragger and its new contact curve. If the new tails are the same as the old tails to a tolerance, the computation fails and the designer must pick different draggers. Otherwise, the next $\delta\mathbf{u}$ is computed.

The user inputs draggers with two mouse clicks. The dragger curve is the closest contact curve to the first click. The tail is the closest point on the curve to the click. The head is the second click. The program obtains the contact equations from a table indexed by feature type (line segment, circular arc) and by motion type (translation, rotation). A typical entry is rotating arc/translating line. The complete table appears in prior work [9].

The table entries have the form $C(\mathbf{p}, \mathbf{f}, \mathbf{g}) = 0$ where $\mathbf{f}$ and $\mathbf{g}$ are parameters that specify the touching features. The parameters of a line are the coordinates of its endpoints. The parameters of an arc are its center coordinates and radius. These parameters are symbolic expressions, $\mathbf{f}(\mathbf{u})$ and $\mathbf{g}(\mathbf{u})$, in the design parameters. The linearized

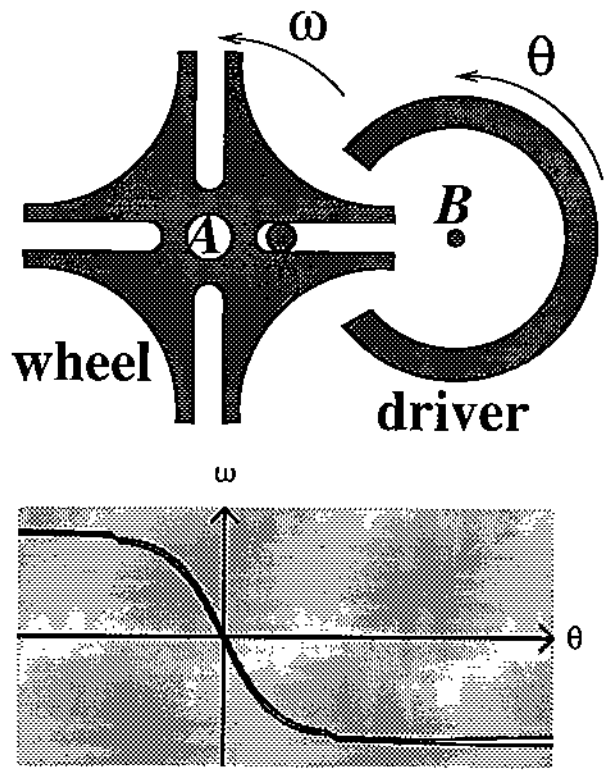

Figure 5: Geneva pair and configuration space detail.

contact equations are obtained by the chain rule

$$\frac{\partial C}{\partial \mathbf{p}}\delta\mathbf{p} + \left(\frac{\partial C}{\partial \mathbf{f}}\frac{\partial \mathbf{f}}{\partial \mathbf{u}} + \frac{\partial C}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{u}}\right)\delta\mathbf{u} = 0. \qquad (2)$$

The term in parenthesis is $\partial C/\partial\mathbf{u}$ in Equation (1). The derivatives $\partial C/\partial\mathbf{f}$ and $\partial C/\partial\mathbf{g}$ come from a second table indexed by feature and motion type. The derivatives $\partial\mathbf{f}/\partial\mathbf{u}$ and $\partial\mathbf{g}/\partial\mathbf{u}$ are computed symbolically.

4 Structural changes

A parameter update that satisfies the dragger constraints can change the system kinematics in undesirable ways. The synthesis program detects changes by matching the new configuration space against the old one. If they have the same structure (defined below), it accepts the update. If not, it adds draggers that prevent the change and recomputes the update. Alternately, the designer can accept the original update if the change is harmless.

We illustrate change detection on a Geneva pair (Figure 5). Rotating the driver causes intermittent rotation of the wheel with drive periods where the driver pin engages the wheel slots and with dwell periods where the outer

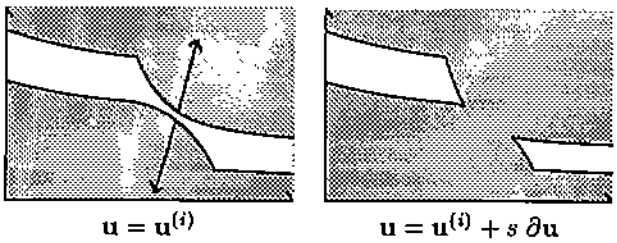

Figure 6: Geneva structural change due to parameter update.

driver arc engages the concave wheel arcs. The configuration space shows this function geometrically. The free space forms a single channel that wraps around the horizontal and vertical boundaries (four shifted copies of the detail in the figure). As the driver rotates, the configuration follows the channel. The wheel rotates in the diagonal segments where the driver pin pushes the wheel slots. It dwells in the horizontal segments where the driver arc engages the wheel arcs.

The design task is to minimize the wheel play: the angle by which the wheel can rotate when the driver is stationary. In configuration space, this is the vertical distance between the bottom and top contact curves (Figure 6). The designer places draggers on the horizontal segments because the play is greatest there. The parameter update reduces the play, but now the channel closes and the free space breaks into multiple components. The modified pair geometry is close to the original, but the configuration space structure is very different.

We have developed a heuristic configuration space equivalence test that quickly finds all structural changes. False positives are possible, but have not been observed and cause no damage beyond increasing the number of parameter updates. Two spaces are equivalent when they have the same number of connected components and these components are pairwise equivalent. Two components are equivalent when they have the same number of boundary curves and there exists a cyclic ordering of the second boundary for which every curve in the first boundary is equivalent to the corresponding curve in the second boundary. Two curves are equivalent when they are generated by the same pair of part features.

Structural changes can involve any number of contact curves, hence can be very complicated. We assume that every change consists of a finite sequence of basic changes where two components merge or where one component splits (Figure 7). For example, the Geneva undergoes a split. The program finds and prevents the first basic change in the sequence. A merge is prevented by two draggers whose tails are the closest points on the two components and whose directions are the normals that point into free

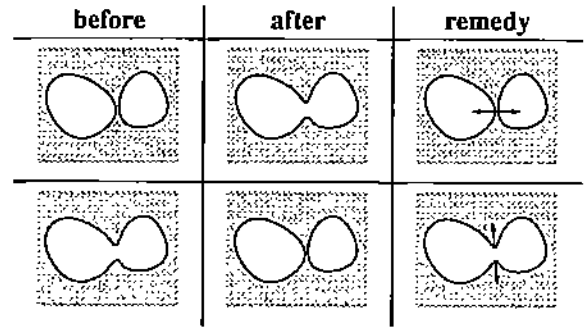

Figure 7: Basic change types: merge and split.

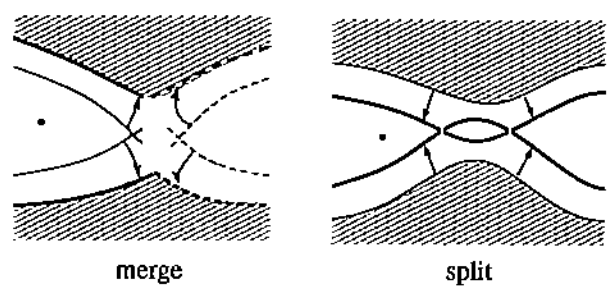

Figure 8: Basic change tests.

space. A split is prevented by two draggers whose tails are the closest points on the narrow neck and whose directions are the normals that point into blocked space.

Suppose a structural change occurs when $\mathbf{u}$ is updated from $\mathbf{u}_0$ to $\mathbf{u}_0 + \delta\mathbf{u}$. The program searches the line segment $\mathbf{u}_0 + s\delta\mathbf{u}$, $s \in [0, 1]$ for an interval $[s_a, s_b]$ in which the first basic change occurs. It performs bisection search starting with $[0, 1]$. At each step, the midpoint space is compared to the initial space. If they have the same structure, s_a is updated to the midpoint and if not s_b is updated. When the interval is narrow enough (0.01 units by default), the program assumes that it contains one basic change. It adds appropriate draggers and resumes the parameter update iteration from $\mathbf{u}_0 + s_a\delta u$.

Basic changes are detected by comparing pairs of contact curves at s_a with the corresponding pairs at s_b. A merge occurs when two curves intersect at s_a, but are disjoint at s_b; a split occurs when they are disjoint at s_a, but intersect at s_b (Figure 8). Several merges or splits can occur simultaneously when the parts, hence the configuration space, contains repeated patterns. For example, the Geneva undergoes eight splits—at the four tops and bottoms of the diagonal configuration space channels—due to the wheel symmetry. Simultaneous changes are detected by comparing all pairs of curves and grouping the changes.

5 Conclusions

We have described a kinematic synthesis program for planar mechanical systems based on configuration space manipulation. The inputs are a parametric system model and a list of kinematic design changes, expressed as geometric changes in the configuration space of kinematic pairs. The program updates the system parameters to achieve the requested changes, while avoiding unintended kinematic changes. We have illustrated the algorithm on a MEMS actuator with several kinematic pairs and on a Geneva pair.

We see several directions for further work. The top priority is to validate the synthesis algorithm on real-world applications. The next priority is to add functionality. Although quite versatile, draggers are not the best way express every design change. For example, the actuator design shows that rotating a contact curve around a point is awkward with draggers. We have developed a rotator constraint for this task and have added it to the program. We can reduce false positives in the matcher by augmenting the symbolic equivalence criterion with a contact curve shape criterion. We are working on a completeness proof for the basic change types, perhaps using a few more types.

The greatest technical challenges are to extend the synthesis algorithm to planar parts with three degrees of freedom and to spatial parts. In the planar case, we have a configuration space computation program [13] and can compute parameter updates as before, but lack a structural change algorithm for these three-dimensional spaces. In the spatial case, we have a configuration space computation program for parts that move along fixed axes [14] and can use the current parameter update and structural change algorithms. All that is lacking is a computer implementation of the linear contact constraints for the various spatial contacts.

Acknowledgments

Sacks is supported by NSF grant CCR-9617600, by the Purdue Center for Computational Image Analysis and Scientific Visualization, by a Ford University Research Grant, by the Ford ADAPT 2000 project, and by grant 98/536 from the Israeli Academy of Science. Kyung is supported by NSF grant CCR-9617600.

References

[1] Panos Papalambros and Douglass Wilde. *Principles of Optimal Design*. Cambridge University Press, 1988.

[2] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C*. Cambridge University Press, Cambridge, England, 1990.

[3] G. Erdman, Arthur. *Modern Kinematics: developments in the last forty years*. John Wiley and Sons, 1993.

[4] Rajan Ramaswamy. *Computer Tools for Preliminary Parametric Design*. PhD thesis, Massachussetts Institute of Technology, 1993.

[5] Jorge Angeles and Carlos Lopez-Cajun. *Optimization of Cam Mechanisms*. Kluwer Academic Publishers, Dordrecht, Boston, London, 1991.

[6] Max Gonzales-Palacios and Jorge Angeles. *Cam Synthesis*. Kluwer Academic Publishers, Dordrecht, Boston, London, 1993.

[7] Leo Joskowicz and Elisha Sacks. Computer-aided mechanical design using configuration spaces. *Computing in Science and Engineering*, 1(6):14–21, 1999.

[8] Michael E. Caine. The design of shape from motion contraints. AI-TR 1425, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA, 02139, 1993.

[9] Elisha Sacks and Leo Joskowicz. Computational kinematic analysis of higher pairs with multiple contacts. *Journal of Mechanical Design*, 117(2(A)):269–277, June 1995.

[10] Stephen M. Barnes, Samuel L. Miller, M. Steven Rodgers, and Fernando Bitsie. Torsional ratcheting actuation system. In *Third International Conference on Modeling and Simulation of Microsystems*, San Diego, CA, 2000.

[11] Leo Joskowicz and Elisha Sacks. Configuration space computation for mechanism design. In *IEEE International Conference on Robotics and Automation*. IEEE Computer Society Press, 1994.

[12] Michael Gleicher. *A Differential Approach to Graphical Interaction*. PhD thesis, Carnegie Mellon University, 1994.

[13] Elisha Sacks. Practical sliced configuration spaces for curved planar pairs. *International Journal of Robotics Research*, 18(1):59–63, January 1999.

[14] Ku-Jim Kim, Elisha Sacks, and Leo Joskowicz. Kinematic analysis of spatial fixed-axes pairs using configuration spaces. Technical Report CSD-TR 99-036, Purdue University, 1999.