Parameter Synthesis of Higher Kinematic Pairs

Min-Ho Kyung and Elisha Sacks (corresponding author)

Computer Science Department
Purdue University
West Lafayette, IN 47907, USA
phone: 1-765-494-9026; fax: 1-765-494-0739

Abstract

We describe a parameter 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. This is an important class of systems and is not addressed by prior synthesis research. Kinematic function is computed from the CAD models of the parts and is represented graphically as configuration spaces. Design flaws appear as incorrectly shaped contact curves, as incorrectly placed intersection points, and as incorrect curve sequences. The designer reshapes the faulty curves with the keyboard mouse. The synthesis algorithm computes new parameter values that realize the changes without harmful side effects, using a novel form of constrained optimization. The algorithm has been tested on industrial applications.

Key words: kinematic synthesis, higher pair, configuration space.

abbreviated title: Parameter Synthesis of Higher Pairs.

Introduction

This paper describes a synthesis algorithm for higher kinematic pairs based on configuration space construction and on constrained optimization. Kinematic synthesis is an iterative process in which a designer devises a mechanical system that performs a specified function. The first step is conceptual design where the designer determines the system structure (linkage, gear chain, ratchet). The second step is parameter synthesis where the designer builds a parametric model of the system

and chooses the parameter values (link lengths, gear radii, engagement angle). The last step is tolerancing where the designer allows for manufacturing variation by calculating error bounds on the nominal parameter values that guarantee correct function. At each step, 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.

We address the parameter synthesis step in this cycle. The parts are specified in terms of parameters with ranges of legal values. The Cartesian product of the ranges defines the set of possible designs. The task is to select parameter values that realize the correct (and ideally optimal) function. Most synthesis is parametric because most designs are revisions of prior designs. We require that the mapping from parameters to parts be continuous. Discrete mappings are uncommon in kinematic and are not amenable to our approach. Replacing a linkage by a cam mechanism is an example of non-parametric design, while changing the number of teeth on a gear is an example of discrete parametric design.

The traditional method of parameter synthesis is for the designer to search the parameter space. This is often impractical. The designer must examine many points to ensure 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 direct search has led researchers to pose parameter synthesis 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 fixed contact topologies, meaning that the same contact constraints apply throughout the work cycle. These systems are comprised of lower kinematic pairs and of specialized higher pairs. The constraints are a fixed set of algebraic equalities, hence nonlinear constrained optimization is applicable. Most research addresses linkage design [2,3]. 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.” An-

geles et al [4,5] synthesize and optimize fixed-contact 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 with contact changes. The parts are represented in a boundary representation comprised of line and circle segments. These shapes suffice for almost all engineering applications. Each part is required to translate along a fixed axis or to rotate around a fixed point. Hence, the system is comprised of higher pairs with two degrees of freedom per pair. These pairs are common 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.

Systems with contact changes pose unique synthesis problems because they are much more complex than systems with fixed contacts. Instead of a few equality constraints, there are many equality and inequality constraints. When two part features (edges or vertices in planar 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 synthesis challenge 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 system contact sequences.

Our synthesis research builds upon our work on kinematic analysis. We analyze the kinematic function of a mechanical system by constructing the configuration spaces of its kinematic pairs. These spaces encode the kinematic function in a geometric format that highlights contacts, contact changes, and contact sequences. Contacts are represented as configuration space curves, so contact changes appear as curve intersection points and contact sequences appear as paths comprised of curve segments. We have developed a fast, robust configuration space construction algorithm [6] and have shown that designers can analyze complex systems by examining their configuration spaces [7–9].

We use configuration spaces to extend optimization based design to systems with

contact changes. The designer constructs a parametric model of a mechanical system, selects initial parameter values, and examines the resulting configuration spaces. Design flaws appear as incorrectly shaped contact curves, as incorrectly placed intersection points, and as incorrect curve sequences. We have found that an effective way to correct such flaws is for the designer to reshape the faulty contact curves and for a synthesis algorithm to compute new parameter values that realize the changes without harmful side effects. The computation is a novel form of constrained optimization that is the technical contribution of our research. We have developed interactive software that implements this paradigm and have tested it on several industrial applications.

The only prior work on kinematic synthesis with contact changes is by Caine [10] 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. Our algorithm goes beyond Caine's in several important ways. It handles simultaneous design changes in a mechanical system comprised of multiple higher pairs, versus one change in one pair. It handles any parametric model, versus a fixed parameterization of the contact features. Most importantly, it prevents harmful side effects, which Caine does not address.

The rest of the paper is organized as follows. The next section presents three scenarios that illustrate the synthesis algorithm and the configuration space framework. The following sections describe the synthesis algorithm. The final section contains conclusions and plans for future work.

Parameter synthesis scenarios

We present three parameter synthesis scenarios involving two mechanisms. The first two scenarios introduce higher pairs, parametric models, configuration spaces, and design modifications. The third scenario introduces side effect prevention.

Dennis clutch mechanism

The first two scenarios involve a Dennis clutch mechanism that consists of an arm, a ratchet, a cam, and a pawl (Figure 1a). The arm is attached to a fixed frame with

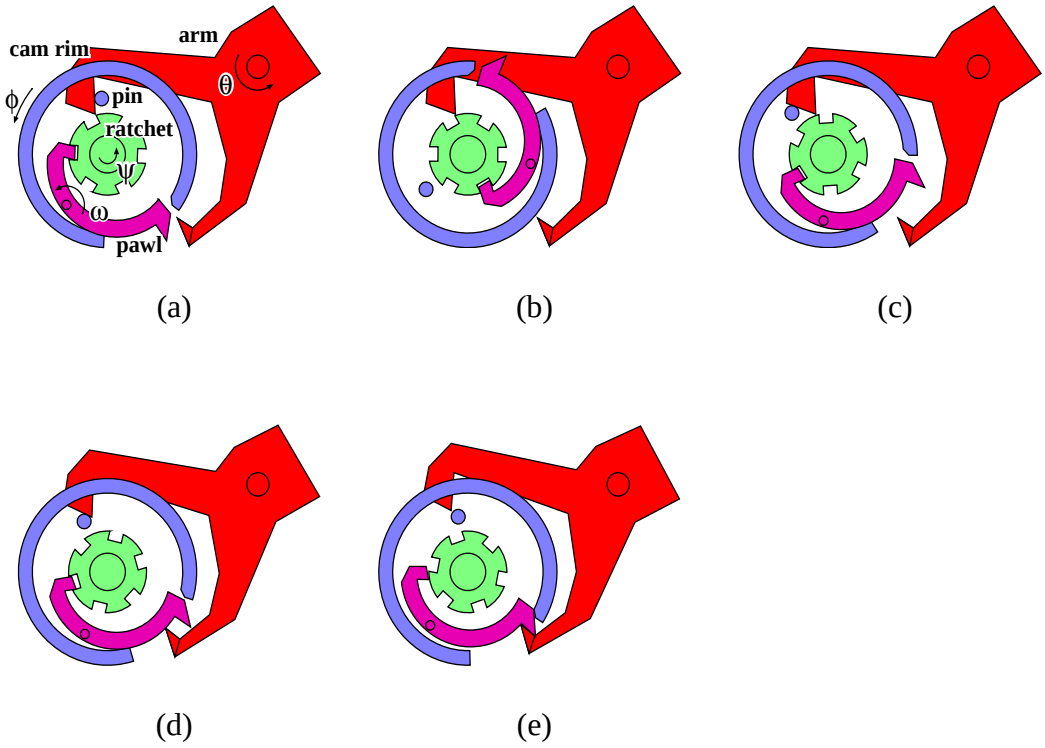


Fig. 1. Dennis clutch mechanism.

a pin joint. The ratchet is mounted on a rotating input shaft. The cam consists of a pin and a rim attached to a plate (not shown), and is mounted on an output shaft. The pawl is attached to the cam plate via a pin joint and is spring loaded to rotate clockwise.

The intended function of the mechanism is as follows. The operator rotates the arm counterclockwise, which releases the pawl (Figure 1a). The pawl rotates clockwise until its tip engages a ratchet slot. The ratchet drives the cam via the pawl (Figure 1b) until the the cam pin hits the upper finger of the arm (Figure 1c). The cam pin rotates the arm clockwise until its lower finger hits the pawl (Figure 1d), disengages the pawl from the ratchet, and blocks further rotation of the cam (Figure 1e).

Parametric model

We parameterize the functional features of the mechanism with 26 parameters. Figure 2 shows our parameterization of the pawl and the ratchet. The pawl parameters are the angles α and β, the radii r and s, and the coordinates of the points $p_1, \ldots, p_7$. The ratchet parameters are the tooth angle t, depth d, and radius e, and the coordi-

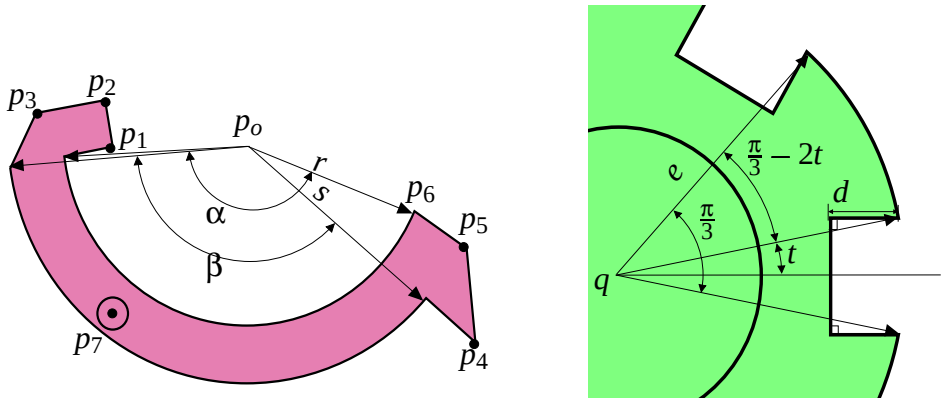


Fig. 2. Pawl and ratchet design parameters.

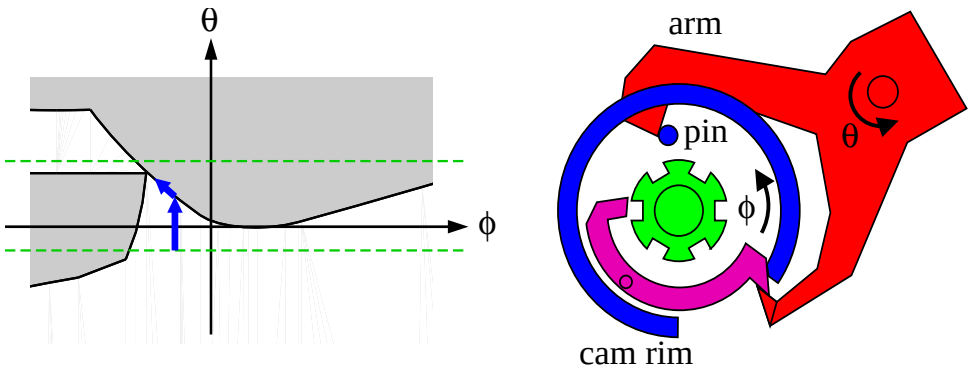
nates of the center of rotation q.

Configuration space

We perform a kinematic analysis of the initial design by constructing configuration spaces for its four kinematic pairs. Figure 3a shows the cam/arm configuration space. The horizontal axis represents the cam orientation, ϕ, and the vertical axis represents the arm orientation, θ. The lower horizontal line marks the initial arm orientation and the upper line marks the orientation at which it releases the pawl. 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 interact (black curves). The contact space consists of curves that represent contacts between pairs of part features.

The configuration space encodes the pair kinematics. The vertical arrow through free space marks the motion path in which the arm rotates until the upper finger contacts the cam pin. The diagonal arrow through contact space marks the coupled motion in which the arm rotates the cam. The motion occurs along a single contact curve that represents contact between the cam pin and the inner line segment of the upper finger. The curve slope equals the instantaneous angular velocity transmission ratio.

The configuration space reveals a design flaw: the coupled motion begins below the pawl release line. When the arm rotates counterclockwise to release the pawl, its upper finger hits the cam pin and rotates the cam clockwise, which pushes the cam rim against the pawl (Figure 3b). The pawl blocks on the lower finger of the arm, which jams the mechanism.



(a) (b)

Fig. 3. Cam/arm pair: (a) configuration space; (b) jamming configuration.

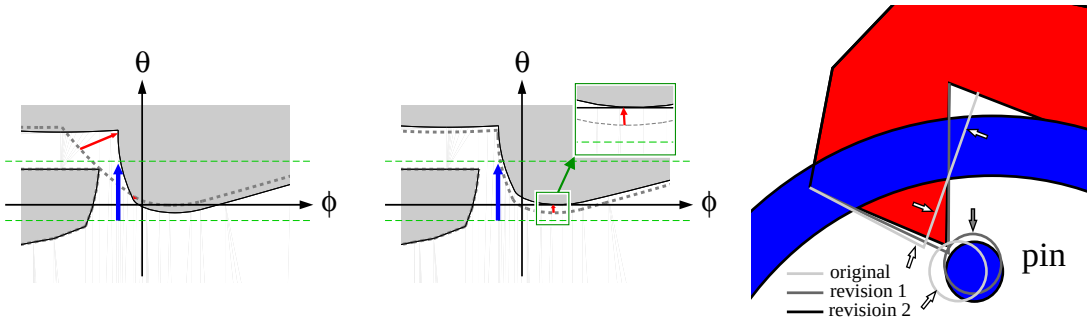
Design modification

We decide to correct the design flaw by forcing the arm to release the pawl before it hits the cam pin. We ask the synthesis program to change the faulty contacts and it updates the parameters accordingly. Without the program, we would have to identify the relevant part contacts, to determine the parameters that effect them, and to search the 26 dimensional parameter space for values that fix the flaw. The solution is hard to find: it occupies a small region of parameter space and several parameters have to vary in unison to reach it. These difficulties are the norm in higher pair design and are much greater in larger designs.

A change is specified as a dragger: an arrow whose tail lies on a contact curve and whose head is a configuration that should lie on that curve. Figure 4a shows how we fix the faulty curve with a long diagonal dragger. The original contact space is drawn with dashed curves for comparison. The right side of the revised contact space is too low, so that the lower finger of the arm hits the pawl before the cam pin disengages the upper finger. Figure 4b shows how we fix this problem with a small vertical dragger and Figure 4c shows the original and revised parts.

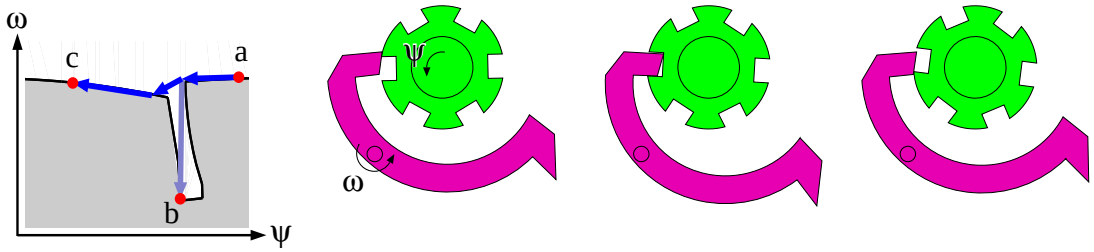
Second scenario

The pawl/ratchet configuration space (Figure 5a) reveals a second design flaw: the pawl can slide over the ratchet teeth instead of engaging in a slot. The configuration space coordinates are the ratchet orientation, ψ, and the pawl orientation, ω.



(a) (b) (c)

Fig. 4. Cam/arm redesign: (a) revision 1; (b) revision 2; (c) revised parts.



(a) (b) (c) (d)

Fig. 5. Pawl/ratchet engagement failure: (a) configuration space; (b) sliding; (c) engages; (d) jumps.

Contact space consists of a well where the pawl engages and of two plateaus where the pawl slides over the ratchet teeth. (This pattern is repeated six times for the six ratchet teeth.) When the arm releases the pawl, the spring force causes ω to decrease until the pawl hits the top of a ratchet tooth (Figure 5b). As the ratchet rotates, the configuration moves left until it reaches the mouth of the well. It can either drop into (Figure 5c) or jump over (Figure 5d) the well depending on the ratchet angular velocity and the spring force.

We modify the design to guarantee that the pawl engages. Our strategy is to raise the left plateau above the right plateau. If the configuration jumps over the well, it will hit the left wall and drop in. Our first revision is specified with a large upward dragger on the left plateau and a small downward dragger on the right plateau (Figure 6a). The revision achieves its goal, but the well becomes very narrow as a side effect. Kinematic tolerance analysis [11] shows that a 0.1% tolerance on the design parameters can produce parts for which the well closes, hence the parts cannot engage. Our second revision widens the well with two horizontal draggers (Figure 6b). The original and revised parts appear in Figure 6c.

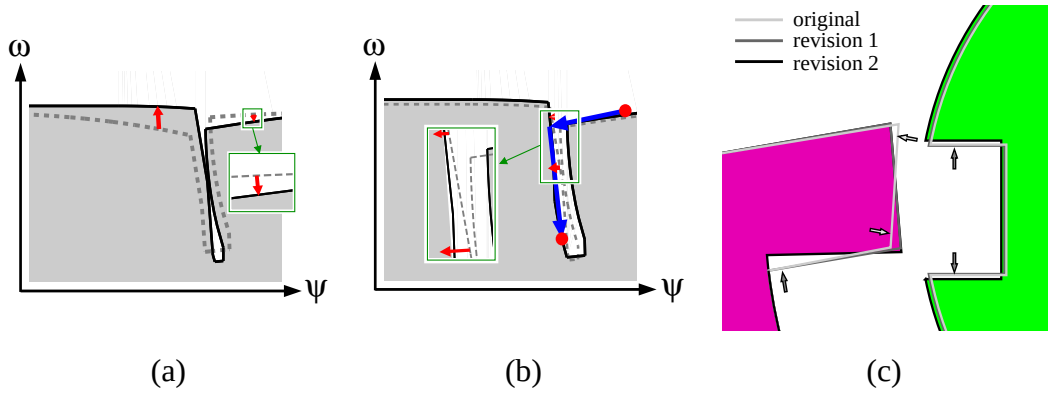


Fig. 6. Pawl/ratchet redesign: (a) revision 1; (b) revision 2; (c) revised parts.

Structure changes

The parameter update algorithm ignores the contact curves that are not selected for change, which are the vast majority of the configuration space. These curves 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, hence changing the configuration space structure (equivalently, the kinematic function) in an unintended way.

We have developed an efficient method of preventing structure changes. Before a parameter update is accepted, the new configuration space is matched against the old space. If the match succeeds, the new space has the same structure as the old and is accepted. Otherwise, a revised parameter update is computed. The line segment in parameter space between the old and the new parameter values is searched for the first point where the structure changes. This point is the closest we can get to the goal in the original parameter update optimization problem. The objective function is modified to prevent the change and the optimizer is restarted at a point just before the change.

We illustrate the algorithm on a cam/follower pair from an optical filter mechanism developed by Israel Aircraft Industries (Figure 7a). The parametric model has 25 functional parameters. The parts are attached to a fixed frame with pin joints. Rotating the cam counterclockwise causes its pin to engage the follower slot and drive the follower clockwise. The follower motion ends when the cam pin leaves the slot, at which point the follower filter covers the lens. As the cam continues to rotate, its locking arc aligns with the complementary follower arc and locks the follower in place. Rotating the cam clockwise returns the filter to the initial state.

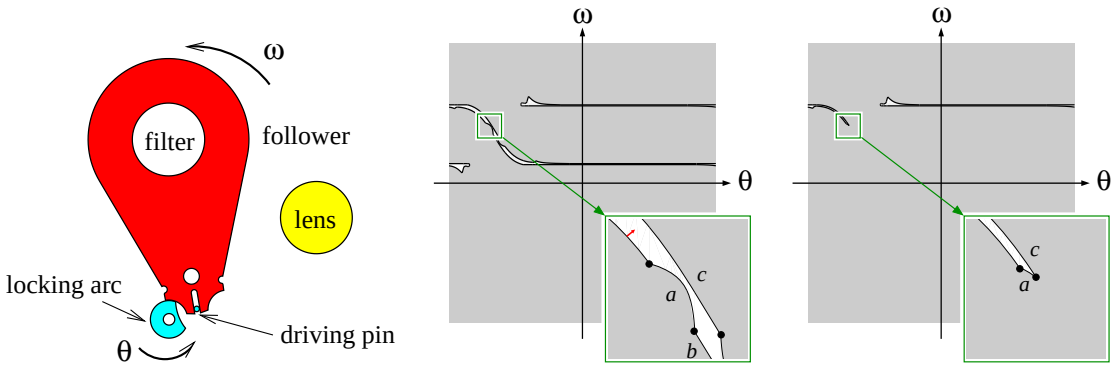


Fig. 7. Cam/follower: (a) parts; (b) initial configuration space; (c) structure change.

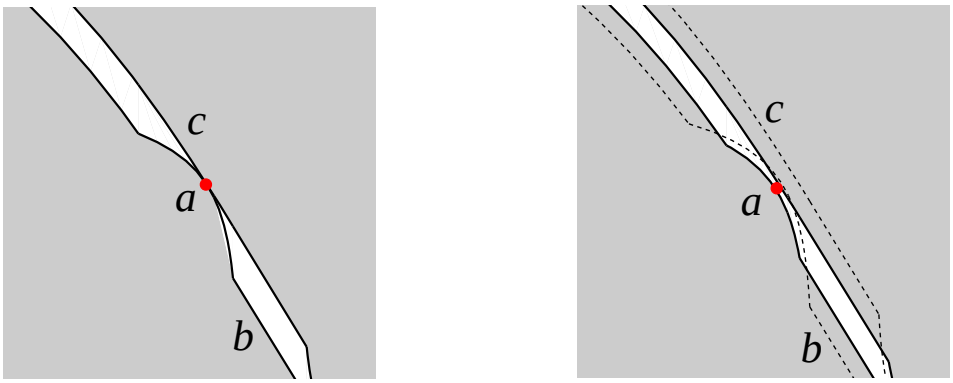


Fig. 8. Structure change prevention: (a) barriers inserted; (b) design goal achieved.

The initial configuration space shows correct function (Figure 7b). The cam drives the follower in the diagonal channel and locks it in the horizontal channels. But the configuration space also reveals excessive follower play in the driving mode. The play appears as the distance between the top and bottom contact curves that bound the channel. The designer tries to reduce the play with an upward dragger on the bottom curve (Figure 7b detail). The parameter updater achieves this goal by raising the entire bottom curve, which blocks the channel at its narrowest point (Figure 7c). The matcher reports that curve *a* intersects curve *c* in the new space, but not in the old space. The searcher finds a structure change where *a* becomes tangent to *c* (Figure 8a). Two penalty terms are added to the objective function: one prevents *a* from crossing a barrier point slightly below the tangency and the other prevents *c* from crossing a barrier slightly above. The barriers are marked with filled circles, which appear as a single circle because they are so close together. The revised objective is minimized and the design goal is achieved without structure changes

Input: parts, **u**, tol, sep.
1. Construct **u** configuration space.
2. Input draggers.
3. If draggers are satisfied within tol, return **u**.
4. Compute new parameter values **v**.
5. Construct **v** configuration space.
6. Match configuration spaces.
7. If they match, set $\mathbf{u} = \mathbf{v}$ and goto 3.
8. Find first structure change, **w**, using sep.
9. Construct barriers and goto 4.
Output: new parameter values.

Fig. 9. Parameter synthesis algorithm.

(Figure 8b).

Parameter synthesis algorithm

Figure 9 shows the parameter synthesis algorithm. The inputs are two parts, a vector **u** of initial parameter values, a tolerance for dragger satisfaction, and a minimum separation for structure change location. The parts are given in a parametric boundary representation. A part is bounded by an outer profile and by zero or more inner profiles, which are simple loops of line and circle segments. Line segments are represented by their endpoints and circle segments are represented by their endpoints and radii. The point coordinates and radii are represented with algebraic expressions whose variables are design parameters. Each part has a translation axis or a center of rotation, which is parameterized in the same way. We set tol to 10^{-5} and sep to 10^{-3} in the examples.

The algorithm consists of nine steps. Steps 1 constructs the initial configuration space. The configuration space is represented as a partition of free space into connected components bounded by loops of contact curves. The partition is encoded in the standard winged-edge representation of computational geometry [12]. Step 2 inputs the draggers via a point and click graphical interface. Each dragger consists of a target contact curve and a goal configuration that should lie on the target curve. Step 3 is the exit point: it returns parameter values whose configuration space satisfies the draggers to the input tolerance and has the same structure as the initial configuration space. Step 4 computes a parameter update via constrained nonlinear

optimization. Steps 5–7 match the new and old configuration spaces. If they mismatch, step 8 finds the first structure change. Step 9 constructs barriers and passes control to the next parameter update step.

The parameter synthesis algorithm handles mechanical systems with multiple higher pairs that share design parameters. Step 1 constructs an initial configuration space for each pair. Step 2 inputs draggers from all the pairs. Steps 3 and 4 are unchanged. Steps 5–7 apply to every pair. Step 8 finds the first structure change in every mismatched pair and returns the global first. Step 9 is unchanged.

The components of the parameter synthesis algorithm are configuration space construction, parameter updating, configuration space matching, structure change location, and barrier construction. The first component is described elsewhere [6]. The rest are described in the following sections.

Parameter updating

We formulate the draggers as a constrained nonlinear optimization problem and compute a parameter update by the conjugate gradient method with an active set strategy [13]. The constraints come from the dragger contact curves and from the design parameter ranges, while the objective comes from the dragger goals and from the barriers.

Every contact curve is the zero set of one equality and several inequalities in its configuration space coordinates. These equations also contain the design parameters. There is one set of equations for circle/line contacts and another for circle/circle contacts. The equality states that the two segments are tangent, while the inequalities state that the tangency point lies on the segments. These equations also cover vertices, which are modeled as circles of radius zero. Line/line contacts need not be analyzed because they are subsumed by their line/endpoint contacts.

The circle/line equations are for a circle segment with center $\mathbf{o}$, radius r, tail $\mathbf{p}$, and head $\mathbf{q}$ that contacts a line segment with tail $\mathbf{l}$, head $\mathbf{m}$, and length d (Figure 10a). The distance between the circle center and the line equals the radius: $(\mathbf{o}-\mathbf{l}) \times (\mathbf{m}-\mathbf{l}) = dr$. The contact point lies on the circle segment: $(\mathbf{p}-\mathbf{o}) \times (\mathbf{n}-\mathbf{o}) \geq 0$ and $(\mathbf{n}-\mathbf{o}) \times (\mathbf{q}-\mathbf{o}) \geq 0$, and on the line segment, $0 \leq (\mathbf{o}-\mathbf{l}) \cdot (\mathbf{m}-\mathbf{l}) \leq d^2$. The circle/circle equations are for a circle segment with center $\mathbf{o}$, radius r, tail $\mathbf{p}$, and

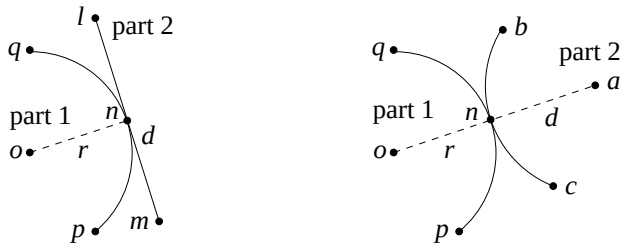


Fig. 10. Contact types: (a) Circle/line; (b) circle/circle.

head **q** that contacts a circle with center **a**, radius d, tail **b**, and head **c** (Figure 10b). The distance between the centers equals the sum of the radii: $(\mathbf{a}-\mathbf{o})^2 = (r+d)^2$. The line that connects the centers intersects both segments: $(\mathbf{p}-\mathbf{o}) \times (\mathbf{a}-\mathbf{o}) \geq 0$, $(\mathbf{a}-\mathbf{o}) \times (\mathbf{q}-\mathbf{o}) \geq 0$, $(\mathbf{b}-\mathbf{a}) \times (\mathbf{o}-\mathbf{a}) \geq 0$, and $(\mathbf{o}-\mathbf{a}) \times (\mathbf{c}-\mathbf{a}) \geq 0$.

Part 1 has configuration (x,y,θ) and part 2 has configuration (s,t,ψ). The points on part 1 have coordinates $\mathbf{o} = (x,y) + R_\theta o(\mathbf{u})$, $\mathbf{p} = (x,y) + R_\theta p(\mathbf{u})$, and $\mathbf{q} = (x,y) + R_\theta q(\mathbf{u})$ where $o(\mathbf{u}), p(\mathbf{u}), q(\mathbf{u})$ are local part coordinates, which are input as parametric algebraic expressions in the vector **u** of design parameters. Likewise, the points on part 2 have coordinates $\mathbf{l} = (s,t) + R_\psi l(\mathbf{u})$ and so on. If both parts rotate, the configuration space coordinates are (θ,ψ) and x,y,s,t are constant. After substituting the coordinate expressions into the contact equations, we obtain an equality $C(\theta,\psi,\mathbf{u}) = 0$ and a set of inequalities $H(\theta,\psi,\mathbf{u}) \geq 0$. For example, if $o(\mathbf{u}) = (u_1,0)$, $l(\mathbf{u}) = (0,u_2)$, and $m(\mathbf{u}) = (1,u_2+1)$ in a circle/line contact then C is $u_1 - u_2 = r$. The derivation is analogous for a translating part: the rotation angle is a constant and the translation is a configuration space coordinate.

The objective function for a dragger is $C(\theta_h,\psi_h,\mathbf{u})^2$ with C its contact equality and with (θ_h,ψ_h) its goal configuration. The constraints are the contact inequalities. The objective function for a set of draggers is the sum of the dragger objectives and the constraints are the union of their constraints. Two barrier terms may be added, as described below.

The updater computes a conjugate gradient step **v** for the objective subject to the constraints. If the objective decreases, **v** is passed to step 5 of the parameter synthesis algorithm (Figure 9). If not, the algorithm fails and the user must select new draggers. This rare case is omitted from Figure 9 for simplicity.

Configuration space matching

The **u** and **v** configuration spaces match when they have the same number of connected components and each **u** component matches a single **v** component. Two components match if they have the same number of boundary curves, every **u** curve matches a single **v** curve, and adjacent **u** curves match adjacent **v** curves. Two curves match if they are formed by the same pair of part features. A feature pair can generate several **u** curves and several **v** curves. The best match is chosen based on slope compatibility: the range of slopes is computed for each curve and the pair with the largest overlap matches.

The matcher compares every segment in every **u** component to every segment in every **v** component. If two segments match, it tests if the components match by comparing consecutive pairs of segments starting from the matched pair. If the components do not match, the first mismatch is passed to structure change location in step 8 of the parameter synthesis algorithm. The mismatch consists of consecutive segments (a,b) in the **u** component and (a',c) in the **v** component such that a matches a' and b does not match c. If all pairs of components match, **u** is updated to **v** and control passes to step 3 of the parameter synthesis algorithm.

The **u** and **v** configuration spaces can match even though structure changes occur on the line $[\mathbf{u},\mathbf{v}]$ (and indeed on every curve between **u** and **v**). A sequence of changes can occur that cancel each other: for example two disjoint contact curves become tangent, intersect for a while, become tangent again, and separate. We could guard against these situations by bounding the parameter update step size or by matching **u** against intermediate points in $[\mathbf{u},\mathbf{v}]$.

The matcher running time is worst-case quadratic in the number of contact curves. Faster algorithms are available, but are not worthwhile because the running time is negligible on real-world applications.

Structure change location

A mismatch between the **u** and **v** configuration spaces indicates that the **v** parameter update causes a structure change. We search the line segment $[\mathbf{u},\mathbf{v}]$ for the first structure change. A naive approach is to compute the configuration space at the

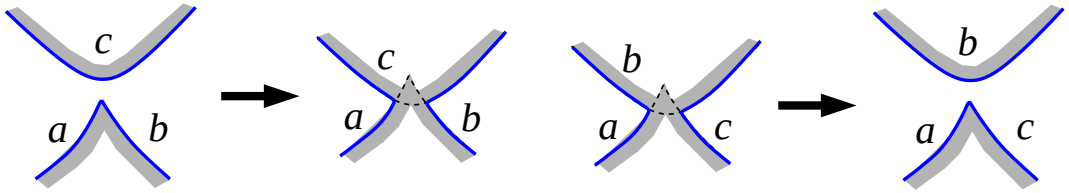


Fig. 11. Contact curve mismatches.

midpoint **m**, replace **u** or **v** with **m** depending on whether the **m** space matches the **v** or **u** space, and continue until the interval width shrinks below the minimum separation. This approach is unacceptably slow because of the configuration space computation in the inner loop, which is quadratic in the number of part features. We have developed a search algorithm that works with the mismatched features only, hence has a constant-time inner loop and is very fast.

A mismatch between (a,b) and (a,c) indicates that c moves between a and b or that b moves from between a and c (Figure 11). We will discuss the first case; the second is handled by swapping the roles of b and c and of **u** and **v**. There are three structure changes that can make c move between a and b: a and c are tangent (Figure 12a), an endpoint of c lies on a (Figure 12b), or an endpoint of a lies on c (Figure 12c). We perform a bisection search on $[\mathbf{u},\mathbf{v}]$ to find the structure change point **w**. If a and c are tangent, the mismatch first appears at **w**. If an endpoint of c lies on a, the mismatch first appears at **w** if c and the other segment, d, that shares the endpoint both lie on the same side of a. Otherwise, a mismatch between (a,b) and (a,d) occurs before **w** (dashed line d' in Figure 12b). Analogously, an endpoint of a that lies on c can be the first mismatch or can be preceded by a mismatch of (b,e) with c, as shown in Figure 12c. We perform a bisection search on $[\mathbf{u},\mathbf{w}]$ to find the structure change that causes the earlier mismatch and so on until we reach the first mismatch.

Figure 13 shows the structure change algorithm. The inputs are a mismatch (comprised of segments a, b, c), **u**, **v**, and a minimum separation. Steps 1–2 initialize the binary search. Step 3 bisects the search interval based on the `afterChange` predicate that returns true when the mismatch occurs after **m**. Bisection ends when the interval width drops below the separation. A reasonable alternate stopping condition is that the distance between c and a drops below a tolerance. When necessary, step 4 initiates another bisection search on $[\mathbf{u},\mathbf{m}]$ in cases b and c of Figure 12. Otherwise, steps 5–6 match the **u** configuration space with the space at a point **w** just before **m**. If a mismatch occurs, another bisection search is initiated on $[\mathbf{u},\mathbf{w}]$. This occurs when the **v** space contains structure changes that are independent of

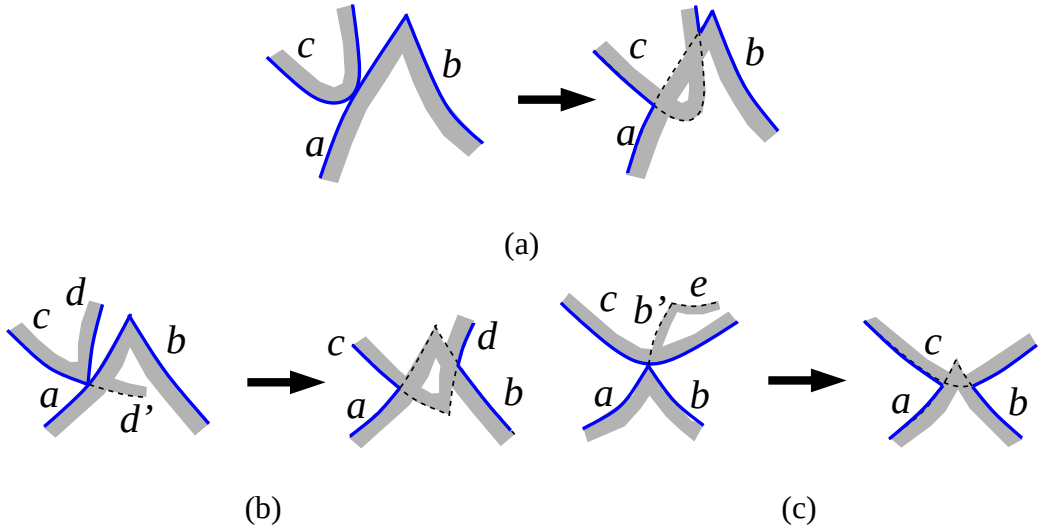


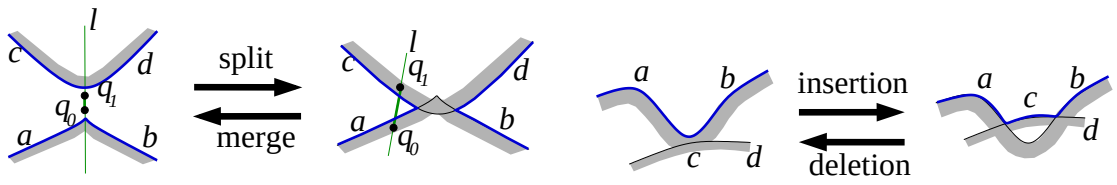
Fig. 12. Structure changes.

```
Input: mismatch, u, v, sep.
1. lb = u; ub = v;
2. m = 0.5(lb + ub);
3. if (ub - lb > sep) {
     if (afterChange(a, b, c, m)) ub = m;
     else lb = m;
     goto 2; }
4. switch (changeType(a, b, c, m)) {
     case B: c = c.next; v = m; goto 1;
     case C: a = b; b = b.next; v = m; goto 1; }
5. w = m - sep;
6. if (!match(u, w, mismatch)) { v = w; goto 1; }
7. return w;
```

Fig. 13. Structure change location.

and prior to the input mismatch. Otherwise, **w** is the first structure change, hence is returned.

The `afterChange` and `changeType` predicates are evaluated together in constant time, which makes the inner loop 2–4 very fast. The curves $a(\mathbf{m})$, $b(\mathbf{m})$, and $c(\mathbf{m})$ are constructed by instantiating the parametric features and calling the contact curve module of the configuration space construction program [6]. The *ab* meeting point can be a shared endpoint or an intersection point. It is computed by intersecting $a(\mathbf{m})$ and $b(\mathbf{m})$. The `afterChange` predicate is true when $ab(\mathbf{m})/ab(\mathbf{v})$ are on the same side of $c(\mathbf{m})/c(\mathbf{v})$. The `changeType` is computed analogously by simple



(a) (b)

Fig. 14. Structure change types: (a) topological; (b) geometric.

computational geometry.

The structure change algorithm is guaranteed to terminate. Executing steps 1–4 takes at most $k = \log_2(||\mathbf{v}-\mathbf{u}||/sep)$ iterations each of which takes constant time. Executing steps 5–7 takes $O(n^2)$ time where n is the number of contact curves. Steps 1–7 are executed $O(n^2)$ time because there are $O(n^2)$ structure changes. The total running time is $O((k+n^2)n^2)$. In practice, steps 1–7 are executed once or twice and the inner constant factor is very small.

The first structure change can occur at $\mathbf{u}$, which causes the parameter synthesis algorithm to fail. This situation occurs when the initial design is unstable. It has not been observed in practice and is omitted from Figure 13 for simplicity.

Barrier construction

Structure changes are prevented by adding penalty terms to the parameter update objective function that grow large as the curves approach the structure change point. The penalty term form depends on the structure change type. A structure change is called topological if it causes two free space components to split or merge (Figure 14a). This type of change involves two curves that bound free space components. A structure change is called geometric if it inserts or deletes a curve in a component (Figure 14b). This type of change involves a curve that moves between blocked and free space.

The penalty terms for topological changes are

$$B(u,\lambda) = \begin{cases} \beta\left(\frac{1}{\exp^{\alpha C_a(r_0(\lambda),u)}} + \frac{1}{\exp^{\alpha C_c(r_1(\lambda),u)}}\right) & \text{for a split} \\ \beta\left(\frac{1}{\exp^{-\alpha C_a(r_0(\lambda),u)}} + \frac{1}{\exp^{-\alpha C_b(r_1(\lambda),u)}}\right) & \text{for a merge} \end{cases}$$

where α and β are positive constants, C_a, C_b, C_c are the contact functions of curve segments a, b, c, and $r_0(\lambda)$ and $r_1(\lambda)$ are stake points that move along a line connecting initial points q_0, q_1 with a constant distance $||q_1 - q_0||$ (Figure 14a). For a split, barriers are inserted in free space, so that C_a and C_c are positive. If c passes over $r_0(\lambda)$ or d passes over $r_1(\lambda)$, C_a or C_c becomes negative and then $B(u,\lambda)$ increases rapidly. For a merge, barriers are inserted in blocked space. We allow geometric changes, hence do not construct barriers, because they do not change the qualitative kinematic function of the pair. It would be straightforward to construct barriers that are analogous to the topological ones.

We set $\alpha = 10^{-5}, \beta = 50$ in the examples. The larger α is, the faster the penalty term increases as the contact curve approaches the barrier. A good α is large enough to prevent the curve from crossing the barrier, but small enough to keep the penalty term well conditioned. The β value determines the relative importance of the barrier and the draggers in the next conjugate gradient step.

The barrier terms are deleted from the objective function after one conjugate gradient step. The rationale is that the the barrier terms are superfluous, since the next step will probably not encounter the same structure change, hence should be deleted because they slow convergence to the dragger objective. If the structure change does recur, it will be redetected and the barriers will be reconstructed.

Conclusions

We have presented a parameter synthesis algorithm for planar mechanical systems comprised of higher pairs where each part rotates around a fixed point or translates along a fixed axis. The system is given in a parametric boundary representation. The designer assigns initial parameter values, analyzes the resulting kinematic function in configuration space, and requests kinematic changes via draggers. The program computes a parameter update that fulfills the requests and preserves the qualitative kinematic function, meaning the configuration space topology. We believe this is an effective paradigm for parameter synthesis based on several case studies involving real-world designs.

We see several directions for further work. The top priority is to validate the synthesis algorithm on additional applications. Multiple draggers can interact in unexpected ways when their contact curves share parameters. We are working on a

technique for visualizing these dependencies in configuration space based on prior work in kinematic tolerance analysis [11].

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 [14] and can compute parameter updates as before, but lack a structure 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 [15] and can reuse the parameter update and structure change algorithms. All that is lacking is a computer implementation of the parametric contact equations for the various spatial contacts.

Acknowledgments

This research was supported by NSF grants CCR-9617600 and IIS-0082339, 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.

References

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

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

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

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

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

[6] 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.

[7] Elisha Sacks and James Allen. Mems functional validation using the configuration space approach to simulation and analysis. In *Second International Conference on Modeling and Simulation of Microsystems, Semiconductors, Sensors, and Actuators*, 1999.

[8] Elisha Sacks and Steven M. Barnes. Computer-aided kinematic design of a torsional ratcheting actuator. In *Proceedings of the Fourth International Conference on Modeling and Simulation of Microsystems*, Hilton Head, SC, 2001.

[9] Elisha Sacks, Leo Joskowicz, Ralf Schultheiss, and Uwe Hinze. Computer-assisted kinematic tolerance analysis of a gear selector mechanism with the configuration space method. In *25th ASME Design Automation Conference*, Las Vegas, 1999.

[10] 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.

[11] Elisha Sacks and Leo Joskowicz. Parametric kinematic tolerance analysis of planar mechanisms. *Computer-Aided Design*, 29(5):333–342, 1997.

[12] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, editors. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 1997.

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

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

[15] Ku-Jim Kim, Elisha Sacks, and Leo Joskowicz. Kinematic analysis of spatial fixed-axis higher pairs using configuration spaces. Technical Report CSD-TR 02-001, Purdue University, 2002. To appear in *Computer-Aided Design*.