Computational Synthesis of Mechanical Systems

Sridhar Kota
Professor of Mechanical Engineering
University of Michigan, Ann Arbor, MI 48105-2125
Email: kota@umich.edu

Abstract
The paper provides a brief outline of methods for automated synthesis of mechanical systems in general, with primary focus on application domains involving kinematics of mechanical motions. Two application domains described in this paper, mechanisms and machine tool design, employ methods of selecting and assembling appropriate building blocks to satisfy given functional (motion) specifications. This is called Synthesis by Composition. The functionality of the precompiled building blocks as well as the design task specifications are captured mathematically in a unified representation scheme. Generation of alternate designs is accomplished by manipulation of matrices representing functions. The paper also presents another method of Synthesis by Elimination to automatically generate topologies of flexible structures which satisfy given input/output forces and displacements. The methodology presented in this paper has been successfully applied to computational synthesis of mechanisms, machine tools, compliant mechanisms, and differentials. Various applications of these methods to automotive, aerospace, and medical and MEMS fields will be presented at the symposium.

1.0 Introduction

A systematic procedure for automated synthesis involves (a) proper understanding of task requirements and (b) creation of a concept design; which is referred as configuration design, or type synthesis (mechanisms), or topology synthesis (structures, machine tools) etc. depending on the application domain. This early phase of design is still a mixture of art and science. In order to develop a more scientific basis for conceptual design process, we need a unified mathematical framework that captures both the design intent (task requirements) as well as the functionality of the basic building blocks of the application domain. Such a mathematical framework, being limited to a specific application domain, can systematically, and therefore computationally, generate alternate solutions and rank them according to various performance requirements and constraints. The same framework can also include information relevant to further analysis and evaluation.

Whenever we attempt to develop a systematic procedure, let alone an algorithmic one, to "automate" an open-ended creative design process, we need to focus on certain aspects of the problem first and set aside the others until later stages [Kota and Lee 1993]. Assuming we start with a "complete" set of design requirements we can sort out the desired primary functions, and performance requirements or operational constraints. A function describes the intended behavior of the artifact to be designed. For instance, in a mechanism, the primary function is purely kinematic; that is say when the input motor is turned on, the output should follow a specified path. Inertia forces due to component masses and the speed of operation affect the dynamic behavior, motor size, vibrations, etc. Focusing on the essentials, we first develop alternate functional (kinematic) solutions to convert the motion of a rotating shaft into a desired trajectory. We can then evaluate each of the alternatives and rank them in terms of their superiority in dynamic performance measures.

The basic premise of this work, like many others reported in the literature, is that we begin the synthesis process by reasoning with functions before addressing the performance requirements and constraints keeping in mind that a complete solution must satisfy all the constraints as well as the functions before it is deemed useful. Alternate designs are generated first by considering functions alone. This functional synthesis process yields alternate configurations, or topologies, or arrangements which satisfy the design requirements in a qualitative sense. Systematic generation of these conceptual alternatives can be performed in at least two ways: (a) Synthesis by Composition; in which each of the higher level functions are mapped to one of more pre-compiled physical building blocks that are stored in the library. We must then keep track of the order and spatial orientations of the candidate building blocks to create a meaningful assembly. This generic method of synthesizing building blocks is similar to creative reasoning of a human designer. (b) Synthesis by Elimination, on the other hand, is something that is very peculiar to computers. Here we start with much lower level building blocks of the same kind (Lego blocks, or beams, or tubes etc.) with a plurality of standardized connections among them. If the design intent is captured correctly, an optimization algorithm automatically eliminates the building blocks that are not needed leaving a feasible topology. Building blocks used in this type of synthesis procedure have finer granularity than the ones used in synthesis by composition. In order for this approach to work without combinatorial explosion, all the building

blocks should share the same functional character (all beams – resist bending; or all tubes – resist torsion; or Lego blocks of certain geometric shapes).

In the following, we will very briefly describe the two approaches to synthesis. Many of the important details and illustrations and the related work by other researchers are omitted due to space constraints. Interested readers should refer to articles cited for detailed and complete descriptions.

2.0 Computational Synthesis of Mechanisms

Conceptual design of mechanisms is still a mixture of art of science. Many researchers in the past have attempted to classify basic elements of mechanisms in order to systematize the creative design process [Reuleaux 1876, Molian 1969, Chironis 1978, Artobolevski 1986]. A detailed account of various approaches to creative design of mechanisms can be found in [Chiou and Kota and Chiou 1992]. Graph enumeration procedures reported in the literature [Freudenstein and Maki 1979, Tsai 2000] are based primarily on structural and topological considerations and do not reflect the desired behavior; where desired behavior is characterized as a set of kinematic functions and operational constraints that capture the intended motion requirements.

Starting with the intended function and operational constraints, the methodology presented by [Chiou and Kota 1999] systematically generates alternate mechanism concepts. The methodology uses symbolic matrices and constraint vectors to represent the functionality of mechanism building blocks. When no direct match exists between the motion requirements and stored building blocks, the matrix representation scheme automatically decomposes the given task into sub tasks to facilitate search of sub-solutions and performs subsequent synthesis. Rather than symbolic matrices, motion transformation matrices based on dual vector algebra [Ball 1900, Bottema and Roth 1979, Murray et al 1993] was later presented by [Moon and Kota 2002].

The dual-vector representation of a rigid-body motion (screw representation) enables us to separate the function from the structure. The design synthesis process comprises of three stages: 1. Functional synthesis. (2) Topological synthesis and (3) Dimensional synthesis. Motion codes capture the kinematic motion transformation required to accomplish the task as well as kinematic functionality of stored building blocks. The necessary coordinate transformation and alignments required to assemble appropriate building blocks are performed using dual vector algebra during topological synthesis. Decomposition of a desired task into subtasks, in order to meet either kinematic function or spatial constraints, is performed by manipulating the dual-vector representations.

2.1 Motion Specifications

The dual representation of a screw can also be directly applied to represent desired motion specifications. Suppose we intend to design (or select or assemble) a mechanism that transforms a rotational input motion into a translational output motion (figure 1), then the motion requirements can be represented as:

$$\hat{\mathbf{S}}_{\mathrm{I}} = (q_{\mathrm{I}} + e0)[\mathbf{L}_I + e\mathbf{R}_I \times \mathbf{L}_I]$$
$$\hat{\mathbf{S}}_{\mathrm{O}} = (0 + eF(q_{\mathrm{I}}))[\mathbf{L}_O + e\mathbf{R}_O \times \mathbf{L}_I]$$

Where the pitch $q_{\mathrm{I}} + e0$ represents the rotational motion, pitch $0 + eF(q_{\mathrm{I}})$ represents the translational motion. The dual vectors represent the directions ($\mathbf{L}_{\mathrm{I}}$ and $\mathbf{L}_{\mathrm{O}}$) and the positions (R_{I} and R_{O}). Therefore the desired kinematic function of the mechanism is to transform $q_{\mathrm{I}} + e0$ to $0 + eF(q_{\mathrm{I}})$. Further, the spatial orientation of the input and the output are defined as $\hat{\mathbf{S}} = \sin^{-1}(|\hat{\mathbf{S}}_{\mathrm{I}} \times \hat{\mathbf{S}}_{\mathrm{O}}|)\ \hat{\mathbf{S}}_{\mathrm{I}} \times \hat{\mathbf{S}}_{\mathrm{O}}$

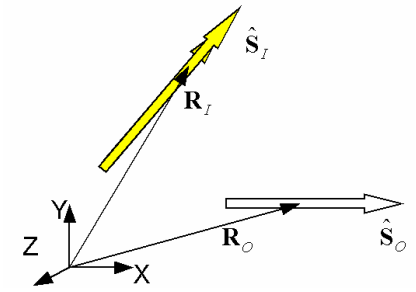


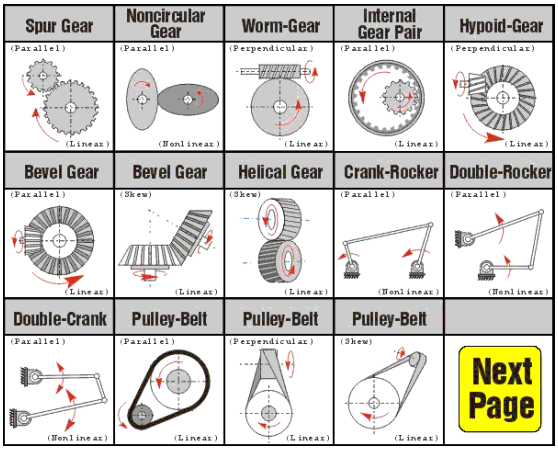
Figure 1: Capturing desired motion requirements as a screw transform

Now that we have a mathematical means to capture the design specifications as well as the functions (motion characteristics) of basic building blocks, we can perform synthesis of mechanisms by selecting and synthesizing appropriate building blocks.

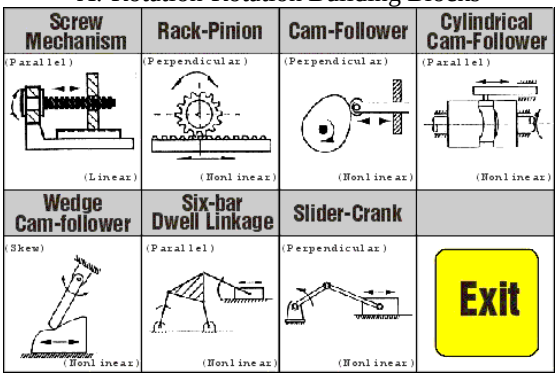
2.2 Kinematic Building Blocks

There are many ways to categorize a set of building blocks. Our approach is based on the type of input and output motions. Since a typical input motion is either a purely rotational motion (motor or a crank) or a purely translational motion (pneumatic or hydraulic cylinder), we identified building blocks as the ones that transform a Rotational motion(R) into a Translational motion (T) or between R-R, T-T, or T-R. Figure 2 below shows a partial list of such building blocks. Although several building blocks satisfy the basic kinematic function of say converting R-T, each one has a unique motion characteristics such as if the output is a reciprocating motion or if it requires input to change direction to alter

the output direction. These critical differences are captured in what is called a Motion Characteristics Code (MCC).



A: Rotation-Rotation Building Blocks



B: Rotation – Translation Building Blocks

Figure 2: Library of kinematic building blocks

2.3 Representation of Building Blocks

A mechanism building block transforms a rigid body motion between input and output terminals and it is represented as two screws *(i) input motion screw, and (ii) output motion screw.* The magnitude of the output motion M_O is a function of input motion M_I, therefore $\hat{M}_O = f(\hat{M}_I)$. In the case of a rack-and-pinion building block, the motion of the rack is a function of rotation angle of the pinion θ_I (figure 3) and therefore $\hat{M}_O = (0 + eGq_I)$.

The kinematic function and the relative orientation of the input and output axes of a rack-and-pinion building block are represented by a pair of screws (input screw and output screw). The input motion screw is

$$\hat{\mathbf{S}}_I = (q_I + e0)\left[\begin{Bmatrix}0\\0\\1\end{Bmatrix} + e\begin{Bmatrix}0\\0\\0\end{Bmatrix}\right]$$; And the output motion screw

$$\hat{\mathbf{S}}_O = (0 + eGq_I)\left[\begin{Bmatrix}0\\1\\0\end{Bmatrix} + e\begin{Bmatrix}0\\0\\D\end{Bmatrix}\right]$$

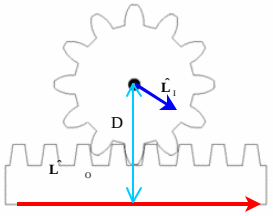


Figure 3: Rack-and-Pinion Building Block

Additionally, a *motion characteristics code* is used to capture information pertaining to motion type, continuity, and linearity of building blocks. The motion characteristics code (MCC) has four entries (ABCD) each has a value of 0 or 1: A. Motion type indicates whether the motion is rotation (0) or translation (1). B. Continuity indicates whether the motion is a continuous (0) or intermittent (1). C. Linearity indicates whether the motion is linearly (0) or non-linearly (1) dependent on time, and 4. Directionality indicates whether the motion is unidirectional (0) or bi-directional (reciprocating or oscillating; value 1). For example, a simple rotating motion of a motor is coded as 0000 which implies rotational, continuous, linear, unidirectional motion.

The motion characteristics of each building block (applies also for an entire mechanism) are represented by a string of two MCCs: the first MCC characterizes the output and the second one characterizes the input. For example, a rack and pinion is represented as 0000-1001. This implies that the output is rotational, continuous, linear and unidirectional and the input is translational, continuous, linear, and non-reciprocating. The fact that the input and output can be interchanged is stored in another field (not shown here). Thus, a mechanism building block is represented by a pair of MCCs and it can also be considered as a *one-edge and a two-node graph*. The library of mechanism building blocks are parameterized by their design variables and function characteristics.

2.4 Functional Synthesis

The desired motion is represented by a pair of MCCs, "XXXX-YYYY". We first search the building block library to select building block(s) with same pair of MCCs as the desired MCC pair. If the initial search fails to identify a matching MCC pair, then the required MCC-pair is decomposed. The process of decomposition of required function into sub-functions involves matching the required output MCC, XXXX in figure 4, with building block MCC pair until at least one building block whose output MCC matches with XXXX. If this search is successful, we will have a XXXX-AAAA building block. Subsequent search

attempts to identify at least one building block whose MCC pair is AAAA-YYYY. The search and decomposition process are iterated until an ordered set of building blocks are identified to match XXXX-YYYY.

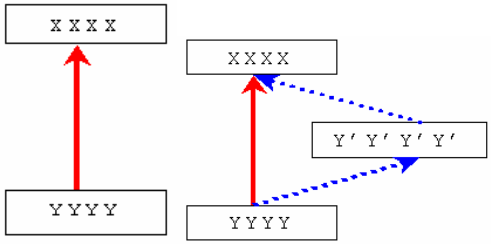


Figure 4: Decomposition of MCC in search for candidates

If search process fails to identify a desired MCC, then the search algorithm attempts to identify matches for individual entries. For instance, if the search process fails to identify a building block say whose output is 0101, then it first attempts to find a building block with 0100. If this goal is met then the algorithm looks for a building block whose output is 0001 so that these two building blocks can be combined to produce the desired output 0101. When two building blocks are combined, the output of the first building block serves as the input to the second building block. A truth table (not shown here) serves as a guide to derive the resulting motion characteristics when any two building blocks are concatenated.

After selecting appropriate building blocks, the spatial configuration of the assembly should be determined. Since all the building blocks are modeled by dual-vectors, a series of line transformations [Moon and Kota 2002] are performed to align the output axis of each building block with the input axis of its succeeding building block.

2.5 Topological Synthesis

The directions of the motions are aligned by mathematical manipulation of dual-*vectors* of the building blocks. This procedure first aligns the axis of output motion of the last building block to the desired output motion. This process involves (a) transformation of each building block coordinate system in to machine (global) coordinate system and (b) line transformation of each building block to align the axes. Note that when all the said transformations are performed on the selected set of building blocks, the input axis of the first building blocks may not match the input axis of the desired motion. This spatial discrepancy can be met by searching the library for additional building blocks guided by spatial decomposition.

2.6 Design Example

The objective is to synthesize a mechanism that draws wire from a spool and cuts the wire to desired length. The two subtasks (corresponding to each output motion) are (i) wire feeding and (ii) wire cutting. It is also required that the mechanism be driven by a single motor and the two outputs are coordinated according to a prescribed timing diagram (not shown). A pair of friction rollers is assumed to serve as end-effectors to feed the wire intermittently to the cutter. Figure 5 shows a layout of the kinematic and topological design specifications.

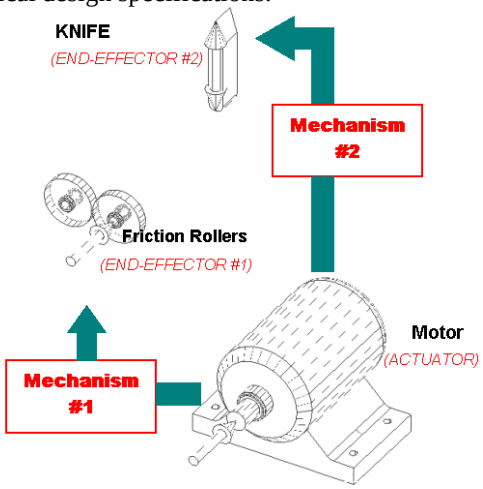
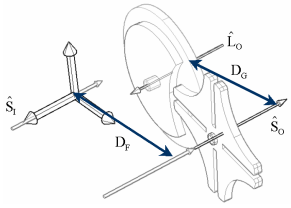


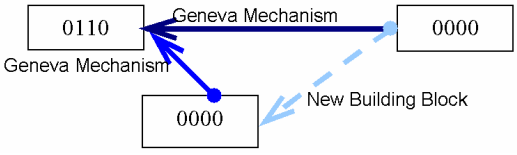
Figure 5: Schematic of synthesis task specifications.

The position and orientations of the input and output motions are captured in a Global reference frame. The wire feeding mechanism has a continuous rotational input and an oscillating output (0110-0000). A Geneva mechanism (in the library) is considered as one of the candidate building blocks since it has the same MCC pair. In order to match the directions and orientations with the axis of the input motor and the friction rollers, a spur gear pair and a bevel gear pair are selected. This process is entirely algorithmic as it involves, matching MCC pairs, deriving new subtasks by decomposition, identifying spatial relations (matches and mismatches) using dual vectors and coordinate transformations. The evolution of one of the solution mechanisms is depicted in figure 6.

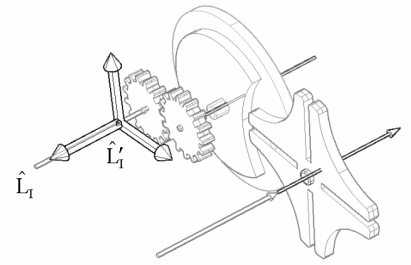
A similar procedure yields solutions to wire cutting mechanism. In the final design shown below, a cam-follower system is used to convert the continuous rotational motion of the motor to reciprocating translational motion of the wire cutter.



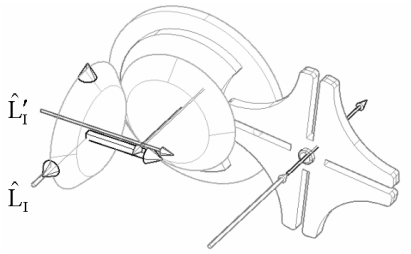
A.



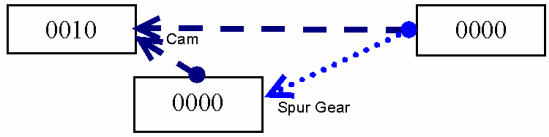
B.



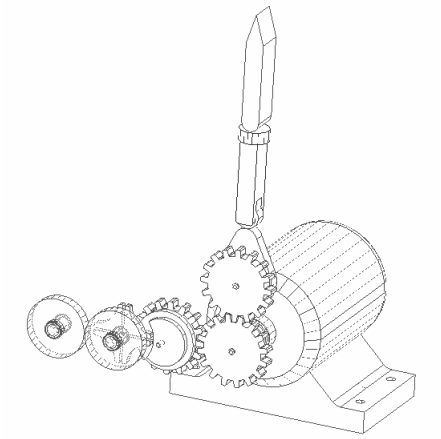
C.



D.



E.



F.

Figure 6: A-E is a snap shot at the evolution of a concept solution during synthesis of the wire-cutting mechanism. F is one of many possible solutions of an automated wire-cutting and wire-feeding mechanism.

3.0 Computational Synthesis of Reconfigurable Machine Tools

Using screw representation of motions, [Moon and Kota 2002b] developed an automated synthesis procedure for designing reconfigurable machine tools. Starting from a set of machining operations (figure 7), milling, drilling etc, to be performed on a set of parts, a set of feasible structural configurations of the machine is determined using graph theory. The required motions of the cutting tool to generate prescribed machining features are captured in the form of dual-vectors. The motion characteristics of various building blocks (figure 8) are also represented as dual-vectors. Using a precompiled parameterized library of commercially available machine modules, each (kinematic) function is then mapped to a feasible set of modules. Process models are used to establish desired dynamic stiffness criteria. This provides a set of kinematically feasible machine tools that provide desired motions. The structural stiffness of each of these machine tools is then evaluated. This results in a dynamically-feasible sub-set of machine tools.

The design procedure consists of three main stages: 1. Conceptual design, 2. Module selection, and 3. Design evaluation. The conceptual design stage consists of three steps: 1. Task clarification; in which the cutter-location data is transformed into a series of homogeneous transformation matrices representing desired motion, 2. Structural configuration; in which graph theory is used to enumerate alternate machine tool architectures (Figure 9), and 3. Function mapping: where the kinematic motion functions are mapped to various elements of the machine tool structure (nodes in the structural graph). The key enabler in developing a systematic procedure is the motion representation scheme.

3.1 Design Representation

The three dimensional motion of any rigid body (cutting tool, machine components etc.) is represented (screw) as

$$\vec{\$} = [M_M \quad M_m \quad M_C]\{P_A + eP_T\}\{\vec{S} + e\vec{S}_0\}$$

The first part of the dual vector representation shows the range of the motion. M_M, M_m and M_C represent maximum, minimum, and current value of the motion. The second term $(P_A + eP_T)$ represents the pitch of the motion. P_A and P_T represent angular and translational pitches respectively. If P_A is 0 the screw represents the pure translation and P_T is 0 then the screw represents the pure rotational motion. P_A and P_T assume values between 0 and 1. P_T is the ratio of translational motion to rotational motion if P_A is non-zero. Finally, $\vec{S}$ is the direction of the motion and $\vec{S}_0$ contains the information of the position of the motion $\vec{r}$.

3.2 Module Selection

The kinematic model of the desired machining task (task matrix T) comprises of a set of motions required to carry out the prescribed machining operations. These motions are represented as a series of screws.

$$T = T_{m\_1} T_{m\_2} \cdots T_{m\_n} T_0$$

Each $T_{m\_}$ represents one motion that can be fulfilled by one or more mechanisms. A complete model of the task specifications include the spindle motion, the type of machining process, power requirements and ranges of

various motions. Since the task matrix transforms the coordinates from work piece to tool, the product of the matrices of modules should be equal to the task matrix. $M_1 M_2 M_3 \cdots M_n = T$; where M_i is ith module matrix. i ∈ {1, 2, 3, ..., n}and T is task matrix. M_1 is the module adjacent to the work piece, and has the function of rotation about Z-axis, as determined during function assignment procedure. The modeled task is decomposed in to several sub-functions during function decomposition procedure and the decomposed functions are assigned to the modules that have been determined during structure design stage.

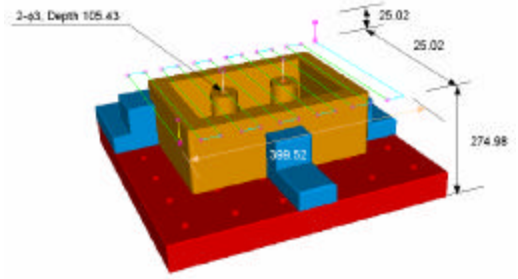


Figure 7: Machining task specifications

Name	Drawing	Name	Drawing	Name	Drawing	Name	Drawing
Angle Structure 1 (1)		Angle Structure 2 (2)		Structure 11 (3)		Structure 12 (4)	
Structure 13 (5)		Structure 14 (6)		Structure 21 (7)		Structure 22 (8)	
Base 1 (9)		Base 2 (10)		Base 3 (11)		Base 4 (12)	
Column Base 1 (13)		Column Base 2 (14)		Motion 11 (15)		Motion 12 (16)	
Motion 21 (17)		Motion 22 (18)		Motion 31 (19)		Motion 32 (20)	
Slide 1 (21)		Slide 2 (22)		Slide 3 (23)		Slide 4 (24)	
Spindle 11 (25)		Spindle 12 (26)		Spindle 21 (27)		Spindle 22 (28)	
Spindle 311 (29)		Spindle 312 (30)		Spindle 321 (31)		Spindle 322 (32)	

Figure 8: A parameterized library of building blocks

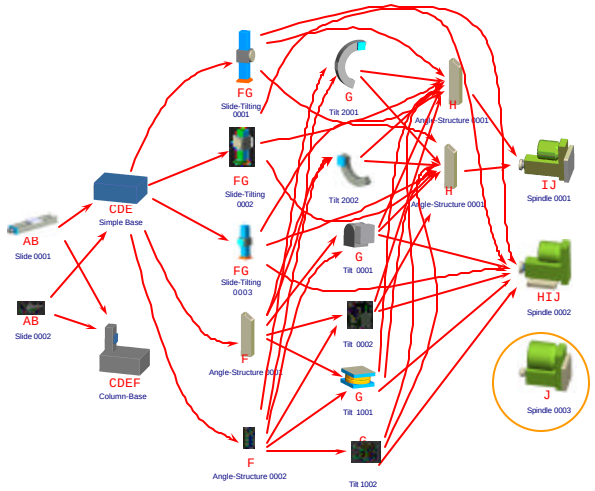
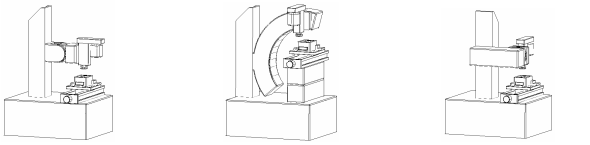


Figure 9: Enumeration of alternate solution paths.



(a)Configuration1 (b) Configuration 2 (c) Configuration 3

Figure 10: Alternate machine configurations generated.

4.0 Computational Synthesis of Compliant Structures.

In this section, we present a paradigm in which topological solutions are generated by eliminating unwanted structural elements. The building blocks in this case are all of the same type (beams). We start out with a network of hundreds of such highly interconnected beams. If the design intent is captured properly as a mathematical objective function, the optimization procedure will determine which of the beam elements in the initial network are really needed to meet the desired functional specifications. The design variable, beam cross section, approaches zero value for the beams that are not needed. This method of "synthesis" is not carried out by composing necessary building blocks in the strict sense of the term synthesis. But it does generate (or synthesize) topological arrangements to meet given functional specifications. We will illustrate this method in the context of designing compliant mechanisms.

Traditionally, engineered artifacts are designed to be strong and stiff. Designs in nature, on the other hand, are strong but not necessarily stiff – they are *compliant*. Compliant mechanisms are single-piece flexible structures that deliver the desired motion by undergoing elastic deformation as opposed to jointed rigid body motions of conventional mechanisms [Kota et al 2001]. Compliance in design leads to joint less, no-assembly, monolithic mechanical devices and is particularly suited for applications with small range of motions

The first step in the design of a compliant mechanism is to establish a kinematically functional design that generates the desired output motion when subjected to prescribed input forces. This is called topological synthesis. Once a feasible topology is established, performance constraints (permissible stress, energy efficiency etc.) can be imposed during the following stage in which size and shape optimization are performed. The key design challenge in this paradigm is to synthesize an optimum configuration of structural elements (beams) which is flexible enough not to minimize the input energy needed to generate desired motion (deformation) and yet stiff enough to withstand resistive external loads.

4.1 Topology Synthesis

An objective function is formulated to capture conflicting design requirements; that is the need for (a) compliance to undergo desired deformation (kinematic requirement), and (b) stiffness to resist external loads (structural requirement) once the mechanism assumes the desired configuration.

In the first part, the "mechanism design", the kinematic requirements are met by maximizing the deflection at a specified point along a specified direction. This is achieved by applying a fictitious force at the point of interest, B, along the direction of the desired output deflection, Δ (Figure 11). This "dummy load" is denoted by f_B and as shown below for a general design domain subject to an applied force, f_A, at the point A and some specified boundary conditions. Maximizing the deflection at the point B in the direction of f_B is equivalent to maximizing the mutual potential energy, $v_B{}^T K u_A$, where u_A is the deflection field due to f_A, v_B is the deflection field due to f_B, and K is the global stiffness matrix.

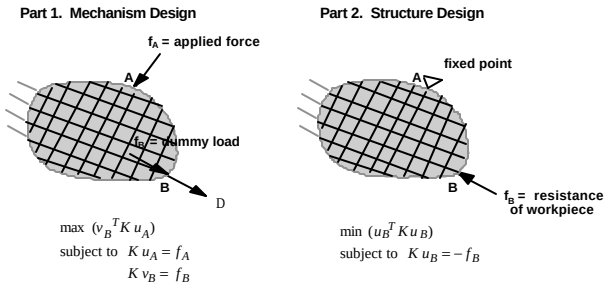


Figure 11: Illustration of synthesis problem formulation.

In the second part, the structural stiffness is maximized. Here the point A is considered fixed, and the external load (resistance) is accounted for by applying the force f_B at point B in the opposite direction. Maximizing the stiffness is equivalent to minimizing the strain energy, $u_B{}^T K u_B$, where u_B is the deflection field due to this set of loading, and K is the global stiffness matrix. The two parts are then combined into a single problem via multi-criteria optimization [Kota et al 2001].

Combined Problem

$$\max \left[\frac{\text{mutual energy}}{\text{strain energy}} \right] = \max \left[\frac{v_B{}^T K u_A}{u_B{}^T K u_B} \right]$$

subject to $K u_A = f_A$

$K v_B = f_B$

$K u_B = -f_B$

total resource constraint

lower and upper bounds

4.2 An Array of Beam Elements

In this method, the prescribed design domain (this is the area within which the mechanism should fit) is first divided into a number of nodes. Each node is connected to several other nodes via modular array of beam elements. This serves as an initial guess. Certain nodes are "fixed" to imply the points where the mechanism is anchored to the substrate. The cross sectional area of each beam element serves as the design variable with specified upper and lower bounds. The resource constraint provides less material than the available space. The objective then is to distribute the material in a way that maximizes the objective function. During the optimization process, those beam elements whose cross sectional areas reach the lower bound are removed (deemed unnecessary) leaving only a network of beam elements whose area reached the upper bound. This establishes the topology of the compliant mechanism.

4.3 Design Example

The compliant gripper synthesis example (figure 12) illustrates the automated synthesis method. The design specifications are that the applied force, F, causes the motion, D, at the indicated location, which will allow the device to grip an object at that point as shown in Figure 12(a). The design domain shown in Figure 12(a) represents the upper-half view since this is assumed to be a symmetric problem without any loss of generality in the solution procedure. The dashed line represents the desired space within which the mechanism should fit.

The initial guess is a modular beam structure (Figure 12(b)) with a uniform distribution of cross-sectional areas. When the algorithm converges, the solution consists of beam members whose design variable reached (or is close to) the upper bound. The beam members whose design variables reached the lower bound constraint are eliminated. The optimized solution and corresponding finite element model are shown in Figure 12(c)), where the un-deformed shape is denoted by the dashed lines and the deformed shape is denoted by the solid lines. Compliant grippers based on this design were fabricated in nylon using a rapid prototyping machine. The methodology described here applies for three-dimensional problems as well as design problems with multiple sets of input/output force/displacements.

5. 0 Conclusions

The methodologies presented in this paper have been successfully implemented in synthesis programs. In mechanisms and machine tools domains, the kinematic functionality can be captured mathematically in the form of dual vectors and transformation matrices to enable algorithmic procedures to generate conceptual designs. Extending the methodology to multiple domains (such as hydraulics, and control systems) is difficult due to the fact that the functions in those domains demand a different representation scheme. If a lower level function that is common to multiple domains is discovered, the granularity or semantic size of the building blocks would become

prohibitively small in that combinatorial explosion would soon occur. If the granularity is too coarse, not only is the novelty in synthesis is hindered but also, in the case of more demanding design task specifications, none of the building block combinations will yield a feasible solution. However, a hierarchical functional scheme in which the functional building blocks are kept coarsest in any given instance, in that the process of function decomposition is halted soon as a mapping to a physical device is discovered. By discretizing the design space and the functional character of the building blocks to a smaller chunks yield intricate (several components) solutions. Any of such solutions can be further refined to enable function sharing and consequent reduction in total number of parts. Even in the early stages, mapping functions to physical building blocks require quantitative comparisons to yield physically meaningful solutions.

If an initial seed design is available, and if it can be abstracted in the form of a graph (nodes and edges), graph enumeration schemes work well in generating all possible combinations of building blocks which yield feasible functional solutions. In the absence of such a seed design, a mathematical framework is required to capture the design intent (ignoring the performance measures) and the functionality of building blocks in a unified representation scheme. Otherwise, a rule-based or a case-based reasoning scheme can only offer a limited, non-scalable approach computational synthesis. Finally, regardless of the ultimate success of a particular computational scheme employed, the effort alone enhances the synthetic reasoning of a human designer.

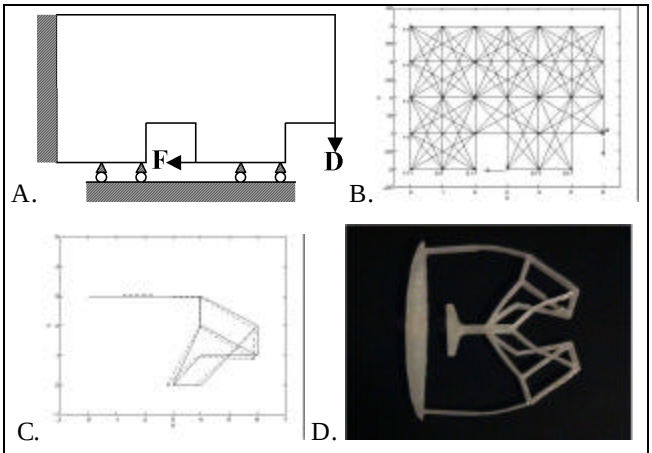


Figure 12: Compliant design methodology. (a) Define the design problem and desired forces. (b) Create an initial guess of what the structure may look like. (c) Obtain the solution through structure optimization (deformed shape shown with dashed lines). (d) Fabricate design.

6.0 Acknowledgements

The author gratefully acknowledges the work of several of his graduate students who worked diligently on a variety of computational synthesis schemes over the years. This work was supported through multiple grants and contracts from National Science Foundation (NSF) – Design Engineering Program, and Air Force Office of Scientific Research (AFOSR).

7.0 References

1. Artobolevsky II, 1986, Mechanisms in Modern Engineering Design, MIR Publishers, Moscow, Volumes1-2. Ball, S., 1900, *"A Treatise on the Theory of Screws"*, Cambridge University Press (Reprinted in 1998).

4. Bottema, O., and Roth, B., 1979, *Theoretical Kinematics*, North-Holland Pub. Co., New York

5. Chironis, N., 1978, Mechanism, Linkages and Mechanical Controls, McGraw Hill, New York

6. Chiou, S. J. and Kota, S., 1999, Automated Conceptual Design of Mechanisms, Mechanism and Machine Theory, Vol. 34, No. 3, 467-495

7. Freudenstein, F., and Maki, E.R., 1979, The creation of mechanisms according to kinematic structure and function, Environment and Planning, Vol. 6, 375-391

8. Kota, S., and Chiou, S.J., 1992, Conceptual design of mechanisms based on computational synthesis and simulation of kinematic building blocks, Journal of Research in Engineering Design, Vol. 4, 75-87

9. Kota, S., and Lee C-L. 1993, General Framework for Configuration Design: Part 1- Methodology; Part 2 – Application to Hydraulic Systems Configuration, Journal of Engineering Design, Vol. 4, No. 4: 277-303

10. Kota, S., Joo J., Li Z., Rodgers S.M., and Sniegowski, 2001, Design of Compliant Mechanisms: Applications to MEMS, Analog Integrated Circuits and Signal Processing, 29, 7-15.

11. Molian, S., 1969, Storage and retrieval of descriptions of mechanisms and mechanical devices according to kinematic type, Journal of Mechanisms, Vol. 4, 311-323.

12. Moon, Y-M, and Kota S. 2002, Design of Reconfigurable Machine Tools, Transactions of the ASME Journal of Manufacturing Science and Engineering, May, Vol. 124: 480-483.

13. Moon Y-M, and Kota S. 2002, Automated Synthesis of Mechanisms using Dual-Vector Algebra, Mechanism and Machine Theory, 37,143-144

14. Murray, R. M., Li, Z. and Sastry, S. S., 1993, *"A Mathematical Introduction to Robotic Manipulator"*, CRC Press

15. Reuleaux, F., 1876, *The Kinematics Of Machinery – Outline Of A Theory Of Machines*, Translated by A. B. W. Kennedy, Macmillan and Co., London.

16. Tsai, L. W., 2000, *Mechanism Design: Enumeration of Kinematic Structures According to Function*, CRC Press.