Kinematic Joint Recognition in CAD Constraint Systems*

Audrey Lee-St.John†

Abstract

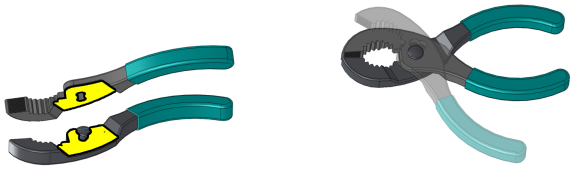
We study the **joint recognition problem**, which asks for the identification and classification of *kinematic joints* from a geometric constraint system. By laying the foundation for a rigidity-theoretic study of flexible body-and-cad frameworks, we obtain an $O(n^3)$ algorithm for identifying prismatic and revolute joints relative to a specific body and an $O(n^4)$ algorithm for finding all pair-wise joints. For a specific subset of body-and-cad frameworks, we present a combinatorial algorithm for identifying all pairs of bodies with prismatic joints in $O(n^2)$ time.

1 Introduction

Computer-aided design (CAD) software allows engineers to design sophisticated mechanical systems by placing intuitive geometric constraints in rigid body *assemblies*. In many cases, motion of the resulting system comprises a large part of the user's **design intent**. We present a novel approach based on rigidity theory for analyzing a flexible structure constructed with CAD constraints. This comprehensive approach considers a CAD system at the global level and can recognize kinematic joints that may be implied by constraints not directly involving the two participating bodies.

Motivation. To provide engineers with meaningful feedback for verifying design intent, many CAD applications offer *motion study* tools to simulate kinematic and dynamic motion. Kinematics simulation allows for quick prototyping, as dynamics is a more computationally expensive task. Once the engineer is satisfied with the kinematics, more robust dynamic simulation can be performed before the product is physically produced.

A common approach to kinematic motion simulation of CAD systems is to perturb the system, then resolve the constraints. This can be computationally expensive as it generally reduces to solving an algebraic system of equations. An alternative approach is to directly model the kinematics, which relies on identifying well-understood mechanical joints, such as the prismatic and revolute studied here.



(a) A plane-plane coincidence constraint used in the design.

(b) Design intent includes rotation.

Figure 1: A pair of pliers, designed in SolidWorks, has a single rotational motion (i.e., a revolute joint). "Slip Joint Pliers" part from `www.3dcontentcentral.com`.

Contributions. In this paper, we establish a rigidity theoretic foundation for understanding *flexible* body-and-cad frameworks. The **joint recognition problem** asks for the recognition of *kinematic joints* described by a geometric constraint system. We consider 3D *body-and-cad* frameworks, which are composed of rigid bodies joined by pairwise coincidence, angular and distance constraints, first introduced in [3]. A constraint is placed between two bodies by identifying a *geometric element* (a point, line or plane) on each body and specifying the relationship between them. We address the identification of two types of kinematic joints, both allowing exactly one degree of freedom: *prismatic* (translational motion) and *revolute* (rotational motion)*.

We present a set of examples highlighting underlying subtleties, lay the mathematical foundation for studying infinitesimal relative motions (or *twists*) of bodies and introduce key concepts for building a theory to analyze them. For motion relative to a particular body, we obtain an $O(n^3)$ algorithm for identifying kinematic joints. A naïve extension provides an $O(n^4)$ algorithm for all pairs of bodies. In the case of prismatic joints, a combinatorial approach for a subset of body-and-cad systems leads to an $O(n^2)$ algorithm.

Related work. Recognition of kinematic joints has a rich history in the CAD community [8], as engineers rely on tools for verifying the kinematic joints dictated by their designed geometric constraint system. Commercial software, such as SolidWorks, Pro/Engineer and the SimMechanics package for MatLab, include a variety of tools for recognizing joints. These proprietary techniques appear to be based on heuristics or mappings from basic constraints to joints. Such mapping

*A preliminary version of this work was presented at the FWCG '10 with Rittika Shamsuddin.

†Department of Computer Science, Mount Holyoke College, `astjohn@mtholyoke.edu`

*In the rigidity theory literature, a *revolute* joint is called a *hinge*.

techniques require explicit constraints between the two bodies where such a joint is recognized; see, e.g., [9]. However, joints may be implied by constraints present in the global system. Recent work [1] presents a *dynamic geometric system* that uses a *filter* to check potential motions with a limited set of *predicates* (currently, this set includes "CircularMotion" and "Rocker").

The methodology in this paper relies on the foundation of body-and-cad rigidity that was presented in [3]. Our approach is similar in flavor to the *witness method* of Thierry et al. [12]. The *witness method* relies on analysis and manipulation, e.g., Gauss-Jordan elimination, of the Jacobian matrix to detect dependencies; maximal rigid components are discovered by using *references*. We refer to the Jacobian matrix as the *rigidity matrix* and *pin* a body in a way that is similar to choosing a reference. However, the references used by the witness method to detect maximal G-well-constrained components must explicitly block the "motions" of a transformation group G. An attempt to extend this approach would require a priori knowledge of the axis of motion for a proposed revolute or prismatic joint. However, our approach not only detects the presence such a kinematic joint, but additionally identifies the axis of motion. Also in contrast to the work of Thierry et al., which uses a *generic* witness for analysis, we work with the rigidity matrix of an *embedded* framework. It is only for prismatic joints in body-and-cad frameworks with certain properties that the behavior appears generic and amenable to combinatorial analysis.

Structure. Section 2 provides motivating examples, then presents preliminaries for the relevant rigidity theory. Section 3 develops the methodology for identification of prismatic and revolute joints. For a special subset of body-and-cad frameworks, Section 4 gives a purely combinatorial approach for detecting prismatic joints. Finally, Section 5 concludes with open questions.

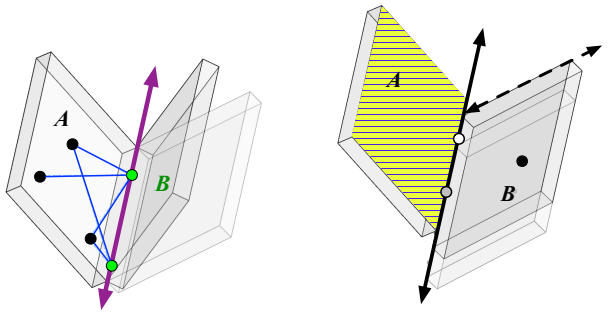
2 Background

In this section, we begin with simple examples to provide intuitions and highlight the subtleties of the **joint recognition problem**. We then provide the preliminaries necessary for describing our contributions.

2.1 Motivating examples

We present several small examples of body-and-cad frameworks that expose some of the difficulties encountered when analyzing motion and identifying kinematic joints.

Simple revolute joint. Figure 1 depicts a simple example of a 2-body system with a single revolute joint. This pair of pliers is composed of two rigid handles with (1) a plane-plane coincidence so they lie adjacent to each other, and (2) a point-point coincidence to force a center of rotation at the desired axis. Since the resulting rotational motion is clearly necessary for the design, identification of this revolute joint would allow verification of user intent. This example contains no dependencies and the rotational degrees of freedom are constrained by exactly two (primitive) angular constraints resulting from the plane-plane coincidence, so analysis seems straightforward: intuitively, we can conclude that there is one rotational degree of freedom.



(a) A revolute joint determined by 5 point-point distance constraints.

(b) A prismatic joint allows a single translational motion.

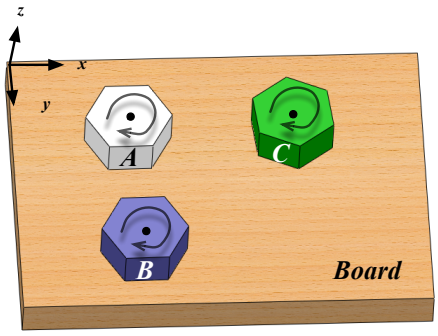
Figure 2: Two-body frameworks with revolute and prismatic joints.

Revolute joint with only blind constraints. Body-and-cad constraints can be separated into "angular" constraints (affecting only rotational degrees of freedom) and "blind" constraints (affecting either rotational or translational degrees of freedom); refer to Section 2.2. This presents a challenge as a revolute joint may be specified without the use of any angular constraints: it is known from classical rigidity theory that a "hinge" (i.e., a revolute joint) may be described by 5 bars (i.e., point-point distances) [11, 14]. Figure 2a depicts this set of bar constraints between two bodies (the three black points lie on body A); there is exactly one rotational motion about the purple (bold) line. Since point-point distance constraints are blind, there does not appear to be straightforward reasoning that would lead us to identify this revolute joint.

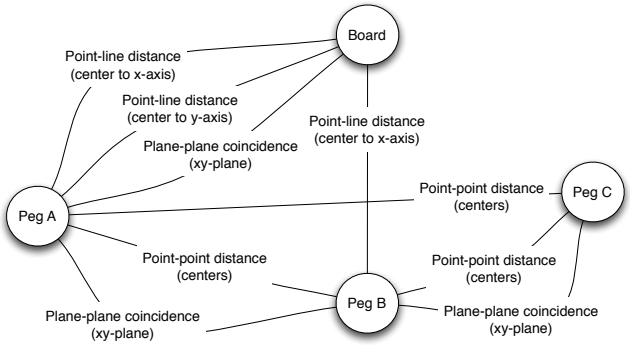
Simple prismatic joint. We return to a simple design for a prismatic joint between a pair of rigid bodies (see Figure 2b). This system is composed of (1) a line-line coincidence along the solid line, and (2) a line-plane perpendicular between the dashed line on body B and the striped plane on body A. These constraints define a prismatic joint: body B may only translate relative to body A along the solid axis. As with the first example, this system seems amenable to analysis as it has no dependencies and the rotational degrees are explicitly eliminated by exactly three angular constraints (two from the coincidence and one from the perpendicular).

Prismatic joint with only blind constraints. This next example again highlights the subtlety of blind constraints. We create an equivalent system to the previous one by using (1) a point-line coincidence between the gray point on body B and the solid line on body A, (2) a point-line coincidence between the white point on body B and the solid line on body A, and (3) a point-plane distance between the black point on body B and the yellow (striped) plane on body A. While the only motion left is a translation along the solid axis, there are no angular constraints present. Intuitively, it seems that three of the bars are somehow modeling angular constraints to eliminate the rotational degrees of freedom.

Pegboard example: implied revolute joints. In the previous examples, the constraints determining a kinematic joint explicitly involved the two bodies. The pegboard shown in Figure 3a provides an example where a revolute joint is implied by constraints not directly involving the two bodies. This system contains 4 rigid bodies: a wooden board along with three "pegs" A, B, and C. The constraints are described by the *cad graph* (formally defined in Section 2.2) in Figure 3b. The pegboard has three non-trivial degrees of freedom associated to three revolute joints: each peg can rotate relative to the board (about the vertical axis through the peg's center point). Notice the lack of constraints between peg C and the board; this revolute joint is implied by the rest of the constraints. In fact, such a design is a realistic result of the difficulties often caused by user interfaces of 3D CAD software (rotating the view to select logical surfaces can be cumbersome).



(a) A 4-body system has a single wooden board with three pegs.



(b) Associated cad graph denotes constraints.

Figure 3: A pegboard example with three revolute joints (one completely implied by indirect constraints).

2.2 Preliminaries

We now present background for body-and-cad rigidity, which provides a foundation for defining relative motion and classifying prismatic and revolute joints.

Body-and-cad infinitesimal rigidity theory. We work with *body-and-cad frameworks*, composed of rigid bodies with pairwise constraints from a set of 21 coincidence, angular and distance constraints. A constraint between two bodies involves a *geometric element* (a point, line or plane) rigidly affixed to each body. Since we rely on the body-and-cad rigidity theory presented by Haller et al. [3], we give a brief overview of the necessary foundations for the infinitesimal rigidity theory.

Formally, a body-and-cad framework is defined by a *cad graph* (G, c) that describes the combinatorics along with a family of 21 "length" functions describing the geometry of the structure. The cad graph (G, c) is a pair, where $G = (V, E)$ is a multigraph with $V = [1..n]$ and c an edge coloring function specifying the cad constraints. See Figure 3b for an example cad graph; we use labels to indicate the edge "colors." In this paper, we will abuse notation slightly and assume that a body-and-cad framework is given by an *embedding* from which the "length" functions can be computed.

Since we work in the infinitesimal rigidity theory, we first consider instantaneous rigid body motion in 3D. As a consequence of Chasles' Theorem (see, e.g., [10]), any instantaneous rigid body motion may be described by a *twist* (translation and rotation about a *twist axis*), which itself is represented by a 6-vector $\mathbf{s} = (\boldsymbol{\omega}, \mathbf{v})$. The 3-vector $\boldsymbol{\omega}$ describes the *angular velocity*: the direction of the twist axis and rotational speed about it. The 3-vector $\mathbf{v}$ can be used to decode the rest of the twist axis and translational speed along it.

Body-and-cad infinitesimal rigidity theory relies on expressing constraints in the Grassmann-Cayley algebra, resulting in the construction of a *rigidity matrix*. This matrix has 6 columns for each body i, corresponding to 2-tensors represented with Plücker coordinates. Since there is a mapping between the 2-tensors in the Grassmann-Cayley algebra and twists, we may interpret these 6 columns representing the degrees of freedom of body i: $\mathbf{s}_i^* = (\mathbf{v}_i, -\boldsymbol{\omega}_i)^\dagger$. The kernel of the rigidity

†The re-ordering and negation are technicalities arising from the development of the rigidity matrix. When referring to elements of the kernel, we will use $\mathbf{s}^*$; when referring to the corresponding twist, we will use $\mathbf{s}$.

matrix describes the *infinitesimal*[‡] *motion space* for a body-and-cad system.

Each body-and-cad constraint is associated to a number of *primitive* constraints (which affect at most one degree of freedom). A primitive constraint between bodies i and j is encoded by a single row in the rigidity matrix that has a vector $\mathbf{x} \in \mathbb{R}^6$ in the columns for body i and $-\mathbf{x}$ in the columns for body j; every other entry in the row is 0. A distinction is made between primitive *angular* and *blind* constraints: a primitive angular constraint may affect only a rotational degree of freedom and corresponds to a row in the rigidity matrix with zeros in the $\mathbf{v}$ columns.

Combinatorics. For a cad graph (G, c), we associate a *primitive cad graph* $H = (V, R \sqcup B)$, which assigns a vertex to each body and red (R) and black (B) edges to primitive angular and blind constraints determined by each cad constraint. Recent work [6] characterizes generic rigidity of body-and-cad frameworks without point-point coincidence constraints.

Theorem 1 *[6] Let F be a body-and-cad framework with no point-point coincidence constraints. Let $H = (V, R \sqcup B)$ be the primitive cad graph associated to F. Then F is generically minimally rigid if and only if there exists a set $B' \subseteq B$ such that $(V, R \cup B')$ and $(V, B \setminus B')$ are each the edge-disjoint union of 3 spanning trees.*

For angular constraints in isolation, *angular rigidity* for *body-and-angle* frameworks was characterized in [7], based on *sparsity counts*.

Theorem 2 *[7] A body-and-angle framework is generically minimally rigid if and only if its associated graph is $(3, 3)$-tight.*

The pebble games of [4] provide $O(n^2)$ algorithms for determining if a graph is (k, ℓ)-sparse (or tight) and for detecting (k, ℓ)-*components* ((k, ℓ)-tight subgraphs that are maximal with respect to vertices).

Relative motions. We consider *relative motions* between a pair of bodies. Let F be a body-and-cad framework, $M(F)$ its rigidity matrix and $\ker(M(F))$ the associated motion space. It must be the case that the 6-dimensional space of trivial rigid body motions is a subspace of $\ker(M(F))$. For a pair of bodies i and j, we restrict the motion space to describe relative motions between the bodies. Formally, we project $\ker(M(F))$ into $\mathbb{R}^{12}$ by a linear transformation described by the following $12 \times 6n$ matrix: the first (respectively, last) 6 rows contain the identity matrix of size 6 in the columns for body i (respectively, j) and zeros everywhere else. Then the *relative motion space* W is the resulting subspace of $\mathbb{R}^{12}$. Again, the 6-dimensional space of trivial motions must be a subspace of W. Define the *number of relative degrees of freedom* to be $\dim(W) - 6$. If this is zero, the two bodies are *relatively rigid*. A *rigid component* is a maximal set of bodies that are pairwise relatively rigid. We observe that the number of relative degrees of freedom is equal to the minimum number of rows (i.e., primitive constraints or edges in the primitive cad graph) whose addition cause i and j to fall into the same rigid component.

Intuitively, to study the non-trivial relative motions, we consider when one body is fixed and seek a description of the allowed motions of the second. To formalize this notion, we may fix body i by appending 6 rows to the rigidity matrix: the identity matrix of size 6 appears in the columns for i and zeros appear in all other entries. We denote the newly obtained *pinned* matrix by $M(F, i)$. The *non-trivial relative motion space for body i* is simply $\ker(M(F, i))$. The *non-trivial relative motion space between bodies i and j* is a projection of the kernel into $\mathbb{R}^6$. The linear transformation used is defined by a $6 \times 6n$ matrix with the identity matrix of size 6 appearing in the columns for body j and zeros everywhere else. The number of relative degrees of freedom is the dimension of this subspace of $\mathbb{R}^6$.

3 Identification of Kinematic Joints

In this paper, we are interested in non-trivial relative motion spaces of dimension 1, spanned by a single twist $\mathbf{s} \in \mathbb{R}^6$. As a consequence of the mapping between twists and 2-tensors in the Grassmann-Cayley algebra, there is a further mapping between twists that are either pure rotations or pure translations and 2-tensors that are decomposable. Decomposable 2-tensors are those that satisfy the Plücker relation, i.e., those that lie on the Grassmannian. We choose the same convention as [3] for the Plücker coordinates, so that the Plücker relation is satisfied for a vector $\mathbf{s} = (\boldsymbol{\omega}, \mathbf{v})$ if $\langle -\boldsymbol{\omega}, \mathbf{v} \rangle = 0$, where the angle brackets denote the dot product. These 6-vectors lie on the Klein quadric and describe lines in 3-dimensional projective space. Furthermore, if $\boldsymbol{\omega} = \mathbf{0}$, the twist has only a translation $\mathbf{v}$ (encoding a prismatic joint). Otherwise, the twist corresponds to pure rotation about the axis (encoding a revolute joint), and the axis of rotation is described by $\mathbf{s}$, the Plücker coordinates of the line. See, e.g., [10, 13], for reference.

We can now describe the **algorithm for identifying infinitesimal prismatic and revolute joints**. To find joints relative to body i in a body-and-cad framework F:

1. Construct the pinned rigidity matrix $M(F, i)$.
2. Compute its kernel $\ker(M(F, i))$.
3. For each body $j \neq i$:

[‡]For brevity, we will omit "infinitesimal" for the remainder of the paper.

(a) Restrict to body j by finding a basis for the non-trivial relative motion space between i and j.

(b) Compute the number of relative degrees of freedom (the dimension of the space).

(c) If there is > 1 degree of freedom, continue to the next body.

(d) Otherwise, consider the single basis vector $\mathbf{s}^* \in \mathbb{R}^6$.

(e) Check if $\mathbf{s} = (\boldsymbol{\omega}, \mathbf{v})$ satisfies the Plücker relation. If not, continue to the next body.

(f) If $\boldsymbol{\omega} = \mathbf{0}$, output *prismatic joint between i and j* with translation $\mathbf{v}$; otherwise, output *revolute joint between i and j* with axis of rotation described via Plücker coordinates $\mathbf{s}$.

Let m be the number of rows (primitive constraints) and n the number of bodies. The dominating factor is step 2 (computing the kernel), which requires $O(\min(n, m)nm)$ time; if we assume a linear number of constraints (e.g., if there are no dependencies), the algorithm has time complexity $O(n^3)$. By executing the algorithm for all $i \in [1..n]$, we obtain an algorithm for identifying all pairs of infinitesimal prismatic and revolute joints that runs in $O(n^4)$ time.

Since the analysis is done at the infinitesimal level, this is not a characterization (i.e., the method may incorrectly identify a relative kinematic joint), but can be used as a filter to identify potential pairs of bodies with prismatic or revolute joints.

4 Combinatorial Identification of Prismatic Joints

We now give a combinatorial algorithm for identifying prismatic joints in a subset of body-and-cad frameworks. We consider frameworks that have no point-point coincidence constraints, dependencies or non-trivial (involving more than one body) rigid components. This allows us to find a combinatorial condition for characterizing when two bodies have a single relative degree of freedom, leading to an algorithm for finding candidate pairs that may have a kinematic joint.

Lemma 3 *For a body-and-cad framework with no point-point coincidence constraints, dependent constraints or non-trivial rigid components, a pair of bodies i and j generically have one relative degree of freedom if and only if they lie in a common $(6, 7)$-component of $H = (V, R \sqcup B)$, the associated primitive cad graph.*

Proof. By [2], a graph is $(6, 7)$-tight if and only if the addition of any edge results in the edge-disjoint union of 6 spanning trees. As a consequence of Theorem 1, $(6, 7)$-components of H are equivalent to subgraphs such that the addition of any edge results in a rigid component. Since the number of relative degrees of freedom between two bodies is the minimum number of edges whose addition results in their being in the same rigid component, i and j have one relative degree of freedom if and only if they lie in a common $(6, 7)$-component. □

To find prismatic joints, we begin by defining *angular-rigid components*. As with Section 2.2, we restrict the motion space to consider only rotational degrees of freedom. For a framework F with rigidity matrix $M(F)$ and motion space $\ker(M(F))$, the *angular motion space* is the space obtained by projecting the kernel into $\mathbb{R}^{3n}$ (intuitively retaining only the rotational coordinates). Formally, we use the linear transformation described by the $3n \times 6n$ matrix with a set of 3 rows for each body i containing the identity matrix of size 3 in the columns for $-\boldsymbol{\omega}_i$ and zeros elsewhere. Instead of rigid body motions, the 3-dimensional space of (trivial) rotations is contained in the angular motion space. The concepts of *relative angular motion space*, *relative angular degrees of freedom*, *relatively angular-rigid* and *non-trivial relative angular motion space* follow analogously. An *angular-rigid component* is a maximal set of bodies that are pairwise relatively angular-rigid.

Lemma 4 *In a body-and-cad framework, a pair of bodies i and j with one relative degree of freedom share a prismatic joint if and only if they lie in a common angular-rigid component.*

Proof. Let F be a body-and-cad framework; let bodies i and j have one relative degree of freedom and denote by W their relative motion space (i.e., $dim(W) = 7$).

Bodies i and j lie in a common angular-rigid component if and only if W_R, their relative angular motion space, has dimension 3. Now consider fixing body i; let W' be the non-trivial relative motion space for j and W'_R the non-trivial relative angular motion space. Then W' must have dimension 1, defined by a single basis vector $\mathbf{s}^* = (\mathbf{v}, -\boldsymbol{\omega})$. W_R has dimension 3 if and only if W'_R has dimension 0 if and only if $\boldsymbol{\omega} = 0$. Thus, bodies i and j share a prismatic joint if and only if they lie in a common angular-rigid component. □

For a subset of body-and-cad frameworks, the detection of (generic) angular-rigid components becomes a combinatorial problem. If F is a framework with rigidity matrix $M(F)$ and primitive cad graph $H = (V, R \sqcup B)$, let $M_R(F)$ be the submatrix determined by the $3n$ columns corresponding to the rotational degrees of freedom $\boldsymbol{\omega}_i$ and the rows associated to the red edges R. We define a body-and-cad framework to be *angular-distinct* if the kernel of $M_R(F)$ is the same as the angular motion space of F. Then, as a consequence of Theorem 2, angular-rigid components in an angular-distinct body-and-cad framework are equivalent to $(3, 3)$-tight components in $H_R = (V, R)$.

We now present a combinatorial **algorithm for detecting prismatic joints** for a body-and-cad framework F satisfying the conditions: (A) F contains no point-point coincidence constraints, (B) F contains no dependent constraints or non-trivial rigid components, and (C) F is angular-distinct.

1. Play the $(6,7)$-pebble game on H to detect $(6,7)$-components.
2. Play the $(3,3)$-pebble game on $H_R = (V,R)$ to detect $(3,3)$-components.
3. Find all pairs of bodies i and j that share a $(6,7)$-component and a $(3,3)$-component; output *prismatic joint between bodies i and j.*

Condition (A) allows the application of Theorem 1. Since dependencies or rigid components result in subgraphs that are not $(6,7)$-sparse, Condition (B) ensures that we find all pairs of bodies with one relative degree of freedom. Without Condition (C), we will still correctly output prismatic joints, but may not find all.

The pebble game algorithms run in $O(n^2)$ time, so Steps 1 and 2 each take quadratic time. In Step 3, the union pair-find data structure developed for rigid components [5] allows us to query if two bodies share a component in constant time and quadratic space. Thus, the entire algorithm has $O(n^2)$ time complexity.

5 Conclusions and Future Work

We initiated the study of understanding flexible body-and-cad frameworks and relative motions from a rigidity-theoretic perspective. This led to the development of an $O(n^3)$ algorithm for detecting infinitesimal prismatic and revolute joints relative to a fixed body and an $O(n^4)$ algorithm for finding all pair-wise kinematic joints. Based on standard linear algebra techniques, this method outputs the type of kinematic joint (revolute or prismatic) as well as the axis of motion.

For the special case of prismatic joints in a restricted set of body-and-cad structures, we gave a purely combinatorial algorithm with $O(n^2)$ complexity, indicating that prismatic joints are more amenable to detection algorithms. This may be due to their correspondence to a 2-dimensional plane in the Klein quadric, whereas revolute joints correspond to all other points on the quadric.

Future work. The combinatorial algorithm for prismatic joints motivates the need for an efficient algorithm for body-and-cad rigidity (including detecting dependencies and rigid components). Further investigation of angular-distinct systems or the transformation (or identification) of blind constraints to angular constraints may elucidate the special treatment of angular constraints with respect to rotational degrees of freedom. More generally, we anticipate that the foundations presented here will allow us to understand motions beyond infinitesimal prismatic and revolute joints.

Acknowledgements. We are grateful to reviewers for insightful suggestions and observations and to Rittika Shamsuddin for participating in the initial work.

References

[1] M. Freixas, R. Joan-Arinyo, and A. Soto-Riera. A constraint-based dynamic geometry system. *Comput. Aided Des.*, 42(2):151–161, 2010.

[2] R. Haas. Characterizations of arboricity of graphs. *Ars Combinatorica*, 63:2002.

[3] K. Haller, A. Lee-St.John, M. Sitharam, I. Streinu, and N. White. Body-and-cad geometric constraint systems. *Computational Geometry: Theory and Applications*, 2010. In press. http://dx.doi.org/10.1016/j.comgeo.2010.06.003.

[4] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 308(8):1425–1437, 2008.

[5] A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. In *Proceedings of the 17th Canadian Conference of Computational Geometry*, Windsor, Ontario, 2005.

[6] A. Lee-St.John and J. Sidman. Combinatorics and the rigidity of cad systems. Accepted to SPM '12: Symposium of Solid and Physical Modeling, 2012.

[7] A. Lee-St.John and I. Streinu. Angular rigidity in 3d: combinatorial characterizations and algorithms. In *Proceedings of the 21st Canadian Conference on Computational Geometry*, pages 67–70, 2009.

[8] K. Lyons, V. Rajan, and R. Sreerangam. Representations and methodologies for assembly modeling. *National Institute of Standards and Technology,Gaithersburg, MD*, 6059, 1997.

[9] O. E. Ruiz. Geometric constraint subsets and subgraphs in the analysis of assemblies and mechanisms. *Ingenieria y Ciencia (Engineering and Science)*, 2(3):103–137.

[10] J. M. Selig. *Geometric Fundamentals of Robotics.* Monographs in Computer Science series. Springer, New York, 2nd edition, 2005.

[11] T.-S. Tay. Rigidity of multi-graphs. I. Linking rigid bodies in n-space. *Combinatorial Theory Series*, B(26):95–112, 1984.

[12] S. E. B. Thierry, P. Schreck, D. Michelucci, C. Fünfzig, and J.-D. Génevaux. Extensions of the witness method to characterize under-, over- and well-constrained geometric constraint systems. *Computer-Aided Design*, 43(10):1234–1249, 2011.

[13] N. White. Grassmann-Cayley algebra and robotics. *Journal of Intelligent and Robotics Systems*, 11:91–107, 1994.

[14] W. Whiteley. The union of matroids and the rigidity of frameworks. *SIAM Journal Discrete Mathematics*, 1(2):237–255, May 1988.