FRONTIER: fully enabling geometric constraints for feature-based modeling and assembly [*]

Jianjun Oung Meera Sitharam [†] Brandon Moro Adam Arbree

CISE department, University of Florida

Abstract

In the full paper [1], we discuss the functionality and implementation challenges of the *Frontier* geometric constraint engine, designed to address the main reasons for the underutilization of geometric constraints in today's 3D design and assembly systems. Here, we motivate the full paper by outlining the advantages of *Frontier.*

1. *Frontier* fully enables both (a) the use of complex, cyclic, spatial constraint structures as well as (b) feature-based design.
   To deal with Issue (a), *Frontier* relies on the efficient generation of a close-to-optimal *decomposition and recombination (DR) plan* for completely general variational constraint systems (see Figure 1). A serious bottleneck in constraint solving is the exponential time dependence on the size of the largest system that is simultaneously solved by the algebraic-numeric solver. In most naturally occurring cases, *Frontier*'s DR-plan is guaranteed to minimize this size (to within a small constant factor). To deal with Issue (b), *Frontier*'s DR-plan admits the independent and local manipulation of features and subassemblies in one or more *underlying feature hierarchies that are input* (Figures 1 and 2).
   A DR-plan satisfying the above requirements is generated by the new *Frontier vertex Algorithm (FA)*: the DR problem and its significance as well as FA and its performance with respect to several relevant and newly formalized abstract measures are described in [2, 3].

2. *Frontier* employs a crucial representation of the DR-plan's subsystems or clusters, their hierarchy and their interaction This representation merges network flow information, as well as other geometric and combinatorial information in a natural manner. Some of this information is obtained from an efficient flow-based algorithm for detecting small rigid subsystems presented in [4]. The clarity of this representation is crucial in the concrete realization of FA's formal performance. More significantly, this representation allows *Frontier* to take advantage of its DR-plan in surprising and unsuspected ways listed below.

[*] Supported in part by NSF Grants EIA0096104, CCR9902025.
[†] Corresponding author: sitharam@cise.ufl.edu

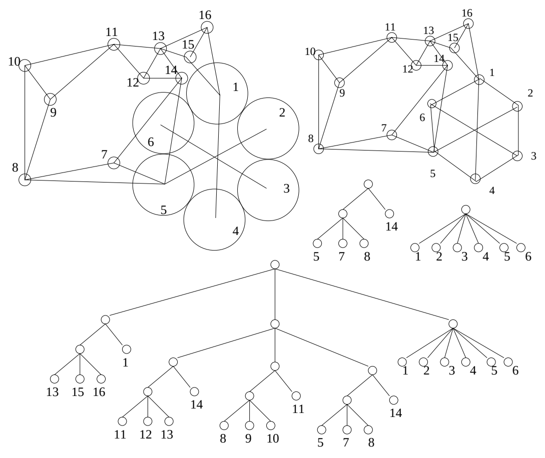


Figure 1: 2D well-constrained variational example: constraint graph with vertices and edges having degree-of-freedom weights 2 and 1; DR plan (big tree) internal vertices represent rigid subsystems sent to the algebraic-numeric solver; DR-plan also incorporates the 2 input feature hierarchies (small trees)

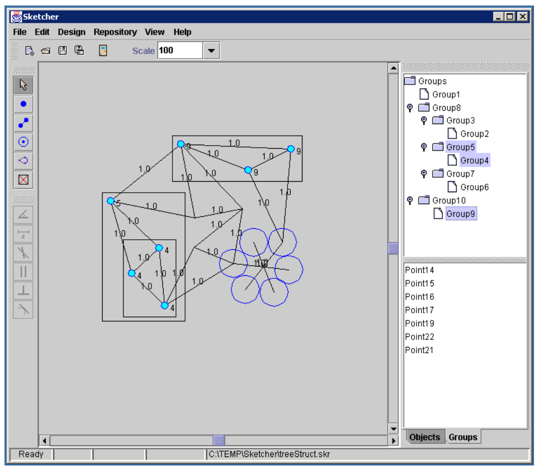
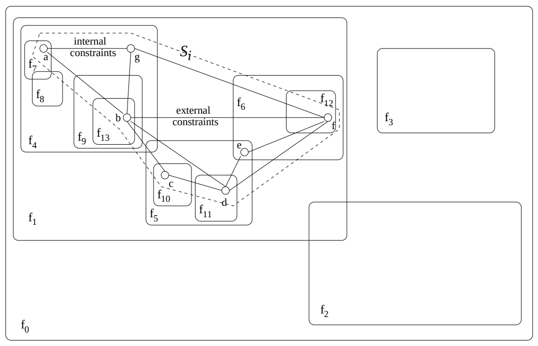


Figure 2: Specifying input partial decomposition in the FUI sketcher; numbers attached to point elements indicate which features they are in; feature hierarchy shown on right; features picked by dragging rectangles or pointed clicks



f_0 f_1	mixed precedural	f_5	Procedural History Based
f_2 f_4 f_{12}	CSG	f_{10}	Parametric Constraints
f_3 f_6	BREP	f_{11}	2D Variational Constraints and Sweeps

Figure 3: A constraint system S_i. Features $f_0, \ldots, f_{11}$ are in mixed procedural representation; a, b, c, d, e are participating feature handles; $f_0, \ldots, f_3$ are features at a higher level of hierarchy in larger CAD model

- It enables complex 3D variational geometric constraint structures that relate parts and features to be mixed in freely with other intuitive representation languages for representing the individual parts and features. Examples of these representations include: explicit drafting on screen, B-rep, CSG, history based specifications, 2d variational constraints along with sweeps extrusions etc; we call these other representation languages collectively as *mixed-procedural representations.* See Figure 3.
- It operates in an optional, on-line setting where it continually renders partial constraint systems as they are incrementally specified. To aid this process, it uses a fast *simple solver* that deals with a certain subset of constraints excluding distance constraints. Using these tools, it also deals with the related problem of under-constrained systems.
- It deals with non-geometric, engineering (equational) constraints; and certain types of soft constraints and inequalities.
- It deals with overconstrained systems, aiding maintenance of multiple constraint views and constraint reconciliation; persistently names shape elements that are not explicitly specified, but may or may not be generated as a result of constraint resolution; and admits parametric constraint solving as a special case.
- Finally, it systematically controls the combinatorial explosion caused by multiple solutions or bifurcations for subsystems, and allows the user to interactively choose intermediate solutions (Figure 4).

3. *Frontier* uses a constraint representation language *frep* that permits all of its components to efficiently share the same datastructures. Moreover it allows new types of shape elements, constraints and features to be easily added to the repertoire. *Frep* is also geared towards mixed procedural representations of features and towards interfacing with the larger CAx system that would call *Frontier.*

4. *Frontier*'s implementation and internal organization of components, coordinated by the Universal Transfer Unit (UTU) - see Figure 5 - satisfies two competing requirements: seamless communication between components as well as complete

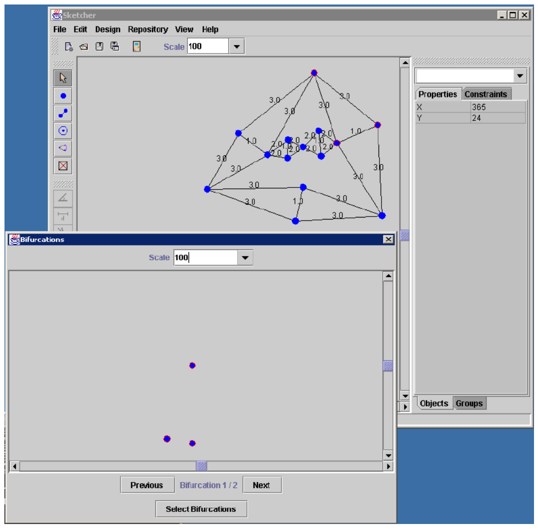


Figure 4: Bifurcation window in FUI sketcher, displaying multiple intermediate solutions to subsystems

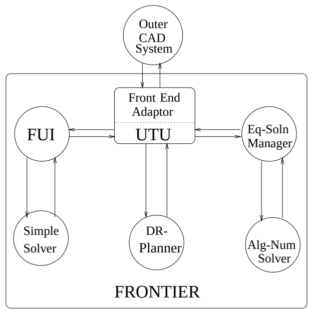


Figure 5: The organization of *Frontier*

portability of each of its components. In addition, *Frontier*'s graphical user interface (FUI) allows full expression of *Frontier*'s capabilities. See, for example Figures 2 4.

References

[1] http://www.cise.ufl.edu/˜sitharam/full-frontier.ps

[2] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition of geometric constraints systems I: performance measures. To appear in *Journal of Symbolic Computation*.

[3] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Decomposition of geometric constraints systems II: new algorithms. To appear in *Journal of Symbolic Computation*.

[4] C.M. Hoffmann, A. Lomonosov, and M. Sitharam. Geometric constraint decomposition. In Bruderlin and Roller Ed.s, editors, *Geometric Constraint Solving*. Springer-Verlag, 1998.

[5] I. Fudos and C. M. Hoffmann. Correctness proof of a geometric constraint solver. *Intl. J. of Computational Geometry and Applications*, 6:405–420, 1996.

[6] J. Owen. Constraints on simple geometry in two and three dimensions. In *Third SIAM Conference on Geometric Design*. SIAM, November 1993.