Extended Abstract

Title: **Applying database optimization technologies to feature recognition in CAD**

Authors: Z. Niu[1], R.R. Martin[1], M. A. Sabin[2], F. C. Langbein[1], J. H. Bucklow[3]
[1] Cardiff University, [2] Numerical Geometry Ltd., [3] TranscenData Europe Ltd.
Corresponding author: Z.Niu, email: mind3str@gmail.com

Keywords: Computer-aided design, Feature recognition, Database optimization

Abstract:

Feature recognition is important in analysis or simulation based on a CAD model [1]; this is done by first meshing the model. Real industrial models normally have many small details, and in many cases, their effect on analysis is minor. Suppressing such details allows meshing to be both quicker and more robust [2, 3], and as a sparser mesh with larger elements results, the time needed for analysis is reduced. Feature recognition can help to find candidates for removal. In computer-aided manufacturing (CAM), computer-aided process planning (CAPP) uses use feature information to generate a sequence of instructions to manufacture a model [2]. Finding features by hand is tedious: automatically finding features as candidates for removal is preferable. Much research has been devoted to this topic [3].

Our approach to feature recognition is based on a high-level declarative feature definition language, which allows users to define new kinds of features to be found. The users merely need to specify *what* constitutes a feature, rather than a specific *algorithm* for finding instances of it. Different applications need different definitions of features: parts of a shape which are important for machining may be quite different to those which can be ignored for analysis for example; features important for heat analysis may be quite different from those for electrostatic analysis. Thus, it is infeasible to hard-code all possible useful features and this must be left to engineers. Our declarative approach has the benefit of allowing the engineers to concentrate on what constitutes a feature rather than how to find it.

In our system, the definition is turned into an SQL query, and a database engine is coupled to a CAD modeler to find instances of entities satisfying the predicates which make up features. The purpose of this paper is to show that such an approach is feasible, and that in particular, the built-in *database optimization techniques* can effectively execute the query in an acceptable time to find features–without optimization, such a declarative form would be far too slow. Database optimization techniques automatically determine a suitable set of sub-queries and an appropriate order to execute them in. We compare this approach with optimization techniques proposed by Gibson [4], who also used a declarative approach, but not based on a relational model.

To test the ideas above, we have implemented a feature finder which integrates the SQLite database engine and CADfix [5] modeler–see Figure 1. The feature finder works as a front-end to the CAD modeler, which reads in the CAD model in which features are to be found. The feature finder reads in feature definitions, and converts requests to find such features into an SQL query. The query planner in the database engine transforms the query into a simpler set of sub-queries, using an optimization strategy to choose the expected cheapest approach based on estimated cost. These sub-queries are then executed, reporting all instances of the defined feature in the model. A tokenizer and parser read feature definitions and execute other commands such as OPEN model, DEFINE type, LOAD / SAVE definition, FIND feature, DRAW result. Features are defined using entities and predicates; note that the same feature can be defined in multiple ways. These definitions can also be translated into SQL in different ways. The general form of feature definition we adopt is shown in Figure 2. After naming the feature, various entities are defined: they may be low-level components of the model such as vertices, edges, and faces; they may also be sub-features e.g. a CYLINDRICAL HOLE. Each entity has a given type which restricts the search domain of the feature, e.g. e1, e2: EDGE means all edges of the model should be considered as candidates for e1 and e2. The predicates indicate relationships that the entities should satisfy. Each predicate refers to one or more entities. Typical unary predicates concern geometric properties, e.g.

CONVEX(e1). Binary predicates concern relationships between entities, e.g. BOUNDS(f1, e1). Externally visible entities are defined after EXPORT when this type is used as a sub-feature.

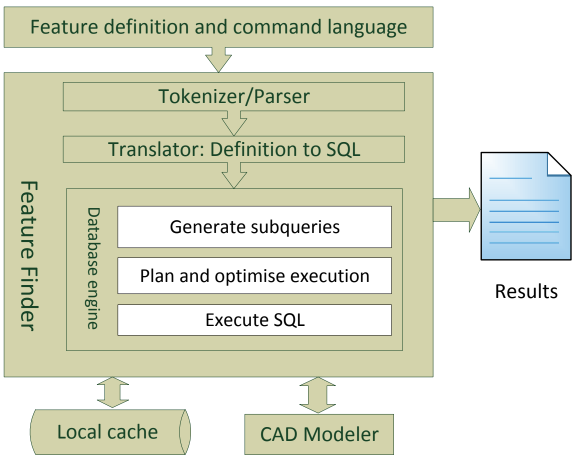

Figure1. Architecture of the feature finder

```
DEFINE FEATURENAME AS
    e1, e2 ...: EDGE;
    f1, f2 ... : FACE;
    h1, h2... : CYLINDRICAL_HOLE;
    ...
SATISFYING
    e1<e2;
    CONVEX(e1);
    BOUNDS(e1, f1);
    ...
EXPORT
    e1, f1
END
```

Figure2. Declarative feature definition.

To test the ideas, SQLite was modified to switch various query optimizations (re-ordering joins, indexing, subquery flattening) on and off. A series of experiments using both carefully constructed artificial models and real industrial models has been carried out to see if this approach is viable, and the extent to which various optimizations help. We have investigated how the feature finding time varies with the size of the model, the number of instances of the feature in the model and the number of entities in the feature. We have also compared the optimizations available to those considered by Gibson [4].

Results show that, as hoped, database optimization provides significant benefits to the time taken to find features (e.g., for notch features, the observed scaling complexity of the algorithms is reduced to about $O(n^2)$ comparing to an unoptimized time of $O(n^6)$, where n is the number of edges in the model. However, optimization does not always provide the best result: different ways of defining the same feature have different performance even after optimization. Future work will consider databases with more powerful optimization, and the benefits of indexing.

Acknowledgements

This work was financially supported by the Framework Program 7 Initial Training Network Funding under Grant No. 289361 "Integrating Numerical Simulation and Geometric Design Technology".

References

1. Lee, K., et al. A small feature suppression/unsuppression system for preparing B-rep models for analysis. in Proceedings of the 2005 ACM symposium on Solid and physical modeling. 2005. ACM.
2. Han, J., M. Pratt, and W.C. Regli, Manufacturing feature recognition from solid models: a status report. Robotics and Automation, IEEE Transactions on, 2000. 16(6): p. 782-796.
3. Babic, B., N. Nesic, and Z. Miljkovic, A review of automated feature recognition with rule-based pattern recognition. Computers in Industry, 2008. 59(4): p. 321-337.
4. Gibson, P., H. Ismail, and M. Sabin, Optimisation approaches in feature recognition. International Journal of Machine Tools and Manufacture, 1999. 39(5): p. 805-821.
5. TranscenData Europe Ltd., CADfix 9.0. http://www.transcendata.com/products/cadfix.