INTERNATIONAL DESIGN CONFERENCE - DESIGN 2006
Dubrovnik - Croatia, May 15 - 18, 2006.

DESIGN 2006

EMBEDDING GENERAL CONSTRAINT RESOLUTION INTO A CAD SYSTEM

B. Singh, J. Matthews, G. Mullineux and A. J. Medland

Keywords: design constraints, constraint modelling, CAD

1. Introduction

Computer aided design (CAD) environments have provided designers with increasingly greater capabilities over the last few decades. Initially only two-dimensional drafting was supported. Now many systems work using fully three-dimensional assembly modelling together with parametric capabilities. They can be integrated within a company's data management, planning and manufacturing facilities. Earlier CAD systems were simply repositories for geometric entities, and it was the user who provided the interpretation of what these represented. As systems have evolved, there has been a desire to make them more "intelligent" and hence be able to take some account of the meaning of the geometry. One of the ways which has been investigated is the use of constraints. These can be imposed, for example, to specify something of the function of the design by showing limiting relations between geometric entities, or to indicating interactions during the operation.
The advent of feature-based CAD systems has meant that components are effectively created parametrically and relations between the individual geometric identities are specified (perhaps implicitly) while a user creates a component. This means that the component can be reconstructed in different forms by changing the defining parameters. The relations are essentially constraints. When a component is reconstructed it is necessary for the system to be able to resolve these successfully. In this way, constraints are applied to geometric entities, and modelling systems can take advantage of the fact that the forms of these are known a priori. As an example, some CAD systems have the ability to perform assembly of components and, more generally, simulate motion of mechanical systems and obtain kinematic data. Those with the appropriate analytical support can provide dynamic information. owever, such tools rely upon the software having specific tools built within it. There is need to allow more general constraints to be imposed upon a design: constraints which involve the design parameters but which may be dependent upon to a particular design application and so cannot be foreseen when the software is created. There are many examples of these. A car engine needs to be powerful enough to drive the vehicle and yet small enough to fit into the space available for it. There are often trade-offs required between performance and "cost", as, for example, when trying to improve the performance of an existing machine but wishing to keep the number of new parts low. In the food industry, there are decisions to be made on pipe work (based upon consideration of non-Newtonian flow) so as to ease production without compromising the quality of the product.
Constraints can also be used in the conceptual stages of design when the underlying geometry is not yet available [Deng et al 2000]. The lack of precise geometry hampers the application of conventional CAD techniques. Instead, the design is effectively represented (notionally rather then geometrically) in terms of the individual components from which it will ultimately be formed. This gives access to design variables in terms of the parameters and properties of the components. The components can be linked together so that output parameters from one act as inputs to another, as in Schemebuilder [Bracewell et al 1996]. There are then constraints which relate how the parameters, particularly the

inputs and outputs, interact and what are the limits of operation. Once an initial design has been obtained, the feasibility of this can be verified by checking that none of the constraints is violated.
To investigate such general constraints, a stand-alone constraint modelling system has been created [Mullineux 2001]. Via a user language, design parameters can be declared and constraints imposed between these. These are resolved using optimisation techniques, which means that no assumptions are made about the form of parameters involved in the constraints. This has several advantages. One drawback however is that the lack of involvement with a CAD system means that geometry has to be imported to constraint modeller and exported from it, and there is a lack of facilities for a user to modify or interrogate the geometric model. The purpose of this paper is to investigate the integration of such a stand-alone constraint system with a commercial CAD system so that the advantages of both may be obtained.
Section 2 discusses the ideas of constraints in general and section 3 introduces the stand-alone constraint modelling system. Section 4 shows how the applications programming interface (API) of the CAD system can be used to allow the constraint modeller to access the underlying geometric data and hence achieve a successful integration. Section 5 gives a case study example of modelling a machine for wrapping confectionary, and the final section gives some conclusions.

2. Constraints

Often in a design process, constraints exist that impose limits on what is possible within the design. These limits can results from many sources, ranging from customer requirements through to the manufacturing methods available. These constraints drive the product design process. The aim is to find a design solution where all the constraints are either fully satisfied or, at least, an acceptable comprise is achieved between them.
In recent years, the ability to handle constraints has started to appear within commercial CAD systems - particularly those which are described as being feature-based. Here constraints are applied to the geometry of the component or components being modelled. Even when three dimensional objects are being created within a CAD system, the starting point for a user is often the creation of a two dimensional profile which is then extruded (along a straight line, circular arc, or some other curve) to produce a solid object. Constraints are often applied to the initial profile to insist that lines are parallel or perpendicular, arcs and lines are tangential, and so on. When three dimensional objects have been created, it is usually possible to assemble these and again constraint relations are often available. These might specify that faces from different objects need to be coplanar, or that lines in different objects are to be collinear.
CAD systems which possess a geometric interface language can also allow further constraints to be defined in terms of parametric variables. For example, it may be possible to specify that the lengths of the sides of a rectangular block need to be in a given proportion, or that the height of a cylinder cannot exceed its diameter.
However these constraints are essentially only concerned with the underlying geometry of a part or parts. There can be a requirement to deal with other forms of constraint. For example, there may be a requirement to limit the speed of motion of a part during the operation of a machine, or to ensure that a part is sufficiently strong to support another.
Much research has been undertaken into the way that constraints can be identified and resolved. A lot of this has been directed towards constraints arising from purely geometric considerations [Bouma et al 1995, Anderl et al 1996, Hoffmann 2005]. Here several approaches are available. One strategy is to try to order the constraints so that they can be solved in sequence. An alternative is to group constraints together and then look to solve the underlying equations simultaneously by an appropriate numerical method.
Comparatively little work has been undertaken on more general constraint-based applications. Presumably this is because work external to an underlying CAD system is required. In the application to functional design verification [Deng et al 2000], the functional description of the design is captured in graph form based upon the interrelation of the components as prescribed by the user. This means that dependency graphs for the design parameters can be created and constraint propagation methods employed.

In another approach [Hicks et al 2001], a stand-alone constraint modelling system is used to help in the design process for the mechanisms required in packaging machinery. While this has been successful, its independence from a full CAD system does bring some limitations in terms of modelling and displaying the underlying geometry. This means that there may be advantages in integrating such a stand-alone system with commercial CAD software. It is this idea that is investigated here. It should be noted that this may result in some overlap of capability, as both systems ought to be able to handle the purely geometric constraints. However, it is thought that the ability to impose additional forms of constraint and to have greater control over the purely geometric ones may offer some benefits. The stand-alone constraint modeller is discussed in the next section.

3. Constraint modeller

The constraint modeller is software that was created to investigate the use of constraint techniques in the design of mechanism and machine systems [Mullineux 2001]. Underlying it is a user interface language in which the parameters for a given design task can be created and manipulated. In its basic form, the software allows the creation of geometry in terms of three dimensional wire-frame entities such as lines and arcs. These are treated as parameters within the user language.

The underlying language acts as a normal programming environment: variables can be declared, expressions between variables can be evaluated and graphical entities can be displayed. However, the system also allows constraints between parameters to be imposed. Constraints are defined as expressions between variables. These expressions are deemed to be true when they are zero; non-zero values are really a measure of the falseness of the constraint. For example, if three parameters *x*, *y* and *z* need to be constrained to have a sum equal to 6, then the following constraint rule is applied.

```
    rule( x + y + z - 6 );
```

Normally several constraints are imposed together and the user specifies which of the design parameters can be varied in order to seek a feasible solution. The system uses optimisation techniques to try to resolve constraints [Ge et al 1999]. The sum of the squares of the constraint values is formed and a minimum is sought by manipulating the free design parameters. This has the advantage of finding some form of "best compromise" solution even when constraints are in conflict.

Constraints can also be imposed between geometric entities. For example, suppose *L2* and *L3* are two lines that have been created. Within the language structure, the expression *L2:e2* denotes the second end-point of *L2*. The following constraint rule

```
    rule( L2:e2 on L3:e2 );
```

is used to specify that the two lines should touch at their end-points. Here *on* is a binary function built into the user language. Effectively it finds the distance between two geometric objects. When that distance is zero the objects coincide and hence the function can be used within constraints to define mating conditions.

The approach can be used to create assemblies of components and to simulate the motion of mechanism systems. An assembly can be created entirely in terms of constraints such as the one above. However it is possible to simplify the process. Geometric entities can be grouped within model spaces [Leigh et al 1989]. Each space is associated with a transformation to indicate how it maps into world space. Alternatively, the transform can map into another model space so that a hierarchy of model spaces in created. This is a tree structure whose "root" is the world space.

The hierarchy allows a partial assembly of objects to be created. When the transform of one model space is modified, the geometry of all spaces closer to the world is left unchanged, and all the geometry of spaces further away moves together. However the assembly created by the hierarchy alone is only partial; typically additional constraints are required to form the full assembly. As an example, consider the four bar linkage illustrated in part (a) of figure 1. This shows the three moving links as line segments. Each line is contained in a model space and the hierarchy of these is shown in part (b), where *W* denotes the world space.

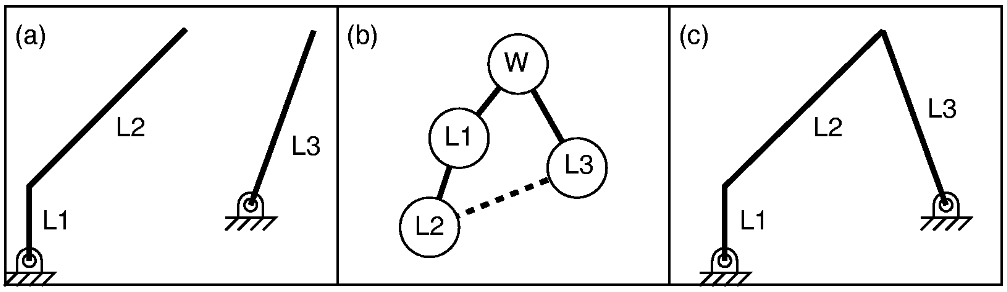


Figure 1. Assembly of a mechanism

The hierarchy ensures that lines *L1* and *L2* stay together. To complete the assembly, the ends of lines *L2* and *L3* needs to be joined as suggested by the dashed line in hierarchy diagram. This requires a constraint which is in fact precisely the one displayed above. This is imposed by allowing the system to vary the angles of the model spaces of lines *L2* and *L3*. The result is the fully assembled mechanism shown in part (c) of figure 1.

4. Incorporation within a CAD system

The main purpose of this paper is to investigate the incorporation of a stand-alone constraint modelling system with an existing CAD package. The aim is to allow a range of constraints greater than that provided by the CAD package alone to be made available to a user. The NX3 software of Unigraphics has been used as the CAD system. This offers a number of modes of application. The most basic is “part modelling” in which the user creates models of individual components. Built over this are a number of applications. One is assembly work in which individual components are put together. Other applications include finite element analysis, and the creation of manufacturing instructions.

This integration of the CAD package with the constraint modeller has been accomplished using Open API which is the application programming interface language of NX3. An application programming interface provides the means to interface different standards of software. Using Open API it is possible to integrate third party applications or programs within NX3 environment. It provides necessary routines, procedures, variables and data structure for the communication with different pieces of software. The integration has been achieved with NX3 working in its part modelling mode.

Figure 2 shows the structure of the interfaced system. The user interaction and display is provided by the CAD system. The language interpreter receives commands from the command window via this user interface and creates and manipulates the user variables. Expressions involving variables, that may include geometric objects, can be evaluated. Constraint expressions can be dealt with using the constraint resolver.

The mai command handling and constraint resolving parts of the constraint modeller operate as previously. The main difference in the integrated system arises when geometry (solid or wire-frame) is created. Here, the appropriate Open API call is made to create the object within the memory assigned to the CAD software. The creation is handled by NX3 itself and it returns a pointer (handle) to the geometric object. This is held with the constraint modelling software as the “value” of the corresponding design variable.

One of the crucial needs is to be able to apply transforms determined by the constraint modeller to objects with the CAD system. Again Open API allows this to be done: the appropriate interface procedure is called, passing to it both the matrix transform and the pointer to the object. In particular, in this way, the constraint modelling software can create assemblies within the CAD system and can then control simulations of their motion.

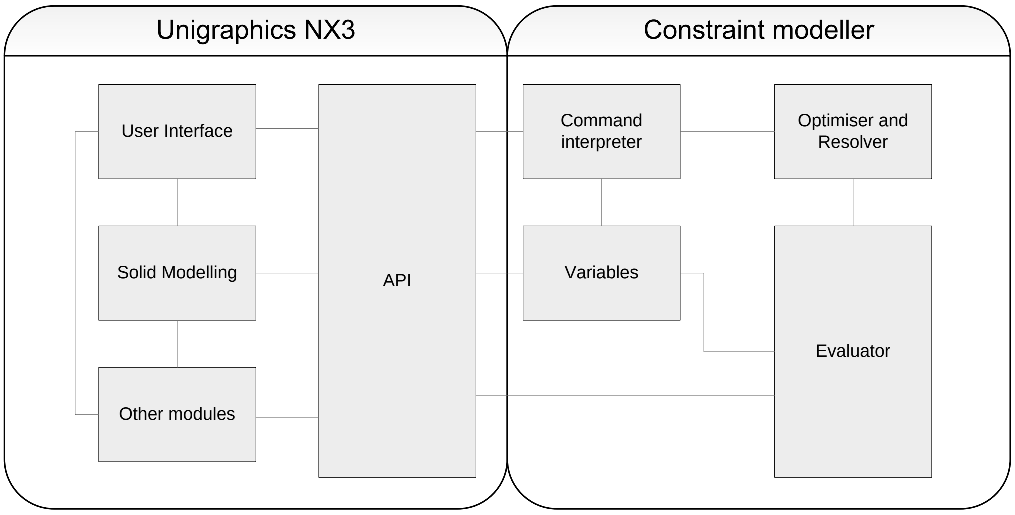
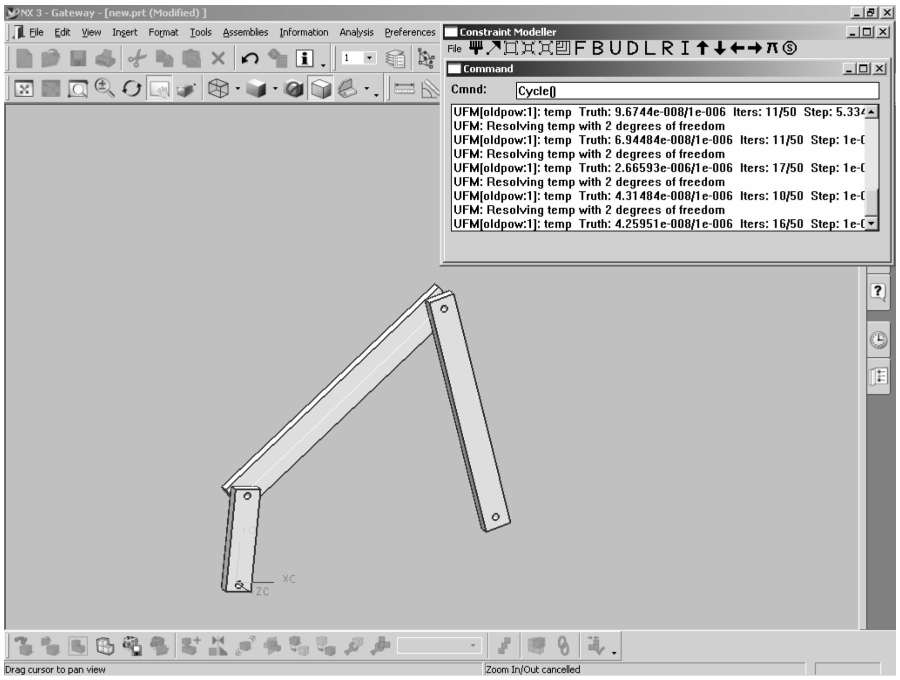


Figure 2. CAD - constraint modeller integrated system

At a basic level, the assembly of objects can be carried out within the CAD system itself. However, transferring the control to external software allows the form of the hierarchy of model spaces to be recast via the user language should other means of driving the system be required. It also allows other constraints beyond mere assembly to be imposed; for example to specify kinematic relations in terms of velocity and acceleration, to avoid clashes with other parts of a design, and to investigate possible failure conditions.


NX 3 - Gateway - [new.prt (Modified)]
File Edit View Insert Format Tools Assemblies Information Analysis Preferences
Constraint Modeller
File
F B U D L R I
Command
XC
ZC
Drag cursor to pan view
Zoom In/Out cancelled

Cmnd: Cycle()

```
UFM[oldpow:1]: temp  Truth: 9.6744e-008/1e-006  Iters: 11/50  Step: 5.334
UFM: Resolving temp with 2 degrees of freedom
UFM[oldpow:1]: temp  Truth: 6.94484e-008/1e-006  Iters: 11/50  Step: 1e-0
UFM: Resolving temp with 2 degrees of freedom
UFM[oldpow:1]: temp  Truth: 2.66593e-006/1e-006  Iters: 17/50  Step: 1e-0
UFM: Resolving temp with 2 degrees of freedom
UFM[oldpow:1]: temp  Truth: 4.31484e-008/1e-006  Iters: 10/50  Step: 1e-0
UFM: Resolving temp with 2 degrees of freedom
UFM[oldpow:1]: temp  Truth: 4.25951e-008/1e-006  Iters: 16/50  Step: 1e-0
```

Figure 3. Integrated system interface

Figure 3 shows a screen shot of the integrated system. This shows the basic CAD screen with a simple mechanism displayed. The additional window is that dealing with constraint modelling commands.

5. Case study application example

The example presented in this section is based upon part of a sweet wrapping machine. More specifically, the gripper mechanism is considered. This is used to pull film out from a reel. Once a sufficient amount of film has been withdrawn, it is cut off. This results in a portion of film being positioned above the sweet to be wrapped. A separate mechanism then moves the sweet vertically upwards and into the film, thus starting the wrapping process. The gripper mechanism is shown in part (a) of figure 4. The mechanism is controlled by two cams: one effectively deals with the forward and backward motion, while the second is used to introduce a slight upward movement of the film as it is being drawn across the next sweet.

Part (b) of the figure shows a "stick diagram" of the mechanism. Each line represents an element of the mechanism and the closed curves are the driving cams. These elements are each embedded into its appropriate model space. As before, the model spaces hierarchy can be used to perform most of the assembly of the links. The assembly is completed by imposing constraints so that the two links which have cam followers lie on the appropriate cam. For example the constraint to position one of the cam followers onto its cam profile is the following.

```
    rule( camfollower1:e1 on cam1 );
```

In the system in which the constraint modeller is integrated with the CAD software, solid objects representing the links can be created. The model space hierarchy and the constraints can be used to assemble these (based on the stick diagram), and part (c) of figure 4 shows their assembly. Allowing the cams to rotate in steps and resolving the constraints at each stage produces a simulation of the motion.

Such a simulation allows the track of the end of the gripper to be investigated. Because the assembly is determined purely by the constraints, it is possible to attempt to improve the design of the output motion. For example, the path of the gripper can be modified. Then the same constraints used for the simulation can be imposed to ensure that the gripper follows the new output; this then determines the profiles of the two driving cams.

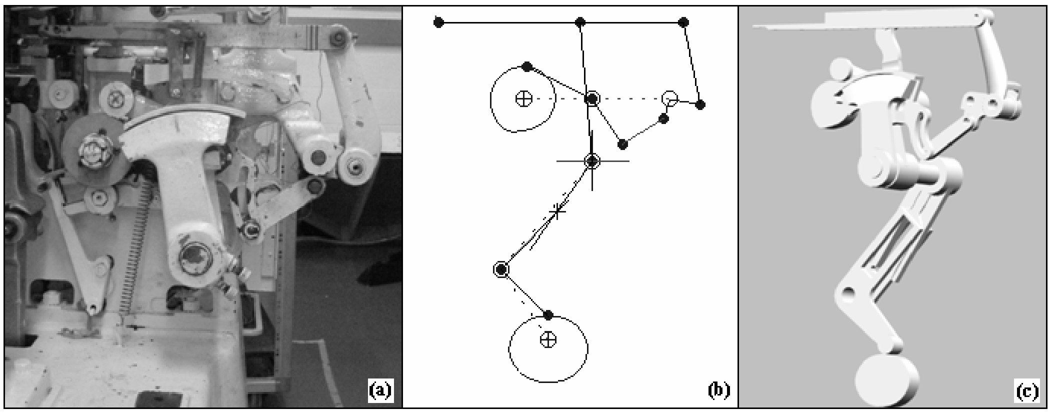


Figure 4. Film grip mechanism

6. Conclusions

Constraints arise frequently in design work and the appearance of feature-based CAD systems means that constraint resolution is now an integral part of such design tools. However, such constraints can often only be applied to geometric entities, and constraints which express more general information about the functionality of a design cannot be dealt with. For these types of application more specialised tools are required, and these can be distinct from the CAD system.

It has been seen that it is possible to integrate a stand-alone constraint modelling system with a commercial CAD package. Use has been made of the application programming interface (API) provided with the CAD tool. This means that the constraint modeller can define the underlying geometric entities and can have subsequent access to these via pointers. In particular, it is possible for the external modeller to apply transforms to the CAD entities and hence take control of assembly processes. This allows the user greater interaction with the assembly and hence provides the ability to investigate different arrangements.

The combined system has been demonstrated in its use to model, assemble and simulate the action of part of a confectionary wrapping machine. This has allowed the motion of the end-effector to be investigated and improved.

Acknowledgement

The research work described in this paper has been carried out with the support of DEFRA research grant given as part of the Food Processing Faraday. This support is gratefully acknowledged together with that of other collaborators.

References

Anderl, R. and Mendgen, R., "Modelling with constraints: theoretical foundation and application", Computer-Aided Design, Vol. 28, No. 3, 1996, pp 155-168.

Bouma, W., Fudos, I., Hoffmann, C., Cai, J., and Paige, R., "Geometric constraint solver" Computer-Aided Design, Vol. 27, No. 6, 1995, pp 487-501.

Bracewell, R. H. and Sharpe, J. E. E., "Functional descriptions used in computer support for qualitative scheme generation - Schemebuilder", Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 10, 1996, pp 333-345.

Deng, Y.-M., Britton, G. A. and Tor, S. B., "Constraint-based functional design verification for conceptual design", Computer-Aided Design, Vol. 32, 2000, pp 889-899.

Ge, J.-X., Shang-Ching, C. and Gao, X.-S., "Geometric constraint satisfaction using optimization methods", Computer-Aided Design, Vol. 31, 1999, pp 867-879.

Hicks, B. J., Medland, A. J. and Mullineux, G., "A constraint-based approach to the modelling and analysis of packaging machinery", Packaging Technology and Science, Vol. 14, 2001, pp 209-225.

Hoffmann, C. M., "Constraint-based computer-aided design", ASME Journal of Computing and Information Science in Engineering, Vol. 5, 2005, pp 182-187.

Leigh, R. D., Medland, A. J., Mullineux, G., "Model spaces and their use in mechanism simulation." Proc. Instn Mech. Engrs - Part B - Journal of Engineering Manufacture, Vol. 203, 1989, pp 167-174.

Mullineux, G., "Constraint resolution using optimisation techniques", Computers & Graphics, Vol. 25, 2001, pp 483-492.

Mr Baljinder Singh
Research Officer
University of Bath
Department of Mechanical Engineering
Bath, BA2 7AY
United Kingdom
Tel.: +44 (0)1225 385937
Fax.: +44 (0) 1225 386928
Email: B.Singh@bath.ac.uk
URL: http://www.bath.ac.uk