



Assembly Feature Tutor and Recognition Algorithms Based on Mating Face Pairs

Adam Dixon and Jami J. Shah

Arizona State University, {adam.c.dixon,jami.shah}@asu.edu

ABSTRACT

Assembly features encode geometric and functional relationships between parts in an assembly. They can be used in assembly design by features, kinematic and structural analyses, assembly planning and assemblability evaluation. Explicit definition of assembly features and archival in libraries can be used to assist in design and analysis of mechanical assemblies. This paper builds on our past work in neutral definition of generic assembly features to provide a mechanism for user-defined custom assembly features using an interactive system, called the assembly feature tutor. The user teaches the system interactively by examples and the tutor outputs an assembly definition file that is written in EXPRESS like language. Central to the definition are pairs of mating part faces and includes algebraic, geometric and parametric relations. Twist and wrench matrices can be extracted from face pair relations. The definition is implementation independent and provides a uniform way to define assembly features. The paper also describes the implementation of an automatic recognition system for user defined assembly features. The algorithm is based on contact pairs which is found to be more efficient than those based on pre-recognition of part features, as the issue of feature interaction is avoided.

Keywords: assembly features, assembly feature definition and recognition.

DOI: 10.3722/cadaps.2010.319-333

1 BACKGROUND

Almost all mechanical and electro-mechanical devices are assemblies of multiple parts. There are many factors why this is so, some related to functionality and some to manufacturing. Functional reasons include requirement for relative motion, interfacing of moving and stationary parts, need for multiple states, power transmission and structural integrity. Manufacturing reasons include use of different materials, shape/size differences, and assemblability. Thus, we can say that assembly design is the very crux of engineering design. Yet CAD is not particularly compatible with assembly level thinking. Designers think in terms of assembly functions but in CAD one must fully define individual parts before assemblies can be created. Incompatibility also exists between CAD and computer aided kinematic analysis: the designer needs to manually define joints and reference, moving coordinate systems even though the information is implicit in the assembly CAD model. Not only is this time consuming but it is also difficult to keep track of redundant and over constraint situations.

There are many types of assemblies: static, rotational, articulated and mixed. The specific design process varies with assembly type and function. Major design tasks typically include shape/size design of parts; interfacing of components; layout, packaging; kinematic, dynamic, structural analyses; motion simulation and interference detection. Auxiliary tasks may include manufacturability (DfM) and assemblability (DfA) analysis. In addition to nominal design of an assembly, there is also tolerance design which determines the GD&T scheme and allowable manufacturing variations to ensure proper functioning and assemblability.

Assembly features can be used in mechanical design for supporting each of the above functions. Designing assemblies using assembly features can allow a designer to create an assembly model in a more fluid fashion. The transition from concept to design is easier since most designers already follow a top-down design approach. As well, assembly features can help in assembly planning. Assembly features contain constraints and restricted motion limits that can be exploited to determine ways that parts can be assembled together or disassembled. Assembly analysis can also take advantage of assembly features. Assembly features can be used to determine degrees of freedom and motion limits of an assembly. These uses of assembly features are not readily available in most commercial CAD systems. With the use of assembly features in conjunction with computer systems, these processes can be automated, and therefore take advantage of assembly features to create more robust CAD systems.

In order to provide a way for a user to define assembly features, this paper describes a method that has been implemented to define a library of assembly features, and also a method to evaluate an assembly for user defined assembly features. The background to this paper starts with the development of an information model for storing assembly data for legacy systems engineering [16, 14]. The model is called OAM+, since it is a modified version of Open Assembly Model (OAM). Next, a neutral definition of assembly features was created [15]. This research lays the ground work for presenting a uniform definition of assembly features. This paper is a continuation of our previous work and describes and demonstrates the implementation of a system that interactively defines assembly features (tutor) and uses these definitions to find assembly features in any assembly CAD model.

2 ASSEMBLY MODELING

An assembly can be viewed many different ways. Figure 1 shows a mechanical assembly of a rotary engine in both assembled and disassembled form. This assembly can be viewed as individual parts with constraints between the parts of the assembly. The hierarchy of the assembly below is shown in Figure 2. In this model, joints can be defined, such as revolute, but much of the information that is contained in assembly features is missing. And these joints and links must be manually input by the user; as shown in Figure 3. With the use of assembly features, these joints and conditions can be automatically determined and further assembly analysis can proceed.

A part feature has been defined as a stereotypical shape with certain topological and geometric properties [21]. Similarly, an assembly feature is defined as a stereotypical association between two part features that are on different parts. Whitehead posits that parts get position from location features and keep it with effector features [27]. Whitney defines assembly features as a description of how parts are joined to each other [28]. Bordegoni and Cugini [2] defined assembly features as attachments, pass-through and connect types. Research by Sambhoos [20] contain assembly feature that are have direct, indirect or interfering relationships.

One thing that all assembly feature definitions have in common is that they all contain some type of connection. We define assembly features as an association between two parts which contain face connections between parts, constraints, parameters, kinematic relations, and structural relations. A formal definition has been created [15] to provide a uniform definition, which is implementation independent, of an assembly feature.

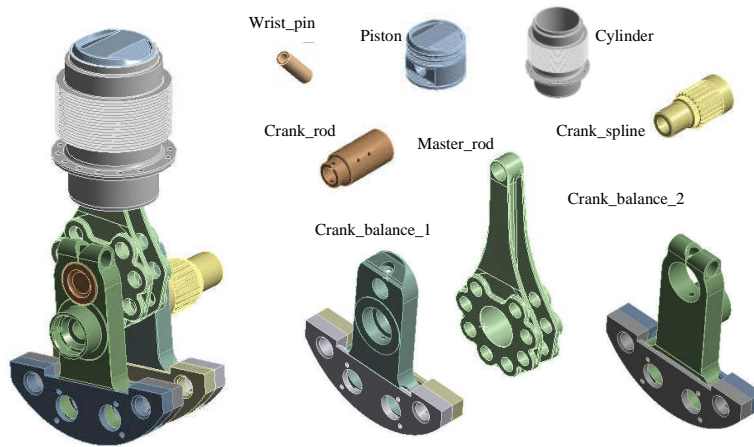


Fig. 1: (a) Engine assembly, (b) disassembled parts.

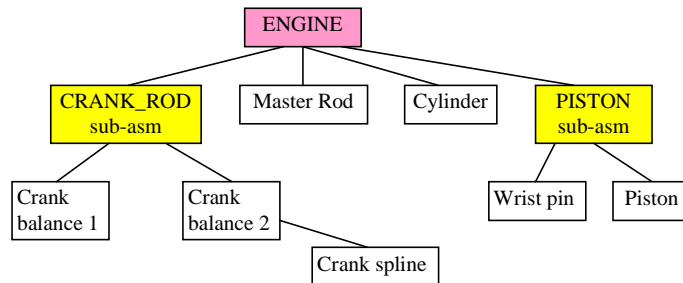


Fig. 2: Assembly hierarchy.

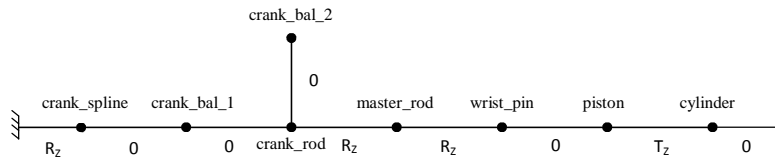


Fig. 3: Kinematic diagram of engine assembly: links, joints, DoFs.

3 LITERATURE REVIEW

Although a great deal of literature exists on part feature definition and recognition, the same cannot be said about assembly features. Brunett [4] developed an assembly model that supports conceptual development according to the Pahl & Beitz procedure [18]. Gupta et. al. [8] created models for assembly sequence generation. Noort et. al. [17] used assembly features to provide multiple views of a product. Liang [11] used a port model to define interaction between a component and its environment or interfaces between components. DeFazio [6] proposed a feature-based assembly system to specify the mating relations between components and relative extraction directions for use in assembly planning. Sodhi [23] used assembly features for specification of relations between components at a high abstraction level. Boredegoni [2] classified assembly features as attachment, pass-through and connect types.

For large assemblies, assembly planning is extremely important. Wang et al [26] created a system that contains an algorithm to generate all available assembly sequences for an assembly planning

problem in automobile assembly lines. Their goal was to automate the process so that it speeds up the assembly planning stage at the prototype design stage. Butterfield [5] established the benefits of virtual simulation of large assemblies in the aerospace industry. The purpose of this research was to show how digital manufacturing techniques could improve existing manufacturing assembly practices in commercial aircraft. By providing a means of assembly planning, the manufacturing engineers are able to assist more directly in the assembly planning process. A method to reorganize large assembly models from as-designed assembly models to as-built assembly models was created by Jayaram [9]. This restructuring allows for the assembly model to be evaluated for assemblability and manufacturability along with creating an assembly hierarchy. There has also been advancement in the use of features to develop methods of assembly planning. Eng [7] developed a method to extract assembly planning from a CAD model using mating features. The goal was to create software that would generate a correct and practical assembly sequence from a CAD model based on already defined features. As well, van Holland [25] illustrated that assembly features can be a significant help in creating assembly plans. The connection relationships between parts of an assembly can be used to bridge the gap between the function of an assembly, and the geometry of an assembly. This information can be used to help with assembly analysis and planning, resulting in more efficient ways of analyzing an assembly and planning how an assembly is assembled.

Many if not most commercial CAD systems do not explicitly support model design or synthesis using assembly features, even though there are advantages to using assembly features in assembly design. Lee [10] presented an assembly synthesis method that uses screw theory for determining proper part constraint in an assembly. By using a predefined joint library and screw theory, Lee is able to determine optimal designs of beam based structures from assemblies that have already been created by an engineer. Yetis [29] and Lyu [12, 13] have focused on a systematic decomposition process during the conceptual design phase. In general, their intended purpose is to provide feedback about possible decompositions to the designer before the detailed design phase. Yetis decomposes a model into an assembly of multiple structural members. These members and their joints are evaluated for structural strength. While Lyu introduces continuing research that provides a method that considers the structural stiffness of the end model. The final model is then determined by joint location between components and its structural stiffness. Singh [22] developed a method that automates applying mating constraints by using a catalog of parts with embed assembly information. Parts are assembled together in a CAD system using ports. These ports are basic geometric entities that undergo mating constraints. Sambhoos [20] created a method for developing mating graphs from legacy CAD models in order to apply the information in assembly variant design. The developed mating graphs contain assembly feature that have direct, indirect or interfering relationships. Indirect assembly relationships are particularly important to assembly variant design.

An important part of assembly design is verifying that the assembly is valid and performing various analyses. To do this, Adams and Whitney [1] used screw theory to develop a method to analyze an assembly's constraint state using one or more assembly features. Their method uses twist and wrench matrices to represent the motion properties of kinematic joints. By uniting multiple twist matrices, contained in an assembly, the motion properties of an assembly can be represented. This method can be useful to determine if features are over-constrained, under-constrained, or properly constrained. Rachuri [19] developed a way to allow the exchange of information between modeling, analysis and planning systems by creating an integrated information model for assembly representation. They have defined assemblies based on their features, and desire that this development is to be integrated with various CAD and engineering analysis systems. Sung [24] used assembly features to generate disassembly sequences of assemblies. An octree representation method is used to define the assembly features. Along with developed algorithms, the defined assembly features are used to determine possible disassembly sequences for a given assembly model.

The above review, though not comprehensive, is representative of much assembly modeling literature. Two themes stand out: most of the work is directed at assembly sequence planning when the design is already fixed and not evolving; the various assembly feature definitions used require manual input into the ad-hoc attribute slots. The latter problem creates a disconnect between kinematic/structural simulation and design. Our assembly feature definition goes beyond geometry by capturing functional definition of generalizable part-part interfaces in the form of knowledge structures.

4 ASSEMBLY FEATURE TEMPLATE

In this paper we demonstrate the implementation of interactively defining user defined assembly features. Below is the template used to define an assembly feature, the complete details of which can be found in our previous paper [15]. The definition of an assembly feature contains slots for the following information:

- Face pairs that constitute the assembly feature (including face types and part IDs)
- Net kinematic DoFs at feature level
- Actual loads and directions

Each face pair needs to be defined with:

- Assembly Parameter Definition
 - Geometric - parameter defined by two geometric entities directly
 - Algebraic - parameter defined by other parameters
- Constraints/ Relations
 - Geometric - constraint between two geometric entities
 - Algebraic - constraint between parameters
- Kinematic Relations - DoFs & motion limits at the face pair level
- Structural Relations: load point, component directions & magnitude, time functions (at face pair level)

The slots in the above template will be populated for each assembly feature and used to create a library of multiple user defined assembly features.

5 ASSEMBLY FEATURE TUTOR

An interactive system can facilitate definition of assembly features. An assembly CAD file can be imported into the systems, and the user can be guided to define an assembly feature that can later be used in the process of assembly feature recognition. Each assembly feature can be expressed in terms of the components of the template described in the last section. By providing a system where the user is guided thru the process of defining assembly features, the user can define any assembly feature that they deem useful. Therefore, the user has ultimate control over what assembly features can be defined and recognized. Such a system has been implemented in C++ on Windows platform using ACIS geometry kernel, with HOOPS for rendering. Figure 4 shows a workflow schematic and a screenshot of the Assembly Feature Tutor system. Throughout the following sections, the steps of defining a round pin and slot assembly feature will be demonstrated using the geometry shown in Figure 4(b).

The Assembly Feature Tutor allows users to define their own features in progressive stages as shown by the bubbles in Figure 5. When complete, each feature is translated into an implementation neutral definition. An output file is created which can be stored in a library of assembly features. The libraries could be used in design by assembly features, feature recognition or any other related application. Details of the tutor are given in the following sections.

5.1 Find Contact Faces

When the Assembly Feature Tutor is started it stores a list of bodies and their faces, contained in the assembly, in their respective classes. As well, the user enters a name for the assembly feature they wish to define; in this circumstance, the name entered is "rd_pin_slot". Since all assembly features depend on their contact faces, the second step deals with finding these contact faces. The user has the option of choosing the faces manually or automatically. If the user chooses to pick contact pairs manually, the system checks to make sure that the faces chosen are on separate bodies and that they face each other. If the user decides to automatically find the contact faces in the assembly, they are able to input either an absolute tolerance (eg. 0.05 units) or a weighted value (from 0 to 1, with 1 being the tightest). The Tutor uses an algorithm called ACON, which was created by a PhD student at ASU [15]. Based on the users input, the contact faces can be automatically detected as shown in Figure 6. The faces automatically found or manually selected are then checked to verify that they have not previously been identified as contact pairs. They are then stored in the *AFFacePair* class and identified as contact faces. The individual faces and bodies are added to the *AFFace* and *AFBody* classes respectively if they have not previously been added. The *AFFace* and *AFBody* classes are used to

identify individual entities that are important to the assembly feature. Also, any appropriate geometric constraints, described in more detail later, are added based on the face pair type. Figure 6 shows that the contact faces were found using the automatic function using an absolute tolerance of 0.1 units. The faces found in contact are highlighted in the figure.

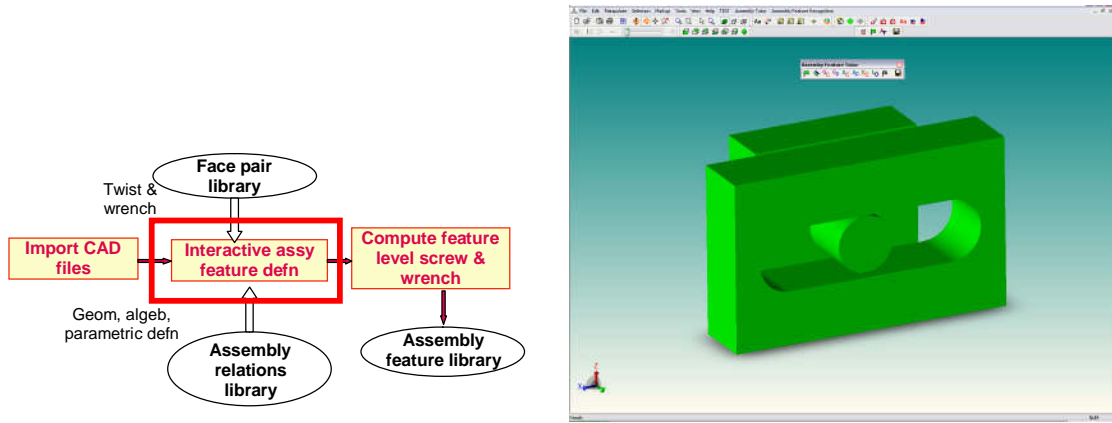


Fig. 4: Interactive system for assembly feature definition: workflow (left) user interface (right).

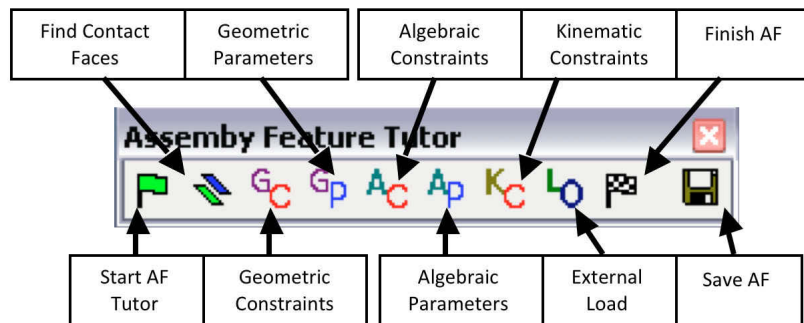


Fig. 5: Assembly feature tutor tasks.

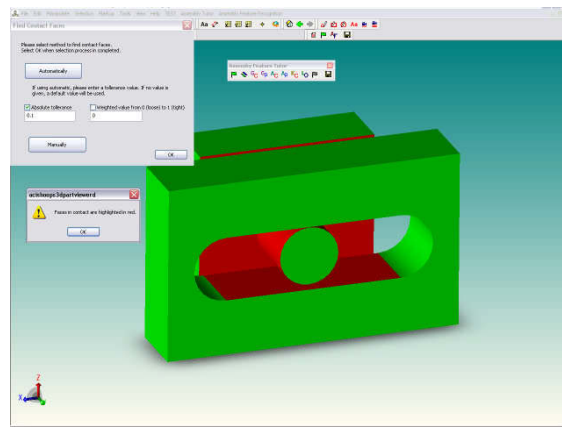


Fig. 6: Automatically found contact faces.

5.2 Geometric Constraints and Parameters

The next step in defining an assembly feature is to explicate its geometric constraints. This allows the user to pick two faces and choose a constraint from a drop down menu. Internally, the system checks to make sure the faces selected are on separate bodies, and that the types of faces selected also match with the constraint type. For example, for a tangent geometric constraint, the two faces selected need to be on separate bodies and one must be a planar face and the other a cylindrical or spherical face. The faces selected and the constraint type is then added to the *AFGeometricConstraints* class. As with the previous step, if the individual faces and their corresponding body are not already added to the *AFFace* and *AFBody* classes, they are added. Figure 7 shows this step using the round pin and slot feature. The faces chosen are the cylinder of the pin and the top planar face of the slot to define a tangent geometric constraint.

Geometric parameters are defined by selecting either one or two faces and defining a parameter name. For example, one may choose a spherical face and define a parameter of “diameter”; or two parallel planar faces may be selected to define a “width”. When the faces are selected and the parameter defined, the system adds the faces to the *AFGeometricParameter* class and the parameter name to the *AFParameter* class. And as with previous steps, the individual faces and bodies are added to their respective classes if not already added. Figure 8 demonstrates the defining of the parameter “pin_dia” for the example assembly feature.

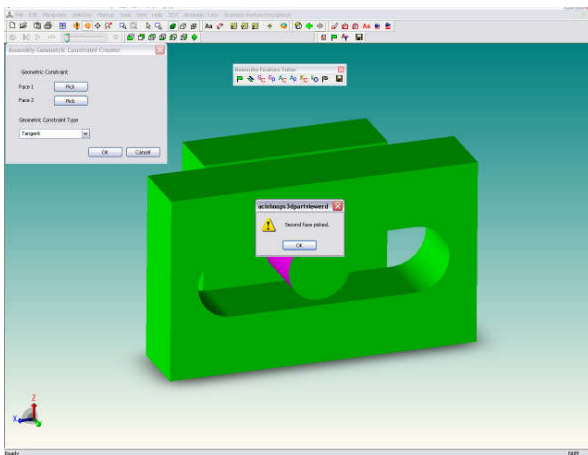


Fig. 7: Geometric constraint.

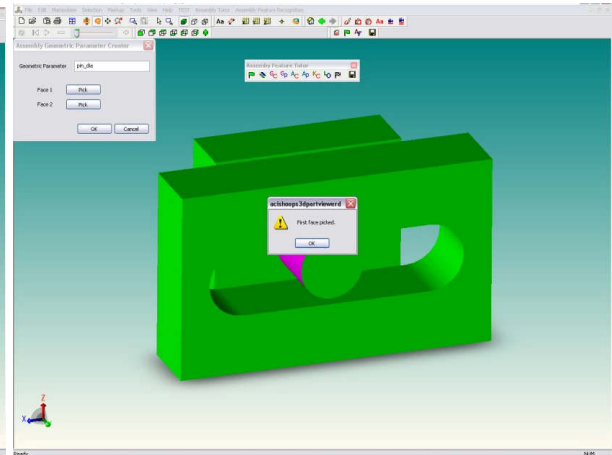


Fig. 8: Geometric parameter.

5.3 Algebraic Constraints and Parameters

Algebraic constraints allow the user to input 2 expressions and choose the algebraic relation between these expressions. The possible relations are \neq , $<$, \leq , $=$, $>$, \geq . Since algebraic constraints involve algebraic parameters, the algebraic parameters used must also be defined. The complete expression is then stored in the *AFAlgebraicConstraint* class. Algebraic parameters are defined using previously defined geometric parameters. The expression input by the user has the form of: algebraic constraint = GeometricParameter1 - (or +, *, /, etc) GeometricParameter2. This entire expression is then stored in the *AFAlgebraicParameter* class. With the previous assembly feature example, an algebraic constraint of “side_clearance>0” and an algebraic parameter of “side_clearance=slot_width-pin_dia” (figure 9) has been defined. These will later be used in the recognition process to verify that the recognized feature meets these pre-defined criteria.

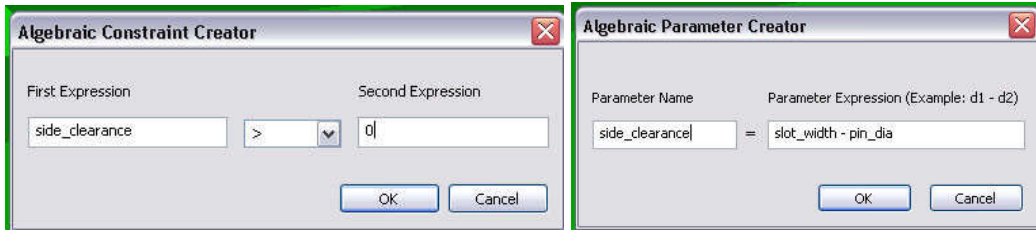


Fig. 9: Algebraic constraints and parameter for example assembly feature.

5.4 Kinematic Constraints

As discussed earlier, kinematic constraints are represented with twist matrices. When defining kinematic constraints in the Assembly Feature Tutor, the user is able to input the twist matrix manually or it can be extracted automatically. When a user is creating a custom assembly feature definition which involves multiple face pairs, the twist and wrench matrices for the assembly feature can be derived using Whitney algorithm [28]. A library of twist matrices has been created, based on contact face pairs, for the automatic extraction of kinematic constraints. They are as follows:

Plane - Plane	Cylinder - Cylinder	Sphere - Sphere	Plane - Cylinder
$T = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$T = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$
$W = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$	$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$	$W = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$

Tab. 1: Kinematic constraints library.

Using the above library with Whitney's algorithms, an assembly feature twist and wrench matrix can be derived. In looking at the plane - plane twist matrix, the first and second row represent translation in the x and y directions respectively, and the third row represents rotation about the z direction. It is important to note that the matrices in the library need to be aligned correctly with the assembly and with the global coordinates if possible in order to produce correct load representation of the assembly. Internally, the system stores the manually entered or automatically derived twist matrix for the entire assembly in the *AFKinematicRelation* class. As well, the face pair twist matrices are stored in the *AFFacePairKinematics* class. Figure 10 shows the output of kinematic constraints for the "rd_pin_slot" assembly feature definition. The movement allowed in this feature is translation along the x direction (along the slot) and rotation about the y axis (axis of the pin).

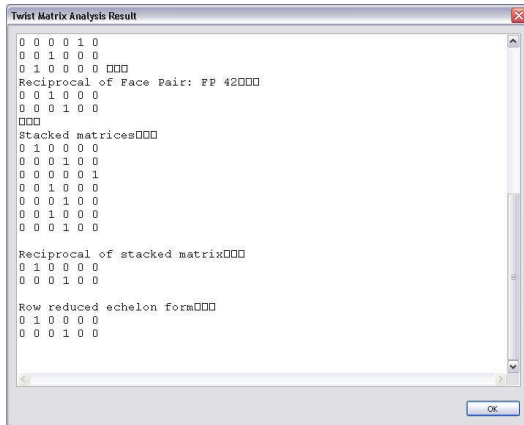


Fig. 10: Output of kinematic constraints.

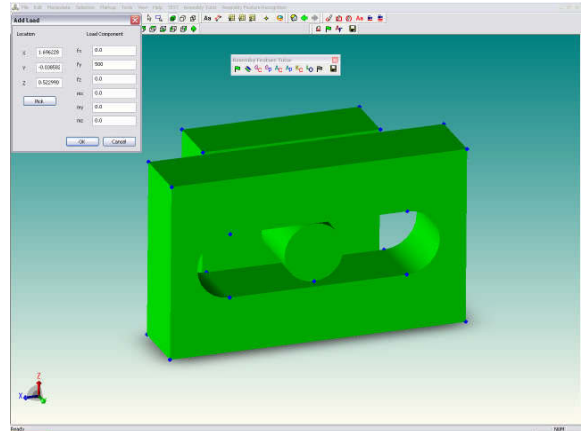


Fig. 11: External load information.

5.5 External Loads

This section of the Assembly Feature Tutor allows the user to add external loads to the assembly feature definition. The user is allowed to either enter the coordinates of the load, or pick a point on the model. Then they enter the load information. Figure 11 shows this being done on the example that has been presented. The load location and load magnitudes are stored in the *AFExternalLoad* class.

5.6 Finish and Save

At this point, all of the major assembly feature definition steps have been completed. The finish stage allows the user to view all of the information entered that defines the current assembly feature. The user is also able to delete any parts of the assembly feature that are not necessary. During this step, the system loops through each individual face that is stored in the *AFFace* class and creates face associations of convex or concave between each other. These associations are stored in the *AFFaceRelation* class. Figure 12 is the tree display for the example assembly feature definition. The last step is saving the assembly feature. At this point the user chooses which assembly feature definition they wish to save, gives it a file name by which it is saved. The system converts the information saved in the various classes to an N-Rep text file that can be used later in the assembly feature recognition process. This file is written in an express-like language and a sample is shown in figure 13.

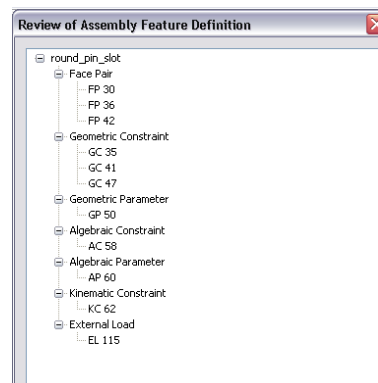


Fig. 12: Summary of defined assembly feature.

```

#8 = ASU_FACE_REFERENCE ('face1', 'planar', '$');
#21 = ASU_FACE_REFERENCE ('face2', 'planar', '$');
#3 = ASU_FACE_REFERENCE ('face3', 'planar', '$');
#22 = ASU_FACE_REFERENCE ('face4', 'cylindrical', '$');
#12 = ASU_FACE_REFERENCE ('face5', 'planar', '$');
#23 = ASU_FACE_REFERENCE ('face6', 'planar', '$');
#11 = ASU_FACE_REFERENCE ('face7', 'cylindrical', '$');
#10 = ASU_FACE_REFERENCE ('face8', 'cylindrical', '$');
#5 = ASU_FACE_REFERENCE ('face9', 'planar', '$');

#30 = ASU_FACE_PAIR ('pair30', (#8, #21), (#13, #24));
#36 = ASU_FACE_PAIR ('pair36', (#3, #22), (#13, #24));
#42 = ASU_FACE_PAIR ('pair42', (#12, #22), (#13, #24));

#191 = ASU_PARAMETER ('pin_dia', $);
#192 = ASU_CONSTRAINT ('pin_dia', #22, #191);
#48 = ASU_ASSEMBLY_GEOMETRIC_PARAMETER (#192);
#193 = ASU_PARAMETER ('pin_length', $);
#194 = ASU_CONSTRAINT ('pin_length', (#21, #23), #193);
#52 = ASU_ASSEMBLY_GEOMETRIC_PARAMETER (#194);
#195 = ASU_PARAMETER ('slot_length', $);
#196 = ASU_CONSTRAINT ('slot_length', (#11, #10), #195);
#59 = ASU_ASSEMBLY_GEOMETRIC_PARAMETER (#196);
#197 = ASU_PARAMETER ('slot_width', $);
#198 = ASU_CONSTRAINT ('slot_width', (#3, #12), #197);
#66 = ASU_ASSEMBLY_GEOMETRIC_PARAMETER (#198);

```

Fig. 13: Assembly feature N-Rep file.

6 ASSEMBLY FEATURE RECOGNITION ALGORITHMS

The motivation for our project comes from the objective to reverse engineer, re-engineer and manufacture replacement parts in legacy system in order to prolong their lives. In many instances, design drawings or CAD models for such parts are not available and one must extract information from failed specimens. It is not enough to just extract geometric information; we also need material, surface, manufacturing and functional information. The re-engineered part must fit into the legacy system and perform the intended function of the replacement part. Although functions can be varied, the two primary functions in mechanical assemblies are structural and kinematic. The assembly feature model presented is aimed at providing form, fit and function specifications for replacement parts.

Because of its advantages, the assembly feature recognition process chosen to implement is the second method presented in our previous paper [15]. Throughout this section, an example will be used to demonstrate how the system works. The pinned slide in figure 14 will be used to extract some sample assembly features. The implementation steps, for assembly feature recognition, are as follows:

- The system reads the input N-Rep files, which contain assembly feature definitions
- Modified face adjacency graphs of the N-Rep files are created and pertinent information from N-Rep files such as parameters and constraints are stored.
- Then the assembly model is loaded by selecting a part to be used to find assembly features.
- Modified face adjacent graph/s of input assembly are created.
- Graph matching is conducted.
- Constraints defined in the assembly feature definitions are checked.
- The system then computes assembly level load directions.
- A tree showing the found assembly features and their parameter values is shown.
- If wanted, the user can output an assembly feature instance file.

The assembly feature recognition module allows the user to input as many or as little N-Rep files as they wish. Ideally, a library of assembly feature definitions would be previously created, using the Assembly Feature Tutor, for use at any time. As little as one N-Rep file can be loaded. For the example of the slide, two assembly features are of importance, therefore two N-Rep files have been loaded; “rd_pin_slot” and “concave_pin_thru_hole”. Once the N-Rep files are loaded, the system reads the definition and creates a modified face adjacency graph of each N-Rep file inputted. This data structure is similar to face adjacency graphs except face connection arcs are included in the graphs. The nodes represent the faces in the definition. The nodes contain an id number and face geometry type. The arcs represent adjacency definitions between faces. These are concave, convex, or contact. The modified face adjacency graphs are created using linked lists. Next, the user selects the part that is of interest in finding its assembly features. This is done so that the system focuses on a particular part of the model. In this step, the contact faces associated with the chosen part are located. This is shown in figure 15 for the sample assembly. Also a list of contact faces is found in Table 2.

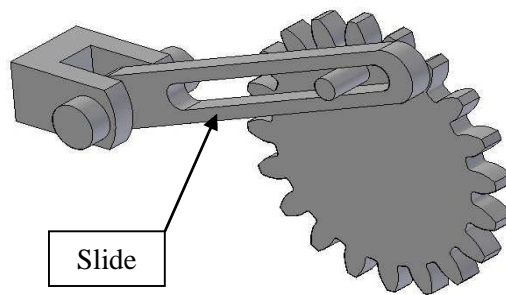


Fig. 14: Sample assembly.

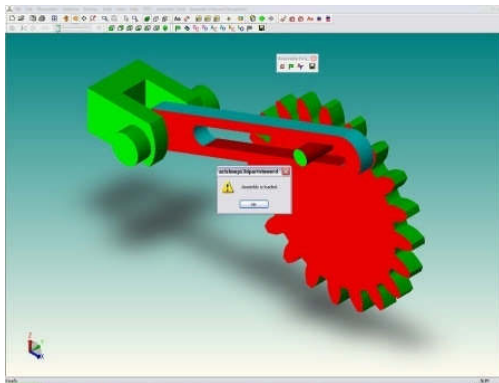


Fig. 15: Loaded assembly.

Face Number	Face Type	Contact Face	Associated Assembly Feature
3	Cylinder	4	round pin through hole
4	Cylinder	3	round pin through hole
8	Cylinder	7 & 9	round pin and slot
7	Planar	8	round pin and slot
9	Planar	8	round pin and slot
5	Planar	6	round pin and slot
6	Planar	5	round pin and slot

Tab. 2: Contact faces and their associated assembly feature.

Once the contact faces are located, a modified face adjacency graph of the input assembly, or Brep model, is created. The nodes and arcs carry the same information as the ones created for the N-Rep modified FAGs. For this step, nodes are created for the faces found in contact and their immediate adjacent faces only. And the arcs are created in the same manner as above. At this point, nodes that contain faces that point to the same geometry as another node are removed, so that there is only one

node per geometric face in the Brep model. In order to decrease search space, separate graphs are created for nodes that do not have any arcs connecting to other groups of nodes, and the graphs are classified based on the face types contained in the graph/s. A graph that contains only planar faces is classified as “planar”, and graphs that contain cylinder or spherical faces are classified as “cylindrical” and “spherical” respectively. The modified FAG for the slide is found in figure 16; for simplicity, only the pertinent nodes and arch are presented in the graph. It can be noted that there is a single graph since all of the contact faces and their immediate adjacent faces are all connected by either a face adjacency or by contact.

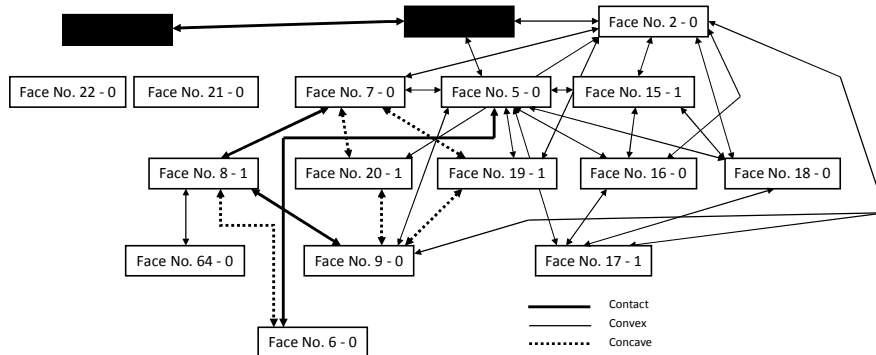


Fig. 16: Modified FAG for slide example.

Once the graphs are created from the inputted N-Rep files and the input Brep assembly model, graph searching techniques can be used to find subgraphs within the Brep graphs that match those created from the N-Rep files. There are many techniques that can be used in this stage. The implemented system starts graph searching at nodes which contain contact arcs and the same classification as the N-Rep graph. For example, for an assembly feature with spherical faces, the search starts at the node that represents one of the spherical faces in contact. Then the search continues outward until all of the N-Rep nodes in the assembly feature definition are found and match in the Brep graph/s. In figure 16, the face numbers 5, 6, 7, 8, and 9 make up the contact faces for the “rd_pin_slot” definition. And faces 3 and 4 are the contact faces for the “concave_pin_thru_hole” definition. This graph matching process continues with each appropriate N-Rep definition until an exhaustive search of all possible assembly features has been conducted. Once the graph matching is complete, the constraints stored from the N-Rep files are checked with the found assembly features in the Brep graph/s. If the constraints are met, the stored parameters are evaluated in the Brep model. Also the assembly level load direction, represented as a twist matrix, is evaluated using Whitney’s algorithms in conjunction with the found assembly feature’s twist matrices. All of the recognized assembly features and their information are then shown in a tree (figure 17). Finally, the user has the option of saving an output assembly feature instance file, which shows the important information about the assembly features and assembly level load directions.

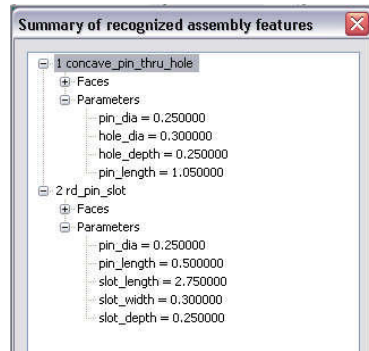


Fig. 17: Found assembly features and their parameters.

7 DISCUSSION

The motivation of this research is to implement a system that interactively defines assembly features and uses these definitions to locate assembly features for use in Reverse Engineering of legacy systems. This allows a user to define their own assembly features that are relevant to them and their own work. In doing so, the user can create a library of assembly features to define any type of assembly feature desired. This library can be used in many different ways. For Reverse Engineering of legacy systems, it is important to verify that the part being redesigned still complies with form, fit, and function of the original part. This means that the new part has the same form (possibly size, generic shape, weight, balance, etc), that the redesigned part still fits in the assembly, and that this part also performs the same function as the previous part. In summary, there should not be any difference between the previous and redesigned assembly.

Assembly features explicitly help to verify that a redesigned part fits and functions identically as the previous part. Fit has to do with the ability of a part to physically interconnect with other parts in an assembly. This relates the associativity of a part to other parts. This means that redesigned parts must have connecting faces that are in the same locations and have the same dimensions as the previous part. For the slide that is in figure 18, the two features of importance are the rounded end slot and the hole. For this reason, when identifying these features, the parameters of the hole diameter, hole depth, slot length, width and depth have been evaluated so that if this part is to be redesigned, these identified features will still have the same dimensions and fit in place of the old part.

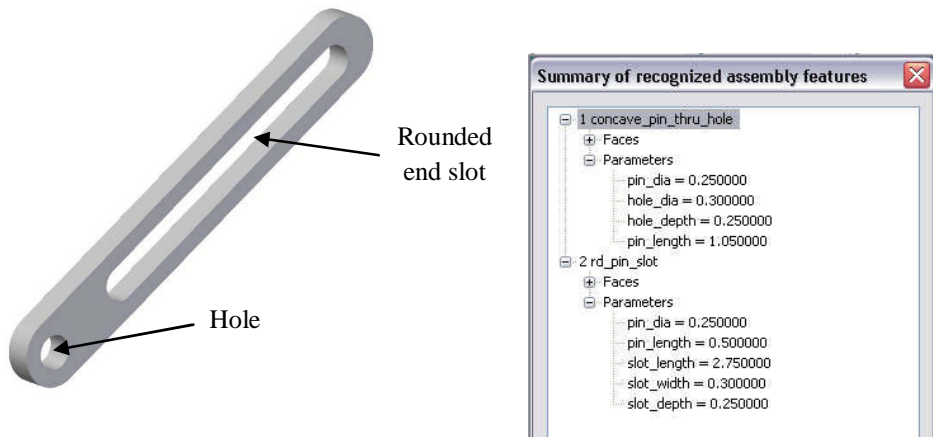


Fig. 18: Slide and found assembly features and their parameters.

Similarly, function is equally important factor in the redesign of a part for an assembly. Function is the action that a part is designed to perform in an assembly. This is extracted from assembly features by determining motion limits and possible load directions. Figure 19 shows what motion limits and load directions of the slide at the assembly level and at each assembly feature. It can be seen that at the assembly level, the slide is fully constrained and has no movements allowed. This information can be used to verify that a redesigned part contains the same motion limits and load directions.

This project has produced a system that has been implemented to generate user defined assembly features. It has also show an implementation of assembly feature recognition of these user defined assembly features, and has shown how it can be applied to Reverse Engineering of legacy parts.

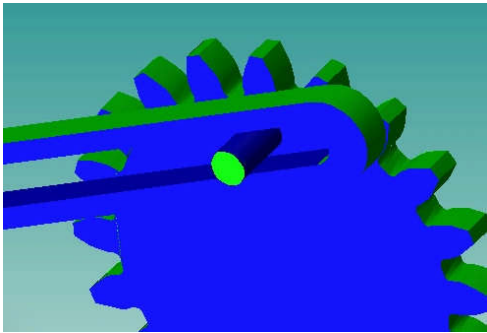
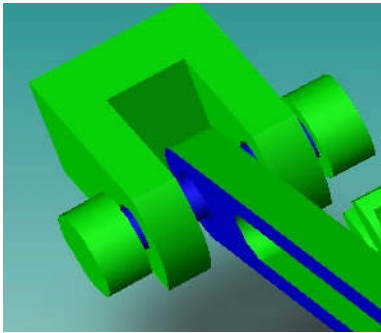
		Assembly Level
Twist Matrix 0 1 0 0 0 0 0 0 0 1 0 0	Twist Matrix 0 1 0 0 0 0	Twist Matrix Empty
Wrench Matrix 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1	Wrench Matrix 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1	Wrench Matrix 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1

Fig. 19: Twist and wrench matrices of sample assembly.

ACKNOWLEDGEMENTS

The work described here was supported by the US Army Research Office, Grant#W911NF-04-R-0005.

REFERENCES

- [1] Adams, J. D.; Whitney, D. E.: Application of Screw Theory to Constraint Analysis of Mechanical Assemblies Joined by Features, Journal of Mechanical Design, 123, 2004, 26.
- [2] Bordegoni, M.; Cugini, U.: Feature-based assembly design: concepts and design environment, DETC97/CIE- 4266.
- [3] Brunett, G.; Golob, B.: A feature-based approach towards an integrated product model including conceptual design information, Computer-Aided Design, 32, 2000, 877-887.

- [4] Butterfield, J.; Crosby, S.; Curran, R.; Price, M.; Armstrong C. G.; Raghunathan S.; McAleenan D.; Gibson C.: Optimization of Aircraft Fuselage Assembly Process Using Digital Manufacturing, *Journal of Computing and Information Science in Engineering*, 7, 2007, 269.
- [5] De Fazio, T. L.: A prototype of feature-based design for assembly, in B. Ravani, ed., 'ASME Advances in Design Automation 1990', Chicago, Illinois, USA, 9-16.
- [6] Eng, T.; Ling, Z.; Olson, W.; McLean, C.: Feature-based assembly modeling and sequence generation, *Computer & Industrial Engineering*, 36, 1999, 17-33.
- [7] Gupta, S. K.; Paredis, C. J. J.; Sinha, R.: Intelligent assembly modeling and simulation, *Assembly Automation*, 21(3), 2001, 215-235.
- [8] Jayaram, U.; Kim, Y.; Jayaram, S.; Jandhyala, V. K.; Mitsui, T.: Reorganizing CAD Assembly Models (as-Designed) for Manufacturing Simulations and Planning (as-Built), *Journal of Computing and Information Science in Engineering*, 4, 2004, 98.
- [9] Lee, B.; Saitou, K.: Three-Dimensional Assembly Synthesis for Robust Dimensional Integrity Based on Screw Theory, *L. Mech. Des.* 128, 2006, 243.
- [10] Liang V-C; Paredis C.: A Port Ontology For Automated Model Composition, Winter Simulation Conference, 2003.
- [11] Lyu, N; Saitou, K: Decomposition-Based Assembly Synthesis for Structural Stiffness, *Journal of Mechanical Design*, 125, 2003, 452.
- [12] Lyu, N; Saitou, K: Decomposition-Based Assembly Synthesis of Space Frame Structures Using Joint Library, *Journal of Mechanical Design*, 128, 2006, 57.
- [13] Murshed, M.: Development of TechSpec model for legacy systems engineering, MS Thesis, ASU, Tempe, AZ, 2008.
- [14] Murshed, M.; Dixon, A.; Shah, J.: Neutral definition and recognition of assembly features for legacy systems reverse engineering, ASME DETC2009-86793, San Diego, Sept. 2009.
- [15] Murshed, M.; Shah, J.; Jagasivami V.; Wasfy A.; Hioslop D.: OAM+: An assembly data model for legacy systems engineering, ASME DETC2007-35723, Las Vegas, Sep 2007.
- [16] Noort, A.; Hoek, G. F. M.; Bronsvort, W. F.: Integrating part and assembly modeling, *Computer-Aided Design*, 34(12), 2002, 899-912.
- [17] Pahl, G.; Beitz, W.: *Engineering Design: A Systematic Approach*. Springer, Springer-Verlag, 2007.
- [18] Rachuri, S.; Han, Y.; Foufou, S.; Feng, S. C.; Roy, U.; Wang, F.; Sriram, R. D.; Lyons, K. W.: A Model for Capturing Product Assembly Information, *Journal of Computing and Information Science in Engineering*, 6, 2006, 11.
- [19] Sambhoos, K.; Koc, B.; Nagi, R.: Extracting Assembly Mating Graphs for Assembly Variant Design, *Journal of Computing and Information Science in Engineering*, 9, 2009, 034501.
- [20] Shah, J.; Mantyla, M.: *Parametric and feature based CAD/CAM*, John Wiley, NY, 1995.
- [21] Singh, P.; Bettig, B.: Port-Compatibility and Connectivity Based Assembly Design, *Journal of Computing and Information Science in Engineering*, 4, 2004, 197.
- [22] Sodhi, R.; Turner, J. U.: Representing tolerance and assembly information in a feature-based design environment, in G. A. Gabriele, ed., 'ASME Design Automation Conference', Vol. DE-Vol. 32-1, Miami, Florida, USA, 1991, 101-108.
- [23] Sung, R. C. W.; Corney, J. R.; Clark, D. E. R.: Automatic Assembly Feature Recognition and Disassembly Sequence Generation, *Journal of Computing and Information Science in Engineering*, 1, 2001, 291.
- [24] Van Holland, W; Bronsvort, W. F.: Assembly features in modeling and planning, *Robotics and Computer Integrated Manufacturing* 16, 1999, 277-294.
- [25] Wang, W.; Chen, G.; Lin, Z.; Lai, X.: Automated Hierarchical Assembly System Construction in Automobile Body Assembly Planning, *Journal of Mechanical Design*, 127, 2005, 347.
- [26] Whithead, T: *Design of instruments & accurate measurements*, Dover NY, 1954.
- [27] Whitney, D. E.: *Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development*, Oxford University Press, 2004.
- [28] Yetis, F. A.; Saitou, K.: Decomposition-Based Assembly Synthesis Based on Structural Considerations, *Journal of Mechanical Design*, 124, 2002, 593.