American Transactions on
Engineering & Applied Sciences

http://TuEngr.com/ATEAS

Characterization of a 2D Geometry Using C^{++} Interface

Vijay K. Goyal [a*], Ricky Valentin [a], Michael J. Cruz [a], Neit J. Nieves [a]

[a] *Department of Mechanical Engineering, University of Puerto Rico at Mayagüez, PR 00680 USA*

ARTICLEINFO

Article history:
Received 14 May 2013
Received in revised form
02 December 2013
Accepted 12 December 2013
Available online
13 December 2013

Keywords:
CAD;
Solid Modeling;
parametric, dimensions;
constraints

ABSTRACT

Problem statement: Usually, we want to represent the final computer aided drawings in a detailed characterization of the design geometry. In some cases, the final design is corrupted with additional geometry parameters, which are not part of the problem, and lack of dimensions. We need an automatic approach to resolve this issue. **Approach:** Here, we develop a toolkit, which integrates SIEMENS NX8 and C++, to improve the characterization process for 2D geometries. **Results:** We applied the tool to several different 2D cross-sections and proved that we were able to removal the unwanted parts from the geometry and apply the proper dimensions and constraints to the geometry. **Conclusion:** This makes the 2D geometry characterization process faster and user-friendly.

1. Introduction

Many tools are currently available for assisting an engineer, or designer, in the dimensioning or constraint characterization process of a solid model or 2D drawing. These systems and tools each advance the design process and help make dimensioning or constraint application more useful within the modeling software environment in use but may require significant and time-consuming user input to achieve the desired results. To characterize a design within Solid Modeling software packages the designer will have to use several tools including parametric and constraint-based

*Corresponding author (V. Goyal), Tel.: 1-787-832-4040; E-mail: vijay.goyal@upr.edu.

modeling and a proper dimensioning scheme for the accurate description of said design. Current solid modeling software requires manual and time consuming manipulation of the geometry for the correct implementation of dimensions and constraints for future development and referencing of the design. We need ~~to~~ that the design's integrity will remain after any modifications are done to it, while avoiding over constraining the geometry. Also, if the drawings were to become corrupted, or if the user tried to extract lines from a section view of a 3D model, or if the user were to import the geometry into the CAD software using a scanning tool, unneeded geometry might become present, which the user must carefully select and manually remove. This process will take a tremendous amount of time if the design consists of hundreds of objects. In addition, the dimensioning and constraint application will be difficult to accomplish, usually requiring multiple steps in the process.

Hence, we created a tool using NX's open architecture in conjunction with Visual C++ to characterize 2D geometries, automatically remove unwanted geometry parts, and apply dimensions and geometric constraints. With this tool, the designer will reduce the characterization time.

2. Background and Motivation

For any successful engineering process, drawings providing a detailed description of ideas and specifications are essential. These drawings must accurately depict a detailed characterization of the design geometry to be successfully implemented in the design and manufacturing process. In some cases, a design might become corrupted or if importing the sketches to NX8 utilizing a scanning tool, errors might be encountered resulting in additional lines, lack of dimensions and geometrical constraining. In such cases, we would need to remove manually the unwanted items before dimensioning and constraining can added to the model. Using SIEMENS NX8 and their open architecture NX OPEN, we developed a toolkit using C++, with the purpose of improving the characterization process of dimensional geometry, in hopes of making the characterization a faster and more user-friendly one. We can use this toolkit to add constraint and dimensioning to unconstrained sketches created out of the sketcher environment, shall the user decide to do so, regardless of whether or not the sketch includes any unwanted geometry.

For the creation and implementation of this tool Microsoft C++ was utilized in conjunction to

SIEMENS NX8. Referencing functions available to the user within the NX OPEN library, we wrote a code and then compiled it into a dynamic-link library (.dll) file. With an OPEN NX8 session and with the geometry of interest in hand, we may execute the toolkit using CTRL+U to call the Execute User Function dialog and navigate to the .dll file. For proof of concept, completely unconstrained geometry created for the tests, this is done outside the Sketch environment. With the tool menu already on screen, the user proceeds to select the adequate lines and dimensioning and constraining will be added to the lines. We need to be careful when selecting the "Cancel" option once the user has selected each type of line. The final product will be a new dimensioned and constrained sketch free of extra lines.

2.1 Solid Modeling

Solid modeling software creates a virtual representation of components for machine design, analysis and manufacturing. Figure 1 shows ~~the many~~several ways to describe a solid model. These methods allow the designer to, accurately, describe a design. Our work is part of the modeling and analysis of 2D views taken from three-dimensional solid models.

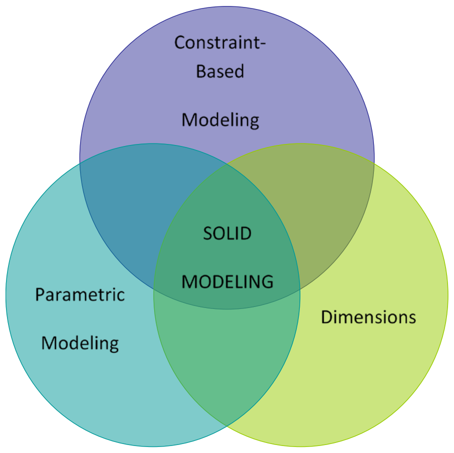
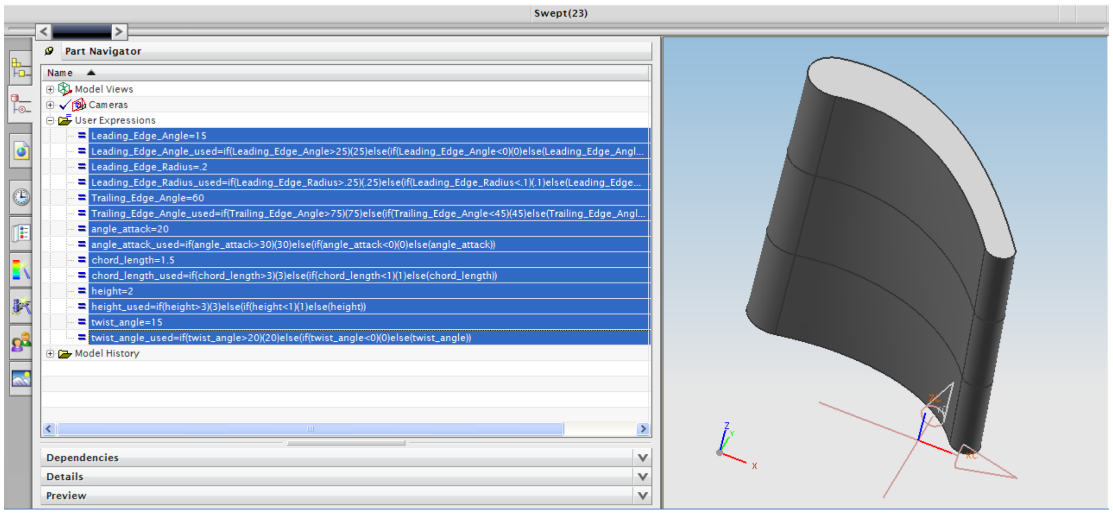


Figure 1: Simple Representation of Solid Modeling.

2.2 Parametric Modeling

Parametric modeling is a characterization process that uses parameters to define a model. The parameters may be modified later, and the model will update to reflect the modification. Figure 2 shows an airfoil model geometry constrained by user-defined parameters. These parameters are highlighted on the left. Typically, there is a relationship between parts, assemblies, and drawings. A part consists of multiple features, and an assembly consists of multiple parts. We can make the drawings from either parts or assemblies. We can visualize the constraint-based geometry as a general framework from which a parameterized system is a special case. Parametric design specification is one of the possible alternatives available within a constraint-based representation [1].

Maarten and Van [2] discussed the creation and modification of parametric solid models by graphical interaction. Their primary concern was to define relations ~~between~~ among geometric properties of constructive solid geometry primitive. The system they described is capable of changing dimensions and position of a geometric model as a whole, without changing the object description. The system cannot handle user-defined constraints and does not address the removal of any excess geometry, restricting the editing and characterization process.



```
Leading_Edge_Angle=15
Leading_Edge_Angle_used=if(Leading_Edge_Angle>25)(25)else(if(Leading_Edge_Angle<0)(0)else(Leading_Edge_Angl...
Leading_Edge_Radius=.2
Leading_Edge_Radius_used=if(Leading_Edge_Radius>.25)(.25)else(if(Leading_Edge_Radius<.1)(.1)else(Leading_Edge...
Trailing_Edge_Angle=60
Trailing_Edge_Angle_used=if(Trailing_Edge_Angle>75)(75)else(if(Trailing_Edge_Angle<45)(45)else(Trailing_Edge_Angl...
angle_attack=20
angle_attack_used=if(angle_attack>30)(30)else(if(angle_attack<0)(0)else(angle_attack))
chord_length=1.5
chord_length_used=if(chord_length>3)(3)else(if(chord_length<1)(1)else(chord_length))
height=2
height_used=if(height>3)(3)else(if(height<1)(1)else(height))
twist_angle=15
twist_angle_used=if(twist_angle>20)(20)else(if(twist_angle<0)(0)else(twist_angle))
```

Figure 2: Parameterized airfoil geometry with list of parameters displayed.

2.3 Constraint-Based Modeling

Constraint-based geometry is a technique by which we may define or constrain an arbitrary geometry using its dimensions, as shown in Figure 3. In constraint-based geometry, a number of characteristic points in a three-dimensional space determines an object. Instead of using the geometry to define the dimensions, constraint-based geometry uses the dimensions to define its geometry. We use dimensions to relate geometrical elements such as points, lines, arcs or circles. We may graphically represent these relations in the design's drawings using dimension parameters. Dimensioning and constraint-based geometry help the designer in the definition of the object and the use of geometrical dimensions as part of design specifications. Our work allows a designer to define constraints to selected objects within 2D geometry.

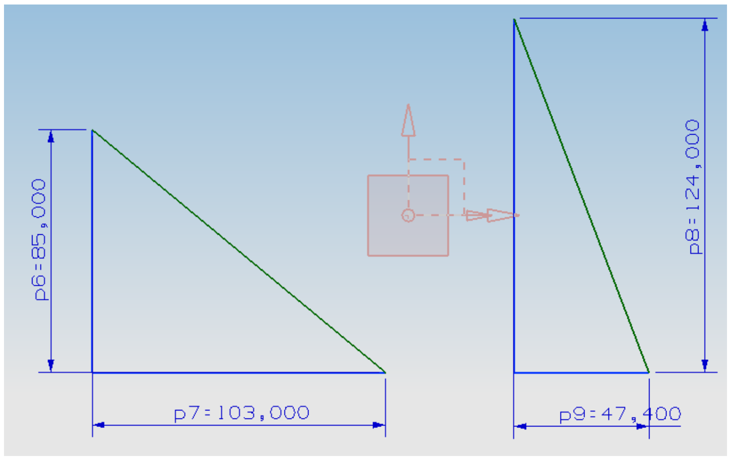


Figure 3: Triangular geometry constrained by its dimensions.

Light and Gossard ~~et al~~ [3] discuss a constraint-based method for modification of geometric models. They present a procedure for minimizing computational effort in the solution of the constraints and a procedure for detecting invalid dimensioning schemes. Suzuki et al [4] discuss the importance of geometric constraints and reasoning in CAD systems. One characteristic of design is constraint solving, an essential portion of these constraints being geometrical. They present a consistent framework for representing geometric models and constraints, and a geometric reasoning mechanism to solve those constraints. Their implementation is limited to the 2D case and they use constraint propagation to determine the parameter values. They clearly state that their method can only make geometrical changes to a fixed topology.

Pérez [5] proposed a methodology to be implemented in a three dimensional constraint-based finite element modeler to allow a designer to interactively construct a geometric model by dimensional changes that are propagated to the finite element model. The system also allowed for optimization analysis without requiring explicit parametric definitions from the user. Pérez [5] validated the system by conducting experiments and showed a significant saving time when compared to previously available systems.

All of these systems propose mathematical methods for constraint or dimension calculation and management but do not provide an actual application to use in an existing drawing or an application for removal of geometries that we do not need.

2.4 Dimensioning

The purpose of dimensioning engineering design drawings is to help the designer understand the spatial arrangement of the objects' geometry, as shown in Figure 4. In order to provide a manufacturing-ready part, we need to ensure correct, precise and complete dimension for size and position. There is more than one way to dimension single parts [1]. We dimension each component individually. We may define a part as a collection of geometric elements having a closed topology. Two parts do not share geometric elements and can be isolated without changing their individual characteristics [1]. This work allows the user to apply dimensions to the desired parts of the 2D geometry, as shown in Figure 4.

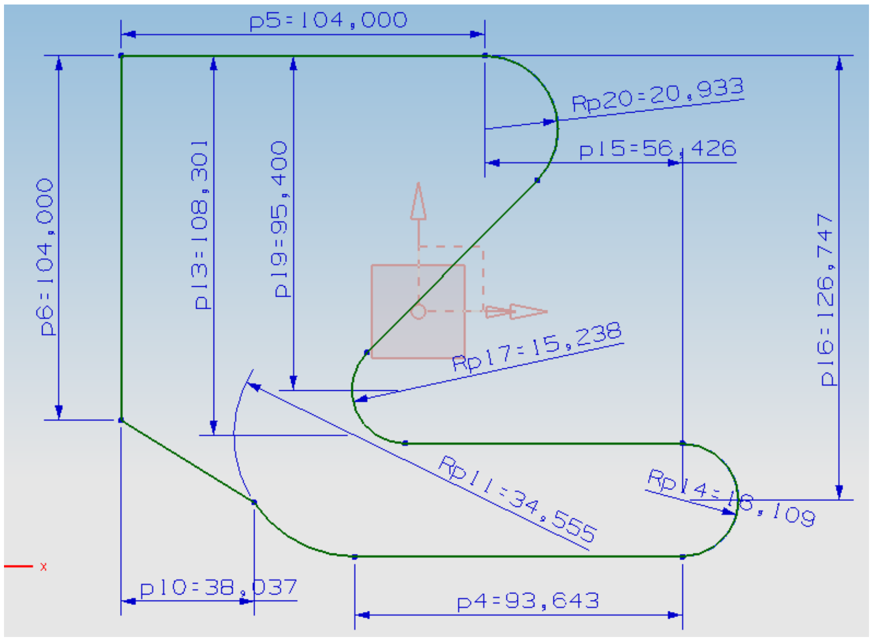


Figure 4: Example Dimensioning Scheme

Yuen et al [6] developed a method to generate automatically dimensions based on boundary representations of solid models expressed as linear and angular dimensions. This solid modeling system can generate an adequate dimensioning layout for a defined solid. Light and Gossard [3] presented a procedure that represents a geometry by a set of dimensions which were~~. They~~ used as constraints limiting the locations of the characteristic points of the object. Aldefield [7] represented a geometric model by a rule-based system for propagating constraint information on geometric structures [8].

Jaramillo [9] presented a methodology for the automatic dimensioning and use of tolerances for 2D geometry. Dones [1] presented an automatic generation of dimensioning schemes using constraint-based geometry. They represented dimensions as dimensional constraint equations, in terms of the characteristic points defining the geometry and dimension parameters. Dones [1] determined possible dimensioning schemes using a modification of the constraint management theory as presented by Serrano [10,11]. Pabón [8] proposed an automatic dimensioning layout methodology to help any CAD system with automatic dimensioning tools to locate properly the generated dimensions on the drawing layout using an intelligent rule based system.

These systems represent mathematical methods for constraint or dimension calculation and management but do not provide an application to use in an existing drawing or an application for removal of unwanted objects from the geometry.

3. Code

This work focuses on developing a toolkit to help the 2D cross-sectional analysis and characterization process. We used Microsoft Visual C++ software [12,13] as the main programming language with applicable functions and programming libraries in order to interact with SIEMENS NX8 [14–16] solid modeling software and analyze a 2D cross-section. The final output will be a 2D drawing free of unneeded geometrical parameters and a fully dimensioned and constrained drawing.

For the purpose of this work, we used the *uf* libraries [17]. These libraries allow the interactions between C++ code, our toolkit and SIEMENS NX8. The program starts by initializing the "Open C API Environment" a command that enables the SIEMENS NX8 capability of being run by an external C++ code. The program runs the six unique functions that we developed. The functions read the geometry's characteristic points in the plane and with that information; they calculate which objects are parts of the "desired" geometry and decide which ones remove. The functions will also create dimensioning and constraining information for the remaining objects. The output of this algorithm will be a geometry free of unwanted objects, with the desired parts dimensioned, and constrained. This is the major contribution of this work.

Figure 5 presents a problem approach flow chart. We load the 2D drawing into the

SIEMENS NX8 environment and then start the toolkit within the SIEMENS NX8. By analyzing and comparing the coordinates of the points that describe geometry and using position tolerances, we will be able to determine if the objects in the drawing are part of an enclosed area, which describes a sectional view. This will allow the toolkit to segregate non-descriptive objects and remove them from the drawing and presenting a clean 2D section for the user to analyze. After the user selects the parts he wants to dimension and constrain, the toolkit will automatically complete the job.

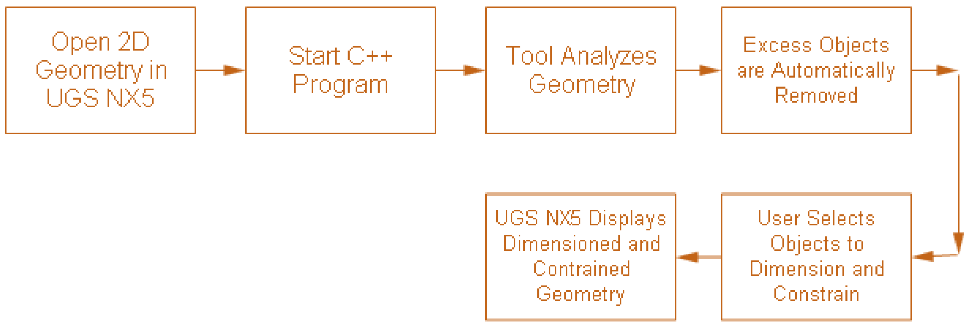

Open 2D Geometry in UGS NX5
Start C++ Program
Tool Analyzes Geometry
Excess Objects are Automatically Removed
User Selects Objects to Dimension and Constrain
UGS NX5 Displays Dimensioned and Contrained Geometry

Figure 5: Problem Approach Flow Chart.

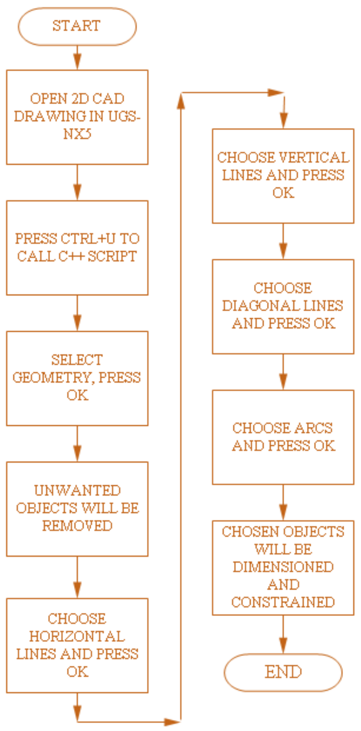

START
OPEN 2D CAD DRAWING IN UGS-NX5
PRESS CTRL+U TO CALL C++ SCRIPT
SELECT GEOMETRY, PRESS OK
UNWANTED OBJECTS WILL BE REMOVED
CHOOSE HORIZONTAL LINES AND PRESS OK
CHOOSE VERTICAL LINES AND PRESS OK
CHOOSE DIAGONAL LINES AND PRESS OK
CHOOSE ARCS AND PRESS OK
CHOSEN OBJECTS WILL BE DIMENSIONED AND CONSTRAINED
END

Figure 6: General SIEMENS NX8 and C++ Code Interaction

Figure 6 shows the interaction between SIEMENS NX8 and C++. Once the 2D geometry is loaded into SIEMENS NX8, the user presses *CTRL+u* to open the external code. The user will select the C++ code from its location in the computer's hard drive and the program will run automatically within the SIEMENS NX8 environment. The user selects all the geometry in the drawing and hits *ok*, and the program will remove the objects that are not part of the intended 2D geometry. This is powerful, especially, when trying to import scanned drawings. After the geometry is "cleaned-up", the users sees the individual line select dialog boxes. The program will then display the dimensions and create a geometric and dimensional constraint of the selected objects. We divided the code into several functions called from the main program. Figure 7 shows the tool function sequence.

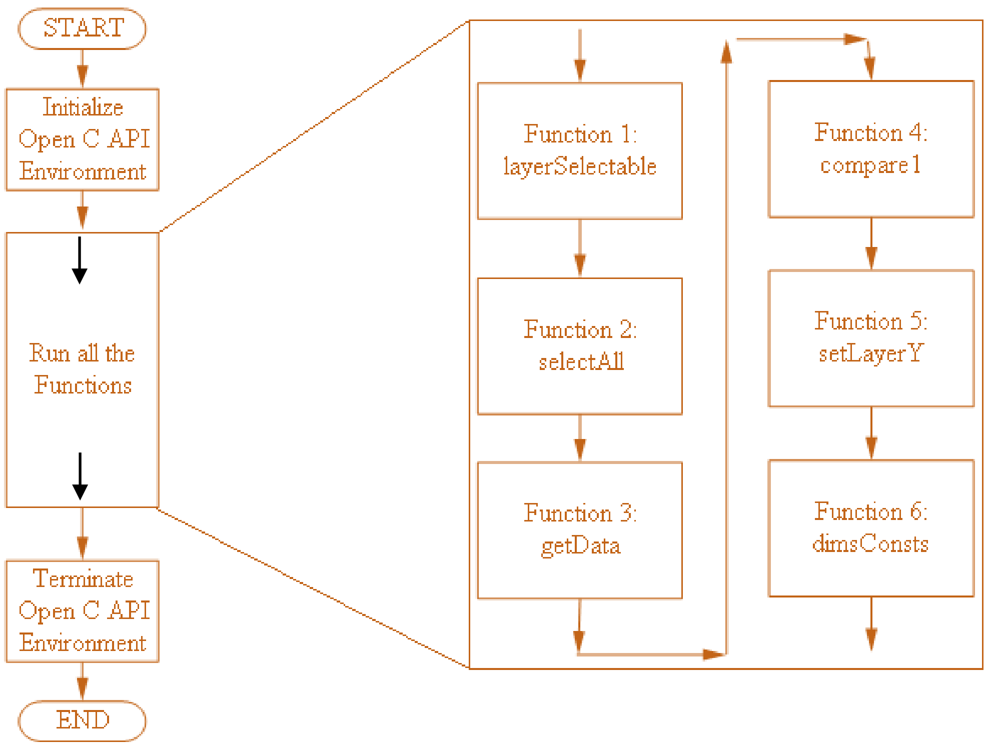


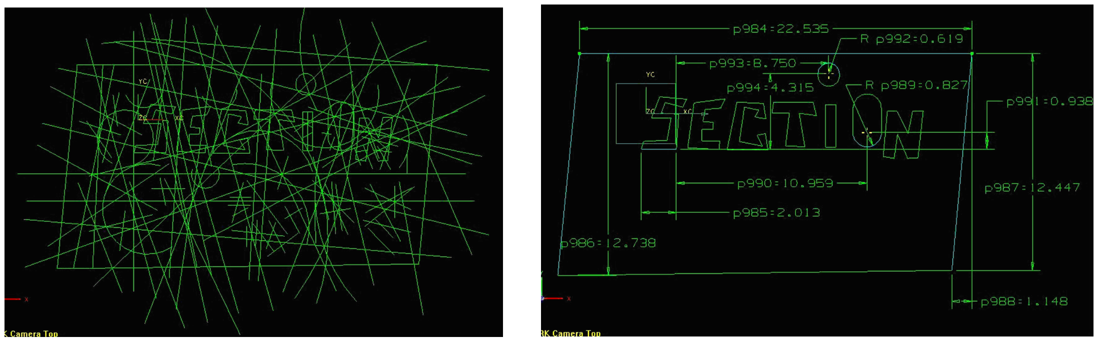
Figure 7: Functions within our C++ code.

Figure 7 Function 1, layerSelectable, enables the 256 available work layers within SIEMENS NX8 to be user-selectable and makes layerZ the active work layer. Function 2, selectAll, displays the object selection dialog, highlights the objects in the geometry and records the ID of objects into an array. It then sets the objects to layerZ and makes layerZ and layerY the only active work layers. Function 3, getData, records the curve properties of the objects in the array. These properties are point, tangent, unit principal normal, unit normal, torsion, and radius of curvature.

Function 4, compare1, compares the start and end-points of all lines and arcs to determine whether they are part of the enclosed area that defines the cross section of interest or are excess lines that need to remove. This function will start the analysis of every individual object to determine if they are part of the cross section of interest. Excess lines will be set to LayerZ and good lines to LayerY. Currently, we used the same SIEMENS NX8 tolerance for the coordinate comparison and we could improve this by defining a user to set a different tolerance. Function 5, setLayerY, will set LayerY, which contains the good lines previously calculated, as the only active and shown work layer effectively removing excess objects from the display. It will remove all lines saved in LayerZ and they thus these lines will not display. The last function, Function 6, dimsConsts, displays the line selection dialog, records the curve properties and highlights the lines or arcs selected by the user and saves the absolute spatial coordinates of the start and end-points. With this information, we are able to display dimensions and also creates dimensional and geometric constraints and applies them to the selected objects. After completing all the line selection subroutines, the last task is to check the constrained status and degrees of freedom of the geometry. The resulting display within the SIEMENS NX8 environment is a geometry free of excess lines and with the user selected objects dimensioned and constrained and a message displayed to the user with information about constrained status, over-constrained, fully constrained, or how many constraints are required.

4. Validation

Now, we apply the toolkit to various examples to show the complete removal of unwanted items and the dimensioning and constraining of the objects within the geometry selected by the user.



(a) Before toolkit application (b) After toolkit application

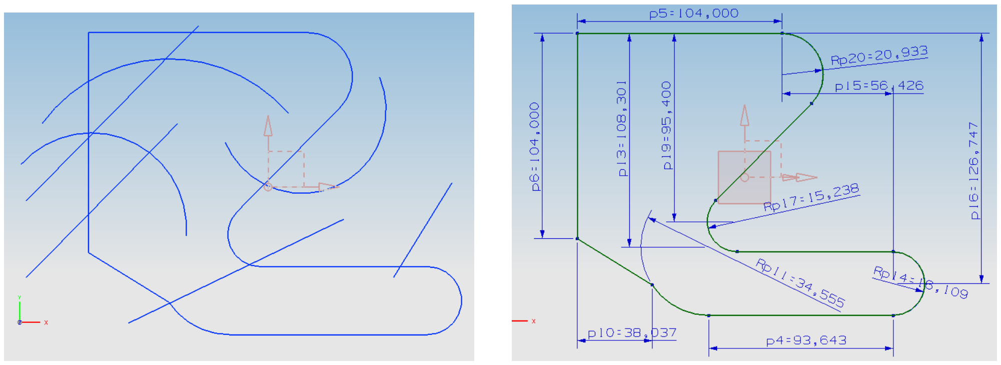
Figure 8: Dimensioning the word "Section".

4.1 Example 1: Sketch of the Word "SECTION"

Figure 8 consists of an arbitrary sketch that contains the word *section* covered by a multitude of unnecessary lines and arcs that cover the geometry. After running our toolkit, we can clearly identify the intended geometry and see the dimensioning and constraints of the parts.

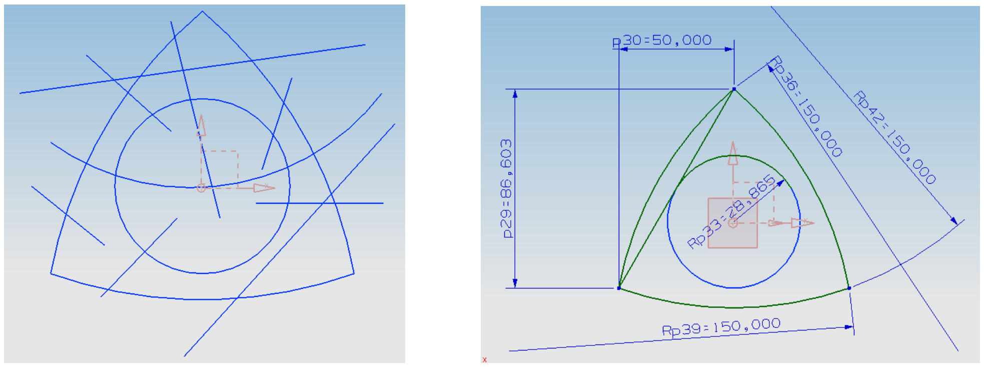
4.2 Example 2: A machined component

Figure 9 represents a machined component covered by a multitude of unnecessary lines and arcs that cover the geometry. After running our toolkit, we can clearly identify the intended geometry and see the dimensioning and constraints of the parts.



(a) Before toolkit application (b) After toolkit application

Figure 9: Dimensioning a machined component.



(a) Before toolkit application (b) After toolkit application

Figure 10: Dimensioning a Wankel-Type Rotor Profile.

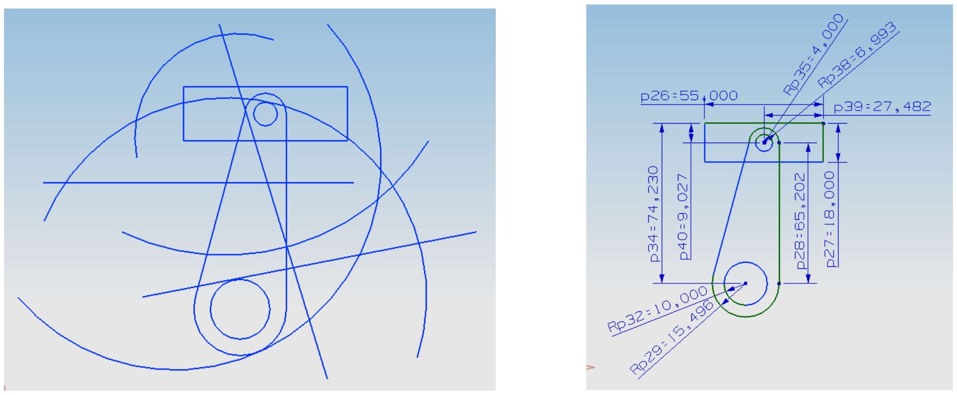
4.3 Example 3: Wankel-Type Rotor from an automotive engine

Figure 10 represents a side view of a Wankel-Type Rotor from an automotive internal combustion engine covered by a multitude of unnecessary lines and arcs that cover the geometry.

After running our toolkit, we can clearly identify the intended geometry and see the dimensioning and constraints of the parts.

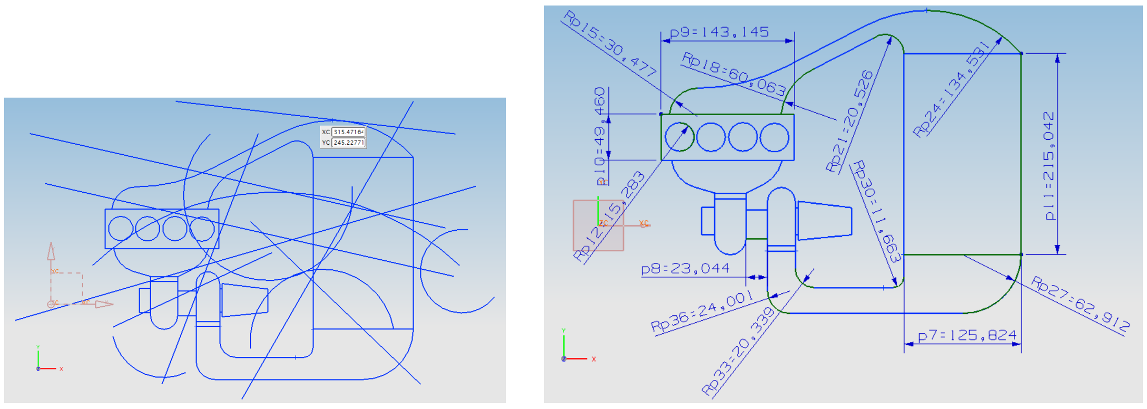
4.4 Example 4: Piston and rod representation

Figure 11 shows the side view geometry representing a piston and rod from an automotive internal combustion engine covered by a multitude of unnecessary lines and arcs that cover the geometry. After running our toolkit, we can clearly identify the intended geometry and see the dimensioning and constraints of the parts.



(a) Before toolkit application (b) After toolkit application

Figure 11. Dimensioning a piston and a rod.



(a) Before toolkit application (b) After toolkit application

Figure 12: Dimensioning a turbocharged and intercooled 4-cylinder internal combustion engine.

4.5 Example 5: Turbocharged and intercooled 4-cylinder internal combustion engine

Figure 12 represents a turbocharged and intercooled 4-cylinder internal combustion engine covered by a multitude of unnecessary lines and arcs that cover the geometry. After running our toolkit, we can clearly identify the intended geometry and see the dimensioning and constraints of the parts.

4.6 Example 6: Airfoil Calculator

Enclosed another example of the versatility of the code. We developed a subroutine that is capable of developing a 2D geometry based on data points. A good example is an airfoil, whose geometry is~~we~~ usually known in discrete points. Given the data points, we are able to generate the geometry automatically in SIEMENS NX8, as shown in Figure 13. For hereon, we can apply the code previously used.

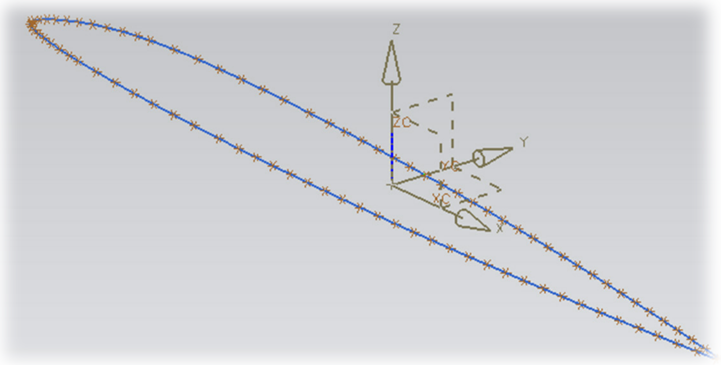


Figure 13: 2D Airfoil generator.

5. Discussion

Our proposed toolkit advances the field of computer aided design and modeling by being able to interact with SIEMENS NX8 and enhance its capabilities to analyze 2D cross-sections. The designer will be able to view any 2D drawing, remove unwanted objects and apply characteristic dimensions and constraints to the geometry, improving the characterization and analysis process and thus saving time. It is worth to mention, that we can expand the current toolkit to interface with any Solid Modeling software packages such as AutoCAD or CATIA with few modifications. The potential exists for this toolkit to become “universal” in its application and being able to interact with many different software packages. This will ~~make this work~~be an attractive proposition for the industry.

Prior research in the area provides a good foundation for developers to improve further the capabilities of our toolkit. Automating the application of dimensions, providing the user with different options for the dimensioning schemes and optimizing constraint management are some areas that were outside the scope of this work but can be developed by future researchers.

6. Final Remarks

The main objectives of this work were to improve ~~how~~ the current modeling software, such as SIEMENS NX8, to handle the dimensioning and constraining processes for 2D geometries as well as the removal of unnecessary objects by developing a toolkit to help make these processes more intuitive and user friendly.

Using C++ programming language with specific software libraries and functions to interface with SIEMENS NX8, we developed a toolkit to analyze the 2D geometries within the modeling environment. The result was a geometry that no longer contains unnecessary geometries and the user-selected objects are dimensioned and geometrically constrained, making it parameterized. This helps the designer visualize and understand the spatial relationships of the different objects within the geometry.

We met the main scope of this with the development of a C++ toolkit that interfaces with SIEMENS NX8 and facilitates the removal of unnecessary objects and the application of dimensions and geometric constraints to the selected objects within the geometry. We did not consider the layout of the dimensions because it was not within the scope of our work and remains as an open issue for future work. The automation of the different processes within the tool, such as dimensioning and constraint application, constraint management and optimization techniques; also remain as open issues for future developments.

Although the resulting output from manual selection is enough for documenting the functionality of this toolkit, we recommend developing algorithms to automate and optimize the dimensioning and constraining processes and for obtaining a better layout for dimensioning geometry.

7. Acknowledgments

This work is a result of the PACE partnership. We are grateful for their input and providing our university with their software.

8. References

[1] Dones Pérez, P. M. (1991) Automatic Dimensioning in Constraint-based Geometry, Thesis (M.S.), University of Puerto Rico Mayagüez Campus.

[2] Maarten, J. and Van, E., (1989). *Creation and Modification of Parameterized Solid Models by Graphical Interaction*, Computer & Graphics, 13(1), 71-76.

[3] Light, R. and Gossard, D., (1982). *Modification of Geometric Models through Variational Geometry*, Computer Aided Design, 14(4), 209-214.

[4] Suzuki, H., Ando, H. & Kimura, F., (1990). *Geometric Constraints and Reasoning for Geometrical CAD Systems*. , Computer & Graphics, 14(2), 211-224.

[5] Pérez Jiménez, A. (1993) *Finite Element Modeling and Optimization in a Constraint-based Environment*, Thesis (M.S.), University of Puerto Rico Mayagüez Campus.

[6] Yuen, M. M., Tan, S. T. and Yu, K. M., (1988). *Scheme for Automatic Dimensioning of CGS Defined Parts*, Computer Aided Design, 20(3), 151-159.

[7] Aldefield, B., (1988). *Variation of Geometries Based on a Geometric-Reasoning Method*, Computer Aided Design, 20(3), 117-126.

[8] Pabón Irizarry, I. U. (1996) *Artificial Intelligence in Automatic Dimensioning Layout*, Thesis (M.S.), University of Puerto Rico Mayagüez Campus.

[9] Jaramillo, H. (1993) *Automatic Dimensioning and Tolerances*, Thesis (M.S.), University of Puerto Rico Mayagüez Campus.

[10] Serrano, D. (1987) *Constraint Management in Conceptual Design*, Thesis (Sc.D.), Massachusetts Institute of Technology.

[11] Serrano, D. (1991) *Automatic Dimensioning in Design for Manufacturing*, ACM OB9791-427-9/91, pp. 379-386.

[12] Ullman, L., Signer, A., (2006) *C++ Programming*, Berkeley: Peachpit Press.

[13] Liberty, J., Horvath, D. B. (2005) *C++*, Indianapolis, Sam's Publishing.

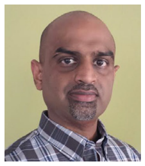
[14] UGS Corp. (2007) *'Intermediate NX Design and Assemblies with Teamcenter Integration – Student Guide'*, Publication Number: MT10056-TC-S – NX 5.

[15] Tickoo, S., Kanthe, A. P. (2007). *NX 5 for designers*. New York: CADCIM Technologies. ISBN: 978-1-932709-40-7.

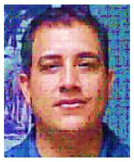
[16] WEB: http://design.osu.edu/carlson/history/lesson10.html . Carlson, W (2003). *A Critical History of Computer Graphics and Animation, Section 10: CAD/CAM/CADD/CAE.* The

Ohio State University.

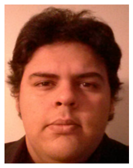
[17] WEB: http://www.open-std.org/jtc1/sc22/wg21/. *ISO/IEC JTC1/SC22/WG21 - The C++ Standards Committee* (2008).



Dr. V. Goyal is an associate professor committed to develop a strong sponsored research program for aerospace, automotive, biomechanical and naval structures by advancing modern computational methods and creating new ones, establishing state-of-the-art testing laboratories, and teaching courses for undergraduate and graduate programs. Dr. Goyal, US citizen and fully bilingual in both English and Spanish, has over 17 years of experience in advanced computational methods applied to structures. He has over 25 technical publications, main author of two books (Aircraft Structures for Engineers and Finite Element Analysis by Pearson Education Publishers), second author of Biomechanics of Artificial Organs and Prostheses (by Apple Academic Press), and has been recipient of several research grants from Lockheed Martin Co., ONR, and Pratt & Whitney.



Dr. Ricky Valentin is an Associate Professor and the Interim Director of the Department of Mechanical Engineering at UPRM. Dr. Valentin completed an engineering degree in 1996 in Mechanical Engineering at the University of Puerto Rico, Mayaguez, a Master of Engineering Science degree in 1997 (Wisconsin-Madison), and a Ph.D. from the University of Maryland at College Park in 2003. Dr. Valentin's major research area is the innovative nano-manufacturing techniques to build templates for electronic packaging, alternative energy, environmental remediation, and biomedical applications.



Michael J. Cruz is a Mechanical Engineering Undergraduate Student and Researcher at the University of Puerto Rico Mayaguez Campus. He will complete his Bachelor's Degree in the Science of Mechanical Engineering in the Fall Semester 2013. He has interned with General Electric Aviation and General Motors. He currently lives in Mayagüez, Puerto Rico. His research interests are PLM CAx enterprise strategy and composite materials. He is currently pursuing a career in the energy industry.

Neit J. Nieves-Flores, P.E. is a Senior level Engineer at Honeywell Aerospace working with Commercial and Military Aircraft Design and Manufacturing Projects. He completed a Bachelor of Engineering degree in Mechanical Engineering in 2000 and a Master of Engineering in Mechanical Engineering in 2009, both at the University of Puerto Rico, Mayaguez. He has worked in Design and Manufacturing in the Automotive Industry for Visteon Corporation and American Axle & Manufacturing, Inc. His research interests include Internal Combustion Engine Flow, Combustion Performance and Heat Transfer as well as Composite and Ceramic Materials for application in Engine Design.

Peer Review: This article has been internationally peer-reviewed and accepted for publication according to the guidelines given at the journal's website.