

Modification to 3D Model from Exchange Format File Using Visual Basic in SolidWorks

Li Jie, Hua Shun Gang

School of Mechanical Engineering, Dalian University of Technology, 2, Linggong Road, Ganjingzi District, Dalian City, China

Abstract: During the process of modelling, analysis and optimization of mechanical system, 3D models need to be saved as exchange format files as well as be modified frequently. The exchange format file would lose certain feature information of the original model, making operations to models inconvenient. Aiming at the model imported from exchange format file, we study programming of relevant operation with Visual Basic in SolidWorks environment, including entity extrusion or compression from a planar surface and fillet/chamfer for edges. Specialized functions are developed to facilitate the modelling and modification for complicated solids by appending menu icons into SolidWorks, thus improving the model operation efficiency and degree of automation significantly.

Keywords: secondary development, exchange format file, model modification, SolidWorks.

1. Introduction

In the process of mechanical parts and engineering design, various component models are created by 3D modeling software in accordance with industry standards, and assembled to constitute sophisticated mechanical system. In order to obtain lightweight mechanical structures that meet strength and stiffness requirements, both the static and dynamic performance of major parts or the whole system are analyzed by using the methods of finite element and multibody dynamics, then the structure and dimensions of the original model are adapted appropriately [1], [2]. For the design of complicated mechanical systems and products, different parts are designed probably by different designers in different departments under heterogeneous environment, thereby the designed results will be saved as exchange format files, such as IGS, STEP, and so on, for facilitating transmission and exchange. Such that, original design characteristics that belong to the model may be lost to make the subsequent operations inconvenient.

Figure 1 shows a model input from an IGS format file in SolidWorks environment. It can be seen that there is no feature information of the model displayed in feature manager on the left. To change a specified dimension of the model, e.g., increasing or decreasing the length in the direction perpendicular to the highlight reference plane, we should enter the sketch mode and use convert entities tool to pick up edges one by one, acquiring a fully closed region to execute the extrusion. However, using mouse to select edges like that is rather cumbersome, furthermore, maybe a certain edge will be selected repeatedly or omitted, such that a closed loop outline will not be constituted. As a consequence, the user may fail to accomplish this operation. In CAD, we hope to operate on models from exchange format files accurately and conveniently. For this purpose, the method of secondary development can be adopted to realize these operations [3], [4].

As one of the most classical 3D modelling software's, apart from its function of modeling, SolidWorks also provides a set

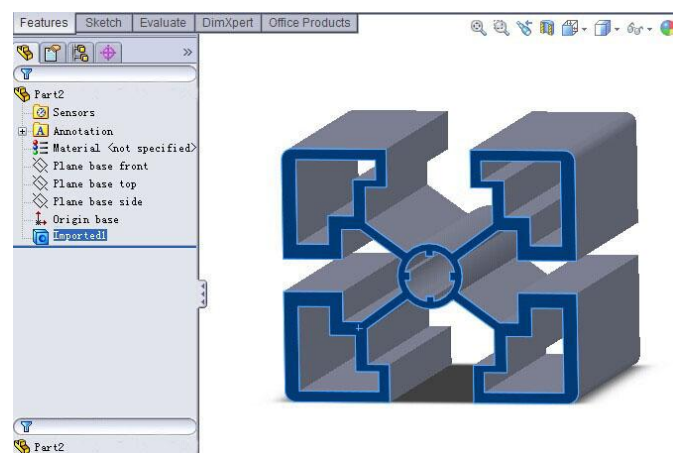


Figure 1: Display of CAD model from IGS format file in SolidWorks

of interface functions, i.e., Application Programming Interface (API). Users can utilize advanced programming languages with API to build desired application system [5], [6]. In this paper, we study the method of secondary development in SolidWorks with Visual Basic, and realize the solid extrusion or compression from a planar surface and fillet/chamfer for edges.

2. Secondary development based on Solid Works

Secondary development based on SolidWorks is achieved via embedded API, which provides a large amount of OLE (Object Linking and Embedding) objects and their properties as well as methods (see fig. 2). With the advanced programming language, users can call OLE objects to access SolidWorks via executing corresponding codes. Thus, similar to interactive operation manually, various operations can be realized to meet the user's needs [7], [8].

In the process of modeling, we can use the methods below to achieve parametric design of parts.

1) Dimension-driving: Dimensions can be driven and changed by modifying the values of design variables that

auto-generated during model building. Design variables are usually auto-generated and given identifies during the process of modeling.

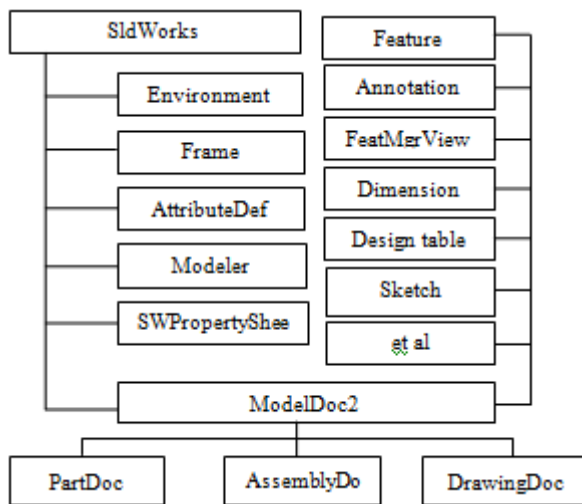


Figure 2: SolidWorks API structure

Dimension-driving method only need to modify the values of variables in SolidWorks and has some advantages of speediness and high-efficiency. However, this method lacks the ability of variant design and is mainly applied for modifying component that possess the same shape but different sizes.

2) Macro-recording: This method can record the operation steps in modeling, and builds the VB program automatically. Users only need to modify the related data and program structure, and run the code again to regenerate the desired model. As the codes are recorded according to the system settings and rules, there will exist some redundant codes and default object names, which make it impossible for users to capture and operate these objects.

3) Encoding with API functions: Users can encode to achieve specific functions by using SolidWorks OLE objects' types, properties and methods [9]. In this paper, we aim at mechanical models from exchange format files, in which some feature information and data will be lost when the original models are saved as exchange format files. Hence, the dimension-driving method cannot be used to drive the dimensions associated with these original features. For the macro-recording, we also fail to encode to deal with models because of the default object names. Therefore, we encode with API functions to accomplish operations to models. Fig. 3 shows the flow diagram of processing models with VB.

3. Extruding from the planar surface

Modifying dimensions, such as length, thickness, height and so on, is one of the most common operations for 3D modeling. In this section, we will give an example of the solid extrusion to illustrate how to utility SolidWorks' OLE objects as well as their types, properties and methods to modifying the model. Users can draw inferences about other cases from one instance.

Figure 4 shows a model of track loader [10]. We import the IGS format file of the chassis part of the loader into

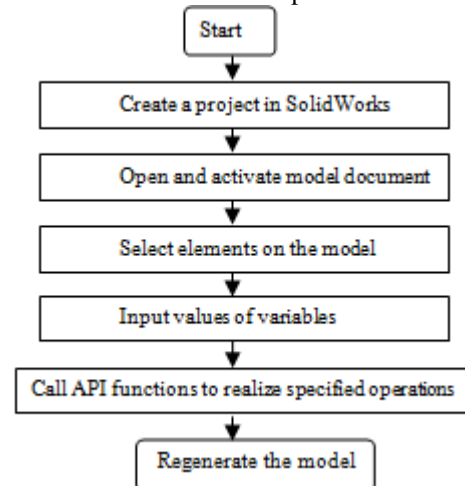


Figure 3: The flow diagram of encoding for model modification

SolidWorks. In the FeatureManager of SolidWorks, there isn't any feature information, hence, we need to encode and call some API functions, capturing the solid and then changing the dimension. The part codes modifying the roof thickness of the chassis part is listed below.

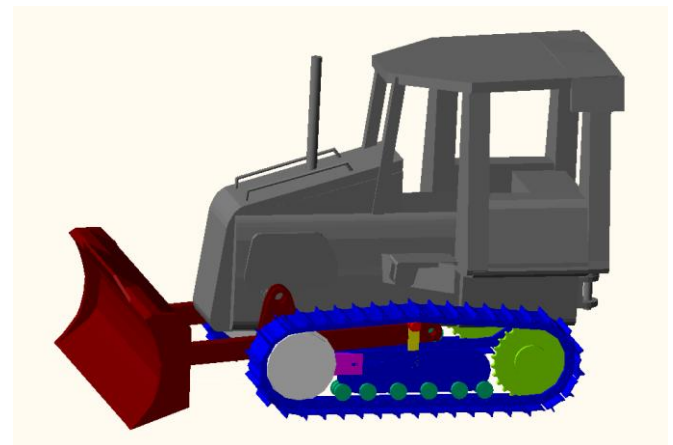


Figure 4: Track loader model

```
Option Explicit
Public m_depth As Double
Sub swmain()
Dim swap As SldWorks.SldWorks
Dim swModel As SldWorks.ModelDoc2
Dim swPart As SldWorks.PartDoc
Dim swNewPart As SldWorks.PartDoc
Dim swModeler As SldWorks.Modeler
Dim swSelMgr As SldWorks.SelectionMgr
Dim swSelFace As SldWorks.Face2
Dim swBody As SldWorks.Body2
Dim swFeat As SldWorks.Feature
Dim swMathe As SldWorks.MathUtility
Dim swNormal As Variant
Dim dirArr(2) As Double
Set swApp = CreateObject("SldWorks.Application")
Set swModel = swApp.ActiveDoc
Set swPart = swModel
```

```

Set swSelMgr = swModel.SelectionManager
Set swModeler = swApp.GetModeler
Set swMathe = swApp.GetMathUtility
Set swSelFace = swSelMgr.GetSelectedObject3(1)
Set swBody = swSelFace.GetBody
swNormal = swSelFace.Normal
dirArr(0) = swNormal(0)
dirArr(1) = swNormal(1)
dirArr(2) = swNormal(2)
Dim dirVector As SldWorks.MathVector
Set dirVector = swMathe.CreateVector((dirArr))
Dim extrudedBody As SldWorks.Body2
Dim swnb As Variant
Set swnb = swSelFace.CreateSheetBody()
Set extrudedBody = swModeler.CreateExtrudedBody(swnb,
dirVector, m_depth / 1000)
Dim errorCode As Long
Dim sswsbody As SldWorks.Body2
Set sswsbody = swBody.Copy()
Dim ResultBodiesPerm As Variant
ResultBodiesPerm=sswsbody.Operations2(SWBODYADD,
extrudedBody, errorCode)
Set swNewPart = swApp.NewPart
Set swFeat= swNewPart.CreateFeatureFromBody3
(sswsbody, False, 0)
End Sub
    
```

The followings are the explanations to the main sentences.

Set swSelFace=swSelMgr.GetSelected Object3(1): selects a face of the object as reference surface for extruding;

Set swBody = swSelFace.GetBody: gets the body which contains the reference surface;

swNormal = swSelFace. Normal: obtains unit normal vector of the reference surface which points towards the outside of the body;

Set extrudedBody = swModeler.CreateExtrudedBody (swnb , dirVector, depth / 1000): creates an extruded body from the reference surface, and *swnb* represents a sheet body; *dirVector* indicates the direction of the extrusion; *m_depth* expresses the depth value to extrusion;

ResultBodiesPerm = sswsbody.Operations2(SWBODYADD, extrudedBody, errorCode): combines the original body (*swbody*) and the extruded body (*extrudedbody*) together to form a entire body (*sswsbody*).

For the case of compression, we only need to modify the *swNormal* to opposite direction, and then replace *SWBODYADD* with *SWBODYCUT*. Thus 3D model compression can be performed through Boolean subtraction operation to the original body with the extruded body.

Figure 5 shows a form of size extruding, such that the extrusion value can be input and the codes above are connected with the *Extrude* button.

By creating the ActiveX DLL and a menu icon in SolidWorks, ones can select a reference plane of the model and click the menu icon for the extrusion. Fig. 6 is an example of increasing thickness of the roof by 100mm while fig. 7 decreasing

thickness of the column by 65mm.

4. Fillet and chamfer

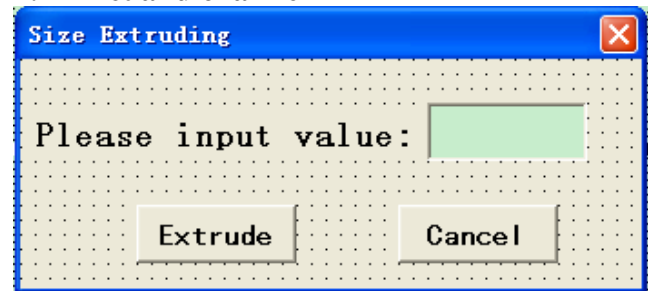
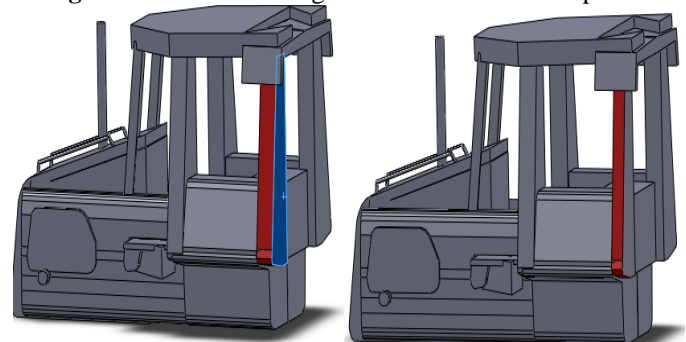


Figure 5: Size extruding form



(a) Original model (b) Increasing the roof thickness

Figure 6: Extrusion along normal direction of the plane



a) Original model b) Decreasing the column thickness

Figure 7: An example of decreasing thickness

In order to meet the need of structure design and avoid stress concentration when analyzed structurally, the region near the sharp edges should be modified with fillet or chamfer. In this paper, we encode to realize the fillet and chamfer with VB.

(1) Fillet Fillet is an operation that creates a rounded internal or external face along one or more edges in solid or surface feature. In SolidWorks, we can use *FeatureFillet* function for this operation.

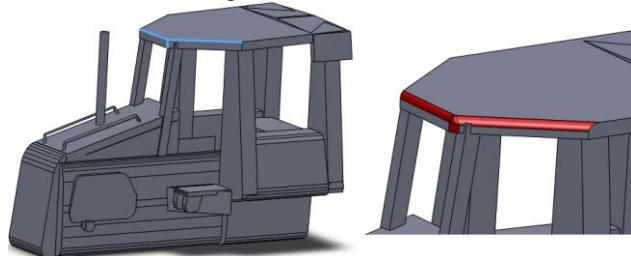
Our fillet operation can deal with the case along one or more edges, and the rounded radius may be same or not. Users can choose the edges and input the corresponding radius values for fillet. The key sentence for this operation is

Value = instance.FeatureFillet (Options, R1, Ftyp, OverflowType, Raddi, SetbackDistances, PointRadiusArray).

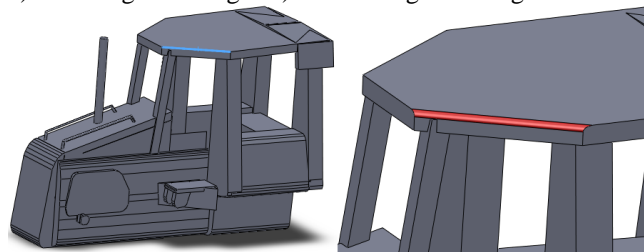
Where *Options* indicates various fillet options; *R1* represents the radius value while choosing the type of *Constant radius*; *Ftyp* denotes the fillet type; *Raddi* refers to an array of

different radius values while choosing the type of *variable radius*.

Figure 8 shows the fillet operation along one or more edges in solid. We select three edges (in blue) in front of the roof in fig.8(a), and the fillet result with rounded radius 40, 50, 80mm is shown in fig. 8(b). The result along one edge with a radius 30mm is shown in fig. 8(d).



a) Selecting three edges b) Fillet along three edges



c) Selecting one edge d) Fillet along one edge

Figure 8: Fillet along one or more edges

(2) Chamfer Chamfer is an operation that forms the flat surface by cutting away the sharp edges of two meeting surfaces. In this paper, we implement the chamfer operation for an edge and for a vertex. The critical sentence for this operation is

```
value = instance.InsertFeatureChamfer (Options,  
ChamferType, Width, Angle, OtherDist, VertexChamDist1,  
VertexChamDist2, VertexChamDist3).
```

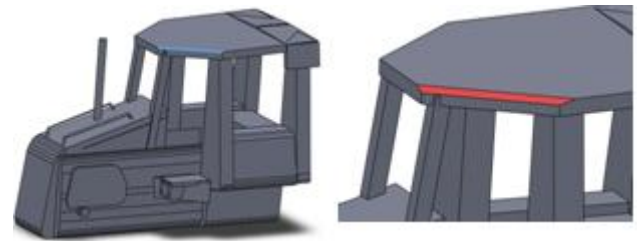
Where Options indicates various chamfer options; ChamferType represents the type of chamfer; width, Angle, etc. are the parameters related with types of chamfer.

Figure 9 shows the examples of chamfer for an edge as well as for a vertex on the roof. We pick up an edge in front of the roof and input two distance values 30 and 60mm in our designed pop-up dialog. Then the chamfer for the edge by way of distance-distance is executed and the result is shown in fig.9(b). For the way of angle-distance, we can input an angle and a distance for that. Fig. 9(d) shows the chamfer operation for a vertex with distances 60, 50, 50mm.

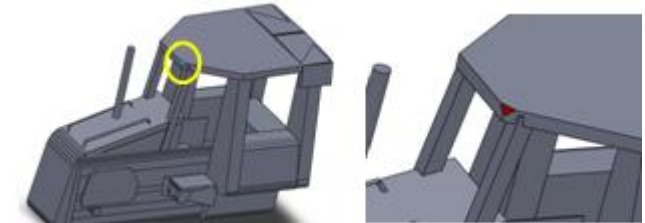
5. Conclusion

User custom-made function modules can be constructed with SolidWorks API functions. Through building a new macro in SolidWorks environment and adding references to SolidWorks libraries, we can encode for specified operations with VB and develop an ActiveX DLL project.

Aiming at the issues of some feature information lost via exchange format file, we program for achieving extrusion,



a) Selecting an edge b) Chamfer by *distance-distance*



c) Selecting a vertex d) Chamfer for the vertex

Figure 9: Chamfer for edges and a vertex

compression, fillet and chamfer operation in this paper. These functions can be applied to modeling and modifications of complicated models, simplifying operation steps and improving the degree of automation. In our investigation of the structure dynamic analysis and parameter optimization to mechanical system, the functions have been adopted for the model modification.

For the future work, we will study the extrusion from the curved surfaces with VB, as well as other functions, in order to facilitate the component modeling and modification.

References

- [1] Y. Yang. "The parametric design and intelligent assembly system based on the secondary development of solidworks", 2nd International Conference on Computer Engineering and Technology (ICCET), IEEE, pp. 602-605, 2010. (conference style)
- [2] S.P. Prince, R.G. Ryan, T. Mincer, "Common API: Using Visual Basic to Communicate between Engineering Design and Analytical Software Tools", ASEE Annual Conference 2005, ASEE, pp. 1939-1951, 2005. (conference style)
- [3] J.L. Tian, S.X. Liu, H. Fu, "CAD System Design on Standard Part Based on Software Reuse", Fourth International Symposium on Knowledge Acquisition and Modeling(KAM), IEEE, pp. 229-232, 2011. (conference style)
- [4] A. Titus, X.B. Liu, "Secondary Development of SolidWorks for Standard Components Based on Database", International Journal of Science and Research, 2(10), pp. 162-164, 2013. (journal style)
- [5] X.B. Ning, Q.S. Jiang, "A digital design method of geometric model for centrifugal fan impeller based on SolidWorks and VB", 2011 International Conference on Electronic and Mechanical Engineering and Information Technology(EMEIT), IEEE, pp. 4023-4026, 2011. (conference style)

- [6] U.H. Farhan, S. O'Brien, M.T. Rad, "SolidWorks Secondary Development with Visual Basic 6 for an Automated Modular Fixture Assembly Approach", .International Journal of Engineering, 6(6), pp. 290-304, 2012. (journal style)
- [7] B. Sun, G.T. Qin, Y.D. Fang, "Research of standard parts library construction for SolidWorks by Visual Basic", International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT), IEEE , pp. 2651-2654, 2011. (conference style)
- [8] S. Danjou, N. Lupa, P. Koehler, "Approach for Automated Product Modeling Using Knowledge-Based Design Features", Computer-Aided Design and Applications, 5(5), pp. 622-629, 2008. (journal style)
- [9] Dassault Systemes SOLIDWORKS Corp.SolidWorks2010\Help\API Help Topics.
- [10] FunctionBay Company.Recurdyn R7V5 \Help \Manual \Tutorials \Toolkit \Track_LM.

Author Profile

Li Jie received the B.E. in Qingdao University in 2013, He is a graduate student in the School of Mechanical Engineering, Dalian University of Technology, China.