Extracting Sensor Data from CAD

SteamVR™ Tracking

Introduction

Creating a JSON file from 3D CAD is a necessary step as one moves from theoretical sensor placement in hmd_designer to one that considers real world geometric constraints. This document presents two methods for generating a JSON file from 3D CAD. The first covers the equations and processes to do it by hand. While labor intensive, this is an effective method of JSON creation. The second discusses a macro for Solidworks. Once the preconditions are met, a JSON file can generated from a Solidworks file in a few clicks. The last step for each process discusses the importance of verifying that the JSON matches up with the exported STL object.

hmd_designer JSON Requirements

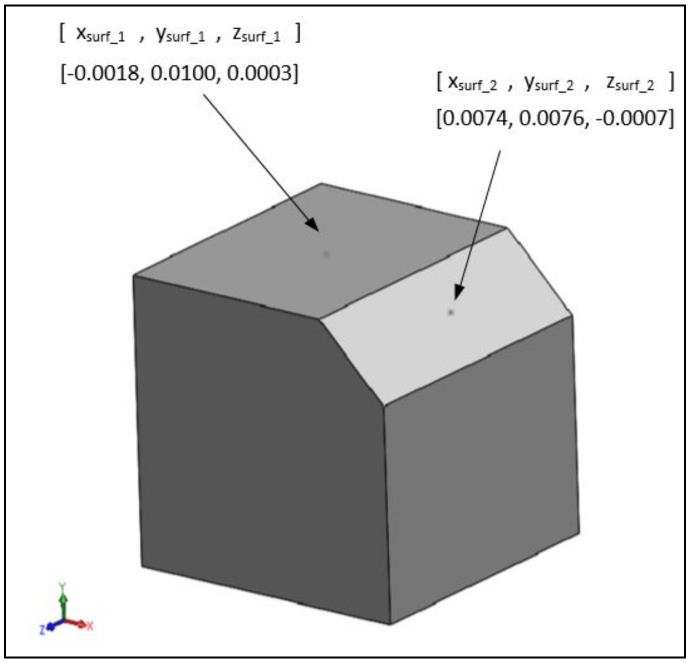
The hmd_designer tool uses the modelPoints, modelNormals, and channelMap arrays in the JSON file to define sensor properties. The elements in the modelPoints array specify the x, y, and z position of a sensor centroid. The values are expressed in meters. The elements in the modelNormals array define the orientation of each sensor using unit normal vectors. The channelMap array holds a label for each sensor. If this array is not present, hmd_designer will number the sensors on its own. All other arrays in a JSON file besides the ones mentioned are ignored. See **The JSON File** for greater detail.

Extraction from 3D CAD

Manual Extraction

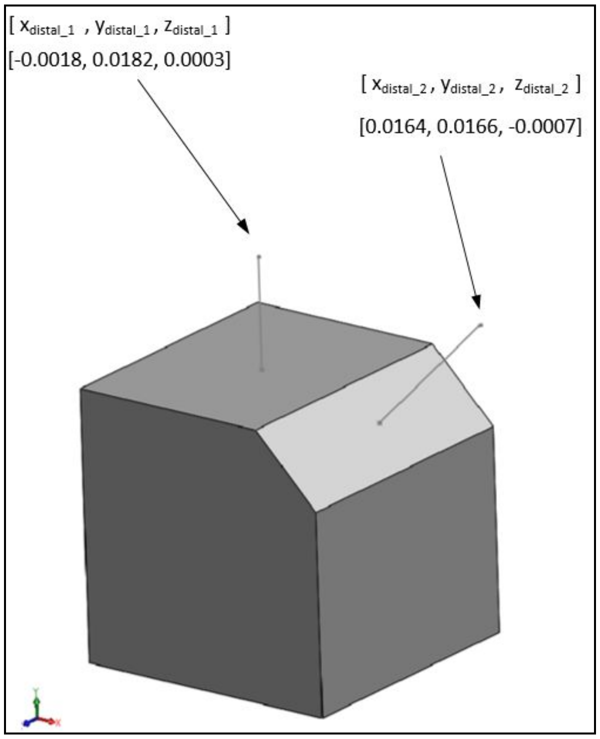
The input data for the modelPoints and modelNormals arrays can be extracted manually from 3D CAD. The process depends on the CAD package so a general description is provided. The first step is to indicate the positions of the sensor centroids. The centroids can be indicated by a sketch point on the surface of the target part. Whatever feature used, ensure that absolute location relative to the primary axis can be measured.

A cube with a 45 degree beveled face was created to demonstrate this process. Points were placed on the top and beveled surfaces. The values in the brackets are in meters. If you are in different units, be sure to convert before inputting them into the JSON file.



[x_{surf_1} , y_{surf_1} , z_{surf_1}]
[-0.0018, 0.0100, 0.0003]
[x_{surf_2} , y_{surf_2} , z_{surf_2}]
[0.0074, 0.0076, -0.0007]

The second step is to indicate the directions that the sensors are facing. Create a line extending from the sensor centroid point in the desired direction. In most cases, the line will be projected perpendicular to the surface at the point of the sensor. The length of the line does not need to be defined. In the image below, lines were projected from the sensor centroids perpendicularly from the surface. A measurement was then taken to determine the position of the distal end of each line.



[x_{distal_1} , y_{distal_1} , z_{distal_1}]
[-0.0018, 0.0182, 0.0003]
[x_{distal_2} , y_{distal_2} , z_{distal_2}]
[0.0164, 0.0166, -0.0007]

The components for each unit normal vector can then be calculated with the following equations.

$$x_{normal} = \frac{x_{distal} - x_{surf}}{\sqrt{(x_{distal} - x_{surf})^2 + (y_{distal} - y_{surf})^2 + (z_{distal} - z_{surf})^2}} \quad (1)$$

$$y_{normal} = \frac{y_{distal} - y_{surf}}{\sqrt{(x_{distal} - x_{surf})^2 + (y_{distal} - y_{surf})^2 + (z_{distal} - z_{surf})^2}} \quad (2)$$

$$z_{normal} = \frac{z_{distal} - z_{surf}}{\sqrt{(x_{distal} - x_{surf})^2 + (y_{distal} - y_{surf})^2 + (z_{distal} - z_{surf})^2}} \quad (3)$$

For example, the x_{normal} component for the second point is calculated as such:

$$x_{normal\,2} = \frac{0.0164 - 0.0074}{\sqrt{(0.0164 - 0.0074)^2 + (0.0166 - 0.0076)^2 + (-0.0007 + 0.0007)^2}} = 0.7071 \quad (4)$$

The y_{normal} and z_{normal} values can be calculated similarly. The information can then be entered into the modelPoints and modelNormal arrays.

```
"modelPoints" : [
         [ xsurf_1, ysurf_1, zsurf_1 ],
         [ xsurf_2, ysurf_2, zsurf_2]
      ]
```

```
"modelNormals" : [
         [ xnormal_1, ynormal_1, znormal_1 ],
         [ xnormal_2, ynormal_2, znormal_2 ]
      ]
```

Putting the example values into the modelPoints and modelNormal arrays gives:

```
"modelPoints" : [
         [ -0.0018, 0.0100, 0.0003 ],
         [ 0.0074, 0.0076, -0.0007]
      ]
```

```
"modelNormals" : [
         [ 0.0000, 1.0000, 0.0000 ],
         [ 0.7071, 0.7071, 0.0000 ]
      ]
```

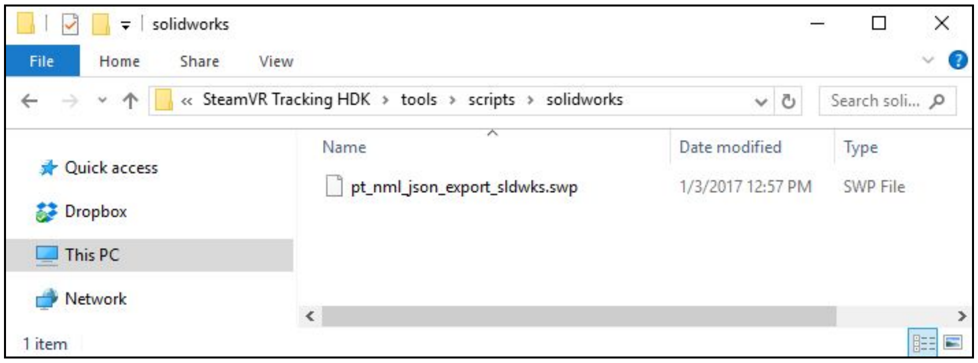
The final step is the verify that the sensors show up in the correct position and orientation on the 3D object. Refer to the “Visualizing an Existing Model” section in the **Simulation User Manual** for the proper procedure.

While the above process relies extensively on the manual entry of data, it is an effective method for prototyping or working with a small number of sensors. The process can be made more efficient by using spreadsheets or numerical analysis software.

Solidworks Export Tool

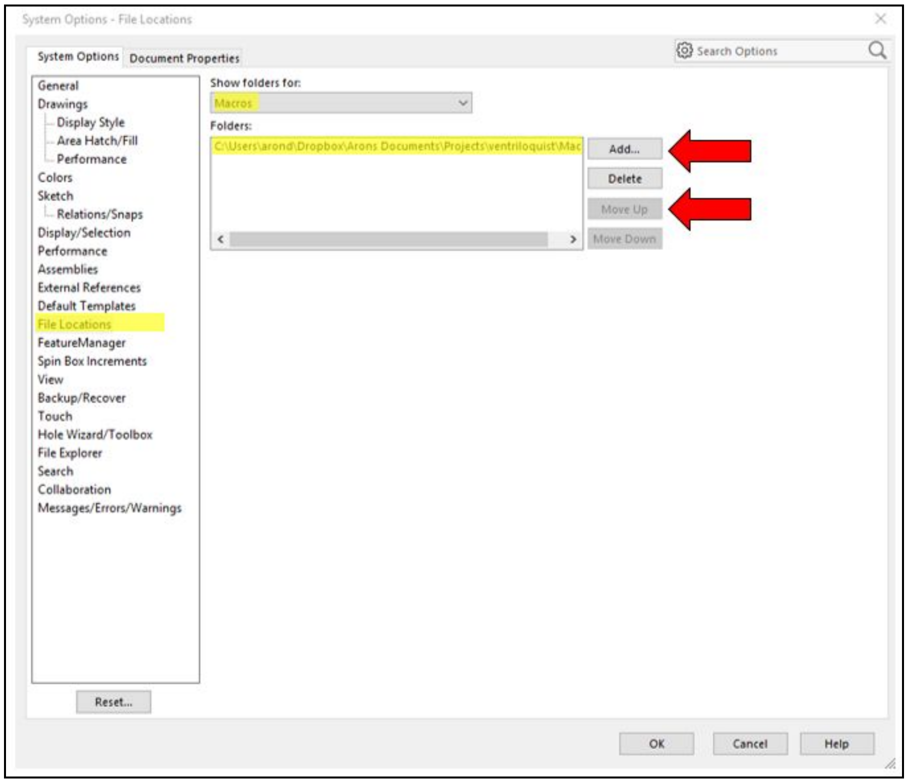
A macro was created for Solidworks to provide an automated method for extraction of sensor information. The macro creates the modelPoints and modelNormals arrays and generates a JSON file. This process reduces both JSON file creation time and user error. A walkthrough of the process is provided. Though it is written for Solidworks 2016, it should be applicable for other Solidworks versions.

The macro is included in the SteamVR Tracking HDK file and located at the folder path tools\scripts\solidworks.



Name	Date modified	Type
pt_nml_json_export_sldwks.swp	1/3/2017 12:57 PM	SWP File

Add this directory to Solidworks macro locations. Navigate to Tools > Options > File Locations > Macros. Add the above directory to the list of folders. Move the newly added location to the top of the list and click ‘OK.’ Alternatively, the macro in the HDK can be added to an existing Solidworks macro folder.

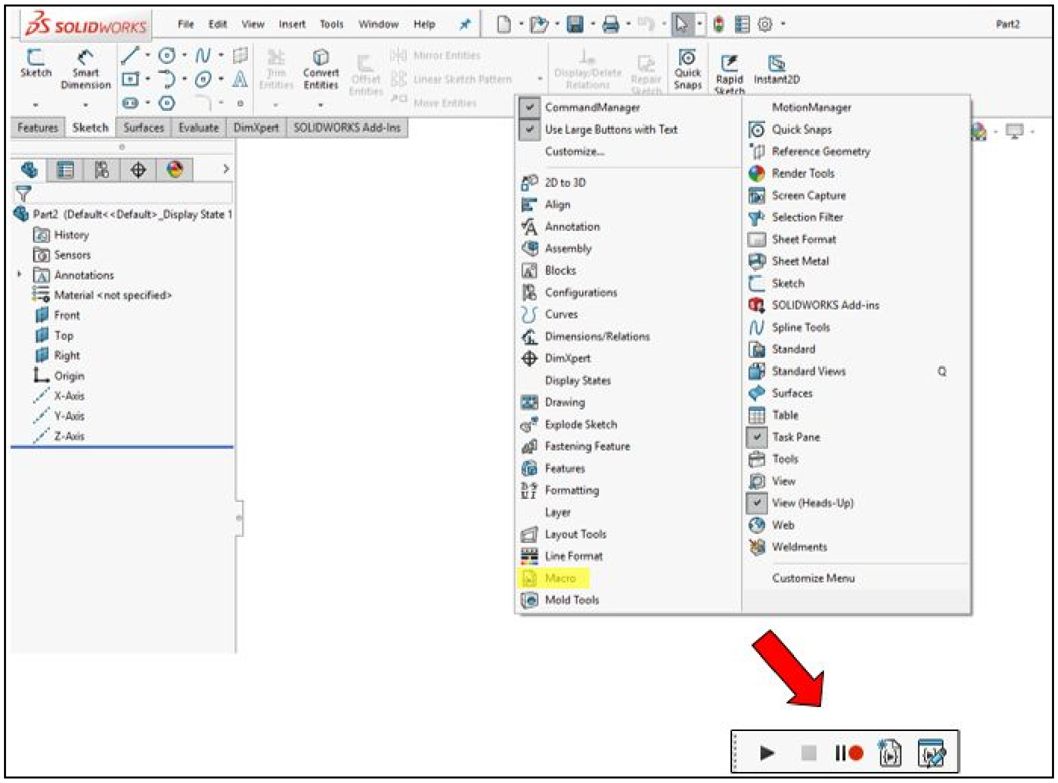


Show folders for: Macros

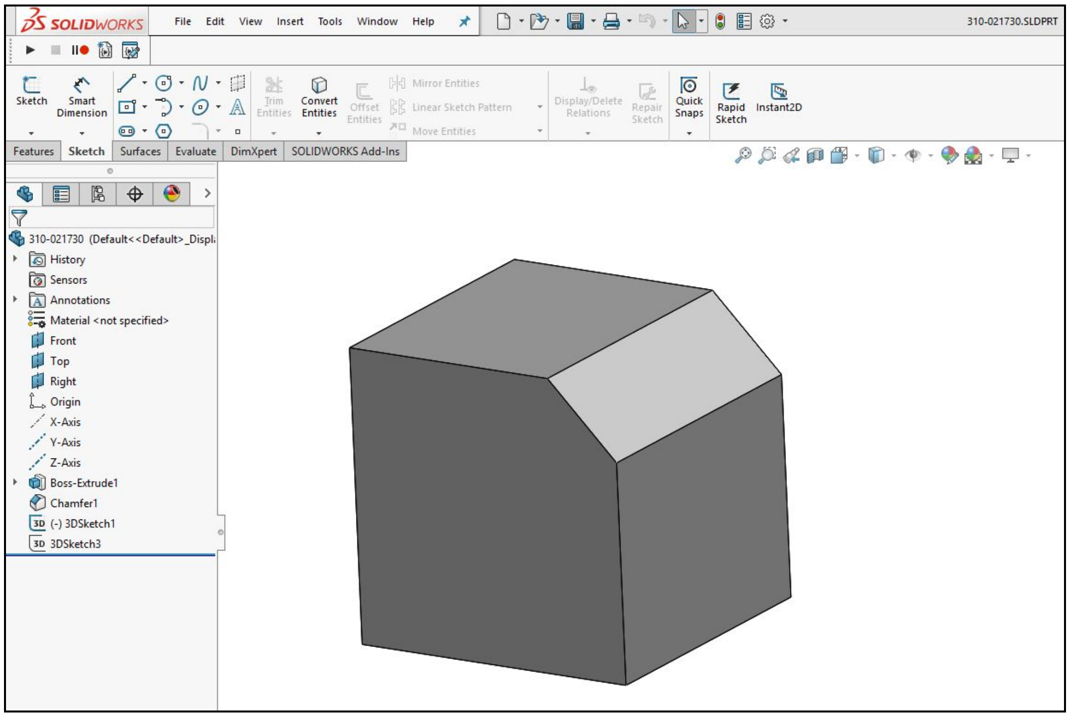
Folders: C:\Users\arond\Dropbox\Arons Documents\Projects\ventriloquist\Mac

.

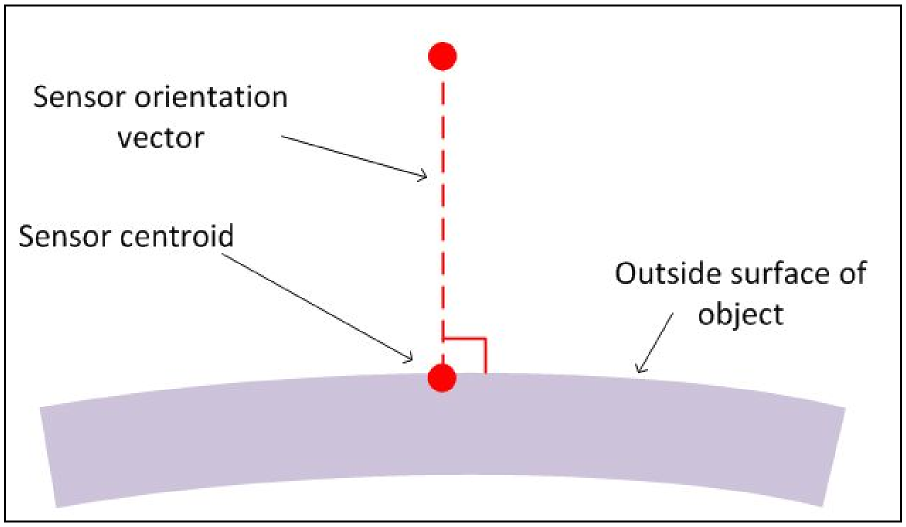
Create or open a new part in Solidworks. Add the macro menu to the Command Manager by right clicking on it and selecting ‘Macro.’



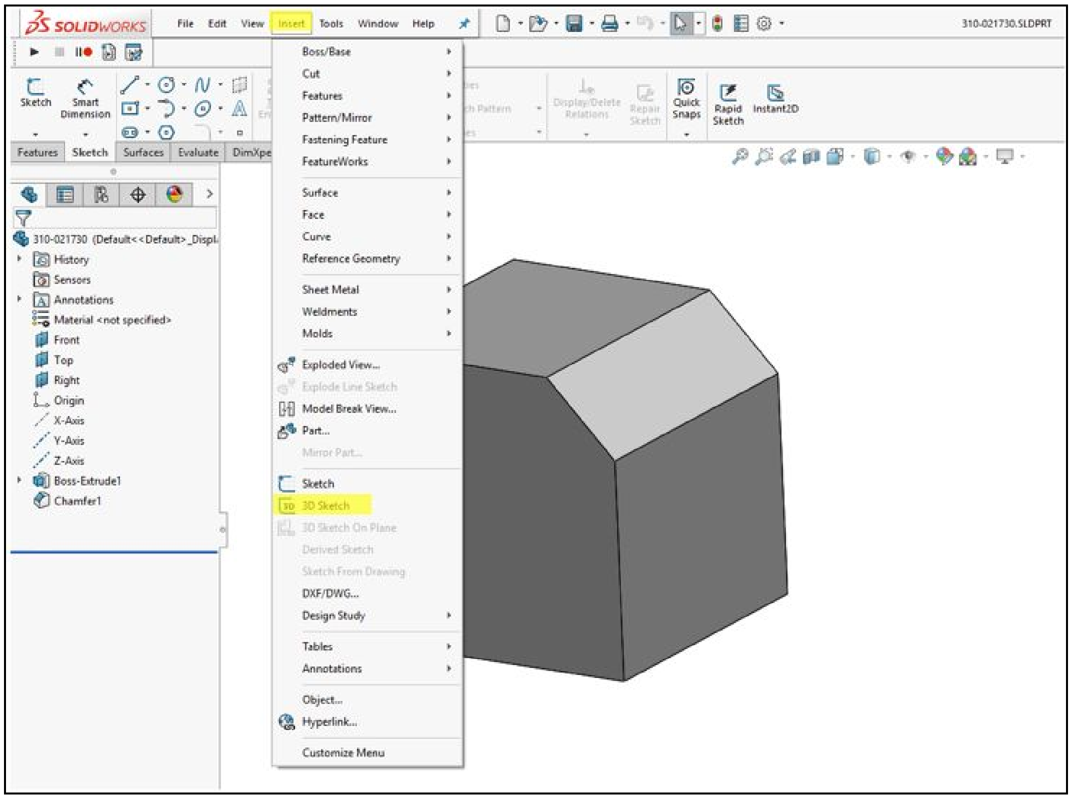
Create a part if you don't already have one open. Modify it to where you would like to create a JSON file. For demonstration purposes, a cube with a 45 degree bevel on one corner was created. This is a great shape for demonstration purposes but won't track well in SteamVR.



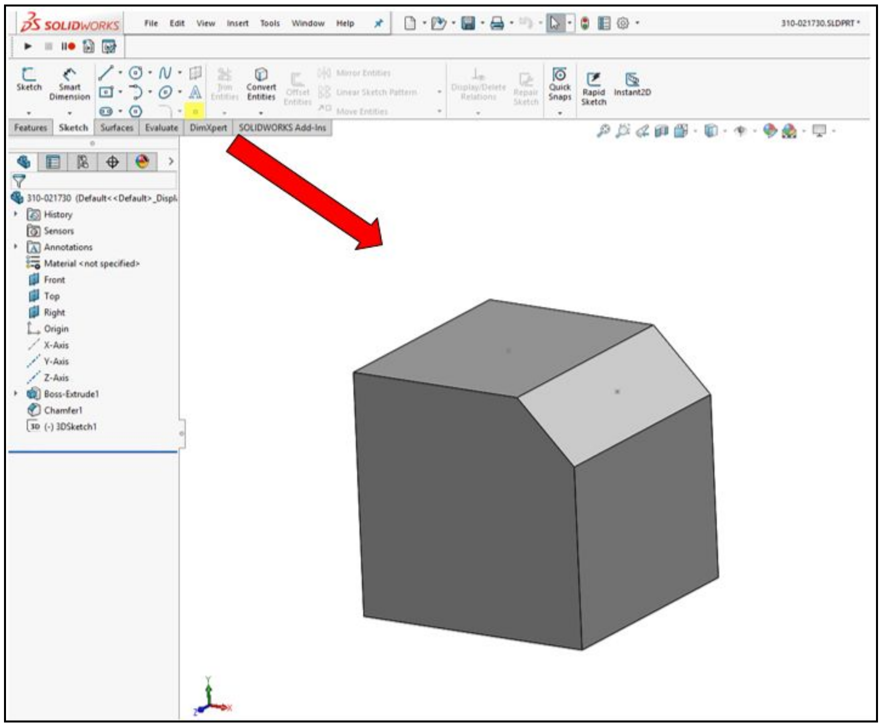
Sensor information can now be added. The assumption is made that the sensor centroid is on the outside of the part and that the orientation is away from the part body and normal to the surface. While these assumptions aren't always valid, they are generally a good place to start.



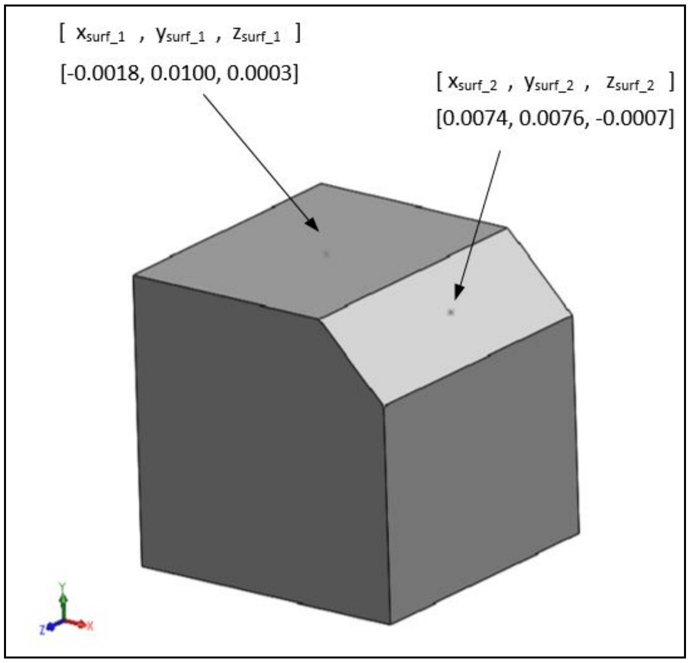
Create a new 3D Sketch by navigating to Insert > 3D Sketch.



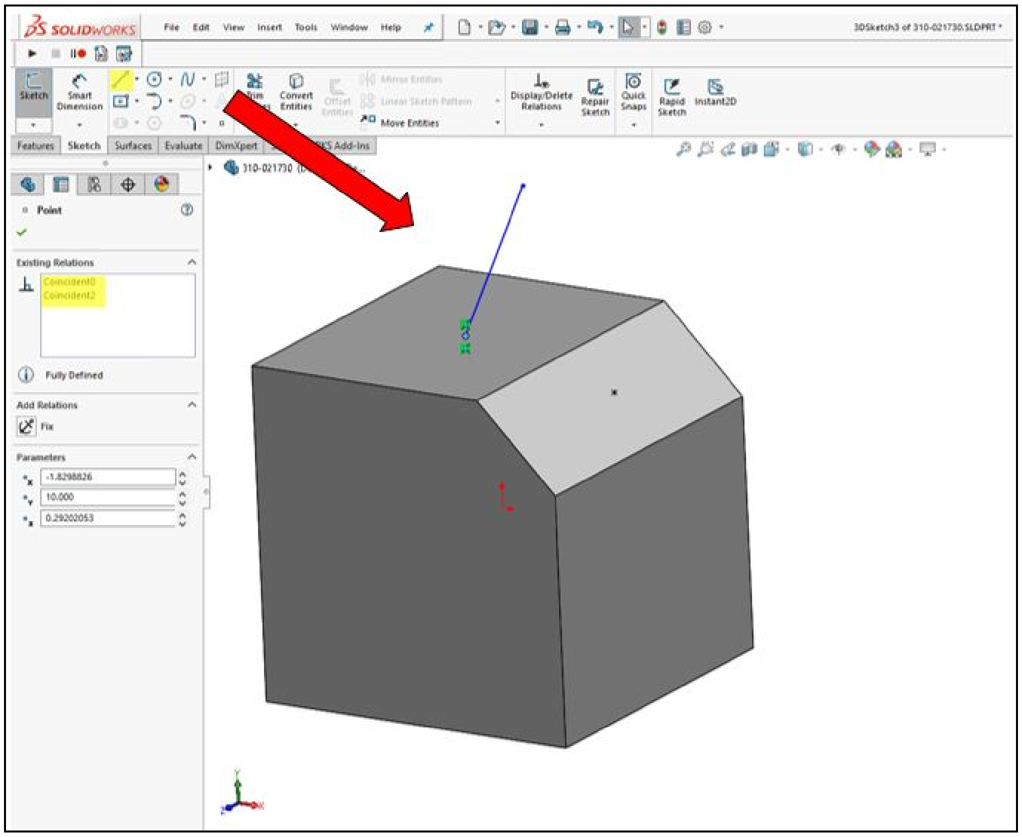
Place sketch points in the desired sensor centroid locations.



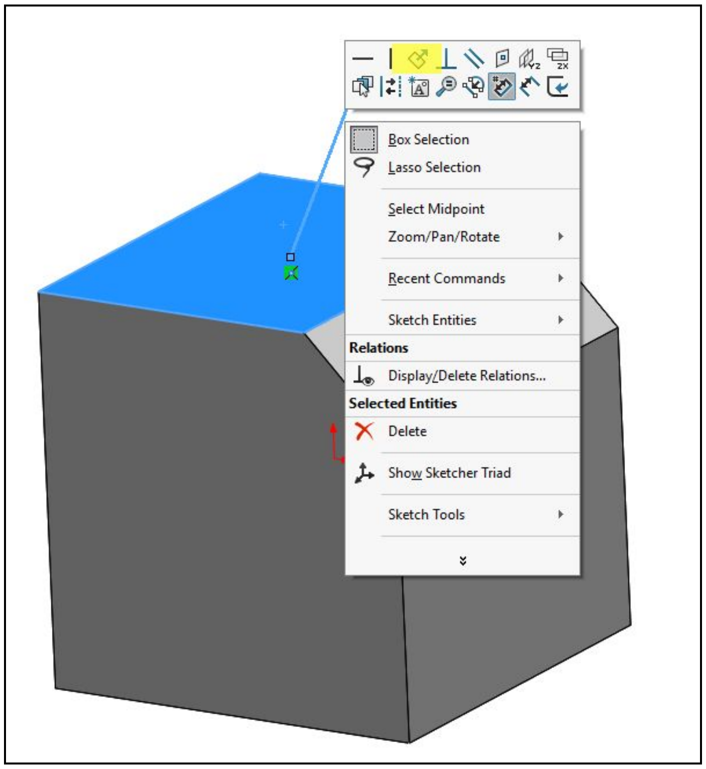
Points were placed on the top and beveled surface in the following locations.


$[\ x_{surf\_1}\ ,\ y_{surf\_1}\ ,\ z_{surf\_1}\ ]$
[-0.0018, 0.0100, 0.0003]
$[\ x_{surf\_2}\ ,\ y_{surf\_2}\ ,\ z_{surf\_2}\ ]$
[0.0074, 0.0076, -0.0007]
Y
X
Z

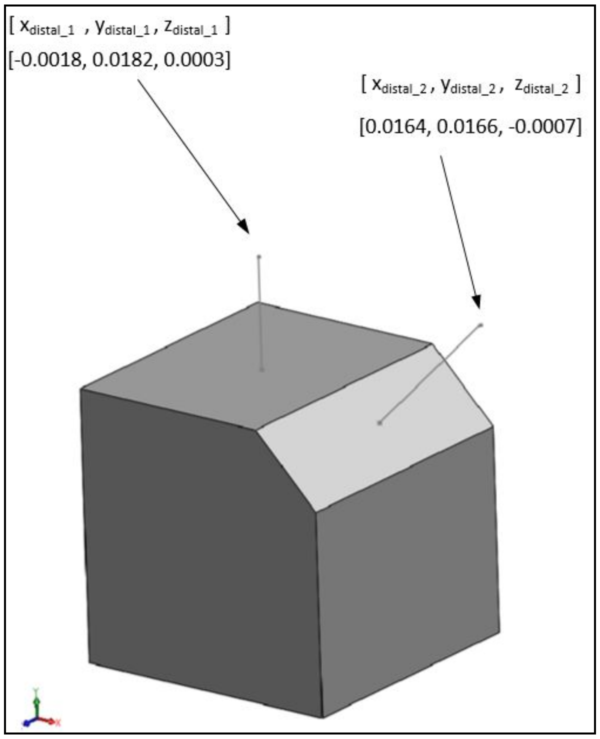
Within the same 3D sketch create a sketch segment. Attach one end to the point indicating the sensor centroid. Ensure that the sketch point and end of sketch segment have a coincident relationship.



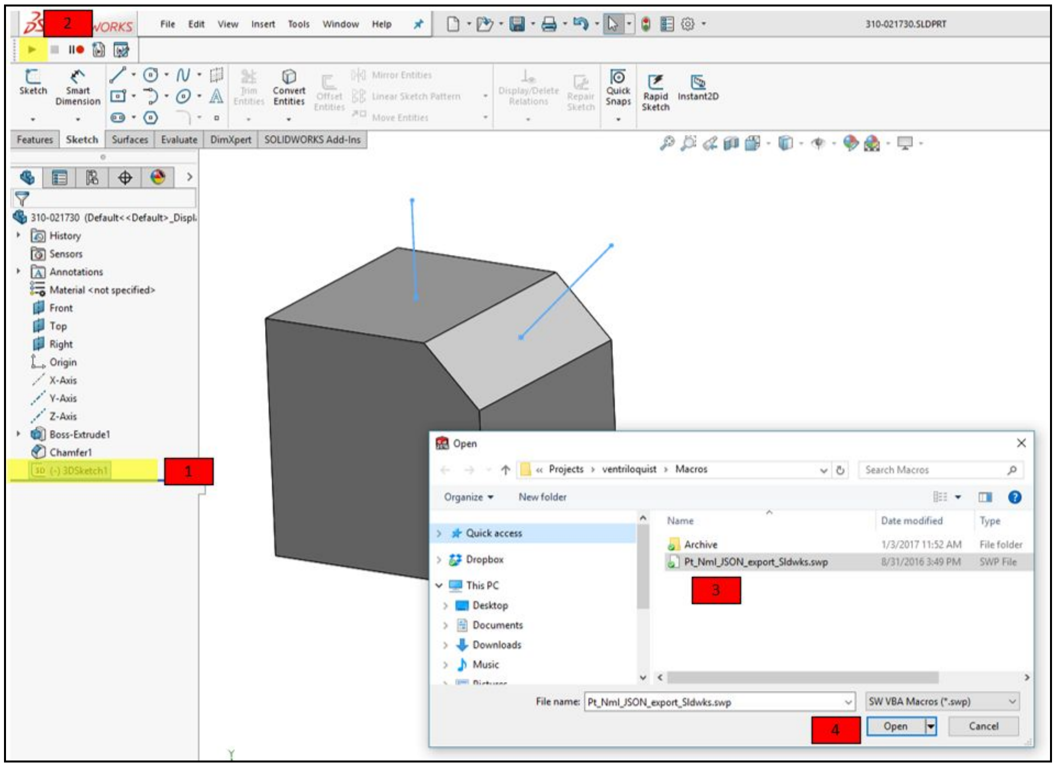
Orient the unconstrained end of the sketch in the viewing direction of the sensor. The normal constraint can be used to ensure the sketch segment is normal to the face. Select both the face and the sketch segment by holding CTRL and left-clicking on each. Right click on either and select the ‘Make Normal’ icon.



Make sure that the sketch segment is extending away from the part. The length of the sketch segment is not important and does not need specified. The macro only looks at the direction of the sketch segment. Repeat the process for any remaining sketch points until each has a sketch segment attached and oriented. Exit the 3D Sketch.


[x_{distal_1} , y_{distal_1} , z_{distal_1}]
[-0.0018, 0.0182, 0.0003]
[x_{distal_2} , y_{distal_2} , z_{distal_2}]
[0.0164, 0.0166, -0.0007]

Save the Solidworks file. The macro works with both Solidworks part and assembly files. (1) Left click the newly created 3D Sketch in the feature tree. (2) While the sketch remains highlighted left click the ‘Run Macro’ button.



The macro directory will be opened. (3) Select the macro file Pt_Nml_JSON_export_Sldwks.swp. (4) Click 'Open.' The macro will then create the JSON file with the same name as the Solidworks file and in the same directory. No messages are displayed if the macro functions properly. If an error box is displayed, see the Appendix for support.

Opening the JSON file confirms that it was created properly. Check that there are sections for modelNormals and modelPoints. Ensure that the number of elements is the same for each array. Lastly check the modelNormals array to ensure there are no null vectors ([0, 0, 0]). A null vector usually indicates that a sketch point and sketch segment endpoint aren't coincident.

```
{
"modelNormals" : [
[0, 1, 0],
[0.707107, 0.70710678, 0]
],
"modelPoints" : [
[-0.00183, 0.01, 0.00029202],
[0.007378, 0.0076221, -0.00072788]
]
}
```

The final step is the verify that the sensors show up in the correct position and orientation on the 3D object. Refer to the "Visualizing an Existing Model" section in the **Simulation User Manual** for the proper procedure.

Appendix

Solidworks Export Tool Operation

The Solidworks JSON export macro tool was written in VisualBasic and uses the Solidworks API. It is not easily portable to other 3D CAD packages but the coding approach may prove useful. The macro takes information from a user selected Solidworks sketch and creates a JSON file for use with hmd_designer.

Preconditions

There are a number of preconditions required in order to run the script. The macro is designed to work with Solidworks 2016 but has been tested to work with versions from 2014-2016. Other versions may work but would have to be tested. Units should not matter as the JSON exporter will convert all values to meters. Millimeters and inches have been tested but other units should work as well. The user must have the Solidworks file open and saved. The file type can be a part (*.sldprt) or assembly (*.sldasm). Keep in mind when exporting that the origin for a part may not be the same as the part in an assembly. This may cause translation errors when visualizing the JSON and STL in hmd_designer. The Solidworks file must contain a sketch containing sketch points and sketch segments. The points should represent the sensor locations. One end of a line segment should attach to a point while the other indicates the viewing direction of the sensor. The sketch should contain only the sketch points and sketch segments being used. Unused points, construction lines, or other miscellaneous sketch elements may cause errors. Lastly, when the macro is run, the sketch should be selected.

Postconditions

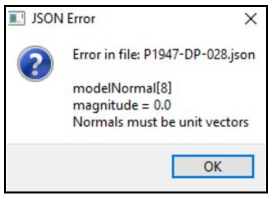
Running the macro will create a JSON file in the same name and directory as the Solidworks part. The JSON file will contain fully populated modelPoints and modelNormals arrays.

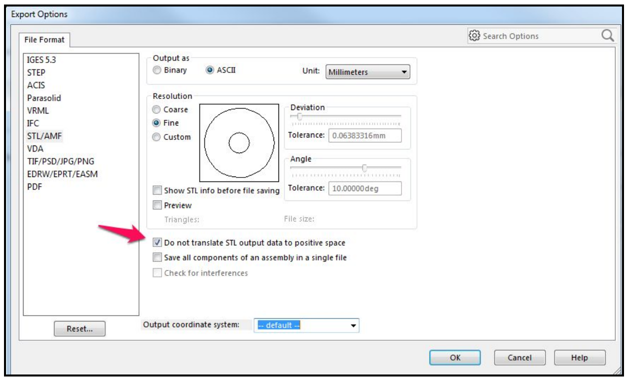
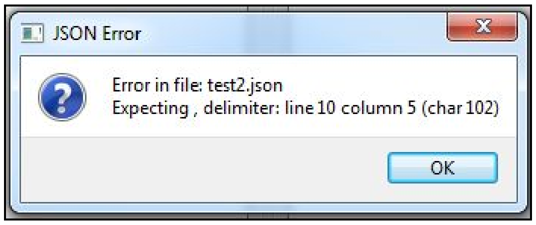
Overview of macro procedure

The macro uses the selected sketch from the Feature Tree and puts it into edit mode to access the sketch information. It then retrieves the sketch segments and puts them into an array whose length is defined by the number of points in the sketch. The macro then uses a for loop to step through each of the sketch segments. Another for loop is used to compare each end of a single sketch segment to all the sketch points. When a match is found, the opposite end of the sketch segment by definition defines the direction of the unit normal vector. The components of the sketch segment are normalized to one by dividing the x, y, and z-components by the sketch segment length. The matched point and normalized sketch segment components are saved to arrays.

The selected sketch in the Feature Tree is then taken out of edit mode. A text file is created with the ".json" filename extension. The arrays are then written to the modelPoints or modelNormals sections of the JSON file. The file is saved with the same name and in the same directory as the Solidworks file.

Solidworks Export Tool Error Guide

Error	**Probably Diagnosis**
Run-time error '91': Object variable or With block variable not set	The sketch is not selected in the Feature Tree when the macro is started.
Run-time error '9': Subscript out of range	The number of sketch points and sketch segments is not the same. The sketch may contain something other than just sketch segments and points. Ensure that any extraneous sketch points, construction lines, and miscellaneous sketch elements are removed.
An array in the modelNormals section of the JSON file contains a [0 0 0] null vector.	The ends of the sketch segments may not be coincident with the sketch points. Ensure that when a sketch segment is placed, it is not attaching to another sketch or feature. A good way to do this is to place the sketch segment without initially constraining it to anything. Then select the sketch point and sketch segment endpoint and apply the coincident constraint.
Error upon importing a JSON file into hmd_designer:  Error in file: P1947-DP-028.json modelNormal[8] magnitude = 0.0 Normals must be unit vectors	
Changes in CAD are not propagating to the JSON file when exported.	Make sure the Solidworks file is saved before the macro is run.
When visualizing the exported JSON and STL in hmd_designer the outside surface of the sensor is red instead of green	The sensor is facing the wrong direction. Ensure that the sketch segment in the Solidworks sketch is oriented in the correct direction.

Error	**Probably Diagnosis**
When visualizing the exported JSON and STL in hmd_designer the center of the sensor pattern is offset from the center of the object.	The export process may be translating the STL file. If using Solidworks, be sure to check the "Do not translate STL output data to positive space" in the 'Options…" window during saving.  Export Options File Format: STL/AMF Output as: ( ) Binary (●) ASCII — Unit: Millimeters Resolution: ( ) Coarse (●) Fine ( ) Custom Deviation — Tolerance: 0.06383316mm Angle — Tolerance: 10.00000deg ☐ Show STL info before file saving ☐ Preview ☑ Do not translate STL output data to positive space ☐ Save all components of an assembly in a single file ☐ Check for interferences Output coordinate system: -- default --
	This error can also occur when the STL file is generated from a different Solidworks file than the one the JSON was generated from. If the two files have different reference coordinates it will show up as an offset when visualized in hmd_designer. Ensure both the STL and JSON are generated from the same Solidworks file.
When visualizing the exported JSON and STL in hmd_designer a delimiter error occurs.  JSON Error Error in file: test2.json Expecting , delimiter: line 10 column 5 (char 102)	This error can be caused by using commas instead of periods in the decimal place. In the United States, a period is used to indicate the decimal place. In many other countries, a comma is used instead. The error can be resolved by adjusting the decimal symbol on your computer. In Windows 7: Start > Type "Region and Language" > Additional Settings... > Change the decimal symbol to "." > OK > OK In Windows 10: Start > Type "Region & language settings" > Additional date, time, & regional settings > Region > Additional settings… > Change the decimal symbol to "." > OK > OK

Solidworks Export Tool Code

```
'-------------------------------------------------------------------
' Preconditions:
' 1. User has Solidworks part open and saved.
' 2. Solidworks part contains a sketch with points and line segements.
' 3. Points should be placed in sensor location. Line segement should attach to point
and indicate direction of sensor viewing.
' 4. Lines can be made normal to the surface on which they lie by using a "normal to
face" constraint
' 5. Sketch must be single selected when the macro is run.
'
' Postconditions:
' 1. JSON file with same name as part file.
' 2. JSON file contains an array of points. There is a second array that contains the
unit normal vectors for those points.
'
'Possible Errors:
' 1. Run-time error '91': Object variable or With block variable not set
'       Sketch needs to be selected in Solidworks
' 2. [0 0 0] normal vectors returned in output.
'       Sketch segments and points are not completely coincident. Despite constraining
a segment endpoint to be coincident to a point,
'       they sometimes are not close enough to be registered by the macro to be
exactly the same. The best practice is to find the
'       offending point and sketch segment, remove all the constraints, drag the
endpoint and point away from each other, make them
'       coincident once again, constrain the two to the sensor location, and make the
line normal to the sensor surface.
' 3. Changes aren't being captured by the macro
'       Be sure to save the Solidworks file before running the macro.
'-------------------------------------------------------------------

Option Explicit
Sub main()
    Dim swApp As SldWorks.SldWorks
    Dim swModel As SldWorks.ModelDoc2
    Dim swSelMgr As SldWorks.SelectionMgr
    Dim swFeat As SldWorks.Feature
    Dim swSketch As SldWorks.Sketch
    Dim vSketchSeg As Variant
    Dim vSketchUserPt As Variant
    Dim SketchUserPtCount As Integer
    Dim i As Long
    Dim j As Long
    Dim vSketchUserPtX() As Double
    Dim vSketchUserPtY() As Double
    Dim vSketchUserPtZ() As Double
    Dim NormalVectorLengthX() As Double
    Dim NormalVectorLengthY() As Double
    Dim NormalVectorLengthZ() As Double
    Dim LineLengthNormalization As Double
    Set swApp = Application.SldWorks
    Set swModel = swApp.ActiveDoc
    Set swSelMgr = swModel.SelectionManager
    Set swFeat = swSelMgr.GetSelectedObject6(1, -1)
    Set swSketch = swFeat.GetSpecificFeature2
    Dim swSketchSeg As SldWorks.SketchSegment
    Dim swSketchLine As SldWorks.SketchLine
    Dim swSkStartPt As SldWorks.SketchPoint
    Dim swSkEndPt As SldWorks.SketchPoint
    Dim tFilename As String
    Dim sFilename As String
    Dim f As Variant
    Dim fs As Variant

    Debug.Print ""
    Debug.Print "----------------"
    Debug.Print "Beginning of Run"
    Debug.Print ""

    swModel.EditSketch 'Puts sketch into edit mode to get access to segment
information
    vSketchSeg = swSketch.GetSketchSegments 'gets all of the sketch segments in
swSketch
    SketchUserPtCount = swSketch.GetUserPointsCount 'grabs the number of user-defined
points in the sketch
    vSketchUserPt = swSketch.GetUserPoints2

    '%%%REDIMENSION LENGTH OF ARRAYS%%%
    ReDim vSketchUserPtX(UBound(vSketchSeg))
    ReDim vSketchUserPtY(UBound(vSketchSeg))
    ReDim vSketchUserPtZ(UBound(vSketchSeg))
    ReDim NormalVectorLengthX(UBound(vSketchSeg))
    ReDim NormalVectorLengthY(UBound(vSketchSeg))
    ReDim NormalVectorLengthZ(UBound(vSketchSeg))

    For i = 0 To UBound(vSketchSeg) 'individually steps through all the sketch
segments
        Set swSketchSeg = vSketchSeg(i)
        Set swSketchLine = swSketchSeg
        Set swSkStartPt = swSketchLine.GetStartPoint2 'gets starting point of sketch
        Set swSkEndPt = swSketchLine.GetEndPoint2 'gets ending point of sketch
        LineLengthNormalization = Sqrt(((swSkEndPt.X - swSkStartPt.X) ^ 2) +
((swSkEndPt.Y - swSkStartPt.Y) ^ 2) + ((swSkEndPt.Z - swSkStartPt.Z) ^ 2)) 'calculates
length of line
        'Debug.Print "  Start Point = " & swSkStartPt.X & ", " & swSkStartPt.Y & ", "
& swSkStartPt.Z
        'Debug.Print "  End Point = " & swSkEndPt.X & ", " & swSkEndPt.Y & ", " &
swSkEndPt.Z

        For j = 0 To SketchUserPtCount - 1 'sets up a loop that runs until the number
of user-defined points is counted. 1 is subtracted because j starts at 0
            vSketchUserPtX(j) = vSketchUserPt(8 * j + 5) 'grabs point x value for the
point from GetUserPoints2 array
            vSketchUserPtY(j) = vSketchUserPt(8 * j + 6) 'grabs point y value for the
point from GetUserPoints2 array
            vSketchUserPtZ(j) = vSketchUserPt(8 * j + 7) 'grabs point z value for the
point from GetUserPoints2 array

            If vSketchUserPtX(j) = swSkStartPt.X And vSketchUserPtY(j) = swSkStartPt.Y
```

```
And vSketchUserPtZ(j) = swSkStartPt.Z Then 'captures the case when a sketch line is
started on normal surface
                NormalVectorLengthX(j) = (swSkEndPt.X - swSkStartPt.X) /
LineLengthNormalization 'unit normal x value
                NormalVectorLengthY(j) = (swSkEndPt.Y - swSkStartPt.Y) /
LineLengthNormalization 'unit normal y value
                NormalVectorLengthZ(j) = (swSkEndPt.Z - swSkStartPt.Z) /
LineLengthNormalization 'unit normal z value

                Debug.Print "Point: x = " & vSketchUserPtX(j) & ", y = " &
vSketchUserPtY(j) & ", z = " & vSketchUserPtZ(j)
                Debug.Print "Unit vector: x = " & NormalVectorLengthX(j) & ", y = " &
NormalVectorLengthY(j) & ", z = " & NormalVectorLengthZ(j)

            End If

            If vSketchUserPtX(j) = swSkEndPt.X And vSketchUserPtY(j) = swSkEndPt.Y And
vSketchUserPtZ(j) = swSkEndPt.Z Then 'captures the case when a sketch line is ended on
a normal surface (not as likely)
                NormalVectorLengthX(j) = (swSkStartPt.X - swSkEndPt.X) /
LineLengthNormalization 'unit normal x value
                NormalVectorLengthY(j) = (swSkStartPt.Y - swSkEndPt.Y) /
LineLengthNormalization 'unit normal y value
                NormalVectorLengthZ(j) = (swSkStartPt.Z - swSkEndPt.Z) /
LineLengthNormalization 'unit normal z value

                Debug.Print "Point: x = " & vSketchUserPtX(j) & ", y = " &
vSketchUserPtY(j) & ", z = " & vSketchUserPtZ(j)
                Debug.Print "Unit vector: x = " & NormalVectorLengthX(j) & ", y = " &
NormalVectorLengthY(j) & ", z = " & NormalVectorLengthZ(j)
            End If

        Next j
    Next i
    swModel.InsertSketch2 False ' Exit edit mode and do not rebuild the model

    '%%%WRITING JSON SECTION%%%
    tFilename = swModel.GetPathName
    sFilename = Left(tFilename, Len(tFilename) - 7)
    Set fs = CreateObject("Scripting.FileSystemObject")
    Set f = fs.CreateTextFile(sFilename & ".json", True)

    f.writeline "{"
    f.writeline "" & Chr(34) & "modelNormals" & Chr(34) & " : ["
    For j = 0 To UBound(vSketchSeg)
        f.write "[" & Round(NormalVectorLengthX(j), 6) & ", " &
Round(NormalVectorLengthY(j), 8) & ", " & Round(NormalVectorLengthZ(j), 8) & "]"
'write unit normal components
        If Not j = UBound(vSketchSeg) Then
            f.write "," & vbCrLf
        End If
    Next j
    f.write "" & vbCrLf
    f.writeline "],"
    f.writeline "" & Chr(34) & "modelPoints" & Chr(34) & " : ["
    For j = 0 To UBound(vSketchSeg)
```

```
        f.write "[" & Round(vSketchUserPtX(j), 6) & ", " & Round(vSketchUserPtY(j), 8)
& ", " & Round(vSketchUserPtZ(j), 8) & "]" 'write point coordinates
        If Not j = UBound(vSketchSeg) Then
            f.write "," & vbCrLf
        End If
    Next j
    f.write "" & vbCrLf
    f.writeline "]"
    f.writeline "}"

    f.Close

End Sub
```