MFG124360

Exploring Toolpath and Probe Path Definition and Visualization in VR/AR

Zhihao Cui
Autodesk

Learning Objectives

- Learn how to export reusable geometries (models, tools, and so on) from PowerMill and PowerInspect
- Learn best practices for using PowerShape models in AR and VR environments
- Discover the potential when visualizing toolpaths and probe paths in AR and VR
- Rethink the potential change of workflow when using PowerMill and PowerInspect

Description

Visualizing and defining 3D models on a 2D screen has always been a challenge for CAD and CAM users. Tool paths and probe paths add other levels of complexity to take into consideration, as the user cannot fully appreciate the problem on a 2D viewer. Imagine yourself trying to define a tool axis on a complex shape—it's very hard to take every single aspect of the shape into account, except by guessing, calculating, and retrying repeatedly. With augmented reality (AR) and virtual reality (VR) technologies, the user gains the ability to inspect and define accurate 3D transformations (position and rotation) for machine tools in a much more natural way. We will demonstrate one potential workflow to address this during the class, which includes how to export relevant models from PowerMill software or PowerInspect projects; how to reconstruct, edit, and optimize models in PowerShape software and 3ds Max software; and eventually how to add simple model interactions and deploy them in AR/VR environments with game engines like Stingray or Unity.

Speaker

Zhihao is a Software Engineer in Advanced Consulting team within Autodesk. His focus for AR and VR technologies is in manufacturing industry and he wishes to continuously learn and contribute to it.

Data Preparation

The first step of the journey to AR or VR is generating the content to be visualized. Toolpath and probe path need to be put into certain context to be meaningful, which could be models for the parts, tools, machines or even the entire factory.

PowerMill Export

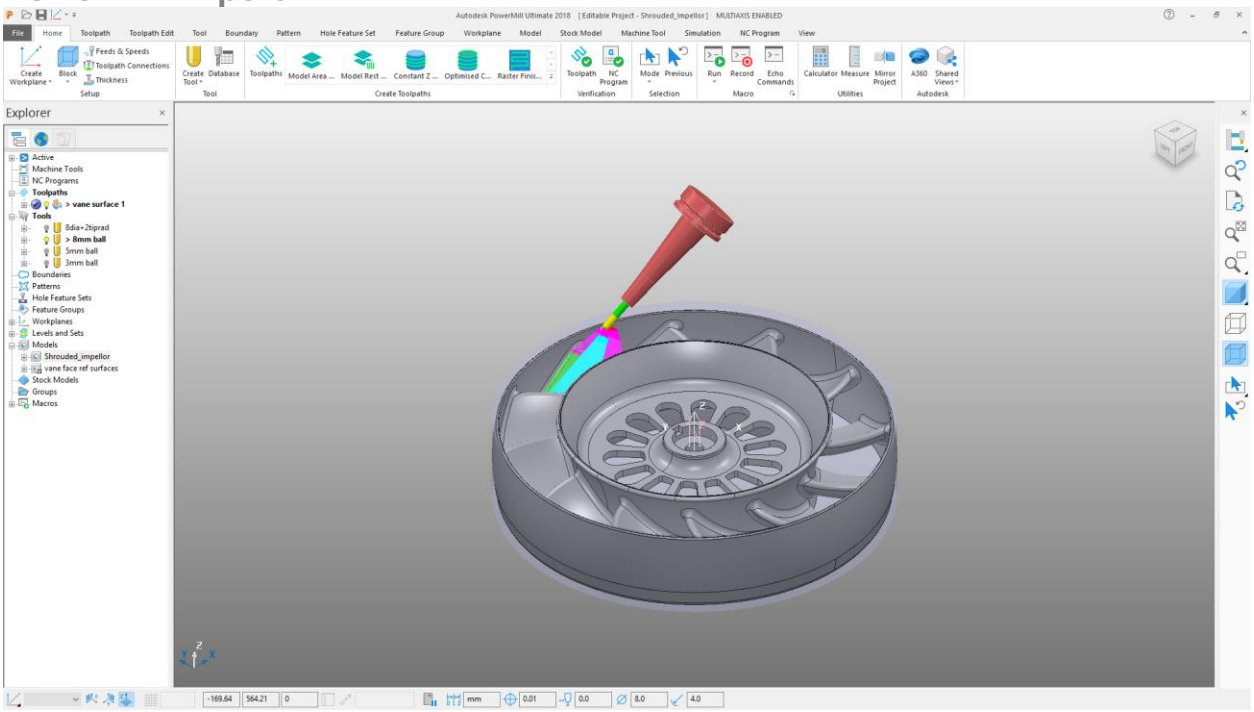

Figure 1 Typical PowerMill Project

Parts

Exporting models of the part is relatively simple.

1. Choose the part from the Explorer -> Models -> Right click on part name -> Export Model…
2. Follow the Export Model dialog to choose the name with *DMT* format[1].

[1] *DGK* file is also supported if additional CAD modification is needed later. See Convert PowerMill part mesh on page 7

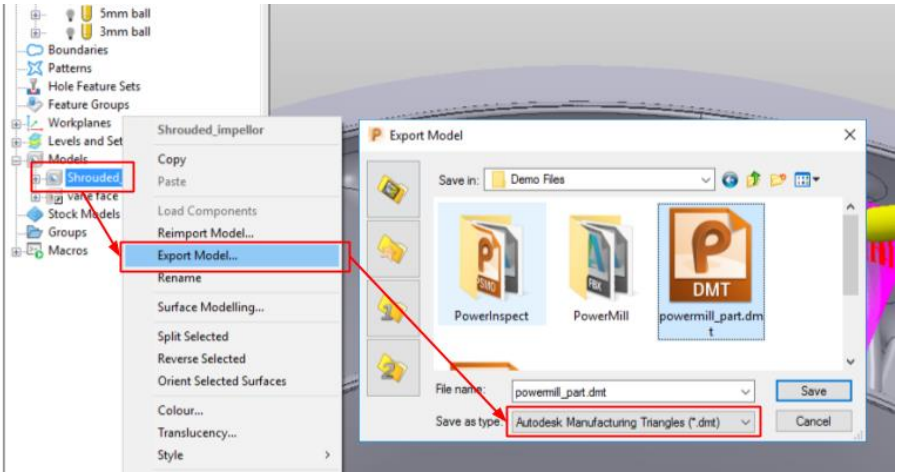

Figure 2 PowerMill – Export Model

Tool

Tool in PowerMill consists three parts – Tip, Shank and Holder.

To export the geometry of the tool, type in the macro commands shown in Figure 3, which would generate *STL* files[2] contains the corresponding parts. Three lines of commands[3] are used instead of exporting three in one file (See Figure 11), or one single mesh would be created instead of three which will make the coloring of the tool difficult.

```
EDIT TOOL ; EXPORT_STL TIP "powermill_tool_tip.stl"

EDIT TOOL ; EXPORT_STL SHANK "powermill_tool_shank.stl"

EDIT TOOL ; EXPORT_STL HOLDER "powermill_tool_holder.stl"
```

Figure 3 PowerMill Macro - Export Tool

Toolpath

Toolpath is the key part of the information generated by a CAM software. They are created based on the model of the part and various shapes of the tool for different stages (e.g. roughing, polishing, etc.). Toolpaths are assumed to be fully defined for visualization purposes in this class, and other classes might be useful around toolpath programming, listed on page 15.

Since there doesn't exist a workflow to directly stream data into AR/VR environment, a custom post-processor[4] is used to extract minimal information needed to describe a toolpath, i.e. tool tip position, normal direction and feed rate (its format is described in Figure 17).

The process is the same way as an NC program being generated for a real machine to operate. Firstly, create an NC program with the given post-processor shown in Figure 4. Then grab and drop the toolpath onto the NC program and write it out to a text file shown in Figure 5.

---

[2] `DDX` file format can also be exported if geometry editing is needed later in PowerShape

[3] The macro is also available in addition al material `PowerMill\ExportToolMesh.mac`

[4] The file is in additional material `PowerMill\simplepost_free.pmoptz`

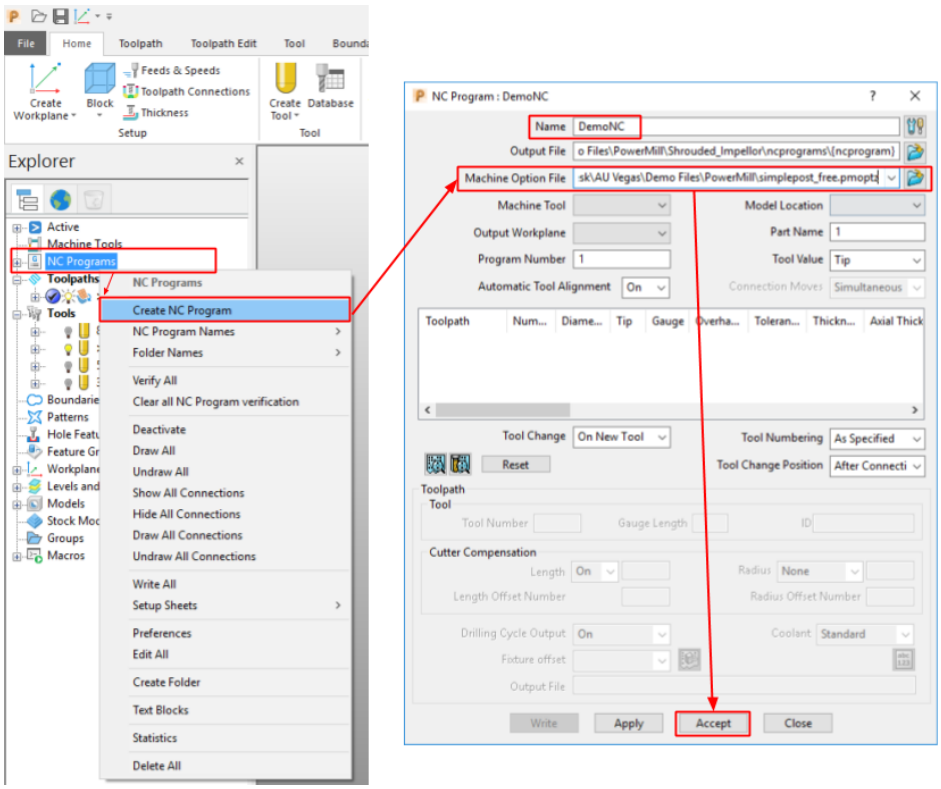

Figure 4 PowerMill Create NC Program

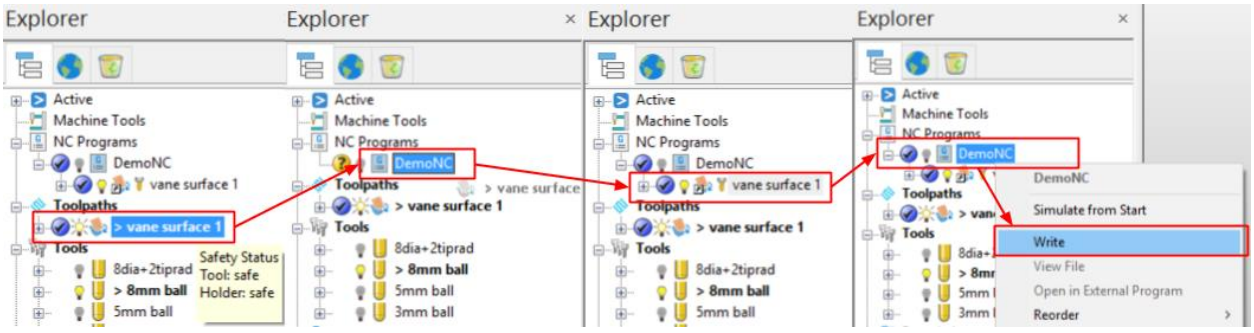

Figure 5 PowerMill Insert NC Program

PowerInspect Export

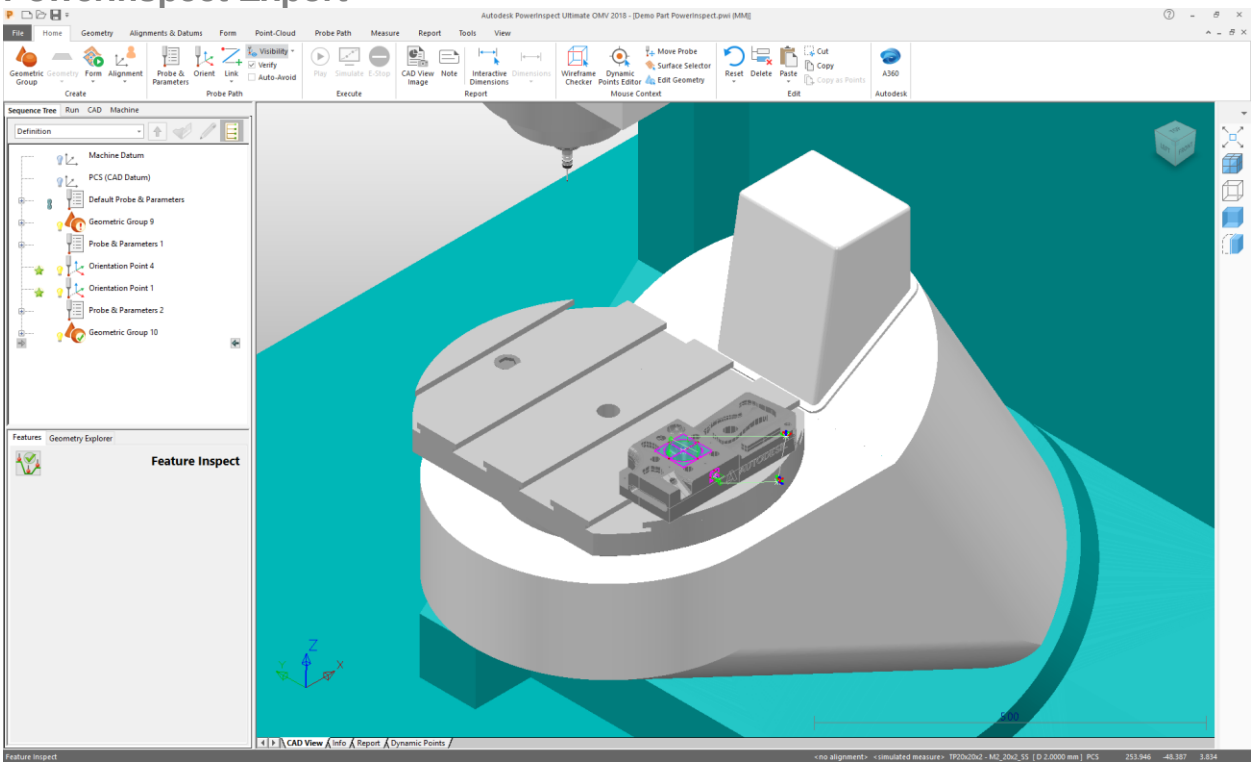

Figure 6 Typical PowerInspect OMV Project

CAD

CAD files can be found in the CAD tab of the left navigation panel. The model can be re-processed into a generic mesh format for visualization using PowerShape, which is discussed in Section Convert PowerMill part mesh on page 7.

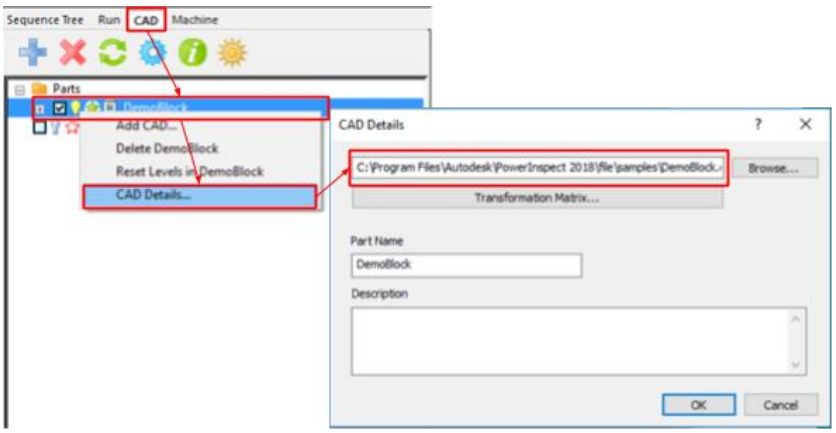

Figure 7 Find CAD file path in PowerInspect

Probe

Default probe heads are installed at the following location:

```
C:\Program Files\Autodesk\PowerInspect 2018\file\ProbeDatabaseCatalogue
```

Probes shown in PowerInspect are defined in `Catalogue.xml` file and their corresponding mesh files are in `probeheads` folder. These files will be used to assemble the probe mentioned in Section Model PowerInspect probe on page 9.

Probe tool

Although probe tool is defined in PowerInspect, they cannot be exported as CAD geometries to be reused later. In Model PowerInspect probe section on page 9, steps to re-create the probe tool will be introduced in detail based on the stylus definition.

Probe path

Like toolpath in PowerMill, probe path can be exported using post processor[5] to a generic MSR file format, which contains information of nominal and actual probing points, measuring tolerance, etc.

This can be achieved from Run tab -> NC Program, which is shown in Figure 8.

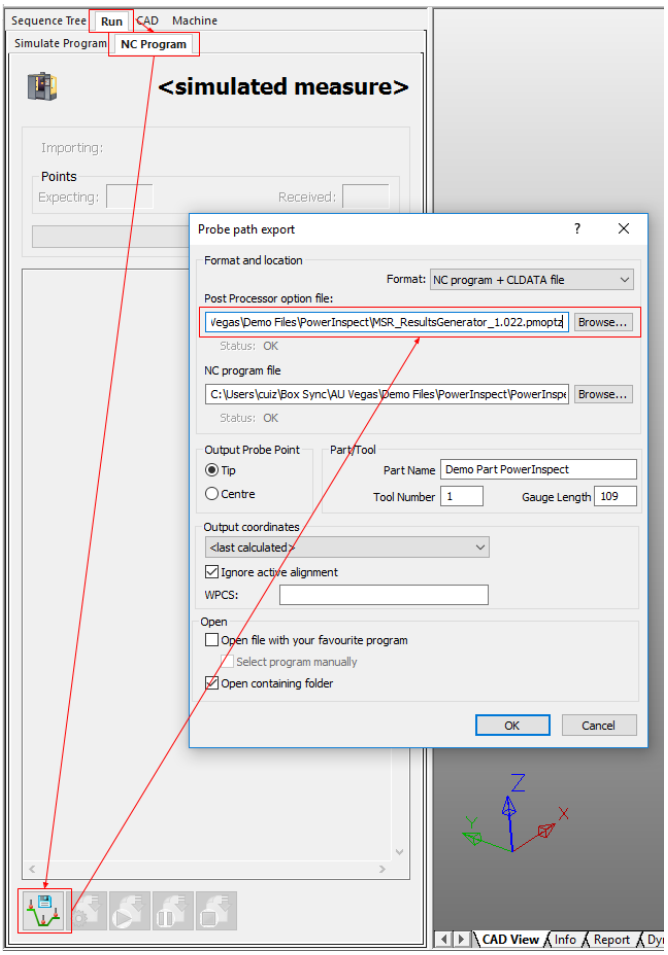

Figure 8 Export Probe Path from PowerInspect

[5] The file is in additional material `PowerInspect\MSR_ResultsGenerator_1.022.pmoptz`

Modelling using PowerShape

Convert PowerMill part mesh

DMT or *DGK* files can be converted to mesh in PowerShape to *FBX* format, which is a more widely adopted format.

DMT file contains mesh definition, which can be exported again from PowerShape after color change and mesh decimation if needed (discussed in Section Exporting Mesh in PowerShape on page 10).

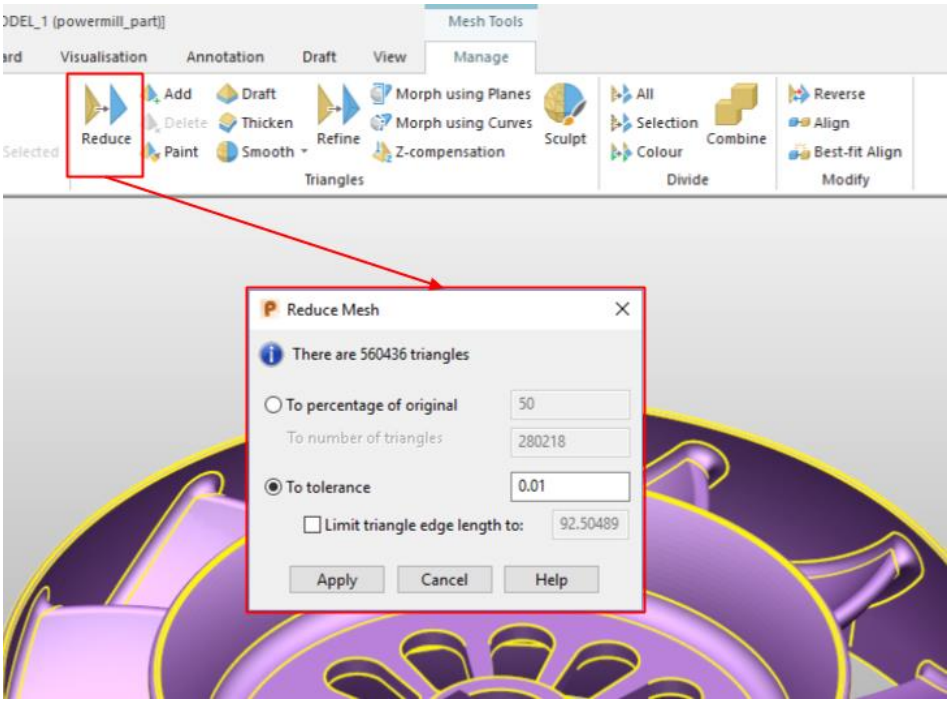

Figure 9 PowerShape reduce mesh

DGK file exported from PowerMill / PowerInspect is still parametric CAD model not mesh, which means further editing on the shape is made possible. Theoretically, the shape of the model won't be changed since the toolpath is calculated based on the original version, but further trimming operations could be carried here to keep minimal model to be rendered on the final device. For example, not all twelve blades of the impeller may be needed to visualize the toolpath defined on one single surface. It's feasible to remove ten out of the twelve blades and still can verify what's going on with the toolpath defined. After editing the model, PowerShape can convert the remaining to mesh and export to *FBX* format as shown below.

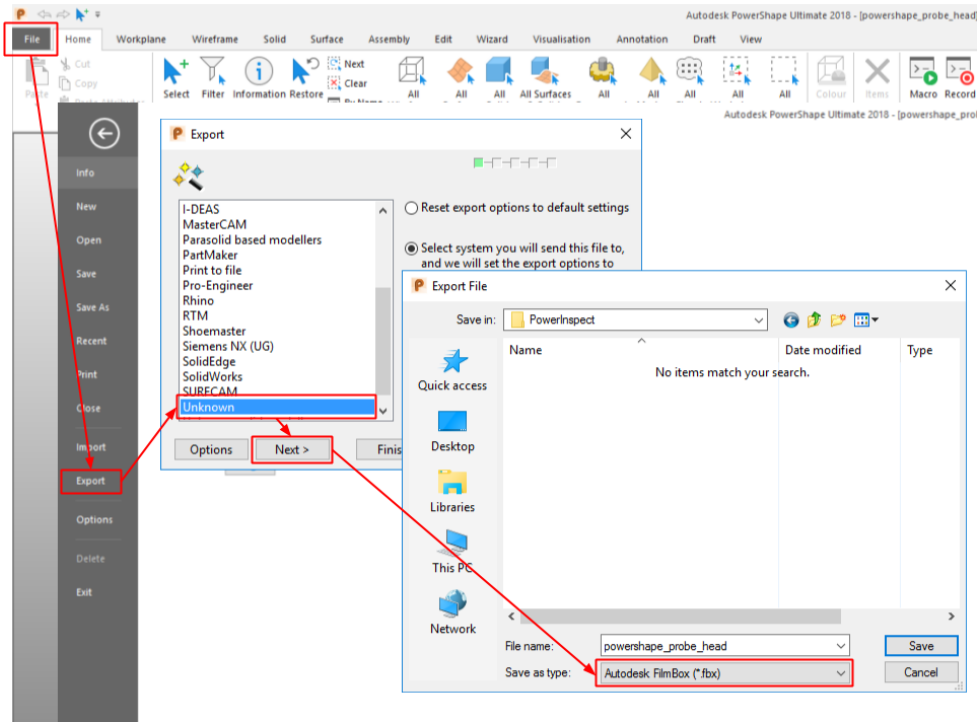

Figure 10 Export FBX from PowerShape

Model PowerMill tool

Import three parts of the tool's *STL* files into PowerShape, and change the color of individual meshes to match PowerMill's color scheme for easier recognition.

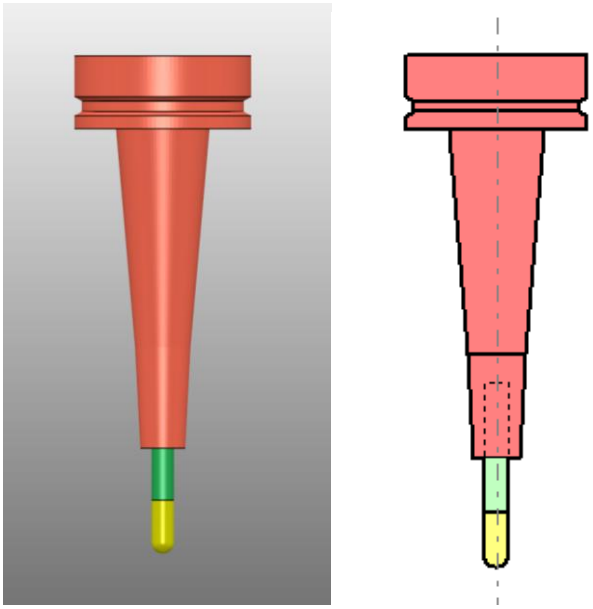

Figure 11 PowerShape Model vs PowerMill assembly view

Before exporting, move the assembled tool such that the origin is at the tool tip and oriented z-axis upwards, which saves unnecessary positional changes during AR/VR setup. Then follow Figure 10 to export *FBX* file from PowerShape to be used in later stages.

Model PowerInspect probe

Take the example Probe OMP400. `OMP400.mtd` file[6] contains where the mesh of individual components of the probe head are located and their RGB color. For most of the probe heads, *DMT* mesh files will be located in its subfolder. They can be dragged and dropped into PowerShape in one go to form the correct shape, but all in the same color (left in Figure 14). To achieve similar looking in PowerInspect, it's better to follow the definition file, and import each individual model and color it according to the `rgb` value one by one (right in Figure 14).

```
<!-- Head -->
<machine_part NAME="head">
  <model_list>
    <dmt_file>
      <!-- Comment !-->
      <path FILE="probeheads/OMP400/body.dmt"/>
      <rgb R="192" G="192" B="192"/>
    </dmt_file>
```

Figure 12 Example probe definition MTD file

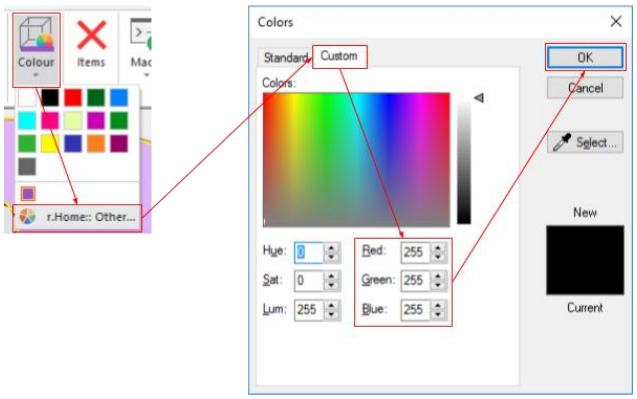

Figure 13 PowerShape apply custom color

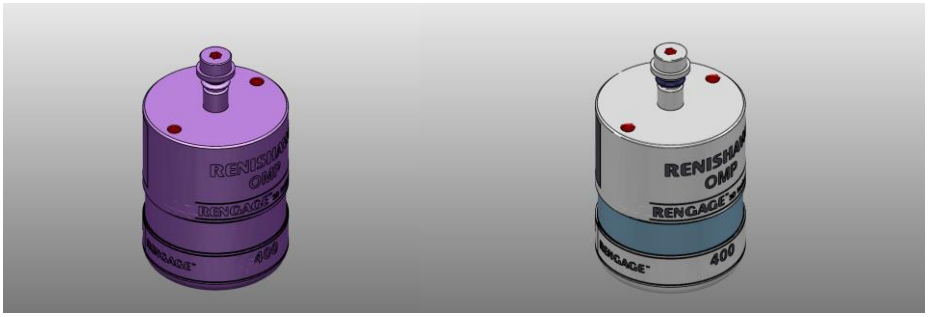

Figure 14 Before and after coloring probe head

For the actual probe stylus, it's been defined in `ProbePartCatalogue.xml` file. For the TP20x20x2 probe used in the example, `TP20` probe body, `TP20_STD` module and `M2_20x2_SS` stylus are used. Construct them one by one in the order of probe body, module and stylus, and each of them contains the definition like the below, which is the `TP20` probe body.

---

[6] `C:\Program Files\Autodesk\PowerInspect 2018\file\ProbeDatabaseCatalogue`

```
<ProbeBody name="TP20" from_mounting="m8" price="15.25" docking_height="0"
to_mounting="AutoMagnetic" length="17.5">
    <Manufacturer>Renishaw</Manufacturer>
    <Geometry>
        <Cylinder height="14.5" diameter="13.2" offset="0" reference_length="14.5"
material="Aluminium" color="#C8C8C8"/>
        <Cylinder height="3.0" diameter="13.2" offset="0" reference_length="3.0"
material="Stainless" color="#FAFAFA"/>
    </Geometry>
</ProbeBody>
```

Figure 15 Example TP20 probe body definition

Almost all geometries needed are cylinder, cone and sphere to model a probing tool. Start with the first item in the `Geometry` section, and use the parameters shown in the definition to model each of the geometries with solid in PowerShape and then convert to mesh. To make the result look as close as it shows in PowerInspect, `color` parameter can also be utilized (Google “color #xxx” to convert the hex color).

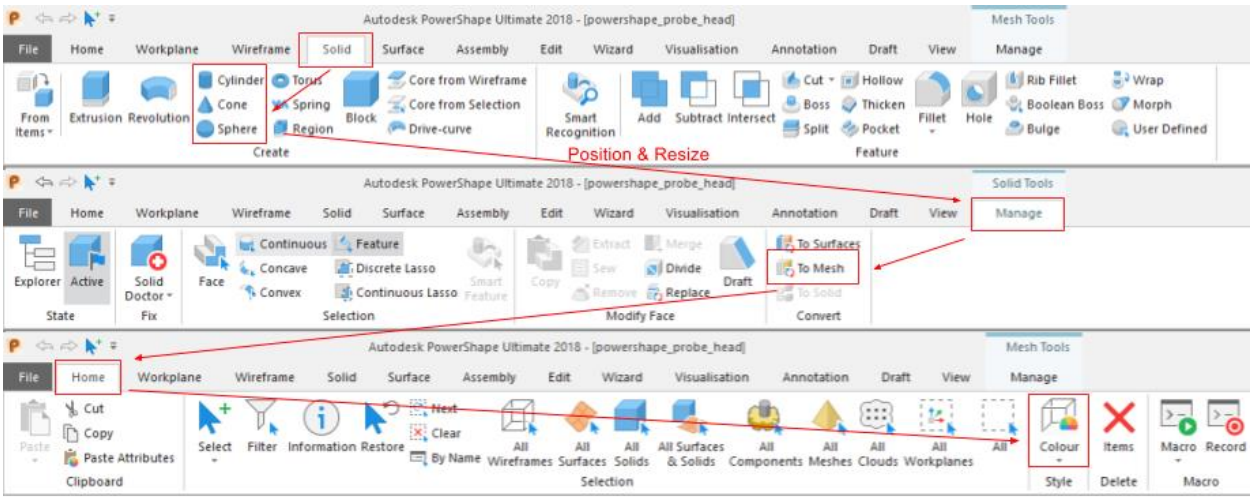

Figure 16 Model PowerInspect Tool

Unlike PowerMill tool, PowerInspect probe’s model origin should be set to the probe center instead of tip, which is defined in the `MSR` file. But the orientation should still be tuned to be z-axis facing upwards.

Discussions

Exporting Mesh in PowerShape

In PowerShape, there are different ways that a mesh can be generated and exported. Take the impellor used in PowerMill project as an example, the end mesh polycount is 786,528 if it’s been converted from surfaces to solid and then mesh with a tolerance set to 0.01. However, if the model was converted straight from surface to mesh, the polycount is 554,630, where the 30% reduce makes a big impact on the performance of the final AR/VR visualization.

Modifying the tolerance could be another choice. For visualization purposes, the visual quality will be the most impactable factor of choosing the tolerance value. If choosing the value is set too high, it may introduce undesired effect that the simulated tool is clipped into the model in certain position. However, setting the tolerance too small will quickly result in a ridiculous big mesh, which will dramatically slow down the end visualization.

Tolerance	Polycount
0.001	4,586,918
0.005	1,586,175
0.01	554,630
0.05	165,296

Table 1 PowerShape mesh Tolerance vs Polycount

Choosing the balance of the tolerance here mainly depends on what kind of end devices will the visualization be running on. If it will be a well-equipped desktop PC running VR, going towards a large mesh won't necessarily be a problem. On the other hand, if a mobile phone is chosen for AR, a low polycount mesh will be a better solution, or it can be completely ignored as a placeholder, which is discussed in Section On-machine simulation on page 12.

Reading data

Same set of model and paths data can be used in multiple ways on different devices. The easiest way to achieve this is through game engines like Stingray or Unity 3D, which has built-in support for rendering in VR environment like HTC Vive and mobile VR, and AR environment like HoloLens and mobile AR.

Most of the setup in the game engine will be the same for varies platform, like models and paths to be displayed. Small proportion will need to be implemented differently for each platform due to different user interaction availability. For example, for AR when using HoloLens, the user will mainly control the application with voice and gesture commands, while on the mobile phones, it will make more sense to offer on-screen controls.

For part and tool models, *FBX* files can be directly imported into the game engines without problem. Unit of the model could be a problem here, where export from PowerShape is usually in millimeter but units in game engines are normally in meters. Unit change in this case could result in a thousand times bigger, which may cause the user seeing nothing when running the application.

For toolpath data, three sets of toolpath information are exported from PowerMill with the given post-processor, i.e. tool tip position, tool normal vector and its feed rate. They can be read line by line, and its positions can be used to create toolpath lines. And together with the normal vector and feed rates, an animation of the tool head can be created.

```
Position(x,y,z)      Normal(i,j,k)      Feed rate

33.152,177.726,52.0,0.713,-0.208,0.67,3000.0
```

Figure 17 Example toolpath output from PowerMill

For probe path data, similar concept could be applied with an additional piece of information[7] – actual measured point, which means not only the nominal probe path can be simulated ahead of time, but also the actual measured result could be visualized with the same setup.

[7] See page 14 for MSR file specification.

```
START
G330 N0 A0.0 B0.0 C0.0 X0.0 Y0.0 Z0.0 I0 R0
G800 N1 X0 Y0 Z25.0 I0 J0 K1 O0 U0.1 L-0.1
G801 N1 X0.727 Y0.209 Z27.489 R2.5
END
```

Figure 18 Example probe path output from PowerInspect

Use cases

On-machine simulation

When running a new NC program with a machine tool, it's common to see the machine operator tuning down the feed rate and carefully looking through the glass to see what is happening inside the box. After several levels of collision checking in CAM software and machine code simulator, why would they still not have enough confidence to run the program?

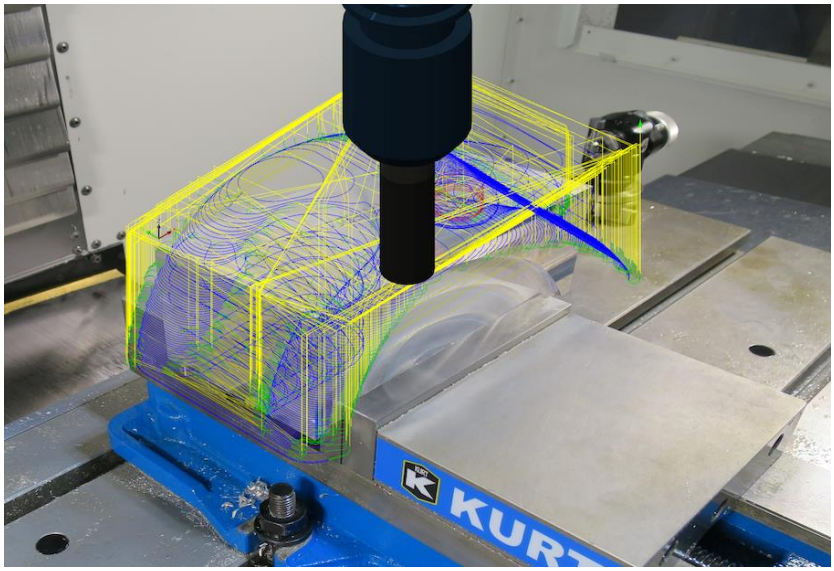

Figure 19 Toolpath simulation with AR by Hans Kellner @ Autodesk

One potential solution to this problem is using AR on the machine. Since how the fixture is used nowadays is still fairly a manual job constrained by operator's experience, variations of fixtures make it a very hard process to verify ahead of machining process. Before hitting the start button for the NC program, the operator could start the AR simulation on the machine bed, with fixtures and part held in place. It will become an intuitive task for the operator to check for collisions between part of the virtual tool and the real part and fixtures. Furthermore, a three-second in advance virtual simulation of the tool head can be shown during machining process to significantly increase the confidence and therefore leave the machine always running at full speed, which ultimately increases the process efficiency.

Toolpath programming assistance

Programming a toolpath within a CAM software can sometimes be a long iterative try and error process since the user always imagines how the tool will move with the input parameters. Especially with multi-axis ability, the user will often be asked to provide not only the basic parameters like step over values but also coordinate or direction in 3D for the calculation to start. Determining these 3D values on a screen becomes increasingly difficult when other

surfaces surround the places needed to be machined. Although there are various ways to let the user to navigate to those positions through hiding and sectioning, workarounds are always not ideal and time-consuming. As shown in Figure 20, there's no easy and intuitive way to analyze the clearance around the tool within a tight space, which is one of the several places to be considering.

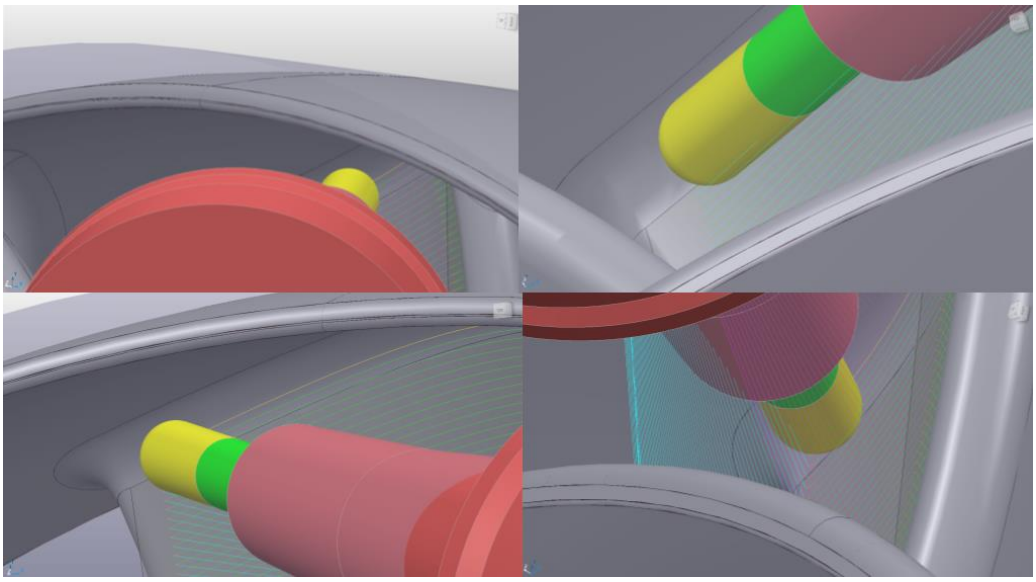

Figure 20 Different angles of PowerMill simulation for a 5-axis toolpath in a tight space

Taking the example toolpath in PowerMill, a user will need to recalculate the toolpath after each modification of the tool axis point, to balance between getting enough clearance[8] and achieving better machining result makes the user and verify the result is getting better or worth. However, this workflow can be changed entirely if the user can intuitively determine the position in VR. The tool can be attached to the surface and freely moved by hand in 3D, which would help to determine the position in one go.

Post probing verification

Probing is a common process to follow a milling process on a machine tool, making sure the result of the manufacturing is within desired tolerance. Generating an examination report in PowerInspect is one of the various ways to control the quality. However, what often happens is that if an out of tolerance position is detected, the quality engineer will go between the PC screen and the actual part to determine what is the best treatment process depending on different kind of physical appearance.

[8] Distance between the tool and the part

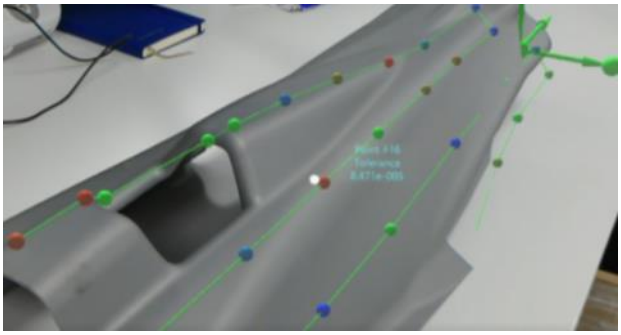

Figure 21 Overlay probing result on to a physical part

Overlaying probing result with AR could dramatically increase the efficiency by avoiding this coming back and forth. Same color coded probed point can be positioned exactly at the place of occurrence, so that the surrounding area can be considered separately. The same technique could also be applied to scanning result, as shown in Figure 22.

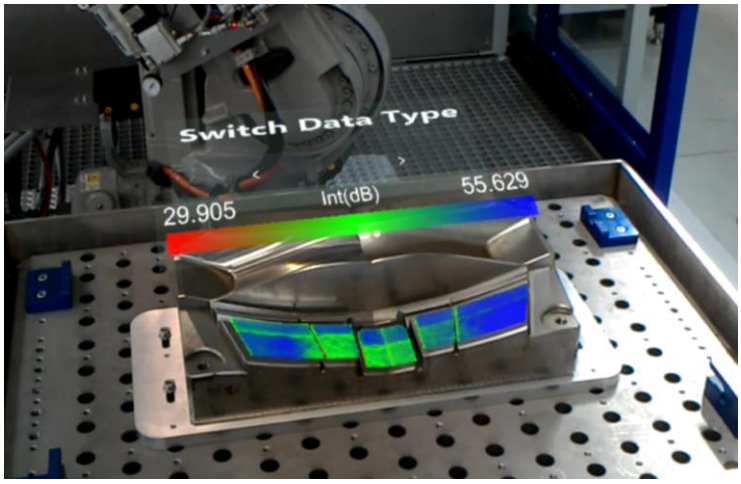

Figure 22 Overlaying scanning result on HoloLens by Thomas Gale @ Autodesk

Appendix

Reference Autodesk University classes

PowerMill

MFG12196-L: PowerMILL Hands on - Multi Axis Machining by GORDON MAXWELL

MP21049: How to Achieve Brilliant Surface Finishes for CNC Machining by JEFF JAJE

MSR File format[9]

G330 Orientation of the probe

G800 Nominal values

G801 Measured values

N Item number

A Rotation about the X axis

B Rotation about the Y axis

C Rotation about the Z axis

XYZ Translations along the X, Y and Z axes (these are always zero)

U Upper tolerance

L Lower tolerance

O Offset

I and R (in G330) just reader values

R (in G801) probe radius

[9] Credit to Stefano Damiano @ Autodesk