Model Builder
An Overture Tool for CAD Cleanup and Modification

William D. Henshaw,
Department of Mathematical Sciences,
Rensselaer Polytechnic Institute,
Troy, NY, USA, 12180.

Kyle K. Chand,
Anders Petersson

Centre for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA, 94551

April 18, 2015

Abstract

This document describes the `ModelBuilder` class. This class can be used to build geometric models, read, edit, repair and modify CAD geometries. Some of the features are

- read CAD geometries from IGES files.
- automatic detection of many errors in the CAD representation such as invalid trimming curves.
- repair CAD geometries through interactive editing of the trimming curves
- modify CAD geometries to remove features, ...
- automatically build the topology (connectivity) of the patched model, (closing gaps and removing overlaps), and constructing a water-tight representation of the CAD surface using a global triangulation.

Contents

1 Introduction

NOTE: This documentation is under construction.

2 Grids for a car rear view mirror (from an IGES file)

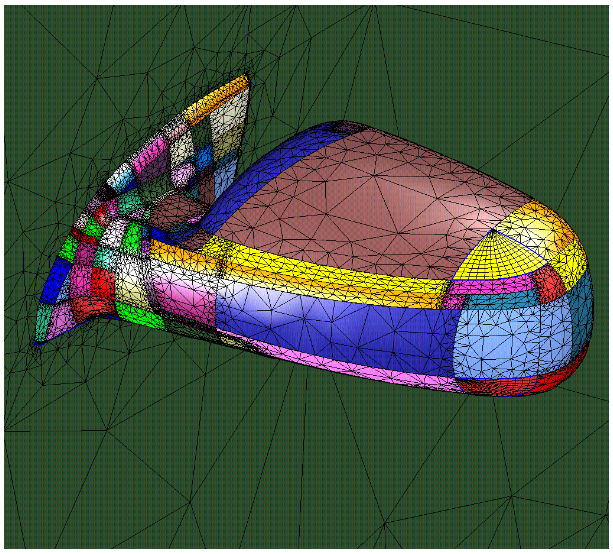


Figure 1: A CAD model for an automobile rear view mirror.

Figure 1 shows a CAD geometry for the rear view mirror of an automobile. We will use this example to illustrate CAD fixup and grid generation.

The CAD geometry for the rear view mirror has been provided courtesy of PSA Peugeot Citroen. Also thanks to Mr. Mehdi Bordji who generated some of the grids shown in this section.

Getting started: The first step is to read in the IGES file and look at all the surfaces:

```
 # the next command is needed by ogen but optional for mbuilder
 create mappings
   read iges file
     myFile.igs
   continue
   choose all
```

Here we assume you are running ogen or mbuilder.

Choosing a set of surfaces to work on: It can often be helpful to only look at a sub-set of the surfaces when getting started or when building a grid for one part of a large geometry. To choose a subset of surfaces one can

1. read in all surfaces (as above),
2. hide surfaces that are not wanted (using the mouse to pick individual or collections of surfaces),
3. delete hidden surfaces, (it is safer to hide first and then delete rather than delete to begin with),
4. ”print parameters” will give a list of the remaining surfaces that can be read in as a list as shown below

Reading in a list of surfaces: A list of surfaces can be read as follows,

```
 create mappings
   read iges file
     myFile.igs
   continue
choose a list
  6 7 8 20 22 70
done
```

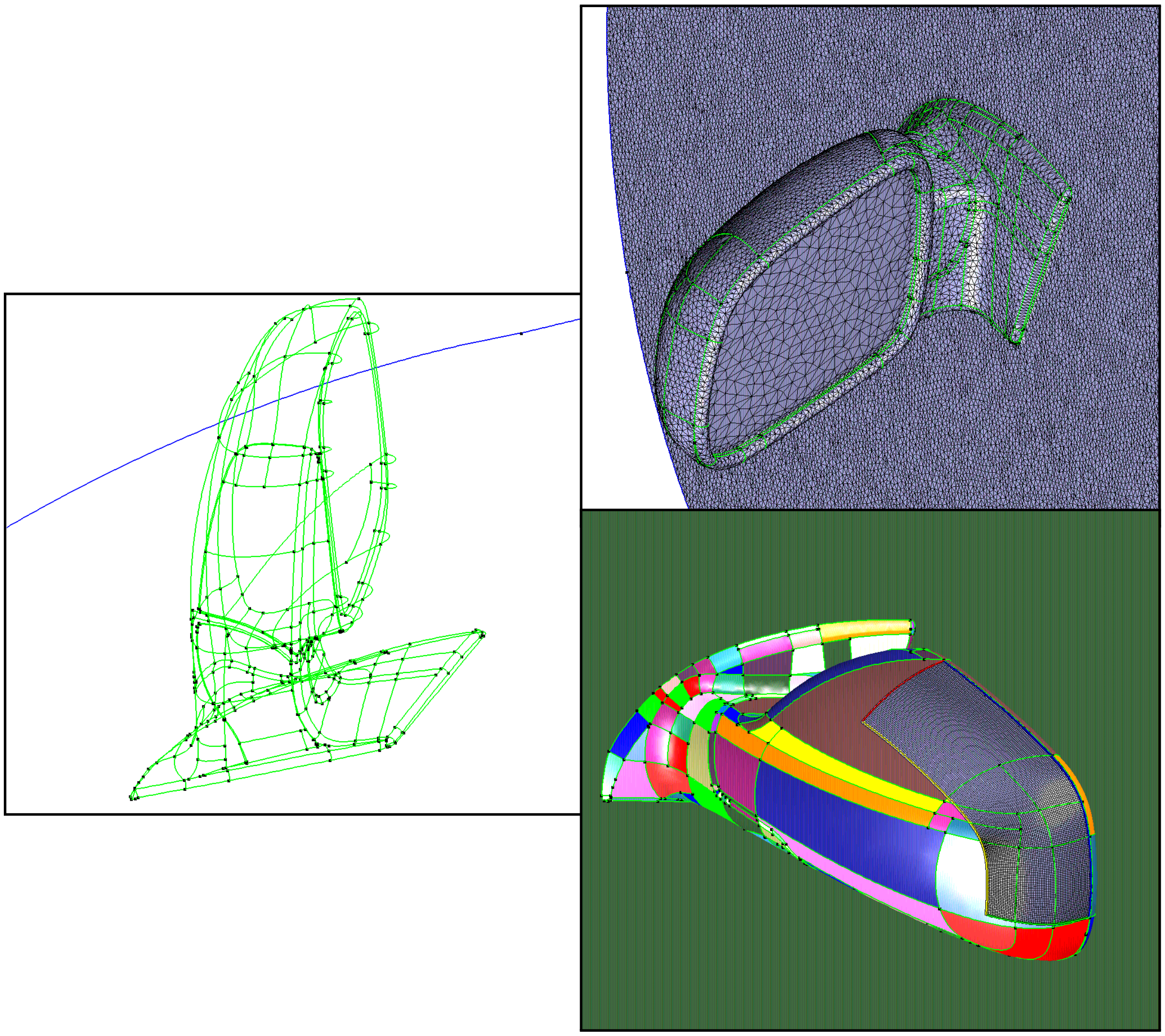


Figure 2: The topology (merged edge-curves, left), water-tight triangulation (top-right) and a surface grid generated on the CAD surface (bottom-right).

Fixing broken surfaces: The command `show broken surfaces` will print a list of any broken surfaces. To fix a broken surface, such as surface 5, type `examine a sub-surface 5`. See section 5 for examples of fixing trimming curves.

3 Building grids on triangulated surfaces

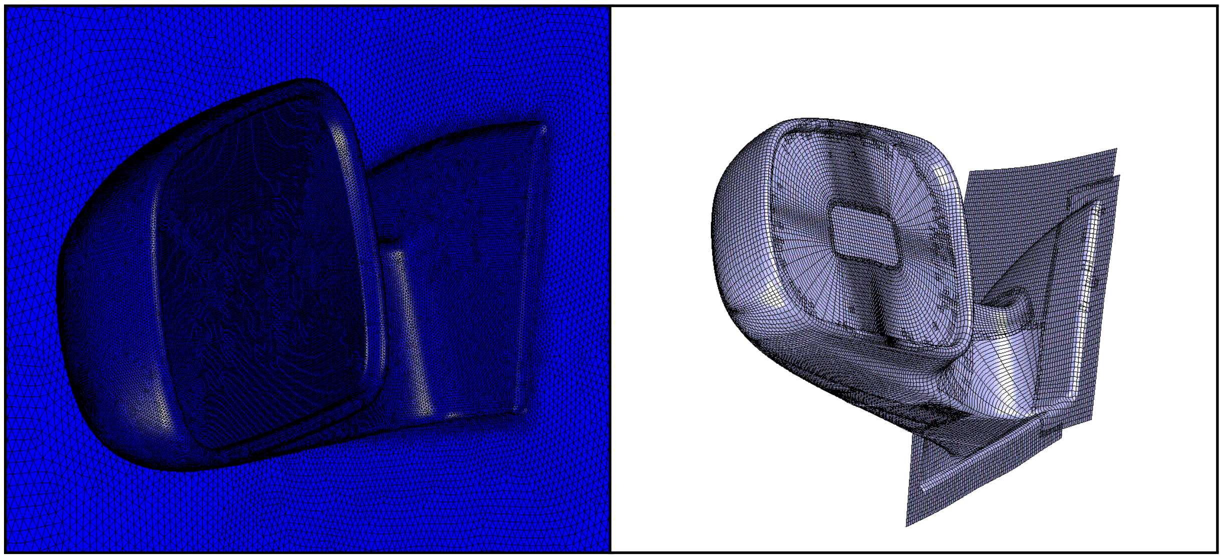


Figure 3: Left: triangulated surface for the automobile rear view mirror read from an AVS file. Right: Overlapping grids built on a triangulated surface constructed with mbuilder and the hyperbolic grid generator.

mBuilder can also be used to build grids directly on a triangulated surface. The triangulated surface can be defined in a number of ways such as an AVS or STL file. Figure 3 shows a triangulated surface read from an AVS file and represented as an UnstructuredMapping,

```
create mappings
  unstructured
    read avs file
    myFile.avs
  exit
```

Given a surface defined by a triangulation, the next step is to build structured surface grids on the surface. The hyperbolic grid generator is one way to build such grids. Figure 3 shows overlapping grids that were built on a surface.

```
  builder
    create surface grid...
```

To define a surface grid one must create a starting curve. This can be done by using the mouse to pick a set of points on the surface to define a curve. After interactively choosing points, the commands in the command file will look like

```
 initial curve:points on surface
   choose point on surface 0 2.277030e+01 -7.074806e+01 1.534555e+02 9.171783e-02 1.666090e-01
   choose point on surface 0 1.400569e+01 -6.563512e+01 1.329498e+02 6.320195e-01 3.748991e-02
   choose point on surface 0 5.563543e+00 -6.254250e+01 1.132309e+02 4.987929e-02 7.274973e-01
     ...
 done
```

The first three numbers are the x, y, z coordinates of the point. The last two numbers are related to the parameter space coordinates of the point. After creating the initial curve one can optionally edit the Mapping that defines the curve. In the next example we indicate that the initial curve is periodic:

```
edit initial curve
  periodicity
  2
exit
```

4 Build a surface of revolution

This section needs to be written.

5 Fixing errors in trimming curves

5.1 Example: Self-intersecting trimming curve

A trimming curve may be invalid if it intersects itself. One common way this can happen is near a cusp in the geometry. The CAD program may not care if the trim curve intersects itself near the end of the cusp since it is not visible. We must fix this problem, however, so that we can build a proper water-tight triangulation.

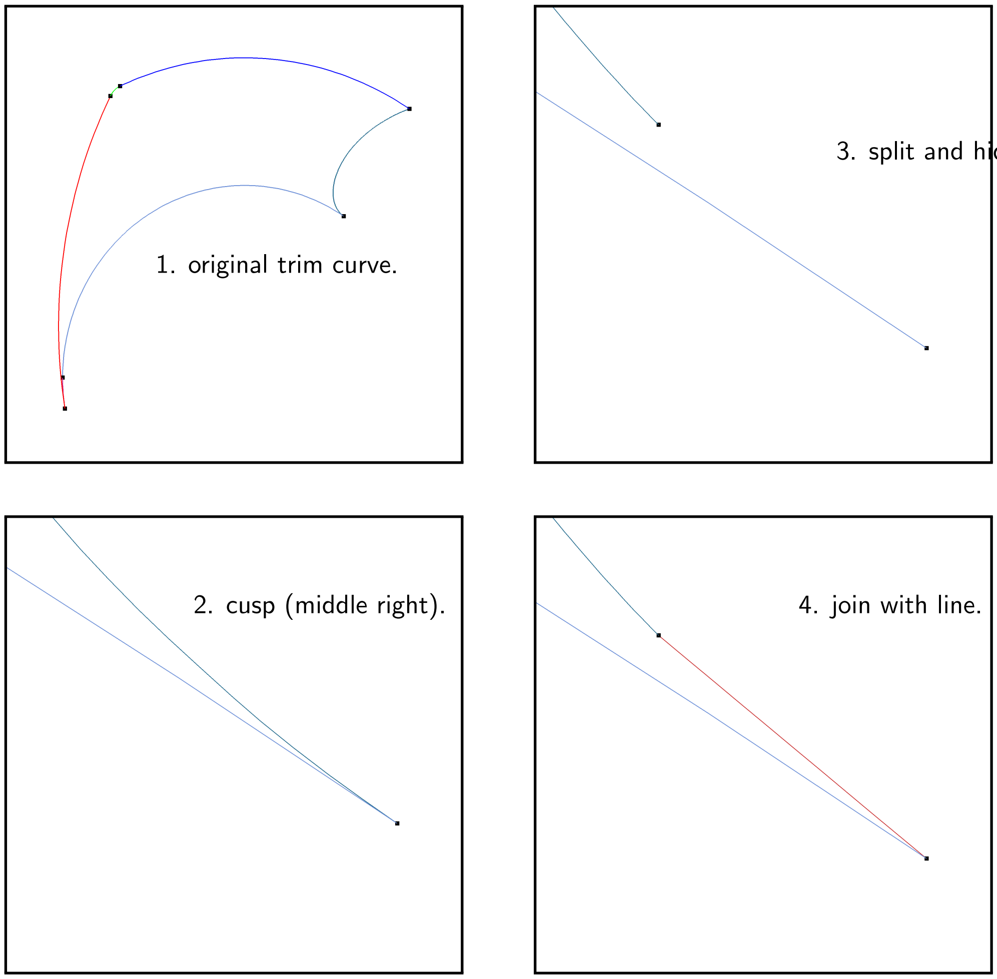


Figure 4: A trimming curve that self-intersects at a cusp can be fixed by splitting one of the curves near the cusp and replacing the small section with a straight line.

Figure 4 shows an example. One good way to fix this problem is to

1. split one of the curves near the cusp but before the intersection point (usually one should split the sub-curve that has more curvature).
2. hide the small portion of the split curve that contains the intersection.
3. Join the split sub-curve to the end-point of the cusp with a straight-line

5.2 Example: A trimming curves with missing and self-intersecting segments

In this example there are a number of mistakes in the trimming curves. Figure 5 shows the original trim curve (plot the sub-curves in difference colours) along with a gap between segments that is fixed by adding a line segment.

Figure 6 show two other problems where the trim curve intersects itself. These are fixed using the *snap-to-intersection* option.

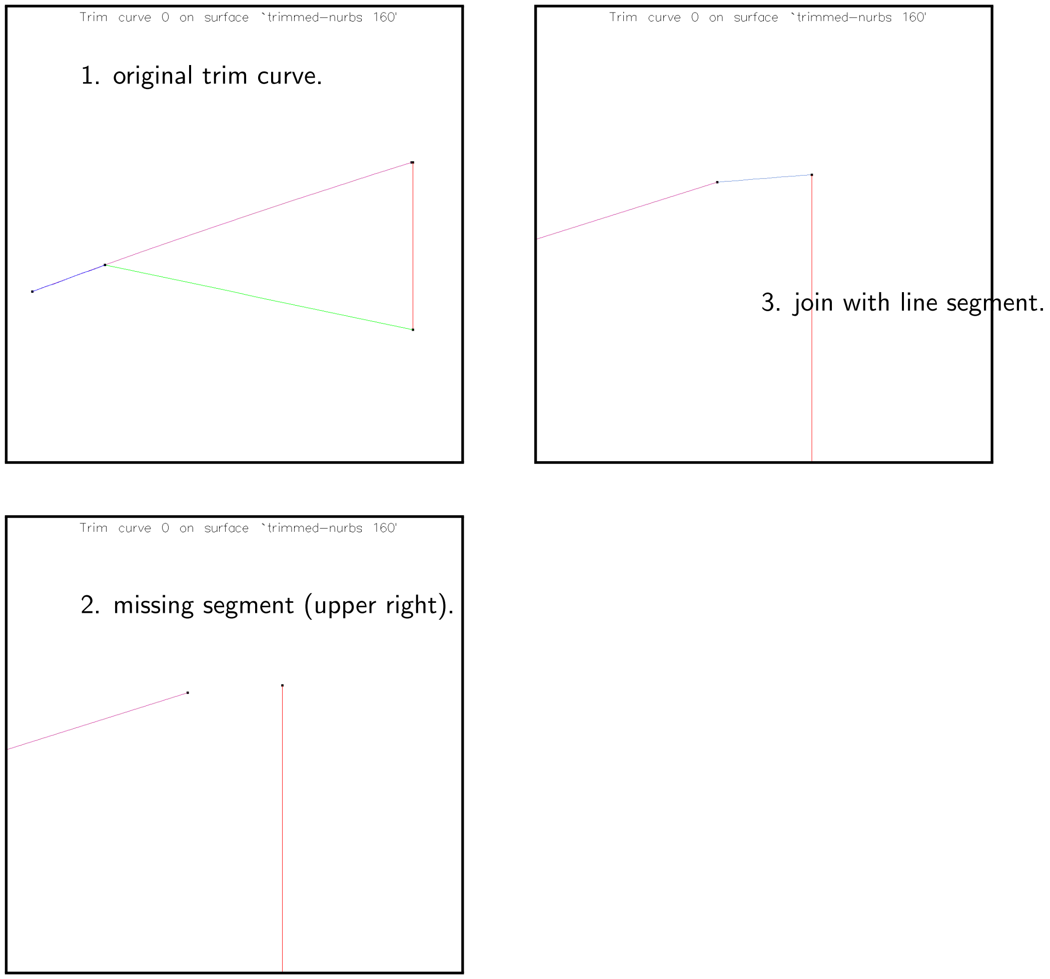

Trim curve 0 on surface `trimmed–nurbs 160'
1. original trim curve.
Trim curve 0 on surface `trimmed–nurbs 160'
3. join with line segment.
Trim curve 0 on surface `trimmed–nurbs 160'
2. missing segment (upper right).

Figure 5: A trimming curve that has a gap. The gap can be filled using the *join-with-line-segment* option and picking one adjacent curve and then the other.

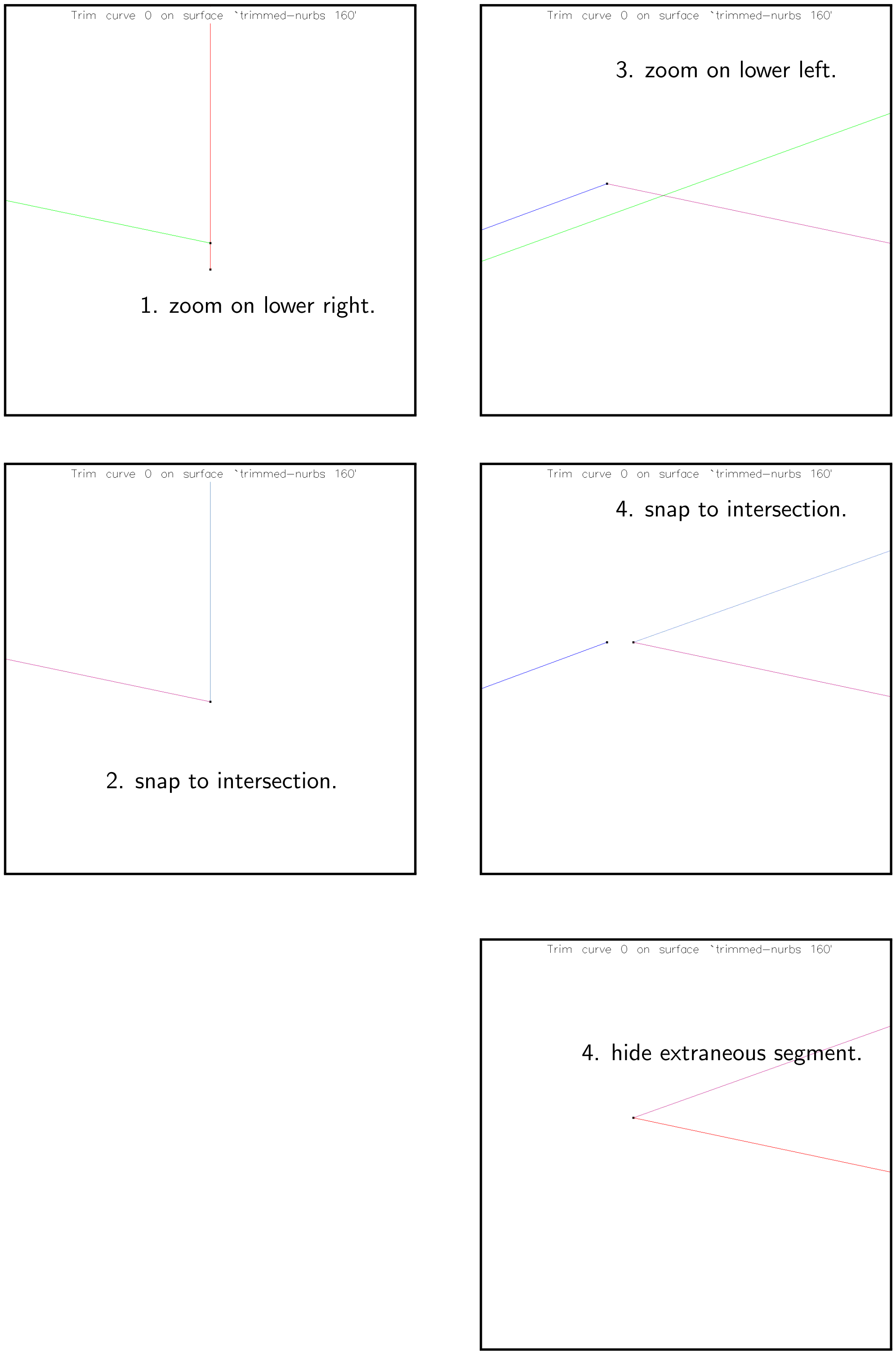

Trim curve 0 on surface `trimmed–nurbs 160'
1. zoom on lower right.
Trim curve 0 on surface `trimmed–nurbs 160'
3. zoom on lower left.
Trim curve 0 on surface `trimmed–nurbs 160'
2. snap to intersection.
Trim curve 0 on surface `trimmed–nurbs 160'
4. snap to intersection.
Trim curve 0 on surface `trimmed–nurbs 160'
4. hide extraneous segment.

Figure 6: The trimming curve has two self intersecting segments. These can be eliminated using the *snap-to-intersection* option and picking one adjacent curve and then the other. The end points of the adjacent curves will be changed to the point of intersection. The small left-over segments are hidden. Longer left-over segments are hidden explicitly.

6 Grid for a nozzle and cavity

In this section we demonstrate how to build an overlapping grid for a geometry of a converging diverging nozzle with a cavity at the throat. The geometry is defined through an IGES file (from CATIA). This geometry is courtesy of Philippe Lafon's group at EDF.

The ogen command file used for this example is called `nozzleAndCavity.cmd`. To start, the CAD geometry is read from an IGES file using the commands: read from an IGES file.

```
create mappings
  read iges file
    /Users/henshaw/res/Overture/ogen/vanne_simplifiee_catia2.igs
    continue
    choose all
```

At this points the CompositeSurface dialog appears and the CAD surface is plotted as in figure 7 (left). The CAD surface consists of a set of (trimmed) patches. The connectivity of the CAD surface is now determined using the topology function,

```
  CSUP:determine topology
    merge tolerance 0.001
    deltaS 0.01
    maximum area 1.e-4
    compute topology
  exit
```

The topology algorithm will merge matching edges from adjacent patches (using the *merge tolerance* to determine when this can be done. Each edge will be discretized with a set of points separated by a distance of approximately *deltaS* and a triangluation will be generated with a given *maximum area*. The triangulation is used later in the grid generation process as an aid in projecting points on to the original CAD patches.

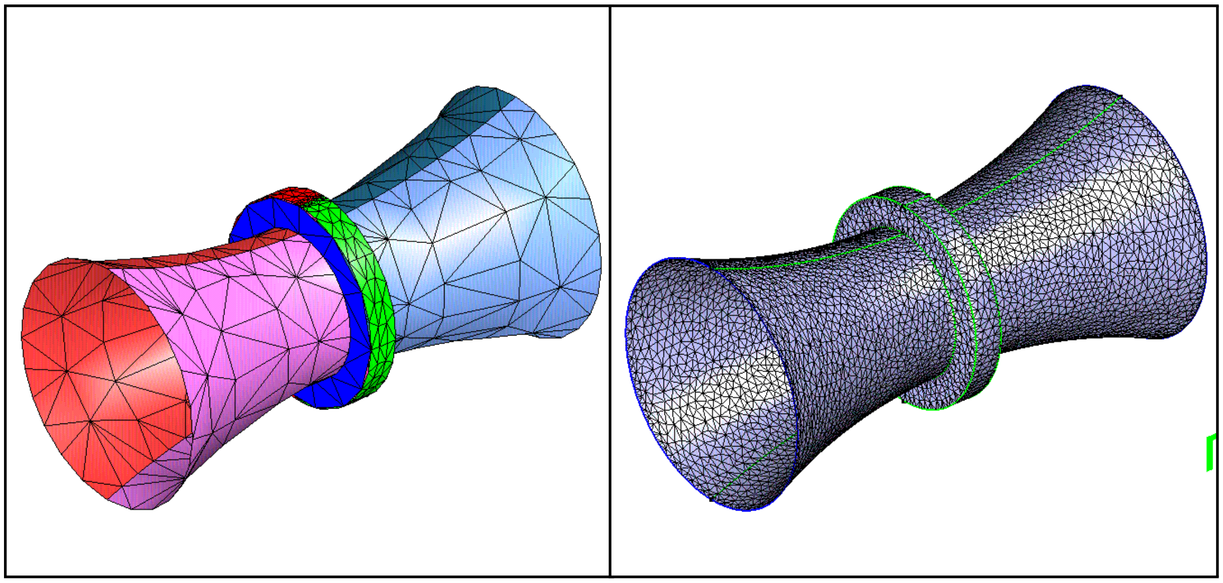


Figure 7: Nozzle and Cavity. Left: CAD geometry represented as a CompositeSurface Mapping. Right: water-tight triangulation generated by the Overture topology routine.

Upon exiting the CompositeSurface, the *builder* option is chosen to open the *Mapping Builder* dialog. The target grid spacing for all subsequent grids is set using the perl variable *$ds*.

```
  builder
    target grid spacing $ds $ds (tang,norm)((<0 : use default)
```

After choosing *create surface grid...* one enters the hyperbolic grid generator (*hype* for short). An initial curve is selected and a surface grid is grown for the left section of the nozzle as shown in figure 8.

```
# left nozzle:
create surface grid...
  choose boundary curve 1
  done
  $ndist=.46;
  $nx = int($ndist/$ds+1.5);
  lines to march $nx
  generate
  name left_nozzle_surface
exit
```

Hype is also used to grow a volume grid,

```
  create volume grid...
    marching options...
    BC: bottom fix x, float y and z
    BC: top fix x, float y and z
    backward
    lines to march $nn
    generate
    name left_nozzle
    Boundary Condition: bottom  3
    Boundary Condition: top     0
    Share Value: back    2
    Share Value: bottom  3
    exit
```

Similiar steps are used to build a surface and volume grid to connect the cavity with the nozzle as shown in figure 9, and grids for the cavity itself, figure 10.

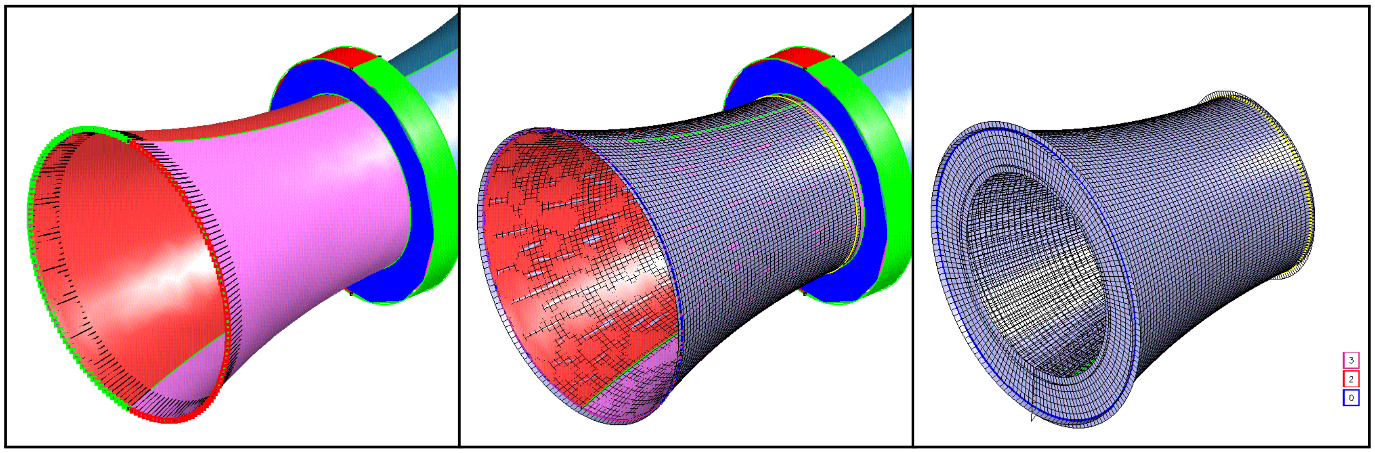


Figure 8: Nozzle and Cavity. Construct a grid for the left section of the nozzle boundary. Left: choose a start curve. Middle: grow a surface grid. Right: grow a volume grid.

The overlapping grid for the geometry, generated by ogen, is shown in figure 11

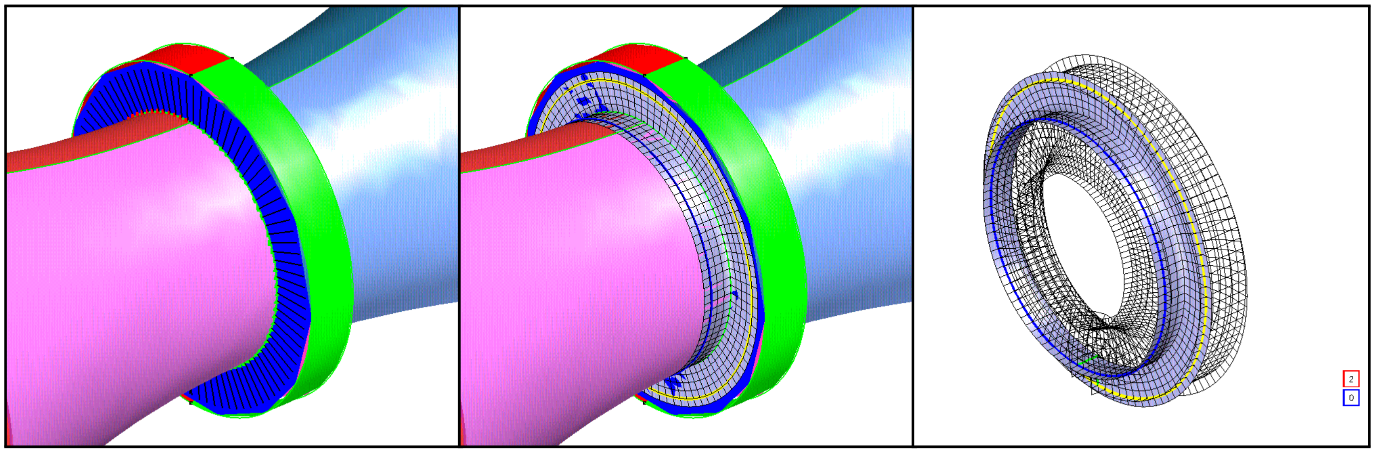


Figure 9: Nozzle and Cavity. Construct a grid to connect the nozzle and cavity (left-side). Left: choose a start curve by joining two edges. Middle: grow a surface grid in both directions from the start curve. Right: grow a volume grid.

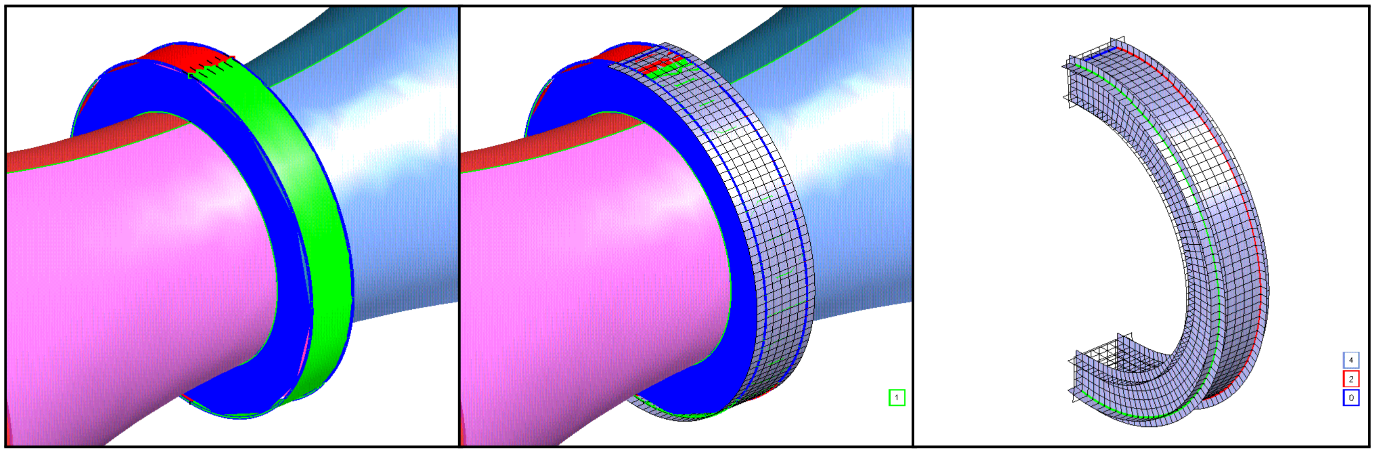


Figure 10: Nozzle and Cavity. Construct a grid for the cavity. Left: choose a start curve from an edge (boundary curves are highlighted in blue). Middle: grow a surface grid along the base of the cavity; the boundary conditions for this surface grid have been set to follow boundary curves that have been previously selected. Right: grow a volume grid.

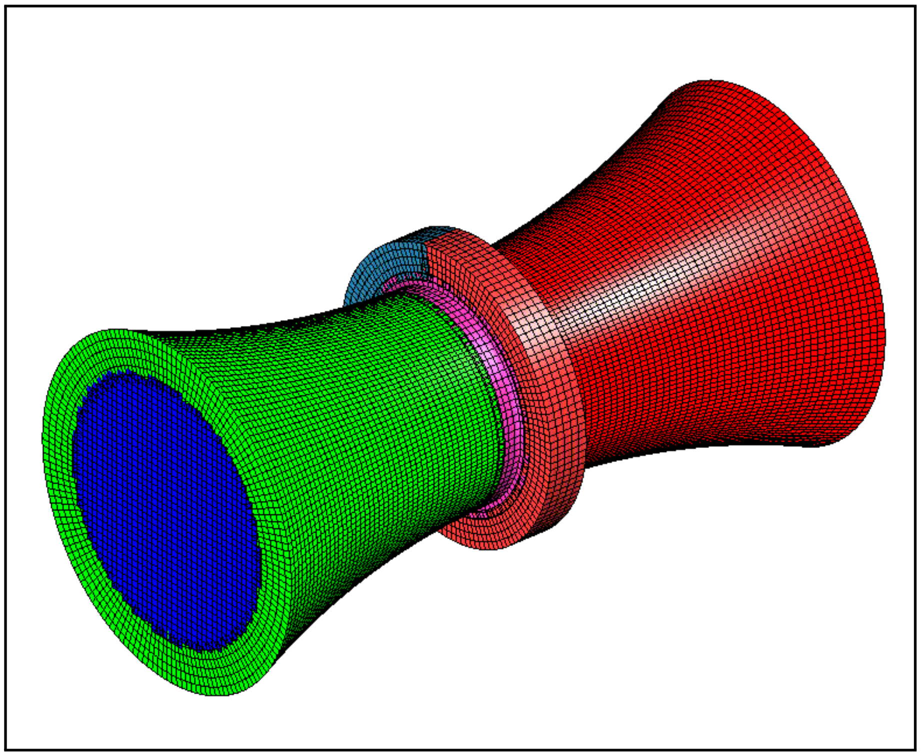


Figure 11: Nozzle and Cavity. Overlapping grid.

7 Overlapping grid for a space capsule

In this section we build an overlapping grid on a CAD geometry for a space capsule. The difficult issue here is that some of the CAD surfaces have singularities.

Figure 12 shows the overlapping grid.

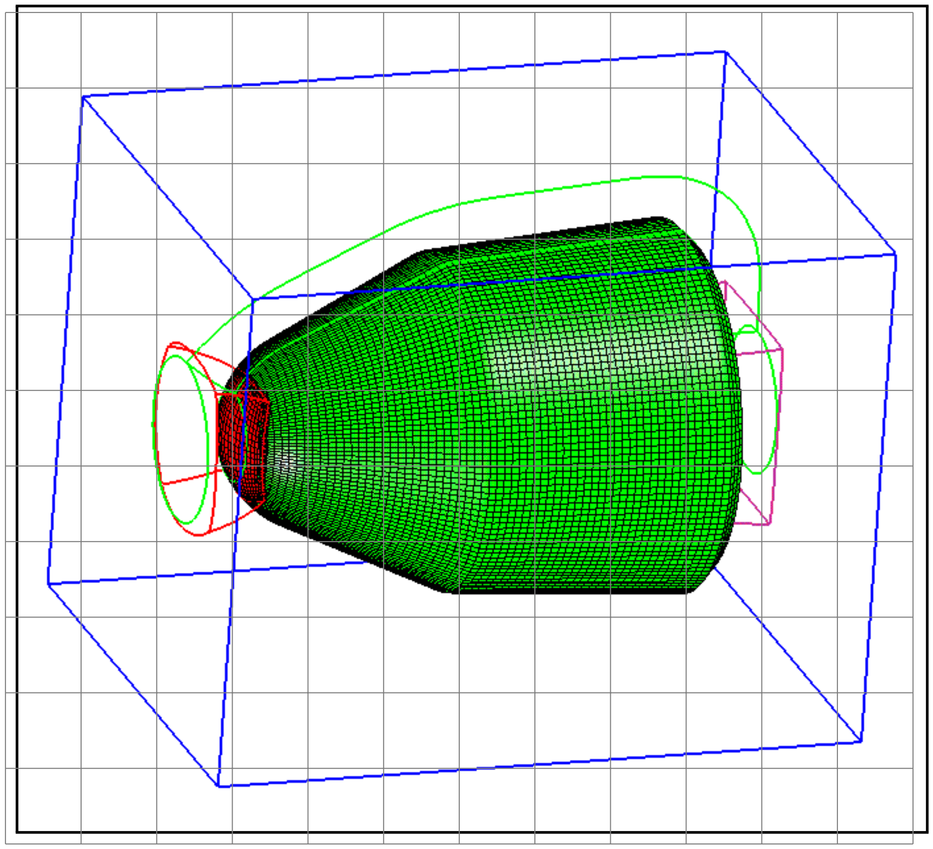


Figure 12: Space capsule overlapping grid.

8 Model Builder Class Member Functions

8.1 Constructor

ModelBuilder()

Description: Default constructor for thr ModelBuilder class

8.2 newModel

bool
newModel(GenericGraphicsInterface & ps, MappingInformation & mapInfo, CompositeSurface &model)

Description: Read a new model from a file.

8.3 addPlaneToModel

bool
addPlaneToModel(real planeCoordinates[3][3], int &planePoints, CompositeSurface &model,
GenericGraphicsInterface &gi)

Description: Given three points, build a plane and add it to the model.

8.4 editModel

void
editModel(MappingInformation &mapInfo, CompositeSurface & model, CompositeSurface & deletedSurfaces,
ListOfMappingRC &curveList, PointList & points)

Description: Edit a model.

8.5 simpleGeometry

void
simpleGeometry(MappingInformation &mapInfo, CompositeSurface & model, ListOfMappingRC &curveList,
PointList & points)

Description: Build a model from simple geometrical tools.

8.6 linerGeometry

int
linerGeometry(CompositeSurface & model, GenericGraphicsInterface& gi, PointList & points,
SphereLoading & sphereLoading)

Purpose: Define different liner shapes.

model (output): holds the liner generated by this routine

gi (input): Holds a graphics interface to use.

points (input/output) : holds existing points on input, plus any new points on output

sphereLoading (input/output): holds a distribution of sphere sizes and volume fractions on input. Holds the sphere centers and radii on output.

9 Appendix: Mouse and button commands

This section describes some of the features and properties of the Overture graphical user interface which is illustrated in figure 13. Some of these features are

- User defined dialog windows that can contain pulldown menus with push or toggle buttons, option menus, text labels (for inputting strings), push buttons, and toggle buttons.
- Multiple graphics windows and a single command window with a scrollable sub-window for outputting text, a command line, and a scrollable list of previous commands.
- Rotation buttons , , ,... which rotate the object on the screen about fixed x, y, and z axes (the x axis is to the right, the y-axis is up and the z-axis is out of the screen).
- Translation buttons , , $\otimes$, $\odot$,... which shift the object on the screen along a given axis. The two last buttons shift the object in and out of the screen, respectively. Since an othographic projection method is used in the graphics interface, these buttons only change the appearance when clipping planes are used.
- Push buttons for making the objects bigger: , or smaller: , and a reset button: to reset the view point, and a **clear** button to erase all objects on the screen.
- A push button to set the rotation center (after selecting this button, pick a point on the displayed grid or surface to use as a new rotation point).
- A **rubber band zoom** feature.
- Mouse driven **translate, rotate and zoom**.
- A pop-up menu that is active on the command and graphics windows. This menu is defined by the application (= user program).
- Static pull-down menus (**file**, **view**, and **help**) on the graphics windows. Here, the screen can be saved in different formats, clipping planes and viewing characteristics can be set, annotations can be made (not fully implemented), and some help can be found.
- Static pull-down menus (**file** and **help**) on the command window. Here command files can be read/saved, new graphics windows can be opened, the window focus can be set, and the application can be aborted.
- An optional pull-down menu (**My menu** in this case) that is defined by the application.
- Pushbuttons (**Pick 3D** and **Plot** in this case) that are defined by the application.
- A file-selection dialog box (not shown in the figure).
- The option of typing any command on the command line or reading any command from a command file. All commands can be entered in this fashion, including any pop-up or pull-down menu item or any of the buttons, **x+r:0**, **y-r:0**, **x+:0**, **y+:0**, **bigger:0**, etc. For the buttons, the :0 refers to the window number where the view should be modified, which in this case is window #0. Furthermore, when typing a command, only the first distinguishing characters need to be entered.
- Recording or retrieving a command sequence in a command file.

9.1 Using the mouse for rotating, scaling and picking

Rotations are performed with respect to axes that are fixed relative to the screen. The x-axis points to the right, the y-axis points upward and the z-axis points out of the screen. Rotations can either be performed about the centre of the window, or about a user defined point. This point can be set by first pressing the **set rotation point** icon on the graphics window and then clicking on the screen with the left mouse button. The rotation point can also be set by opening the "Set View Characteristics" dialog from the "View" pull-down menu (see section 9.3 for details). Note that the centre of the window is changed with the translation commands **x+**, **x-**, **y+**, etc.

Typing a rotation command with an argument (on the command line), such as **x+r:1 45**, will cause the view in window number 1 to rotate by 45 degrees about the x-axis. If no window number is given with the commands, such as, **y-y 30**, the command will apply to window 0.

Typing a translation command with an argument, such as **x+:0 .25** will cause the view in window number 0 to move to the right .25 units (in normalized screen coordinates; the screen goes from -1 to 1).

Rubber band zoom: The middle mouse button is used to ZOOM in. Press the middle button at one corner of a square, drag the mouse to another corner and lift the button. The view will magnify to the square that was marked. Use 'reset' to reset the view.

Mouse driven translate, rotate and zoom: All these operations are performed with the SHIFT key down. To translate, you hold the SHIFT key down, press the left mouse button and drag the cursor; the plotted objects will translate in the same direction as the mouse is moved. To rotate the view, you hold the SHIFT key down and press the middle mouse button and drag the cursor. Moving the cursor left or right will rotate about the y-axis (the vertical screen direction) and moving up or down will rotate about the x-axis (horizontal screen direction). To zoom in or out, you hold the SHIFT key down, press the right mouse button and drag the cursor vertically. To rotate about the z-axis, you press the left mouse buttons and drag the cursor horizontally.

Picking (aka selecting): While clicking the left mouse button, you select an object and get the (x, y, z) coordinate of the point on the object where you clicked. The object that was selected is reported in the selectionInfo data structure, which holds the global ID number, the front and back z-buffer coordinates and the window and 3–D coordinates. The global ID number can be used by the application to identify the selected object.

It is also possible to select several objects on the screen by specifying a rectangular region. To do this, you press the left mouse button in one corner of the rectangle and drag it to the diagonally opposite corner of the rectangle, where the mouse-button is released. The program will draw a rectangular frame to indicate the selected region. When you are happy with the selected region, you release the mouse button. An imaginary viewing volume is defined by translating the rectangular region into the screen and all objects that intersect the viewing volume are selected. However, only the 3–D coordinate of the closest object is computed. We remark that the object with the lowest front z-buffer value is the closest to the viewer. Also note:

1. Objects that are hidden by another object are also selected by this method.
2. The selection takes clipping planes (see below) into account, so it is not possible to select an object that has been removed by a clipping plane.

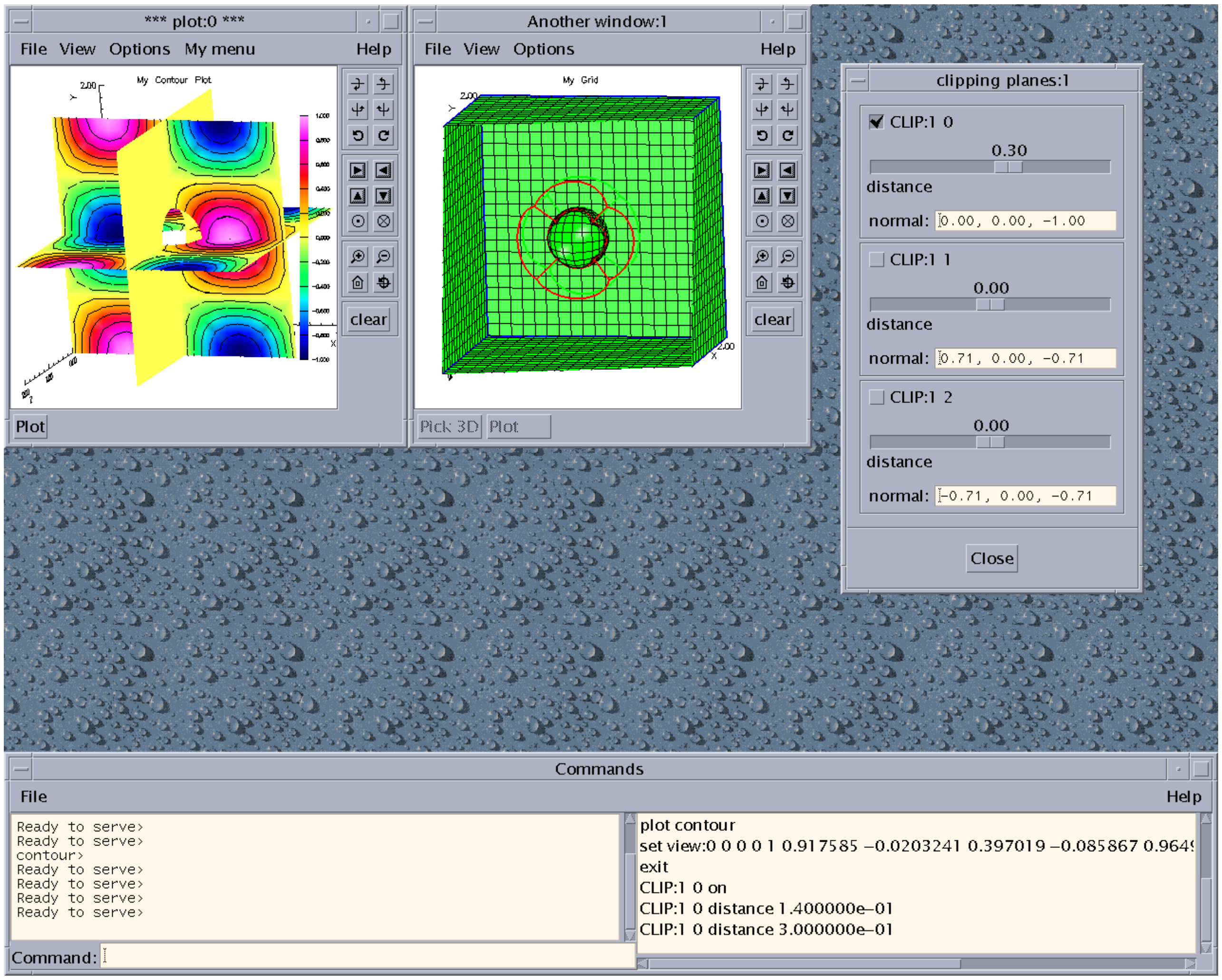
The mouse driven features are summarized in table 1.

9.2 Using clipping planes

The clipping plane dialog for each graphics window is opened from the 'View' pull-down menu on the menu bar in the graphics window. Figure 13 shows an example from the test program. The first clip plane is activated by clicking on the toggle button in the top left corner. After it is activated, we can look inside the cube and see the sphere. By dragging the slider bar for the clipping plane, the clipping plane is moved

Modifier	Mouse button	Function
	left	picking
	middle	rubber band zoom
	right	pop-up menu
<SHIFT>	left	translate
<SHIFT>	middle	rotate around the x & y axes
<SHIFT>	right	zoom (up & down) and z-rotation (left & right)

Table 1: Mouse driven features.



☑ CLIP:1 0
0.30
distance
normal: 0.00, 0.00, -1.00

☐ CLIP:1 1
0.00
distance
normal: 0.71, 0.00, -0.71

☐ CLIP:1 2
0.00
distance
normal: -0.71, 0.00, -0.71

```
Ready to serve>
Ready to serve>
contour>
Ready to serve>
Ready to serve>
Ready to serve>
Ready to serve>
```

```
plot contour
set view:0 0 0 0 1 0.917585 −0.0203241 0.397019 −0.085867 0.964
exit
CLIP:1 0 on
CLIP:1 0 distance 1.400000e−01
CLIP:1 0 distance 3.000000e−01
```

Command:

Figure 13: The Overture graphics interface. The clipping plane dialog window is also shown

closer or further away from the eye. The direction of the normal of the clipping plane can also be changed by editing the numbers in the 'Normal' box.

9.3 Setting the view characteristics

The view characteristics dialog for each graphics window is opened from the 'View' pull-down menu on the menu bar in the graphics window. Figure 14 shows an example from the test program. NOTE: When entering numerical values into a text box, it is necessary to hit <RETURN> before the changes take effect.

Here follows a brief description of the functionality in this window:

Background colour: Select a background colour from the menu by clicking on the label with the current colour. In this example, the background colour is white. Changing the background colour will take effect immediately.

Text colour: Select a text (foreground) colour from the menu by clicking on the label with the current colour. In this example, the text colour is steel blue. Note that the text colour is used to colour the axes, the labels, and sometimes also the grid lines. However, only the axes will change colour immediately after a new colour is chosen. To update the colour of the labels and the grid lines, it is necessary to replot the object on the screen.

Axes origin: Click on the radio buttons to set the origin of the coordinate axes either at the default location (lower, left, back corner of the bounding box), or at the rotation point.

Rotation point: Enter the (X, Y, Z) coordinates of the rotation point. The rotation point will remain fixed to the screen during both interactive rotation with <SHIFT>+middle mouse button, and during rotation with the buttons **x+r**, **y+r**, etc.

Pick rotation point: After clicking on this button, set the rotation point by clicking with the left mouse button on a point on an object in the graphics window. NOTE:

1. Picking will only work in the window from which the view characteristics dialog was opened. If you are unsure which window to pick in, you can press the right mouse button and read the window number from the title of the popup menu.

Lighting: Activate/deactivate lighting in the graphics window.

Light #i: Turn on/off light source number i, $i = 0, 1, 2$. Turning off all light sources is the same as deactivating lighting with the above function.

Position (X, Y, Z): The location (X, Y, Z) of light source number i.

Ambient (R, G, B, A): The ambient colour (R, G, B, A) of light source number i.

Diffusive (R, G, B, A): The diffusive colour (R, G, B, A) of light source number i.

Specular (R, G, B, A): The specular colour (R, G, B, A) of light source number i.

X-colour material properties: The following properties characterize the default material, which is used when a X-colour is chosen for a lit object. The X-colours are distinguished from the predefined "special" materials in that only their ambient and diffuse reflections are defined, but not their specular and shininess properties. (For those fluent in OpenGL, this functionality is obtained by calling the function glColorMaterial with the argument GL_AMBIENT_AND_DIFFUSE.) Note that the reflective properties only influence objects that are lit. Also note that the objects need to be replotted in order for the changes to take effect.

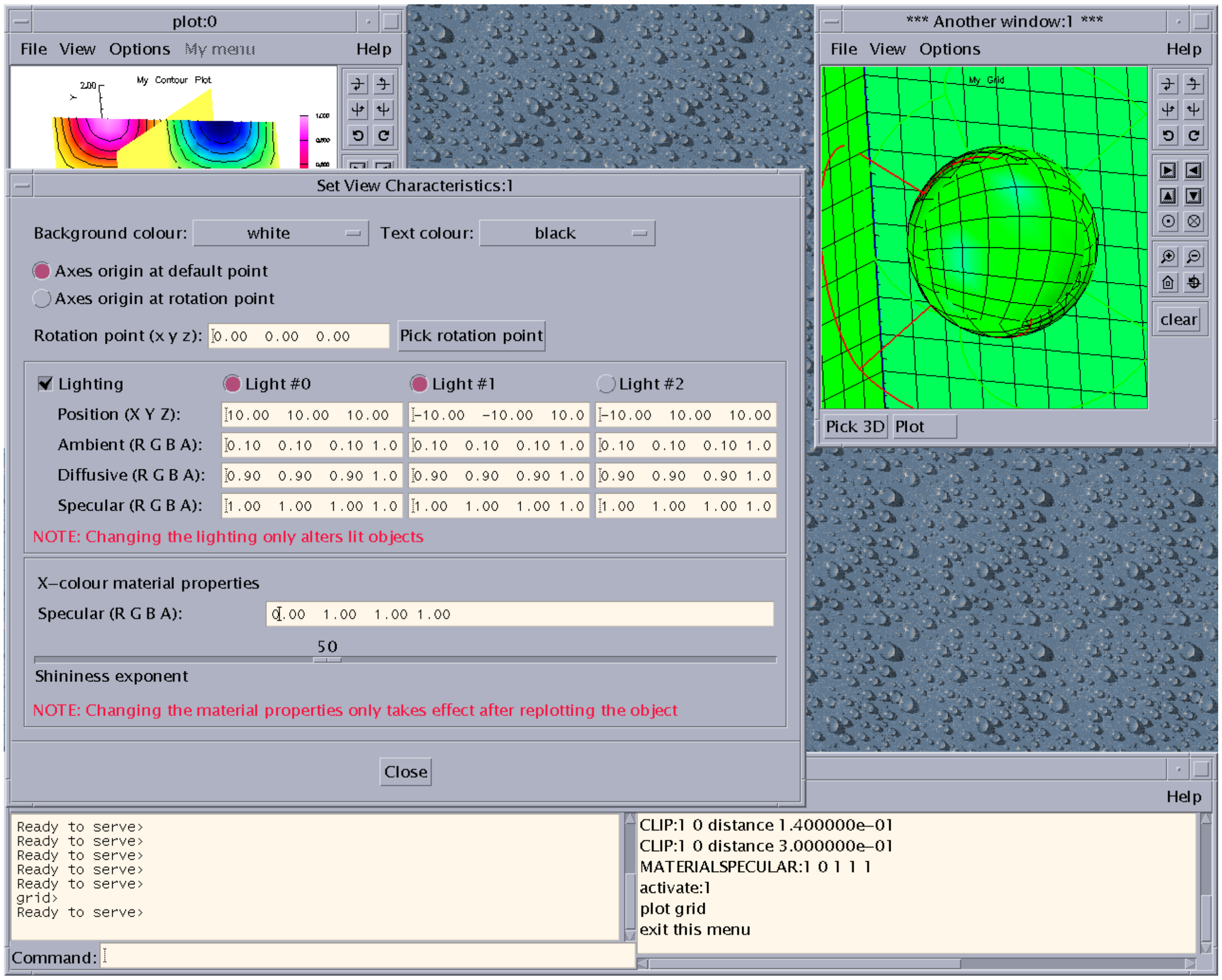
The predefined materials are listed in table 2. Hence, any colour that is not in the table is considered to be an X-colour.

Specular (R, G, B, A): The specular reflective property of the X-colour material. For example, by setting the first number to zero, you will get a bluish reflection on the green sphere in the example application. Note that the same effect could have been obtained by changing the specular colour of the light sources.

Shininess exponent: A number between 0 and 128 that describes how "narrow" the specular reflection of the X-colour material will be. A lower number gives a wider reflection.

emerald	jade	obsidian	pearl
ruby	turquoise	brass	bronze
chrome	copper	gold	silver
blackPlastic	cyanPlastic	greenPlastic	redPlastic
whitePlastic	yellowPlastic	blackRubber	cyanRubber
greenRubber	redRubber	whiteRubber	yellowRubber

Table 2: Predefined materials.



Background colour: white Text colour: black

[✓ Axes origin at default point]

Axes origin at rotation point

Rotation point (x y z): 0.00 0.00 0.00

[✓ Lighting] [✓ Light #0] [✓ Light #1] Light #2

```
Position (X Y Z):      10.00  10.00  10.00   -10.00 -10.00  10.0   -10.00  10.00  10.00
Ambient (R G B A):     0.10  0.10  0.10 1.0  0.10  0.10  0.10 1.0  0.10  0.10  0.10 1.0
Diffusive (R G B A):   0.90  0.90  0.90 1.0  0.90  0.90  0.90 1.0  0.90  0.90  0.90 1.0
Specular (R G B A):    1.00  1.00  1.00 1.0  1.00  1.00  1.00 1.0  1.00  1.00  1.00 1.0
```

NOTE: Changing the lighting only alters lit objects

X–colour material properties

Specular (R G B A): 1.00 1.00 1.00 1.00

50

Shininess exponent

NOTE: Changing the material properties only takes effect after replotting the object

```
Ready to serve>
Ready to serve>
Ready to serve>
Ready to serve>
Ready to serve>
grid>
Ready to serve>
```

Command:

```
CLIP:1 0 distance 1.400000e–01
CLIP:1 0 distance 3.000000e–01
MATERIALSPECULAR:1 0 1 1 1
activate:1
plot grid
exit this menu
```

Figure 14: The view characteristics dialog window

9.4 Changing the default appearance of the windows

Default settings for some parameters can be changed through a file named .overturerc in your HOME directory. An example of an .overturerc file is

```
commandwindow*width: 800
commandwindow*height: 150
graphicswindow*width:  650
graphicswindow*height: 500
backgroundcolour: mediumgoldenrod
```

```
foregroundcolour: steelblue
```

The window sizes is specified in pixels. It is not necessary to specify both the width and height, and the default size is obtained either by omitting the command completely, or by setting the size to -1. The default foreground colour is black and the default background colour is white. If you change them, you must use one of the following colours:

black	white	red	blue	green
orange	yellow	darkgreen	seagreen	skyblue
navyblue	violet	pink	turquoise	gold
coral	violetred	darkturquoise	steelblue	orchid
salmon	aquamarine	mediumgoldenrod	wheat	khaki
maroon	slateblue	darkorchid	plum	

Index