Terminator meets Simulator: CGI Tools used to drive a Virtual Product Evaluation

Christopher M. Pieper and Janis Hughes

Kimberly-Clark Corporation

Abstract: When computer generated imagery (CGI) first started to appear on movie and video game screens, the emphasis was to make something look close enough to its real counterpart to satisfy audiences. As graphics and computing capability progressed, audiences became less satisfied with "cartoon-like" animation; they demanded more realistic visuals. The entertainment industry reached across the aisle to the scientific community in attempt to incorporate more physical realism into animations. As a result, today's physics- based CGI can fool even a scrutinizing scientist into believing what he's seeing is real.

While the entertainment industry profited by collaborating with the scientific community, the benefit is mutual. For example, in order to study the fit performance of a dust mask with facial movement, we borrow a CGI technique to make simulation feasible. Specifically, the complex and intricate movements of the face are represented using a high resolution motion capture technique. Similarity between motion capture and finite element data forms allows their interchangeability.

In this study, detailed motion capture data of real facial movement was interpreted to produce a finite element model (node and element definitions) along with a time history of nodal displacements. The finite element data was stored as an Abaqus output database and subsequently used as a global model to drive a similar facial submodel to evaluate the fit and sealing performance of a pouch style face mask design under realistic use conditions. This paper outlines the analysis approach and shows results from the simulation.

Keywords: Abaqus, Submodel, c3d, Motion Capture, Moving Surface, Contact, CGI.

1. Introduction

This paper outlines an analysis approach used to evaluate the interaction between a man's face and a pouch style face mask used for respiratory protection (dust mask) that he is wearing. The process developed is, in general terms, a method of simulating a moving surface in a contact analysis. The specifics of the application are secondary to the methodology. At the time of this writing, the analysis for which this approach was developed is still in progress; therefore, details about the analysis are included only to illustrate the analysis methodology and show its advantage for special applications.

The methodology to be described involves using capabilities of Abaqus that likely were not developed with this purpose in mind. Specifically, a detailed set of position histories for hundreds of points on a face as it moves through various expressions and mouth open positions (e.g., a

moving surface) is used to defined nodal displacements in an Abaqus output database (ODB). This ODB is subsequently used as a global model to drive points on a similar finite element surface structure (depicting a face). The facial surface interacts with a virtual product (a dust mask) as a means to study the mechanical interaction between the face and the product. The process that was developed and simulation results to date were performed as a test to establish the feasibility of this methodology. Fortunately, demonstration of a successful analysis has shown the process to be both feasible and useful for virtual product evaluation and potentially other applications.

2. Motivation for Methodology

The analysis methodology described in this paper involves several steps. Why go to such lengths to perform such an analysis? The complexity of the human face in its design and function (see figure 1) provides the motivation. The human face has been a focus of study in the computer graphics world almost since its inception (Hung, 1995). Hung writes, "Human faces are among some of the most difficult objects to model in computer graphics… Facial expressions are the result of the movement of skin layered atop muscles and bone structures, and may have thousands of combinations." For similar reasons, representing the surface motion of a human face is a challenge for a structural model.

Representation of the positions and movement of the face is a significant challenge in evaluating the sealing performance of a dust mask in use. Part of the product analysis is to evaluate the relative benefit of potential design modifications. Since our interest lies in the mask, the facial motion merely represents a moving surface that the product interacts with. Therefore, ideally, the facial motion that causes the product to deform has to be relatively simple to represent. It is not practical to derive a modeling technique to represent facial motion based on the underlying anatomic structure for this analysis. The challenge of representing realistic facial motion is not unique to this structural analysis, as suggested by Hung, it has intrigued computer graphic professionals for many years.

In the recent movie, "The Incredible HULK," (Universal Studios, 2008) a process called "Contour™ Reality Capture" (Perlman, 2006) was used to capture an actor's facial movement. This motion was used to drive a graphical model depicting another face (the HULK's) in a computer generated animation. The capability to capture a high resolution moving (deformable) surface (e.g., Contour™ Reality Capture) has been developed and offered by Mova® LLC[1]. Since the problem of representing a realistic moving surface (such as a moving face) is common from both a CGI and structural analysis perspective, it follows that a solution from one perspective will benefit both. The measurement and replay of facial movement has already been demonstrated as a solution for CGI, and now has also been used to solve a problem in a structural analysis. The methodology developed is generally useful for describing any complex moving surface in a contact analysis.

[1] Mova® LLC, http://www.mova.com

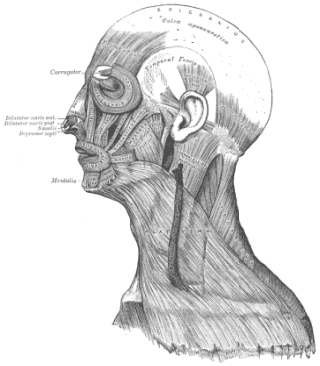

Figure 1a. Muscles of the head, face and neck (Gray, 1918 [*fig 378*])

Figure 1b. Side view of the skull (Gray, 1918 [*fig 188*])

3. Analysis Process Overview

The moving face represents a contact surface with which an object, such as a dust mask, interacts. Using Abaqus/Explicit, we are able to successfully simulate the interaction between our deformable product and the deforming facial surface.

In summary, the process of creating a moving facial surface model involves: 1) reading facial motion capture data, 2) building a finite element model using points from the motion capture data, 3) generating an output database (ODB) containing the facial finite element definitions, 4) populating the facial output database with nodal displacements based on the motion capture data and 5) using the updated ODB as a global model to drive a submodel representation of similar facial geometry. The submodel may also contain additional objects such as the dust mask in our example case. A flowchart of the process is shown in figure 2.

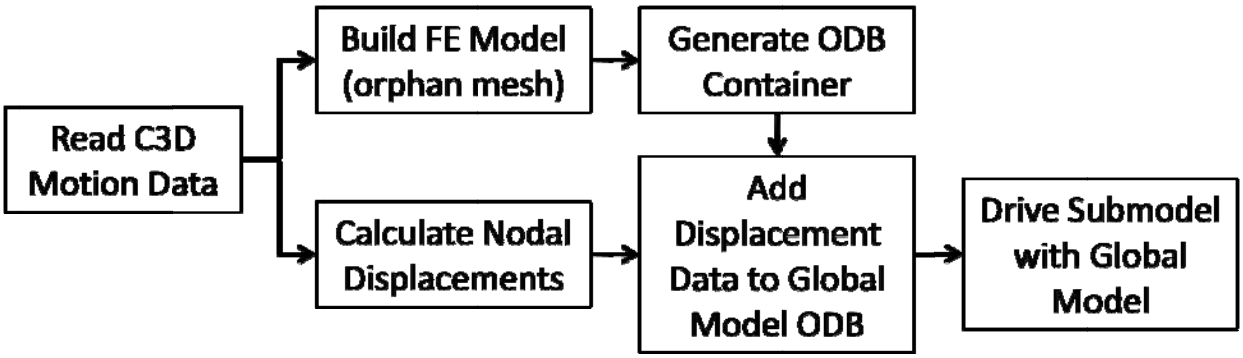

Figure 2. Flowchart of Analysis Process

The first step was to extract surface point position data from a sample set of facial motion data provided by Mova® LLC. The data was provided in an open format called C3D[2] which is described as "a public domain, binary file format that is used in Biomechanics, Animation and Gait Analysis laboratories to record synchronized 3D and analog data." A public domain Python module (Perin, J.B., 2005) written to run in Blender[3] provided the fundamental functionality[4] needed to extract position data from the C3D file.

The next step was to use the initial positions of the points as the basis for a finite element model definition. The points became nodes, and element definitions were completed using surfacing software (Geomagic[5]) to establish the nodal connectivity. Nodes and elements were written to an Abaqus input file (using a Python program) and subsequently imported into Abaqus/CAE as an orphan mesh part. Using the orphan mesh as the basis for a minimal model definition, a step definition was added and a sparse output database (ODB) was generated by performing a datacheck. This ODB served as a container into which displacement data was added to become a global model used to drive a submodel representing a moving face.

The global ODB was completed by adding nodal displacements as fieldOutput using the Abaqus Python scripting interface. The updated ODB was viewed as an animation using Abaqus/Viewer to verify that all data was converted correctly. Figure 3 shows visualizations of several frames from the updated ODB.

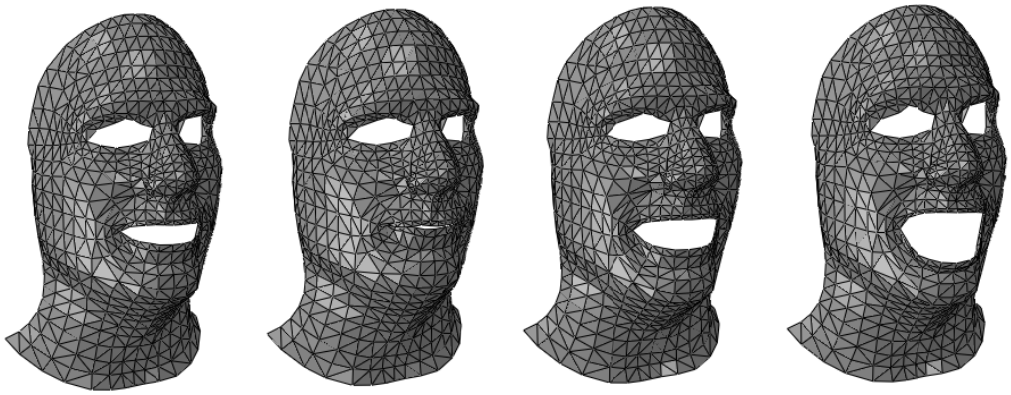

Figure 3. Deformed Shapes of Deformable Surface (face) at Points in Time

The updated global model was used to drive the moving surface portion of the submodel. In this case, the submodel involved more than just the moving surface; it also included the representation

[2] C3D format, http://www.c3d.org
[3] Blender Software, http://www.blender.org
[4] The Python code was modified but the C3D reading capability was left intact
[5] Raindrop Geomagic Software, http://www.geomagic.com

of the product being evaluated and non-moving parts of the head. With the submodel main assembly completed, a submodel boundary condition was applied to the moving portions of the facial structure. Additional loads and boundary conditions were added and the model was run to completion.

Finally, the analysis was completed by post processing the results. Figure 4 shows the simulated face mask in place on the virtual user along with a picture of a real person wearing a similar mask. Post processing of the simulation results revealed several regions (such as the high curvature areas of the nose) that exhibit gapping between the mask seal and the face as evidenced by gaps in contact pressure contours as shown in Figure 5. These areas represented design challenges that will be addressed with product design changes. The new designs will be evaluated using this simulation technique as a means to get a quick look at the benefits of each change.

Figure 4. KIMBERLY-CLARK PROFESSIONAL DUCKBILL® Dust Mask (Real and Simulated)

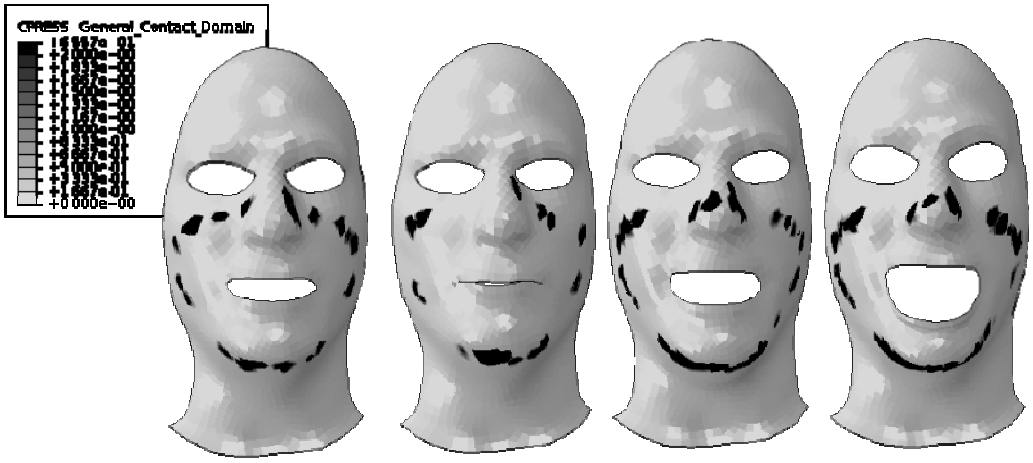

Figure 5. Contact Pressure Contours as an Estimate of Sealing Effectiveness on Face at Various Points in Time

4. Simulation of Face Mask

Although the analysis process is involved, the benefit is worth the effort. As is generally the case with other virtual prototyping applications, the advantage of unhindered exploration of a design space provides a means to quickly test the merit of changes and can help to shrink the product design cycle. For cases involving human (or animal, etc.) product testing, subtle differences in behavior can influence the measured benefit of a design change (by adding a source of test variability). Additionally, physical measurement of low contact pressures between two deformable bodies of irregular geometry is difficult at best and in many cases impossible. Simulation provides a means to obtain a repeatable and non-biased evaluation of product performance under specifically prescribed conditions. This type of evaluation is extremely difficult using human subjects and physical measurements.

5. Conclusion

An analysis procedure for the evaluation of a virtual face mask respirator has been shown to be feasible using commercial and custom software. This method has advantages over other possible means of representing a complex moving surface such as a moving face in the test case. The method should be feasible for other applications that require interaction with a moving surface that would otherwise be difficult to simulate.

The methodology described in this paper is an example of using a combination of products and features that were initially created for another purpose to achieve new capability. The methodology is useful for representing "living surfaces." The built in features of Abaqus including the Python interface and submodeling capability make this methodology possible. The long time demand for realistic CGI provided the motivation for developing moving surface data

capture. This work represents another example of the convergence of interests between the analytical and entertainment communities. When Terminator meets Simulator, they both win.

6. References

1. Perlman, S., "Volumentric Cinematography: The World No Longer Flat," white paper available at http://www.mova.com/pages/whitepaper.html, 2006
2. Hung, D. D, and Huang, S. T, "Project Deidre (I) - Modeling Human Facial Expressions," CS 718 Topics in Computer Graphics, http://www.nbb.cornell.edu/neurobio/land/OldStudentProjects/cs490-95to96/hjkim/deidre.html, 1995
3. Gray, H, "Anatomy of the Human Body," Figures 188 & 378, Philadelphia: Lea & Febiger, 1918
4. Perin, J. B., "C3D Importer", http://pagesperso-orange.fr/jb.perin/, 2005

7. Appendix

7.1 Methodology Details

The methodology described in this paper is hereby further expanded to allow its application to other problems that might benefit from this approach.

7.1.1 Collection of Motion Data

The collection of motion data in (provided in C3D format) was performed by Mova® LLC using a proprietary technique (Perlman, 2006) allowing the capture of a moving surface. The data used for this study was a sample dataset provided by Mova allowing development of the methodology. With the methodology and supporting software in place, a more directed and specific set of data will be requested and purchased from Mova as the basis for future product development analysis.

7.1.2 Conversion of Motion Data

Motion data in C3D format is directly usable by several commercial software packages; however, since a commercial package was not available for this feasibility study, public domain software was used to extract data from the binary C3D file for use. Specifically, a Blender module (Perin, 2005), written in Python, was used as the basis for the C3D extraction and conversion program. Below is a portion of the modified code used to calculate point (node) displacements.

```
# displacements
for i in range(1, len(Markers), 1):
  filename='points_%d.txt' % i
  outfile=open(filename,'w')
  PointData=[]
  Displacements=[]
  for j in range(len(Markers[i])):
    pstr=`Markers[i][j]`
    pstr=string.replace(pstr, 'x = ','')
```

```
    pstr=string.replace(pstr, '[','')
    pstr=string.replace(pstr, ']','')
    pstr=string.replace(pstr, 'y = ',' ')
    pstr=string.replace(pstr, 'z = ',' ')
    fields=string.splitfields(pstr)
    PointData.append((eval(fields[0]),eval(fields[1]),eval(fields[2])))
    dx=PointData[j][0]-Frames[i-1][0][j][0]
    dy=PointData[j][1]-Frames[i-1][0][j][1]
    dz=PointData[j][2]-Frames[i-1][0][j][2]
    Displacements.append((dx,dy,dz))
    outfile.write('%s\n' % pstr)
  Frames.append((PointData, Displacements))
  outfile.close()
```

7.1.3 Generation of Global Output Database

The initial position of surface points are used as the basis for an orphan mesh definition by directly generating an Abaqus input file using the following portion of Python code.

```
outfile=open("faceMesh.inp","w")
outfile.write('*part, name=face\n')
outfile.write('*node\n')
for i in range(len(nodePoints)):
    outfile.write('%d, %f, %f, %f\n' % (i+1,nodePoints[newOrder[i]][0],
                                        nodePoints[newOrder[i]][1],
                                        nodePoints[newOrder[i]][2]))
outfile.write('*element, type=s3r, elset=face\n')
eid=1
for i in range(nnodes+1, len(lines), 1):
    fields=string.splitfields(lines[i])
eid,newOrder[eval(fields[0])]+1,newOrder[eval(fields[1])]+1,newOrder[eval(fiel
ds[2])]+1
    outfile.write('%d, %d, %d, %d\n' % (eid,
                                        newOrder[eval(fields[0])]+1,
                                        newOrder[eval(fields[1])]+1,
                                        newOrder[eval(fields[2])]+1))
    eid+=1
outfile.write('*end part\n')
outfile.write('*Assembly, name=Assembly\n')
outfile.write('*Instance, name=face-1, part=face\n')
outfile.write('*End Instance\n')
outfile.write('*End Assembly\n')
outfile.close()
```

A simple means of generating an output database to be used as a global model is to build a model using the orphan mesh and run a datacheck. This output database is populated with displacement fieldOutput data using the Abaqus Python interface and code such as the following.

```
from odbAccess import *
# build Abaqus odb
odb=openOdb('face.odb')
myStep=odb.steps['Step-1']
instance=odb.rootAssembly.instances['PART-1-1']
```

```
time=0.0
inc=0.1
nodeIds=[]
for node in instance.nodes:
  nodeIds.append(node.label)
for i in range(1,len(Frames),1):
  frame=myStep.Frame(incrementNumber=i, frameValue=time)
  dFieldOutput=frame.FieldOutput(name='U', description='Displacement',
                                 type=VECTOR)
  time+=inc
  displacements=[]
  labels=[]
  label=1
  for dvect in Frames[i][1][1:]:
    if (label in nodeIds):
      displacements.append(dvect)
      labels.append(label)
    label+=1
  displacements=tuple(displacements)
  labels=tuple(labels)
  dFieldOutput.addData(position=NODAL, instance=instance,
                       labels=labels, data=displacements)
odb.save()
odb.close()
```

7.1.4 Simulation using Sub-modeling

The newly generated global model is used to drive a submodel representing the moving surface in the simulation of interest. This (driven) moving surface can interact with other modeled objects as a complex moving boundary that would be difficult to produce using a more direct approach (e.g., driving nodes directly with individual boundary conditions).

8. Acknowledgements

The authors thank Mova® LLC for the sample set of facial motion data used to establish the feasibility of this analysis method. We would also like to acknowledge and thank Brett Conard from the Simulia Central office for his help with some of the technical difficulties.