This document contains the draft version of the following paper:

M. Schwartz, S.K. Gupta, D.K. Anand, J.E. Brough and R. Kavetsky. Using virtual demonstrations for creating multi-media training instructions. *CAD Conference*, Hawaii, June 2007.

Readers are encouraged to get the official version from the conference proceedings or by contacting Dr. S.K. Gupta (skgupta@umd.edu).

Using Virtual Demonstrations for Creating Multi-Media Training Instructions

Maxim Schwartz[1], Satyandra K. Gupta[2], John E. Brough[3], Davinder K. Anand[4], Robert Kavetsky[5]

[1]Center for Energetic Concepts Development, University of Maryland, maxim@umd.edu
[2] Center for Energetic Concepts Development, University of Maryland, skgupta@umd.edu
[3]Naval Surface Warfare Center at Indian Head, Maryland, john.e.brough@navy.mil
[4] Center for Energetic Concepts Development, University of Maryland, dkanand@umd.edu
[5]Naval Surface Warfare Center at Indian Head, Maryland, bob.kavetsky@verizon.net

ABSTRACT

Often generating training instructions for virtual environments is a long and tedious process. In this paper, we discuss the development of a virtual environment (VE) instruction generating tool called Virtual Author which is the main component of the Virtual Training Studio (VTS). VTS is a virtual environment-based training system that provides instructors with a tool to create training instructions and allows trainees to learn assembly operations in a personal virtual environment. The Virtual Author tool is designed to allow an instructor to perform virtual demonstrations using CAD models in the virtual environment in order to quickly generate VE-based training instructions for use in VTS. This paper describes the algorithms used to carry out motion smoothening of instructor's actions, automated text instruction generation based on part and assembly motions, and extraction of alignment constraints from 3D CAD models to support instruction generation. We also present examples to illustrate how the use of the Virtual Author tool leads to a significant reduction in the training instruction generation time.

Keywords: Virtual environment, authoring, virtual demonstrations, training, instructions.

1. INTRODUCTION

The workforce in most industries requires continued training and update. Current training methods, for the most part, involve a combination of paper-based manuals, DVD/video-based instructions and/or hands on master-apprentice training. Due to the rapid influx of new and changing technologies and their associated complexities, accelerated training is a necessity in order to promote and maintain an advanced and educated workforce. Existing training methods can be further improved in terms of cost, effectiveness, time expenditure and quality through the use of digital technologies such as virtual environments (VE) [3]. Specifically, VE-based techniques have been shown to be useful for simulating mechanical assembly operations [4], [11], [5], [1], [7], [2], [9]. The advent of personal virtual environments offers many new possibilities for creating accelerated training technologies.

One of the biggest challenges associated with using virtual environments for training is the ability to rapidly create VE-based training material. Such training material often consists of animations, text, video, audio, and often interactive simulations that allow the trainees to pick up virtual parts and carry out the assembly themselves. Use of software engineers to write new code for each tutorial is too expensive and time consuming. A better alternative is to build an application that is capable of generating most of the training material with minimal input from the instructor and no coding effort. This eliminates the need for specialized coding for individual tutorials and reduces the work load on the instructor. Implementing such an application presents significant challenges, however.

Some researchers have used interactive demonstrations as a means of conveniently creating tutorials. An example of this type of authoring tool is the Cognitive Tutor Authoring Tool (CTAT) [6]. CTAT builds math tutorials by demonstration. The author performs a scenario using a windows based GUI while the system records the procedure. The author must also explicitly demonstrate all alternative solution paths both correct and incorrect. Another example of this type of authoring tool is RIDES [8], which can train amongst other things, nurses to use medical equipment. In simple mode RIDES allows an author to 'record' a procedure that students must learn,

simply by carrying out the procedure in a window based user interface. Another example of virtual authoring is a system called UVAVU [10], which generates assembly sequences by observing an author perform an assembly in a virtual environment. UVAVU makes use of known information about final part locations within the assembly to perform snap-ons upon collision during the author's demonstration. Testing conducted with UVAVU showed that assembly plans produced in virtual reality were similar to those produced in a real environment.

In this paper, we discuss the development of a virtual environment instruction generating tool called Virtual Author which is a main component of the Virtual Training Studio (VTS). VTS is a virtual environment-based training system that allows instructors to create training instructions and allows trainees to learn assembly operations in a virtual environment. The most novel aspect of the Virtual Author is the automated instruction generation based on demonstrations of motions in the virtual environment by an instructor. The Virtual Author allows an instructor to import CAD models of parts belonging to a product into the virtual environment and define a dictionary by naming parts and certain features on those parts. The instructor then simply picks up CAD models in the virtual environment and assembles them in the order he/she wants the assembly process to be taught to trainees. The system monitors and records the instructor's actions and then performs motion smoothening and alignment adjustment to account for the fact that the instructor's motion and placement may not be precise. Virtual Author then automatically generates (1) text instructions, (2) 3D animations, and (3) 3D interactive simulations. These training instructions are generated based on observed motion, feature and part alignment, and collision detection during the virtual demonstration. All the generated data associated with the tutorial is then stored in a file which can later be loaded by the VTS during training sessions.

The greatest benefit of the Virtual Author is the rapid creation of VE-based training instructions without the need for coding. This reduces the time and the cost of building new tutorials. Demonstrations in VE also make the system user friendly and allow the process expert, who is most likely a mechanical or manufacturing engineer, to directly control how the instructions for device assembly are generated without relying on programmers. Another benefit of Virtual Author is the consistency of the generated tutorials. Hard-coding the data for each tutorial may often lead to inconsistent tutorials with slight differences in either the look and feel or the behavior of features. Generating training instructions automatically from virtual demonstrations also ensures that the instructor can visually verify that the training instructions being generated are complete and no step is missing in the sequence. Consistent tutorials are expected to streamline the training process for trainees once they have become accustomed to the system.

2. SYSTEM OVERVIEW

2.1 Overview of Virtual Training Studio

The Virtual Training Studio is a suite of tools, which currently consists of the Virtual Author, Virtual Workspace, Virtual Mentor, and CAD model importing utilities. With Virtual Training Studio, training instructors have the option of employing a wide variety of multi-media options such as 3D animations, videos, text, audio, and interactive simulations to create multi-media training instructions. The virtual environment enables trainees to practice training instructions using interactive simulation and hence reduces the need for practicing with physical components. Our system is mainly geared toward cognitive skills: training users to recognize parts, learn assembly sequences, and correctly orient the parts in space for assembly. The VTS was designed to be an affordable Personal Virtual Environment (PVE) for training. We developed a low cost wand design and use an off the shelf head mounted display (HMD). The level of physics based modeling that has been implemented as well as the hardware selected reflects this design decision. The VTS system architecture is shown in Figure 1.

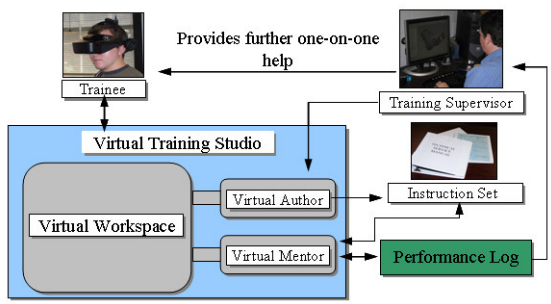

Fig. 1: Virtual Training Studio architecture.

The user interacts with the tutorial using a Head Mounted Display (HMD) and a wireless wand. Four optical trackers (infrared cameras) and two gyroscopes are used to track the position and orientation of the user's head and the wand. The wand consists of an off the shelf wireless presenter, an infrared LED, and a wireless gyroscope. Inside the Virtual Reality environment, the user can manipulate the CAD models and the buttons using a virtual laser pointer, which is controlled by the wireless wand. A wireless gyroscope and another infrared LED are mounted on the HMD. The cameras track the two LEDs and use triangulation to return the x,y,z, positions. Use of haptics and gloves was avoided in order to keep the cost of the system down and make the user interface as easy and user friendly as possible. After user testing with a glove-based version of the system, utilizing 2 5DT DataGloves, we made the decision to create a wand based system due to the complications of the user interface in the use of gloves and the simplicity and user friendliness of the wand interface.

The software infrastructure of the VTS was built using a combination of programming languages: C/C++, Python, and OpenGl. Additionally, a number of libraries were used: WorldViz's Vizard for general purpose loading and transformation of VRML models, ColDet for collision detection, Gnu Triangulated Surface library (GTS) for segmentation, and wxPython for Graphical User Interface. Figure 2 shows the software infrastructure.

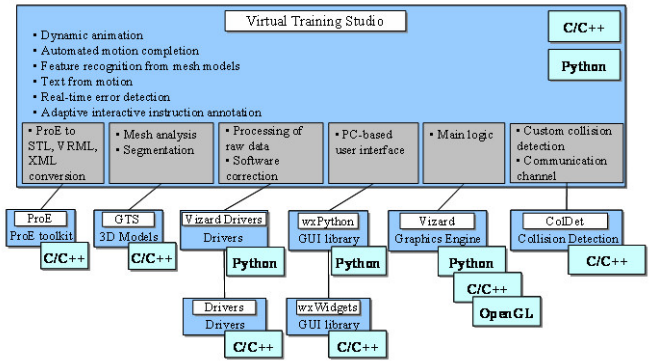

Fig 2: Software infrastructure and libraries of the VTS.

Trainees interact with a component of the VTS called Virtual Workspace. Virtual Workspace provides the basic infrastructure for multimodal training and incorporates the appropriate level of physics-based modeling consistent with the operation of a low cost PVE. Virtual Workspace houses the necessary framework to allow manipulation of virtual objects, collision detection, execution of animations, and it integrates the hardware with the software to provide the user an intuitive and easy to use interface to the virtual environment. The current version of the Virtual Workspace places the user in a furnished room with a table at the center and a projector screen on one of the walls. Parts used in the tutorial are placed on the table, while video as well as text instructions are displayed on the projector screen. The user interacts with the VE using a single wand, represented in the VE as a virtual laser pointer, to pick up, move and rotate objects and to click on buttons located on the control panel at the front of the room. The implementation of the Virtual Workspace also includes the option to interact with the VE through a desktop personal computer (PC) interface. Virtual Workspace offers three primary modes of training: (1) 3D Animation Mode which allows users to view the entire assembly via animations, (2) Interactive Simulation Mode which is a fully user driven mode that allows users to manually perform the assembly tasks, and (3) Video Mode which allows users to view the entire assembly via video clips. Trainees can switch between these modes at any time with the click of a button.

Virtual Mentor is a component of the VTS that is designed to monitor trainees inside the virtual environment, log their activities, perform detection of errors, and present very precise messages and hints. Virtual Mentor is currently embedded in the Virtual Workspace.

Please see Figure 3 for a screenshot of the VTS environment as the user sees it through the HMD and see Figure 4 for a photograph of a user immersed in the virtual environment.

2.2 Overview of Virtual Author

The goal of the Virtual Author is to enable the instructor to quickly create multi-media training instructions for use in the Virtual Workspace without writing any code. The Virtual Author package includes a ProEngineer

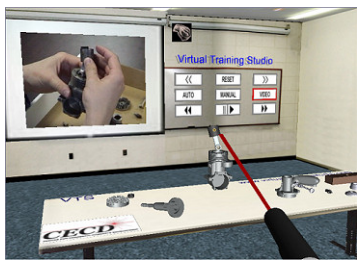

Fig. 3 (left): VTS environment as the user would see it through the HMD, Fig. 4 (right): A user interacting with VTS.

(ProE) plug-in written in ProE Toolkit, which allows an engineer to load an assembly into ProE and export it to the file formats used in the VTS – VRML and STL. We decided to use VRML and STL formats to ensure that the VTS system can work with a wide variety of CAD systems.

The instructor begins the authoring process by loading a set of VRML and STL CAD models into a tool called Part Loader where the instructor declares a tutorial specific dictionary for Virtual Author. The dictionary is created by giving names to parts and selected features. The instructor also uses the tool to give all the CAD models initial arrangement on the virtual table. At the end of the dictionary declaration process, the tool generates a data file which is loaded into Virtual Author on start up. The instructor then steps into the virtual environment and performs a virtual demonstration which the system monitors and records.

During the virtual demonstration, the instructor picks up one part or subassembly with a single virtual laser pointer and inserts it into another part or subassembly. Hence, there is always a moving subassembly and a receiving subassembly which remains stationary. After the instructor carries out the assembly inside the virtual environment for a particular step, the Virtual Author performs motion smoothening by calculating the final assembly path, calculating the insertion point, and more precisely realigning the held assembly with the receiving assembly. Motion smoothening is necessary due to the fact that the system does not prevent one object from passing through another upon collision and the fact that highly precise placement and alignment of parts may not be possible inside the virtual environment. Not permitting the parts to intersect at all during the motion would have required computationally expensive constraint management techniques that may slow down the scene refresh rate. Hence we allow CAD models to intersect with each other during virtual demonstrations. Most such intersections are eliminated from the training instructions using a motion smoothening technique described in section 5. Motion smoothening allows Virtual Author to deal with minor placement and orientation errors during virtual demonstrations that result due to no enforcement of non-penetration constraints and lack of haptics feedback.

For each step that the instructor demonstrates in the virtual environment the instructor also declares symmetries and specifies the symmetry types. The part symmetry information is later used by Virtual Workspace to allow trainees to assemble parts using alternate insertion locations and orientations. For each step, highly detailed text instructions are generated automatically by combining data about collision detection, part motion and alignment constraints with the dictionary declared by the instructor. Creation of text instructions is discussed in section 4. Text instructions enable trainees to refresh their memories about the assembly process at the shop floor where VE installations are not available. Automated generation of text instructions reduces the text instruction generation time and ensures that there is no missing step in the text instructions. In addition to the text instructions, Virtual Author also automatically generates data for dynamic animations and interactive simulation for later use in Virtual Workspace.

During the final phase the instructor also has the option of loading video clips (.avi files) and audio (.wav files) and associating them with each step. Both the motion smoothening techniques as well as automatic text from motion generation are heavily dependent on extraction of alignment constraints from polygonal models discussed in Section 3.

3. EXTRACTION OF SURFACES AND ALIGNMENT CONSTRAINTS FROM POLYGONAL MODELS

In order to generate proper text instructions and perform motion smoothening, we need location and orientation information about parts' faces. In order to make our system independent of any specific CAD system, we use STL and VRML models. Both of these file formats represent parts as a collection of triangles (also called triangulated meshes) and do not maintain part surface information. Hence we need to extract part surface location and orientation data by analyzing a triangulated mesh of each part. The required data consists of the following: extracted surfaces which are classified as cylindrical, planar and curved, axes associated with planar and cylindrical surfaces, and areas of the surfaces.

As a part of the Virtual Author, we have developed a module which is capable of performing segmentation of part geometries and extraction of axes of planar and cylindrical patches. In order to extract axes of parts, the system must first perform segmentation on all the parts the instructor has chosen to use in the VE tutorial. Our current algorithm classifies all extracted patches into three categories: planar, cylindrical, and curved. All patches that are not determined to be planar or cylindrical are placed in the generic curved category. Segmentation and classification is conducted in three stages. In the first stage, the algorithm calculates angles between all adjacent facets and places all adjacent, coplanar facets into planar groups. In the second stage, the algorithm visits all planar groups (patches), converting connected planar groups into curved groups. For example, Virtual Author may take multiple small planar patches which are not co-planar and convert them into a curved patch based on a set of heuristics such as area of a planar patch relative to total part area, area of planar patches relative to the area of neighboring planar patches, and angles between neighboring planar patches. Cylindrical patches are extracted in phase 3. Our algorithm for finding cylindrical surfaces involves going through each patch labeled as curved and ascertaining whether the facets that make up the curved patch have a mostly cylindrical arrangement. That algorithm first visits every facet and places the area of each facet into a bucket that matches the facet's normal. The buckets are sorted in descending order based on surface area. The algorithm then chooses the largest bucket and uses it as a reference. It then visits every other bucket and computes the cross product between the normal of the reference bucket and the normal of each visited bucket. The results of the cross products (resulting vector and the surface area of visited bucket) are placed in a separate set of buckets which are sorted by surface area in descending order. If the largest bucket contains at least 70 percent of the surface area, then the patch is mostly cylindrical and the vector associated with the biggest bucket is used as the axis. The axis of a planar patch is simply the normal of the surface. Axes of curved patches are not extracted.

Figures 5 and 6 show how two representative parts were automatically segmented into patches by our system. For the part shown in Figure 5, a model airplane engine piston, the segmentation algorithm correctly produced 13 planar patches, 6 cylindrical patches and 1 generic curved patch. For the part shown in Figure 6, a parachute deployment device cartridge, the segmentation algorithm correctly produced 4 planar patches, 2 cylindrical patches and 1 generic curved patch.

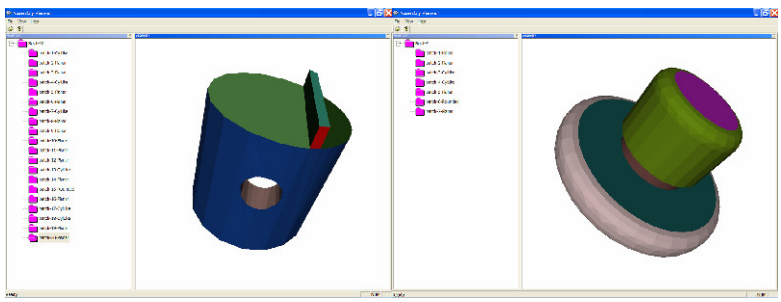

Fig 5 (left): Surface segmentation performed on model airplane engine piston, Fig 6 (right): Surface segmentation performed on a parachute deployment device cartridge.

Tests of this algorithm on a large number of parts with varying shape and complexity revealed good extraction accuracy. We have also conducted extensive performance testing of this algorithm with respect to file size. The computational results indicate that our algorithm can support interactive applications. These tests were conducted on a machine with a dual-core Intel processor (1.8 Ghz) and 1GB of RAM. For a set of 5 MB Stereolithography (STL) files in ASCII format the extraction of alignment constraints was completed on average in 5.50 seconds. For 10 MB STL files in ASCII format the algorithm completed on average in 39.74 seconds. For a set of 20 MB STL files the algorithm completed on average in 64.94 seconds. Note that segmentation and

extraction of alignment constraints is a one time operation performed when parts are loaded with the Part Loader.

4. AUTOMATIC GENERATION OF TEXT INSTRUCTIONS

Automatic generation of highly detailed text instructions is one of the ways in which Virtual Author augments the trainer's productivity. The instructor later has the option of editing generated text and adding new text, but most of the work is done by the Virtual Author. Automatic generation of text instructions also insures that the generated text matches the generated animations and reduces the probability of the instructor forgetting to mention certain training details when generating text. Text is an important component of training instructions because it can be used not only during training in the virtual environment, but also as a reference, along with 2D illustrations, on the shop floor.

The Virtual Author generates text instructions during the replay phase of the authoring process after the instructor has demonstrated a particular step and the system has performed all the necessary motion smoothening. In the replay phase, Virtual Author plays the animation of the assembly for a step the way a trainee would see it. While the animation is playing the system performs collision detection between virtual parts, assembles, and instructor declared features. At the end of the animation it determines which features are aligned and what parts are internal or external. All the gathered information is combined to choose an appropriate set of instruction templates. The instruction templates are then completed with the insertion of part, feature, and assembly names in appropriate places and combined into sentences. A list of template groups for assembly text instructions is shown in Figure 7.

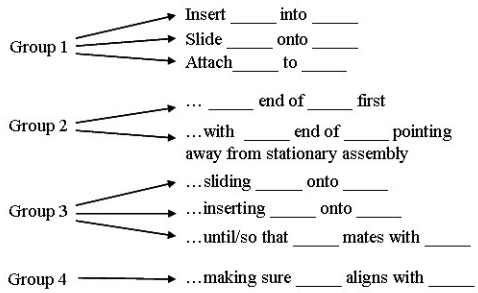

Fig. 7: Text instruction template groups.

The blanks represent part names or feature names which are inserted into the template dynamically. Each template has certain rules or conditions associated with it. The Virtual Author picks templates by determining which conditions were met during the animation of a particular step. Virtual Author always picks a single template from Group 1. This is done by recording the position at which the moving assembly collided with the receiving assembly and recording the position where the moving assembly came to a stop. A vector is then calculated from the point of collision to the stop point. Next, Virtual Author creates three axes which are aligned with the mobile assembly's three bounding box dimensions and determines which of those axes has the smallest angle with the insertion vector. The Virtual Author divides the distance between collision point and stop point by the length of the dimension associated with the nearest axis. If the ratio is greater than a certain number, then either a "slide on" or a "insert into" template must be chosen. Otherwise, the "attach to" template is used. To determine whether it should use the "slide on" or a "insert into" template, Virtual Author draws several lines that are perpendicular to the insertion vector toward the collision point. If most of the lines intersect the mobile assembly first, then "slide on" template is used, otherwise Virtual Author uses the "insert into" template.

Templates from Groups 2, 3, and 4 are optional. Virtual Author may pick several or none from each group depending on the conditions. Virtual Author may pick one of the templates in Group 2 by calculating centers of features and parts and comparing them to the center and dimensions of the mobile part or assembly. To pick a template from this group Virtual Author also compares the direction of motion of mobile assembly to the vector from center of assembly to the center of one of the parts or features. If a declared feature or part is found near the end of the assembly that is facing toward the receiving assembly, then "X end of Y first" template is used. Depending on the conditions, several templates may be picked from Group 3. The method used to pick one of

these templates is very similar to the method described for template Group 1. Only feature names are used inside templates from Group 3. As each declared feature collides with another feature, Virtual Author records the position associated with that interaction. After the animation completes, Virtual Author records the final positions of all features that collided. As for Group 1 templates, Virtual Author determines which of the three dimensions of a feature's bounding box are closest to being parallel to the direction of motion. For each moving feature that collided with a stationary feature, Virtual Author calculates the ratio between feature travel distance and this dimension length. If the travel distance is large enough then Virtual Author calculates the cross section of the colliding moving feature and stationary feature. The cross section is perpendicular to the bounding box dimension that is closest, in terms of the angle, to the direction of motion. The feature with the bigger cross section is assumed to be on the outside so the "...sliding <feature A> onto <feature B>..." template is used with the names of the appropriate moving feature and stationary feature inserted.

Lastly, Virtual Author picks none or several templates from Group 4 by checking the alignment between all the features on the receiving assembly to all the features on the moving assembly. Since each feature may be composed of several planar, cylindrical and curved surfaces, Virtual Author picks the dominant axis associated with the greatest amount of surface area in a feature. When checking for alignment Virtual Author only compares features of same type. If most of the surface area and hence the dominant axis is associated with planar surface(s), then the feature is of type "planar". Then only planar stationary features are compared to planar moving features. Once the system picks all the necessary templates and fills them with the names of the appropriate features, parts, and assemblies, the templates are combined into sentences.

For one of the steps, the instructor attached an engine cover assembly to engine case assembly. The text instruction that was automatically generated by the Virtual Author was, "Attach engine cover to engine case of the engine body assembly, so that engine cover bottom mates with engine case top. During this step please make sure cooling fins is aligned with crankshaft rod." Figure 8 shows the attachment operation being demonstrated in the virtual environment. After performing motion filtering, described in the next section, Virtual Author was able to correctly detect that one of the moving, cylindrical features called "cooling fins" is aligned with one of the stationary, cylindrical features called "crankshaft rod". Virtual Author also correctly detected that this is an "attach" operation because engine cover does not travel far after initial collision with case. Virtual Author also detected that engine cover "mates with" engine case. One template was picked from Group 1, Group 3 and Group 4.

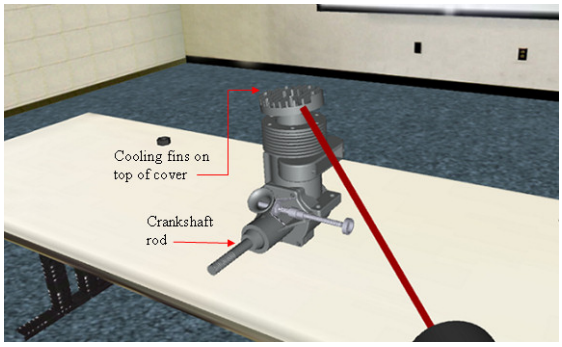

Fig 8: Virtual demonstration of attachment of engine cover to top of engine case.

We used the Virtual Author module to automatically generate text instructions for the eight steps of a rocket motor assembly and the twenty two steps of a model airplane engine assembly. For each step the quality and accuracy of text instructions was similar to the example above. The average time needed to generate a text instruction was 0.0043 seconds.

5. MOTION SMOOTHENING AND FEATURE ALIGNMENT

Motion smoothening is an important capability of the Virtual Author. This feature allows the instructor to have the freedom to perform the virtual demonstration completely unrestricted, without being hindered by any system enforced constraints on movement. As the instructor attaches or inserts one part into another, Virtual Author signals that a collision has occurred by making all colliding parts translucent, but the instructor may continue to move one part through another. Such freedom requires Virtual Author to carry out slight adjustments to the instructor's final placement of a part or subassembly due to the fact that highly precise alignment and placement

may be difficult inside the virtual environment. Precise placement is difficult for a number of reasons. Although the stereo view provided by the HMD in a virtual environment greatly helps to visualize and move the objects, it still does not match the quality of depth perception in the real world. Also, the total freedom of movement and the lack of haptics feedback reduce the chance of the user knowing if there is an invalid placement due to geometries intersecting each other. Motion smoothening utilizes the extracted surfaces (patches) and alignment constraints discussed in Section 3.

The first step in motion smoothening is the logging of the motion of the held part. When the instructor picks up an object, Virtual Author stores the positions of the held part in a queue of points. The size of the queue is adjustable. Assuming the queue is of size S then the newest point is at position 0 and the oldest point is in position S – 1.

Once the user releases the part and signals to Virtual Author to perform motion smoothening, the first task that Virtual Author performs is processing of the point queue to determine the insertion vector and the insertion position. Insertion position must be ascertained because in the Virtual Workspace, during interactive simulation, the trainee must place a part at the insertion position and then signal to the Virtual Mentor to complete the process with animation. Virtual Mentor checks if the position and orientation are correct and activates animation. The animation itself first gradually translates the part to the insertion position and then to the final position within another assembly. Virtual Author uses a recursive function to find the insertion point and the insertion vector. Because of space constraints, we will present a high level description of the function. The recursive function receives a queue as an argument and breaks it up into two subqueues. It then calculates the average vector for each subqueue. Next, it compares the two vectors. If angle between the two vectors is greater than limit L (currently set to 5 degrees), then the function calls itself, passing in the left subqueue containing points 0 through queue length divided by two. If the angle is less than L, then the function declares the right end point of the right queue as the current insertion point and calls itself, passing as the argument the neighboring right subqueue of same size (if there are any). In other words, if a significant difference between left subqueue and right subqueue is found, then the function keeps moving left (toward the newest points). If it does not find much of a difference between left queue and right queue then it assumes that it went too far left and that it needs to go back to the right (toward the older points) to find the insertion point.

After the insertion vector is found, Virtual Author processes a certain number of the largest planar and cylindrical patches on the stationary assembly and finds a patch with an axis that has the smallest angle with the insertion vector (i.e. nearest patch). It follows the same procedure for the moved assembly. Virtual Author then calculates a vector that is perpendicular to the nearest stationary patch axis and nearest mobile patch axis. It then rotates the mobile assembly around that vector so that the nearest mobile patch axis aligns with the nearest stationary patch axis. The system adjusts the original insertion vector established by the recursive function so that it aligns with the nearest stationary patch axis. The insertion point is also adjusted to reflect this.

The previous rotational adjustment is meant to align the moved assembly with the stationary assembly using the insertion vector as a guide. The next rotational adjustment rotates the moved assembly around the new insertion vector in order to align closest similar features. Each patch on the stationary assembly with an axis that is not parallel to the new insertion vector is added to a list of non-parallel stationary patches and the patch axes are projected onto the plane that is perpendicular to the insertion vector. The same is done for patches on the mobile assembly. For each stationary non-parallel patch, Virtual Author finds mobile non-parallel patches whose projected axes are within D degrees of rotation from the projected stationary patch axis, where D is adjustable. (If no mobile patches are found within D degrees, then the system continues to the next stationary patch.) Next, Virtual Author repeatedly rotates the mobile assembly around the insertion axis so that the current stationary *projected* patch axis aligns with a *projected* mobile patch axis. After each rotation, the system compares the *unprojected* mobile patch axis to the *unprojected* stationary patch axis. If they are parallel and of same type, Virtual Author stores the number of degrees that was required for rotation to make them match. Finally, Virtual Author finds the smallest such angle and rotates the mobile assembly one last time by that amount. Rotation around the insertion axis is limited to D degrees. The reasoning is that if the instructor wanted to align such distant features, then he would have done a better job of manually aligning the mobile assembly with the stationary one.

The Virtual Author simulates this process for cylindrical mobile and stationary patches that are parallel to the new insertion vector and have about the same surface area by computing the vector from the center of the assembly to the center of the cylindrical patch, projecting that vector to the plane perpendicular to insertion

vector, and adding an abstract patch with an axis matching this vector to the list of non-parallel stationary or mobile patches. This way Virtual Author is capable of aligning cylindrical features that are parallel to the insertion vector by rotating the mobile assembly around the insertion vector. An example of this is the rotation of the crankshaft so that the crankshaft pin aligns with a hole in the piston rod.

The final step in the motion smoothening process is the translational adjustment. Virtual Author shifts the mobile assembly along the plane that is perpendicular to the insertion axis and then translates along the insertion axis so that the nearest similar planar patches or cylindrical patches align.

Figure 9 shows the screen capture of a demonstration in the virtual environment for a model airplane engine assembly. A crankshaft is being attached to the piston assembly. The manually placed crankshaft is clearly misaligned with the piston assembly. The stored points in the queue are being displayed in this virtual demonstration in the form of cubes.

Figure 10 shows the adjusted placement of the crankshaft after motion smoothening. The red line represents the unadjusted insertion vector and the red prism represents the system resolved insertion point. In the new placement, the crankshaft is perpendicular to the piston rod and the crankshaft has been rotated around the adjusted insertion vector so that the crankshaft pin matches the piston rod hole. It should be noted that normally the piston assembly is inside the engine block, but for better illustration crankshaft is assembled directly onto the piston assembly in this step.

Fig. 9 (left): Crankshaft is being manually attached to piston assembly in a virtual demonstration, Fig. 10 (right): Automatically adjusted placement of crankshaft after motion smoothening.

We have tested motion smoothening and feature alignment on a large collection of parts and have confirmed that the employed algorithms work accurately on a wide variety of parts and assemblies. Testing has shown that the algorithms are sufficiently fast to be used inside a virtual environment where the system must be highly responsive to user's actions. Smoothening and alignment is carried out, on average, in 0.044 seconds for assemblies, where the average assembly is composed of 6.7 parts.

6. DISCUSSION AND CONCLUSIONS

The ability of the Virtual Author, a component of the Virtual Training Studio, to quickly generate device assembly training instructions for the virtual environment is supported by three novel capabilities implemented within Virtual Author.

- First, the Virtual Author tool is able to extract surfaces and alignment constraints needed for performing virtual assembly operations using polygonal models represented by STL files. The ability to extract this information from STL files makes our tool independent of any particular CAD system. The fact that the vast majority of CAD systems on the market can export to the STL format, makes Virtual Author, and VTS in general, highly versatile and able to accommodate engineers working with a wide variety of CAD tools.
- Second, the Virtual Author performs logging and motion smoothening on the assembly demonstrations inside the virtual environment, allowing instructors to easily demonstrate how they want the assembly process taught to trainees and make sure that the relevant features of various parts in the assembly align and mate perfectly after a quick and imprecise virtual demonstration. Not enforcing non-penetration constraints makes it possible for us to achieve very high refresh rates on an inexpensive computer during virtual demonstrations. By using motion smoothening, the system is able to cope with minor errors during virtual demonstration that result due to lack of haptics feedback and lack of enforcement of non-penetration constraints.

- Finally, the Virtual Author automatically generates (1) 3D animations of the assembly steps for the instructor to verify, (2) highly detailed text instructions by employing an instructor declared dictionary, and (3) data for 3D interactive simulations. Detailed text instructions can not only be used inside the virtual environment but also on the shop floor as a quick reference. Automatic generation of text instructions ensures that the text instructions are in complete agreement with the 3D animations.

The design and implementation of these novel capabilities has led to a huge improvement in the amount of time it takes to generate detailed, high quality multi-media training instructions for the virtual environment. Before the implementation of the Virtual Author, we needed about one week (40 hours) to manually generate training instructions for use in the Virtual Workspace. Virtual Author has reduced the time commitment for generating training instructions down to just several hours. As a specific example, the twenty three part model airplane engine assembly tutorial used in many examples and screen captures in this paper was re-generated after completion of Virtual Author tool in just one hundred thirty three minutes. Previously, it took us just over a full work week to generate the same data for the model airplane engine tutorial.

Currently, there are a number of limitations in the Virtual Author that will be addressed in the future. Tool use is currently not supported in the Virtual Workspace by trainees or in the Virtual Author by instructors. In the future, we will design and implement algorithms that will allow instructors to import CAD models of tools into Virtual Author, define their behaviors, and incorporate them into the assembly instructions. Some of the future effort will also be invested in the improvement of the quality and robustness of generated text instructions. Current limitations often prevent the Virtual Author from using correct grammar when composing sentences. One example of this is the use of the word “is”, and never “are”, regardless of the plurality of the declared feature. “Cooling fins” is declared as one feature but must be treated as plural when choosing a verb.

ACKNOWLEDGEMENTS. This research is supported in parts by the Center for Energetic Concepts Development at the University of Maryland and Naval Surface Warfare Center at Indian Head.

7. REFERENCES

[1] Banerjee, A.; Cecil, J.: A virtual reality based decision support framework for manufacturing simulation, Computers and Information in Engineering Conference, Chicago, Illinois, September 2003.

[2] Gupta, R.; Whitney, D.; Zeltzer, D.: Prototyping and design for assembly analysis using multi-modal virtual environments, Computer-Aided Design, Vol 29. No 8. pp. 585-597, 1997.

[3] Hodges, M.: Virtual reality in training, Computer Graphics World, Vol 21. No 8 1998.

[4] Jayaram, U.; Jayaram, S.; DeChenne, C.; Jun Kim, Y.: Case studies using immersive virtual assembly in industry, Computers and Information in Engineering Conference, Salt Lake City, UT, September – October 2004.

[5] Kim, C.; Vance, J.: Using VPS (voxmap pointshell) as the basis for interaction in a virtual assembly environment, Computers and Information in Engineering Conference, Chicago, Illinois, September 2003.

[6] Koedinger, K.R.; Aleven, V.; Heffernan, N.; McLaren, B.; Hockenberry, M.: Opening the door to non-programmers: authoring intelligent tutor behavior by demonstration, Lecture Notes in Computer Science 3220, 162-174, 2004.

[7] Mikchevitch, A.; Leon, J.C.; Gouskov, A.: Numerical modeling of flexible components for assembly path planning using a virtual reality environment, Computers and Information in Engineering Conference, Chicago, Illinois, September 2003.

[8] Munro, A.; Towne, D.M.: Productivity tools for simulation-centered training development, Educational Technology Research and Development, 40 (4): 65-80, 1992.

[9] Rickel, J.; Johnson, W. L.: Animated agents for procedural training in virtual reality: perception, cognition and motor control, Applied Artificial Intelligence, 13 (4-5): 343-382, June-August 1999.

[10] Ritchie, J.M.; Dewar, R.G.; Simmons, J.: The generation and practical use of plans for manual assembly using immersive virtual reality. In: Proceedings of the Institution of Mechanical Engineers, Journal of Engineering Manufacture, 213(5): 461-474, 1999.

[11] Wan, H.; Gao, S.; Peng, Q.; Dai, G.; Zhang, F.: MIVAS: a multi-modal immersive virtual assembly system, Computers and Information in Engineering Conference, Salt Lake City, UT, September – October 2004.