SANDIA REPORT

SAND2003-3501
Unlimited Release
Printed November 2003

Automation Tools for Flexible Aircraft Maintenance

William Drotning, David Kozlowski, Clifford Loucks, William Prentice, and Peter Watterberg

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under Contract DE-AC04-94AL85000.

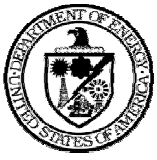
Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online



SAND2003-3501
Unlimited Release
Printed November 2003

Automation Tools for Flexible Aircraft Maintenance

William Drotning, David Kozlowski, Clifford Loucks,
William Prentice, and Peter Watterberg
Intelligent Systems and Robotics Center
Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-1007

Abstract

This report summarizes the accomplishments of the Laboratory Directed Research and Development (LDRD) project 26546 at Sandia, during the period FY01 through FY03. The project team visited four DoD depots that support extensive aircraft maintenance in order to understand critical needs for automation, and to identify maintenance processes for potential automation or integration opportunities. From the visits, the team identified technology needs and application issues, as well as non-technical drivers that influence the application of automation in depot maintenance of aircraft. Software tools for automation facility design analysis were developed, improved, extended, and integrated to encompass greater breadth for eventual application as a generalized design tool. The design tools for automated path planning and path generation have been enhanced to incorporate those complex robot systems with redundant joint configurations, which are likely candidate designs for a complex aircraft maintenance facility. A prototype force-controlled actively compliant end-effector was designed and developed based on a parallel kinematic mechanism design. This device was developed for demonstration of surface finishing, one of many in-contact operations performed during aircraft maintenance. This end-effector tool was positioned along the workpiece by a robot manipulator, programmed for operation by the automated planning tools integrated for this project. Together, the hardware and software tools demonstrate many of the technologies required for flexible automation in a maintenance facility.

1. Introduction

One of the goals of DOE's Robotics and Intelligent Machines Roadmap was to identify classes of processes that could be automated. The Roadmap identified the automation technology bases that underlie non-contact, contact and multi-processes. A Laboratory Directed Research and Development (LDRD) project was begun in FY01 to develop technology aimed at automation for classes of processes. The goals of this particular project were to develop advanced design tools for flexible robotic workcells, to apply these tools for analysis of an automation-intensive maintenance facility for a class of complex systems, and to demonstrate use of the analysis on a key process technology. In addition, the project set an additional goal of developing new hardware technology to support automation of this key process in maintenance operations. The eventual expected benefits of this effort are the increased utilization of facilities and capital equipment, and reduced cost in automation programming, leading to reduced overall maintenance and operations costs.

We envision a facility that is designed for automated operations and can accommodate an entire class of complex systems, including weapon systems, so that the cost can be amortized over multiple systems and processes. The facility will have multi-process automation equipment to reduce maintenance staffing requirements. Figure 1 below shows many of the issues associated with adding flexibility to a maintenance facility. Multiple processes and multiple aircraft within size groups comprise the maintenance spectrum. The initial target environment for this work is military aircraft maintenance, although the resulting tools will be applicable to the design of robotic workcells for other complex systems.

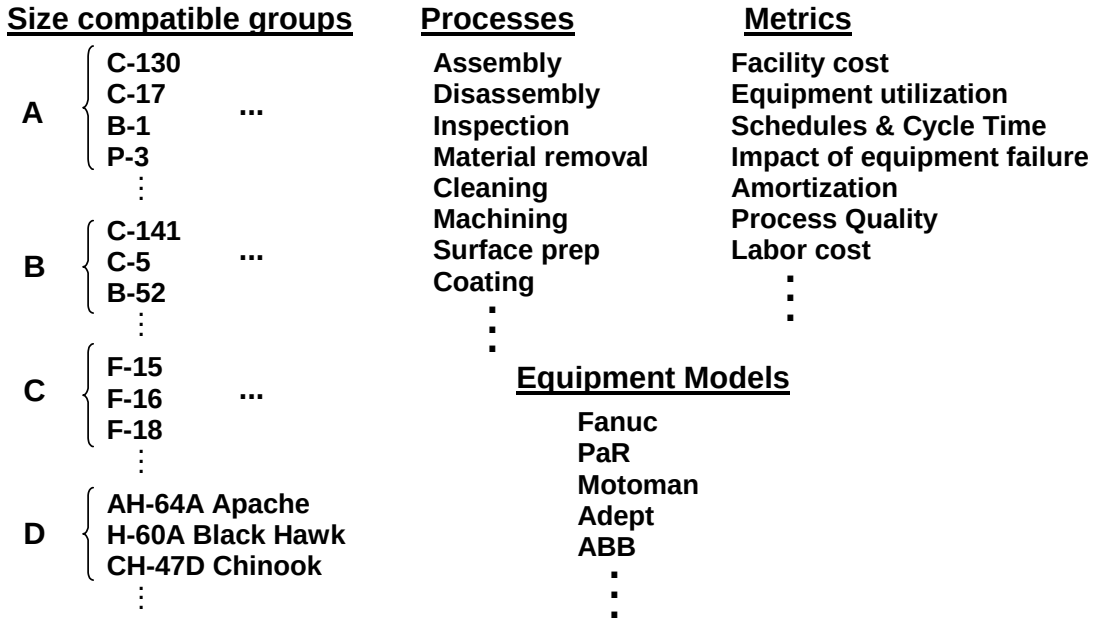


Figure 1. Considerations for a flexible aircraft maintenance facility.

Elements of this vision have been realized for the aircraft painting/coating process and for nuclear weapon component spray cleaning. In developing advanced painting/coating/cleaning systems, we have discovered the need for more advanced design tools; development of such tools was a principle focus of this project. These tools will be targeted at environments requiring many changes in task or workpiece. The tools will allow rapid analysis of a workcell configuration, including both commercial and custom

hardware, for operations on any of a class of systems. The tools will integrate modeling and simulation, automated path planning and generation, and minimization methods to support analysis of workcell architectures. They will allow assessment of designs with variable workpiece position. These tools will use automated methods to minimize the number of robot positions that must be specified by the operator.

The initial task of the current project was a system study of military aircraft depot maintenance to identify high-payoff automation opportunities and to gain an understanding of aircraft maintenance operations. The project team visited aircraft maintenance depots to understand critical needs for automation, and to identify maintenance processes for potential automation or integration opportunities. In the second phase, the task was to continue the process of identifying technology and integration needs, and to identify automation design tool enhancements to fill these needs. The goal was to develop automation design tools for flexible maintenance facilities. The focus was on multiple aircraft size groups, and multiple maintenance operations and processes. In the third phase of the project, a significant maintenance process was selected for technology development and pilot demonstration, showing application of the analysis and design tools.

This report summarizes the accomplishments of the LDRD project 26546 at Sandia, during the period FY01 through FY03. All of the original goals of LDRD project 26546 were met and are reported here. In addition, the identification of a particular technology need, namely automation control during in-contact operations, led to a challenging hardware development effort, also described in this report. The major section headings of the report reflect the three major phases of the project. In section 2, the survey of U.S. military aircraft depots is summarized from the site visits. The third section describes the accomplishments in enhancing and integrating software modeling and analysis tools that led to design analysis of flexible automation facilities for aircraft maintenance. Section 4 demonstrates the application of these software tools for design analysis of a flexible automation workcell for aircraft maintenance operations. From needs identified by the depot site visits and an understanding of the technology gaps hindering the expansion of automation for maintenance, a hardware technology employing automated in-contact operations was selected for early development. Together, the developed hardware system and software analysis tools combined into an integrated demonstration, described in the section 5.

2. Depot Maintenance Survey

In FY02, we completed the task to study maintenance operations of U.S. military aircraft, as related to flexible automation capabilities, needs, and opportunities. The work performed in FY01 had developed a comprehensive compilation of information on the military logistics centers and depots. This information was reviewed and evaluated to select sites for staff visits, based on the types of aircraft served, maintenance processes employed, and technologies in place. In addition, a comprehensive set of questions was developed to guide the site visit teams in the technology surveys. By the end of FY01, the project team had visited three DoD depots that support extensive aircraft maintenance. The three depots visited were Warner Robins Air Logistics Center WR-ALC (Warner Robins, GA); Naval Aviation Depot NADEP-Jax, Jacksonville, FL; and Ogden Air Logistic Center OO-ALC, Ogden, UT. Due to the military activities following the September 11th attack, the team's visit to the Corpus Christi Army Depot CCAD, Corpus Christi, TX, was postponed until FY02. This last visit allowed the team to observe and discuss maintenance operations at another branch of the military, and to expand the scope of flexible automation concepts to rotary wing airframes (helicopters).

The four DoD depot visits resulted in some modification to the original proposal hypothesis that automation technologies are severely underutilized. Generally, existing automation systems are utilized

often, but as expected, these systems are very aircraft-type-specific and do not contain features that support flexibility.

A number of common trends and experiences were observed during the depot visits. Programmatic and political drivers have historically influenced the assignment of specific types and sizes of aircraft to depots. Thus, an efficient grouping of automation processes for size classes of aircraft is often superseded by non-technical drivers. Automation drivers tend to be environmental, productivity, quality, and manpower needs; in any case, a business case with a strong ROI is essential for successful implementation at the depots. Efficiency and efficacy of operation is critical; for example, improving the automated depaint coverage on an airframe from 80% to 90% can be an essential achievement for retaining an automation process. A continuing challenge facing maintenance staff is to reduce the process turn-around time (TAT) in order to improve military systems availability. Workforce issues, such as job skill grades, manpower levels, system maintenance staff, system programmer staff, and operator training are all important considerations for the successful introduction of automation systems into depot maintenance tasks.

On the technical side, there is a common need for addressing the path and task planning, motion, and location of automation systems for maintenance operations on large aircraft. For larger structures, the maintenance automation system may likely be moved around the aircraft, as opposed to statically occupying a large workcell footprint. It is clear that, especially for automation operations on larger aircraft, the reachability analysis tool and tools for automated path generation and registration are needed and will benefit broad process areas such as cleaning, coating, surface preparation, finishing, decoating, sealing, fastener removal/insertion, and inspection. To optimize efficiency, a reachability analysis for the automation system is essential to minimize movement around the aircraft. Once an operational location has been selected, the system must be moved into the selected position, and its location registered with respect to the aircraft. The reachability analysis and robot-to-aircraft registration methods require new technology development to meet the needs seen in the field.

Another theme that was heard at a number of the depot sites is that automation of processes is inhibited by the wide variety of parts to be handled in a process. If the number of part variations and types is high, and the quantity of each part is not large, then the costs for automation -- mostly programming and system maintenance - often are too high to justify automation. Some examples observed were the painting of a variety of smaller parts; preparing aircraft hydraulic tubing systems with customized, multiple bends; and the overhaul of a wide variety of wheel brake assemblies, to name a few. This is clearly an opportunity for development and deployment of the types of advanced automation concepts that Sandia has been developing but that are not yet available in the marketplace. Flexible workcells that perform automated programming of the robot system, based on inputs from part models and sensors, can bridge this technology gap and provide automation solutions for a much broader variety of maintenance operations.

A number of automation and robotics systems for large structures were observed at the depots. Some examples included an automated aircraft wing rework system, a robotic gantry ultrasonic scanner, FlashJet® depaint heads on a mobile platform and a robotic end-effector, a laser automated decoating system, robotic media blast depaint systems, and robotic painting systems.

The depot site visits revealed that the originally-proposed hardware demonstration concept, that of an automated de-rivet and rivet operation that would be used for skin replacement on wings, for example, was not a suitable choice for LDRD laboratory development. This technology is already commercially available, and is in use at Warner-Robins ALC in Georgia.

However, several Sandia automation technologies in development are applicable to depot maintenance needs. Reachability analysis software tools use 3D models to determine the dexterous working volume

and optimize workcell layout. This will reduce setup and programming time, increase coverage on the aircraft, and reduce process duration. Automated path generation methods create task paths and robot motions from 3D models and process rules and constraints. This results in reduced programming time, optimized robot motions, and minimized process cycle times. Sensor systems can be used for robot-to-aircraft registration, and for in-contact operations, where compliance is achieved through feedback to the sensor-based motion control of the robot or its tool. Actively compliant force-controlled tools, for example, lead to improved process quality for in-contact operations on curved surfaces. For operations on large aircraft, use of mobile robot platforms has the advantage of not requiring the expensive movement of aircraft during processing at the depot.

From the depot survey and prior experience, a list of aircraft maintenance operations was generated. Figure 2 lists many of the processes amenable to automation, grouped by whether the maintenance tool is in contact, in proximity to, or not in contact with the workpiece. Many of the non-contact operations have been successfully automated. There is a need for improved automation control for in-contact operations, which comprise a broad spectrum of required maintenance processes.

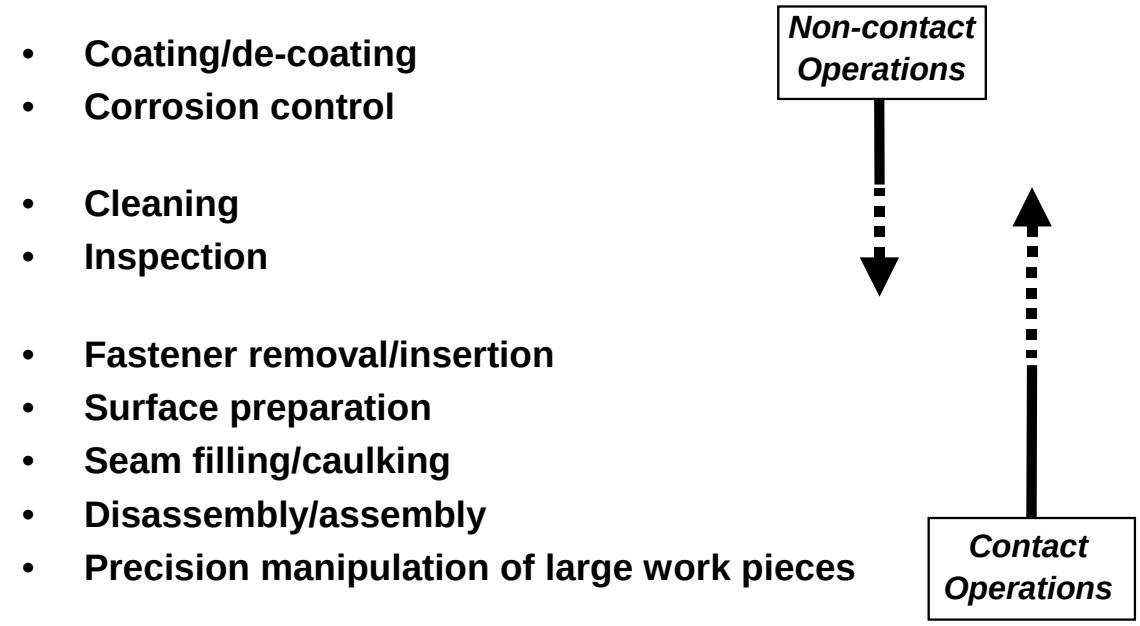


Figure 2. Processes for a flexible automation workcell

A candidate process technology was selected for development and demonstration. The process of precision surface finishing and preparation appears to be an important and broadly needed technology where automation has been hindered by lack of advanced technology. It is also representative of a wide variety of in-contact processes that may employ model-based position analysis, sensor-based workpiece characterization, robot motion control, and process monitoring/feedback. Development of a hardware demonstration of this technology will be described in section 5.

In early FY02, the results of the depot site visits and technology assessment were presented at the Defense Manufacturing Conference, DMC'01.

3. Software Tools for Maintenance Facility Analysis

Software tools for design analysis were improved and extended to encompass greater breadth for eventual application as a generalized design tool. The design tools for automated path planning and generation have been enhanced to incorporate those complex robot systems with redundant joint configurations, which are likely candidate designs for a complex aircraft maintenance facility. By employing advanced inverse kinematic solution methods and optimization techniques, the prior design tools for automated path generation have been enhanced to provide greater flexibility and generality, as well as improved performance for complex robotic systems. As a by-product, the results of this current work were implemented in the F-117 and F-22 coatings projects.

The development of flexible automation design analysis tools focused on integrating and enhancing the automated reachability analysis tool and the automated path generation and optimization tools. The goal was to build these into a framework that is accessible to engineers in a more robust and generalized analysis tool, and eliminates the *ad hoc* nature of past implementations of these tools. To this end, the project team integrated these tools into Umbra [GOTTLIEB-2001], an application framework for advanced controls, simulation, and visualization. From prior tools, the robot-specific knowledge needed to be separated from application knowledge, so that new robot objects can be rapidly implemented without modification to the process application or user interface software. The final product is a demonstration of the integrated design tool, which shows the flexibility of the tool for analysis of maintenance operations. The demonstration will comprise the analysis of an automation facility (robot workcell) containing two different workpieces (*e.g.,* P-3 and B-2 aircraft) for operation with two different processes, such as sanding (contact) and RAM coating (non-contact).

The software analysis algorithms that this project has developed fall into three different areas: collision-free path planning, task planning and reachability analysis. In collision-free path planning, the software is given starting and ending configurations for a robot from which a path is computed for the robot to follow that gets from one to the other while avoiding all known obstacles in the workspace. The task planning part of the software computes a path for the robot to follow that accomplishes a specific task such as sanding or painting a given surface geometry. The reachability software displays a volume which can represent either where the robot can be placed to accomplish a given task or (if the robot is fixed) where the object might be placed so the robot can reach all necessary parts. These three parts will be described in detail below.

All three of these software components are based on previous *ad hoc* software that had been developed for specific projects. The challenge for this project was to see if the software could be generalized so that it could be readily used for any robot and for any geometry. Further, it was important that each package could all use the same robot and geometry descriptions so as not to have to reformulate the data to use different components.

The first step was to isolate the robot specific functions of each of the three modules from the generic algorithms. Looking at all the required robot functionality, the solution to the common interface problem was to create a C++ class to describe a basic robot. This class instantiated most of the functionality that a robot class needed to provide for the three modules. Any individual robot simply inherits from this class and adds a few routines and parameters necessary to complete the functionality to support the three primary modules. This class will be described in a later section. Each of the three main components was then rewritten to use this general robot class.

The idea of a generalized, independent, C++ class that can represent robots for projects across the entire Intelligent Systems and Robotics Center has been around for a long time among various groups. The class

developed for this project has stimulated the discussion of what that class should look like. It is currently filling that role for several projects at this time and will probably serve as the basis for such a center-wide class.

3.1 Collision-Free Path Planning

The central algorithm for this task was implemented from a paper by Kuffner and LaValle for ICRA 2000 [KUFFNER-2000]. It is called RRT for rapidly expanding, random tree. In brief, the algorithm tries to explore the space around the start location and the goal location. It grows a tree at each that describes where the robot can move and not collide with anything. At each step in the iterative algorithm, it picks a random point in the robot's configuration space (the space of all its possible joint values) and attempts to go there from either the start tree or the goal tree. It finds the closest point in the existing tree and tests to see if it can move in a direct line from the tree to the random point. As it steps along toward the random point, it adds the points at which there is no collision to the tree it is trying to expand. From the farthest point it reaches (possibly all the way to the random point) it tries to connect to the closest point in the other tree. If it fails, it then tries to grow the other tree. RRT is wonderful in its robustness and its simplicity relative to previous algorithms. Either in spite of or because of its simplicity, it is also significantly faster than previous algorithms. Its advantage grows exponentially as the number of degrees of freedom of the robot increases.

RRT requires only four interface routines with the generic robot class. First, it needs to know if the robot is in collision at a given set of joint values. Second, it needs to have the robot provide random points in configuration space. Third, it needs the robot to provide a trajectory from one point to another. And finally, it needs the robot to tell it how far it is between two points in configuration space. If we restricted the planner to work with only with robots that have Euclidean spaces and trajectories that are linear in the space, the planner would only need the collision routine. The rest could be calculated by the planner and are provided as a default. However, this would eliminate non-holonomic robots and robots for which the optimum trajectory is not linear in configuration space. For normal robots, these routines are implemented once in the generic parent class and are only modified if there are abnormal constraints.

3.2 Task Planning

Task planning is the process of creating a series of locations and orientations (call it a coordinate frame, control frame, tag or just frame) in world space that will cause a task to be accomplished if the robot moves from one to the next while some tool (e.g. a sander or paint gun) attached to the robot is turned on. The task planner for this project is designed to cover a surface area that is not highly curved with such frames. There are many parameters that can be adjusted to suit the desired process requirements. Precursors to this algorithm have been used for painting and spraying cleaning solvent. It is currently being used to do sanding tasks as well and is expected it to handle most generic tasks where area coverage is required.

The task planner can be controlled via its parameters. One parameter controls how far off the surface the tool is placed. For spraying and blasting operations, this can be whatever is appropriate for the given task. For contact operations like sanding, it is set to zero. The control frames are placed along the surface in stripes. Stripe spacing can be set. The distance between frames along a stripe can also be set. One area for improvement would be to have the planner only put out tags along the stripe when the surface has curved a certain amount. The amount of curvature required to trigger a new frame would be a settable parameter.

When the operation in question is a non-contact operation such as painting, it is generally optimal for the tool to be directly over the point on the surface to which the spray is being directed. However, in tight corners, this may not be possible and the tool will have to be offset so as not to collide with surrounding objects. How close the tool is allowed to get to surrounding objects is a settable parameter. In addition, the maximum offset allowable is settable. If the tool cannot be placed without exceeding the maximum

offset, an unavoidable collision is detected and the planner aborts. This requires a change of parameters or surrounding geometric constraints and a replanning.

The primary direction of each stripe with relation to the surface is calculated by default but can be changed if desired. In addition, each stripe can be offset a given distance from the place it would nominally cover. This allows planning multiple passes, each slightly offset from the previous. This can be useful in ensuring that the coverage for painting or sanding is more uniform.

In some operations such as painting, it is necessary to make sure the tool has moved completely off the surface before the tool is turned off, moved in another direction, or slowed down. This distance is called the kick distance and is added to each end of a stripe. It can even be made negative if it is desired that no part of the tool ever leave the surface. Generally, it is desired that the tool move at a constant velocity. When changing from one stripe to the next or before the first or after the last, it is necessary to accelerate or decelerate the tool off the stripe so that it is moving with the desired velocity as it engages the stripe or does not speed up or slow down until it has left the stripe. There is a parameter that sets how many frames are placed off the ends of the stripe to bring the tool to a halt or get it moving. This can be zero if needed.

One of the main improvements to the task planner that must be addressed in the future is the ability to track surfaces of high curvature. At the moment, the orientation for each stripe is the same and based on the orientation of the surface as a whole. If the surface is highly curved, this orientation becomes inappropriate as the surface curves away and distortions or coverage gaps may ensue. Algorithms have been identified to address this shortcoming and can be implemented subject to future funding.

3.3 The Reachability Tool

This is one of the main advances to promote flexibility in design and use of robot workcells. Once a task plan is completed, all the frames are placed that will get the task accomplished. But whether the robot can reach them or not hasn't been taken into account. The task plan is useless if the robot cannot execute it. The reachability tool computes and shows graphically the volume of space in which the robot can be placed and reach all the task frames. If the robot is fixed, it can show the volume of space in which the surface can be placed at a given orientation and have the robot reach everything. This is a critical function in getting the most out of the robot and workcell.

This tool may be even more useful in the design process for creating flexible workcells. As stated above, it can show where to place things but what is more important to the designer, it can show how the reachable volume is effected by changes in tool design or robot kinematic design, or even robot selection. This can make a dramatic difference in the usefulness of the workcell. Or, it can help pinpoint what parts of the system need to have more options such as the set of tools and their capabilities or the length of track that the robot might move on.

3.4 The Generic Robot Class

As mentioned above, to support the three software components, a generic robot C++ class was created. The functions that need to be provided by the generic class and those classes that represent actual robots will be discussed here. To support the RRT path planner, a robot class must provide functions to return a random point in configuration space, a trajectory between two points in configuration space, the distance between two points and whether or not the robot is in collision at a point. The task planner has an option to check whether the robot can reach a given frame. To support this, a robot class must provide an inverse kinematics routine. This is also the only routine that is required for the reachability tool. The generic base class provides default routines for random point, distance and trajectory functions for robots whose configuration spaces are Euclidean spaces with fixed limits in each dimension. This is true for most standard robots. Hence, a specific robot could be implemented for these tools by writing only three routines. There must be an initialization routine that sets the number of degrees of freedom (dimensions

of configuration space) and the limits of travel of each joint. There must also be a routine that checks for collision at a given point. (The generic class manages the list of obstacles that might cause collisions.) And there must be an inverse kinematics routine.

Writing these three routines would be all that's necessary to support the analysis tools described here. However, the entire package of tools and robots is embedded in a larger system, Umbra, and to make the robots really usable, a few more functions should be provided. The users of the system will want to see the robot displayed in their world. So, a VRML file describing the geometry of the robot should be provided. Since some representation of the geometry of the robot is usually needed to do the collision checking, this is virtually always present already. It just needs to be read into Umbra's scene graph. A generic function is provided for this and all the specific robot class must do is provide the file name. A function to change the displayed model's joint values should be provided to visually display the current configuration of the robot. Most robots are used in conjunction with various tools mounted on the last joint. As a convenience, a mount function is written so the user simply states what Umbra model is to be attached.

4. Application of Software Design Analysis Tools to an Aircraft Maintenance Process

4.1. Flexible Maintenance Software Tool Design and Implementation

The software tool functionality was developed to include the ability to have a variety of robots do a variety of work on a variety of aircraft parts (which could be expanded to any type of part). This would necessitate the use of many of the Intelligent Systems and Robotics Center's key technologies. The objective was to then bring them together into one functioning group. The challenge was to find or design an environment that allowed these key technologies to be integrated together in a relatively simple and timely way.

The environment chosen was Umbra, a software tool developed in the Center. The gains from using Umbra were many. First, Umbra is an application framework that allows for the rapid development and testing of reusable software modules. It also is a powerful graphics program that facilitates the rendering of the simulation models and work functions that the tool has to perform. Lastly, it was used as the foundation of the GUI that the software tool is built under.

One of the key developments was that of the Umbra COTS robot class. Up to this point, robots in the Umbra environment were complex mechanisms made up of separate modules that communicate using connectors. We saw the need for an integrated Umbra robot class that would contain common attributes and functionality of a robot such as geometry, joints, and inverse kinematics, along with others. We also wanted this robot class to be easily loadable from a user selection so that the tool could accommodate a variety of COTS robots. This new class was designed and implemented, and became the foundation of the software design and implementation within the Umbra framework. In addition to the robots used in the present project, these robot classes have also been applied to a funded Mobile Robots project. We were able to model their mobile robot and build in the appropriate functionality to allow them to quickly substitute this robot class into their Umbra project and gain the added functionality provided.

Once the robot class was defined, many of the other necessary technologies such as reachability and work (collision-free path) planning were incorporated into its functionality. Various test parts were then modeled and saved as VRML part files. The user interface was designed such that the operator had the option of loading a variety of developed COTS robots along with a variety of test parts. With the use of

process parameters for the task and path planning functions, different types of work can then be planned on the parts, such as painting and sanding, simply by changing key parameters.

Umbra works from the concept of reusable modules that can stand-alone or communicate using standard connector classes. It also uses the popular scripting language Tcl as its front end to instantiate, parameterize, and connect modules. This allowed for our modules to be developed rapidly, and for testing and integration to be a dynamic process.

4.2. Software Tool GUI Description

At the center of the GUI is the Umbra scene graph. At the top are menus that allow for the loading of robots and parts, along with various other functions such as reachability and work path editing. On the right side are areas that list the major pieces as they are brought into and displayed in the Umbra scene graph. This area consists of a list of devices which includes each robot that has been loaded, a list of each test part that has been loaded, and tag series and points for work paths that have been planned.

A key feature of the GUI is to allow the operator maximum flexibility in configuring and reconfiguring a workcell. Once a workcell is configured, the operator can plan a work path on the workpiece and do reachability analysis on robot placement. Any of these major elements can be replaced or repositioned for a new analysis, as described below.

4.3. Workcell Analysis Process Example

When the system first comes up the operator is presented with an empty scene containing only the world coordinate axes. Typically the first step is the loading of a robot to be analyzed. Opening the file menu, where the operator is presented with various submenus including “Load Robot”, does this. Once “Load Robot” is selected, a standard file browser dialog is presented allowing the operator to go to the directory where robot VRML files are stored. Once the desired robot is selected, the Umbra robot module is created and the robot model is displayed in the scene as shown in Figure 3 for the Fanuc S430iW used in our sanding demonstration.

The Umbra scene is a true three-dimensional representation of the workcell. As such, the scene can be rotated and viewed at any angle, along with zoom in and out, using mouse bindings. At the top right of the screen is position and rotation information of the currently selected (highlighted) item. The item’s position and rotation can also be changed using other key/mouse bindings to allow for maximum flexibility.

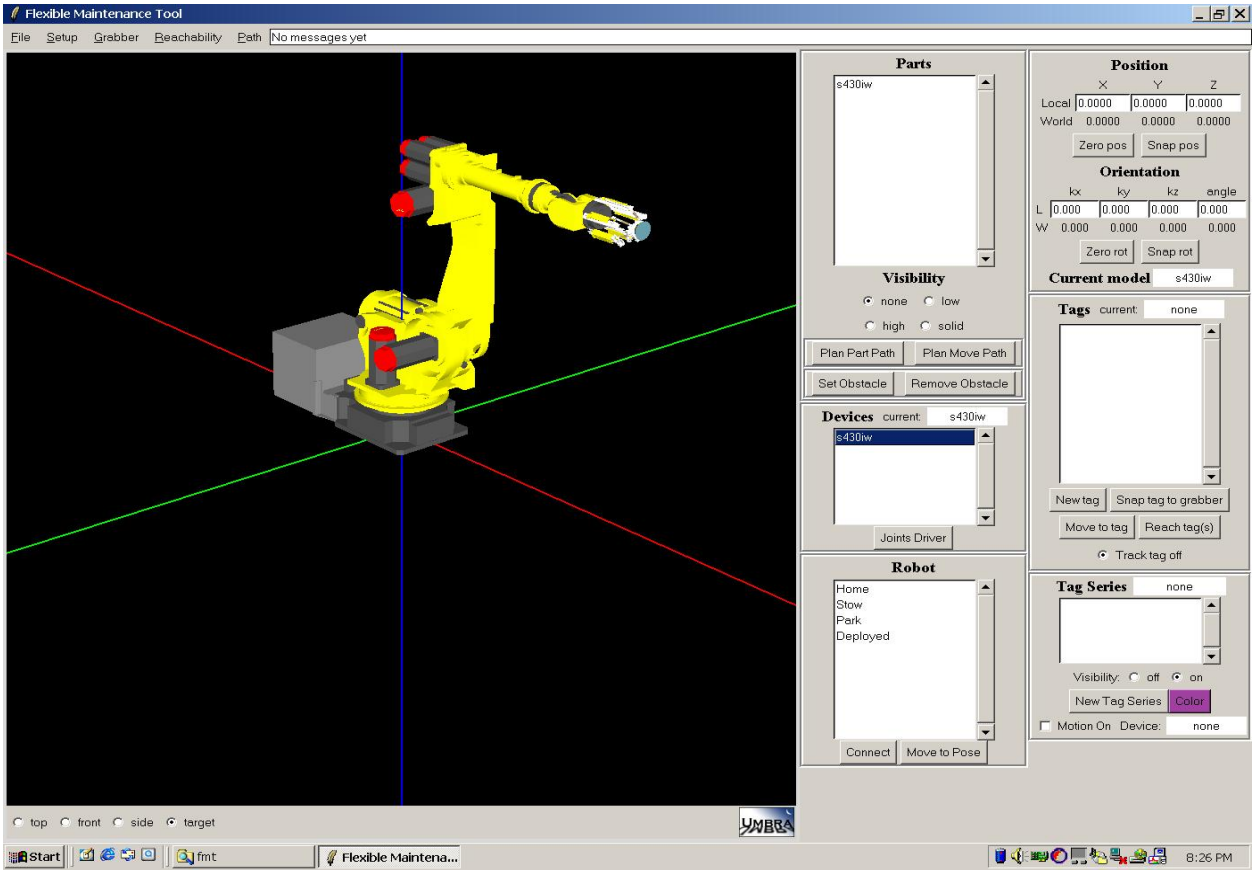


Figure 3. Fanuc S430iW robot model in Umbra scene.

The next step typically is to load a workpiece using the same File main menu and selecting the "Load VRML Part" submenu. Once the part is loaded, it appears in the active workcell scene as well as its name in the parts list. The same manipulation options apply to the part as to the robot. The aircraft part used in the sanding demo is shown loaded in Figure 4 (it actually appears as three parts: a main part and two sanding areas). The currently selected (highlighted) part is show in yellow in the workcell scene.

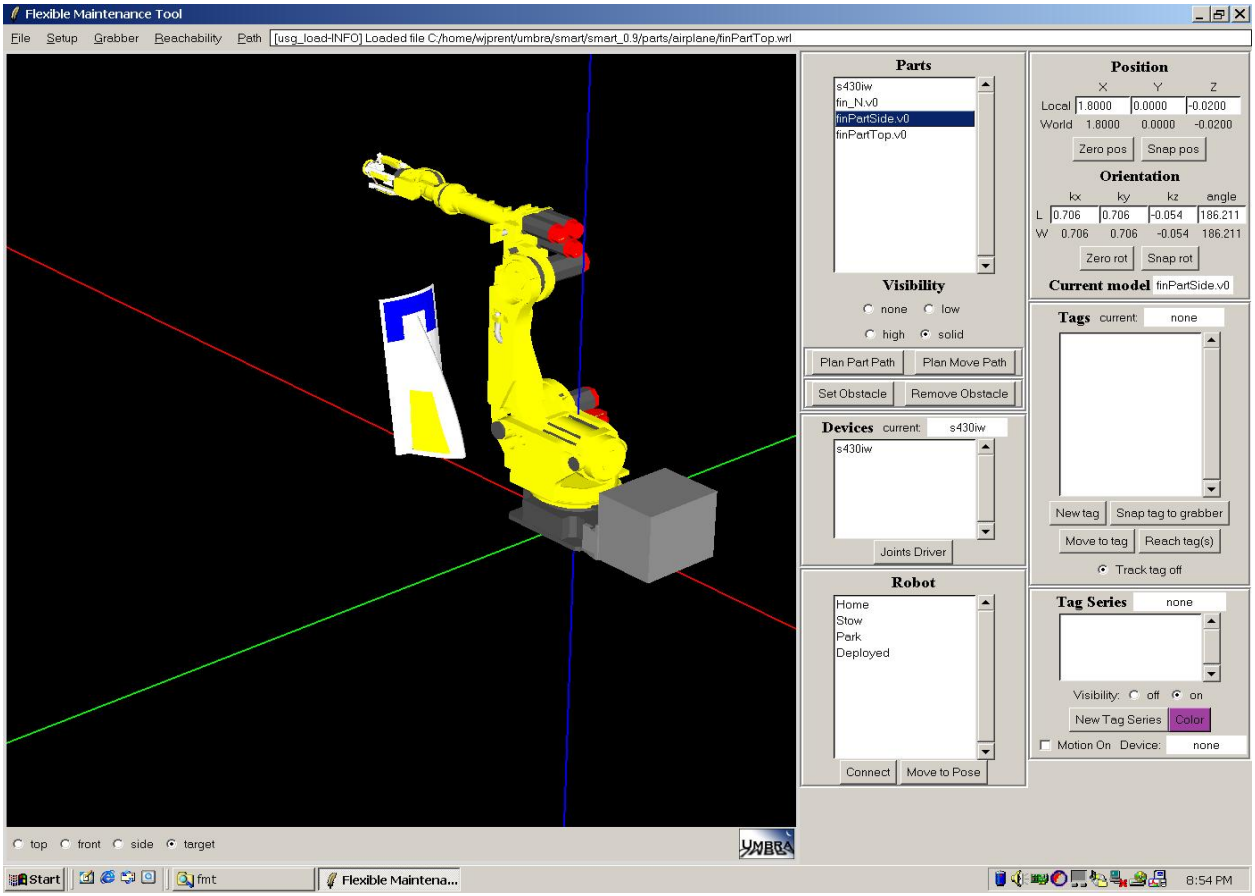


Figure 4. Workpiece part added to Umbra scene.

A work path can now be planned on the desired workpiece. If the path edit function is enabled using the Path menu, key parameters can be changed to allow for various work paths to be generated and the desired one chosen. The "Plan Part Path" button in the Parts frame is used to generate the work path. Once the path has been generated, a tag representation is displayed on the workpiece, as shown in Figure 5. Note that a new tag series with individual tags are now displayed in the "Tag Series" and "Tag" lists, respectively.

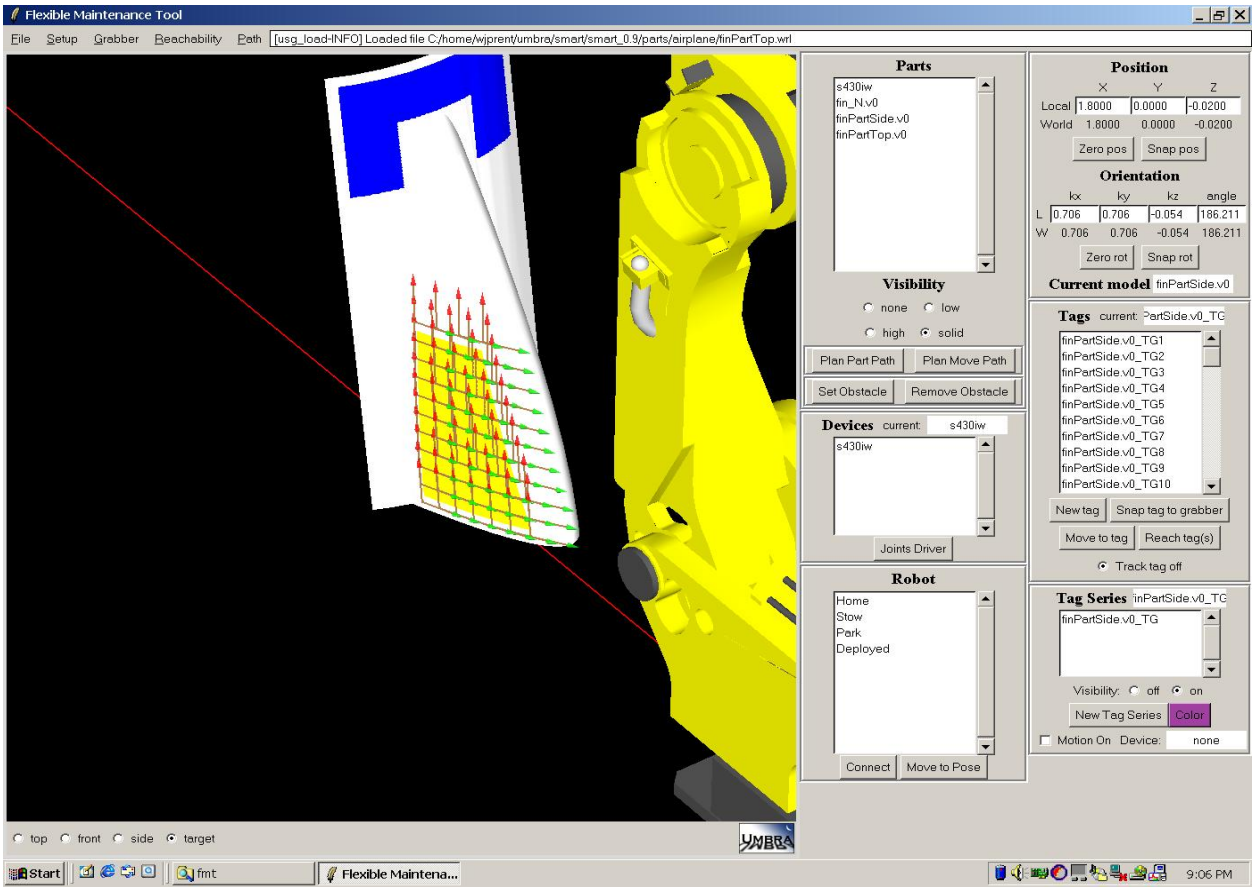


Figure 5. Tag points showing the work path on the workpiece.

A reachability analysis can now be done on the work path tag series for the current robot using the "Reach Tag(s)" button in the Tags frame. A reachability cloud is generated as a default, or, by turning restricted Z on under the Reachability menu, only a surface representing a slice of the reachability volume is generated at the robot's current base Z position, as shown in Figure 6.

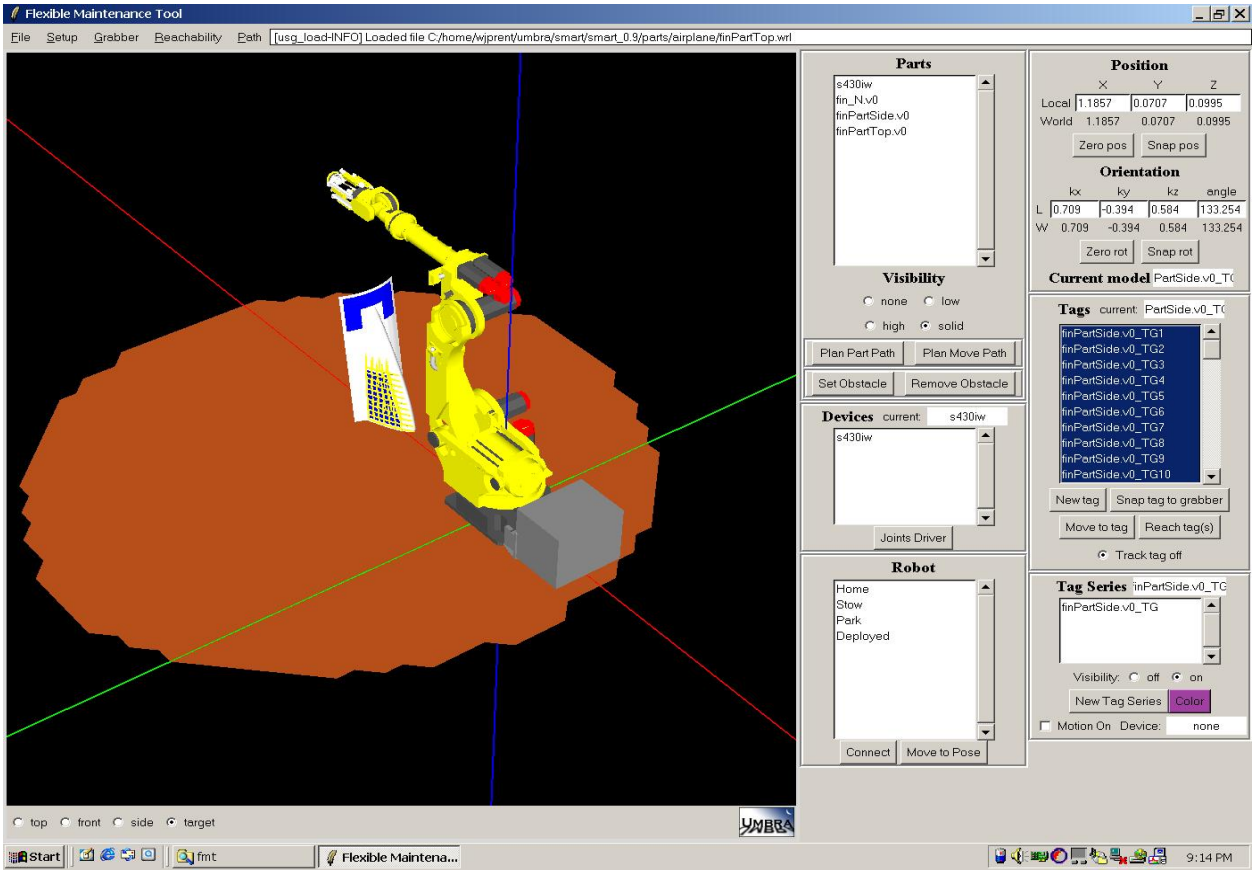


Figure 6. Surface generated from reachability analysis.

The robot's inverse kinematics are used to generate the reach volume.

A collision-free robot joint path can now be generated for robot approach, departure, and work motion. These paths can be viewed in the simulation workcell using the "Motion On" checkbox to track tags. The robot's sequence of joint positions is also saved to disk for future downloading to the robot controller. Figure 7 shows the robot at a mid-path sanding pose while executing the sanding work path in simulation.

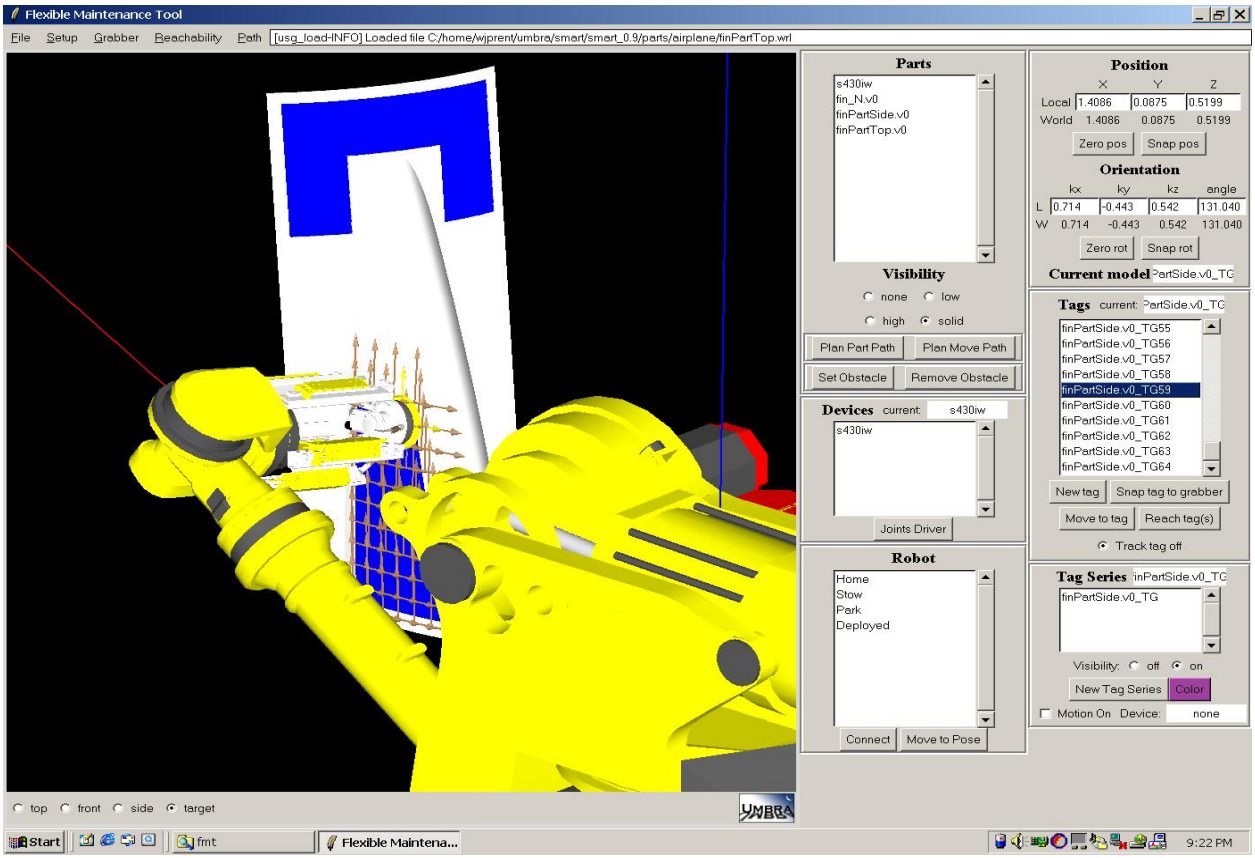


Figure 7. Test in simulation of the generated sanding work path.

After checking all robot motion using simulation, the robot sanding work path is run on the actual robot sanding the real aircraft workpiece. The aircraft workpiece is secured in a fixture so that its position relative to the robot matches that of the model. As previously mentioned, the joint path file is saved in a format compatible to the R-J3 controller so that it can be executed. The current programmatic, online interface with the R-J3 controller is only compatible with the Visual Basic language, and cannot be linked with our tool. We would like in the future for either Fanuc to make their interface compatible with the C++ environment, or to have a Visual Basic interface to the R-J3 controller written so that the joint path can be streamed down to the robot and run online. Our communication method at this time is to use the Fanuc PC File Services to download the saved joint path program, as described in another section.

4.4 Future Tool Enhancements

Other key technologies exist in the Center that should be integrated into the Flexible Maintenance Tool's environment, many of which have already been integrated into the Umbra environment and are available as Umbra modules. We have demonstrated flexibility in three areas. The first is the ability to bring multiple robots into the work environment for analysis and comparison. The second is the ability to do multiple work functions with those robots (painting, sanding, etc.) using our configurable work path planning. The third is the ability to load different workpieces into the system for rapid processing.

At the present time, the workpieces have to be modeled offline and then loaded into the system. We currently have Umbra technology in place that will use laser scanning (structured lighting) to scan a workpiece and then create its model geometry in the Umbra scene. The dynamically created workpiece model will also contain accurate positional information to be used with the work path planners. This

would allow for a workcell to have the flexibility to bring new workpieces in and have the system recognize and dynamically configure itself to process the new piece.

5. Development of an Automation Tool for Aircraft Surface Preparation

For this project, a surface finishing ("sanding") device with active compliance was developed as an end-effector for a robot manipulator arm. Active compliance was achieved by attaching a portable sanding tool to a 6-DOF parallel kinematic manipulator, called the "hexapod". Section 5.1 describes the hexapod design, while 5.2 describes the implementation of the hexapod tool with the robot manipulator arm as an integrated demonstration.

5.1. Analysis of the 6-DOF Hexapod

5.1.1 Introduction

A serial manipulator consists of several links connected in series by various types of joints, typically revolute or prismatic. Although the positional workspace can be large for these types of manipulators, maneuvering appropriate sized payloads can only be accomplished by the use of gear trains on the motor drives. This presents a myriad of other technical problems when the serial robotic device attempts to "interact" with the environment, as discussed below.

Conversely, a parallel manipulator typically consists of a distal moving platform that is connected to a fixed base by several limbs. For example, a parallel manipulator with an n-DOF platform is connected to a fixed base through n independent kinematic chains each having a single actuated joint. Parallel manipulators possess the advantage of high stiffness, low inertia, and large payload capacity. These come at the expense of relatively small workspace, hardware design difficulties, and difficult control issues. In addition, the only closed form forward kinematics solution that has been reported in the literature is for special mechanical forms of the Stewart-Gough platform. The general Stewart-Gough platform forward kinematics solution requires numerical techniques.

Historically, in 1949 Gough [GOUGH-1962] developed the first full 6-DOF parallel structure (a six-linear jack system for use as a universal tire testing machine). In 1965 Stewart [STEWART-1965] popularized it for use as a flight simulator for pilot training, and his name is now associated with this device. Since that time many variants [MERLET] of the Stewart Platform have been researched, but industrial application has been fairly limited to machine tools, flight simulators, and micromanipulators. On the other hand, their serial link chain counterparts have seen widespread growth and are now applied to many industrial applications. Yet, other than "guarded moves", the implementation of contact sensor information to serial mechanisms for compliant control contact problems has not matured. There are several reasons for this, with the most prominent listed below.

- Large geared serial manipulators present a myriad of other force control issues such as coulomb and viscous friction, backlash, large moving mass, etc.
- Implementation of a force control loop around a position controlled robot leads to slow and sluggish performance; poor assembly cycle times result. In addition, contact instability is common if control gains are too high.
- Elegant solutions for teaching and assembly search strategies are required for effective use on the factory floor.

5.1.2 Kinematics

The number of degrees of freedom that the distal platform exhibits, and the type of limb configuration, typically define a parallel manipulator. The numbers of degrees of freedom that a manipulator exhibits are typically given by Grubler's formula described below [TSAI-1999]:

$$F = \lambda(n - j - 1) + \sum_i f_i \tag{1}$$

F: degrees of freedom of a mechanism.
f_i: degrees of relative motion.
j: number of joints in a mechanism, assuming that all the joints are binary.
n: number of links in a mechanism, including the fixed link.
λ: degrees of freedom of the space in which a mechanism is intended to function.

Limb configurations of parallel manipulators are typically made up of a combination of revolute (R), prismatic (P), or spherical (S) joints attached together to make up a kinematic chain (an assembly of links connected together by joints). The numbers of degrees of freedom for each of the above joints are 1, 1, and 3 respectively (a Hooke, or universal, joint is made up of 2 R joints).

Thus, for a typical 6-DOF Stewart-Gough parallel manipulator we have 6 links made up of an R-R-P-S type of configuration. Plugging this into Grubler's formula (1) yields

$$F = 6(14 - 18 - 1) + (6x3 + 6x2 + 6) = 6\,. \tag{2}$$

For the ParaDex [KOZLOWSKI-2002] type manipulator currently under investigation, we have the exact same equation.

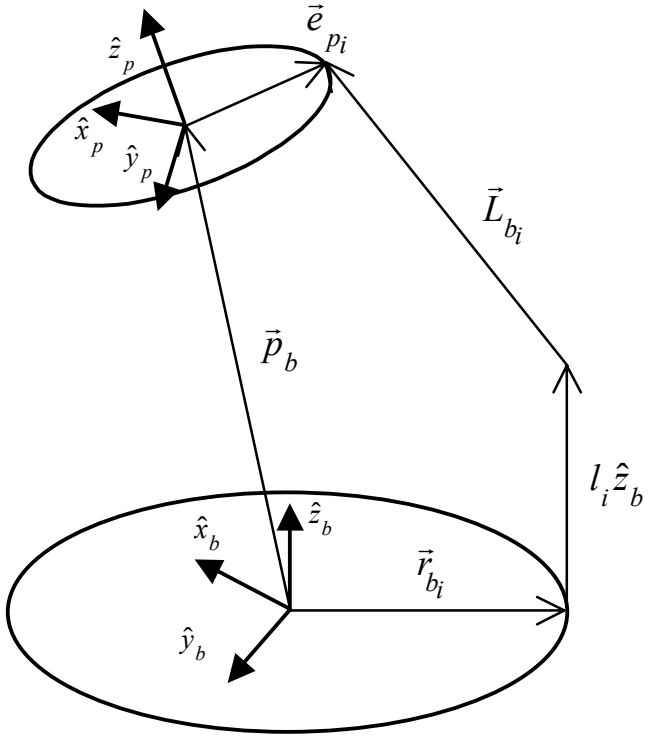


Figure 8: ParaDex 2 kinematics, I =1-6.

The inverse kinematic solution can be derived from the drawing in Figure 8 and simple vector arithmetic as seen below

$$\vec{p}_b + \vec{e}_{p_i} = \vec{r}_{b_i} + l_i\hat{z}_b + \vec{L}_{b_i} \tag{3}$$

where $l_i\hat{z}_b$ is the actuator position.

Rearranging equation (3) and transforming $\vec{e}_{p_i}$ into the base frame yields,

$$\vec{L}_{b_i} = \vec{p}_b + {}^b_pR\vec{e}_{p_i} - \vec{r}_{b_i} - l_i\hat{z}_b \tag{4}$$

$$= \vec{c}_i - l_i\hat{z}_b \tag{5}$$

where $\vec{c}_i = \vec{p}_b + {}^b_pR\vec{e}_{p_i} - \vec{r}_{b_i}$ and ${}^b_pR$ is the 3x3 direction cosine matrix relating the platform to the base frame. Taking the inner product of (5) yields

$$L_i^2 = c_{ix}^2 + c_{iy}^2 + (c_{iz} - l_i)^2$$

$$|c_{iz} - l_i| = \sqrt{L_i^2 - c_{ix}^2 - c_{iy}^2}$$

$$l_i = c_{iz} - \sqrt{L_i^2 - c_{ix}^2 - c_{iy}^2} \tag{6}$$

Thus, given a desired platform position and orientation, the active link length vectors of the actuator are uniquely determined.

5.1.3. Jacobian Analysis

For an arbitrary parallel manipulator, the Jacobian is defined as [TSAI-1999]

$$\underline{\dot{l}} = \underset{\sim}{J}\underline{\dot{x}}$$

or,

$$\begin{bmatrix} \dot{l}_2 \\ \dot{l}_3 \\ \dot{l}_4 \\ \dot{l}_5 \\ \dot{l}_6 \\ \dot{l}_1 \end{bmatrix} = J \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \\ \dot{z}_b \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{7}$$

where $\underline{\dot{l}}$ is the actuator velocities and $\underline{\dot{x}}$ is a column vector of linear and angular velocities of the platform relative to the base frame (the dual definition for the serial manipulator is $\underline{\dot{x}} = \underset{\sim}{J}\underline{\dot{\theta}}$ where $\underline{\dot{\theta}}$ is a vector of joint velocities and $\underline{\dot{x}}$ is as before). This can be derived for the ParaDex style parallel manipulator by differentiating equation (3) relative to the base frame.

$$\dot{l}_i\hat{z}_b + \vec{\dot{L}}_{b_i} = \vec{\dot{p}}_b + \vec{\omega}_b \times \vec{e}_{p_i} \tag{8}$$

Since $\vec{L}_{b_i}$ is of fixed length, we know $\vec{L}_{b_i} \bullet \vec{\dot{L}}_{b_i} = 0$. Thus, taking the dot product of equation (8) with $\vec{L}_{b_i}$ yields

$$\dot{l}_i\left(\hat{z}_b \bullet \vec{L}_{b_i}\right) = \vec{\dot{p}}_b \bullet \vec{L}_{b_i} + \left(\vec{\omega}_b \times \vec{e}_{p_i}\right) \bullet \vec{L}_{b_i}$$

or

$$\dot{l}_i = \left(\frac{1}{\hat{z}_b \bullet \vec{L}_{b_i}} \right) \left(\vec{L}_{b_i} \bullet \vec{\dot{p}}_b + \left(\vec{e}_{p_i} \times \vec{L}_{b_i} \right) \bullet \vec{\omega}_b \right) \tag{9}$$

Equation (9) can be rearranged such that $\dot{l} = J\dot{x}$ ($\dot{l}$ and $\dot{x}$ are defined above), and

$$J_i = \left[\frac{\vec{L}_{b_i}^T}{\hat{z} \bullet \vec{L}_{b_i}}, \frac{\left(\vec{e}_{p_i} \times \vec{L}_{b_i} \right)^T}{\hat{z} \bullet \vec{L}_{b_i}} \right] \tag{10}$$

is the i[th] row of the Jacobian.

Since the Jacobian matrix (along with its inverse and transpose) is a mapping that relates forces and velocities between the actuator space and the end effector, it is a natural starting point for analyzing the "dexterity" of a manipulator design. There are many optimization design criteria for Jacobian analysis [TSAI-1999, STOUGHTON-1993, ST.-ONGE-1996], but the main design parameter involves the condition number of the Jacobian. The condition number of an arbitrary matrix A (denoted $\kappa(A)$) is defined as the ratio of the maximum to minimum singular values. Thus, $\kappa(A)$ is a measure of a manipulator's ability to generate velocities and forces independent of direction [STOUGHTON-1993]. The desired $\kappa(A)$ for a Jacobian is to be as close to unity as possible over the entire working volume of the manipulator for the above condition to occur (the last 3 rows of the above Jacobian must be scaled by the radius of the platform for appropriate results). At singular positions for a parallel manipulator, the device gains one or more degrees of freedom and loses stiffness completely; in this case, the scaled $\kappa(A)$ approaches a very large value. Thus, the inverse of the Jacobian does not exist.

5.1.4. 5-DOF and 6-DOF Implementations

Utilizing the above formulation of the Jacobian, the original idea was to develop a 5 degree of freedom parallel manipulator (5 active links), with a constraint on the platform roll axis essentially freezing a degree of freedom of the manipulator. For processes that do not require roll axis manipulation (such as for a rotary sanding tool), the reduction from 6-DOF to 5-DOF can result in a significant saving in tool weight. (A technical advance (TA) has been submitted for the 5-DOF variant of the hexapod.) Thus, optimization analysis was done on a 5x5 Jacobian, with l_i a 5x1 vector, and $\dot{x} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \omega_x & \omega_y \end{bmatrix}^T$ since $\omega_z = 0$ for all time (constrained). This is summarized below:

1. initialize the radius of the platform, and the radius of the base
2. constrain the attachment points (base and platform) of one passive link, which also constrains the passive link length
3. utilizing Monte Carlo methods, allow the rest of the link attachment points to "float," and calculate $\kappa(A)$ for each "design", storing all design information for $\kappa(A) < 2$ at the home position of the manipulator
4. throw out all designs not feasible
5. test $\kappa(A)$ over the central workspace region for feasible designs

Due to the kinematic coupling between the links and the platform roll axis, this design procedure yielded a 5-DOF platform with a singularity at the home position. Thus, the original analysis was flawed. While reasons for this result (discussed below) were researched, the platform was modified to add a sixth linear actuator making the final system a 6-DOF parallel platform -- a known stable configuration.

The original 5x5 Jacobian that was used in the above analysis was modified to include the effects of ω_z, and by adding a fictitious rotational motor whose only "function" is to control the center tool roll axis. Thus, the Jacobian now takes on the form shown below:

$$\begin{bmatrix} \dot{l}_1 \\ \dot{l}_2 \\ \dot{l}_3 \\ \dot{l}_4 \\ \dot{l}_5 \\ \dot{\theta}_6 \end{bmatrix} = \underset{\sim}{J} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \qquad \underset{\sim}{J} = \begin{bmatrix} \underline{J}_1 \\ \underline{J}_2 \\ \underline{J}_3 \\ \underline{J}_4 \\ \underline{J}_5 \\ \underline{J}_6 \end{bmatrix} \qquad \begin{aligned} \underline{J}_i &= \left[\frac{\vec{L}_{b_i}^T}{\hat{z} \bullet \vec{L}_{b_i}}, \frac{\left(\vec{e}_{p_i} \times \vec{L}_{b_i}\right)^T}{\hat{z} \bullet \vec{L}_{b_i}} \right] \\ i &\ni [1,2,3,4,5] \\ \underline{J}_6 &= [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1] \end{aligned} \tag{11}$$

An analysis of the above Jacobian does indeed show the singularity of the 5-DOF device, as $\kappa(A)$ was well above 1e06 for the device as originally built. In order to confirm that a 5-DOF design solution does exist, the above Jacobian optimization analysis was run, with results shown below.

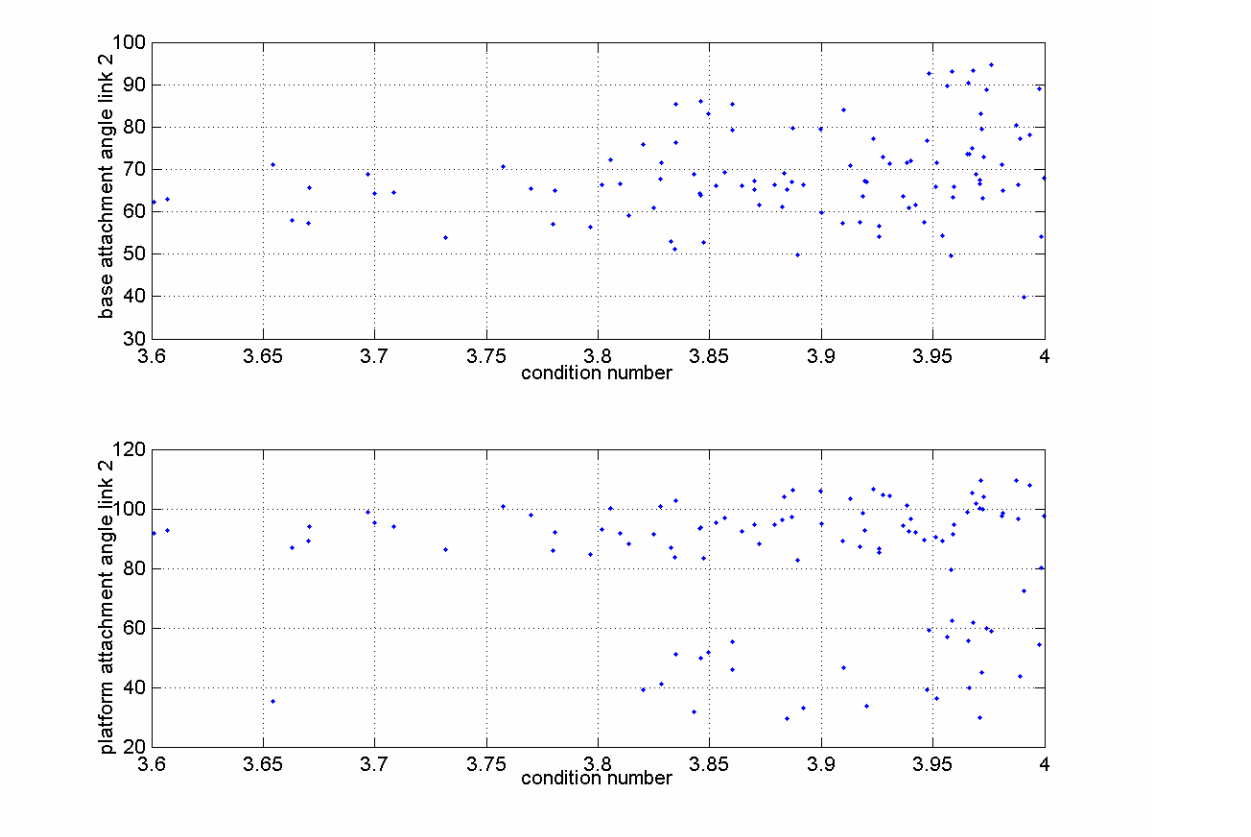


Figure 9: Different base and link attachment points versus condition number for link 2.

Figure 9 and Figure 10 illustrate how the condition number of the Jacobian changes for different base and platform passive link attachment points of a 5-DOF parallel manipulator. (Link 5 is symmetric to link 2; link 3 is symmetric to link 4. The link attachment points are shown in Figure 11.) The best condition number achieved for this optimization analysis was 3.6 (with 1 being the optimum).

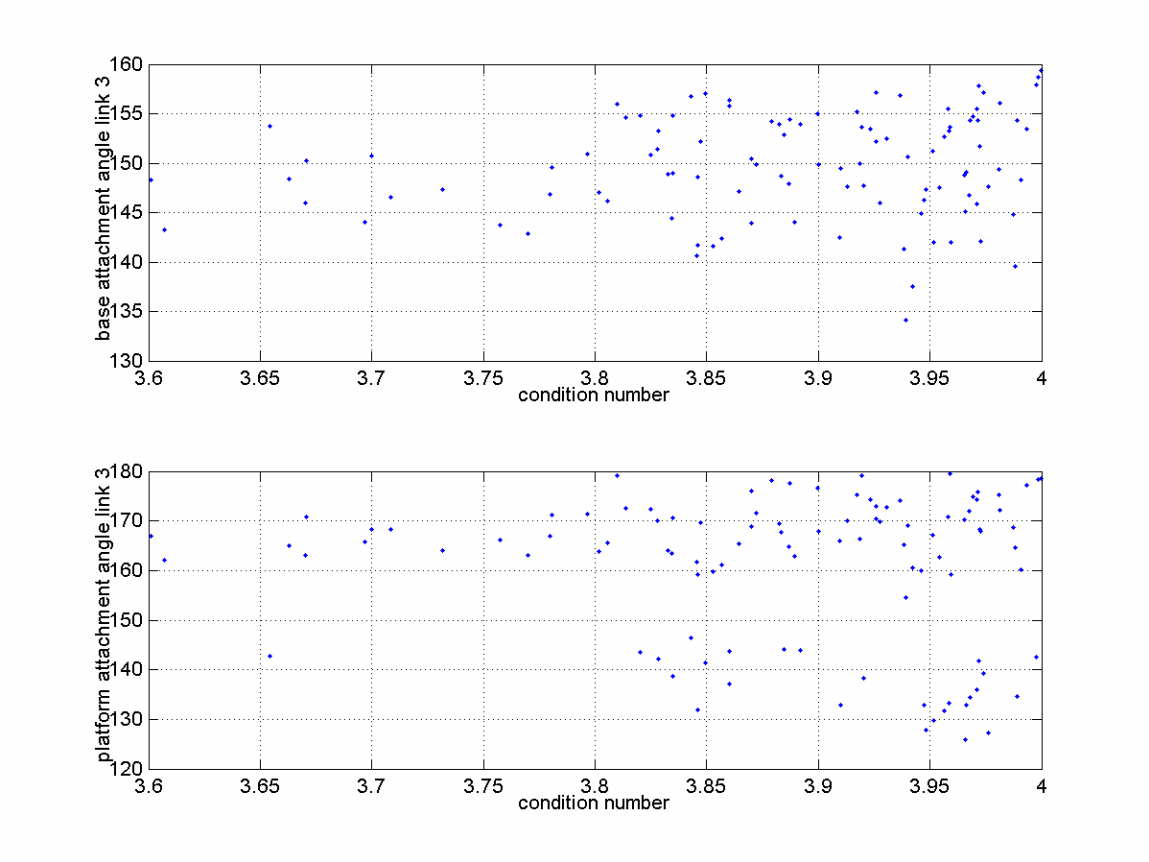


Figure 10: Different base and link attachment points versus condition number for link 3.

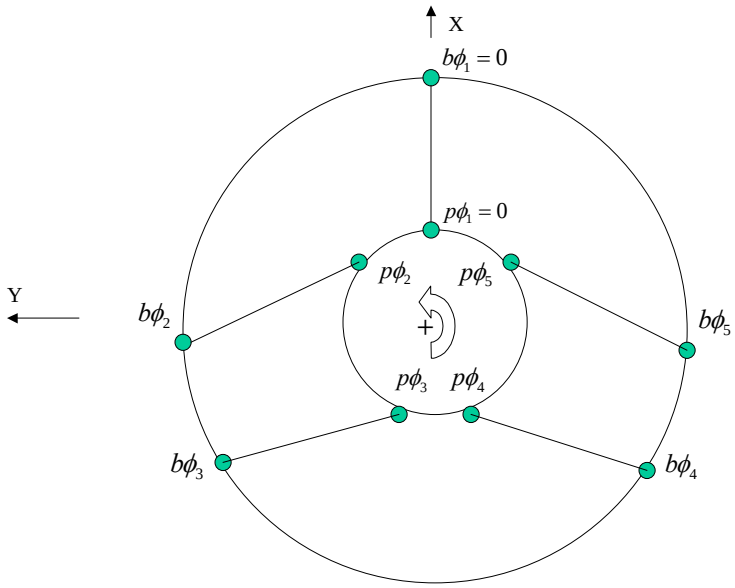


Figure 11: 5-DOF base and platform link attachment point definitions.

Thus, the updated optimization analysis for the 5-DOF parallel manipulator yielded design parameters shown in the following table:

Passive link number	Passive link length (mm)	Base Attachment Angle (degrees)	Platform Attachment Angle (degrees)
1	88.22	0	0
2	108.40	62	92
3	96.93	148	167
4	96.93	-148	-167
5	108.40	-62	-92
Base radius		**Platform radius**	
145.71 mm		**101.6 mm**	

Table 1. 5-DOF parallel manipulator optimization analysis results.

5.2. Robotic Workcell with Hexapod End-effector

A robotic workcell was developed using an actively compliant in-contact tool mounted on the robot manipulator. Together with the software analysis tools for path planning, the workcell demonstrates the application of the analysis tools to generate a path, which was then used by the robot to drive the movement of the force-controlled in-contact tool on a demonstration aircraft workpiece.

5.2.1. Robot Workcell

The robot workcell was built around a Fanuc S-430iW robot and is shown in Figure 12.

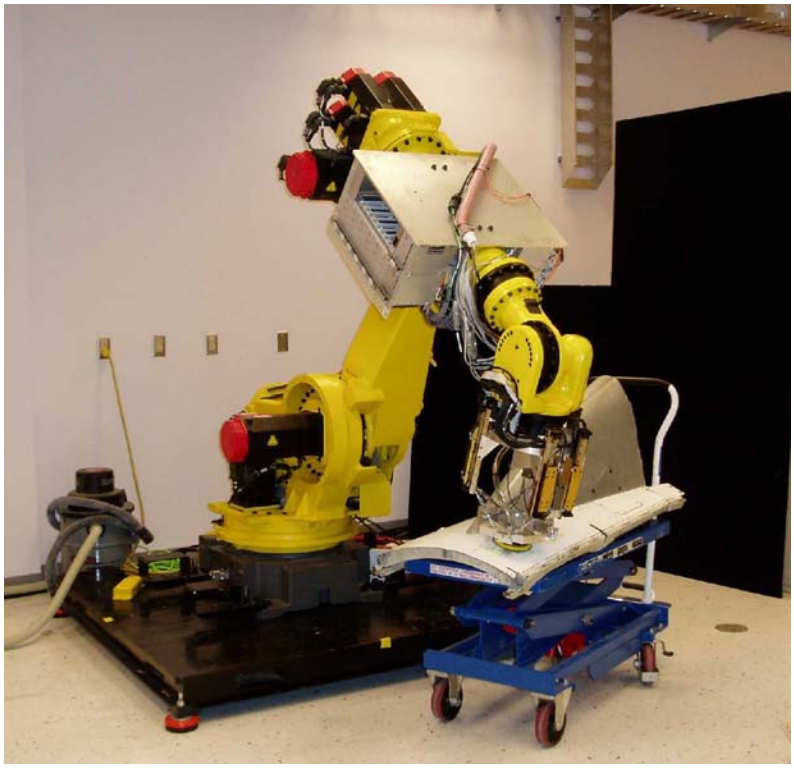


Figure 12. Robot workcell, showing robot, hexapod end-effector, and workpiece.

This large, six-axis robot is designed for material handling and welding tasks. The robot is a typical serial link manipulator design with 6 axes. It has a 360 lbs. working load and a reach (to the faceplate) of 104 in. The specified repeatability is 0.012 in.

The controller is the R-J3 model from Fanuc with version 5.2 of Fanuc's operating system and Karel programming language. The controller is responsible for interpreting the sanding paths, specified as tool positions in a Cartesian coordinate frame, and solving the inverse kinematic equations to compute the required joint positions to get the tool to these poses. The controller also uses digital I/O signals to enable the sanding head and to tell the hexapod when to begin and end its contact force-controlling mode.

The hexapod sanding end-effector is carried by the robot. The current system divides the sanding tasks into two parts. The robot is used to drive the head through the sanding path, while the end-effector adjusts the contact force of the sanding head against the part. The robot thus performs the gross positioning while the hexapod compensates for part curvature and positioning errors. These errors come from robot inaccuracy, uncertainty in the model of the workpiece, and part fixturing errors.

5.2.2. Downloading and Running a Sanding Path

The Fanuc R-J3 controller includes an ethernet interface. A Windows-based utility called PC File Services is available to perform tasks such as backing up the controller software (including user programs) to a PC and transferring files from the PC into the controller's memory.

The sanding path is created by the path planning software on the PC and output to a text file as a sequence of points, defined by X, Y, Z, Yaw, Pitch, Roll in a World coordinate frame. Through a manual point and click operation, we send this file into the robot controller's RAM disk memory. (This manual operation could be automated through an Active X interface on the PC. However, we used the manual technique since there is not currently a path from the Umbra environment to the Active X level.)

With the file in the robot controller's memory, a robot program is executed to move the sanding head along the path. This program, written in Fanuc's Karel programming language, begins by setting up the motion parameters for the robot, such as tool tip speed, using linear interpolation between path points, and blending of the path segments together with constant speed through each point. It then opens the file and reads the first line, which specifies the number of points in the file. Next it reads the first point and executes an approach move toward that point and stops there. This allows the robot to get in a starting position that is the same pose as the first point on the path.

The hexapod controller now reads the values from the six load cells and uses those readings as offsets so that the gravity loading (which varies with tool orientation) is biased to zero. In the current configuration, we are limited to sanding paths with no change in tool Pitch (i.e. no change in gravity loading). We have, however, left provisions to incorporate a 3-axis accelerometer into the hexapod that will allow gravity compensation.

With the hexapod ready to go, the operator signals the robot to continue. The robot now drives to the first point in the path. Just before the robot reaches that point, the sander is turned on and the hexapod begins its contact force control mode. The robot now continues reading subsequent points from the file and moving through them in a smooth, continuous motion.

When the last point in the path is reached, the hexapod is signaled to stop its force control mode and the sanding tool departs from the surface and the sander is turned off. The robot is then driven to a "safe position" and halted.

5.2.3. The Hexapod Sanding End-effector and Controller

The sanding end-effector, shown in Figure 13, is a parallel kinematic mechanism design commonly referred to as a Stewart platform. In this 6-DOF design, there are six parallel linear actuators who's individual motions are coordinated to produce the desired motion (or static force) of the platform. The sanding head is a commercially available, pneumatic, random-orbital unit from Dynabrade. It has a 5 in.-diameter sanding pad, to which various grit sanding disks are adhered.

The linear actuators are a package (motor, encoder, and slide) from Trilogy, using their moving coil design and Linear Encoder Modules. Each actuator has 3.4 in. of travel and is capable of 10 lbs. of force. This design places the six actuators parallel to each other and uses passive links to connect the motor slides to the platform. Traditional Stewart platforms use the linear actuators as the links themselves. We chose the current design to help keep the end-effector more compact. The passive links have universal

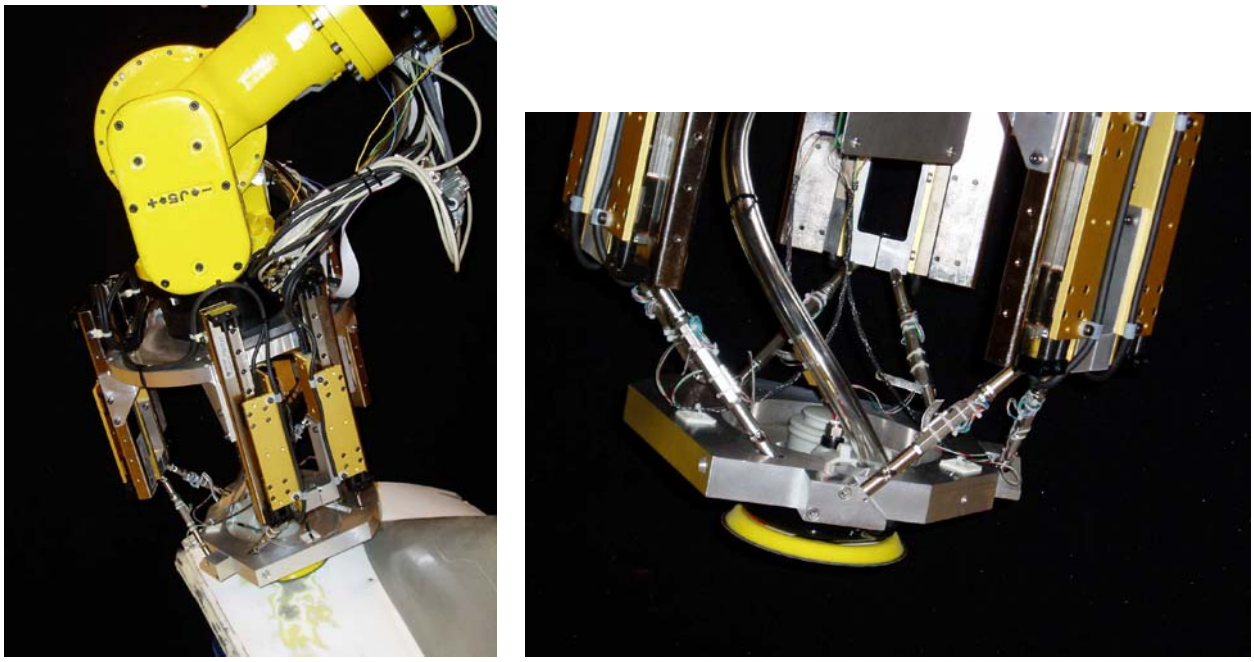


Figure 13. Two views of the hexapod sanding end-effector.

joints at each end. A thrust bearing is also incorporated into the platform so that the links have 3 degrees of freedom at the platform and 2-DOF at the motor slide. The u-joints and bearings are off-the-shelf models available through McMaster-Carr. The range of motion of the sanding head on the moving platform is approximately 3 inches vertically, a lateral motion of about 2 inches, and pitch and yaw angles of ±12 degrees.

To measure the contact force on the sander (and therefore the platform), each link includes a load cell from Entran. These are strain gauge based units packaged as small, flat disks with 10x32 studs. With the 2- and 3-DOF mounting of the passive links, the links are in pure tension or compression. The cells measure this load, which can then be converted into the net loading on the sanding head using the Jacobian of the parallel kinematic mechanism. Amplification and filtering electronics are mounted to the base of the hexapod. The conditioned signals then to go an A/D converter and then on to the PC.

Servo control loops are closed around each motor by UMAC axis controllers from Delta Tau and amplifiers from Western Servo Design. Using the encoder for feedback, the Delta Tau boards command the motor velocities through sinusoidal commutation of the motor's 3 phases.

A PC with both Windows and QNC boot partitions sits at the top of the hexapod control hierarchy. Windows is used for software development while QNC, a real-time UNIX variant, is used when running the hexapod. A fiber-optic link connects the PC with the Delta Tau controllers and A/D boards. All control and feedback signals are multiplexed through this link.

5.2.4. Integrated Robotic Workcell

The sanding path generated by the path planning system includes the tool position and surface normal at each path point. The robot controller drives the tool through each point in succespsion and uses the surface normal to orient the tool at each point. But the robot's inaccuracy and errors in modeling and positioning of the workpiece will result in both force and torque errors at the sanding head.

To control the net force on the sanding head (typically 2 pounds of force normal to the surface), the tension/compressions measurements in each link are converted into the net force of the platform (and sander) using the Jacobian of the structure. The Jacobian is a 6x6 matrix formed from the kinematic equations of the parallel mechanism. The Jacobian relates the six motor velocities to the platform velocity. The transpose of the Jacobian relates the six motor forces to the net platform force. Once the net force is known, the error from the desired force is computed. The inverse of the Jacobian transpose is then used to compute the forces needed from each motor. These calculations are repeated for each update cycle.

The net torques around the axes of the tool coordinate system at the platform are also calculated along with the net forces along the tool axes. The reaction to the spinning sanding disk produces a torque around the tool Z-axis. This torque is monitored but no compensation is done on it explicitly. Any net torque around the tool X or Y axes means that the sander is not normal to the surface being worked. By driving these torques to zero, the sander can be maintained normal to the surface.

The position of the platform within its own working volume must also be monitored. With a parallel mechanism, freedom of motion is greatly reduced as the platform moves away from its central (or Home) position. Sanding force compensation means that the platform must be driven away from its Home. If this drift becomes too great, it would be desirable to have the robot offset it's path to compensate. While this capability will not be very difficult to implement, it is beyond the scope of this demonstration project.

6. Conclusion

The project team visited four DoD depots that support extensive aircraft maintenance in order to understand critical needs for automation, and to identify maintenance processes for potential automation or integration opportunities. From the visits, the team identified technology needs and application issues, as well as non-technical drivers that influence the application of automation in depot maintenance of aircraft.

Software tools for automation facility design analysis were developed, improved, extended, and integrated to encompass greater breadth for eventual application as a generalized design tool. The design tools included automated path planning and path generation, integrated into a single application framework for interaction, design, analysis, and control.

An actively compliant 6-DOF tool was designed, analyzed, and developed based on a Stewart platform design. A 5-DOF concept was also designed, optimized, and analyzed. A robotic workcell was developed using the 6-DOF actively compliant in-contact tool mounted on the robot manipulator. Together with the software analysis tools for path planning, the workcell demonstrates the application of the analysis tools to generate a path, which was then used by the robot to drive the movement of the force-controlled in-contact tool on a demonstration aircraft workpiece.

From this investigation, several areas for follow-on research were identified. These include: 1) extension of the design analysis tools in order to automatically extract geometric sections and patches from workpiece models for process-based path planning; 2) integration of optical scanning methods for input of workpiece surface geometry; 3) advanced kinematic tools that perform stability analyses for parallel mechanisms; 4) incorporation of accelerometer sensors in the 6-DOF hexapod to implement gravity compensation for the tool's response; 5) sensory feedback to implement process control of sanding effectiveness (*e.g.* area of coverage on curved surfaces).

7. Bibliography

[GOTTLIEB-2001] Gottlieb, E., R. Harrigan, M. McDonald, F. Oppel, and P. Xavier, *The Umbra Simulation Framework*, SAND2001-1533. Sandia National Laboratories, Albuquerque, NM, June 2001 (UNCLASSIFIED).

[GOUGH-1962] Gough, V.E. and S. G. Whitehall, *"Universal tire test machine"*, Proc. 9th Int. Tech. Congr. F.I.S.I.T.A., May 1962, p. 117.

[KOZLOWSKI-2002] Kozlowski, D. M., R. S. Stoughton, R. Hebbar, and W. S. Newman, *"Automated Force Controlled Assembly Utilizing a Novel Hexapod Manipulator*", Proc. World Automation Congress, 2002.

[KUFFNER-2000] Kuffner, J. J. and S. M. LaValle, *RRT-Connect: An Efficient Approach to Single-Query Path Planning*, Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, 2000, pp. 995-1001.

[MERLET] Merlet, J-P, INRIA, http://www-sop.inria.fr/coprin/equipe/merlet/Archi/archi_robot.html.

[ST.-ONGE-1996] St.-Onge, B. M., and C. Gosselin, "Singularity Analysis and Representation of Spatial Six-DOF Parallel Manipulator," in *Recent Advances in Robot Kinematics*, pp. 389-398, 1996.

[STEWART-1965] Stewart, D., *"A Platform with Six Degrees of Freedom",* Proc. Inst. Mech. Engrs., Vol. 180, No. 15, 371-386, 1965.

[STOUGHTON-1993] Stoughton, R. S., and T. Arai, *"A Modified Stewart Platform Manipulator with Improved Dexterity"*, IEEE Transactions on Robotics and Automation, Vol. 9, April 1993.

[TSAI-1999] Tsai, L.W., 1999, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators,* John Wiley & Sons, New York, N.Y.

Distribution:

1	MS0323	D. L. Chavez	LDRD Office
1	MS1002	S. Roehrig	15200
1	MS1004	R. Harrigan	15221
1	MS1004	P. Watterberg	15221
1	MS1007	L. Shipers	15272
3	MS1007	W. Drotning	15272
1	MS1007	C. Loucks	15272
1	MS1007	W. Prentice	15272
1	MS1007	D. Kozlowski	15272
1	MS9018	Central Technical File	8945-1
2	MS0899	Technical Library	9616