# ADACS - An Automated System for Part Finishing

*Keith Stouffer, John Michaloski, Bob Russell and Fred Proctor*

**National Institute of Standards and Technology**
**Robot Systems Division**
**March 1993**

*Abstract* – **This paper describes an automated finishing system called the Advanced Deburring and Chamfering System (ADACS). ADACS uses the Real-Time Control System (RCS), a hierarchical controller architecture that was developed at the National Institute of Standards and Technology. ADACS uses a graphical user interface that prompts an operator to specify chamfering edges and cutting parameters for the part. Given the operator-designated chamfering edges, ADACS uses this edge information to extract features – such as inside corner – to generate a finishing process plan. ADACS interprets the finishing plan to generate motion trajectories that are tightly coupled to the tooling control. Because of the inaccuracies in robotic position control, ADACS uses active force control in the tool to compensate for any small position errors along the finishing path. A prototype ADACS has successfully processed aerospace test parts.**

## 1. INTRODUCTION

Large material removal required to machine parts has always been the job of powerful machine tools. Once a part has been machined, a finishing operation is usually required to perform small material removal (removal of excess material or burrs) to bring the part into tolerance of the specification. The primary finishing processes are deburring and chamfering. In the past – and still presently – this finishing step has been accomplished manually with a hand held spindle grinder. As expected, there are problems with manual finishing of parts. The manual finishing process is inconsistent and inaccurate, leading to irreparable part damage that results in increased machining costs. Manual finishing is also very time consuming. Pratt & Whitney has estimated that 12% of the total machining hours are devoted to manual deburring and chamfering [4]. In addition, manual finishing increases health care costs that result from the cumulative trauma of using the hands in a small work area for great lengths of time. The most notable of these health–related illnesses is carpel tunnel syndrome.

Automation of the finishing process would prove to be very beneficial. At the present time, manual finishing accounts for 12% of the total labor cost and that approximately 10–30% of the manufactured parts need rework after the manual finishing process; by automating the finishing and chamfering process, tolerances could be held to less than 0.07 mm (0.003 in), the finishing costs could be reduced as much as 50%, and the rework rates could be eliminated – assuming a practical and efficient automated finishing system. Automating through traditional teach programming is impractical since it is tedious, time consuming, and prone to inaccuracies. For complex geometries, such as arcs and splines, hundreds if not thousands of points need to be taught along the surface for a robot to perform the trajectory accurately. Therefore, a usable autonomous finishing system must have the capability to use CAD models to generate the necessary robot trajectories based on this knowledge.

Within the Automated Manufacturing and Research Facility (AMRF) at the National Institute of Standards and Technology (NIST), the Advanced Deburring and Chamfering System (ADACS) is a prototype autonomous finishing workcell that uses operator-designated features from graphical CAD models to generate robot trajectories. Working in cooperation with United Technologies Research Center (UTRC) and Pratt and Whitney, ADACS has been developed as a finishing workcell which is capable of processing aerospace parts made from hard materials such as titanium and inconel. The ADACS finishing system serves as an intermediate processing facility between the machining operation and the soft brush finishing operation. The primary objective of the ADACS is to remove the majority of the material from a part's edge in a controlled and timely manner. The ADACS must produce the intermediate, 45 degree break edge or chamfer for part edge geometries such as modified and full radii.

This article was prepared by United States Government employees as part of their official duties and is therefore a work of the U.S. Government and not subject to copyright. This article references certain commercial equipment, instruments or materials and such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

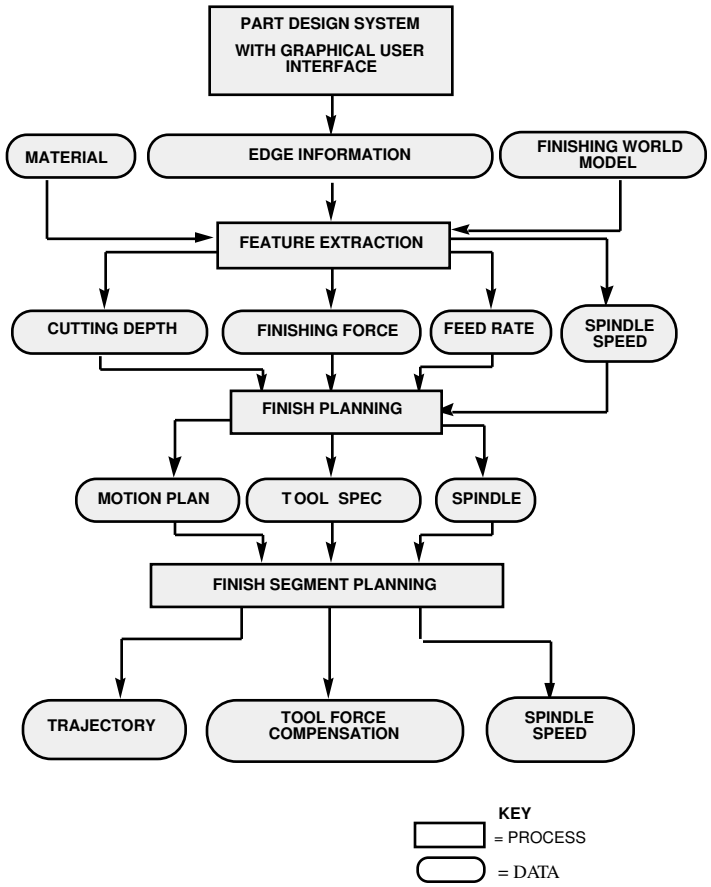


FIGURE 1. ADACS Finishing Process Model

ADACS is a second-generation deburring cell which utilizes ongoing technology developed in the AMRF to deburr and chamfer hard materials with active hard–tooling to compensate for robot errors [8]. Features of ADACS include:

- operator–controlled, off–line graphical user interface exploiting CAD part models to direct chamfering
- automated extraction of features from edge data
- feature and part based chamfering process model
- tightly coupled coordination of tool forces and motion control to achieve ramping and smoothing
- functionally–based position transformations to achieve non-linear and sensor–based trajectory motion
- active tooling to compensate for small position errors and to maintain a constant cutting force

In the following sections, we discuss the requirements, features, components, and demonstration scenario of the ADACS system. Section 2 presents the functional requirements for chamfering. Section 3 describes the details of the ADACS system architecture. Section 4 reviews a demonstration system and a test scenario for chamfering Pratt & Whitney aerospace parts.

## 2. ADACS DATA FLOW

The machining process takes a workpiece and transforms it into a part. Given a machined part, the finishing operation removes any remaining material from the machining operation. The finishing operations of concern in this paper are chamfering and deburring. Figure 1 reviews the data flow within a typical ADACS finishing operation.

At the top level, or part design system, we will assume that a part model exists in a boundary representation. Using a graphical rendering of the part model and a mouse, an operator selects part edges to finish. Edge information including starting and ending points and the two normals of the surfaces that construct the edge are extracted from the CAD data. These normals are required to determine the correct tool orientation to produce a 45 degree chamfer on the edge. The collection of edge information as well as material and finishing model information is sent to the feature extraction subsystem.

The feature extraction subsystem accepts part edge information and extracts part feature information. Edge knowledge is determined from the collection of edge information, including concavity, starting and ending tool orientation, and the state transition. The state transition between edges yields features such as “inside corner” versus “outside corner.” Further, state transition studies include the transition from free–space to contact as well as

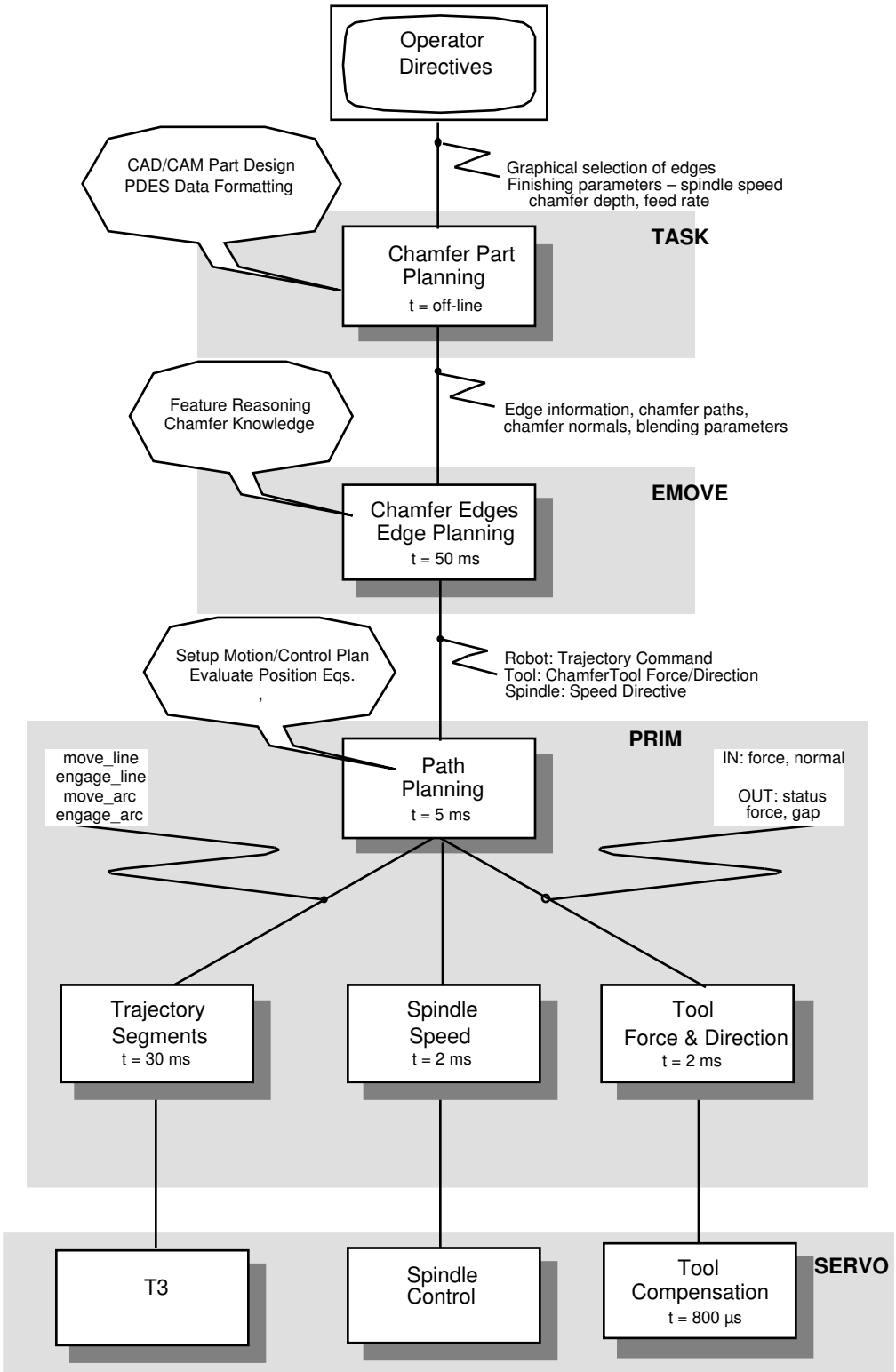


Figure 2. ADACS Control Architecture

continuous versus discontinuous features. In addition, the state transition from free–space to contact imparts ramping requirements on the motion and tooling control.

The finish planning subsystem generates a process model that relates the amount of material removed by the cutter to the tool cutting force, chamfer depth, feed rate, and spindle speed.

The finish segment planning subsection generates motion primitives based on the chamfer features defined in the system world model. The system must account for proper machine tool setup and fixturing, proper tooling, and account for any interference from the fixturing. Each motion primitive must then in turn be transformed into coordinated position and force–control motion segments. These segments are then in turn transformed into a series of set points, or trajectories, and are downloaded to the robot controller. The required tool force compensation along with the spindle speed are sent to the tool controller.

## 3. ADACS SYSTEM ARCHITECTURE

At the National Institute of Standards and Technology, ongoing study into the design and development of intelligent control systems is taking place. This work has led to the Real–Time Control System (RCS) – a concurrent architecture for real-time control applications [1]. RCS specifies an architecture of an application to be constructed as a hierarchy of communicating controller nodes. Within RCS, task decomposition and equipment composition are used to derive the architecture consisting of the hierarchy of controller nodes. An RCS controller node is a canonical control unit that handles sensing, world modeling and behavior generation. RCS names the levels 1–4 of the architecture diagram SERVO, PRIMITIVE, ELEMENTAL MOVE, and TASK respectively. For ADACS, the stratified architecture is sketched in Figure 2. The nodes form a controller hierarchy with a TASK level root node and SERVO level leaf nodes. Within this architecture sketch, each level has a set of controller nodes denoted by a rectangle which contains its name and timing requirements. Any noteworthy functionality of a controller node is outlined within the attached bubble.

The ADACS hierarchy communicates through message passing. These messages are defined by a language that is not a programming language; it is a type of non–procedural language. The purpose of a command message is to ask a subordinate controller node to do something, leaving the command interpretation and execution to the subordinate. Communication links are defined between nodes with the semantic interfaces briefly described in the text between the controller nodes.

For the ADACS, the TASK level control coordinates the part and finishing information with a the graphical user-interface and CAD model. The ELEMENTAL MOVE (EMOVE) level interprets the relationship between the part edges and the finishing operation. The PRIMITIVE (PRIM) level calculates trajectory path planning with real–time synchronization of motion control with tooling forces. The SERVO level communicates with each actuator and sensor to move the robot and account for the tool compensation. The following sections will further outline the levels of functionality and interfaces.

### *3.1 TASK - Graphical User Interface*

The TASK level develops the part and finishing modeling. For ADACS, the most important aspect of this is programming the part's edges that need to be finished. A graphical user interface to the TASK node allows a user to select edges to be chamfered. This eliminates the need to teach program the robot.

Chamfer features consist of a series of part edges. For an edge that is a line, this information includes the starting point of the edge, the ending point of the edge, and the two normals of the surfaces that construct the edge. These normals are required to determine where the material is, which is required to obtain the proper tool orientation to produce a 45 degree chamfer on the edge. The extracted information for an edge that is an arc is similar. The data contains the starting point of the arc, the ending point of the arc, the center of the arc, the length of the arc, the normal to the plane that the arc lies in and the normal at the starting point and the normal at the ending point of the arc. This data allows the control system to keep a 45 degree tool orientation to the edge as it traverses along the arc.

The inputs to the TASK controller node are finishing edges and series of finishing parameters – spindle speed, finishing force, and finishing depth. These could automatically be derived but it was felt that the operator should control these parameters to tailor the finish as necessary.

The required functionality within TASK includes:

- utilizing CAD model with a graphical user interface
- using PDES or other CAD model data exchange format
- extracting edge information from CAD model
- calibrating CAD data to system fixturing
- using Menu–based or Dialogue–based query system for the user-interface
- generating edge finishing orientations from feature knowledge

### *3.2 ELEMENTAL MOVE- Coordinated Motion*

The ELEMENTAL MOVE (EMOVE) level is the interface between the geometrical part description and the tooling action. In general, a controller node at the EMOVE level coordinates machine robot motion with tool cutting. This coordinated effort results in tool-path motion poses to the robot and auxiliary commands to the related control devices – in this case spindle speeds and tool cutting forces. EMOVE must also check motion pathways for obstacle avoidance – in this case clearance of the tool cutter with the fixture and other part features.

The inputs to the EMOVE controller node are a chamfer normal and series of labelled edges– line, arc, parabola, etc. The required functionality within the EMOVE level includes:

- extracting features from a series of edges
- performing gross–motion path-planning based on part features, including obstacle avoidance of fixtures and nearby features
- using a finishing model to define the tool finishing parameters
- orchestrating tool changes, estimating tool wear
- cuing the operator for part-fixturing
- sending the primitive level a series of trajectory paths with force directives and spindle speeds

The finishing parameters (depth of chamfer, feed rate, and spindle speed), entered from the graphical user interface, are evaluated in the process model and the required normal cutting force is calculated for the operator-designated chamfer depth. This information, along with tool position and orientation commands, is updated and sent to the tool every 2 milliseconds.

### *3.3 PRIMITIVE - Motion Trajectory Generation*

The PRIMITIVE (PRIM) level generates a time sequence of closely–spaced manipulator goal states from a static description of a desired motion. The output commands of the EMOVE level are time–independent descriptions of motions, for example static position, or position and force paths, or directional fields. The position and/or force commands are in the form of parameterized paths to be followed.

The inputs to the PRIM controller node are a motion type– e.g., arc, line, engage_arc, engage_line – and a set of defining parameters including tooling forces and the spindle commands. The required functionality within the PRIMITIVE level includes:

- interpreting the motion, tool and spindle elements
- generating a tightly–coupled motion plan that determines the proper motion position primitive that accounts for synchronizing the spindle and tooling forces
- defining time–based motion primitives to handle non-linear motions or sensor–based motions
- generating motion trajectories

### *3.4 SERVO - Tooling Compensation*

The SERVO level accepts the PRIM outputs and drives the actuators in each piece of hardware to either move to the desired cartesian position or apply the desired force. To remove material from a part manufactured from a hard material, a hard cutter must be used. Hard cutters require compliant tool holders, either passive or active, to reduce chatter and to account for robot inaccuracies in the planned trajectory. Robot arms, unlike structurally stiff machine tools, have a relatively low stiffness that allows large amplitude resonances that cause chatter. It is shown that in Assada and Goldfine [2] that chatter is reduced when the tangential and normal stiffnesses differ by a factor of 10, as shown in Figure 3.

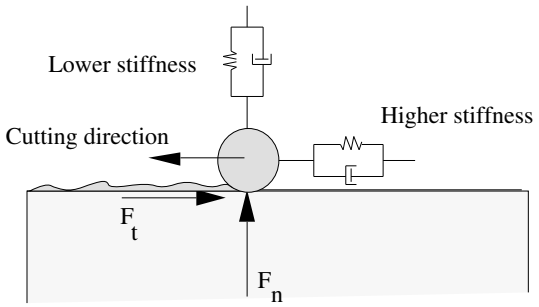


FIGURE 3. Tool Chamfering Stiffness

When following an edge, robot accuracy is not sufficient to keep a hard cutter on the edge. Therefore, the normal direction of the tool must be made to be compliant, so that the cutter will remain in contact with the edge and apply the necessary normal force to achieve the required break edge. Compliance can either be implemented passively (with spring and damper system) or actively through force control.

## 4. ADACS DEMONSTRATION

In this section, we provide a short overview of the ADACS prototype, and how the ADACS system is used to finish parts within the AMRF. Chamfering in the ADACS demonstration workstation is accomplished through the coordination of two robots and a servo positioning table.

These machines are controlled using the NIST developed Real–time Control System (RCS). For the initial scenario, two parts were selected to validate the ADACS. These parts include a jet aircraft turbine hub and a bearing housing. The deburring and chamfering of these parts is a very manually intensive endeavor and is therefore very costly. Both parts also need heavy material removal and are manufactured from hard materials which dictate the use of hard tooling.

The turbine hub, shown in Figure 4, is a highly stressed rotating engine component with many blade retention slots that are broached into the body of the hub. Each of these slots needs to be deburred and chamfered. The small dimensions (5mm across at the back edge) of the retention slots have tight tolerances as well as tightly radiused features. The back edges also have a break edge requirement of 0.25 – 0.75 mm, blended at each end, to reduce stress concentrations in the hub.

The bearing housing, shown in Figure 5, is a non–rotating part that has heavy material removal requirements in the area of the lightening holes. These lightening holes are an interesting challenge in that they are milled in a conical shape.

FIGURE 4. Turbine hub

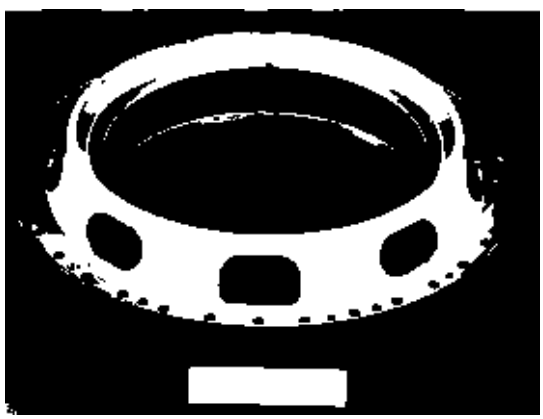

FIGURE 5. Bearing housing

The following section will discuss the equipment, briefly review the major software components, and describe the ADACS graphical user interface.

### *4.1 Equipment*

For the prototype ADACS, two separate robots are coordinated to perform the chamfering operation on the parts in the workcell. A T3–646 six axis electric robot is used as a macropositioner and an Adaptive Deburring Tool (ADT) is used as a micropositioner. A servo table was used to fixture the part to be chamfered.

The T3–646 is a six degree of freedom electric robot with a three roll wrist. The T3-646 is shown below in Figure 6. A battery of tests were performed on the T3–646 to determine induced resonant frequencies and robot–induced dynamic path errors. These tests were performed to determine the tool performance requirements necessary to filter out these disturbances. A mock tool designed to simulate the mass distribution of the ADACS chamfering tool was mounted to the flange of the robot. An accelerometer was mounted to either of the five arms of the mock tool and was used in conjunction with an impact hammer to determine the robot dynamic characteristics. The accelerometer was mounted on the mock tool and the mock tool was struck in several directions and the hammer force and robot acceleration times were recorded using a data analyzer. These tests were performed to obtain the performance requirements for a new active tool being designed by the engineers at UTRC. This Chamfering and Deburring End-of-arm Tool (CADET), Figure 7, is a voice coil actuator active tool. The test procedures and results are discussed in

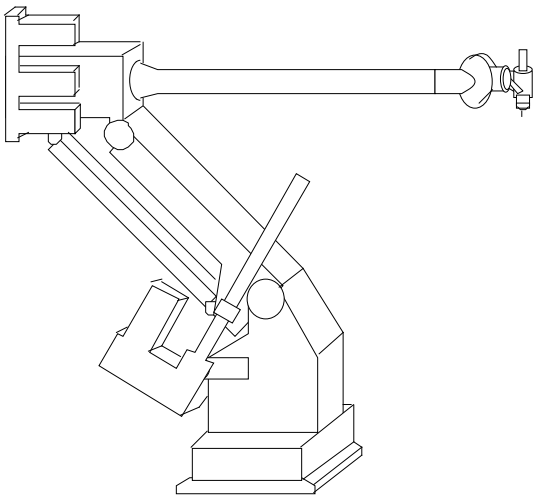

FIGURE 6. T3-646 Robot

detail in [4]. The CADET will be integrated into the ADACS in the fall of 1993.

The current tool used on the ADACS is the Adaptive Deburring Tool (ADT) developed by Trikinetics, shown in Figure 8. The control of the ADT is described in [7].

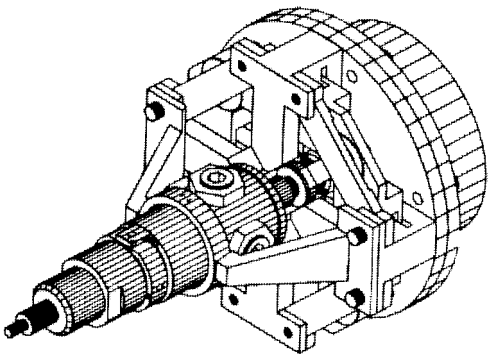

FIGURE 7. Chamfering and Deburring End-of-arm Tool (CADET)

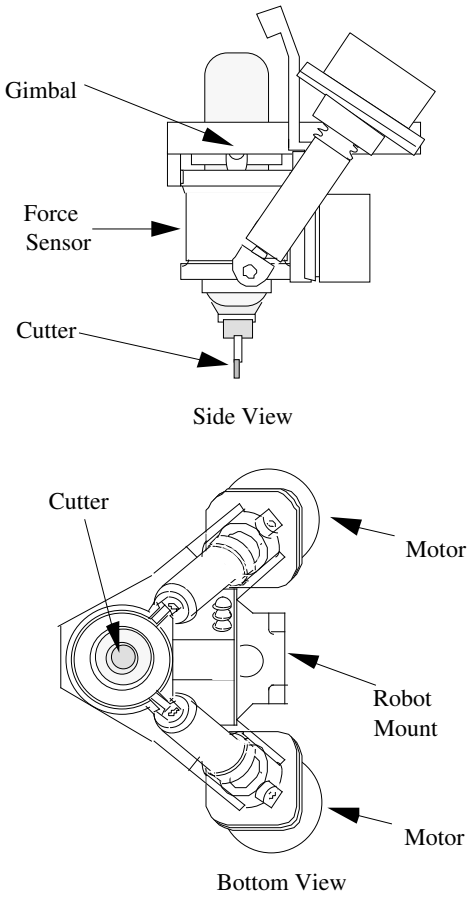


FIGURE 8. Active Deburring Tool (ADT)

### *4.2 RCS Software Tools*

For ADACS, RCS consists of an archive containing platform–generic and platform–specific source code, header files, executables, libraries and documentation for control applications. The archive consists of C and C++ code that supplies application-independent libraries– such as communication, tasking, vector math, etc. – as well as application-specific routines, such as device kinematics, I/O drivers, etc. The application–independent services are designed to be platform–independent so that code is transparently portable across platforms. A set of RCS shell commands are provided within the archive as a programming convenience. These commands automate much of the tedious programming chores and provide a consistent programming paradigm.

The primary programming construct is the control node which is a self–contained control engine. On its own, a control node has the baseline ability to send and receive command and status as well as communicate to a user. At run-time, a control node self–configures to determine its supervisor and its subordinates. This self–configuration hides much of the communication setup from the user. Users then extend the control node by adding commands to a command list. Each command has an accompanying execution routine and source file. These commands then become part of the control node interface. A variety of command types, interfaces and execution modes (task spawns, execution call, or deterministic cycle) are available.

### *4.3 Level II Library*

The ADACS workstation uses the Level II Library (supported by the Indiana Business Modernization and Technology Office) – an ANSI C library of robot control routines for trajectory generation within the PRIMITIVE level. These routines provide a user hooks to control one or many kinematic devices simultaneously. The user is responsible for supplying the kinematics, configuration and dynamic profile when installing the device in the library. Level II then provides routines to generate time–based trajectory set points for a device based on these kinematic routines [6]. The Level II Library uses position equations consisting of homogeneous transformations to represent motion, e.g.,

[BASE][T3][TOOL] = [TABLE][FIXTURE][PART]

Level II uses homogeneous transforms to describe the coordinate transformations, but supports other representations. This was necessary to translate between homogeneous representation and the T3–646 Euler angle representation. More importantly, Level II allows

functionally–defined transforms to modify a position equation in real–time. Thus, while the trajectory generator is evaluating a position equation, a functionally–defined transform can provide for time–dependent transformations such as curvilinear motions (arc, circles, ellipses, etc.) or sensor–dependent offset values.

For motion control, one defines a series of moves using position equations that are queued. Level II executes cyclically with a fixed frequency to sequentially process each position equation on the queue and compute the kinematic parameters– either joint or cartesian– necessary to move the robot to the desired position. Several different parameters can be set to control the attributes of the motion as the robot follows the desired path. These include acceleration, velocity, total time, fly–through vs. stop, and mode of travel (cartesian vs. joint).

The ability to use functionally–defined transforms allows ADACS to define non-linear edge–based motion, e.g., arc and ellipses, as well as straight lines. The ability to apply functionally defined transforms with the position equation allows tight control of the tooling force synchronized with the current position.

### *4.4 World Model*

The main components of the world model for ADACS consists of device kinematics, feature knowledge, chamfer knowledge, and tooling force compensation. Of most interest to this paper is the software development of the feature knowledge. ADACS uses the base class concept of C++ to define a chamfer edge object. A chamfer edge includes the typical data definitions of a starting, entry, and ending position and orientation. However, the chamfer edge applies the C++ virtual function to include functions to derive direction, concavity, orientation and input and output format interface. Depending on the feature, the default definitions of the virtual function might be overridden. By defining a free–space and none–left edge objects, determining features during edge state transition was greatly simplified.

The process model for chamfering inconel, empirically developed by the engineers at UTRC, is shown in Figure 9.

### *4.5 User-Interface*

The graphical user–interface is based on the AutoCAD system. Using AutoLISP and the Advanced Modeling Extension (AME) within AutoCAD Release 12, a LISP

$$F_n = C_1(1 + C_2\,MR)D_c^{K_3}F_r^{K_4}N_s^{K_5}$$

$F_n$ = normal cutting force (lb)
MR = material removed (mil2-in)
$D_c$ = chamfer depth (mil)
$F_r$ = feed rate (in/min)
$N_s$ = spindle speed (krpm)

$C_1 = 0.0084$
$C_2 = 0.0012$
$K_3 = 1.88$
$K_4 = 0.57$
$K_5 = 0.40$

FIGURE 9. Chamfering Process Model

program was written to extract the necessary edge information from a part to calculate the finishing features. With the interactive user interface, the user highlights the edge, and the exact cartesian feature description is automatically calculated from the CAD data base. After the user selects an edge to be finished, it is highlighted and a directional cone is then superimposed on the edge to indicate the direction of the trajectory to follow. A prompt then asks if this is the required direction of travel. If the user answers yes, the directional cone remains on the edge and the user is promoted to select another edge. If the user answers no, the previous directional cone is erased and the correct cone that indicates direction is superimposed on the edge and then user is prompted to select another edge. When the user is finished, the result is a drawing of the part with the robotic trajectories superimposed on the edges that need to be finished. This allows the user to easily visualize the paths that the robot will follow.

The user also specifies several machining parameters for the finishing operation. These include the spindle speed, the required chamfer depth for the edges and the required feed rate. The necessary force to obtain the chamfer depth is then calculated, based on these parameters, and is downloaded to the active tool, which updates the force it is applying to the part's edge every one–thousandth of a second.

A file is generated which contains a series of part edges and chamfering information. This information is passed to the EMOVE levels which interprets the edges, generates the chamfer features. Given a chamfer feature, EMOVE develops a command using

the chamfering process model based on the chamfer depth. The command is interpreted by PRIM to generate real-time motion and force control to achieve the proper material cutting. PRIM sends the next position, spindle velocity and force commands to the SERVO level which interprets these for the given piece of equipment.

## 5. SUMMARY

ADACS supplies a CAD–based graphical interface of a part, wherein the operator uses a mouse to select feature edges to chamfer and optionally supplies chamfer forces and cutting strategies. ADACS subsequently generates the finishing process model and performs the finishing operation that includes the following key features:

- graphically–instructed edge definition and automated edge extraction from CAD model
- feature recognition based on edge–to–edge transitions including free–space to contact, continuous versus discontinuous, and force information.
- feature–based knowledge to define material removal. Feature–based knowledge defines a process model based on part material, cutting force, depth of chamfer, feed rate and spindle speed. Tool wear estimation is also monitored.
- feature based generation of multiple finishing paths to prevent dead-reckoning problems, and scarring.
- tightly motion and tool forces control that allows linear and curvilinear (arc, ellipses, etc.). Ramping of tool forces and spindle speeds allows smooth transition from free–space to contact–space.
- active force control tooling to compensate for positioning errors.

Future improvements to the ADACS include 1) the addition of extracting and interpreting part and fixture information to prevent collisions and fixturing interference, 2) adding new parts and chamfering features to the feature–based knowledge, and 3) experimenting with a stiffer motion control device–such as a machine tool instead of the robot.

From our experiences, the ADACS system has proven a flexible and useful system. It has applicability beyond part chamfering and deburring and in the future will be adapted to perform feature-based grinding and welding, as well as other edge–related feature applications.

Unfortunately, the wrist kinematics of the robot pose a problem. Although the three wrist axes intersect at a point, guaranteeing a closed-form solution for the inverse kinematics, this point is located mechanically inside the wrist. The tool must be mounted so that the point of contact between the cutter and the edge is relatively far from this intersection point. This means that pure orientation changes, which are common during a dextrous task such as chamfering, require motion of all six joints. The effect is that inaccuracies in the base joints are magnified by the large distance between the base and tool.

A preferred wrist design would place the point of intersection of the three wrist axes in space, so that the cutting point could be located at this point. If this were possible, pure rotation would only require motion of the three wrist axes, eliminating the dynamics of the base joints.

There are research projects currently ongoing at NIST and UTRC to determine if a machine tool would provide an improved platform for the ADACS. After the machining of the part is completed, the machine tool would change tools to the deburring/chamfering tool and the finishing would take place on the same machine. There would be no need to remove the part from the machine and perform the finishing operations elsewhere in the manufacturing facility, which can add hours to the manufacturing time.

## 6. REFERENCES


1. Albus, J. S., “RCS: A Reference Model Architecture for Intelligent Control,” IEEE Journal on Computer Architectures for Intelligent Machines, May 1992.

2. Asada H., Slotine, J. J., Robot Analysis and Control, John Wiley and Sons, New York, New York, 1986.

3. Craig, J. J., Introduction to Robotics Mechanics & Control, Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

4. Engel, T. W., Roberts, R. K., Proctor, F. M., “Specification of an Active Force Control Tool for Performing Deburring and Chamfering on a Robot Platform,” Proceedings of the International Conference on Industrial Electronics, Control, Instrumentation, and Automation, San Diego, California, November 9-13, 1992.

5. Gillespie, L. K., Robotic Deburring Handbook, Published by Society of Manufacturing Engineers, Robotics International of SME, Dearborn, Michigan, 1987.

6. Guptill, R., Stahura, P., “Multiple Robotic Devices: Position Specification and Coordination,” Proceedings of the IEEE International Conference on Robotics and Automation, Raleigh, North Carolina, March 31 - April

3, 1987.

7. Hollowell, R., “Hybrid Force/Position Control for Robotic Light Machining,” Robotics and Remote Systems Conference, Charleston, South Carolina, March 1989.

8. Murphy, K. N., Proctor, F. M., “An Advanced Deburring and Chamfering System,” Presented at Third International Symposium on Robotics and Manufacturing, British Columbia, Canada, July 1990.