Path Generation for Planar Spray Painting

by

C.James Bartlett
B.Sc.(M.E.), University of Manitoba, 1985

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the
Department of Mechanical Engineering

We accept this thesis as conforming
to the required standard.

[redacted]
Dr. R. P. Podhorodeski, Supervisor (Dept. of Mechanical Engineering)

[redacted]
Dr. Z. Dong, Department Member (Dept. of Mechanical Engineering)

[redacted]
Dr. E. Manning, Outside Member (Dept. of Electrical and Computer Engineering)

[redacted]
Mr. B. Pooni, External Examiner (Civil & Mech. Eng. Technology, Camosun College)

University of Victoria

Supervisor: Dr. Ron P. Podhorodeski

Abstract

Robotic spray paint application offers cost, environmental and safety advantages over manual spray paint application. The benefits derived from robotic paint application depend in part on the path the robot is programmed to follow. Robots can be programmed manually or automatically. Manual path programming is limited by the operators ability to visually judge what the optimum path might be. Manual programming also requires the production line be stopped while the path is programmed.

An algorithm is developed here for the generation of paint paths based upon a geometric model of the workpiece. The algorithms are applicable to planar workpieces defined by contiguous line segments.

The spray pattern is modelled as an ellipse with uniform paint distribution. The generated path is comprised of a set of parallel horizontal spray passes, starting from the upper edge of the workpiece and descending to the lower edge in equal increments. The endpoints of each individual pass are determined by the intersection of the pass center with a curve offset from the workpiece boundary.

A process cost model is developed to quantify the economic cost of spraying an individual workpiece. The model includes the cost of the paint itself, operating and maintenance costs of the painting system, and costs for filtration and disposal of waste paint.

An algorithm is presented for minimizing the process cost model. The algorithm illustrates that minimum process cost is obtained with the highest possible spray gun

velocity, subject to the constraints of maximum paint flowrate and required paint thickness, and the spray pattern adjusted for minimum aspect ratio.

For general workpiece geometry an exhaustive search over the feasible range of spray pattern size and pass orientation is required. For the special case of rectangular workpiece geometry the minimum cost occurs for spray passes oriented either parallel or perpendicular to the base of the rectangle. For rectangular workpieces there also exists a set of spray pattern sizes that result in local minima of the process cost function. The global minimum can be found through comparison of the local minima.

For a given deposition profile the uniformity of paint thickness over consecutive spray passes is determined by the overlap between them. Three measures for paint thickness uniformity are presented. For the elliptical spray pattern used here the optimum uniformity occurs between 16-18 percent overlap, depending which measure of uniformity is used.

The algorithms presented here have been written in the 'C' programming language and can be executed within the AutoCAD environment

Examiners:

[redacted]

Dr. R. P. Podhorodeski, Supervisor (Dept. of Mechanical Engineering)

[redacted]

Dr. Z. Dong, Department Member (Dept. of Mechanical Engineering)

[redacted]

Dr. E. Manning, Outside Member (Dept. of Electrical and Computer Engineering)

[redacted]

Mr. B. Pooni, External Examiner (Civil & Mech. Eng. Technology, Camosun College)

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Dr. Ronald Podhorodeski, for his support, guidance and patience during this project. Thanks also to Kenneth Pittens, who has endured and answered my countless questions.

To those who make me smile.

Chapter 1

Introduction

1.1 Purpose and Definition of Paints

The application of paint is a fundamental step in many manufacturing operations. The purpose of the paint may be to make the workpiece aesthetically pleasing or to protect it from a corrosive environment. Principles and practices of industrial painting are discussed in general in [19].

The traditional description of a paint is a liquid containing a binder, solvent, additives and possibly a pigment. The binder holds the paint components in a continuous system and provides the basis of the film formed on the workpiece after curing. Solvent is added to reduce paint viscosity for easier application. The solvent evaporates after application. Additive is a general term describing a wide variety of materials added to paint to improve its functional performance. Pigments are insoluble solid particles added to paint for either appearance (color) or function (corrosion protection).

A variety of other coating materials currently in use do not fit the traditional description of a paint, but do perform the same function. One example is powder coatings, which are applied to the workpiece as a dry powder and subsequently heated

to their melting temperature. Upon melting the powder flows out into a continuous film. The dry powder may contain all the components of a traditional paint with the exception of the solvent. Another example is electrocoating, where the workpiece is submersed into a conductive solution of water and paint. Upon application of a potential the paint is deposited on the workpiece.

For the purposes of this work the terms *paint* and *coating* will be used interchangeably to refer to any liquid that must be applied with uniform surface thickness to a workpiece.

1.2 Spray Coating

A variety of methods can be used to apply coatings, however the application method considered in this work is air atomizing spray. With this method of application the liquid paint is emitted from a nozzle into a high velocity stream of air. The paint stream is subsequently broken into a fine mist of particles that distribute over the surface of the work. The fine distribution of paint throughout the spray pattern results in a uniform surface finish, which is the primary attribute of spray painting.

The device used to meter the paint into the air stream is known as a *spray gun.* The spray gun is connected to a paint supply and a source of compressed air. Control valves on the spray gun allow variation in the shape of the spray pattern and quantity of paint emitted.

Industrial spray painting can be performed manually, with fixed automation, or with programmable automation.

1.2.1 Concerns Facing the Spray Coating Industry

Industries using air atomizing spray finishing in their manufacturing process are facing several concerns. This section will discuss these concerns.

Environmental and Health Concerns

The viscosity of paints must typically be lowered before they can be effectively atomized by the spray gun. The solvents used to lower paint viscosity are collectively known as volatile organic compounds, or VOCs.

VOCs contribute to the formation of ground level ozone, and ultimately, urban smog. It has been reported [6] that in the summer months more than one half of all Canadians are exposed to ozone concentrations that are recognized as having adverse effects on health. Agriculture is also affected. For example, crop damage from ozone in Ontario, Canada, is reported to reach $70 million annually. By the year 2005 solvents are expected to become the largest source of VOC emissions, and the most common use of solvents are those in paints and coatings [6].

In the United States both federal and state governments are overseeing the implementation of the 1990 Clean Air Act. The Environmental Protection Agency (EPA) has set maximum VOC emission levels which must be met by all states. Each state has an equivelent organization which may impose restrictions more stringent that those of the EPA. The most stringent of state regulations are those administered by the South California Air Quality Management District. Their standards are expected to form the basis of many other state regulations.

In Canada the Canadian Council of Ministers of the Environment responded to the problem of ground level ozone in 1988 and began development of a NO_x and

VOC management plan. As a result, new VOC emission standards for industrial and commercial painting facilities are to take effect in 1994. The nature of the new standards are currently under development [6].

Health Concerns

Many solvents, binders and catalysts used in coating materials present a direct health hazard to anyone inhaling their vapours. Both United States and Canadian governments have established occupational safety standards which limit permissible exposure to a wide class of coating materials. In addition, employer's premiums to workmen's compensation plans are increased when personnel are exposed to painting environments.

Cost of Coatings

Spray painting is noted for its low application efficiency. Only a portion of the paint sprayed is deposited on the workpiece surface. This, combined with the cost of paint materials, results in a high cost due to waste. Furthermore, the portion of paint not deposited on the work may often have to be filtered from the air, and disposed of, which adds to the overall cost.

Quality of Surface Finish

Political changes which reduce or eliminate trade barriers are exposing manufacturers to increased levels of foreign competition. All aspects of product quality, including surface finish, are becoming more critical as organizations strive for a competitive advantage.

1.2.2 Addressing the Concerns

Manufacturers and paint suppliers are addressing the aforementioned industry concerns in several ways. Paint suppliers are altering paint chemistry and introducing coatings with reduced VOC levels, such as waterborne paints and high solids coatings. Spray gun manufacturers are developing guns which are able to atomize the paint with lower pressures, thus reducing overspray. Finally, manufacturers are employing robotic automation systems to apply the paint.

1.3 Robotic Spray Coating

Robotic spray coating is a process where a robot system guides a spray gun over the workpiece surface in a specified pattern. In this work, the term robotic system is meant to include any programmable system which can guide a spray gun through a given path.

Robotic paint application offers the potential for eliminating or reducing many of the concerns manufacturers face in their finishing operations. Robotic application can result in reduced paint consumption over manual application. Using less paint lowers the direct paint cost and reduces the overall VOC emissions. Robotic paint application is also consistent. Once an acceptable trajectory is defined, the surface quality remains at the same level. Finally, with robotic application, human exposure to toxic paint vapours can be eliminated.

General discussions and experiences with robotic spray coating are provided in [15, 14, 11, 21, 8, 5, 4]. More specific consideration of robotic spray painting hardware is found in [20, 8]

1.3.1 Robot Path Planning

The extent to which the benefits of robotic spray coating are realized depends upon the path the robot is instructed to follow. The process of designing this path is termed *path planning.*

In practice path planning for robotic spray application is typically performed by an experienced operator who leads the robot through the desired path. This teaching method is termed *on-line path planning.* The results of on-line planning are largely dependent on the operator's judgement. An alternate technique is to plan the robot path based on a geometric model of the workpiece. This teaching method is termed *off-line path planning.*

1.3.2 Off-Line Path Planning

Off-line path planning is performed remotely from the spraying equipment. It involves creating a set of motion instructions based on a geometric model of the workpiece and spray gun, and a set of mathematical relationships between the spraying parameters. Discussion and experiences with off-line path planning are given in [18, 13, 5, 4, 1]. Off-line path planning for electrostatic spray paint application is considered by Duelen [9].

Off-line path planning offers the potential of extending or enhancing the benefits of robotic spray finishing. The attributes are:

1. The time and cost of programming the robot are reduced.

2. The resulting path accuracy is restricted only by the validity of the model used to generate the path.

3. Programming the robot off-line does not require the use of the actual production equipment. Therefore, the need to interrupt the production line is eliminated.

Off-line path planning also permits consideration of the optimization of the spray path with respect to process time and process cost. This optimal off-line path planning of spray paths is the goal of this work.

In this thesis, techniques for off-line path planning for planar workpieces are presented. A process cost model is developed which allows quantifying the economic cost of spraying a given workpiece. An algorithm for minimizing the process cost is also presented. The algorithm considers spray pattern size and spray pass orientation.

1.3.3 Current Research in Off-Line Path Planning

Techniques have been developed for off-line path planning for robotic spray painting.

Suh et al. [23, 24] have developed a trajectory planning system for painting robots. The path is first planned based on a CAD model of the workpiece, which may be a three dimensional sculpted surface. The spray pattern is assumed circular and the path interval is taken to equal to the radius of the spray pattern. The sweeping direction is assumed to lie along one of the coordinate directions defining the surface. In particular, the one with the least curvature. The velocity of the spray tool is determined such that the average coating thickness equals the required nominal. In [23], techniques have also been developed for the analysis of the subsequent coating thickness. The main assumptions may be summarized as:

- sweeping direction chosen as being parallel to one of the two coordinate directions defining the surface.

- planned path is based upon a circular spray pattern with uniform coating distribution.
- path interval is taken as the radius of the circular spray pattern.

Klein [16, 17] has developed an interactive path planning algorithm that models the spray pattern as a cone. The thickness of paint is assumed constant through a center portion and to fall off sinusoidally to zero at the edges. An equation describing the spray density at any point within the cone is used to determine the total paint deposited at any point on the workpiece surface.

Goodman [12, 13] has proposed a method for quantifying an arbitrary spray distribution based on thickness measurements of a standard test spray path. The results of the test are then used as an input to a spray path analysis program to provide a detailed coating thickness analysis of a chosen path.

1.4 Objectives

The objectives of this work are:

1. to investigate the painting process and develop an algorithm to generate spray painting paths.
2. to form an economic cost model of the painting process.
3. to develop an algorithm which selects painting parameters for minimum process cost, while maintaining surface finish quality.

4. to implement the results of the work within the AutoCAD Computer Aided Design software system [1].

[1] AutoCAD is the world's most predominant CAD system with over 470,000 users [2]. Furthermore, there are currently no automatic spray path generation programs that run within the AutoCAD environment [3].

Chapter 2

Quantifying the Painting Process

2.1 Introduction

Prior to generating paths for spray painting, several characteristics must be modelled in a fashion suitable for analysis. The purpose of this chapter is to provide a brief discussion of the painting process and to introduce and quantify the characteristics.

When spray painting the spray gun is held at a fixed distance or *stand off distance* away from the part being sprayed. As the gun is drawn across the workpiece its *orientation* is generally maintained normal to the surface being painted. The characteristic shape the paint makes as it contacts the work surface is termed the *spray pattern.*

The thickness of paint to be deposited on the work is generally known at the outset of the task. The amount actually deposited depends on several factors which include the *paint flow rate*, the *gun velocity* and the spray pattern size. To help insure the desired amount is applied the distribution of paint over the spray pattern must be quantified. This is achieved with the *deposition profile.*

Painting is typically performed by a series of parallel passes over the part. To ensure complete paint coverage each pass overlaps the previous to some extent. The

distance between consecutive spray pass centers is termed the *path interval.* The path interval and spray pattern size determine the *overlap* between passes.

Due to local and bulk movement of air, not all of the finely atomized paint sprayed from the gun is available for deposit on the work. The amount actually available is described by the *transfer efficiency.*

The painting parameters mentioned above are fundamental to the painting process. This chapter will describe each in preparation for development of the spray path generation problem.

2.2 Paint Flow Rate and Transfer Efficiency

Of major importance in painting is the rate of paint flow from the spray gun. This variable, termed the *nominal flowrate* and denoted q', is a measure of the total volume of paint emitted from the paint gun per unit time. One consequence of the high degree of atomization achieved with spray painting is that only a portion of the total paint flow is deposited upon the workpiece surface. A portion of the spray is carried away by air currents from both the spray gun and air filtration system. The portion carried away is termed *overspray.* The variable used to account for the loss due to overspray is the *transfer efficiency*, η. The transfer efficiency describes the portion of the nominal flowrate that is available for deposit on the workpiece surface. The paint flow available for deposit on the work is termed the *net flowrate*, q, and can be found from

$$q = \eta q' \tag{2.1}$$

In practice the transfer efficiency varies widely depending on the type of painting equipment employed and the type of paint being used.

2.3 Gun Velocity

Another factor in determining the quantity of paint deposited on the workpiece surface is the gun velocity, v. The gun velocity is the velocity with which the spray gun moves with respect to the workpiece surface. Since planar spraying is considered in this work the velocity is always in a direction parallel to the plane of the workpiece as shown in Figure 2.1.

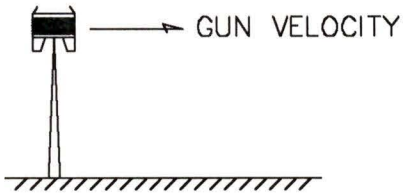

Figure 2.1: Gun velocity parallel to workpiece

In the course of this work there will be two velocities of concern: one where the paint is on and; one where the paint gun is off. The paint on and paint off velocities are denoted v_i and v_o, respectively.

2.4 Gun Orientation

The spray gun orientation is the angular orientation of the gun with respect to the workpiece surface. A generally accepted practice in spray painting is to maintain the gun alignment along a vector normal to the workpiece. Again, since planar painting is being considered the gun will maintain an orientation perpendicular to the plane of the work, as illustrated below in Figure 2.2.

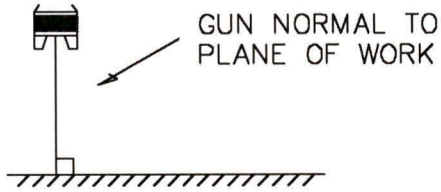

Figure 2.2: Gun normal to workpiece

2.5 Stand Off Distance

The standoff distance, d, of the spray gun is the distance from the workpiece to the spray tip as illustrated in Figure 2.3. The standoff distance affects the dispersion

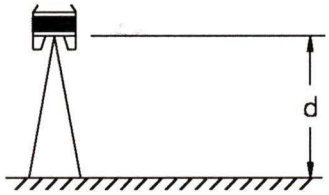

Figure 2.3: Stand off distance

of the paint spray. Increasing d tends to result in a greater amount of spray being carried away from the work. This is reflected in a reduced transfer efficiency, η. A larger stand off also results in a larger spray pattern as shown in Figure 2.4. For a conical shaped spray fan making an angle α with the vertical the diameter of the spray pattern, D, is given by

$$D = 2\ d\ \tan\alpha \tag{2.2}$$

where α is the angle between the conical pattern and workpiece normal. Therefore D is proportional to d. In practice an approximate stand off distance of 15 to 20 cm is recommended [19]. This distance is to remain constant during the painting process.

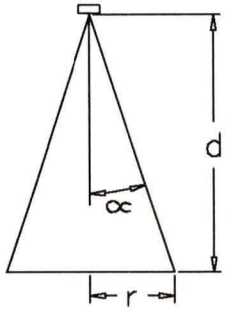

Figure 2.4: Conical fan shape

2.6 Spray Pattern Shape

The *spray pattern shape* is defined as the shape of the paint pattern that a stationary spray gun would make on a workpiece surface directly below it at a given stand off distance. In general, this pattern is elliptical in shape. The elliptical shape is physically accomplished by impinging streams of air into the conical paint flow. The streams of air tend to compress the cone from a circular cross section into an elliptical cross section. The streams of air are directed from a set of air horns on side of the spray nozzle. A cross section of a typical spray nozzle showing the air horns is illustrated in Figure 2.5. The quantity of air passing through the air horns is variable and may

Figure 2.5: Nozzle cross section

typically be set on the paint gun itself. The progression in spray pattern shape as

the fan air is increased is shown in Figure 2.6. In this work the spray pattern will be

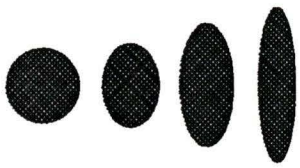

Figure 2.6: Spray pattern progression

considered as an ellipse. For the elliptical spray pattern the ellipse dimensions will be noted as illustrated in Figure 2.7. In figure 2.7 the variable a denotes the ellipse

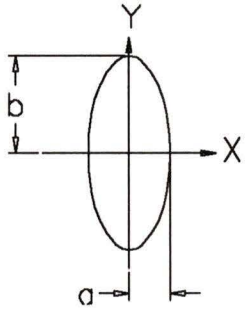

Figure 2.7: Ellipse centered at origin

minor axis size and b represents the major axis dimension.

An important characteristic of the spray pattern is the distribution of paint over its area. Little data exists to substantiate the distribution of paint over the spray pattern [22]. Goodman [12] has proposed a technique to quantify any specific distribution based on a standard test of the hardware configuration.

In this work, the distribution of paint over the spray pattern will be assumed uniform. However, the techniques developed for optimizing paint paths can be used for other distributions.

2.7 The Path Interval

The *path interval* is the distance between the center lines of 2 consecutive spray passes. This is shown as s in Figure 2.8.

2.8 The Overlap

The overlap is defined as the ratio of the overlapped region of consecutive passes to the total height of the spray pattern. The overlap of the two passes is illustrated in Figure 2.8. The absolute value of the overlap is given by $2b - s$, where 2 b is the total

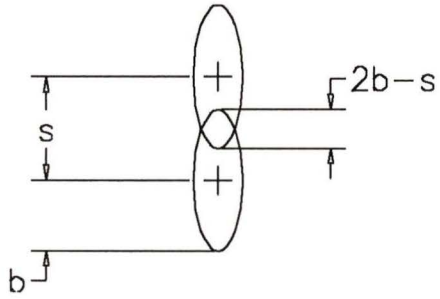

Figure 2.8: Spray pattern overlap

height of the spray pattern and s is the path interval. Therefore, expressed as a ratio of the total spray pattern height the overlap is

$$\beta = \frac{2b - s}{2b} = 1 - \frac{s}{2b} \tag{2.3}$$

Overlap is often expressed as a percentage.

2.9 Development of the Deposition Profiles

The *deposition profile* is defined as the profile of paint thickness deposited in a planar cross section perpendicular to the velocity of the gun. A deposition profile (greatly exaggerated) for a single pass of the sprayhead is illustrated in Figure 2.9. To help

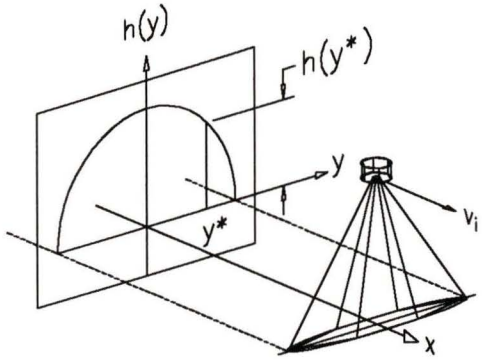

Figure 2.9: Deposition profile

explain the deposition profile consider the manner in which paint is deposited as the gun moves over the surface of the workpiece. If the spray head is given a constant velocity v_i, the resulting spray will be deposited as illustrated in Figure 2.10.

Since the spray pattern is wider in the center than at the ends, the deposition profile will be thickest along the path centerline, (i.e. the x axis in figure 2.10) and will decrease to zero at $y = \pm b$.

To determine the deposition profile for an assumption of uniform distribution over the pattern, it is useful to consider the film build up for a stationary gun. The rate of film build for a stationary spray gun is simply the net flow rate divided by the spray pattern area. If the thickness of paint is denoted h the rate of film build for a

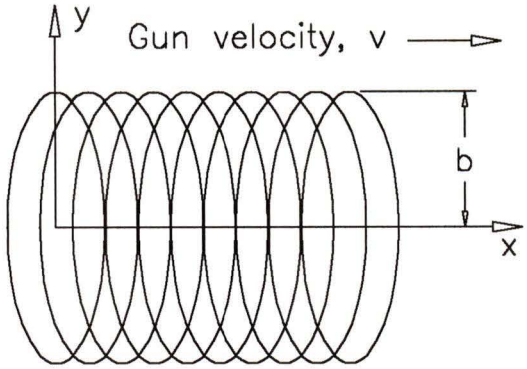

Figure 2.10: Paint deposition

stationary spray gun is given by

$$\dot{h} = \frac{q}{A} \tag{2.4}$$

where A is the ellipse area and is given by

$$A = \pi ab$$

Therefore

$$\dot{h} = \frac{q}{\pi ab} \tag{2.5}$$

where the dimensions are [L]/[t].

Consider the spray pattern illustrated in Figure 2.11. The amount of paint deposited on an arbitrary point y^* is proportional to the duration of time the spray pattern is over that point. This time duration is directly related to the width of the ellipse at that point. Consider the line AB at y^* in Figure 2.11. The length of AB may be determined by rearranging the standard equation for an ellipse and solving for x, i.e.,

$$\begin{aligned} 1 &= \frac{x^2}{a^2} + \frac{y^2}{b^2} \\ \Rightarrow x &= \frac{a}{b}\sqrt{b^2 - y^2} \end{aligned} \tag{2.6}$$

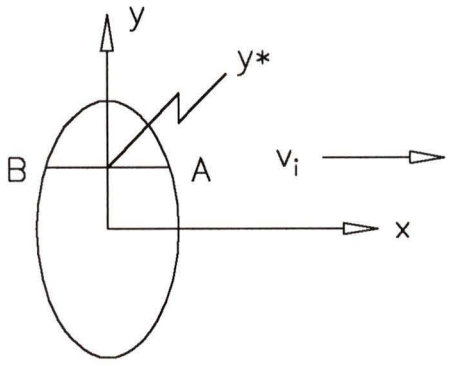

Figure 2.11: Quantifying the deposition profile

Therefore, the x coordinates of the intersection of AB and the ellipse are given by

$$x = \pm\frac{a}{b}\sqrt{b^2 - (y^*)^2} \tag{2.7}$$

and the length $|AB|$ is simply the difference between the x coordinates or

$$|AB| = 2\,\frac{a}{b}\,\sqrt{b^2 - y^2} \tag{2.8}$$

The duration of time the spray pattern is over point y^* is inversely proportional to the gun velocity or

$$t = \frac{|AB|}{v} = \frac{2a}{vb}\,\sqrt{b^2 - (y^*)^2} \tag{2.9}$$

and the amount of paint deposited at y^* is given by

$$h(y^*) = \frac{2\,q}{v\,\pi\,b^2}\,\sqrt{b^2 - (y^*)^2} \tag{2.10}$$

Or in general for any point along the y axis

$$h(y) = \frac{2q}{v\pi b^2}\,\sqrt{b^2 - y^2} \quad , -b \leq y \leq b \tag{2.11}$$

Equation (2.11) represents the thickness profile of paint deposited in any plane perpendicular to the velocity of the spray gun. It is interesting to note that the minor

axis length of the ellipse, a, is not present in equation (2.11). This implies that the aspect ratio of the ellipse is irrelevant to the shape of the deposition profile. Any elliptical spray pattern with overall height 2 b would result in the same deposition profile if the paint flow and velocity were maintained at a constant rate. A typical deposition profile is shown in Figure 2.12.

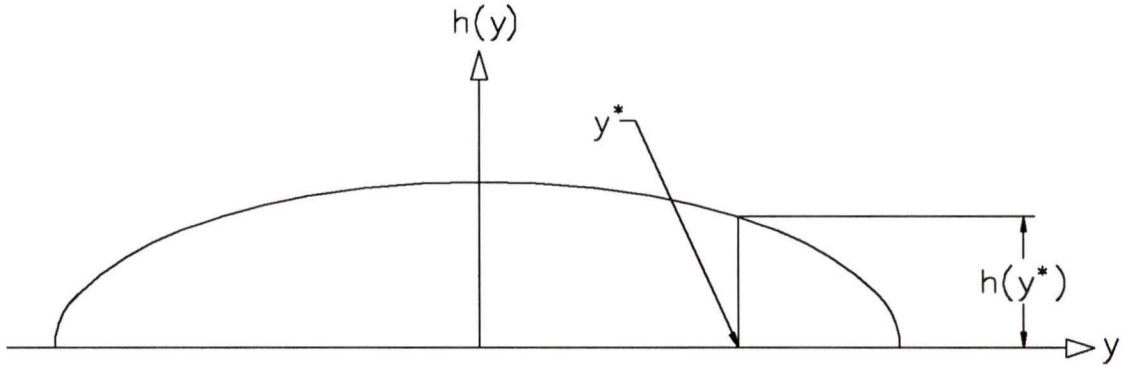

Figure 2.12: Deposition profile

2.10 Deposition Profile Across the Path Interval

The deposition profile of two overlapping and identical profiles is illustrated in Figure 2.13. Note that in Figure 2.13 the coordinate axis y runs across the path interval. The resulting paint thickness over the range $0 < y < s$ is a composite function given by:

$$h = \begin{cases} h(y) & \text{if } 0 \leq y < (s-b) \\ h(y) + h(s-y) & \text{if } (s-b) \leq y \leq b \\ h(s-y) & \text{if } b < y \leq s \end{cases}$$

where $b \leq s \leq 2b$ and $h(y)$ is as given in equation (2.11).

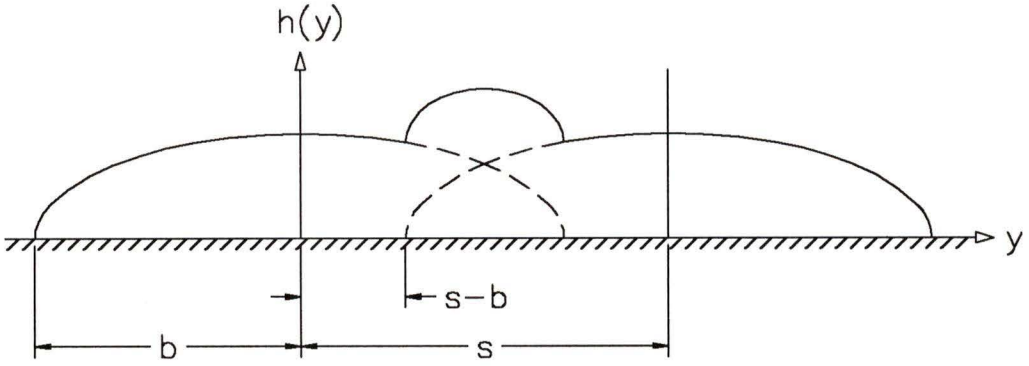

Figure 2.13: Overlapping deposition profiles

2.11 Determination of Average Film Build Across Path Interval

An expression for average film build may be obtained by integrating the equation for the deposition profile over the range of the path interval and dividing by the length of the path interval.

Observation of Figure 2.13 illustrates that the total amount of paint deposited over the path interval is the sum of 2 half deposition profiles, which is the same as one complete deposition profile. Therefore, the average film thickness may be obtained by integrating the deposition profile (equation (2.11)) over the range

$$-b \leq y \leq b$$

and dividing the result by the length of the path interval, s. Denoting the average film build over s by $\overline{h}$,

$$\overline{h} = \frac{1}{s} \int_{-b}^{b} h(y)\, dy \tag{2.12}$$

where $b \leq s \leq 2b$.

Evaluating equation (2.12)

$$\begin{aligned}\frac{1}{s}\int_{-b}^{b} h(y)\,dy &= \int_{-b}^{b} \frac{2q'\eta}{s\,v_i\pi b^2}\sqrt{b^2-y^2}\,dy \\ &= \frac{2q'\eta}{s\,v_i\pi b^2}\left\{\frac{y}{2}\sqrt{b^2-y^2}+\frac{b^2}{2}\arcsin\frac{y}{b}\right\}_{-b}^{b} \\ &= \frac{q'\eta}{s\,v_i} \qquad (2.13)\end{aligned}$$

Equation 2.13 is easily seen to be correct from a physical standpoint. The net volume of paint deposited in unit time is $q'\eta$. The area it is distributed over in that unit time is given by the product of the velocity and the path interval, v_i and s. Therefore, the average height required to constitute the known volume may be found.

In later work it will be useful to express $\overline{h}$ in terms of the overlap, β and spray pattern height, b. Using the expression for β of equation(2.3)

$$\overline{h} = \frac{q'\,\eta}{v_i\,s} = \frac{q'\,\eta}{2\,v_i\,b\,(1-\beta)} \qquad (2.14)$$

At this point the basic relationships between painting variables have been defined. The focus of the next chapter will be to apply these relationships to the path planning problem.

Chapter 3

Paint Path Generation

3.1 Introduction

The purpose of this chapter is to discuss techniques and algorithms that will be utilized to automatically generate spray painting paths.

Path generation based upon the special case of circular and more general elliptical spray patterns will be considered. A fundamental part of determining spray pass endpoints will be the use of offset curves. Methods of creating offset curves will be discussed for both circular and elliptical spray patterns.

Some workpieces may have voids that allow the paint to be shut off at particular intervals. In order to determine the shut off points the intersection points of a pass center with the offset curve have to be sorted in a logical sequence. A method allowing this sorting will be described.

Several factors, such as spray path orientation and path interval will be assumed fixed in this chapter. This allows explanation of the path generation in a direct manner. Chapter 4 will consider the optimization of these factors for improved paint performance.

3.2 Procedure for Path Generation

It will be assumed from the outset that the painting task will be accomplished using a series of parallel passes with a fixed path interval. The orientation of the passes will be horizontal through the workpiece, or equivelently, parallel to the X axis of the workpiece coordinate system. This choice of orientation is made for convenience in determining spray pass endpoints. Parallel horizontal passes are shown in Figure 3.1. Chapter 4 will discuss the approach for arbitrary path orientation.

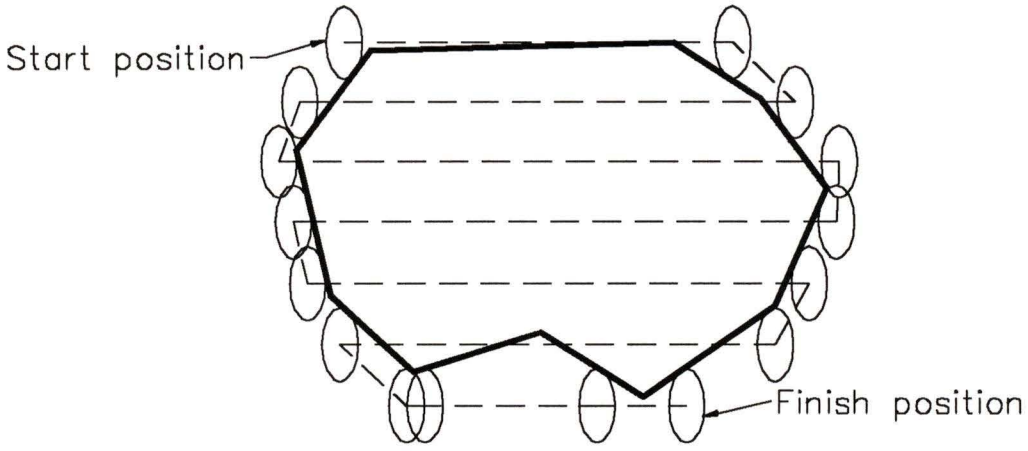

Figure 3.1: Parallel, horizontal spray passes over workpiece

With a fixed path interval and orientation the problem of path determination is effectively reduced to one of: a) locating suitable starting and ending points; and b) finding the individual spray pass endpoints. These will both be discussed in more detail in the sections below.

3.2.1 Locating Initial and Final Pass Heights

The painting path consists of a series of parallel passes that start at the top of the workpiece and descend in regular increments to the bottom of the workpiece. Let the y coordinate of each pass be termed the *pass height*, y_i. Then the pass height for each of n required passes would be

$$y = y_1, y_2, \ldots, y_n$$

A decision must be made as to what the first and last pass heights, y_1 and y_n, will be. A convenient method of defining these is to specify borders inset from the highest and lowest points on the workpiece. In Figure 3.2, y_s is the *upper border* defining where the first pass, y_1 can occur, and y_f is the *lower border* defining where the last pass, y_n, can occur, i.e.,

$$y_1 = y_s \tag{3.1}$$

$$y_n \leq y_f \tag{3.2}$$

The upper and lower borders, y_s and y_f, are defined with respect to y_{max}, and y_{min}, the highest and lowest points on the workpiece, and a specified constant, c, termed the *inset*, i.e.,

$$y_s = y_{max} - c \tag{3.3}$$

$$y_f = y_{min} + c \tag{3.4}$$

In this work y_1 is made to coincide with y_s, however, y_n will not in general coincide with y_f. Since each pass descends from the previous one an amount equal to the path

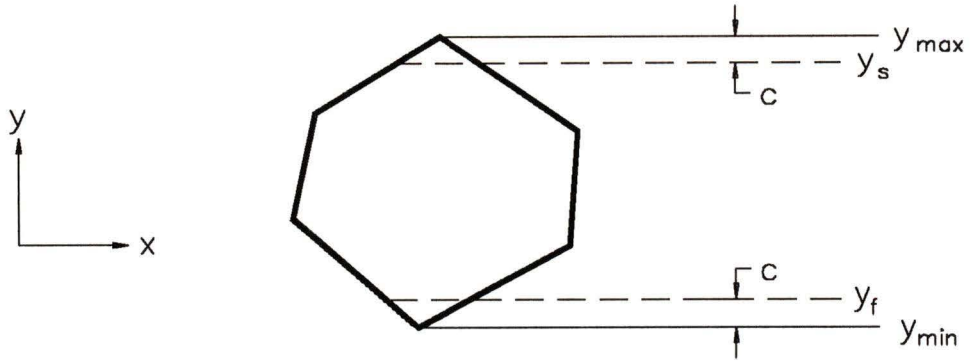

Figure 3.2: Initial and final pass heights

interval, s, y_n will only equal y_f when

$$\frac{y_s - y_f}{s} = \text{a whole number}$$

Together, equations (3.1) and (3.2) simply state that the spray passes start at y_s and continue to descend the work until such time the pass height falls below y_f. Note that (3.2) places an upper bound on y_n. Since additional spray passes beyond the lower edge of the work are of no value, equation (3.2) can be rewritten to include a lower bound, i.e.,

$$y_f - s < y_n \leq y_f \tag{3.5}$$

Clearly the inset, c, has to be determined before y_s and y_f can be found. In practice, satisfactory paint coverage can be obtained by setting $c = 0$. However, chapter 4 will describe two methods for determining non-zero c that decrease overall process cost.

3.2.2 Locating Spray Pass Endpoints

The individual spray pass endpoints are the points at which the spray pattern just clears the edge of the workpiece. Consider one line segment within the boundary of a workpiece. Furthermore, consider a circular spray pattern tangent to this boundary in several locations as shown in Figure 3.3. Note that the locus of spray pattern center points forms a line parallel to the boundary segment, and offset by a distance equal to the spray pattern radius.

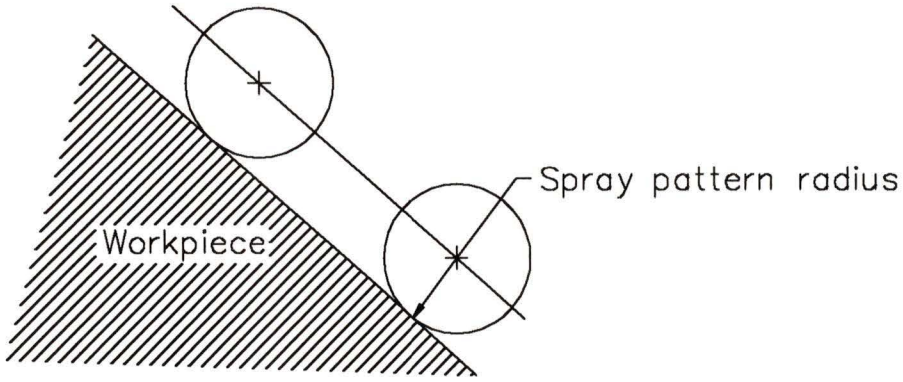

Figure 3.3: Spray pass end points

If all the lines segments forming a closed workpiece boundary were offset by the spray pattern radius and extended until they intersect one another they would appear as shown in Figure 3.4.

If an arbitrary horizontal line representing a spray pass center were laid over a boundary and its offset curve, the pass endpoints would occur at the intersection of the horizontal line and the offset curve. This is illustrated in Figure 3.5.

The type of offset curve required for determination of spray pass endpoints depends upon the shape and size of the spray pattern. This work will consider offset curves

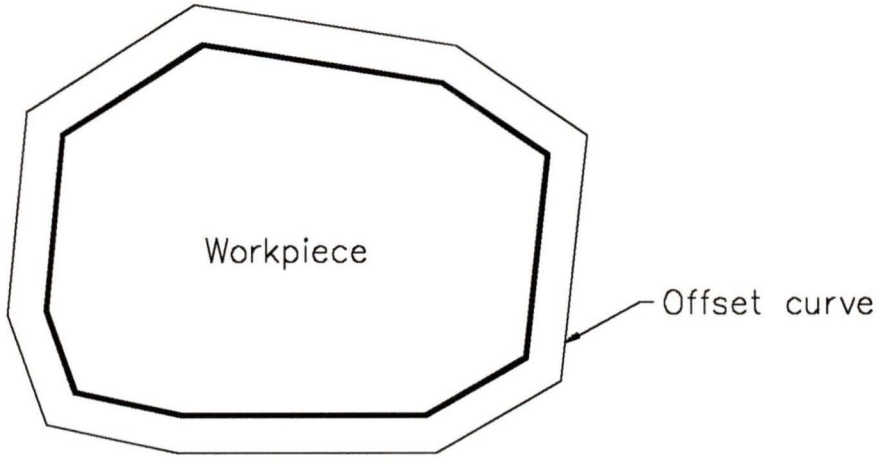

Figure 3.4: Complete offset curve

for circular and elliptical spray patterns.

The intersections of the pass center lines and the offset curve for the workpiece define a set of spray pass endpoints. To assemble the endpoint information into a painting path they must be sorted in some logical fashion. The method used in this work will be to sort from top to bottom and alternate horizontal directions between passes. Note that the paint is to be turned on between pass endpoints and turned off in the transition stage between passes.

The following sections of this chapter will discuss the techniques used to determine the offset curves, find pass endpoints, and sort the information into an appropriate spray path file.

3.3 Calculation of Offset Curves

The calculation of offset curves is performed based upon the geometric description of the workpiece boundary and the required offset distance. The format for representing

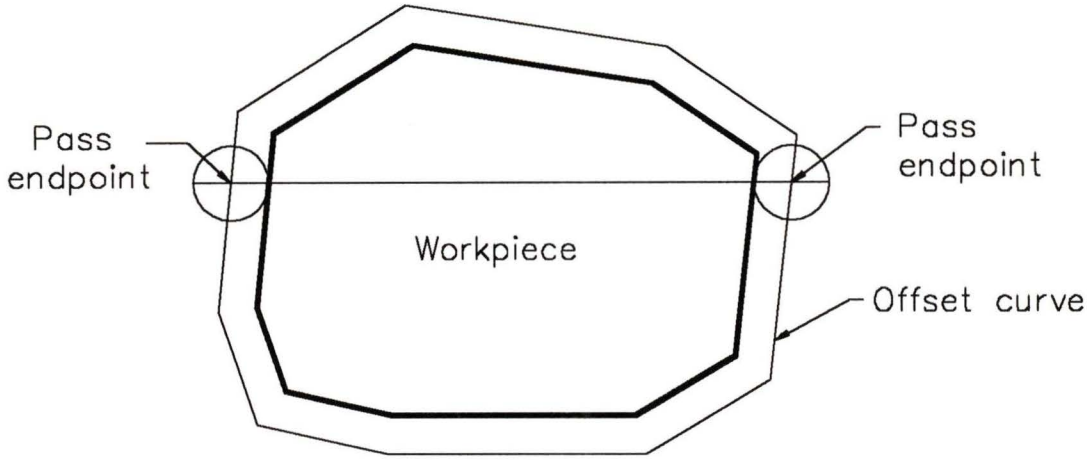

Figure 3.5: Spray pass endpoints

the workpiece boundary is presented below.

3.3.1 Workpiece Definition Format

In this work, the workpiece boundaries are represented by a series of contiguous line segments. Furthermore, it is assumed that the vertex list traverses the boundary in a counterclockwise fashion. This is an important consideration when creating offset curves and calculating workpiece area. In computer files the vertex list will form an array as follows

$$
\begin{array}{cc}
x_1 & y_1 \\
x_2 & y_2 \\
x_3 & y_3 \\
\vdots & \\
x_{n-1} & y_{n-1} \\
x_n & y_n
\end{array}
$$

Since the boundary is by definition closed the first vertex is the same as the last or

$$x_1 = x_n$$

$$y_1 = y_n$$

3.3.2 Offset Curves for a Circular Spray Pattern

To calculate the offset curve the boundary is processed in a pairwise fashion. Two connecting line segments are considered at one time. Consider the segments illustrated in Figure 3.6.

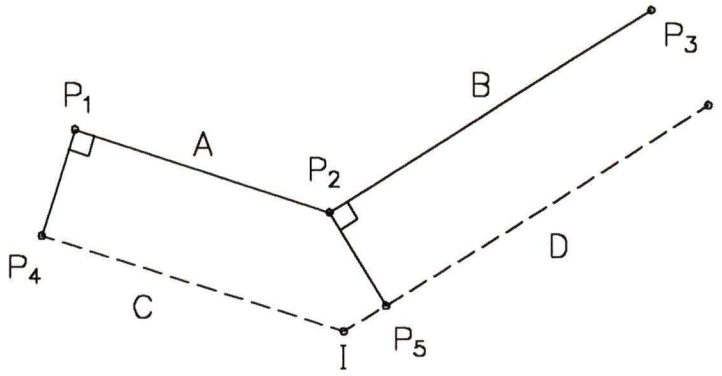

Figure 3.6: Workpiece boundary and offset curve line segments

The boundary segments A and B have known endpoints P_1, P_2 and P_3. The offset segments, C and D, are parallel to A and B respectively. Points P_4 and P_5 are defined by the intersection of the normals to A and B at the points P_1 and P_2 and the offset curves C and D. Point I is defined by the intersection of curves C and D. The objective is to determine the coordinates of I. To begin, the coordinates of points P_4 and P_5 are required.

Let $\underline{a}$ and $\underline{b}$ be unit vectors along A and B, respectively. Also let $\underline{c}$ and $\underline{d}$ be unit vectors along P_1P_4 and P_2P_5 as shown in Figure 3.7. Vector $\underline{c}$ can be found by taking

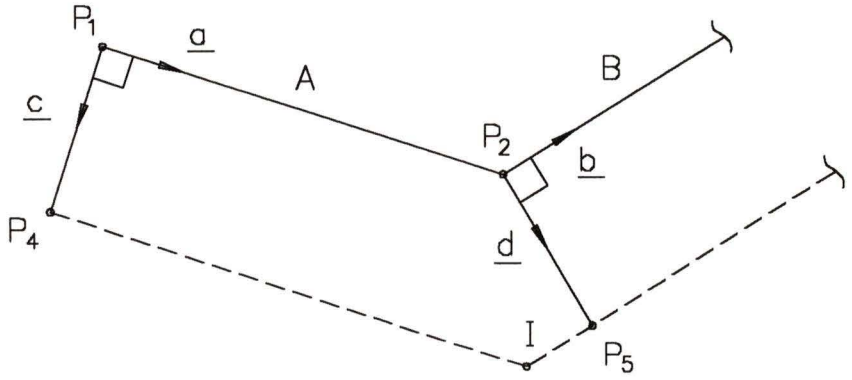

Figure 3.7: Unit vectors along A and P_1P_4

the vector product of $\underline{a}$ and $\underline{k}$, where $\underline{k}$ is a unit vector perpendicular to the plane of the work, i.e.,

$$\underline{c} = \underline{a} \times \underline{k} \tag{3.6}$$

Note that if the desired offset curve was on the left hand side of the boundary the unit vector $\underline{c}$ would be obtained from $\underline{c} = \underline{a} \times -\underline{k}$. The unit vector $\underline{c}$ is obtained from the coordinates P_1 and P_2. Let the length of A be denoted L_A. Then

$$L_A = \sqrt{(P_{2_x} - P_{1_x})^2 + (P_{2_y} - P_{1_y})^2} \tag{3.7}$$

and $\underline{a}$ is given by

$$\underline{a} = \frac{1}{L_A}\{P_{2_x} - P_{1_x}, P_{2_y} - P_{1_y}\}^T \tag{3.8}$$

Therefore, $\underline{a}$ has the components

$$a_x = \frac{P_{2_x} - P_{1_x}}{\sqrt{(P_{2_x} - P_{1_x})^2 + (P_{2_y} - P_{1_y})^2}} \tag{3.9}$$

$$a_y = \frac{P_{2_y} - P_{1_y}}{\sqrt{(P_{2_x} - P_{1_x})^2 + (P_{2_y} - P_{1_y})^2}} \tag{3.10}$$

Substituting the components of $\underline{a}$ and performing the vector product leaves

$$\underline{c} = \{a_y, -a_x\} \tag{3.11}$$

Now the point $\underline{P_4}$ may be obtained from

$$\underline{P_4} = \underline{P_1} + r\,\underline{c} \tag{3.12}$$

where r is the radius of the spray pattern, which is the desired offset value. Substituting for $\underline{c}$ and L_A yields

$$\underline{P_4} = \underline{P_1} + r\,\frac{\{P_{2_y} - P_{1_y}, P_{1_x} - P_{2_x}\}}{\sqrt{(P_{2_x} - P_{1_x})^2 + (P_{2_y} - P_{1_y})^2}} \tag{3.13}$$

In a similar fashion $\underline{P_5}$ may be found from

$$\underline{P_5} = \underline{P_2} + r\,\frac{\{P_{3_y} - P_{2_y}, P_{2_x} - P_{3_x}\}}{\sqrt{(P_{3_x} - P_{2_x})^2 + (P_{3_y} - P_{2_y})^2}} \tag{3.14}$$

To determine the intersection point, I, of offset curves C and D, their equations are written in parametric form, i.e., the coordinates of any point along C may be described by

$$\underline{P_C} = \underline{P_4} + \alpha_C\,\underline{a} \tag{3.15}$$

In a similar manner any point along offset curve D is described by

$$\underline{P_D} = \underline{P_5} + \alpha_D\,\underline{b} \tag{3.16}$$

The coordinates of the intersection point I are defined by the condition

$$\underline{P_C} = \underline{P_D} = \underline{I} \tag{3.17}$$

Equating the expressions for $\underline{P_C}$ and $\underline{P_D}$

$$\underline{P_4} + \alpha_C\,\underline{a} = \underline{P_5} + \alpha_D\,\underline{b}$$

rearranging into matrix form

$$[\underline{a} \ -\underline{b}] \left\{ \begin{array}{c} \alpha_C \\ \alpha_D \end{array} \right\} = \{\underline{P_5} - \underline{P_4}\}$$

and solving for $\underline{\alpha}$ gives

$$\begin{aligned} \underline{\alpha} &= \left\{ \begin{array}{c} \alpha_C \\ \alpha_D \end{array} \right\} = [\underline{a} \ -\underline{b}]^{-1} \{\underline{P_5} - \underline{P_4}\} \\ &= \frac{\left[\begin{array}{cc} -b_y & b_x \\ -a_y & a_x \end{array} \right]}{a_y b_x - a_x b_y} \{\underline{P_5} - \underline{P_4}\} \end{aligned}$$

Solving for α_C and α_D yields

$$\alpha_C = \frac{-b_y(P_{5_x} - P_{4_x}) + b_x(P_{5_y} - P_{4_y})}{a_y b_x - a_x b_y} \tag{3.18}$$

$$\alpha_D = \frac{-a_y(P_{5_x} - P_{4_x}) + a_x(P_{5_y} - P_{4_y})}{a_y b_x - a_x b_y} \tag{3.19}$$

Now either of equations (3.15) or (3.16) may be used to determine the intersection point.

Example-Offset Curves for a Circular Spray Pattern

As an example of offset curve determination consider the boundary line segments illustrated in Figure 3.8. The spray pattern radius is 7 units in length.

The known values of points $\underline{P_1}$, $\underline{P_2}$ and $\underline{P_3}$ are as follows

$$\underline{P_1} = \left\{ \begin{array}{c} 4.5 \\ 10.0 \end{array} \right\} \quad \underline{P_2} = \left\{ \begin{array}{c} 23.0 \\ 3.5 \end{array} \right\} \quad \underline{P_3} = \left\{ \begin{array}{c} 41.0 \\ 10.0 \end{array} \right\}$$

The lengths of A and B are determined from

$$L_A = \sqrt{(P_{2_x} - P_{1_x})^2 + (P_{2_y} - P_{1_y})^2} = \sqrt{(23.0 - 4.5)^2 - (3.5 - 10.0)^2} = 19.61$$

$$L_B = \sqrt{(P_{3_x} - P_{2_x})^2 + (P_{3_y} - P_{2_y})^2} = \sqrt{(41.0 - 23.0)^2 - (10.0 - 3.5)^2} = 19.14$$

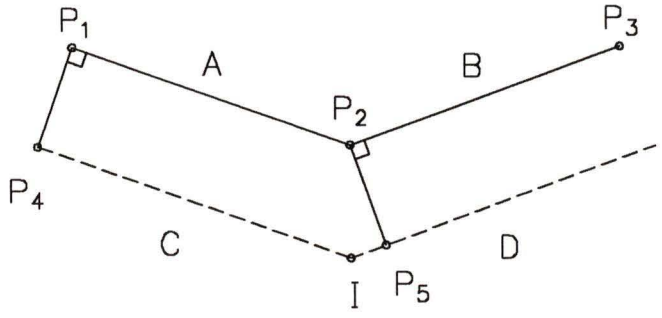

Figure 3.8: Example offset curves

and the unit vectors along A and B are given by

$$\begin{aligned} a_x &= \frac{P_{2_x} - P_{1_x}}{L_A} = \frac{23.0 - 4.5}{19.61} = 0.943 \\ a_y &= \frac{P_{2_y} - P_{1_y}}{L_A} = \frac{3.5 - 10.0}{19.61} = -0.332 \\ b_x &= \frac{P_{3_x} - P_{2_x}}{L_B} = \frac{41.0 - 23.0}{19.14} = 0.940 \\ b_y &= \frac{P_{3_y} - P_{2_y}}{L_B} = \frac{10.0 - 3.5}{19.14} = 0.340 \end{aligned}$$

The unit vectors along P_1P_4 and P_2P_5 may be found.

$$\begin{aligned} \underline{c} &= \left\{ \begin{array}{c} c_x \\ c_y \end{array} \right\} = \left\{ \begin{array}{c} a_y \\ -a_x \end{array} \right\} = \left\{ \begin{array}{c} -0.332 \\ -0.943 \end{array} \right\} \\ \underline{d} &= \left\{ \begin{array}{c} d_x \\ d_y \end{array} \right\} = \left\{ \begin{array}{c} b_y \\ -b_x \end{array} \right\} = \left\{ \begin{array}{c} 0.340 \\ -0.940 \end{array} \right\} \end{aligned}$$

The points $\underline{P_4}$ and $\underline{P_5}$ may now be determined.

$$\begin{aligned} \underline{P_4} &= \underline{P_1} + r\underline{c} = \left\{ \begin{array}{c} 4.5 \\ 10.0 \end{array} \right\} + 7.0 \left\{ \begin{array}{c} -0.332 \\ -0.943 \end{array} \right\} = \left\{ \begin{array}{c} 2.18 \\ 3.40 \end{array} \right\} \\ \underline{P_5} &= \underline{P_2} + r\underline{d} = \left\{ \begin{array}{c} 23.0 \\ 3.5 \end{array} \right\} + 7.0 \left\{ \begin{array}{c} 0.340 \\ -0.940 \end{array} \right\} = \left\{ \begin{array}{c} 25.38 \\ -3.08 \end{array} \right\} \end{aligned}$$

Now the parameters for each offset curve can be found. However, only one is required

to locate the intersection point. α_c will be used in this example.

$$\begin{aligned}\alpha_C &= \frac{-b_y(P_{5_x} - P_{4_x}) + b_x(P_{5_y} - P_{4_y})}{a_y b_x - a_x b_y} \\ &= \frac{-0.340(25.38 - 2.176) + 0.940(-3.08 - 3.400)}{(-0.332)(0.940) - (0.943)(0.340)} = 22.10\end{aligned}$$

Finally the intersection point is given by

$$\underline{I} = \underline{P_4} + \alpha_C \underline{a} = \left\{ \begin{array}{c} 2.176 \\ 3.400 \end{array} \right\} + 22.10 \left\{ \begin{array}{c} 0.943 \\ -0.332 \end{array} \right\} = \left\{ \begin{array}{c} 23.02 \\ -3.94 \end{array} \right\}$$

This process of locating intersection points is continued until the entire boundary has been traversed and the offset curve completely defined.

The techniques just illustrated apply to the case of a constant offset distance, as would be the case with a circular spray pattern. However, the majority of painting is accomplished with an elliptical spray pattern. Therefore the offset curve determination must be modified to suit. This will be done in the following section.

3.3.3 Offset Curves for Elliptical Spray Pattern

For painting with an elliptical spray pattern the distance the pattern must extend beyond the part boundary is dependent upon the slope of each individual line segment. Specifically, the pattern must extend until it is tangent to the workpiece boundary as illustrated in Figure 3.9.

For a particular boundary line segment each ellipse must extend to the tangency point. The locus of all pass endpoints will form a line parallel to the workpiece boundary as shown in Figure 3.10.

Each segment of the boundary has an associated offset line. When considered together the offset lines will amount to a variable distance offset curve, as illustrated

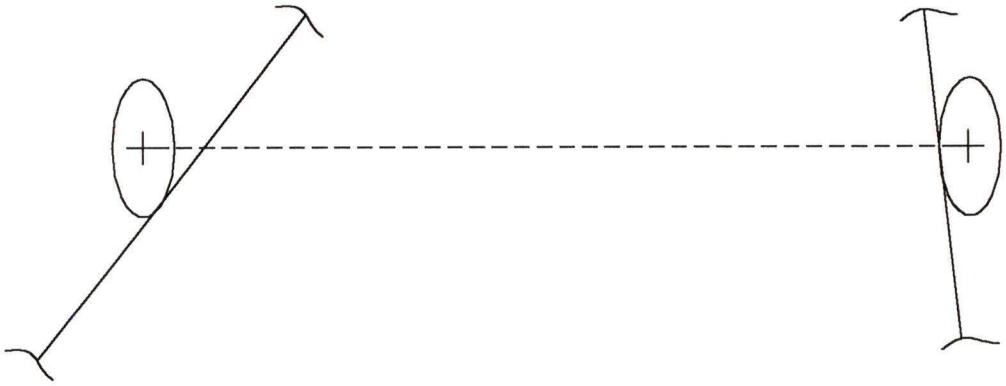

Figure 3.9: Spray pattern tangent to boundary

in Figure 3.11. The development of the variable distance offset curves is similar to that of the regular offset curves. The distinction is that for the variable offset curves the offset distance must be calculated separately for each individual segment.

The offset distance is the perpendicular distance between the ellipse center and the boundary line segment as shown in Figure 3.12.

To solve for the offset distance d consider the ellipse aligned with the coordinate directions as shown in Figure 3.12. The point of tangency of the ellipse and the boundary line is indicated as T in Figure 3.12. The slope of the boundary line is known, or at least may be found from its endpoint coordinates. Consider Figure 3.13. The tangency point can be found by solving the general equation of an ellipse for y and differentiating with respect to x. The standard form for the ellipse equation is

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Solving for y gives

$$y = b\sqrt{1 - \frac{x^2}{a^2}} \ , \ a \neq 0 \tag{3.20}$$

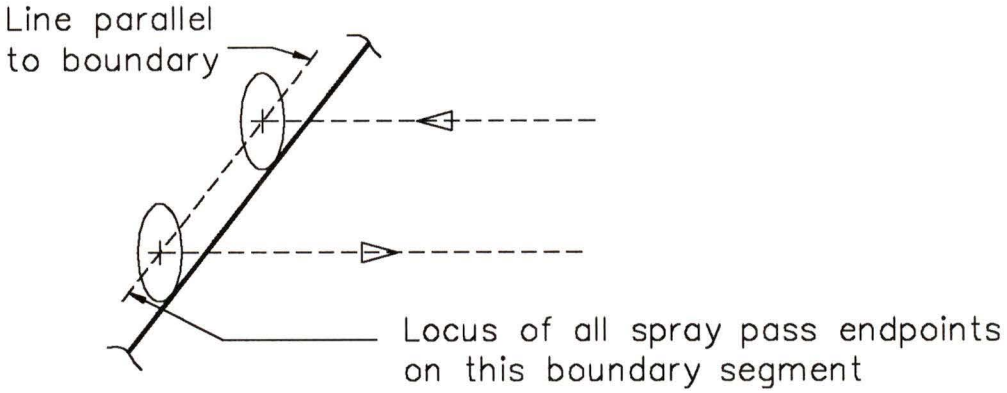

Figure 3.10: Locus of spray pattern endpoints

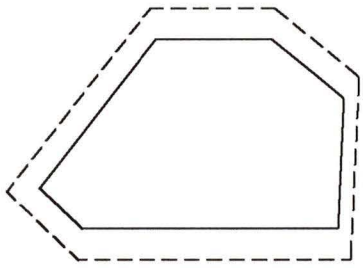

Figure 3.11: Variable distance offset curve

Differentiation with respect to x provides

$$\begin{aligned} \frac{dy}{dx} &= \frac{-bx}{a^2}\frac{1}{\sqrt{1-\frac{x^2}{a^2}}} \\ &= \frac{-bx}{a\sqrt{a^2-x^2}} \quad , -a < x < a \end{aligned}$$

Denoting dy/dx by y', squaring both sides, and solving for x yields

$$x = \frac{a^2\, y'}{\sqrt{b^2 + y'^2 a^2}} \tag{3.21}$$

Since y' is known, equation (3.21) can be solved. Then y can be determined from equation (3.20).

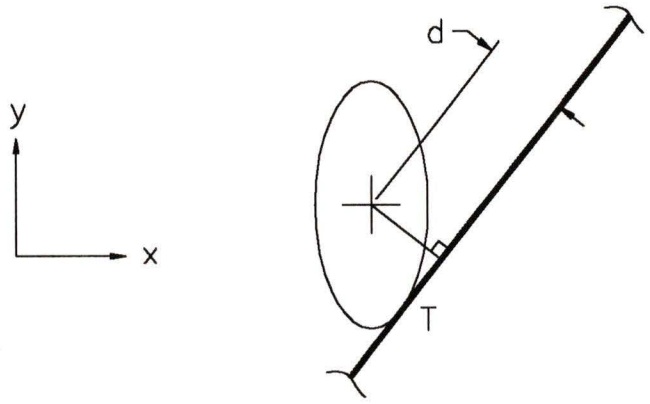

Figure 3.12: Offset distance

With the coordinates of T known, the distance, ℓ, from the ellipse center to the tangency point may be obtained through

$$\ell = \sqrt{x^2 + y^2}$$

To find the shortest (the offset) distance d, the angle γ between d and l is required. This angle can be resolved by

$$\gamma = \theta - \phi$$

where ϕ is the angle between ℓ and the y axis direction, i.e.,

$$\phi = \arctan\left(\frac{|x|}{|y|}\right)$$

Finally the required offset distance is

$$d = \ell \cos\gamma$$

Using the previous results for ℓ, ϕ and γ, d can be rewritten as

$$d = \sqrt{x^2 + y^2}\ \cos\left(|\theta| - \arctan\left(\frac{x}{y}\right)\right) \tag{3.22}$$

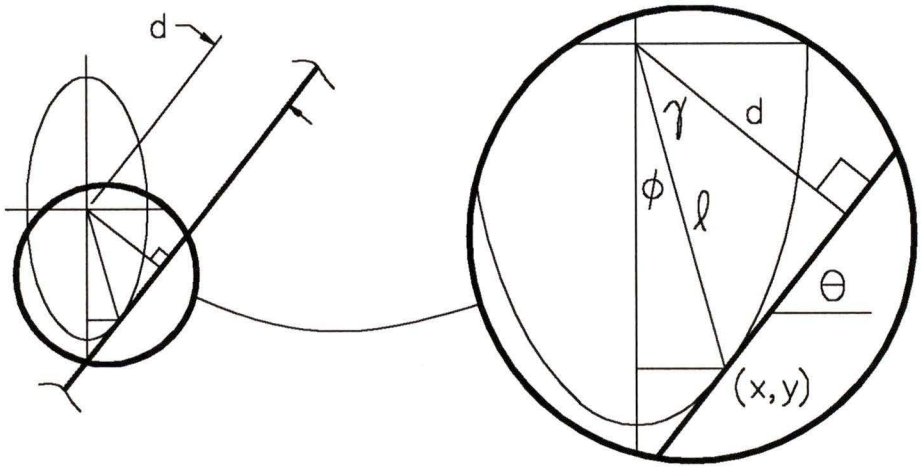

Figure 3.13: Calculation of offset distance

It was stated previously that the input to this problem is the boundary segment slope. To maintain consistency in sign the absolute value of the slope is used. This is due to the fact that the desired offset distance is always positive. To illustrate, four possible configurations of a 30^o angle are shown in Figure 3.14. All yield the same offset distance.

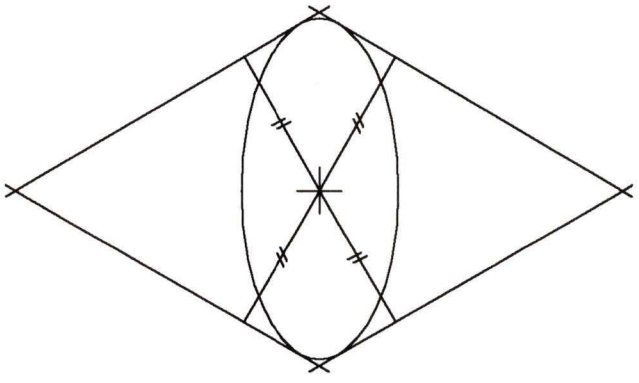

Figure 3.14: Constant offset distance for all signs of angle

Example-Offset Curve for Elliptical Spray Pattern

Two line segments, A and B, are part of a workpiece boundary. It is required to find the appropriate offset distances, g_1 and g_2 and the intersection point I, for the ellipse size shown. The coordinates of points $\underline{P_1}$, $\underline{P_2}$ and $\underline{P_3}$ are as follows.

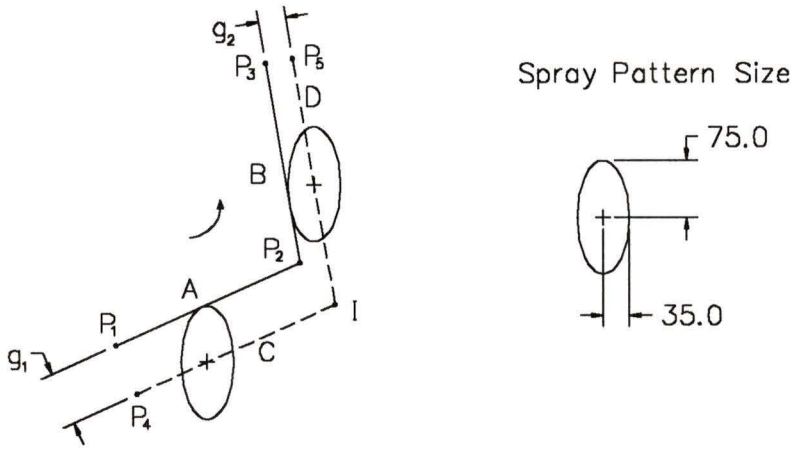

Figure 3.15: Offset curves for elliptical spray pattern

$$\underline{P_1} = \begin{Bmatrix} 16.0 \\ 108.0 \end{Bmatrix} \quad \underline{P_2} = \begin{Bmatrix} 261.0 \\ 218.0 \end{Bmatrix} \quad \underline{P_2} = \begin{Bmatrix} 216.0 \\ 478.0 \end{Bmatrix}$$

The required offset distances g_1 and g_2 will be calculated first. Starting with g_1 the slope of A is denoted as y'_A.

$$y'_A = \frac{P_{2_y} - P_{1_y}}{P_{2_x} - P_{1_x}} = \frac{218.0 - 108.0}{261.0 - 16.0} = 0.449$$

For g_1 the point of tangency between the ellipse and boundary segment A with respect to a coordinate frame centered at the ellipse origin is given by equations (3.21) and (3.20).

$$x = \frac{a^2 y'_A}{\sqrt{b^2 + y'^2_A a^2}} = \frac{(35.0)^2(0.449)}{\sqrt{75.0^2 + (0.449)^2(35.0)^2}} = 7.178$$

$$y = b\sqrt{1-\frac{x^2}{a^2}} = (75.0)\sqrt{1-\frac{7.178^2}{35.0^2}} = 73.41$$

g_1 may now be obtained from equation 3.22

$$\begin{aligned} g_1 &= \sqrt{x^2+y^2}\,\cos\left(Tan^{-1}(y'_A) - Tan^{-1}\left(\frac{x}{y}\right)\right) \\ &= \sqrt{7.178^2+73.406^2}\,\cos\left(Tan^{-1}(0.449) - Tan^{-1}\left(\frac{7.178}{73.406}\right)\right) = 69.906 \end{aligned}$$

The same procedure may now be used to determine g_2. The slope of B is given by

$$y'_B = \left|\frac{P_{3_y} - P_{2_y}}{P_{3_x} - P_{2_x}}\right| = \left|\frac{478.0 - 218.0}{216.0 - 261.0}\right| = 5.778$$

Now the point of tangency is found from

$$\begin{aligned} x &= \frac{a^2\, y'_B}{\sqrt{b^2 + y'^2_B a^2}} = \frac{(35.0)^2(5.778)}{\sqrt{75.0^2 + (5.778)^2(35.0)^2}} = 32.816 \\ y &= b\sqrt{1-\frac{x^2}{a^2}} = (75.0)\sqrt{1-\frac{32.816^2}{35.0^2}} = 26.080 \end{aligned}$$

g_2 may now be obtained from equation (3.22)

$$\begin{aligned} g_2 &= \sqrt{x^2+y^2}\,\cos\left(Tan^{-1}(y'_B) - Tan^{-1}\left(\frac{x}{y}\right)\right) \\ &= \sqrt{32.816^2+26.080^2}\,\cos\left(Tan^{-1}(5.7778) - Tan^{-1}\left(\frac{32.816}{26.080}\right)\right) = 36.78 \end{aligned}$$

With the offset curves determined work can begin on determining the pass endpoints.

3.4 Determination of Pass Endpoints

Consider the workpiece boundary and its offset curve shown in Figure 3.16. A horizontal line representing the center line of the spray pattern is passed through the boundary.

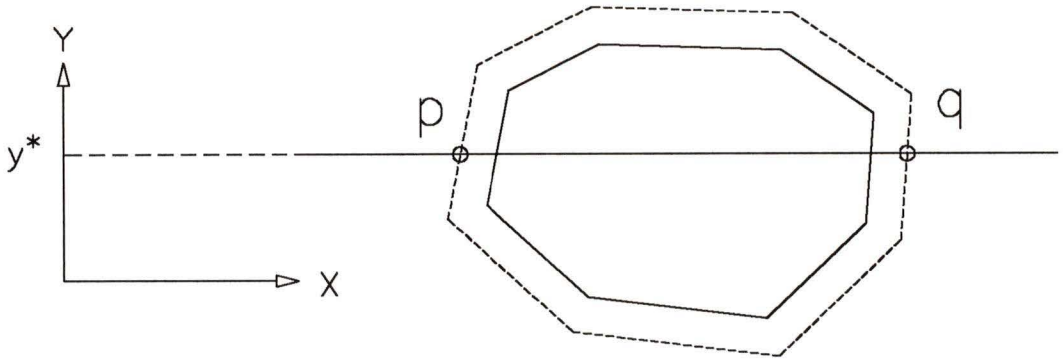

Figure 3.16: Horizontal path through workpiece

The point at which the horizontal line intersects the boundary *offset curve* is either the beginning or end of that particular spray pass. Each element of the offset curve is tested for intersection with the line pq. If the offset curve comprises n vertices, $v_1, v_i, v_{i+1}, \ldots, v_n$, then the following test is performed for $i = 1 \ldots n$. An intersections exists if either

$$v_i < y^* \quad and \quad v_{i+1} > y^*$$

or

$$v_{i+1} < y^* \quad and \quad v_i > y^*$$

This test will find either no intersections, or an even number of intersections for any given horizontal line. If there are two or more intersections along a given line they are sorted in order of increasing X coordinate depending upon whether the pass is travelling from left to right or from right to left. For a given pass the first intersection point requires that the paint be turned on, successive intersections toggle it off and on until the final intersection point is reached. At the final intersection point the paint is turned off.

After all intersections are determined and sorted for a given line, y^* is decreased

by an amount equal to the path interval and the process is repeated. This is continued until the pass center is lower than y_{min}.

At this point the techniques for generating paint paths for a given workpiece geometry are in place. The next chapter will discuss methods for measuring spray path performance, and the optimization of painting parameters for improved performance.

Chapter 4

Optimization of the Spray Path

4.1 Introduction

In chapters 2 and 3 techniques were developed for generation of spray painting paths from the workpiece boundary. Several painting parameters were assumed fixed in the development, including the overlap, the spray pattern size, the flowrate, the velocity, the spray path orientation, and selection of the inset.

In this chapter methods are developed for optimization of the spray path with respect to economic cost. Section 4.2 deals with the development of a process cost model that permits objective measurement of spray path cost. In section 4.3 it is shown that the extent of overlap between two consecutive spray passes determines the uniformity of paint thickness across the path interval. Three measures of uniformity will be presented as a basis for optimization of the overlap. Section 4.4 offers two alternative methods for selection of the inset.

The minimization of process cost is performed by an exhaustive search over the spray pattern size and spray pass orientation. Sections 4.5 and 4.6 examine the relationship between process cost and spray pattern size and pass orientation, re-

spectively. A description of the program which implements the work of this thesis is provided in section 4.7. Finally, section 4.8 illustrates spray path optimization through the consideration of three examples.

4.2 Development of the Process Cost Model

The objective of painting in an industrial environment is to obtain the desired finish quality at the lowest possible economic cost. This economic cost will be referred to as the *process cost* and denoted by Φ. The process cost model developed here consists of five components. Two are proportional to process time and three are proportional to paint usage.

4.2.1 Costs Proportional to Process Time

Let the *process time*, T, be defined as the time required for the sprayhead to traverse the entire spray path. The process time is comprised of two components, the *paint on time* and the *paint off time*. The paint on time, T_i, is the time required to travel the portion of the path where the sprayhead is on, and the paint off time, T_o, is the time required to travel the portion of the path where the paint is off. T can be written

$$T = T_i + T_o$$

Let X_i and X_o be the path lengths when the paint is on and off respectively, then

$$X_i = \sum x_i$$

$$X_o = \sum x_o$$

where x_i and x_o are individual path segment lengths. Now if v_i and v_o represent the paint on and paint off velocities respectively, T_i and T_o can be obtained from

$$T_i = \frac{X_i}{v_i} \tag{4.1}$$

$$T_o = \frac{X_o}{v_o} \tag{4.2}$$

To relate the process times of (4.1) and (4.2) to process cost, a proportionality constant is required. Let k_1 represent the cost per unit time of machine operation. The constant k_1 may include labour, maintenance, energy consumption or other costs that are proportional to the time the machine is in operation. The actual value of k_1 is particular to the painting process under consideration. The dimensions of k_1 are $[\$/T]$, where \$ represents cost. The process cost due to paint on time, Φ_{T_i}, is given by

$$\Phi_{T_i} = k_1 \, T_i = k_1 \, \frac{X_i}{v_i} \tag{4.3}$$

and the process cost due to paint off time, ϕ_{T_o}, is given by

$$\Phi_{T_o} = k_1 \, T_o = k_1 \, \frac{X_o}{v_o} \tag{4.4}$$

For comparison purposes, a minimum process time may be found for painting any given workpiece. Given a maximum volume flowrate from the gun, q'_{max}, the surface area of the workpiece to be coated, A_s, and a desired film thickness, $\overline{h}$, the minimum process time may be calculated from the relation

$$T_{min} = \frac{A_s \overline{h}}{\eta q'_{max}} \tag{4.5}$$

where η is the transfer efficiency. Due to several factors this minimum time may be approached but not reached. Specifically, influences such as tool movement with

the paint off and spraying outside of the part boundary increase the time required to complete the spraying process. However, T_{min} may be used as a convenient benchmark for evaluation of proposed spray paths.

4.2.2 Costs Proportional to Paint Usage

Paint usage is divided into three components in this work.

1. Paint loss due to overspray
2. Paint loss due to spray by
3. Paint deposited on the work

Each component is a portion of the nominal flowrate, q'. The three components are described below, beginning with the overspray.

Overspray

Overspray, as introduced in section 2.2, is the portion of the finely atomized paint that is carried away from the workpiece by air currents within the work environment. The paint lost due to overspray, Q_{os}, is given by

$$Q_{os} = q' (1 - \eta) T_i = q' (1 - \eta) \frac{X_i}{v_i} \tag{4.6}$$

Denoting the cost of Q_{os} as Φ_{os}, it can be expressed

$$\Phi_{os} = (k_2 + k_3) Q_{os} = (k_2 + k_3) q' (1 - \eta) \frac{X_i}{v_i} \tag{4.7}$$

where k_2 is a constant reflecting the cost of the paint itself and k_3 is a constant representing the cost for filtering, collection and disposal of the overspray. The dimensions of both k_2 and k_3 are cost per unit volume, $[\$/L^3]$.

Spray by

The amount of paint flow available for deposit on the work after the loss due to overspray is termed the net flowrate. The net flowrate q, as described in chapter 2, is the product of the nominal flowrate and the transfer efficiency, i.e.,

$$q = q' \, \eta$$

The portion of paint represented by q can do one of two things: it may land on the workpiece; or it may be sprayed past the edge of the workpiece boundaries. The portion sprayed past the workpiece boundaries is termed *spray by.* Spray by generally occurs during the first and last spray passes, and at the ends of each intermediate pass, or segments of a pass. The volume of paint lost due to spray by, Q_{sb}, is found by subtracting the volume of paint that lands on the work from q, i.e.,

$$Q_{sb} = q' \, \eta \, T_i - A_s \, \overline{h} = q' \, \eta \, \frac{X_i}{v_i} - A_s \, \overline{h} \tag{4.8}$$

where A_s[1] is the area of the workpiece and $\overline{h}$ is the specified paint thickness. The costs associated with the spray by, Φ_{sb}, are attributable to the cost of the paint itself and its collection and disposal. Φ_{sb} is given by

$$\Phi_{sb} = (k_2 + k_4) \, Q_{sb} = (k_2 + k_4) \left\{ \eta \, q' \, \frac{X_i}{v_i} - A_s \, \overline{h} \right\} \tag{4.9}$$

Note that the cost constant for collection and disposal of spray by, k_4, may be different than the cost constant for overspray, k_3.

[1]The method used to determine workpiece area is described in appendix A

Paint Deposited on the Work

The portion of the net paint spray that is deposited on the work, Q_w, is given by the product of the workpiece area and the specified paint thickness, i.e,

$$Q_w = A_s \, \overline{h} \tag{4.10}$$

The cost of paint deposited on the work, Φ_w, is given by

$$\Phi_w = k_2 \, A_s \, \overline{h} \tag{4.11}$$

The only cost constant used in the determination of Φ_w is k_2 since there are no collection or disposal costs associated with Q_w.

The overall process cost model can now be formed by combining equations (4.3), (4.4), (4.7), (4.9) and (4.11), i.e.,

$$\begin{aligned} \Phi &= \Phi_{T_i} + \Phi_{T_o} + \Phi_{os} + \Phi_{sb} + \Phi_w \\ &= k_1 \frac{X_i}{v_i} + k_1 \frac{X_o}{v_o} + (k_2 + k_3)(1 - \eta) \, q' \frac{X_i}{v_i} + \\ &\quad (k_2 + k_4) \left\{ \eta \, q' \frac{X_i}{v_i} - A_s \, \overline{h} \right\} + k_2 \, A_s \, \overline{h} \end{aligned} \tag{4.12}$$

The variables q' and v_i can be eliminated from the third and fourth terms of (4.12) by rearranging equation (2.14) for q', i.e.,

$$\begin{aligned} \overline{h} &= \frac{q' \, \eta}{2 \, v_i \, b \, (1 - \beta)} \\ \Rightarrow q' &= \frac{2 \, \overline{h} \, v_i \, b \, (1 - \beta)}{\eta} \end{aligned} \tag{4.13}$$

Substituting the RHS of (4.13) into (4.12) for q' and simplifying gives

$$\begin{aligned} \Phi &= k_1 \frac{X_i}{v_i} + k_1 \frac{X_o}{v_o} + 2 \, (k_2 + k_3) \frac{(1 - \eta)}{\eta} \, \overline{h} \, b \, (1 - \beta) \, X_i + \\ &\quad (k_2 + k_4) \left\{ 2 \, b \, X_i \, \overline{h} \, (1 - \beta) - A_s \, \overline{h} \right\} + k_2 \, A_s \, \overline{h} \end{aligned} \tag{4.14}$$

Equation (4.14) represents the overall process cost for painting an arbitrary planar workpiece. To minimize the process cost, Φ, an appropriate choice of variables in (4.14) must be made. Section 4.2.3 will examine how each of the variables are treated in the optimization process.

4.2.3 Minimization of the Process Cost Model

Of the terms appearing in equation (4.14), k_1, k_2, k_3, k_4, $\overline{h}$, η, v_o and A_s are specified inputs or constants. Only the variables X_i, X_o, v_i, b and β can be considered in the optimization.

Section 4.3 will show that the overlap, β, determines the uniformity of paint thickness across the path interval. Therefore, β will be selected independently prior to the minimization of the process cost and can be considered a constant for the remainder of this discussion.

The paint on and paint off path lengths, X_i and X_o are not independant variables but instead functions of the spray pass orientation, θ, and the spray pattern size, b. The relationships for

$$\begin{aligned} X_i &= f(b, \theta) \\ X_o &= g(b, \theta) \end{aligned}$$

cannot be written for a general workpiece. Furthermore, the relationship does not lend itself to minimization using an optimization scheme. Therefore, the minimization of the cost, Φ, given in equation (4.14) will be performed by an exhaustive search over the feasible regions of b and θ. Since b and θ are both bounded, a search using practical search increments does not pose an unreasonable computational burden.

Each iteration of the search over b and θ yields values for X_i and X_o so that all terms of equation (4.14) can be evaluated with the exception of the first, which contains the as yet undetermined variable, v_i.

The spray on velocity, v_i, is set to the maximum value possible in order to minimize the first term of equation (4.14). The maximum value of v_i is limited by two factors. The first limiting factor is the maximum machine velocity, v_o. The paint on velocity cannot be greater than the maximum velocity of the robot, i.e.,

$$0 < v_i \leq v_o$$

The second limiting factor of v_i is the constraint of equation (2.14). For a particular spray pattern size, b, and a feasible upper limit on nominal flowrate, q'_{max}, the upper limit on v_i is obtained by rearranging (2.14), i.e.,

$$v_i = \frac{q'_{max}\,\eta}{2\,b\,\overline{h}\,(1-\beta)} \tag{4.15}$$

If v_i from equation(4.15) exceeds v_o it is reset so that

$$v_i = v_o$$

and q' is determined by rearranging (2.14), i.e.,

$$q' = \frac{2v_i b\overline{h}(1-\beta)}{\eta} \tag{4.16}$$

At this point Φ can be determined and recorded along with current values of b and θ. This process is repeated over the feasible range of b and θ. The minimum resulting Φ represents the optimal paint path.

As mentioned earlier in this section, the overlap, β, may be optimized independently prior to minimization of the process cost model. The next section will discuss techniques for obtaining the optimum overlap for several objectives.

4.3 Optimization of Overlap

4.3.1 Introduction

The overlap, β, describes the extent to which two consecutive spray passes extend over one another. When spray coating is performed manually the overlap is often set at a value of 50 percent. This provides a convenient target for each successive spray pass. In robotic spray application an overlap of 50 percent is also frequently used [25].

In this section, it will be shown that it is the overlap that controls the uniformity of surface finish thickness. Several measures for uniformity of paint thickness will be developed and discussed. Using the developed measures, optimum values of β will be determined.

Before proceeding the reader is reminded that this discussion and results are based on the assumption of an elliptical spray pattern with uniform paint distribution. However, the introduced measures of uniformity can be applied to other paint distributions and spray patterns.

Recall that the thickness of paint across the path interval is expressed as a composite function resulting from the overlap of two deposition profiles (§2.10). In this discussion the composite function is termed the *interval profile*. The interval profile is simply the curve representing the actual thickness of paint deposited over the path interval.

Ideally the interval profile would be a constant function equal in magnitude to the specified paint thickness $\overline{h}$, as shown in Figure 4.1(a). Instead it is of the form shown in Figure 4.1(b). Note that the graphs of Figure 4.1 have been normalized; the

ordinate values with respect to the average paint thickness, $\overline{h}$, and the abscissa values with respect to the path interval, s. Figure 4.2 shows two more interval profiles, each

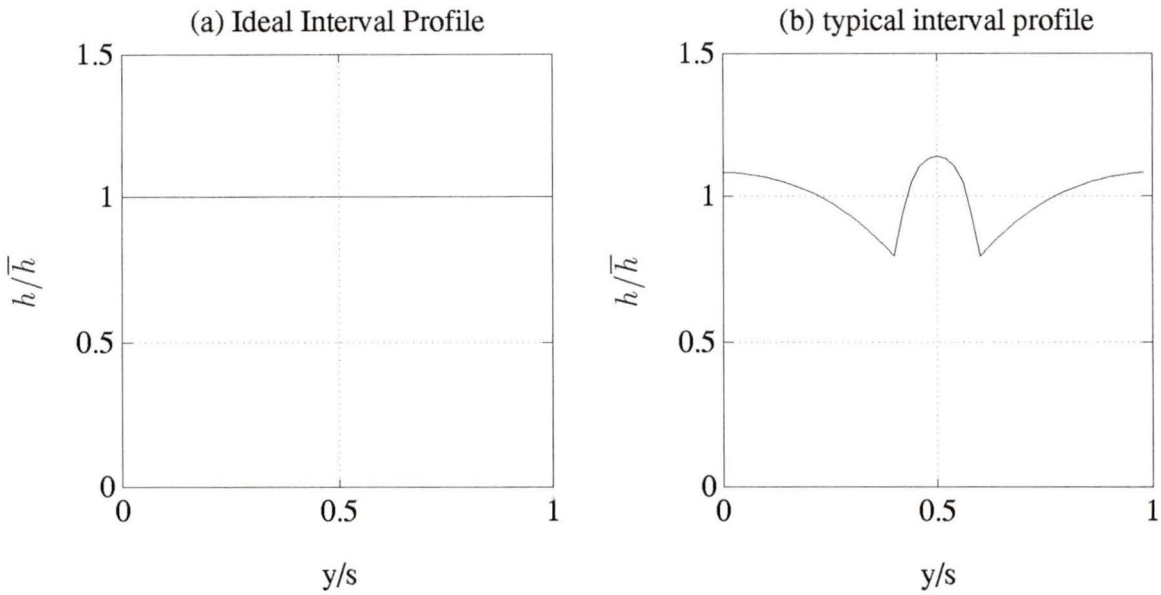

Figure 4.1: Ideal and typical interval profiles

based on a different value of β. The interval profile of Figure 4.2(a) which is based on $\beta = 15\%$, appears to maintain a closer proximity to the average paint thickness, (represented by $h/\overline{h} = 1$), than that of Figure 4.2(b) which is based on $\beta = 35\%$. Before a judgement can be made as to which level of overlap is better, an objective measure of uniformity is required that will permit meaningful comparison between the two. The purpose of the following discussion is to develop such measures and to determine β such that the uniformity is optimal.

The measures introduced to quantify paint thickness uniformity will be based on the deviation of paint thickness from the average paint thickness. Three particular measures of this type will be considered. The first measure is the average arithmetic

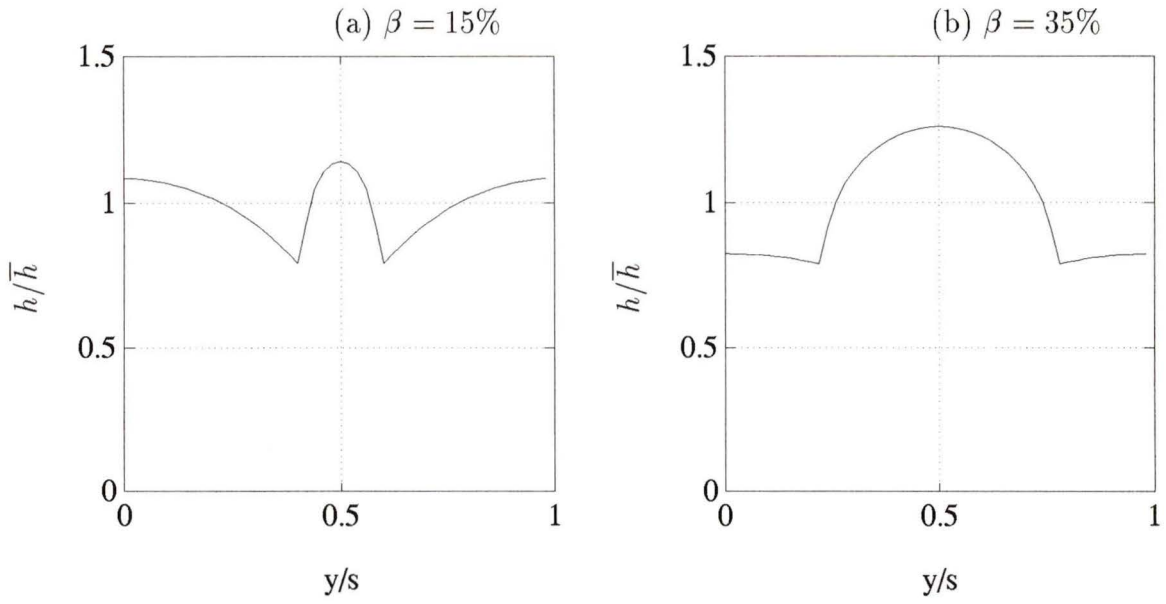

Figure 4.2: Interval profiles for different β

deviation from the mean. The second measure is the root mean square deviation, and the final measure is the maximum deviation from the mean. Each of these measures will be developed and discussed in the following text.

4.3.2 Average Arithmetic Deviation from the Mean

The average arithmetic deviation from the mean corresponds to a measure of *roughness* used in machine design [10]. Denoting the roughness as R_1, it may be obtained from

$$R_1 = \frac{1}{s} \int_0^s |h - \overline{h}| \, dy \tag{4.17}$$

where $h = h(y)$ is the interval profile. To generalize (4.17), normalize with respect to the average paint thickness $\overline{h}$, i.e.,

$$R_1 = \frac{1}{s} \int_0^s \left| h/\overline{h} - 1 \right| dy \tag{4.18}$$

To perform the integration of (4.18) numerically, it is rewritten in the form

$$\begin{aligned} R_1 &= \frac{1}{s}\sum_{i=1}^{n}\left|\frac{\hat{h}_i+\hat{h}_{i+1}}{2}-1\right|\frac{s}{n} \\ &= \frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{h}_i+\hat{h}_{i+1}}{2}-1\right| \end{aligned} \tag{4.19}$$

where n is the number of intervals used to approximate the integration and $\hat{h}$ is given by

$$\hat{h} = h(y)/\overline{h}$$

The integration of (4.19) was performed and the results are shown in the graph of Figure 4.3. From the data used to generate Figure 4.3 it can be found that the minimum deviation R_1 occurs at $\beta = 0.175 = 17.5\%$.

4.3.3 Root Mean Square Deviation from the Mean

The root mean square (RMS) deviation is also used as a measure of roughness in machine design [10]. The RMS deviation is obtained from

$$R_2 = \left\{\frac{1}{s}\int_0^s (\hat{h}-1)^2\,dy\right\}^{1/2} \tag{4.20}$$

Rewriting (4.20) in a form suitable for numerical integration provides

$$R_2 = \left\{\frac{\sum_{i=1}^{n}\left(\frac{(\hat{h}_i+\hat{h}_{i+1})}{2}-1\right)^2}{n}\right\}^{1/2} \tag{4.21}$$

The values of R_2 as a function of β were calculated and are presented in the graph of Figure 4.4. The RMS deviation is minimum at $\beta = 16\%$.

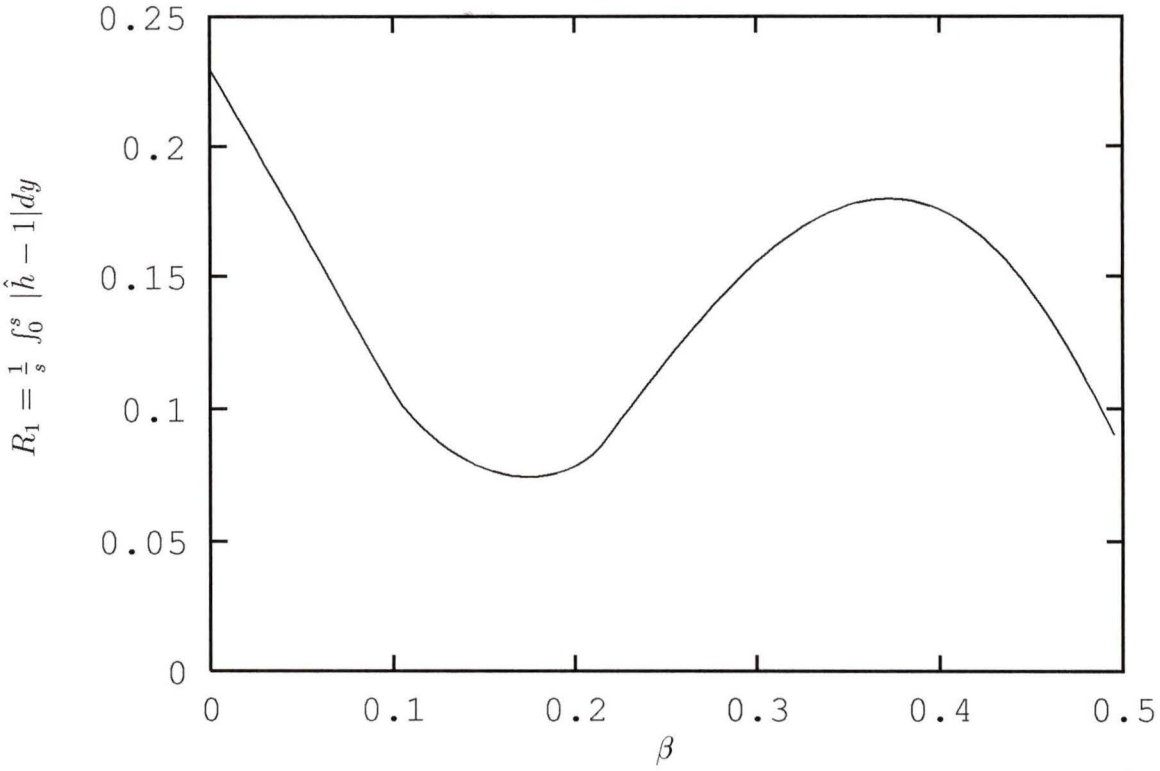

Figure 4.3: R_1 as a function of β

4.3.4 Maximum Deviation from the Mean

A third technique to quantify uniformity is to base it on the magnitude of the maximum deviation of paint thickness from the average paint thickness across the path interval. Let R_3 denote the maximum deviation, then

$$R_3 = max_i \, |\hat{h} - 1|$$

The magnitude of R_3 was calculated as a function of β and the results were plotted in Figure 4.5. The minimum R_3 occurs for $\beta = 18\%$.

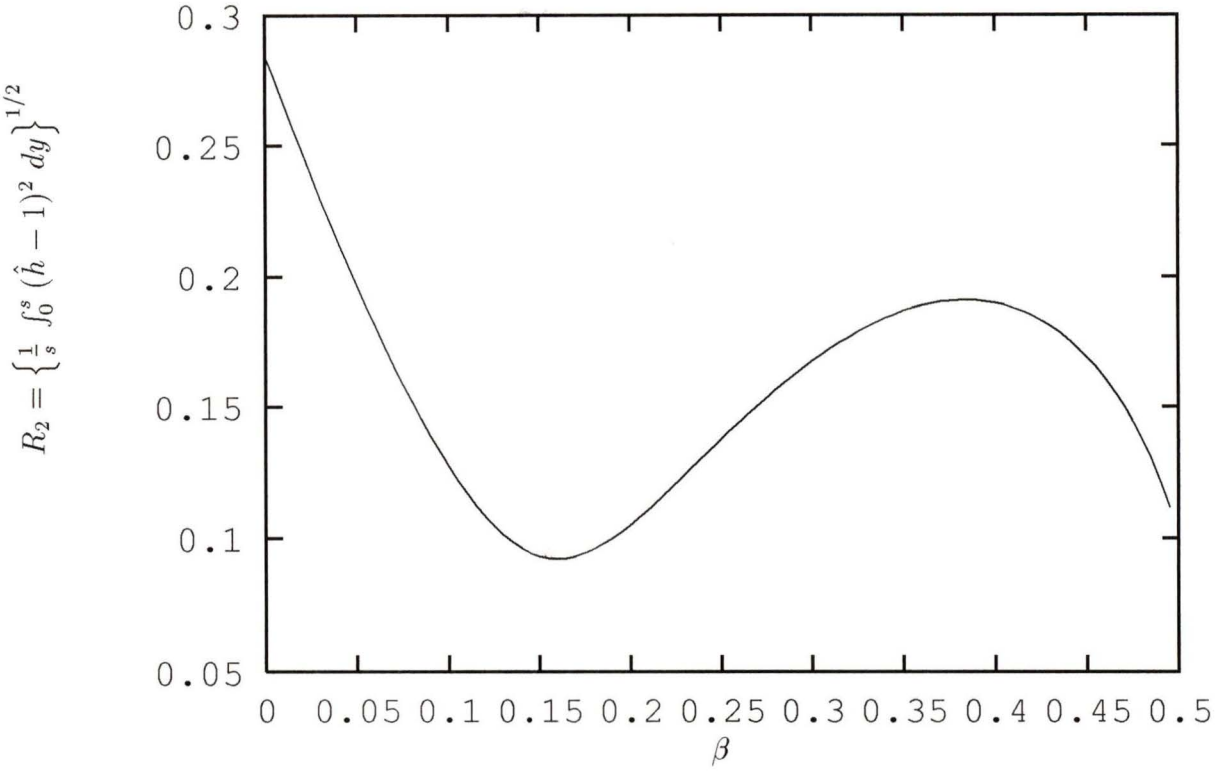

Figure 4.4: RMS deviation as a function of β

4.3.5 Discussion of Overlap Results

Each measure of uniformity described above exhibited change for different values of overlap. Examination of figures 4.3, 4.4 and 4.5 illustrate that optimum values of overlap, as discussed in the previous subsections, exist for each measure. As stated in the introduction to this section, the specific results obtained apply to the case of an elliptical spray pattern, (including the special case of a circular spray pattern) where the paint deposition rate is uniform throughout the pattern.

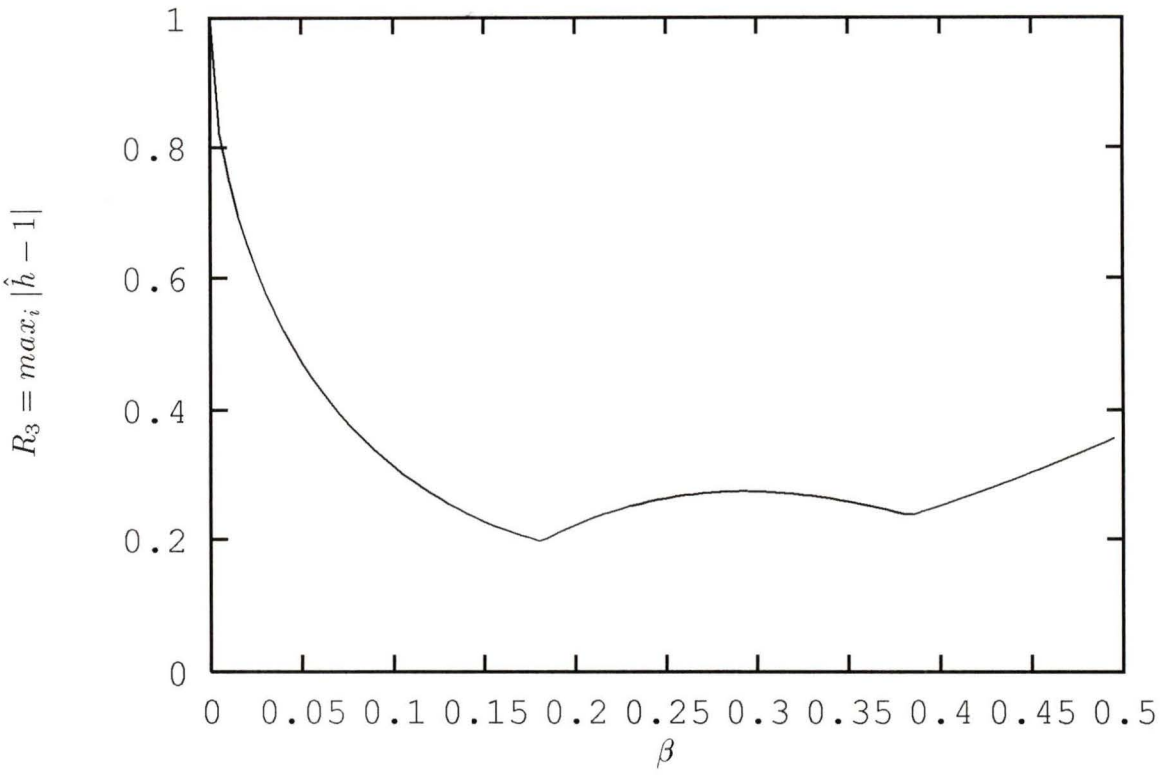

Figure 4.5: Maximum deviation as a function of β

The assumptions of a uniform deposition rate and elliptical spray patterns have been used in spray painting research [24]. Under these assumptions the optimum values of overlap, for each of the three uniformity measurement schemes presented here, apply. However, if the assumption of uniform deposition is unjustified the methods and measures of this section can still be applied. All that is required is a model of the deposition profile.

A model of the deposition profile can be obtained through experimental testing

of the painting hardware under consideration. Obtaining the model requires taking thickness measurements in a plane perpendicular to the velocity of the spray gun. The relationship of the deposition profile to changes in velocity, spray pattern size and flowrate may not be linear so a series of tests over the feasible operating range of the hardware may be required.

Paints remain viscous after application on the work, and can continue to flow. This flow can help improve paint thickness uniformity over time. However, if the coating has a rapid drying time or the workpiece is oriented unfavorably there may be little or no improvement in the thickness uniformity after the paint is applied. Applying the paint as uniformly as is possible reduces the dependence on post application flow.

The optimum for each measure of uniformity in this section are very similar. This, combined with paint's ability to flow, makes the results from each measure practically equivalent.

The optimum overlaps found here are smaller than those often used in industry [25]. Until the limit of maximum feasible flowrate, a lower overlap permits a lower process time.

To summarize the results of this section it can be stated that based upon a known deposition profile it is possible to arrive at a value of β that optimizes the paint thickness uniformity. Once β is found, it can be treated as a constant in the process cost model of equation (4.14). The overlap used in the examples at the end of this chapter is 17.5%, which corresponds to the optimum for the mean deviation objective.

4.4 Selection of Inset

4.4.1 Introduction

Section 3.2.1 provided an introduction to the selection of initial and final pass heights. It was stated that an inset, c, of zero would yield a satisfactory paint path. This section presents two methods of determining non-zero c that decrease process cost.

Increasing the inset reduces the spray by at the top and bottom of the workpiece. It can also decrease process time by lowering the required number of spray passes. Clearly, c cannot be increased without limit. Some minimum paint thickness must be maintained at the edges of the work. Methods one and two in this section offer two choices for that minimum paint thickness.

For a known deposition profile, $h(y)$, a particular choice of c fixes the thickness of paint at the highest point on the work, y_{max}. Specifically, the thickness will be

$$h_{y_{max}} = h(c)$$

Note that if the inset is any larger than the spray pattern major axis, b, the paint thickness will be zero at y_{max}. Therefore the choice of c is restricted to the range

$$0 < c < b$$

A particular choice of c within the feasible range also guarantees that the thickness of paint at y_{min} will be no less than that at the lowest point on the interval profile.

4.4.2 Method One

In method one, the inset, c, is determined such that $h(c) = \overline{h}$ as shown in Figure 4.6.

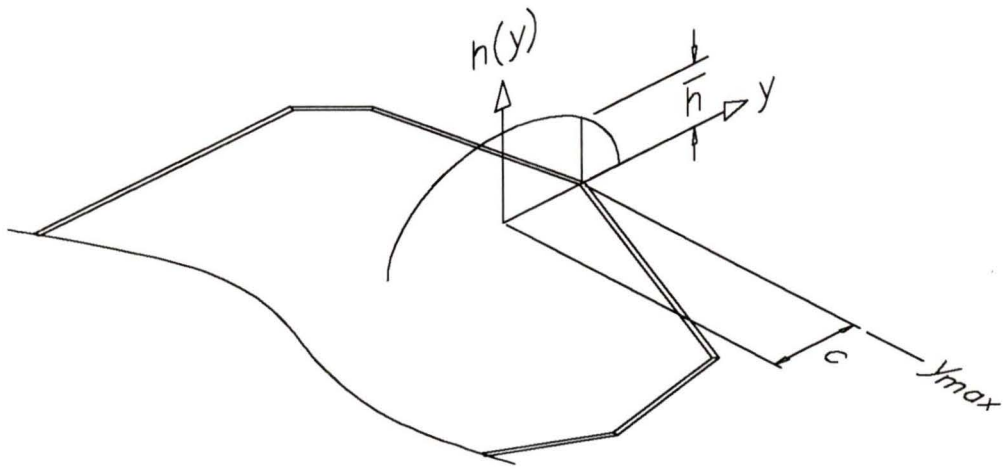

Figure 4.6: Paint thickness at $y_{max} = \overline{h}$

To determine the inset substitute $\overline{h}$ and c in equation (2.11), i.e.,

$$\overline{h} = \frac{2\,q'\,\eta}{v_i\,\pi\,b^2}\,\sqrt{b^2 - c^2}$$

Solving for c gives

$$c = b\,\sqrt{1 - \left(\frac{\overline{h}\,v_i\,\pi\,b}{2\,q'\,\eta}\right)^2} \tag{4.22}$$

Recall equation (2.14), i.e.,

$$\overline{h} = \frac{q'\,\eta}{v_i\,s} = \frac{q'\,\eta}{2\,v_i\,b\,(1-\beta)}$$

Substituting the RHS of (2.14) into (4.22) for $\overline{h}$ and simplifying yields

$$c = b\,\sqrt{1 - \left(\frac{\pi}{4\,(1-\beta)}\right)^2} \tag{4.23}$$

where β is the offset between spray passes.

Note that for c to be real valued, the expression under the radical in (4.23) must be positive, i.e.,

$$1 - \left(\frac{\pi}{4\,(1-\beta)}\right)^2 \geq 0$$

Solving the inequality for β yields

$$\beta \leq 0.215 = 21.5\% \tag{4.24}$$

Equation (4.24) arises from the fact that if $\beta > 21.5\%$, $\overline{h}$ will be greater than the maximum value of the deposition profile, $h(0)$, and the requirement that $h(c) = \overline{h}$ would not be satisfied.

For the optimum overlap of 17.5% (§4.3), the inset can be calculated as

$$\begin{aligned} c &= b\sqrt{1-\left(\frac{\pi}{4\,(1-0.175)}\right)^2} \\ &= b\,(0.3061) \end{aligned} \tag{4.25}$$

Now, for a given workpiece the upper and lower borders can be determined by substituting the RHS of (4.25) for c in equations (3.3) and (3.4).

4.4.3 Method Two

This method selects an inset, c, such that the average paint thickness between y_{max} and y_s (§3.2.1) is equal to $\overline{h}$.

Consider the hatched area under the deposition profile of figure 4.7. The average paint thickness, h_{avg}, over the range

$$0 \leq y \leq c$$

is given by

$$h_{avg} = \frac{1}{c}\int_0^c h(y)\,dy = \frac{1}{c}\int_0^c \frac{2\,q'\,\eta}{v_i\,\pi\,b^2}\sqrt{b^2-y^2} \tag{4.26}$$

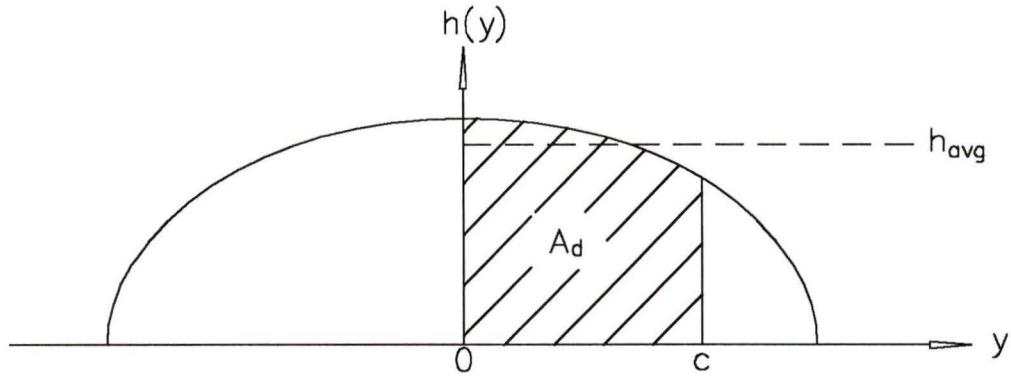

Figure 4.7: Average thickness over range $0 \leq y \leq c$

To determine c such that $h_{avg} = \overline{h}$, equate (4.26) with the expression for $\overline{h}$ given in equation (2.14), i.e.,

$$\frac{q'\,\eta}{2\,v_i\,b\,(1-\beta)} = \frac{1}{c}\int_0^c \frac{2\,q'\,\eta}{v_i\,\pi\,b^2}\sqrt{b^2-y^2}\,dy \tag{4.27}$$

Evaluating the integral and simplifying (4.27) gives

$$\frac{c}{b} - \sin\left(\frac{\pi\,c}{b\,(1-\beta)} - \frac{c\,\sqrt{b^2-c^2}}{b^2}\right) = 0 \tag{4.28}$$

The LHS of equation (4.28) is plotted for $\beta = 0.175$ and $b = 1$ as a function of c in Figure 4.8. Observation of Figure 4.8 shows that equation (4.28) is satisfied for

$$c \approx 0.74\,b \tag{4.29}$$

That is, h_{avg} over the range

$$y_{max} - .74\,b \leq y \leq y_{max}$$

is $\overline{h}$.

With the inset determined, the upper and lower borders can be found from equations (3.3) and (3.4), respectively.

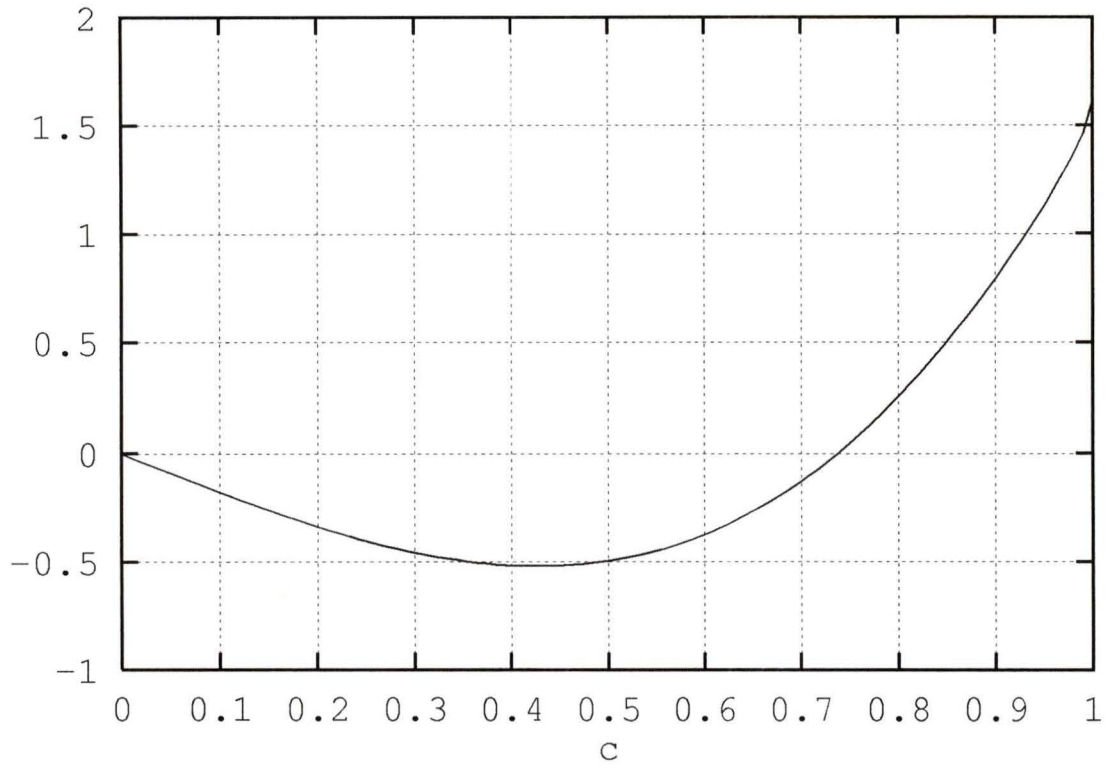

Figure 4.8: LHS of equation (4.28) vs. c

4.4.4 Comparison of Method One and Method Two

Use of method two for determination of the inset, c, will result in a reduction of spray by at the upper and lower workpiece edges. It may also permit fewer spray passes. The choice between the two depends on the amount of paint desired at the edge. Method two may be all that is necessary since it results in a paint thickness of $\overline{h}$ over the entire workpiece. Method one, however, provides a thickness greater than $\overline{h}$ at the upper and lower edges.

4.5 Relationship Between Spray Pattern Size and Process Cost

4.5.1 Introduction

The spray pattern in this work is modelled as an ellipse. The spray pattern size is defined by the length of the ellipse minor and major axis lengths, a and b, respectively. This section discusses the effect of a and b on process cost. The existence of a set of major axis lengths that result in local minima for process cost will be also be illustrated.

4.5.2 Ellipse Minor Axis Length

Despite its absence from the process cost model of equation (4.14), the minor axis length, a, does influence Φ. This influence is through the effect of a on the path length, X_i.

Recall that the point at which each pass segment intersects the offset curve indicates a paint shut off point. Since a is directly involved in the determination of the offset curve (§3.3.3), it follows that a must also affect X_i.

Consider Figure 4.9. Two spray patterns sit at the edge of a workpiece boundary. Clearly the spray pattern with the smaller aspect ratio, Figure 4.9(a), will shut off earlier than that of Figure 4.9(b). In general then, a larger minor axis results in a longer spray path, and consequently, in a higher process cost, Φ. Therefore, a should be set at the smallest value possible.

The spray pattern's minor axis length is physically set with the *fan air* adjustment on the spray gun (§2.6). It will be assumed here that the smallest possible minor axis

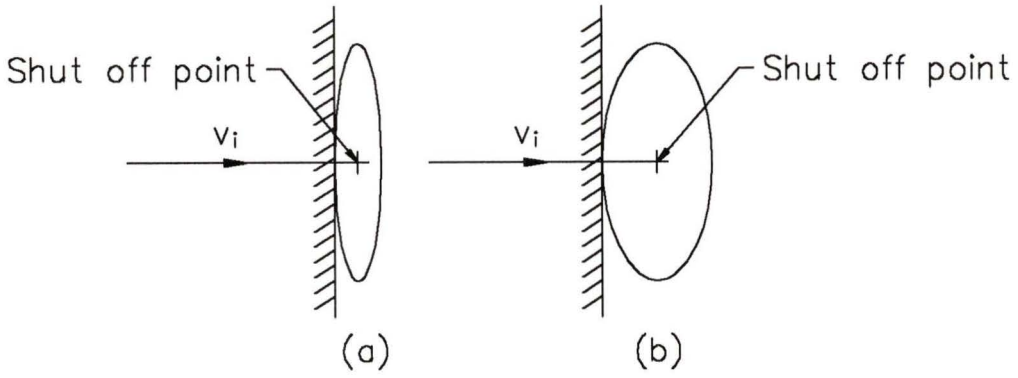

Figure 4.9: Influence of a on X_i

length can be expressed as a constant proportion of the major axis length, b. The proportionality constant is defined as the *aspect ratio*, a_r, and is given by

$$a_r = \frac{a}{b} \tag{4.30}$$

Now a may be determined from

$$a = a_r \, b \tag{4.31}$$

It will be assumed that the minimum aspect ratio is constant over the range of feasible flowrate, q'.

In all further discussion the terms *spray pattern size* and b will be used synonymously, but it should be understood that b and a_r are needed to provide a complete description of the spray pattern size.

4.5.3 Ellipse Major Axis Length

The length of the major axis, b, influences the path length, X_i. Although an explicit relationship is not available for general workpiece geometry, the trend is that for larger major axis length, spray path length will decrease.

Consider the plot of X_i vs. b in Figure 4.10, which is based on a rectangular workpiece. The step pattern of the graph is explained by recognizing that each step is associated with a particular number of spray passes. Note that on any given step the path length increases with major axis size. The increase in X_i with b on the individual steps is a consequence of the associated increase of the minor axis length, i.e., $a = a_r\, b$. As the minor axis increases the spray pattern travels further past the edge of the work before it shuts off. This was shown in Figure 4.9. When b reaches a size such that the workpiece can be coated with one less spray pass, X_i drops sharply.

4.5.4 Preferred Values of Spray Pattern Size

The value of b corresponding to the start of each step in Figure 4.10 results in a local minima for path length. These values of b will be referred to as *preferred values* of spray pattern size. The preferred values of spray pattern size are values of b for which the initial and final pass heights coincide with the upper and lower borders (§3.2.1). To illustrate the concept of preferred b values consider the rectangular workpiece of Figure 4.11.

In Figure 4.11 the spray passes are numbered 1 through 9 and the upper and lower borders are shown as dashed lines. Note that spray pass number 8 is higher than the lower border, and therefore another pass is required. Spray pass 9 extends below the workpiece boundary and the majority of its spray is lost to spray by. To reduce this waste it would be beneficial to either: increase the spray pattern size so that pass number 8 would coincide with y_f; or decrease the spray pattern size so that pass number 9 would coincide with y_f. The values of b in either case are preferred

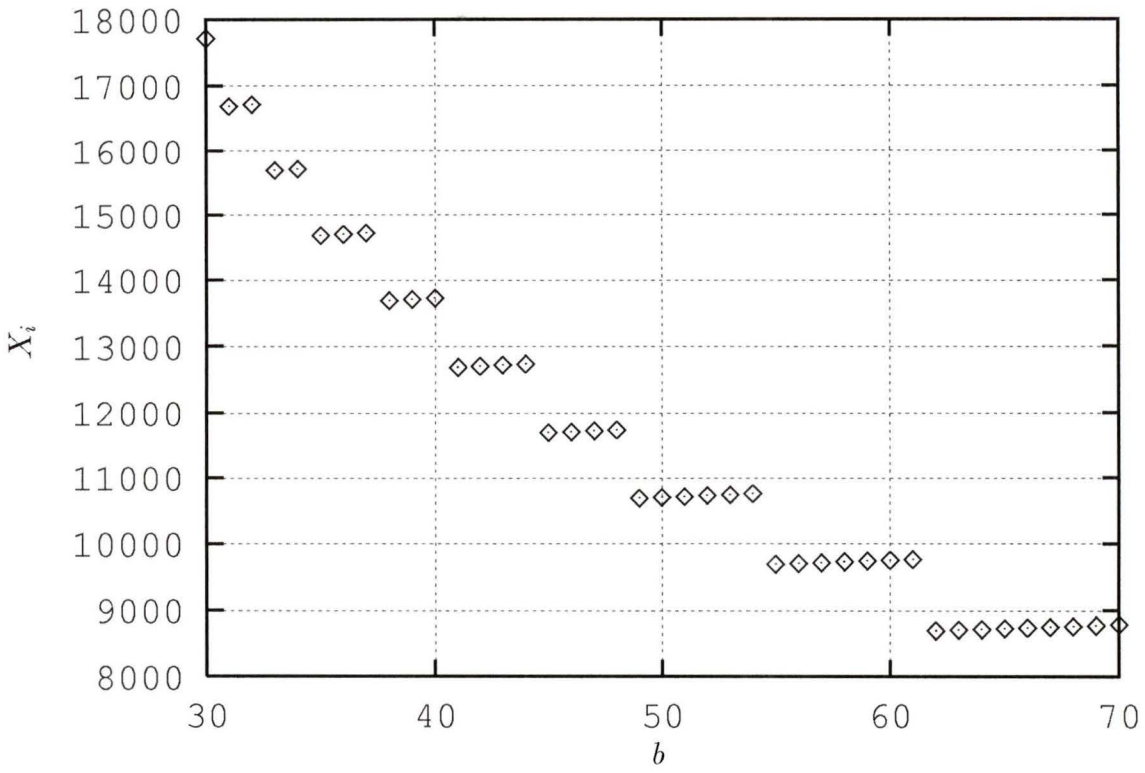

Figure 4.10: Relationship between X_i and b

values of b. To generalize, a preferred value of b is any one in the feasible range

$$b_{min} \leq b \leq b_{max} \tag{4.32}$$

that results in

$$y_1 = y_s$$

$$y_n = y_f$$

Within the feasible range of spray pattern size there are a finite number of pre-

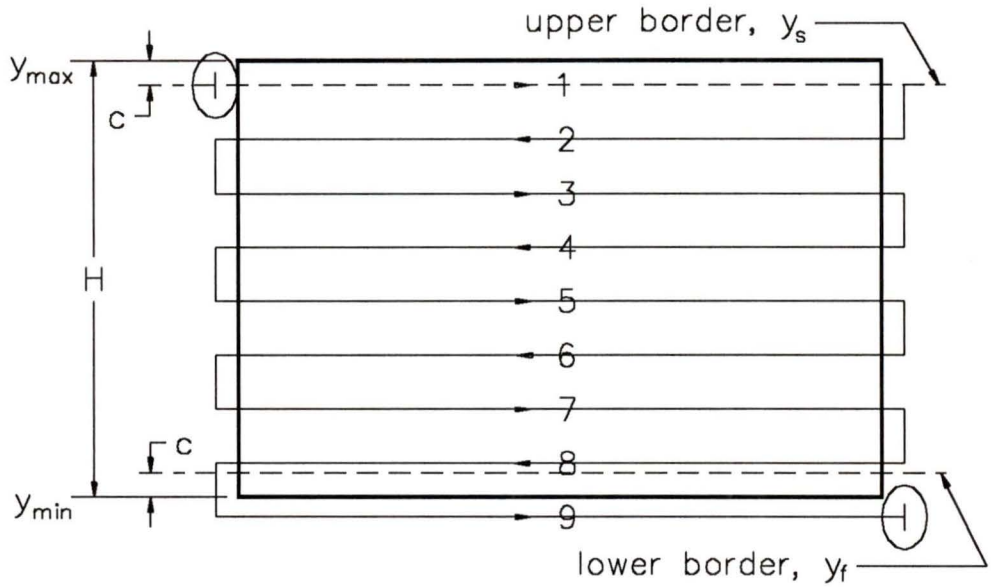

Figure 4.11: Spray passes

ferred values of b. Let $\{b_{pref}\}$ denote the set of preferred values of b. To determine $\{b_{pref}\}$ the relationship between workpiece height, H, and the number of required spray passes, n, must be established. Referring to Figure 4.12 we can write

$$H = (n-1)\, s + 2\, c \tag{4.33}$$

Solving equation (2.3) for s provides

$$s = 2\, b\, (1-\beta) \tag{4.34}$$

and equation (4.29) for c (see §4.4) gives

$$c = 0.74\, b$$

Substituting the results of equations (4.34) and (4.29) into equation (4.33) yields

$$H = (n-1)\, 2\, b\, (1-\beta) + 2\, (0.74)\, b$$

Solving for n provides

$$n = \frac{H - 2\,(0.74)\,b}{2\,b\,(1-\beta)} + 1 \tag{4.35}$$

Preferred b in equation (4.35) result in integral n. The range of feasible n is determined by substituting the bounding values of b from equation (4.32). The maximum number of passes, n_{max}, occurs when $b = b_{min}$

$$n_{max} = \frac{H - 2\,(0.74)\,b_{min}}{2\,b_{min}\,(1-\beta)} + 1 \tag{4.36}$$

The result of (4.36) must be rounded *down* to the next whole number. Similarly, n_{min} is obtained from

$$n_{min} = \frac{H - 2\,(0.74)\,b_{max}}{2\,b_{max}\,(1-\beta)} + 1 \tag{4.37}$$

The result of (4.37) must be rounded *up* to the next whole number. Equation (4.35) can be solved for b, i.e.,

$$b = \frac{H}{2\,[(n-1)\,(1-\beta) + 2\,(0.74)]} \tag{4.38}$$

allowing $\{b_{pref}\}$ to be determined by substitution of

$$n = n_{min}, n_{min} + 1, \ldots, n_{max}$$

into equation (4.38).

Example-Determination of Preferred Values of Spray Pattern Size

Consider a rectangular workpiece with

$$\begin{aligned} H &= 1000 \\ \beta &= 0.175 \\ 40.0 \leq\ & b \leq 60.0 \end{aligned}$$

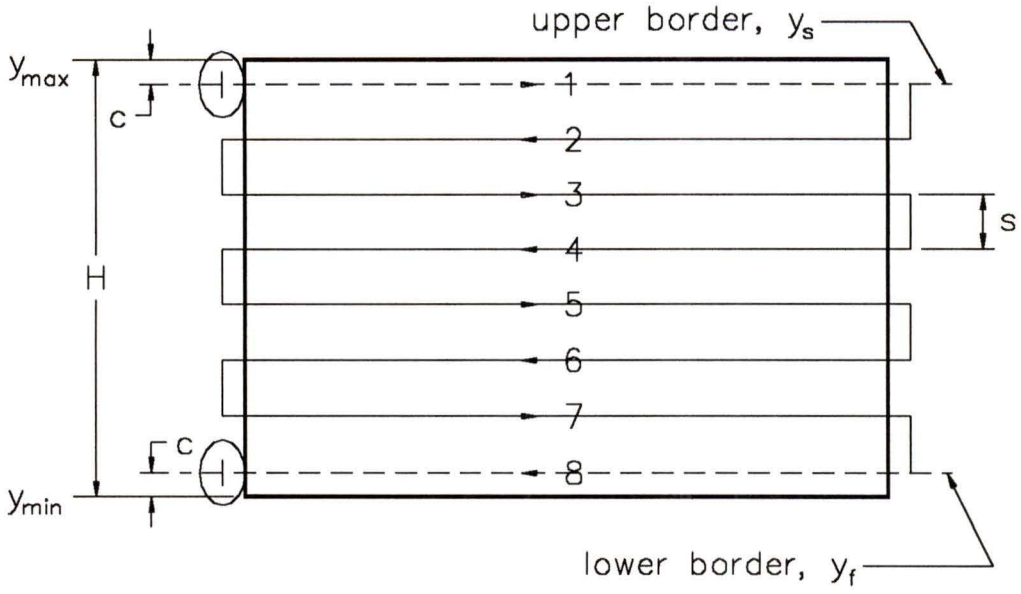

Figure 4.12: Determination of the preferred values of b

The minimum and maximum number of spray passes are obtained from (4.37) and (4.36), i.e.,

$$n_{min} = \frac{H - 2\,(0.74)\,b_{max}}{2\,b_{max}\,(1-\beta)} + 1 = \frac{1000 - 2\,(0.74)(60)}{2\,(60)(1-0.175)} + 1 = 10.2$$

$$n_{max} = \frac{H - 2\,(0.74)\,b_{min}}{2\,b_{min}\,(1-\beta)} + 1 = \frac{1000 - 2\,(0.74)(40.0)}{2\,(40.0)(1-0.175)} + 1 = 15.3$$

After rounding n_{min} and n_{max} become

$$n_{min} = 11$$

$$n_{max} = 15$$

and evaluation of b from (4.38) for

$$n = 11, 12, \ldots, 15$$

yields

$$\{b_{pref}\} = \{55.6\ 50.9\ 47.0\ 43.6\ 40.7\}$$

4.6 Relationship of Spray Pass Orientation and Process Cost

4.6.1 Introduction

The relationship between process cost and spray pass orientation is generally unknown. Finding the minimum cost over a range of θ usually requires an exhaustive search. However, for a rectangular workpiece the search only requires checking spray pass angles parallel to its sides. The following discussion illustrates the process cost characteristics as a function of spray pass orientation for an irregularly shaped workpiece, and a rectangular workpiece.

4.6.2 Irregular Shaped Workpiece

An irregularly shaped workpiece is shown in Figure 4.13. The process cost was evaluated for spray pass angles in the range

$$0 \leq \theta \leq 180$$

in 2 degree increments and using a constant spray pattern size of $b = 60\text{mm}$. The results are illustrated in the graph of Figure 4.14. The erratic nature of Figure 4.14 suggests the difficulty of predicting which orientation would result in the least process cost, Φ, without an exhaustive search approach.

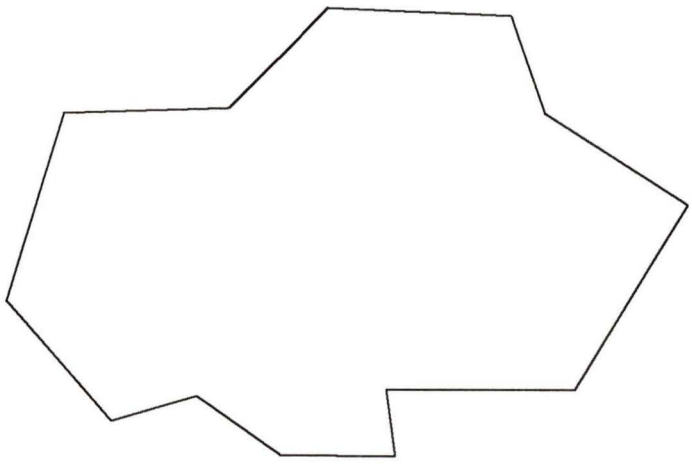

Figure 4.13: Irregular shaped workpiece

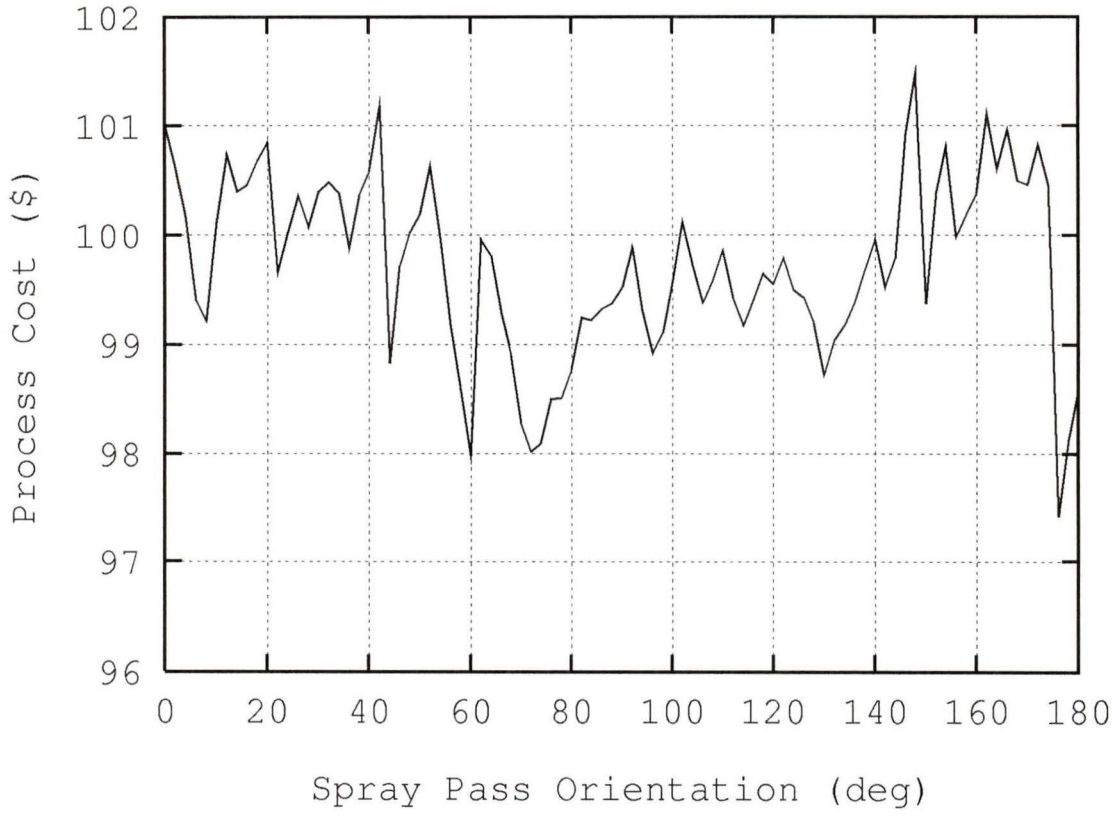

Figure 4.14: Process cost as a function of spray pass orientation

4.6.3 Rectangular Workpiece

The process cost for the rectangular workpiece of Figure 4.15 was evaluated for

$$0 \leq \theta \leq 180^o$$

with $b = 60$mm. The results are plotted in Figure 4.16.

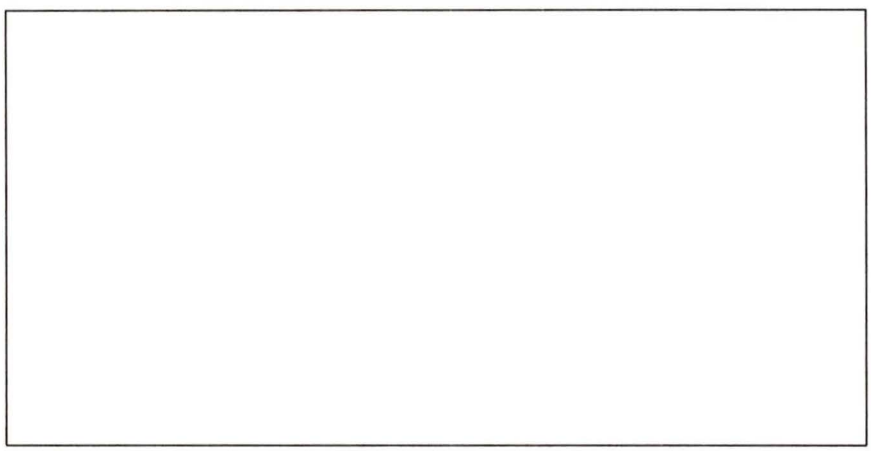

Figure 4.15: Regularly shaped workpiece

The graph is symmetrical about $\theta = 90^o$ and has local minima at $\theta = 0^o$, 90^o, and 180^o. The local minima correspond to the spray pass orientation being parallel with the sides of the workpiece. This result can be extended for rectangular workpieces in general; the minimum process cost for rectangular workpieces occurs when $\theta = 0^o$ or $\theta = 90^o$.

To illustrate why Φ is minimized when $\theta = 0^o$ or 90^o consider Figure 4.17. In Figure 4.17(a) $\theta = 0$. Increasing θ as shown in Figure 4.17(b) increases the quantity of paint sprayed by the edge of the work and increases the path length, X_i. The result is an increase in process cost, Φ.

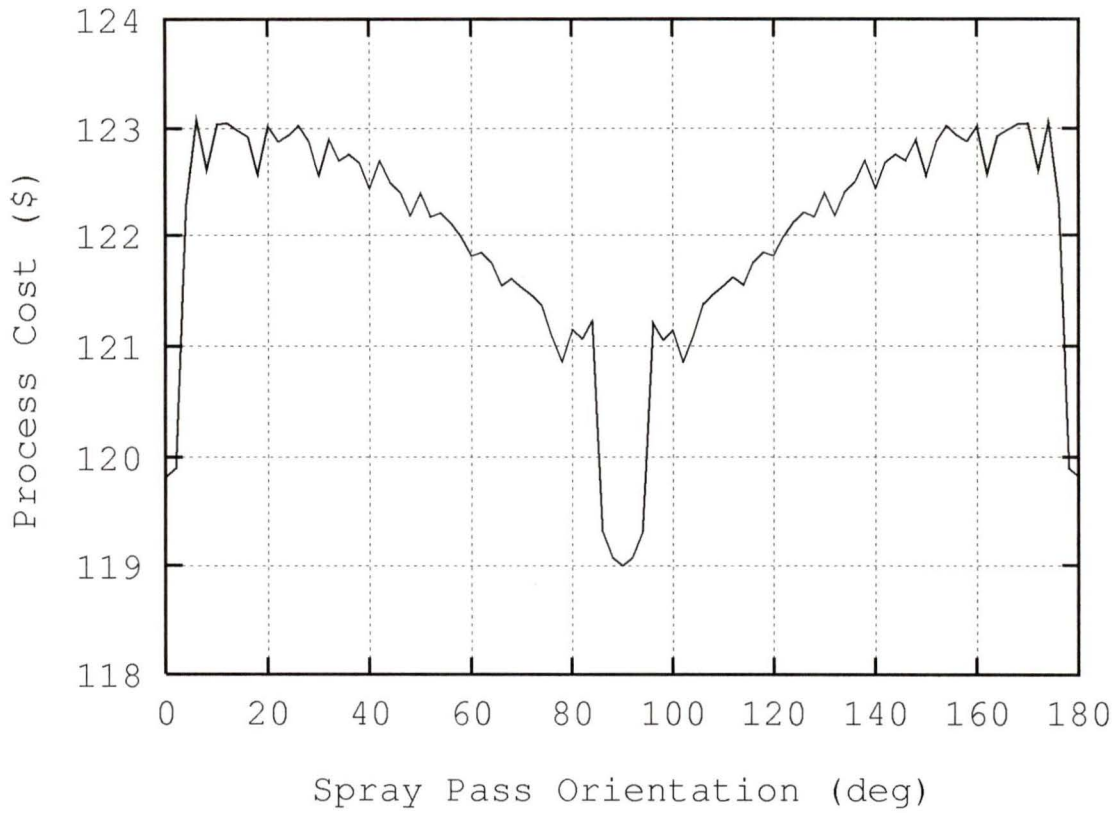

Figure 4.16: Process cost as a function of spray pass orientation

The search for the spray pass orientation yielding the lowest process cost for a rectangular workpiece need only check the two angles parallel to each workpiece edge.

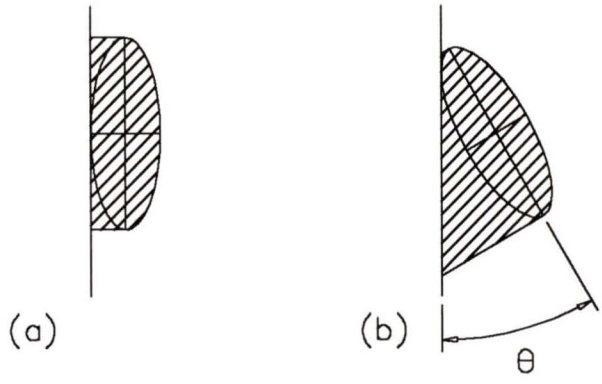

Figure 4.17: Spray by with increase in θ

4.7 Program Description

The algorithms presented in this work have been implemented in the C programming language and run within the AutoCAD environment. The source code for the algorithms is presented in appendix B.

4.7.1 Program Overview

The first step in determining the optimum spray path is to define the workpiece geometry using the standard AutoCAD line command. Once the workpiece geometry is defined the presented algorithms are invoked by typing *paint* at the AutoCAD command prompt. The program first prompts the user to indicate any line segment on the workpiece geometry. It then searches for all connecting segments in the AutoCAD drawing database and organizes the vertex coordinates into a file.

With the workpiece model determined the program evaluates the spray path and process cost at desired increments over the feasible range of spray pattern size and spray pass orientation. For each iteration the process cost, velocity, spray pass angle and other spraying parameters are written to a results file. Once the search is complete the results file is examined for the minimum process cost. Using the spraying parameters associated with the minimum process cost the spray path is recalculated and superimposed on the workpiece geometry.

4.7.2 Program Flow

The program follows the sequence enumerated below.

1. Input spraying parameters from user specified file

   ($\overline{h}$, θ_s, θ_f, $\Delta\theta$, $v_{i_{max}}$, q'_{max}, η, b_{min}, b_{max}, Δb, a_r, β, k_1, k_2, k_3, k_4)

2. Prompt user to indicate segment of workpiece boundary.

3. Create file describing workpiece boundary

4. Calculate workpiece area, A_s.

5. Set $b = b_{min}$

6. $b \leq b_{max}$?

   Y $\Rightarrow$ goto line 7

   N $\Rightarrow$ goto line 25

7. Calculate path interval: $s = 2\,b(1 - \beta)$

8. Calculate ellipse minor axis length: $a = a_r\,b$

9. Set $\theta = \theta_s$

10. $\theta \leq \theta_f$?

    Y $\Rightarrow$ goto line 11

    N $\Rightarrow$ goto line 23

11. Rotate workpiece by angle $(-\theta)$

12. Calculate offset curve for rotated workpiece

13. Determine spray path lengths, X_i and X_o

14. Calculate q' based upon maximum spray gun velocity, $v_{i_{max}}$

15. $q' = (\overline{h}\, v_{i_{max}}\, s)/\eta$

16. $q' > q'_{max}$?

    Y $\Rightarrow$ goto line 17

    N $\Rightarrow$ goto line 19

17. $q' = q'_{max}$

18. $v = q'_{max}\, \eta/(\overline{h}s)$

19. Calculate process costs: (Φ_{T_i}, Φ_{T_o}, Φ_{os}, Φ_{sb}, Φ_w)

20. Write costs and associated spraying parameters to results file

21. $\theta = \theta + \Delta\theta$

22. Goto line 10

23. $b = b + \Delta b$

24. Goto line 6

25. Search results file for minimum Φ.

26. Calculate s and a corresponding with minimum Φ.

27. Rotate work by angle $-\theta_{opt}$

28. Calculate offset curve corresponding to θ_{opt} and b_{opt}.

29. Generate optimum spray path.

30. Rotate optimum spray path by θ_{opt}.

31. Display resulting spray path, (superimpose over workpiece geometry).

4.8 Examples

4.8.1 Example 1 - A Rectangular Workpiece

Consider a rectangular workpiece with face dimensions of 1000mm × 800mm as shown in Figure 4.18. For an initial trial the spray path is optimized with respect to process

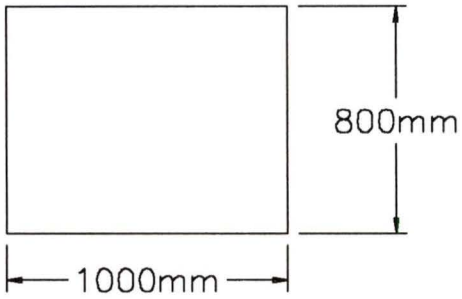

Figure 4.18: Rectangular workpiece

cost using the parameters shown in table 4.1. Note that the input values for spray painting parameters vary widely depending on the application. The values used in this and the following examples were estimated based on painting hardware capabilities [7], and discussion with personnel in the painting industry [25].

Note that since the workpiece is rectangular not all pass orientation angles have to be checked. As indicated in section 4.6.3 the minimum process cost for a rectangular workpiece occurs when the spray pass orientation is parallel to either the bottom or side of the rectangle. Therefore a comparison of the costs with $\theta = 0$ and $\theta = 90^o$ will ensure the global minimum will be found.

An exhaustive over the range of feasible spray pattern size, b, i.e.,

$$20.0 = b_{min} \leq b \leq b_{max} = 75.0mm$$

Item	Variable description	Symbol	Units	Value
1	specified paint thickness	$\overline{h}$	mm	0.051
2	starting angle	θ_s	degrees	0.0
3	ending angle	θ_f	degrees	90.0
4	angle increment	$\Delta\theta$	degrees	90.0
5	maximum velocity	v_{max}	mm/s	1000.0
6	maximum total flowrate	q'_{max}	ℓ/min	1.0
7	transfer efficiency	η	dimensionless	0.65
8	minimum ellipse height	b_{min}	mm	20.0
9	maximum ellipse height	b_{max}	mm	75.0
10	ellipse increment size	Δb	mm	2.0
11	minimum aspect ratio	a_r	dimensionless	0.7
12	overlap	β	dimensionless	0.175
13	time cost	k_1	dollars/hr	60.0
14	material cost	k_2	dollars/ℓ	10.00
15	disposal cost (overspray)	k_3	dollars/ℓ	2.0
16	disposal cost (spray by)	k_4	dollars/ℓ	2.0

Table 4.1: Input values-example 1

Variable	Φ_{min}	b	θ	v_i	q'
Result	\$.9127	62.0 mm	0^o	1000 mm/s	0.481 ℓ/min

Table 4.2: Results-example 1

was performed using the increment $\Delta b = 2.0$ mm. The minimum process cost was found to occur at $\theta = 0^o$ and $b = 62$mm. Table 4.2 summarizes the cost, paint flow rate, and gun velocity associated with the optimal b and θ. The resulting optimal spray path is illustrated in Figure 4.19 and a graph illustrating the process cost as a function of b for $\theta = 0^o$ is shown in Figure 4.20.

The search algorithm used in this optimization does not determine process cost specifically for preferred spray pattern sizes. Instead it steps through the feasible range of b using the specified increment value. The following discussion considers process cost for preferred spray pattern sizes and compares the result with that found using the exhaustive search.

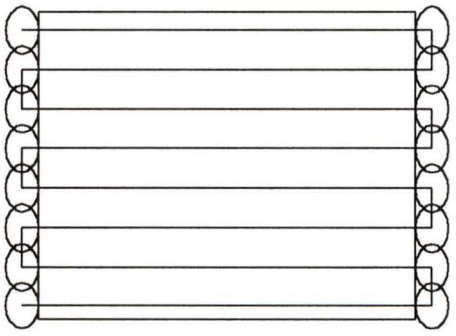

Figure 4.19: Spray path for rectangular workpiece

Preferred Spray Pattern Sizes

As discussed in section 4.5.4, there exists a set of preferred values of spray pattern size, $\{b_{pref}\}$, for which Φ achieves local minima. Several local minima are evident in Figure 4.20, however it would only be chance that they correspond precisely with $\{b_{pref}\}$. The increment for spray pattern size in the search was 2mm. The search algorithm therefore determined Φ for

$$b = b_{min}, b_{min} + 2, b_{min} + 4, \ldots, b \leq b_{max}$$

which does not guarantee evaluation precisely at any element of $\{b_{pref}\}$.

For comparative purposes Φ will be determined for the preferred values of b. The first step in determining $\{b_{pref}\}$ is to determine the range of integral passes, which is defined by n_{min} and n_{max}. Recalling equation(4.36)

$$n_{max} = \frac{H - 2\,(0.74)\,b_{min}}{2\,b_{min}\,(1-\beta)} + 1 = \frac{800 - 2\,(0.74)\,(20.0)}{2\,(20.0)\,(1-0.175)} + 1 = 23.35 \tag{4.39}$$

and

$$n_{min} = \frac{H - 2\,(0.74)\,b_{max}}{2\,b_{max}\,(1-\beta)} + 1 \frac{800 - 2\,(0.74)\,(75.0)}{2\,(75.0)\,(1-.175)} + 1 = 5.57 \tag{4.40}$$

Now n_{min} must be rounded up and n_{max} rounded down so that

$$\begin{aligned} n_{min} &= 6 \\ n_{max} &= 23 \end{aligned}$$

Now $\{b_{pref}\}$ is obtained by evaluating b for

$$n = n_{min}, n_{min} + 1, \ldots, n_{max} = 6, 7, \ldots, 23$$

using equation (4.38).

$$b = \frac{H}{2[(n-1)\,(1-\beta) + 2\,(0.74)]}$$

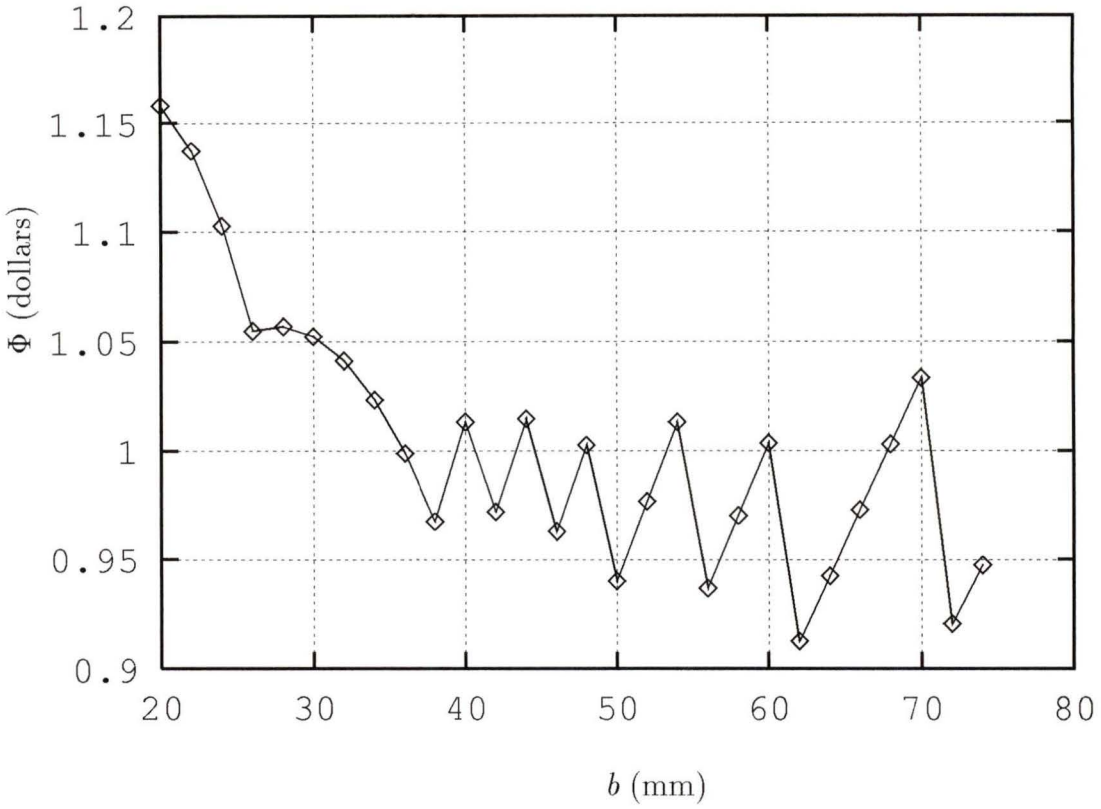

Figure 4.20: Variation of Φ with b

The preferred values of b and the resulting values of process cost, Φ, are provided in table 4.3. The lowest cost found using the preferred values of spray pattern size was Φ = \$.9126, corresponding to a value b_{pref} = 71.4mm. This represents the global minimum for process cost and is only a reduction of \$.0001 from Φ_{min} found in the original search.

The reduction in process cost available through use of $\{b_{pref}\}$ depends upon the value of Δb used in the search. Small Δb increases the likelihood of finding Φ close to

n	6	7	8	9	10	11	12	13	14
b_{pref}	71.4	62.2	55.1	49.5	44.9	41.1	37.9	35.1	32.8
Φ	.9126	.9157	.9219	.9312	.9411	.9526	.9651	.9767	.9919
n	15	16	17	18	19	20	21	22	23
b_{pref}	30.7	28.9	27.2	25.8	24.5	23.3	22.2	21.3	20.4
Φ	1.005	1.020	1.032	1.048	1.063	1.078	1.092	1.110	1.125

Table 4.3: Preferred spray pattern sizes and associated Φ

Item	Variable description	Symbol	Units	Value
1	time cost	k_1	\$/hr	20.0
2	material cost	k_2	$\$/\ell$	25.00
3	disposal cost (overspray)	k_3	$\$/\ell$	15.0
4	disposal cost (spray by)	k_4	$\$/\ell$	15.0

Table 4.4: Revised cost constants-example 1

the minimum available with $\{b_{pref}\}$. Utilizing the values of $\{b_{pref}\}$ greatly reduces the number of necessary evaluations. However, for irregular object shapes $\{b_{pref}\}$ cannot be found as in this example of a rectangular workpiece. Furthermore, it should be noted that setting and maintaining the spray pattern size at precise $\{b_{pref}\}$ values may not be feasible in practice.

Material Costs vs. Time Costs

Observation of Figure 4.20 shows for the example workpiece and the cost constants of table 4.4, that as the spray pattern size increases, the general trend for Φ is to decrease. To illustrate the influence of time and material costs on the overall process cost, the optimization was rerun with the revised constants shown in table 4.4. Note that the material cost constant, k_2, and the disposal cost constants, k_3 and k_4 have

been increased, while the time cost constant, k_1 has been decreased.

The plot of Φ vs. b for the revised data is shown in Figure 4.21. The graph of

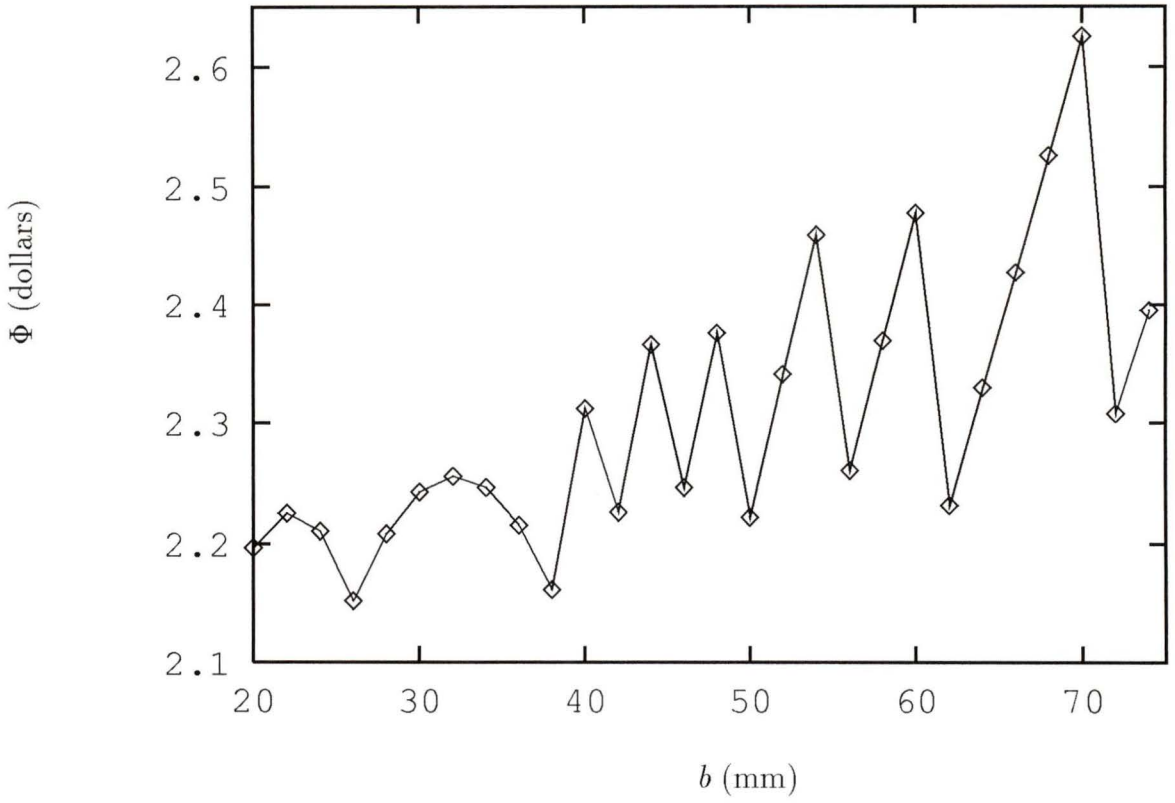

Figure 4.21: $\Phi = f(b)$

Figure 4.21 illustrates that for increasing spray pattern size, the process cost now rises, which is the opposite to the situation of Figure 4.20. Increasing the material and disposal cost constants in equation (4.14) increases the cost of overspray and spray by. This has the effect of reducing b, since both overspray and spray by costs are reduced with a smaller spray pattern. Therefore, depending on the relative sizes

of the cost constants the overall trend of the slope of $\Phi = f(b)$ can be change from positive to negative.

4.8.2 Example 2 - An Irregular Shaped Workpiece

As a second example consider the workpiece of Figure 4.22. The search for minimum

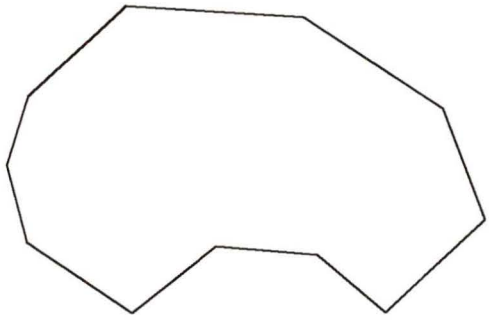

Figure 4.22: Irregular shaped workpiece

process cost will be performed over the range

$$0 \leq \theta \leq 360^o$$

The spray pattern size limits are taken to be

$$40 \leq b \leq 70$$

The remaining input parameters for the optimization are provided in table 4.5.

The optimal process cost occurs at $b = 44$ mm and $\theta = 45^o$. The corresponding flowrate, velocity, and process cost are provided in table 4.6. The resulting optimal spray path is shown in Figure 4.23. Figure 4.24 illustrates that the process cost as a function of spray pass orientation and spray pattern size. The erratic nature of Figure 4.24 indicates the difficulty in predicting optimal process cost.

Item	Variable description	Symbol	Units	Value
1	specified paint thickness	$\overline{h}$	mm	0.051
2	starting angle	θ_s	degrees	0.0
3	ending angle	θ_f	degrees	360.0
4	angle increment	$\Delta\theta$	degrees	5.0
5	maximum velocity	v_{max}	mm/s	1000.0
6	maximum total flowrate	q'_{max}	ℓ/min	1.0
7	transfer efficiency	η	dimensionless	0.65
8	minimum ellipse height	b_{min}	mm	40.0
9	maximum ellipse height	b_{max}	mm	70.0
10	ellipse increment size	Δb	mm	2.0
11	minimum aspect ratio	a_r	dimensionless	0.7
12	overlap	β	dimensionless	0.175
13	time cost	k_1	dollars/hr	100.00
14	material cost	k_2	dollars/ℓ	50.0
15	disposal cost (overspray)	k_3	dollars/ℓ	10.0
16	disposal cost (spray by)	k_4	dollars/ℓ	10.0

Table 4.5: Input values-example 2

Variable	Φ_{min}	b	θ	v_i	q'
Result	\$2.455	44.0 mm	45^o	1000 mm/s	0.3418 ℓ/min

Table 4.6: Results-example 2

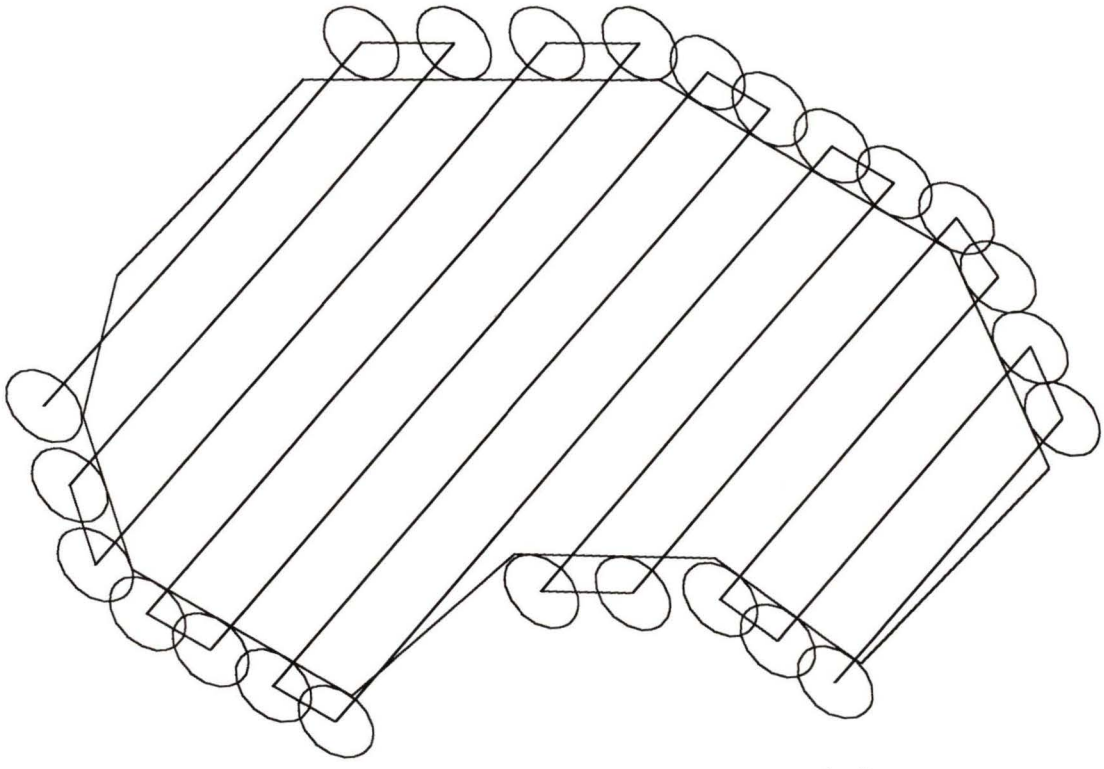

Figure 4.23: Optimal spray path-example 2

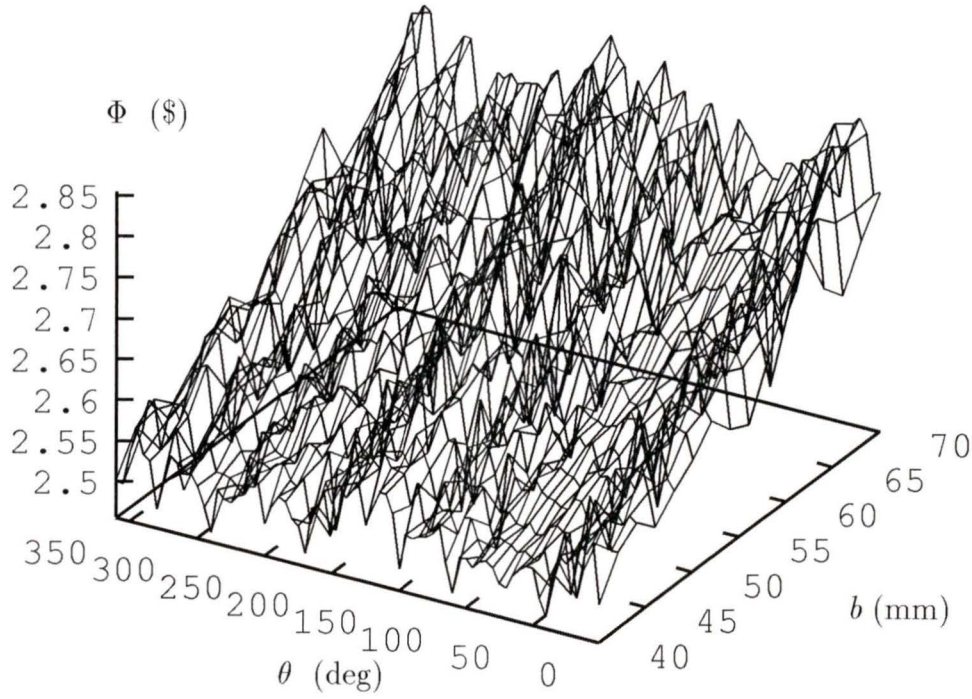

Figure 4.24: $\Phi = f(b, \theta)$

4.8.3 Example 3 - A Hexagonal Workpiece

As a third example consider the hexagonal workpiece of Figure 4.25.

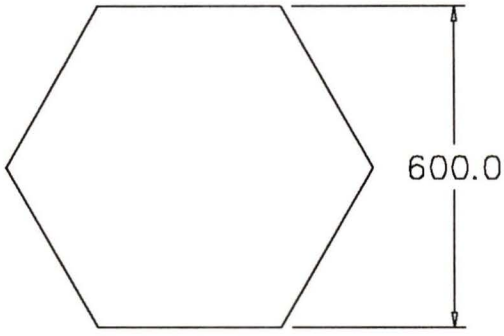

Figure 4.25: Polygon workpiece

Since the workpiece is hexagonal it is expected that the process cost results will repeat through every 60^o of spray pass orientation. To determine if this is the case a preliminary evaluation of process cost was performed over the range

$$0 \leq \theta \leq 180^o$$

with $b = 60$mm. The results are illustrated in Figure 4.26. Figure 4.26 clearly shows that the process cost does repeat through every 60^o of spray pass orientation.

The search for minimum process cost can now be confined to the range

$$0 \leq \theta \leq 60^o$$

The spray pattern size limits are taken to be

$$50 \leq b \leq 75$$

The remaining input parameters for the optimization are provided in table 4.7.

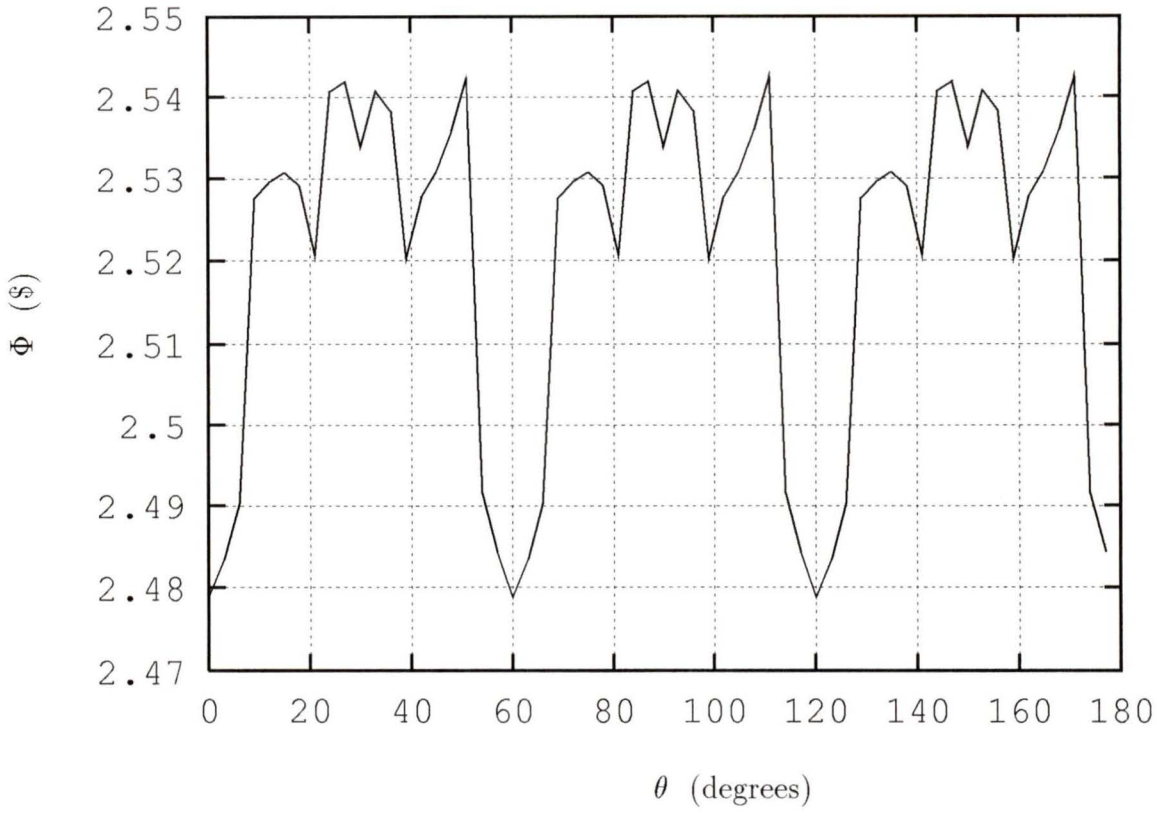

Figure 4.26: Process cost, Φ, as a function of spray pass angle, θ

The optimal process cost occurs at $b = 74$ and $\theta = 0$. The corresponding flowrate, velocity, and process cost are provided in table 4.8. The resulting optimal spray path is shown in Figure 4.27. Figure 4.28 illustrates that the process cost tends to decrease with increasing spray pattern size, and when close to $\theta = 0^o$ and 60^o.

Figure 4.28 shows that even for small changes in b and θ a cost variation of \$0.05 can occur. In high volume manufacturing this cost could be considered very significant.

Item	Variable description	Symbol	Units	Value
1	specified paint thickness	$\overline{h}$	mm	0.051
2	starting angle	θ_s	degrees	0.0
3	ending angle	θ_f	degrees	60.0
4	angle increment	$\Delta\theta$	degrees	1.0
5	maximum velocity	v_{max}	mm/s	500.0
6	maximum total flowrate	q'_{max}	ℓ/min	0.5
7	transfer efficiency	η	dimensionless	0.65
8	minimum ellipse height	b_{min}	mm	50.0
9	maximum ellipse height	b_{max}	mm	75.0
10	ellipse increment size	Δb	mm	2.0
11	minimum aspect ratio	a_r	dimensionless	0.7
12	overlap	β	dimensionless	0.175
13	time cost	k_1	dollars/hr	60.00
14	material cost	k_2	dollars/ℓ	15.0
15	disposal cost (overspray)	k_3	dollars/ℓ	5.0
16	disposal cost (spray by)	k_4	dollars/ℓ	5.0

Table 4.7: Input values-example 3

Variable	Φ_{min}	b	θ	v_i	q'
Result	\$2.350	74.0 mm	0^o	500 mm/s	0.287 ℓ/min

Table 4.8: Results-example 3

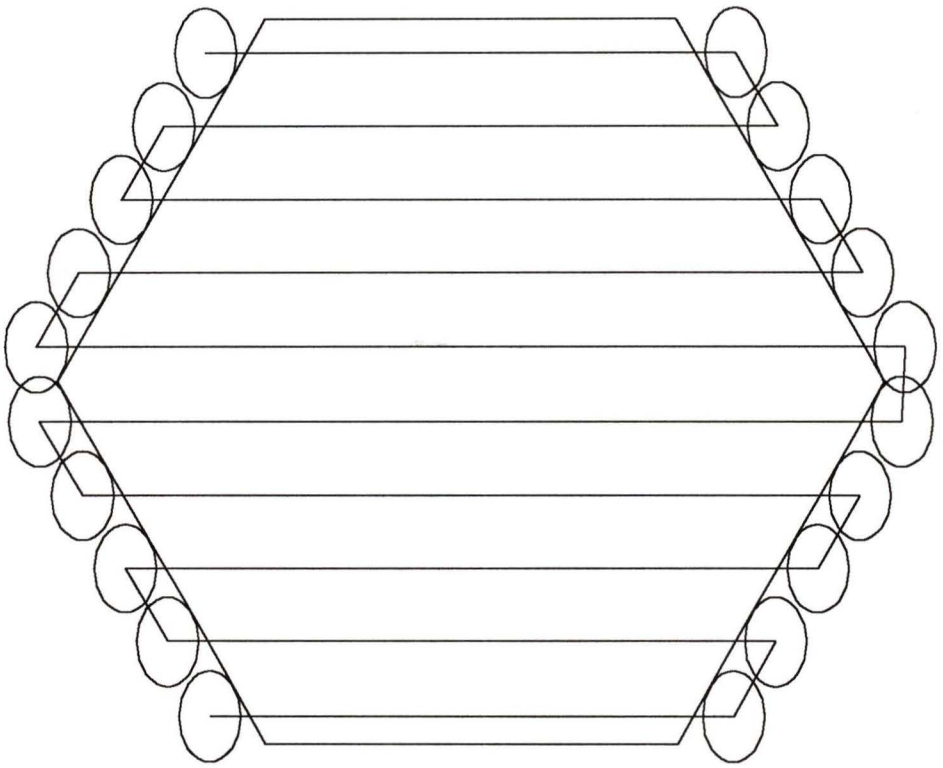

Figure 4.27: Optimal spray path-example 3

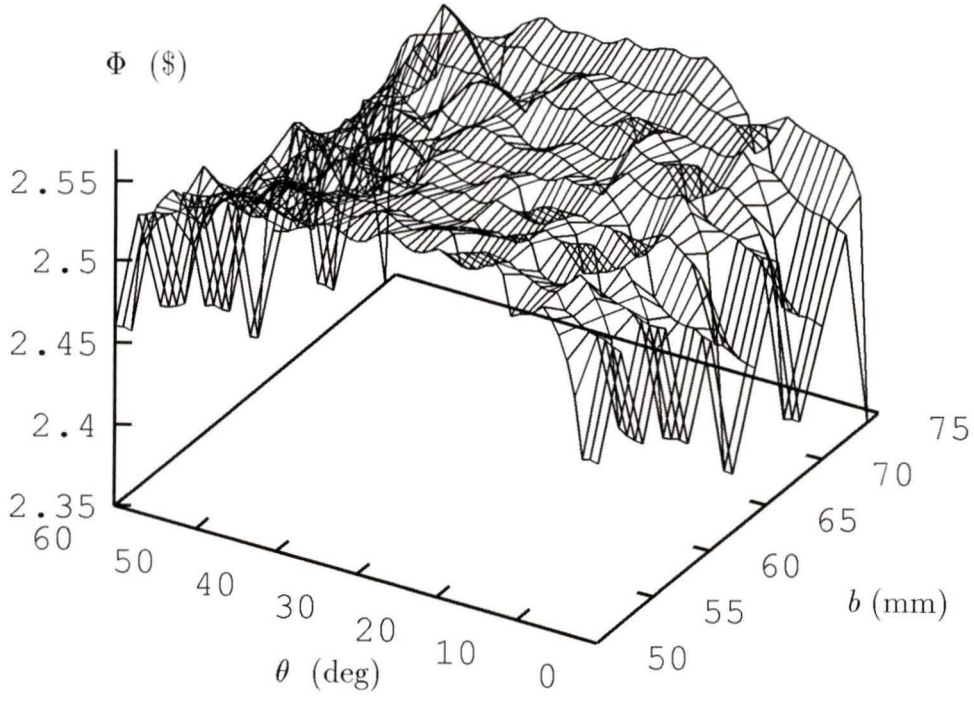

Figure 4.28: $\Phi = f(b, \theta)$

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

An algorithm was developed for automatic spray path generation for planar workpieces. The algorithm modelled the spray pattern as an ellipse with uniform distribution of paint over its area.

It was shown (§4.3) that the uniformity of paint thickness over the workpiece is controlled by the degree of overlap between consecutive spray passes. Three measures for the variation in paint thickness uniformity were presented. The measures were the average arithmetic deviation from the mean, the root mean square deviation from the mean and the maximum deviation from the mean. The optimum uniformity was concluded to range between 16-18% depending upon the applied measure.

A process cost model was developed (§4.2) that considered paint costs, labour and maintenance costs, and waste paint disposal costs. The following conclusions can be made regarding the minimization of the process cost model.

1. The spray pattern is determined by the length of its major and minor axis

lengths. Since a larger minor axis length increases path length, the minimum process cost will occur for the smallest possible minor axis length.

2. For rectangular workpiece geometry there exists a finite set of spray pattern sizes that produce local minima in the cost function. These spray pattern sizes result in equal inset values on the top and bottom of the workpiece and thus the top and bottom passes provide only the paint required to coat the workpiece to the specified thickness, and no more. The global optimal process cost can be determined by comparison of the local minima.

3. For rectangular workpiece geometry the minimum process cost will occur when the spray passes are aligned with either the horizontal or vertical sides of the rectangle. Other orientations will result in increased path length, without any other benefit. The search for the minimum need not consider other spray pass orientations.

4. For general workpiece geometry there is no explicit expression of path length as a function of spray pattern size and spray pass orientation. Therefore the minimization of process cost requires an exhaustive search over the feasible range of spray pattern size and spray pass orientation. The size of variable increments in the search should be limited to the tolerance that can be practically set and maintained on the physical equipment being used.

5. The amount of paint used tends to decrease as spray pattern size is decreased.

6. Process time tends to decrease with increasing spray pattern size.

5.2 Recommendations for Further Research

1. Optimization of spray pattern overlap for other spray pattern paint distribution models using the introduced uniformity measures could be considered. Experimental tests could be conducted to validate and improve if necessary the paint distribution models.

2. An investigation of the relationship of transfer efficiency with paint flowrate and gun velocity could provide an improvement in accuracy of the process cost model.

3. Development of alternative spray painting strategies could provide further process cost minimization. For example, a preliminary spray pass circumscribing the workpiece boundary, followed by parallel passes in interior region could reduce the paint lost to spray by.

References

[1] Adler, A., "Simulation and Off Line Programming of Robots", *Tecnomatix GmbH*, Federal Republic of Germany.

[2] "Autodesk 1991 Annual Report", Autodesk Inc., Sausalito, CA, USA, 1991.

[3] "The Autocad Resource Guide", summer-fall 1992, *Autodesk Inc.*, Sausalito, CA, 1991.

[4] Bidanda, B., Rubinovitz, J., Narayanan, V., "Automatic CAD-Based Off-Line Programming of a Glaze Spraying Robot", SME Technical Paper MS91-335, *Fourth World Conference on Robotics Research*, Pittsburgh, Sept 17-19, 1991, 9 pages.

[5] Bishop, W., "Experience with a Rail Car Painting System for CN Rail", SME Technical Paper MS89-151, *Robots in Aerospace Manufacturing conference*, Irvine, California, 1989, 12 pages.

[6] "Management Plan for Nitrogen Oxides and Volatile Organic Compounds", *Canadian Council of Ministers of the Environment*, (ISBN 0-191074-70-7), November, 1990.

[7] "Spray Guns and Accessories" Sales Catalog, Form I-2008-A, Devilbiss Ransburg Industrial Coating Equipment, Barrie, Ontario, Canada, 1991.

[8] Dobson, D., "Leading Edge Robotic Finishing Technologies", SME Technical Paper FC89-608, *Finishing '89 Conference*, Cincinnati, Ohio, October 16-19, 1989, 27 pages.

[9] Duelen, G. et al, "An Off-Line Planning and Simulation System for the Programming of Coating Robots", *C.I.R.P. Annals*, V.38, n.1, 1989, pp.369-372.

[10] Faires, V.M., *Design of Machine Elements*, Collier-Macmillan Canada, Ltd., Toronto, Ontario, 1965.

[11] Fredriksson, L.B., "Robotics in a spray-up process", *Proc. of the 14th Int. Symposium on Industrial Robots*, Gothenburg, 1984, pp.297-303.

[12] Erik D. Goodman, E.D., Hoppensteadt, L.T.W., "A Method for Accurate Simulation of Robotic Spray Application Using Empirical Parameterization", *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991, pp.1357-1368.

[13] Goodman, E.D., "Users Guide - SPRAYTOOL - Release 1.1",*CASE Center for Computer-Aided Engineering and Manufacturing*, Michigan State University, East Lansing, MI 48824, April 6, 1989.

[14] Grunewald, P., "Car body painting with the spine spray system",*Proc. of the 14th Int. Symposium on Industrial Robots*, Gothenburg, 1984, pp.633-641.

[15] Jacobs, M.P., "Experiences from Programming Robots", SME Technical Paper MS89-159, *Robots in Aerospace Manufacturing Conference*, Irvine, California, February 20-23, 1989, 24 pages.

[16] Klein, A., "CAD-Based off-line programming of painting robots", *Robotica*, Vol.5, 1987, pp.267-271.

[17] Klein, A., "Off-line programming for robot painters", *Proc. of the 14th Int. Symposium on Industrial Robots*, Gothenburg, 1984, pp. 497-504.

[18] Pillon, G., "OLP: the key to expanding robot usage", *The Industrial Robot*, December 1988, pp.206-210.

[19] Roobol, N.R., *Industrial Painting Principles and Practices*, Hitchcock Publishing Co., Carol Stream IL, 1991.

[20] Schmidt, T.D., "Robotic Painting Equipment Packaging for the Painting Environment", *GMF Robotics*, Troy, Michigan.

[21] Schreiber, R.R., "Portrait of a Painting Robot", *Manufacturing Engineering*, November 1991, pp.55-60.

[22] Lefebvre, A.H., Senser, D.W., "Research Unravels Spray Mysteries", *Industrial Finishing*, June 1990, pp.16-20.

[23] Suh, S., Noh, S., "Automatic Trajectory Planning System (ATPS) for Spray Painting Robots", *Journal of Manufacturing Systems*, V.10, n.5, 1991, pp.396-406.

[24] Suh, S., Woo, I., Noh, S., "Development of An Automatic Trajectory Planning System (ATPS) for Spray Painting Robots", *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991, pp.1948-1955.

[25] Personal communication with staff of GM Fanuc Robotics, Detroit, Michigan, October 92.

Appendix A

Determination of Workpiece Area

The workpiece area, A_s is required for process cost calculations. If the workpiece is represented by a piecewise continuous, non-intersecting and closed curve its area may be found from Green's theorm of calculus. The area, A_s, enclosed by a contiguous set of n continuous curves may be determined from

$$A_s = \sum_{i=1}^{n} \int_{u_{i1}}^{u_{i2}} xy'du \tag{A.1}$$

where x and y are defined parametrically as functions of u. The order of performing the integration is important and determines the limits of integration. The limits shown correspond to traversing the boundary in a counterclockwise fashion keeping the interior region of the workpiece to the left. If the boundary is comprised solely of line segments, the area may be determined as follows. Express x and y parametrically as functions of u;

$$x = x_1 + u(x_2 - x_1) \tag{A.2}$$

$$y = y_1 + u(y_2 - y_1) \tag{A.3}$$

Now y' can be obtained by differentiating equation (A.3) with respect to u, i.e.,

$$y' = \frac{dy}{du} = (y_2 - y_1) \tag{A.4}$$

Substituting the RHS of equations (A.2) and (A.4) into equation (A.1) gives

$$A_s = \sum_{i=1}^{n} \int_{u_{i1}}^{u_{i2}} (x_1 + u(x_2 - x_1))(y_2 - y_1)du \tag{A.5}$$

which can be simplified to

$$A_s = \sum_{i=1}^{n} \left[(x_1(y_2 - y_1)u + \frac{u^2(x_2 - x_1)(y_2 - y_1))}{2} \right]_{u_{i1}}^{u_{i2}} \tag{A.6}$$

Substituting the bounds of parameter u, namely $u_{i1} = 0$ and $u_{i2} = 1$ into equation (A.6) yields

$$A_s = \sum_{i=1}^{n} \left[(x_{1_i}(y_{2_i} - y_{1_i}) + \frac{(x_{2_i} - x_{1_i})(y_{2_i} - y_{1_i}))}{2} \right] \tag{A.7}$$

Given the endpoint vertices of all line segments defining the workpiece, equation (A.7) can be used to determine the total workpiece area.

Appendix B

Program Source Code

The programs implemented in this work are broken into several different modules. The source code for each module is presented in the following sections. A brief description of what the module does preceeds the code.

B.1 Tools.c

This module contains the main line of the *paint* program.

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define ELEMENTS(array) (sizeof(array)/sizeof((array)[0]))
#define pi 3.14159265359
#define LEFT -1
#define RIGHT 1

struct func_entry{
  char *func_name;
  int (*func) ();
};
int goff();
int elloff();
int paint();
int paint1();
int paint2();
int paint3();
int phase2();
void drawout _((array,size,color));
void rotate_path _((file,theta));
```

```
double area _((polygon,size));
void minimum_cost  _((file,THETA,B,COST));

struct func_entry func_table[]={
  {/*MSG0*/"c:paint",paint},
  {/*MSG0*/"c:paint1",paint1},
  {/*MSG0*/"c:paint2",paint2},
  {/*MSG0*/"c:paint3",paint3},
  {/*MSG0*/"c:elloff",elloff},
  {/*MSG0*/"c:goff",goff},
  {/*MSG)*/"c:phase2",phase2},
};


int dofun      _((void));
int funcload   _((void));

/* FUNCTION: main ***************************************************************/
/********************************************************************************/
void main(argc,argv)
int argc;
char *argv[];
{
    short scode = RSRSLT;            /* Normal result code (default) */
    int stat;

    ads_init(argc, argv);            /* Open communication with AutoLISP */

    for ( ;; ) {                     /* Request/Result loop */

        if ((stat = ads_link(scode)) < 0) {
            printf(/*MSG1*/"FACT: bad status from ads_link() = %d\n", stat);
            fflush(stdout);
            exit(1);
        }

        scode = RSRSLT;              /* Reset result code */

        switch (stat) {

        case RQXLOAD:                /* Load & define functions */
            scode = funcload() == RTNORM ? RSRSLT : RSERR;
            break;

        case RQSUBR:                 /* Handle external function requests */
            scode = dofun() == RTNORM ? RSRSLT : RSERR;
            break;

        default:
            break;
        }
    }
}

/* FUNCTION: ********************************************************************/
/* Define this application's external functions.                                */
/* Return RTERROR on error, else RTNORM.                                        */
/********************************************************************************/
static int funcload()
{
```

```
    int i;

    for (i = 0; i < ELEMENTS(func_table); i++) {
        if (!ads_defun(func_table[i].func_name, i))
            return RTERROR;
    }
    return RTNORM;
}

/* FUNCTION: dofun ************************************************************/
/* Execute external function (called upon an RQSUBR request).                 */
/* Return value from the function executed, RTNORM or RTERROR.                */
/******************************************************************************/
static int dofun()
{
    struct resbuf *rb;
    int val;

    /* Get the function code and check that it's within range.
       (It can't fail to be, but paranoia doesn't hurt.) */
    if ((val = ads_getfuncode()) < 0 || val >= ELEMENTS(func_table)) {
        ads_fail(/*MSG2*/"Received nonexistent function code.");
        return RTERROR;
    }

    /* Fetch the arguments, if any. */
    rb = ads_getargs();

    /* Call the handler and return its success-failure status. */
    return (*func_table[val].func)(rb);
}

/* FUNCTION: paint ************************************************************/
/* This function is the final version of paint, it allows the user to         */
/* optimize the spray pattern                                                 */
/******************************************************************************/
int paint()
{
  /* VARIABLES FOR FILE NAMES */
  char *file="input_data";        /* INPUT DATA */
  char *file1="part_outline";     /* WORKPIECE BOUNDARY */
  char *outfile1="offset_curve"; /* OFFSET CURVE VERTICES */
  char *outfile2="paint_path";   /* DESCRIPTION OF PAINT PATH */
  char *outfile3="results";      /* LOOP RESULTS */
  char *outfile4="plotres";      /* RESULTS FILE FOR PLOTTING */

  /* VARIABLES FOR WORKPIECE AND OFFSET ARRAYS */
  long worksize;    /* WORKPIECE ARRAY SIZE */
  long offsetsize;  /* OFFSET ARRAY SIZE */
  double **work;    /* WORKPIECE ARRAY */
  double **work1;   /* WORKPIECE ARRAY AFTER ROTATION */
  double **offset;  /* OFFSET CURVE ARRAY */

  /* MISCELLANEAOUS VARIABLES */
  double CT,ST;             /* COS THETA & SIN THETA */
  long i,i1;                /* LOOP COUNTERS */
  double dtr=0.01745329;    /* CONVERSION FACTOR-DEGREES TO RADIANS */
  double ltmm=16666.6667;   /* CONVERSION FACTOR-l/min to mm3/sec */

  /* SPECIFIED PAINT THICKNESS */
```

```
double h;

/* PARAMETER INPUT VARIABLES */
char variable[40]; /* STRING VARIABLE FOR PARAMETER INPUT */
double value;      /* DOUBLE VARIABLE FOR PARAMETER INPUT */

/* OVERLAP RATIO */
double beta;

/* WORKPIECE AREA */
double A;

/* ELLIPSE DIMENSIONS */
double a;      /* MINOR AXIS DIMENSION */
double b;      /* MAJOR AXIS DIMENSION */
double b_min;  /* MINIMUM FEASIBLE MAJOR AXIS SIZE */
double b_max;  /* MAXIMUM FEASIBLE MAJOR AXIS SIZE */
double b_d;    /* ELLIPSE INCREMENT SIZE */
double ar;     /* ELLIPSE ASPECT RATIO */

/* SPRAY ORIENTATION VARIABLES */
double theta;    /* SPRAY ORIENTATION */
double theta_s;  /* STARTING ORIENTATION */
double theta_f;  /* FINISHING ORIENTATION */
double theta_d;  /* INCREMENTAL ANGLE FOR SEARCH */
double angle;    /* TEMPORARY VARIABLE FOR RADIANS/DEGREES */

/* VARIABLES FOR SPRAY VELOCITIES */
double v;      /* SPRAY VELOCITY */
double v_min;  /* MINIMUM VELOCITY (FOR ACCEPTABLE SPRAYING) */
double v_max;  /* MAXIMUM VELOCITY OF MACHINE */

/* VARIABLES FOR FLOWRATE */
double qp;      /* FIXED TOTAL FLOWRATE */
double q;       /* FIXED NET FLOWRATE */
double qp_max;  /* TOTAL VOLUME FLOWRATE */
double q_max;   /* NET VOLUME FLOWRATE */
double eta;     /* TRANSFER EFFICIENCY */

/* VARIABLES FOR PROCESS TIMES */
double Ti;  /* TOTAL TIME WITH SPRAY ON */
double To;  /* TOTAL TIME WITH SPRAY OFF */

/* VARIABLES FOR SPRAY DISTANCES */
double Xi; /* SPRAY OFF DISTANCE */
double Xo; /* SPRAY ON DISTANCE */
double s;  /* PATH INTERVAL */

/* VARIABLES FOR PROCESS WASTE */
double Wos;  /* WASTE DUE TO OVERSPRAY */
double Wsb;  /* WASTE DUE TO SPRAY BY */

/* VARIABLES FOR OPTIMUM VALUES */
double b_opt;     /* OPTIMUM ELLIPSE MAJOR AXIS SIZE */
double q_opt;     /* OPTIMUM FLOWRATE */
double v_opt;     /* OPTIMUM VELOCITY */
double theta_opt; /* OPTIMUM SPRAY ORIENTATION */

/* PROCESS COST VARIABLES */
double cost;           /* PROCESS COST (DOLLARS) */
```

```
double C1,C2,C3,C4,C5; /* COMPONENTS OF PROCESS COST */
double matl_cost;      /* PAINT COST ($/CUBIC MM)*/
double disp_cost;      /* PAINT DISPOSAL COST ($/CUBIC MM)*/
double time_cost;      /* COST OF TIME ($/SECOND) */

/* SYSTEM VARIABLES */
int DOCRES=0;
int INITMODE=0;
int PREFSPS=0;

/* FILE VARIABLES */
FILE *fp;   /* INPUT FILE */
FILE *fp1;  /* RESULTS FILE */
FILE *fp2;  /* RESULTS FILE FOR PLOTTING */
struct resbuf val;
val.restype = RTSHORT;
val.resval.rint = 0;

ads_setvar("cmdecho",&val);

fp=fopen(file,"r");
while(fscanf(fp,"%[^:]: %lf",variable,&value) != EOF){
  if(strstr(variable,"specified paint thickness")) h=value;
  if(strstr(variable,"starting angle") != NULL) theta_s=value*dtr;
  if(strstr(variable,"ending angle") != NULL) theta_f=value*dtr;
  if(strstr(variable,"angle increment") != NULL) theta_d=value*dtr;
  if(strstr(variable,"minimum velocity") != NULL) v_min=value;
  if(strstr(variable,"maximum velocity") !=NULL) v_max=value;
  if(strstr(variable,"maximum total flowrate") !=NULL) qp_max=value*ltmm;
  if(strstr(variable,"transfer efficiency") != NULL) eta=value;
  if(strstr(variable,"minimum ellipse height") !=NULL) b_min=value;
  if(strstr(variable,"maximum ellipse height") !=NULL) b_max=value;
  if(strstr(variable,"ellipse increment size") !=NULL) b_d=value;
  if(strstr(variable,"minimum aspect ratio") !=NULL) ar=value;
  if(strstr(variable,"overlap") !=NULL) beta=value;
  if(strstr(variable,"material cost") !=NULL ) matl_cost=value/1000000.0;
  if(strstr(variable,"disposal cost") !=NULL ) disp_cost=value/1000000.0;
  if(strstr(variable,"time cost") !=NULL) time_cost=value/3600.0;
  if(strstr(variable,"INITMODE") !=NULL ) INITMODE=(int)value;
  if(strstr(variable,"PREFSPS") !=NULL ) PREFSPS=(int)value;
}
fclose(fp);
ads_textpage();
ads_printf("\nThe parameters of the paint function are as follows\n");

ads_printf("\n%-25s = %.2lf mm","PAINT THICKNESS",h);
ads_printf("\n%-25s = %.2lf deg","STARTING ANGLE",theta_s/dtr);
ads_printf("\n%-25s = %.2lf deg","ENDING ANGLE",theta_f/dtr);
ads_printf("\n%-25s = %.2lf deg","ANGLE INCREMENT",theta_d/dtr);
ads_printf("\n%-25s = %.2lf mm/s","MINIMUM VELOCITY",v_min);
ads_printf("\n%-25s = %.2lf mm/s","MAXIMUM VELOCITY",v_max);
ads_printf("\n%-25s = %.2lf l/min","MAXIMUM FLOWRATE",qp_max/ltmm);
ads_printf("\n%-25s = %.2lf","TRANSFER EFFICIENCY",eta);
ads_printf("\n%-25s = %.2lf mm","MINIMUM ELLIPSE HEIGHT",b_min);
ads_printf("\n%-25s = %.2lf mm","MAXIMUM ELLIPSE HEIGHT",b_max);
ads_printf("\n%-25s = %.2lf mm","ELLIPSE INCREMENT",b_d);
ads_printf("\n%-25s = %.2lf","ELLIPSE ASPECT RATIO",ar);
ads_printf("\n%-25s = %.2lf","OVERLAP",beta);
ads_printf("\n%-25s = %.6lf $/l","MATERIAL COST",matl_cost*1000000.0);
ads_printf("\n%-25s = %.6lf $/l","DISPOSAL COST",disp_cost*1000000.0);
```

```
ads_printf("\n%-25s = %.4lf $/hr","TIME COST",time_cost*3600.0);
ads_printf("\n%-25s = %d", "INITMODE",INITMODE);
ads_printf("\n%-25s = %d", "PREFSPS",PREFSPS);


ads_printf("\n\nType enter if you want to continue. If one or more of the ");
ads_printf("parameters are to be changed press Cntl-C to exit the ");
ads_printf("function, then make the necessary changes in the text file ");
ads_printf("'input_data'.");

if(ads_getstring(0,NULL,variable) == RTCAN)
   ads_exit(0);
else ads_graphscr();

/* OPEN RESULTS FILE AND PLOTTING FILE */
fp1=fopen(outfile3,"w");
fp2=fopen(outfile4,"w");

/* CREATE FILE DESCRIBING WORKPIECE */
get_boundary(file1);
ads_printf("\nCalculations in progress, please wait...");
work = file_to_array(file1,&worksize);

/* CALCULATE WORKPIECE AREA */
ads_printf("\ncalculating workpiece area...");
A=area(work,worksize);

/* ALLOCATE MEMORY FOR ROTATED WORKPIECE ARRAY */
work1=(double **)malloc((unsigned)(worksize)*sizeof(double*));
for(i=0;i < worksize; i++)
  work1[i]=(double *)malloc(3*sizeof(double));

/* LOOP FOR RANGE OF ELLIPSE SIZE */
b=b_min;
while(b<=b_max){

  /* CHECK FOR USER BREAK */
  if(ads_usrbrk())
    ads_abort("\nProgram cancelled at users request, goodbye!");

  ads_printf("\nsearch %.0lf percent complete, <Cntl-C to exit>"
            ,(100.0*(b-b_min)/(b_max-b_min)));

  /* CALCULATE PATH INTERVAL */
  s=(1.0-beta)*2.0*b;

  /* CALCULATE ELLIPSE MINOR AXIS LENGTH */
  a=ar*b;

  /* LOOP FOR RANGE OF WORKPIECE ORIENTATION */
  theta=theta_s;
  i1=0;
  while(theta<=theta_f){
    ads_printf(".");

    CT = cos(-theta);
    ST = sin(-theta);

    /* ROTATE WORKPIECE BY ANGLE -THETA */
    for(i=0; i<worksize;i++){
```

```
      work1[i][0]=work[i][0]*CT-work[i][1]*ST;
      work1[i][1]=work[i][0]*ST+work[i][1]*CT;
      work1[i][2]=work[i][2];
    }

    /* CALCULATE OFFSET CORRESPONDING TO ROTATED WORKPIECE */
    ellipse_offset(work1,worksize,RIGHT,a,b,outfile1);
    offset = file_to_array(outfile1,&offsetsize);

    /* CALCULATE SPRAY PATH */
    Xi=0.0;
    Xo=0.0;
    spray_path4(work1,offset,offsetsize,b,s,&Xi,&Xo,INITMODE);

    /* CALCULATE PAINT FLOWRATE BASED ON MAXIMUM VELOCITY */
    qp=h*v_max*s/eta;
    v=v_max;

    /* IF FLOWRATE > SPECIFIED MAXIMUM, RESET TO MAXIMUM  */
    if(qp>qp_max){
      v=qp_max*eta/h/s;
      qp=qp_max;}

    /* CALCULATE NET FLOWRATE */
    q = qp*eta;

    /* CALCULATE PROCESS TIMES */
    Ti = Xi/v;
    To = Xo/v_max;

    /* CALCULATE PAINT WASTE */
    Wos = (1-eta)*qp*Ti;
    Wsb = q*Ti-A*h;

    C1=time_cost*Ti;
    C2=time_cost*To;
    C3=(matl_cost+disp_cost)*Wos;
    C4=(matl_cost+disp_cost)*Wsb;
    C5=matl_cost*A*h;

    cost = C1+C2+C3+C4+C5;

    /* OUTPUT RESULTS TO FILES */
    if(i1) fprintf(fp2,"%.4lf %.4lf %.4lf",b,theta,cost);
    else fprintf(fp2,"\n%.4lf %.4lf %.4lf",b,theta,cost);

    fprintf(fp1,"\n%.8lf %.8lf %.8lf",b,theta/dtr,v);
    fprintf(fp1," %.8lf %.8lf %.8lf %.8lf",qp/ltmm,cost,C1,C2);
    fprintf(fp1," %.8lf %.8lf %.8lf",C3,C4,C5);
    fprintf(fp1," %.8lf %.8lf %.8lf %.8lf",Xi,Xo,Ti,To);

    theta=theta+theta_d;

  }
  fprintf(fp2,"\n");
  fprintf(fp1,"\n");
  b=b+b_d;
}

fclose(fp1);
```

```
  fclose(fp2);

  /* FIND OPTIMUM VALUE IN FILE */
  minimum_cost(outfile3,&theta,&b,&v,&qp,&cost); /* RETURNS THETA IN RADIANS */
  ads_printf("\nmin cost = %lf, b=%.2lf, theta=%.2lf, v=%.1lf, qp=%.6lf",
             cost,b,theta/dtr,v,qp);

  /* RECALCULATE s & a CORRESPONDING TO OPTIMUM b */
  s=(1.0-beta)*2.0*b;
  a=b*ar;

  /* ROTATE WORK TO OPTIMUM VALUE OF THETA */
  CT = cos(-theta);
  ST = sin(-theta);
  for(i=0; i<worksize;i++){
    work1[i][0]=work[i][0]*CT-work[i][1]*ST;
    work1[i][1]=work[i][0]*ST+work[i][1]*CT;
    work1[i][2]=work[i][2];
  }

  /* CALCULATE OFFSET CORRESPONDING TO OPTIMUM ANGLE OF WORK */
  ellipse_offset(work1,worksize,RIGHT,a,b,outfile1);
  offset = file_to_array(outfile1,&offsetsize);

  /* CALCULATE SPRAY PATH */
  spray_path3(work1,offset,offsetsize,b,s,outfile2,INITMODE);

  /* ROTATE SPRAY PATH TO ANGLE THETA */
  rotate_path(outfile2,theta);

  /* DISPLAY RESULTING SPRAY PATH */
  display_path2(outfile2,a,b,theta);

  return(RTNORM);
}

/* FUNCTION: minimum_cost ***********************************************/
/* Given the file name of the results file this routine searches for the      */
/* record with the least cost and copies select parameters from that record   */
/************************************************************************/
void minimum_cost(file,THETA,B,V,QP,COST)
char *file;
double *THETA;
double *B;
double *COST;
double *V;
double *QP;
{
  int count=0;
  double b,theta,cost,v,qp;   /* TEMPORARY READ VARIABLES */
  double dtr=0.01745329; /* CONVERSION FACTOR-DEG TO RADIANS */
  FILE *fp;


  fp=fopen(file,"r");
  while(fscanf(fp,"%lf %lf %lf %lf %lf %*f %*f %*f %*f %*f %*f %*f %*f %*f",
               &b,&theta,&v,&qp,&cost)!=EOF){
    if(!count){
      *COST = cost;
      *B = b;
```

```
      *THETA = theta*dtr;
      *V = v;
      *QP = qp;
      count++;
    }
    if(cost < *COST){
      *COST = cost;
      *B = b;
      *THETA = theta*dtr;
      *V = v;
      *QP = qp;
    }
  }
  fclose(fp);
}


/* FUNCTION: rotate_path ***********************************************/
/* Given the file containing the paint path description this function rotates */
/* the coordinates about a given angle theta, and rewrites the file with the  */
/* same name                                                                  */
/*****************************************************************************/
void rotate_path(file,theta)
char *file;
double theta;
{
char keyword1[30],keyword2[30],*tempfile="fileXXXXXX";
double x1,x2,y1,y2,X1,X2,Y1,Y2;
double CT,ST;
FILE *fp1,*fp2;

/* OPEN PAINT PATH FILE AND TEMPORARY FILE */
fp1 = fopen(file,"r");
fp2 = fopen(tempfile=mktemp(tempfile),"w");

/* CALCULATE THE COSINE AND SINE OF THETA */
CT = cos(theta);
ST = sin(theta);

/* READ THROUGH PAINT PATH FILE , ROTATE AND COPY TO TEMPORARY FILE */
while(fscanf(fp1,"%s",keyword1) != EOF){
  if(!strcmp(keyword1,"system")){
    fscanf(fp1,"%s",keyword2);
    fprintf(fp2,"\n%s %s",keyword1,keyword2);
  }

  if(!strcmp(keyword1,"line")){
    fscanf(fp1,"%lf %lf %lf %lf",&x1,&y1,&x2,&y2);
    X1=x1*CT-y1*ST;
    Y1=x1*ST+y1*CT;
    X2=x2*CT-y2*ST;
    Y2=x2*ST+y2*CT;
    fprintf(fp2,"\n%s %lf %lf %lf %lf",keyword1,X1,Y1,X2,Y2);
  }
}
fclose(fp1);
fclose(fp2);

/* DELETE PAINT PATH FILE */
remove(file);
```

```
/* CHANGE NAME OF TEMPORARY FILE TO PAINT PATH FILE */
rename(tempfile,file);

}

/* FUNCTION: drawout ************************************************************/
/* Draws polygon based on input array of vertices                                */
/* Number of vertices specified by "size"                                       */
/* Drawing color specified by "color"                                           */
/********************************************************************************/
void drawout(array,size,color)
char *color;
long size;
double **array;
{
long i;
ads_command(RTSTR,"color",RTSTR,color,RTNONE);

for(i=0; i<size-1; i++)
  ads_command(RTSTR,"line",RTPOINT,array[i],RTPOINT,array[i+1],RTSTR,"",RTNONE);

}

/* FUNCTION: paint1 *************************************************************/
/* Spray pattern = rectangle                                                    */
/* No offset curve used                                                         */
/* Paints to horizontal workpiece extremes                                      */
/* Does not handle islands                                                      */
/********************************************************************************/
int paint1()
{
  char *file = "part_outline";
  char *outfile = "paint_path";
  double **data;
  long size,i;
  double a = 20.0;
  double b = 45.0;
  double beta = 0.175;

  get_boundary(file);
  data = file_to_array(file,&size);
  spray_path(data,size,a,b,beta,outfile);
  display_path(outfile);
  return(RTNORM);
}

/* FUNCTION: paint2 *************************************************************/
/* Spray pattern = ellipse                                                      */
/* Elliptical based offset curve used                                           */
/* Paints to horizontal workpiece extremes                                      */
/* Does not handle islands                                                      */
/********************************************************************************/
int paint2()
{
  char *file = "part_outline";
  char *outfile = "paint_path";
  double **data;
  long size,i;
  double a = 20.0;
```

```
  double b = 45.0;
  double beta = 0.175;

  get_boundary(file);
  data = file_to_array(file,&size);
  spray_path2(data,size,a,b,beta,outfile);
  display_path2(outfile);
  return(RTNORM);
}

/* FUNCTION: paint3 ************************************************************/
/* Spray pattern = ellipse                                                     */
/* Elliptical based offset curve used                                          */
/* Turns off paint in workpiece voids                                          */
/* Does not handle islands                                                     */
/*******************************************************************************/
int paint3()
{
  char *file = "part_outline";
  char *outfile1 = "part_curve";
  char *outfile2 = "paint_path";
  double **data1,**data2;
  long size1,size2,i;
  double a = 20.0;
  double b = 40.0;
  double beta = 0.175;

  get_boundary(file);
  data1 = file_to_array(file,&size1);
  ellipse_offset(data1,size1,RIGHT,a,b,outfile1);
  data2 = file_to_array(outfile1,&size2);
  spray_path3(data2,size2,a,b,beta,outfile2);
  display_path2(outfile2);
  return(RTNORM);
}

/* FUNCTION goff (group offset) ************************************************/
/* Creates constant distance offset curve                                      */
/* Prompts user for curve to offset,and offset ammount                         */
/*******************************************************************************/
int goff()
{
  char *file = "part_outline";
  char *outfile = "offset_curve";
  long size;
  double **data;
  ads_real off;
  struct resbuf val;

  val.restype = RTSHORT;
  val.resval.rint = 0;


  ads_getreal("Enter the desired offset distance:",&off);

  ads_setvar("cmdecho",&val);
  get_boundary(file);
  data = file_to_array(file,&size);
  offset(data,size,RIGHT,off,outfile);
  display_path(outfile);
```

```
  return(RTNORM);
}


/* FUNCTION elloff (ellipse offset) ***************************************/
/* This function requests the major and minor axes of an ellipse and creates  */
/* an offset curve around a boundary such that the ellipse is always tangent  */
/* to it                                                                      */
/**************************************************************************/
int elloff()
{
  char *file = "part_outline";
  char *outfile = "offset_curve";
  long size;
  double **data;
  ads_real a,b;
  struct resbuf val;

  val.restype = RTSHORT;
  val.resval.rint = 0;


  ads_getreal("Enter the ellipse major axis length:",&a);
  ads_getreal("Enter the ellipse minor axis length:",&b);

  ads_setvar("cmdecho",&val);
  get_boundary(file);
  data = file_to_array(file,&size);
  ellipse_offset(data,size,RIGHT,a,b,outfile);
  display_path(outfile);
  return(RTNORM);
}

/* FUNCTION: phase2 *******************************************************/
/* phase2 gets a boundary description from the user and generates a list of   */
/* intersection vertices at a defined height                                  */
/**************************************************************************/
int phase2()
{
  char *file="boundary";
  int i,num_intsect;
  long size;
  double **data,*ptr;
  ads_point pt;


  get_boundary(file);
  data=file_to_array(file,&size);
  ads_getpoint(NULL,"\nIndicate horizontal line for vertices",pt);
  ptr = intloop2(data,size,pt[Y],&num_intsect);

  for(i=0; i< num_intsect; i++)
    ads_printf("\nintersection %d = %.3lf",i+1,ptr[i]);

  return(RTNORM);
}
```

B.2 Display.c

This module contains routines for displaying the spray path on the AutoCAD graphics screen.

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359

void display_path    _((file));
void display_path2   _((file));

/* FUNCTION: display_path ************************************************/
/* Display path opens the file specified by the parameter "file" and displays */
/* its contents on the screen. The input file must be in the standard format  */
/* specified in the appendix. The contents are displayed in red if the paint  */
/* is on and in yellow if the paint is off. The drawing color is reset to     */
/* upon completion of the subroutine.                                         */
/*************************************************************************/
void display_path(file)
char *file;
{
  char etype[10];
  char system_command[10];
  double cp[2],radius,sa,da,start[2],end[2];
  FILE *fp;
  fp = fopen(file,"r");

  /* SET DRAWING COLOR TO RED */
  ads_command(RTSTR,"color",RTSTR,"red",RTNONE);

  while(fscanf(fp,"%s",etype) != EOF){
    if(!strcmp(etype,"line")){
      fscanf(fp,"%lf %lf %lf %lf",&start[X],&start[Y],&end[X],&end[Y]);
      ads_command(RTSTR,"line",RTPOINT,start,RTPOINT,end,RTSTR,"",RTNONE);
    }
    if(!strcmp(etype,"arc")){
      fscanf(fp,"%lf %lf %lf %lf %lf",&cp[X],&cp[Y],&radius,&sa,&da);
      start[X] = cp[X] + radius * cos(sa/180.0*pi);
      start[Y] = cp[Y] + radius * sin(sa/180.0*pi);
      ads_command(RTSTR,"arc",RTSTR,"c",RTPOINT,cp,RTPOINT,start,RTSTR,"a",
      RTREAL,da,0);
    }
    if(!strcmp(etype,"system")){
      fscanf(fp,"%s",system_command);
      if(!strcmp(system_command,"painton"))
        ads_command(RTSTR,"color",RTSTR,"red",RTNONE);
      if(!strcmp(system_command,"paintoff"))
        ads_command(RTSTR,"color",RTSTR,"yellow",RTNONE);
    }
  }

  /* SET DRAWING COLOR BACK TO WHITE */
  ads_command(RTSTR,"color",RTSTR,"white",RTNONE);
}
```

```
/* FUNCTION: display_path2 ***************************************************/
/* Display path opens the file specified by the parameter "file" and displays */
/* its contents on the screen. The input file must be in the standard format  */
/* specified in the appendix. The contents are displayed in red if the paint  */
/* is on and in yellow if the paint is off. The drawing color is reset to     */
/* white upon completion of the subroutine. An ellipse is drawn at pass endpts*/
/*****************************************************************************/
void display_path2(file,a,b,theta)
char *file;
double a,b,theta;
{
  char etype[10],astring[20],bstring[20],thetastring[20],value1[60];
  char system_command[10];
  double cp[2],radius,sa,da,start[2],end[2];
  double rtd=57.2957795; /* CONVERSION FACTOR-RADIANS TO DEG */
  FILE *fp;
  fp = fopen(file,"r");

  theta=theta*rtd;
  strcpy(system_command,"painton");

  /* CONVERT a,b AND theta INPUT PARAMETERS TO STRINGS */
  gcvt(a,12,astring);
  gcvt(b,12,bstring);
  gcvt(theta,12,thetastring);

  strcpy(value1,"@");
  strcat(value1,astring);
  strcat(value1,"<");
  strcat(value1,thetastring);

  /* SET DRAWING COLOR TO RED */
  ads_command(RTSTR,"color",RTSTR,"red",RTNONE);

  while(fscanf(fp,"%s",etype) != EOF){

    if(!strcmp(etype,"line")){
      fscanf(fp,"%lf %lf %lf %lf",&start[X],&start[Y],&end[X],&end[Y]);
      ads_command(RTSTR,"line",RTPOINT,start,RTPOINT,end,RTSTR,"",0);
      if(!strcmp(system_command,"painton")){
        ads_command(RTSTR,"ellipse",RTSTR,"C",RTPOINT,start,RTSTR,value1,
                    RTSTR,bstring,0);
        ads_command(RTSTR,"ellipse",RTSTR,"C",RTPOINT,end,RTSTR,value1,
                    RTSTR,bstring,0);
      }
    }

    if(!strcmp(etype,"arc")){
      fscanf(fp,"%lf %lf %lf %lf %lf",&cp[X],&cp[Y],&radius,&sa,&da);
      start[X] = cp[X] + radius * cos(sa/180.0*pi);
      start[Y] = cp[Y] + radius * sin(sa/180.0*pi);
      ads_command(RTSTR,"arc",RTSTR,"c",RTPOINT,cp,RTPOINT,start,RTSTR,"a",
      RTREAL,da,0);
    }

    if(!strcmp(etype,"system")){
      fscanf(fp,"%s",system_command);
      if(!strcmp(system_command,"painton"))
        ads_command(RTSTR,"color",RTSTR,"red",RTNONE);
      if(!strcmp(system_command,"paintoff"))
```

```
        ads_command(RTSTR,"color",RTSTR,"yellow",RTNONE);
    }
  }

  /* SET DRAWING COLOR BACK TO WHITE */
  ads_command(RTSTR,"color",RTSTR,"white",RTNONE);
}
```

B.3 Path.c

Path.c contains routines that assemble the workpiece boundary and determine intersection points between the spray pass centers and the workpiece offset curve.

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359
#define LEFT -1
#define RIGHT 1

int get_boundary     _((file));
void next_entity     _((point,entity));
void intsect         _((y,p1,p2,result));
void intloop         _((boundary,size,h,left_vertex,right_vertex));
double *intloop2  _((boundary,vertices,h,num_vertices));
double **intloop3 _((boundary,vertices,h,num_intsect));
int compress_path _((path,size));

/* FUNCTION: get_boundary ***********************************************/
/************************************************************************/
int get_boundary(file)
char *file;
{
  char etype[10];
  int i=0,test;
  double tolerance=0.001;
  ads_name current_entity,first_entity;
  ads_point first_search_pt,search_pt,start,end,pt1;
  FILE *fp;

  /* REQUEST USER TO INDICATE ONE ELEMENT ON BOUNDARY */
  test=ads_entsel("\nIndicate an element on the boundary",
                  first_entity,search_pt);
  if(test != RTNORM){
    ads_fail("Could not find entity");
    ads_exit(0);
  }

  /* OPEN FILE FOR RESULTING BOUNDARY DESCRIPTION */
  fp = fopen(file,"w");

  /* EQUATE ENTITY NAMES */
  equate_name(current_entity,first_entity);
```

```
  /* DETERMINE THE ENDPOINT CLOSEST TO THE SEARCH POINT */
  closest_end(current_entity,search_pt,pt1);

  /* WRITE DESCRIPTION OF FIRST ENTITY TO FILE */
  file_entity(fp,current_entity,pt1);

  /* RESET SEARCH POINT */
  furthest_end(current_entity,pt1,search_pt);

  /* GET NEXT ENTITY */
  next_entity(search_pt,current_entity);

  while(current_entity[0] != first_entity[0] &&
        current_entity[1] != first_entity[1]){
    file_entity(fp,current_entity,search_pt);
    furthest_end(current_entity,search_pt,search_pt);
    next_entity(search_pt,current_entity);
    }
  fclose(fp);
}

/* FUNCTION: ***************************************************************/
/***************************************************************************/
void next_entity(point,entity)
ads_point point;
ads_name entity;
{
  ads_name temp_ss;
  double tolerance = 0.001;

  entity_search(point,tolerance,temp_ss);
  ads_ssdel(entity,temp_ss);
  ads_ssname(temp_ss,0,entity);
  ads_ssfree(temp_ss);
}

/* FUNCTION INTLOOP ********************************************************/
/* DETERMINES IF ANY LINE IN GIVEN BOUNDARY INTERSECTS A HORIZONTAL LINE OF A */
/* GIVEN HEIGHT                                                            */
/***************************************************************************/
void intloop(boundary,num_vertices,h,left_vertex,right_vertex)
double **boundary;
long num_vertices;
double h;
double left_vertex[3],right_vertex[3];
{
  double result[3];
  int i=0,j=0,n;
  double vertex_list[50][3];

  n=num_vertices;
  for(i=0;i < n-1;i++){
    if(boundary[i][Y]<=h && boundary[i+1][Y]>h ||
       boundary[i][Y]>=h && boundary[i+1][Y]<h){
       intsect(h,boundary[i],boundary[i+1],result);
       vertex_list[j][X]=result[X];
       vertex_list[j][Y]=result[Y];
       vertex_list[j][Z]=result[Z];
       j++;
    }
```

```
  }

  /* PULL LEFT AND RIGHT VERTICES OUT OF VERTEX LIST */
  left_vertex[X] = vertex_list[0][X];
  right_vertex[X] = vertex_list[0][X];
  for(i=0;i <= j-1;i++){
    if(vertex_list[i][X] < left_vertex[X])
      equate_point(left_vertex,vertex_list[i]);
    if(vertex_list[i][X] > right_vertex[X])
      equate_point(right_vertex,vertex_list[i]);
  }
}

/* FUNCTION: intloop2 ********************************************************/
/* Given an array containing the definition of a closed boundary and a height */
/* this function determines the intersections of a line at the given height   */
/* with the boundary. It returns a pointer to the vector of x coordinates and */
/* sets num_intsect to the number of intersections that exist                 */
/*****************************************************************************/
double *intloop2(boundary,vertices,h,num_intsect)
int *num_intsect;
double **boundary;
long vertices;
double h;
{
  double result[3];
  int i=0,j=0,n;
  double *ptr;

  /* COUNT THE NUMBER OF INTERSECTIONS */
  n = 0;
  for(i=0;i < vertices-1;i++)
    if(boundary[i][Y]<=h && boundary[i+1][Y]>h ||
       boundary[i][Y]>=h && boundary[i+1][Y]<h)
       n++;

  /* ALLOCATE MEMORY TO HOLD "X" COMPONENTS OF INTERSECTION VERTICIES */
  ptr=(double *)malloc((unsigned)n * sizeof(double));

  /* ASSIGN VALUES TO VECTOR */
  j=0;
  for( i=0; i < vertices-1 ; i++){
    if(boundary[i][Y]<=h && boundary[i+1][Y]>h ||
       boundary[i][Y]>=h && boundary[i+1][Y]<h){
       intsect(h,boundary[i],boundary[i+1],result);
       ptr[j]=result[X];
       j++;
    }
  }

  /* SORT VECTOR IN ASCENDING ORDER */
  picksort1(n,ptr);

  *num_intsect = n;
  return(ptr);
}

/* FUNCTION: intloop3 ********************************************************/
/* Given an array containing the definition of a closed boundary and the y    */
/* coordinate of a line this function determines the intersections of the     */
```

```
/* line with the boundary. It returns a pointer to a two dimensional array    */
/* containing the x coordinates of the intersections and the corresponding    */
/* index of the boundary element with which the intersection was made.        */
/* The array is sorted in ascending order of the x coordinates. The function  */
/* also sets the variable num_intsect to equal the number of intersection     */
/* points that exist.                                                         */
/******************************************************************************/
double **intloop3(boundary,vertices,h,num_intsect)
int *num_intsect;
double **boundary;
long vertices;
double h;
{
  double result[3];
  int i=0,j=0,n;
  double **ptr;

  /* COUNT THE NUMBER OF INTERSECTIONS */
  for(i=0,n=0;i < vertices-1;i++)
    if(boundary[i][Y]<=h && boundary[i+1][Y]>h ||
       boundary[i][Y]>=h && boundary[i+1][Y]<h)
       n++;

  /* ALLOCATE MEMORY TO HOLD "X" COMPONENTS OF INTERSECTION VERTICIES */
  ptr=(double **)malloc((unsigned)n * sizeof(double *));
  for(i=0;i<n;i++)
    ptr[i]=(double *)malloc(2 * sizeof(double));

  /* ASSIGN VALUES TO ARRAY */
  for(i=0,j=0; i < vertices-1 ; i++){
    if(boundary[i][Y]<=h && boundary[i+1][Y]>h ||
       boundary[i][Y]>=h && boundary[i+1][Y]<h){
       intsect(h,boundary[i],boundary[i+1],result);
       ptr[j][0]=result[X];
       ptr[j][1]=(double)i+1;
       j++;
    }
  }

  /* SORT ARRAY IN ASCENDING ORDER */
  picksort2(n,ptr);

  *num_intsect = n;
  return(ptr);
}

/* FUNCTION: INTSECT **********************************************************/
/* CALCULATES INTERSECTION OF HORIZONTAL LINE AND GIVEN LINE                  */
/******************************************************************************/
void intsect(y,p1,p2,result)
double y;
double *p1,*p2,*result;
{
  double eps=0.000001,alpha;
  if(fabs(p1[X]-p2[X]) < eps){
    result[X]=p1[X];
    result[Y]=y;
  }
  else{
    alpha=(y-p1[Y])/(p2[Y]-p1[Y]);
```

```
    result[X]=p1[X]+alpha*(p2[X]-p1[X]);
    result[Y]=y;
  }
}

/* FUNCTION: compress_path ***************************************************/
/* Given a n x 2 array describing the on/off points of the spray this function*/
/* checks whether the gaps between on and off are less than the value of 'x'  */
/* If so the gap is eliminated and the array reduced so that all gaps are > x */
/*****************************************************************************/   int compress_path(path,size,x)
int size;
double **path,x;
{
   int i=1,p,size2;
   double a = 10,S,F;

   /* IF PATH SIZE IS ONLY ONE SEGMENT RETURN AND LEAVE PATH UNCHANGED */
   if(size == 1) return(size);

   /* OTHERWISE CONTINUE */
   S = path[0][0];
   F = path[0][1];
   size2 = size;
   while(i < size2){
      if(path[i][0] - path[i-1][1] < 2.0*a){
         path[i-1][1] = path[i][1];
         size2--;
         for(p=i;p < size2;p++){
            path[p][0] = path[p+1][0];
            path[p][1] = path[p+1][1];
         }
      }
      else i++;
   }
   return(size2);
}
```

B.4 Algebra.c

This module contains routines that determine the dot and cross product of vectors.

```
#include  <math.h>
#include  "adslib.h"

void dot_product _((v1,v2,result));
void cross_product _((V1,V2,result));

/* FUNCTION: dot_product *****************************************************/
/* Given 2 vectors this function calculates their dot or scalar product and  */
/* places the result in result                                               */
/*****************************************************************************/
void dot_product(v1,v2,result)
double *v1,*v2,*result;
{
  *result = v1[X]*v2[X]+v1[Y]*v2[Y]+v1[Z]*v2[Z];
}
```

```
/* FUNCTION: cross_product ************************************************/
/* Performs the cross product on the given vectors, ie (V1 X V2), and places  */
/* the result in the array pointed to by "result"                             */
/***************************************************************************/
void cross_product(V1,V2,result)
double *V1,*V2,*result;
{
  result[X]=V1[Y]*V2[Z]-V2[Y]*V1[Z];
  result[Y]=V2[X]*V1[Z]-V1[X]*V2[Z];
  result[Z]=V1[X]*V2[Y]-V2[X]*V1[Y];
}
```

B.5 Utility.c

This module contains routines that equate and compare point coordinates and calculate the workpiece area.

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359
#define LEFT -1
#define RIGHT 1

void equate_name     _((name1,name2));
void equate_point    _((pt1,pt2));
int test_point_equality _((pt1,pt2,error));
void picksort1  _((n,arr));
void picksort2 _((n,arr));
int next_x _((A,asize,a,B,bsize,b,X));
void bool_add _((A,asize,B,bsize));
double work_area _((work,size));

/* FUNCTION: equate_point *************************************************/
/* Equate the elements of the two points passed, ie pt1[0]=pt2[0]...,etc      */
/***************************************************************************/
void equate_point(pt1,pt2)
ads_point pt1,pt2;
{
  pt1[X]=pt2[X];
  pt1[Y]=pt2[Y];
  pt1[Z]=pt2[Z];
}

/* FUNCTION: ***************************************************************/
/* Equate name copies the contents of one name into another. The format is    */
/* name1=name2. In ADS names are defined by: typedef long ads_name[2];        */
/***************************************************************************/
void equate_name(name1,name2)
ads_name name1,name2;
{
  name1[0]=name2[0];
  name1[1]=name2[1];
```

```
}

/* FUNCTION: test_point_equality ***********************************************/
/* Tests if each member of two ads_point variables are equal, if so function  */
/* function returns 1, otherwise returns 0.                                   */
/******************************************************************************/
int test_point_equality(pt1,pt2,error)
double *pt1,*pt2,error;
{
  if(pt1[X]-pt2[X]<error && pt1[Y]-pt2[Y]<error && pt1[Z]-pt2[Z]<error)
    return(1);
  else
    return(0);
}

/* FUNCTION: picksort1 *********************************************************/
/* Sorts a single dimensional array in ascending order using straight         */
/* insertion method. (See Numerical Recipies in C, page 242)                  */
/******************************************************************************/
void picksort1(n,arr)
int n;
double *arr;
{
  int i,j;
  double a;

  for(j=1;j<n;j++){
    a=arr[j];
    i=j-1;
    while(i >= 0 && arr[i] > a){
       arr[i+1]=arr[i];
       i--;
    }
    arr[i+1]=a;
  }
}

/* FUNCTION: picksort2 *********************************************************/
/* Sorts a two dimensional array in ascending order using straight            */
/* insertion method. (See Numerical Recipies in C, page 242                   */
/******************************************************************************/
void picksort2(n,arr)
int n;
double **arr;
{
  int i,j;
  double a,b;

  for(j=1;j<n;j++){
    a=arr[j][0];
    b=arr[j][1];
    i=j-1;
    while(i >= 0 && arr[i][0] > a){
       arr[i+1][0]=arr[i][0];
       arr[i+1][1]=arr[i][1];
       i--;
    }
    arr[i+1][0]=a;
    arr[i+1][1]=b;
  }
```

```
}

/* FUNCTION: bool_add ************************************************************/
/* Adds regions to be painted for the case of a rectangular spray pattern and */
/* no offset curve                                                              */
/*******************************************************************************/
void bool_add(A,asize,B,bsize)
double *A,*B;
int asize,bsize;
{
  int a=0,b=0,AA=0,BB=0,value,i,start,count=0;
  double result[50][2],S,xval;

  /* SET START POINT (S) */
  value = next_x(A,asize,a,B,bsize,b,&S);
    if(value == 1){
      AA = 1;
      a++;}
    if(value == 2){
      BB = 1;
      b++;}

   start = 0;

  /* LOOP THROUGH ARRAY VALUES SEQUENTIALLY */
  while(value = next_x(A,asize,a,B,bsize,b,&xval)){
    if(start){ S = xval; start = 0;}
    if(value == 1){
      if(AA) AA = 0; else AA = 1;
      a++;}
    if(value == 2){
      if(BB) BB = 0; else BB = 1;
      b++;}

    if(!AA && !BB){
      result[count][0] = S;
      result[count][1] = xval;
      count++;
      start = 1;}

  } /* while */
  ads_printf("\nFinal Path");
  for(i=0;i<count;i++)
    ads_printf("\n%.3lf, %.3lf",result[i][0],result[i][1]);

} /* function */

/* FUNCTION: next_x **************************************************************/
/* Given two vectors, their sizes and the current index this function find the*/
/* next highest value from either array and places the result in xval          */
/*******************************************************************************/
int next_x(A,asize,a,B,bsize,b,xval)
int a,b,asize,bsize;
double *A,*B,*xval;
{
  if(a < asize && b < bsize)
    if(A[a] < B[b]){
      *xval = A[a];
      return(1);}
    else{
```

```
      *xval = B[b];
      return(2);}
  if(a < asize && b >= bsize){
    *xval = A[a];
    return(1);}

  if(a >= asize && b < bsize){
    *xval = B[b];
    return(2);}

  if(a >= asize && b >= bsize)
    return(0);
}

/* FUNCTION: AREA ************************************************************/
/* GIVEN AN NX3 ARRAY OF COORDINATES DESCRIBING AN ENCLOSED POLYGON THIS      */
/* FUNCTION WILL DETERMINE ITS ENCLOSED AREA                                  */
/*****************************************************************************/
double area(polygon,size)
long size;        /* SIZE (NUMBER OF ROWS) OF THE n x 3 ARRAY "polygon" */
double **polygon; /* POINTER TO THE n x 3 ARRAY DESCRIBING ENCLOSED POLYGON */
{
  long i;
  double area=0.0,AREA=0.0;
  double x1,x2,y1,y2;

  for(i=0;i< size-1;i++){
    x1 = polygon[i][X];
    x2 = polygon[i+1][X];
    y1 = polygon[i][Y];
    y2 = polygon[i+1][Y];
    area = x1*(y2-y1) + (y2-y1)*(x2-x1)/2.0;
    AREA = AREA + area;
  }
  return(AREA);
}
```

B.6 Elpsoffset.c

This module contains routines that determine the elliptical offset curve of a given workpiece.

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359
#define LEFT -1
#define RIGHT 1

/* DECLARE FUNCTIONS */
void ellipse_offset  _((boundary,size,side,outfile));
void intersect1    _((p1,p2,p3,side,offset,I));

/* FUNCTION: ellipse_offset **************************************************/
```

```
/*****************************************************************************/
void ellipse_offset(boundary,size,side,a,b,outfile)
double **boundary,a,b;
int size,side;
char *outfile;
{
  int i,curve;
  double p1[2],p2[2],p3[2];
  double I[2];
  double S[2],E[2],SA,DA;
  double C[2],B[2],ang;
  FILE *fp;

  fp = fopen(outfile,"w");

  /* INITIALIZE FIRST THREE POINTS */
  p1[X]=boundary[0][X];
  p1[Y]=boundary[0][Y];
  p2[X]=boundary[1][X];
  p2[Y]=boundary[1][Y];
  p3[X]=boundary[2][X];
  p3[Y]=boundary[2][Y];

  /* LOOP THROUGH BOUNDARY AND DETERMINE OFFSET VERTICES */
  for(i=3;i <= size+1 ;i++){

    /* DETERMINE IF SEGMENT 2 CURVES LEFT OR RIGHT (USING CROSS PRODUCT AxB) */
    if( ((p2[X]-p1[X])*(p3[Y]-p2[Y])-(p2[Y]-p1[Y])*(p3[X]-p2[X])) > 0)
      curve=LEFT;
    else
      curve=RIGHT;

    intersect1(p1,p2,p3,side,a,b,I);
    if(i>3){    /* FIRST PASS THROUGH THE LOOP ? */
       fprintf(fp,"\nline %.4f %.4f %.4f %.4f",S[X],S[Y],I[X],I[Y]);
       S[X]=I[X];
       S[Y]=I[Y];
      }
      else{ /* IF IT IS FIRST TIME THROUGH SET CLOSE POINT AND START POINT */
        C[X]=S[X]=I[X];
        C[Y]=S[Y]=I[Y];
      }

    /* INCREMENT VERTICES */
    p1[X]=p2[X];
    p1[Y]=p2[Y];
    p2[X]=p3[X];
    p2[Y]=p3[Y];

    if(i < size){
      p3[X]=boundary[i][X];
      p3[Y]=boundary[i][Y];
    }
    else if(i==size){  /* PREPARE FOR FINAL LOOP */
      p3[X]=boundary[1][X];
      p3[Y]=boundary[1][Y];
    }
    else if(i > size){
      fprintf(fp,"\nline %.4lf %.4lf %.4lf %.4lf",S[X],S[Y],C[X],C[Y]);
    }
```

```
  }
  fclose(fp);
}

/* FUNCTION: INTERSECT1 ***************************************************/
void intersect1(p1,p2,p3,side,a,b,I)
int side;
double *p1,*p2,*p3,*I;
double a,b;
{
  double p4[2],p6[2];
  double L1,L2,t;
  double slope,theta,gamma,phi;
  double x,y,l,d_A,d_B;

  /* CALCULATE THE LENGTHS OF EACH LINE */
  L1=sqrt((p2[X]-p1[X])*(p2[X]-p1[X])+(p2[Y]-p1[Y])*(p2[Y]-p1[Y]));
  L2=sqrt((p3[X]-p2[X])*(p3[X]-p2[X])+(p3[Y]-p2[Y])*(p3[Y]-p2[Y]));

  /* CALCULATE THE OFFSET OF LINE A */
  if((slope=fabs((p2[Y]-p1[Y])/(p2[X]-p1[X]))) > 10000.0)
    d_A=a;
  else{
    theta = atan(slope);
    x = a*a*slope/sqrt(b*b+a*a*slope*slope);
    y = b*sqrt(1-x*x/a/a);
    l = sqrt(x*x+y*y);
    phi = atan(x/y);
    gamma = theta-phi;
    d_A = l*cos(gamma);
  }

  /* CALCULATE THE OFFSET OF LINE B */

  if((slope=fabs((p3[Y]-p2[Y])/(p3[X]-p2[X]))) > 10000.0)
    d_B=a;
  else{
    theta = atan(slope);
    x = a*a*slope/sqrt(b*b+a*a*slope*slope);
    y = b*sqrt(1-x*x/a/a);
    l = sqrt(x*x+y*y);
    phi = atan(x/y);
    gamma = theta-phi;
    d_B = l*cos(gamma);
  }

  /* CALCULATE THE PRELIMINARY START POINT OF EACH OFFSET LINE */
  p4[X]=p1[X]+d_A*side*(p2[Y]-p1[Y])/L1;
  p4[Y]=p1[Y]-d_A*side*(p2[X]-p1[X])/L1;

  p6[X]=p2[X]+d_B*side*(p3[Y]-p2[Y])/L2;
  p6[Y]=p2[Y]-d_B*side*(p3[X]-p2[X])/L2;

  /* CALCULATE PARAMETER FOR INTERSECTION POINT */
  t=L1*((p3[Y]-p2[Y])*(p6[X]-p4[X])+(p3[X]-p2[X])*(p4[Y]-p6[Y]))/
    ((p2[X]-p1[X])*(p3[Y]-p2[Y])-(p2[Y]-p1[Y])*(p3[X]-p2[X]));

  I[X]=p4[X]+t*(p2[X]-p1[X])/L1;
  I[Y]=p4[Y]+t*(p2[Y]-p1[Y])/L1;
}
```

B.7 Spray3.c

This module contains routines that determine the paint on and paint off distances for a particular spray path

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359
#define LEFT -1
#define RIGHT 1
#define ON 1
#define OFF 0

void spray_path3    _((work,offset,size,b,s,outfile,INITMODE));

/* FUNCTION: SPRAY_PATH3 ************************************************/
/* Inputs: pointer to array describing workpiece                        */
/*         pointer to array describing offset curve                     */
/*         dimensions of the spray pattern ellipse                      */
/*         desired overlap                                              */
/*         name of output file                                          */
/* Output: spray path geometry                                          */
/* Characteristics:                                                     */
/*         spray is turned off for workpiece voids                      */
/*         does not handle islands                                      */
/************************************************************************/
void spray_path3(work,offset,size,b,s,outfile,INITMODE)
int INITMODE; /* TOGGLE FOR INITIAL PASS HEIGHT DETERMINATION METHOD */
double **work,**offset;
long size;
double b,s;
char *outfile;
{
  int i,count=0,ips,spraying_right=1,paint;
  double higho,lowo,highw,loww,y,*iplist;
  double  hx,hy;
  double c; /* offset from center of deposition profile for reqd thickness */
  double test;
  FILE *fp;

  /* LOCATE THE HIGHEST AND LOWEST POINT ON THE WORKPIECE */
  highw = work[0][Y];
  loww = work[0][Y];

  for(i=1;i < size;i++){
    if(work[i][Y] > highw)
      highw = work[i][Y];
    if(work[i][Y] < loww)
      loww = work[i][Y];
  }

  /* LOCATE THE HIGHEST AND LOWEST POINT ON THE OFFSET CURVE */
  higho = offset[0][Y];
  lowo = offset[0][Y];

  for(i=1;i < size;i++){
```

```
    if(offset[i][Y] < lowo)
      lowo = offset[i][Y];
    if(offset[i][Y] > higho)
      higho = offset[i][Y];
  }

  /* DETERMINE MINIMUM HEIGHT TO START SPRAYING AT */
  if(INITMODE)
    c=0.75*b;
  else
    c=b*sqrt(1.0-(pi*b/2.0/s)*(pi*b/2.0/s));

  /* SET PASS_CENTER */
  y = highw-c;

  /* OPEN DATA FILE */
  fp=fopen(outfile,"w");

  test = loww -s + c;
  /* MAIN PRINTING LOOP ******************************************/

  while(y >= test){
    iplist = intloop2(offset,size,y,&ips);

    if(spraying_right){
      if(count) fprintf(fp,"\nline %.6f %.6f %.6f %.6f",hx,hy,iplist[0],y);
      else count=1;
      paint=ON;
      for(i=0;i<ips-1;i++){
        if(paint) {fprintf(fp,"\nsystem painton"); paint=OFF;}
        else { fprintf(fp,"\nsystem paintoff"); paint=ON;}
        fprintf(fp,"\nline %.6f %.6f %.6f %.6f",iplist[i],y,iplist[i+1],y);
      }
      fprintf(fp,"\nsystem paintoff");
      hx = iplist[ips-1];
      hy = y;
    }
    else{
      fprintf(fp,"\nline %.6f %.6f %.6f %.6f",hx,hy,iplist[ips-1],y);
      paint=ON;
      for(i=ips-1;i>=1;i--){
        if(paint) {fprintf(fp,"\nsystem painton"); paint=OFF;}
        else {fprintf(fp,"\nsystem paintoff"); paint=ON;}
        fprintf(fp,"\nline %.6f %.6f %.6f %.6f",iplist[i],y,iplist[i-1],y);
      }
      fprintf(fp,"\nsystem paintoff");
      hx = iplist[0];
      hy = y;
    }

    /* RESET PASS_CENTER */
    y = y - s;

    /* RESET SPRAY DIRECTION */
    if(spraying_right) spraying_right=0; else spraying_right=1;
  }
  fclose(fp);
}
```

B.8 Spray4.c

This module contains routines that determine the paint on and paint off distances for a particular spray path

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359
#define LEFT -1
#define RIGHT 1
#define ON 1
#define OFF 0

void spray_path4    _((work,offset,size,s,&Xi,&Xo,INITMODE));

/* FUNCTION: SPRAY_PATH4 ************************************************/
/* Inputs: array describing a workpiece boundary                        */
/*         array describing offset boundary                             */
/*         dimensions of the spray pattern ellipse                      */
/*         desired overlap                                              */
/*         name of output file                                          */
/* Output: Paint ON and Paint OFF distances                             */
/* Characteristics:                                                     */
/*         spray is turned off for workpiece voids                      */
/*         does not handle islands                                      */
/************************************************************************/
void spray_path4(work,offset,size,b,s,Xi,Xo,INITMODE)
double **work;   /* POINTER TO WORKPIECE ARRAY */
double **offset; /* POINTER TO OFFSET CURVE ARRAY */
long size;     /* NUMBER OF VERTICES IN WORK AND OFFSET ARRAY */
double s;      /* PATH INTERVAL */
double b;      /* ELLIPSE MAJOR AXIS */
double *Xi;    /* PAINT ON DISTANCE */
double *Xo;    /* PAINT OFF DISTANCE */
int INITMODE; /* TOGGLE FOR INITIAL PATH HEIGHT DETERMINATION METHOD */

{
  int i,count=0,ips,spraying_right=1,paint;
  int INITMODE=1;    /* TOGGLE FOR INITIAL PATH HEIGHT DETERMINATION METHOD */
  double highw,loww; /* DEFINE WORKPIECE EXTREME Y COORDINATES */
  double y;          /* DENOTES PATH CENTER */
  double *vlist;     /* VLIST IS LIST OF INTERSECTIONS POINT X COORDINATES*/
  double hx,hy;      /* END OF SPRAY PATH COORDINATES */
  double c;          /* OFFSET FROM PATH CENTER WHERE h>=h_specified */
  double test;       /* LOOPING CONDITION VARIABLE */

  /* LOCATE THE HIGHEST AND LOWEST POINTS ON THE WORKPIECE */
  highw = work[0][Y];
  loww = work[0][Y];
  for(i=1;i < size;i++){
    if(work[i][Y] > highw)
      highw = work[i][Y];
    if(work[i][Y] < loww)
      loww = work[i][Y];
  }
```

```
  /* CALCULATE c, THE DISTANCE FROM THE SPRAY PATH CENTER WITHIN WHICH  */
  /* THE PAINT THICKNESS IS >= SPECIFIED PAINT THICKNESS */
  if(INITMODE)
    c=0.75*b;
  else
    c=b*sqrt(1.0-(pi*b/2.0/s)*(pi*b/2.0/s));

  /* SET PASS_CENTER */
  y = highw-c;

  /* SET LOOP CONDITION */
  test = loww -s + c;

  /* MAIN LOOP ***************************************************/
  while(y >= test){

    vlist=intloop2(offset,size,y,&ips);
    /* CALCULATE DISTANCES */
    paint=ON;

    /* THIS LOOP SUMS THE PAINT ON AND PAINT OFF DISTANCES ALONG 1 PASS */
    for(i=0;i<ips-1;i++){
      if(paint){*Xi=*Xi+fabs(vlist[i]-vlist[i+1]); paint=OFF;}
      else     {*Xo=*Xo+fabs(vlist[i]-vlist[i+1]); paint=ON; }
    }

    /* THIS LOOP DETERMINES THE TRANSFER DISTANCE FROM THE PREVIOUS PASS */
    /* AND ADDS IT TO THE SPRAY OFF DISTANCE VARIABLE, Xo.               */
    if(spraying_right){
      if(count) *Xo=*Xo+sqrt(pow(hx-vlist[0],2.0)+pow(hy-y,2.0));
      hx=vlist[ips-1];
      hy=y;
      spraying_right=OFF;
    }
    else{
      if(count) *Xo=*Xo+sqrt(pow(hx-vlist[ips-1],2.0)+pow(hy-y,2.0));
      hx=vlist[0];
      hy=y;
      spraying_right=ON;
    }
    y=y-s;
   count++;
  }
}
```

B.9 Edata.c

This module contains routines that determine chacteristics of database entities.

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359
#define LEFT -1
```

```
#define RIGHT 1

void entity_type   _((entity));
void closest_end   _((entity,pt1,pt2));
void furthest_end  _((entity,pt1,pt2));
void line_data     _((entity,start,end));
int arc_data       _((entity,center,start,end,radius,start_angle,end_angle));
void entity_search   _((point,tolerance,temp_ss));

/* FUNCTION: ***********************************************************/
void furthest_end(entity,pt1,pt2)
ads_name entity;
ads_point pt1,pt2;
{
  char etype[10];
  double radius,sa,ea;
  ads_point start,end;

  entity_type(entity,etype);

  if(!strcmp(etype,"LINE") || !strcmp(etype,"line"))
    line_data(entity,start,end);

  if(ads_distance(pt1,start) <= ads_distance(pt1,end))
    equate_point(pt2,end);
  else
    equate_point(pt2,start);
}

/* FUNCTION: ***********************************************************/
void closest_end(entity,pt1,pt2)
ads_name entity;
ads_point pt1,pt2;
{
  char etype[10];
  double radius,sa,ea;
  ads_point start,end;

  entity_type(entity,etype);

  if(!strcmp(etype,"LINE") || !strcmp(etype,"line"))
    line_data(entity,start,end);

  if(ads_distance(pt1,start) <= ads_distance(pt1,end))
    equate_point(pt2,start);
  else
    equate_point(pt2,end);
}

/* FUNCTION: ***********************************************************/
void line_data(entity,start,end)
ads_point start,end;
{
  struct resbuf *eb;

  /* OBTAIN THE DXF LISTING OF THE DESIRED ENTITY */
  eb = ads_entget(entity);

  /* LOOP THROUGH THE DXF LISTING UNTIL THE DXF GROUP CODE IS ZERO */
  while(eb->restype != 10)  eb = eb->rbnext;
```

```
  start[X] = eb->resval.rpoint[X];
  start[Y] = eb->resval.rpoint[Y];
  start[Z] = eb->resval.rpoint[Z];

  eb=eb->rbnext;

  end[X] = eb->resval.rpoint[X];
  end[Y] = eb->resval.rpoint[Y];
  end[Z] = eb->resval.rpoint[Z];
}

/* FUNCTION: arc_data ************************************************************/
/* Given the entity name of an arc this function determines the arc           */
/* characteristics and returns them to the calling function                   */
/*****************************************************************************/
int arc_data(entity,center,start,end,radius,start_angle,end_angle)
ads_name entity;
double *center,*start,*end,*radius,*start_angle,*end_angle;
{
  struct resbuf *eb;

  /* OBTAIN THE DXF LISTING OF THE DESIRED ENTITY */
  eb = ads_entget(entity);

  /* LOOP THROUGH THE DXF LISTING UNTIL THE GROUP CODE IS 10 (CENTER POINT) */
  while(eb->restype != 10)
    eb=eb->rbnext;

  /* COPY THE CENTER VALUE INTO THE DATA PTR PROVIDED */
  center[X] = eb->resval.rpoint[X];
  center[Y] = eb->resval.rpoint[Y];
  center[Z] = eb->resval.rpoint[Z];

  /* LOOP THROUGH THE DXF LISTING UNTIL THE GROUP CODE IS 40 (RADIUS) */
  while(eb->restype != 40)
    eb=eb->rbnext;

  /* COPY THE RADIUS VALUE INTO THE DATA PTR PROVIDED */
  *radius = eb->resval.rreal;

  /* LOOP THROUGH THE DXF LISTING UNTIL THE GROUP CODE IS 50 (START ANGLE) */
  while(eb->restype != 50)
    eb=eb->rbnext;

  /* COPY THE START ANGLE INTO THE DATA PTR PROVIDED */
  *start_angle = eb->resval.rreal;

  /* LOOP THROUGH THE DXF LISTING UNTIL THE GROUP CODE IS 51 (END ANGLE) */
  while(eb->restype != 51)
    eb=eb->rbnext;

  /* COPY THE END ANGLE INTO THE DATA PTR PROVIDED */
  *end_angle = eb->resval.rreal;

  /* CALCULATE START AND END POINTS */
  start[X]=center[X] + *radius * cos(*start_angle);
  start[Y]=center[Y] + *radius * sin(*start_angle);
  start[Z]=center[Z];
```

```
  end[X]=center[X] + *radius * cos(*end_angle);
  end[Y]=center[Y] + *radius * sin(*end_angle);
  end[Z]=center[Z];

  return(RTNORM);
}

/* FUNCTION: ************************************************************/
void entity_search(point,tolerance,temp_ss)
ads_point point;
double tolerance;
ads_name temp_ss;
{
  ads_point pt1,pt2;

  pt1[X] = point[X] - tolerance;
  pt1[Y] = point[Y] - tolerance;
  pt1[Z] = 0.0;

  pt2[X] = point[X] + tolerance;
  pt2[Y] = point[Y] + tolerance;
  pt2[Z] = 0.0;

  ads_ssget("C",pt1,pt2,NULL,temp_ss);
}

/* FUNCTION: ************************************************************/
void entity_type(entity,etype)
ads_name entity;
char *etype;
{
  struct resbuf *eb;

  /* OBTAIN THE DXF LISTING OF THE DESIRED ENTITY */
  eb = ads_entget(entity);

  /* LOOP THROUGH THE DXF LISTING UNTIL THE DXF GROUP CODE IS ZERO */
  while(eb->restype != 0)  eb = eb->rbnext;

  /* COPY THE ENTITY TYPE INTO ETYPE */
  strcpy(etype,eb->resval.rstring);
}
```

B.10 File.c

This module contains routines that manipulate files and arrays.

```
#include  <stdio.h>
#include  <string.h>
#include  <math.h>
#include  "adslib.h"

#define pi 3.14159265359
#define LEFT -1
#define RIGHT 1
```

```
void file_entity        _((entity,current_entity,point));
double **file_to_array  _((file,data,size));

/* FUNCTION: ***********************************************************/
void file_entity(fp,current_entity,point)
FILE *fp;
ads_name current_entity;
ads_point point;
{
  char etype[10];
  static int j=0;
  int i;
  ads_point temp,start,end,center,v1,v2,result1;
  double radius,sa,ea,da,result2;

  /* DETERMINE ENTITY TYPE */
  entity_type(current_entity,etype);

  /* ENTITY TYPE == LINE */
  if(!strcmp(etype,"LINE") || !strcmp(etype,"line")){
    furthest_end(current_entity,point,temp);
    if(j) fprintf(fp,"\n");
    fprintf(fp,"line %.8lf %.8lf %.8lf %.8lf",point[X],point[Y],
    temp[X],temp[Y]);
    j++;
  }

  /* ENTITY TYPE == ARC */
  if(!strcmp(etype,"ARC")){
    arc_data(current_entity,center,start,end,&radius,&sa,&ea);

    /* CREATE VECTORS V1 AND V2 TO CHECK DELTA ANGLE */
    for(i=0;i<=2;i++){
    v1[i]=start[i]-center[i];
    v2[i]=end[i]-center[i];
    }

    cross_product(v1,v2,result1);
    dot_product(v1,v2,&result2);

    if(result1[Z] < 0.0)
      da = 2.0*pi - acos(result2/(radius*radius));
    else
      da = acos(result2/(radius*radius));

    if(!test_point_equality(point,start,0.0001)){
      da = -da;
      sa=ea;
    }

    fprintf(fp,"\narc %.8lf %.8lf %.8lf",center[X],center[Y],radius);
    fprintf(fp," %.8lf %.8lf",sa,da);
  }
}

/* FUNCTION: file_to_array *********************************************/
/***********************************************************************/
double **file_to_array(boundary,size)
char *boundary;  /* NAME OF PART DESCRIPTION FILE */
long *size;
```

```
{
  char etype[10];
  int j=0;
  long i=0;
  double **data;
  ads_point start,end;
  FILE *fp;

  /* DETERMINE REQUIRED SIZE OF ARRAY */
  fp = fopen(boundary,"r");
  while(fscanf(fp,"%s",etype) != EOF){
    if(!strcmp(etype,"line") || !strcmp(etype,"LINE")){
      i++;
      lseek(fp,4*sizeof(double),1);
    }
  }
  *size=i+1;

  /* ALLOCATE MEMORY FOR ARRAY */
  data = (double **)malloc((unsigned)(*size) * sizeof(double *));
  for(i=0;i < *size;i++)
    data[i] = (double *)malloc(3*sizeof(double));

  /* RESET FILE POINTER AND READ IN FIRST LINE DATA */
  rewind(fp);
  fscanf(fp,"%s",etype);

  if(!strcmp(etype,"line") || !strcmp(etype,"LINE")){

    fscanf(fp,"%lf %lf %lf %lf",&start[X],&start[Y],&end[X],&end[Y]);
    data[0][X] = start[X];
    data[0][Y] = start[Y];
    data[0][Z] = 0.0;
    data[1][X] = end[X];
    data[1][Y] = end[Y];
    data[1][Z] = 0.0;
  }

  /* LOOP THROUGH FILE AND ASSIGN VALUES TO ARRAY */
  for(i=2;i < *size;i++){
    fscanf(fp,"%s",etype);
    if(!strcmp(etype,"line") || !strcmp(etype,"LINE")){
      fscanf(fp,"%lf %lf %lf %lf",&start[X],&start[Y],&end[X],&end[Y]);
      data[i][X] = end[X];
      data[i][Y] = end[Y];
      data[i][Z] = 0.0;
    }
  }
  fclose(fp);
  return(data);
}
```

VITA

Surname: Bartlett

Given Names: Cyril James

Place of birth: Montreal, Quebec, Canada

Date of Birth: 16 March 1958

Educational Institutions Attended:

University of Victoria	1990 to 1992
University of Manitoba	1980 to 1985

Degree Obtained:

B.Sc.(M.E.)	Unitversity of Manitoba	1985

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis: Path Generation for Planar Spray Painting

Author: [redacted]

Cyril James Bartlett

23 December 1992