UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Geração de estratégias de medição de superfícies complexas em sistema CAD para máquinas de medir por coordenadas

Henrique Neves de Lucena
Orientador: Prof. Dr.-Ing Klaus Schützer

Santa Bárbara d'Oeste, SP

2009

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Geração de estratégias de medição de superfícies complexas em sistema CAD para máquinas de medir por coordenadas

Henrique Neves de Lucena
Orientador: Prof. Dr.-Ing Klaus Schützer

Dissertação de Mestrado apresentada no Programa de Pós-Graduação em Engenharia de Produção da Faculdade de Engenharia, Arquitetura e Urbanismo, da Universidade Metodista de Piracicaba – UNIMEP.

Santa Bárbara d'Oeste, SP

2009

Agradecimentos

Ao Professor Dr.-Ing. Klaus Schützer pela orientação e incentivo para a conclusão deste trabalho, obrigado por ter confiado em mim.

Aos companheiros de trabalho do Laboratório de Sistemas Computacionais para Projeto e Manufatura (SCPM) pelo apoio, incentivo e orientação.

À CAPES - Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pela concessão da bolsa de estudos.

Ao meu grande amigo Antonio Álvaro de Assis Moura, por toda a ajuda e companheirismo em todos os momentos do meu trabalho.

Aos meus muitos amigos que sempre me ajudaram e estiveram ao meu lado, em especial, Carlos Miralles, Nilton Júnior e Rafael Assim, e todos os outros que não haveria possibilidades de inserir aqui, mas que vivem sempre em minha lembrança.

À minha família, pelo apoio que tive durante toda a minha vida e os princípios que me passaram e que carrego até hoje. Sem vocês eu não estaria aqui.

À minha querida Valéria Orlandi, pela dedicação, apoio e ajuda de revisão dos textos.

Ao meu querido mestre Kykwo Syllwa, que sempre me guiou e ajudou em todos os momentos, ajudando a me tornar antes de tudo, em um homem de bem.

À Deus acima de todas as coisas, que nunca me deixou nem por um momento em toda a minha vida.

“Procure ser uma pessoa de valor, em vez de procurar ser uma pessoa de sucesso. O sucesso é consequência...”

Albert Einstein

Sumário

Sumário III
Lista de Figuras V
Lista de Tabelas VIII
Lista de Tabelas VIII
Lista de Abreviações IX
Resumo X
Abstract XI
1. Introdução 1
2. Revisão Bibliográfica 4
2.1 Máquina de medir por coordenadas 5
2.1.1 Importância da medição 9
2.1.2 Diferença entre a tecnologia de medição por coordenadas e os métodos convencionais 12
2.1.3 Requisitos para utilização do sistema de medição por coordenadas 15
2.1.4 Influência do operador no resultado da medição 17
2.2 Curvas paramétricas 19
2.2.1 Curva Bézier 22
2.2.2 Curva B-Spline 24
2.2.3 NURBS 31
2.3 CAD (Computer Aided Design) 34
2.3.1 CAD 3D 35
2.3.1.1 Modeladores sólidos 36
2.3.1.2 Modeladores de superfícies 37
2.3.2 Interface de Programação 38
2.3.2.1 API 39
2.3.3 Vantagens e desvantagens do sistema CAD 43

2.4 CAI (Computer-Aided Inspection) 44
2.4.1 Integração CAD-CAI 46
2.4.2 Ferramentas de ajuda de um sistema CAI 49
2.4.3 Representação Gráfica 52
2.4.4 Alinhamento da peça em relação ao modelo virtual 53
3. Objetivo e Métodos 56
3.1 Objetivos 56
3.2 Métodos 57
4. Desenvolvimento e implementação do aplicativo 59
4.1 Desenvolvimento de aplicativos 59
4.2 Desenvolvimento de aplicativo em sistema CAD para definição de pontos de inspeção 66
4.2.1 Lógica de funcionamento do aplicativo 69
4.2.2 Estratégia de distribuição pelo parâmetro “t” 73
4.2.3 Estratégia de distribuição pelo comprimento da curva 73
4.3 Exportação de dados para o sistema CAI 77
4.4 Interface com o usuário 79
5. Conclusões 85
6. Referências 87
Anexos 93
Anexo A – Código Fonte do aplicativo desenvolvido 93
Anexo B – Cálculo do exemplo da figura 4.1. 107
Anexo C – Função do UGOPEN para obtenção dos dados da curva. 110
Anexo D – Estrutura de dados da curva. 111
Anexo E – Função do UGOPEN para obtenção do comprimento de curva. 113
Anexo F – Exemplo de arquivo DMIS gerado pelo aplicativo. 114

Lista de Figuras

Figura 2.1: Fluxo de utilização do aplicativo..... 5

Figuras 2.2: Componentes de um sistema de medição por coordenadas (Laboratório de Metrologia, UNIMEP). 6

Figura 2.3: Modelos de sistemas de apalpação: tipo (a) Comutador e (b) Analógico [5]..... 8

Figura 2.4: Qualificação de um apalpador por meio de uma esfera padrão (Laboratório de Metrologia, UNIMEP). 8

Figura 2.5: Realimentação baseada nos resultados de inspeção MMC [14]..... 10

Figura 2.6: Resultados de tomadas de decisões baseadas em uma má inspeção [16]. 11

Figura 2.7: Exemplo de peças inspecionadas por meio de uma MMC [20]..... 12

Figura 2.8: Influência causada por elementos externos à inspeção [10]..... 16

Figura 2.9: Comparação gráfica dos itens que influenciam a medição [27]. 18

Figura 2.10: Influência da estratégia de distribuição de pontos..... 19

Figura 2.11: Exemplo de curva com pontos e derivadas definidas. 20

Figura 2.12: Exemplo de curva fechada ou com um laço..... 21

Figura 2.13: Exemplo de curva que passa por pontos dados..... 21

Figura 2.14: Representação de uma curva tridimensional parametrizada (a) e pelo modelo matemático de Lagrange (b). 22

Figura 2.15: Representação da Curva de Bézier. 23

Figura 2.16: Funções de Passo..... 26

Figura 2.17: Diagrama do esquema de Computação Triangular..... 27

Figura 2.18: Representação de vetores uniformes e não uniformes [36]. 28

Figura 2.19: Exemplo de tipos de curva B-Spline: (a)Periódica e (b)Aberta. 29

Figura 2.20: Segmento de Curvas Polinomiais na Curva B-Spline [36]. 29

Figura 2.21: Exemplos de continuidade de curvas: continuidade G^0, G^1 e C^1..... 30

Figura 2.22: Curva NURBS definida com diferentes pesos (weight)..... 33

Figura 2.23: Exemplo de utilização de operações Booleanas em modelador sólido. 37

Figura 2.24: Edição de superfícies através da alteração dos pontos de construção. 38

Figura 2.25: Tela de desenvolvimento de interface visual (Siemens NX). 41

Figura 2.26: Exemplo de arquivos gerados no desenvolvimento de uma caixa de diálogo (Siemens NX). 42

Figura 2.27: Exemplo de estrutura de programação de um caixa de diálogo. 43

Figura 2.28: Exemplo de utilização de ferramenta de auxílio (PC-DIMS). 50

Figura 2.29: Seleção de sistema de apalpação (PC-DIMS). 52

Figura 2.30: Criação de alinhamento. 54

Figura 2.31: Criação de alinhamento referente ao modelo CAD. 55

Figura 4.1: Acesso ao User Interface Styler. 60

Figura 4.2: Janela de construção do User Interface Styler. 61

Figura 4.3: Seqüência de criação de novo projeto. 62

Figura 4.4: Inclusão de arquivos no projeto 62

Figura 4.5: Configuração do programa como executável. 63

Figura 4.6: Configuração das bibliotecas do API 64

Figura 4.7: Texto contendo informações para a criação do menu 65

Figura 4.8: Configuração de Variáveis de Ambiente. 66

Figura 4.9: Influência dos polígonos de controle no cálculo de posicionamento dos pontos. 68

Figura 4.10: Cálculo de primeira ordem de Rbasis. 69

Figura 4.11: Cálculo das demais ordens de Rbasis. 70

Figura 4.12: Normalização dos resultados. 70

Figura 4.13: Cálculo dos pontos. 71

Figura 4.14: Exemplo de cálculo de ponto e suas funções Rbasis. 72

Figura 4.15: Estratégia de distribuição pelo parâmetro “t” da curva. 73

Figura 4.16: Estratégia de distribuição pelo comprimento da curva. 74

Figura 4.17: 1º Parte de exemplo de cálculo de ponto (distribuição pelo comprimento). 75

Figura 4.18: 2º Parte de exemplo de cálculo de ponto (distribuição pelo comprimento). 76

Figura 4.19: Criação de vetores para a estratégia de distribuição pelo comprimento da curva. 77

Figura 4.20: Exemplo de ponto gerado no arquivo DMIS. 77

Figura 4.21: Caminho de importação de arquivo DMIS (Software PC-DIMS). 78

Figura 4.22: Interface com o usuário. 79

Figura 4.23: Seqüência de uso do aplicativo. 80

Figura 4.24: Janela de opções do aplicativo. 81

Figura 4.25: Janela de seleção do caminho de medição (curva). 81

Figura 4.26: Janela de seleção do sólido. 82

Figura 4.27: Janela de seleção da face. 83

Figura 4.28: Criação de pontos e vetores. 84

Lista de Tabelas

Tabela 2.1: Diferenças entre os métodos de medição [5]. 14

Tabela 2.2: Recursos geométricos contidos nas interfaces normalizadas [72]. 49

Tabela-Anexo B: Tabela de cálculo de coordenadas dos pontos relacionados à Figura 4.9. 109

Lista de Abreviações

ANSI - American National Standard Institute

API - Application Programming Interface

BRep - Boundary Representation

CAA - Computer Aided Accuracy

CAD - Computer Aided Design

CAE - Computer Aided Engineering

CAI - Computer Aided Inspection

CAM - Computer Aided Manufacturing

CSG - Constructive Solid Geometry

CNC - Computer Numerical Control

DES - Data Encryption Standard

DIMS - Dimensional Measuring Interface Specification

DIN - Deutsches Institut für Normung

IGES - Initial Graphics Exchange Specification

ISO - International Organization for Standartization

MMC - Máquina de Medir por Coordenadas

STEP - Standard for the Exchange of Product Model Data

VDA-FS - Verband der Deutschen Automobilindustrie-Flächenschnittstelle

Resumo

Atualmente, as empresas visam atingir o mercado consumidor com produtos dotados de geometrias complexas, seja para satisfazer as exigências em relação a uma estética inovadora, com formas mais harmônicas ou no desenvolvimento de funções específicas que requerem uma forma diferenciada. Visando estes produtos, os processos produtivos necessitam de um constante aprimoramento para atender os padrões de qualidade, mantendo a exatidão geométrica necessária, mas sem esquecer-se sempre de levar em consideração a redução no tempo do ciclo de desenvolvimento de produto. Um método de medição amplamente utilizado na área industrial é a tecnologia de medição por coordenadas, devido a sua exatidão no processo de inspeção, juntamente com a possibilidade de integração com outras áreas do processo produtivo. Visando uma melhor comunicação entre a área de concepção do produto com a de inspeção, este trabalho desenvolve um aplicativo para um sistema CAD que permite a criação de pontos de inspeção para o uso em máquinas de medir por coordenadas. Esses pontos são definidos em uma trajetória de medição numa secção previamente selecionada no modelo virtual. Após a definição dos pontos, os mesmos podem ser exportados juntamente com o modelo CAD para um sistema CAI, onde poderão ser inspecionados, colaborando assim para reduzir o tempo necessário para a geração de pontos de medição, além de diminuir as incertezas obtidas pela falta de conhecimento da funcionalidade da peça pelo operador de uma MMC.

Palavras-chave: Estratégias de Medição, Integração CAD/CAI, Interface de Programação em Sistema CAD.

Abstract

Currently, the companies aim to reach the consumer market with products with complex geometry to satisfy the requirements for an innovative esthetic with more harmonics shapes or the development of specific functions that require a different shape. Aiming these products, the production processes need a constant improvement to reach the quality standards and maintain the necessary geometric accuracy, always considering the time reduction of the product development cycle. A measurement method widely used in industry is the coordinate measuring technology, due to its accuracy in the inspection process together with the possibility of integration for the other areas of the production process. Aiming to a better communication between the product design area with the inspection area, this work develops an application for a CAD system that allows the creation of the inspection points to use in coordinate measuring machine. These points are defined by a measurement trajectory created in the virtual model previously. After the creation of the points, they can be exported with the CAD model for the CAI system that can be inspected and thus help to reduce the time required for the strategy creation, also decreasing the uncertainties obtained by the lack of knowledge of the part function by the CMM's operator.

Keywords: Measuring Strategies, CAD/CAI Integration, Programming Interface in CAD Systems.

1. Introdução

O mercado se encontra atualmente em um ambiente cada vez mais globalizado, onde o sucesso de uma empresa depende que o desenvolvimento de seus produtos seja feito dentro das qualificações que o consumidor exige, em um tempo menor que seus concorrentes [1].

Com a necessidade de se atingir o mercado consumidor, que deseja cada vez formas mais harmoniosas, ou mesmo na fabricação de itens para as indústrias que utilizam um alto grau de complexidade geométrica em seus produtos, como por exemplo, a indústria aeronáutica, automotiva ou de moldes e matrizes, um intensivo desenvolvimento tecnológico nos processos de usinagem vem acontecendo, visando manter a qualidade exigida e ao mesmo tempo obter um ciclo de desenvolvimento do produto reduzido.

Paralelamente, as exigências de conformidade geométrica também tiveram seu aumento, resultando em especificações mais severas de projeto para garantir o elevado desempenho funcional aos quais estes produtos são designados.

Com o advento das máquinas de medir por coordenadas, as geometrias dotadas de formas complexas puderam ser inspecionadas com uma margem de confiabilidade e em um tempo que fosse plausível às empresas [2].

A exigência da exatidão geométrica dos novos produtos, aliada à crescente informatização e integração dos sistemas de manufatura, torna a MMC uma ferramenta de fundamental importância para o controle de qualidade dos produtos, considerando a sua versatilidade, exatidão e velocidade (em relação aos métodos convencionais de aferição) na execução das medições.

A aplicação em larga escala da tecnologia de medição por coordenadas, tornou-se viável com o desenvolvimento principalmente de sistemas para auxiliar o processo de inspeção, ou seja, os sistemas CAI.

Com a utilização de uma MMC, tem-se uma maior praticidade na inspeção de superfícies complexas devido a sua flexibilidade de utilização, oferecendo um

método de obtenção de coordenadas espaciais de um ponto medido, podendo ser comparado com um modelo virtual proveniente de um sistema CAD [3].

Contudo, para a obtenção de resultados satisfatórios na medição, atualmente, ainda depende-se quase que exclusivamente da experiência do operador da MMC. Parâmetros importantes de uma inspeção, como sua velocidade, a quantidade de pontos que a inspeção terá, além da estratégia de medição que será utilizada, são de total responsabilidade do operador.

Este trabalho apresenta como proposta o desenvolvimento de um aplicativo para sistema CAD, a fim de auxiliar a construção de estratégias de medição para posterior uso em Máquina de Medir por Coordenadas. Este aplicativo permite ao usuário definir pontos de medição baseando-se em planos de medição definidos pelo usuário no modelo virtual utilizando conceitos de distribuição diferentes pela região selecionada.

Este trabalho está dividido em sete capítulos que serão detalhados brevemente a seguir.

Capítulo 1 - Introdução - Apresentação do ambiente no qual está inserido este trabalho, apresentando as dificuldades encontradas na área de medição por coordenadas e a proposta que este trabalho considera.

Capítulo 2 - Revisão Bibliográfica - Revisão Bibliográfica sobre a inspeção com máquinas de medir por coordenadas, e uma breve explicação sobre a parametrização de curvas, enfocando os modelos de curva do tipo *Spline*. Em seguida, é apresentada uma revisão sobre sistema CAD, as formas de construção de modelos sólidos e modelos baseados em superfície, juntamente com uma explicação do funcionamento de uma interface de programação em um sistema CAD. Por fim, é discutido o sistema CAI, sua aplicação e a apresentação de ferramentas que facilitam o processo de inspeção.

Capítulo 3 - Objetivos e métodos - Detalhamento dos objetivos do trabalho, divididos entre objetivo e métodos utilizados para o desenvolvimento deste trabalho.

Capítulo 4 - Desenvolvimento do Aplicativo - Apresenta o desenvolvimento do aplicativo mediante a descrição da criação de um aplicativo para sistema CAD, o algoritmo utilizado para o cálculo de distribuição de pontos. Além disso, é demonstrado como foi inserido o aplicativo no sistema CAD *Unigraphics-NX* utilizando sua interface de programação. Também é visto o funcionamento do aplicativo em relação a um modelo virtual, e quais são os fatores necessários para a sua correta utilização, juntamente com a exportação e identificação dos pontos pelo sistema CAI.

Capítulo 5 - Conclusões e sugestões para trabalhos futuros - Conclusão dos resultados obtidos, e sugestões baseadas nos resultados para o uso em futuros trabalhos.

Capítulo 6 - Bibliografia referenciada - Apresenta a listagem dos material bibliográfico em que o projeto de pesquisa está fundamentado.

2. Revisão Bibliográfica

Para o desenvolvimento do aplicativo, diversas áreas foram estudadas por serem necessárias em alguma etapa do desenvolvimento.

Por se tratar de um aplicativo que visa a criação de estratégias de inspeção para um sistema de medição por coordenadas, foi necessário o conhecimento de suas formas de utilização, fatores que o influenciam e o panorama que esta tecnologia tem no mercado atualmente.

Visando a implantação do aplicativo em um sistema CAD, foi necessário um aprofundamento principalmente no funcionamento de seus modelos matemáticos (utilizados para o desenvolvimento de suas funções) e o modo de se introduzir novas funções para a utilização simultânea à do sistema CAD.

A Figura 2.1 ilustra o fluxo de utilização do aplicativo, iniciando no sistema CAD, onde através do modelo matemático de curvas parametrizadas (utilizadas para a construção de formas complexas), o aplicativo permite gerar pontos em um plano de medição definido em um modelo geométrico.

Estes dados são então exportados para um sistema CAI, onde são utilizados para a inspeção do produto finalizado. A seguir, estas áreas serão abordadas em tópicos individuais, facilitando o entendimento geral deste trabalho.

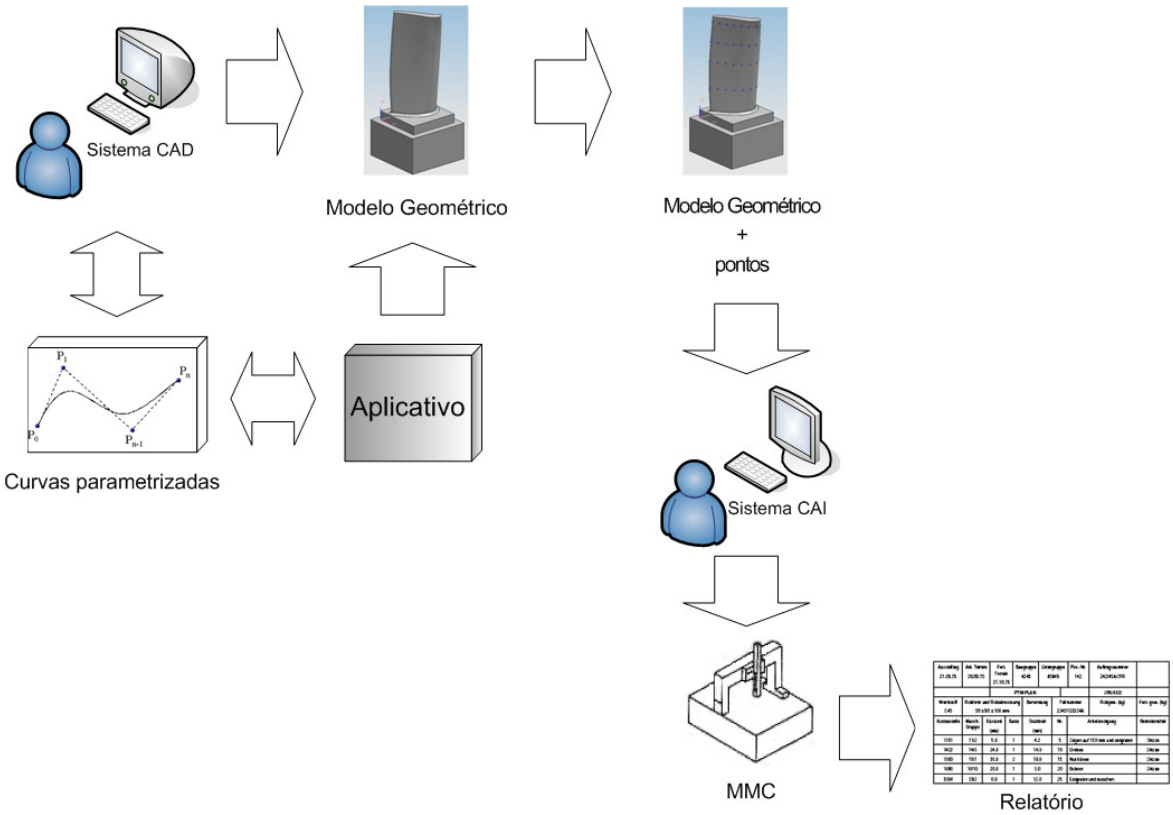


Figura 2.1: Fluxo de utilização do aplicativo.

2.1 Máquina de medir por coordenadas

Por meio de métodos convencionais, é difícil obter uma medição confiável (dentro dos padrões de tolerância especificado) de uma superfície complexa, devido as suas limitações. Devido a isto, as máquinas de medir por coordenadas surgiram como uma solução para esse problema.

Neste tópico serão apresentados os métodos de utilização da MMC, juntamente com seu desempenho metrológico, o custo necessário para aquisição e manutenção, finalizando com suas vantagens perante os outros métodos convencionais.

Uma MMC pode ser dividida basicamente em cinco componentes [4]:

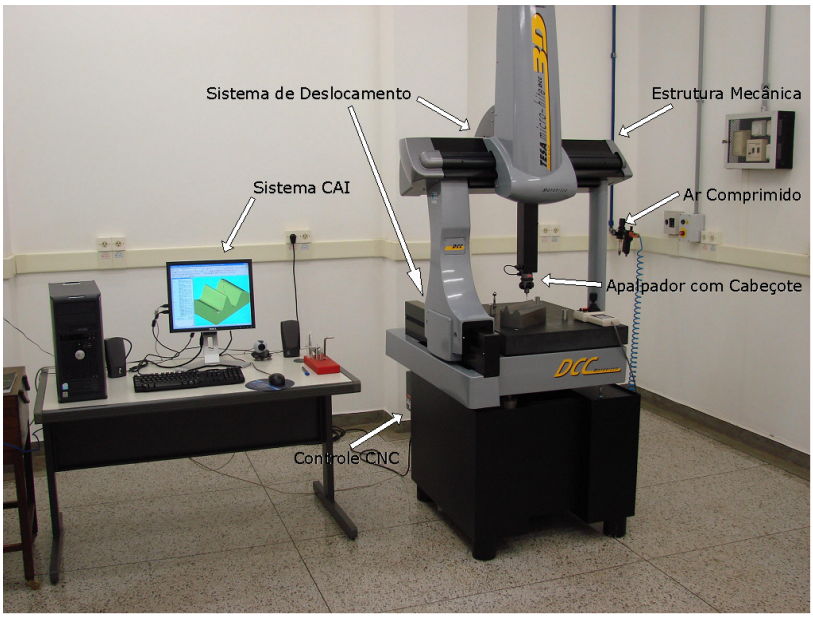
- Estrutura mecânica;
- Sistema de deslocamento;
- Apalpador com cabeçote;

- Controle CNC;
- Software de medição.

A estrutura mecânica básica consiste normalmente de guias de direcionamento, colunas de suporte da máquina, a mesa de medição, sistemas de acionamento e movimentação e o sistema de deslocamento do cabeçote.

A Figura 2.2 exemplifica o posicionamento de cada um desses componentes em uma máquina de medir por coordenada disponível comercialmente. O modelo apresentado é o encontrado no Laboratório de Metrologia da UNIMEP, Campus Santa Bárbara d' Oeste.

O sistema de medição é controlado por escalas eletro-ópticas que conferem uma exatidão dentro da escala de milésimos de milímetros.



Figuras 2.2: Componentes de um sistema de medição por coordenadas (Laboratório de Metrologia, UNIMEP).

De acordo com SOUSA [5], os erros que podem ser relacionados à estrutura mecânica juntamente com o sistema de deslocamento são:

- Erros de perpendicularidade;

- Erros de rotação, onde se dividem em:
    - Erros de rolamento (roll);
    - Erros de guinamento (pitch);
    - Erros de tombamento (yaw).
- Erros de translação, que são subdivididos em:
    - Erros de posicionamento;
    - Erros de retilineidade.

Estes erros são levados em consideração inclusive pelos fabricantes das máquinas de medição por coordenadas e devem ser sempre revistos após a devida instalação no ambiente preparado para a utilização da máquina.

O apalpador juntamente com o cabeçote ao qual ele está acoplado, é denominado de Sistema de Apalpação. Este sistema tem algumas influências significantes no resultado de medição [6].

O nível de influência do Sistema de Apalpação nas incertezas do resultado depende da configuração a ser utilizada de cabeçote/apalpador, e de parâmetros como freqüência de uso e o nível de conservação [7].

Na Figura 2.3 é apresentada a configuração estrutural básica de dois dos mais utilizados tipos de cabeçotes apalpadores, ou seja, os de tipo Comutador ("*touch-trigger*") e o Analógico ("*Analogic measuring*").

No primeiro modelo, existem pequenos dispositivos que depois de acionados, as coordenadas de cada eixo são adquiridas. Já o segundo modelo funciona através da medição da deflexão de molas planas paralelas aos eixos através de transdutores indutivos quando o apalpador toca a peça [6].

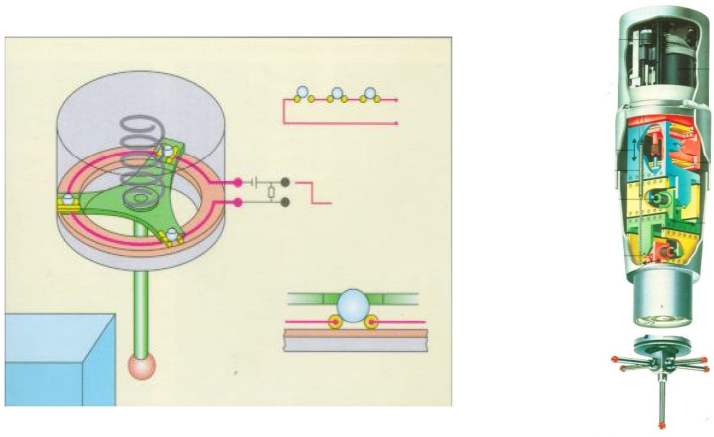


Figura 2.3: Modelos de sistemas de apalpação: tipo (a) Comutador e (b) Analógico [5].

Parte dos erros sistemáticos do Sistema de Apalpação pode ser compensada pela qualificação do apalpador com a utilização prévia de uma esfera padrão como a apresentada na Figura 2.4. Contudo, incertezas como a flexão da haste, histerese e repetitividade devem ser levadas em consideração nos resultados da medição.

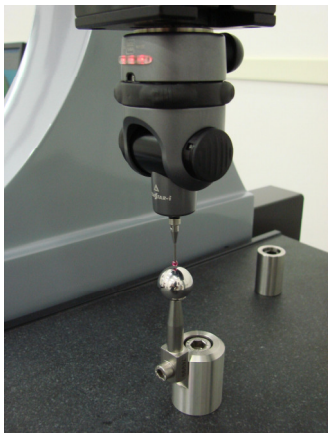


Figura 2.4: Qualificação de um apalpador por meio de uma esfera padrão (Laboratório de Metrologia, UNIMEP).

O controle CNC (*Computer Numerical Control*), assim como em qualquer outro equipamento dotado de sistema NC (*Numerical Control*), é responsável por monitorar e controlar os sistemas mecânicos, eletrônicos e ópticos do equipamento, além de interfacear com o *software* de medição CAI (*Computer Aided Inspection*) que está sendo utilizado.

O software de medição, ou CAI é responsável por permitir ao operador o total controle do processo de medição, juntamente com a possibilidade de criação ou alteração da execução dos programas CNC gerados, além da produção dos relatórios contendo os resultados da medição [8]. O sistema CAI será melhor descrito no tópico 2.4.

2.1.1 Importância da medição

O desafio que se coloca aos sistemas produtivos atualmente é que produzam peças cada vez mais exatas, com formas geométricas cada vez mais complexas e com custos cada vez menores. O atendimento destas necessidades requer uma contínua melhoria no ciclo de desenvolvimento de produto ao longo de suas etapas [9].

Um fator importante para este ciclo de desenvolvimento é a etapa de medição [10], pois devido aos produtos fabricados possuírem requisitos funcionais estéticos que precisam ser atendidos de forma satisfatória para o seu sucesso na função para o qual foram desenvolvidos, estes requisitos podem ser verificados apenas através de um processo de medição. Para isto, cada requisito a ser inspecionado é dotado de uma faixa de tolerância que ajude a decidir quais os limites que a conformidade geométrica deverá ter para uma eventual aprovação do controle da qualidade [11].

A garantia de que os produtos tenham uma conformidade geométrica dentro dos padrões aceitáveis é uma obrigatoriedade para uma empresa conseguir se manter em um mercado cada vez mais competitivo [12], além de abrir um caminho seguro para uma futura cooperação entre os demais setores ou mesmo outras empresas coexistindo em um ambiente cada vez mais globalizado.

Com os sistemas produtivos buscando cada vez mais diminuir o retrabalho e o tempo de produção e conseqüentemente os custos, os novos processos de medição surgem como grandes aliados à otimização dos processos já que, além de inspecionar a conformidade geométrica das peças produzidas, com os

dados gerados, podem-se identificar prováveis variações no processo de fabricação, tendo assim um indicador confiável de onde se deve investir em melhorias para as futuras correções [13]. A Figura 2.5 mostra que o sistema de medição pode ser uma realimentação do processo produtivo, servindo tanto como avaliador de peças como também um indicador do desempenho do processo em si. Os dados obtidos pelo processo de inspeção podem ser utilizados para a tomada de decisões, definindo melhorias nos processos ou indicativos de setores que necessitem de maior investimento para corrigir erros pertinentes à suas funções.

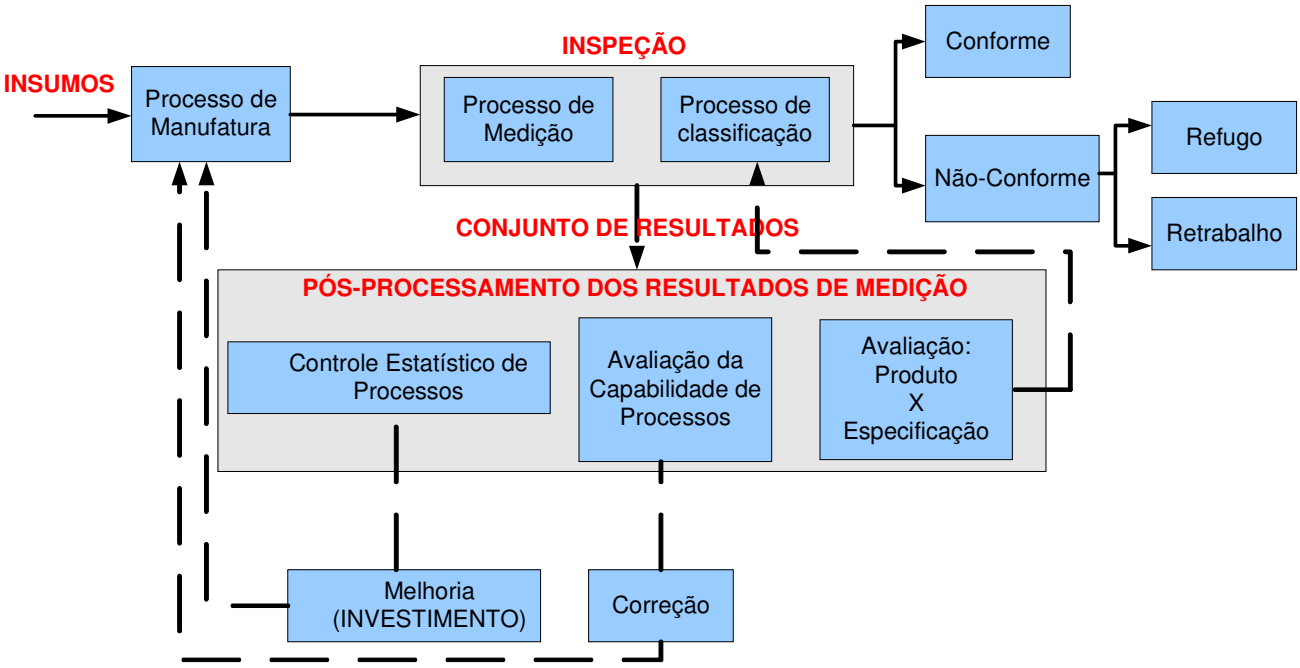


Figura 2.5: Realimentação baseada nos resultados de inspeção MMC [14].

Vista a importância que uma tecnologia de inspeção tem no sistema produtivo, é fundamental que os processos de medição adotados por uma empresa sejam capazes de realizar medições confiáveis, evitando a classificação equivocada de produtos (peças ruins aprovadas ou peças boas reprovadas) ou que o processo de fabricação seja realimentado com informações que não condizem com a real situação da qualidade de fabricação, fazendo com que decisões erradas ou precipitadas nas melhorias do processo de fabricação sejam tomadas [4, 15]. A Figura 2.6 ilustra o que resultados provenientes de uma inspeção errônea podem gerar.

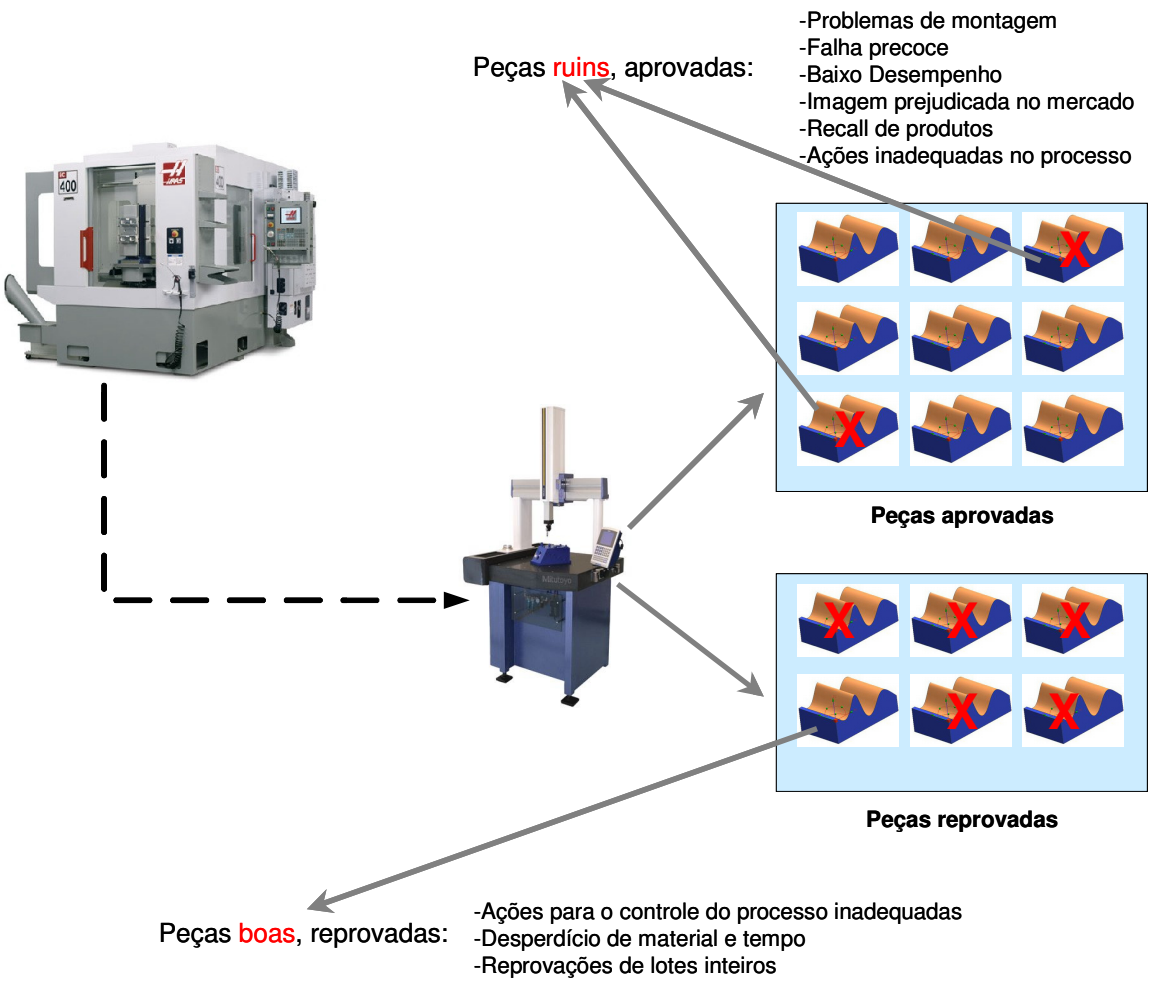


Figura 2.6: Resultados de tomadas de decisões baseadas em uma má inspeção [16].

Na década de 90, a utilização de máquinas de medir por coordenadas para aferição de peças e controle do processo produtivo no setor industrial cresceu de forma expressiva. No Brasil, o número de empresas que agregaram o uso de máquina de medir por coordenadas ao seu controle da qualidade chegou a ter um crescimento de 20% no ano de 1999 em relação aos anos anteriores [17].

Muitos fatores têm levado o ambiente industrial a procurar as máquinas de medir por coordenadas como uma importante ferramenta. Dentre as potencialidades deste instrumento de medição pode-se destacar [18]:

- Flexibilidade e rapidez na medição de peças dotadas de qualquer tipo de geometria;

- Possibilidade de integração com sistemas CAD/CAM;
- Uso de acessórios de medição e magazine para troca de apalpadores;
- Alto grau de informatização dos resultados das medições e, com a utilização de sistemas computacionais, uma vasta gama de facilidades como a programação de estratégias e emissão de relatórios;
- Programações *off-line* permitindo a programação de peças sem o desperdício de tempo de máquina parada.

2.1.2 Diferença entre a tecnologia de medição por coordenadas e os métodos convencionais

As máquinas de medir por coordenadas vêm substituindo os métodos convencionais de medição por suas vantagens estratégicas, como por exemplo, a redução do tempo necessário das medições e menor exigência de conhecimento dos diferentes métodos por parte do usuário, permitindo o controle dos mais diversos tipos de geometrias em um único equipamento [19]. A Figura 2.7 apresenta exemplos de peças que podem ser medidas por uma MMC.

Figura 2.7: Exemplo de peças inspecionadas por meio de uma MMC [20].

Segundo Sousa [5] os métodos convencionais de inspeção metrológica apresentam vantagens e desvantagens. Entre as desvantagens, algumas que podem-se destacar são:

- Os instrumentos são dedicados e com pouca ou nenhuma flexibilidade;
- Maior tempo, custo e menor confiabilidade para a medição de peças com formas complexas;
- Grande dificuldade de integração com ambientes automatizados.

As principais vantagens dos métodos convencionais são:

- Menor qualificação exigida do operador,
- Menor custo de investimento.

Analogamente, as vantagens que podem ser relacionadas à MMC, são [21]:

- Fácil adaptação às mais diferentes tarefas de medição;
- Medições obtidas por meio de modelos matemáticos;
- Maior confiabilidade em tarefas com alto grau de complexidade;
- Menor tempo e custo para a medição de peças com formas complexas;
- Maior facilidade na integração de ambientes automatizados.

Pode-se citar em relação aos métodos de medição por coordenadas as seguintes desvantagens do método de medição por coordenadas:

- Maior qualificação do operador, inclusive com a necessidade de uma atualização constante do seu conhecimento;
- Maior custo de investimento.

A seguir, a Tabela 2.1 exemplifica com maiores detalhes os dados, de forma comparativa entre os métodos de medição convencionais e o método de medição por coordenadas mostrando suas vantagens e desvantagens:

Tabela 2.1: Diferenças entre os métodos de medição [5].

Medição Convencional	**Medição por Coordenadas**
Alinhamento manual e demorado da peça	Não é necessário o alinhamento manual da peça
Instrumentação dedicada e pouco flexíveis	Flexibilidade e adaptação simples às tarefas de medição
Determinação separada de dimensões, desvios de forma e posição, utilizando diferentes instrumentos de medição	Determinação conjunta de dimensão, forma e posição, na maior parte das vezes, em uma única medição
Dificuldade de integração em ambientes automatizados	Possibilidade de integração em ambientes com automação flexível
Menor confiabilidade em tarefas complexas	Maior confiabilidade em tarefas complexas
Maior tempo de inspeção para grande quantidade de peças complexas	Menor tempo de inspeção para grande quantidade de peças (possibilidade de programação CNC)
Maior custo de inspeção de peças com geometrias complexas	Menor custo de inspeção de peças com geometrias complexas
Menor custo de investimento	Maior custo de investimento
Menor qualificação do operador	Maior qualificação do operador

Juntamente com estes fatos, a tecnologia de medição por coordenadas possibilita uma interação com o ambiente de produção [4] como, por exemplo, a utilização de recursos que permitem a verificação da temperatura ambiente para a compensação das incertezas da medição [22].

Esta tecnologia também permite a integração com outros meios de produção, como a integração com máquinas-ferramenta, por exemplo, sendo possível

corrigir eventuais dados do processo produtivo automaticamente de acordo com os resultados da medição.

A integração da tecnologia de medição por coordenadas com outros meios de produção, permite inclusive o envio de dados diretamente para os setores que monitoram o desempenho dos processos, possibilitando a troca de informações necessárias para a otimização do processo produtivo [14].

Devido à extrema facilidade de automação, de integração ao ambiente industrial e a rapidez de medição, as máquinas de medir por coordenadas apresentam vantagem em relação aos sistemas convencionais que atualmente já não conseguem acompanhar a evolução tecnológica da manufatura [23].

2.1.3 Requisitos para utilização do sistema de medição por coordenadas

Geralmente quando as empresas investem na aquisição de uma MMC, elas devem ter em mente que a máquina é apenas uma parcela de todo o investimento requerido para a utilização adequada da tecnologia em questão [24].

Para se obter resultados confiáveis, o usuário da máquina deve procurar investir em determinados requisitos que são essenciais para obter-se o retorno desejado do investimento. Um dos requisitos mais importantes é o ambiente em que a máquina de medição será instalada.

A temperatura que os fabricantes utilizam para estabelecer as incertezas de seus equipamentos é de 20ºC. Portanto, para garantir que tais faixas de incerteza, como a deformação da peça e da MMC devido à variação de temperatura possam ser consideradas nos resultados das medições, o equipamento de medição, juntamente com o ambiente em que se encontra, devem ser mantidos nesta temperatura controlada de forma estável e homogênea [15].

Muitos usuários acreditam que a simples utilização da máquina de medir em um ambiente com ar condicionado seja o suficiente. As variações de temperatura causadas pela proximidade às fontes de calor juntamente com o fluxo de ar frio do condicionador de ar diretamente sobre a máquina podem gerar diversas variações na MMC como na própria peça a ser medida, já que contrações ou dilatações de ambas podem levar os resultados a terem uma incerteza de até quatro vezes o valor especificado pelo fabricante [5]. A Figura 2.8 ilustra essa questão.

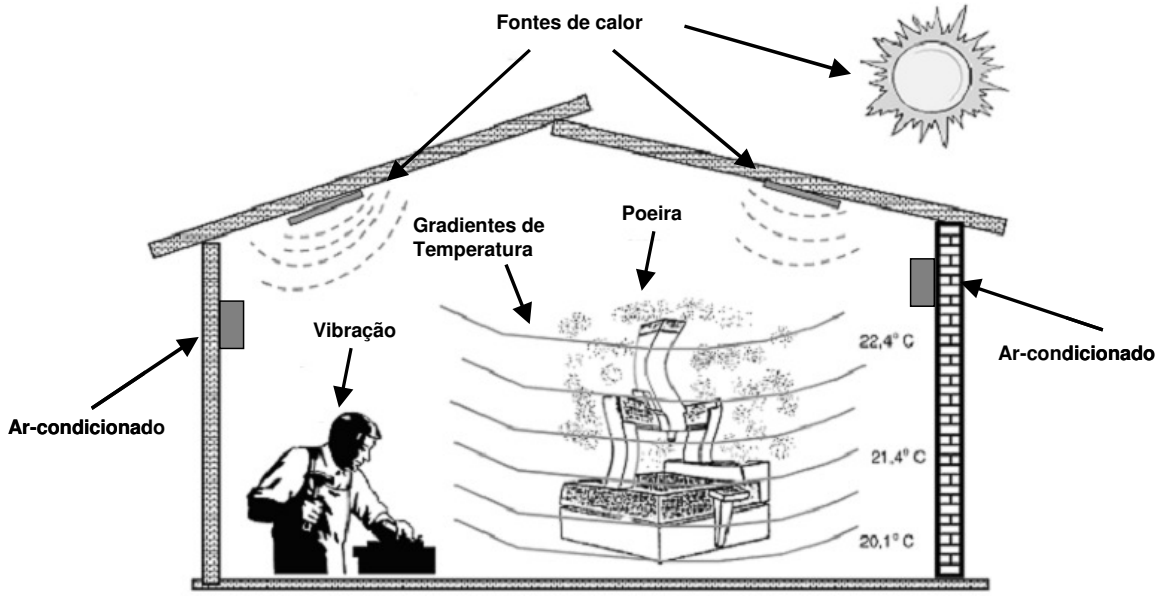


Figura 2.8: Influência causada por elementos externos à inspeção [10].

O ambiente, além de contar com a condição de temperatura a ser considerada, ainda é influenciável por outros dois fatores: poeira e partículas indesejáveis que podem se alojar nos equipamento ou na peça e vibrações que causam interferência nos resultados de medição [25].

A correta instalação da MMC longe de fontes de vibração e com o devido controle do ambiente a fim de se evitar acúmulo de resíduos, é de extrema importância para a obtenção de resultados mais precisos [22].

Além disso, podem-se destacar outros fatores que influenciam nos resultados da medição, como:

- Correta utilização do software CAI;
- Devida capacitação do operador em métodos de medição por coordenadas;
- Utilização de estratégias de medição adequadas ao nível de exatidão requerida;
- Seleção adequada de acessórios, como apalpadores e suportes de fixação para as peças;
- Conhecimento da real aplicação da peça e dos processos de fabricação envolvidos.

2.1.4 Influência do operador no resultado da medição

O operador da MMC é o responsável por todo o processo de inspeção. É dele a responsabilidade de posicionar, fixar a peça de forma que ela fique estável durante a medição, selecionar e montar o cabeçote com uma configuração adequada do apalpador e, o mais importante, estabelecer os parâmetros da estratégia de medição.

Na estratégia de medição os parâmetros que são de total responsabilidade do operador são:

- Velocidade de apalpação;
- Número de pontos;
- Distribuição de pontos pela superfície;
- Escolha de filtros que auxiliem na medição;
- Métodos de avaliação e produção do relatório final.

Estes parâmetros devem ser definidos levando-se em consideração o projeto da peça que está sendo medida, sua funcionalidade e exatidão requerida, além do processo pelo qual foi fabricada.

Entre as influências que podem gerar erros em um resultado de medição, o operador é o que tem maior destaque, mesmo com a utilização de um sistema CAI. A responsabilidade da estratégia de medição recai totalmente no operador, sendo que a direção, quantidade de pontos e a velocidade referente à estratégia, são definidas mais por experiência profissional do que por algum embasamento científico [26]. A Figura 2.9 apresenta uma relação do nível de influências nos resultados, apresentando dados comparativos entre o operador, a MMC e o ambiente em que a inspeção está sendo produzida. Na situação em que os resultados são obtidos em condições ideais de medição, as influências são encontradas em menor escala. Conforme os resultados tendem à condição crítica, na qual devem ser totalmente rejeitados, pode-se notar que as ações do operador são as que têm maior influência.

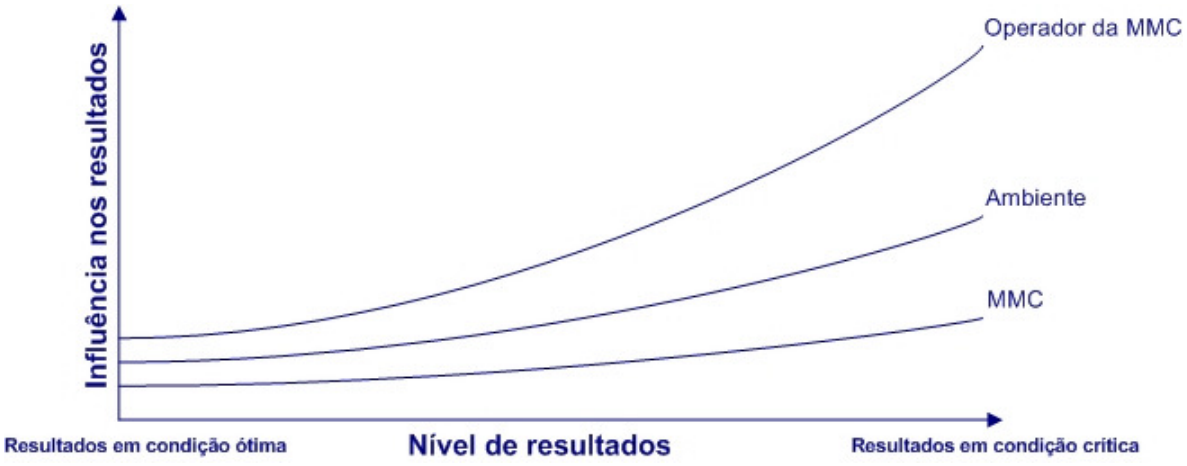


Figura 2.9: Comparação gráfica dos itens que influenciam a medição [27].

Uma distribuição dos pontos de medição feita de forma errônea induz o sistema CAI a uma interpretação incorreta da superfície real; conseqüentemente se a peça possuir algum acabamento superficial de má qualidade ou existirem verdadeiros erros da geometria, eles podem ser camuflados. A Figura 2.10 apresenta um exemplo de erro induzido por má distribuição de pontos na estratégia de medição.

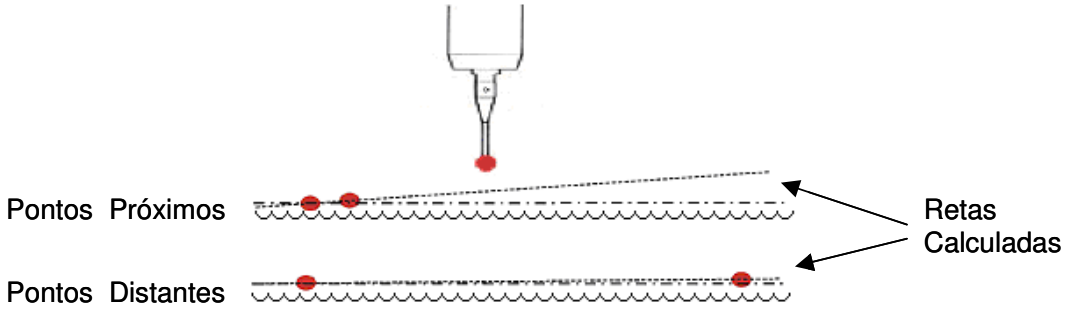


Figura 2.10: Influência da estratégia de distribuição de pontos.

A capacitação dos operadores de MMC deve estar relacionada à importância que os resultados da inspeção têm para os produtos finais.

Se a margem de tolerância que a empresa requer para seus produtos for grande, o operador necessitará apenas de cursos básicos sobre a utilização da tecnologia de medição por coordenadas.

No entanto, se a empresa necessita de um controle maior em relação à qualidade de seus produtos, o operador deve conhecer as influências dos processos de fabricação da peça, tais como estratégias de usinagem utilizada, material utilizado para a produção da peça, forma em que a peça foi fixada para sua fabricação, entre outros fatores importantes, e então gerar possíveis estratégias de medição que permitam inspecionar as zonas críticas de determinada peça.

A seguir serão apresentados o modelo de parametrização de curvas e a geração de curvas do tipo *Spline*, que representa o modelo matemático das curvas que serão utilizadas para a geração da estratégia de medição utilizadas neste trabalho.

2.2 Curvas paramétricas

Matematicamente, o conceito de uma curva tem o objetivo de capturar a idéia intuitiva de um objeto contínuo no espaço cujo domínio é um intervalo finito deste objeto [28]. As curvas podem ser expressas por três tipos de representação:

- Forma implícita;
- Forma explícita;
- Forma paramétrica.

A forma explícita é representada pela função de y=f(x). Neste tipo de formulação, devido à falta de parâmetros que limitem sua forma (objeto finito), torna-o inviável para o uso em modelamento virtual [29].

A representação na forma implícita é descrita por uma função do tipo f(x,y)=c, permitindo apresentar sob certas condições, mais de um valor em y para o mesmo valor em x. Ainda tem como um fator limitante, a falta de um parâmetro que defina um campo finito de trabalho.

Estas duas formas matemáticas permitem determinar rapidamente se um ponto pertence a uma curva, ou ainda em que lado da curva este se localiza, mas se mostram limitadas para expressar muitas das formas requeridas para modelagem geométrica necessárias em um sistema CAD [29].

A representação paramétrica é descrita nas funções de x=f(t) e y=f(t), controladas pela variação de um mesmo parâmetro para ambas (parâmetro t), com maior controle de suas formas, permitindo que sejam desenvolvidas formas finitas. [28, 30].

Para a representação de modelos virtuais em sistemas CAD, como os que serão utilizados neste trabalho, algumas características que estão presentes apenas na forma paramétrica se fazem necessárias, são elas:

- Definição de uma curva por meio de seus pontos e de suas derivadas em tais pontos (Figura 2.11):

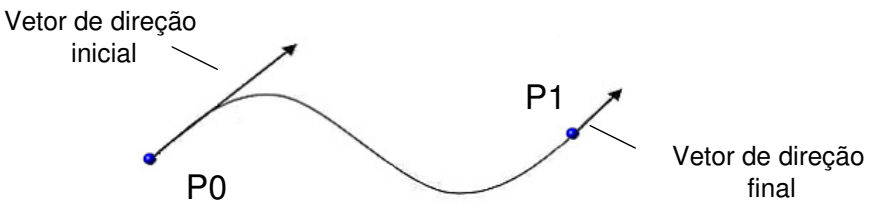


Figura 2.11: Exemplo de curva com pontos e derivadas definidas.

- Criação de curvas fechadas ou com laços, características de múltiplos valores de y para um mesmo valor de x (Figura 2.12):

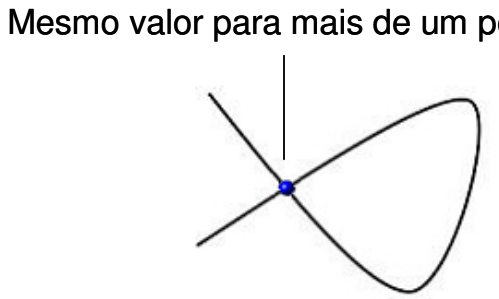


Figura 2.12: Exemplo de curva fechada ou com um laço.

- Obtenção de uma curva suave que passe por um conjunto de pontos (Figura 2.13):

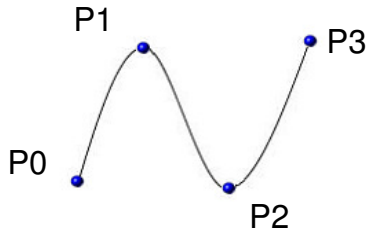


Figura 2.13: Exemplo de curva que passa por pontos dados.

No entanto, devido aos graus de liberdade (parâmetros independentes) existentes em uma curva parametrizada tridimensional, sua manipulação se torna praticamente inviável sem a utilização de modelos matemáticos que permitam controlá-los a fim de facilitar o seu uso em um modelamento geométrico [29].

A Figura 2.14 mostra a representação matemática de uma curva tridimensional parametrizada e a representação de uma curva do mesmo tipo através do modelo matemático de Lagrange.

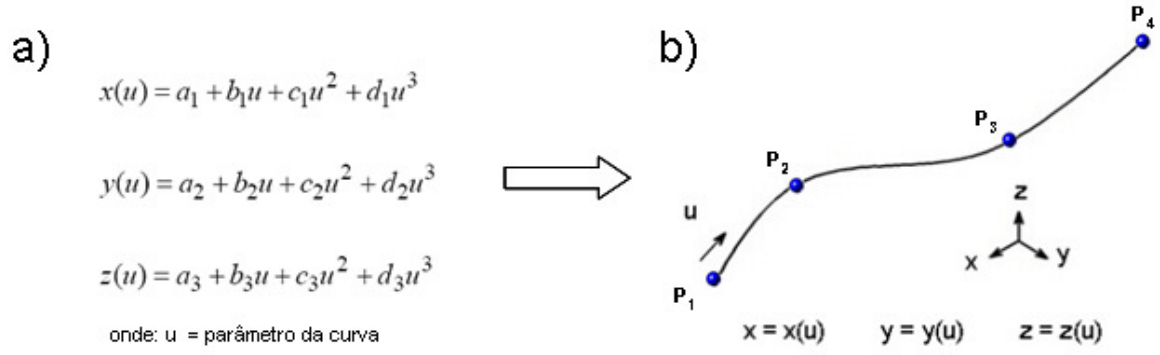


Figura 2.14: Representação de uma curva tridimensional parametrizada (a) e pelo modelo matemático de Lagrange (b).

Conforme foi aumentando a necessidade de maior controle na descrição de uma curva (recursos para construção e edição), outros modelos matemáticos surgiram, como por exemplo, as Curvas de Bézier ou curvas do tipo *Spline*. Por estes modelos serem utilizados no modelamento de geometrias complexas nos sistemas CAD, eles terão um maior destaque a seguir.

2.2.1 Curva Bézier

O conceito matemático da Curva de Bézier foi originalmente desenvolvido pelo francês Pierre Bézier para a indústria automobilística, nos anos 60. Tratava-se de uma nova ferramenta de desenho baseada em curvas matemáticas. Posteriormente, em 1972, seu conceito se tornou a base do sistema CAD *Unisurf* para a representação de formas complexas de peças automotivas. Essas construções matemáticas tinham como mérito o fato de proporcionarem uma definição fácil das curvas de maneira que os computadores pudessem representá-las [31].

A Curva de Bézier emprega no mínimo 3 pontos para sua definição, podendo chegar a “n” pontos de controle. O grau de interpolação polinomial de uma curva Bézier é relacionado ao número de vértices de seu polígono de controle, ou seja, alterando um dos pontos, toda a curva é alterada.

Uma característica da Curva de Bézier é passar pelo primeiro e último ponto do polígono de controle (também conhecidos como *endpoints*), e não passar por seus pontos intermediários (também conhecidos como *control points*) [32].

Outra característica da Curva de Bézier é ser tangente aos vetores definidos pelo segmento formado pelos pontos P_0 e P_1 e o segmento formado pelos pontos P_{n-1} e P_n de seu polígono de controle. A Figura 2.15 ilustra uma Curva de Bézier formada por n pontos em seu polígono de controle [33].

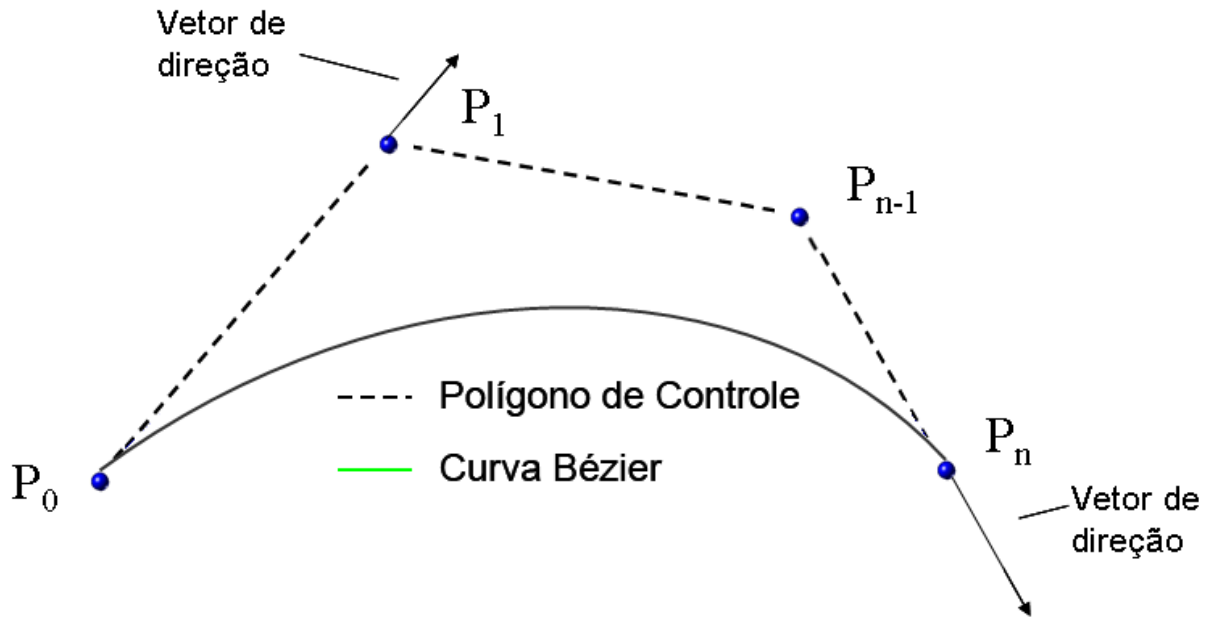


Figura 2.15: Representação da Curva de Bézier.

Matematicamente, pode-se definir uma curva Bézier por [33]:

$$P(t) = \sum_{i=0}^{n} B_i J_{n,i}(t)$$

onde:

P(t) = Curva de Bézier;

B_i = Pontos do Polígono de Controle;

n = Grau do Polígono de Controle;

t = Parâmetro da curva que varia de t_{min} à $t_{máx}$;

$J_{n,i}(t)$ = Função de Suavização.

A equação que define a função de suavização para uma curva Bézier é dada por [33]:

$$J_{n,i}(t) = \left[\frac{n!}{i!(n-i)!} \right] t^i (1-t)^{n-i}$$

Por meio desta função de suavização, são determinadas as características da curva Bézier que serão agregadas aos vértices do polígono de controle para então ser definida a curva em si [33].

2.2.2 Curva B-Spline

O nome *Spline* designa curvas especiais formadas por segmentos de polinômios, tendo sua origem a partir de uma prática comum em que se utilizam réguas juntamente com pesos para a determinação de curvas utilizadas na construção de barcos. O intuito era alterar a forma da curva para obtenção de maior suavidade, deslocando estes pesos ao longo da mesma, tentando posicioná-los de maneira que a energia de deformação da *Spline* elástica fosse a mínima possível.

As curvas *Spline*s começaram a ganhar espaço primeiramente substituindo os polinômios em interpolação de pontos, requerendo para isto um menor grau de funções. Logo depois, em outras ferramentas gráficas para se construir curvas suaves, uma vez que as *Spline*s se adequam perfeitamente a problemas de formas complexas.

Em uma curva *Spline* ou *Bézier,* ao se modificar um dos pontos de controle, toda a curva é afetada. No entanto, as curvas *B-Spline* exibem a característica de que os pontos de controle não são transpassados pela curva, ao contrário do que ocorre em uma *Spline* simples, permitindo com que sejam mais simples

as alterações locais e a simplicidade de cálculos requeridos para a possibilidade destas alterações.

Para desenhar uma curva *B-Spline* precisa-se de um conjunto de pontos de controle, um vetor de nós com ao menos um coeficiente para cada ponto de controle, de forma que todos os segmentos de curva sejam unidos satisfazendo certa condição de continuidade.

A definição da curva *B-Spline*, pode ser representada pela equação [33]:

$$P(t) = \sum_{i=1}^{n+1} B_i N_{i,k}(t)$$

onde se tem:

P(t) = Curva *B-Spline*;

B_i = Vértices do Polígono de Controle;

n+1 = Quantidade de pontos do Polígono de Controle;

t = Parâmetro da curva que varia de t_{min} à $t_{máx}$;

k = Ordem da Curva *B-Spline*, podendo ser definida no intervalo $2 \leq k \leq n+1$;

$N_{i,k}(t)$ = Função de Suavização (Função Cox De-Boor).

A função de suavização para uma curva *B-Spline* é definida por [33]:

$$N_{i,1}(t) = \begin{cases} 1 \text{ se } x_i \leq t < x_{i+1} \\ 0 \text{ para demais casos} \end{cases}$$

$$N_{i,k}(t) = \frac{(t - x_i)\, N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)\, N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}$$

onde :

x_i = vetores de nó em relação ao parâmetro t que respeitam a relação $x_i \leq x_{i+1}$.

A função de suavização também é conhecida como equação de recursão Cox De-Boor [34]. Permite uma variação polinomial de acordo com o parâmetro de ordem k. Se o grau é zero, isto é, k = 0 (grau utilizado apenas para o cálculo a equação de Recursão, conforme Figura 2.17) estas funções base são funções de passo e isto é o que a primeira expressão diz. Por exemplo, possuindo quatro nós $x_0 = 0$, $x_1 = 1$, $x_2 = 2$ e $x_3 = 3$, os períodos de nó 0, 1 e 2 são [0,1], [1,2], [2,3] e as funções base de grau 0 são:

- $N_{0,0}(x) = 1$ em [0,1], e 0 para o restante;
- $N_{1,0}(x) = 1$ em [1,2] e 0 para o restante;
- $N_{2,0}(x) = 1$ em [2,3] e 0 para o restante.

A Figura 2.16 apresenta os valores de passo a serem considerados para cada função.

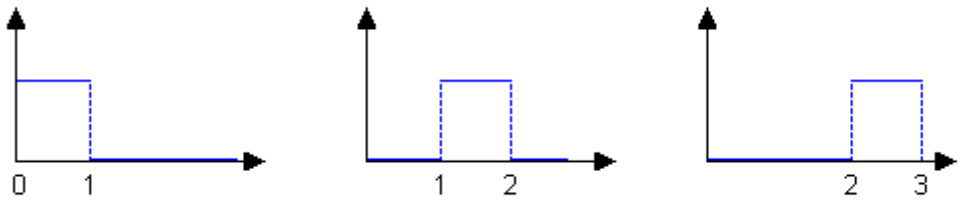


Figura 2.16: Funções de Passo.

O modo de se calcular $N_{i,k}(t)$ para k maior que 0, utiliza-se o esquema de computação triangular [35], conforme apresentado na Figura 2.17.

Para o cálculo de $N_{i,k}(t)$, as funções $N_{i,k-1}(t)$ e $N_{i+1,k-1}(t)$ são exigidas. Então, conseqüentemente, é necessário calcular também $N_{i,k-2}(t)$, $N_{i+1,k-2}(t)$ e $N_{i+2,k-2}(t)$, e assim sucessivamente. Este processo continua até que todos $N_{i,k}(t)$ requeridos sejam calculados.

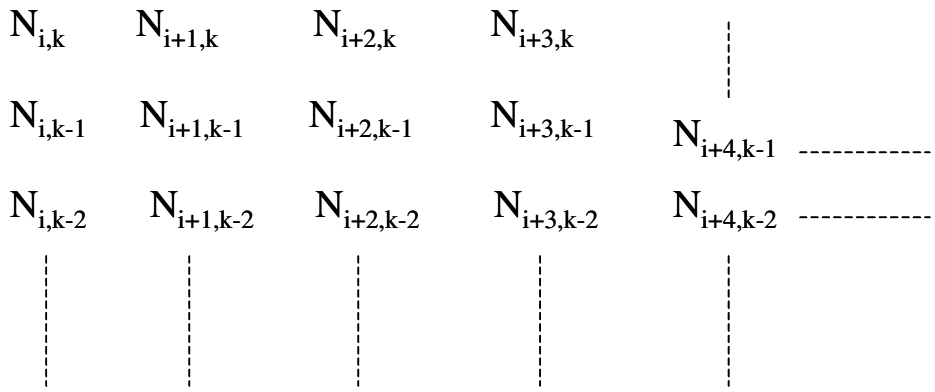


Figura 2.17: Diagrama do esquema de Computação Triangular.

Outra característica importante na representação de uma Curva *B-Spline*, é o seu vetor de nó (knot vectors). Esses vetores influenciam no cálculo da função de suavização, e conseqüentemente, na curva como um todo.

Estes vetores de nó são definidos a partir do parâmetro t, e podem ser descritos como [33]:

- Vetores de nós periódicos;
- Vetores de nós abertos, onde podem ser distribuídos de forma:
    - Uniforme,
    - Não uniforme.

Para os vetores de nós periódicos e abertos uniformes, o seu intervalo de distribuição é constante em toda a curva. A quantidade de um vetor uniforme é definida por n+k. Como por exemplo, o vetor:

X=[0 1 2 3 4 5 6 7]

Para uma curva com 4 vértices em polígono de controle (n+1 = 4) e tendo grau 3 (k = 4). Para a obtenção deste vetor dentro de um parâmetro t, variando entre 0 e 1, se tem para o mesmo exemplo a seguinte representação:

X=[0 0,14 0,29 0,43 0,57 0,71 0,86 1]

Nos vetores de nós abertos (uniformes e não uniformes), é utilizada uma propriedade que aplica repetidos valores no início e no fim do vetor. Esta repetição ocorre sempre em relação ao valor da ordem da curva (k). Como exemplo da representação deste tipo de vetor para uma curva de ordem 3 (k = 3), se tem:

$$X=[0\ 0\ 0\ 1\ 2\ 3\ 3\ 3]$$

Esta propriedade obriga a curva em questão a passar através do primeiro e último vértice do polígono de controle tangenciando os seus respectivos segmentos, conforme Rogers [33].

Ainda para vetores de nó abertos, eles podem ter uma distribuição uniforme ou não uniforme. Para os vetores uniformes, com a exclusão dos valores iniciais e finais que são repetidos de acordo com o valor da ordem (k), o restante segue um intervalo de distribuição constante. Já para o vetor de nós não uniforme, tais valores podem variar, fazendo com que as funções de suavização sejam igualmente não uniformes. A Figura 2.18 apresenta um exemplo de um vetor com distribuição uniforme e um vetor com distribuição não uniforme, ambos baseados em uma curva de ordem 4 (k = 4).

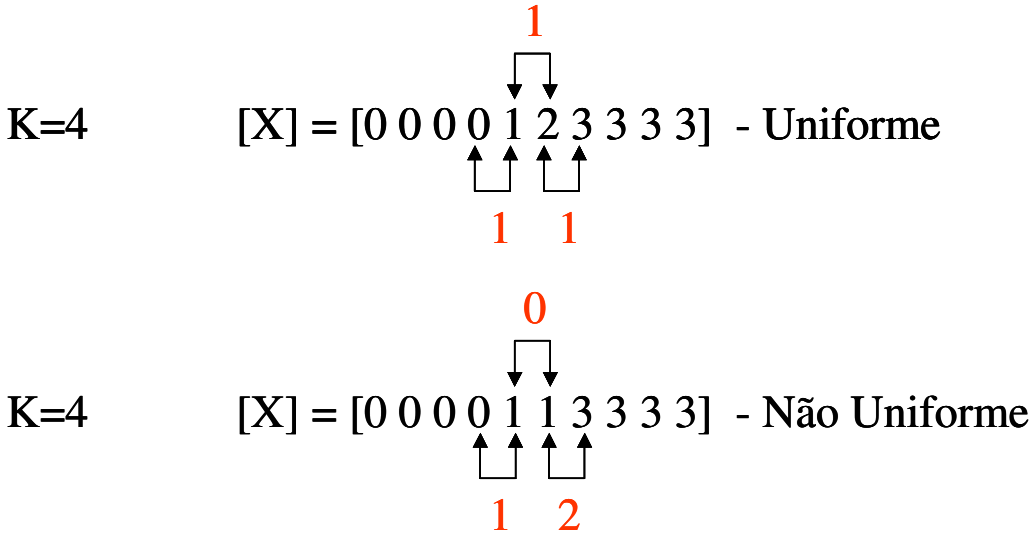
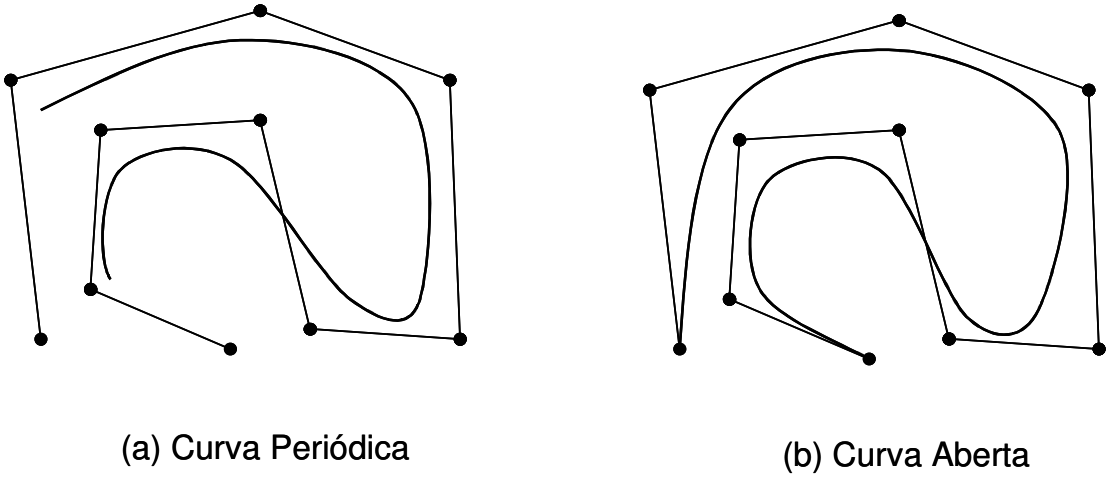


Figura 2.18: Representação de vetores uniformes e não uniformes [36].

Para uma curva periódica, seu início e fim se dão em locais diferentes do ponto inicial e final de seu polígono de controle, conforme Figura 2.19 (a). Para as

curvas abertas, devido à sua propriedade de repetição de valores em seu vetor de nós, o ponto inicial da curva é o mesmo do primeiro ponto de seu polígono de controle, assim como seus pontos finais respectivamente, como mostra a Figura 2.19 (b).



(a) Curva Periódica (b) Curva Aberta

Figura 2.19: Exemplo de tipos de curva B-Spline: (a)Periódica e (b)Aberta.

Uma propriedade das curvas *B-Spline* que será utilizada em uma das estratégias deste trabalho, é a obrigatoriedade de uma curva deste tipo ser representada por segmentos de curvas polinomiais de grau k-1, presentes em cada intervalo do parâmetro da curva, dados por $x_i \leq t \leq x_{i+1}$.

A Figura 2.20 ilustra os segmentos de curva polinomiais encontrados em uma curva B-Spline.

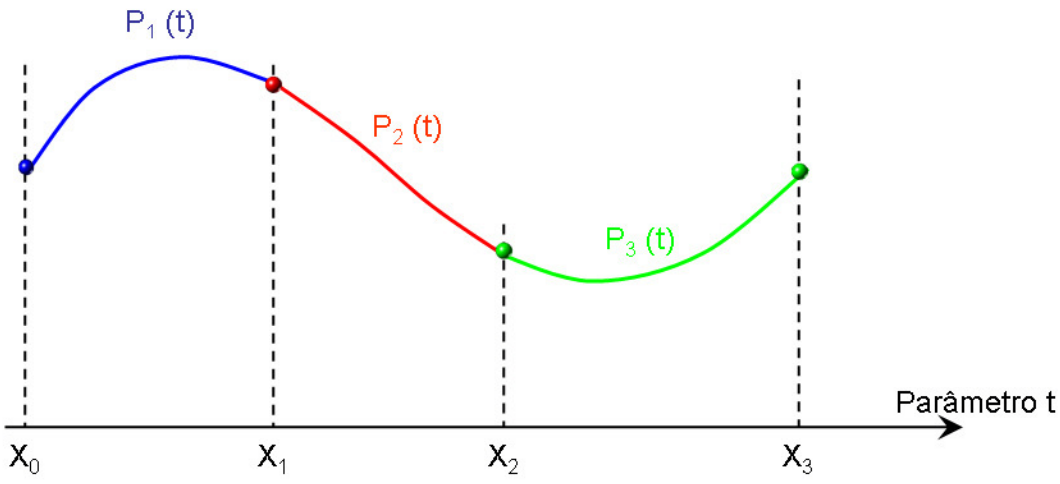


Figura 2.20: Segmento de Curvas Polinomiais na Curva B-Spline [36].

Para assegurar a continuidade entre os segmentos de curva, criaram-se restrições adicionais, chamadas de Continuidades de Curva [33]. Existem dois tipos de continuidade:

- Continuidade Geométrica, sendo representada por G^n, onde n = grau de continuidade;
- Continuidade Paramétrica, representada por C^n, onde n = grau de continuidade.

Na situação de continuidade geométrica, para um grau de continuidade zero (G^0), os segmentos de curva são unidos por um ponto em comum. Para um grau de continuidade igual a um (G^1) além da união pelos pontos em comum, as tangentes de cada um têm a mesma direção [37].

Para a situação de continuidade paramétrica, a continuidade de grau zero (C^0) apresenta a mesma situação da continuidade geométrica de mesmo grau, ou seja, a união dos segmentos ocorre somente através de um mesmo ponto. Já no caso de uma continuidade paramétrica com grau maior que zero (C^n), os segmentos se unem em relação a n-ésima derivada nos pontos de união, ou seja, além da união por pontos, suas tangentes têm a mesma direção e mesma magnitude [33].

A Figura 2.21 ilustra exemplos de continuidade de curvas, apresentando um modelo com uma continuidade de grau zero, (G^0), além da continuidade geométrica com grau 1 (G^1), e uma continuidade paramétrica de grau 1 (C^1).

Figura 2.21: Exemplos de continuidade de curvas: continuidade G^0, G^1 e C^1.

2.2.3 NURBS

As curvas B-*Spline* podem ser modeladas por polinômios. Embora sejam flexíveis e apresentem propriedades úteis para os sistemas CAx, como a continuidade de uma curva em suas 1° e 2° derivadas, elas não podem representar uma curva mais simples, como por exemplo um círculo. Os círculos podem ser representados com funções racionais (funções que são quocientes de dois polinômios). Para lidar com círculos, elipses e muitas outras curvas que podem ser representadas por polinômios, é necessária uma extensão para as curvas *B-Spline* [38].

Para se ter um grau adicional de flexibilidade na manipulação geométrica de curvas *B-Splines* foi então adicionado um parâmetro denominado de peso (h), aos vértices do polígono de controle.

Com este parâmetro adicional, tornou-se possível o controle de uma curva Spline por [33]:

- Seu grau polinomial;
- Quantidade de vértices no polígono de controle;
- União de segmentos de curva através do controle não uniforme dos nós;
- Controle local em cada segmento através do parâmetro peso, definindo o grau de atração que o vértice de controle tem em relação à curva.

Como esta propriedade visa gerar um grau total de flexibilidade na manipulação da curva, ela foi introduzida nas curvas *B-Splines* abertas não uniformes. Daí se tem o nome NURBS (*Non-Uniform Rational B-Spline*), ou seja, uma curva B-Spline não uniforme em sua forma racional devido ao controle local que o parâmetro peso (h) proporciona.

Atualmente, este tipo de curva é amplamente usado na construção de superfícies complexas, sendo utilizada nos mais diversos sistemas CAD [39].

Naturalmente, com a adição de mais um parâmetro, a sua forma matemática é ligeiramente diferente de uma curva *B-Spline* não uniforme. Pode-se definir uma NURBS por [33]:

$$P(t) = \sum_{i=1}^{n+1} B_i R_{i,k}(t)$$

onde se tem:

P(t) = Curva NURBS,

B_i = Pontos do Polígono de Controle,

n+1 = quantidade de pontos no Polígono de Controle,

t = Parâmetro da curva que varia de t_{min} à $t_{máx}$,

$R_{i,k}(t)$ = Função de Suavização.

A diferença entre uma curva B-Spline não uniforme e uma NURBS, está em sua função de suavização (ou função Rbasis) [36], já que está em sua forma racional, recebe a adição do parâmetro peso (h) [33]. Matematicamente a função de suavização para uma NURBS é dada por:

$$R_{i,k} = \frac{h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)}$$

onde:

h_i = Peso de atração do vértice do Polígono de Controle;

$N_{i,k}$ = Função de Suavização de uma curva *B-Spline* não uniforme.

A Figura 2.22 ilustra uma curva NURBS com diferentes exemplos de variação de seu parâmetro peso.

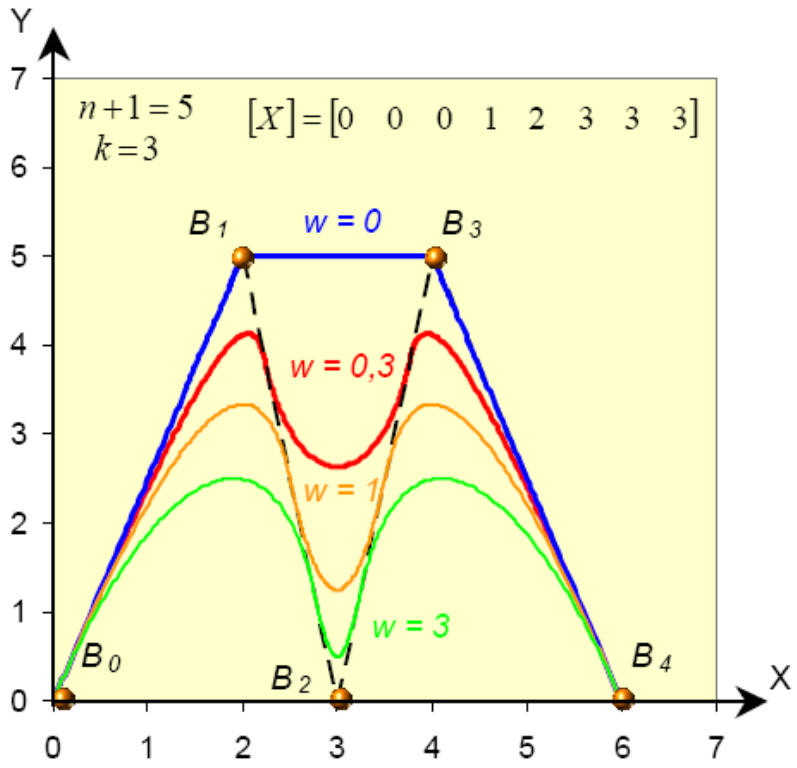


Figura 2.22: Curva NURBS definida com diferentes pesos (weight).

Como pode ser notado, de acordo com o modelo matemático, se todos os pesos forem iguais, uma curva NURBS torna-se uma curva B-*Spline* do tipo não uniforme. Se, além disso, n = k (isto é, o grau de uma curva B-*Spline* é igual ao número de pontos de controle menos 1) e existirem 2(k + 1) = 2 (n+1) vetores de nós com k + 1 deles semi-fechados em cada fim, esta curva NURBS reduz para uma Curva de Bézier. Por conseguir descrever as geometrias complexas normalmente utilizadas para a modelagem de superfícies complexas, este trabalho utiliza como função de cálculo em seu aplicativo, os conceitos matemáticos NURBS.

2.3 CAD (Computer Aided Design)

Com o advento dos computadores, cada dia aumenta o número de profissionais que se especializa na utilização de algum CAD [40] como ferramenta de trabalho, bem como nos diversos sistemas CAx, cuja integração CAD-CAI é a área de estudo deste trabalho.

Os sistemas CAD são classificados de acordo com as suas capacidades de utilização e são definidos em 3 categorias:

- Pequeno Porte - Sistemas CAD que requerem um menor custo para sua utilização e de fácil aprendizado. No entanto esses sistemas possuem algumas limitações, tais como: capacidades reduzidas de modelagem, comunicação de baixa ordem com outros sistemas, sistema não paramétrico e não associativo, sendo a sua representação geométrica em 2D por *wireframe*;

- Médio Porte - Sistemas CAD que requerem um custo intermediário para sua utilização. Algumas das suas principais características são: representação por sólidos ou por superfícies 3D, representação foto-realística com recursos de visualização em *Shading*, é possível efetuar algumas análises de propriedades mecânicas, associatividade no sistema possibilitando, por exemplo, a geração automática de desenhos 2D;

- Grande Porte - Possui todos os recursos das classes anteriores além da capacidade de integração de módulos que personalizam o sistema de acordo com as necessidades do usuário, e modelamento híbrido, que será descrito ainda neste tópico. Em função dessas características, é necessário que o usuário possua conhecimento especializado e experiência.

Com a utilização crescente de sistemas de médio e grande porte, alguns recursos foram gerados para a utilização dos projetistas, estando presentes na maioria dos sistemas CAD:

- Paramétrico - Consiste na criação de modelos que possibilitam que suas dimensões variem de acordo com a necessidade do projetista [41]. Por exemplo, medidas podem ser vinculadas a uma equação de controle, permitindo com que sejam feitas montagens onde, havendo uma variação de uma medida, todos os elementos do produto se atualizam automaticamente;

- *Features* - Um *feature [*42*]* tem como definição um elemento físico que tenha um significado específico tecnicamente. Para ser considerado um *feature*, algumas condições devem ser satisfeitas: deve ser mapeável, ter alguma significância técnica, e ser parte integrante de uma peça. O modelamento por *features* vêm facilitando cada vez mais a tarefa dos projetistas que podem contar com este método. Com uma biblioteca variável e muitas vezes expansível, o método permite a geração de furos, rasgos, canais, entre uma vasta gama de itens, para serem inseridos em algum projeto que esteja sendo desenvolvido;

- Associativo - Permite a geração de desenhos bidimensionais a partir do modelo tridimensional, atualizando-os automaticamente para cada modificação do projeto. Essa atualização não é restrita unicamente para geração de desenhos, permitindo assim a utilização em outros módulos do mesmo sistema, tais como: CAM e CAE entre outros.

2.3.1 CAD 3D

A partir da década de 70, foi desenvolvido o primeiro sistema CAD que possibilitava a representação de objetos tridimensionais.

A vantagem da utilização do sistema CAD com representação gráfica em 3D sobre a representação bidimensional, é que o usuário trabalha com a forma real do objeto a ser projeto, não fazendo necessária a utilização de vistas auxiliares para a interpretação do mesmo, principalmente quando o objeto a ser projeto faz uso de formas complexas e possui um alto grau de complexibilidade, desta maneira eliminando possíveis erros de interpretação.

Além da representação do modelo do produto, os sistemas CAD deste tipo possuem diversas ferramentas que auxiliam no desenvolvimento de um produto. Alguns exemplos de ferramentas que um sistema CAD 3D pode oferecer são [43]:

- Cálculo de volume;
- Análise da forma geométrica;
- Propriedades de massa;
- Verificação de interferências entre as peças de um mesmo conjunto.

Adicionalmente é possível a análise de elementos finitos conseguindo, assim, o estudo prévio de tensões, temperatura, entre outros.

Os sistemas CAD 3D devem ser capazes de representar fielmente as características reais do produto. Para conseguir fornecer as ferramentas ideais aos projetistas, os sistemas CAD 3D atuais de grande porte, são encontrados com modeladores híbridos que permitem a visualização de modelos sólidos e superfícies, na qual possui ferramentas integradas que auxiliam no desenvolvimento do produto, tais como: KBE [44] e GD&T [45].

2.3.1.1 Modeladores sólidos

Os modeladores sólidos têm por característica principal a representação dos produtos juntamente com seus aspectos estruturais, dessa maneira, tem a possibilidade de representar fielmente o produto a ser projetado [44].

Os sistemas que utilizam este modelador são capazes de gerar objetos tridimensionais por meio de sólidos básicos, como paralelepípedos ou cilindros. Estes sólidos podem interagir entre si através de operações *Booleanas* (adição, subtração, intersecção, entre outros). A Figura 2.23 ilustra a utilização de uma operação de adição de uma peça à outra, seguida de uma operação de subtração.

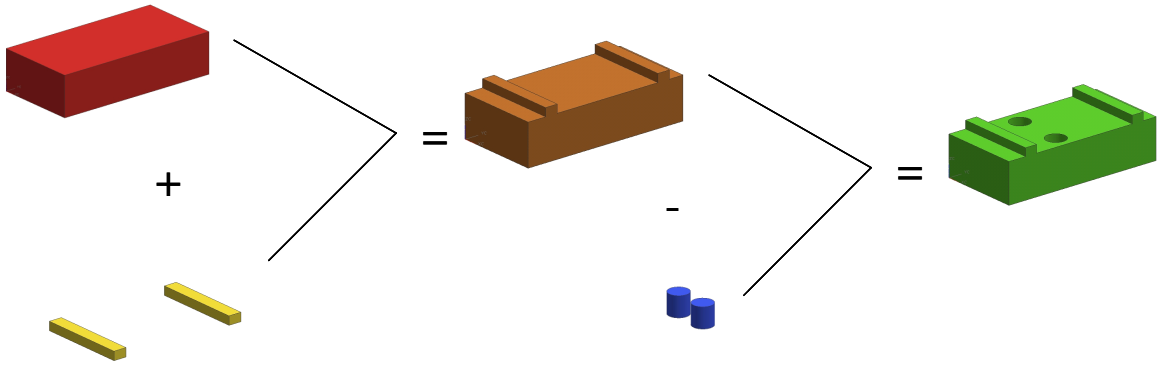


Figura 2.23: Exemplo de utilização de operações Booleanas em modelador sólido.

Para a representação de geometrias sólidas, alguns métodos podem ser utilizados. Os métodos normalmente utilizados para representar as geometrias sólidas são:

- CSG (*Constructive Solid Geometry*)- Este método permite a construção de objetos sólidos, com informações de forma e as propriedades físicas do material utilizado [45];
- BRep (*Boundary Representation*) – Representa a geometria dos sólidos por meio de um conjunto de polígonos que delimita uma região fechada no espaço [48];

2.3.1.2 Modeladores de superfícies

Quando se trata da obtenção de formas complexas, a utilização de modeladores sólidos não se mostra suficiente para a construção de um modelo geométrico que forneça uma representação fiel para estes tipos de produtos. Para preencher esta lacuna, surgem então os modeladores de superfícies.

Um modelo tridimensional gerado por superfícies não possui espessura, contém as informações sobre as arestas de construção de um objeto, incluindo o espaço contido entre elas. Através de formulações matemáticas [49], os modeladores de superfícies permitem a geração de formas geométricas complexas, com a possibilidade de edição dos pontos de construção a qualquer momento, alterando a estrutura do objeto localmente e mantendo as características restantes inalteradas [46]. A Figura 2.24 ilustra a utilização do recurso de edição de posição de pontos em uma superfície.

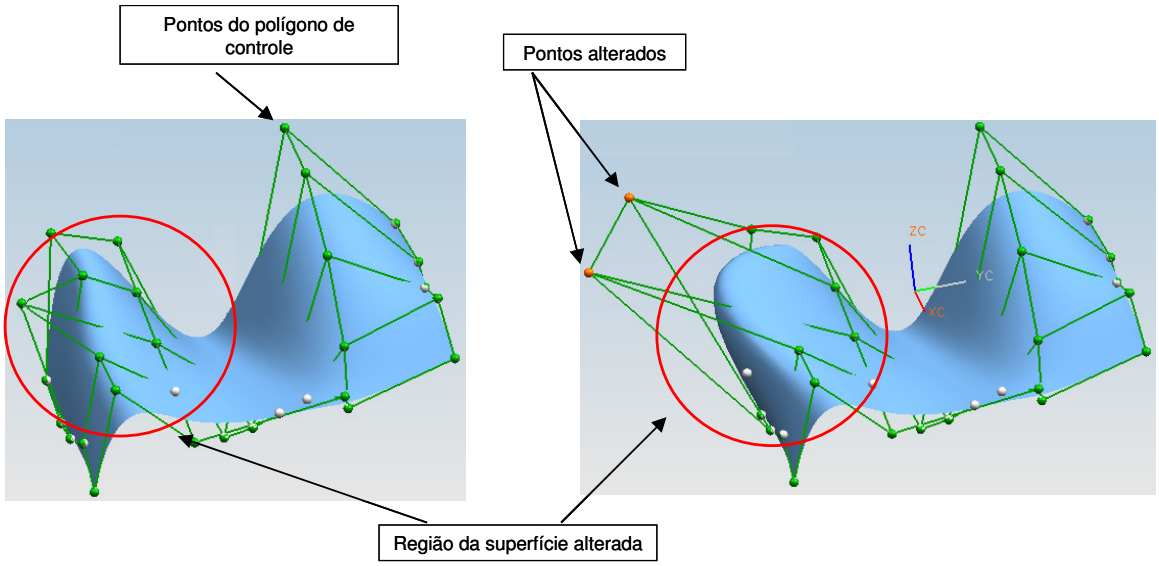


Figura 2.24: Edição de superfícies através da alteração dos pontos de construção.

Os modelos desenvolvidos com superfícies têm um maior nível de descrição dos objetos (informações sobre a geometria construída), permitindo aplicações que necessitem de maiores informações para sua produção. Entretanto, uma das principais desvantagens para a utilização de modeladores de superfícies, é a dificuldade de aprendizado, exigindo elevado investimento em um treinamento nesta área [49].

Os *softwares* de grande porte têm a opção de utilizar ambos os modeladores (sólido e superfícies) conjuntamente. A esta opção, é dado o nome de modeladores híbridos. Com isto, é possível criar produtos que tenham elementos sólidos e de superfícies [47].

2.3.2 Interface de Programação

A maioria dos sistemas CAD de grande porte permite uma interação com o usuário, através da personalização de determinadas tarefas.

Por meio da criação de aplicativos um usuário dotado de conhecimentos em programação consegue acessar funções internas do sistema CAD, permitindo a automação de processos.

Através de uma nova função desenvolvida pelo usuário, em conjunto com as funções já conhecidas do sistema do sistema CAD, é possível então a automação de diversos processos.

Algumas vantagens relacionadas à criação de um aplicativo em um sistema CAD são:

- Automatizar processos dentro do sistema CAD;
- Criar caixas de diálogo utilizando a mesma plataforma de interface que o sistema CAD em questão;
- Manipulação de dados direta no modelo nativo, não necessitando de exportações para outros *softwares*;
- Efetuar cálculos através de um novo algoritmo, utilizando parâmetros do modelo nativo;
- Gerenciar os tipos de dados de saída do aplicativo desenvolvido,
- Interface de comunicação com outros programas.

Normalmente, os sistemas CAD de grande porte são dotados de ferramentas que fornecem ao programador um meio de desenvolver a interface do aplicativo com o próprio sistema. A ferramenta mais conhecida para esta função é o API.

2.3.2.1 API

O *Application Programming Interface*, ou API de um programa de *software*, é uma coleção de rotinas e tipos de dados que dão acesso ao núcleo de funcionamento de um dado sistema [52].

Existem diversos tipos de APIs, cada um facilitando uma atividade na construção de um aplicativo [52]. Exemplos de API que são normalmente utilizados na construção de um aplicativo em um sistema CAD [48]:

- API de interface ao usuário - Inclui o desenvolvimento de menu, textos de diálogo com o usuário, mensagens de apresentação das funções, etc;

- API para seleção de modelo - Permite uma interação com a interface virtual do modelo geométrico;

- API de manutenção de dados - Acesso aos dados do modelo virtual;

- API de funções - Inclui os cálculos matemáticos, processamento de informação, etc.

Alguns APIs possuem a opção de construção de seus aplicativos de duas formas [49]:

- Aplicativos externos - Permite a execução do aplicativo sem a necessidade da execução do *software* principal, utilizando apenas as ferramentas e bibliotecas disponíveis do sistema. Este tipo de aplicativo normalmente é utilizado quando o tempo de carregamento do *software* ou a utilização de todas as suas funções (como interação visual, por exemplo) é desnecessário;

- Aplicativos internos - Este tipo de aplicativo é compilado como uma biblioteca dinâmica do *software* principal. No caso do *software* CAD NX, um arquivo do tipo dll, (veja Capítulo 4), sendo utilizado diretamente com o sistema em funcionamento. Uma das vantagens de um aplicativo desenvolvido por este modo é utilização simultânea com as outras ferramentas disponíveis no *software* para o qual foi criado.

Segundo ROZVANY [55], as principais desvantagens da criação de programas personalizados API são os custos iniciais de desenvolvimento do software, além do custo de aquisição de conhecimento inicial requerido para a utilização de um API.

Para demonstrar o funcionamento de um modelo API, o funcionamento de construção de uma interface ao usuário para o sistema CAD Siemens NX, é exposto a seguir.

Através da função *User Interface Styler* (API de interface visual do NX), é possível construir a interface visual do aplicativo, utilizando o próprio módulo de construção de interface do sistema CAD. Com esta ferramenta, é possível criar botões, inserir textos informativos em cada ação necessária que o aplicativo requer para o seu funcionamento, entre outros. Após o desenvolvimento de toda a interface visual, esta ferramenta ainda proporciona toda a construção de programação que será utilizada para inserir um programa criado pelo usuário.

A Figura 2.25, a seguir, apresenta a tela de desenvolvimento da interface visual do *software* Siemens NX. A caixa de interface é o resultado do processo, ou seja, é o objeto com que o usuário interage quando o aplicativo está em execução.

Com o editor de funções, o programador do aplicativo direciona quais tipos de dispositivos, como botões ou barras de rolagem, além dos textos, serão aplicados à caixa de interface. No editor está, também, o local onde são inseridos os nomes de chamada de cada dispositivo. Estes nomes são os endereços utilizados para a inserção de um comando da programação que controla a função desejada. A lista de funções apresenta quais dispositivos serão criados na caixa de interface.

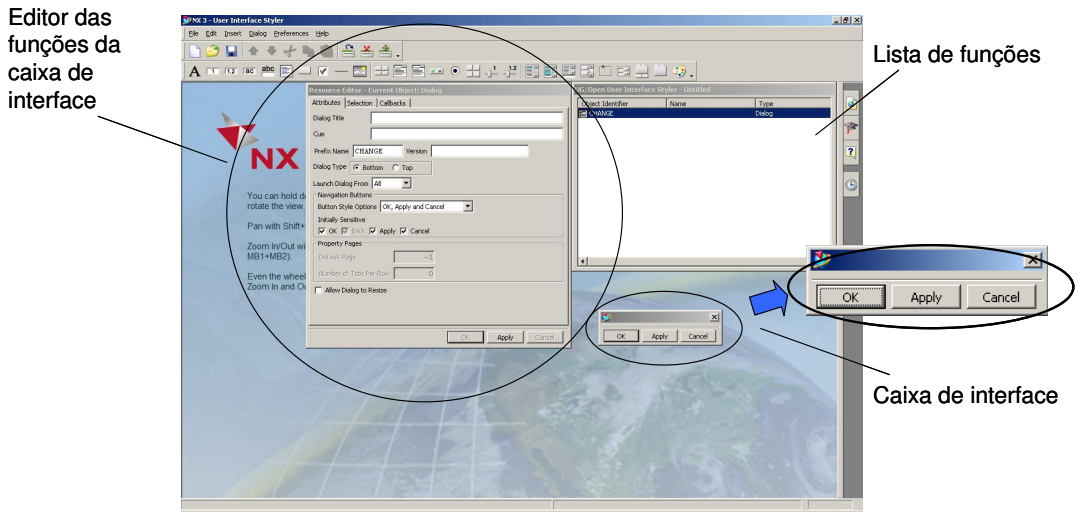


Figura 2.25: Tela de desenvolvimento de interface visual (Siemens NX).

Após o desenvolvimento da caixa de interface, é dada ao programador, a opção de selecionar qual linguagem de programação será utilizada. O *software* NX gera, então, os arquivos que serão utilizados no aplicativo. A Figura 2.26, apresenta o resultado desta função para uma caixa de interface gerada para um aplicativo que será desenvolvido em linguagem C. O nome "*hello*" é dado somente para a criação deste exemplo. O arquivo "*dlg*" é a própria caixa de interface, o arquivo "*h*" é o cabeçalho utilizado para indicar quais funções são utilizadas, e o arquivo "*c*" é o programa em linguagem C da caixa em si.

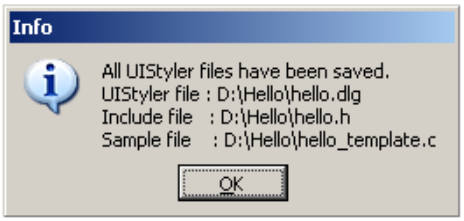


Figura 2.26: Exemplo de arquivos gerados no desenvolvimento de uma caixa de diálogo (Siemens NX).

Com estes arquivos, pode-se iniciar a utilização de outra ferramenta, como o *Visual C*. Esta ferramenta é uma interface de programação que permite aos usuários criarem aplicativos personalizados utilizando a linguagem de programação C ou C + +.

O Microsoft Visual C++ por exemplo, possibilita a interface com o sistema CAD NX. Através deste software é possível construir um projeto onde, inseridos os arquivos gerados na criação da caixa de diálogo, têm-se uma estrutura básica de programação: A Figura 2.27 ilustra tal estrutura. O arquivo gerado insere o comando de chamada da biblioteca de funções API do software (arquivo "*uf.h"*), e com a função de chamada (*void*), cria o vinculo com o software principal. O local denominado "body" é onde os comandos são inseridos.

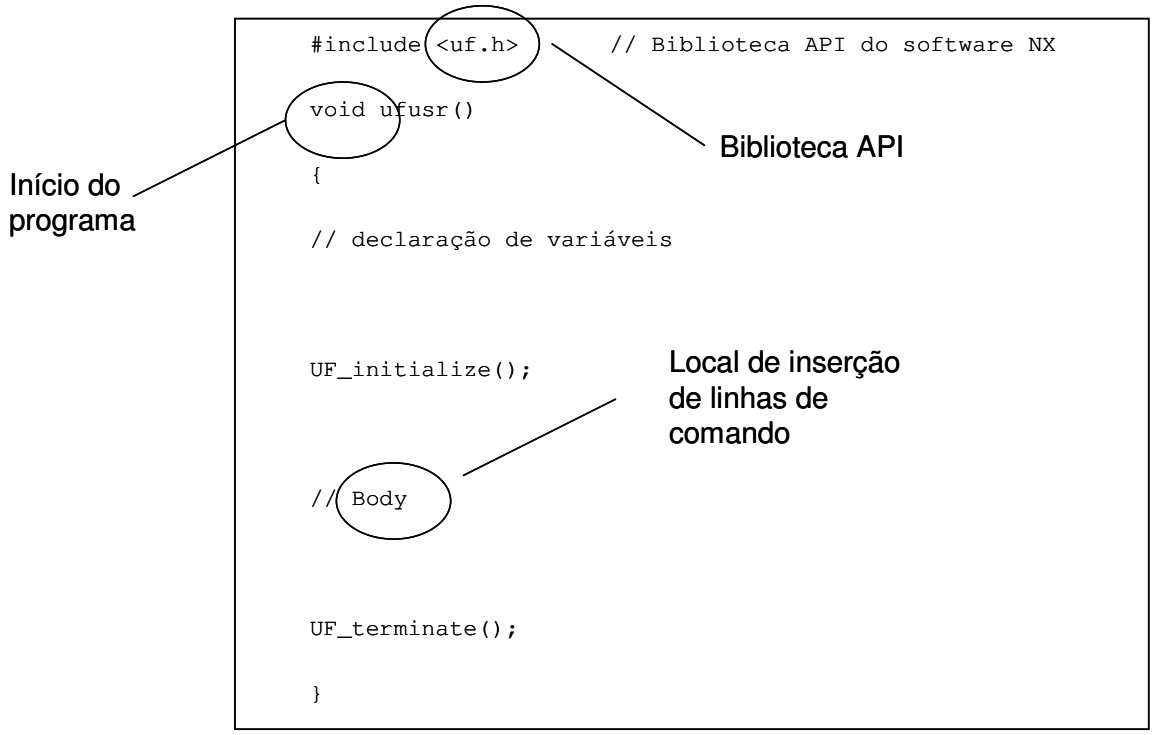


Figura 2.27: Exemplo de estrutura de programação de um caixa de diálogo.

2.3.3 Vantagens e desvantagens do sistema CAD

Em relação às vantagens que um sistema CAD propicia ao projetista, podem-se citar importantes fatores como facilidades ao operador, devido a velocidade de desenvolvimento de um novo projeto, com a possibilidade de visualizar o projeto completo em apenas um único arquivo, incluindo a possibilidade de armazenamento digital dos dados. Além disso, o CAD permite a integração com outros sistemas que são de extrema valia para o projetista [51, 52], tais como:

- CAM (Comp*uter Aided Manufacturing*): integração do projeto com a manufatura;
- CAE (*Computer Aided Engineering*): integração com software capaz de realizar simulações e cálculos a partir de um projeto CAD;
- CAI (*Computer Aided Inspection*): integração com *softwares* que interagem diretamente com métodos de medição e validação do produto. Este item será abordado no tópico 2.4.

A maior desvantagem atribuída aos sistemas CAD está relacionada aos custos de aquisição e operação:

- Custo de aquisição do *software*: dependendo da necessidade que a empresa tem, este custo pode ser a maior desvantagem, devido aos altos valores para a aquisição de uma licença de uso, principalmente para a utilização de sistemas CAD de grande porte, que englobam as mais diversas funções em um único *software*;

- Custo de aquisição do hardware: a utilização de um sistema CAD implica na aquisição de um equipamento com características próprias para o bom funcionamento do mesmo. Normalmente o hardware dispõe de processadores com velocidade adequada de processamento além de um processamento gráfico que acompanhe os requisitos do sistema;

- Custo de formação e salários: dependendo das necessidades da empresa, os cursos para a formação do projetista CAD podem ter valores altos além de salários elevados dos profissionais especializados na área.

2.4 CAI (Computer-Aided Inspection)

Como observado no tópico 2.1, o software de medição está entre os cinco componentes principais de uma MMC. Atualmente não apenas reconhecido como um programa de medição, mas sim como autênticas estações de assistência à inspeção [53, 54]. Esses *softwares* são comumente conhecidos como CAI (*Computer Aided Inspection*), ou Inspeção Assistida por Computador em português [8].

Além da tarefa de possibilitar ao usuário a programação de uma estratégia de medição, esses *softwares*, atualmente, permitem o posicionamento da peça por meio da interação com modelos CAD [55].

Estes parâmetros são tomados como referência para a especificação do posicionamento da peça na mesa da MMC, tornando a posição “zero-peça” do modelo virtual igual à posição “zero-peça” da inspeção.

Outra grande vantagem é a possibilidade da programação *off-line*, ou seja, a partir de um modelo importado de um sistema CAD, toda a estratégia de medição pode ser definida e então testada virtualmente, onde apenas depois ela é utilizada em uma MMC para eventuais testes reais. Essas opções aumentam a confiabilidade e segurança dos programas gerados, além de uma considerável redução no tempo de máquina utilizado, não tendo o desperdício de hora-máquina parada [56].

Em alguns sistemas CAI, podem ser inseridos todos os dados que influenciam o desempenho de uma MMC para serem levados em consideração no cálculo da incerteza da medição, como por exemplo, a incerteza gerada pelo sistema de apalpação ou a temperatura estimada no ambiente entre outros. Com essas variáveis, é possível o programa estimar a incerteza da tarefa que será executada pela máquina.

Pode-se concluir que através do método de programação *off-line*, se tem várias vantagens. Algumas que se pode citar são [57]:

- Avaliação de estratégias complexas e muitas vezes custosas;
- Estimativa do tempo gasto para a medição entre outros.

Os *software*s permitem a inclusão de normas como a ISO 1101 [58, 59] e a ASME Y14.5 já nas especificações geométricas em seus modelos, para a maior rapidez na inclusão de faixas de tolerância de forma adotadas internacionalmente.

O sistema de programação *off-line* demonstra o quanto um sistema CAI pode ser recompensador e extremamente útil se bem implantado [60, 61].

Um sistema CAI é composto por diversos módulos ou processos que funcionam como filtros de resultados [62, 63]. Cada módulo consegue tratar os resultados retirando dados indesejáveis da inspeção final. Como exemplo de filtros pode-se citar:

- O primeiro módulo de tratamento dos dados que é relativo a pontos que não fazem parte da peça em questão, mas sim a vibrações do ambiente ou deslocamentos impróprios do sensor, como toques obtidos por colisão;
- O segundo módulo ajusta os pontos medidos a uma geometria ideal, para que os desvios da inspeção possam ser quantificados.

As máquinas de medir por coordenadas têm como método de aferição inicial, por questões conceituais, a esfera de referência [64], sendo que seu processamento é extremamente rápido além de oferecer uma maior gama de ferramentas para utilização de filtros em geral. Mais tipos de filtros serão citados adiante neste capítulo quando forem tratadas as ferramentas do sistema CAI.

2.4.1 Integração CAD-CAI

A integração dos sistemas CAI diretamente com os modelos importados de sistemas CAD surgiu como uma ferramenta que facilita o trabalho dos usuários de máquinas de medir por coordenadas.

Em relação ao custo-benefício desta ferramenta para a área de inspeção, o tempo utilizado para o desenvolvimento de uma estratégia de medição pode cair em até 75% em relação ao método tradicional [65].

Com a opção de se selecionar uma geometria diretamente de um modelo CAD, os parâmetros de comparação da inspeção são os próprios dados do modelo, onde o sistema CAI consegue então de acordo com uma tolerância pré-

estipulada, gerar um relatório com a comparação entre o modelo virtual e a peça.

Atualmente os sistemas CAIs oferecem diversos métodos para a transferência de dados do CAD, tanto para a importação dos dados de um modelo, como para a exportação de resultados de medição.

Como formato de dados que podem ser gerados através de um sistema CAI, pode-se citar:

- Uma nuvem de pontos, onde podem ser utilizados para a reconstrução de uma geometria. Esta função é o ponto principal na engenharia reversa [66];
- Forma de relatório, demonstrando a comparação entre os pontos gerados com os medidos.

Estes dois formatos são os mais utilizados para a obtenção de resultados de medição, mas com as atuais formas de integração com sistemas de outras áreas do ciclo de desenvolvimento do produto, outros tipos de dados são fornecidos com a utilização de um sistema CAI [67].

Os formatos que normalmente são aceitos para a importação ou exportação de dados entre os sistemas CAD/CAI são:

- IGES - o IGES (*Inicial Graphics Exchange Specification*), Padrão normalizado norte-americano (ANSI Y14.26), onde o modelo é transformado em informação de texto, que pode ser reconhecido por qualquer plataforma de programação conhecida [26];
- STEP - o padrão STEP (*Standard for the Exchange of Product Model Data*) [68] é uma norma ISO (*International Organization for Standartization*) sendo conhecida por ISO10303, e que abrange o desenvolvimento do produto ao longo de toda a cadeia de processo, de forma que as informações permaneçam integradas e consistentes

e não perdem sua integridade. A principal diferença entre este padrão e o padrão IGES é que todos os dados de engenharia e não somente elementos gráficos são inseridos no modelo para ser transferido. Atualmente nem todos os dados são suportados por tal norma, mas as contínuas atualizações fazem com que o mesmo fique cada vez mais completo e integrado às novas tecnologias. Atualmente é o padrão mais completo em representação do modelo, como demonstra a Tabela 2.2;

- VDA-FS - o VDA-FS (*Verband der Deutschen Automobilindustrie-Flachenschnittstelle*) foi concebido para a transferência de superfícies complexas. Isso significa que as dimensões geométricas 2D e textos são deixados de fora. Este formato de arquivo foi registrado como uma norma DIN (*Deutsches Institut für Normung*), sendo reconhecida por DIN 66301 [69];

- DES - o DES (*Data Encryption Standard*) é um antigo padrão utilizado nos Estados Unidos em meados da década de 80, com pouca utilização atualmente. Ele basicamente transforma o modelo em um sistema criptografado, fornecendo alguma segurança ao mesmo, mas limitando a transferência do mesmo para outros sistemas [70];

- Formato XYZIJK - formato básico de informações que compõem uma coordenada dotada das dimensões x, y e z além do seu vetor i, j e k. O formato em que estes vetores são apresentados é um vetor contendo primeiramente os parâmetros de posição nas coordenadas seguidas dos parâmetros de direção do vetor. Os dados podem ser obtidos de um arquivo comum de texto ou importados de uma tabela;

- DIMS - norma desenvolvida para permitir o controle de dados a serem transferidos de sistemas CAD para sistemas de aferição em geral. Um programa de inspeção criado seguindo o padrão DMIS,

pode ser exportado a qualquer MMC que tenha uma interface de aceitação neste formato [71].

Tabela 2.2: Recursos geométricos contidos nas interfaces normalizadas [72].

Tipo de Elemento	VDAFS	IGES	STEP
Ponto	X	X	X
Vetor	X		X
Reta	U	X	X
Arco de Círculo	X	X	X
Curva de secção cônica		X	X
Curva Polinomial	X	U	X
Curva B-Spline		U	X
Curva Nurbs		X	X
Plano	U	X	X
Cilindro		X	X
Cone		X	X
Esfera		X	X
Elipsoide		X	X
Superfície anelar		X	X
Superfície Polinomial	X	E	X
Superfície B-Spline		U	X
Superfície Nurbs		X	X
Curva em superfície polinomial	X	U	X
Superfície de Regulagem	E	X	X
Superfície polinomial limitada	X	U	X
Sólidos		E	X
(X=contém; E=limitado; U=contido como sub-elemento)			

Alguns sistemas CAI ainda permitem o uso de modelos nativos advindos de um sistema CAD, com isto, permite-se utilizar os dados sem a probabilidade de alterações ou perda dos mesmos, por problemas com conversões [73, 74].

Analogamente à importação, o sistema CAI permite também exportar os dados obtidos de uma inspeção para um sistema CAD. Os modelos oferecidos a isso são normalmente os mesmos que se encontram disponíveis para a importação dos dados. A seguir serão expostas algumas ferramentas que vêm facilitando cada vez mais a utilização destes *softwares*, tornando a programação de estratégias uma tarefa mais dinâmica e eficiente.

2.4.2 Ferramentas de ajuda de um sistema CAI

Como já foram citados anteriormente, os sistemas CAI vêm oferecendo cada vez mais ferramentas que ajudam ao usuário uma programação melhor e

eficiente. A seguir, serão apresentados de como as novas ferramentas podem facilitar ainda mais uma inspeção.

Neste trabalho, será utilizado o sistema CAI denominado PC-DIMS®. O software em questão oferece um método simples de execução. Ao invés de normalmente requerer a especificação do elemento que será medido, como por exemplo, um círculo ou um quadrado, ele permite que os toques sejam efetuados manualmente pelo operador através do sistema de apalpação. Quando o conjunto de toques estiver concluído pelo operador, ele finaliza a medição, onde o software CAI determina de acordo com algoritmos de estimativa, qual o elemento que está sendo gerado. A Figura 2.28 ilustra a utilização desta ferramenta na construção de um círculo. Os toques são armazenados, e os vetores resultantes do toque são apresentados. Após a confirmação do operador, os dados são então avaliados e inseridos no melhor elemento que represente a geometria final. Os dados são então mantidos na lista do programa, com as coordenadas obtidas.

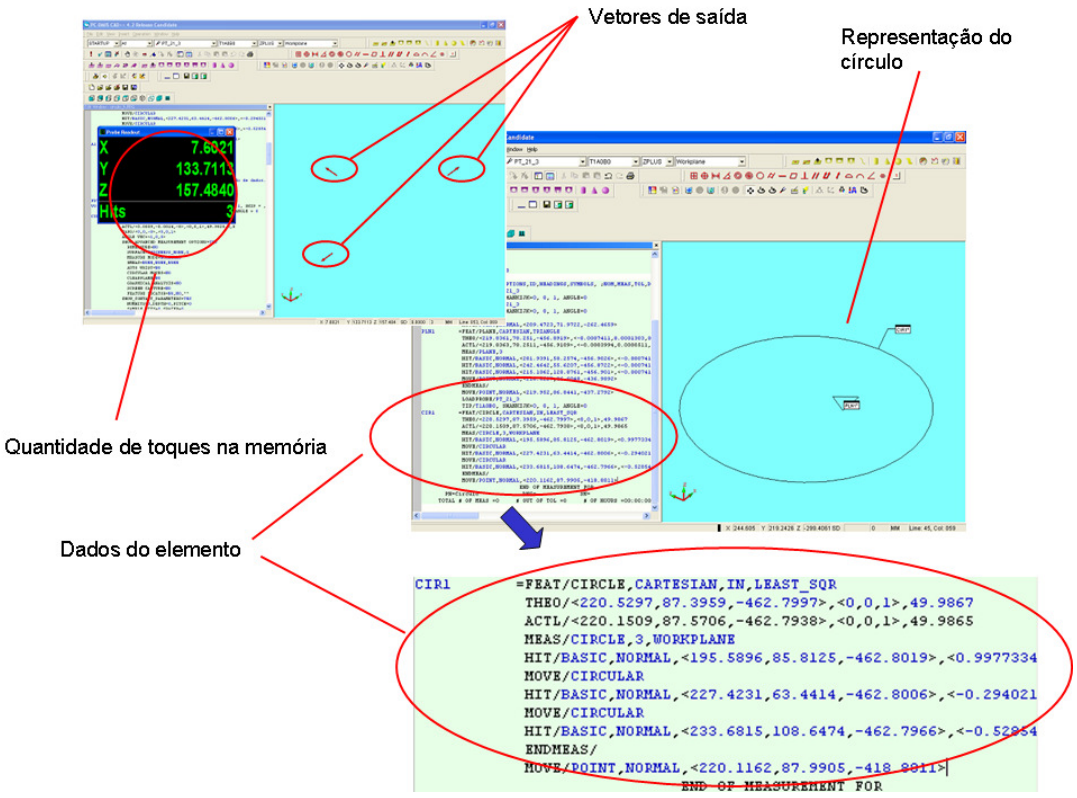


Figura 2.28: Exemplo de utilização de ferramenta de auxílio (PC-DIMS).

Para ilustrar o que foi escrito, se o conteúdo da inspeção for três toques, ambos respeitando um mesmo padrão circunscrito e sendo que o ângulo entre o primeiro e o último toque seja maior que 180º, o *software* o considera um círculo [75].

Os toques mínimos que um *software* dotado deste tipo de algoritmo de ajuda necessita antes de identificar um determinado elemento são:

- Ponto: Apenas um toque. O *software* armazena as coordenadas encontradas nesta posição;
- Linha: Dois pontos. O sistema CAI produz uma linha entre os dois pontos, criando assim uma reta;
- Círculo: Três pontos. Se estes pontos seguirem um mesmo padrão de circunferência, mas forem medidos com menos de 180° entre o primeiro e o último toque, o *software* produz um arco entre eles;
- Plano: Três pontos. Excluindo a situação de círculo, quando houver a quantidade de pontos, o sistema CAI o reconhece como um plano;

Com estes elementos básicos, praticamente qualquer elemento mais complexo pode ser inspecionado [76], com exceção dos elementos regidos por um modelo matemático Spline, como foi discutido no tópico 2.2.

Outra ferramenta importante é o que considera a montagem do cabeçote e o sensor de apalpação que está sendo utilizado. A Figura 2.29 apresenta o funcionamento desta ferramenta no momento de definição de um sistema de apalpação. Por exemplo, quando o sistema CAI mede um toque, ele compensa automaticamente o raio do apalpador, em relação à direção à normal ao toque. Isto significa que se o ponto tiver sido medido na direção de Y negativo, o software compensará automaticamente o raio do sensor em +Y [75].

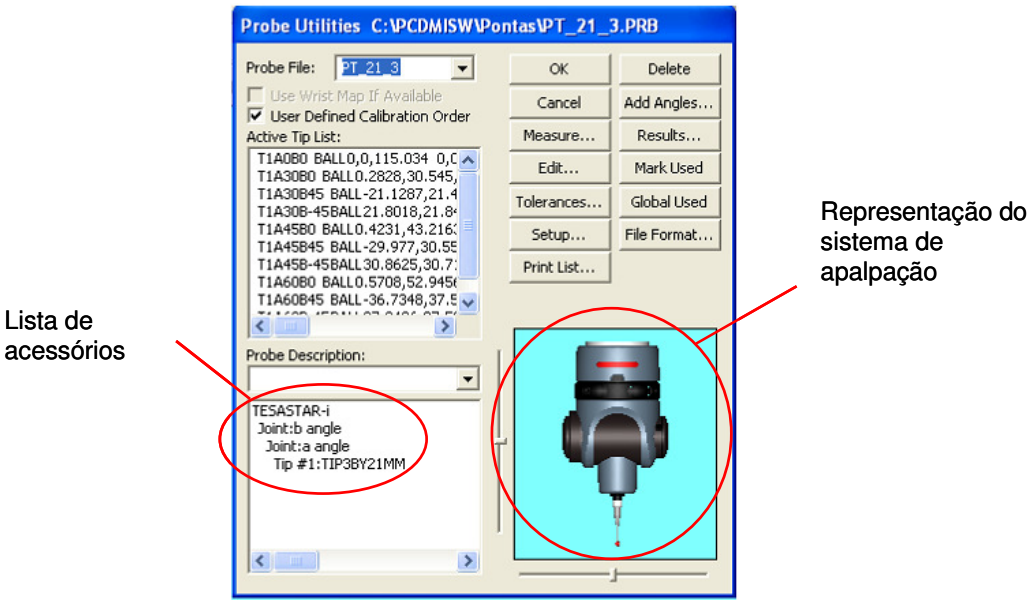


Figura 2.29: Seleção de sistema de apalpação (PC-DIMS).

2.4.3 Representação Gráfica

Quando um sistema CAI exibe uma geometria, tais representações podem advir de diversas formas. Destes modelos, podem-se exemplificar três possíveis formatos:

- De um arquivo de importação baseado em um modelo CAD;
- De valores advindos do programa da peça;
- De uma inspeção de uma peça com o mesmo modelo previamente medido.

Se a imagem vier da segunda ou terceira opção, a imagem será apenas parcial, sem todos os dados do projeto, ou melhor, a peça só conterá os elementos que foram medidos pela MMC.

Já a primeira opção apresenta o modelo completo, com todas as medidas idealizadas que serão posteriormente comparadas com os valores da inspeção como citado anteriormente neste tópico.

Outra exibição gráfica disponível ao usuário é a opção que alguns sistemas CAI têm de visualizar modelos importados do CAD em 2D em perspectiva. Para isto se tornar possível, o sistema CAI cria uma visão em perspectiva a partir do modelo 2D, os níveis tridimensionais desejados, mantendo os dados originais advindos do modelo CAD [75].

Com as opções de representação gráfica que os sistemas CAI oferecem, é possível visualizar inclusive os vetores de um toque, onde será apresentado o vetor da medição, o vetor de direção da inspeção e o vetor normal ao plano.

2.4.4 Alinhamento da peça em relação ao modelo virtual

Uma das ferramentas utilizadas constantemente neste trabalho é o alinhamento da peça em uma MMC em relação ao seu modelo virtual importado de um sistema CAD [77].

A criação de um alinhamento deste tipo no software PC-DIMS, requer a construção de dois alinhamentos: um que identifica a posição do modelo físico em relação à posição zero máquina, e outro que, sendo baseado já no primeiro alinhamento, é utilizado para igualar a posição “zero-peça” do modelo virtual à posição “zero-peça” da inspeção.

Para a construção do primeiro alinhamento, é necessária a designação de pelo menos três elementos. Um exemplo de elementos que podem ser utilizados é:

- Plano: para definir a posição zero em relação ao eixo z,
- Linha: definindo uma direção em um dos eixos (x ou y),
- Ponto: é utilizado para apontar o início do eixo restante.

Com estes elementos é possível se criar o primeiro alinhamento, para identificar o posicionamento da peça na mesa da MMC. Este procedimento é ilustrado na Figura 2.30.

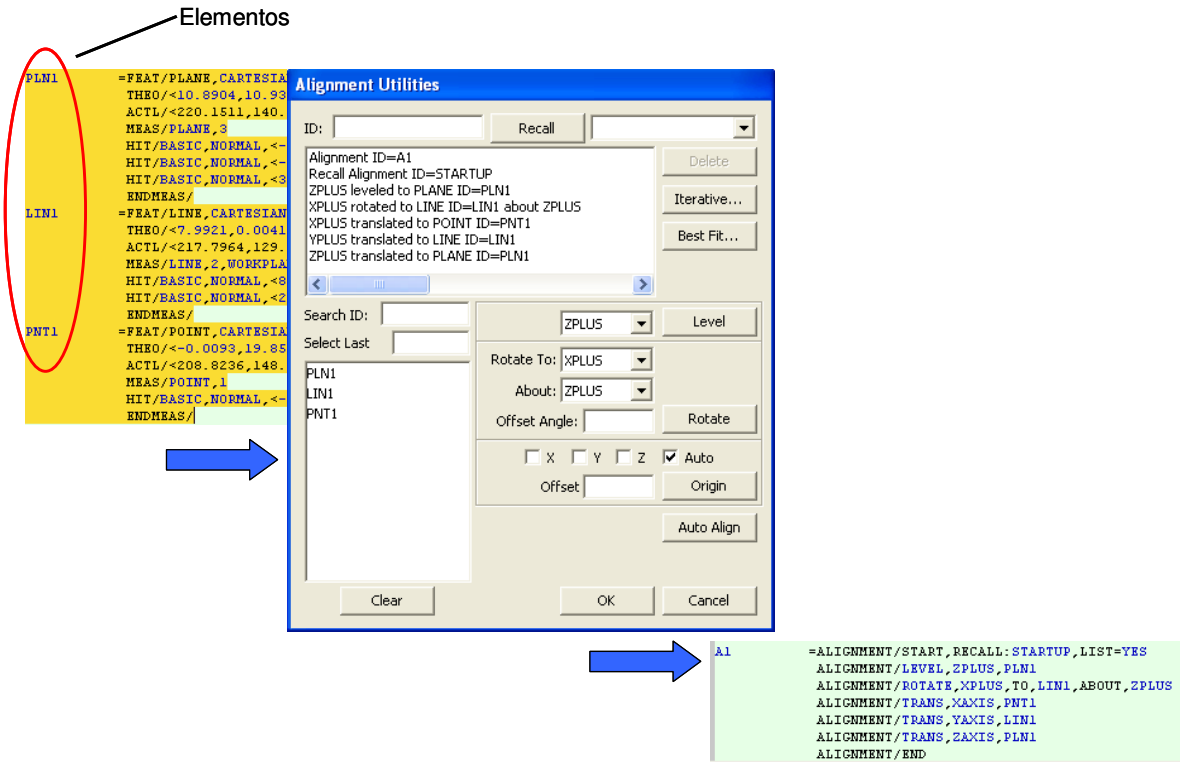


Figura 2.30: Criação de alinhamento.

Para o segundo alinhamento, primeiramente deve-se importar o modelo virtual CAD, para o software reconhecer seus dados, e então, fazer novamente os mesmos três elementos, para utilizar as coordenadas já referenciadas pelo primeiro alinhamento.

A diferença deste alinhamento em relação ao primeiro, é que após a definição de todos os fatores, o botão “PART=CAD”, deve ser utilizado, fazendo com que o software iguale o ponto “zero-peça” encontrado neste alinhamento seja igualado à do modelo virtual (Figura 2.31).

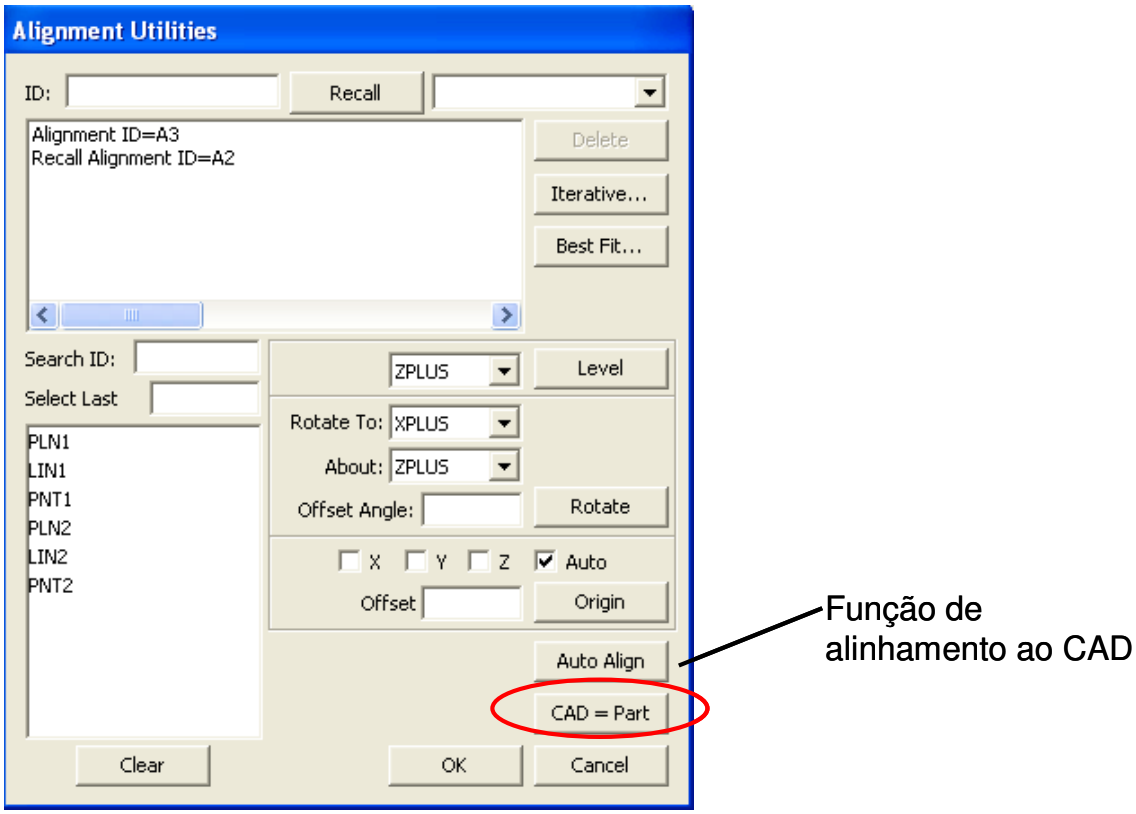


Figura 2.31: Criação de alinhamento referente ao modelo CAD.

3. Objetivo e Métodos

Neste capítulo serão expostos os objetivos desta dissertação, juntamente com os métodos utilizados para que os objetivos sejam alcançados com êxito.

3.1 Objetivos

O objetivo deste trabalho fica definido como:

Desenvolvimento de um aplicativo para sistema CAD para auxiliar a definição de pontos de inspeção de superfícies complexas em Máquinas de Medir por Coordenadas considerando diferentes estratégias de distribuição dos pontos sobre a superfície.

Este aplicativo deverá implantar pelo menos duas formas de distribuição de pontos sobre a superfície:

- Uniforme pelo comprimento da curva: os pontos serão distribuídos em espaços eqüidistantes pelo comprimento total da curva (em unidades de medida);
- Uniforme pelo parâmetro “t” de construção da curva: os pontos serão divididos uniformemente pelo valor do parâmetro “t” da curva selecionada.

Outro objetivo é analisar a possibilidade de exportação dos dados gerados pelo aplicativo para um sistema CAI de tal forma que a inspeção da superfície possa ser automatizada.

Com os objetivos propostos alcançados, pode-se preparar um documento sobre a implantação deste aplicativo no sistema CAD de tal forma que outras estratégias de distribuição de pontos para a inspeção de superfícies complexas possam vir a ser implantadas em futuros trabalhos.

Com isto, este trabalho visa oferecer uma ferramenta que suporte a definição de pontos de medição em um plano definido pelo usuário, priorizando áreas que necessitem de uma inspeção mais rigorosa, ou elementos-chave, que

realmente definem a funcionalidade da peça e que conseqüentemente necessitam de uma maior exatidão na fabricação, contribuindo assim no ciclo de desenvolvimento do produto.

3.2 Métodos

Para adquirir o conhecimento necessário sobre os temas abordados, foi realizada uma pesquisa bibliográfica sobre diferentes assuntos relacionados às áreas que se fazem pertinentes a este trabalho. Os assuntos que foram estudados:

- Sistemas CAI;
- Sistemas CAD;
- Máquinas de medir por coordenadas;
- Desenvolvimento de aplicativos;
- Modelamento de superfícies.

A escolha da linguagem de programação utilizada (linguagem C) para o desenvolvimento do aplicativo a fim de viabilizar sua inserção, foi definida por utilizar a mesma linguagem que a do sistema CAD utilizado.

Devido à flexibilidade das curvas Splines para a geração de modelos virtuais, o aplicativo utiliza seus conceitos matemáticos para os cálculos de distribuição de pontos.

O aplicativo foi desenvolvido com 2 estratégias de definição de pontos. A distribuição por espaçamento uniforme é baseada em tipos de distribuição já pesquisados [78] enquanto que a distribuição pelo parâmetro “t” da curva de curva é baseada no modelo matemático que define uma curva *Spline*, conforme vistos no Capítulo 2. A utilização deste segundo tipo de estratégia serve para representar a possível inserção de futuras estratégias ao aplicativo.

A verificação do funcionamento do aplicativo será feita através da definição de pontos de medição em um modelo geométrico no sistema CAD. Já a

verificação da utilização dos pontos gerados para a construção de uma estratégia de medição, se dá através da importação do modelo virtual juntamente com os pontos definidos pelo aplicativo. Com o reconhecimento dos pontos pelo sistema CAI, é possível então a realização de uma inspeção baseada na medição dos mesmos.

4. Desenvolvimento e implementação do aplicativo

A implantação do aplicativo para o Software Siemens NX, se deu por meio da criação de uma biblioteca com vínculos dinâmicos (DLL), tendo todas as suas funções em um único arquivo compilado que pode ser acessado pelo sistema CAD.

Para o desenvolvimento do aplicativo, foi utilizado o próprio API existente no *software* NX. A decisão de uso deste sistema CAD se deve a familiaridade e disponibilidade do mesmo para o seu uso. A escolha da utilização da linguagem C como via de programação se deve ao maior conhecimento de programação nesta linguagem e a disponibilidade do API, que também oferece a possibilidade de programação em VB.Net e C++. A criação da biblioteca foi feita com o uso do software Microsoft Visual C++, que é o compilador utilizado pelo API.

4.1 Desenvolvimento de aplicativos

Como visto anteriormente no capítulo 2, existem *softwares* que permitem a criação de novas funções, viabilizando o uso de operações internas já existentes em conjunto com um aplicativo implantado. O *software* utilizado neste trabalho, o sistema CAD/CAM Siemens NX, conta com uma ferramenta que permite que estes aplicativos sejam produzidos.

Para o início do desenvolvimento de um aplicativo para este sistema CAD, é necessário o uso de sua ferramenta, o *User Interface Styler*, que permite criar uma interface de comunicação com o usuário, juntamente com sua programação de funcionamento.

Dentro do próprio software, encontra-se a opção de uso desta ferramenta. Para sua utilização, deve-se ir ao menu principal e na opção “*Application*”, acessá-lo, conforme Figura 4.1.

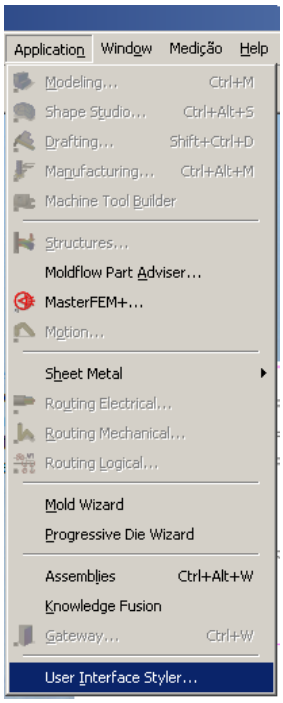


Figura 4.1: Acesso ao User Interface Styler.

Através desta opção, a janela de criação de uma interface visual com o usuário é apresentada (Figura 4.2), contendo as diversas funções que podem ser agregadas e posteriormente vinculadas a alguma operação que o usuário desenvolva.

Conforme estas funções são selecionadas, o resultado visual já é apresentado na caixa de interface e fica visível na lista de funções. Com o editor, é possível alterar textos, mensagens e os nomes de chamada de cada função que serão utilizados posteriormente na programação.

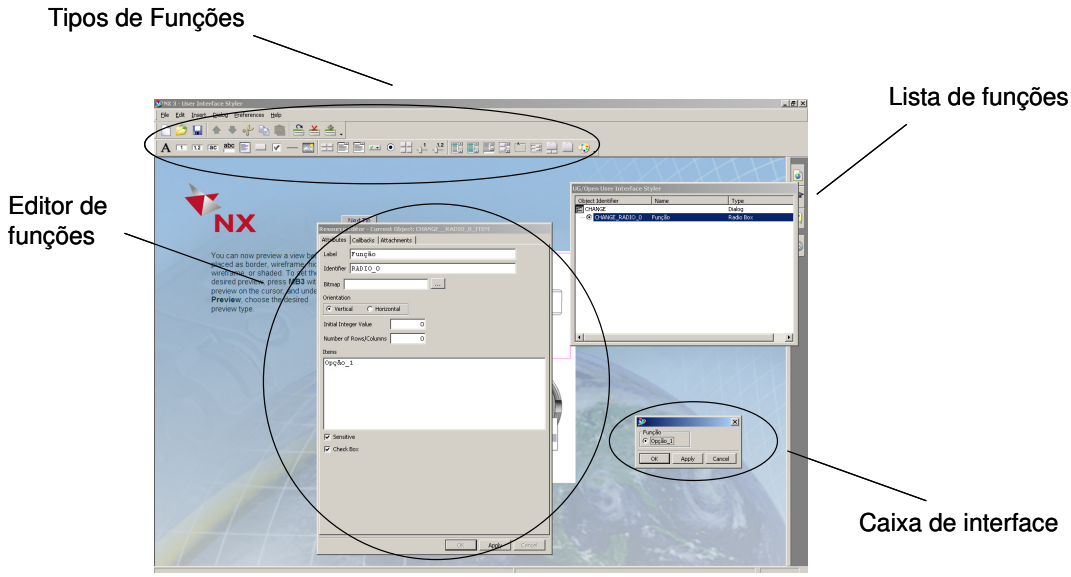


Figura 4.2: Janela de construção do User Interface Styler.

Após a criação da caixa de interface, quando o arquivo é salvo, três arquivos com extensões diferentes são salvos:

- *.dlg - é a caixa de diálogo em si;
- *.h - o cabeçalho utilizado posteriormente para a programação;
- *.c -o programa em si em linguagem C, C++ ou dot Net.

Após a criação destes arquivos, o Siemens NX pode ser desligado para o início da fase de programação. O software utilizado neste trabalho para esta fase é o Microsoft Visual C++, devido a sua viabilidade de construir bibliotecas de vínculos dinâmicos para programas gerados para uma plataforma Windows, cuja extensão é “.dll”.

Iniciando um novo projeto, têm-se a opção de denominar o projeto e selecionar o tipo de resultado que o mesmo gera. Para o caso de desenvolvimento de um aplicativo para o sistema Siemens NX, a opção correta é um arquivo do tipo DLL para Win 32. Após esta definição, na tela seguinte, a opção de um projeto vazio deve ser selecionada (Figura 4.3).

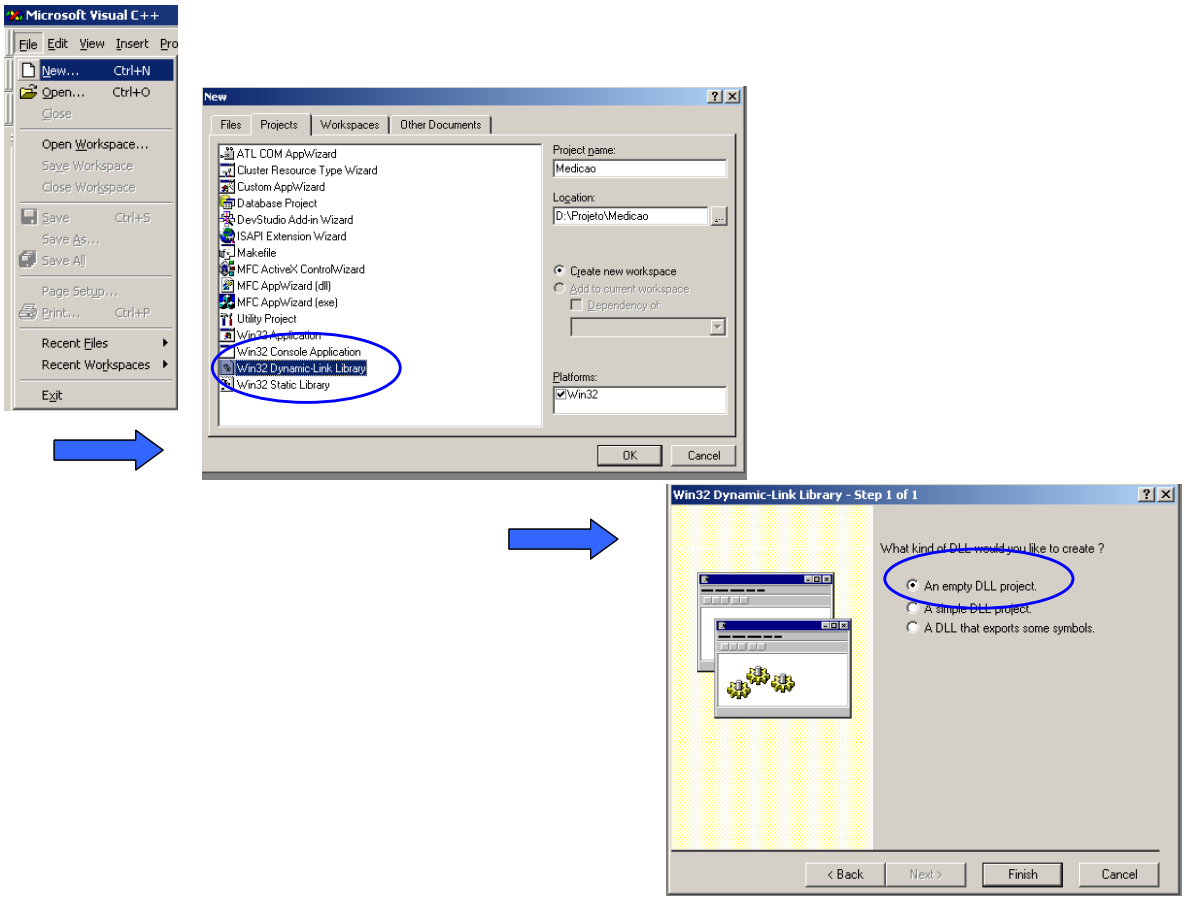


Figura 4.3: Seqüência de criação de novo projeto.

No ambiente de trabalho do projeto, na opção "*File View*", os arquivos gerados pelo API (extensões "*.c" e "*.h") devem ser inclusos nas pastas "*Source Files*" e "*Headers Files*" respectivamente (Figura 4.4).

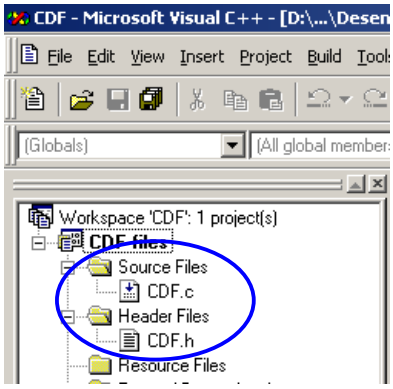


Figura 4.4: Inclusão de arquivos no projeto.

Nas configurações do projeto são necessárias algumas adições ao convencional que o próprio software utiliza. Na opção "*Project->Settings*", pode-se alterar os dados necessários para a correta compilação. Selecionando a

opção “*Debug*”, o próprio Siemens NX deve ser utilizado como executável (Figura 4.5).

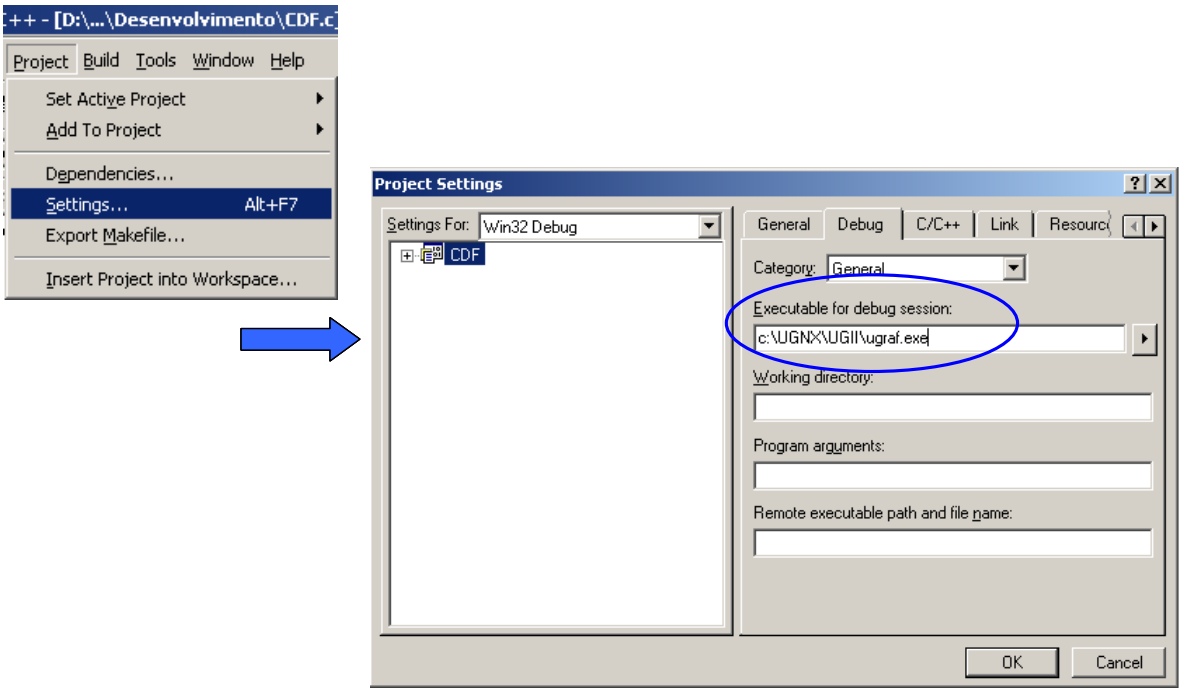


Figura 4.5: Configuração do programa como executável.

Ainda em “*Settings*”, na opção “*Link*”, para o caminho do resultado da compilação, deve ser indicado à pasta “*Application*”, gerada anteriormente na pasta principal do projeto. As bibliotecas de informações do próprio API devem ser indicadas em “*object/libraries modules*”.

Elas são:

- ugopen\libufun.lib;
- ugopen\libopenintpp.lib;
- ugopen\libugopenint.lib.

Estas bibliotecas devem ser precedidas do restante do caminho em que se encontra o software Siemens NX. A Figura 4.6, ilustra como exemplo, a configuração destas opções para um aplicativo que gera um arquivo DLL denominado “CDF” e a inserção das bibliotecas para um software Siemens NX instalado no caminho apropriado, neste caso em: “c:\UGNX”.

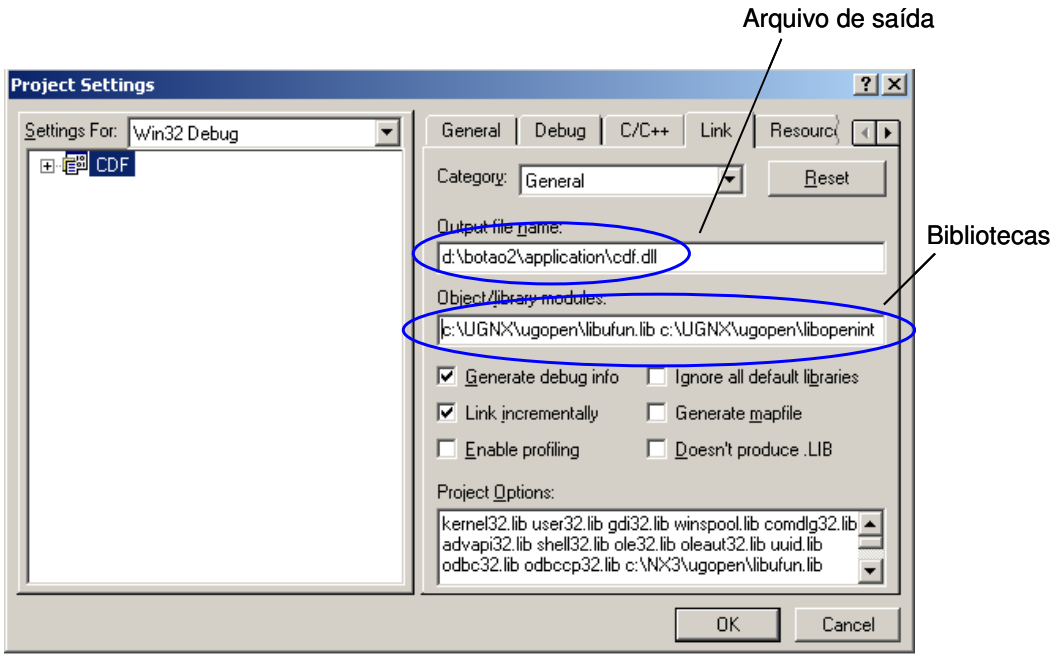


Figura 4.6: Configuração das bibliotecas do API.

O próximo passo é dado pela disposição em que os arquivos devem se encontrar para que a compilação ocorra corretamente. Na pasta onde estão os arquivos gerados pelo NX, devem ser criadas duas pastas: uma denominada "*Application*" e outra com o nome "*Startup*".

A criação do menu que será apresentado pelo sistema Siemens NX, é gerado através de um texto existente no arquivo de extensão "*.c" (Figura 4.7). Este conteúdo é destinado especificamente para desenvolver o caminho de inclusão ao menu do CAD. O texto deve ser copiado integralmente em um arquivo do tipo bloco de notas na pasta "*Startup*".

Para o nome do arquivo deve-se utilizar o mesmo nome do projeto, com a extensão ".men". Para nomeá-lo devem-se utilizar aspas para a alteração do tipo do arquivo de bloco de notas para formato "men". Para exemplificar, o arquivo CDF gerado anteriormente, seria denominado "CDF.men".

Neste texto estão as informações da posição em que o aplicativo estará no menu e qual o formato que ele utilizará, como por exemplo o efeito cascata. Todas estas informações podem ser alteradas de acordo com a necessidade do usuário.

```
Move the contents between the dashed lines to your Menuscript file.
!   ------------------------------------------------------------
VERSION 120

EDIT UG_GATEWAY_MAIN_MENUBAR

BEFORE UG_HELP
  CASCADE_BUTTON UISTYLER_DLG_CASCADE_BTN
  LABEL Dialog Launcher
END_OF_BEFORE

MENU UISTYLER_DLG_CASCADE_BTN
  BUTTON HW_BTN
  LABEL Display HW dialog
  ACTIONS HW.dlg
END_OF_MENU
!   ------------------------------------------------------------
```

Figura 4.7: Texto contendo informações para a criação do menu.

Por fim, é necessário criar a base e os caminhos onde os dados serão requeridos para a compilação. Inicialmente, o caminho que o software Siemens NX deve encontrar as informações do aplicativo devem ser informados por meio de uma variável de ambiente. Utilizando uma plataforma Windows, na opção de “Propriedades” do ícone “Meu computador” há uma opção denominada “Avançado”. Na opção “Variáveis de Ambiente”, deve-se criar uma nova variável denominada “UGII_USER_DIR”, e o seu valor correspondendo ao local em que se encontra o aplicativo (Figura 4.8).

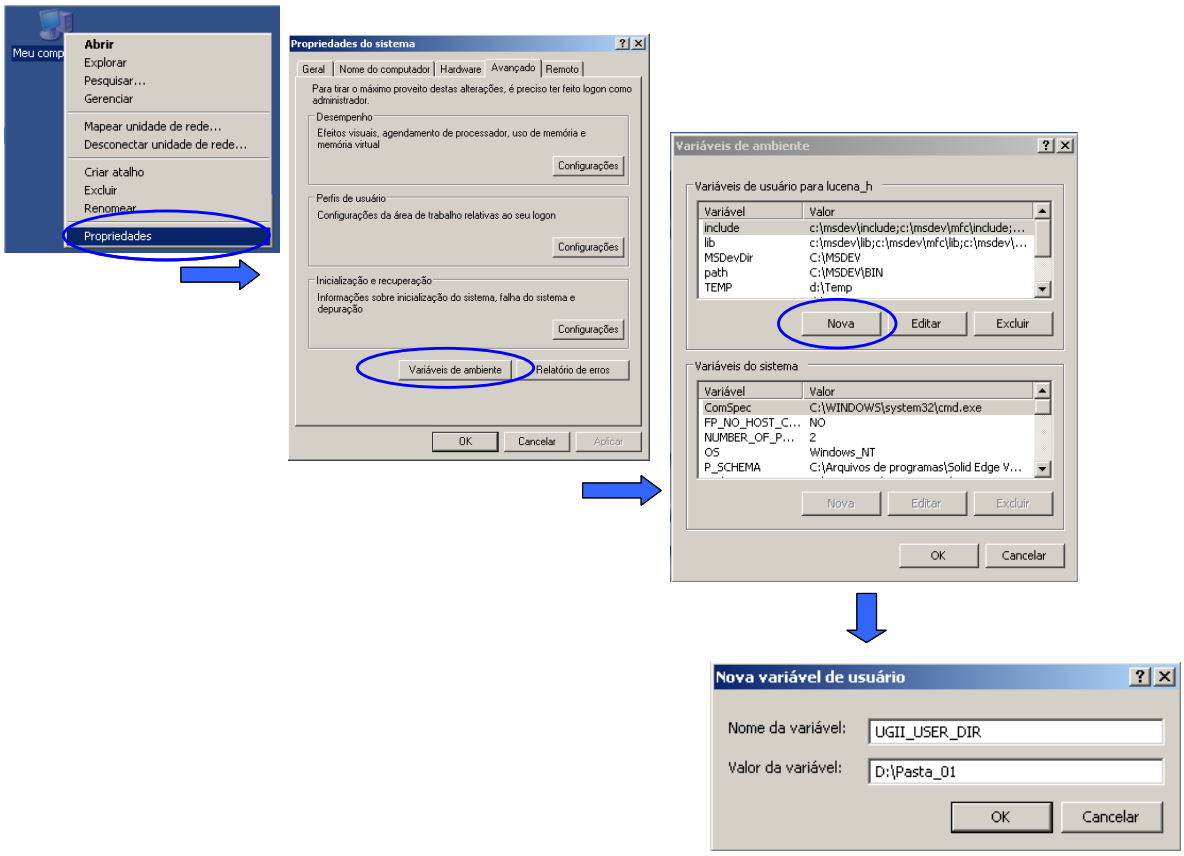


Figura 4.8: Configuração de Variáveis de Ambiente.

Após estes tópicos, ao se compilar o projeto, ele irá gerar o arquivo no formato DLL que será reconhecido pelo software CAD como uma nova função.

4.2 Desenvolvimento de aplicativo em sistema CAD para definição de pontos de inspeção

Para que o aplicativo possa desempenhar as funções que foram designadas, primeiramente foi necessária a utilização de um algoritmo que consiga calcular qualquer caminho de medição (secção desenvolvida previamente) selecionado pelo usuário.

Como visto no capítulo de revisão bibliográfica, o modelo matemático de curvas *Splines* permite representar formas complexas, sendo então o modelo matemático utilizado para base de cálculos do aplicativo. O algoritmo utilizado neste trabalho é baseado no modelo matemático desenvolvido por Rogers [33]. O código fonte do aplicativo desenvolvido se encontra no Anexo A.

Os dados utilizados pelo aplicativo são extraídos do próprio modelo virtual. Estes dados são obtidos por meio das funções da interface de programação do próprio software.

Com estes dados é possível então a definição de pontos por este caminho de medição, de acordo com estratégias de distribuição.

A escolha da primeira estratégia é baseada no próprio modelo de construção que uma curva *Spline* através da divisão do parâmetro “t”, sendo seu valor 0 para o ponto inicial da curva e variando até 1 para seu último ponto.

O passo (valor que define a variação de espaço entre os pontos) é definido de acordo com a divisão do valor do parâmetro t por uma quantidade de pontos definida pelo usuário.

Após a definição do primeiro ponto, o passo é então somado ao valor de passo anterior para o próximo ponto.

Para cada passo, são calculadas as funções *Rbasis* para cada um dos pontos do polígono de controle. As funções *Rbasis* dos polígonos de controle mais próximas do passo terão um valor de influência maior que os outros. Estas funções são então multiplicadas pelas coordenadas de seus respectivos pontos do polígono de controle. Por fim, a posição do ponto é dada pela soma vetorial destes resultados Figura 4.9. Para as equações e dados deste exemplo, vide Anexo B.

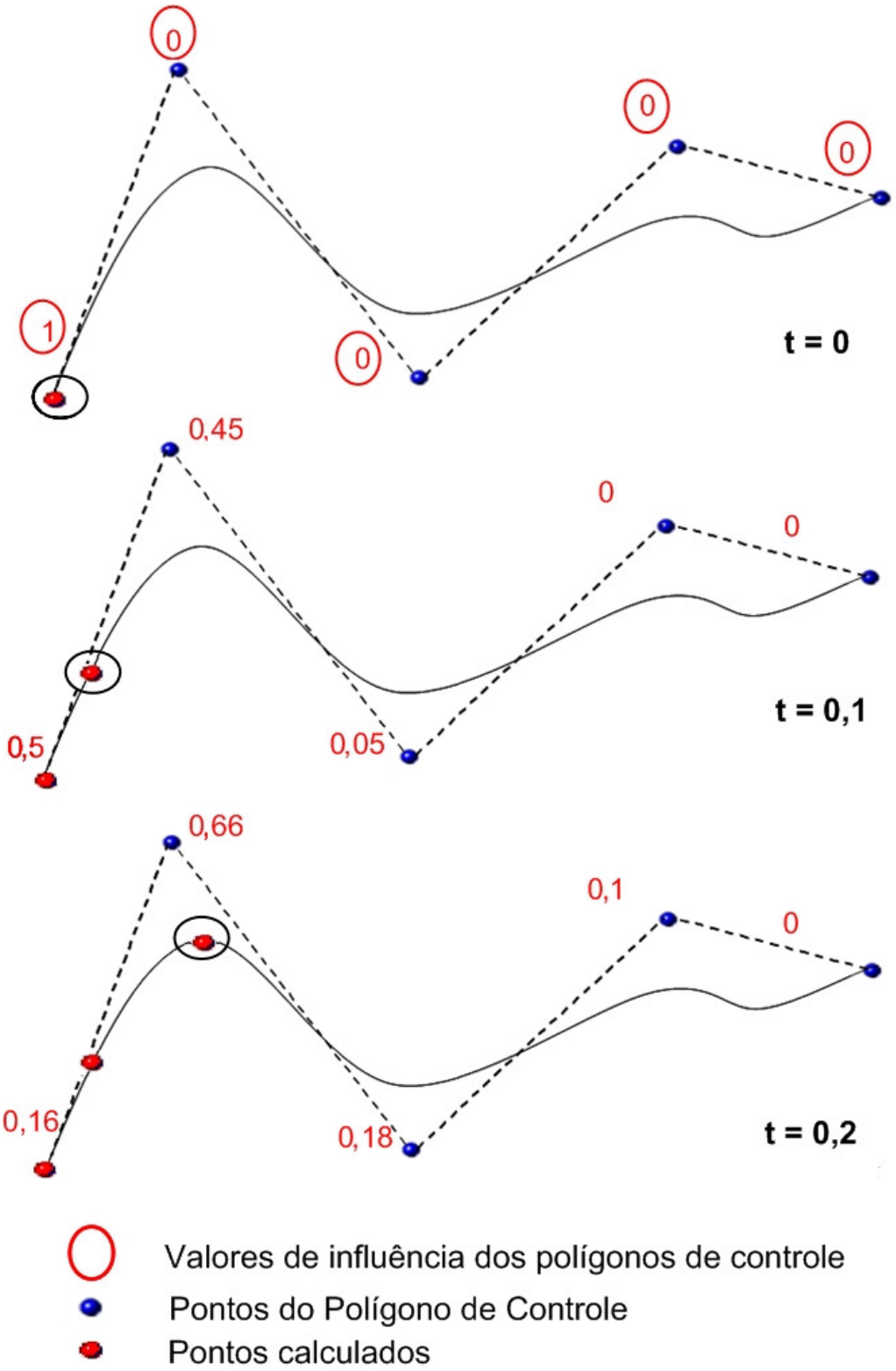


Figura 4.9: Influência dos polígonos de controle no cálculo de posicionamento dos pontos.

Com esta estratégia é possível então distribuir mais pontos na região em que contiver mais pontos no polígono de controle.

Para a segunda estratégia, o comprimento total da curva é obtido por uma função existente no próprio API do software. Com este valor adquirido, o aplicativo o distribui pelo número de pontos estabelecido pelo usuário. Estes pontos serão distribuídos em espaços eqüidistantes pela curva, sendo o primeiro ponto definido na posição inicial da curva e a cada ponto subseqüente, um passo é calculado para que o comprimento seja o mesmo entre todos os pontos, tendo por fim um ponto definido na posição final da curva.

4.2.1 Lógica de funcionamento do aplicativo

Ambas as estratégias presentes são baseadas na matemática de definição de uma curva NURBS. Após a seleção do caminho (curva) e da face, com a quantidade de pontos a serem calculados e a estratégia de distribuição definida, o aplicativo busca do próprio modelo virtual do sistema CAD, os dados necessários para o início dos cálculos (Através de APIs do próprio sistema CAD) (Anexos C e D). Os dados relativos à curva são:

- Ordem da curva;
- Pólos (vértices do polígono de controle),
- Peso (peso de cada um dos pólos);
- Vetor de nós da curva.

Após a obtenção destes dados, inicia-se o algoritmo de cálculos. O primeiro cálculo (Figura 4.10) determina a função *Rbasis* de primeira ordem para todos os nós, como visto anteriormente no Capítulo 2.

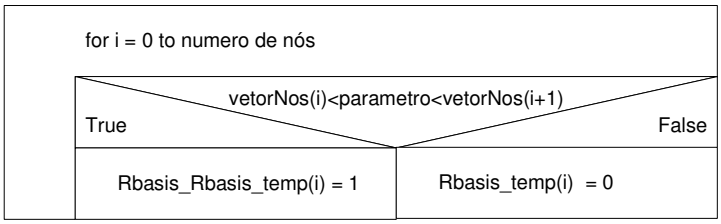


Figura 4.10: Cálculo de primeira ordem de Rbasis.

Em seguida, são calculadas em duas partes (primeiro termo e segundo termo), as funções *Rbasis* para as demais ordens da curva. Após este passo, os pesos

de cada ponto do polígono são multiplicados por seu respectivo valor da função *Rbasis* (Figura 4.11).

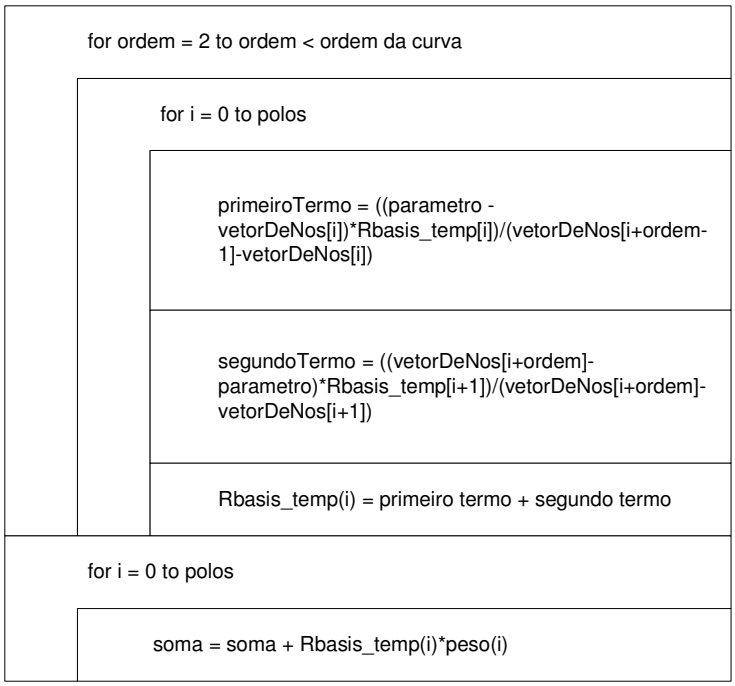


Figura 4.11: Cálculo das demais ordens de Rbasis.

No 3º passo, é utilizado um método de normalização para que todos os valores estejam inclusos entre zero e um. Este passo é necessário para que os dados estejam de acordo com o parâmetro “t” da curva (Figura 4.12).

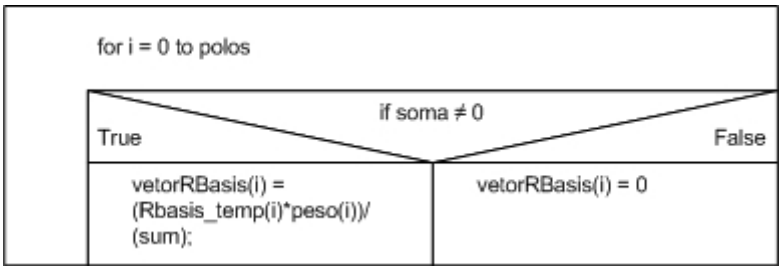


Figura 4.12: Normalização dos resultados.

O último passo do algoritmo é multiplicar as coordenadas com os respectivos *Rbasis* de cada ponto do polígono de controle, dando então a influência daquele vértice à posição final do ponto calculado (Figura 4.13).

Com os pontos calculados, a função finaliza e retorna para a estratégia definida pelo usuário.

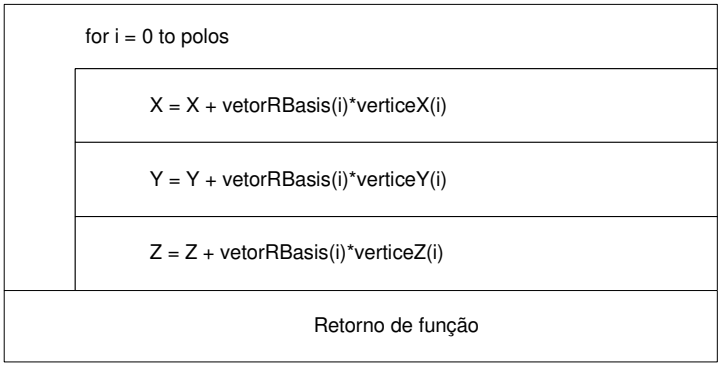


Figura 4.13: Cálculo dos pontos.

Deste modo, o valor de um ponto em uma curva com quatro vértices no polígono de controle, com grau três e vetor de nós distribuído uniformemente, é calculado da seguinte forma: é feito o cálculo das funções de suavização de primeira ordem para todos os nós (fig. 4.10), em seguida são calculadas as funções de suavização de demais ordens (fig. 4.11). A somatória das funções de suavização é normalizada (fig. 4.12), obtendo-se assim os valores de influência de cada um dos vértices do polígono de controle. Estes valores são multiplicados pelas coordenadas dos vértices do polígono de controle (fig.4.13), para finalmente obter-se o valor do ponto desejado, como pode ser visto na figura 4.14.

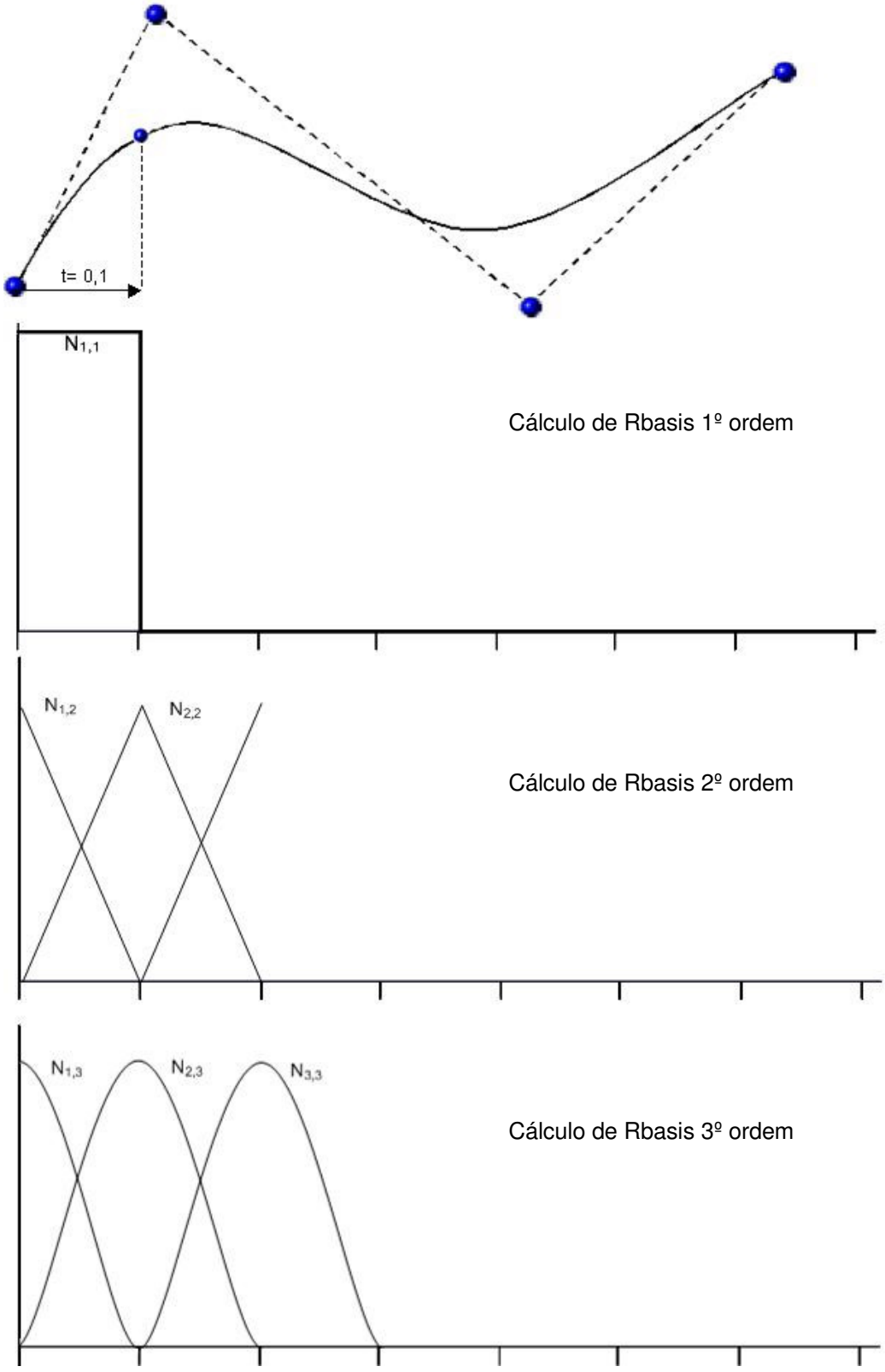


Figura 4.14: Exemplo de cálculo de ponto e suas funções Rbasis.

4.2.2 Estratégia de distribuição pelo parâmetro “t”

Para a estratégia de distribuição pelo parâmetro “t” (Figura 4.15), a função utiliza os pontos gerados pela função de cálculo dos pontos, calculando para cada um deles, o vetor normal em relação à face selecionada. Com estes dados (pontos com seus respectivos vetores), a estratégia os cria no mesmo modelo virtual do sistema CAD.

Estratégia de distribuição pelo parâmetro “t”
Função Cálculo Pontos
Função Cálculo Vetores
Cria pontos com vetores

Figura 4.15: Estratégia de distribuição pelo parâmetro “t” da curva.

4.2.3 Estratégia de distribuição pelo comprimento da curva

Para a estratégia de distribuição uniforme pelo comprimento da curva (Figura 4.16), se tem como ponto de partida a obtenção do comprimento total da curva (Anexo E). Com este valor, obtém-se o valor de passo, dividindo-o pelo número de pontos a ser criado.

Para cada um destes pontos, temos dois parâmetros utilizados para o cálculo:

- Estimativa Inferior;
- Estimativa Superior.

A função inicia-se com os valores destes parâmetros para o primeiro ponto da seguinte forma: zero para a estimativa inferior e o valor da posição do último nó da curva para a estimativa superior (valores dados em relação ao parâmetro “t” da curva, para o uso na função de cálculo de pontos). Enquanto a diferença destes valores for menor que um valor de tolerância estipulado como irrelevante, o aplicativo recalcula somente o comprimento até o ponto em questão. Este valor (comprimento parcial) é então comparado ao passo total, sendo o seu valor para o primeiro ponto igual à zero. Se o comprimento parcial

for maior, então o valor da Estimativa Inferior é alterado, senão será o valor da Estimativa Superior que será alterado.

Após a criação deste ponto, soma-se o valor de Passo ao Passo_Total para o cálculo do próximo ponto.

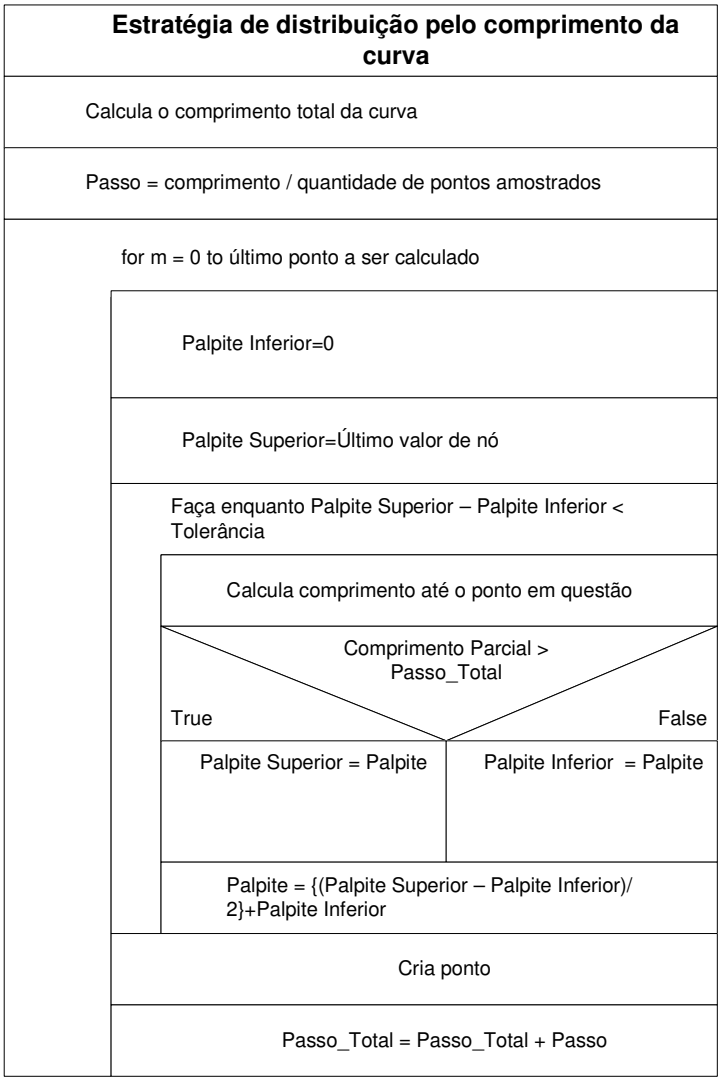


Figura 4.16: Estratégia de distribuição pelo comprimento da curva.

A Figura 4.17 ilustra o uso desta estratégia para o cálculo de um ponto. Para este exemplo, o passo foi definido como sendo 20 mm, portanto serão calculados pontos na curva a cada 20 mm. Na 1ª etapa, é definido o valor um (fim da curva) garantindo assim que o ponto a ser encontrado está entre estes dois valores, sendo esta a primeira etapa do cálculo por aproximações sucessivas.

Com estas considerações iniciais o comprimento parcial (valor calculado do comprimento da curva da estimativa inferior até a estimativa superior) é de 100 mm, ou seja, o comprimento total da curva. (2ª etapa)

Como o valor do comprimento parcial é maior que o passo definido, uma nova estimativa superior é calculada, considerando o valor médio entre as estimativas anteriormente definidas – se o comprimento parcial é maior o ponto desejado está aquém da estimativa superior. Um novo valor de comprimento parcial é calculado, por ainda ser maior que passo, uma nova estimativa superior deve ser considerada. (3º etapa).

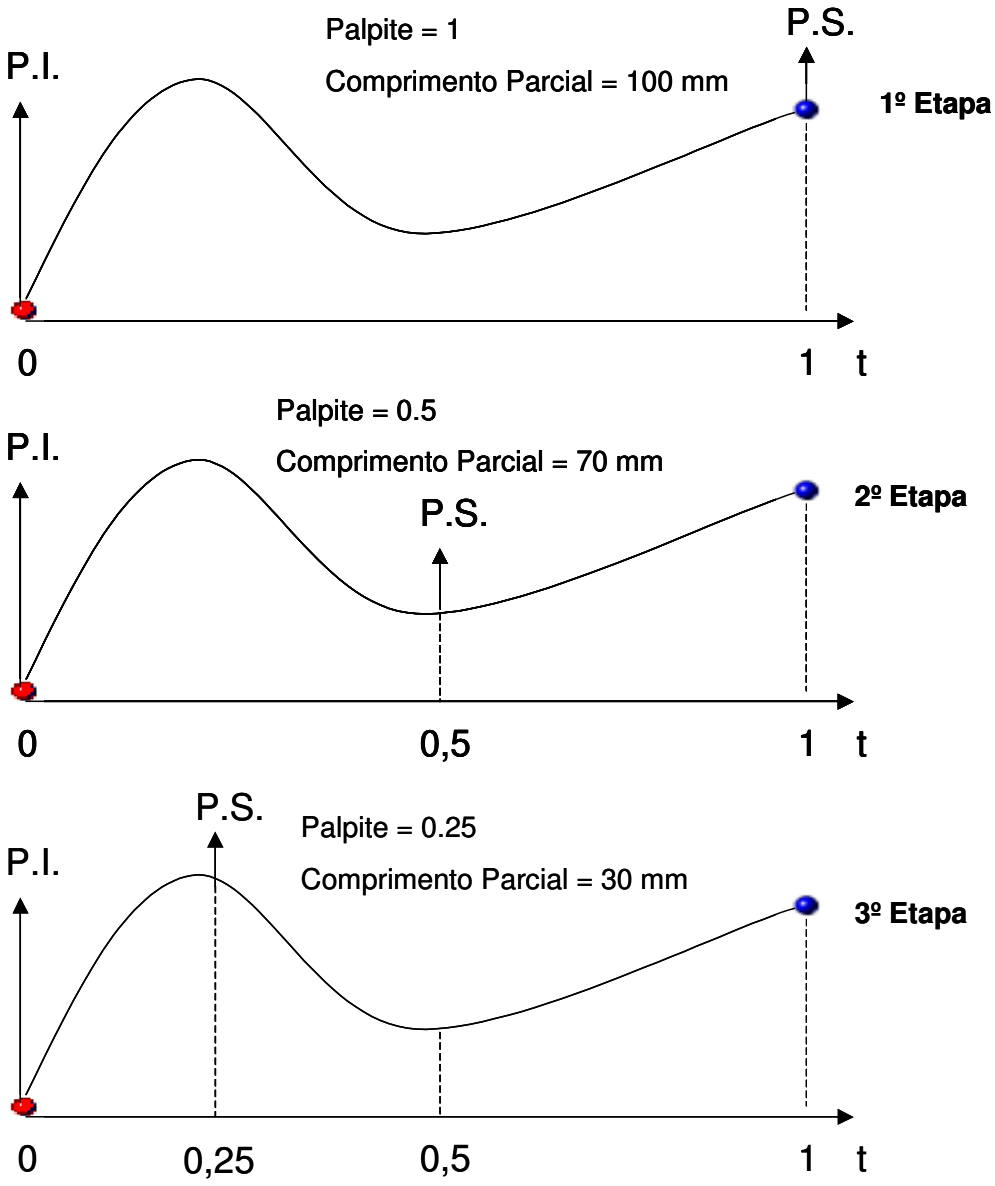


Figura 4.17: 1º Parte de exemplo de cálculo de ponto (distribuição pelo comprimento).

Na 4º etapa, como o valor do comprimento até o novo Estimativa (0.125) é menor que o Passo_Total (20 mm), então o valor alterado é o da Estimativa Inferior. Para a 5º etapa, novamente um novo Estimativa é calculado, avaliado e, tendo o valor do Parâmetro Inferior alterado (Figura 4.18).

Este processo de aproximação se dá até que os valores estejam próximos o suficiente para que a Estimativa calculada a seguir, tenha um valor desprezível fisicamente (6º etapa).

Por fim, o ponto é criado nesta posição e o aplicativo soma ao Passo_Total o valor de Passo, dando início ao cálculo do próximo ponto.

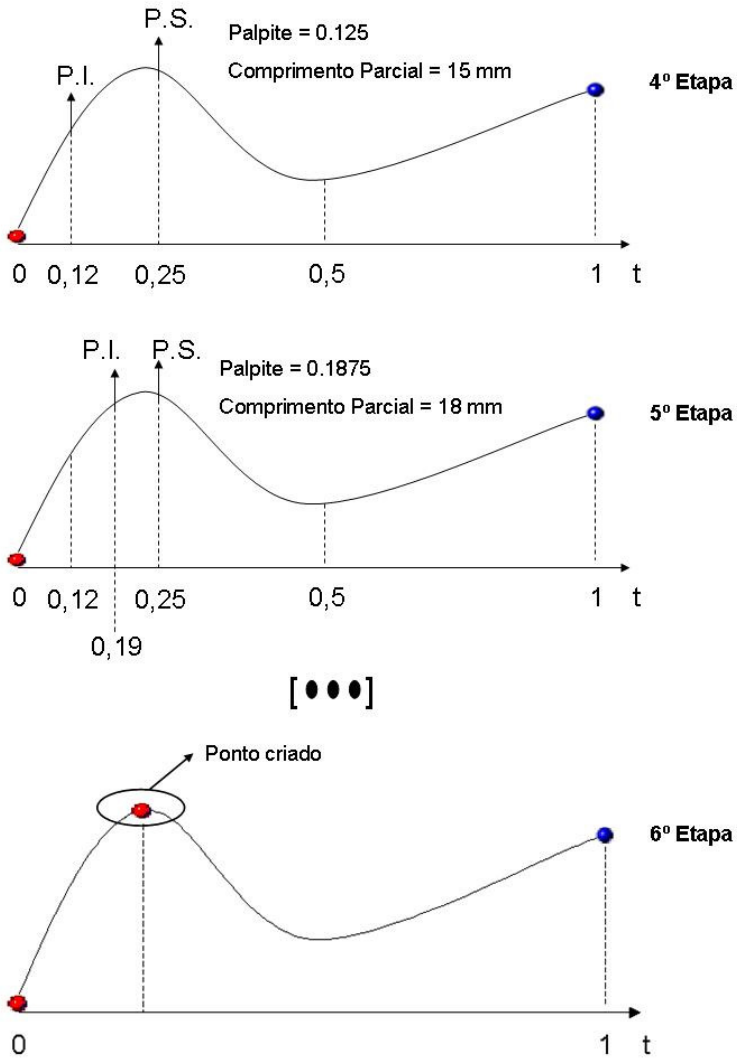


Figura 4.18: 2º Parte de exemplo de cálculo de ponto (distribuição pelo comprimento).

Com os pontos calculados, assim como na estratégia anterior, os vetores são calculados em relação à face selecionada (Figura 4.19).

Estratégia de distribuição pelo comprimento da curva
Função Cálculo Vetores
Cria vetores

Figura 4.19: Criação de vetores para a estratégia de distribuição pelo comprimento da curva.

4.3 Exportação de dados para o sistema CAI

Para o reconhecimento dos pontos gerados no *software* PC-DIMS, o aplicativo gera automaticamente um arquivo denominado “saída.DIMS”, onde os dados são dispostos em uma sintaxe que o sistema CAI reconheça como uma programação de inspeção. A Figura 4.20 ilustra o modelo de um arquivo gerado pelo aplicativo. Para uma visualização de um arquivo-exemplo de 10 pontos, vide Anexo F.

```
F(PM1)=FEAT/POINT,CART,6.467532,10.562838,19.723828,-0.206382,-0.382874,0.900452
MEAS/POINT, F(PM1),1
  PTMEAS/CART,6.467532,10.562838,19.723828,-0.206382,-0.382874,0.900452
ENDMES
```

Figura 4.20: Exemplo de ponto gerado no arquivo DMIS.

O formato reconhecido pelo sistema CAI deve ter uma disposição onde, exista um nome referente a cada ponto gerado, seguido pela designação de seu modelo, ou seja, uma *feature* de formato ponto. Seguindo o seu conteúdo, existe a informação de que os dados estão dispostos em um modelo cartesiano, seguido de suas coordenadas x, y e z e os vetores i, j e k.

A próxima linha especifica o comando para que o ponto dado seja medido, e com quantos toques são necessários para fazê-lo, no caso de um ponto, apenas um toque é requerido, como visto na formação de elementos no software PC-DIMS no tópico 2.4.2.

Por fim, se têm a informação de resultado do ponto, onde as coordenadas e vetores serão os mesmo que o nominal, pois eles ainda não foram inspecionados. Os dados relacionados à linha em questão, após importado para o PC-DIMS e utilizados para a inspeção, serão alterados para o resultado obtido na medição daquele ponto. Já os dados existentes na primeira linha, serão considerados os valores nominais.

Com os dados expostos desta maneira, podem ser importados ao software pelo caminho *File->Import->DIMS*, ilustrado na Figura 4.21. Para uma melhor eficácia do uso dos dados na importação, o programa já deve estar com a seleção do sistema de apalpação escolhido e o alinhamento em relação ao modelo virtual definidos, conforme vistos nos tópicos 2.4.2 e 2.4.5 respectivamente.

No modo de importação DMIS, a opção "*Merge"* deve ser selecionada para a inclusão dos dados após os dados já existentes no programa (como o alinhamento da peça por exemplo). Após a inclusão dos pontos, um novo alinhamento deve ser feito, tomando como referência o alinhamento anterior (alinhamento feito em relação ao modelo virtual). Este novo alinhamento é necessário para alinhar os pontos em relação ao modelo CAD. Com isto, os pontos são então reconhecidos na posição correta.

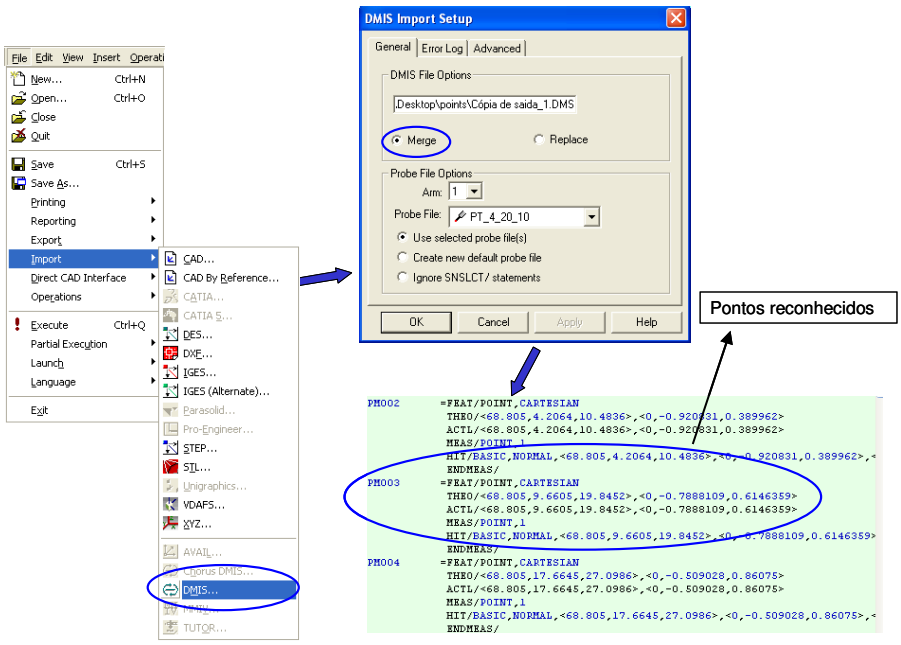


Figura 4.21: Caminho de importação de arquivo DMIS (Software PC-DIMS).

4.4 Interface com o usuário

Com a possibilidade da inserção de aplicativos em sistemas CAD, foi possível a inserção de muitas funções específicas para as mais diversas tarefas que os usuários venham a precisar.

No entanto, é necessário que estas novas funções tenham uma interface amigável, permitindo uma utilização rápida e simples às suas funções.

Para o aplicativo proposto para este trabalho, foi estruturada uma interface simples e com acesso lógico às suas operações (Figura 4.22). Esta estrutura foi baseada no próprio menu do sistema CAD, onde o usuário seleciona uma aplicação e todas as funções relacionadas a ela são apresentadas.

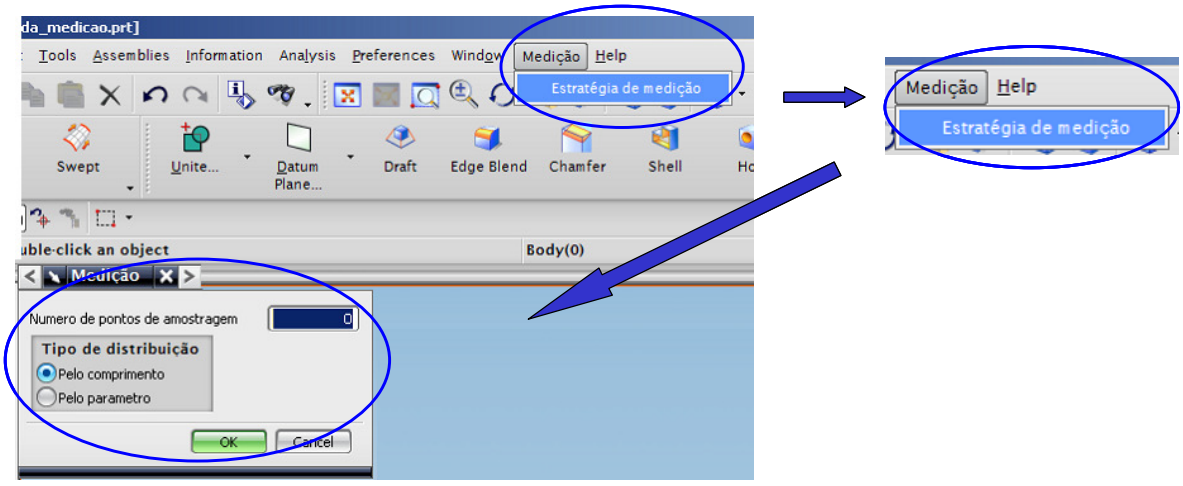


Figura 4.22: Interface com o usuário.

Para a utilização do aplicativo o usuário desempenhará funções de ação em cinco pontos importantes:

- Criação do caminho de medição;
- Número de pontos;
- Definição de estratégia de distribuição;
- Seleção do caminho de medição e a face do modelo CAD onde os pontos serão posicionados.

Em um determinado modelo virtual pré-concebido no sistema CAD, ele irá produzir uma curva sobre a superfície a ser inspecionada utilizando os métodos que o sistema CAD já dispõe para esta finalidade (como a opção *Curve on Surface*). Nesta curva estarão os pontos criados para medição, e que posteriormente a MMC irá seguir para inspecionar o produto.

Com a utilização do aplicativo, o usuário irá definir a quantidade de pontos a serem criados e o tipo de estratégia em que serão distribuídos ao longo da curva criada anteriormente. Com estes dados, o aplicativo consegue calcular as coordenadas de cada ponto e os seus respectivos vetores em relação à superfície, distribuindo-os então no modelo virtual. A Figura 4.23 ilustra a seqüência de passos necessários, incluindo as funções que dependem da interação com direta com o usuário.

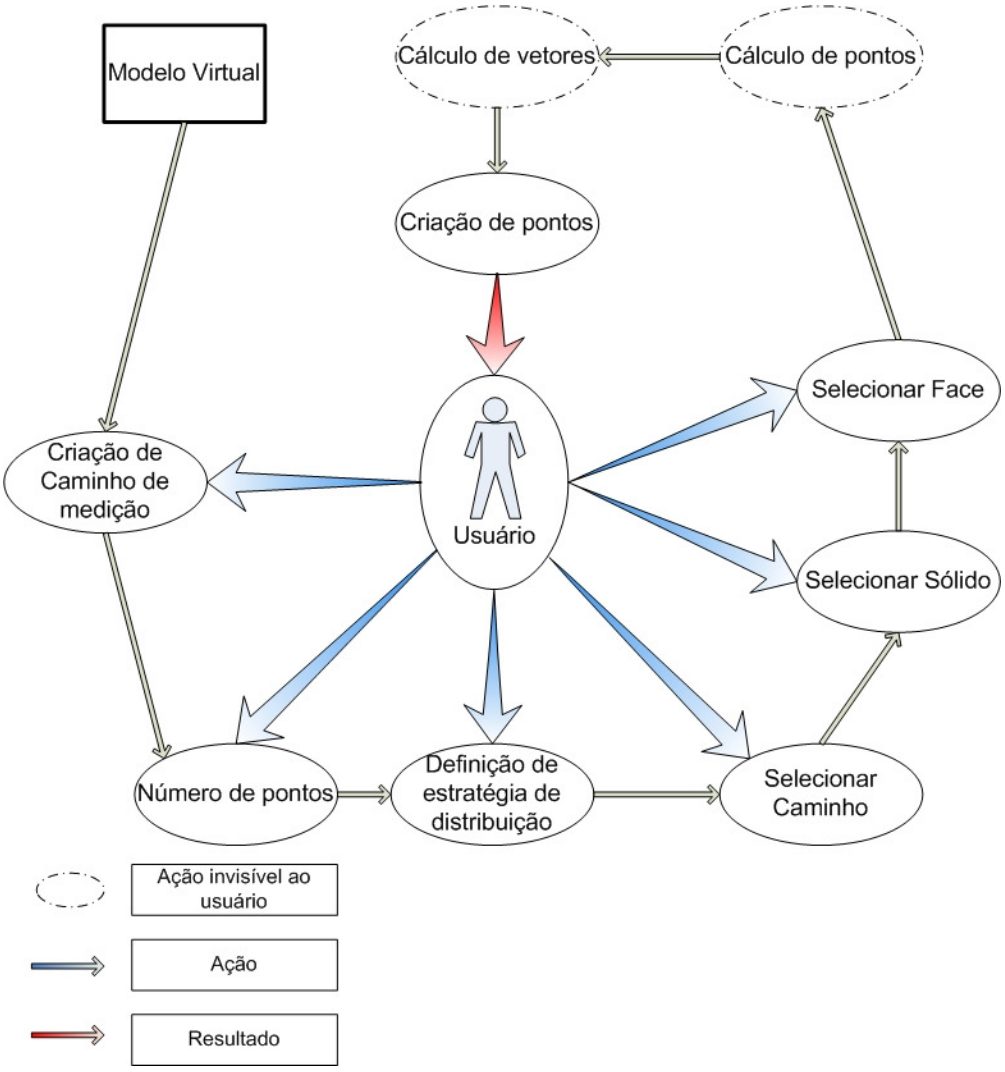


Figura 4.23: Seqüência de uso do aplicativo.

Para a utilização do aplicativo, o usuário deve selecionar na barra de opções principal do software, a opção “medição”, localizada à esquerda da opção “help”.

Após a seleção da opção criada para o aplicativo, uma barra de menu é apresentada com a opção “estratégia de medição”. Com a seleção desta opção tem-se início a janela de trabalho do aplicativo (Figura 4.24).

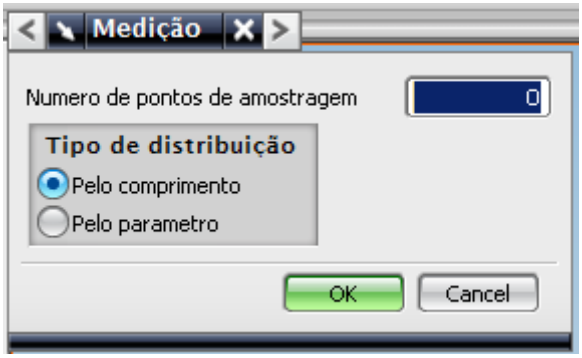


Figura 4.24: Janela de opções do aplicativo.

O primeiro item é o número de pontos que são definidos pelo usuário. Este número deve ter um limite máximo de 1000 pontos (valor estipulado para a concepção do aplicativo, sendo suficiente para a avaliação de criação de pontos e exportação dos mesmos).

O próximo passo é a seleção de qual estratégia se baseia a distribuição de pontos, por meio da distribuição eqüidistante do comprimento de curva ou por distribuição uniforme do parâmetro de construção da curva “t”.

Confirmando o número de pontos e o tipo de distribuição no botão “OK”, uma janela informativa definindo o momento de seleção do caminho de medição (curva) criado pelo usuário (Figura 4.25).

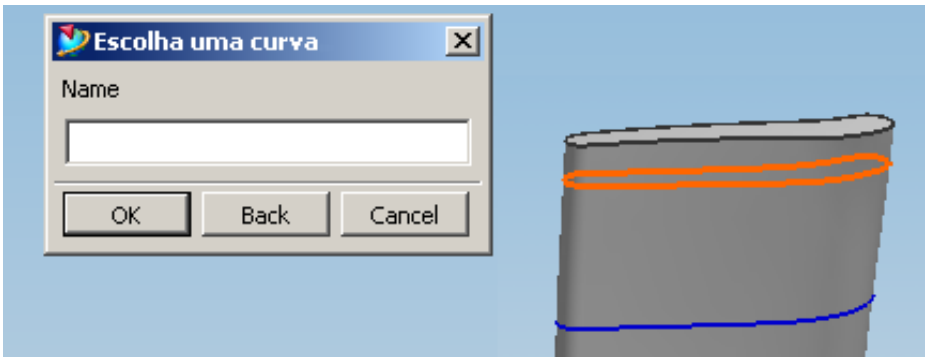


Figura 4.25: Janela de seleção do caminho de medição (curva).

Após a seleção do caminho de medição, uma nova janela informativa aparece, sendo o momento de seleção do sólido do corpo ou superfície do modelo utilizado. A Figura 4.26 ilustra esta janela.

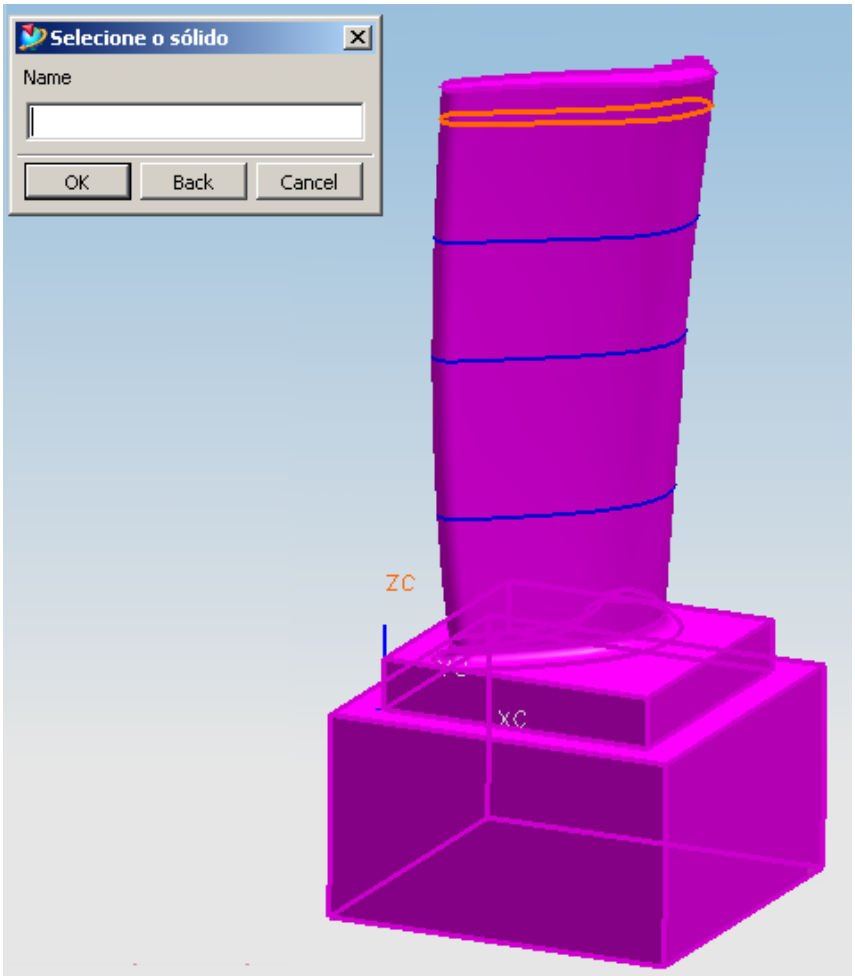


Figura 4.26: Janela de seleção do sólido.

Com a seleção do sólido, o próximo passo é a seleção da face (Figura 4.27). A face selecionada é utilizada para o cálculo do vetor de saída do cabeçote de medição. Estes vetores são normais à face selecionada.

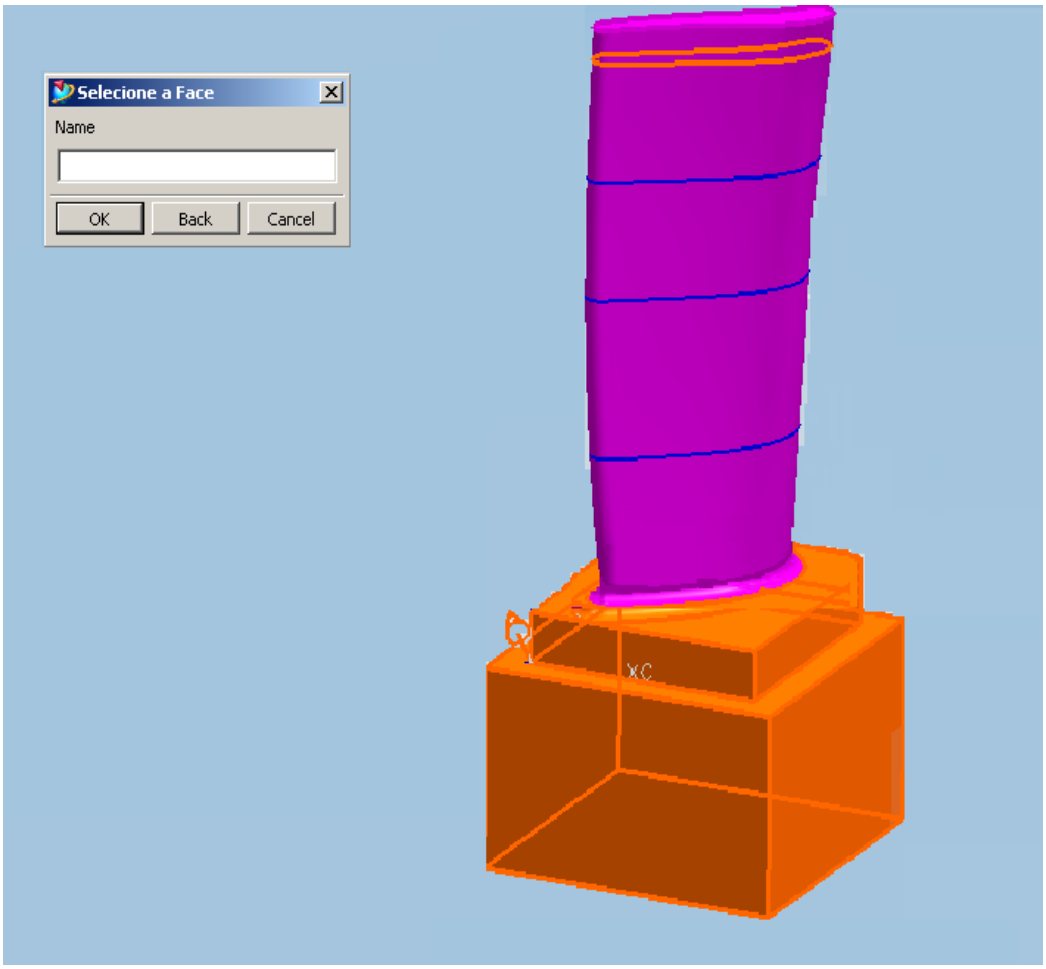


Figura 4.27: Janela de seleção da face.

Por fim, o programa calcula então os pontos e os vetores em relação à superfície selecionada. O resultado final é gerado no modelo virtual, e uma janela informativa confirma a criação de um arquivo “DIMS” com os pontos e seus respectivos vetores gerados (Figura 4.28).

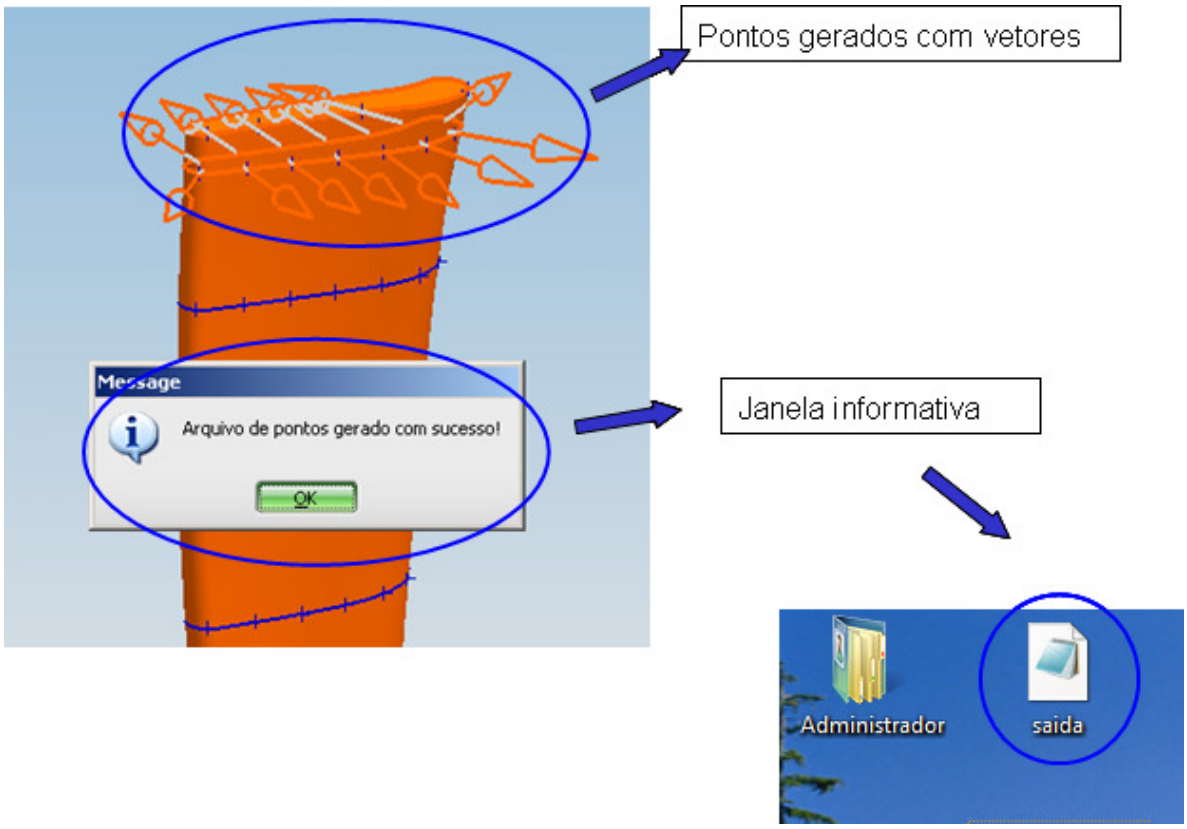


Figura 4.28: Criação de pontos e vetores.

5. Conclusões

Segundo os objetivos propostos neste trabalho, a principal meta era criar um aplicativo para uso em um sistema CAD, que permitisse a distribuição de pontos em um dado plano definido no produto pelo usuário, para posterior exportação em um sistema CAI.

O projeto iniciou-se com o desenvolvimento e implantação de um aplicativo que utiliza os modelos matemáticos NURBS para o cálculo de distribuição de pontos por meio de duas estratégias distintas:

- Uniformemente pelo comprimento da curva e;
- Uniformemente pelo parâmetro “t” da curva.

A utilização deste aplicativo ocorre com o auxílio de textos explicativos para cada etapa da execução de seu funcionamento: quantidade de pontos, seleção da estratégia de distribuição, seleção da curva (plano de medição) e superfície do produto.

Para a exportação dos dados ao sistema CAI, foram gerados duas saídas diferenciadas:

- Pontos com seus vetores normais em relação à face, definidos no próprio modelo virtual do sistema CAD e;
- Dados dos pontos em um arquivo no formato DMIS, para a importação direta à programação de sistemas que suportem este tipo de arquivo.

Com a adição do uso do arquivo DMIS, os pontos puderam então ser importados diretamente para a programação de inspeção, tornando a utilização do aplicativo mais eficiente.

Para futuras melhorias do aplicativo, novas estratégias poderiam ser desenvolvidas de acordo com a necessidade do usuário, e então, introduzidas no sistema CAD.

Outra sugestão é adicionar uma integração que além dos pontos distribuídos, o aplicativo possa simular possíveis colisões do sistema de medição durante o processo de inspeção.

Um estudo do impacto que a inclusão do software pode causar em um processo produtivo poderia ser feito, levando em consideração a diminuição do tempo de construção da estratégia de medição do produto, assim como a exatidão da mesma comparando os métodos de inspeções.

6. Referências

[1] TOLEDO, E. M.; BRITO, E. P. Z. - O Desenvolvimento de Produtos a partir de novas Tecnologias. EnANPAD, p.1-15, 1999.

[2] KIMOTHI, S. K. The Uncertainty of Measuremants. ASQ Quality Press, Milwaukee, 2002.

[3] OLIVEIRA, L. A.; SOUSA, A.R. Validação de processos de medição por coordenadas em operações de controle da qualidade. Anais de Metrologia 2003, Recife, Brasil, 2003.

[4] SOUSA, A. R. Garantia da Confiabilidade Metrológica na Medição por coordenadas, Apostila de. Curso oferecido pela Fundação CERTI, 2003.

[5] SOUSA, A.R. Padrões Corporificados e a Tecnologia de Medição por Coordenadas Inovando a Qualificação Geométrica de Centros de Usinagem. Tese de Doutorado, Universidade Federal de Santa Catarina, 2000.

[6] CAUCHICK, P.A.; KING, T.G. Factors wich influence CMM Touch Trigger Probe Performance. Int. J. Mach. Tools Manufacturing, v. 38, n. 4, p.363-374, 1998.

[7] NEUMANN, H.J. Industrial Coordinate Metrology: Ten years of innovations, Carl Zeiss, Vaerlag Moderne Industrie, 2000.

[8] TIEN, F. -C., TSAI C. -Y. Computer-aided inspection for substrate design. International Journal Advanced Manufacturing Technology, v. 23, p. 279–287, 2004.

[9] GIGO, L.G; SCHNEIDER, C.A. Estação De Medição Por Coordenadas Na Produção De Peças Complexas – Metodologia De Especificação. Sociedade Brasileira de Metrologia (SBM), p.1-8, 2003.

[10] OLIVEIRA, A.L.; SOUSA, A.R.; NETO, A.A.B. Influências da Incerteza da Medição por Coordenadas na Conformidade Dimensional de Peças Seriadas. ENQUALAB, Anais, 2002.

[11] VEIGA, C.L.N.; et al. É difícil calibrar CMMs?. Controle da Qualidade, Ed. Banas, v. 50, p. 48-54,1997.

[12] BOUTELLIER, R., GASSMANN, O. AND VON ZEDTWITZ, M.; Managing Global Innovation, Uncovering the Secrets of Future Competitiveness. 3. ed. Berlin, Germany: Springer-Verlag, 2008.

[13] URRUTIA,J.I.D. Avaliação dos processos de medição na indústria, baseada no impacto econômico da operação de controle geométrico. Dissertação de

mestrado, curso de Pós-Graduação em Metrologia Científica e Industrial, Universidade Federal de Santa Catarina, 2000.

[14] GIGO,L.G. Estação de Medição por Coordenadas na Produção de Peças complexas: Método de Especificação. Dissertação de mestrado, curso de Pós-Graduação em Metrologia Científica e Industrial, Universidade Federal de Santa Catarina, 1999.

[15] SOUSA, A.R. de; SCHNEIDER, C.A.; MAAS, G.A. Recomendações para uma utilização eficiente confiável da medição por coordenadas. Certi, p. 1-8, 2003.

[16] SOUSA, A.R. A medição por coordenadas na garantia da qualidade da produção do 3º milênio. Congresso Brasileiro de Metrologia, 2003.

[17] TANI,F. Curva Ascendente nas Vendas de Máquinas de Medição Tridimensional. Revista Metal Mecânica, p. 8-19, 2000.

[18] ROLIM, T.L.; et al. Comparação Entre Modos De Calibração De Máquinas De Medição Por Coordenadas. Sociedade Brasileira de Metrologia, Recife p. 1-7, 2003.

[19] BOSCH, J.A. Coordinate Measuring Machines and Systems, Book, CRC, New York, p. 443, 1995.

[20] TUPY S.A. Joinville, Brasil, Disponível em www.tupy.com.br, visualização em 22 de Junho de 2008.

[21] WERNER, A.; et al. Reverse engineering of free-form surfaces. Journal Of Materials Processing Technology, Bialystok, v. 76, p. 128-132, 1998.

[22] FENG, C.J.; et al. Design and analysis of experiments in CMM measurement uncertainty study. Precision Engineering, v. 31, p. 94-101, 2007.

[23] PEREIRA, P.H.; HOCKEN, R.J. Characterization and compensation of dynamic errors of a scanning coordinate measuring machine. Precision Engineering, v. 31, p. 22-32, 2007.

[24] OPPERMANN M.; et al. New Quality Cost Models to Optimize Inspection Strategies. Ieee Transactions On Electronics Packaging Manufacturing. V. 26, N. 4, 2003.

[25] KRUTH, J., VAN DEN BERGH, C.; VANHERCK, P. Correcting steady-state temperature influences on coordinate measuring machines. Journal of Manufacturing Systems, v. 19, n. 6, p. 365-374, 2000.

[26] YANG, J; HAN, S; PARK, S. A method for verification of computer-aided design model errors. Journal Of Engineering Design, v. 16, n.3, p. 337-352, 2005.

[27] PORATH, M. C.; SOUSA, A.R. de. A Tecnologia De Medição Por Coordenadas No Ciclo de Desenvolvimento De Produtos Plásticos. Certi, p. 1-8, 2002.

[28] GLEICHER, M. A Curve Tutorial for Introductory Computer Graphics. Department Of Computer Sciences, p.1-25, 2004.

[29] MAHON, MC.; BROWNE, J. CAD/CAM from Principles to Practice. UK, Suffolk: Addison-Esley, ISBN 0-201-56502-1, 1993.

[30] CHANDLER, R.E. A recursive technique for rendering parametric curves. Computers and Graphics, v. 14, p. 477–479, 1990.

[31] CHU, C.; WANG, C.C.l.; TSAI, C. Computer aided geometric design of strip using developable Bezier patches. Computers In Industry, v. 59, n. 6, p.601-611, 2008.

[32] PARENTE JUNIOR, E. Análise de Sensibilidade e Otimização de Estruturas Geometricamente Não-Lineares. Tese de Doutorado, Departamento de Engenharia Civil, PUC - Rio, p. 1-25, 2000.

[33] ROGERS, D. F. An introduction to NURBS: with historical perspective. Ed. Morgan Kaufmann, 2001.

[34] LAI, Y.; et al. Degree reduction of NURBS curves. International Journal Advanced Manufacturing Technology, v. 27, p. 1124-1131, 2006.

[35] HUGHES, T.j.r.; COTTRELL, J.A.; BAZILEVS, Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods Application in Mechanic. Engineering, Austin, v. 194, p. 4135-4195, 2005.

[36] HELLENO,A.L. Contribuição para a manufatura de superfícies complexas com altas velocidades baseada em novos métodos de interpolação da trajetória da ferramenta. Tese de Doutorado, Universidade Metodista de Piracicaba, 2008.

[37] STODDART, A. J.; HILTON, A.; ILLINGWORTH, J. Slime: A new deformable surface. British Machine Vision Conference, p. 285-294, 1994.

[38] LEI, W.T.; et. Al. Fast real-time NURBS path interpolation for CNC Machine tools. International Journal of Machine Tools & Manufacturing, v.47, p. 1530-1541, 2007.

[39] BOUJELBENE, M.; et al. Product enhancement in dies and molds manufacturing by the use of C1 continuous tool path. International Journal of Machine tools & manufacture, v.44,p. 101-107, 2004.

[40] SANSONI, Gi.; DOCCHIO, F. Three-dimensional optical measurements and reverse engineering for automotive applications. Robotics And Computer-integrated Manufacturing, v. 20, p. 359-367, 2004.

[41] HOFFMANN, C. M., KIMB, K. -J. Towards valid parametric CAD models. Computer-Aided Design v. 33, n. 1, p. 81-90, 2001.

[42] WOO, Y., LEE, S. H. Volumetric modification of solid CAD models independent of design features. Advances in Engineering Software, v.37, p. 826–835, 2006.

[43] BARBIERI, L. et al.;Innovative integration techniques between Virtual Reality. International Journal of Advanced Manufacturing Technology, v. 38, p. 1085–1097, 2008.

[44] HUI-XIA, L.; et al.; Development of a Knowledge-Based Intelligent cad system for automotive panel die. Journal of advanced Manufacturing Systems, vol.7, Nº1 p.51-54, 2008.

[45] PASIN, A.; Análise crítica de software para controle dimensional aplicado a máquinas de medir por coordenadas com relação ao uso da linguagem GD&T segundo a norma Y 14. 5M-1994. Monografia de final de curso, UNESP Guaratinguetá-SP, 2003.

[46] CUNHA, R.R.M.; DIAS, A. Uma revisão das tecnologias de integração de dados em CAD/CAM. CONEM 2000.

[47] NATEKAR, D.; ZHANG, X.; SUBBARAYAN, G. Constructive solid analysis: a hierarchical, geometry-based meshless analysis procedure for integrated design and analysis. Computer-aided Design, v. 36, p.473-486, 2004.

[48] SALAMON, C.; CORNEY, J.; RITCHIE, J. Information hiding through variance of the parametric orientation underlying a B-rep face. 10th International Workshop on Information Hiding, p. 268-282, 2008.

[49] SUNIL, V.B.; PANDE, S.S. Automatic recognition of features from freeform surface CAD models. CAD Computer Aided Design, v., 40, n. 4, p. 502-517, 2008.

[50] HELLENO, A.L. Investigação de Métodos de Interpolação para Trajetória da Ferramenta na Usinagem de Moldes e Matrizes com Alta Velocidade, 2004, 140 p. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, Universidade Metodista de Piracicaba, 2004.

[51] REQUICHA, A. A. G. GEOMETRIC MODELING: A First Course. Acesso em Janeiro de 2009. http://www-lmr.usc.edu/~requicha/ch3.pdf.

[52] HOGGE, D. Integrating Commercial CAx Software to Perform Multidisciplinary Design Optimization. Thesis, Brigham Young University, 2002

[53] DING, S.; et al. The implementation of adaptive isoplanar tool path generation for the machining of free-form surfaces. International Journal Advanced Manufacturing Technology, v. 26, p. 852–860, 2005.

[54] LU, Z.; CHEN, D.; CHENG, Y. Effective Distributed Collaborative Design FrameBased on UG Software. 2nd International Conference on Pervasive Computing and Applications, p. 158-161, 2007.

[55] ROZVANY, G.I.N.; ZHOU, M.; BIRKER, T. Generalized shape optimization without homogenization. Structural and Multidisciplinary Optimization. V. 4, p. 250–252, 1992.

[56] SCHÜTZER, K.; HELLENO, A. L.; PEREIRA, S. C. The influence of the manufacturing strategy on the production of molds and dies. Journal of Materials Processing Technology, v. 179, p. 172-177, 2006.

[57] LE, J. J., CHOU, C. C.; CHEN, P. Study of CAE crash signatures for airbag sensor calibration. International Journal of Vehicle Safety, v. 2, n. 1-2, p. 20-43, 2007.

[58] ZHANG, S.G.; et al. A feature-based inspection process planning system for co-ordinate measuring machine (CMM). Journal Of Materials Processing Technology, v. 107, p. 111-118, 2000.

[59] CHEN J. Computer-aided accuracy enhancement for multi-axis CNC machine tool International Journal of Machine Tools and Manufacture v. 35, n. 4, p. 593-605, 1995.

[60] BARREIRO, J.; et al. Functional model for the development of an inspection integration framework. International Journal Of Machine Tools & Manufacture, v. 43, p. 1621-1632, 2003.

[61] DONATELLI, G. D.; et al. Metrologia Geométrica na Indústria: Tendências e Desafios. SENAI/SC, 2005.

[62] WOODBINE, K. It's not just CMM software. Quality, v. 46, n. 1, p. 52-59, 2007.

[63] ISO 1101. Geometrical Product Specifications (GPS) - Geometrical tolerancing - Tolerances of form, orientation, location and run-out, 2004.

[64] GASS, S., WITZGALL, C.; HARARY, H. H. Fitting circles and spheres to coordinate measuring machine data. International Journal of Flexible Manufacturing Systems, v. 10, n. 1, p 5-25, 1998.

[65] PAHK, H. J. ;et al. Development of computer-aided inspection system with CMM for integrated mold manufacturing, CIRP annals, v. 42, n.1, p. 557-560, ISSN 0007-8506, 1993.

[66] FENG, C., WANG, X. Digitizing uncertainty modeling for reverse engineering applications: regression versus neural networks. Journal of Intelligent Manufacturing. V. 13, p. 189-199, 2002.

[67] CHO, M.-w.; SEO, T.-i.. Inspection Planning Strategy for the On-Machine Measurement Process Based on CAD/CAM/CAI Integration. International Journal Advanced Manufacturing Technology, v. 19, p. 607-617, 2002.

[68] PAN, C.; SMITH, S. S.; SMITH, G. C. Automatic assembly sequence planning from STEP CAD files. International Journal Of Computer Integrated Manufacturing, v. 19, n. 8, p. 775-783, 2006.

[69] REHWALD, P. Vdafs - An Interface To Transfer Surface Description Data Between CAD Systems. Computers & Graphics, v. 9, n.1, p. 69-70, 1985.

[70] SMID, M. E., BRANSTAD, D. K. The Data Encryption Standard: Past and Future. Proceedings of the IEEE, v. 76, n. 5, p. 550-560, 1988.

[71] "Making the CAD to CMM interface more effective". Modern Machine Shop. FindArticles.com. 01 Dec. 2008. http://findarticles.com/p/articles/mi_m3101/is_n9_v61/ai_7370245.

[72] DÜRR, H.; SCHÜNEMANN, R.; SCHULZE, J. CADEIA de processo baseada em NURBS. Máquinas e Metais, São Paulo, v. 33, n. 415, p. 18-27, ago. 2000.

[73] NASR, E. S. A.; KAMRANI, A. K. A new methodology for extracting manufacturing features from CAD system. Computers & Industrial Engineering, v. 51, p. 389-415, 2006.

[74] SHE, C.; et al. A study on the computer-aided measuring integration system for the sheet metal stamping die. Journal Of Materials Processing Technology, p. 138-141, 2006.

[75] PC-DMIS™ 3.5 Portuguese Version Reference Manual, 2003.

[76] PAHK, H. J.; AHN, W. J. Precision Alignment Technique For Parts Having Thin Features Using Measurement Feedback Iterative Method In Cad/Cai Environment. International Journal of Machine Tools & Manufacture, v. 36, n. 2, p. 217-227, 1996.

[77] SKALSKI, K.; et al. Identification and geometrical modelling of complex shape surfaces using coordinate measuring machine and CAD:CAM systems. Journal of Materials Processing Technology, Warsaw, v. 76, p. 49-55, 1998.

[78] ELKOTT, D.F.; ELMARAGHY, H.A.; ELMARAGHY, W. H. Automatic sampling for CMM inspection planning of free-form surfaces. International Journal of Production Research, v. 40, n. 11, p. 2653-2676, 2002.

Anexos

Anexo A – Código Fonte do aplicativo desenvolvido.

```
/* These include files are needed for the following template code.
*/
#include <stdio.h>
#include <uf.h>
#include <uf_defs.h>
#include <uf_exit.h>
#include <uf_ui.h>
#include <uf_styler.h>
#include <uf_mb.h>
#include "CDF.h"
#include <windows.h>
#include <malloc.h>
#include <uf_disp.h>
#include <uf_obj.h>
#include <uf_view.h>
#include <uf_curve.h>
#include <uf_eval.h>
#include <iostream.h>
#include <uf_modl.h>
#include <uf_assem.h>

/* The following definition defines the number of callback entries */
/* in the callback structure:                                      */
/* UF_STYLER_callback_info_t CDF_cbs */
#define CDF_CB_COUNT ( 4 + 1 ) /* Add 1 for the terminator */
#define POR_COMPRIMENTO 0
#define POR_PARAMETRO 1
#define DENTRO_SOLIDO 1
#define FORA_SOLIDO 2
#define NO_SOLIDO 3
 /*-------------------------------------------------------------------
--
The following structure defines the callback entries used by the
styler file.  This structure MUST be passed into the user function,
UF_STYLER_create_dialog along with CDF_CB_COUNT.
-------------------------------------------------------------------
*/
static UF_STYLER_callback_info_t CDF_cbs[CDF_CB_COUNT] =
{
 {UF_STYLER_DIALOG_INDEX, UF_STYLER_OK_CB, 1, CDF_OK_CB},
 {UF_STYLER_DIALOG_INDEX, UF_STYLER_CANCEL_CB, 0, CDF_CANCEL_CB},
 {CDF_N_POINTS, UF_STYLER_ACTIVATE_CB, 0, CDF_PONTOS_ACTIVE_CB},
 {CDF_DISTRIBUICAO, UF_STYLER_VALUE_CHANGED_CB, 0, CDF_TIPO_CB},
 {UF_STYLER_NULL_OBJECT, UF_STYLER_NO_CB, 0, 0}
};
/*-------------------------------------------------------------------
--
UF_MB_styler_actions_t contains 4 fields.  These are defined as follows:

Field 1: the name of your dialog that you wish to display.
Field 2: any client data you wish to pass to your callbacks.
Field 3: your callback structure.
```

```
Field 4 : flag to inform menubar of your dialog location.  This flag
MUST
match the resource set in your dialog! Do NOT ASSUME that changing
this field will update the location of your dialog.  Please use the
UIStyler to indicate the position of your dialog.
------------------------------------------------------------------------
*/
static UF_MB_styler_actions_t actions[] = {
    { "CDF.dlg",  NULL,   CDF_cbs,  UF_MB_STYLER_IS_NOT_TOP },
    { NULL, NULL, NULL, 0} /* This is a NULL terminated list */};

extern void ufsta (char *param, int *retcode, int rlen)
{
    int  error_code;
   FILE *fp; //prepare a file to allocate the console

    if ( (UF_initialize()) != 0)
          return;

    if ( (error_code = UF_MB_add_styler_actions ( actions ) ) != 0 )
    {
          char fail_message[133];

          UF_get_fail_message(error_code, fail_message);
          printf ( "%s\n", fail_message );
    }

    UF_terminate();
    return;
}

//************************************************************************
//
/*------------------------------------------------------------------------
*/
/*------------------- UIStyler Callback Functions ----------------------
*/
/*------------------------------------------------------------------------
*/
/* ------------------------------------------------------------------------
-
 * Callback Name: CDF_OK_CB
 * This is a callback function associated with an action taken from a
 * UIStyler object.
 *
 * Input: dialog_id   -     The dialog id indicate which dialog this
callback
 *                             is associated with.  The dialog id is a
dynamic,
 *                         unique id and should not be stored.  It is
 *                         strictly for the use in the NX Open API:
 *                                UF_STYLER_ask_value(s)
 *                                UF_STYLER_set_value
 *        client_data -    Client data is user defined data associated
 *                         with your dialog.  Client data may be bound
 *                         to your dialog with UF_MB_add_styler_actions
 *                         or UF_STYLER_create_dialog.
```

```
 *        callback_data - This structure pointer contains information
 *                        specific to the UIStyler Object type that
 *                        invoked this callback and the callback type.
 * ---------------------------------------------------------------------
*/



//************************************************************************
//
/*-----------------------------------------------------------------------
*/
/*---------------------- Função Principal ----------------------*/
/*-----------------------------------------------------------------------
*/
/* ---------------------------------------------------------------------
--
*----Desenvolvedor - Henrique Neves de Lucena
*----10/03/2009---------------------------
*----Função principal do Aplicativo
*----Desenvolvido no Laboratório SCPM - Universidade Metodista de
Piracicaba
//************************************************************************
//

int CDF_OK_CB ( int dialog_id,
             void * client_data,
             UF_STYLER_item_value_type_p_t callback_data)
{

////////////////////////////////////////////////////////////////////////
//
   ///------------------Variáveis------------------///
////////////////////////////////////////////////////////////////////////
//

    //Variáveis para a escolha da curva através da caixa de diálogo
   int retornoCaixa;
   tag_t curva_bs;
   double cursor[3];
   tag_t vista;

   //Variáveis para os dados da curva escolhida
   double *dadosCurva;
   double comprimento;
   int tipoCurva;
   int ultimoValorDeNo;
   UF_CURVE_struct_p_t estruturaCurva;

   //Variáveis para a leitura da caixa de diálogo
   int tipoErro=0;
   char mensagem[133];
   int contador;
   int numeroPontos;
   UF_STYLER_item_value_type_t dadosCaixa[2];
```

```
   int quantidadeValores=0;
   int modoCalculo;

   //Variáveis para a determinação do ponto a ser lido na curva
   double estimativa;
   double estimativaInferior;
   double estimativaSuperior;
   double parametro=0;
   double passo;
   double ponto[3];
   double tamanho;
   tag_t pontoCriado;

   //variaveis para vetores
   tag_t solidoEscolhido;
   double pontoVerificacao[3];

   //Variaveis para a face
   tag_t superficieEscolhida;
   double parametrosUV[2];
   double pontoNaFace[3];
   double u1[3];        //primeira derivada em U
   double v1[3];        //primeira derivada em V
   double u2[3];        //segunda derivada em U
   double v2[3];        //segunda derivada em V
   double normalFace[3];//Vetor normal à face no ponto
   double radii[2];     //Raio de curvatura


   const char  *filenameOUT;  //arquivo de escrita
   FILE *arquivoSaida;
   char saida[256];


   int pontoStatus;
   int retval;


     if ( UF_initialize() != 0)
          return ( UF_UI_CB_CONTINUE_DIALOG );

    //Captura os pontos a serem examinados
    filenameOUT="C:\\Documents                                          and
Settings\\lucena_h\\Desktop\\saida.DMS";
    arquivoSaida=fopen(filenameOUT,"w"); //abre o arquivo de escrita
    dadosCaixa[0].item_id = CDF_DISTRIBUICAO;
    dadosCaixa[0].item_attr = UF_STYLER_VALUE;
    dadosCaixa[1].item_id = CDF_N_POINTS;
    dadosCaixa[1].item_attr = UF_STYLER_VALUE;
     tipoErro=
     UF_STYLER_ask_values(dialog_id,2,dadosCaixa,&quantidadeValores);
    if ((tipoErro <0)||(tipoErro >1))
    {
      UF_get_fail_message(tipoErro,mensagem);
      UF_UI_set_status (mensagem);
    }
    else
```

```
    {
     modoCalculo=dadosCaixa[0].value.integer;
     numeroPontos=dadosCaixa[1].value.integer;
     for (contador=0;contador<quantidadeValores;contador++)
     UF_STYLER_free_value(&dadosCaixa[contador]);
    }



////////////////////////////////////////////////////////////////////////
//
   ///------------------Seleção dos objetos------------------///
////////////////////////////////////////////////////////////////////////
//

   //Escolha da curva - - -
   UF_UI_select_with_single_dialog("Escolha uma b-spline para ser
calculada.","Escolha uma curva",UF_UI_SEL_SCOPE_NO_CHANGE, selec_curva,
NULL,&retornoCaixa, &curva_bs, cursor, &vista);
   //Escolha da face - - -
   UF_UI_select_with_single_dialog("Escolha a Face para o cálculo do
vetor normal","Selecione a Face",UF_UI_SEL_SCOPE_NO_CHANGE, selec_face,
NULL,&retornoCaixa, &superficieEscolhida, cursor, &vista);
   //Escolha do sólido - - -
   UF_UI_select_with_single_dialog("Escolha o corpo do sólido
utilizado","Selecione o sólido",UF_UI_SEL_SCOPE_NO_CHANGE, selec_solido,
NULL,&retornoCaixa, &solidoEscolhido, cursor, &vista);
   //Cálculo do comprimento da curva - - -
   UF_CURVE_ask_arc_length (curva_bs, 0.0 , 1.0,UF_MODL_UNITS_PART,
&comprimento);
   //Leitura dos dados da curva (em especial do vetor de nós) - - -
   UF_CURVE_ask_curve_struct(curva_bs,&estruturaCurva);
   UF_CURVE_ask_curve_struct_data(estruturaCurva,&tipoCurva,&dadosCurva)
;
   ultimoValorDeNo =
(int)dadosCurva[(int)dadosCurva[3]+(int)dadosCurva[4]+4];
   //Liberação de memória após a leitura dos dados da curva
   UF_CURVE_free_curve_struct(estruturaCurva);
   UF_free(dadosCurva);
////////////////////////////////////////////////////////////////////////
//
//----Cálculo por distribuição uniforme pelo comprimento da curva-------
//
////////////////////////////////////////////////////////////////////////
//

   if (modoCalculo==POR_COMPRIMENTO) //Calculo dos pontos por
comprimento da curva
   {
    passo=comprimento/(numeroPontos-1);
    tamanho=0.0;
    for (contador = 0; contador< numeroPontos; contador++)
    {
     estimativaInferior = 0.0;
     estimativaSuperior= ultimoValorDeNo;
     estimativa =estimativaSuperior;
   //Ponto Inicial
```

```
      if (contador==0)
        estimativa=0.0;
   //Ponto Final
      if (contador==numeroPontos-1)
        estimativa=ultimoValorDeNo;
      while ((estimativaSuperior-estimativaInferior)>5E-12)
      {
        //Cálculo do comprimento do segmento da curva até a estimativa
        UF_CURVE_ask_arc_length (curva_bs, 0.0 ,
estimativa,UF_MODL_UNITS_PART, &comprimento);
        if (comprimento>tamanho)
          estimativaSuperior=estimativa;
        else
          estimativaInferior=estimativa;
        estimativa=((estimativaSuperior-
estimativaInferior)/2)+estimativaInferior;
      }

      curva(curva_bs,estimativa,ponto);


   ////////////////////////////////////////////////////////////////////
///
   ///--------------------Criação dos pontos--------------------------
///

////////////////////////////////////////////////////////////////////

      UF_CURVE_create_point(ponto,&pontoCriado);
      //Leitura dos parâmetros da curva (em especial dos parâmetro u e
v)
      UF_MODL_ask_face_parm(superficieEscolhida,ponto_p, parametrosUV,
pontoNaFace);
      //De posse dos parâmetros u e v, leitura do vetor normal ao ponto
escolhido
      UF_MODL_ask_face_props(superficieEscolhida, parametrosUV,
pontoNaFace, u1, v1, u2, v2, normalFace, radii);
      pontoVerificacao[0]=ponto[0]+normalFace[0];
      pontoVerificacao[1]=ponto[1]+normalFace[1];
      pontoVerificacao[2]=ponto[2]+normalFace[2];
      //Verifica se o vetor normal à face está contido no sólido
      UF_MODL_ask_point_containment(pontoVerificacao,solidoEscolhido,
&pontoStatus );
      if (pontoStatus!=FORA_SOLIDO){ //se o vetor normal estiver contido
no sólido é invertido
        normalFace[0]=-normalFace[0];
        normalFace[1]=-normalFace[1];
        normalFace[2]=-normalFace[2];
      }
      //Apresenta o vetor normal na tela
      UF_DISP_conehead(UF_DISP_ALL_ACTIVE_VIEWS,pontoVerificacao,
normalFace, 0);
      //Escreve no arquivo de saída

   sprintf(saida,"F(PM%03d)=FEAT/POINT,CART,%f,%f,%f,%f,%f,%f\n",contado
r+1,ponto[0],
ponto[1],ponto[2],normalFace[0],normalFace[1],normalFace[2]);
```

```
      fputs(saida,arquivoSaida);
      sprintf(saida,"MEAS/POINT,F(PM%03d),1\n", contador+1);
      fputs(saida,arquivoSaida);
      sprintf(saida,"  PTMEAS/CART,%f,%f,%f,%f,%f,%f\n",ponto[0],
ponto[1],ponto[2],normalFace[0],normalFace[1],normalFace[2]);
      fputs(saida,arquivoSaida);
      fputs("ENDMES\n\n",arquivoSaida);
      tamanho=tamanho+passo;
     }
   }

////////////////////////////////////////////////////////////////////////
//
//-------Cálculo por distribuição uniforme pelo parâmetro da curva------
//
////////////////////////////////////////////////////////////////////////
//

   if (modoCalculo==POR_PARAMETRO) //Calculo dos pontos por parametro da
curva
   {
    passo = (double)ultimoValorDeNo/(numeroPontos-1);
    parametro=0.0;
    for (contador=0; contador<numeroPontos;m++)
   {
      curva(curva_bs,parametro,ponto);

////////////////////////////////////////////////////////////////////////
//
 ///--------------------Criação  dos  pontos--------------------------
///
////////////////////////////////////////////////////////////////////////
//

      UF_CURVE_create_point(ponto,&pontoCriado);
      //Leitura dos parâmetros da curva (em especial dos parâmetro u e
v)
      UF_MODL_ask_face_parm(superficieEscolhida,ponto_p, parametrosUV,
pontoNaFace);
      //De posse dos parâmetros u e v, leitura do vetor normal ao ponto
escolhido
      UF_MODL_ask_face_props(superficieEscolhida, parametrosUV,
pontoNaFace, u1, v1, u2, v2, normalFace, radii);
      pontoVerificacao[0]=ponto[0]+normalFace[0];
      pontoVerificacao[1]=ponto[1]+normalFace[1];
      pontoVerificacao[2]=ponto[2]+normalFace[2];
      //Verifica se o vetor normal à face está contido no sólido
      UF_MODL_ask_point_containment(pontoVerificacao,solidoEscolhido,
&pontoStatus );
      if (pontoStatus!=FORA_SOLIDO){ //se o vetor normal estiver contido
no sólido é invertido
        normalFace[0]=-normalFace[0];
        normalFace[1]=-normalFace[1];
        normalFace[2]=-normalFace[2];
      }
      //Apresenta o vetor normal na tela
      UF_DISP_conehead(UF_DISP_ALL_ACTIVE_VIEWS,pontoVerificacao,
```

```
normalFace, 0);
      //Escreve no arquivo de saída

   sprintf(saida,"F(PM%03d)=FEAT/POINT,CART,%f,%f,%f,%f,%f,%f\n",contado
r+1,ponto[0],
ponto[1],ponto[2],normalFace[0],normalFace[1],normalFace[2]);
      fputs(saida,arquivoSaida);
      sprintf(saida,"MEAS/POINT,F(PM%03d),1\n", contador+1);
      fputs(saida,arquivoSaida);
      sprintf(saida,"  PTMEAS/CART,%f,%f,%f,%f,%f,%f\n",ponto[0],
ponto[1],ponto[2],normalFace[0],normalFace[1],normalFace[2]);
      fputs(saida,arquivoSaida);
      fputs("ENDMES\n\n",arquivoSaida);
      parametro=parametro+passo;

    }
   }
   fclose(arquivoSaida);       //fecha o arquivo de saída
   uc1601("Arquivo de pontos gerado com sucesso!",1);
    UF_terminate ();

    return (UF_UI_CB_EXIT_DIALOG);

}

/* ------------------------------------------------------------------------
--
 * Callback Name: CDF_CANCEL_CB
 * This is a callback function associated with an action taken from a
 * UIStyler object.
 *
 * Input: dialog_id    -   The dialog id indicate which dialog this
callback
 *                             is associated with.  The dialog id is a
dynamic,
 *                         unique id and should not be stored.  It is
 *                         strictly for the use in the NX Open API:
 *                                UF_STYLER_ask_value(s)
 *                                UF_STYLER_set_value
 *        client_data -   Client data is user defined data associated
 *                         with your dialog.  Client data may be bound
 *                         to your dialog with UF_MB_add_styler_actions
 *                         or UF_STYLER_create_dialog.
 *        callback_data - This structure pointer contains information
 *                         specific to the UIStyler Object type that
 *                         invoked this callback and the callback type.
 * ------------------------------------------------------------------------
*/
int CDF_CANCEL_CB ( int dialog_id,
             void * client_data,
             UF_STYLER_item_value_type_p_t callback_data)
{
     if ( UF_initialize() != 0)
          return ( UF_UI_CB_CONTINUE_DIALOG );


     UF_terminate ();
```

```
    return ( UF_UI_CB_EXIT_DIALOG );
}



    /* Callback acknowledged, terminate dialog             */
    /* It is STRONGLY recommended that you exit your       */
    /* callback with UF_UI_CB_EXIT_DIALOG in a cancel call */
    /* back rather than UF_UI_CB_CONTINUE_DIALOG.          */




/* ------------------------------------------------------------------
-
 * Callback Name: CDF_PONTOS_ACTIVE_CB
 * This is a callback function associated with an action taken from a
 * UIStyler object.
 *
 *  Input: dialog_id     -   The dialog id indicate which dialog this
callback
 *                             is associated with.  The dialog id is a
dynamic,
 *                         unique id and should not be stored.  It is
 *                         strictly for the use in the NX Open API:
 *                                UF_STYLER_ask_value(s)
 *                                UF_STYLER_set_value
 *        client_data -    Client data is user defined data associated
 *                         with your dialog.  Client data may be bound
 *                         to your dialog with UF_MB_add_styler_actions
 *                         or UF_STYLER_create_dialog.
 *        callback_data - This structure pointer contains information
 *                         specific to the UIStyler Object type that
 *                         invoked this callback and the callback type.
 * ------------------------------------------------------------------
*/
int CDF_PONTOS_ACTIVE_CB ( int dialog_id,
             void * client_data,
             UF_STYLER_item_value_type_p_t callback_data)
{
     /* Make sure User Function is available. */
     if ( UF_initialize() != 0)
          return ( UF_UI_CB_CONTINUE_DIALOG );

     /* ---- Enter your callback code here ----- */

     UF_terminate ();

    /* Callback acknowledged, do not terminate dialog */
    return (UF_UI_CB_CONTINUE_DIALOG);

    /* or Callback acknowledged, terminate dialog.    */
    /* return ( UF_UI_CB_EXIT_DIALOG );               */

}
```

```
/* ----------------------------------------------------------------
-
 * Callback Name: CDF_TIPO_CB
 * This is a callback function associated with an action taken from a
 * UIStyler object.
 *
 * Input: dialog_id     -   The dialog id indicate which dialog this
callback
 *                              is associated with.  The dialog id is a
dynamic,
 *                          unique id and should not be stored.  It is
 *                          strictly for the use in the NX Open API:
 *                                 UF_STYLER_ask_value(s)
 *                                 UF_STYLER_set_value
 *        client_data -   Client data is user defined data associated
 *                          with your dialog.  Client data may be bound
 *                          to your dialog with UF_MB_add_styler_actions
 *                          or UF_STYLER_create_dialog.
 *        callback_data - This structure pointer contains information
 *                          specific to the UIStyler Object type that
 *                          invoked this callback and the callback type.
 * ----------------------------------------------------------------
*/
int CDF_TIPO_CB ( int dialog_id,
             void * client_data,
             UF_STYLER_item_value_type_p_t callback_data)
{
     /* Make sure User Function is available. */
     if ( UF_initialize() != 0)
          return ( UF_UI_CB_CONTINUE_DIALOG );

     /* ---- Enter your callback code here ----- */

     UF_terminate ();

    /* Callback acknowledged, do not terminate dialog */
    return (UF_UI_CB_CONTINUE_DIALOG);

    /* or Callback acknowledged, terminate dialog.  */
    /* return ( UF_UI_CB_EXIT_DIALOG );             */

}

//**************************************************************************
//
/*-------------------------------------------------------------------------
*/
/*---------------------- Função Seleção de Objetos ---------------------
*/
/*-------------------------------------------------------------------------
*/
/* ----------------------------------------------------------------
-
*----Desenvolvedor - Henrique Neves de Lucena
*----10/03/2009---------------------------
*----Função caracterizada para a seleção da curva, a face de refrência e
```

```
o sólido utilizado
*----Desenvolvido no Laboratório SCPM - Universidade Metodista de
Piracicaba
//*********************************************************************
//

static int selec_curva(UF_UI_selection_p_t select,void* user_data)
{
    int num_triples = 2;
    UF_UI_mask_t mask_triples[] = {
     UF_spline_type,    0, 0,
         UF_solid_type,       0, UF_UI_SEL_FEATURE_ANY_EDGE};


if(UF_UI_set_sel_mask(select,UF_UI_SEL_MASK_CLEAR_AND_ENABLE_SPECIFIC,nu
m_triples, mask_triples) == 0)
        return (UF_UI_SEL_SUCCESS);
    else
        return (UF_UI_SEL_FAILURE);
}

static int selec_solido(UF_UI_selection_p_t select,void* user_data)
{
    int num_triples = 1;
    UF_UI_mask_t mask_triples[] = {
     UF_solid_type,           0, 0
         };


if(UF_UI_set_sel_mask(select,UF_UI_SEL_MASK_CLEAR_AND_ENABLE_SPECIFIC,nu
m_triples, mask_triples) == 0)
        return (UF_UI_SEL_SUCCESS);
    else
        return (UF_UI_SEL_FAILURE);
}

static int selec_face(UF_UI_selection_p_t select,void* user_data)
{
    int num_triples = 1;
    UF_UI_mask_t mask_triples[] = {
     UF_face_type,            0, 0
         };


if(UF_UI_set_sel_mask(select,UF_UI_SEL_MASK_CLEAR_AND_ENABLE_SPECIFIC,nu
m_triples, mask_triples) == 0)
        return (UF_UI_SEL_SUCCESS);
    else
        return (UF_UI_SEL_FAILURE);
}

/*********************************************************************
*/
/* Subroutine to generate rational B-spline basis functions--open knot
vector
```

```
   by David F. Rogers. Copyright (C) 2000 David F. Rogers,
   All rights reserved.

   Name: rbasis
   Language: C
   Subroutines called: none
   Book reference: Chapter 4, Sec. 4. , p 296

   order      = order of the B-spline basis function
    d          = first term of the basis function recursion relation
    e          = second term of the basis function recursion relation
   weight[]   = array containing the homogeneous weights
    poles      = number of defining polygon vertices
    n_values    = constant -- poles + order -- maximum number of knot
values
    r[]        = array containing the rationalbasis functions
                 r[1] contains the basis function associated with B1
etc.
    t          = parameter value
    temp[]  = temporary array
    k_vector[] = knot vector
*/

//**********************************************************************
//
/*---------------------------------------------------------------------
*/
/*---------------------- Função Cálculo da curva ----------------------
*/
/*---------------------------------------------------------------------
*/
/* ---------------------------------------------------------------------
--
*----Desenvolvedor - Henrique Neves de Lucena
*----10/03/2009---------------------------
*----Função que calcula a curva Spline
*----Desenvolvido no Laboratório SCPM - Universidade Metodista de
Piracicaba
//**********************************************************************
//

void curva(tag_t cur, double parametroU,double c_points[3])
{

   //Variáveis para os dados da curva do UG
   double *curve_data;
   int curve_type;
   UF_CURVE_struct_p_t curve;

   //Variáveis para os dados da curva em uso
   double *vetorDeNos;
   double *verticeDoPoligono;
   double *peso;
   int numNos;
   int ordemDaCurva;
   int polos;
```

```
   //Variáveis usadas no cálculo dos pontos
   double *vetorRBasis;       //vetor contendo a função Basis
   double *temp;              //vetor temporario para o calculo
recursivo da função rational basis
   double primeiroTermo;      //primeiro termo da relação recursiva da
função rational basis
   double segundoTermo;       //segundo termo da relação recursiva da
função rational basis
   double sum;                //soma dos pesos e da função rational
basis
   int ordem;                 //contador para as ordens de 2 a n
   int  i,j,k;                //contadores

   //Alocação de Memória para até 1.000 pontos
   vetorDeNos = (double *)malloc(9000);
   temp = (double *)malloc(9000);
   vetorRBasis = (double *)malloc(9000);
   verticeDoPoligono = (double *)malloc(9000);
   peso = (double *)malloc(1000);

   //Lê o valores da curva
   UF_CURVE_ask_curve_struct(cur,&curve);
   UF_CURVE_ask_curve_struct_data(curve,&curve_type,&curve_data);

   //Atrela os valores da curva para serem usados na rotina
   polos=(int)curve_data[3];
   ordemDaCurva=(int)curve_data[4];
   numNos=polos+ordemDaCurva;

   //Atrela os valores da curva para vetor de Nos
   for (i=0;i<numNos;i++)
    vetorDeNos[i]= curve_data[5+i];

   //Atrela os valores da curva para os vertices e pesos
   for (i=0,j=0,k=numNos+5;i<polos;i++,j+=3,k+=4)
   {
    verticeDoPoligono[j] = curve_data[k];
    verticeDoPoligono[j+1] = curve_data[k+1];
    verticeDoPoligono[j+2] = curve_data[k+2];
    peso[i]=curve_data[k+3];
   }
   //Libera os vetores da curva original
   UF_CURVE_free_curve_struct(curve);
   UF_free(curve_data);

   //Inicio da rotina de calculo dos pontos
   if (vetorDeNos[numNos-1] - parametroU < 5E-6)
    parametroU = vetorDeNos[numNos-1];
   //Cálculo da função não racional Basis de primeira ordem
   for (i = 0; i< numNos-1; i++)
   {
    if (( parametroU >= vetorDeNos[i]) && (parametroU <
vetorDeNos[i+1]))
     temp[i] = 1.0;
    else
     temp[i] = 0.0;
   }
```

```
   //Cálculo da função não racional Basis de ordem 2 a seguintes
   for (ordem = 2; ordem <= ordemDaCurva; ordem++)
   {
    for (i = 0; i < polos; i++)
    {
     if (temp[i] != 0) /* se a função de ordem inferior é nula, não faz
o cálculo */
         primeiroTermo =
((parametroU-vetorDeNos[i])*temp[i])/(vetorDeNos[i+ordem-1]-
vetorDeNos[i]);
       else
      primeiroTermo = 0;
     if (temp[i+1] != 0)  /* se a função de ordem inferior é nula, não
faz o cálculo */
         segundoTermo =
((vetorDeNos[i+ordem]-parametroU)*temp[i+1])/(vetorDeNos[i+ordem]-
vetorDeNos[i+1]);
     else
      segundoTermo = 0;
     temp[i] = primeiroTermo + segundoTermo;
    }
   }
   if (parametroU == vetorDeNos[numNos-1]) /*leitura para o último ponto
   */
    temp[polos-1] = 1;
   // Cálculo da soma para o denominador da função racional
   sum = 0;
   for (i = 0; i < polos; i++)
    sum += temp[i]*peso[i];

   // Popular o vetor de funções RBasis considerando o peso
   for (i = 0; i < polos; i++)
   {
    vetorRBasis[i]=0.;
    if (sum != 0)
     vetorRBasis[i] = (temp[i]*peso[i])/(sum);
   }

   // Geração do ponto na curva
   for (i = 0; i < 3; i++)
   {
    c_points[i] = 0.;
    for (j = 0, k=i; j < polos; j++,k+=3){
     c_points[i] = c_points[i] + vetorRBasis[j]*verticeDoPoligono[k];
    }
   }
}
```

Anexo B – Cálculo do exemplo da figura 4.1.

Pode-se definir uma NURBS por [33]:

$$P(t) = \sum_{i=1}^{n+1} B_i R_{i,k}(t)$$

onde se tem:

P(t) = Curva NURBS,

B_i = Pontos do Polígono de Controle;

n+1 = quantidade de pontos no Polígono de Controle;

t = Parâmetro da curva que varia de t_{min} à $t_{máx}$;

k = Ordem da Curva B-Spline, podendo ser definida no intervalo $2 \leq k \leq n+1$;

$R_{i,k}(t)$ = Função de Suavização.

Matematicamente a função de suavização para uma NURBS é dada por:

$$R_{i,k} = \frac{h_i N_{i,k}(t)}{\sum_{i=1}^{n+1} h_i N_{i,k}(t)}$$

onde:

h_i = Peso de atração do vértice do Polígono de Controle;

$N_{i,k}$ = Função de Suavização de uma curva *B-Spline* não uniforme.

A função de suavização para uma curva *B-Spline* é definida por [33]:

$$N_{i,1}(t) = \begin{cases} 1 \text{ se } x_i \leq t < x_{i+1} \\ 0 \text{ para demais casos} \end{cases}$$

$$N_{i,k}(t) = \frac{(t - x_i)\, N_{i,k-1}(t)}{x_{i+k-1} - x_i} + \frac{(x_{i+k} - t)\, N_{i+1,k-1}(t)}{x_{i+k} - x_{i+1}}$$

onde:

x_i = vetores de nó em relação ao parâmetro t que respeitam a relação $x_i \leq x_i+1$.

Dados:

Ordem da curva (k): 3;

Peso dos pontos de polígono de controle: 1;

Número de pontos do polígono de controle (n+1): 5;

Vetor de nós: [0 0 0 0,33 0,66 1 1 1];

Valores do parâmetro t utilizados: [0 0,1 0,2].

Coordenadas dos pontos do polígono de controle:

Pi – [x, y]

P1 – [0, 1]

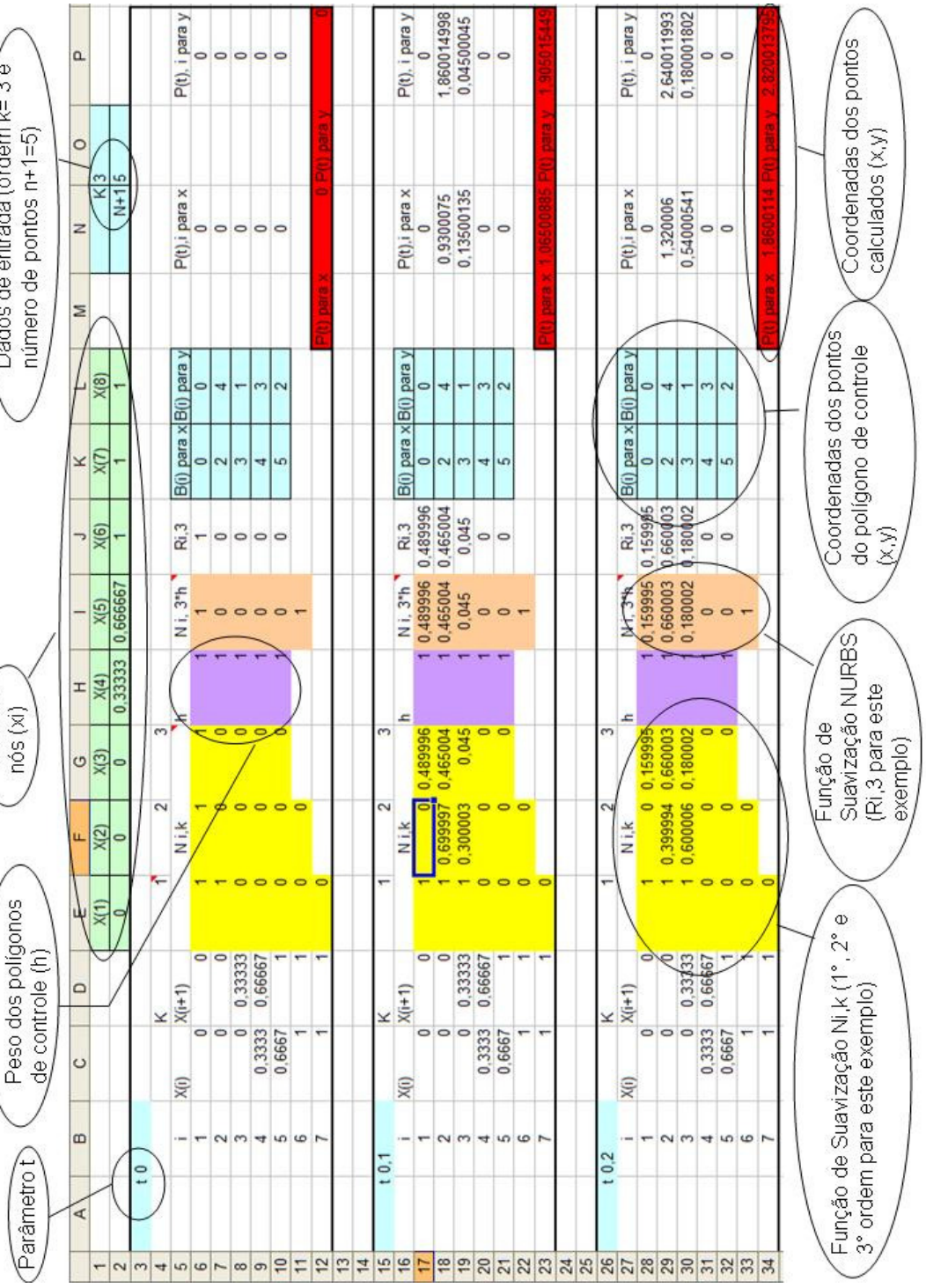
P2 – [2 , 4]

P3 – [3 , 1]

P4 – [4, 3]

P1 – [5, 2]

Tabela-Anexo B: Tabela de cálculo de coordenadas dos pontos relacionados à Figura 4.9.



Anexo C – Função do UGOPEN para obtenção dos dados da curva.

```
*****************************************************************
*/

Gets the structure pointer corresponding to the specified curve id.

NOTE: This routine is to be permanently withdrawn in the near
future. Use the routines in the UF_EVAL module instead.
Gets the structure pointer corresponding to the specified curve id.

Environment: Internal  and  External
*****************************************************************
*/
extern UFUNEXPORT int UF_CURVE_ask_curve_struct(
tag_t  curve_id ,/* <I>
                 Object identifier of the curve
                 */
UF_CURVE_struct_p_t * curve_struct  /*
<OF,free:UF_CURVE_free_curve_struct>
                                    Pointer to pointer of type
                                    UF_CURVE_struct_p_t. This routine
                                    allocates the structure and fills in
                                    the data.  This argument must be
                                    freed by calling
                                    UF_CURVE_free_curve_struct.
                                    */
);
```

Anexo D – Estrutura de dados da curva.

```
*************************************************************************
*/
Allows you to get the curve type and data for a specified curve
structure.

NOTE: This routine is to be permanently withdrawn in the near
future. Use the routines in the UF_EVAL module instead.
Allows you to get the curve type and data for a specified curve
structure. You may obtain the curve structure id using
UF_CURVE_ask_curve_struct.

Depending on the object type, curve_data may have the following data:
POINT                   curve_data[0 - 2]

LINE                    curve_data[0]           =  t0
                        curve_data[1]           =  tscale
                        curve_data[2]           =  periodicity
                        curve_data[3 - 5]       =  first point
                        curve_data[6 - 8]       =  second point

ARC                     curve_data[0]           =  t0
                        curve_data[1]           =  tscale
                        curve_data[2]           =  periodicity
                        curve_data[3]           =  start angle
                        curve_data[4]           =  end angle (radians)
                        curve_data[5]           =  radius
                        curve_data[6 - 8]       =  center
                        curve_data[9 - 11]      =  X axis of the
                                                   construction plane
                        curve_data[12 - 14]     =  Y axis of the
                                                   construction plane

CONIC                   curve_data[0]           =  t0
                        curve_data[1]           =  tscale
                        curve_data[2]           =  periodicity
                        curve_data[3]           =  T1
                        curve_data[4]           =  T2
                        curve_data[5]           =  K1
                        curve_data[6]           =  K2
                        curve_data[7]           =  conic type
                        curve_data[8-10]        =  center
                        curve_data[11-13]       =  axis1
                        curve_data[14-16]       =  axis2

B-CURVE                 curve_data[0]           =  t0
                        curve_data[1]           =  tscale
                        curve_data[2]           =  periodicity
                        curve_data[3]           =  number of poles
                        curve_data[4]           =  order
                        curve_data[*]           =  knot sequence
                                                   ( curve_data[3] +
                                                     curve_data[4] )
                        curve_data[*]           = array of homogeneous
                                                  poles (wx, wy, wz, w),
```

```
                                                        for each pole.
Environment: Internal  and  External
******************************************************************************
*/
extern UFUNEXPORT int UF_CURVE_ask_curve_struct_data(
UF_CURVE_struct_p_t curve_struct ,/* <I>
                                    Curve structure pointer.
                                    */
int * type ,/* <O>
            Object type
            */
double ** curve_data  /* <OF>
                      Array of object data.  This array is allocated by
                      this routine.  The caller must free it by calling
UF_free.
                      */
);
```

Anexo E – Função do UGOPEN para obtenção do comprimento de curva.

```
*****************************************************************************
*/
Computes the arc length of a curve between two input parameters. It will
also
compute the length of a solid edge.

Environment: Internal  and  External

*****************************************************************************
*/
extern UFUNEXPORT int UF_CURVE_ask_arc_length (
tag_t curve_tag ,/* <I>
                 Tag of curve to query for arc length
                 */
double start_param ,/* <I>
                    Start parameter of the curve from which the arc
                    length is to be calculated.
                    */
double end_param ,/* <I>
                  End parameter of the curve to which the arc length
                  is to be calculated.
                  */
UF_MODL_units_t unit_flag ,/* <I>
                           Any one of the following enumerated
constants:
               UF_MODL_UNITS_PART - same as parts units
               UF_MODL_INCH       - inches
               UF_MODL_MMETER     - millimeters
               UF_MODL_CMETER     - centimeters
               UF_MODL_METER      - meters
                           */
double* arc_length  /* <O>
                    arc length of the curve from start_param to
end_param.
                    */
);
```

Anexo F – Exemplo de arquivo DMIS gerado pelo aplicativo.

```
F(PM001)=FEAT/POINT,CART,3.677417,11.806711,75.000000,-0.555380,0.831597,-0.000000
MEAS/POINT,F(PM001),1
  PTMEAS/CART,3.677417,11.806711,75.000000,-0.555380,0.831597,-0.000000
ENDMES

F(PM002)=FEAT/POINT,CART,8.271700,14.679445,75.000000,-0.530377,0.847762,0.000000
MEAS/POINT,F(PM002),1
  PTMEAS/CART,8.271700,14.679445,75.000000,-0.530377,0.847762,0.000000
ENDMES

F(PM003)=FEAT/POINT,CART,12.861420,17.559990,75.000000,-0.532562,0.846391,-0.000000
MEAS/POINT,F(PM003),1
  PTMEAS/CART,12.861420,17.559990,75.000000,-0.532562,0.846391,-0.000000
ENDMES

F(PM004)=FEAT/POINT,CART,17.420985,20.487779,75.000000,-0.553116,0.833105,-0.000000
MEAS/POINT,F(PM004),1
  PTMEAS/CART,17.420985,20.487779,75.000000,-0.553116,0.833105,-0.000000
ENDMES

F(PM005)=FEAT/POINT,CART,21.828425,23.636609,75.000000,-0.635395,0.772187,0.000000
MEAS/POINT,F(PM005),1
  PTMEAS/CART,21.828425,23.636609,75.000000,-0.635395,0.772187,0.000000
ENDMES

F(PM006)=FEAT/POINT,CART,25.337904,27.727327,75.000000,-0.846327,0.532664,0.000000
MEAS/POINT,F(PM006),1
  PTMEAS/CART,25.337904,27.727327,75.000000,-0.846327,0.532664,0.000000
ENDMES

F(PM007)=FEAT/POINT,CART,29.308270,30.585897,75.000000,0.313062,0.949733,0.000000
MEAS/POINT,F(PM007),1
  PTMEAS/CART,29.308270,30.585897,75.000000,0.313062,0.949733,0.000000
ENDMES

F(PM008)=FEAT/POINT,CART,30.480677,25.976247,75.000000,0.942445,-0.334361,0.000000
MEAS/POINT,F(PM008),1
  PTMEAS/CART,30.480677,25.976247,75.000000,0.942445,-0.334361,0.000000
ENDMES

F(PM009)=FEAT/POINT,CART,27.756134,21.331104,75.000000,0.744742,-0.667352,0.000000
MEAS/POINT,F(PM009),1
  PTMEAS/CART,27.756134,21.331104,75.000000,0.744742,-0.667352,0.000000
ENDMES

F(PM010)=FEAT/POINT,CART,23.463251,18.068817,75.000000,0.501243,-0.865307,-0.000000
MEAS/POINT,F(PM010),1
  PTMEAS/CART,23.463251,18.068817,75.000000,0.501243,-0.865307,-0.000000
ENDMES
```