JOÃO WALTER BRUNO FILHO

PROGRAMAÇÃO DE MÁQUINAS-FERRAMENTAS DE 3 EIXOS ASSISTIDA POR EDITOR PARA MODELAGEM E SIMULADOR GRÁFICO 3D DE PEÇAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Hélio Pedrini

CURITIBA

2008

JOÃO WALTER BRUNO FILHO

PROGRAMAÇÃO DE MÁQUINAS-FERRAMENTAS DE 3 EIXOS ASSISTIDA POR EDITOR PARA MODELAGEM E SIMULADOR GRÁFICO 3D DE PEÇAS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Hélio Pedrini

CURITIBA

2008

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

B-Rep	*Boundary Representation*	Representação por Fronteiras
CAD	*Computer-Aided Design*	Projeto Auxiliado por Computador
CAE	*Computer-Aided Engineering*	Engenharia Auxiliada por Computador
CAM	*Computer-Aided Manufacturing*	Manufatura Auxiliada por Computador
CAPP	*Computer-Aided Process Planning*	Planejamento de Processos Auxiliado por Computador
CD	*Cell Decomposition*	Decomposição de Células
CIM	*Computer Integrated Manufacturing*	Manufatura Integrada por Computador
CNC	*Computer Numerical Control*	Controle Numérico por Computador
CSG	*Constructive Solid Geometry*	Geometria Sólida Construtiva
FMS	*Computer Integrated Manufacturing*	Manufatura Integrada por Computador
GPU	*Graphics Processing Unit*	Unidade de Processamento Gráfico
ISO	*International Organization for Standardization*	Organização Internacional para Padronização
LWJGL	*Lightweight Java Gaming Library*	Biblioteca para Jogos em Java Lightweight
NC	*Numerical Control*	Controle Numérico
OpenGL	*Open Graphics Library*	Biblioteca OpenGL
PPI	*Pure Primitive Instancing*	Instanciação de Primitivas Puras
RS	*Recommended Standard*	Padrão Recomendado
S	*Sweeping*	Varredura
SOE	*Spatial Occupancy Enumeration*	Enumeração de Ocupação Espacial
STEP-NC	*Standard for the Exchange of Product Model Data - Numerical Controller*	Padrão para Troca de Dados de Produto - Controlador Numérico
STL	*STereoLitography*	Estereolitografia

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A atividade de planejamento de processos é de grande importância para a fabricação mecânica, pois possibilita a racionalização das decisões a fim de obter eficazmente a peça a ser usinada de acordo com as especificações de projeto. Dessa forma, a redução do tempo de produção e dos custos com material e mão-de-obra torna-se uma questão fundamental.

Esforços internacionais têm sido realizados para promover maior integração entre projeto, processo e fabricação. Uma iniciativa é a utilização de um conjunto de informações, conhecidas como *features*, para descrever a forma e os atributos de uma peça. As máquinas-ferramentas tradicionais executam comandos escritos em linguagem G/M, os quais correspondem aos movimentos de eixos da máquina e funções das ferramentas. O uso de *features* permite a usinagem da peça por meio de uma seqüência de operações de alto nível de abstração para a remoção de material.

Este trabalho, inserido em um projeto multidisciplinar nas áreas de Ciência da Computação e Engenharia Mecânica, descreve a metodologia para desenvolvimento de um protótipo que visa auxiliar o processo de usinagem em máquinas-ferramentas de 3 eixos por meio do emprego de *features*.

Os principais módulos que compõem o protótipo são o de edição da peça, simulação do modelo e transmissão do código para a máquina-ferramenta. O módulo de edição permite a inserção dos parâmetros geométricos relacionados com as features. Após a edição, o modelo pode ser visualizado e avaliado pelo usuário. Este modelo permite ainda a geração do código (programa) a ser interpretado pela máquina-ferramenta. A validação do programa é facilitada com o auxílio de um simulador gráfico. Os recursos de simulação propostos baseiam-se na representação gráfica da trajetória das ferramentas de corte. Após a verificação do programa, o usuário pode transmiti-lo para o comando da máquina-ferramenta. Para a transferência de dados, adotou-se o protocolo RS-232C entre as portas seriais do computador e da máquina-ferramenta.

O modelo da peça é descrito pela combinação de duas técnicas de representação, a geometria sólida construtiva e a enumeração de ocupação espacial, a partir de um conjunto de primitivas gráficas como cubos, paralelepípedos e cilindros. As primitivas são combinadas para formar um novo objeto sólido por meio de uma seqüência ordenada de operações Booleanas. Uma estrutura hierárquica é usada para controlar a aplicação das operações Booleanas.

CAPÍTULO 1

INTRODUÇÃO

1.1 Caracterização do Problema

Com o contínuo avanço tecnológico, a indústria de manufatura requer constante atualização. Há algum tempo, a utilização de máquinas eletromecânicas era o cenário habitual, onde as máquinas necessitavam de controle manual em conjunto com a entrada de dados para a manufatura. Hoje, a utilização de um ambiente computacional, onde não há necessidade de intervenção manual no processo, é uma realidade em grande parte do cenário industrial. Entretanto, algumas tecnologias hoje empregadas ainda vêm de um ambiente pouco amigável e de técnicas que perduram desde a época dos cartões perfurados, como é o exemplo de algumas máquinas-ferramentas.

As mais conhecidas delas, chamadas CNCs (Controle Numérico por Computador, do inglês *Computer Numerical Control*), embora sejam máquinas de alta precisão e desempenho e estejam bastante à frente de suas antecessoras, ainda são utilizadas de forma muito semelhante à época em que foram lançadas. Poucas mudanças ocorreram desde a sua criação até os dias de hoje. Isso pode ser visto como uma grande barreira ao processo de evolução do projeto e desenvolvimento das peças de manufatura e de mantenabilidade das empresas que produzem essas peças.

O emprego de softwares de apoio à criação para a indústria de manufatura, os chamados CAD/CAM (do inglês *Computer-Aided Design* e *Computer Aided Manufacturing*) têm se tornado mais comuns, mas ainda assim não há uma grande integração entre esses sistemas, e muito menos entre eles e as máquinas utilizadas para a fabricação das peças finais, as máquinas-ferramentas.

O processo de manufatura com a utilização de CNCs pode ser realizado de várias formas, tais como entrada de dados manual, programação manual ou programação assistida por computador [7]. Todas elas necessitam de um alto grau de conhecimento técnico sobre o

funcionamento dos CNCs e utilização dos softwares que apoiam o desenvolvimento de peças de manufatura, e não apenas de conhecimento sobre o produto. Além disso, adaptações são necessárias em todas as etapas da produção para atender às demandas de uma máquina-ferramenta e, mesmo que haja um sistema ágil e de produtividade para uma determinada máquina-ferramenta, a cadeia anterior à ela geralmente é projetada para atender aos fins específicos daquela máquina, pois não há uma padronização entre as máquinas-ferramentas ou CNCs, isto é, não se consegue utilizar um mesmo código para mais de uma máquina-ferramenta.

Desde o projeto e a digitação do código como processo manual até o projeto digital em CAD (importação em um sistema CAM), despende-se muito tempo para planejar e desenvolver um bom produto. Hoje em dia, a necessidade de qualidade e agilidade em todo o processo de desenvolvimento é uma questão não somente de expansão de negócios, pois não há garantias de posição no mercado, mas da constante busca por novos meios e técnicas de produção.

O padrão ISO 14649 [33, 44] (comumente chamado STEP-NC) está sendo desenvolvido com o intuito de melhorar a integração das técnicas de manufatura, criando um padrão de dados baseado nas características dos produtos, as chamadas *features*. Por meio do emprego das *features*, pode-se explorar todo o processo de criação em uma só ferramenta, ao invés de necessitar da utilização de várias em conjunto, além de possibilitar o desenvolvimento de peças de manufatura com muito menos treinamento e especialização.

1.2 Objetivos e Metas

Este trabalho tem como principal finalidade investigar a criação de uma ferramenta computacional que incorpore algumas *features* descritas na norma ISO 14649 e desenvolver um protótipo para tal. Este sistema é composto, basicamente, por um editor, um simulador e um módulo de transmissão. O editor permitirá o projeto de uma peça a ser usinada baseando-se nas características da norma. O simulador permitirá a visualização de um modelo virtual da peça, além de gerar o código a ser executado na máquina-ferramenta. O módulo de transmissão permitirá o envio do código gerado à máquina ferramenta para execução.

Uma meta deste trabalho é responder questões relativas ao processo de manufatura basea-

do em máquinas-ferramentas e nas *features* apresentadas na norma ISO 14969, aplicando-se conhecimento de recursos computacionais de forma a contribuir com a área de usinagem, sem a pretensão de se construir um protótipo superior aos aplicativos CAD e CAM disponíveis, os quais são mantidos e atualizados por empresas (corporações) especializadas nesse ramo de atividade industrial, mas sim a investigação e incorporação de novas funcionalidades, tanto com o emprego das *features* quanto de algoritmos para modelagem e simulação.

Um modelo é criado a partir de uma vista tridimensional de algumas das *features* encontradas na norma ISO 14649 para permitir a fabricação de peças de usinagem. Uma interface gráfica permite ao usuário a entrada dos valores dos atributos de cada *feature* e a sua exibição de acordo com os diferentes valores especificados. Pela aplicação de um conjunto de *features*, o projetista pode criar um modelo virtual da peça e, com o simulador, verificar se ele atende aos requisitos especificados no projeto. Caso seja necessário algum ajuste, o projetista tem a possibilidade de realizar as devidas modificações até que o modelo corresponda à peça desejada, reduzindo-se assim custos com material e mão-de-obra.

1.3 Contribuições

O protótipo desenvolvido, que agrega metodologias das áreas de Computação e Engenharia Mecânica, propicia uma redução de tempo e custo do processo de usinagem de peças em máquinas-ferramentas.

O protótipo pode ser utilizado no treinamento de projetistas em atividades de usinagem, favorecendo a auto-aprendizagem e o treinamento fora do ambiente operacional.

A metodologia utilizada no desenvolvimento do protótipo incluiu a etapa de análise de requisitos, o estudo das *features*, a seleção da representação dos objetos, a escolha do ambiente de desenvolvimento, a implementação do protótipo e a avaliação dos resultados. Uma arquitetura do sistema foi proposta, constituída dos módulos de edição, simulação e transmissão.

O modelo da peça é descrito pela combinação de duas técnicas de representação, a geometria sólida construtiva (CSG) e a enumeração de ocupação espacial (SOE), a partir de um conjunto de primitivas gráficas como cubos, paralelepípedos, cilindros e elipsóides. As primitivas são combinadas para formar um novo objeto sólido por meio de uma seqüência ordenada

de operações Booleanas. Uma estrutura hierárquica é usada para controlar a aplicação das operações Booleanas.

1.4 Estrutura da Dissertação

Este trabalho está organizado como segue. O capítulo 2 apresenta as máquinas-ferramentas, especificamente os CNCs, seu funcionamento, diferenças e a sua evolução. Em seguida, alguns procedimentos para a operação dessas máquinas-ferramentas são verificados, além do estudo de parte da norma ISO 14649. Alguns métodos de representação de objetos tridimensionais estudados para o desenvolvimento da ferramenta proposta são apresentados. As alternativas analisadas do ambiente de desenvolvimento do protótipo são descritas. O capítulo 3 descreve a metodologia proposta neste trabalho, destacando-se os processos a serem empregados no desenvolvimento a partir de uma visão geral do sistema e os principais componentes do protótipo, sendo eles o módulo de edição, o módulo de simulação e o módulo de transmissão. O capítulo 4 apresenta os resultados obtidos a partir da aplicação da metodologia, assim como as dificuldades encontradas no processo de desenvolvimento. O capítulo 5 apresenta algumas considerações finais. O apêndice A descreve as características e propriedades geométricas de algumas das *features* que fazem parte da norma ISO 14649.

CAPÍTULO 2

REVISÃO BIBLIOGRÁFICA

Desde as civilizações antigas, o ser humano tem buscado ferramentas com a finalidade de facilitar suas tarefas. Com o passar do tempo, a necessidade por produtos cresceu significativamente, necessitando-se do desenvolvimento de novos métodos de produção e exploração.

Essas novas técnicas criaram a necessidade de novas ferramentas de trabalho em todas as etapas do processo de produção. A evolução dessas ferramentas propiciou o surgimento de novas máquinas para suprir as necessidades de produção da época, sendo estas chamadas de máquinas-ferramentas. No início, essas máquinas operavam com controle totalmente manual.

No princípio da revolução industrial, as máquinas-ferramentas adicionaram poder aos seres humanos e precisão aos processos. Com elas, bens puderam ser produzidos mais rapidamente e com melhor qualidade, aumentando drasticamente a produtividade, tal que os produtos manuais deram lugar aos bens industriais [22].

Essas máquinas foram construídas na Europa, mas com o tempo se expandiram ao redor do mundo. No início da revolução industrial, esses produtos ainda eram feitos de maneira personalizada, com máquinas-ferramentas operadas manualmente. Posteriormente, a criação de partes intercambiáveis trouxe uma melhora significativa à manufatura. A partir dessa idéia, surgiu o conceito de linha de produção e a produção em massa tornou-se realidade, trazendo uma produção maior, melhor e mais acessível de bens. Chang et al. [7] cita que, ainda assim, nessa época a variedade de itens era limitada ao alto valor de mudança no sistema de manufatura.

Desde a sua criação, até meados da segunda guerra mundial, as máquinas pouco haviam mudado. Elas haviam ganho precisão, pois agora possuíam controladores mecânicos, eletromecânicos ou pneumáticos. Entretanto, foi durante a segunda guerra e pela disputa da produção de armas que essas máquinas ganharam destaque, sendo que a redução dos custos de manufatura tornou-se o objetivo principal, principalmente para o desenvolvimento de aviões. A

partir dessa procura, surgiu o conceito de controle numérico (NC, do inglês *Numerical Control*, com o objetivo de substituir a habilidade humana por uma máquina programável.

2.1 Controle Numérico

O controle de uma máquina-ferramenta utilizando entrada variável, tal como cartões perfurados ou programa armazenado, é conhecido como Controle Numérico (NC, do inglês *Numerical Control*). NC foi definido pela *Electronic Industries Association* (EIA) como "sistema em que as ações são controladas pela inserção direta de dados numéricos em algum ponto. O sistema precisa automaticamente interpretar pelo menos parte desses dados" [7].

As máquinas NC são geralmente descritas pelo número de eixos de movimento. Uma máquina de 3 eixos é capaz de mover a ferramenta de corte em três direções relativas à peça [22]. Há também a descrição é chamada de $2\frac{1}{2}$ eixos, onde geralmente um dos eixos se mantém constante enquanto os outros dois eixos realizam a operação de corte. Esse tipo de máquina possui grande utiliação, pois abrange uma grande quantidade de operações e consegue manufaturar uma grande quantidade de peças, servindo a diversas áreas com essa configuração. Já uma máquina de 5 eixos, pode posicionar a ferramenta em um espaço tridimensional e controlar sua orientação [22].

Um sistema de máquina-ferramenta de controle numérico contém uma unidade de controle de máquina (MCU, do inglês *Machine-Control Unit*) e a máquina-ferramenta em si. A MCU é então dividida em dois elementos: a unidade de processamento de dados (DPU, do inglês *Data Processing Unit*) e a unidade de controle de laços de repetição (CLU, do inglês *control-loops unit*), como pode ser visto na figura 2.1. A DPU processa os dados codificados lidos do dispositivo de entrada e passa informação da posição de cada eixo, sua direção de movimento, alimentação, e sinais de controle de funções auxiliares para a CLU. A CLU opera o mecanismo de direção da máquina, recebe o sinais de realimentação (*feedback*) sobre a posição atual e velocidade de cada um dos eixos, então sinaliza quando uma operação é concluída. A DPU lê seqüencialmente os dados quando cada linha teve execução concluída quando informada pela CLU.

Chang et al. [7] descrevem os componentes de uma DPU como sendo:

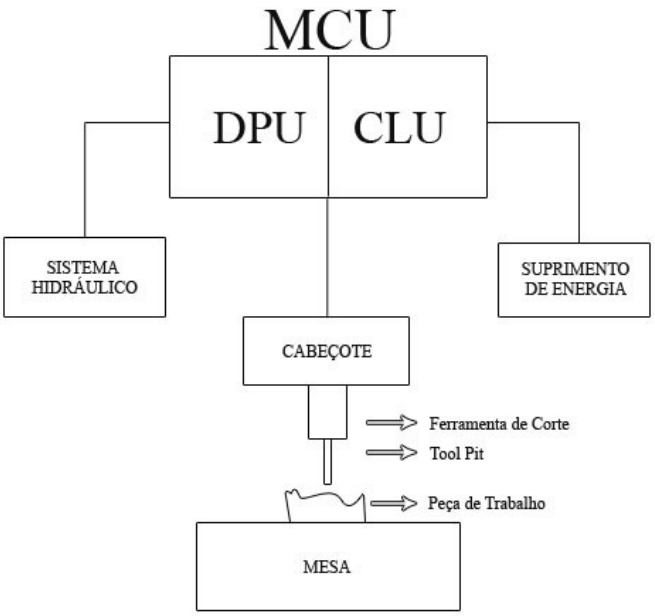


Figura 2.1: Sistema de controle numérico. Adaptado de [7].

- dispositivo de entrada de dados, como leitor de cartão perfurado, leitor de fita magnética, porta RS-232-C, e assim por diante;
- circuitos de leitura de dados e lógica de verificação de paridade;
- circuitos de decodificação para dados descritos entre os eixos do controlador;
- um editor.

e também uma CLU que contém:

- um interpolador que supre os comandos de movimentos de máquina entre os pontos de dados para o movimento da ferramenta;
- *hardware* de controle de posição de laços para todos os eixos de movimento, em que cada eixo tem um laço separado de controle;
- laço de controle de velocidade, onde alimentação é necessária;
- circuitos captadores de desaceleração e retrocesso;

- controle de funções auxiliares, como fluído refrigerante *liga/desliga*, troca de engrenagem e *eixo árvore liga/desliga*.

Como visto anteriormente, o controle de movimento das máquinas-ferramentas NC é feito traduzindo códigos NC em comandos de máquina. Os códigos NC podem ser, de maneira geral, classificados em dois grupos:

- comandos para controle individual de componentes de máquina, tais como controle de motor *liga/desliga*, seleção de velocidade do *eixo árvore*, troca de ferramenta, controle de resfriamento (tarefas realizadas pela execução de pulsos eletrônicos ao sistema de transmissão ou controle lógico;

- comandos para o controle do movimento relativo da peça de trabalho e ferramentas de corte. Estes comandos são compostos de informações como eixo e distância a ser movida em uma determinada unidade de tempo por uma mesa de trabalho (peça de trabalho) ou ferramenta de corte. Tais informações são traduzidas em comandos de execução de máquina e comandos de controle de movimento e são levados ao sistema eletromecânico de controle.

As máquinas-ferramentas são divididas em NC, CNC e DNC (Controle Numérico Distribuído, do inglês *Distributed Numerical Control*) [46], de acordo com o controlador. O sistema NC foi originalmente construído utilizando um circuito dedicado, onde todas as funções tinham que ser projetadas e implementadas em *hardware*. Cada vez que um programa NC precisava ser executado para gerar uma nova peça, o cartão perfurado (ou outro dispositivo de leitura) precisava ser lido, e a DPU consistia apenas neste dispositivo de leitura, não possuindo nenhum sistema de edição.

Em uma máquina CNC, microcomputadores foram introduzidos nos controladores, com um sistema operacional dedicado, ambiente de desenvolvimento, além dos *software* já descritos. Chang et al. [7] apresentam a interface RS-232 como talvez o método mais típico de comunicação de dados para os CNCs, onde o computador integrado lê e armazena o programa, como um arquivo. Além disso, deve-se mencionar o cartão perfurado, fita magnética, fita *mylar*, disco flexível e a comunicação direta(Chang et al. [7] *apud* Pressman e Williams [31]).

Segundo Zeid [46], o conceito de CNC possui algumas falhas. Um mesmo programa NC utilizado em várias máquinas-ferramentas tem que ser carregado separadamente em cada máquina, uma tarefa improdutiva e repetitiva. Ainda mais importante, sistemas CNC são limitados quanto a realimentação de dados. Vários dados que poderiam ser obtidos diretos do chão de fábrica, como ociosidade de máquinas, *status* do trabalho atual*work-in-progress*, taxas de produção, taxas de sobras (*scrap*) e outros dados de produção, poderiam ser melhor explorados com um sistema de comunicação eficiente para melhorar a gerência dos processos de manufatura.

Já um sistema DNC possui um computador central e é responsável pela comunicação e controle de várias máquinas CNC. Este computador pode carregar os programas em quaisquer máquinas na rede, evitando a necessidade de fazê-lo individualmente. Também executa o retorno das tarefas como os relatórios de produção, processamento e coleção de dados, além da comunicação entre vários componentes do sistema DNC [46].

Em alguns casos, há vários níveis hierárquicos de computadores e redes entre as máquinas CNC e o computador central. Os computadores intermediários (em alguns casos chamados computadores satélites) fornecem vários níveis de controle local para diferentes máquinas CNC ao longo do sistema DNC.

A maior vantagem do sistema DNC é que o sistema pode ser monitorado centralmente. Isto é importante ao lidar com diferentes operadores, em diferentes turnos, trabalhando em máquinas diferentes [46].

2.1.1 Programação NC

Um programa NC é uma combinação de código de máquina-ferramenta e instruções específicas da máquina. Ele contém informação geométrica sobre a peça e informações de movimento para a ferramenta de corte.

Sob o aspecto da programação NC, pode-se citar algumas de suas características principais. Zeid [46] apresenta quatro métodos comuns utilizados para criar programas NC:

1. programação manual da peça: o programador NC escreve o código manualmente e o armazena em um arquivo, similar à programação C, C++ ou Java;

2. programação da peça assistida por computador: o programador NC usa um software específico para gerar os programas NC;

3. programação da peça usando um sistema CAD/CAM: o programador NC usa um pacote de sistema CAD/CAM para gerar programas NC. Esta abordagem é semelhante à abordagem 2. Entretanto, o benefício aqui é que o programador utiliza uma base de dados das geometrias de peças e operações, não havendo necessidade de traduzir e importá-la;

4. entrada manual de dados: o programador NC usa o controlador de uma máquina-ferramenta para entrar com dados NC diretamente. O programador pode usar o controlador para verificar o caminho da ferramenta após a entrada dos dados para assegurar sua corretude antes de usá-la na usinagem. O controlador ainda permite ao programador salvar os seus dados e programa.

Independentemente do método escolhido, o código ou programa deve ser entendido pelo MCU. Um processo de conversão (pós-processamento) pode ser necessário quando linguagens de alto nível são usadas para escrever o programa NC da peça.

Como dito, atualmente a operação de máquinas-ferramentas CNCs ainda necessita de um grande nível de especialização. A utilização da chamada linguagem G/M (ISO 6983), que se baseia em instruções de movimento (G1, G2, G3) e de troca (M1, M8) de ferramenta, embora esteja bastante próxima e de forma bastante clara dos princípios fundamentais da construção das peças ou partes de manufatura, ainda possui muitos problemas com relação à padronização da sua utilização. O grande problema é que, na criação dessa linguagem, algumas características não haviam sido previstas, ou estavam de alguma forma incompletas, necessitando de alterações e adaptações. Com o passar do tempo, essas "falhas" da chamada Linguagem G, ou norma G, tiveram que ser cobertas pelas empresas que produziam as máquinas-ferramentas. Cada uma dessas empresas produziu soluções próprias, sendo que cada máquina apresentava uma ramificação particular da linguagem G.

A partir da metade da década de 1990, uma nova interface de dados, chamada STEP-NC (ISO 14649), passou a ser desenvolvida sob a direção do comitê técnico TC184 da ISO (do inglês *International Organization for Standardization*), dentro dos subcomitês SC1 e SC4 [33], o

que facilitou o desenvolvimento para a área de manufatura e, com isso, possibilitou a mudança de características que hoje estão presentes quando se fala em máquinas-ferramentas. Buscou-se padronizar o processo de manufatura e, especificamente, estabelecer um modelo de dados para os controladores numéricos a serem criados, deixando para trás uma linguagem da época dos cartões perfurados, de programas muito extensos, difíceis de manter, corrigir e com limitado controle de execução.

A geometria tanto do bloco bruto quanto da peça final é descrita pelo padrão conhecido como STEP (*STandard for Exchange of Product model data*), descrita na norma ISO 10303, inicialmente dedicado a aplicações de manufatura, que define um padrão para modelos de dados específicos para uma ampla variedade de produtos, tais como da indústria naval, automobilística e eletrônica [24]. Com a utilização dessa sintaxe, pretende-se realizar troca direta de dados entre o software de CAD/CAM e o NC. Dados poderiam ser importados diretamente de sistemas CAD, sem a necessidade de transformação ou adaptação dos mesmos, eliminando a necessidade do pós-processamento, descrito anteriormente.

Por meio de uma nova linguagem, a STEP pretende substituir as instruções G/M por tarefas de manufatura, tornando possível a utilização das chamadas *features* de manufatura, como furos e rebaixos, entre outros.

O Manual de Aplicação STEP [36] descreve alguns termos relacionados a *features*:

- *features*: os elementos geométricos e a orientação que definem informação de superfície e volumes sem qualquer semântica;
- semântica: o significado ou implicação associada com o processo de manufatura para remoção de um volume de material;
- *feature* de usinagem (*machining feature*): é a combinação de uma *feature* e semântica, ou seja, a informação geométrica sobre uma forma bem definida e as semânticas de manufatura que ajudam na associação do processo de remoção de volume;
- *feature* de manufatura (*manufacturing feature*): é a combinação de uma *feature* e a implicação de que um volume de material foi removido criando uma *feature*.

Neste trabalho, o termo *feature* será empregado como *feature de manufatura*, especificamente no que diz respeito a ela na norma ISO 14649. Na norma, todas as operações necessárias para produzir a peça a partir do bloco inicial são descritas por uma seqüência de tarefas de manufatura. Segundo Weck et al. [44], a ISO 14649 fornece um modelo de dados orientado a objetos para CNC, com uma estrutura detalhada de interface de dados que incorpora a programação baseada em *features*, onde há um grande conjunto de informações como *feature* a ser usada na usinagem, tipo de ferramenta e operações a realizar e o plano de trabalho, ou, de forma mais abrangente, ela define um conjunto de estratégias de usinagem [33].

Para cada operação a realizar sobre uma ou mais *features* é definida uma instrução chamada *workingstep*. Esses *workingsteps* fornecem as bases para o funcionamento dos planos de trabalho definidos em instruções chamadas *workplan* (plano de trabalho) que definem a fabricação do componente [33].

Uma breve descrição das diferenças entre a situação atual das máquinas-ferramentas e a proposta da norma STEP-NC pode ser vista na figura 2.2.

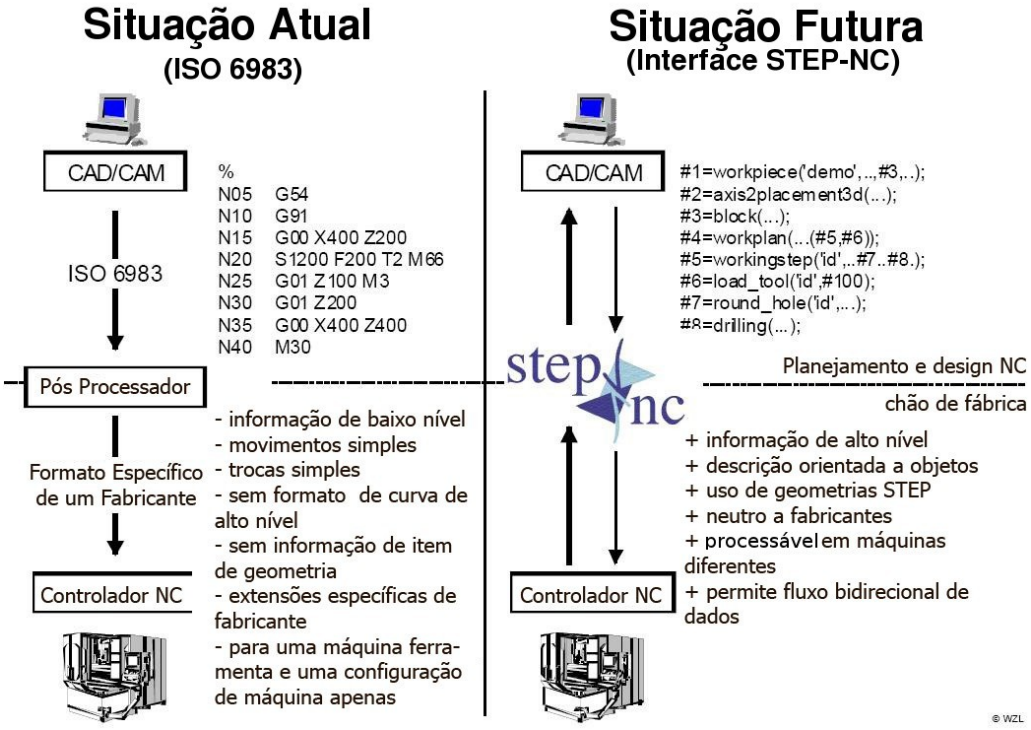


Figura 2.2: Diferenças entre interfaces Atual e STEP-NC. Adaptado de [44].

Para demonstrar algumas de suas características, um subconjunto de *features* de manufa-

tura é apresentado no apêndice A. Estas *features* são descritas na STEP-NC e estão restritas a apenas um pequeno conjunto, sem as subdivisões que apresentam na versão final da norma.

A meta da ISO 14649 de não haver necessidade de qualquer adaptação em código ao se implantar um programa de uma máquina-ferramenta para outra está a caminho de se tornar realidade [33]. Prevê-se que códigos em STEP-NC sejam processados por controladores inteligentes a médio prazo. Além disso, alguns trabalhos como [34, 42] já apresentam alternativas de implementação para sistemas desenvolvidos para máquinas que possuam suporte ao STEP-NC, mas essas máquinas ainda não foram desenvolvidas ou produzidas comercialmente.

A ISO 14649 é realmente um grande avanço em inúmeros aspectos, mas é importante ressaltar que a implantação da norma não faz referência às máquinas utilizadas atualmente, ou seja, sua aplicação está voltada exclusivamente para máquinas já criadas com suporte à ela. Com a utilização desta norma, como já dito, pretende-se que não haja a necessidade de qualquer tradução entre o código da máquina e os arquivos gerados pelos sistemas CAD/CAM. Dessa maneira, todo um parque de equipamentos de grande precisão e qualidade estaria sendo descartada; as máquinas que não seguem este padrão não poderiam ser utilizadas. É importante, então, criar um meio termo entre estes padrões. Algo que esteja próximo à estratégia de *features* e ainda assim seja compatível com as máquinas-ferramentas atuais, ou que possa interpretar de alguma forma as definições da ISO 14649 para utilização das máquinas-ferramentas não aderentes a norma.

Uma abordagem para este problema é possibilitar a utilização das *features* de usinagem através de um sistema CAD/CAM. Este sistema pode oferecer suporte às máquinas-ferramentas atuais e ainda assim entrar em concordância com a norma ISO 14649 futuramente.

A construção de um sistema CAD/CAM envolve vários processos e diferentes campos da computação ligados com Geometria e a Engenharia Mecânica (a área fim para o sistema, neste caso). Neste trabalho, duas características importantes para o desenvolvimento de um protótipo são ressaltadas: a Representação de Sólidos e o Ambiente de Desenvolvimento.

2.2 Métodos de Representação de Sólidos

Modelos sólidos são conhecidos por serem representações dos objetos completos, válidos e não ambíguos. Um sólido completo é aquele que possibilita a um ponto no espaço ser classificado em relação ao objeto, se está dentro, fora ou no objeto. Essa classificação também é chamada de endereçamento espacial [46].

Em meados dos anos 70, Brown et al. [4] previram que sistemas de modelagem geométrica seriam reconhecidos como os componentes centrais dos sistemas flexíveis para projeto e produção automática de uma ampla variedade de bens mecânicos, por fornecerem meios para a criação, edição e manutenção de várias representações alternativas da geometria de objetos sólidos (partes móveis, ferramentas, entre outros) e servir como fontes de dados geométricos para um grande número de aplicações.

Brown et al. [4] apresentaram também um conjunto de características desejáveis em um sistema de modelagem industrial para peças mecânicas e montagens (mecanismos). Eles especificaram a abrangência das peças que podem ser construídas e seus tipos, como peças não esculpidas, esculpidas funcionalmente e esculpidas aninhadamente, e delimitaram seu trabalho sob peças não esculpidas (como *brackets* e *shafts*), bens predominantemente feitos por métodos de produção em lote ou seriais, os mais encontrados. As características desejáveis em um sistema são:

- Cobertura geométrica e tolerância: ao menos 90 a 95% das peças encontradas em produtos típicos feitos por métodos de produção serial ou em lote devem ser cobertos, e a informação variacional com as especificações atuais da peça deve ser fornecida. Implicações técnicas: o domínio de qualquer esquema de representação para sólidos nominais incorporados no sistema deve incluir a classe de conjuntos (*r-sets*) definível pelo conjunto de operações regularizado em meio-espaços arbitrariamente posicionados, limitados por superfícies planares, cilíndricas, cônicas, esféricas e toroidais. Representações variacionais apropriadas ao domínio devem ser fornecidas.

- Completude informacional: o sistema deve conter todas as características geométricas sobre cada objeto representado para assegurar suficiência de informações para aplicações

atuais e futuras. Implicações técnicas: o sistema deve conter ao menos um esquema de representação para geometria nominal e variacional que seja formalmente completo.

- Eficiência: uma variedade de aplicações deve ser suportada eficientemente. Implicações técnicas: o sistema deve conter múltiplas representações dos objetos (em esquemas distintos) porque nenhum esquema único de representação é uniformemente melhor para todas as aplicações e sobre qualquer critério (Ex. facilidade de criação ou transmissão).

- Confiança: o sistema deve garantir, sem intervenção humana, a corretude de seus dados. Implicações técnicas: cada representação deve se formalmente válida, e todas as representações de cada objeto devem ser formalmente consistentes com uma representação completa do objeto.

- Facilidade e extensibilidade: representações de objetos devem ser fáceis de se criar, editar e acessar por usuários (humanos ou autômatos) e devem ser possíveis com apenas esforço moderado de se aumentar a cobertura geométrica e a abrangência das aplicações suportadas. Implicações técnicas: essas propriedades e outras como portabilidade não são bem compreendidas no âmbito formal. De um modo geral, implicam em boa prática de planejamento e implementação.

Brown et al. [4] colocam como senso comum a não existência de sistemas de modelagem geométrica com as características mostradas, apontando principalmente o fato de serem deixadas, para os usuários humanos, tarefas como detecção de erros e inconsistências.

Apontando alguns desses termos como completude, validade e não-ambigüidade, Zeid[46] descreve duas abordagens que sistemas oferecem para a criação de modelos sólidos: primitivas e *features*. A abordagem permite aos projetistas o uso de formas pré-definidas (primitivas) como blocos de construção (como peças Lego) para criar sólidos complexos. Projetistas devem usar operações Booleanas para combinar as primitivas. Essa abordagem é limitada pelas formas das primitivas.

As *features* são mais flexíveis por permitirem a construção de sólidos mais complexos e elaborados do que as primitivas oferecem [46]. A abordagem de *features* possui características diferentes das utilizadas pela norma, mesmo que algumas denominações sejam análogas, não

correspondendo diretamente a *features* de manufatura. Ela é definida como uma forma e uma operação para construir peças [46] ou como, essencialmente, macros de formas [43], enquanto as *features* da norma, de acordo com [24], são definidas como uma *feature* geométrica (uma entidade sólida conectada) que é o subconjunto de um volume removido pela ferramenta em uma operação de manufatura. Esse volume corresponde ao material que é removido na operação de manufatura. A união de todas as *features* geométricas (ou volume delta) é equivalente ao total de material removido do bloco para obter a peça finalizada. Alguns sistemas CAD, como Unigraphics, CATIA e I-DEAS, oferecem ambas as abordagens, enquanto outros (como Solidworks e Pro/E) oferecem apenas a abordagem de *features* [46].

Zeid [46] afirma que a abordagem de *features* é considerada por alguns como uma generalização da abordagem das primitivas. Sistemas que oferecem as *features* ocultam de seus usuários as operações Booleanas necessárias para criar o modelo sólido final.

Utilizando a descrição de Ji e Marefat [18], um esquema de representação para modelagem de sólidos é definido como uma transformação S que mapeia objetos físicos de um domínio M para um modelo espacial R, denotada $S : MR$. Em métodos de representação não ambíguos, cada representação no espaço de modelo corresponde a um objeto físico no domínio. Esquemas de representação únicos asseguram que cada objeto físico admite (pode ser mapeado para) apenas uma representação sintaticamente correta.

Ji e Marefat [18] citam seis principais técnicas (esquemas) usadas para representar e manter um modelo 3D por um sistema de modelagem CAD: instanciação de primitivas puras (PPI, do inglês *Pure Primitive Instancing*), enumeração de ocupação espacial (SOE, do inglês *Spatial Occupancy Enumeration*), decomposição de células (CD, do inglês *cell decomposition*), varredura (S, do inglês *Sweeping*), Representação por Fronteiras (B-Rep, do inglês *Boundary Representation*) e geometria sólida construtiva (CSG, do inglês *Construtive Solid Geometry*).

Nas próximas seções serão apresentados esses métodos de representação, destacando-se tanto seus aspectos de modelagem quanto de manipulação dos sólidos.

2.2.1 Instanciação de Primitivas Puras

A representação de objetos por instanciação de primitivas é uma das representações mais usadas devido a sua grande simplicidade e flexibilidade [18]. Esse tipo de representação tem por base a definição de objetos geométricos tridimensionais (primitivas) que possuem atributos (parâmetros), cujos valores são definidos pelo usuário no momento da criação de uma nova instância.

O conceito de parâmetro não está confinado às dimensões. Assim, a primitiva pirâmide, por exemplo, pode ter como parâmetro o número de faces laterais, além de parâmetros tipicamente geométricos como altura tamanho da sua base, ou um cilindro pelo diâmetro da base e também a sua altura.

Pode-se ainda estender o conceito de parâmetro para englobar propriedades dos objetos reais a modelar, tais como a definição dos materiais constituintes dos objetos e o acabamento das suas superfícies.

2.2.2 Enumeração de Ocupação Espacial

A representação por enumeração de ocupação espacial (SOE) subdivide o espaço 3D em volumes pequenos idênticos, chamados *voxels* (acrônimo para elementos de volume, do inglês *volume elements*) que juntos representam o volume ocupado pelo sólido.

Para representar um objeto, classifica-se esse volume como vazio ou contendo um sólido. Esta representação é uma variação de *octrees* em que o espaço 3D é subdividido recursivamente em octantes e classificado como cheio, vazio ou parcialmente cheio (Ji e Marefat [18] apud Juan-Arinyo e Sole [19] apresentam uma conversão de SOE para uma variante de octrees).

A SOE é única e não ambígua, mas tem a falha de ser potencialmente prolixa, devido a sua natureza enumerativa, ou seja, são necessários muitos elementos de volume, pois estes crescem em número segundo a lei cúbica. A forma dos voxels impede a correta representação dos objetos, não representando de forma exata as fronteiras existentes em planos oblíquos, já que não permite ocupação parcial de voxels, como pode ser visto na figura 2.3.

Lopes [25] e Mantÿlä [29] citam a representação de objetos por enumeração de ocupação

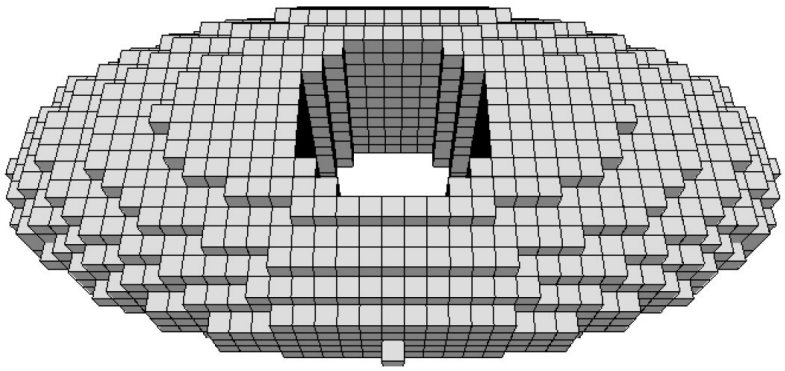


Figura 2.3: Exemplo de modelo de elipsóide com furo central representado por enumeração de ocupação espacial. Nota-se pouca precisão na descrição da fronteira. Adaptado de [25].

espacial como sendo muito usada na visualização de dados de caráter espacial, particularmente em aplicações biomédicas. Os dados obtidos por tomografia computadorizada ou por ressonância magnética (figura 2.4) representam propriedades dos volumes e não de pontos, devido à resolução espacial permitida pelos equipamentos de aquisição de dados médicos. Isto torna ideal o uso da representação por enumeração de ocupação espacial.

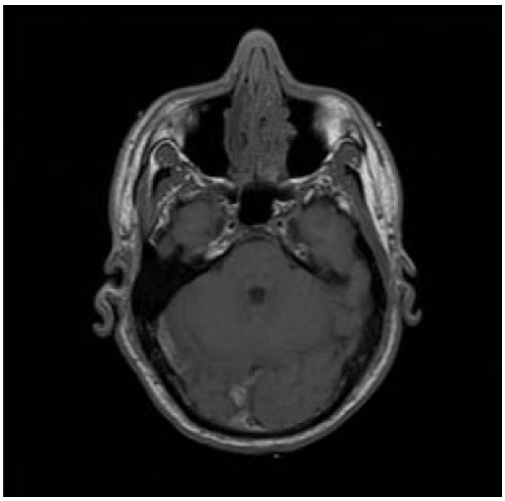


Figura 2.4: Imagem de ressonância magnética. Fonte [25]: imagem produzida a partir de dados do projeto *Visible Human Project*, U. S. National Library of Medicine, National Institutes of Health.

Mantÿlä [29] cita que em casos onde não se tem um objeto inicial, não se pode utilizar esta técnica de descrição para a modelagem de sólidos, e sugere utilizá-la através da conversão de

outra forma de representação. Segundo Mantÿlä [29], outros objetos podem então ser criados por operações Booleanas ou outros algoritmos disponíveis para enumerações exaustivas (sua nomenclatura para SOE). Além disso, Mantÿla diz ser possível a aplicação de técnicas de processamento de imagens para modelagem de sólidos, apontando uma operação deste tipo como útil para geração de dados para máquinas-ferramentas com controle numérico (CN) e em robótica.

2.2.3 Decomposição de Células

O método de decomposição de células possui como base a representação do objeto final através de elementos primitivos paramétricos. A decomposição do espaço proíbe expressamente a intersecção e impõe a justaposição das primitivas que partilham pontos, arestas ou faces. Esta é também a diferença essencial entre a decomposição do espaço em células e a instanciação de primitivas puras. Um exemplo de representação de um objeto através de decomposição de células é mostrado na figura 2.5.

A decomposição de células parece ser um recurso eficiente de geometria para cálculos de propriedade de massa e para alguns tipos de problemas de planejamento, mas, em algoritmos gerais para criação e combinação, tal representação não é disponível, e a validação não é trivial [4].

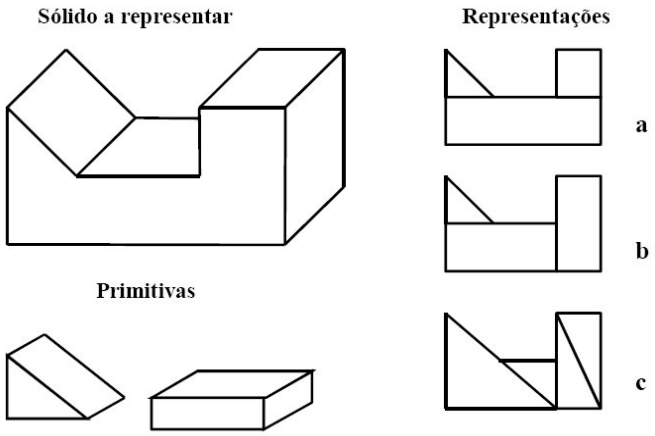


Figura 2.5: Representação de um sólido por decomposição em células, apresentando três soluções distintas (a, b e c) em que a solução c recorre a uma única primitiva.

2.2.4 Varredura

A representação de sólidos tridimensionais por varredura tem por base a descrição do volume gerada quando um objeto é deslocado segundo uma dada trajetória e forma um dado volume. O exemplo mais simples é o de um círculo deslocado segundo uma trajetória linear perpendicular ao círculo. O volume varrido resultante é um cilindro obtido por translação.

Um processo de fabricação em que um material plástico é obrigado a passar por um crivo com um orifício circular produz um resultado idêntico. O material toma a forma de um cilindro à saída do crivo e o volume descrito resulta de uma varredura por extrusão. Se o círculo considerado descrever uma trajetória circular, o volume varrido pelo círculo toma a forma de um *torus*, obtido por uma varredura por rotação. A figura 2.6 apresenta essas duas formas, além de ilustrar a dificuldade da representação por varredura em aplicar certas operações a objetos.

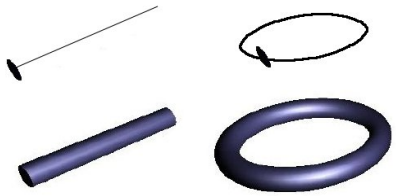


Figura 2.6: Círculo sobre trajetória linear e círculo sobre trajetória circular. A união de volumes descritos por varredura pode não ser descritível. Fonte [25].

Em qualquer caso, as varreduras produzem volumes com propriedades geométricas fáceis de calcular. Essa situação complica-se um pouco se as seções bidimensionais que varrem os volumes apresentarem formas irregulares.

Bohez et al. [3] apresentam um algoritmo para verificação de caminho de ferramenta para CNCs de 5 eixos por meio da utilização de um algoritmo plano de varredura. Eles apresentam ainda uma tabela, mostrada em 2.1, que contém uma comparação entre os algoritmos verificadores de caminho mais conhecidos.

A representação de varredura geral parece ser eficaz para alguns tipos de problema de definição de geometria e planejamento, mas pesquisas extensivas são necessárias para produzir a ambos uma base teórica matemática e procedimentos computacionais apropriados (figura 2.7).

Algorithm	Workpiece model blank	Tool model	Swept volume model	Cut workpiece model	Overcut undercut errors	View Dependent	Computing time
Surface normals	Surface Normals	3D polygons	3d Polygons	Cut Surface	Yes	Yes	Slow
Dexels	Depth element	Depth Element	Depth element	Image Frame Buffer	Yes	Yes	Slow
Z-buffer	Z-buffer of blank	Z-buffer of tool	Swept volume z-buffer	Image Frame Buffer	Yes	Yes	Fast
Solid Modeling CSG/B-rep	CSG tree/B-rep	CSG tree/B-rep	CSG tree/B-rep	CSG tree/B-rep	Yes	Yes	Very Slow
Graftree Octree	Volume elements (voxels) CSG tree/B-rep	Volume elements (voxels)	Volume elements (voxels) CSG tree/B-rep	Volume elements (voxels) CSG tree/B-rep	Yes	Yes	Very Slow
Stencil Buffer	Slices of 3D	Canonical equations	3D polygon slices	Image Frame	Yes	Yes	Very Slow
Sweep plane	polygon mesh	of 3D surfaces	3D polygon slices of tool sweep	Image Frame buffer	Yes	Yes	Slow

Tabela 2.1: Comparação de algoritmos para verificação de caminho de ferramenta com 5 eixos. Fonte [3].

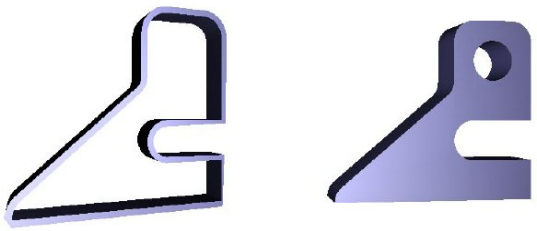


Figura 2.7: Representação por varredura do volume a ser removido pelo processo de corte por remoção de cavaco (à esquerda da figura) na fabricação de uma peça (à direita). Fonte [25].

Além disso, a técnica de varredura também é utilizada em conjunto com outros modelos de representação para explorar as melhores características de ambas, como em Shiroma et al. [37], onde é empregada em conjunto com um método de modelagem baseado em geometria sólida construtiva (CSG).

Zeid [46] considera a representação por varredura útil onde o seu domínio de modelagem pode ser estendido através de objetos $2\frac{1}{2}D$ e também ressalta que peças, como parafusos e componentes que necessitam de formas helicoidais e especiais, podem ser representados por varredura.

2.2.5 Representação por Fronteiras

A modelagem geométrica por fronteiras descreve os objetos por meio das superfícies que os limitam. As superfícies limitam os objetos mas não indicam que lado da superfície estes se encontram. Esta distinção é feita pela forma como cada elemento da superfície é definido.

Em geral, pode-se considerar a representação por limites (fronteira) em que os elementos das superfícies que a constituem podem assumir qualquer forma geométrica, mas isto significa passar os problemas derivados de formas complexas para cada um dos elementos constituintes da fronteira. Um exemplo de objeto representado por fronteiras pode ser visto na figura 2.8.

Dentre os vários tipos de representação de fronteira, Lopes [25] apresenta os seguintes:

- representação por poliedros.
- representação por arestas estendidas ou "aresta alada" (*winged-edge*).
- representações não poliédricas.

Representações por fronteiras são fontes eficientes de geometria para gráficos e, dada informação apropriada de dimensionamento e tolerância, para operações realizadas em máquinas NCs. Infelizmente, elas são difíceis de serem criadas e custosas para serem armazenadas, transmitidas e validadas [2].

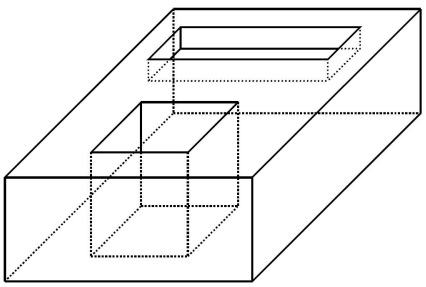


Figura 2.8: Exemplo de objeto representado por poliedro com um furo que o atravessa completamente e outro furo que não o atravessa. O poliedro tem um único componente e apresenta 36 arestas, 24 vértices, 15 faces (três apresentam furos) e um furo que atravessa completamente o poliedro [25].

2.2.6 Geometria Sólida Construtiva

A representação de objetos por geometria sólida construtiva (CSG) consiste em criar representações de objetos a partir de uma hierarquia de um conjunto de operações Booleanas (união, intersecção, diferença) aplicada a um conjunto de formas simples, chamadas primitivas. Os operandos das operações Booleanas podem ser tanto primitivas como os objetos resultantes das operações anteriores.

Internamente, um objeto CSG pode ser representado como uma árvore binária (árvore CSG), cujos nós folhas representam as primitivas sólidas e os nós internos (compostos) representam as operações Booleanas. Transformações podem ser armazenadas tanto nos nós folhas como nos nós internos [17].

Os nós internos representam os sólidos parciais resultantes da aplicação das operações Booleanas associadas a esses nós e aos dois subsólidos das árvores inferiores à esquerda e direita. O nó raiz representa o objeto composto resultante.

A aplicação de técnicas de CSG, especificamente a sistemas de CAD/CAM, já é bastante difundida. O emprego de CSG é definido como uma das duas principais técnicas de modelagem de sólidos por vários autores ([21, 22, 37, 46], entre outros).

A representação CSG parece ser eficiente para muitos problemas de projeto e planejamento (por exemplo, verificação *rough-machining* e de colisão), mas não são fontes eficientes de geometria para gráficos ou *finish machining* [4]. Representações CSG são fáceis de serem criadas, armazenar, e transmitir e são intrinsecamente válidas se os operadores combinacionais são gerais, ou seja, aplicáveis para todo objeto a ser representado. Esquemas tendo operadores não-gerais possuem características de validade similares aos esquemas de decomposição de célula.

Várias técnicas da aplicação de CSG têm sido criadas para suprir algumas das deficiências que o algoritmo apresentava em sua concepção (como é o exemplo de Salesin e Stolfi [35], Stewart et al. [40, 41], Cameron e Yap [6], Duff [9], Middleditch e Read [28], Buchele e Crawford [5]). Além disso, técnicas combinadas de CSG com outros métodos de representação como B-Rep (Benouamer e Michelucci [2]) e varredura (Shiroma et al. [37]), vêm sido estudadas e aplicadas para suprimir essas deficiências.

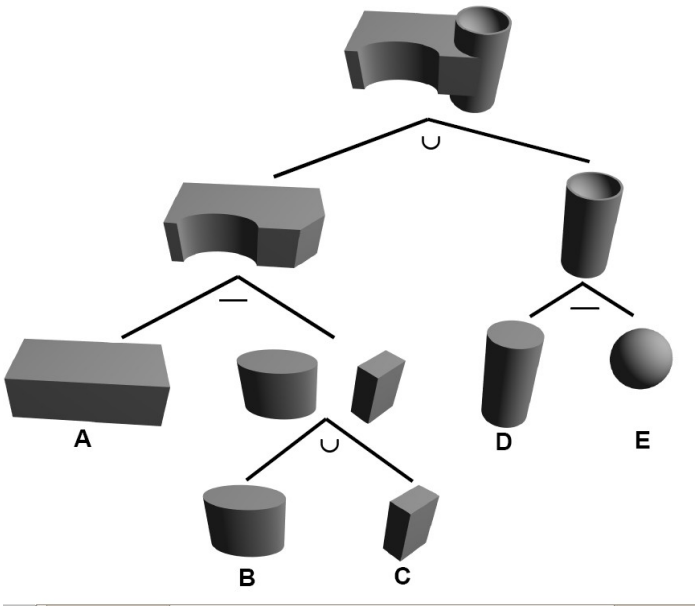


Figura 2.9: Construção de um objeto por CSG a partir de objetos primitivos sujeitos a operações Booleanas e transformações estruturadas hierarquicamente. Fonte [25].

Brown et al [4] realiza comparações entre alguns esquemas de representação geométrica (representação por fronteiras, geometria sólida construtiva e decomposição por células). Embora até então estivessem limitadas a essas técnicas, ele apresenta várias considerações importantes na relação entre os esquemas de representação geométrica e a sua aplicação para os sistemas de manufatura. Uma delas é que um sistema de modelagem geométrica para uso em indústrias mecânicas deve conter múltiplas representações de cada objeto, entretanto, essas representações podem apresentar problemas de projeto em sistemas não triviais para garantir a consistência entre as várias representações de cada objeto.

Outro aspecto importante a abordar são as ferramentas de desenvolvimento estudadas e utilizadas na implementação do protótipo.

2.3 Ambiente para Desenvolvimento do Protótipo

A escolha das ferramentas de programação foi determinada principalmente pela afinidade com o desenvolvimento do simulador e a possibilidade de implementar os métodos de representação descritos no capítulo 2. A seguir encontram-se as alternativas analisadas para a implementação

da ferramenta.

2.3.1 Java

A linguagem de programação Java, bastante difundida tanto no meio comercial como no meio científico, inclui recursos de orientação a objetos e não possui dependência quanto à plataforma. Ela é executada sobre uma máquina virtual, através do chamado *bytecode*, ao contrário das linguagens convencionais que necessitam ser compiladas para um código nativo. A linguagem também possibilita a execução de aplicativos diretamente sobre um navegador.

Devido à difusão da Internet, a linguagem Java apresenta vários aplicativos auxiliares para o desenvolvimento desse tipo de ferramenta e ainda um grande conjunto de material didático relacionado.

Algumas distribuições Java também oferecem a possibilidade de executar programas diretamente sobre o navegador, o que traz a possibilidade de abranger ainda mais as capacidades da utilização da ferramenta desenvolvida.

Pode-se antecipar que uma das deficiências apresentadas na linguagem Java é o seu baixo desempenho em determinadas situações, geralmente necessitando de grandes capacidades de processamento e memória. Esses aspectos deverão ser levados em consideração durante a escolha do melhor ambiente de desenvolvimento.

2.3.2 jMonkey Engine

jMonkey é um motor gráfico de alto desempenho que trabalha com as características de OpenGL associadas às mais modernas placas gráficas, sobre a linguagem de programação Java. O que chama atenção no jMonkey é o fato de ele possuir um grande conjunto de funcionalidades já implementadas, o que torna o trabalho de desenvolvimento bastante simplificado. Por exemplo, recursos de iluminação, controle de câmeras, detecção de colisão podem ser incorporados e utilizados pela chamada de poucos métodos.

Uma das características que torna o jMonkey tão interessante é o seu desempenho. Ele utiliza o chamado grafo de cena, que organiza as ligações entre os componentes da aplicação em uma árvore. Esta árvore é normalmente organizada de forma que os componentes não

utilizados na cena não necessitem ser processados, tornando a execução do aplicativo mais eficiente.

O jMonkey possibilita a escolha do sistema de renderização para sua execução, embora apenas o LWJGL (do inglês *LightWeight Java Game Library*) seja o único suportado atualmente. Além disso, ao explorar as habilidades do Java, o jMonkey pode ser executado como aplicação remota.

2.3.3 MATLAB/Octave

MATLAB [12, 26] é um software interativo para cálculo numérico, análise e visualização de dados que oferece auxílio na resolução de problemas para uma ampla variedade de aplicações, como processamento de sinais, biologia computacional, estatística, análise de imagens, entre outras.

O pacote foi concebido no final da década de 1970 por Cleve Moler, do Departamento de Ciência da Computação da Universidade do Novo México. Houve uma rápida disseminação do pacote em várias universidades, bem como grande aceitação pela comunidade de matemática aplicada. O engenheiro Jack Little reconheceu o potencial comercial do MATLAB e se uniu a Cleve Moler e Steve Bangert, que fundaram a empresa *The MathWorks* em 1984, onde o pacote continua em desenvolvimento.

MATLAB oferece um grande conjunto de funções que podem ser integradas para resolver problemas de alta complexidade. Os programas são expressos em forma matemática, ao contrário da programação tradicional. MATLAB provê um conjunto de pacotes de funções, conhecidos como *toolboxes*, para as mais diversas áreas de cálculo científico. Há também uma ferramenta para o desenvolvimento de interfaces gráficas ao usuário, reunindo em uma só ferramenta várias funcionalidades.

Octave [1] é uma linguagem de alto nível para cálculos de dados numéricos e oferece uma interface para a resolução de problemas lineares e não-lineares numericamente. O pacote utiliza uma linguagem geralmente compatível com MATLAB.

2.3.4 OpenGL

O OpenGL (do inglês *Open Graphics Library*) é uma API (do inglês *Application Programming Interface*) gráfica multilinguagem que permite a criação de aplicações utilizando gráficos computacionais 2D e 3D. O pacote possui aproximadamente 250 funções diferentes para desenhar cenas tridimensionais complexas. A OpenGL é bastante difundida na indústria dos jogos eletrônicos e compete diretamente com o pacote comercial DirectX da Microsoft. Além de sua utilização para jogos eletrônicos, o OpenGL é bastante utilizado em aplicações científicas, CAD, realidade virtual, possuindo várias ferramentas para cada uma dessas áreas.

Embora o OpenGL seja uma ferramenta com muitos recursos, sua utilização implicaria no desenvolvimento de vários algoritmos auxiliares para a construção da ferramenta, como detecção de colisão entre objetos e funções relativas à modelagem, os quais farão parte do Módulo de Simulação.

Para resolver esse problema, procurou-se ferramentas que auxiliassem o desenvolvimento em OpenGL. Foram encontradas algumas bibliotecas que trazem facilidades à implementação utilizando Open GL. São Elas:

- GLUT: do inglês *openGL Utility Toolkit*: é um conjunto de ferramentas responsável pelo gerenciamento de janelas e por fornecer uma API OpenGL para várias plataformas, além de fornecer suporte a vários dispositivos de interação, como mouse, teclado e *joyticks*;

- GLEW: do inglês *openGL Extension Wrangler*, é uma biblioteca que gerencia a inicialização de extensões OpenGL, tornando necessário apenas iniciar a biblioteca para que esta verifique a disponibilidade das extensões;

- GLUI: biblioteca dependente do GLUT, fornece controles mais refinados para OpenGL, e uma melhor aparência às janelas, além da facilidade na manipulação desses controles e janelas;

- OpenCSG: Biblioteca que implementa funções de Geometria Sólida Construtiva, tornando mais fácil a realização das operações Booleanas necessárias para a criação dos modelos de representação.

2.3.5 Visual Basic

Visual Basic é um ambiente de desenvolvimento baseado na linguagem Basic, concebido para programação no sistema operacional *Windows*. A linguagem é estruturada e possui ampla utilização em sistemas comerciais.

Alguns aplicativos utilizados como base no trabalho estavam disponíveis em Visual Basic, tal que esse ambiente foi tomado como referência no desenvolvimento do protótipo.

2.3.6 VTK

VTK (do inglês *Visualization Toolkit*) é um motor gráfico aberto e independente de plataforma com suporte à renderização paralela. Várias aplicações do VTK são conhecidas nas mais diversas áreas, inclusive para fins militares [15].

O VTK consiste em uma biblioteca de classes C++ e em várias camadas de interface interpretadas, incluindo Tcl/Tk, Java e Python. VTK suporta uma ampla variedade de algoritmos de visualização, incluindo métodos escalares, vetoriais, tensores, texturais e volumétricos, além de técnicas de modelagem avançadas como modelagem implícita, redução de polígonos, suavização de malhas, cortes, contornos e triangulação de Delaunay. O pacote possui vários algoritmos integrados, permitindo ao usuário combinar algoritmos de gráficos 3D, imagens 2D e dados. Possui projeto e implementação fortemente influenciado pelos princípios de orientação a objeto e suporte a várias plataformas como Unix, Windows, MacOSX.

CAPÍTULO 3

METODOLOGIA

Este capítulo descreve a metodologia utilizada no desenvolvimento de um sistema para auxiliar as atividades de usinagem de uma peça, destacando-se seus componentes, os aplicativos e o ambiente computacional.

O levantamento dos requisitos foi realizado por meio de reuniões, discussões técnicas e utilização da máquina-ferramenta. Reuniões freqüentes com usuários e especialistas da área permitiram a coleta de informações para o desenvolvimento do sistema, a definição das funcionalidades consideradas e a análise de aplicativos disponíveis no mercado que oferecem recursos semelhantes. A utilização da máquina-ferramenta permitiu uma melhor compreensão sobre o processo de desenvolvimento atual, como as interfaces de programação e utilização da norma ISO 6983 (códigos na linguagem G/M).

Além disso, as reuniões e discussões técnicas permitiram, de forma rápida, uma grande troca de conhecimentos diretamente relacionados ao problema, reduzindo dificuldades inerentes à interdisciplinaridade das áreas envolvidas. Detalhes específicos a respeito das *features*, restrições e propriedades geométricas e da norma ISO 14649 foram analisados e discutidos.

A metodologia proposta neste trabalho teve como objetivo desenvolver um sistema que agrega as novas funcionalidades que as *features* de manufatura podem trazer ao processo de desenvolvimento de peças de usinagem utilizando uma máquina-ferramenta atual. A ISO 14649 não foi abordada em sua totalidade, foi utilizada como base para o desenvolvimento da metodologia, apenas com o propósito de avaliar a possibilidade de sua utilização. O equipamento utilizado como base para o desenvolvimento e testes do protótipo foi o centro de usinagem ROMI Discovery 4022 (figura 3.1), uma máquina-ferramenta de 3 eixos com comando numérico MACH9.

Este capítulo está dividido da seguinte forma. Primeiramente será apresentada uma visão geral do protótipo desenvolvido. Os componentes principais desse sistema serão explicados

Figura 3.1: Painel de Comando da Máquina-ferramenta ROMI Discovery 4022. Fonte [30].

com maiores informações quanto a sua subdivisão, suas funcionalidades e sua relação interna. Depois serão abordadas as questões mais técnicas quanto à Representação Geométrica, com demonstrações de sua utilização. As primitivas e a sua composição para a formação das *features* serão apresentadas. Por fim é exemplificada a aplicação de uma *feature* simples a um bloco para demonstrar a seqüência de eventos no processo de modelagem e então uma *feature* mais complexa para demonstração da criação de uma árvore CSG.

3.1 Componentes da Ferramenta

Para uma maior compreensão do funcionamento do protótipo e de seu processo de desenvolvimento, em um nível alto de abstração, o sistema foi dividido em quatro entidades principais: os usuários, a norma ISO 14649, o sistema desenvolvido propriamente dito e a máquina-ferramenta (CNC). Eles estão descritos a seguir:

- Usuários: são os responsáveis pela entrada de dados e seleção de características do programa. Eles escolhem as *features* utilizadas, especificam valores aos atributos de cada uma delas e as posicionam em relação à peça. Os usuários também definem quais

ações devem ser executadas para a modelagem da peça.

- Norma ISO 14649: fornece informações geométricas e restrições das *features* utilizadas no sistema.

- CNC: recebe o código com instruções para a usinagem. O fato de possuir algumas particularidades, como o número de eixos, número máximo de ferramentas no *magazine*, o espaço de trabalho é considerado como entrada de dados, pois são informações relevantes ao processo de uma peça de manufatura.

- Sistema para Edição/Simulação/Transmissão: permite ao processista que estiver familiarizado com o desenvolvimento, modelar, simular e gerar o código para usinagem de uma peça, sendo uma interface entre os usuários, as *features* e a máquina-ferramenta.

Com base nessa divisão, o diagrama mostrado na figura 3.2 representa as entidades e a comunicação realizada entre elas, através das setas.

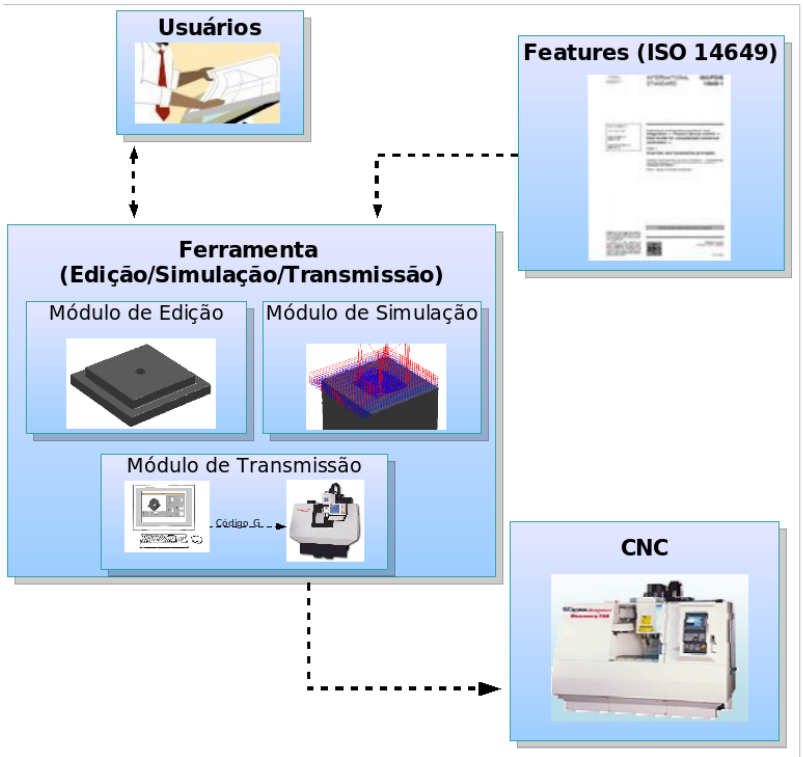


Figura 3.2: Componentes envolvidos no desenvolvimento do protótipo.

Com uma visão mais focada no protótipo, seus principais módulos são o de edição da peça, simulação da peça e transmissão do código para a máquina-ferramenta. O módulo de edição permite a inserção dos parâmetros geométricos relacionados com as *features*. Após a edição, o modelo pode ser visualizado e avaliado pelo usuário. Este modelo permite ainda a geração do código interpretado pela máquina-ferramenta. A validação do programa é facilitada com o auxílio de um simulador gráfico. Os recursos de simulação propostos baseiam-se na representação gráfica da trajetória da(s) ferramenta(s) de corte em um plano escolhido. Após a verificação do programa, o usuário pode transmiti-lo para o comando da máquina-ferramenta. Para a transferência de dados, adotou-se o protocolo RS-232C entre as portas seriais do computador e da máquina-ferramenta. A seguir encontra-se a descrição desses módulos.

O sistema para edição/simulação é constituído pelos módulos de edição e simulação, e é chamado assim porque o conjunto desses módulos representa o núcleo do protótipo desenvolvido e há uma forte dependência entre eles. Neste sistema ocorre a maior interatividade com o usuário e, assim, requer um maior detalhamento devido à extensão de suas funcionalidades. Uma representação da interface desses módulos pode ser observada na figura 3.3. O capítulo 4 apresenta uma descrição mais detalhada da interface desse módulos.

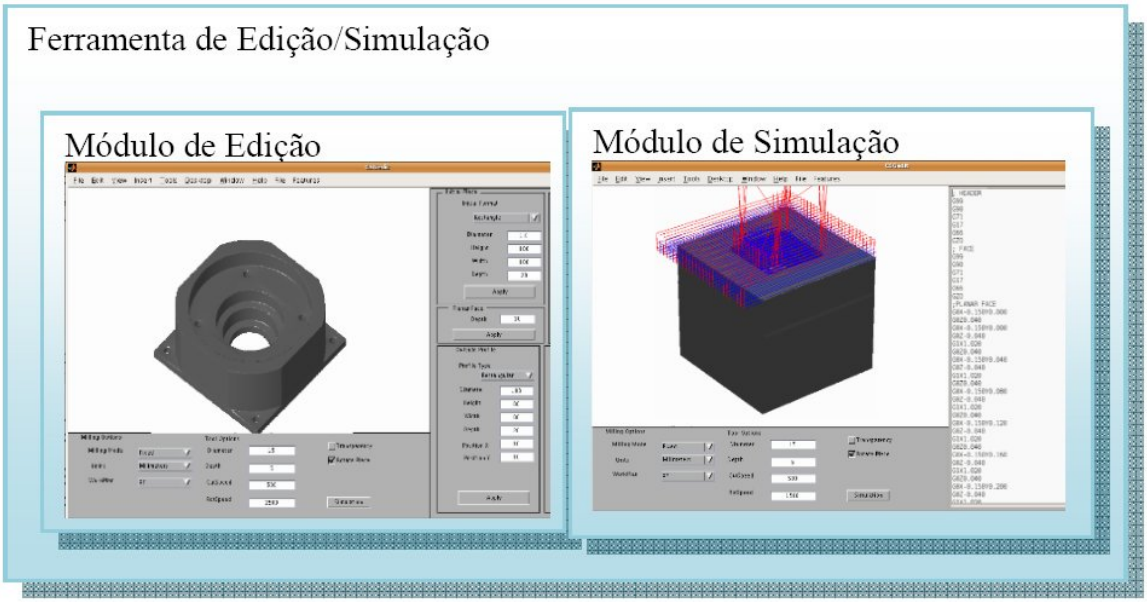


Figura 3.3: Representação das interfaces de Edição e Simulação.

3.1.1 Módulo de Edição

A interação inicial com o sistema é feita como o Módulo de Edição. Nesse módulo, o usuário tem como objetivo a modelagem da peça a partir da aplicação das *features* e dos parâmetros necessários para o projeto da peça de manufatura.

A figura 3.4 apresenta a arquitetura do Módulo de Edição. Esse módulo é composto de vários submódulos que, em conjunto, formam a interface para a modelagem da peça de manufatura.

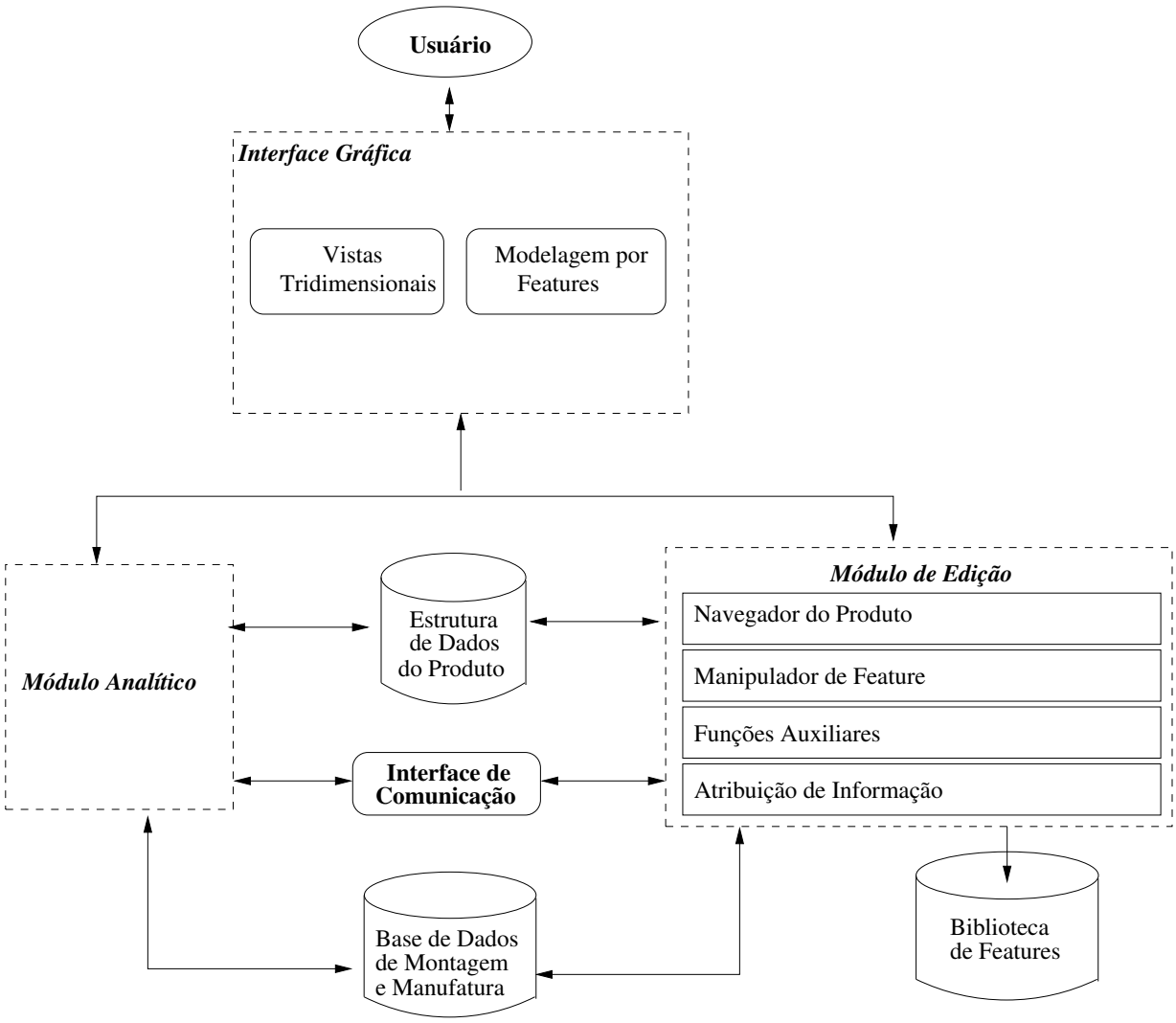


Figura 3.4: Arquitetura da interface de Edição. Adaptado de Maziero et al. [27].

O módulo de modelagem apresenta os seguintes componentes:

- Um submódulo, chamado Navegador do Produto, é utilizado para navegar sobre a es-

trutura de dados da peça. O navegador possui uma interface gráfica para permitir que o usuário visualize graficamente a estrutura da peça.

- Um submódulo chamado de Manipulador de *Feature* é responsável pela instanciação das *features*. O usuário pode selecionar dentre as *features* disponíveis e, então, fornecer os valores de seus atributos.

- Um submódulo de Funções Auxiliares inclui um conjunto de funções de suporte, como o mapeamento da estrutura da peça em uma estrutura gráfica.

- Um submódulo Atribuição de Informação é responsável por atribuir tolerâncias e definir a finalização de superfície a determinadas partes, baseado no tipo de ajuste desejado entre as *features* que constituem as peças. Esta função é realizada de acordo com a verificação das restrições estabelecidas pela norma, que definem a Base de Dados de Manufatura.

O Módulo Analítico realiza diferentes tipos de análise, que incluem: a) dimensionamento; b) análise de tolerâncias; c) identificação de acabamento de superfície. A descrição de um modelo analítico pode ser encontrado em Maziero et al. [27].

A Interface de Comunicação é responsável pela comunicação entre os Módulos Analíticos e de Modelagem, transferindo os valores com as devidas proporções e limites da estrutura gráfica. É importante não confundir a Interface de Comunicação com o Módulo de Transmissão, que será discutido mais tarde.

O usuário, ao desejar criar uma nova peça, deve inicialmente definir o tipo e as dimensões do bloco inicial. O bloco inicial deve representar o mais próximo possível as características da peça bruta a ser usinada. Após a definição dessas dimensões, o bloco bruto é demonstrado graficamente.

Depois de definir o bloco inicial, o usuário escolhe qual *feature* de manufatura ele deseja incorporar ao bloco. Ao escolher a *feature* desejada, ele deve atribuir valores aos atributos da *feature*, e então definir o posicionamento da *feature* sobre o bloco, onde as estruturas geométricas serão mostradas ao usuário.

Ao terminar a modelagem da peça de manufatura, o processista pode realizar a simulação da usinagem da peça. Para isto, a estrutura de dados da peça modelada é transferida ao Módulo de Simulação, gerando uma nova representação, baseada na trajetória da ferramenta de corte sobre o bloco inicial, possuindo mais informações com relação a manufatura da peça, e fornecendo assim, um nível de detalhe superior ao anterior.

3.1.2 Módulo de Simulação

Assim como o Módulo de Edição, o Módulo de Simulação possui uma estrutura bastante diferenciada. A figura 3.5 apresenta a arquitetura do Módulo de Simulação.

A interface gráfica possui dois componentes principais: a simulação tridimensional da peça a ser usinada, com a trajetória da ferramenta de corte sobre o modelo tridimensional gerado no módulo de edição e características tais como precisão escolhida da ferramenta e a área reservada ao código G gerado e utilizado na simulação.

Em relação à figura 3.5, a Biblioteca de *Features* é única, compartilhada com o Módulo de Edição. Pode-se também verificar que alguns submódulos possuem a mesma nomenclatura entre os módulos de Edição e Simulação, mas estes apresentam diferenciações em suas funcionalidades, a citar, os submódulos analítico, a base de manufatura, a estrutura do produto e a interface de comunicação. As diferenças entre esses submódulos, bem como os submódulos exclusivos do Módulo de Simulação, estão descritas a seguir:

- submódulo Navegador do Produto, que utiliza uma estrutura de dados diferente, mesmo possuindo a mesma finalidade do Módulo de Edição.
- submódulo Interpretador de Superfície faz a conversão da estrutura de dados gerada pelo Módulo de Edição, a Base de Dados de Manufatura e o conjunto de *features*.
- submódulo Gerador de Código G faz a análise da estrutura de dados da Peça através da Base de Dados de Manufatura em conjunto com o Módulo Analítico para gerar um código específico da máquina-ferramenta utilizada.
- submódulo Funções Auxiliares inclui funções de suporte tanto à estrutura gráfica da simulação quanto à geração do código G.

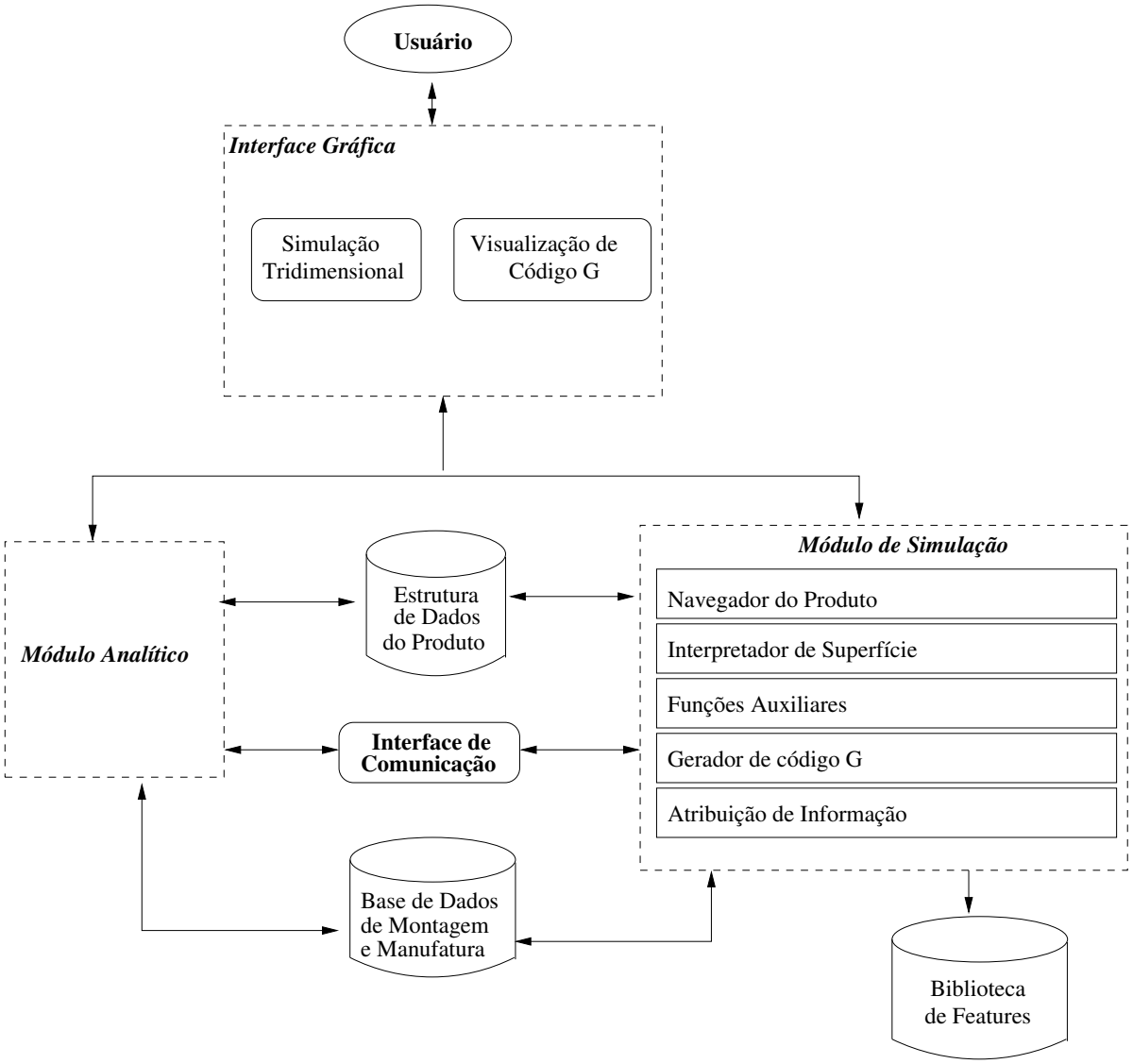


Figura 3.5: Representação da arquitetura da interface de Simulação. Adaptado de Maziero et al. [27].

O Módulo Analítico é responsável por análises direcionadas à Simulação, envolvendo as propriedades da máquina-ferramenta, do modelo gerado e das *features*. A Base de Dados de Manufatura contém as especificações e tolerância no que diz respeito às técnicas de manufatura. O método de representação utilizado na interface de simulação foi escolhido em razão da natureza do problema e ao grande nível de detalhe gerado na tarefa de simulação.

A simulação ocorre em conjunto com a geração de código G. Este processo se inicia quando o usuário define os valores ou as dimensões do bloco inicial da peça de trabalho. Então, o modulo de simulação define seus limites de trabalho, assim como os deslocamentos seguros

em relação à peça de trabalho e também as especificações referentes à ferramenta de corte (como diâmetro, profundidade, diâmetro máximo de corte, profundidade máxima de corte, velocidade de avanço, velocidade de posicionamento, etc) e aspectos de usinagem como plano de trabalho, utilização de fluido de corte, modo de trabalho (absoluto e incremental), entre outros.

No Módulo de Simulação, a versão usinada da peça é apresentada. A geometria da ferramenta e aspectos como precisão são levados em consideração, assim os detalhes da usinagem poderão ser exibidos e modificados, antes da execução da peça na máquina-ferramenta.

Após a entrada de valores e confirmação da inclusão de uma *feature*, o número de passos necessários para concluir a operação é calculado, ou seja, a quantidade de movimentos que a ferramenta de corte necessita para remover todo o volume referente à *feature* escolhida. Esses passos são calculados com base nas dimensões da peça de trabalho, nas dimensões das ferramentas de trabalho (os chamados diâmetro efetivo, profundidade efetiva, etc), nas dimensões da *feature* especificada e parâmetros de segurança, como plano de retração, utilizado para auxiliar na remoção de cavaco, entre outros. Cada ponto é então calculado com base na especificação da *feature* e no valor dos atributos selecionados pelo usuário.

Para diminuir a probabilidade de erros, em cada ponto calculado para a máquina, um ponto é atribuído à simulação gráfica do traçado da ferramenta. Nos casos onde uma função determina uma seqüência de pontos com características comuns, também há uma função que gera os pontos correspondentes para a simulação de traçado.

As estratégias de desbaste foram escolhidas pela sua simplicidade, visando sempre à segurança da máquina e da peça de trabalho, procurando sempre posicionar a ferramenta de corte fora da peça, antes de iniciar qualquer movimento de corte. Alguns poucos movimentos são realizados em velocidade de posicionamento dentro da peça, como por exemplo os movimentos de retração para a usinagem de furos. Mas estes movimentos são sempre executados em direção a um eixo livre.

A partir desses fatores é gerado o código G correspondente ao comando de corte/posicionamento necessário. Depois de gerado o código G, este mesmo ponto é inserido no conjunto de simulação de traçado, também levando em conta o tipo de operação realizado. Por razões de

segurança, após a execução de toda *feature*, o código para o retorno da ferramenta de corte à posição de troca (ou à posição chamada G0Z0O, como atribuido pelo controle MACH 9) é incluído.

A interface do Módulo de Simulação permite ao usuário analisar a peça em diferentes ângulos de observação para verificar sua corretude e o detalhamento da usinagem, além de verificar o código de usinagem gerado para a máquina-ferramenta.

3.1.3 Módulo de Transmissão

A transmissão do código G para a máquina-ferramenta é uma característica importante para a conclusão do projeto da peça de manufatura. Embora seja possível a digitação do código diretamente no painel da máquina-ferramenta, isto se torna inviável devido ao tempo necessário para inserir a quantidade de informação gerada, visto também a sub-utilização de equipamento disponível (neste caso, o computador).

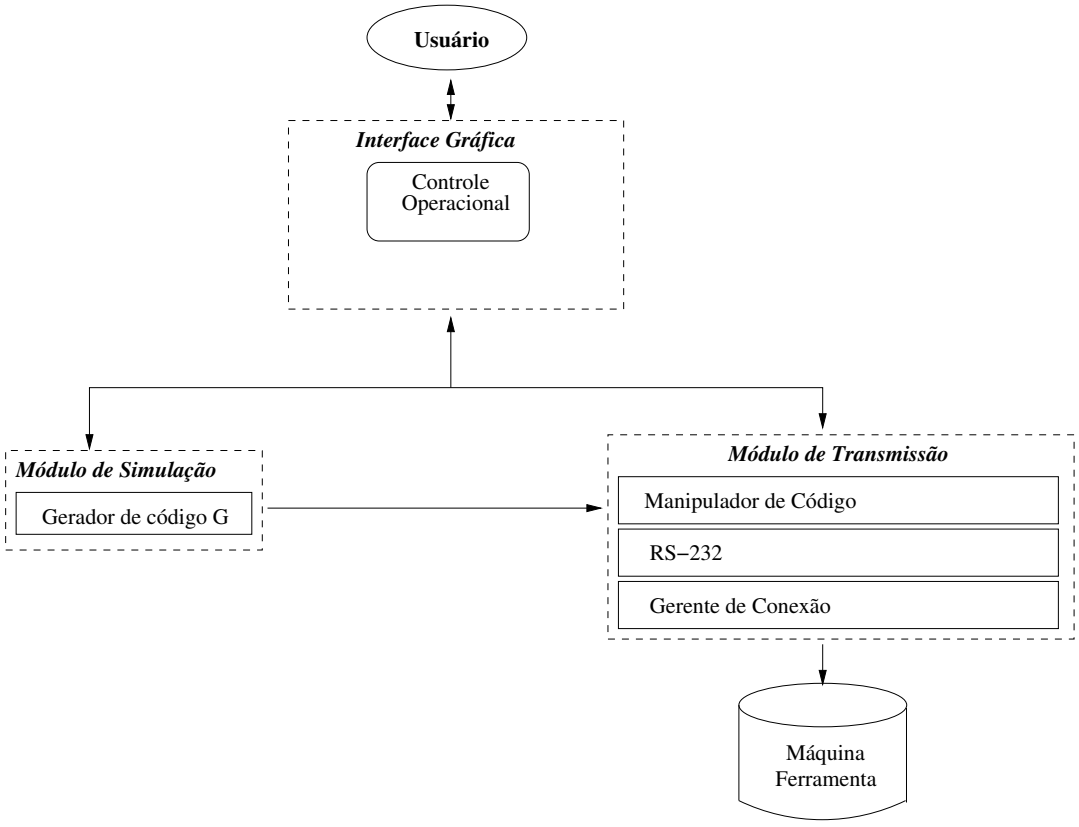


Figura 3.6: Representação da arquitetura da interface de Transmissão. Baseado em Maziero et al. [27].

Uma alternativa é a utilização de programas já existentes que realizam esta tarefa. Entretanto, é interessante que uma ferramenta consiga englobar todos esses processos. Visto isso, o módulo de transmissão tem como finalidade enviar para a máquina-ferramenta o código gerado para a usinagem da peça desejada.

Como visto na seção 3.1.2, o código G para a manufatura é gerado para cada *feature* independentemente. Como o número de funções ou arquivos que a máquina-ferramenta pode receber é limitado e não há como prever a quantidade de *features* utilizada na usinagem de uma peça, todo o processo de usinagem é enviado em apenas um arquivo para a máquina-ferramenta.

Diferente das etapas de Edição e Simulação, a Transmissão de dados apresenta uma arquitetura relativamente simples, como pode ser visto na figura 3.6, apresentando os seguintes componentes:

- submódulo Manipulador de Código controla o posicionamento dentro do código G para o envio do mesmo à máquina-ferramenta.
- submódulo RS-232 contém as funções de envio e recebimento do protocolo RS-232.
- submódulo Gerente de Conexão estabelece uma conexão entre o sistema e a máquina-ferramenta para o envio dos dados.

A transmissão dos dados entre o computador e a máquina-ferramenta ocorre de maneira simples. Como a máquina utilizada nos testes não possui um protocolo próprio para a transferência de dados e não oferece um tratamento detalhado de retorno, foi utilizado o protocolo RS-232C para realizar a entrega do programa.

Inicialmente, a máquina-ferramenta deve ser colocada em modo de recepção por meio da porta serial que possui. O usuário então deve escolher a opção de efetuar conexão com a máquina. O módulo de conexão tenta iniciar a comunicação entre a máquina-ferramenta e o computador e, assim que estabelecida, envia os dados gerados para a máquina-ferramenta. A velocidade da conexão foi estipulada em 19200 Kbps (Kilobytes por segundo), pois a quantidade de informação a ser enviada é relativamente pequena e esta velocidade oferece uma segurança relativa à entrega dos dados.

3.2 Representação Geométrica

Para representar a modelagem de objetos foi utilizada a técnica de enumeração de ocupação espacial, onde o espaço é subdivido em pequenos volumes de tamanho fixo (os chamados *voxels*), em conjunto com as operações de geometria sólida construtiva (CSG). A representação geométrica é realizada da seguinte forma. Primeiramente estabelece-se um valor para a resolução, que implicará no número de subdivisões do espaço e, conseqüentemente, na precisão da visualização. Esse número deve estar entre o intervalo aberto de extremos 0 e 1.

Embora esse intervalo esteja definido matematicamente, é importante ressaltar que a escolha desse valor deve ser realizada de forma lógica à precisão que se deseja encontrar na modelagem e que valores superiores a 0.1 são impraticáveis devido à baixa qualidade de representação que oferecem. Quanto maior proximidade do zero, maior a quantidade de *voxels* gerada, tal como a precisão do objeto modelado.

Para demonstrar, considera-se um espaço bidimensional, com o valor de 0.1 de resolução, e o espaço de trabalho entre 0 e 1 (incluindo os mesmos). Sabe-se que esse espaço será dividido em 11 partes iguais, como pode ser observado na figura 3.7 (devido à inclusão dos limites). Pelo método de enumeração de ocupação espacial, esse valor será o número de subdivisões do espaço de trabalho em questão. Uma matriz é criada com essas dimensões, em que um determinado valor (por exemplo, 0) implicará na não ocupação de um espaço, enquanto um outro (por exemplo, 1) representará a ocupação do mesmo. Considera-se a matriz 3.1 como a representação desses valores em conjunto.

$$O = \begin{bmatrix} 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{3.1}$$

Considerando um círculo de raio r igual a 0.5, posicionado sobre o centro do objeto (x_c, y_c),

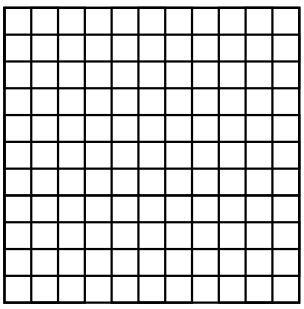


Figura 3.7: Figura representando o espaço de trabalho com 11 subdivisões pela técnica de Enumeração da Ocupação Espacial.

o cálculo de suas dimensões é dado pela equação 3.2.

$$r^2 = (x - x_c)^2 + (y - y_c)^2 \tag{3.2}$$

Como demonstração, para os valores de raio e centro apresentados, a equação teria a forma da equação 3.3 e sua representação segundo o método adotado é dada na matriz MR (matriz 3.4).

$$0.5^2 = (x - 0.5)^2 + (y - 0.5)^2 \tag{3.3}$$

$$MR = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.4}$$

Dessa forma, o objeto resultante é ilustrado na figura 3.8. A diferença entre o objeto real e o obtido com a resolução especificada pode ser vista pelo círculo sobrescrito à matriz. Essa diferença é a principal limitação da técnica de SOE, devido aos limites impostos pela resolução.

A implementação do sistema estende essas características ao espaço tridimensional. O espaço inicial onde a peça de trabalho será modelada possui suas dimensões normalizadas.

A criação do espaço inicial somente é dada quando da criação do objeto inicial ou bloco

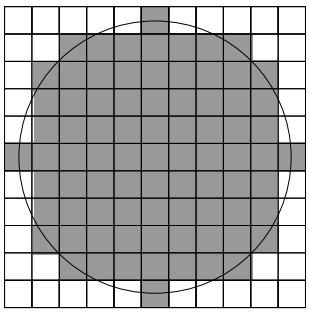


Figura 3.8: Figura representando um círculo arbitrário em um espaço com 11 subdivisões pela técnica de Enumeração da Ocupação Espacial.

inicial. Ao definir esse objeto, ele é posicionado internamente ao espaço de trabalho, normalizando seus valores para que trabalhem dentro dos limites deste espaço, tomando-se o maior valor como referência, para que o modelo ocupe o maior volume possível, sem sofrer danos.

O objeto inicial pode possuir três formas principais: o cubo, o paralelepípedo ou o cilindro. Essas formas são utilizadas durante todo o processo de modelagem. Todas as *features* são compostas pela aplicação de operações de CSG a uma ou mais dessas formas.

3.2.1 Cubo

O cubo é um objeto sólido tridimensional limitado por seis faces quadradas que se encontram em um determinado ângulo correto. Também pode ser chamado de poliedro uniforme U_6 e modelo Wenninger W_3 [45]. O sólido cubo é gerado a partir de sua coordenada central em conjunto com o raio da esfera inscrita a ele. Pela representação adotada, para cada dimensão (eixo), todo espaço que possui coordenadas iguais ou inferiores ao raio é atribuído o valor verdadeiro e para cada valor fora dessas coordenadas é atribuído o valor falso.

3.2.2 Paralelepípedo

O sólido retângular ou paralelepípedo é criado de forma semelhante ao sólido cubo, exceto que cada dimensão possui um valor diferente de comparação. Esses valores são largura, altura e profundidade. Como o objeto é descrito com origem em seu centro, suas coordenadas devem ser divididas pela metade.

3.2.3 Cilindro

O cilindro é gerado através da criação de círculos paralelos entre si ao longo de um determinado eixo, sendo perpendiculares a este eixo. Assim como os outros sólidos, o cilindro é gerado a partir de seu centro. Caso uma unidade de volume (*voxel*) seja interna ou concordante ao círculo, esta será parte do sólido resultante.

O objeto cilindro é gerado por toda a extensão do espaço de trabalho (ao longo do eixo escolhido). Isso requer uma adequação do cilindro a sua profundidade desejada, o que é realizado por meio da aplicação de técnicas de CSG.

A partir da geração do modelo desse bloco inicial, um conjunto de operações CSG é efetuado sobre o modelo. Como mencionado no capítulo 2, as operações utilizadas são união (ou adição), interseção e subtração. Essas operações são executadas sucessivamente de acordo com a especificação das *features* escolhidas para modelagem da peça de usinagem. O conjunto de funções utilizadas é descrito com mais detalhes no capítulo 4.

3.2.4 União

A união de dois conjuntos A e B é um conjunto de todos os elementos que pertencem a A ou a B ou a ambos [23]. Segundo a abordagem utilizada, representa o menor conjunto de *voxels* (elementos) com todos os valores obtidos de dois objetos de volume. Como a representação utilizada define valores apenas para os componentes do objeto, a aplicação da função mínimo retorna a união dos objetos.

3.2.5 Interseção

A interseção de dois conjuntos A e B é o conjunto dos elementos que são comuns a A e a B, isto é, os elementos que pertencem a A e que também pertencem ao conjunto B (ou todos os elementos de B que também pertencem ao conjunto A) [8, 23]. A interseção representa o maior conjunto com os elementos pertencentes aos dois sólidos simultaneamente.

3.2.6 Subtração

A subtração refere-se à aplicação do operador de diferença ou complemento. A subtração entre os conjuntos A e B corresponde ao conjunto de todos os elementos de A que não pertencem a B [8, 23, 39].

3.3 Aplicação de Features a um Bloco

Essa seção tem como objetivo demonstrar a metodologia utilizada sob a interface. A aplicação de uma *feature* a um bloco depende de alguns fatores referentes à peça inicial (suas dimensões e sua forma), assim como características referentes às *features* aplicadas anteriormente.

A *feature* furo (*hole*) pode ser vista como um dos exemplos mais simples de *feature*. Ela é composta basicamente pela remoção de material no formato cilíndrico a partir de um ponto no plano superior, seguindo a direção negativa do eixo Z até a profundidade desejada. Foram implementados dois dos tipos principais de furos, o furo cego (*blind hole*), com profundidade especificada pelo usuário a partir da interface e o furo passante (*through hole*), onde a profundidade do furo é dada pelos limites da peça de trabalho.

Considerando a aplicação de um furo passante em um bloco inicial com a forma ilustrada na figura 3.9, o processista deve informar os valores dos atributos designados para a *feature* furo. Como o furo é do tipo passante, não há a necessidade de especificar o valor de profundidade para a *feature*, sendo apenas necessário estabelecer o seu valor de diâmetro e o posicionamento sobre o bloco de trabalho.

Ao aplicar a *feature*, seus atributos são computados, depois é criado um modelo positivo do volume do sólido a ser retirado do bloco. Esse volume é criado a partir das primitivas e das operações definidas na seção 3.2. No caso do furo, esse volume positivo é representado por um cilindro, sendo apenas necessário instanciar a primitiva cilindro diâmetro e profundidade especificados. Como o furo utilizado é passante, a profundidade é automaticamente definida para ultrapassar a face inferior do modelo da peça de trabalho. Uma ilustração do cilindro utilizado na operação de subtração é dada na figura 3.10.

O próximo passo da modelagem da aplicação da *feature* furo sobre uma peça é a remoção

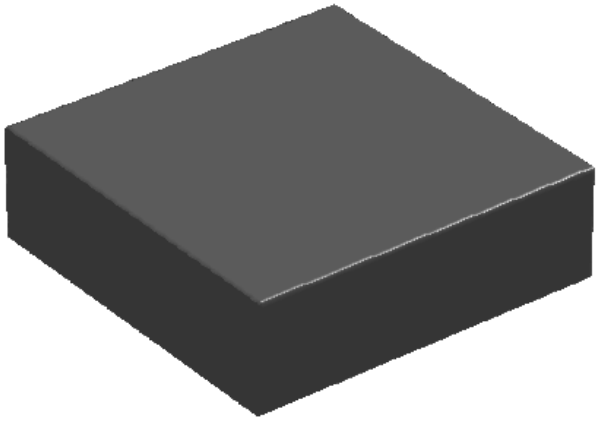


Figura 3.9: Um bloco inicial de formato retângular de dimensões arbitrárias.

Figura 3.10: Exemplo do volume a ser removido para a *feature* furo passante posicionada sobre o centro do objeto da figura 3.9.

do volume da peça de trabalho na posição escolhida. Essa remoção é realizada pelo emprego das técnicas de CSG, tal que a peça de trabalho e o cilindro são as primitivas e a operação é a subtração. A figura 3.11 demonstra a árvore gerada após a aplicação da *feature* furo passante sobre um paralelepípedo, bem como o objeto resultante dessa aplicação.

Para as aplicações subseqüentes de *features*, o modelo gerado é armazenado para que os cálculos referentes às operações geométricas não sejam refeitos. A árvore é mantida pela aplicação sucessiva de *features* à peça de trabalho.

O exemplo a seguir ilustra a criação de uma *feature* mais complexa, chamada de *pocket*. A *feature pocket*, utilizada neste trabalho, é especificada pelos atributos altura, largura, profundidade, raio ortogonal e posição X e Y (considerando o plano XY). Como a *feature pocket* possui os cantos arredondados (mesmo que seja apenas pelo raio da ferramenta de corte), essa posição em X e Y é referente à projeção de suas bordas laterais.

Após a especificação desses parâmetros pelo usuário e outros atributos referentes à *feature*,

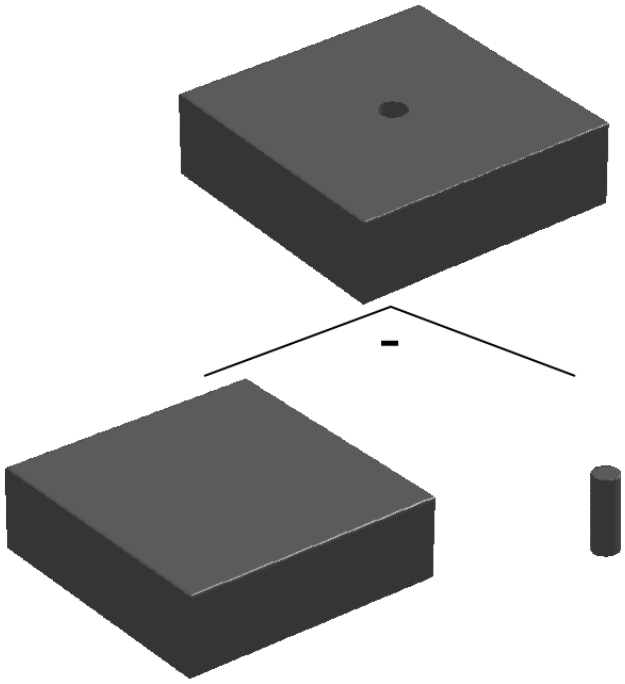


Figura 3.11: Uma árvore CSG para a aplicação da *feature* furo sobre um paralelepipedo.

como *Tipo de Pocket*, os pontos de posicionamentos dos componentes do *pocket* na interface são calculados. Após isso, cria-se um modelo positivo do *pocket* (figura 3.12). Como o *pocket* representa um quantidade de volume a ser retirada da peça de trabalho, o modelo referente a esse volume é construído a partir das informações fornecidas pelo usuário. O modelo do pocket é formado basicamente por seis estruturas principais, sendo quatro cilindros referentes ao cantos do *pocket* e dois paralelepípedos referentes às suas laterais.

Figura 3.12: Exemplo da estrutura principal da *Feature Pocket*.

Como mencionado anteriormente, o modelo positivo é criado pela união desses quatro cilindros, mais os dois paralelepípedos, em um único objeto, que depois será posicionado

corretamente sobre a peça de trabalho e então removido por meio de operações CSG. De fato, essas duas etapas (a criação do modelo positivo e a sua remoção) são realizadas pelo emprego de técnicas de CSG e, juntamente com o restante da modelagem e aplicação de *features*, produzem a chamada *árvore CSG*. O diagrama da figura 3.13 ilustra a criação do modelo por meio de suas partes, assim como a sua remoção, resultando a peça após a aplicação da *feature*, ou seja, um paralelepípedo com uma lacuna central no formato de um retângulo de cantos arredondados.

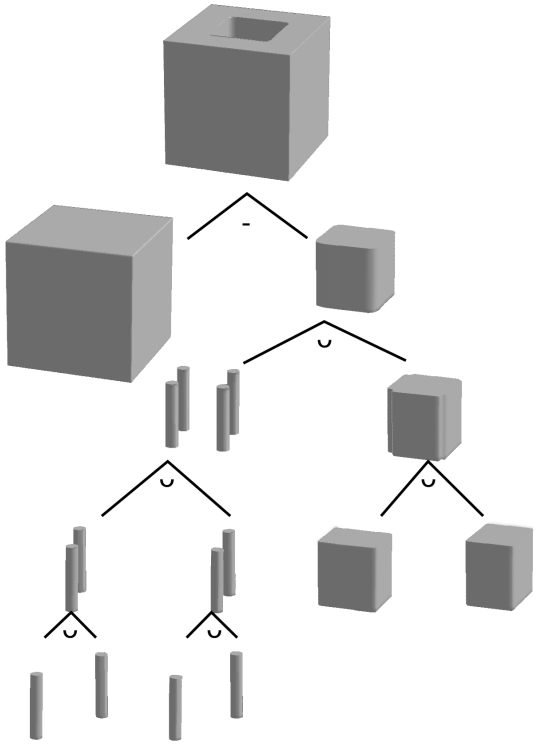


Figura 3.13: Uma árvore CSG para a aplicação da *Feature Pocket* sobre um paralelepípedo.

Simultaneamente à modelagem desse sólido, os valores atribuídos tanto às dimensões e formato inicial da peça de trabalho quanto às *features* aplicadas a essa peça, são armazenados para serem utilizados posteriormente pela simulação da trajetória da ferramenta de corte e geração do código G correspondente.

CAPÍTULO 4

RESULTADOS OBTIDOS

Este capítulo tem como objetivo a demonstração da aplicação da metodologia descrita no capítulo 3. A implementação das *features* é relatada através dos códigos das primitivas e operações descritas nos capítulos 2 e 3. A usinagem de uma peça encontra-se ilustrada por meio de exemplos, tanto pela abordagem convencional quanto pela metodologia proposta.

4.1 Processo de Usinagem

O processo de usinagem de uma peça inicia-se com sua especificação. Como dito anteriormente no capítulo 2, a aplicação da ISO 6983 ainda é utilizada de forma manual para a programação de máquinas CN. As seções seguintes descrevem o processo convencional para programação de máquinas-ferramentas e por meio da metodologia proposta.

4.1.1 Processo Convencional

Primeiramente o processista necessita fazer um desenho da peça ou um esboço dele. Este desenho pode ser produzido manualmente ou através de uma ferramenta CAD. A figura 4.1 ilustra o desenho de uma peça a ser usinada, com suas cotas.

O código para a usinagem da peça deve ser então escrito, digitando-se diretamente no CNC ou transferindo-o a partir de um arquivo no formato ASCII. O programa deve iniciar com um conjunto de instruções para a máquina, como escolha de ferramenta, corretor de ferramenta, compensação do raio de corte, entre outros, incluídas no cabeçalho do programa. Nos exemplos a seguir, as informações após o símbolo ";" são explicativas, considerados comentários para o programa.

```
;Inicio do Cabeçalho
G99; Cancela G92
G90; Modo Absoluto
;G91 para Modo Incremental
G71; Dimensões em mm
```

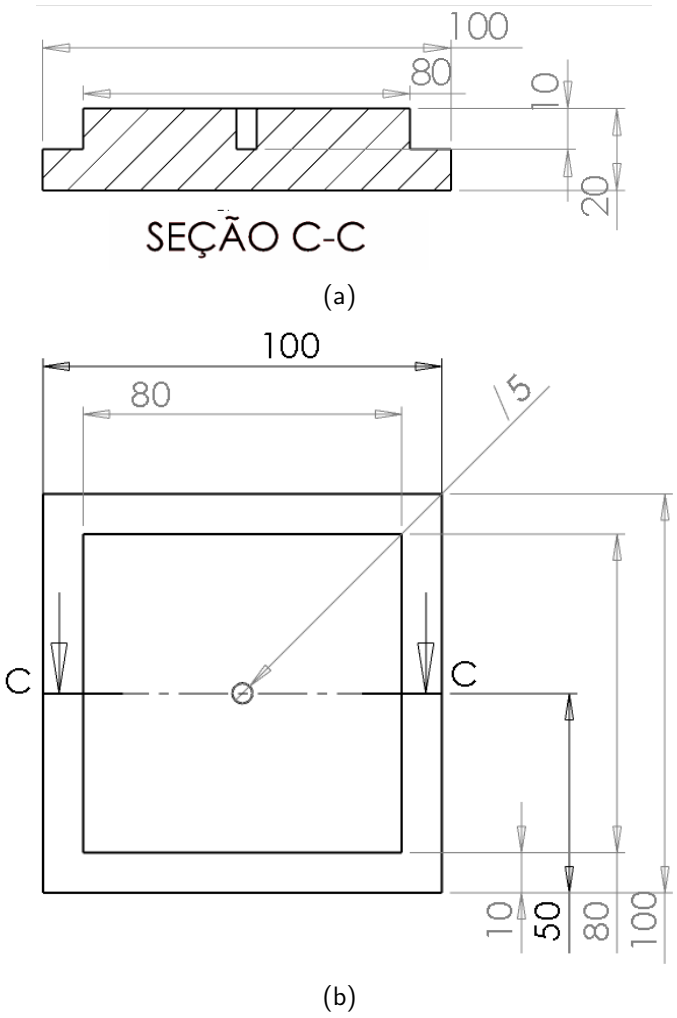


Figura 4.1: Desenho cotado representando uma peça a ser usinada. (a) seção transversal (C-C) e (b) vista superior da peça.

```
;G70 para dimensões em polegada
G17; Plano de trabalho XY
;G18 para plano XZ
;G19 para plano YZ
G66; limpa a tela de simulação do CNC
GZO; Ou G0Z000 move o eixo para a origem
;Fim do Cabeçalho
```

O processista deve, então, desenvolver o código de máquina a partir da geometria das partes da peça, demonstrando o percurso da ferramenta de corte sobre o bloco a ser usinado com o uso dos códigos da linguagem G/M.

Muitos cuidados devem ser tomados no processo de escrita do programa para que, com a execução do código, não ocorram problemas como choque da ferramenta de corte com a peça, com a mesa de trabalho ou que movimentos de avanço e posicionamento sejam executados

em locais incorretos. O processista precisa conhecer o diâmetro das ferramentas de corte que possui e saber qual delas está utilizando no momento.

Dependendo do material utilizado, uma estratégia de desbaste deve ser aplicada antes dos movimentos de acabamento. O código abaixo considera que não houve essa necessidade e que os movimentos são executados diretamente na fase de acabamento.

O trecho a seguir exemplifica o código gerado para o rebaixo do retângulo central da ferramenta. Para evitar a repetição excessiva de código, foi considerada uma ferramenta de corte fictícia que executa o rebaixo em apenas uma passada.

```
;Inicio do desbaste
T1M6;        Seleciona a ferramenta 1 e executa o comando de troca
O1S2000M3;   Estabelece a velocidade de rotação e liga rotação
             à direita para ferramenta 1
G41;         Liga compensação de raio a esquerda
G0X-15.Y10.; Move a ferramenta para fora da peça.
G0Z-10.;     abaixa a ferramenta
G1X110.F2500;     movimento na direção x
G1Y110.;     movimento na direção y
G1X-10.;     movimento na direção x
G1Y-15.;     movimento na direção y
G0Z2.;       movimento na direção z
;Fim do trecho de desbaste
```

Para a execução do furo central, pode ser necessário trocar a ferramenta de corte. Na máquina-ferramenta utilizada, há comandos (ciclos fixos) para execução de furos que seriam úteis caso fosse maior o número de furos ou se o material usinado necessitasse de controle de cavaco. Aqui será utilizada a maneira mais simples, utilizando apenas uma broca com diâmetro ideal (5 mm) e sem utilizar ciclos fixos do comando.

```
G40;         Descompensação de raio
G0X10.Y10.; movimento em vazio
GZO;         Mover ferramenta atual para origem
M5;          parar a rotação da ferramenta
T2M6;        Seleciona a ferramenta 2 e troca de ferramenta
O2S2000M3;   Estabelece a velocidade de rotação e liga
             rotação a direita para ferramenta 2
G0X50.Y50.; posiciona a ferramenta sobre o centro da peça
             (em velocidade de posicionamento)
G0Z3.;       desce a ferramenta até 3 mm acima da peça
G1Z-10.F2500;    executa o furo
G0Z2.;       sobre a ferramenta
GZO;         volta para a origem da máquina
M5;          para a rotação da ferramenta
M2;          finaliza o programa
```

Embora o processo pareça simples devido ao comprimento do código gerado, o processista deve prever quaisquer erros que possam ocorrer durante o processo de planejamento, pois o

custo para realizar uma nova usinagem é geralmente elevado. Caso ele execute a usinagem e a precisão desejada não seja atendida, o processista deverá escrever outro programa com as devidas modificações. Também deve-se considerar que cada mudança dessas implica, dependendo do caso, no desenvolvimento de vários cálculos que, se não automatizados, consomem ainda mais tempo do processo de criação.

4.1.2 Processo com a Metodologia Proposta

Com a utilização do protótipo desenvolvido, o processista deverá trabalhar de forma diferente da convencional. Ele poderá ou não iniciar o projeto por meio de um desenho. Caso ele possua o desenho, este servirá de base ou roteiro para a modelagem da peça que deseja. Caso não possua, ele deverá ter em mente as dimensões da peça que deseja fabricar.

Modelagem da Peça a Partir das Features

Ao iniciar o protótipo, o usuário terá acesso ao Módulo de Edição (semelhante ao mostrado na figura 4.2). Ele deverá possuir uma área de trabalho e um conjunto de funções para modelagem.

O primeiro passo para a modelar uma peça é selecionar a criação de um novo projeto. Ao fazê-lo, o usuário é questionado sobre as dimensões e formato do bloco inicial. Ao atribuir esse valor, no exemplo, $100 \times 100 \times 30$ mm, o usuário terá um sólido com a forma semelhante à mostrada na figura 4.3. Além disso, o usuário deve especificar as dimensões de uma ferramenta de corte a ser utilizada. Essas serão as dimensões utilizadas para a simulação e geração do traçado de ferramenta.

O próximo passo para a criação da peça é a seleção das *features* que a descrevem. O usuário poderá escolher dentre um conjunto de *features* qual delas melhor se aplica. A modelagem por *feature* deve ser realizada por etapas, sempre uma a uma.

Primeiramente, deve-se nivelar o topo do objeto, para retirar as possíveis imperfeições na face superior e garantir que ela esteja plana. Geralmente são extraídos apenas um ou alguns milímetros, mas para efeito visual, considera-se a extração de 10 milímetros de material. O

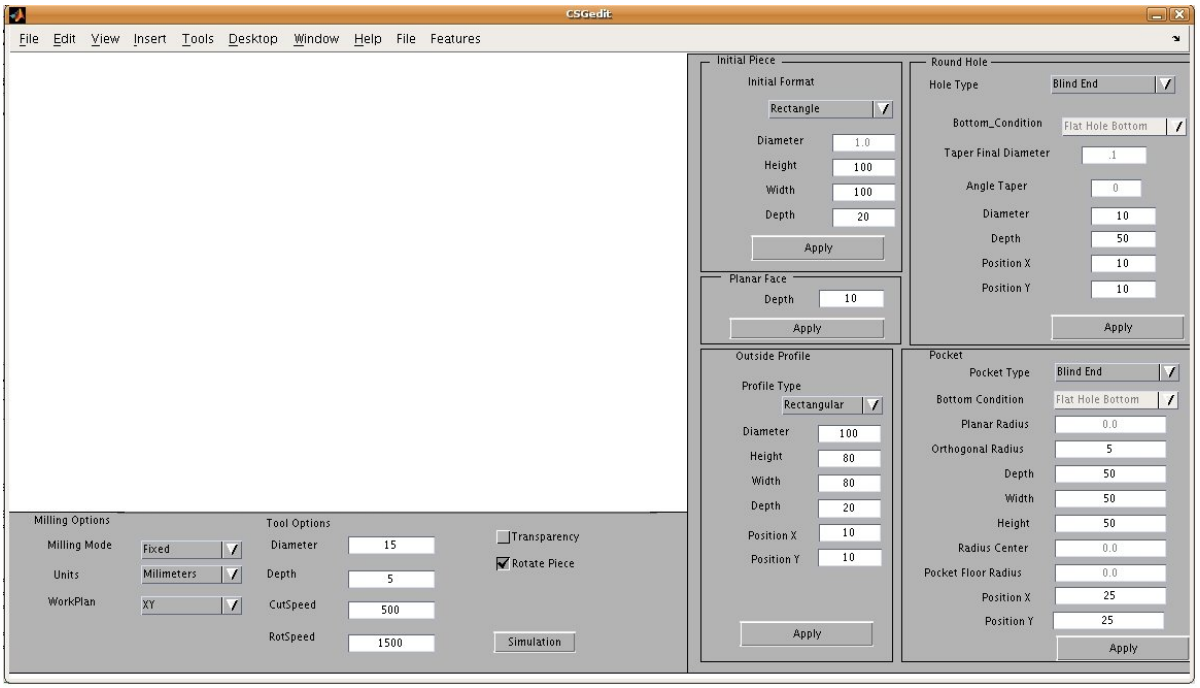


Figura 4.2: Esboço da interface do Módulo de Edição.

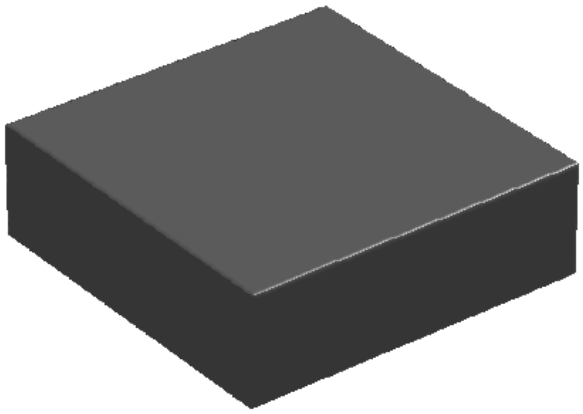


Figura 4.3: Representação do bloco inicial para a modelagem da peça.

usuário deve inserir este valor no campo *Depth* da *feature Planar Face* e aplicar a *feature*, tendo como resultado visual a figura 4.4.

Para usinar o quadrado central mostrado na figura 4.1, o usuário pode escolher a *feature General Outside Profile* que remove material ao redor de uma forma específica. A interface da ferramenta apresentará as opções para o preenchimento dos atributos, neste caso, *Depth* e *Shape Profile*, que definem, respectivamente, a profundidade e a forma da *feature*. O usuário deve então escolher a posição da *feature* sobre a peça. A figura 4.4 ilustra a aplicação da

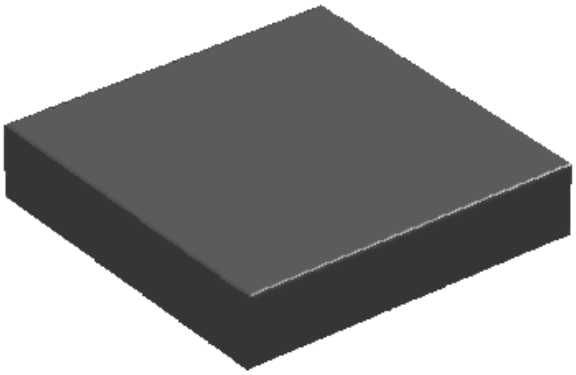


Figura 4.4: Representação da peça após a aplicação da *feature Planar Face*.

feature.

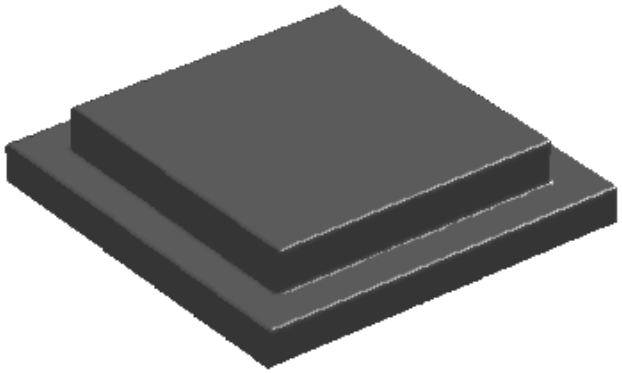


Figura 4.5: Representação da modelagem da peça após execução da *feature General Outside Profile*.

Finalmente, é executado o furo sobre a peça. Para tal, o usuário deverá escolher a *feature Round Hole*, no menu de *features*. Após escolher a *feature*, deve-se atribuir os valores para diâmetro e tipo de fundo (*diameter* e *bottom_condition*). Neste caso, o diâmetro é de 5 mm e a condição do fundo é *blind bottom condition*. Depois, deve-se posicionar o furo na região desejada. A figura 4.6 demonstra a modelagem da peça após a aplicação do furo.

Assim como visto anteriormente, essas aplicações sucessivas são modeladas através de operações de geometria sólida construtiva. A representação da árvore de geração da peça descrita pode ser vista na figura 4.7.

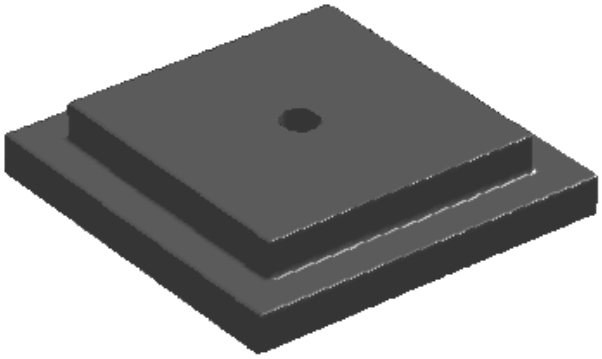


Figura 4.6: Representação da modelagem da peça após aplicar a *feature Round Hole*.

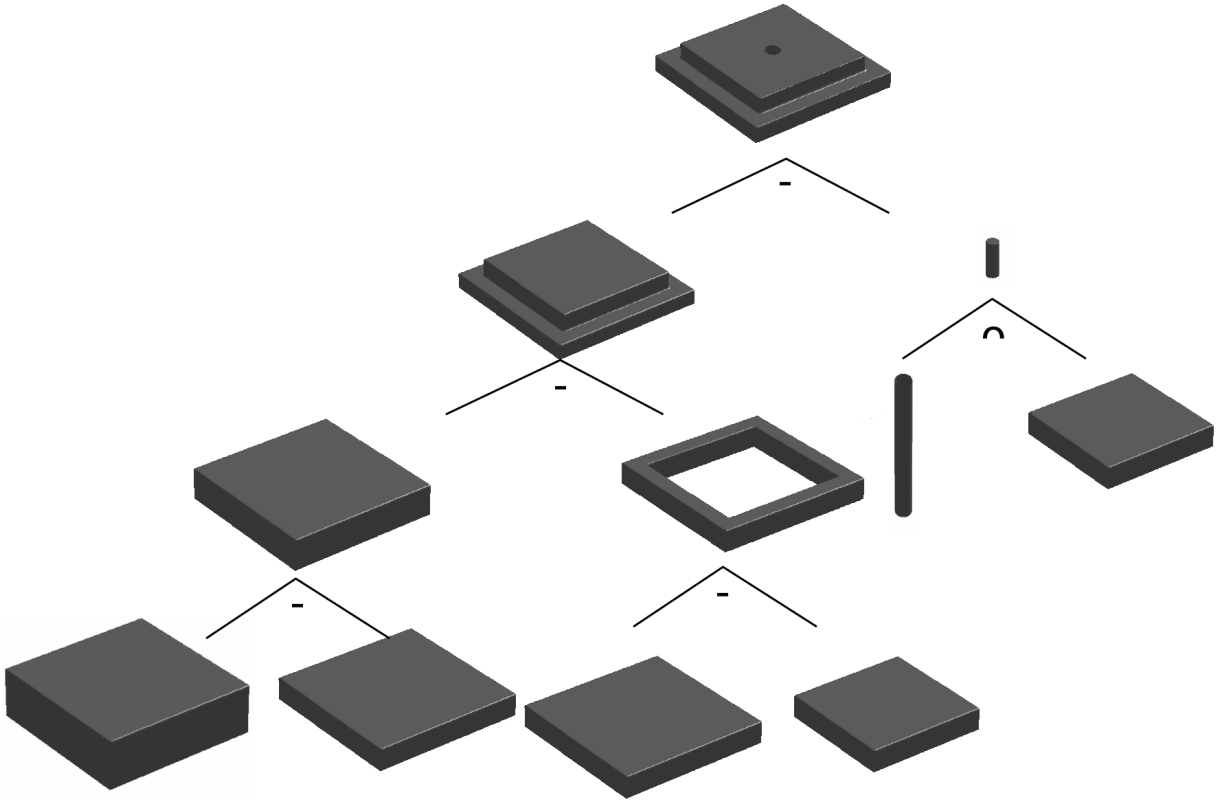


Figura 4.7: Árvore CSG da modelagem de uma peça com um *Rectangular Outside Profile* e um *Hole* central.

Simulação do Modelo e Transmissão de Código

O usuário pode visualizar a simulação da usinagem selecionando a opção de simulação. O Módulo de Simulação é apresentado ao usuário (como ilustrado na figura 4.8) e irá gerar uma representação tridimensional da peça usinada, e o percurso da ferramenta de corte para gerar

a peça desejada sobre ela.

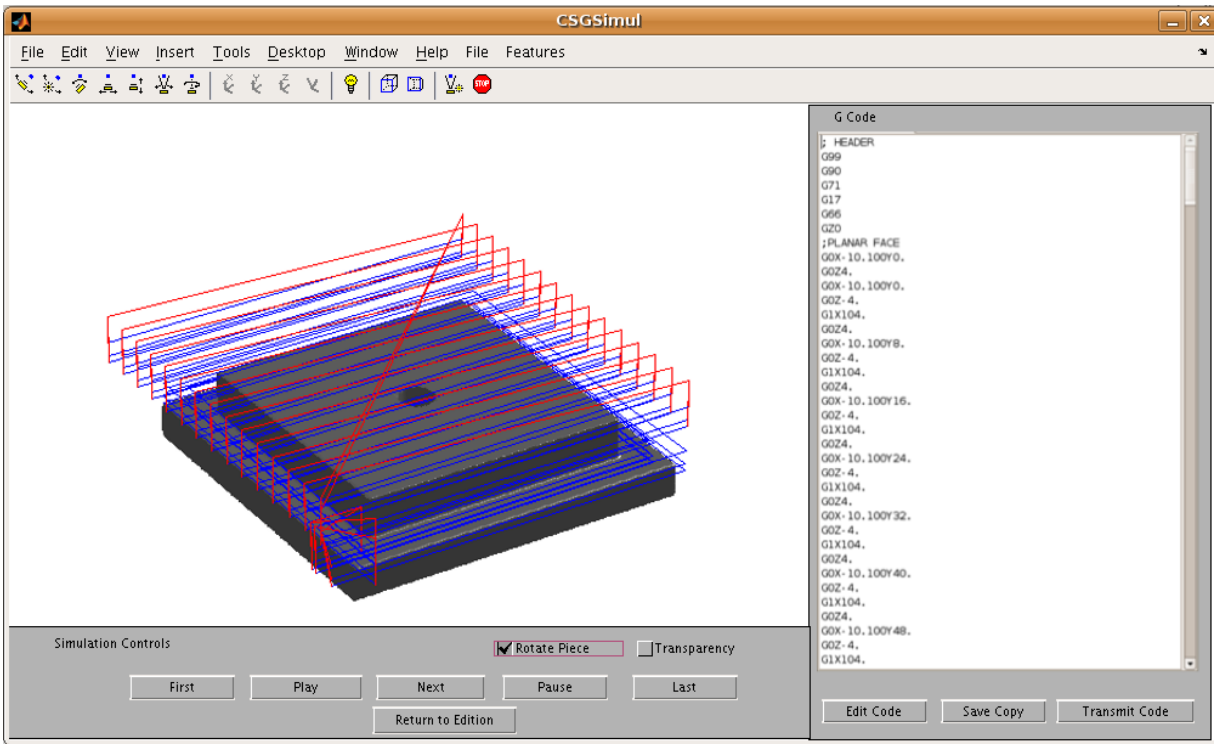


Figura 4.8: Interface do Módulo de Simulação.

Inicialmente, o usuário pode observar linhas representando todo o trajeto da ferramenta de corte sobre o modelo gerado no Módulo de Edição. Para facilitar o entendimento, devido à complexidade da visualização, o usuário pode navegar pela simulação e seguir a ordem na qual as instruções das *features* foram geradas. Essa ordem é mostrada de acordo com o tipo de movimento executado, se movimento de posicionamento ou movimento de avanço, representados por cores distintas.

As imagens da figura 4.9 demonstram alguns passos da execução da *feature Planar Face* através dos controles apresentados na interface de simulação. Primeiramente se tem o modelo gerado pelo Módulo de Edição e então os caminhos da ferramenta de corte são apresentados sucessivamente até o termino da simulação.

Para as *features* internas, a interface permite a manipulação da peça através de comandos de rotação e aproximação, demonstrados na figura 4.10, além de possuir o recurso de transparência (figura 4.11) para que a parte interna das *features* seja observada.

Dessa forma, o usuário pode analisar cuidadosamente a peça, seu código e a usinagem pro-

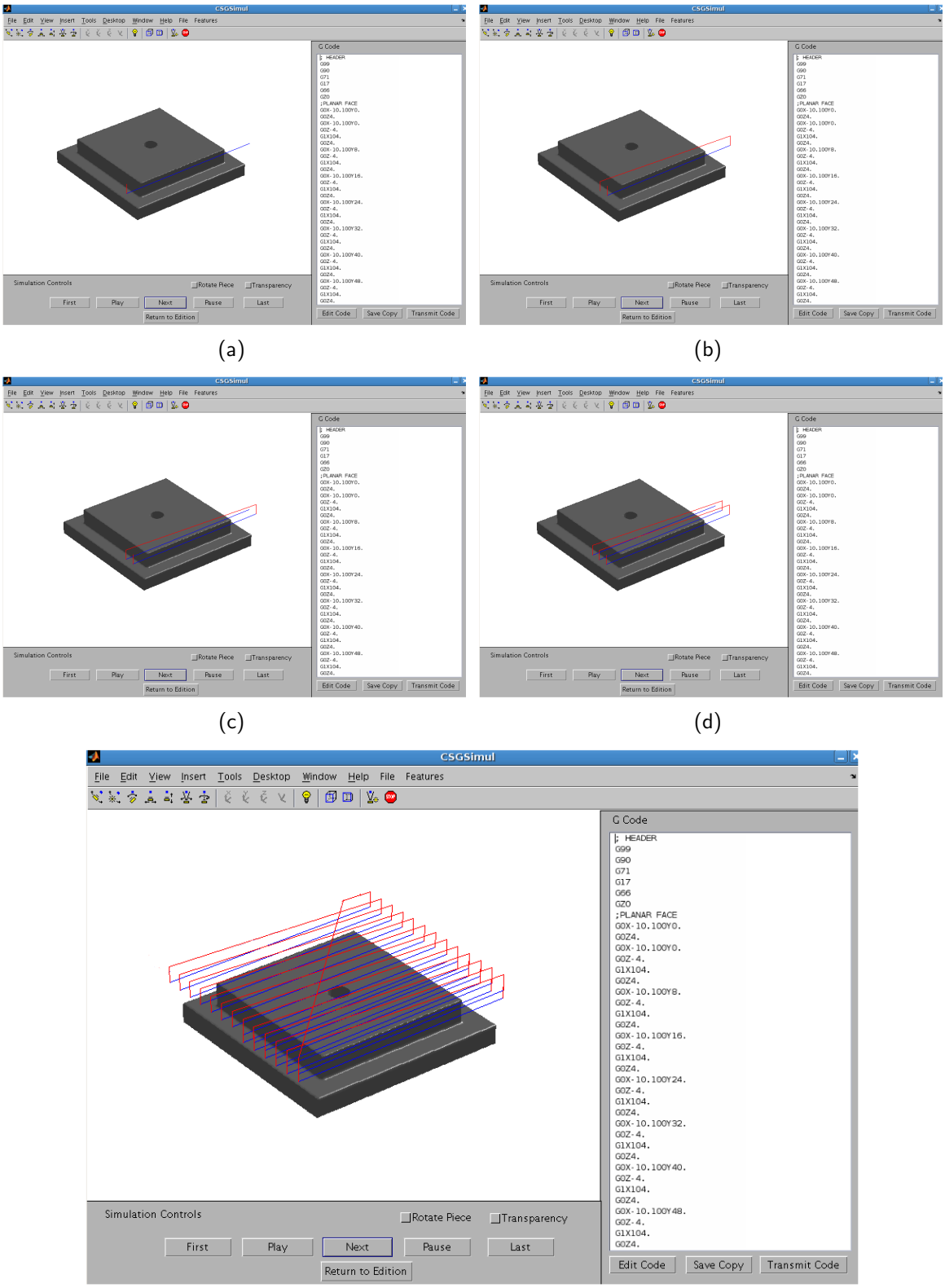


Figura 4.9: Representação da Simulação de trajetória para parte da *feature Planar Face*.

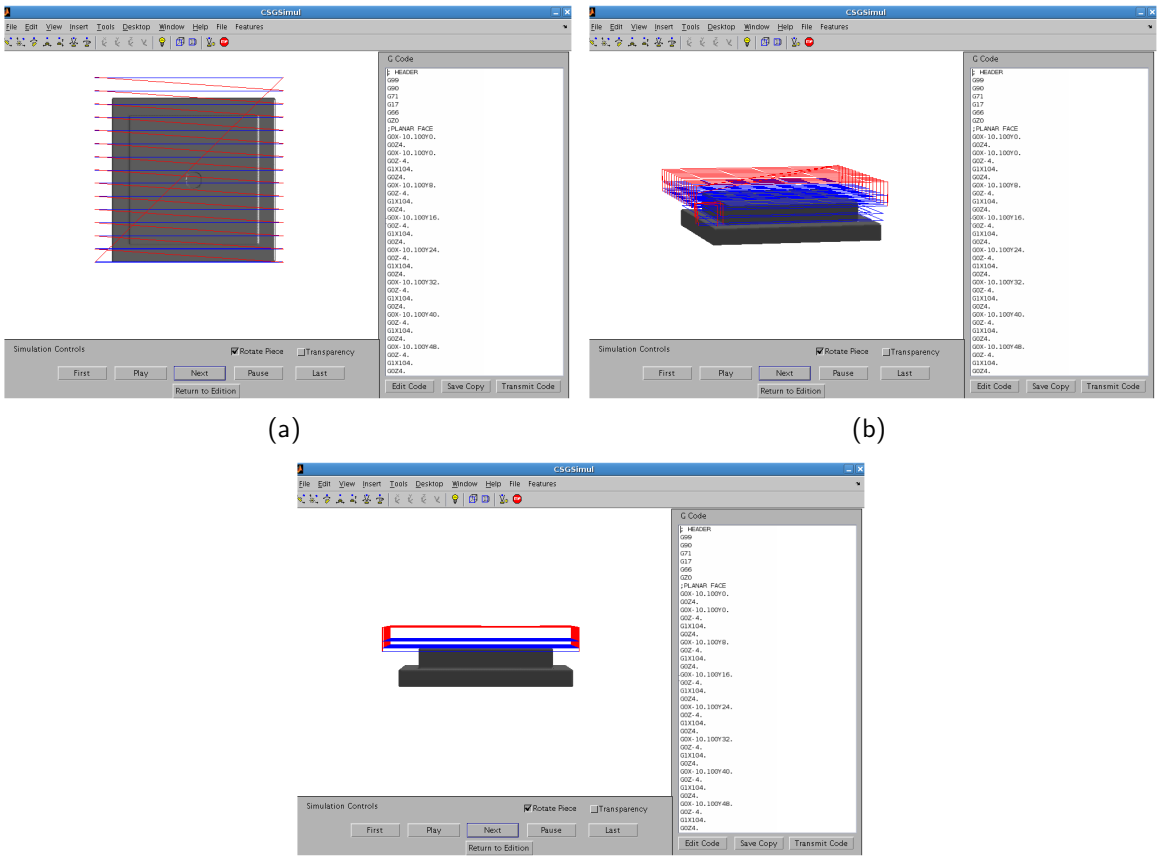


Figura 4.10: Utilização dos controles de navegação da Interface de simulação.

posta, permitindo retornar à modelagem para realizar as correções necessárias. O usuário pode alternar a utilização do Módulo de Edição com o Módulo de Simulação até que a peça esteja de acordo com o esperado. Após o processo de simulação, o código apresentado ao usuário pode ser enviado à máquina-ferramenta por meio do módulo de transmissão (figura 4.12).

Geração de uma Peça Complexa

Para melhor demonstrar a utilização das *features* implementadas, considera-se uma peça mais complexa ilustrada na figura 4.13. Essa peça é composta por onze furos, sendo nove deles passantes, dois com fundo cego e três deles furos fresados. Além disso, ela possui um faceamento e um *General Outside Profile*.

Assim como no exemplo anterior, é necessário definir as dimensões da peça inicial de trabalho. Pode-se observar pela cotas que a peça possui profundidade de 80 mm e um espaço

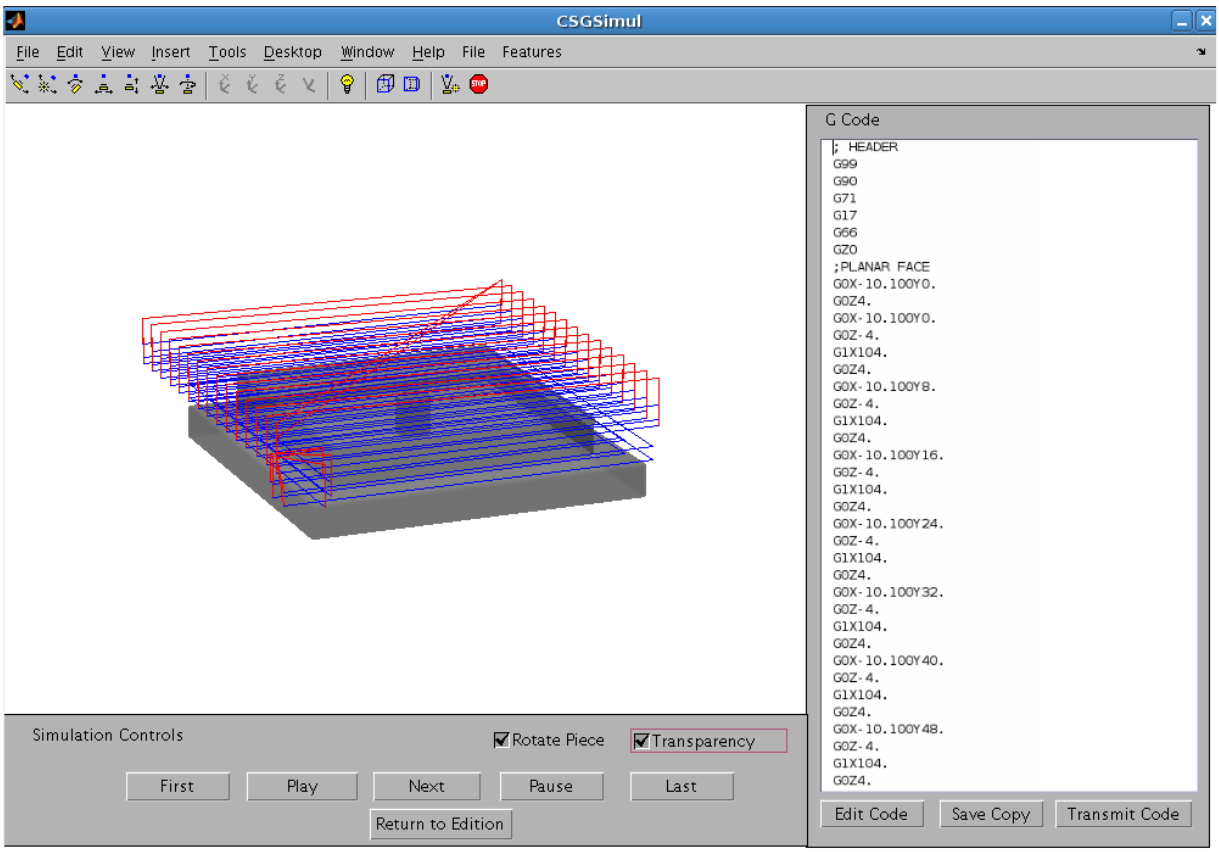


Figura 4.11: Utilização de transparência no Módulo de Simulação.

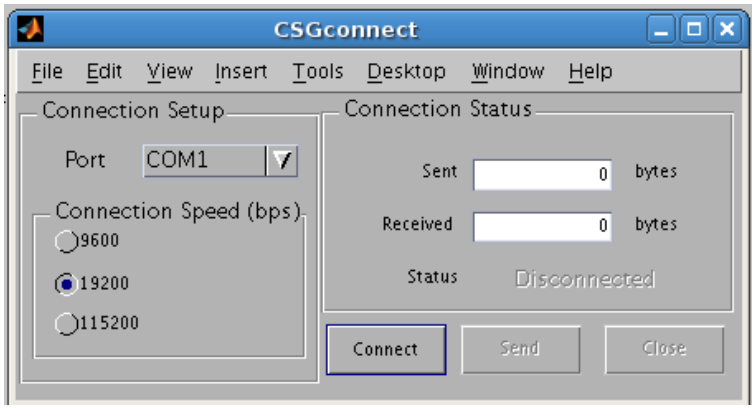


Figura 4.12: Interface do Módulo de Transmissão.

de trabalho de 100 mm de largura por 100 mm de altura. Deve-se supor que o bloco inicial de trabalho possui dimensões semelhantes. Assim como no exemplo apresentado anteriormente, considera-se ser necessário um faceamento inicial para nivelar a face superior ao objeto. Para isso, deve-se criar o objeto com a profundidade um pouco maior para que esse faceamento

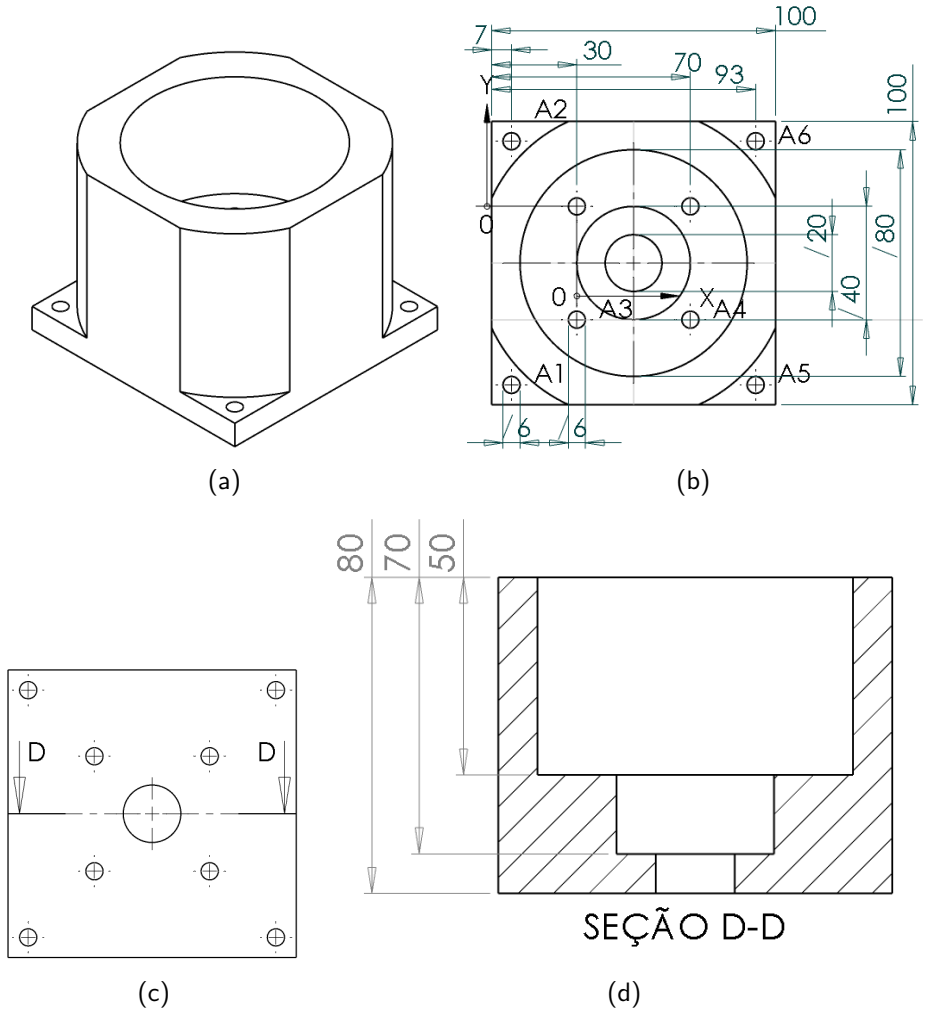
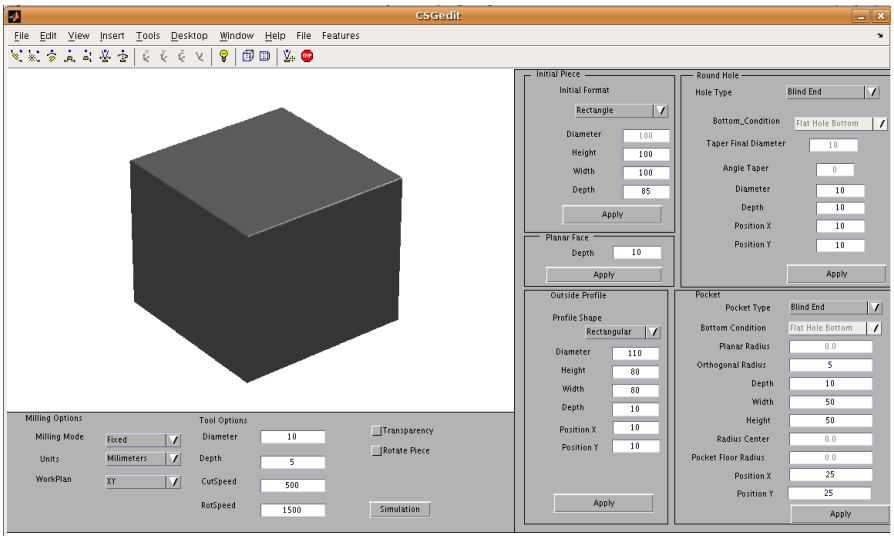


Figura 4.13: Desenho cotado representando uma peça a ser modelada. (a) visão isométrica da peça; (b) vista superior; (c) vista inferior; (d) vista seção D-D.

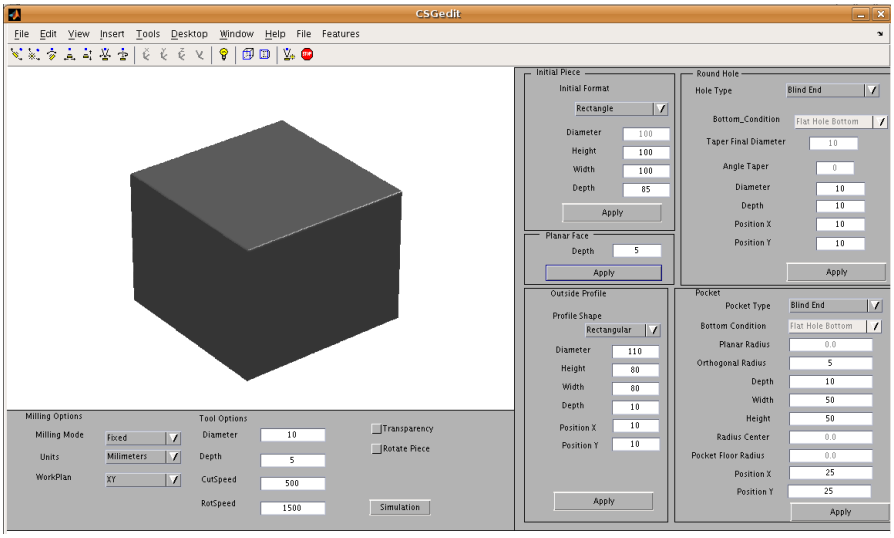
não influencie nas dimensões finais da peça. Para esse exemplo, criou-se um objeto de forma retangular, com 100 mm de largura, 100 mm de altura e 85 mm de profundidade para utilizar a função de faceamento. Ao aplicar a criação do objeto, tem-se a figura 4.14(a).

Como já citado, o bloco inicial foi especificado para que a função de faceamento seja executada sobre o objeto. Como a profundidade desejada do objeto de início é de 85 mm, o valor para a realização do faceamento é de 5 mm. Esse valor deve ser inserido para a função *Planar Face* da interface que, após ser executada, resulta no objeto visto na figura 4.14(b).

Depois do faceamento, a aplicação das *features* de *General Outside Profile* é executada com formato circular, a partir do centro e com diâmetro de 110 mm para que esteja conforme a figura 4.13(b). Os valores a serem atribuídos para a interface são: forma circular, diâmetro



(a)

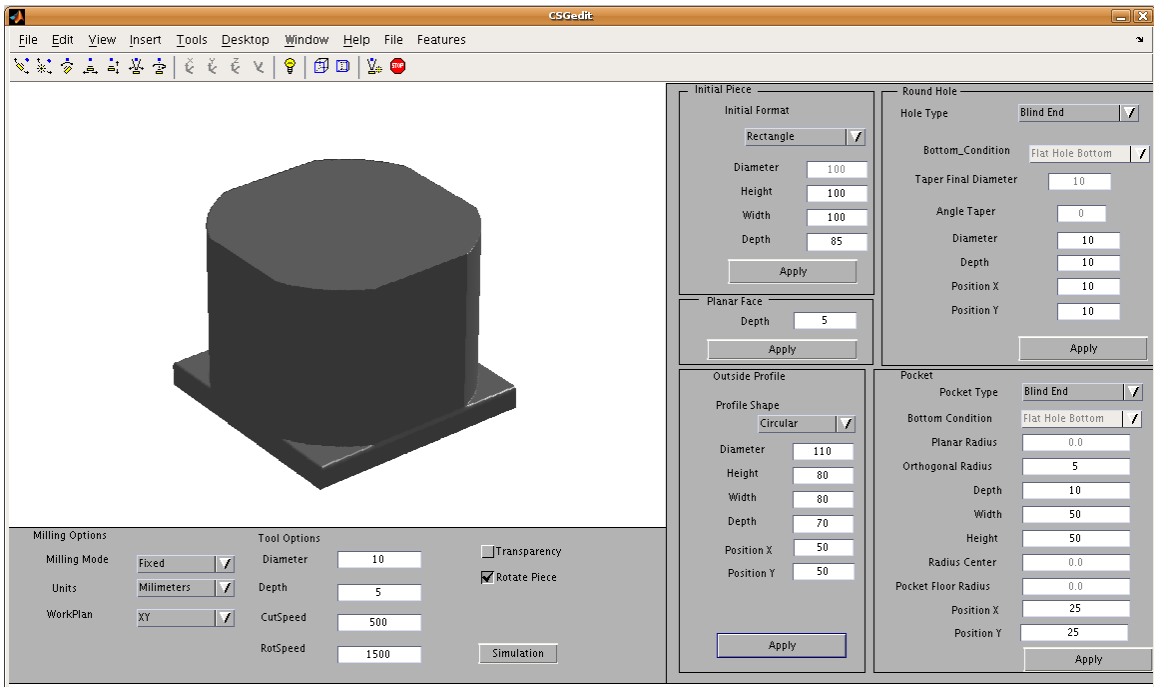


(b)

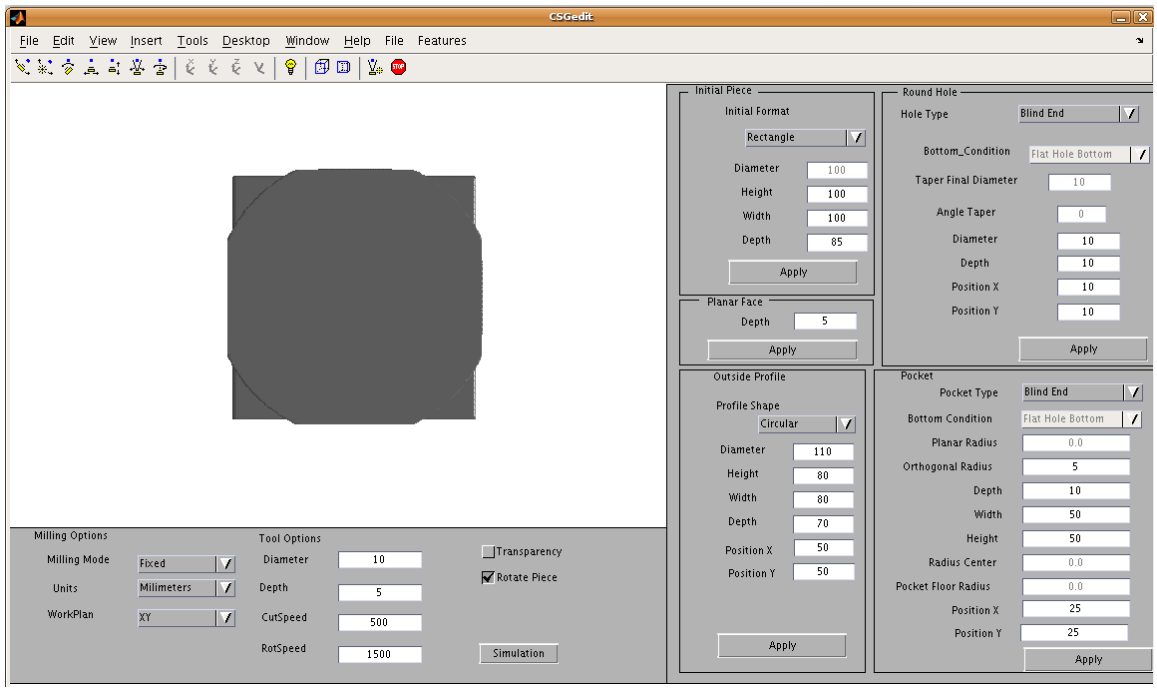
Figura 4.14: (a) Representação do bloco inicial para a modelagem da peça; (b) representação do bloco após a aplicação da *feature Planar Face*.

de 110 mm, profundidade de 70 mm e posição central em $X = 50$ mm e $Y = 50$ mm. Como resultado, tem-se a figura 4.15.

Resta a usinagem dos furos da peça. Inicialmente são feitos os furos fresados, por possuírem maior diâmetro. O primeiro deles possui diâmetro de 80 mm e 50 mm de profundidade com centro em $X = 50$ mm e $Y = 50$ mm. O segundo deles possui diâmetro de 40 mm e 70 mm de profundidade, enquanto o terceiro é do tipo passante e possui diâmetro de 20 mm. Após a inserção das *features* na interface com os valores especificados, a peça resultante é ilustrada na figura 4.16.



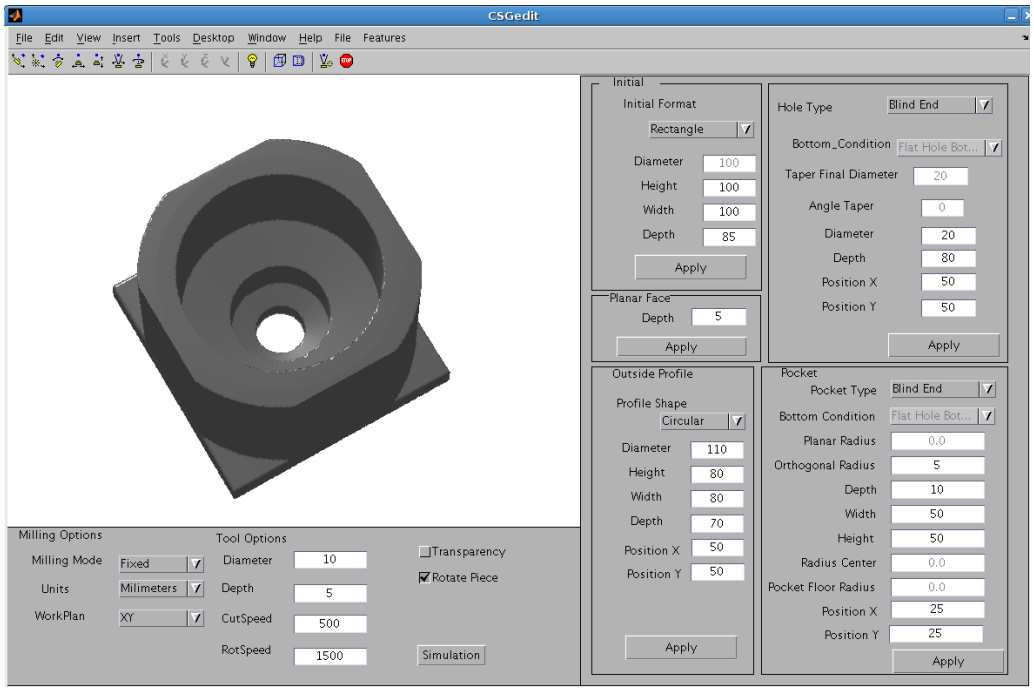
(a)



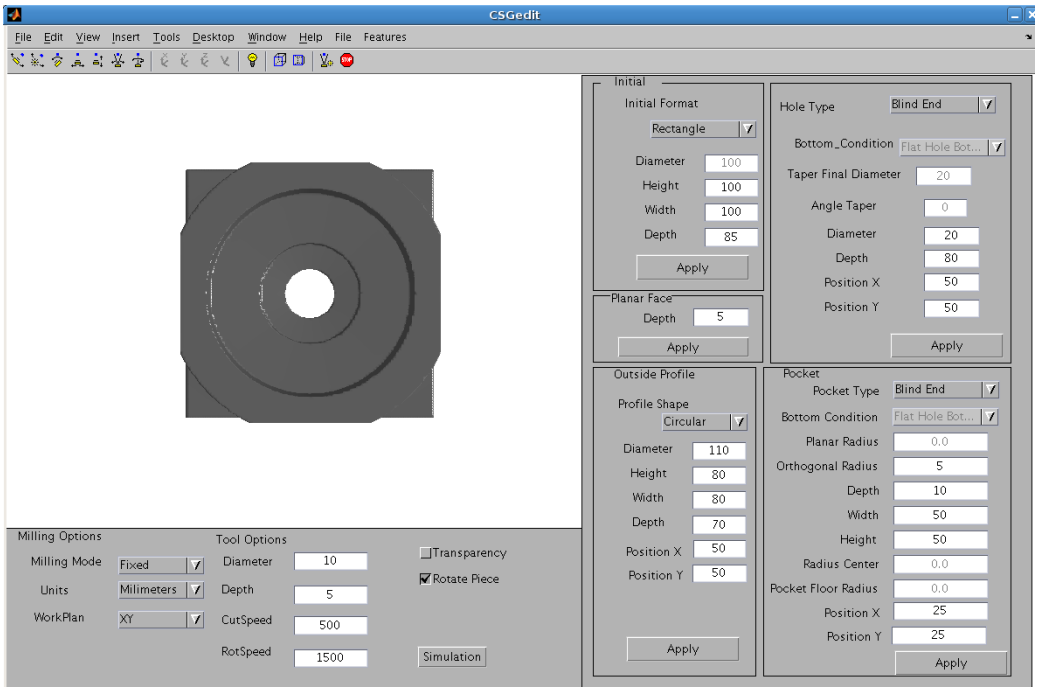
(b)

Figura 4.15: (a) Representação da aplicação da *feature Outside Profile* sob vista isométrica. (b) representação da *feature Outside Profile* sob vista superior.

Finalmente, devem ser aplicados os 8 furos restantes, de tipo passante (*through holes*) e diâmetro de 6 mm. Suas coordenadas são $X = 7$ mm, $Y = 7$ mm; $X = 93$ mm, $Y = 7$ mm; $X = 7$ mm, $Y = 93$ mm; $X = 93$ mm, $Y = 93$ mm; $X = 30$ mm, $Y = 30$ mm; $X = 70$ mm,



(a)



(b)

Figura 4.16: (a) Representação da aplicação de dois furos cegos e um furo passante a uma peça sob vista isométrica. (b) representação da aplicação de dois furos cegos e um furo passante sob vista superior (em perspectiva).

$Y = 30$ mm; $X = 30$ mm, $Y = 70$ mm e $X = 70$ mm, $Y = 70$ mm. Depois de sua aplicação, o modelo terá a forma apresentada na figura 4.17, concluindo-se, assim, a fase de modelagem

da peça. Quando o usuário estiver certo de que a peça está de acordo com a sua especificação, ele pode utilizar o módulo de simulação.

De forma semelhante ao apresentado para a peça anterior, o usuário pode observar linhas representando todo o trajeto da ferramenta de corte utilizandos controles de navegação do módulo de simulação, e assim que verificado, enviar o código de geração para a máquina-ferramenta.

4.2 Representação Geométrica

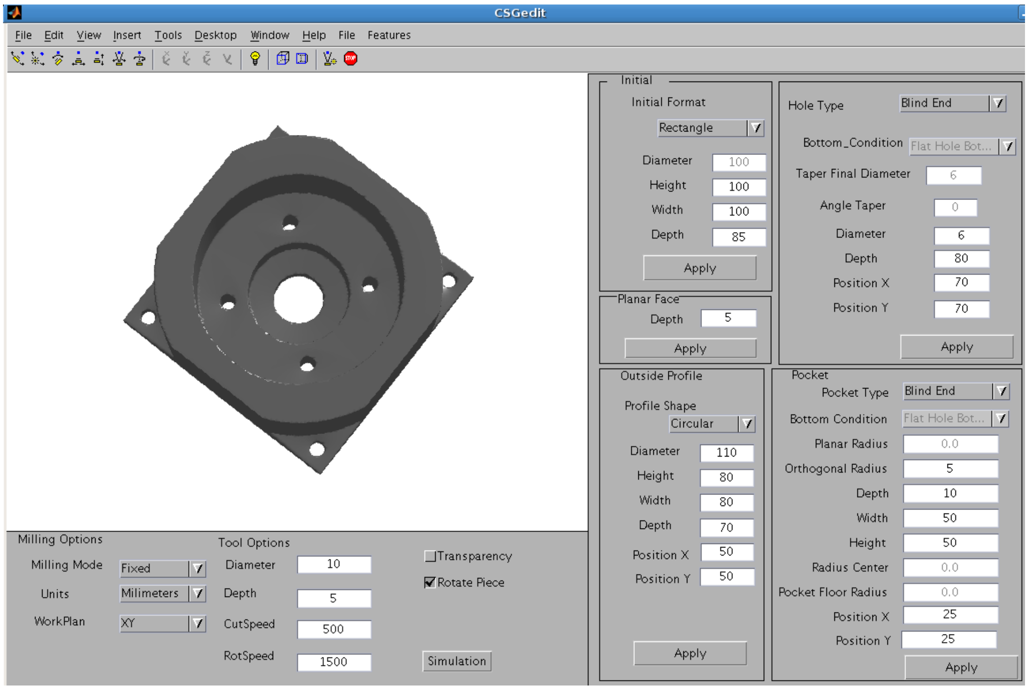
O pacote Matlab trouxe inúmeras facilidades à aplicação da técnica de enumeração de ocupação espacial, principalmente relacionadas à manipulação de matrizes, forma escolhida para representar o espaço neste trabalho.

Para demonstrar, considera-se o exemplo citado na seção 3.2: um espaço bidimensional, com o valor de 0.1 de resolução e o espaço de trabalho entre 0 e 1. O espaço dividido em 11 partes iguais será o espaço de trabalho. Utilizando o pacote Matlab, uma matriz é criada com essas dimensões, onde os valores 0 e 1 representam a *não ocupação* e a *ocupação* de um espaço, respectivamente. Essa matriz é gerada facilmente pelo Matlab através de uma única linha de código (algoritmo 4.1). O valor res representa a resolução do espaço. Na realidade, o Matlab aloca espaço vetorial para cada valor de x e y dentro do espaço de trabalho.

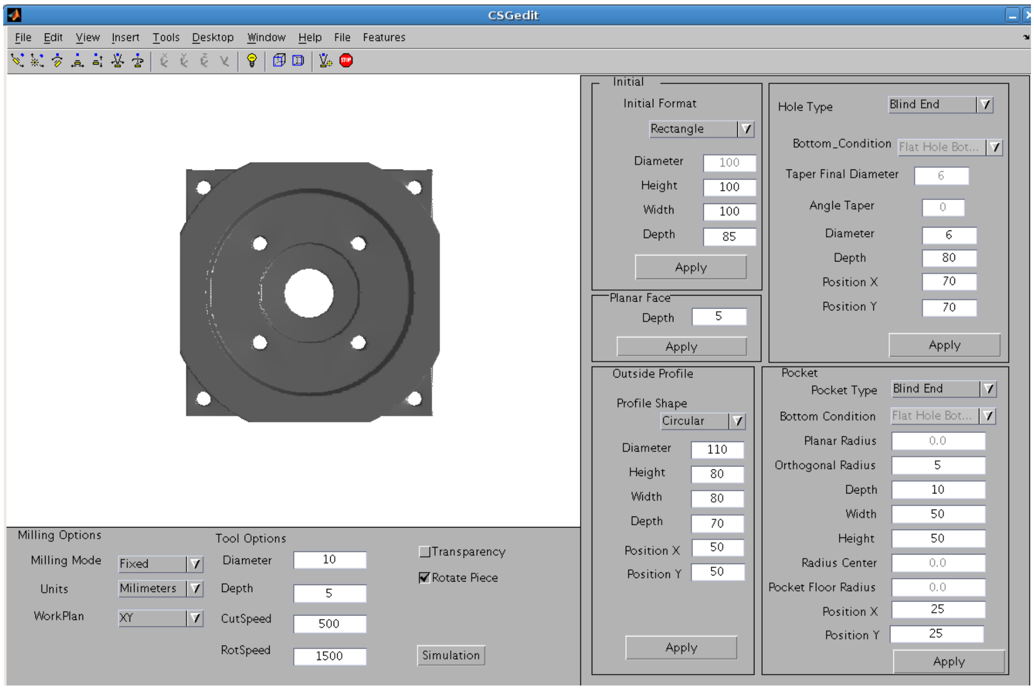
```
[x,y] = meshgrid(0:res:1, 0:res:1);
```

Algoritmo 4.1: Implementação em Matlab para a criação de um espaço bidimensional de valores.

A matriz M (equação 4.1) representa os valores de x, a matriz N (equação 4.2) contém os valores de y. Os valores das matrizes de x e y são então utilizados para os cálculos geométricos, onde assumem a forma vista na figura 3.7 e pela matriz 3.1.



(a)



(b)

Figura 4.17: (a) Representação da aplicação de oito furos passantes em vista isométrica. (b) representação da aplicação de oito furos passantes em vista superior (em perspectiva).

$$M = \begin{bmatrix} 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \\ 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \\ 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \\ 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \\ 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \\ 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \\ 0 & 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 & 0.7 & 0.8 & 0.9 & 1.0 \end{bmatrix} \quad (4.1)$$

$$N = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 & 0.3 \\ 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 & 0.4 \\ 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \\ 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 & 0.7 \\ 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 & 0.8 \\ 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \end{bmatrix} \quad (4.2)$$

Assim como a representação do espaço, pode-se demonstrar a implementação do exemplo do círculo visto na seção 3.2 pelo comando 4.2, cuja representação encontra-se na matriz MR (matriz 3.4).

```
circulo = sqrt( (x-0.5)^2 + (y-0.5)^2 ) - 0.5 <= 0;
```

Algoritmo 4.2: Comando para implementação em Matlab para a criação do sólido circular.

O algoritmo 4.3 encontra a função correspondente à criação da subdivisão do espaço tridimensional.

```
[x,y,z] = meshgrid(0:res:1, 0:res:1, 0:res:1);
```

Algoritmo 4.3: Implementação em Matlab para a criação de um espaço tridimensional de valores.

4.2.1 Implementação dos Sólidos

A seção 3.2 teve como objetivo apresentar a forma como os objetos principais da modelagem são construídos. Esta seção abordará aspectos de implementação referentes aos métodos de construção das primitivas e das operações utilizadas para a modelagem de tais objetos, assim como o fluxo que as informações de entrada do usuário seguem através da geração dos objetos.

4.2.1.1 Cubo

O sólido cubo é gerado a partir de sua coordenada central e do raio da esfera inscrita a ele. O algoritmo 4.4 representa a geração do sólido cubo através de seus atributos xc, yc e zc, que são as suas coordenadas centrais, de r, que corresponde ao raio e do parâmetro de resolução res. Inicialmente, o espaço é definido gerando-se três matrizes semelhantes às matrizes 4.1 e 4.2, uma para cada eixo de coordenadas. Depois, verifica-se para cada conjunto de valores de x, y e z se este pertence ao objeto cubo, pela comparação da diferença dos valores das coordenadas com o centro do objeto e o valor de r, ou seja, como já dito, para cada eixo, a todo espaço que possui coordenadas iguais ou inferiores ao raio é atribuído o valor verdadeiro e para cada valor fora dessas coordenadas é atribuído o valor falso.

Como pode ser observado, os valores de verdadeiro e falso utilizados pela implementação são os valores -1 e 1. Isso é feito para facilitar a utilização das operações posteriores sobre os conjuntos, assim como a representação tridimensional dos modelos.

```
function solidcube=CSGcube(xc,yc,zc,r,res)
[x,y,z]=meshgrid(0-res:res:1+res, 0-res:res:1+res, 0-res:res:1+res);
solidcube = 1 - 2*((abs(x-xc)<=r) & (abs(y-yc)<=r) & (abs(z-zc)<=r));
```

Algoritmo 4.4: Implementação em Matlab para a criação do sólido cubo.

4.2.1.2 Retângulo

O retângulo possui valores de largura, altura e profundidade especificados, em vez de possuir apenas o valor de um lado como no caso do cubo. Entretanto, o sólido retangular, ou paralelepípedo, é criado de forma semelhante ao sólido cubo, como pode ser visto no algoritmo 4.5. Como o objeto também é descrito com a origem em seu centro, suas coordenadas devem ser

divididas pela metade.

```
function solidrect=CSGrectangle(xc,yc,zc, height, width, depth,res)
[x,y,z]=meshgrid(0-res:res:1+res, 0-res:res:1+res, 0-res:res:1+res);
solidrect = 1 - 2*((abs(x-xc)<=width/2) & (abs(y-yc)<=height/2) &
(abs(z-zc)<=depth/2));
```

Algoritmo 4.5: Implementação em Matlab para a a criação do sólido retangular.

4.2.1.3 Cilindro

Visto que o sólido cilindro é criado através de círculos paralelos ao longo de um determinado eixo, sua função de criação (algoritmo 4.6) é dividida de acordo com o eixo escolhido, e os círculos dos quais é formado através dos outros dois eixos restantes em conjunto com seu raio. Caso uma unidade de volume (*voxel*) seja interna ou concordante ao círculo, esta fará parte do sólido resultante.

```
function solidcylinder=CSGcylinder(xc,yc,zc,r,axis,res)
[x,y,z]=meshgrid(0-res:res:1+res, 0-res:res:1+res, 0-res:res:1+res);
if (axis=='x') solidcylinder=sqrt( (y-yc).^2 + (z-zc).^2 ) - r ;
elseif (axis=='y') solidcylinder=sqrt( (x-xc).^2 + (z-zc).^2 ) - r ;
elseif (axis=='z') solidcylinder=sqrt( (x-xc).^2 + (y-yc).^2 ) - r ;
else error('axis must be x,y,or z') end
```

Algoritmo 4.6: Implementação em Matlab para a a criação do sólido cilíndrico.

A *feature* cilindro possui uma particularidade de não estar limitada em todos os eixos que a compõe. Por isso, para sua utilização é necessário utilizar apenas uma seção do sólido gerado. Essa seção é definida através de técnicas CSG. As operações CSG utilizadas para a criação do sólido cilíndrico, assim como o restante das *features* são descritas a seguir.

É importante ressaltar que não é necessário ao usuário especificar diretamente a criação das primitivas utilizadas. Essas informações são extraídas e tratadas de acordo com as informações fornecidas para cada *feature*.

Um exemplo que demonstra a representação gerada pela função de criação de um cilindro pode ser observado na figura 4.18

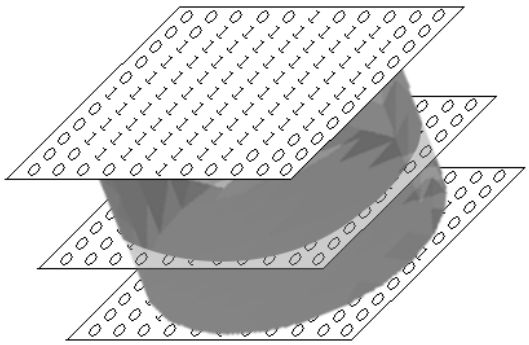


Figura 4.18: Representação matricial de um cilindro no espaço tridimensional.

4.2.1.4 União

Como o conjunto de valores dos objetos está limitado à ocupação ou não do espaço, a união de dois conjuntos representa o menor conjunto de elementos ocupados de espaço entre dois objetos de volume. A aplicação da função mínimo retorna a união desses dois objetos. A sua implementação pode ser vista no algoritmo 4.7.

```
function unioncombine=CSGunion(CSGobj1, CSGobj2)
unioncombine = min(CSGobj1, CSGobj2);
```

Algoritmo 4.7: Código em Matlab referente à operação de união de sólidos.

4.2.1.5 Interseção

A interseção de dois conjuntos A e B é o conjunto dos elementos que são comuns a A e B, isto é, os elementos pertencem a A e também pertencem ao conjunto B (ou todos os elementos de B que também pertencem ao conjunto A) [8, 23]. A interseção representa o maior conjunto com os elementos pertencentes aos dois sólidos simultaneamente. A implementação em Matlab

da operação de interseção pode ser vista no algoritmo 4.8.

```
function intersectcombine=CSGintersection(CSGobj1, CSGobj2)
intersectcombine = max(CSGobj1, CSGobj2);
```

Algoritmo 4.8: Código em Matlab, referente à operação de interseção de sólidos.

4.2.1.6 Subtração

A subtração entre os conjuntos A e B corresponde ao conjunto de todos os elementos de A que não pertencem a B [8, 23, 39], ou seja, refere-se ao maior conjunto dado entre o volume do primeiro objeto e o volume contrário ao segundo objeto. A implementação da subtração é mostrada no algoritmo 4.9.

```
function subtractcombine=CSGsubtract(CSGobj1, CSGobj2)
subtractcombine = max(CSGobj1, -CSGobj2);
```

Algoritmo 4.9: Código em Matlab referente à operação de subtração.

4.3 Implementação das Features

O uso das *features* incluídas neste trabalho visa à avaliação da viabilidade da implementação descrita pela norma ISO 14649, assim como a avaliação da possibilidade de utilizá-las em conjunto com a ISO 6983.

Quanto à conversão para sua utilização em concordância com a ISO 6983, apenas as principais características de cada *feature* foram implementadas. Dessa forma, atributos como *bottom condition* (condições de fundo) e *tapered end* (tipo de furo onde o diâmetro final é diferente do inicial) não foram cobertos pela implementação, mesmo que descritos na norma 14649. O conjunto dessas *features* encontra-se no apêndice A.

4.3.1 Planar Face

Essa *feature* geralmente é utilizada para nivelar a face superior da peça de trabalho a uma determinada profundidade. Como o topo da peça de trabalho é tomado como referência, cabe

ao usuário apenas especificar a profundidade para a *feature*. Essa profundidade é comparada com os parâmetros informados às ferramentas de corte, como profundidade e diâmetro efetivos, para executar a operação de usinagem. Entretanto, essa informação fica transparente ao usuário, necessitando a ele apenas a definição dos atributos relativos à peça, ferramenta e *feature*. Sua representação é realizada por meio da remoção de um retângulo com a forma desejada da face superior do objeto de trabalho. A árvore CSG da aplicação da *feature Planar Face* pode ser vista na figura 4.19

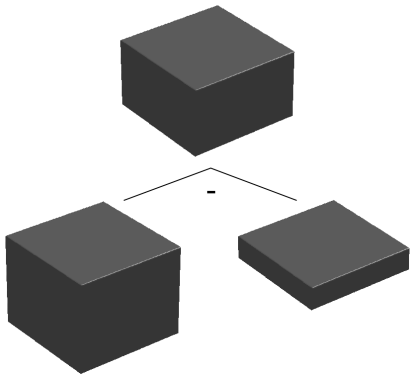


Figura 4.19: Árvore de CSG para a aplicação da *feature Planar Face*.

4.3.2 Round Hole

Essa *feature* foi definida considerando que a ferramenta de corte possibilita a descida direta (ou furação). Assim como para as outras *features*, foram obedecidas as regras quanto aos valores máximos de descida, incluindo plano de retração, permanência e descarga, através da utilização das funções disponibilizadas pela máquina-ferramenta estudada.

Já para os furos maiores que o diâmetro da ferramenta, utilizou-se uma função de interpolação helicoidal disponível pela máquina para a usinagem apropriada. Os furos usinados são aplicados quando o diâmetro informado é maior que o diâmetro da ferramenta de corte, possibilitando essa operação.

A diferenciação dos tipos de furos é transparente ao usuário, evitando que ele necessite conhecer funções específicas da máquina-ferramenta. A construção da *feature hole* é ilustrada na árvore na figura 4.20.

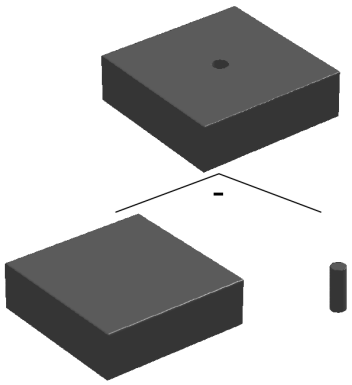


Figura 4.20: Árvore de CSG para a aplicação da *feature Round Hole*.

4.3.3 Pocket

Como descrito no capítulo 2, algumas *features* possuem divergências quanto as suas definições e especificações. Isto ocorre com a *feature Pocket*. Segundo a norma ISO 14649, a *feature Pocket* possui sua geometria definida pelo contorno da face superior e sua profundidade, possuindo os tipos aberto e fechado, bem como algumas divisões quanto à condição do fundo do *pocket*. Sua descrição não especifica o formato da geometria de seu contorno superior, sendo apenas especificado o tipo "perfil fechado".

Neste trabalho, essa geometria foi implementada como um tipo de retângulo de cantos arredondados. A *feature pocket* é definida pelos seus atributos altura, largura, profundidade, raio ortogonal e posicionamento, como já descrito no capítulo 3. Ela foi descrita possuindo sempre um raio ortogonal, mesmo que este seja o do raio da ferramenta, quando o valor especificado for menor que o da ferramenta de corte ou quando não especificado. Por meio desses parâmetros, inicialmente são efetuados os cortes referentes aos quatro cantos que compõem a *feature*, utilizando as mesmas operações encontradas na operação de furação. Depois são traçados os cortes referente ao interior do *pocket*.

A árvore CSG da aplicação (figura 4.21) se inicia com a junção espacial dos cilindros que compõem os quatro cantos da *feature*. Como dito anteriormente, eles possuem a extensão de toda a peça, e depois são seccionados através da operação de intersecção. Paralelamente, dois retângulos são unidos para que formem o restante do *pocket*. Então, a operação de união é aplicada entre esses dois objetos para formarem o modelo positivo do *pocket*, para depois

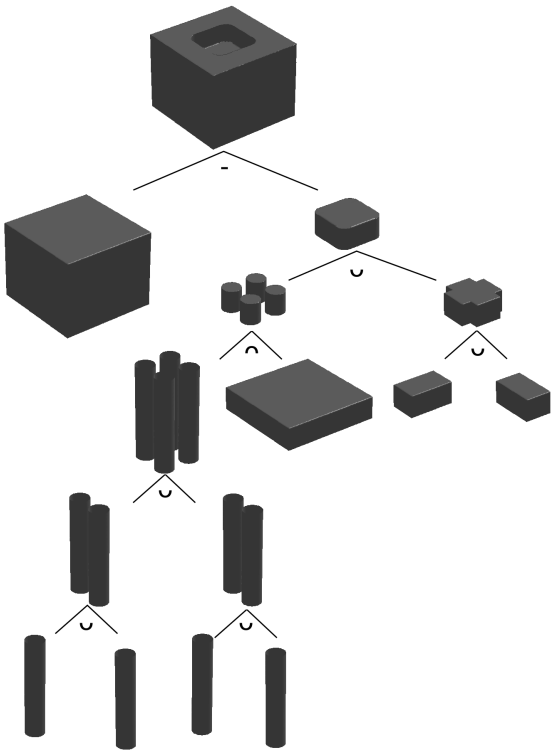


Figura 4.21: Árvore de CSG implementada para a aplicação da *feature Pocket*.

este volume de material ser removido da peça.

4.3.4 Outside Profile

Assim como a *feature Pocket*, embora a *feature Outside Profile* ou *General Outside Profile* descrita na norma ISO 14649 possua forma indefinida, foram utilizadas duas formas básicas para sua implementação, para não ser necessário um módulo particular para criação de tal modelo. As formas criadas foram a retangular e a circular, devido à grande abrangência de sua aplicação. A *feature* possui os atributos altura, largura, profundidade e posição X e Y, para a forma retangular, necessários para representar o formato e posicionamento da *feature* sobre a peça de trabalho.

A modelagem dessa peça se dá através da remoção do formato externo à peça em questão. Isso é feito subtraindo um retângulo com o formato informado de outro com largura e altura correspondentes à peça de trabalho e profundidade igual a desejada. Este objeto gerado é então removido da peça, como demonstrado na figura 4.22.

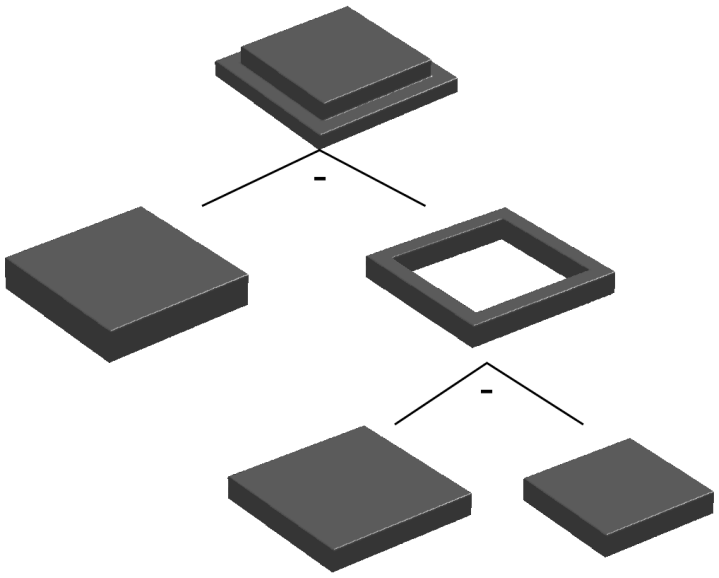


Figura 4.22: Árvore de CSG para a aplicação da *feature Rectangular Outside Profile*.

Já para o formato circular, ao invés de possuir os atributos altura e largura, estes foram substituídos pelo atributo diâmetro. Assim como no caso da *feature* de formato retângular, o modelo é criado através da remoção de material externo aos parâmetros informados. A diferença é se inclui um passo para a criação do cilindro, devido as propriedades vistas na seção 4.2.1. A árvore padrão para geração da *feature* pode ser vista na figura 4.23.

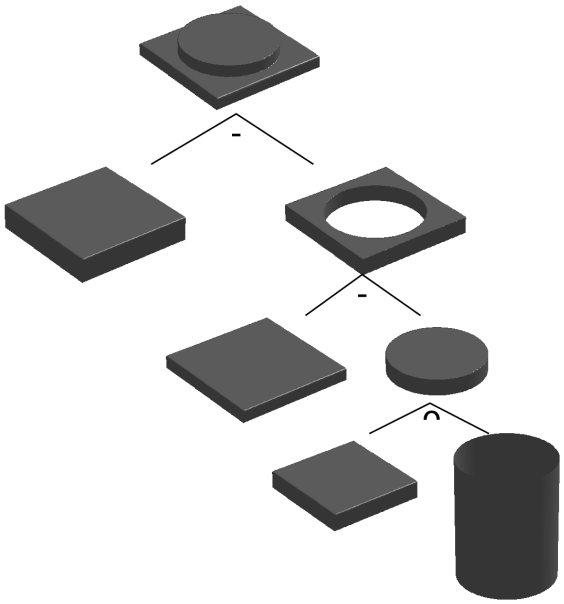


Figura 4.23: Árvore de CSG para a aplicação da *feature Circular Outside Profile*.

4.4 Dificuldades Encontradas

Várias dificuldades surgiram no decorrer do desenvolvimento do protótipo. Devido à natureza do problema abranger tópicos multidisciplinares, nas áreas de Mecânica e Computação, os problemas encontrados podem ser divididos nessas duas áreas, além dos problemas encontrados na conjunção das mesmas. Em um âmbito geral, algumas dificuldades ocorreram devido a alguns problemas não previstos durante o levantamento bibliográfico, como ambientes que não atenderam a todas as características desejadas, à complexidade das *features* em sua abrangência, entre outras.

Esta seção apresenta os principais problemas encontrados nas áreas de Mecânica e Computação. Especificamente às dificuldades associadas com a resolução de problemas na área da Mecânica e com a operação das máquinas-ferramentas, são destacadas a seguir.

4.4.1 Dificuldades Relativas à Manufatura

O desenvolvimento de uma ferramenta para edição e simulação de peças traz alguns desafios para a área de Manufatura. Algumas dificuldades observadas neste trabalho estão descritas a seguir.

Estratégia de Desbaste

Um dos principais problemas relativos ao desenvolvimento de um simulador ou aplicativo para modelar uma peça em um centro de usinagem através de suas características (*features*) é a formulação de uma boa estratégia de desbaste. O desbaste consiste na retirada do material desnecessário ao redor da geometria final da peça e depende de vários fatores, como conjunto de peças e ferramentas disponíveis e a otimização do caminho a ser percorrido, entre outros.

Para o problema do desbaste, foram pesquisadas algumas estratégias existentes, encontradas em algumas ferramentas de *software* utilizadas atualmente, que possuem comprovado funcionamento. Elas foram divididas de acordo com as *features* em que são empregadas.

Uma situação extremamente difícil relacionada ao desbaste é estabelecer um caminho ótimo para o traçado das ferramentas e também verificar quais são as impossibilidades de desbaste

diante de um conjunto de ferramentas. O processista deve ter a certeza de que, com o seu conjunto de ferramentas, todo o material seja retirado antes do processo de acabamento.

O processo de desbaste está intimamente ligado ao processo de acabamento, pois define quais as áreas possíveis e limites para a entrada e traçado das ferramentas de acabamento. As três principais estratégias de desbaste, como indicadas pelo sistema SolidCAM [38] são:

- Desbaste por contorno: é a principal estratégia para limpar grandes áreas e remover grandes volumes de material eficientemente. O procedimento deve garantir uma determinada profundidade de corte e remover o material sem deixar falhas. A profundidade de corte deve permitir a usinagem de faces planas. Tanto entrada em rampa como em hélice podem ser usadas. Nas melhores estratégias, arcos suaves são criados automaticamente, assim como passos e movimentos de ligação, eliminando vibrações e melhorando as velocidades de corte e a vida útil da ferramenta.

- Desbaste de machos: é uma estratégia otimizada para usinagem de placa macho de moldes de fora para dentro. Todos os caminhos da ferramenta iniciam no lado externo, na profundidade em Z e operam até o contorno do macho. O contato de corte deve ser mantido ao máximo possível para evitar redução da vida útil da ferramenta.

- Desbaste de sobra de material: esse desbaste é realizado com uma ferramenta menor, após o uso de uma ferramenta maior, em áreas não alcançadas pela operação anterior. Para grandes peças, pode-se executar mais de uma operação de desbaste de sobra de material, com ferramentas cada vez menores, dependendo da precisão desejada.

Há ainda outras técnicas avançadas, como a de Lefebvre e Lauwers [20], que utiliza *morphing* para gerar uma superfície intermediária para executar o desbaste, estratégia de mergulho, entre outras.

A estratégia para o desbaste foi baseada no desbaste por contorno, por possuir relação com os métodos utilizados para a descrição dos modelos gráficos gerados e devido a sua segurança quanto à geração do trajeto da ferramenta de corte. Como mencionado na seção de simulação e geração do código G (seção 3.1.2) , procurou-se estabelecer um valor limite para o máximo

desbaste de cada ferramenta de corte, com relação ao raio de corte e a profundidade máxima de corte, para evitar a super-utilização e desgaste excessivo de cada uma delas.

Tipo de Material Usinado

Estratégias de usinagem são classificadas de acordo com o material a ser trabalhado. Além da definição da ferramenta, cada material se comporta de maneira diferente.

É preciso estudar profundamente a relação entre cada material para poder determinar os fatores relacionados à usinagem, principalmente se for necessário atribuir automaticamente valores aos diversos parâmetros desse processo. Por esse motivo, não foi abordada a implementação de fatores relacionados aos tipos de materiais diferentes para a peça de trabalho ou para as ferramentas de corte utilizadas.

Velocidade de avanço

Algumas características da usinagem possuem estreita relação com o material trabalhado e com a ferramenta de corte. Uma dessas características é a velocidade de avanço, que é a velocidade com que o CNC movimenta a ferramenta de corte quando está em trabalho de remoção de material.

Neste trabalho não foram implementadas as relações entre o material da ferramenta de corte e o material da peça de trabalho. Por isso, a velocidade de avanço para cada ferramenta deve ser definida pelo usuário antes do processo de modelagem, para que a geração da simulação de traçado e código de execução estejam em concordância com essas relações.

Controle de Cavaco

Cavaco é o produto resultante do movimento relativo entre a ferramenta e a peça. É importante que, no processo de usinagem, o cavaco gerado não se acumule nas áreas ainda a serem usinadas, pois, dependendo do material que estiver sendo trabalhado, o contato de um excesso de cavaco com a área não cortante da ferramenta pode gerar danos à ferramenta.

Como o controle de cavaco tem estreita relação com a velocidade de avanço e estratégia

de desbaste escolhidos, então parte desse processo foi implementada junto à estratégia de desbaste. Através do desbaste de contornos, procura-se retirar o máximo de material com o menor desgaste da ferramenta de corte. Além disso, para a reentrada da ferramenta de corte, procura-se ter uma área livre ao redor da ferramenta de corte compatível com o procedimento, para que não haja acúmulo de material desnecessário, e para que este acúmulo possa ser tratado pela própria movimentação da ferramenta de corte.

Para o casos dos furos, como já dito na seção 4.3.2, utilizou-se um procedimento que retira parte do cavaco gerado a cada descida da ferramenta de corte. Já a velocidade de avanço deve ser prevista pelo usuário antes da execução da usinagem, conforme mencionado na seção 4.4.1.

Fixação

Um dos problemas inerentes à usinagem, principalmente em peças com geometria complexa, é a fixação. Cada peça deve possuir em seu planejamento o uso de uma estratégia de fixação, e esta não deveria interferir na usinagem.

Apesar disso, algumas precauções devem ser tomadas. Um choque (contato) pode ocorrer entre a ferramenta de trabalho e o mecanismo de fixação durante os movimentos de posicionamento ou aproximação no processo de usinagem.

Não foram inseridos na ferramenta final meios para o controle de adição ou choque aos fixadores, ficando a critério do usuário a análise desse risco.

Ferramentas de Trabalho

Assim como o processo de desbaste influi no desenho final da peça, outra parte fundamental é a ferramenta de trabalho. São inúmeras as ferramentas de trabalho para o processo de usinagem, tal que cada uma apresenta suas características e restrições. Essas características têm um papel importante na fabricação da peça, pois, em conjunto com as restrições da máquina, determinará o desenho (geometria) final da peça.

Para o problema dos limites das máquinas, foram estabelecidos os limites da máquina-ferramenta utilizada como padrão. Para o limite das ferramentas, foi estabelecido um tipo

de ferramenta como padrão, o qual pode descrever todas as funcionalidades abrangidas pelo sistema (desbaste, furação). Embora não esteja explícito, a implementação executa comparações e o código para troca de ferramenta de corte e adequação a novas ferramentas está implementado, seguindo o modelo proposto. É necessário que o usuário atribua o valor de raio ou diâmetro de corte, assim como a sua profundidade máxima de corte, para cada uma das ferramentas com que esteja trabalhando na máquina.

Um problema comum, mas que também deve ser apontado, é o fato de o conjunto de ferramentas determinar quais procedimentos (ou *features*) podem ser executados. Com isso, uma das ferramentas pode influenciar significativamente o processo geral da usinagem. Para solucionar esse problema, o conjunto de ferramentas foi limitado a uma ferramenta funcional e foram determinados seus procedimentos a partir da análise de alguns *software* existentes no mercado.

Outro aspecto não abordado pelo sistema é a calibragem da máquina-ferramenta. O sistema considera pontos iniciais de trabalho, tal que a ferramenta toca a peça de trabalho em seu canto inferior esquerdo, tomando todos os outros pontos relativos a este. O sistema não necessita de correção de ferramenta, pois todo o posicionamento é calculado em consideração ao raio especificado para a ferramenta de corte, sempre tomando precauções de segurança quanto ao diâmetro e profundidade da mesma.

Em vários casos, a peça não possui apenas uma face a ser usinada. Como a máquina-ferramenta possui o comando chamado de $2\frac{1}{2}$ eixos, rotações na peça são necessárias na usinagem caso possua mais de uma dimensão. Essas faces adicionais a serem usinadas devem ser consideradas em novas usinagens, ficando a cargo do usuário estabelecer suas inter-relações.

4.4.2 Dificuldades Relativas à Computação

Além das dificuldades encontradas na área de Manufatura, outras dificuldades estão diretamente ligadas à área de Computação, principalmente referentes ao ambiente de desenvolvimento e à forma de representação utilizada. Algumas dessas dificuldades são descritas a seguir.

Dois conjuntos de pacotes de *software* de desenvolvimento diferentes foram utilizados

para a implementação da metodologia. O primeiro deles procurou utilizar a linguagem de programação C++ e a biblioteca OpenCSG para a geração dos modelos em conjunto com a API gráfica OpenGL. O segundo, apresentado neste trabalho, utilizou o pacote matemático Matlab para o desenvolvimento de todo o sistema.

A utilização do primeiro conjunto não apresentou o resultado esperado, principalmente em relação à utilização da biblioteca para geração dos modelos gráficos (tridimensionais). Embora a biblioteca OpenCSG possua um bom desempenho para a geração dos modelos, operações e manipulação dos objetos, ela apresentou problemas funcionais específicos para o desenvolvimento deste trabalho, referente à visualização da modelagem e também à simulação da peça de trabalho. Dessa forma, essa tentativa foi abandonada após avaliação dos requisitos para a resolução de tais problemas. O autor conseguiu desenvolver um protótipo de acordo com o proposto, mesmo assim, houve uma perda em relação a sua abrangência. Essas perdas são citadas adiante.

Mesmo sendo uma ferramenta com diversos recursos e tendo auxiliado tanto na tarefa de gerar modelos matemáticos quanto em gerá-los graficamente, o pacote Matlab possui algumas deficiências quando comparado com outras ferramentas de desenvolvimento. A visualização dos modelos é constantemente solicitada e, a cada modificação no modelo, é necessária uma conversão do modelo guardado na forma matricial, como explicado nas seções anteriores, para uma representação gráfica suportada pelo pacote. Com isso, são necessárias várias conversões ao longo da execução do protótipo, gerando um gargalo no sistema que prejudica a interação do usuário com a ferramenta. Além disso, como o número de objetos gráficos é geralmente extenso, a manipulação direta dos objetos gráficos pode ser afetada de acordo com o *hardware* utilizado para execução do sistema.

A utilização de um sistema próprio para a visualização também ocasionou dificuldades com o Matlab, pois não se consegue fazer uso de técnicas que fazem uso de *hardware* específico, que seria utilizado através da API OpenGL ou enviando comandos gráficos diretamente à unidade de processamento gráfico (GPU) por funções ou bibliotecas específicas (considerando que a versão utilizada para implementação do protótipo não possui suporte direto às funções de OpenGL). Com isso, técnicas como as apresentadas em Guha et al. [13], Romeiro et al. [32],

que utilizam comandos de GPU para suas aplicações, e outras como Bohez et al. [3], Erhart e Tobler [10], Stewart et al. [40] ou Herres [14], que utilizam a API OpenGL e linguagem C ou C++; e mesmo em Fang e Liao [11] que utiliza uma versão mais antiga desses pacotes, não puderam ser testadas ou comparadas. Essas técnicas atuam principalmente com o desempenho da criação e manipulação dos objetos tridimensionais utilizados na etapa de modelagem.

A técnica de SOE provou ser eficiente por meio de sua representação matricial. Outras técnicas poderiam ser aplicadas com sucesso para a representação de sólidos. Métodos consagrados como os algoritmos de GoldFeather e SCS [**?**], além de alternativas a esses algoritmos como os descritos em Stewart et al. [40] e Erhart e Tobler [10], não foram testados devido à falta de recursos para a implementação desses algoritmos e métodos.

Em termos de usabilidade, alguns problemas foram encontrados para desenvolver o sistema de acordo com características padrões às ferramentas do tipo CAD/CAM, como por exemplo a não representação das dimensões dos modelos após terem sido gerados, sendo necessário ao usuário ter como verdadeiras as dimensões a ele apresentadas, sem possuir um retorno direto sobre elas. Finalmente, outra dificuldade foi o conjunto escasso de objetos para o desenvolvimento de uma Interface Gráfica de Usuário (GUI, do inglês *Graphical User Interface*) mais sofisticada.

CAPÍTULO 5

CONCLUSÕES

Este trabalho apresentou uma metodologia para desenvolvimento de um protótipo para auxiliar a criação de peças de usinagem, utilizando um tipo especial de máquina-ferramenta, comandado por CNC, por meio do emprego de *features* de usinagem.

Espera-se contribuir com o desenvolvimento do processo de manufatura atual por meio do uso de técnicas mais modernas e eficazes, como a utilização de *features* especificadas na norma ISO 14649, bem como a aplicação de abordagens para modelagem, programação, simulação e apresentação gráfica 3D das peças a serem usinadas.

O protótipo implementado permite o uso de *features* com uma máquina-ferramenta de grande utilização na indústria de manufatura, aproximando o desenvolvimento da tecnologia de manufatura atual às máquinas antigas, não descartando as máquinas-ferramentas hoje disponíveis no mercado e procurando adequá-las aos passos futuros que pretendem ser dados na área. O protótipo é composto por um Módulo de Edição para a modelagem da peça por meio de características da norma, um Módulo de Simulaçao com recursos gráficos para permitir a visualização e avaliação do modelo criado e um Módulo de Transmissão do código automaticamente gerado para execução na máquina-ferramenta.

A combinação das técnicas de representação por geometria sólida construtiva e por enumeração de ocupação espacial, a partir de um conjunto de primitivas gráficas, mostrou-se adequada para a modelagem dos objetos. As primitivas foram combinadas para formar o objeto sólido resultante por meio de uma seqüência ordenada de operações Booleanas. Uma estrutura hierárquica, chamada árvore CSG, foi usada para controlar a aplicação das operações Booleanas.

Algumas pontos para investigação futura são propostos para melhorar algumas etapas relativas à metodologia desenvolvida e à implementação do protótipo.

Embora o projeto do protótipo tenha recebido colaboração de especialistas da área de

mecânica, uma avaliação detalhada do protótipo está prevista de ocorrer com um número maior de usuários do Departamento de Engenharia Mecânica da Universidade Federal do Paraná.

Apenas um conjunto limitado de *features* da norma ISO-14649 foi atualmente implementado no protótipo. Pretende-se incorporar um número maior de *features*, ampliando-se a abrangência de uso do protótipo. Da mesma forma, pretende-se prover suporte a outros tipos de ferramenta de corte, o que também auxiliará a inclusão de novas *features*. A aplicação de outras estratégias de usinagem pode oferecer ao usuário uma melhor escolha de acordo com os materiais e ferramentas utilizados.

Um estudo mais profundo é necessário para investigar novas formas de representação gráfica que possuam complexidade menor de visualização.

Finalmente, embora não tenha sido o objetivo inicial deste trabalho, o uso de pacotes de código aberto e de livre distribuição permitirão maior flexibilidade na extensão do protótipo, assim como redução dos custos com as licenças de softwares. Nesse sentido, o pacote Octave poderia substituir o Matlab atualmente utilizado no desenvolvimento do protótipo.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Alfio Quarteroni, F. S. *Scientific Computing with MATLAB and Octave*. Springer, 2006.

[2] Benouamer, M. O. e Michelucci, D. Bridging the Gap between CSG and Brep via a Triple Ray Representation. In *SMA '97: Proceedings of the Fourth ACM Symposium on Solid Modeling and Applications* (New York, NY, Estados Unidos, 1997), ACM Press, pp. 68–79.

[3] Bohez, E. L., Minh, N. T. H., Kiatsrithanakorn, B., Natakuson, P., Ruei-Yun, H. e Son, L. T. The Stencil Buffer Sweep Plane Algorithm for 5-axis CNC Tool Path Verification. *Computer-Aided Design 35*, 12 (2003).

[4] Brown, C. M., Requicha, A. A. G. e Voelcker, H. B. Geometric Modelling Systems for Mechanical Design and Manufacturing. In *ACM '78: Proceedings of the 1978 Annual Conference* (New York, NY, Estados Unidos, 1978), ACM Press, pp. 770–778.

[5] Buchele, S. F. e Crawford, R. H. Three-dimensional Halfspace Constructive Solid Geometry Tree Construction from Implicit Boundary Representations. In *SM '03: Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications* (New York, NY, Estados Unidos, 2003), ACM Press, pp. 135–144.

[6] Cameron, S. e Yap, C.-K. Refinement Methods for Geometric Bounds in Constructive Solid Geometry. *ACM Transactions on Graphics 11*, 1 (1992), 12–39.

[7] Chang, T.-C., Wysk, R. A. e Wang, H.-P. *Computer-Aided Manufacturing*. Prentice-Hall, Inc., Upper Saddle River, NJ, Estados Unidos, 1998.

[8] de Alencar Filho, E. *Teoria Elementar dos Conjuntos*. Livraria Nobel S.A., São Paulo, SP, Brasil, 1978.

[9] DUFF, T. Interval Arithmetic Recursive Subdivision for Implicit Functions and Constructive Solid Geometry. In *SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, Estados Unidos, 1992), ACM Press, pp. 131–138.

[10] ERHART, G. E TOBLER, R. General Purpose Z-buffer CSG Rendering with Consumer Level Hardware. VRVis 003, VRVis Zentrum für Vurtual Reality und Visualisierung Forschungs-GmbH, 2000.

[11] FANG, S. E LIAO, D. Fast CSG Voxelization by Frame Buffer Pixel Mapping. In *VVS '00: Proceedings of the 2000 IEEE Symposium on Volume Visualization* (New York, NY, USA, 2000), ACM, pp. 43–48.

[12] GILAT, A. *MATLAB: An Introduction with Applications*, 2nd ed. John Wiley & Sons, 2004.

[13] GUHA, S., KRISHNAN, S., MUNAGALA, K. E VENKATASUBRAMANIAN, S. Application of the Two-sided Depth Test to CSG Rendering. In *I3D '03: Proceedings of the 2003 Symposium on Interactive 3D Graphics* (New York, NY, Estados Unidos, 2003), ACM, pp. 177–180.

[14] HERRES, G. Real Time Constructive Solid Geometry Rendering using 3D Texture Mapping. *Journal of Computing Sciences in Colleges 19*, 5 (2004), 333–335.

[15] HIGHAM, D. Sandia National Labs Achieves Breakthrough Performance Using NVIDIA Technology for Scientific Visualization. `http://www.nvidia.com/object/IO_19962.html`. Acessado em 15 de junho de 2008.

[16] ISO/FDIS. Industrial automation systems and integration - Physical device control - Data model for computerized numerical controllers - Part 10: General process data. `http://www.steptools.com/library/stepnc/tech_resources/`.

[17] JANSEN, F. W. Depth-order Point Classification Techniques for CSG Display Algorithms. *ACM Transactions on Graphics 10*, 1 (1991), 40–70.

[18] Ji, Q. e Marefat, M. M. Machine Interpretation of CAD Data for Manufacturing Applications. *ACM Computing Surveys 29*, 3 (1997), 264–311.

[19] Juan-Arinyo, R. e Sole, J. Constructing Face-octrees from Voxel-based Volume Representations. *Computer-Aided Design 27*, 10 (1995), 783–791.

[20] Lefebvre, P. P. e Lauwers, B. 3D Morphing for Generating Intermediate Roughing Levels in Multi-Axis Machining. *Computer-Aided Design and Applications. Vol. 2, no. 1-4* (2005), 115–123.

[21] Leff, L. e Yun, D. Y. Y. Constructive Solid Geometry: A Symbolic Computation Approach. In *SYMSAC '86: Proceedings of the Fifth ACM Symposium on Symbolic and Algebraic Computation* (New York, NY, Estados Unidos, 1986), ACM Press, pp. 121–126.

[22] Leondes, C. T. *Computer Aided Design, Engineering, and Manufacturing: Systems Techniques and Applications*. CRC Press, 2001.

[23] Lipschutz, S. *Teoria dos Conjuntos*. AO Livros Técnicos S.A., Av. Rio Branco, 81/12o. Andar Z.C. 21, Rio de Janeiro, RJ, 1968.

[24] Loffredo, D. Fundamentals of STEP Implementation. `http://www.steptools.com/library/fundimpl.pdf`.

[25] Lopes, J. M. B. Modelação Geométria. `http://disciplinas.ist.utl.pt/leic-cg/programa/livro/Modelacao.pdf`. Acessado em 15 de junho de 2008.

[26] MathWorks, I. MATLAB Documentation, 2004. `http://www.mathworks.com/`.

[27] Maziero, N. L., Ferreira, J. C. E., Pacheco, F. S. e Prim, M. F. A Feature-Based Object-Oriented Expert System to Model and Support Product Design. *Journal of the Brazilian Society of Mechanical Sciences Vol 22* (2000).

[28] Middleditch, A. E. e Reade, C. A Kernel for Geometric Features. In *Symposium on Solid Modeling and Applications* (Atlanta, GA, Estados Unidos, Maio 1997), pp. 131–140.

[29] MÄNTYLÄ, M. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, MD, Estados Unidos, 1988.

[30] PEREIRA, A. G. Desenvolvimento e Avaliação de um Editor para Programação CN em Centros de Usinagem. Dissertação de Mestrado, Universidade Federal do Paraná, Curitiba - PR, 2003.

[31] PRESSMAN, R. E WILLIAMS, J. E. *Numerical Control and Computer-Aided Manufacturing*. Wiley, New York, NY, Estados Unidos, 1977.

[32] ROMEIRO, F., VELHO, L. E DE FIGUEIREDO, L. H. Hardware-assisted CSG rendering. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Research Posters* (New York, NY, Estados Unidos, 2006), ACM, p. 119.

[33] ROSSO JR., R. S. U. E NEWMAN, S. T. Estrutura de Dados para Sistemas CAD/CAM Aderente à STEP. In *Proceedings of VI Congresso Ibero-Americano de Engenharia Mecânica - Universidade de Coimbra, Coimbra, Portugal* (Outubro 2003).

[34] ROSSO JR., R. S. U., NEWMAN, S. T. E RAHIMIFARD, S. The Adoption of STEP-NC for the Manufacture of Asymmetric Rotational Components. *Proceedings - Institution of Mechanical Engineers - Part B Journal of Engineering Manufacture 218* (2004), 1639–1644.

[35] SALESIN, D. E STOLFI, J. Rendering CSG Models with a ZZ-buffer. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, Estados Unidos, 1990), ACM Press, pp. 67–76.

[36] STEP APPLICATION HANDBOOK - ISO 10303 - VERSION 3, 2006.

[37] SHIROMA, Y., KAKAZU, Y. E OKINO, N. A Generalized Sweeping Method for CSG Modeling. In *SMA '91: Proceedings of the First ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications* (New York, NY, Estados Unidos, 1991), ACM Press, pp. 149–157.

[38] SolidCAM, I. Estratégias High Speed Machining: Desbaste. http://www.solidcam.com/hsm_roughing_strategies_pt,25155.html. Acessado em 15 de juno de 2008.

[39] Spindler, K. *Abstract Algebra with Applications*. Marcel Dekker, Inc, New York, NY, Estados Unidos, 1994.

[40] Stewart, N., Leach, G. e John, S. An Improved Z-Buffer CSG Rendering Algorithm. *1998 Eurographics/Siggraph Workshop on Graphics Hardware* (Agosto 1998), 25–30.

[41] Stewart, N., Leach, G. e John, S. Improved CSG Rendering using Overlap Graph Subtraction Sequences. *International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia - GRAPHITE 2003* (Fevereiro 2003), 47–53.

[42] Suh, S. H., Lee, B. E., Chung, D. H. e Cheon, S. U. Architecture and Implementation of a Shop-floor Programming System for STEP-compliant CNC. *Computer-Aided Design 35*, 12 (2003).

[43] Vandenbrande, J. e Requicha, A. Geometric Computation for the Recognition of Spatially Interacting Machining Features. In *Advances in Feature-Based Manufacturing, JJ. Shah, D. Nau, and M. Mantyla, Eds. Elsevier/North Holland, Amsterdam* (1994), pp. 83–106.

[44] Weck, M. e Wolf., J. STEP-NC. The STEP Compliant NC Programming Interface: Evaluation and Improvement of the Modern Interface. *In:IMS Forum. Ascona/Switzerland* (Outubro 2001).

[45] W.Weisstein, E. Cube., 1998. http://mathworld.wolfram.com/Cube.html.

[46] Zeid, I. *Mastering CAD/CAM*. McGraw-Hill, Inc., New York, NY, Estados Unidos, 2005.

APÊNDICE A

NORMA 14649 - FEATURES

Este apêndice apresenta algumas *features* da norma ISO-14649, descritas em ISO/FDIS [16] :

- *Planar Face:* utilizada para descrever a parte usinada da face externa de uma peça. A geometria de uma face planar é dada pelo limite e profundidade. A profundidade denota o fundo do material que precisar ser retirado da peça para atingir a forma final da *feature* (figura A.1).

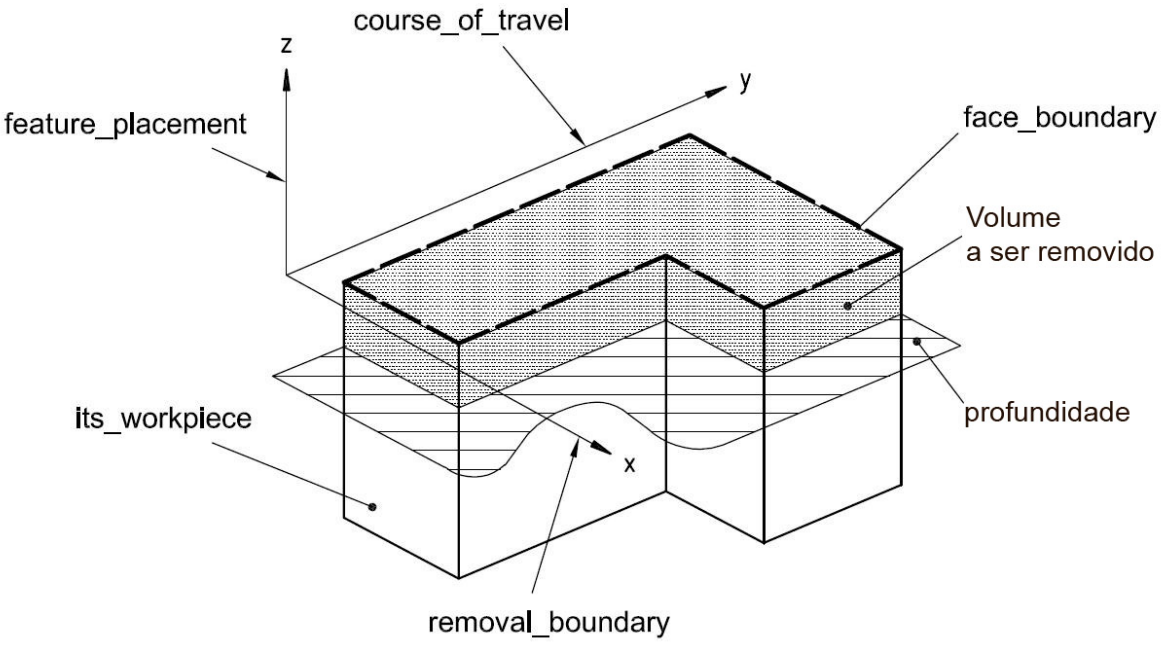


Figura A.1: *Feature Planar Face*. Adaptado de [44].

- *Pocket*: é o supertipo abstrato para *pockets* implícitos e explícitos diferentes. Pockets abertos e fechados são derivados deste supertipo. A geometria de um *pocket* é definida pelo seu contorno da face externa da peça e sua profundidade.

  - *Closed Pocket*: é um *pocket* cercado por material ao redor de toda a sua circunferência. Seu limite é dado por um *profile* fechado.

- *Open Pocket*: é um *pocket* não fechado. Seu limite é dado por um contorno de parede.

- *Slot*: a entidade *slot* é um tipo especial de *pocket*. Tipicamente, um *slot* será manufaturado por um simples *sweep* da ferramenta ao longo do curso. Neste caso, a largura do *slot* é igual ao formato da ferramenta. No caso de uma operação de faceamento o formato é dado pelo diâmetro da ferramenta. Se um *workingstep* que usine este *slot* chamar por uma ferramenta menor, mais de um corte terá que ser efetuado.

  - *Woodruf Slot end*: o final do *slot* deve ser um raio tangente ao fundo do *slot* e curvado para cima sobre um eixo.
  - *Radiused Slot end*: o final do *slot* consiste em um arco. O diâmetro é igual a largura da ferramenta. O centro do arco é idêntico ao ponto final do trajeto.
  - *Flat Slot end*: o final do *slot* consiste em uma linha plana com dois arcos conectando o final dos lados do *slot*. O raio dos dois arcos são dados. Atravessando o *slot* através de sua linha central do inicio ao final, o raio do lado esquerdo do canto é corner_radius1, o raio do lado direito da linha central é o corner_radius2. O ponto final do caminho (*course_of_travel*)está no final plano do *slot*.
  - *Loop Slot end*: o *slot* forma um laço. A profundidade é usada para determinar a profundidade escalar do *slot*. A profundidade escalar é igual a menor distância entre o trajeto de curso e a face planar descrita pelo atributo "*depth*" da *Feature*.
  - *Open Slot end*: *slot* cujo fim é aberto.

- *Step*: *Step* (ou degrau) é um volume de material removido do topo para os lados de uma peça de trabalho. Como um *pocket* aberto, seu contorno é aberto em seus lados. A parte do "perfil V" descrevendo o fundo do *step* encontra-se na superfície elementar definida pelo atributo "profundidade" da *Feature*.

- *Profile*: a *Feature* profile é um volume de material removido de uma forma do limite de uma peça de trabalho. É um supertipo abstrato de *General Outside Profile* e *Shape Profile*.

- *General Outside profile*: é o volume retirado de uma forma arbitrária da forma externa de uma peça de trabalho. Ele pode remover material de toda a forma externa ou de apenas uma parte da forma. O contorno da forma é dado pelo atributo "*feature_boundary*". E a profundidade é dada pelo atributo "*depth*".

- *Shape Profile*: é o volume da remoção de um *profile* cuja forma vem da forma do limite de uma peça de trabalho. O fundo da forma do limite é limitado pela condição de chão (*floor condition*). É o supertipo abstrato de um "*general shape profile*", "*partial circular shape profile*", "*circular closed shape profile*", "*rectangular open shape profile*", e "*rectangular closed shape profile*".

- *Profile Select*: a condição de chão de um profile de forma é um "*through profile floor*" ou uma "*profile floor*".

- *Through Profile Floor*: descreve a condição de fundo de um "*shape profile*" que é aberto.

- *Profile Floor*: descreve a condição de fundo de um "*shape profile*" que pode ser plano ou de uma forma arbitrária.

- *Round Hole*: define furos, furos cônicos, furos passantes e cegos. Seus atributos são o ponto central do furo localizado a $x = y = 0$ no sistema de coordenadas local. A profundidade especificada (positiva) faz a ferramenta avançar o furo na direção negativa em z, e o tipo do fundo não é levado em consideração com a profundidade do furo.

  - *Taper Select*: indica a maneira pela qual a conicidade é descrita, podendo ser por diâmetro ou ângulo.

  - *Hole bottom condition*: Supertipo abstrato para a descrição para o fundo de um furo.

  - *Through bottom condition*: Descreve um fundo aberto de um furo.

  - *Blind bottom condition*: Supertipo que descreve diferentes tipos de condições de fundo cego. O furo pode passar sobre o fundo da peça de trabalho, mas não é totalmente aberto. A profundidade do furo é o comprimendo da seção cilíndrica

do furo. Os subtipos de *blind bottom condition* são: *Flat hole condition*, *Flat with radius hole condition*,*Spherical hole condition* e *Conical hole condition*.

- *Tool Path Feature*: tem por função permitir a definição de movimentos não cobertos pelas *Features* anteriores. Caminhos de ferramentas tem que ser atribuidos para as operações associadas a esta *feature*.

- *Boss*: é uma *Feature* que necessita ser relacionada a uma outra *Feature*. Nenhum *workingstep* pode ser atribuído para manufaturar um *boss*. Ao invés disso, *boss* é aquele material que fica sem ser trabalhado após a manufatura de uma *Feature* com um *boss*. Por esta razão, *boss* não pode ser uma *Feature* de manufatura independente mas existe apenas como um atributo.

- *Spherical Cap*: consiste-se de todos os pontos a uma dada distância de um ponto constituindo seu centro. O centro é definido pela sua colocação, que não é localizado no ponto mais alto da *Feature*, mas sim no centro da esfera.

- *Rounded End*: é uma forma particular circular que passa sobre um caminho linear.

- *Compound feature*: é a *Feature* composta de duas ou mais features, sem espaçamento regular entre os elementos da *compound feature*. Em geral, também não são *features* do mesmo tipo.

- *Replicate Feature*: é a montagem de um número de *Features* similares, por exemplo um círculo de furos ou uma malha de furos. A *Feature* é descrita apenas uma vez e o número e espaçamento da *Feature* é descrita. Os atributos descrevem a localização da it Feature relativa a sua posição no padrão de replicação como especificado pelo subtipo de *Replicate Feature*.

  - *Circular Pattern*: é um padrão circular de *Features*. Um círculo completo de *Features* é um caso especial de *circular pattern*.

  - *Rectangular pattern*: é uma descrição de elementos arranjados em um padrão retangular. Pode ser tanto um *grid* de elementos com n linhas e m colunas e um número total de nxm elementos ou ou uma linha única de m elementos ($n = 1$).

- *Transition Feature*: é uma *Feature* que pode ser adicionada a borda de duas *Features*. Exemplo: um canto arredondado ou um chanfro entre duas faces planares ou entre uma face planar e um *pocket*.

- *Thread*: é uma cadeia de seções uniformes na forma de uma hélice na superfície interna ou externa de um cilindro.

- *Profile*: é um contorno planar usado na definição de uma *Feature*. Um *profile* pode ser do tipo aberto ou fechado. Um *profile* deve estar no plano X-Y e pode ter uma orientação que o posicionará em referência ao sistema de coordenada local de uma *Feature* de manufatura, que pode exigir um *profile* como uma parte de sua definição.

    - *Open profile*: é um tipo de *profile* que é um contorno ou forma sem limites fechados ou confinados. Os finais do *profile* podem se estender indefinidamente.

    - *Linear profile*: é um tipo de *open profile* que é uma linha reta de comprimento especificado. O *linear profile* deve ser orientado paralelo ao eixo X.

    - *Closed profile*: é um tipo de *profile* que é um contorno ou forma que confina uma área fechada sem uma abertura.

- *Travel Path*: é um conjunto de curvas que define a direção de um curso. Essas curvas não se interceptam ou duplicam. Um *travel path* pode possuir sua própria orientação em referência ao sistema de coordenadas local da *Feature* de manufatura que o exige como parte da definição da *Feature*.

- *Surface texture parameter*: é a combinação do parâmetro "nome" e possivelmente indica descrever um parâmetro particular de uma textura de superfície tal como aspereza ou ondulação.