VI CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA
VI NATIONAL CONGRESS OF MECHANICAL ENGINEERING
18 a 21 de agosto de 2010 – Campina Grande – Paraíba - Brasil
August 18 – 21, 2010 – Campina Grande – Paraíba – Brazil

UM SISTEMA CAM PARA MÁQUINAS DE PROTOTIPAGEM RÁPIDA DE BAIXO CUSTO

Silvio Roberto Montenegro Martins, silvio.montenegro@gmail.com[1]
Branilson Luiz Santos Costa, branilson@gmail.com [1]
Herman Augusto Lepikson, herman@ufba.br[1]

[1]Universidade Federal da Bahia - Programa de Pós-Graduação em Mecatrônica.
Rua Aristides Novis, 02, Escola Politécnica, Federação, CEP 40210-630, Salvador, BA, Brasil

Resumo: *Com a evolução das técnicas de manufatura, novos produtos são lançados no mercado com uma velocidade cada vez maior. Uma das formas de viabilizar a redução no ciclo de desenvolvimento de produtos é através da junção de etapas do processo, como projeto, testes e fabricação. A prototipagem rápida está inserida neste contexto, permitindo a construção de objetos físicos a partir de modelos CAD. Essa técnica de construção assistida por computador provê uma alternativa rápida para produção de protótipos funcionais, se comparada aos modelos convencionais de criação. Entretanto, a maioria das soluções disponíveis no mercado são proprietárias, fato que resulta em tecnologias fechadas e em custos elevados. O foco deste trabalho é no desenvolvimento de um sistema CAM capaz de atuar em uma máquina de prototipagem rápida de baixo custo. Esse sistema será responsável pelo tratamento dos arquivos tridimensionais, apresentando soluções para o fatiamento e planejamento das rotas de deposição.*

Palavras-chave: *Prototipagem Rápida, CAM (Computer Aided Manufacturing),Manufatura aditiva.*

1. INTRODUÇÃO

A integração do mercado mundial aumentou a concorrência entre as empresas. Para manter a competitividade é preciso investimentos constantes visando ao desenvolvimento de novos produtos e processos. Tais investimentos devem priorizar a redução de tempo e custo, melhorando-se também a capacidade de testes funcionais e produtibilidade dos protótipos.

A materialização de sólidos a partir de arquivos no computador atrai a atenção de variados setores da economia. Entretanto, a prototipagem rápida está restrita a alguns segmentos por envolver um alto custo para aquisição de equipamentos, manutenção e produção dos protótipos. O desenvolvimento de um sistema de prototipagem rápida de baixo custo que permita a difusão desta tecnologia constitui um desafio que tem sido buscado por diversos grupos de pesquisa. Dentre os projetos de maior destaque podemos citar (RepRap, 2009; Fab@Home, 2009 e MakerBot, 2009).

O presente trabalho está focado na manufatura aditiva (construção de sólidos através da adição de material). O objetivo é o desenvolvimento de um sistema CAM capaz de gerenciar máquinas de prototipagem rápida.

Existem diversas técnicas diferentes de manufatura aditiva, dentre elas: estereolitografia (SLA), modelagem por fusão e deposição (FDM), sinterização seletiva a laser (SLS), manufatura de objetos em lâminas (LOM) e impressão tridimensional (3DP). A principal diferença entre estas tecnologias está no princípio físico de construção do protótipo, porém, no que tange ao aspecto computacional, as etapas do processo de planejamento possuem pontos em comum. De modo geral, este processo pode ser dividido em fases. A Figura (1) mostra o fluxo das atividades que está descrito a seguir (Koc, 2001):

- **projetar em um software CAD o sólido a ser prototipado**. Assim que o modelo estiver concluído, este deve ser exportado para um formato de arquivo específico. Atualmente o STL é o formato de arquivo tridimensional mais utilizado em máquinas de prototipagem;
- **definir a orientação do objeto a ser prototipado**. A definição da orientação pode afetar diversas características do protótipo final. A posição de prototipagem influencia em aspectos como a quantidade de suporte necessária e o total de fatias geradas;
- **calcular a necessidade de suporte.** O suporte é necessário sempre que a ação da gravidade possa afetar alguma face do sólido, provocando a queda ou desestabilização da mesma. No caso da Fig. (1), há um exemplo de faces que podem ser afetadas: o nariz e o queixo do crânio.

- **fatiamento do sólido.** Nesta etapa são geradas todas as camadas do sólido. As fatias obtidas serão construídas de forma seqüencial na máquina de prototipagem.
- **planejamento da rota.** Consiste na tradução das informações de cada camada no modelo físico real. O planejamento envolve definição de rotas para preenchimento de bordas, deposição de material e deposição de suporte.

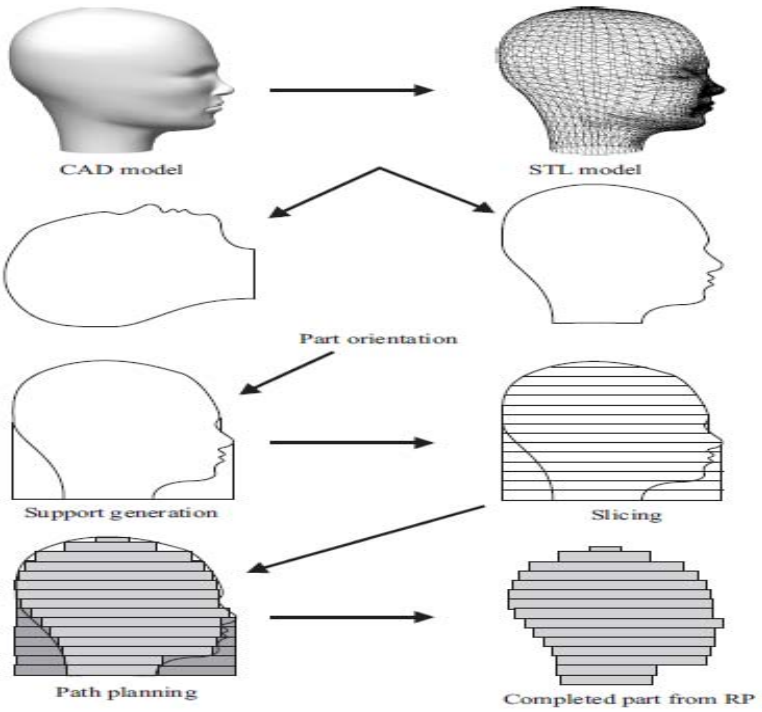

Figura 1 - Etapas do processo de prototipagem de Koc (2001).

A partir dessas informações, podemos definir quatro tarefas básicas como pertencentes ao processo de planejamento: fatiamento, orientação, suporte e planejamento de rota (Kulkarni et al, 2000).

Como pode ser observado na Fig. (2), estas etapas podem ser subdivididas em dois domínios. O domínio do modelo envolve o processamento tridimensional e irá definir questões como orientação e suporte. O domínio da camada tratará da definição do planejamento da rota da ferramenta. A tarefa responsável por realizar a conversão das informações tridimensionais para o domínio das camadas (bidimensional) é o fatiamento.

O fato do processo de prototipagem ser realizado em camadas reduz a complexidade geométrica do objeto modelado. Este fator permite que os sistemas de prototipagem rápida funcionem com pouca intervenção humana (Kulkarni et al, 2000).

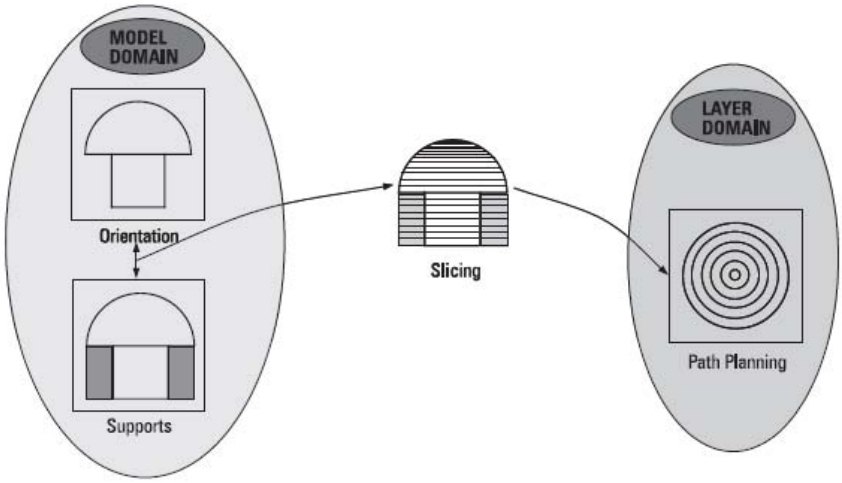

Figura 2 – Mapeamento dos domínios da informação de Kulkarni et al. (2000).

Visando apresentar o sistema CAM de prototipagem rápida, este artigo está organizado como descrito a seguir: na seção 2 será demonstrada a estrutura do sistema e a solução encontrada para o fatiamento; os testes e validações aplicados ao sistema serão descritos na seção 3; finalmente, na seção 4, algumas considerações finais e direções de trabalho futuros serão abordadas.

2. ESTRUTURA DO SISTEMA

A partir da análise do estado da arte da prototipagem rápida, foi definido o ambiente em que a proposta deste trabalho está inserida. O objetivo é atender às etapas compreendidas ao software no PC. Porém, para validar o sistema é necessário atuar em conjunto com outras etapas, pois na prototipagem rápida o software no PC é integrante de um sistema maior que envolve também um software embarcado e um hardware que serão responsáveis por atender os comandos do software e realizar as tarefas necessárias para gerar o protótipo final.

A Figura (3) mostra o ciclo completo definido para a prototipagem de um objeto. As áreas relacionadas ao software no PC estão incluídas no escopo do software. Neste artigo será apresentada com mais detalhes a solução para o fatiamento.

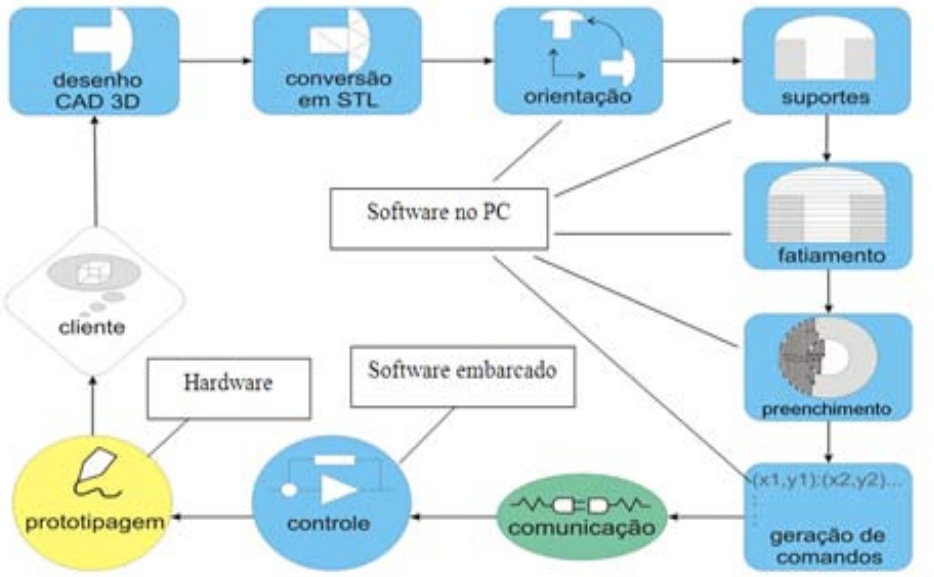

Figura 3 – Ambiente em que o software se enquadra.

O fluxo apresentado se inicia a partir da necessidade do cliente de prototipar um determinado objeto. Esta idéia é modelada em um software CAD e exportada para o formato STL. A partir deste ponto, inicia-se a interação com o sistema CAM de prototipagem. O formato STL é a interface de entrada do sistema.

Assim que o arquivo STL é carregado pelo software, deve ser escolhida uma orientação para o objeto. Em seguida, o fatiamento é realizado, sendo definidas também as estruturas de suporte. Com a conclusão de tais etapas, a informação é convertida para o domínio da camada através do fatiamento.

Para cada camada obtida será definida a rota de preenchimento. Esta rota será passada para o software embarcado através de um protocolo definido. O software embarcado deve receber a lista de comandos e executá-los sequencialmente. Desta forma, ao final deste processo será obtido o sólido tridimensional. Este deve estar de acordo com o modelo vislumbrado pelo cliente no seu software CAD de modelagem 3D.

A proposta deste trabalho é oferecer um modelo genérico, capaz de ser gerado em qualquer linguagem de programação. Porém, para fins de validação, é necessário codificar a proposta. A linguagem Java foi escolhida para o desenvolvimento do software, visto que esta possui algumas vantagens em relação aos requisitos do software:

- orientação a objetos: este paradigma, além de trazer conceitos que ajudam na codificação, permite a utilização da UML (*Unified Modeling Language*). Este facilitador é importante nas fases de análise, projeto e documentação;
- multi-plataforma: como o Java trabalha com o conceito de máquina virtual, o programa desenvolvido nesta linguagem pode rodar em diversos sistemas operacionais sem necessitar ser recompilado. Isto acontece porque o código fonte é compilado para uma linguagem intermediária denominada *bytecode*. No momento da execução, a máquina virtual é responsável por traduzir para linguagem de máquina as instruções do programa.
- suporte à computação 3D: a Sun, empresa responsável pelo desenvolvimento da linguagem Java, disponibiliza uma API (Application Programming Interface) para permitir que o desenvolvedor crie aplicações que necessitam de computação gráfica tridimensional. Esta API é denominada JAVA 3D API (Sun, 1999).

No Java, para criar um ambiente 3D, é necessário criar um mundo virtual denominado *VirtualUniverse*, o qual contém uma série de elementos que podem controlar ou influenciar este ambiente. A classe *Shape3D* representa qualquer objeto visível e encapsula todas as informações deste objeto dentro do Universo. Ao *Shape3D* podem ser associadas outras informações que vão determinar como o objeto será renderizado (Selman, 2002).

Cada objeto contido no universo virtual possui formas e posições diferentes no espaço. O Java armazena essas informações de forma hierárquica. Tal estrutura é mantida em árvore e é denominada *SceneGraphObject*.

O *ViewPlatform* representa um ponto de observação do universo. A partir desta classe é possível renderizar os diversos objetos do universo na tela do monitor. O volume de visão de cada *ViewPlatform* é determinado pela distância entre dois parâmetros que são configuráveis: *near clipping plane* (plano de visão mais próximo) e *far clipping plane* (plano de visão mais distante). Só serão renderizados os objetos do universo que estiverem dentro do volume de visão. A Figura (4) ilustra este processo.

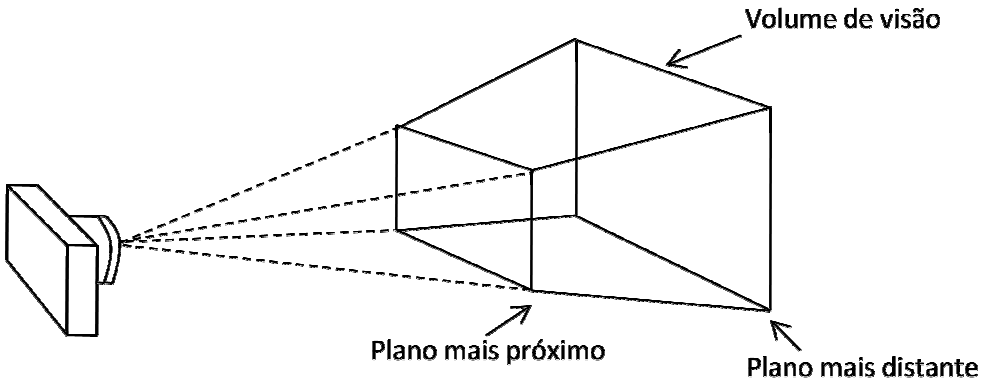

Figura 4 - Modelo de visualização das câmeras.

A partir destes conceitos, a proposta de planejamento do software foi montada. A seguir, será mostrado como o universo virtual foi concebido segundo o modelo conceitual do sistema, posteriormente será realizada uma avaliação mais detalhada em cada uma das atividades incluídas no escopo deste projeto com uma ênfase maior no processo de fatiamento.

2.1. Universo Virtual

O universo virtual a princípio possui apenas uma câmera. O volume de visão (*view frustum*) desta câmera é reduzido; assim, à medida que um objeto estiver passando por dentro da câmera, esta será capaz de visualizar apenas uma fatia deste objeto. A Figura (5) demonstra o estado inicial do universo no momento de sua criação.

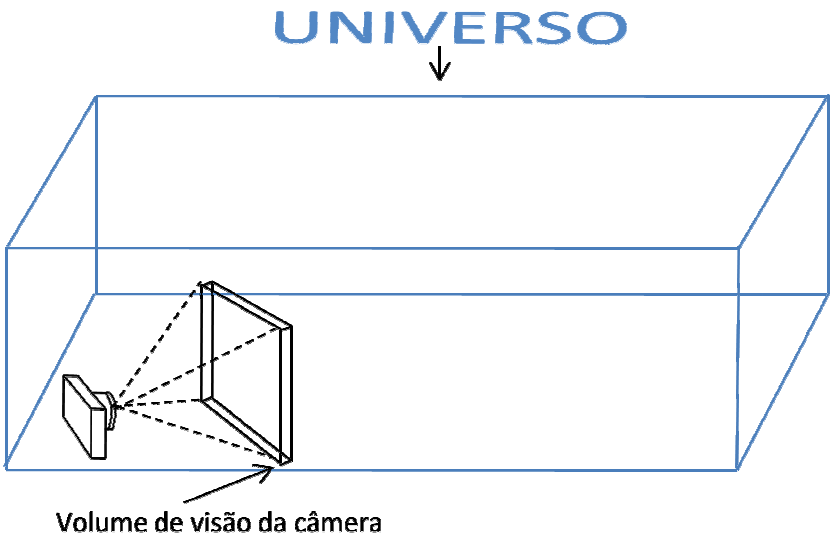

Figura 5 - Estado inicial do universo

Neste estado, o universo virtual está preparado para carregar os modelos tridimensionais a serem prototipados. Quando a importação é realizada, o objeto será carregado como um *Shape3D*. Outras informações associadas ao comportamento do objeto STL são necessárias, pois é preciso que este realize algumas tarefas. Foi criada uma classe denominada *ObjetoSTL* para encapsular todas as transformações essenciais ao objeto dentro do universo, como: mover, girar, definir a cor, alterar a escala, etc. A Figura (6) exemplifica como ficaria o universo após carregar um cilindro.

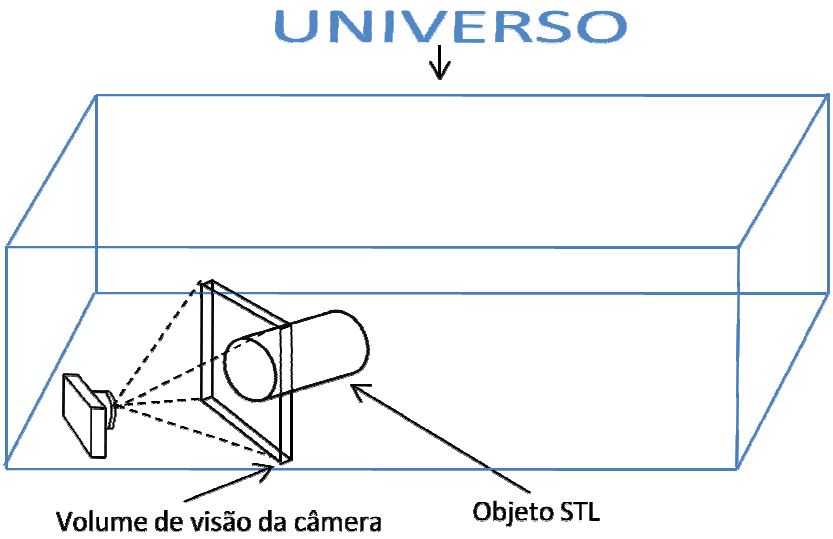

Figura 6 - Universo com um objeto STL carregado.

Quando o universo se encontra com um arquivo tridimensional carregado, o seu grafo de cena fica da seguinte maneira:

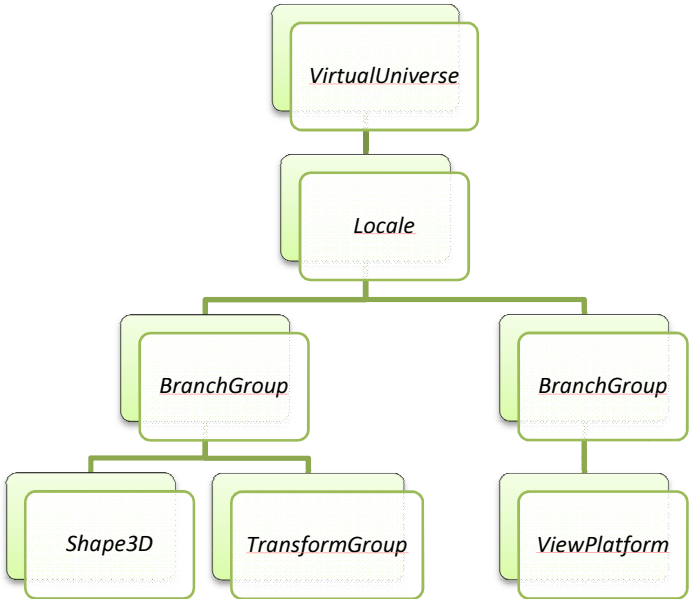

Figura 7 - Grafo de cena.

É importante ressaltar que o formato STL não possui definição de escala. Esse é um problema a ser tratado pelo sistema, pois não se sabe quais as dimensões projetadas pelo usuário em seu modelo CAD. A solução proposta para este problema é utilizar o *Locale* como referencial. O *Locale* armazena informações de coordenadas; um exemplo de sua utilização seria em uma aplicação que necessitasse representar partes da nossa galáxia. Poderíamos definir um *Locale* para representar o sistema solar, com os planetas orbitando em torno do Sol, enquanto outro *Locale* seria utilizado para representar uma casa dentro do planeta terra. Com este exemplo fica clara a necessidade de dois *Locales*: um para representar as geometrias do planeta terra com alguns metros de precisão e outro para representar o sistema solar com uma visão muito mais ampla (Selman, 2002).

Associando-se um determinado objeto a um *Locale*, é possível definir as dimensões que este objeto terá ao ser renderizado. Desta forma, é possível determinar quais as dimensões do objeto STL em relação ao universo virtual. O *TransformGroup e BranchGroup* auxiliam neste processo. Enquanto o *BranchGroup* vai agrupar todos os objetos pertencentes a um *Locale*, o *TransformGroup* armazena as informações de rotação, translação e escala desses objetos.

2.2. Interface Homem Máquina

A interface foi desenvolvida com o intuito de facilitar a prototipagem para o usuário final. Há três funções básicas a serem executadas pelo software: abrir, fatiar e imprimir. Estas funções podem ser observadas no canto superior esquerdo da Fig. (8).

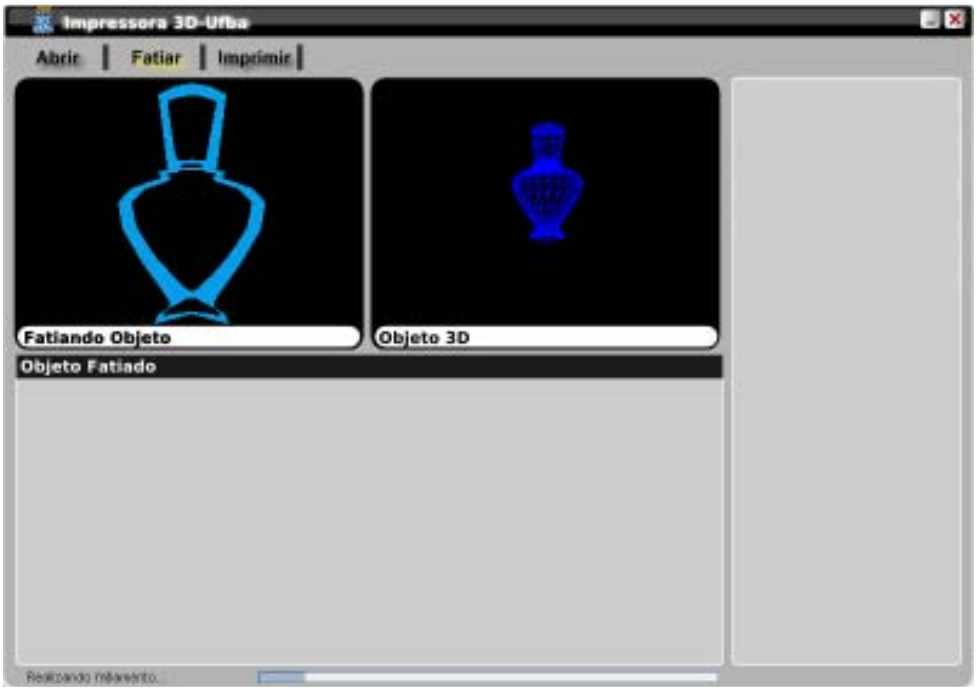

Figura 8 - Interface gráfica do software.

Durante a inicialização do software, o universo virtual é criado com duas câmeras. Uma será responsável pelo fatiamento e a outra apenas mostra para o usuário o objeto que foi carregado. Inicialmente o universo virtual não contém nenhum objeto 3D. Para carregar o arquivo STL o usuário deve clicar em Abrir e escolher o arquivo; em seguida o software transformará este arquivo em um objeto 3D (*Shape3D*) e o demonstrará para o usuário.

A visão de cada uma das duas câmeras é apresentada nos painéis denominados: *Objeto 3D* e *Fatiando Objeto*. O painel *Objeto 3D* apenas mostra o objeto carregado, permitindo que o usuário rotacione com o mouse e tenha uma noção da forma física do protótipo a ser construído. O painel denominado *Fatiando Objeto* é utilizado para demonstrar o fatiamento em tempo real.

No momento em que o usuário escolhe o arquivo STL, o objeto (*Shape3D*) é posicionado exatamente no ponto inicial do volume de visão da câmera. Pelo fluxo definido para o software, sua próxima ação possível é iniciar o fatiamento através do botão "*Fatiar*".

2.3. Fatiamento

Neste momento, através da classe *Transform3D*, aplica-se um movimento retilíneo e constante sobre o objeto em direção a câmera. Esta passará a visualizar apenas camadas do objeto, pois tem um volume de visão bastante reduzido. As fatias do objeto são visualizadas à medida que o objeto vai atravessando o volume de visão da câmera. A Figura (8) exemplifica como ocorreria o fatiamento de um cubo.

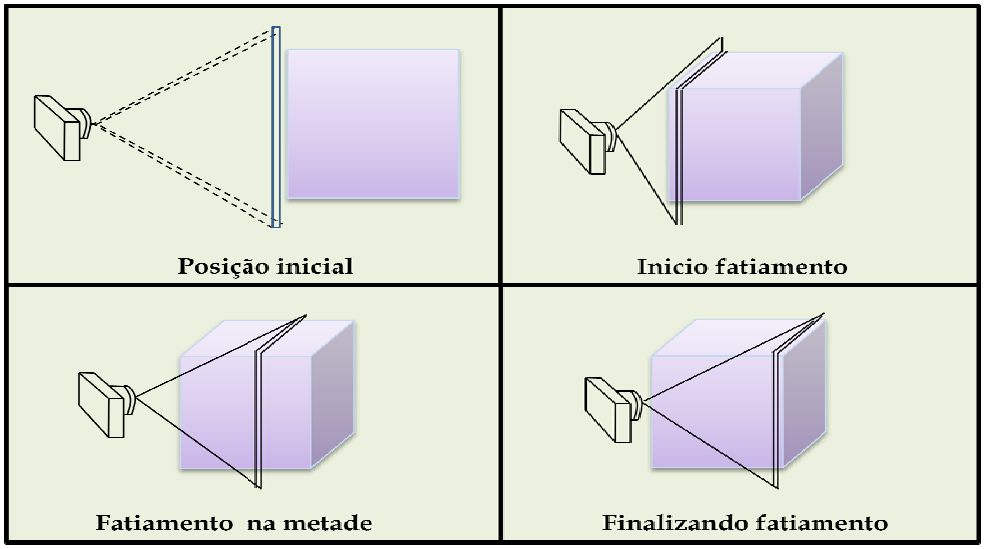

Figura 9 - Processo de fatiamento.

Para gerar as diversas fatias de um determinado objeto tridimensional, basta fazer com que este atravesse completamente o volume de visão da câmera. A quantidade de fatias obtidas é inversamente proporcional ao incremento de translação do objeto, ou seja, quanto maior for o deslocamento do sólido, menor a quantidade de fatias geradas e, consequentemente, menor a quantidade de detalhes obtidas deste objeto. Desta forma, a qualidade do fatiamento pode ser facilmente configurada através da definição do incremento de translação do objeto.

A cada vez que a imagem da câmera for alterada, esta deve capturar a nova imagem e salvá-la como uma fatia. Neste momento, estamos convertendo a informação tridimensional (sólido obtido a partir do arquivo STL) em bidimensional (imagem obtida a partir da renderização da câmera).

Com esta proposta, segundo a classificação de (Dragomatz and Mann, 1997), temos um modelo de fatiamento baseado em pixels. Cada fatia obtida do objeto é representada por uma imagem. As principais desvantagens associadas a este modelo estão ligadas ao consumo excessivo de memória, pois as fatias possuíram a informação de cada ponto da imagem. Em compensação, a partir das próximas etapas do processo é possível utilizar diversos algoritmos voltados para o processamento de imagens.

2.4. Planejamento da rota e Prototipagem

Após abrir e fatiar o arquivo STL, a próxima ação a ser realizada pelo usuário do sistema é solicitar a impressão do objeto clicando em Imprimir. Quando esta função for executada o software percorrerá todas as fatias, realizando o planejamento da rota para cada uma delas. Inicialmente foi implementado um algoritmo que percorre cada fatia linha a linha, identificando as linhas que necessitam de deposição.

As linhas são passadas para o software embarcado através de comunicação serial. Para viabilizar a comunicação entre o software PC e o software embarcado foi definido e implementado um protocolo de comunicação. O software embarcado é capaz de realizar um comando por vez. De forma resumida, estes comandos podem ser: desenhar uma linha entre dois pontos x e y, mover o efetuador até um determinado ponto, mover o eixo Z, ativar e desativar extrusor.

Todas as imagens possuem duas cores possíveis representando os pontos vazios e os pontos de deposição. Para identificar estes pontos é necessário percorrer a imagem lendo a cor de cada pixel.

A estratégia de planejamento linha a linha codificada não é considerada uma solução elegante. A literatura indica que sejam geradas rotas de deposição continua. Segundo Ramaswani (1997), os caminhos de deposição podem ser geometricamente classificados em espiral ou zigzag. No espiral a deposição é gerada com uma série de contornos que são paralelos às bordas da fatia. No preenchimento em zigzag a rota segue uma direção fixa e são traçadas retas paralelas interligadas, gerando um caminho contínuo em zigzag. A partir destas duas formas básicas de preenchimento é possível gerar outras estratégias mais elaboradas.

Estas técnicas de planejamento são originárias de máquinas fresadoras. A diferença básica na sua aplicação à prototipagem rápida é que as rotas são geradas para os locais onde deve haver deposição de material, enquanto nas máquinas fresadoras a rota é planejada nos locais onde deve haver a subtração do material.

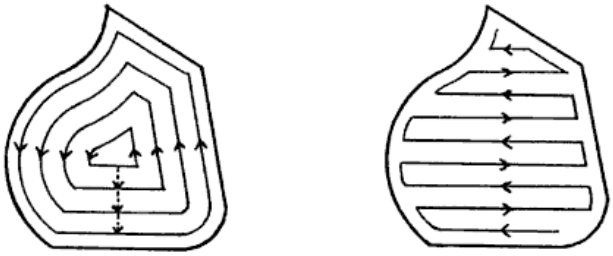

Figura 10 - - Tipos de rotas de preenchimento (Held, 1991).

Estes dois algoritmos estão sendo implementados e serão incorporados a este trabalho. A proposta é aliar estas duas opções de planejamento ao novo hardware que está em processo de montagem. Com isto, espera-se aumentar a precisão da máquina e a qualidade final dos protótipos produzidos. Até então, os principais grupos de pesquisa voltados para a prototipagem rápida (RepRap.org, FabAtHome.org) possuem apenas o algoritmo zigzag. Portanto a implementação do algoritmo espiral representa uma inovação na área.

2.5. Compatibilidade com trabalhos correlatos

É possível integrar este sistema com *hardwares* propostos por outros grupos de pesquisa. Para viabilizar tal integração, existe um projeto denominado Replicat.org que disponibiliza um *software* chamado ReplicatorG.

O ReplicatorG é um *software* capaz de interpretar arquivos com código G gerados por sistemas CAM e repassá-los para uma máquina de prototipagem. Foi desenvolvido originalmente para ser utilizado em máquinas RepRap, porém, atualmente ele é utilizado em outros projetos. O MakerBot, por exemplo, utiliza o ReplicatorG e foca suas pesquisas apenas no desenvolvimento do *hardware*.

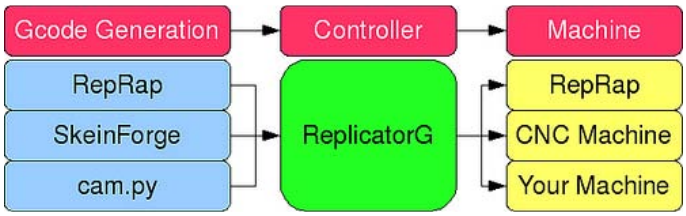

Figura 11 – Processo de prototipagem utilizando o ReplicatorG (Replicat, 2009).

A Figura (11) mostra as três etapas do processo de prototipagem. É possível perceber que o ReplicatorG realiza a integração entre os sistemas CAM e máquinas de prototipagem rápida.

O sistema apresentado neste artigo se encaixa na etapa denominada "*Gcode Generation*". A diferença na utilização do sistema é que ao se clicar no botão imprimir, ao invés de se iniciar o processo de prototipagem propriamente dito, é gerado no computador um arquivo seguindo o padrão G estabelecido pelo Replicat.org. Em seguida o usuário deve iniciar o ReplicatorG e carregar o arquivo gerado anteriormente. Desse ponto em diante, o ReplicatorG passa a controlar a máquina de prototipagem rápida. Desse modo, pode ser feita integração do sistema proposto neste trabalho com qualquer máquina que seja compatível com o ReplicatorG.

3. TESTES E VALIDAÇÕES

De forma sucinta, o protótipo do hardware possui as seguintes características: a mesa X-Y utiliza motores de corrente contínua que giram sistemas de correia dentada e engrenagem para a movimentação prismática dos blocos moveis sobre guias cilíndricas de aço inox. A movimentação da mesa em Z e o bombeamento de material era feita por motores de passo acionando fusos e sistemas de guias similares ao da mesa X-Y. O sistema de injeção utiliza uma seringa plástica onde é também é armazenado o material construtivo (silicone acético). Utilizou-se drivers transistorizados para amplificação de corrente dos motores. O sensoriamento de posição da mesa X-Y utiliza encoderes tipo fita reutilizado de impressoras. A medição de posição em Z e do sistema de bombeamento de material foi feito através da contagem de pulsos dos motores de passo. Sensores eletromecânicos foram utilizados para a sinalização de final de curso em todos os sistemas de movimentação. Diversos experimentos foram realizados com o software atuando em conjunto com o hardware e alguns protótipos foram gerados pela máquina.

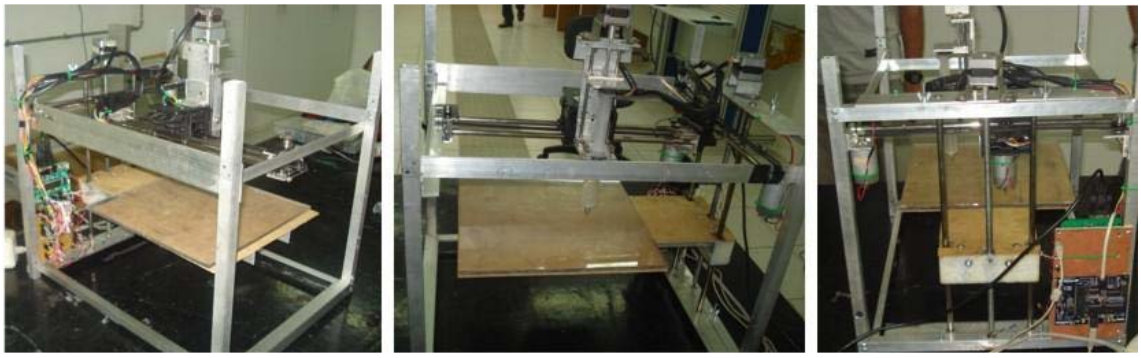

Figura 12 - Fotos do protótipo construído.

Os resultados dos experimentos foram satisfatórios, porém este ainda é um trabalho em andamento e a proposta é que diversos requisitos sejam melhorados tanto no hardware, quanto no software. O objetivo é incrementar a qualidade dos protótipos produzidos e aliar a um baixo custo de produção do hardware.

Nestes primeiros ensaios foi utilizado um bico de deposição com espessura de saída de 0,9mm e a altura das camadas foi de aproximadamente 1mm. A Figura (13) ilustra os primeiros protótipos produzidos pela máquina.

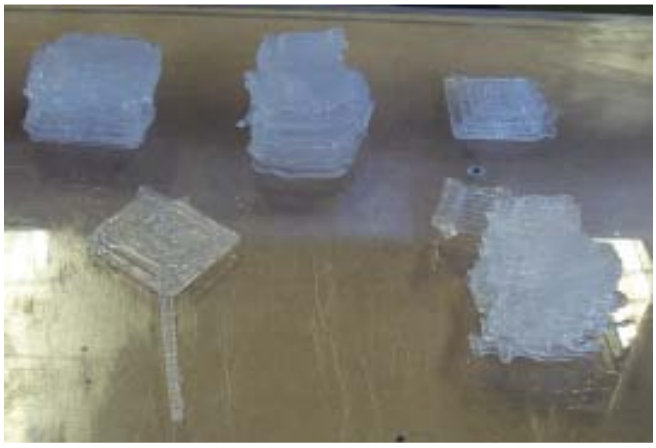

Figura 13 - Protótipos produzidos.

Desta forma foi possível realizar o teste em um sistema completo de prototipagem rápida. Todo o processo de prototipagem foi controlado pelo software desenvolvido neste trabalho.

4. CONSIDERAÇÕES FINAIS

Este trabalho apresentou um sistema CAM capaz de atuar em máquinas de prototipagem rápida. A partir da primeira etapa do desenvolvimento do sistema e dos testes em um protótipo funcional, foi detectada a possibilidade de incrementar o trabalho. Desse modo, estão em andamento reformulações tanto no hardware quanto no software, visando melhorar a qualidade dos protótipos gerados. A principal meta, nesta nova etapa, é alcançar camadas de espessura equivalentes a 0.15 mm.

A proposta de melhoria para o software aplicou algoritmos de processamento de imagens para aperfeiçoar a geração das trajetórias das camadas. Duas opções de algoritmos de preenchimento (ZigZag e Espiral) foram codificadas visando gerar uma rota de deposição contínua, em detrimento da deposição linha a linha utilizada anteriormente. Estas novas funcionalidades permitirão aumentar consideravelmente a qualidade do protótipo final, além de reduzir o tempo de prototipagem. Estes novos algoritmos inicialmente foram testados em ambiente simulado, utilizando a geração de código G. Assim que a nova versão do hardware estiver concluída, novos testes serão realizados no sistema.

5. REFERÊNCIAS

Dragomatz, D. and Mann, S., 1997, "A Classified Bibliography of Literature on NC Tool Path Generation", Computer-Aided Design, Vol. 29, pp. 239-247.

Fab@Home, 2009, “The open-source personal fabricator project”. 28 Nov. 2009, <http://fabathome.org>

Held, M., 1991, "On the Computational Geometry of Pocket Machining", Springer-Verlag, Berlin, Germany.

Koc, B., 2001, "Computional Geometric Analysys and Planning for 3D Rapid Prototyping Processes", Ph.D. thesis, North Carolina State University, North Carolina.

Kulkarni, P., Marsan, A. and Dutta, D., 2000, "A Review of Process Planning Techniques in Layered Manufacturing", Rapid Prototyping Journal, Vol. 6, No. 1, pp. 18-35.
MakerBot, 2009, “Robots that make things”. 28 Nov. 2009, <http://www.makerbot.com>
Ramaswami, K., 1997, "Process Planning for Shape Deposition Manufacturing", Ph.D. thesis, Department of Mechanical Engineering, Stanford University, Stanford, California.
Replicat, 2009, “A simple, open source machine controller”. 28 Nov. 2009, < http://replicat.org>.
RepRap, 2009, “Replicating rapid prototyper”. 28 Nov. 2009, <http://reprap.org>
Selman, D., 2002, "Java 3D Programming", Ed. Manning Publications, GreenWich, CT, USA, 400 p.
Sun, 1999. "Java 3D API Specification". 28 Nov. 2009, <http://java.sun.com/javase/technologies/desktop/java3d/forDevelopers/j3dguide/j3dTOC.doc.html>.

6. DIREITOS AUTORAIS