

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

Santiago Viertel

AVALIAÇÃO DA PREENSÃO DE UM ROBÔ VIRTUAL

Marcelo da Silva Hounsell
Orientador

Joinville – SC

2008

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

Santiago Viertel

AVALIAÇÃO DA PREENSÃO DE UM ROBÔ VIRTUAL

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Marcelo da Silva Hounsell
Orientador

Joinville, julho de 2008

AVALIAÇÃO DA PREENSÃO DE UM ROBÔ VIRTUAL

Santiago Viertel

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação do CCT/UDESC.

Banca Examinadora

Marcelo da Silva Hounsell (orientador)
PhD.

Alessandro Dorow
B.Sc.

Carlos Norberto Vetorazzi Jr.
M.Eng.

Roberto Silvio Ubertino Rosso Jr.
PhD.

Izabel Zattar
M.Eng.

AGRADECIMENTOS

Agradeço primeiramente a Deus que me deu energia durante cada dia de dedicação a este trabalho de Conclusão de Curso.

Agradeço aos meus familiares que me deram força para continuar a pesquisa através de incentivos verbais, compreensão e carinho oferecendo-me tudo o que é psicologicamente primordial para a conclusão do trabalho.

Agradeço aos amigos que sempre estiveram ao meu lado me apoiando nos momentos decisivos necessários para a aceitação deste trabalho pela academia.

Agradeço também ao meu professor orientador e aos membros da banca que avaliaram este trabalho e me ajudaram a deixá-lo no conceito que ele se encontra atualmente.

RESUMO

Um simulador de robô articulado está sendo desenvolvido pelo grupo de pesquisa Laboratório de Realidade Virtual Aplicada (LARVA). Este simulador está sendo implementado com o intuito de ser didático e faz uso da ferramenta de Realidade Virtual VRML integrada a linguagem Java. Devido ao uso destas duas tecnologias, o projeto pode ser executado em um ambiente *web*. Este simulador possuía a ausência de avaliação da preensão (pega) de objetos até este trabalho.

São apresentadas neste trabalho algumas pesquisas que estão sendo realizadas sobre o problema da preensão de objetos feita por uma garra robótica. Com a análise destas é mostrada uma proposta de solução levando em consideração alguns raciocínios encontrados nos trabalhos estudados. Como existem vários tipos de tratamentos para este problema, que envolvem física ou detecção de colisão, por exemplo, optou-se por avaliar as diversas situações através de um conjunto de raciocínios geométricos que gera uma resposta demandando pouco processamento. Esta abordagem foi encontrada espalhada em diversos trabalhos, mas não de maneira estruturada.

Como a aplicação é um simulador gráfico 3-D, buscou-se uma solução eficiente e conceitualmente simples. A solução é apresentada em dois blocos de raciocínios: um que identifica situações que garantidamente não há condição de preensão e; outro que confirma as condições para a preensão. Estes raciocínios são apresentados e detalhados neste trabalho para os objetos rígidos primitivos como cilindros, paralelepípedos e esferas. O primeiro bloco de raciocínios é abrangente e não requerem detalhes geométricos dos objetos. O segundo bloco utiliza o conceito de geometria e tratamento de folga de maneira que seja permitida ao usuário flexibilidade para manipulação. Testes foram feitos na solução implementada e foram confirmados desempenho computacional e precisão.

ABSTRACT

A simulator of articulated robot has being developed at the Laboratory of Applied Virtual Reality (LARVA – LAboratório de Realidade Virtual Aplicada) research group. This simulator is being implemented with didactics purposes and uses the VRML Virtual Reality tool integrated to the Java language. Thanks to the use of these two technologies, the project can be executed in a web environment. This simulator lacked the evaluation of object grasping until this work.

In this work some research that has being realized about the problem of object grasping made by a robotic grip are presented. With the analysis of these, it is shown a solution proposal considering some reasonings found in the studied works. There are many approaches for this problem, involving physics or collision detection, for instance, so it was opted to evaluate the miscellaneous situations through a set of geometric reasonings that generate a reply that demands little processing time. This approach was found scattered in many works, but not in a structured way.

As the application is a 3-D graphic simulator, it was aimed for an efficient and conceptually simple solution. The solution is presented in two blocks of reasonings: one that identifies situations that guarantee that there isn't a grasp condition and; another one that confirms the conditions for grasping. These reasonings are presented and detailed in this work for primitive rigid objects such as cylinders, blocks and spheres. The former block of reasonings is in such an abstract level that they do not require geometric details of the objects. The second block uses the concept of geometric and approach lenience in such a way that a user is allowed a goal level of flexibility for manipulation. Tests were made the implemented solution are confirmed computational performance and accuracy.

LISTA DE ILUSTRAÇÕES

Figura 1 – Simulador do robô ER-4PC em ambiente web.....	1
Figura 2 – Algumas situações de preensão	2
Figura 3 – Disposições que indicam colisão, porém não representam a preensão.....	4
Figura 4 – Intersecção entre triângulos	5
Figura 5 – Robô de coordenadas cartesianas	10
Figura 6 – Robô de coordenadas cilíndricas	11
Figura 7 – Robô de coordenadas esféricas.....	11
Figura 8 – Robô do tipo SCARA	12
Figura 9 – Robô articulado.....	12
Figura 10 – Pinça Parallel Jaw Gripper.....	14
Figura 11 – Garra Barret Hand.....	14
Figura 12 – Garra DLR Hand.....	15
Figura 13 – Garra Robonaut Hand	16
Figura 14 – Partes da garra Rutgers Hand.....	16
Figura 15 – Anatomia do robô Scorbob ER-4PC.....	17
Figura 16 – Volume de trabalho ocupado pelo robô Scorbob	18
Figura 17 – Poliedro geral.....	21

Figura 18 – Modo de representação por B-Rep	22
Figura 19 – Cone de atrito formado no contato de corpos	23
Figura 20 – Criação do polígono de intersecção	26
Figura 21 – Ajuste dos pontos para minimização de torque e atrito	27
Figura 22 – Soma dos cones de atrito sendo feita em um vértice côncavo	28
Figura 23 – Espaço de trabalho do robô	30
Figura 24 – As três formas de serem pegos os objetos pela pinça	31
Figura 25 – Superfície dos objetos e suas representações angulares	33
Figura 26 – Região possível formada pelos pontos de contato	34
Figura 27 – Objeto complexo modelado em primitivas	35
Figura 28 – Os dois tipos de disposição da garra Barrett Hand	35
Figura 29 – Pega de caixas feita pela garra	36
Figura 30 – Pega de esferas feita pela garra	37
Figura 31 – Pega de cilindros pela lateral feita pela garra	37
Figura 32 – Pega de cilindros pela extremidade feita pela garra	38
Figura 33 – Obstáculos sendo avaliados no momento da tentativa de preensão	39
Figura 34 – Simulador do robô Scorbobot ER-4PC	45
Figura 35 – Comparativo entre a interface antiga e projetada	46
Figura 36 – Entradas e saídas de dados do algoritmo proposto	47
Figura 37 – Dimensões da pinça Parallel Jaw Gripper	49
Figura 38 – Vetores de orientação da pinça Parallel Jaw Gripper	50
Figura 39 – Pontos de referência para a preensão	50
Figura 40 – Corte lateral do cilindro	54
Figura 41 – Esquema dos testes de Exclusão	57

Figura 42 – Exclusão por proximidade	58
Figura 43 – Área traseira inacessível pelo robô	59
Figura 44 – Exclusão por posição fora do volume de trabalho	60
Figura 45 – Divisão das análises de Folga	66
Figura 46 – Folga de Abordagem num quadrado 2-D	67
Figura 47 – Raciocínio Geométrico para Paralelepípedos	68
Figura 48 – Padrão para a pega de Cilindros por cima	69
Figura 49 – Padrão para a pega de cilindros pela lateral.....	69
Figura 50 – Representação da Folga Geométrica Interna ao Objeto.....	72
Figura 51 – Máquina de Estados do botão de teste de preensão	74
Figura 52 – Mudança de sistema de coordenadas	76
Figura 53 – Detalhamento da classe Garra.....	78
Figura 54 – Detalhamento da classe TrataPreensao.....	80
Figura 55 – Relação entre os métodos da classe TrataPreensao	80
Figura 56 – Limites da mesa	86
Figura 57 – Colisão errônea ao pegar um objeto	87
Figura 58 – Bug na criação das OBBs da garra	89
Figura 59 – Diagrama de Seqüência – Pegar Objeto	90
Figura 60 – Diagrama de Seqüência – Liberar Objeto.....	91
Figura 61 – Diferentes estados do botão de teste de Preensão.....	91
Figura 62 – Teste Funcional – Bounding Box de trabalho	93
Figura 63 – Teste Funcional – Fatia Traseira.....	94
Figura 64 – Teste Funcional – Proximidade	94
Figura 65 – Teste Funcional – Abordagem para Paralelepípedos.....	95

Figura 66 – Teste Funcional – Abordagem para Cilindros	96
Figura 67 – Teste Funcional – Folga Geométrica paralela ao Apg.....	97
Figura 68 – Teste Funcional – Folga Geométrica paralela ao Upg.....	98
Figura 69 – Teste Funcional – Folga Geométrica paralela ao Elg.....	99
Figura 70 – Reprovação errônea do algoritmo de Raciocínios Geométricos.....	100

LISTA DE TABELAS

Tabela 1 – Especificações técnicas do Scrobot ER-4PC.....	18
Tabela 2 – Coeficientes de atrito estático.....	24
Tabela 3 – Valores limites constantes da geometria da garra	48
Tabela 4 – Eixo de coordenadas da pinça	49
Tabela 5 – Dados de representação específicos dos objetos	52
Tabela 6 – As diferentes situações de preensão	62
Tabela 7 – Similaridade entre as situações.....	63
Tabela 8 – Limites para geração de Objetos	86

LISTA DE ABREVIATURAS

AABB	Axis-Aligned Bounding Box
CG	Centro de Gravidade
DH	Denavit-Hartenberg
DOF	Degrees Of Freedom
EAI	External Authoring Interface
FG	Fechar Garra
GUI	Graphical User Interface
HTML	HyperText Markup Language
IK	Inverse Kinematics
LARVA	Laboratório de Realidade Virtual Aplicada
NASA	National Aeronautics and Space Administration
OBB	Oriented Bounding Box
PFP	Ponto Final da Pinça
PRP	Ponto de Referência do Punho
SCARA	Selective Compliant Assembly Robot Arm
VM	Virtual Machine
VRML	Virtual Reality Modeling Language

SUMÁRIO

1. INTRODUÇÃO	1
1.1. PREENSÃO PELA DETECÇÃO DE COLISÃO	4
1.2. PREENSÃO PELA SIMULAÇÃO FÍSICA	4
1.3. PREENSÃO POR RACIOCÍNIOS GEOMÉTRICOS	6
1.4. OBJETIVOS	7
1.4.1. Objetivo Geral	7
1.4.2. Objetivos Específicos	7
1.4.3. Escopo	8
1.5. ESTRUTURA DO TRABALHO DE CONCLUSÃO DE CURSO	8
2. FUNDAMENTAÇÃO TEÓRICA	9
2.1. ROBÓTICA	9
2.1.1. Classificação dos Robôs	10
2.1.2. Tipos de Garras.....	13
2.1.2.1. <i>Parallel Jaw Gripper</i>	13
2.1.2.2. <i>Barret Hand</i>	14
2.1.2.3. <i>DLR Hand</i>	15
2.1.2.4. <i>Robonaut Hand</i>	15
2.1.2.5. <i>Rutgers Hand</i>	16
2.1.3. Robô Scorbobot ER-4PC	17
2.2. MODELAGEM GEOMÉTRICA	19
2.2.1. Representação por Fronteiras (<i>Boundary Representation</i>).....	19
2.3. FÍSICA APLICADA A OBJETOS RÍGIDOS	22
2.3. Modelos de Contato	22
3. TRABALHOS RELACIONADOS	25
3.1. UTILIZAÇÃO DO CENTRO DE MASSA PARA EQUILÍBRIO DA PEGA	25
3.2. USO DE HEURÍSTICAS PARA ELIMINAÇÃO DE CASOS INVÁLIDOS	28
3.3. PROPAGAÇÃO DE CONES DE ATRITO EM OBJETOS RÍGIDOS	31
3.4. RACIOCÍNIOS APLICADOS A OBJETOS PRIMITIVOS	34
3.5. CARACTERÍSTICAS RELEVANTES AO TEMA PROPOSTO.....	39

4. SOLUÇÃO	42
4.1. A INTERFACE.....	43
4.2. INFORMAÇÕES NECESSÁRIAS.....	47
4.2.1. Dados da Garra.....	48
4.2.2. Dados dos Objetos	51
4.2.3. Dados do Ambiente	55
4.3. ESTRUTURA DA SOLUÇÃO	55
4.3.1. Etapa da Exclusão.....	56
4.3.2. Etapa de Confirmação.....	61
4.4. POSSÍVEIS SITUAÇÕES.....	61
4.5. RACIOCÍNIOS GEOMÉTRICOS	63
4.5.1. Folga	64
4.5.1.1. Folga de Abordagem	66
4.5.1.2. Folga Geométrica	70
5. RESULTADOS.....	73
5.1. IMPLEMENTAÇÃO	73
5.1.1. Mudanças no Projeto.....	73
5.1.2. Classes Incluídas no Simulador.....	77
5.1.3. Códigos Modificados	85
5.1.4. Como Utilizar a Avaliação de Preensão.....	90
5.2. TESTES FUNCIONAIS	92
5.2.1. Testes com a Etapa de Exclusão	92
5.2.2. Testes com a etapa de Confirmação	95
5.2.2.1. Testes de Folga de Abordagem.....	95
5.2.2.2. Testes de Folga Geométrica	96
5.2.3. Teste de Performance	100
5.3. CONSIDERAÇÕES FINAIS.....	101
6. CONCLUSÃO	103
6.1. CONSIDERAÇÕES FINAIS.....	103
6.2. TRABALHOS FUTUROS	105
REFERÊNCIAS	108
ANEXOS.....	112

1. INTRODUÇÃO

Simuladores virtuais podem possuir propósitos didáticos de forma que seus usuários sejam aprendizes de manipulação da estrutura robótica simulada. Usuários iniciantes ou inexperientes de robôs podem fazer uso de simuladores para aprender a manipular a ferramenta real. Desta forma, podem-se evitar possíveis acidentes e, conseqüentemente, danos à máquina causados durante o processo de aprendizagem, como o caso de colisão entre a estrutura do robô real e outro objeto qualquer.

O uso da Realidade Virtual está presente em diversas áreas da tecnologia da informação. Aplicações gráficas como simuladores de ambientes industriais são apenas um exemplo que referencia a importância dessa tecnologia para os *softwares* atuais. Através do Laboratório de Realidade Virtual Aplicada (LARVA, 2007), foi desenvolvido um protótipo do robô Scorbót ER-4PC fazendo uso de Realidade Virtual. Este protótipo, mostrado na Figura 1, pode ser executado em um ambiente *web* através do *browser* do usuário possibilitando, assim, o seu uso em qualquer lugar e a qualquer hora bastando instalar o *plug-in* apropriado. Atualmente é possível criar e verificar programas no aplicativo com ações que o robô virtual é capaz de realizar, como a sua movimentação e comandos de abrir e fechar pinça.

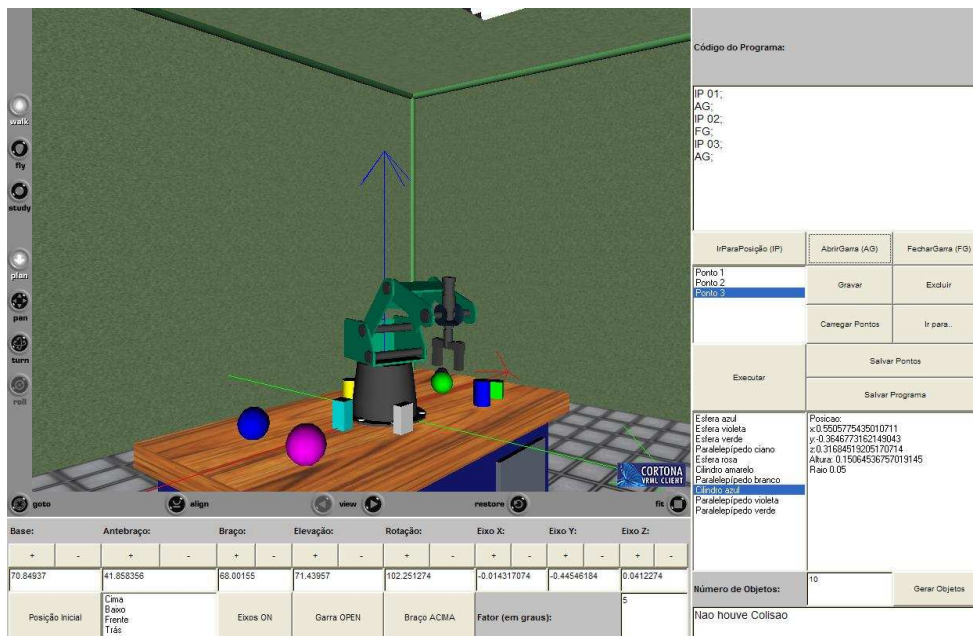


Figura 1 – Simulador do robô ER-4PC em ambiente *web*
(LARVA, 2007)

Após a criação do protótipo virtual baseado na anatomia do robô real (REDEL e HOUNSELL, 2004), foi adicionada a ele a implementação da cinemática inversa desenvolvida e analisada por Zwirtes (2004). Mais uma funcionalidade, sendo esta a de adição de objetos aleatórios no ambiente robótico, também foi incluída no simulador por Santos et al. (2007b). Logo após foi adicionada na última atualização deste simulador de robô a funcionalidade de detecção de colisão no ambiente virtual (SANTOS, 2007a).

O simulador do robô está disponível na *web* e, apesar de estarem implementadas as funcionalidades citadas, existiam ainda algumas limitações para que o simulador pudesse ser utilizado de forma didática. Uma das tarefas que o robô físico é capaz de desempenhar consiste na preensão (pega) de pequenos objetos, estes contendo pouco peso e acessibilidade ao tipo de garra instalada no robô. Um problema existente no simulador até o presente trabalho era a falta de um tratamento para a preensão de objetos feita pela garra do robô virtual, como se pode perceber na Figura 2 (a). Este problema é conceituado e analisado neste trabalho.

A preensão é uma das maneiras primárias criada para o robô interagir com objetos inseridos em seu ambiente. Robôs de linha de montagem fazem uso de efetadores terminais como pinças ou ventosas de sucção para realizar tarefas de “pegar e colocar” com um alto grau de confiabilidade (MILLER, 2001, p. 01). Desta maneira, o problema citado pode ser denominado como problema da preensão. Este tratamento pode ser estendido a qualquer tipo de objeto geométrico e a qualquer tipo de garra para robô, assim como é ilustrado na Figura 2. Devido à infinita quantidade de situações diferentes com garras e objetos distintos, podem existir diversas maneiras de pegar objetos, proporcionando equilíbrio durante a pega, como mostrado na Figura 2 (b), ou não, como mostrado na Figura 2 (a). Desta maneira, pode-se afirmar que o problema da preensão não possui uma solução trivial.

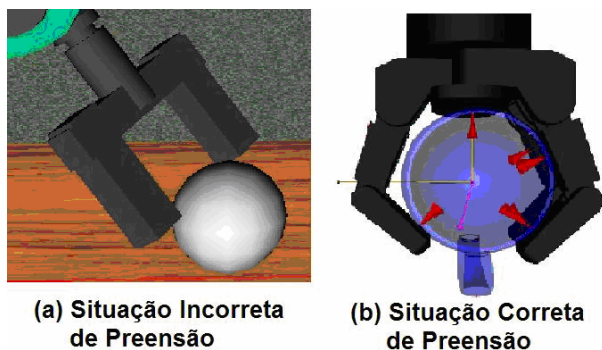


Figura 2 – Algumas situações de preensão
(MILLER, 2001)

Como se pode visualizar na Figura 2, tanto a geometria da garra como a geometria do objeto impõem grandes influências nas análises de preensão. A anatomia de cada garra deve ser levada em consideração para tratar a preensão. Quantidade de dedos, tamanho dos dedos, dimensões de suas fronteiras geométricas, quantidade de DOF's, entre outros, são informações necessárias para que seja tratada a ação de preensão feita pela garra.

A anatomia do objeto também necessita ser definida, pois se sua geometria variar, também mudará a forma como a garra o pegará. Todos estes dados de geometria da garra e do objeto são necessários para que a pega de um objeto seja efetuada, garantindo um equilíbrio maior no ato.

Antes de buscar uma solução para o problema é importante analisá-lo a fim de obter um aplicativo eficiente. Como esse problema teve origem durante a implementação de um simulador, e simuladores buscam ser semelhantes ao ambiente real, aspira-se por ter a execução do aplicativo de maneira satisfatória e rápida. Para isso, é necessário considerar a limitação de processamento gráfico presente na máquina a qual a aplicação está sendo submetida para execução.

Para que a preensão fosse feita de maneira eficiente e rápida em um computador com recursos gráficos e capacidade de processamento limitantes, foi levantada uma série de raciocínios geométricos que garantem a preensão de objetos rígidos dispostos no ambiente. Como consequência disto, os tipos de geometria, as orientações, as localizações e as dimensões, tanto da garra como do objeto a ser pego, necessitavam ser definidos para que seja avaliada a ação. Percebe-se a importância destas informações ao tentar utilizar o simulador para segurar um objeto como o cubo ou o cilindro, por exemplo. Ao submeter um cubo à análise de preensão, percebe-se a necessidade de informações como altura, largura e comprimento, bem como a orientação espacial do objeto geométrico. Para a geometria primitiva cilindro, no entanto, necessita-se de outros tipos de informações distintas às necessárias para o cubo, como raio da geometria cilíndrica, por exemplo.

Com as geometrias da garra e dos objetos definidas, parte-se para a análise de suas posições. Para isso é usado como referência o centro de gravidade do objeto bem como o ponto final existente na ponta da pinça, sendo o último utilizado nos cálculos da cinemática inversa. A cinemática inversa é um procedimento que determina as variáveis de juntas do robô, ou seja, os ângulos em cada junta baseando-se nas coordenadas cartesianas do seu punho (SCHILLING, 1990).

Baseando-se na análise de trabalhos relacionados, durante a avaliação da preensão, pelo menos, três aspectos foram considerados:

1. Detecção de Colisão;
2. Simulação Física;
3. Raciocínios Geométricos;

Estes três aspectos são inter-relacionados, mas, para efeitos de entendimento, algumas considerações precisavam ser feitas:

1.1. Preensão pela Detecção de Colisão

Quanto ao aspecto de detecção de colisão, estes são cálculos capazes de identificar onde e de que forma duas geometrias se encostam, mas ainda são necessárias outras avaliações para indicar se isto corresponde ou não a uma preensão. A Figura 3 indica duas situações onde colisão não leva a preensão: na primeira (a) a colisão levará à queda do cilindro antes mesmo de a garra fechar-se completamente. Na segunda (b), apesar da colisão, não existe uma forma de fixar a garra com o objeto (até mesmo por questões físicas) e, portanto, a garra não segura o objeto, este acabará escapando da garra.

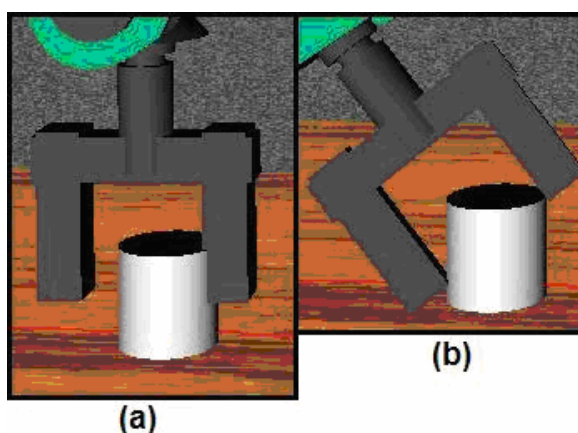


Figura 3 – Disposições que indicam colisão, porém não representam a preensão
(LARVA, 2007)

1.2. Preensão pela Simulação Física

A simulação física responde através das condições reais que ocorrem no momento de realização da pega como o atrito, por exemplo. O atrito é necessário para que os corpos do objeto e da garra interajam e proporcionem uma ação antideslizante no contato entre eles.

Desta situação advém o estudo dos pontos de preensão (relacionado à geometria) e dos cones de atrito (que serão detalhados adiante). Uma vez garantida a pega inicial, a física ainda influencia na avaliação desta em função de restrições que, causam efeitos adversos como torção no objeto, oscilação ou até sobrecarga da garra.

Ao se inserir cálculos físicos na simulação, também se faz necessária a localização dos pontos exatos, dispostos na superfície dos corpos, onde deverão ser inseridos os cones de atrito (tratamento físico). A obtenção destes pontos corresponde à uma etapa do tratamento de colisões denominada determinação da colisão (MÖLLER e HAINES, 2002). Esta etapa localiza o ponto exato onde ocorre a colisão através uma série de testes feitos com duplas de polígonos, sendo um de cada objeto testado. A intersecção entre os polígonos é feita como é mostrada na Figura 4, onde são realizados testes através da intersecção entre pares de arestas (lado esquerdo da figura), entre a face de um e o vértice do outro (parte central da figura) e face com face (parte direita da figura).

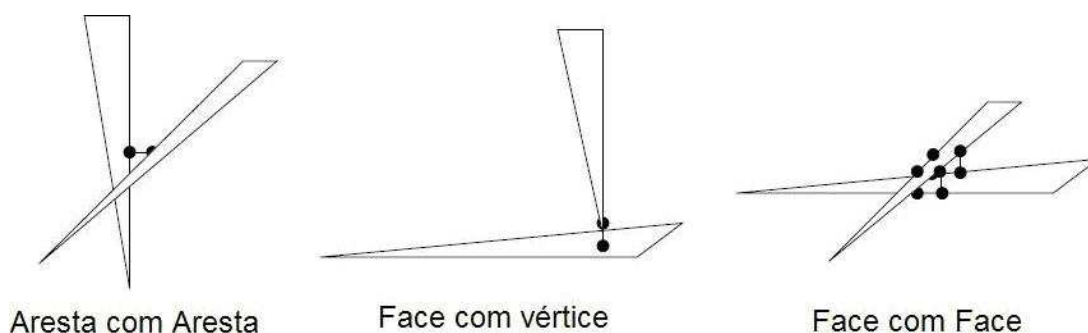


Figura 4 – Intersecção entre triângulos
(MILLER, 2001, p. 40)

Devido ao aumento na quantidade de testes de acordo com a quantidade de triângulos dos dois objetos testados, conclui-se que a adição desta etapa fará com que a busca da solução necessite de mais tempo de processamento. Outro fato a se comentar é que identificar somente um ponto de contato não basta. Para que sejam gerados os cones de atrito proporcionando uma preensão mais eficiente é necessário identificar pelo menos dois pontos.

Mesmo após descobrir os pontos exatos de colisão, é necessário incluir os cones de atrito partindo destes pontos para o interior do objeto “preendido” para que seja utilizada a análise física. Somente após as análises feitas com os cones de atrito, é que será concluída se a preensão é possível de ser feita ou não. Desta maneira, o uso de tratamento físico torna-se inviável para o simulador em questão, visto que todos estes algoritmos estarão executando na

máquina do cliente e com interação entre VRML e a linguagem Java. Como a máquina cliente pode variar em termos de recursos de *hardware*, vê-se a necessidade de aplicar outro tipo de solução para o problema apresentado.

Desta maneira, informações físicas que referenciam a geometria da garra e dos objetos, podem ser utilizadas para encontrar a solução. Outras informações como peso, porém, têm a possibilidade de serem excluídas da análise de preensão, visto que se pode garantir uma boa proximidade com a situação real apenas aplicando restrições geométricas a este tratamento.

Como cálculos físicos representam soluções custosas computacionalmente (SMITH et al., 1999), pois carecem ao atender outras restrições que consideram aspectos da geometria do objeto e da garra, então tem-se o terceiro aspecto que deve ser considerado na avaliação da preensão: os raciocínios geométricos.

1.3. Preensão por Raciocínios Geométricos

Raciocínios geométricos são a representação de condições quaisquer na forma de restrições que envolvam a geometria dos corpos envolvidos (MORTENSON, 2006). Estas condições podem complementar outras análises (como as de física e detecção de colisão) ou até substituí-las, pois são capazes de traduzir em cálculos, soluções aceitáveis para certas situações.

Assim, cálculos físicos podem indicar como corretas as preensões de objetos, com um mínimo de área de contato entre eles, ao passo que raciocínios geométricos são capazes de traduzir a experiência de que a preensão é correta se, pelo menos uma pequena área do objeto, determinada de antemão, é encostada pela garra.

Apesar da evidente perda de precisão e otimização de soluções, o uso de raciocínios geométricos, entretanto, traz duas outras vantagens: são capazes de tornar explícita a heurística do fato e são menos custosos computacionalmente. A dificuldade está, porém, em catalogar as respectivas heurísticas.

A maioria dos trabalhos estudados aborda tema propondo uma solução ótima para a preensão de objetos, com o uso de cálculos físicos ou de outros tratamentos. Apesar de vários dos trabalhos da literatura fazerem uso de cálculos físicos, várias soluções ainda contemplam ou requerem decisões que podem ser obtidas por raciocínios geométricos.

Este trabalho não busca uma solução ótima para o problema da preensão de objetos, mas sim avalia a disposição que resultou da manipulação do usuário sobre o robô e sua garra perante o objeto-alvo a ser pego e, após um processamento, responde se é possível, ou não, acontecer a preensão. Desta forma, vê-se que esta aplicação efetivamente avalia a viabilidade da abordagem que o usuário definiu quando for solicitada uma tarefa de preensão.

É importante que esta funcionalidade esteja incluída no simulador, pois o *software* está sendo desenvolvido também com o propósito de ser utilizado como ambiente de treinamento de programação *off-line* de robôs. Através do simulador, é possível identificar e avaliar tipos de erros produzidos pelo usuário estudante que podem acontecer durante a preensão.

Além de o enfoque do trabalho ser voltado para o ensino, percebe-se que a avaliação de preensão tem sua importância em ambientes industriais. Para estes casos, esta funcionalidade permite o estudo e definição de heurísticas, por parte do usuário, levando à identificação das “melhores práticas” de preensão. Desta forma, várias situações podem ser visualizadas através do simulador a fim de otimizar o programa a ser executado pelo robô no ambiente fabril.

1.4. Objetivos

1.4.1. Objetivo Geral

Desenvolver uma funcionalidade de software, a ser incorporada em um simulador de robô, capaz de avaliar situações de preensão geradas pela manipulação de um usuário.

1.4.2. Objetivos Específicos

- Investigar condições geométricas de preensão para o tipo de garra pinça de dois dedos e objetos propostos;
- Desenvolver uma solução com raciocínios geométricos para a preensão com um robô virtual;
- Avaliar o seu funcionamento para os principais tipos de sólidos geométricos em várias orientações (em pé ou deitado, por exemplo).

1.4.3. Escopo

Para que fosse possível a inclusão da análise de preensão ao simulador, foi necessário que existissem objetos dispostos no ambiente capazes de serem pegos pelo robô virtual. Como o problema foi resolvido apenas com o uso de raciocínios geométricos, busca-se a simplicidade de representação destes objetos. Desta maneira, serão usadas as seguintes geometrias primitivas: cilindros, esferas e paralelepípedos. Estas geometrias serão todas consideradas objetos rígidos com massa homogeneamente distribuída e possuindo a sua representação na técnica de modelagem geométrica denominada *Boundary Representation* (B-rep).

O robô que faz a ação de preensão é o protótipo virtual feito pelo LARVA (2007) que possui 5 DOF e uma garra do tipo pinça de dois dedos (*two-fingers*), mais especificamente a *parallel jaw gripper* (pinça de dedos paralelos).

O simulador tem propósito didático, não com o intuito de encontrar a melhor resposta para a preensão do objeto, mas sim, avaliar se as disposições do robô e do objeto apresentam uma situação válida para preensão. Desta forma, não é automatizada a determinação do caminho percorrido para a pega do objeto, deixando esta tarefa a cargo do usuário aprendiz.

1.5. Estrutura do Trabalho de Conclusão de Curso

Este trabalho de conclusão de curso está disposto em 6 Capítulos.

O Capítulo 1 apresenta a introdução ao trabalho e alguns aspectos importantes a serem tratados no desenvolver.

O Capítulo 2 apresenta um estudo de informações teóricas de robótica, modelagem geométrica e informações físicas importantes para um claro entendimento do trabalho em etapas posteriores.

O Capítulo 3 apresenta alguns trabalhos relacionados com o trabalho que será desenvolvido. Este capítulo fornece informações importantes de modos de soluções adotadas.

O Capítulo 4 apresenta a proposta de solução para o problema apresentado com base nos trabalhos que estão sendo desenvolvidos.

O Capítulo 5 apresenta as considerações sobre a implementação do aplicativo e alguns testes funcionais realizados.

O Capítulo 6 apresenta a conclusão tomada com base em todas as informações apresentadas no trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Serão apresentados alguns conceitos necessários para o entendimento deste trabalho. Inicialmente será mostrada uma introdução sobre a robótica, alguns conceitos, alguns tipos de garras existentes em simuladores e será detalhado o robô Scrobot ER-4PC. Ao final serão explicadas algumas propriedades do material, como atrito.

2.1. Robótica

A robótica é uma ciência ou ramo da tecnologia que estuda o projeto e a construção de máquinas com capacidade de desempenhar tarefas realizadas pelo ser humano (ROMANO, 2002).

Atualmente, as tarefas a serem desenvolvidas em um ambiente industrial exigem cada vez mais eficiência e precisão, muitas vezes colocando a vida humana em risco. Tarefas deste tipo necessitam serem realizadas por dispositivos de robótica. Segundo Romano (2002) existem algumas razões para a utilização de robôs:

- Reduzir custos de produtos, através da minimização da quantidade de pessoas trabalhando e, conseqüentemente, aumentando a produtividade;
- Melhorar as condições de trabalho para os trabalhadores;
- Realizar atividades impossíveis de serem feitas manualmente, como a solda em locais muito estreitos;

Segundo Groover et al. (1989) os robôs trabalham com a junção de três dispositivos diferentes: os sensores, as ferramentas e a garras, apresentados abaixo:

- Os sensores possuem a tarefa de fornecer os parâmetros sobre o comportamento do robô, geralmente em termos de posição e velocidade em função do tempo;
- As ferramentas consistem em todo o aspecto mecânico e estrutural do robô juntamente com os componentes que transformam a energia necessária em trabalho mecânico;
- As garras, também chamadas de efetadores, são periféricos responsáveis por realizar uma aplicação particular da tarefa do robô.

A movimentação do robô é realizada por meio de juntas acionadas. Estas juntas conectam os membros rígidos, denominados de elos, formando uma cadeia elo-junta-elos. As juntas são capazes de realizar movimentação entre os elos, sendo que esta movimentação

pode ser linear ou rotacional (GROOVER et al., 1989). Cada movimentação realizada por uma junta individual é denominada Grau de Liberdade (DOF).

As juntas podem ser classificadas em três tipos:

- a. Juntas Rotacionais: capacitadas por realizar movimentos rotativos dos elos;
- b. Juntas Translacionais (Prismática): capacitadas por realizar movimentos deslizantes ou translacionais dos elos;
- c. Juntas Esféricas (*ball and socket*): capacitadas por realizar movimentos rotacionais dos elos nos três eixos. Possuindo assim um total de 3 DOF's, realizados pelas movimentações rotativas de *pitch*, *yaw* e *roll*.

2.1.1. Classificação dos Robôs

Os robôs podem ser classificados de acordo com o volume de trabalho que ele alcança no momento de sua utilização, ou seja, o volume que o robô pode alcançar com a garra. Schilling (1990) classifica os robôs em cinco tipos diferentes: Configuração de coordenadas cartesianas, cilíndrica, esférica, SCARA e articulada;

- **Configuração de coordenadas cartesianas:** faz uso de três eixos perpendiculares entre si formando os eixos X, Y e Z. Como é possível analisar na Figura 5, ao deslocar uma régua (eixo) em relação ao outro se consegue um volume cúbico alcançado pelo robô.

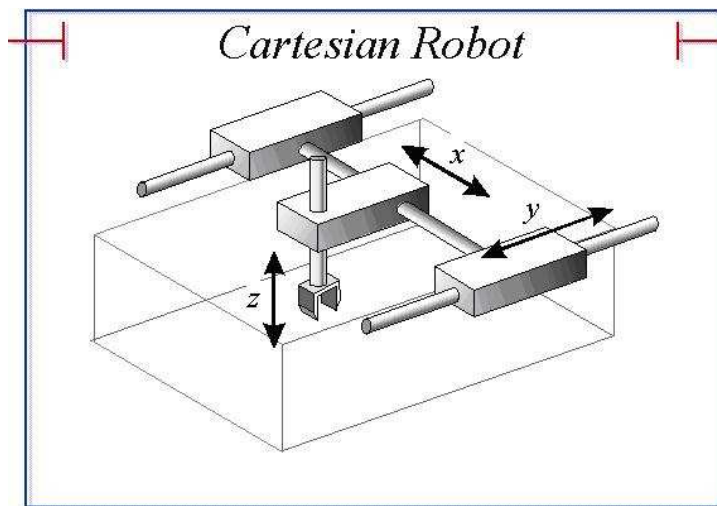


Figura 5 – Robô de coordenadas cartesianas
(RANCH, 2007)

- **Configuração cilíndrica:** faz uso de uma coluna vertical e um suporte integrado a ela que pode se movimentar para cima ou para baixo ao longo da coluna, como é possível visualizar na Figura 6. Ao girar a coluna, pode-se criar um envoltório de trabalho cilíndrico.

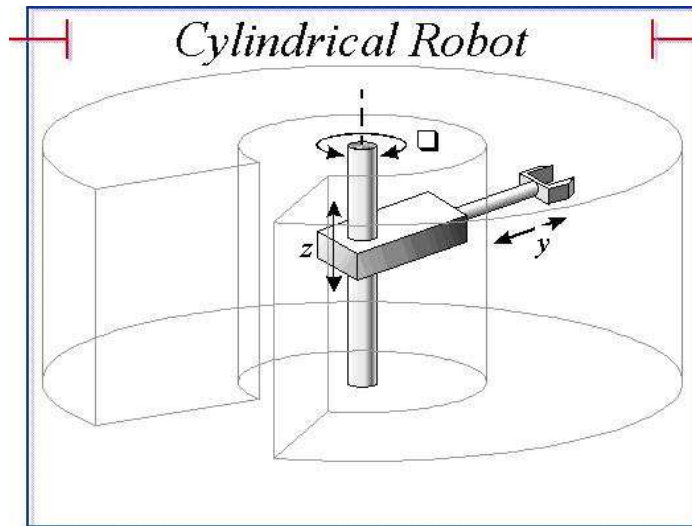


Figura 6 – Robô de coordenadas cilíndricas
(RANCH, 2007)

- **Configuração esférica:** faz uso de um braço robótico que pode ser rotacionado em relação a uma coluna horizontal, assim como é mostrado na Figura 7. A coluna é fixa em uma base rotativa e o braço do robô possui a capacidade de se estender e encolher.

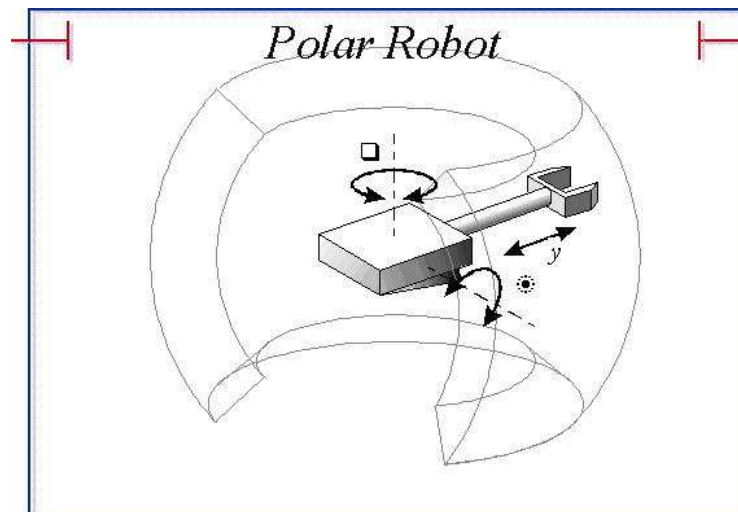


Figura 7 – Robô de coordenadas esféricas
(RANCH, 2007)

- **Robô SCARA:** possui um braço integrado com duas juntas rotacionais acoplado em um eixo vertical e uma junta com movimentação deslizante verticalmente ao final do braço, como é possível visualizar na Figura 8.

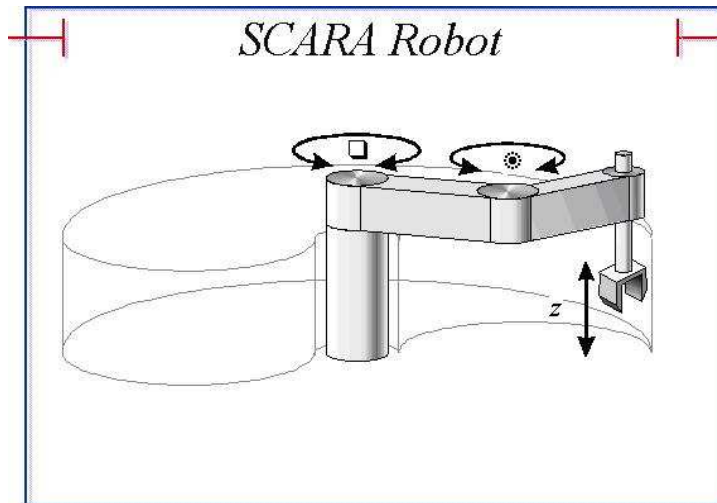


Figura 8 – Robô do tipo SCARA
(RANCH, 2007)

- **Robô articulado:** possui a configuração semelhante ao braço humano. Como é possível visualizar na Figura 9, o robô articulado consta de dois componentes retos, correspondendo ao antebraço e ao braço. Nas pontas dos dois componentes retos são instaladas juntas rotacionais. O braço é conectado ao antebraço e o antebraço é conectado à base, podendo esta também realizar um movimento de rotação.

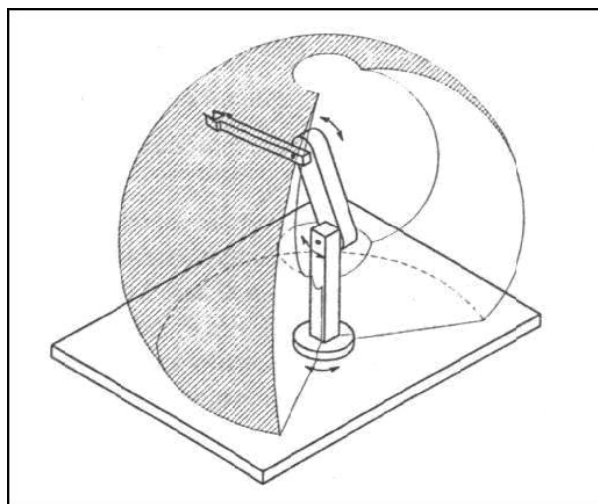


Figura 9 – Robô articulado
(SANTOS, 2006)

2.1.2. Tipos de Garras

Existe atualmente uma série de tipos de garras diferentes com o intuito de atender a necessidade da indústria moderna. Apesar de existirem garras com maior complexidade, como as que tentam imitar a anatomia da mão humana, nas indústrias estes tipos de garras geralmente não são utilizadas.

As garras possuem a tarefa de realizar a pega de objetos, podendo estes serem providos de geometria complexa. Existem diferentes maneiras utilizadas para a preensão de objetos e, conseqüentemente, diferentes tipos de garras e no que elas se baseiam para realizar esta atividade. As garras podem ser classificadas segundo o princípio de trabalho que elas utilizam em: garras com dedos de movimentação mecânica, garras a vácuo, eletroímãs, ganchos e adesivos (PAZOS, 2002).

- As garras com dedos de movimentação mecânica são normalmente compostas por dois ou mais dedos que se abrem e fecham mecanicamente;
- As garras a vácuo consistem em ventosas ligadas a uma bomba de vácuo através de uma eletro-válvula;
- As garras magnéticas são semelhantes às garras a vácuo, porém as ventosas são substituídas por eletroímãs;
- Os ganchos são utilizados para transportar peças que possuem formato irregular;
- As garras adesivas consistem na utilização de uma substância para a operação de pega do objeto.

Como este trabalho será focado no estudo de um tipo específico de garra de movimentação mecânica, sendo esta aplicada em um simulador, será mostrada uma série de garras deste tipo que são referenciadas em alguns simuladores atuais.

2.1.2.1. *Parallel Jaw Gripper*

Esta é uma garra do tipo pinça e é conhecida como pinça de dois dedos. Esta garra é simples, comum e desenvolvida por vários fabricantes. Ela possui apenas um DOF, sendo este para os dois dedos simultaneamente (MILLER, 2001, p. 26).

Estão integrados nela dois dedos paralelos entre si que possuem juntas translacionais. Estes dedos se movem no mesmo eixo e ao mesmo tempo em direções opostas, como é possível ver na Figura 10. O ponto de contato entre os dois dedos localiza-se exatamente no centro da pinça, caso esta esteja totalmente fechada.

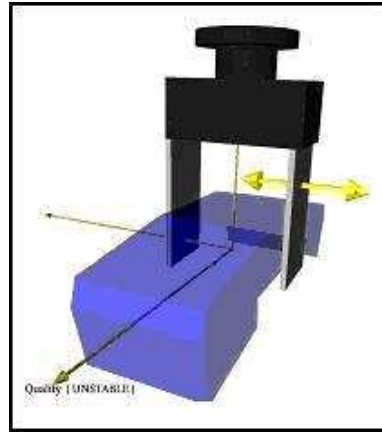


Figura 10 – Pinça *Parallel Jaw Gripper*
(MILLER, 2001, p. 35)

2.1.2.2. *Barret Hand*

Esta garra tem seu design inovador desenvolvido pela Universidade da Pennsylvania. Esta garra possui três dedos articulados com duas juntas cada, como é possível verificar na Figura 11. Um dedo é estacionário (dedo F3) e os outros dois (dedos F1 e F2) podem variar a angulação da base de maneira síncrona em sentidos opostos, ou seja, ao variar a angulação da base de um dedo automaticamente é variada a angulação do outro. Esta angulação entre os dois dedos não estacionários pode chegar a 180°.



Figura 11 – Garra *Barret Hand*
(CLEMSON, 2007)

Esta garra possui ao todo quatro graus de liberdade, sendo esta controlada por quatro motores (MILLER, 2001, p. 28). Um motor gira a base da garra, outro motor é encarregado de manter a sincronia entre os dois dedos com base móvel (dedos F1 e F2). Os outros dois motores são encarregados cada um de variar a angulação de uma junta existente nos três dedos (MILLER, 2001, p. 27-28). Desta maneira as angulações da primeira junta de cada dedo são iguais e assim para as outras juntas consecutivas.

2.1.2.3. DLR Hand

A *DLR Hand* foi desenvolvida pelo Centro Espacial da Alemanha e possui quatro dedos articulados semelhantes ao dedo humano (DLR, 2007), como é possível analisar na Figura 12, porém com seu tamanho 1,5 vezes maior. Todos os dedos são idênticos e cada um possui três juntas. Esta garra tem ao todo doze graus de liberdade internos sendo divididos em três juntas independentes para todos os quatro dedos (MILLER, 2001, p. 28).

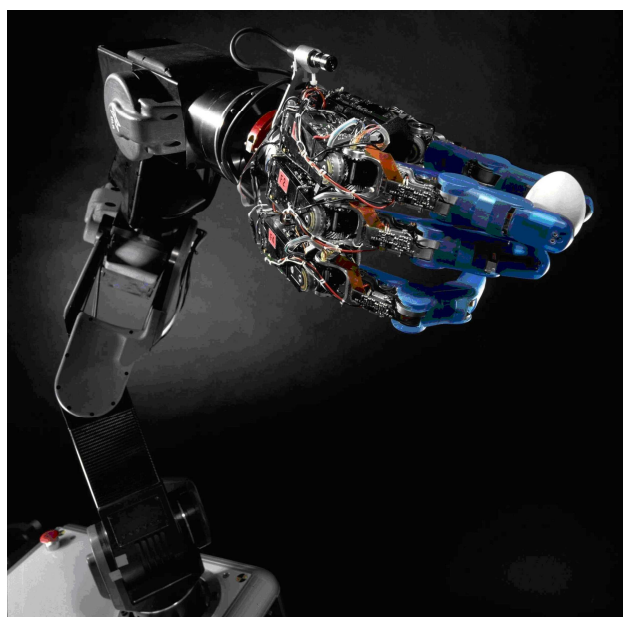


Figura 12 – Garra *DLR Hand*
(DLR, 2007)

2.1.2.4. Robonaut Hand

Esta garra foi desenvolvida pelo Centro Espacial Johnson (2007) da NASA e possui cinco dedos, como é possível analisar na Figura 13, com quatorze graus de liberdade ao todo. Esta garra possui 95% do tamanho médio de uma mão humana masculina.

Os dedos, indicador, do meio e polegar, são considerados dedos de manipulação primária em quanto que os outros dois garantem uma boa pega de forma a “abraçar” o objeto segurado.

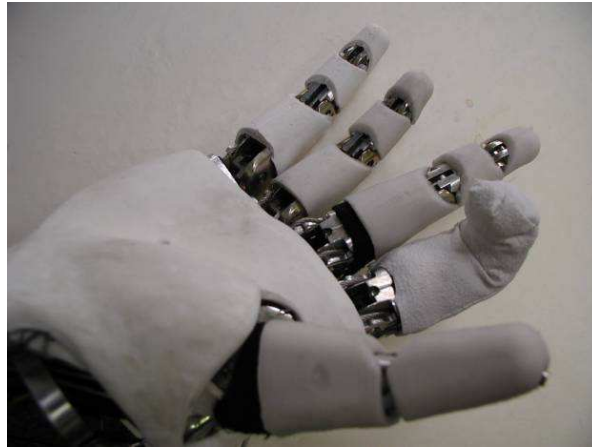


Figura 13 – Garra *Robonaut Hand*
(NASA, 2007)

2.1.2.5. *Rutgers Hand*

Esta garra foi desenvolvida pelos Departamentos de Engenharia Mecânica e Aeroespacial da Universidade de Rutgers em Nova Jersey (DELAURENTIS e MAVROIDIS, 2000). Esta garra possui cinco dedos com anatomia antropomórfica sendo que, quatro dedos possuem três juntas (Figura 14 (b)) e o polegar possui apenas duas (Figura 14 (c)). Suas juntas operam de maneira similar às fibras musculares pertencentes às juntas existentes no dedo humano (MILLER, 2001, p. 30). Como é possível visualizar também na Figura 14 (b), cada junta existente no dedo robótico é determinada por um par de fios que podem contrair ou relaxar causando a rotação na junta. Os cinco dedos, incluindo o polegar, são conectados na palma robótica ilustrada na Figura 14 (a).

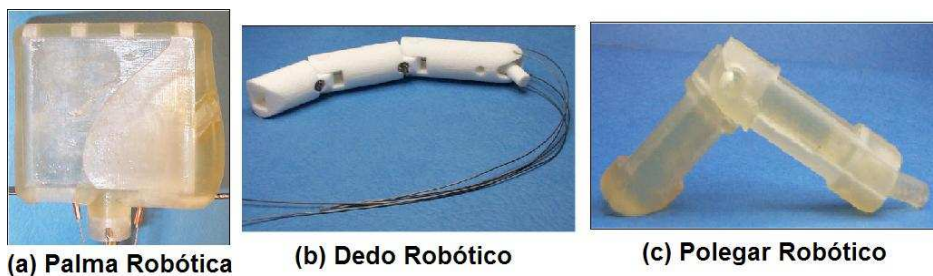


Figura 14 – Partes da garra *Rutgers Hand*
(DELAURENTIS e MAVROIDIS, 2000)

Todos os dedos são ligados na palma da mão com uma junta do tipo *ball and socket*, possuindo assim, dois graus de liberdade independentes proporcionados pela rotação em dois eixos. Desta forma, a mão possui ao todo dezenove graus de liberdade, ou seja, quatro distribuídos para quatro dedos mais três para o polegar.

2.1.3. Robô Scorbot ER-4PC

Este robô possui um total de cinco juntas rotacionais. Sua disposição é análoga ao braço humano possuindo, assim, uma anatomia denominada antropomórfica. A capacidade de carregamento do robô é limitada a objetos com peso relativamente pequeno, estes não passando de 1kg incluindo o peso da própria garra (ROBOTEC, 1982, p. 4).

No ponto final do braço robótico está instalada uma garra do tipo pinça de dois dedos, podendo esta ter orientação arbitrária internamente ao espaço de trabalho. Pode-se perceber à direita da Figura 15 os eixos de rotação existentes e a anatomia do robô comparada ao braço humano. O Scorbot é composto pela base, corpo, braço, antebraço e garra, sendo todos interligados como mostrado na parte esquerda da Figura 15.

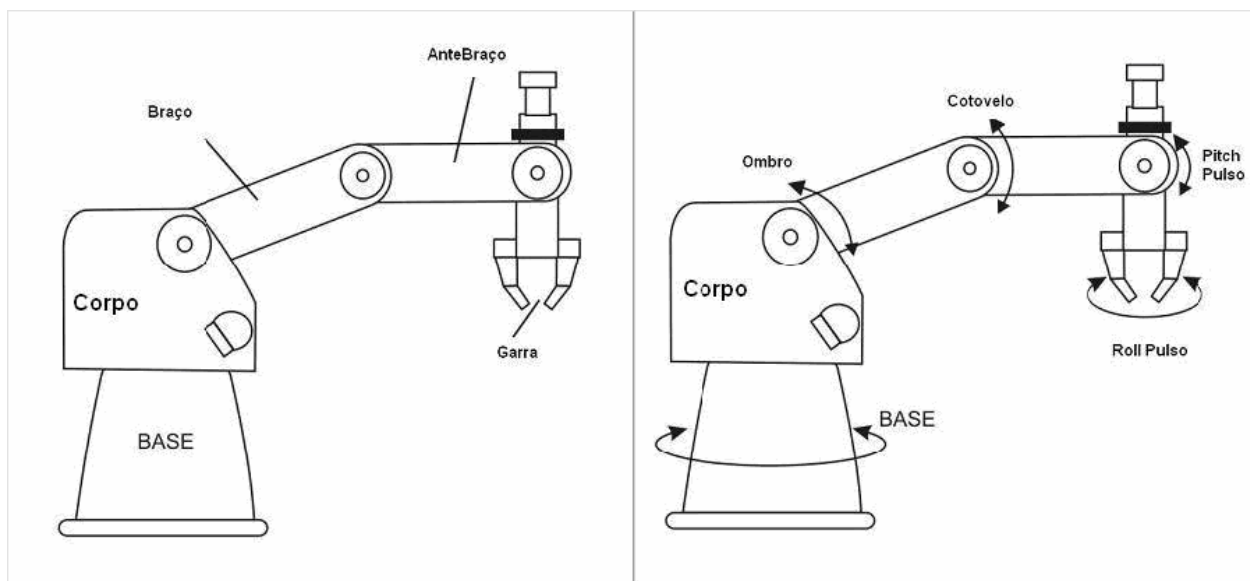


Figura 15 – Anatomia do robô Scorbot ER-4PC
(ROBOTEC, 1982, p. 5)

Como foram mostrados na Figura 15, os possíveis movimentos de rotação são feitos nas seguintes juntas:

1. Base-Corpo;
2. Corpo-Braço;

3. Braço-Antebraco;
4. Antebraco-Garra-*Pitch*;
5. Antebraco-Garra-*Roll*.

O movimento de *Pitch* é feito através da rotação da garra no eixo horizontal do braço enquanto que o movimento de *Roll* é realizado através da rotação da garra em relação ao seu próprio eixo.

O volume de trabalho pertencente a este robô, mostrado na Figura 16, é determinado através do tamanho de seus elos e da capacidade máxima de rotação de cada junta. Toda a região mostrada na Figura 16 pode ser alcançada pelo Scorbot, sendo este volume semelhante a uma esfera. A Tabela 1 mostra algumas especificações do fabricante que serão importantes para o trabalho.

Especificações Técnicas	
Estrutura Mecânica	Verticalmente Articulado
Número de Eixos	5 Eixos Rotacionais + Pinça
Raio máximo de Operação	610 mm
Abertura máxima da Pinça	75 mm
Rotação dos Eixos	Eixo 1 – Rotação Base: 310° Eixo 2 – Rotação do ombro: +130°/-35° Eixo 3 – Rotação do cotovelo: +/-130° Eixo 4 – <i>Pitch</i> do Punho: +/-130° Eixo 5 – <i>Roll</i> do Punho: Ilimitado

Tabela 1 – Especificações técnicas do Scorbot ER-4PC
(ROBOTEC, 1982, p. 4)

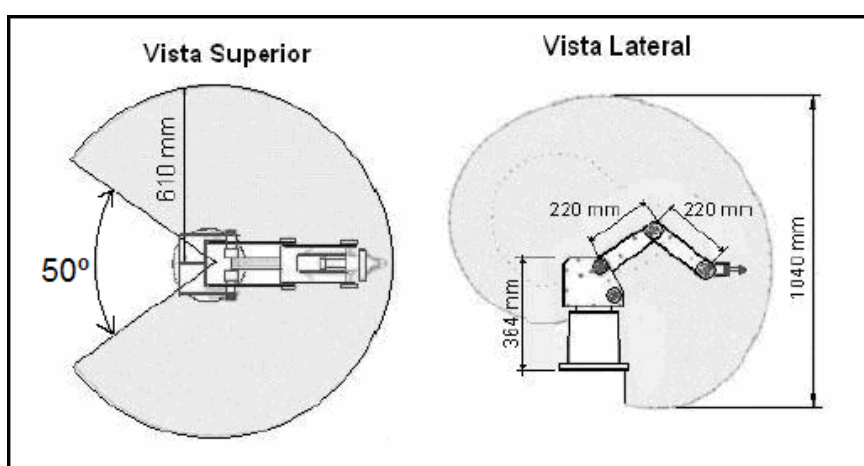


Figura 16 – Volume de trabalho ocupado pelo robô Scorbot
(ROBOTEC, 1982, p. 6)

A pinça existente no robô é a pinça *parallel jaw gripper* e pode ser desinstalada do robô para incluir outro efetuador qualquer. Os dedos da pinça se fecham em sentidos opostos de forma paralela.

2.2. Modelagem Geométrica

Modelagem Geométrica é a maneira como podem ser descritos os modelos dos objetos sólidos que serão manipulados por um sistema computacional. Esta representação matemática precisa ser completa e não pode representar mais de um objeto, ou seja, não apresenta ambigüidade (REQUICHA, 1980).

Atualmente existem várias técnicas de modelagem geométrica, porém será explicada a técnica utilizada no protótipo virtual do robô Scrobot, denominada representação por fronteiras (*Boundary Representation*). Além de ser usada no protótipo virtual, a técnica B-rep também é utilizada na representação dos objetos a serem tratados neste trabalho, nos trabalhos estudados na literatura (que fazem uso de poliedros B-rep) e também é internamente utilizada na ferramenta de Realidade Virtual VRML.

2.2.1. Representação por Fronteiras (*Boundary Representation*)

Esta é a técnica de representação mais familiar utilizada por cientistas da computação. Um sólido representado por B-rep possui sua parte externa segmentada em um número finito de faces interligadas em suas bordas por arestas que são determinadas por vértices (REQUICHA, 1980, p. 452).

Alguns tipos de representações B-rep são restritos a objetos tipicamente planares, que possui fronteiras poligonais convexas ou podem também serem restritos somente ao uso de triângulos. Objetos que possuem geometria curva podem possuir uma representação mais exata, porém são necessários algoritmos mais complexos de intersecção de curvas (SPECK, 2001, p. 48). Por isto, tais tipos de geometrias também podem ser representados por aproximações lineares de suas bordas curvas. Esta representação de objetos curvos acaba formando um efeito denominado “facetamento” (SPECK, 2001, p. 48). Este efeito faz com que o objeto tenha a aparência de um conjunto de faces em sua superfície curva.

A representação de um sólido poliédrico possui suas fronteiras compostas de polígonos e estes de arestas. Cada aresta necessita compartilhar um número uniforme (par) de polígonos. Um poliedro simples, ou seja, que não possui furos (exceto o *torus*), deve

satisfazer as condições impostas pela fórmula de Euler (FOLEY et al., 1995, p. 543), mostrada a seguir:

$$V - E + F = 2 \quad (2.1)$$

Sendo:

- Número de vértices: V
- Número de arestas: E
- Número de faces: F

A Fórmula (2.1), apesar de ser aplicável a um poliedro simples, não garante que o objeto modelado seja um objeto sólido. Para garantir que o objeto seja um sólido válido, é necessário que cada aresta do objeto seja compartilhada por somente duas faces, que cada vértice seja compartilhado por pelo menos três arestas e suas faces não se interpenetram (FOLEY et al., 1995). As condições para um poliedro geral são garantidas pela generalização da fórmula de Euler, mostrada a seguir:

$$V - E + F - H = 2*(C - G) \quad (2.2)$$

Sendo:

- Número de vértices: V
- Número de arestas: E
- Número de faces: F
- Número de furos existentes nas faces: H
- Número de componentes separados do objeto: C
- Número de furos que passam completamente o objeto: G

Como é possível verificar na Figura 17, o poliedro apresentado respeita a condição imposta pela Fórmula (2.2). A quantidade de vértices do poliedro é 24, a quantidade de arestas é 36, a quantidade de faces é 15, a quantidade de furos existentes nas faces é 3, a quantidade de componentes do objeto é 1 e a quantidade de furos que ultrapassam todo o objeto de ponta a ponta é 1.

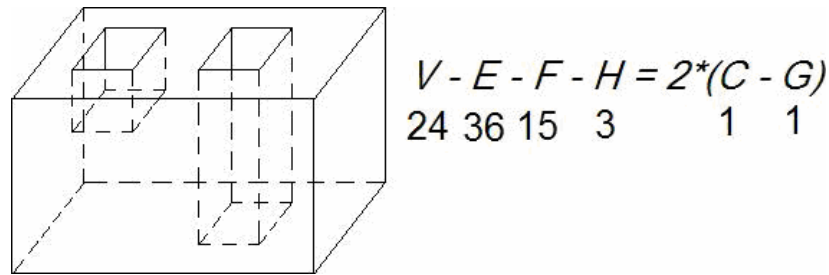


Figura 17 – Poliedro geral
(FOLEY et al., 1995, p. 544)

Todo objeto composto por um conjunto de faces possui sua geometria bem definida, porém não identifica qual lado de cada face é o exterior. Isto é solucionado através de um vetor normal atribuído às faces do objeto. Estes vetores normais são encontrados através de uma convenção onde o vetor normal aponta sempre para o lado exterior do objeto. Para isto, cada face é tratada como um plano e, a partir dele, é encontrado o vetor normal. O sentido do vetor normal é obtido seguindo uma seqüência de avaliação dos vértices das faces em sentido horário. Outros elementos como texturas podem ser incluídas facilmente no modelo B-rep, bastando atribuir uma imagem a cada face do objeto.

Devido à popularidade de utilização do modelo B-rep em computação gráfica, diferentes técnicas foram desenvolvidas para representar objetos poligonais eficientemente. A Figura 18 ilustra um modo de representação de objetos em termos de estrutura de dados apresentado por Requicha (1980). O objeto é representado por um grafo que possui seus nós contendo as faces, as arestas e os vértices. Estes nós são interligados formando a topologia do objeto, evitando a representação ambígua. O esquema mostrado na figura apresenta um modo de representação baseado em uma malha de triângulos, denominado triangulação (REQUICHA, 1980, p. 452). Como é possível verificar na Figura 18, onde somente uma parte do objeto está representada, as faces que determinam a fronteira do objeto ($f_1, f_2, f_3, \dots$) possuem um conjunto de arestas ($e_1, e_2, e_3, \dots$). As arestas por sua vez possuem duplas de vértices ($v_1, v_2, v_3, \dots$) que representam pontos existentes no espaço tridimensional.

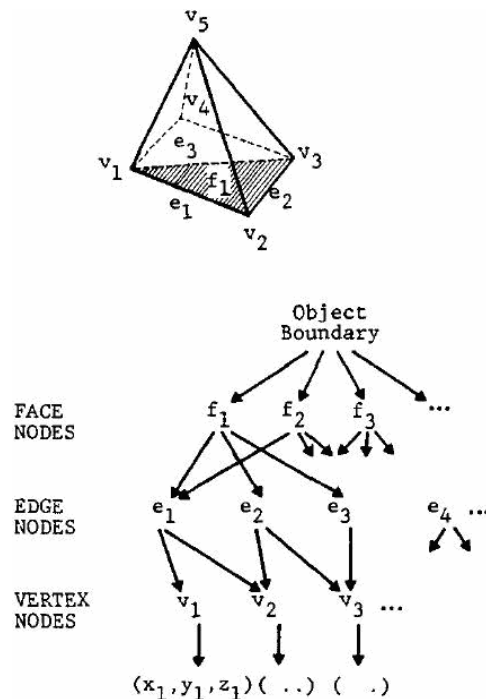


Figura 18 – Modo de representação por B-Rep
(REQUICHA, 1980, p. 452)

O método de representação B-rep é bastante utilizado para a modelagem de objetos sólidos rígidos devido a sua simplicidade conceitual de representação e por fazerem uso de métodos de renderização de polígonos já implementados em *hardware*, próprios para computação gráfica.

2.3. Física aplicada a Objetos Rígidos

2.3. Modelos de Contato

Como existe o contato entre dois corpos no momento da prensão, é importante ter conhecimento sobre os cálculos físicos que envolvem o contato e o atrito entre corpos.

O contato entre as superfícies de dois corpos pode ser descrito como um mapeamento de forças exercidas no ponto de contato e a resultante das forças e momentos para ambos os corpos (LEITE, 2005, p. 44). Em outras palavras, no instante do contato entre dois corpos, indiferente do material ou outra informação física, são exercidas forças para ambos os corpos exatamente no ponto em que ocorre a intersecção.

Caso não seja considerado o atrito no ponto de contato entre os corpos, a força resultante de cada corpo apenas terá a direção normal à superfície dele. Porém, em situações práticas, é raro ocorrer situações como essa (LEITE, 2005, p. 44).

O modelo de atrito de Coulomb é um modelo empírico onde é considerado o atrito no momento do contato entre dois corpos diferentes. Segundo Leite (2005, p. 45) uma força resultante tangencial é proporcional à força aplicada em um corpo, sendo que a constante de proporcionalidade é dada em função dos materiais que estão em contato. Esta constante é conhecida como coeficiente de atrito estático e é representada por μ_s .

O conjunto de forças aplicadas em um ponto de contato deve permanecer interno a um cone centrado sobre a normal da superfície do objeto, como é mostrado na Figura 19. Este cone, projetado internamente ao objeto, é denominado cone de atrito e seu ângulo interno (α_s) com respeito a normal é dado pelo resultado da fórmula:

$$\alpha_s = \arctan(\mu_s) \quad (2.3)$$

Sendo:

- Ângulo interno do cone de atrito: α_s
- Coeficiente de atrito estático: μ_s

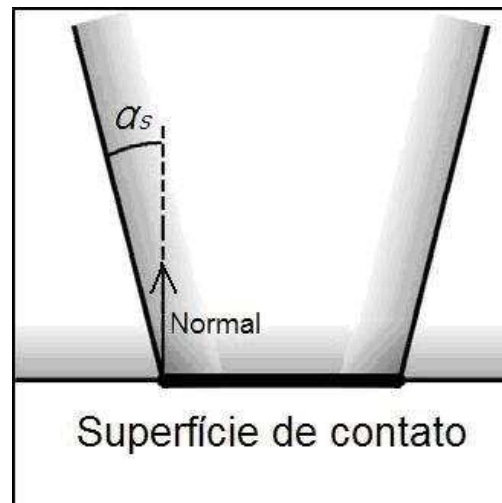


Figura 19 – Cone de atrito formado no contato de corpos
(MORALES et al., 2006)

A força resultante do contato pode seguir qualquer direção interna ao cone de atrito, mostrado na Figura 19 e determinado pela Fórmula (2.3). Alguns coeficientes de atrito entre materiais são mostrados na Tabela 2. Desta forma, o coeficiente de atrito estático depende da natureza das duas superfícies de contato (LEITE, 2005, p. 46).

Material	μ_s
Aço sobre Aço	0,58
Plástico sobre Aço	0,3 – 0,35
Plástico sobre Plástico	0,5
Madeira sobre Madeira	0,25 – 0,5
Madeira sobre Metal	0,25 – 0,6

Tabela 2 – Coeficientes de atrito estático
(LEITE, 2005)

3. TRABALHOS RELACIONADOS

Vários trabalhos analisam o problema da preensão de objetos sendo feita por um robô, sendo este simulado graficamente ou real. É proposta uma vasta quantidade de idéias de soluções distintas e que são melhor adaptadas ao tipo específico do problema. Serão detalhadas a seguir algumas propostas de solução.

3.1. Utilização do Centro de Massa para equilíbrio da pega

O propósito do trabalho feito por Smith et al. (1999) foi garantir a pega feita por um robô a partir de objetos modelados pelo usuário em uma Java *applet*. Através da *applet* são criadas soluções que, posteriormente, são mostradas através de um *site* (LEE, 1999) na internet. A solução é voltada ao tipo de garra *parallel jaw gripper* podendo, essa, segurar objetos levando em consideração características físicas como deslizamento e torque submetido pela pinça de dois dedos.

Foi desenvolvido no trabalho um algoritmo eficiente para pega de objetos poliédricos rígidos, sendo este executado por um robô do tipo SCARA que possui quatro graus de liberdade. Como este tipo de robô faz uso de coordenadas cartesianas, a pega dos objetos somente pode ser realizada por cima e com objetos que não ultrapassem um limite de altura estipulado no algoritmo. O valor de altura máxima foi determinado de acordo com o volume de trabalho que o robô SCARA possui e sua acessibilidade para a pega do objeto.

Todos os dados necessários para o funcionamento do algoritmo são cedidos pelo usuário através da manipulação de uma aplicação gráfica que possui a função de modelar a geometria do objeto. Parâmetros como o centro de massa e o coeficiente de atrito do material também são definidos através de uma interface *GUI* inclusa no aplicativo.

Inicialmente é criado um polígono obtido através do resultado da intersecção do objeto modelado em três dimensões com um plano formado na horizontal a partir do centro de massa do objeto analisado, como é mostrado na Figura 20. Todas as análises do problema e obtenção do resultado são feitos utilizando este polígono encontrado na intersecção, que também é mostrado na Figura 20. Isto faz com que o problema, inicialmente disposto em três dimensões, seja reduzido a um problema planar. O algoritmo gera duplas de pontos dispostas nas arestas do referido polígono. Importante ressaltar que entre estas duplas de pontos, cria-se

um *eixo de pega* o qual deve passar o mais próximo possível do centro de massa do objeto, como é mostrado na Figura 21.

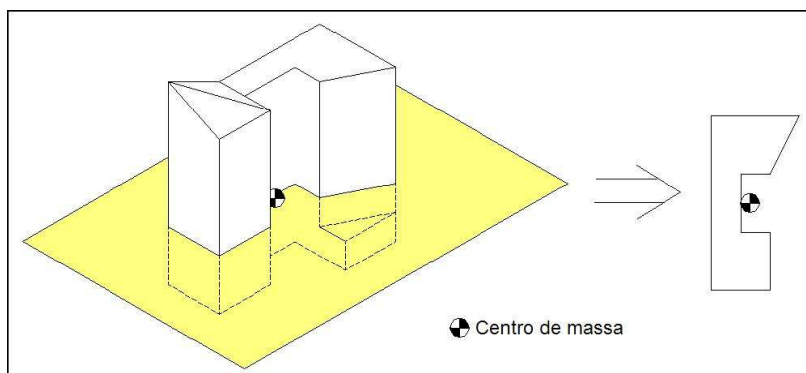


Figura 20 – Criação do polígono de intersecção
(SMITH et al., 1999)

O algoritmo criado pelos autores faz basicamente uso de geometria para determinar os pontos de pega. Estes pontos são distribuídos sobre as arestas existentes no polígono encontrado inicialmente através da intersecção. Cada dupla de pontos é criada de acordo com a angulação existente entre as arestas do polígono nas quais os dois pontos da dupla fazem parte. O teste é feito para todas as arestas e é aceita uma tolerância na angulação entre as arestas que a dupla de pontos faz parte.

Os critérios utilizados por Smith et al. (1999) para o tratamento da preensão são:

- Não é possível a preensão próxima a vértices convexos, ou seja, evita-se a preensão pelas bordas do objeto. Isto é controlado por um parâmetro ϵ ;
- O eixo de pega deve estar interno aos cones de atrito para que os dedos da pinça não escorreguem;
- O eixo de pega deve se alinhar, o máximo possível, aos vetores normais das arestas da peça;
- O eixo de pega deve estar o mais próximo possível do centro de massa do objeto, como é mostrado na Figura 21. Isto minimiza o torque para suspender a peça;
- Os pontos de preensão devem ser acessíveis aos dedos da garra.

Também é avaliado se o eixo de pega é perpendicular ao bissetor. O bissetor é a linha que divide ao meio o ângulo entre as duas arestas analisadas, sendo este mostrado na Figura 21. Também são utilizados cálculos físicos na simulação, através da análise dos cones de atrito gerados durante o contato dos dedos da pinça com a superfície do objeto.

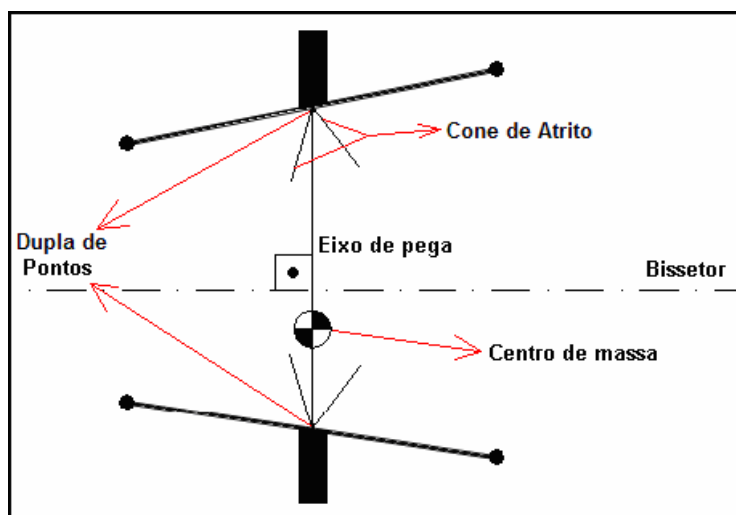


Figura 21 – Ajuste dos pontos para minimização de torque e atrito
(SMITH et al., 1999)

A solução do problema foi dividida em três etapas: em teste de atrito, seção de pega e acessibilidade. O teste de atrito executa em um tempo $O(n^3)$, a seção de pega executa em um tempo de $O(1)$ e a acessibilidade executa em um tempo $O(n^3)$. O valor n equivale ao número de arestas existentes no polígono analisado.

O teste de atrito é capacitado por fazer a verificação do ângulo formado entre as arestas que a dupla de pontos de pega fazem parte. Caso este ângulo seja menor do que a soma dos ângulos que formam os cones de atrito pertencentes aos dois pontos, a dupla de pontos é aprovada no teste. Este teste considera uma folga para a aceitação ou não do par de pontos submetido.

Nesta etapa, é executada uma heurística capaz de detectar superfícies côncavas no objeto, como é mostrada na Figura 22. Este tipo de tratamento garante, dependendo da geometria do objeto, maior robustez durante a pega. Como se pode perceber na figura, o cone de atrito projetado a partir do ponto de concavidade e_{cv} possui uma angulação maior do que os outros dois projetados por e_j e e_k . Devido à soma das angulações dos cones de atrito causados pelos pontos vizinhos e_j e e_k , sendo o resultado desta soma um cone projetado no ponto de concavidade e_{cv} , este método possui uma abordagem interessante a ser considerada para a preensão. A angulação do cone de atrito projetado sobre e_{cv} é maior do que a angulação dos cones de atrito projetados sobre os pontos e_j e e_k e, conseqüentemente, a área também é

maior. Isto faz com que a força exercida durante o contato da pinça com o objeto possa ser dissipada com mais eficiência e em uma área maior interna ao objeto.

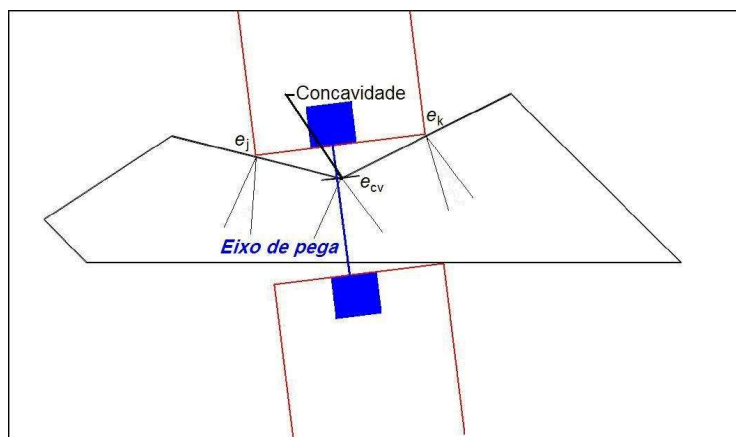


Figura 22 – Soma dos cones de atrito sendo feita em um vértice côncavo (SMITH et al., 1999)

Percebe-se que o uso de heurísticas, como a apresentada, pode encontrar ou melhorar soluções que talvez não fossem averiguadas posteriormente por algoritmos que têm ausência deste tratamento.

Logo após é iniciada a seção de pega que procura minimizar o torque e o atrito exercidos. Nesta etapa cada ponto da dupla é movido sobre sua respectiva aresta da qual faz parte procurando fazer com que o eixo de pega corte o centro de massa do objeto, como é mostrado na Figura 21.

Finalizando estas etapas, é feito o teste de acessibilidade que garante a ausência de colisão entre a pinça, estando essa aberta, e outra parte do objeto modelado. Como não existem outros objetos inseridos na cena, basta verificar a acessibilidade somente entre a pinça e o objeto modelado. Para isso, é traçada uma reta ao longo do eixo de pega e é testado se essa reta intercepta qualquer outra parte do objeto. Os pontos são reposicionados de acordo com este teste de interseção entre a reta e o objeto em si.

3.2. Uso de Heurísticas para eliminação de casos inválidos

É apresentada no trabalho feito por Sunil e Pande (2003) uma ferramenta com o intuito de criar programas para robôs através da internet. Ou seja, o projetista especifica as tarefas que o robô executará através de um modelo gráfico do mundo, e o *software* gera o programa para o robô e envia os dados ao robô real que se encontra remotamente.

Todo o ambiente pode ser modelado no aplicativo desenvolvido por Sunil e Pande. Esta modelagem é realizada adicionando objetos 3-D que possuem informações geométricas de dimensão e orientação. Existem dois tipos diferentes de objetos que podem ser inseridos no ambiente: os objetos móveis e os objetos estacionários. O usuário possui a liberdade de inserir esses dois tipos de objetos, que podem ser blocos ou cilindros rígidos.

O *software* representa o ambiente em três modelos diferentes: o Modelo de Alto Nível, o Modelo Geométrico e o Modelo de Renderização. No Modelo de Alto Nível a cena é representada através de arquivos criados pelo usuário, estes contendo parâmetros de características dos objetos inseridos na cena como orientação e dimensão. Já no Modelo Geométrico esses objetos são modelados por B-Rep, facilitando o cálculo da cinemática inversa. Enquanto que no Modelo de Renderização o ambiente é representado por um conjunto de faces e polígonos, ou seja, é mostrado o resultado de maneira gráfica.

O robô utilizado no experimento possui o tipo de pinça *parallel jaw gripper* e é um robô do tipo articulado. As coordenadas dos pontos de destino e origem alcançados pela pinça necessitam de algumas sub-rotinas pré-implementadas que são capazes de fazer a conversão destes pontos em variáveis de junta do robô (cinemática inversa).

O *software* desenvolvido realiza a tarefa de “pegar e colocar” de maneira automática, onde tanto os objetos a serem pegos como o lugar onde eles serão colocados podem ser determinados pelo usuário da aplicação. Tanto os objetos como os lugares onde eles são colocados são modelados como objetos móveis da aplicação diferenciando no seguinte aspecto: os objetos a serem pegos são geometrias primitivas, citadas anteriormente (caixa e cilindro), enquanto que o lugar onde são colocados os objetos é modelado como sendo uma placa que possui vários furos cilíndricos. São nestes furos que podem ser encaixados os objetos possíveis de serem pegos adicionados na cena.

O robô para o qual a aplicação foi implementada possui articulações que possibilitam somente a preensão por cima do objeto. Como a preensão é realizada apenas verticalmente, o volume de trabalho do robô é linear no eixo de coordenadas Z, como mostrado na Figura 23. O volume de trabalho é todo o volume que o robô pode alcançar com a sua pinça, mapeado em um ambiente 3-D (GROOVER et al., 1989). A figura mostra também os limites do volume de trabalho e ilustra um quarto deste volume, ou seja, um quadrante do sistema de coordenadas do robô.

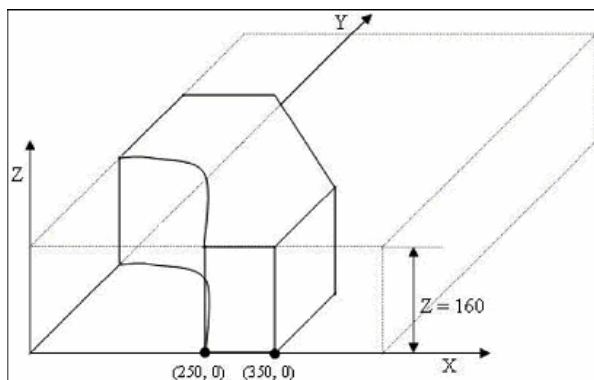


Figura 23 – Espaço de trabalho do robô
(SUNIL e PANDE, 2003)

Como se pode perceber, situações como a localização do objeto fora do volume de trabalho ou altura do objeto excedendo a altura máxima do volume de trabalho, são situações que podem ser eliminadas, pois não atendem os requisitos básicos necessários para que o robô possa executar a apreensão. Antes de o *software* ser executado, são feitos testes para eliminar situações semelhantes às citadas e, assim, evitar processamento desnecessário. Entre estes testes, existem os que avaliam condições entre a placa com furos e o objeto, que são:

- a. Verificar se o diâmetro do objeto visto de cima é menor do que o máximo permitido pela abertura da pinça;
- b. Verificar se o objeto se encontra interno ao volume de trabalho;
- c. Verificar se o tamanho do objeto não excede a altura máxima permitida pelo robô.

Após estas verificações, é calculada a cinemática inversa para determinar as variáveis de juntas do robô, fazendo com que a pinça chegue ao objeto automaticamente. Antes que seja realizada a apreensão, é executada uma heurística de acessibilidade da pinça ao objeto. Este teste verifica se existe espaço suficiente ao redor do objeto para que a pinça possa realizar a apreensão, evitando colisões entre ela e outros objetos vizinhos.

Realizado o teste de acessibilidade, a pinça efetua a apreensão do objeto e então é iniciada a etapa de colocar o objeto em seu devido lugar. Para que esta ação ocorra de maneira satisfatória (ausente de colisões), é realizada uma análise geométrica do ambiente para identificar as regiões não ocupadas por outros objetos. Para que o programa seja executado pelo robô sem que ocorram colisões durante a sua movimentação, o *software* planeja antecipadamente trajetória através dessa análise. Esse teste garante estabilidade durante a movimentação feita pelo robô e livra-o de colisões com outras geometrias.

Foi criada também uma maneira de evitar colisões com os objetos em cena no momento em que é colocado o objeto em seu devido lugar, este tratamento é ilustrado na Figura 24. Para isso, existe a possibilidade de modificar a configuração da pinça para colocar o objeto em seu destino de três maneiras distintas: com a pinça paralela ao eixo X, paralela ao eixo Y e na diagonal. Como se pode perceber são utilizadas as distâncias entre que cada furo e seus vizinhos e, a partir destas informações, é decidida qual será a disposição da pinça para liberar o objeto em seu devido lugar. Caso uma das distâncias entre os furos, representadas por dy e dx , possuem seus valores inferiores à largura do dedo da pinça, é utilizada outra disposição mais adequada, evitando assim a colisão de um dedo da pinça com um objeto vizinho colocado anteriormente na placa.

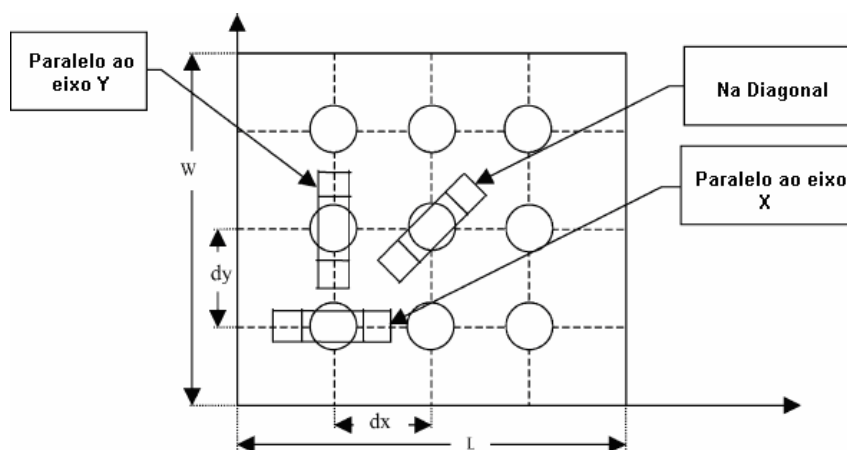


Figura 24 – As três formas de serem pegos os objetos pela pinça
(SUNIL e PANDE, 2003)

As informações de posição e orientação da garra são representadas por uma matriz de transformação de coordenadas homogêneas DH. Isso garante que seja analisada a cinemática direta de maneira mais rápida e eficiente.

O trabalho desenvolvido por Sunil e Pande (2003) não utiliza simulação física no momento da preensão. A solução se baseia em análises geométricas 3-D para realizar a tarefa de pegar e colocar (*pick and place*) de maneira automática.

3.3. Propagação de Cones de Atrito em objetos rígidos

A solução proposta pelo trabalho feito por Morales et al. (2006) consiste em uma síntese de pega genérica de objetos inicialmente desconhecidos, podendo ser pegos por garras de dois ou

três dedos. Somente é utilizada uma imagem do objeto como entrada do sistema, sendo esta extraída através do módulo de processamento da visão do robô.

Como é feito o processamento de uma imagem capturada, a solução do problema se reduz a uma resposta com análise 2-D. O sistema possui uma restrição que é a limitação de serem utilizáveis somente objetos rígidos, tipicamente planares e com massa homogeneamente distribuída.

O trabalho foi dividido em quatro fases: a primeira fase consiste na captura da imagem através de uma câmera focada no volume de trabalho juntamente com o processamento do contorno do objeto; a segunda fase consiste na síntese de pega, onde o sistema computa uma lista de pegas a partir do contorno do objeto encontrado na fase anterior; a terceira fase consiste na determinação do ponto de pega, onde um algoritmo inteligente localiza o ponto central em cada candidato existente na lista encontrada na fase anterior; e finalmente é realizada a execução da tarefa, onde o robô efetua a pega do objeto.

Inicialmente é identificado o contorno do objeto para que posteriormente seja realizada uma avaliação que possa garantir estabilidade durante a preensão do objeto. Este mapeamento de superfície está relacionado com a curvatura natural que o objeto real possui. Para garantir uma área de contato maior no momento da preensão, a curvatura não pode variar bruscamente entre um ponto localizado em sua superfície e seus vizinhos. As áreas de contato são determinadas através de uma tolerância de curvatura entre um ponto da superfície do objeto e seu vizinho.

Quando o sistema reconhece um alinhamento na superfície, através de uma sequência de pontos que não excedem as tolerâncias de curvatura máxima e mínima, é detectada uma região que possível de ser utilizada como contato do objeto com um dedo da pinça. Estas regiões são denominadas de regiões de pega. As regiões de pega, denotadas na Figura 25 por GR_n , têm todos os seus pontos internos não excedendo as tolerâncias de angulação entre seus vizinhos. São mostrados na Figura 25, dois objetos com superfícies distintas (na parte superior da figura) e suas respectivas funções de variação de ângulo entre os pontos da superfície (na parte inferior da figura). A função de variação de ângulo é determinada seguindo o sentido mostrado pelas setas existentes na parte exterior do objeto.

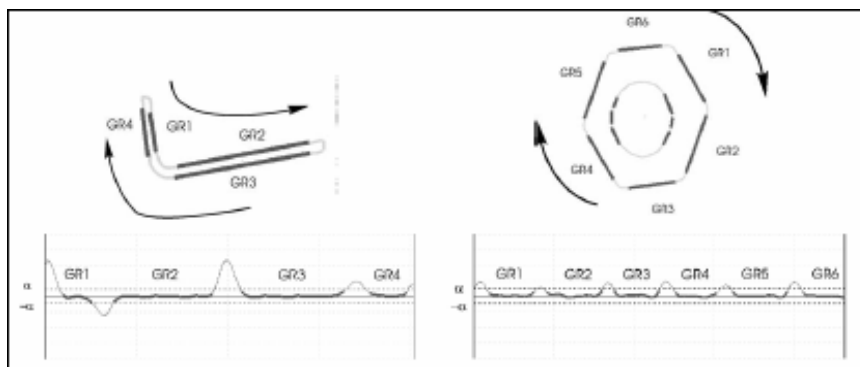


Figura 25 – Superfície dos objetos e suas representações angulares
(MORALES et al., 2006)

Podem ser encontradas superfícies planas segmentadas através de um método de análise de angulação entre os pontos, que são denominadas pelos autores de segmento da região de pega. Caso existam regiões contínuas muito longas, estas são divididas em partes menores levando em consideração uma constante máxima de tamanho para as regiões de pega.

Após a finalização da fase de captura e processamento da imagem, é iniciada a fase de síntese de pega. Nesta parte é criada uma lista de duplas de regiões de pega. As duplas são encontradas através da interligação entre duas regiões de pega através de uma reta, e logo após são determinados os cones de atrito projetados pelo contato dos dedos da pinça nas regiões escolhidas. Desta forma, é utilizado no trabalho análises físicas através do cálculo de forças e torques que a pinça exerce no objeto. Estes cálculos são realizados com base no coeficiente de atrito entre o material do objeto e da pinça.

São escolhidos pares de regiões de pega que satisfaçam ao seguinte critério: o ângulo entre os vetores normais das duas regiões de pega, deve estar entre o intervalo de $180^\circ \pm 2\beta$, sendo β o ângulo do cone de atrito formado internamente ao objeto, como é mostrado na Figura 26. Para evitar eliminação de pares de regiões de pega possíveis como resposta, existe esta tolerância na angulação entre os vetores normais da dupla.

Após formar os dois cones de atrito que são projetados nas duas regiões de contato, é analisado se ocorre intersecção entre os cones, isto é mostrado na Figura 26. Caso aconteça esta intersecção, a área encontrada pela intersecção é chamada de *região factível*, como é mostrada na Figura 26. Existindo uma região factível e a condição de angulação entre os vetores normais ter sido satisfeita, o par de regiões é aprovado para ser submetido à próxima

fase. Estes passos são repetidos para cada par de regiões encontradas na fase anterior. A Figura 26 mostra também a tolerância de angulação entre os vetores normais dessas regiões.

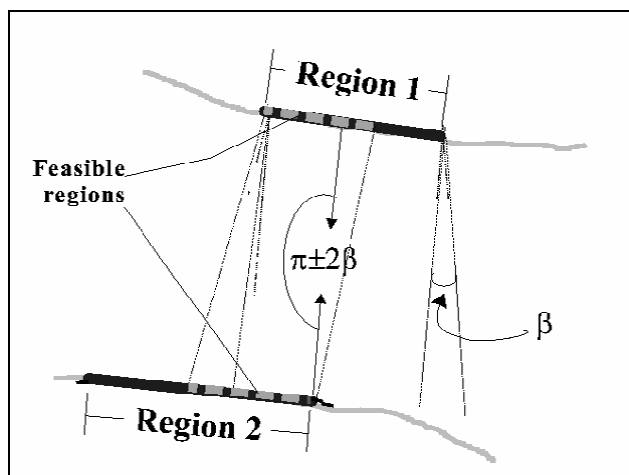


Figura 26 – Região possível formada pelos pontos de contato
(MORALES et al., 2006)

Finalizando a análise de cada par de regiões, é iniciada a fase de determinação do ponto de pega pertencente a cada par de regiões de pega aprovados pela fase anterior. Este ponto está localizado no centro da região factível formada pelo par e é o ponto onde a pinça realiza a preensão. Desta maneira, todas as etapas do trabalho consistem na criação de soluções para a pega do objeto.

Para finalizar, é iniciada a fase de execução da tarefa, onde o robô efetua a preensão do objeto. Foi utilizado um robô articulado que faz a pega de acordo com a posição da câmera e calcula a colisão com o retorno de um detector de força encontrado na pinça.

3.4. Raciocínios aplicados a objetos primitivos

A proposta do trabalho estudado por Miller et al. (2003) consiste na pega de objetos com geometria complexa, sendo estes transformados em uma soma de primitivas geométricas. Os tipos de volumes primitivos utilizados para representar o objeto são: esferas, cilindros, cones e caixas, sendo todos estes sólidos rígidos.

A transformação de um objeto complexo em uma soma de geometrias primitivas faz com que ele seja representado de uma forma mais simples e direta, assim como é ilustrado na Figura 27. Como é possível visualizar na figura, um objeto representado por uma geometria complexa foi transformado em uma soma de duas geometrias primitivas, sendo elas um cilindro e um paralelepípedo. Desta maneira, a aplicação de regras para gerar a pega do objeto

remodelado passa a ser também mais simples computacionalmente. Foi utilizado o simulador denominado “*GraspIt*” (GRASPIT, 2002) para a simular a preensão dos objetos submetidos. O tipo de garra utilizada neste trabalho foi a garra *Barrett Hand*.

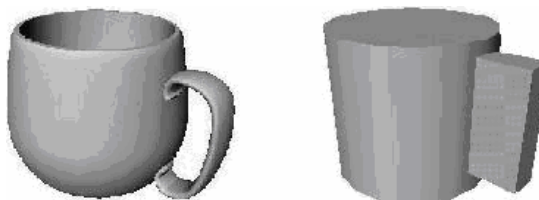


Figura 27 – Objeto complexo modelado em primitivas
(MILLER et al., 2003)

O sistema implementado é constituído por duas etapas principais. Na primeira etapa os autores geram localizações de preensão na superfície do modelo simplificado do objeto. Na segunda etapa, são avaliadas e testadas as possibilidades de uso de cada localização encontrada para preensão.

Inicialmente a garra é movida até uma posição melhor para a realização da pega. Posteriormente o sistema testa várias possibilidades de preensão geradas a partir da posição encontrada e mostra ao usuário a melhor entre elas.

Buscando diminuir a complexidade na escolha da disposição dos dedos da garra *Barrett Hand*, foram definidas quatro disposições padrão. Somente duas destas são utilizadas por esta aplicação para os tipos de primitivas apresentadas no trabalho. As disposições utilizadas no trabalho são: a disposição esférica e a cilíndrica, como é mostrada na Figura 28. Como é possível analisar na figura, enquanto a disposição esférica proporciona uma preensão mais estável para objetos esféricos, a disposição cilíndrica proporciona mais estabilidade na pega de objetos cilíndricos. É possível visualizar também a diferença de angulação entre os dedos nos dois casos.



Figura 28 – Os dois tipos de disposição da garra *Barrett Hand*
(MILLER et al., 2003)

Para gerar um conjunto de preensões possíveis, o sistema necessita computar a versão simplificada da geometria do objeto, que consiste na soma de suas primitivas separadas. O resultado do processamento para determinar as possíveis preensões dependerá unicamente das primitivas encontradas e suas disposições para a representação simplificada do objeto.

Foi definida pelos autores uma série de raciocínios geométricos e heurísticas, sendo estes separados para cada primitiva atendida pela aplicação. Para que seja determinado como será efetuada a tarefa, se faz necessário o conhecimento da posição e orientação 3-D da garra bem como a direção sendo tomada por ela no momento da pega. Estes raciocínios geométricos são separados para cada tipo de primitiva que possa estar contida no objeto. Apesar de os autores terem especificado somente os raciocínios geométricos para a garra *Barrett Hand*, será mostrado o tratamento para as primitivas mais relevantes:

- **Caixas:** podem ser pegas fazendo uso da disposição cilíndrica para a garra. Como é mostrado na Figura 29, enquanto dois dedos seguram em uma face da caixa o outro dedo segura a face oposta. Para isso, a palma da garra deve estar paralela às faces que conectam as faces que estão em contato com os dedos e a direção dos dedos deve ser perpendicular às faces em contato.

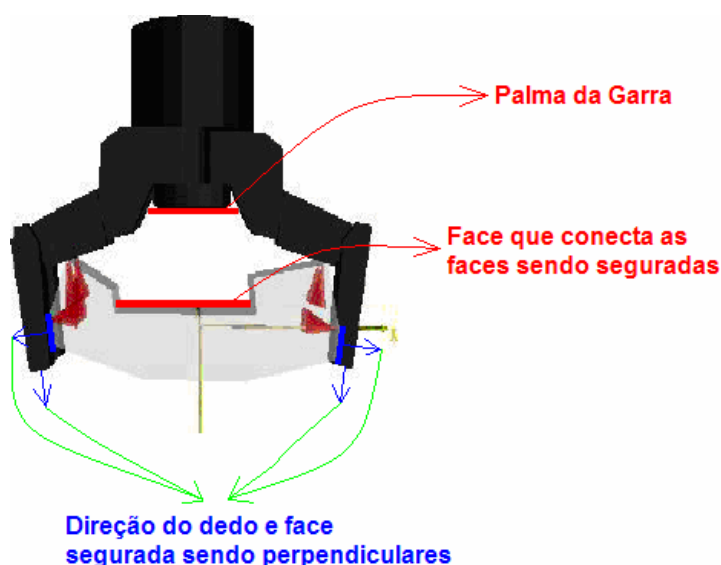


Figura 29 – Pega de caixas feita pela garra
(MILLER et al., 2003)

- **Esferas:** podem ser pegas através da disposição esférica para a garra, como é ilustrado na Figura 30. Para ocorrer a preensão, é verificado se o centro de massa da esfera está

entre a palma da garra e o plano formado sobre o ponto entre os dedos da garra, sendo que este plano possui seu vetor normal igual ao vetor de aproximação da garra.



Figura 30 – Pega de esferas feita pela garra
(MILLER et al., 2003)

- **Cilindros:** podem ser pegos pela lateral ou pelas extremidades.
- **Pela Lateral:** é utilizada a disposição cilíndrica na garra. A Figura 31 mostra que o vetor de aproximação da garra deve ser perpendicular à lateral do objeto. Também é verificado se todos os dedos da garra são perpendiculares ao eixo central da geometria.

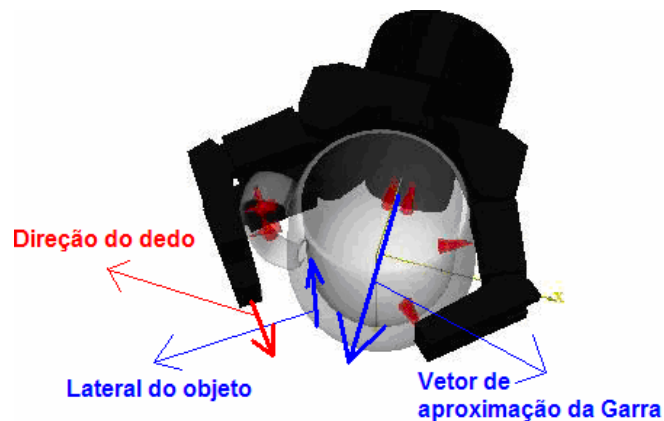


Figura 31 – Pega de cilindros pela lateral feita pela garra
(MILLER et al., 2003)

- **Pelas Extremidades:** a disposição esférica pode ser utilizada nesse caso. O Vetor de aproximação da garra deve ser paralelo ao eixo central do objeto. A Figura 32 mostra o paralelismo entre o eixo central do objeto e o vetor de aproximação da garra.

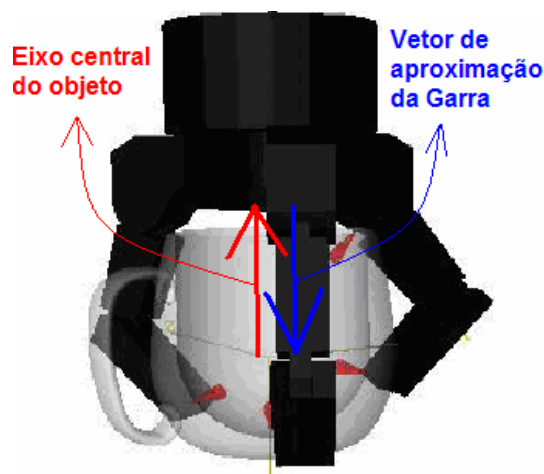


Figura 32 – Pega de cilindros pela extremidade feita pela garra
(MILLER et al., 2003)

A aplicação dessas regras, porém, não chegam a atender a todas as situações de preensão. Apesar de se ter estas regras estabelecidas, alguns casos necessitam ser tratados com outras regras mais específicas que fazem uso de parâmetros automaticamente calculados de acordo com as dimensões do objeto.

Após terem sido criados vários candidatos para resposta final fazendo uso das regras mencionadas, cada situação aprovada na etapa anterior é testada novamente. Este teste consiste em eliminar todas as situações impossíveis de preensão, levando em consideração as orientações da garra e do objeto. A execução desta etapa faz com que se evitem testes desnecessários e, conseqüentemente, aumente o tempo de processamento do algoritmo.

Em seguida é testada a acessibilidade para cada resposta para a pega, sendo que caso ocorram muitas colisões entre a garra e outro objeto disposto no ambiente em determinada resposta encontrada, esta é descartada. É mostrada na Figura 33 uma situação similar à explicação dada, onde ocorre uma série de colisões entre a garra e dois objetos vizinhos ao desejado para preensão. Quando esta situação acontece em uma resposta, esta é desconsiderada para se evitar preensões com muitos obstáculos. Não acontecendo o imprevisto mencionado, os dedos são fechados ao redor do objeto até que sejam detectadas colisões entre o objeto selecionado e os dedos da garra.

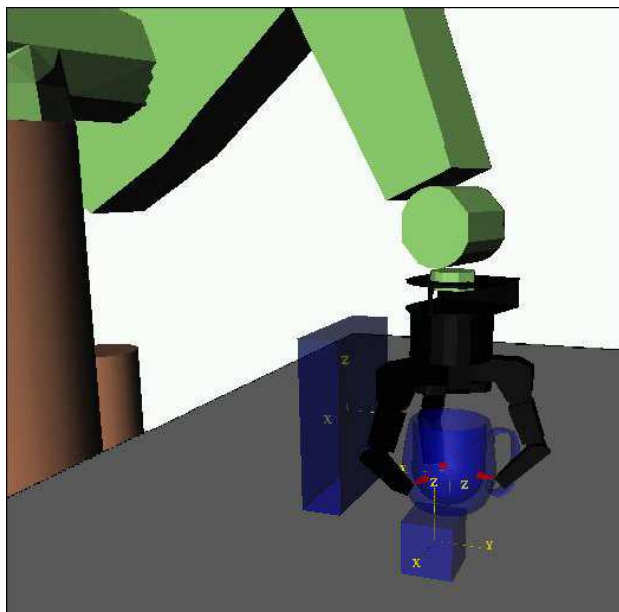


Figura 33 – Obstáculos sendo avaliados no momento da tentativa de preensão
(MILLER et al., 2003)

A última etapa avalia todos os resultados adquiridos até o momento através de valores estáticos de qualidade atribuídos a cada resposta. Estes valores estáticos de qualidade são encontrados através de uma avaliação feita com informações pertencentes aos cones de atrito proporcionados no contato dos dedos com o objeto.

Esse trabalho apresenta uma solução 3-D para o problema e faz uso de um simulador de robô articulado.

3.5. Características relevantes ao tema proposto

Pode-se perceber em todos os trabalhos estudados a necessidade de ser bem definida a geometria do objeto bem como a garra que realiza a ação da pega. Alguns trabalhos utilizam algoritmos que fazem uma análise do espaço 3-D e o reduz a um espaço 2-D como é o caso do trabalho desenvolvido por Smith et al. (1999) e do trabalho desenvolvido por Morales et al. (2006). Porém, este trabalho resolve o problema da preensão fazendo uso dos modelos 3-D de todos os objetos da cena buscando criar algoritmos de rápida execução. O trabalho desenvolvido por Miller et al. (2003) mostra uma solução algorítmica simples e rápida computacionalmente dividindo o objeto em várias geometrias primitivas e analisando-as de maneira isolada.

Outro fato interessante é que o tipo de robô manipulador apenas contribui no problema da preensão determinando a orientação e localização da garra, sendo estas informações necessárias para o cálculo da cinemática inversa. Ou seja, apenas a localização e orientação da garra são afetados pelo tipo de robô utilizado.

Foram utilizados também somente objetos rígidos com massa disposta homogeneamente em todos os trabalhos analisados. O uso de objetos com estas propriedades físicas simplifica a determinação do centro de massa do objeto.

Em algumas propostas foram incluídos cálculos de física para garantir que a preensão seja efetuada de uma maneira que apresenta maior semelhança com mundo real, como foi feito por Smith et al. (1999) e Miller et al. (2003). Segundo Smith (1999), o tempo de processamento da aplicação desenvolvida, utilizando cálculos físicos, está na ordem de $O(n^3)$. Como foi esperado o máximo de desempenho possível na aplicação proposta neste trabalho, o uso de cálculos de física serão evitados, sendo utilizados somente raciocínios geométricos semelhantes aos que foram implementados no trabalho apresentado por Sunil e Pande (2003).

Para que seja garantida a pega mesmo existindo pequenas variações no ambiente, Smith et al. (1999) e Morales et al. (2006) adicionaram uma tolerância na angulação de propagação dos cones de atrito projetados internamente ao objeto. Dessa forma, podem-se obter respostas à pega com variações mínimas na angulação aumentando o domínio da solução do problema.

Raciocínios geométricos e heurísticas também são utilizados no trabalho desenvolvido por Miller et al. (2003) e na solução apresentada por Sunil e Pande (2003). Heurísticas de eliminação de soluções impossíveis, que utilizam raciocínios geométricos, estão presentes nos dois trabalhos citados. Regras como verificar se o objeto encontra-se interno ao volume de trabalho do robô ou se a abertura máxima da pinça é muito pequena em comparação ao tamanho do objeto, podem ser eficientes para eliminar logo no início da execução situações impossíveis de pega.

Pode-se perceber que todos os trabalhos apresentados visam obter soluções ótimas em relação a objeto-garra sugerindo as melhores posições e orientações para preensão, alguns mostrando ao usuário uma seqüência de passos a serem seguidos pelo robô para garantir a melhor forma encontrada pelo aplicativo através da programação automática (VETORAZZI, 1994, p. 19). Isto elimina a etapa de programação do robô, fazendo com que o usuário não se preocupe em como será acessado o objeto escolhido pela máquina manipuladora. Porém, este

tipo de abordagem não avalia os usuários que necessitam de conhecimento em programação *off-line* de robôs.

Desta maneira, simuladores com propósitos didáticos necessitam fazer a avaliação da disposição do robô e do objeto impostos pelo usuário aprendiz. Ou seja, partindo da disposição dos objetos no ambiente virtual, modificado pelo usuário, o simulador mostra uma resposta positiva ou negativa sobre a preensão partindo das informações geométricas apresentadas. Caso a resposta seja negativa mostra-se ao usuário o motivo da análise.

Portanto, como o simulador em questão tem potencial para o uso didático é necessário desenvolver esta funcionalidade.

4. SOLUÇÃO

A solução para o problema apresentado neste trabalho consiste em um conjunto de algoritmos, denominados Raciocínios Geométricos, capazes de avaliar a possibilidade de preensão de objetos. Fazendo uso das disposições da pinça do robô Scorbob virtual, do objeto alvo para preensão e de outras informações existentes na cena, os Raciocínios Geométricos apresentam uma resposta positiva de preensão, caso o ambiente a favoreça, ou negativa caso contrário. Ou seja, o usuário posiciona a pinça do robô em uma posição que ele achar mais favorável à pega do objeto e, quando for solicitado o fechamento dos dedos da pinça para efetuar a suposta preensão, inicia-se um conjunto de tratamentos que avalia se realmente é possível, ou não, pegar o objeto selecionado.

A lógica utilizada para definição de preensão possível ou não, consiste em uma lógica negativa. Não se pode ter certeza que a preensão realmente ocorre com o uso somente de informações geométricas, porém pode-se ter certeza caso a situação seja reprovada. Sendo a situação avaliada e reprovada pelo algoritmo de Raciocínios Geométricos, ela não promove uma preensão estável em um ambiente real. Devido ao fato de não serem satisfeitas as restrições geométricas durante a preensão, pode-se afirmar que a situação pode levar a uma preensão onde o objeto caia ao ser movimentado, possam ocorrer torções ou outros fatos indesejáveis.

Os Raciocínios Geométricos utilizam informações como: tipo de geometria, orientações espaciais, localizações e dimensões, tanto da garra como do resto da cena, que devem ser definidas para que seja possível realizar as análises algébricas propostas na solução.

Como é avaliada a preensão de diferentes tipos de sólidos rígidos primitivos (paralelepípedo, cilindro e esfera), fez-se a necessidade de garantir uma representação que atenda as análises a serem realizadas posteriormente. Desta forma, como são submetidas diferentes geometrias, é necessário que cada tipo de sólido tenha um tratamento especial, visto que não são utilizados conceitos de física que fornecem um tratamento mais geral.

Também é necessário definir a geometria da pinça instalada no robô Scorbob virtual, pois é esta que efetuará a ação da preensão. Desta maneira, a preensão tem tratamentos distintos para cada tipo de garra e para cada tipo diferente de objeto que pode estar na cena.

Como este trabalho é focado no simulador projetado pelo LARVA (2007), foi representado somente o tipo de garra *parallel jaw gripper*, existente no protótipo.

Estas informações sobre a cena, incluindo garra e objetos, são manipuladas pelos diferentes Raciocínios Geométricos, sendo estes dispostos de maneira organizada. No momento da execução da preensão, o simulador identifica qual dos algoritmos se encaixa melhor para o tratamento do tipo de objeto geométrico a ser pego.

De modo geral, este trabalho propõe um algoritmo que faz uso do tipo de modelagem geométrica B-rep implementado na linguagem Java e integrado ao VRML. Este algoritmo possui a função de definir se a posição e orientação da pinça e do objeto, determinados pelo usuário aprendiz, promove, ou não, a preensão do objeto desejado. Caso não seja possível a preensão, o simulador informa ao usuário qual o problema ocorrido numa área de *feedback* ao usuário (vide item 4.1 do trabalho).

Antes de detalhar melhor a lógica do algoritmo, é necessário especificar:

1. A interface e o seu funcionamento;
2. As informações que representam a garra, o objeto e o resto da cena;
3. As situações a serem tratadas;
4. Os raciocínios geométricos respectivos;

Todos os vetores que serão apresentados neste capítulo são normalizados.

4.1. A Interface

O Simulador do Robô Virtual ER-4PC, mostrado na Figura 34, vem sendo desenvolvido pelo Laboratório de Realidade Virtual Aplicada (LARVA, 2007); (REDEL e HOUNSELL, 2004); (ZWIRTES, 2004); (SANTOS, 2007a) com a utilização das linguagens VRML e Java. Quanto a sua interface, uma vez selecionado o objeto a ser pego, pode-ser ter duas formas de funcionamento:

1. O usuário posiciona a garra na posição que supõe promover a preensão e então solicita o teste por Raciocínios Geométricos recebendo então um veredicto. Este tipo de aplicativo seria mais bem empregado com o intuito de ensino (abordagem discreta);
2. À medida que o usuário manipula a estrutura robótica, os Raciocínios Geométricos são executados informando uma mensagem no primeiro instante em que a preensão for válida. Este tipo de *software* seria mais bem empregado para ambientes de produção (fabris) (abordagem contínua).

A abordagem discreta tem algumas vantagens: sob o ponto de vista de *performance* o algoritmo executa somente quando o usuário solicita a avaliação da preensão, fazendo com que o processamento dos Raciocínios Geométricos não seja concorrente com o sistema de detecção de colisão e; como o usuário indica os momentos em que acredita ser possível a preensão, estas crenças podem ser avaliadas pelo educador. A abordagem contínua possui a vantagem de facilitar o processo de interação levando mais facilmente à identificação de situações de preensão válidas.

A interface do simulador pôde ser dividida em três áreas distintas, como destacado na Figura 34: a área de visualização 3-D, a área de criação do programa do robô e a área de entrada e saída de dados. A área de visualização 3-D, que representa o simulador propriamente dito, está representada na área 1 da Figura 34, onde o usuário pode efetuar manipulação direta do robô e visualizar os movimentos a serem realizados por ele. É possível interagir com o robô mantendo pressionado o botão esquerdo do *mouse* sobre uma junta ou um elo dele e arrasta-lo, causando a respectiva movimentação.

A área 2, existente na Figura 34, representa a área de criação de um programa simplificado para a programação *off-line* do robô Scorbot (REDEL e HOUNSELL, 2004). Ainda na área 2, existem elementos com a função de incluir objetos aleatórios na cena (SANTOS et al., 2007b) para que possa ser calculada a detecção de colisão entre o robô e os objetos gerados (SANTOS, 2007a). É na área 2 onde se encontra uma caixa de texto capacitada em mostrar informações de *feedback* para o usuário (último elemento inferior da área 2). Existe também uma interface para a entrada e saída de dados, representados na área 3 da Figura 34, onde o usuário pode efetuar a movimentação das juntas através dos botões de incremento e decremento, ou definir diretamente em uma caixa de texto os valores dos ângulos correspondentes a cada junta. Ao alterar um dos valores de ângulo das juntas, automaticamente o robô virtual assume uma nova disposição movimentando-se de acordo com o respectivo valor modificado.

Apesar de já estarem incorporados elementos de detecção de colisão na interface do simulador, somente estes elementos não atenderam ao propósito deste trabalho. Para isto foi necessária a inclusão de alguns elementos na interface. Para que seja realizadas tais modificações, foi necessária a remoção de alguns elementos já existentes na interface devido a grande densidade informacional existente. Os dois elementos localizados ao lado do botão

(Executar), mostrados na área 2 da Figura 34 foram removidos da interface do simulador por não conter funções implementadas. Os elementos a serem removidos, porém, eram desprovidos de qualquer função, fazendo com que o simulador não perca nenhuma funcionalidade implementada até então.

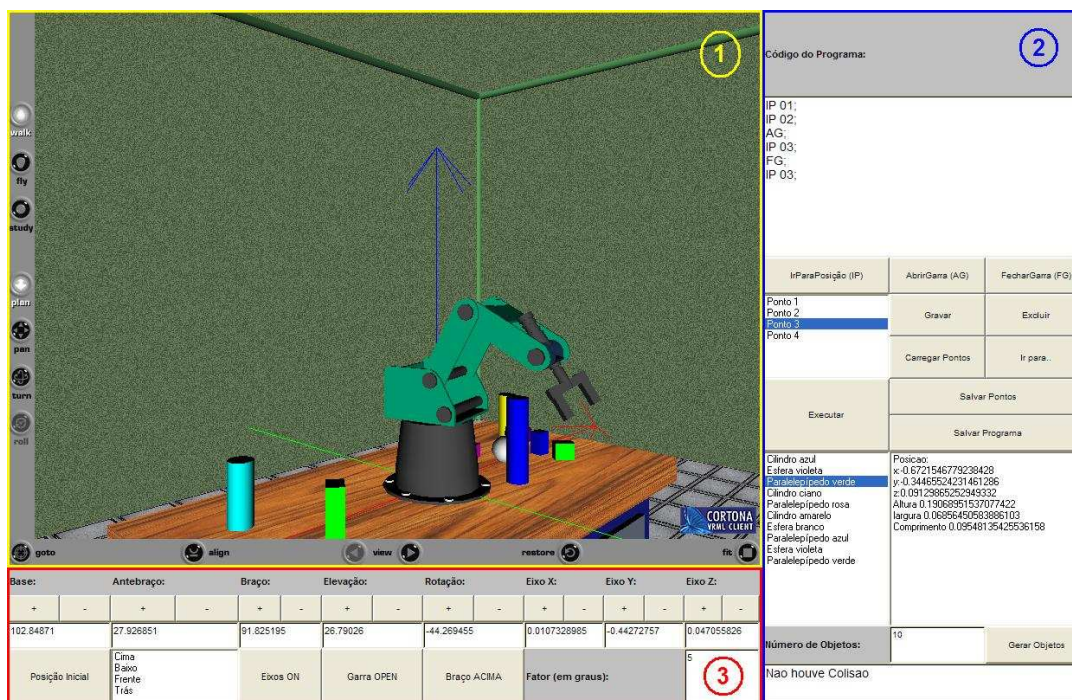


Figura 34 – Simulador do robô Scrobot ER-4PC
(SANTOS, 2007a)

A remoção destes dois elementos faz com que o botão logo ao lado deles (Executar) fosse menor e mais alongado, ocupando toda a extensão horizontal da área 2 mostrada na Figura 34. É possível visualizar na Figura 35 (antes) que os botões “Salvar Pontos” e “Salvar Programas” foram retirados. Isto aumentou a área livre na interface possibilitando a inserção dos novos elementos para avaliação da prensão. Apesar de estes elementos removidos já estarem na interface, eles são desprovidos de funções, possibilitando as suas remoções.

Alguns elementos existentes foram redimensionados e outros foram reposicionados de forma que seja coerente as suas funções com as suas localizações na interface, como pode ser visualizado na Figura 35. Elementos de geração de objetos estão localizados em uma mesma área da interface, enquanto que elementos de avaliação de prensão estão localizados em outra área. A caixa à esquerda da Figura 35 (depois) representa os elementos responsáveis pela geração de objetos, já a caixa à direita representa os elementos adicionados e com a função de avaliar a prensão.

O último elemento localizado na parte inferior possui a função de mostrar para o usuário mensagens de *feedback*, estas vindas tanto da detecção de colisão como da avaliação da prensão. Percebe-se também na Figura 35 que a área de programação do robô foi expandida, fazendo com que mais linhas do código possam ser visualizadas pelo programador. Percebe-se na parte superior (antes) havia uma área na interface mau utilizada, sendo esta aproveitada na nova versão da interface. A Figura 35 mostra um comparativo entre a área 2 da interface antiga e a área 2 da interface projetada.

Código do Programa:		Código do Programa:	
IP 01; IP 02; AG; IP 03; FG;		IP 01; IP 02; AG; IP 03; FG;	
IrParaPosição (IP)	AbrirGarra (AG)	IrParaPosição (IP)	AbrirGarra (AG)
Ponto 1	Gravar	Ponto 1	Gravar
Ponto 2	Excluir	Ponto 2	Excluir
Ponto 3		Ponto 3	
	Carrregar Pontos		Carrregar Pontos
	Ir para..		Ir para..
Executar	Salvar Pontos	Executar	Salvar Pontos
	Salvar Programa		Salvar Programa
Cilindro azul Cilindro violeta Cilindro verde Esfera ciano Paralelepípedo rosa Cilindro amarelo Paralelepípedo branco Paralelepípedo azul Cilindro violeta Paralelepípedo verde	Posicao: x:0.9765730835497337 y:-0.25318535440394315 z:-0.0524339397457586 Altura: 0.3736280911921137 Raio 0.05	Cilindro azul Esfera violeta Esfera verde Cilindro ciano Esfera rosa Paralelepípedo amarelo Esfera branco Esfera azul Esfera violeta Cilindro verde	Objeto Cilindro ciano Posicao: x:-0.6039727048728099 y:0.118768897071448211 z:-0.20201237122562204 Altura: 0.25503796676093826 Raio 0.058743894974773045
Número de Objetos: 10	Gerar Objetos	Qtd Obj: 10	Valor Folga Abordagem: 9
geracao completa		Gerar Objetos	Testa Prensão

Antes Depois

Figura 35 – Comparativo entre a interface antiga e projetada
(LARVA, 2007)

Os elementos incluídos são: uma caixa de texto que captura o parâmetro de Folga de Abordagem (explicado no Item 4.5.1.1. deste trabalho) e um botão com a função de avaliar a prensão. Percebeu-se a necessidade de incluir o botão de teste de prensão para evitar que a avaliação seja feita de forma automática no momento de fechar a garra do robô. Como o ato de fechar ou abrir a garra do robô nem sempre está associado à prensão de objetos, será posto a execução do algoritmo de Raciocínios Geométricos somente ao pressionar este botão. É necessário em certas situações, por exemplo, fechar a garra somente com o intuito de ser

evitada uma colisão entre um dedo da garra e um objeto no ambiente durante a movimentação da estrutura robótica.

4.2. Informações Necessárias

Antes de partir para o desenvolvimento da solução, foi necessário organizar as informações a serem utilizadas. Estas informações são: a representação da garra do robô e a representação dos objetos.

Como se pode perceber na Figura 36, foi necessário obter informações sobre a geometria e disposições tanto do objeto como da garra, bem como informações sobre o ambiente virtual. São importantes estes dados de entrada do algoritmo para que a avaliação da preensão seja realizada através de raciocínios geométricos, semelhantes aos implementados por Miller et al. (2003). Após o término do processamento do algoritmo, é mostrada uma resposta que informa se é possível, ou não, a preensão de acordo com os dados de entrada. Caso a preensão não seja satisfeita é mostrada uma mensagem de *feedback* para o usuário informando-o sobre que aconteceu através da caixa de texto de *feedback* existente na interface. Caso a preensão ocorra de maneira satisfatória, o objeto é incluso na árvore estrutural que define o robô Scrobot implementada em VRML através na função *addChildren()*. Isto faz com que o objeto se mova juntamente com toda a estrutura do robô, caso uma junta seja rotacionada, por exemplo. Desta forma, foi necessário especificar os dados referentes à garra, aos objetos e ao ambiente, mostrados na Figura 36. Definidas tais informações, serão explicadas como os Raciocínios Geométricos estão divididos e como eles se comportam.

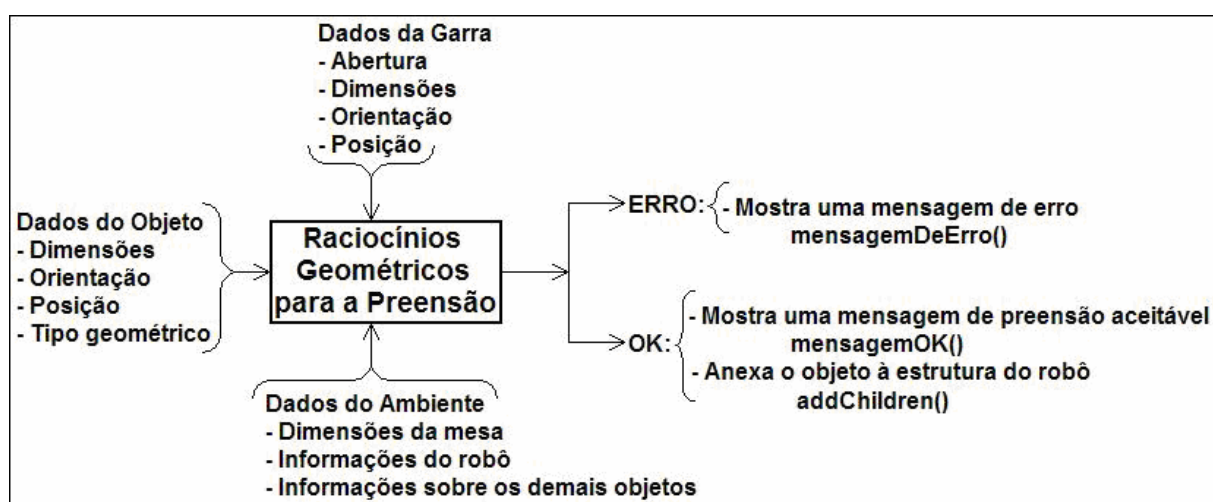


Figura 36 – Entradas e saídas de dados do algoritmo proposto

4.2.1. Dados da Garra

Existem atualmente diferentes tipos de geometrias de garras, sendo que cada uma necessita de uma análise isolada. Cada garra precisa tratar a prensão de objetos de acordo com a sua geometria e dimensões. Desta forma, existem diferentes conjuntos de restrições que são voltados a cada tipo de garra em particular, sendo que essas restrições precisam satisfazer todos os sólidos existentes no ambiente. Como o trabalho é focado no modelo virtual do robô Scorbót ER-4PC, foram apenas analisados raciocínios geométricos para a garra do tipo pinça de dois dedos, esta acoplada no robô virtual.

Como são postos em prática algoritmos que necessitam de informações cartesianas, existiu a necessidade de definir a geometria da garra do robô bem como sua representação. Dados exatos de altura da pinça, tamanho dos dedos e abertura total, por exemplo, são de extrema importância para garantir que uma resposta, exata ou aproximada, seja encontrada.

O tipo de pinça utilizada no simulador do robô Scorbót ER-4PC denomina-se *parallel jaw gripper*. Esta pinça possui dois dedos paralelos entre si, como mostra a Figura 37 e a Figura 10, que se movimentam em direções opostas. A Tabela 3 mostra os parâmetros que caracterizam a pinça mostrada na Figura 37 com sua respectiva descrição e valores no ambiente virtual do VRML.

Descrição	Parâmetros	Valor (metros)
Altura Total (<i>Height</i>)	<i>H</i>	0.25875
Altura dos dedos	<i>A</i>	0.099
Largura dos dedos	<i>L</i>	0.036
Profundidade dos dedos	<i>P</i>	0.036
Abertura máxima entre os dedos (<i>Opening</i>)	<i>O</i>	0.1125

Tabela 3 – Valores limites constantes da geometria da garra

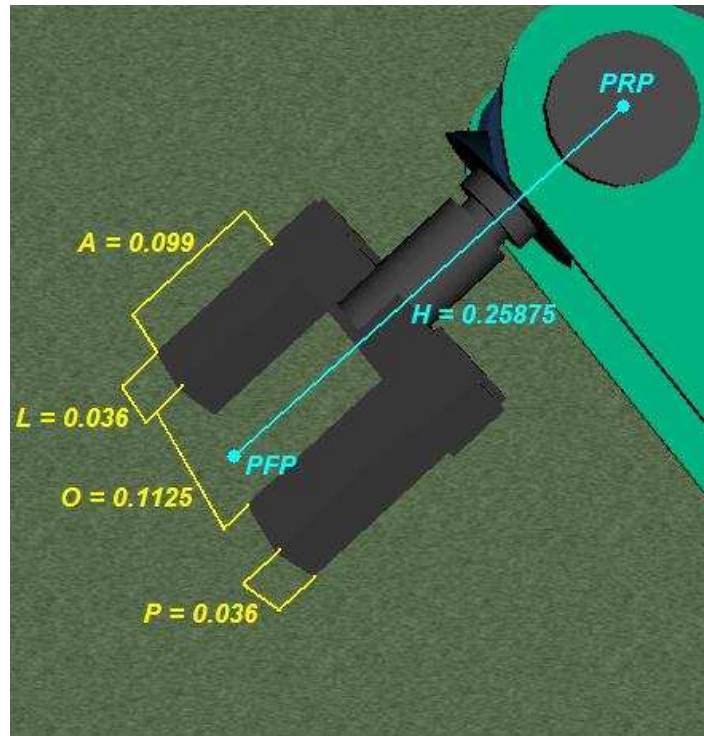


Figura 37 – Dimensões da pinça *Parallel Jaw Gripper*

Além das dimensões que a garra possui, foi necessário definir sua orientação no espaço. Como a pinça pode estar em diferentes disposições no ambiente, percebeu-se a necessidade de definir também vetores que são presos à garra. Esses vetores são rotacionados juntamente com a pinça, fazendo com que seja definida, através deles, a orientação da pinça. Como são mostrados na Figura 38, estes vetores estão dispostos de maneira a formar o sistema de coordenadas da pinça, sendo que este sistema localiza-se no ponto *PRP* (Ponto de Referência da junta do Punho), mostrado na Figura 37. A Tabela 4 descreve o sistema de coordenadas que está associado à orientação da garra mostrada na Figura 38.

Vetores	Significado
$\vec{U}_{pg}$	Informa para que lado a parte superior da pinça está voltada
$\vec{A}_{pg}$	Informa para qual direção a pinça está Apontando
$\vec{El}_g$	Complementa o sistema de Coordenadas, Eixo lateral

Tabela 4 – Eixo de coordenadas da pinça

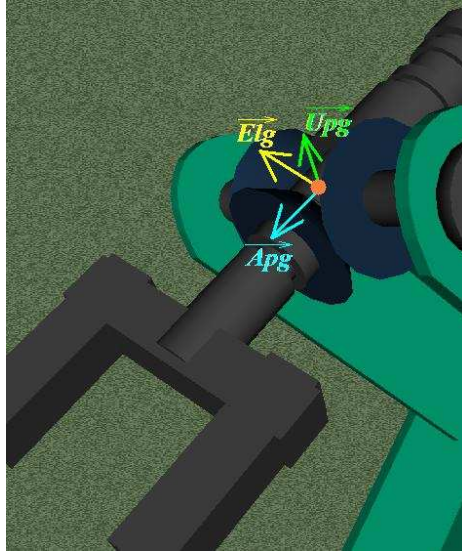


Figura 38 – Vetores de orientação da pinça *Parallel Jaw Gripper*

Foi utilizado, como referência para os cálculos, o ponto *PFP* (ponto final da pinça), localizado entre a ponta dos dedos, conforme a Figura 39. Para a obtenção deste ponto, deve-se partir do ponto *PRP* e, a partir daí, acrescentar informações específicas da garra, como a altura total (H) e o vetor de aproximação da garra ($\vec{A_{pg}}$), necessárias para a obtenção do *PFP*. É derivado o *PFP* através de uma translação a partir do *PRP*, de módulo H no sentido do vetor de aproximação da garra ($\vec{A_{pg}}$). Esta translação é semelhante às realizadas durante as etapas de cálculo de Cinemática Direta.

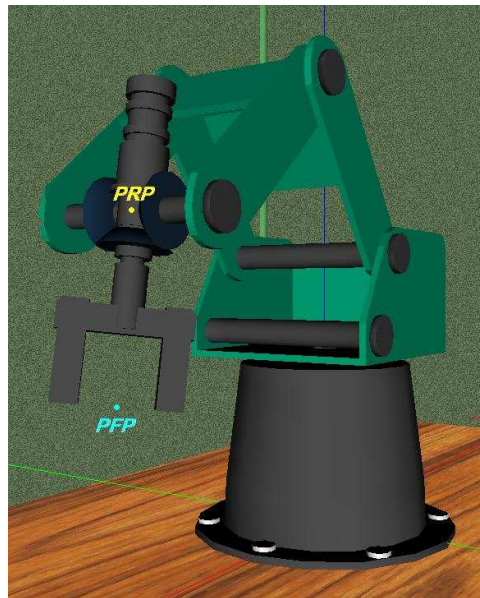


Figura 39 – Pontos de referência para a prensão

Em várias situações, é necessário encontrar um dos pontos através das informações algébricas existentes no outro. Caso seja calculada a cinemática inversa, é necessário encontrar o *PRP* através do *PFP*. Caso seja calculada a cinemática direta, o contrário ocorre.

Para o caso da Cinemática Direta, o *PRP* é encontrado fazendo uso dos parâmetros cinemáticos encontrados na estrutura de elos e juntas existente no robô. Alguns destes valores são constantes e outros são encontrados através da angulação existente em cada junta do robô. Através de um conjunto de derivações iterativas, encontra-se o sistema de coordenadas final, sendo este localizado no punho do robô. Estas derivações determinam para cada junta do robô um sistema de coordenadas. Este sistema iterativo é denominado Representação de Devanit-Hartenberg (SCHILLING, 1990). Para se obter o valor do *PFP*, caso seja realizada a Cinemática Direta, basta encontrar o resultado através da seguinte fórmula:

$$PFP = PRP + \overrightarrow{Apg} * H \quad (4.1)$$

Sendo:

- Ponto Final da Pinça: *PFP*.
- Ponto de Referência do punho: *PRP*.
- Vetor de aproximação da Garra: $\overrightarrow{Apg}$.
- Altura da Pinça: *H*.

Para o caso da Cinemática Inversa, partindo do ponto *PFP*, basta derivar *PRP* através da fórmula:

$$PRP = PFP + (-\overrightarrow{Apg}) * H \quad (4.2)$$

4.2.2. Dados dos Objetos

Os objetos são sólidos rígidos primitivos, ou seja, não são deformáveis, e possuem o formato de esfera, paralelepípedo ou cilindro. Os tipos de sólidos como paralelepípedo e cilindro podem ser dispostos tanto em pé como deitado. Nos dois casos, eles estão em pé caso o vetor *Up* do objeto esteja paralelo ao vetor $\vec{Y}$, e estão deitados caso a angulação esteja

perpendicular. Desta forma, também foi necessário especificar as restrições geométricas que cada tipo de objeto impõe bem como sua orientação.

Como é possível perceber, tanto a garra como os objetos possuem em suas representações vetores de orientação. Os vetores de orientação dos objetos têm suas notações semelhantes aos vetores de orientação da garra, porém diferem somente na última letra de seus nomes. No caso dos vetores de orientação da garra, suas notações terminam com a letra ‘g’, como, por exemplo, o vetor $\overrightarrow{Upg}$. Já no caso da notação dos vetores dos objetos, eles terminam suas notações com a letra ‘o’, como, por exemplo, o vetor $\overrightarrow{Upo}$.

Durante a simulação é possível gerar randomicamente os objetos a serem inseridos na cena. Seus valores de dimensão, orientação e posição são aleatórios, sendo que os valores de dimensão são necessariamente maiores que zero. Como se pretendeu que cada sub-rotina existente nos Raciocínios Geométricos pudesse vir a ser executada, foram utilizados valores durante a geração dos objetos que possam gerar objetos das mais diferentes formas e tamanhos. Isto fez com que o algoritmo de Raciocínios Geométricos pudesse ser testado nos seus diferentes módulos.

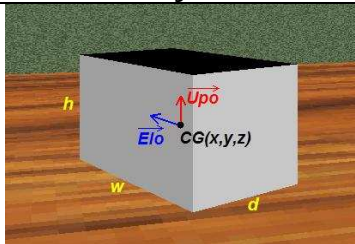
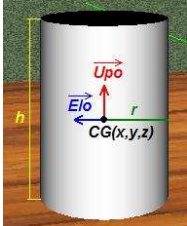
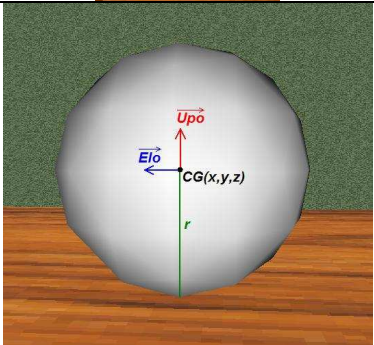
Objeto	Dados
	$\overrightarrow{Upo}$: Vetor <i>Up</i> do objeto; $\overrightarrow{Elo}$: Vetor de <i>Eixo lateral</i> do objeto; <i>CG</i> : Centro de Gravidade do objeto; <i>h</i> : Altura do objeto (<i>height</i>); <i>w</i> : Largura do objeto (<i>width</i>); <i>d</i> : Profundidade do objeto (<i>depth</i>).
	$\overrightarrow{Upo}$: Vetor <i>Up</i> do objeto; $\overrightarrow{Elo}$: Vetor de <i>Eixo lateral</i> do objeto; <i>CG</i> : Centro de Gravidade do objeto; <i>h</i> : Altura do objeto (<i>height</i>); <i>r</i> : raio do objeto.
	$\overrightarrow{Upo}$: Vetor <i>Up</i> do objeto; $\overrightarrow{Elo}$: Vetor de <i>Eixo lateral</i> do objeto; <i>CG</i> : Centro de Gravidade do objeto; <i>r</i> : raio do objeto.

Tabela 5 – Dados de representação específicos dos objetos

Ao gerar os objetos, percebe-se que existem valores específicos para cada tipo de geometria. Para um objeto do tipo esfera, por exemplo, basta definir o seu ponto central e o seu raio, enquanto que para um cilindro, é necessário conhecer a sua altura também. Desta forma, foram definidas as representações peculiares ao tipo de geometria. São mostradas na Tabela 5 as representações dos objetos através de uma imagem e as explicações referentes aos dados apresentados na imagem. Percebe-se que a tabela limita-se em ilustrar os dados referentes ao respectivo tipo do objeto.

Foi feita uma tentativa de utilizar estes valores vindos diretamente do ambiente VRML, porém percebeu-se a limitação ao acessar estes dados especificamente. Valores numéricos de orientação, posição, e dimensão criados no ambiente VRML possuem acesso limitado, sendo somente utilizado internamente ao VRML. Desta forma, foram criadas classes em Java que armazenam os dados pertencentes a estes objetos, havendo, assim, certa duplicidade de informações. Ao serem criadas estas informações, aproveitou-se para definir os parâmetros e sistema de coordenadas acompanhando a nomenclatura adotada para a garra.

Existem informações adicionais que, independente do tipo de geometria, complementam a representação dos objetos. Para que sejam realizados alguns testes existentes nos Raciocínios Geométricos, todos os objetos têm um volume envolvente. A função deste volume é testar a proximidade dele em relação à pinça do robô Scorbot. Este volume é uma esfera e é dado somente por um raio e com seu ponto central localizado no centro de gravidade do objeto. Foi optado por um volume envolvente esférico devido ao fato de que o cálculo de distâncias 3-D é computacionalmente rápido e conceitualmente simples. Apesar de o raio do volume envolvente ser um dado existente em todos os tipos de geometrias, o cálculo desta informação é específico ao tipo de volume.

Analisando especificamente o paralelepípedo na Tabela 5, este pode variar suas dimensões nos três eixos de coordenadas. Desta forma, é indispensável ter conhecimento de seus valores de aresta para cada eixo, seus vetores de orientação e seu centro de gravidade. Para encontrar o raio da esfera envolvente (R) do objeto paralelepípedo, é realizado um cálculo com a Fórmula 4.3:

$$R = \frac{\sqrt{h^2 + w^2 + d^2}}{2} \quad (4.3)$$

Sendo:

- R : O raio da esfera envolvente para Paralelepípedos.

Apesar de a representação de Cilindros possuir dois vetores de orientação, como é mostrado na Tabela 5, para este trabalho bastou obter o vetor Up . Os valores de altura, raio, centro de gravidade e o vetor Up do cilindro ($\overrightarrow{Upo}$) são suficientes para representar corretamente o objeto no espaço tridimensional. Para encontrar o raio da esfera envolvente (R) do objeto cilindro, é realizado um cálculo com a Fórmula (4.4) baseada na Figura 40:

$$R = \sqrt{r^2 + \left(\frac{h}{2}\right)^2} \quad (4.4)$$

Sendo:

- R : O raio da esfera envolvente para Cilindros.

Cilindro Corte lateral

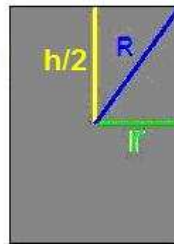


Figura 40 – Corte lateral do cilindro

A representação do objeto Esfera também possui vetores de orientação, mostrado na Tabela 5, porém é desnecessário guardar qualquer informação de orientação para a esfera para este trabalho também. Desta maneira, os dados necessários para a representação são apenas o centro de gravidade e o raio, estando ausente tais vetores de orientação. Para encontrar o raio da esfera envolvente (R) do objeto esfera, simplesmente é atribuído o valor de seu próprio raio ($R = r$).

Estas entidades foram criadas em classes Java para que fosse possível realizar cálculos de colisão também avaliar a preensão. Os dados referentes à orientação e dimensão dos objetos são adquiridos através dos volumes envolventes criados para os cálculos de colisão implementados por Santos (2007a).

4.2.3. Dados do Ambiente

O ambiente também possui informações para que os raciocínios geométricos possam ser postos em execução. Poucas informações do ambiente são necessárias, porém não menos importantes elas são. As informações de dimensão da mesa, como largura, comprimento e altura são utilizadas pelo simulador. Os dados pertencentes ao robô também são necessários para o simulador, pois é através deles que são calculadas a Cinemática Direta e Inversa. As informações dos outros objetos são mostradas na interface do simulador e podem vir a serem utilizadas pelos raciocínios geométricos.

4.3. Estrutura da Solução

O algoritmo proposto busca solucionar o problema da preensão especificamente para o ambiente em que o robô ER-4PC virtual está inserido. Atualmente a interação com o robô está implementada de maneira que toda a sua movimentação seja programada pelo usuário. Desta forma o simulador tem a capacidade de avaliar se ocorre, ou não, a preensão na posição/situação definida pelo usuário/aprendiz.

Como podem existir vários objetos inseridos no ambiente virtual, é necessária a identificação de qual objeto será o alvo do robô para que seja possível realizar os testes necessários. A solução para este problema está na possibilidade de o usuário do simulador poder selecionar previamente, através da interface do aplicativo, qual objeto será o alvo para a preensão. Através do objeto escolhido e da posição imposta pelo usuário, é avaliada a possibilidade de preensão de acordo com o tipo de geometria, dimensões e orientações do objeto e da garra.

Quando se evita o uso de cálculos físicos e de detecção de colisão devido ao tempo gasto durante o processamento, é necessário utilizar outro tipo de abordagem que possa oferecer o tratamento adequado ao problema. O uso de heurísticas para tratamento da preensão se mostra uma solução possível, como as implementadas por Sunil e Pande (2003) e apresentadas por Smith et al. (1999).

A sequência de passos seguida pelo algoritmo durante a preensão, denominada neste trabalho como regras para *Robot Grasping*, consiste em duas etapas principais:

- **Exclusão:** Etapa que possui raciocínios geométricos responsáveis por identificar situações impossíveis de promoverem preensão. Basta um destes raciocínios ser válido para se ter certeza de que não haverá preensão;

- **Confirmação:** Etapa que possui uma série de raciocínios divididos e organizados para o tratamento de diferentes geometrias. Esta etapa é mais específica ao tipo de objeto submetido. Cada raciocínio avaliado como válido confirma aquela respectiva condição para a preensão mas, não necessariamente esta é a definitiva. Em último caso a confirmação definitiva virá somente com o processo de *try-out*. É na etapa de Confirmação que é descoberto qual tipo geométrico é avaliado e, conseqüentemente, é determinado qual raciocínio geométrico será utilizado.

Estas duas etapas principais da solução serão mais detalhadas adiante.

4.3.1. Etapa da Exclusão

Esta etapa tem como função eliminar casos impossíveis para preensão. Casos que podem ser excluídos logo no início e possuem a disposição espacial do robô e do objeto sem as mínimas condições geométricas, como o fato da pinça estar posicionada muito longe do objeto, por exemplo. Este tipo de abordagem mostrou-se presente nos trabalhos desenvolvidos por Sunil e Pande (2003) e por Miller et al. (2003).

Não é levado em consideração o tipo de geometria do objeto alvo, fazendo com que esta etapa possua análises mais gerais. São feitas estas análises iniciais de eliminação para se evitar avaliações posteriores desnecessárias que fazem uso de testes mais refinados próprios para a geometria específica do objeto. Caso algum destes testes iniciais seja aprovado, o algoritmo retorna uma mensagem de erro informando que a disposição da garra e do objeto não promove a preensão.

Os Raciocínios Geométricos de Exclusão estão relacionados a duas questões próprias do robô em relação ao objeto: a anatomia do robô e a garra. O esquema mostrado na Figura 41 ilustra os testes de Exclusão dispostos de acordo com a explicação dada. Os testes de exclusão correspondentes à garra (órgão terminal) fazem uso de informações referentes somente à garra e ao objeto, enquanto que os testes correspondentes à anatomia (estrutura do robô) fazem uso de informações da estrutura robótica e do objeto. Os testes correspondentes à anatomia do robô estão relacionados às dimensões dos elos e limites de junta do robô, portanto, fazem verificações se o robô alcança o objeto.

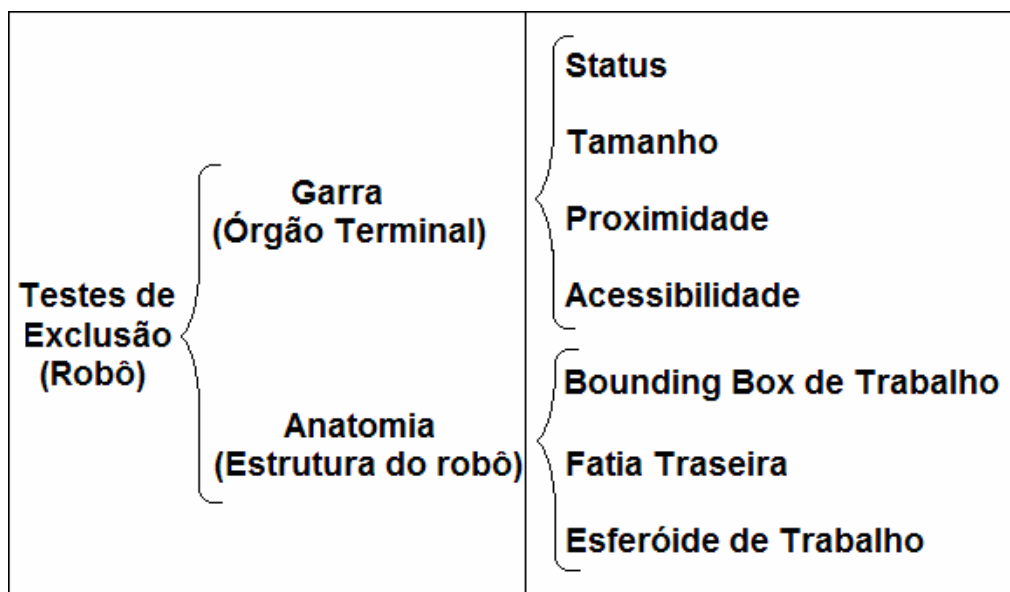


Figura 41 – Esquema dos testes de Exclusão

Estes testes foram dispostos em uma seqüência que inicia pelos casos mais comuns de exclusão para os menos comuns. Desta maneira elimina-se a parte dos casos de exclusão logo no início do algoritmo resultando uma resposta rápida ao problema da preensão, portanto, eficiência. A seqüência e descrições dos testes de exclusão é: Status da garra, Tamanho do objeto *versus* Garra, Proximidade da Garra, Acessibilidade, *Bounding Box* de Trabalho, Fatia Traseira e Esferóide de Trabalho.

- **Status da garra:** Caso a garra esteja fechada, será impossível pegar o objeto, por isso, esta situação é eliminada. Como está implementada a detecção de colisão no robô virtual, este teste não é necessário ser realizado. Como é impossível que a garra fechada seja levada até uma posição possível de preensão, este teste não foi implementado no simulador;
- **Tamanho do objeto *versus* Garra:** Caso o menor valor das três dimensões da OBB do objeto for maior do que o valor da abertura da garra (O). Quando este teste é comprovado e como a OBB está ligada somente a geometrias primitivas, a situação é eliminada, pois é impossível pegar um objeto com tais dimensões. Esta situação ocorre quando os valores de diâmetro e altura de um cilindro forem maiores do que a abertura máxima da pinça, por exemplo;
- **Proximidade da Garra:** Será excluída a resposta caso o objeto e a garra se encontrem muito longe um do outro, podendo o objeto estar no volume de trabalho do robô, ou

não. Duas hipóteses foram levantadas para testar a proximidade do objeto à garra: através dos limites da OBB do objeto e através de distância euclidiana. Foi optado por avaliar a proximidade através de distância euclidiana devido a simplicidade conceitual e rapidez de processamento. O raio do volume envolvente é calculado como foi explicado no tópico 4.2.2. deste trabalho. O teste de proximidade é baseado na Figura 42 e é feito através da seguinte maneira:

- o Se ($dist(CG, PFP) > R$)
 - Reprovou no teste de Proximidade;

Sendo:

- o *dist*: Uma função que calcula a distância euclidiana entre os dois pontos passados;
- o *CG*: O Centro de Gravidade do objeto;
- o *PFP*: O Ponto Final Pinça.

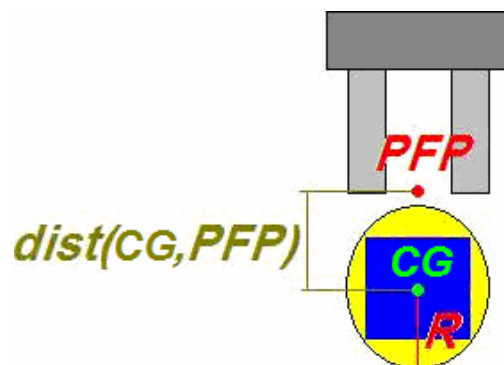


Figura 42 – Exclusão por proximidade

- **Acessibilidade:** São testes que verificam se a pinça tem acesso ao objeto, através de detecções de colisão entre a pinça e o objeto e entre a pinça e outros objetos inseridos em cena. São excluídas neste teste as situações onde ocorrem inúmeras colisões na pinça no momento de realizar a preensão. Apesar de testes de acessibilidade estarem presentes na avaliação da preensão, para o simulador em questão eles foram desnecessários. Devido à inclusão de algoritmos de detecção de colisão ao modelo virtual, implementados por Santos (2007a), não se fazem necessários testes de exclusão levando em conta o acesso da garra ao objeto. A análise de acessibilidade é suprimida pelos testes de colisão que são executados antes de poder ser realizada a

preensão. Caso a pinça chegue aberta em uma posição possível de apreensão, o objeto é acessível a ela, pois cabe ao teste de detecção de colisão decidir a acessibilidade de antemão;

- **Bounding Box de Trabalho:** Este teste verifica se o centro de gravidade do objeto se localiza interno a uma caixa que envolve todo o volume de trabalho do robô. Caso o objeto não se encontre interno ao *Bounding Box* de trabalho, o robô é incapaz de pegá-lo. Esta caixa é criada logicamente e não está inserida graficamente no ambiente virtual. Os eixos da *Bounding Box* são paralelos aos eixos de coordenadas do universo, o que caracteriza um AABB. Este teste é computacionalmente leve, pois são apenas verificados se o centro do objeto se encontra entre valores limitantes nos três eixos de coordenadas;
- **Fatia Traseira:** Como foi visto na Figura 43, o robô Scorbot possui limites de rotação para a base. Desta forma, existe uma área ao qual é inacessível ao robô devido a este limite de rotação (sua parte traseira) mesmo estando dentro do *Bounding Box* de trabalho. O teste Fatia Traseira consiste em verificar se o objeto não está localizado interno a esta área inacessível ao robô devido ao limite de rotação da base. Estima-se pouco tempo de processamento neste teste, pois é verificado apenas se a rotação da base necessária para pegar o objeto está entre os limites determinados;

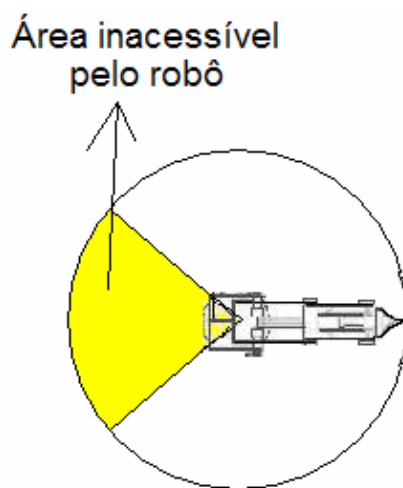


Figura 43 – Área traseira inacessível pelo robô

- **Esferóide de Trabalho:** Será excluído caso o objeto esteja com o seu centro de gravidade externo ao volume de trabalho do robô, impossibilitando o robô de pegá-lo, como mostrado da Figura 44. Idealmente este teste deveria ser feito se os dois

anteriores não excluíssem a apreensão, pois a aproximação do *Bounding Box* de Trabalho não considera a geometria curva do Esferóide de Volume de Trabalho do robô virtual. Desta forma, este teste é feito através do cálculo da cinemática inversa. Para efetivar este teste, o centro de gravidade do objeto é passado ao algoritmo da cinemática inversa e, caso este não retorne uma configuração de junta válida, então não está ao alcance do robô. Devido aos limites de rotação da base do robô e de seu respectivo volume de trabalho (vide Figura 16), o cálculo da cinemática inversa é o que melhor identifica esta situação. Diferente dos testes anteriores (*Bounding Box* de Trabalho e Fatia Traseira), este teste é exato e realmente verifica se o objeto é acessível sem realizar aproximações. Mas é comparativamente mais custoso computacionalmente que os dois anteriores.

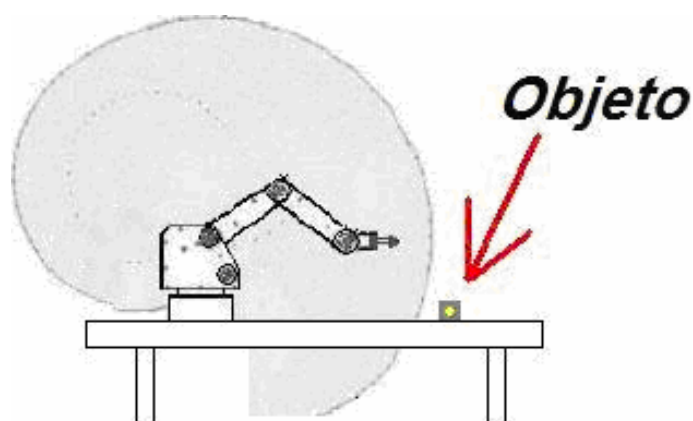


Figura 44 – Exclusão por posição fora do volume de trabalho

Alguns testes de anatomia são computacionalmente leves, como o caso do teste de *Bounding Box* de Trabalho e do teste de Fatia Traseira, porém o teste de Esferóide de Trabalho é custoso. Devido ao fato de o volume de trabalho do robô ser irregular (um esferóide), este teste demanda cálculos de Cinemática Inversa, exigindo, assim, um tempo considerável de processamento. Foram criados os testes de *Bounding Box* de Trabalho e de Fatia Traseira buscando uma otimização do teste onde se deriva o esferóide de trabalho. Observa-se que, mudando-se a garra ou o tipo de robô, é possível que os Raciocínios Geométricos também mudem (uma garra magnética não gera restrições de tamanho) ou até sejam otimizados (um robô SCARA não tem a parte traseira inacessível, por exemplo).

4.3.2. Etapa de Confirmação

Após o término da etapa de Exclusão, o algoritmo inicia o bloco de avaliações mais específicas. Este bloco é separado organizadamente de acordo com cada tipo de geometria. Cada tipo de sólido tem seu tratamento específico através de um conjunto de raciocínios geométricos implementado especificamente para ele, de forma similar a que Miller et al. (2003) classificou em seu trabalho. Desta maneira, é nesta etapa que são utilizadas as informações mais específicas ao tipo de objeto, mostradas nos itens 4.2.1. e 4.2.2. deste trabalho.

Para que comesse a etapa mais específica ao tipo do objeto que será pego, faz-se necessário o conhecimento de qual tipo é o objeto e as suas dimensões e orientações. Mais especificamente é necessário identificar qual será o objeto alvo para que possa ser feita a série de tratamentos adequados para a prensão. Para isto, o usuário aprendiz do simulador pode selecionar qual será o objeto que será pego pelo robô virtual através do objeto selecionado na lista de objetos existente na interface do simulador.

4.4. Possíveis Situações

Devido as diferentes formas possíveis de um objeto ser pego através da pinça, podendo a mesma estar por cima, por baixo ou de forma inclinada, foi necessário fazer uma análise das diferentes formas de orientação da pinça e do objeto.

Foi possível dividir as situações de abordagem da pinça em quatro maneiras distintas: por cima, pela frente, por trás, inclinada pela frente e inclinada por trás. Iniciar a prensão dos objetos por baixo é realisticamente impossível, visto que os objetos não estarão flutuando e também não estarão presos em algum lugar. A prensão por trás possui um tratamento análogo à prensão feita pela frente, pois em ambos os casos o objeto é pego pela pinça frontalmente.

A prensão do objeto feita de forma perpendicular ao plano do robô nunca ocorrerá, pois o robô Scorbob é um robô articulado com 5 DOF, não existindo o tipo de rotação *yaw* em sua estrutura. Ou seja, o robô é estruturalmente incapaz de efetuar a prensão dos objetos pelas laterais. A Tabela 6 mostra as situações de prensão que são tratadas através dos raciocínios geométricos, sendo estes separados para cada situação em especial.

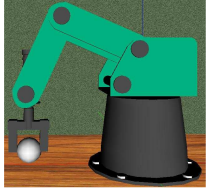
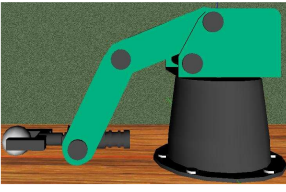
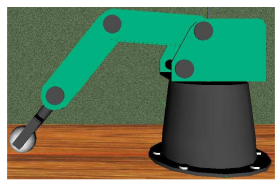
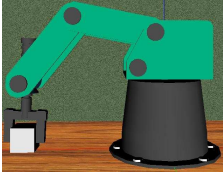
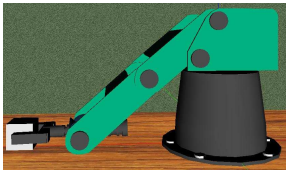
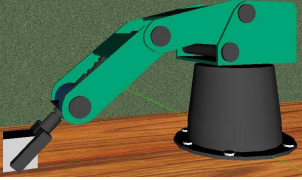
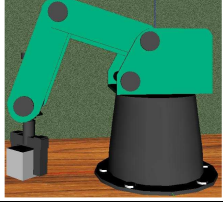
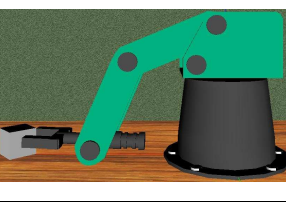
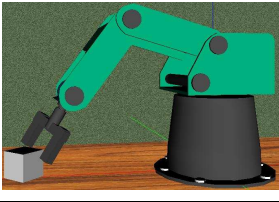
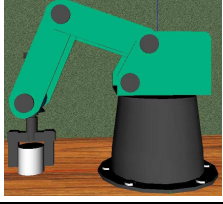
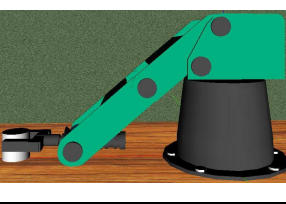
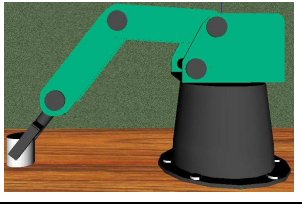
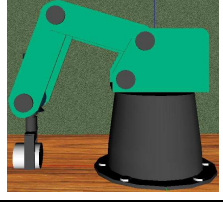
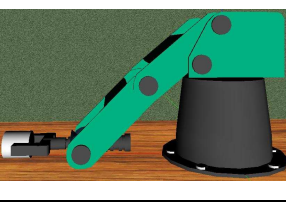
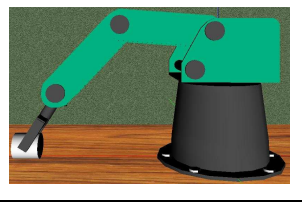
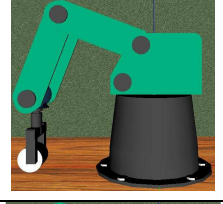
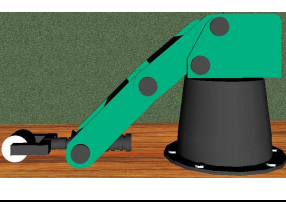
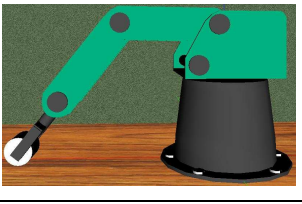
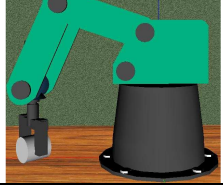
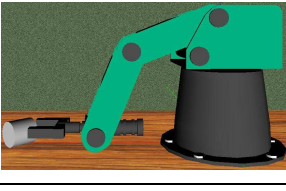
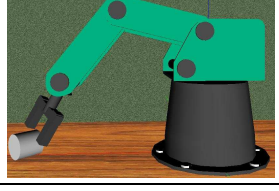
Geometria	Disposição		Pinça por cima	Pinça pela frente	Pinça inclinada
Esfera	Qualquer				
Paralelepípedo	Alinhado				
	Diagonal				
Cilindro	Ereto				
	Deitado	Alinhado			
		Transversal			
		Diagonal			

Tabela 6 – As diferentes situações de prensão

4.5. Raciocínios Geométricos

Enumeradas as diferentes situações possíveis que podem acontecer no momento da apreensão, parte-se para a análise destes casos. Ao visualizar a Tabela 6, percebe-se que existem casos que podem ser resolvidos por um mesmo conjunto de regras. Um mesmo algoritmo pode solucionar a apreensão por cima de um cilindro deitado disposto longitudinalmente e a apreensão pela frente de um cilindro em pé, por exemplo. Visualizando cada caso apresentado na Tabela 6 e agrupando situações semelhantes, chega-se na Tabela 7.

Percebe-se que estas situações semelhantes possuem um padrão nas angulações entre os vetores de orientação da pinça e do objeto. Isto faz com que diferentes situações apresentadas na Tabela 6 possam ser identificadas por um mesmo algoritmo. As situações semelhantes foram enumeradas na Tabela 7, onde cada caso semelhante possui um índice que identifica ele.

Geometria	Disposição			Pinça por cima	Pinça pela frente	Pinça inclinada
Esfera	Qualquer			1	1	1
Paralelepípedo	Alinhado			2	2	2
	Diagonal			2	Exceção	Exceção
Cilindro	Ereto			3	4	5
	Deitado	Alinhado	Longitudinal	4	3	5
			Transversal	6	6	6
		Diagonal		4	Exceção	Exceção

Tabela 7 – Similaridade entre as situações

Ao comparar a Tabela 6 com a Tabela 7, as situações (células da tabela) que possuem índices iguais foram identificadas como situações semelhantes e, consequentemente, puderam ser classificadas como possuindo uma mesma abordagem entre garra e objeto. Os casos onde não existe um índice na tabela (Exceções) são automaticamente excluídos, pois apresentam situações geométricas aparentemente impossíveis para a apreensão. Existem situações de Exceção que até podem levar a apreensão, porém estas não apresentam estabilidade e, consequentemente, são eliminadas pelos Raciocínios Geométricos.

Visualizando a Tabela 7, foram definidas 6 abordagens capazes de traduzir as 17 situações apresentadas na Tabela 6. Quatro casos identificados na Tabela 6 foram

considerados impossíveis para apreensão e são automaticamente eliminados ao serem detectados pelos raciocínios geométricos. Partindo dos padrões identificados através das diferentes situações da cena, foram criados os raciocínios geométricos.

No instante do reconhecimento do padrão ao qual a situação passada para o algoritmo pertence, foram consideradas possíveis folgas (pequenas diferenças geométricas que não comprometeriam a apreensão numa situação real). Devido ao fato de a manipulação do robô estar sendo realizada por um usuário, é preciso considerar que a garra pode não estar perfeitamente alinhada ao objeto. Pequenas diferenças entre angulações precisam ser relevadas, pois são raros os casos onde as angulações que identificam um padrão são perfeitamente atendidas. O mesmo acontece para pequenas variações de posição. São raros os casos onde a apreensão ocorre exatamente no centro de gravidade do objeto, que é o caso ideal.

Desta forma, a solução, a grosso modo, consiste em uma série de testes que avaliam folgas angulares e folgas de posição.

4.5.1. Folga

Levando em consideração a possibilidade de manipulação do robô através de uma interface gráfica (área 1 da Figura 34) feita através do usuário, percebeu-se que muitas vezes existirão situações imperfeitas no momento que seja solicitada a apreensão. É necessário que sejam incluídas folgas para que pequenas variações não possam ser invalidadas pelos Raciocínios Geométricos. Com isto, passa a ser desnecessário que seja promovido um posicionamento perfeito da garra, que muitas vezes só é alcançado modificando diretamente os valores de juntas do robô. Além de este tipo de manipulação do robô ser pouco intuitiva para o usuário (modificação direta das variáveis de junta), é necessário que sejam calculadas de antemão estas variáveis.

Outros fatores também contribuíram para a inclusão de folgas nas análises. Imprecisões nos cálculos computacionais são comuns (propagação de erros matemáticos), fazendo com que uma sequência de cálculos resulte em valores com imperfeições. Com a sequência de cálculos exigida pela Cinemática Direta, é possível que ocorra imprecisão ao atualizar os valores do *PFP*. Este problema é minimizado com o uso da folga, pois pequenas imperfeições de posicionamento do *PFP* são relevadas. Ainda assim, problemas de folgas mecânicas do próprio robô levam a necessidade de flexibilizar os Raciocínios Geométricos.

Após a análise destes fatores, percebeu-se a necessidade de incluir lógicas de folga nos durante a validação por Raciocínios Geométricos.

A estratégia geral da solução consiste em determinar três planos de referência internos a geometria do objeto e utilizá-los como limite para análises com o *PFP*. Os planos possuem seus vetores normais ortogonais e são capazes de gerar uma primitiva cúbica interna ao objeto. Caso o *PFP* esteja localizado dentro deste volume e não ocorra colisão no ambiente, o caso é dado como válido. Este volume interno ao objeto é derivado por dados vindos da garra e do objeto. Os eixos do cubo de folga são paralelos aos vetores de orientação da garra (características da garra), enquanto que a sua localização é abaixo do centro de gravidade do objeto (características do objeto). Outro fator que diz respeito à estratégia geral, é a prevenção de um alinhamento entre a garra e o objeto, feito através de uma folga angular. Percebe-se que a estratégia geral é aplicável pelo fato de existir tratamento de detecção de colisões, garantindo que o objeto se encaixa por entre os dedos da garra.

Durante a manipulação do robô virtual feita pelo usuário estudante da robótica, existe a possibilidade de ocorrer pequenas variações entre a orientação da garra e do objeto. Seria muito restritivo exigir que a garra segurasse um objeto exatamente por cima sem mínimas variações angulares entre seu vetor de aproximação ($\overrightarrow{Apq}$) e o eixo Y de coordenadas, por exemplo. Igualmente, seria muito restritivo exigir que a preensão ocorresse sempre exatamente pelo centro de gravidade do objeto. Como existem casos onde estas variações são irrelevantes, é necessário considerar uma folga a fim de tornar o sistema usável.

É necessário inserir este tratamento para que situações que fisicamente promovem a preensão possam ser avaliadas de maneira geométrica e com pequenas variações. Caso existam imperfeições de posicionamento ou orientação que sejam fisicamente toleráveis para a preensão, esta é considerada uma resposta válida e não é descartada pelos Raciocínios Geométricos. Esta é uma forma de os Raciocínios Geométricos estarem representados através da experiência e do tratamento de questões que são, na verdade, aspectos físicos. No caso têm-se restrições quanto ao atrito em situação estática e, torques e vibrações, em situações dinâmicas.

Levando em consideração as pequenas variações angulares e espaciais existentes no momento da preensão, foram criados cálculos de confirmação que se baseiam em limites angulares e espaciais para determinar se uma preensão é realizável ou não. Estes dois

diferentes tratamentos são denominados de Folga de Abordagem e de Folga Geométrica. A Figura 45 ilustra a divisão do tratamento de folga.

$$\text{Folga} \left\{ \begin{array}{l} 1^{\text{a}} \text{ Abordagem} \rightarrow f(T_Obj, O_Obj, O_Gar) \\ 2^{\text{a}} \text{ Geométrica} \rightarrow f(G_Obj, G_Gar) \end{array} \right.$$

Figura 45 – Divisão das análises de Folga

Como é mostrado na Figura 45, a Folga de Abordagem é o bloco de análises que são realizadas anteriores ao bloco que compõe a Folga Geométrica. Foi observado que a Folga de Abordagem (orientação) depende do tipo de objeto primitivo (T_Obj) e das orientações do objeto (O_Obj) em relação à da garra (O_Gar). Já a Folga Geométrica (posição) depende da geometria do objeto (G_Obj) e da garra (G_Gar). Difere-se tipo do objeto como sendo a distinção de qual primitiva que o objeto é, e geometria do objeto como sendo o conjunto de dados que definem as dimensões geométricas do objeto.

4.5.1.1. Folga de Abordagem

Para esta análise, cada objeto possui um tratamento específico a ele. Este tratamento consiste em identificar em qual padrão mostrado na Tabela 7 a situação submetida pertence.

Com base nas angulações entre os vetores de orientação da garra e do objeto é identificado em qual padrão disposto na Tabela 7 a situação pertence. Caso uma das angulações entre os vetores exceder certo limite, caracterizando nenhum dos padrões apresentados, a situação é considerada como instável para preensão.

O valor limite de angulação, mostrado na Figura 46, é denominado como Folga de Abordagem (Δ_{ab}) e é utilizado para avaliar as variações angulares existentes em cada eixo de coordenadas da garra. Como está presente essa tolerância de angulação entre vetores da garra e do objeto, é formado um volume cônico sobre cada eixo do sistema de coordenadas da garra que garante se a variação é tolerável. Caso a angulação entre os dois vetores, um pertencente à garra e outro pertencente ao objeto, estarem fora deste volume cônico, a situação é reprovada por exceder o limite angular (Folga de Abordagem).

O efeito do Δ_{ab} é permitir imprecisões de orientação, como se o objeto pudesse ter sofrido um tremor, como mostra na Figura 46 apenas para um eixo. O valor do Δ_{ab} é passado para o algoritmo através da captura dos dados existentes em um componente da interface do

simulador destinado unicamente a esta função (vide Figura 35). Note que Δ_{ab} é aplicado tanto numa direção quanto na outra. Como a Figura 46 mostra um objeto 2-D, a angulação formada pela Folga de Abordagem consiste em um “V”. Porém como esta folga é estendida a um ambiente 3-D, ela deriva os cones projetados nos eixos do sistema de coordenadas da garra, explicado anteriormente.

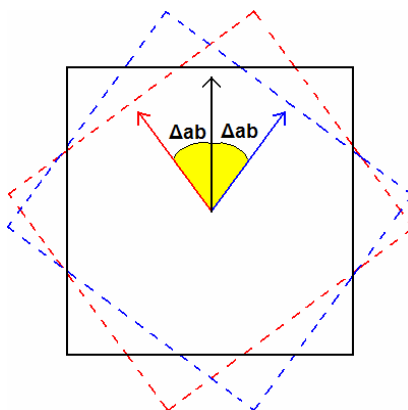


Figura 46 – Folga de Abordagem num quadrado 2-D

Inicialmente, nesta etapa, é identificado qual o tipo de objeto para posteriormente serem feitas as análises de folgas angulares específicas ao objeto identificado. Por exemplo, análises angulares ao paralelepípedo são diferentes de análises angulares de um cilindro, por envolverem quantidades diferentes de ângulos a comparar. Esta distinção pode ser visualizada na própria Tabela 7, onde nenhum dos padrões pertence a objetos com geometrias distintas (cada padrão pode identificar somente um tipo de primitiva). Desta forma, os Raciocínios Geométricos, estes capazes de identificar os padrões da Tabela 7, necessitam ser classificados e explicados de acordo com tipo de geometria ao qual eles foram projetados.

(1) Raciocínios Geométricos de Abordagem para a Esfera:

Devido ao fato de o volume do objeto ser esférico, testes angulares são desnecessários para este caso. Como a Folga de Abordagem é especificamente angular, inexistente teste de Folga de Abordagem quando o tipo de geometria do objeto a ser pego for esfera. O padrão correspondente ao caso de preensão de esferas é o padrão 1, mostrado na Tabela 7.

(2) Raciocínios Geométricos de Abordagem para o Paralelepípedo:

Para o caso de preensão de paralelepípedos uma única regra precisa ser satisfeita. Esta regra consiste na verificação de paralelismo entre o vetor de eixo lateral da garra ($\overrightarrow{Elg}$) e qualquer

vetor de orientação do objeto. Caso qualquer um dos vetores do objeto esteja paralelo ao vetor $\overrightarrow{Elg}$, pode-se garantir, para o caso de preensão com a garra do tipo pinça de dois dedos, que o paralelepípedo está sendo segurado por duas de suas faces paralelas. Isto ocorre devido o fato de os três vetores de orientação do paralelepípedo serem paralelos aos vetores normais de suas faces.

Como é possível visualizar na Figura 47, os vetores $\overrightarrow{Elg}$ e $\overrightarrow{Elo}$ são paralelos para os três casos apresentados caracterizando, assim, três situações possíveis para preensão. Apesar de nos três casos apresentados na figura o vetor do objeto paralelo ao vetor $\overrightarrow{Elg}$ ser o vetor $\overrightarrow{Elo}$, este teste pode ser aplicado a qualquer um dos três vetores de orientação do objeto. Percebe-se ainda que esta regra também atende a preensão de paralelepípedos com a abordagem da pinça de forma inclinada (caso c da Figura 47).

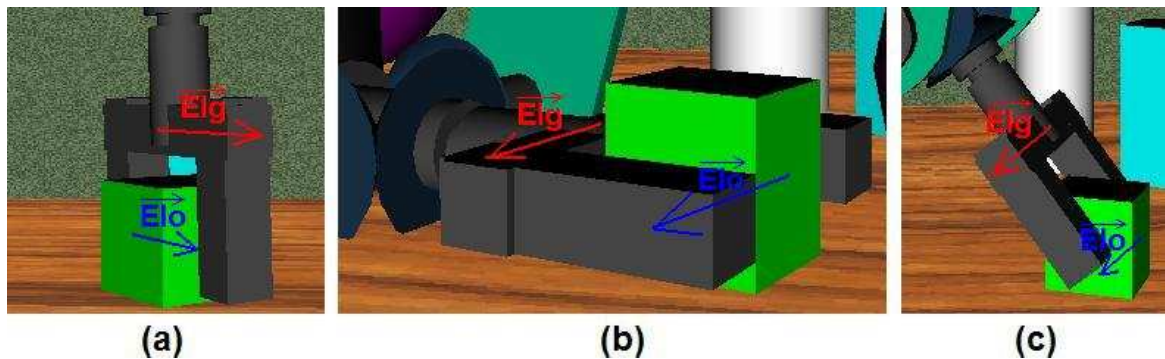


Figura 47 – Raciocínio Geométrico para Paralelepípedos

Como visto na Figura 46, este teste de paralelismo pode garantir que o paralelepípedo está alinhado com a garra e leva a uma preensão aceitável. Caso a angulação entre o vetor $\overrightarrow{Elg}$ e os três vetores de orientação do objeto for superior ao Δab , a situação é classificada por este raciocínio geométrico como sendo instável para a preensão. O padrão correspondente ao caso de preensão de paralelepípedos é o padrão 2, mostrado na Tabela 7.

(3) Raciocínios Geométricos de Abordagem para o Cilindro:

O caso de preensão de cilindros é mais complexo conceitualmente em comparação aos outros casos explicados. Inicialmente é verificado se existe paralelismo entre o vetor $\overrightarrow{Apq}$ e o vetor $\overrightarrow{Upo}$. Caso esses vetores serem paralelos, é identificado o padrão 3 da Tabela 7. Este padrão define a preensão sendo realizada pela parte superior do cilindro, mostrado na Figura 48.

Como é possível visualizar na Figura 48, para qualquer valor de rolamento da pinça (*roll*) este caso é identificado no padrão 3.

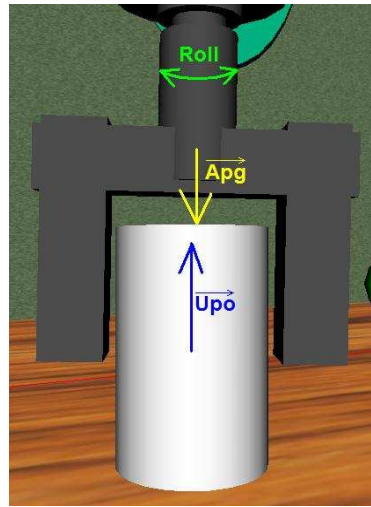


Figura 48 – Padrão para a pega de Cilindros por cima

Ao analisar a Figura 48, percebe-se que mesmo a angulação entre os vetores $\overrightarrow{Apg}$ e $\overrightarrow{Upo}$ excedendo o valor de Folga de Abordagem, ainda podem ocorrer preensões de diferentes abordagens. Desta forma, caso a angulação não satisfaça este padrão, o raciocínio geométrico não realiza nenhuma conclusão sobre a situação. São necessários outros testes para concluir tal situação.

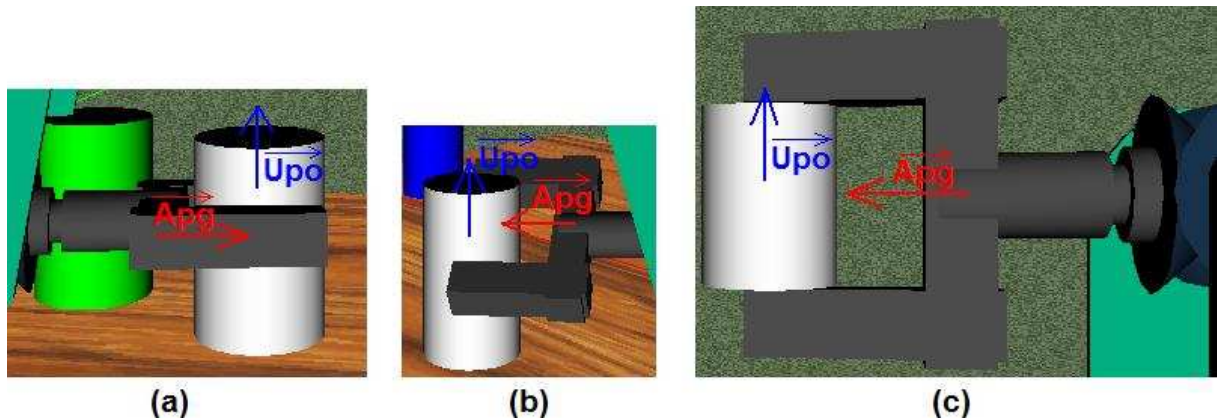


Figura 49 – Padrão para a pega de cilindros pela lateral

Caso o padrão 3 da Tabela 7 não seja identificado, o raciocínio geométrico para o cilindro é passado adiante com outro teste de angulação. O próximo teste consiste em verificar se os vetores $\overrightarrow{Apg}$ e $\overrightarrow{Upo}$ são perpendiculares entre si, como os casos apresentados na Figura 49. Caso sejam, pode-se concluir que a situação pode corresponder ao padrão 4, ao

padrão 6 ou a um caso de preensão instável (Exceção). Como é possível verificar na Figura 49, mesmo após a verificação de paralelismo entre os vetores $\overrightarrow{Apq}$ e $\overrightarrow{Upo}$, feita anteriormente, e a verificação angular de perpendicularismo ainda não pode identificar um padrão da tabela.

Como é possível visualizar na Figura 49, o caso *a* corresponde ao padrão 4, o caso *b* corresponde a uma exceção que, devido à abordagem da garra, não leva a preensão e o caso *c* corresponde ao padrão 6 da Tabela 7. Para que seja decidido em qual padrão a situação está inserida ou se ela é uma exceção, é preciso realizar um terceiro teste. Este teste consiste em avaliar a angulação entre o vetor $\overrightarrow{Upq}$ e o vetor $\overrightarrow{Upo}$. Caso a angulação entre estes vetores seja igual a 90° , a situação está inserida no padrão 6 e pertence ao caso *c* da Figura 49. Caso a angulação corresponder a um paralelismo, a situação está inserida no padrão 4 e pertence ao caso *a* da figura. Não sendo paralelo e nem possuindo 90° de angulação entre os vetores $\overrightarrow{Upq}$ e $\overrightarrow{Upo}$, o raciocínio geométrico identifica como uma exceção da Tabela 7, caso correspondente ao *b* da Figura 49.

Caso a angulação entre os vetores $\overrightarrow{Apq}$ e $\overrightarrow{Upo}$ não corresponda a um paralelismo e nem perpendicularismo (90° ou 270°), pode corresponder ao padrão 5 ou a um caso de preensão instável. Para identificar se a situação pertence ao padrão 5, basta verificar se o ângulo entre o vetor $\overrightarrow{Upo}$ e o vetor $\overrightarrow{Elq}$ são perpendiculares entre si. Caso seja, a situação é identificada com o padrão 5 da Tabela 7, caso contrário ela é identificada como uma exceção e precisa ser eliminada.

Todos os testes angulares existentes nos Raciocínios Geométricos de abordagem para o cilindro levam em consideração a Folga de Abordagem (Δab), possuindo, assim, tolerância de angulação no momento do teste.

4.5.1.2. Folga Geométrica

Seguido das análises angulares e depois de identificado qual padrão da Tabela 7 a situação pertence, são feitos os testes de folga de posicionamento. As análises por raciocínios geométricos são finalizadas nesta parte do tratamento de folga. Diferente da etapa anterior, esta possui testes mais genéricos e não leva em consideração qual o tipo/classe do objeto (exemplo: cilindro, paralelepípedo e esfera), ou seja, os testes realizados são os mesmos, independente do padrão identificado anteriormente e da geometria do objeto e da garra.

Como foi mostrada na Figura 45, esta etapa se utiliza das dimensões do objeto (geometria) que estão representadas na sua respectiva geometria envolvente (OBB). Estas

informações são importantes para que se saiba onde a garra e o objeto estão posicionados e também para poder derivar, através das dimensões do objeto, a folga geométrica propriamente dita.

A estratégia geral adotada consiste em verificar se o ponto *PFP* localiza-se interno a um pequeno volume de folga posicionado a partir do centro de gravidade (CG) do objeto a ser pego. Caso o ponto esteja interno a este volume, o teste é aprovado e então a preensão é dada como válida. O volume de folga resulta em um cubo e o valor de suas arestas é denominado Folga Geométrica (Δ_{ge}). O Δ_{ge} corresponde à metade do menor valor entre as três dimensões do volume envolvente aproximado (no caso, é usada uma OBB) do objeto (valores peculiares ao objeto) e a altura dos dedos da garra (valor peculiar da garra), como mostrado na Fórmula (4.5). Foi decidido que o valor de Folga Geométrica fosse automaticamente calculado fazendo uso dos dados citados, e a Fórmula (4.5) garante que o volume de folga não entre em contato com a parte externa do objeto.

$$\Delta_{ge} = \frac{\text{menor}(\text{Obj.getObb()}[0], \text{Obj.getObb()}[1], \text{Obj.getObb()}[2], \text{Garra.A})}{2} * PI\Delta_{ge} \quad (4.5)$$

Sendo:

- Δ_{ge} : O valor de Folga Geométrica;
- *menor*: Uma função que retorna o menor dos valores passados por parâmetro;
- *A*: O valor da altura do objeto;
- *PI Δ_{ge}* : Parâmetro para Folga Geométrica.

O valor *PI Δ_{ge}* é um valor de porcentagem e possui a função de definir o tamanho do volume de folga desejado pelo usuário. Este valor define a precisão no momento de realizar a preensão (preensão mais próxima ao Centro de Gravidade) e é passado através da interface. Este volume é projetado abaixo do centro de gravidade do objeto e sua orientação acompanha a orientação da garra (e portanto, varia nas direções $\overrightarrow{Apq}$, $\overrightarrow{Upq}$ e $\overrightarrow{Elq}$, vide Figura 38), conforme pode ser observado na Figura 50. Basicamente, se *PFP* estiver dentro do cubo inscrito ao objeto então este teste resulta positivo para condição de preensão.

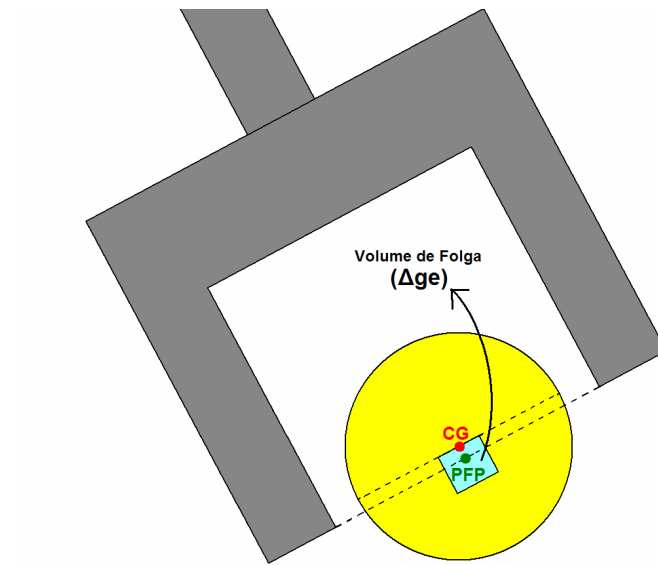


Figura 50 – Representação da Folga Geométrica Interna ao Objeto

5. RESULTADOS

5.1. Implementação

5.1.1. Mudanças no Projeto

Apesar de ter sido definido no projeto a funcionalidade de avaliação de preensão, alguns imprevistos ocorreram ao implementar esta função no robô Scrobot virtual. Muitas informações explicitadas no projeto foram desnecessárias para o desenvolvimento e execução do algoritmo final. Alguns problemas foram visualizados e foram superados com a inclusão de novas informações não previstas na fase de projeto.

Ao iniciar a implementação deste trabalho, percebeu-se uma limitação ao incluir a avaliação da preensão. Os componentes responsáveis pela avaliação da preensão estavam limitados a aparecer somente na *applet* capacitada pela manipulação de comandos do robô (área 2 da Figura 34). Esta limitação ocorre devido ao fato de as três áreas executarem ao mesmo tempo em *threads* distintas e sem possibilidade de troca de informações. Como as três *threads* são criadas através do documento HTML ao iniciar a aplicação, elas não possuem referência uma da outra. A única forma aparente de comunicação entre elas é através do EAI (SANTOS, 2007a), sendo esta uma interface que permite a comunicação entre um ambiente VRML e uma *applet*.

Devido ao fato de existir mais espaço disponível na *applet* responsável em manipular os comandos do robô, foram inclusos nela os componentes de geração de objetos e de avaliação de preensão. Como a avaliação da preensão necessita de informações mais específicas aos objetos, percebe-se a necessidade de os componentes pertencentes a estas duas funções (geração de objetos e avaliação de preensão) estarem localizados na mesma *applet*, e de preferência próximos. Assim, a detecção de colisão e o tratamento de preensão estão intimamente relacionados e necessitam estar executando na mesma *thread*, visto que ambos fazem uso de informações mais específicas aos objetos.

Em sua versão anterior, o simulador do Robô Scrobot ER-4PC já possuía a funcionalidade de pseudo-preensão implementada. A preensão estava presente unicamente com o intuito de teste de colisão entre o objeto pego e outros elementos dispostos no ambiente. Nenhuma avaliação era realizada ao solicitar a preensão e o objeto mais próximo da

garra era o selecionado para preensão. O objeto era incluso na garra através do botão FG existente da interface (vide Figura 35). Este botão, porém possuía outra função e, conseqüentemente, não fazia sentido incluir a funcionalidade da avaliação da preensão neste componente.

O botão FG possui a função de incluir na lista de comandos do robô a operação de fechamento de garra, independente se o *status* da garra seja aberto ou fechado. Desta forma, mesmo que o robô já esteja com a garra fechada, é possível incluir este comando para ele executar novamente. Isto faz com que a funcionalidade de avaliação de preensão não possa ser simplesmente colocada neste componente, visto que o robô poderá já estar segurando outro objeto. Após análises feitas na interface, foi realizada uma modificação com o intuito de passar esta funcionalidade para um novo elemento. Este elemento é um botão próprio responsável em iniciar as análises de preensão e está localizado na parte inferior da *applet* de gerenciamento de comandos do robô.

O botão responsável por iniciar a avaliação de preensão (vide Figura 35), além de ser um novo elemento a ser adicionado na interface, passou a ter a possibilidade de sofrer modificações em seu rótulo (*label*), evitando-se, assim, a inclusão de um novo componente. A necessidade de incluir esta modificação neste componente se mostrou necessária no momento em que foi incluída no simulador a funcionalidade de adição do objeto na estrutura hierárquica do braço do robô a partir da garra. Estando o objeto incluso na hierarquia do braço, viu-se a necessidade de retirar o objeto da estrutura. Estas duas funções foram adicionadas no mesmo componente, evitando-se a inclusão de outro. O comportamento do componente é variado de acordo com a máquina de estados representada na Figura 51.

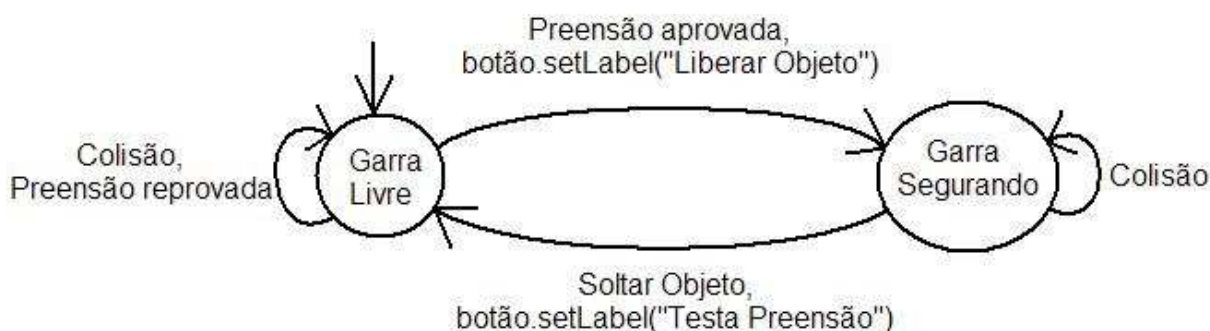


Figura 51 – Máquina de Estados do botão de teste de preensão

Como é possível visualizar na Figura 51, o comportamento do botão está ligado às transições dos dois diferentes estados que a garra do robô pode estar. Caso o botão seja

clicado, a garra esteja livre, não exista nenhuma colisão no ambiente e a preensão for aprovada, o *label* do botão avisa o usuário de que o robô agora pode liberar o objeto e este passa a fazer parte da hierarquia do braço do robô. Caso o robô esteja segurando algo, não ocorre colisão no ambiente e o botão seja clicado, o *label* do botão é modificado avisando o usuário de que é possível pegar um novo objeto e o objeto que a garra estava segurando é removido da estrutura do robô.

Ainda durante a implementação foram realizadas mudanças na interface no que se diz respeito ao *feedback* proporcionado ao usuário. As cores das fontes dos elementos capacitados por apresentar os valores do *PFP* foram modificadas de acordo com qual eixo de coordenadas pertence o valor apresentado em cada componente (vide área 2 da Figura 34), onde são mostrados os valores respectivos ao eixo X, ao eixo Y e ao eixo Z. Percebeu-se um aumento de usabilidade ao relacionar a cor da fonte com a cor do eixo ao qual o valor pertence. Ainda pensando na usabilidade foi realizada outra mudança no componente que possui a função de mostrar as informações do objeto selecionado (vide Figura 35). Anteriormente este elemento não especificava a qual objeto pertence as informações que são apresentadas por ele. Na versão atual do simulador este componente foi atualizado e já informa a qual objeto do ambiente pertence as informações apresentadas.

Uma mudança brusca no projeto aconteceu com a eliminação do teste Esferóide de Trabalho da etapa de exclusão dos Raciocínios Geométricos. O principal motivo de retirar este teste da etapa de exclusão foi a busca por uma execução mais rápida do algoritmo. A solução da Cinemática Inversa disponível hoje no Robô Virtual apenas considera situações onde a garra faz a preensão com abordagem exatamente por cima ($-\vec{Y}$), ou seja, não é suficientemente genérica para ser usada no teste de Esferóide de Trabalho. Para compensar a ausência deste teste foram criados dois novos testes de exclusão, sendo estes os testes de verificação se o objeto está interno ao *Bounding Box* de trabalho do robô e o teste de verificação se o objeto não se encontra na região que excede os limites de rotação da base do robô.

Mesmo estando ausente o teste de Cinemática Inversa, foram utilizadas no simulador lógicas do algoritmo de Cinemática Inversa implementados por Zwirter (2004) para criar o último dos dois testes citados. É verificado se o ângulo de rotação da base necessário para pegar o objeto é maior do que o limite de junta da base do robô. Este ângulo é calculado pela

fórmula que determina a rotação da base no algoritmo de Cinemática Inversa, mostrado na Fórmula 5.1.

$$\text{angulo} = \arctan2(Z, -X) \quad (5.1)$$

Sendo:

- *angulo*: O valor do ângulo de rotação da base necessário para pegar o objeto.
- *Z*: O valor da coordenada Z do centro do objeto.
- *X*: O valor da coordenada X do centro do objeto.
- *arctan2*: A função arco-tangente para os quatro quadrantes do plano XZ

Para aplicar esta função é necessário que XZ seja o plano paralelo à mesa onde o robô está preso. Devido ao fato de o robô estar voltado para o lado esquerdo do ambiente, todo o cálculo da Cinemática Inversa deve ser feito levando em consideração a transformação do sistema de coordenadas global para o sistema de coordenadas do robô. A Figura 52 mostra a mudança de sistema de coordenadas necessária para os cálculos algébricos. A transformação é feita passando de um sistema de coordenadas utilizado por Schilling (1990) em seus cálculos (Figura 52 (a)) para o sistema de coordenadas utilizado no robô virtual (Figura 52 (b)). A Fórmula (5.1) calcula o ângulo diretamente para o sistema de coordenadas do robô virtual.



Figura 52 – Mudança de sistema de coordenadas

Algumas informações especificadas no projeto não foram úteis para a implementação da funcionalidade de avaliação de apreensão. Por exemplo, dados da garra (item 4.2.1 deste trabalho) foram desnecessários para concluir o algoritmo que corresponde aos Raciocínios

Geométricos. Os valores de altura total (H), largura dos dedos (L) profundidade dos dedos (P) e abertura máxima entre os dedos (O) foram desnecessários para o algoritmo. Percebeu-se que somente um dos dados dimensionais da garra foi utilizado, sendo este o valor da altura dos dedos (A). Apenas na etapa de teste da Folga Geométrica é utilizado este valor para calcular o Δg . Mesmo caindo em desuso, estes dados foram mantidos na implementação para possibilitar futuras atualizações do simulador. Alguns dados dimensionais foram desnecessários, ao passo que foram importantes os dados de posicionamento da garra (PFP).

Durante o projeto da funcionalidade, foi previsto que seria necessário um teste unicamente para verificar se a garra está aberta no momento da preensão. Percebeu-se que este teste poderia ser eliminado pelo fato de a garra ser inacessível ao objeto estando ela fechada. Caso se insista na tentativa de acessar o objeto com a garra fechada, os cálculos de detecção de colisão detectariam a colisão assim que a aproximação ocorresse, identificando, desta forma, o problema de acessibilidade. Portanto, percebeu-se que este teste poderia ser retirado da implementação do simulador, visto que estariam executando algoritmos de colisão juntamente com a avaliação da preensão.

Outra informação mencionada no projeto que não foi utilizada na implementação foi o uso do PRP . Como a Cinemática Inversa não foi implementada nos testes de exclusão, o PRP passou a ser obsoleto, visto que este ponto foi criado unicamente para os cálculos de Cinemática Inversa. Desta forma, este ponto foi retirado da implementação.

Como foi previsto, eram poucos os dados necessários para o simulador, pertencentes ao ambiente. Dados do robô foram utilizados para derivação do PFP e do *Bounding Box* de trabalho do robô. Os dados pertencentes à mesa foram utilizados somente na geração de objetos, estando estes ausentes do tratamento de preensão.

Resumidamente, houve considerável quantidade de mudanças na interface para a sua adequação à funcionalidade implementada neste trabalho. Algumas informações especificadas anteriormente não precisaram ser utilizadas na implementação, porém algumas foram mantidas para futuros trabalhos que venham a agregar no Scorbots Virtual.

5.1.2. Classes Incluídas no Simulador

Foi necessário incluir novas classes Java no simulador para adicionar a funcionalidade de avaliação de preensão. Como grande parte do projeto do simulador já foi especificado por

Santos (2007a), serão mostradas apenas as explicações referentes aos métodos e constantes das classes incluídas no simulador decorrentes deste trabalho.

A classe `PrincipalComandos` já estava presente nas versões anteriores do simulador, sendo ela responsável pelo gerenciamento da *applet* de manipulação de comandos para o robô. Ao incluir o tratamento de preensão de objetos, esta classe passou a instanciar uma nova classe denominada `Garra`. Os Raciocínios Geométricos se encontram na classe `TrataPreensao`. Esta classe possui somente um método público e estático que possui a função de avaliar a preensão através dos Raciocínios Geométricos implementados nele. Como a classe `TrataPreensao` possui apenas um método estático, é desnecessário que a classe `PrincipalComandos` possua uma instância de um objeto da classe `TrataPreensao`. Isto faz com que a execução dos Raciocínios Geométricos seja encapsulada e possa ser utilizada em qualquer lugar da *applet* e em qualquer momento, exigindo apenas uma instância da classe `Garra` e a lista de objetos.

A classe `Garra` tem como principal função guardar informações relativas à garra do robô `Scorbot`, que são atualizadas cada vez que se deseja realizar o teste de preensão. Ou seja, quando o usuário solicitar a avaliação de preensão, um objeto desta classe é atualizado. Os dados que esta classe armazena são utilizados pela classe `TrataPreensao` para a execução dos Raciocínios Geométricos. É mostrado na Figura 53 o detalhamento da classe `Garra`.

Garra
+ H : double = 0.25875 + A : double = 0.099 + L : double = 0.036 + P : double = 0.036 + O : double = 0.1125 - pfp : double[] - upg : double[] - apg : double[] - elg : double[]
+ setPFP(p : double[]) : void + setOrientacao(o : Obb) : void + setOrientacao(up : double[], ap : double[], el : double[]) : void + getPFP() : double[] + getUpg() : double[] + getApg() : double[] + getElg() : double[]

Figura 53 – Detalhamento da classe `Garra`

A Figura 53 mostra as constantes, variáveis e métodos referentes à classe `Garra`. As cinco constantes declaradas (*H*, *A*, *L*, *P* e *O*) são públicas e fazem referência aos dados

dimensionais da Garra apresentados na Tabela 3. Apenas o valor de A (altura dos dedos da garra) foi utilizado pelos Raciocínios Geométricos, porém foram mantidos os dados para futuras atualizações do simulador.

As variáveis da classe Garra correspondem ao PFP e os vetores de orientação $\overrightarrow{Upg}$, $\overrightarrow{Apq}$ e $\overrightarrow{Elg}$, respectivamente. Como é possível perceber, existem somente métodos *setters* e *getters*, caracterizando, assim, uma classe responsável para armazenamento e estruturação. Os dados referentes ao PFP são adquiridos, através do EAI , de funções Javascript existentes no arquivo VRML do robô capacitadas em calcular a Cinemática Direta. Os dados pertencentes à orientação da garra são adquiridos através de sua OBB, implementada por Santos (2007a). Existem duas formas diferentes de informar a orientação da garra: passado a OBB da garra por parâmetro através do método `setOrientacao(o : Obb)`, e informando diretamente os vetores de orientação através do método `setOrientacao(up : double[], ap : double[], el : double[])`.

Tendo a garra representada através de um objeto Garra, é possível realizar a execução dos Raciocínios Geométricos através de um método específico da classe `TrataPreensao`. Além dos Raciocínios Geométricos, esta classe possui constantes inteiras correspondentes às situações que levam a preensão e aos erros específicos que os Raciocínios Geométricos são capazes de identificar. Caso a situação submetida aos Raciocínios Geométricos seja aprovada, o método que inicia a execução dos Raciocínios Geométricos retorna o valor da constante de classe `APROVADO`, caso contrário ele retorna uma outra constante pertencente ao erro detectado pelo algoritmo. A Figura 54 mostra o detalhamento da classe `TrataPreensao`.

Como é possível visualizar na Figura 54, todas as constantes que identificam erros correspondem a um valor negativo enquanto que a aprovação corresponde a um valor positivo. Foram atrelados valores negativos a erros e positivos a aprovação para que a programação seja mais intuitiva, fazendo com que o programador possa identificar as situações possíveis e impossíveis (instáveis) apenas verificando se a resposta for um valor positivo ou negativo. Os casos de exceção, listados na Tabela 7, até podem corresponder a uma preensão, porém instável. Desta forma, o algoritmo de Raciocínios Geométricos reprova os casos onde é impossível a preensão (Exclusão) e os casos onde a preensão é instável (Folga de Abordagem e Folga Geométrica).

TrataPreensao	
+ APROVADO : int = 1	
+ ERRO_BOUNDING_BOX_TRABALHO : int = -1	
+ ERRO_FATIA_TRASEIRA : int = -2	
+ ERRO_TAMANHO_OBJETO_GARRA : int = -3	
+ ERRO_PROXIMIDADE : int = -4	
+ ERRO_ABORDAGEM : int = -5	
+ ERRO_DELTA_APG : int = -6	
+ ERRO_DELTA_UPG : int = -7	
+ ERRO_DELTA_ELG : int = -8	
+ preensao(lista : ListaQuadrante, id : int, g : Garra, deltaAb : double) : int	
- exclusao(lista : ListaQuadrante, id : int, obj : char, g : Garra) : int	
- boundingBoxTrabalho(lista : ListaQuadrante, id : int) : boolean	
- fatiaTraseira(lista : ListaQuadrante, id : int) : boolean	
- tamanhoObjetoGarra(lista : ListaQuadrante, id : int, g : Garra) : boolean	
- proximidade(lista : ListaQuadrante, id : int, obj : char, g : Garra) : boolean	
- confirmacao(lista : ListaQuadrante, id : int, obj : char, g : Garra, deltaAb : double) : int	
- folgaAbordagem(lista : ListaQuadrante, id : int, obj : char, g : Garra, deltaAb : double) : int	
- folgaAbordagemCilindro(c : Cilindro, g : Garra, deltaAb : double) : int	
- folgaAbordagemParalelepipedo(p : Paralelepipedo, g : Garra, deltaAb : double) : int	
- folgaGeometrica(lista : ListaQuadrante, id : int, obj : char, g : Garra) : int	
- testeFolgaGeometrica(pontoCentralFolga : double[], g : Garra, deltaGe : double) : int	
- distanciaPontoPonto(a : double[], b : double[]) : double	
- anguloEntreVetores(v1 : double[], v2 : double[]) : double	
- distanciaPontoPlano(pontoPlano : double[], vetorNormal : double[], ponto : double[]) : double	

Figura 54 – Detalhamento da classe TrataPreensao

Já os métodos existentes são todos privados, com exceção do método `preensao(lista : ListaQuadrante, id : int, g : Garra, deltaAb : double)`. Este é o método que é chamado quando é solicitada a avaliação de preensão. Ele retorna valores inteiros correspondentes às constantes declaradas na classe. É possível identificar através do valor de retorno do método se é realizável a preensão ou, se ocorreu algum erro, qual é o motivo de a situação não levar à preensão. É neste método que é separado a etapa de exclusão da etapa de confirmação. A Figura 55 mostra como os Raciocínios Geométricos estão efetivamente divididos nos métodos da classe TrataPreensao.

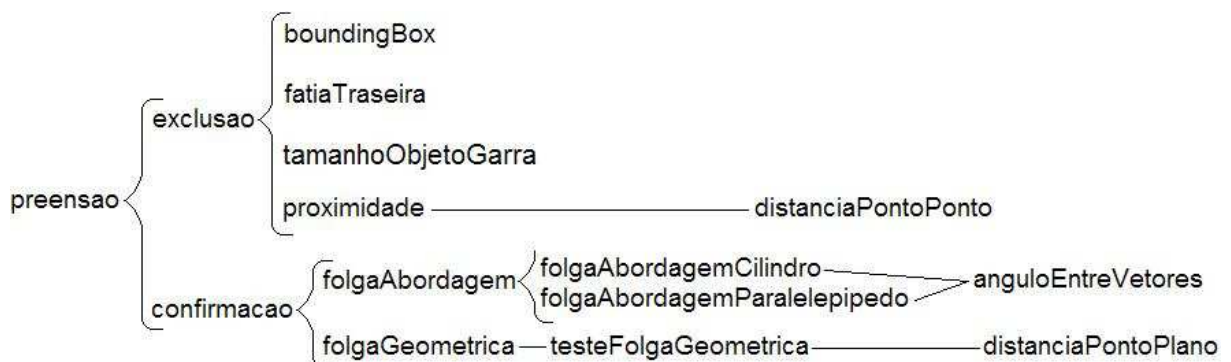


Figura 55 – Relação entre os métodos da classe TrataPreensao

Apesar de o método exclusao somente fazer chamadas aos métodos que possuem os Raciocínios Geométricos de exclusão, e do método confirmacao fazer chamadas unicamente aos métodos de Folga de Abordagem e de Folga Geométrica, eles possuem uma semelhança entre si. Ambos retornam erro caso uma de suas sub-rotinas retornarem erro. Ou seja, o método exclusao retornará erro caso um dos métodos chamados por ele também retorne. Acontece da mesma forma no método confirmacao, onde ele retorna erro caso o método folgaAbordagem ou o método folgaGeometrica retornem erro. Existem ainda métodos auxiliares que fazem cálculos matemáticos de distância entre dois pontos (distanciaPontoPonto), distancia entre um ponto e um plano (distanciaPontoPlano) e ângulo entre dois vetores (anguloEntreVetores).

Serão mostrados os pseudocódigos implementados referentes aos Raciocínios Geométricos especificados no projeto, bem como em qual método da classe TrataPreensao eles estão implementados.

Raciocínios Geométricos para validação do *Bounding Box* de Trabalho: Estes Raciocínios Geométricos foram implementados no método boundingBox e pertencem ao conjunto de raciocínios para exclusão. O método getCG() tem a função de retornar a posição do Centro de Gravidade do objeto.

Pseudocódigo:

```
Se(objeto.getCG()[0] < robo.getCentro()[0] - (robô.getBBTrabalho.getExtensão()[0]/2) ||
objeto.getCG()[0] > robo.getCentro()[0] + (robô.getBBTrabalho.getExtensão()[0]/2))
```

ERRO_BOUNDING_BOX_TRABALHO;

```
Se(objeto.getCG()[2] < robo.getCentro()[2] - (robô.getBBTrabalho.getExtensão()[2]/2) ||
objeto.getCG()[2] > robo.getCentro()[2] + (robô.getBBTrabalho.getExtensão()[2]/2))
```

ERRO_BOUNDING_BOX_TRABALHO;

APROVADO;

Raciocínios Geométricos para validação da Fatia Traseira do robô: Estes Raciocínios Geométricos foram implementados no método fatiaTraseira e utiliza a Fórmula (5.1). Eles também pertencem ao conjunto de raciocínios para exclusão.

Pseudocódigo:

```
angulo = atan2(objeto.getCG()[2], -objeto.getCG()[0]);
```

```

Se(angulo<0)
    angulo = 360° + angulo;
Se(angulo>310°)
    ERRO_FATIA_TRASEIRA;
APROVADO;

```

Raciocínios Geométricos para validação do Tamanho do Objeto: Estes Raciocínios Geométricos foram implementados no método tamanhoObjetoGarra. O método menor possui a função de retornar o menor valor dos três passados por parâmetro. Eles pertencem ao conjunto de raciocínios para exclusão.

Pseudocódigo:

```

Se(menor(objeto.getOBB().getExtensao()[0], objeto.getOBB().getExtensao()[1],
objeto.getOBB().getExtensao()[2]) > Garra.O)
    ERRO_TAMANHO_OBJETO_GARRA;
APROVADO;

```

Raciocínios Geométricos para validação de Proximidade do Objeto: Estes Raciocínios Geométricos foram implementados no método proximidade. É utilizado o método distanciaPontoPonto calcular a distância entre os pontos e verificar se eles estão próximos o suficiente. Estes também são considerados raciocínios de exclusão.

Pseudocódigo:

```

Se(distanciaPontoPonto(objeto.getCG(), g.getPFP()) > objeto.getR())
    ERRO_PROXIMIDADE;
APROVADO;

```

Raciocínios Geométricos de Abordagem para o Paralelepípedo: Estes Raciocínios Geométricos foram implementados no método folgaAbordagemParalelepípedo. É utilizado o método anguloAntreVetores para validar se os vetores são paralelos ou perpendiculares. No pseudo-código é utilizado uma variável que armazena o vetor ortogonal aos vetores $\overrightarrow{Upo}$ e $\overrightarrow{Elo}$. Este vetor corresponde ao outro eixo de coordenadas do objeto.

Pseudocódigo:

```

vetorOrtogonal = produtoVetorial(paralelepípedo.getUpo(), paralelepípedo.getElo());

```



```

Se (paralelismo(pinça.getElg(), paralelepípedo.getUpo())==verdadeiro)
    Padrão 2 identificado;
    APROVADO;
Senão (paralelismo(pinça.getElg(), paralelepípedo.getElo())==verdadeiro)
    Padrão 2 identificado;
    APROVADO;
Senão (paralelismo(pinça.getElg(), vetorOrtogonal)==verdadeiro)
    Padrão 2 identificado;
    APROVADO;
Senão
    ERRO_ABORDAGEM;

```

Raciocínios Geométricos de Abordagem para o Cilindro: Estes Raciocínios Geométricos foram implementados no método `folgaAbordagemCilindro`. É utilizado o método `anguloEntreVetores` para validar se os vetores são paralelos ou perpendiculares.

Pseudocódigo:

```

Se(paralelismo(pinça.getApg(), cilindro.getUpo())==true)
    Padrão 3 identificado;
    APROVADO;
Senão(perpendicular(pinça.getApg(), cilindro.getUpo())==true) {
    Se(paralelismo(pinça.getUpg(), cilindro.getUpo())==true)
        Padrão 4 identificado;
        APROVADO;
    Senão(perpendicular(pinça.getUpg(), cilindro.getUpo())==true)
        Padrão 6 identificado;
        APROVADO;
    Senão
        ERRO_ABORDAGEM;
}
Senão {
    Se(perpendicular(pinça.getElg(), cilindro.getUpo())==true)
        Padrão 5 identificado;

```

```

        APROVADO;
    Senão
        ERRO_ABORDAGEM;
}

```

Raciocínios Geométricos de Folga Geométrica: Estes Raciocínios Geométricos foram implementados no método testeFolgaGeometrica.

Pseudocódigo:

```

pontoCentralFolga = objeto.getCG() + (deltaGe/2)*g.getAp();
Se(distanciaPontoPlano(pontoCentralFolga, g.getUpg(), g.getPFP())>deltaGe)
    ERRO_DELTA_UP;
Se(distanciaPontoPlano(pontoCentralFolga, g.getElg(), g.getPFP())>deltaGe)
    ERRO_DELTA_EL;
Se(distanciaPontoPlano(pontoCentralFolga, g.getApg(), g.getPFP())>deltaGe)
    ERRO_DELTA_AP;
APROVADO;

```

Os Raciocínios Geométricos de exclusão foram priorizados levando em consideração a usabilidade do simulador. Buscou-se uma facilidade maior para identificação dos erros por parte do usuário. Desta forma, primeiramente são realizados os testes de exclusão que utilizam informações de alcance do braço robótico e posteriormente testes de distância. Os testes de alcance são realizados na seguinte sequência: teste de *Bounding Box* de Trabalho e teste de Fatia traseira do Robô. Desta forma, primeiramente é identificado se o objeto está longe o suficiente que se torna impossível de o braço robótico alcançar. Em seguida é identificado se a base do robô consegue rotacionar o suficiente para pegar o objeto. Terminando a análise de exclusão é feito o teste de distância euclidiana entre a garra e o objeto. Não existe priorização nos testes de Folga de Abordagem de Geométrica, pelo fato de ambos possuírem análises mais específicas ao objeto.

As verificações de angulação existentes nos Raciocínios Geométricos de Folga de Abordagem são realizadas levando em consideração a Folga de Abordagem. Para evitar que somente a situação perfeita de paralelismo, por exemplo, seja aceita pelos raciocínios, o algoritmo faz uso do valor de Folga de Abordagem para estabelecer os limites deste

paralelismo. Ou seja, situações que possuem angulações imperfeitas também são consideradas paralelas ou perpendiculares, desde que satisfaça o limite de Folga de Abordagem.

5.1.3. Códigos Modificados

Para que fosse possível testar o algoritmo implementado, foram necessárias modificações em algumas classes já existentes no simulador bem como correção de alguns *bugs* encontrados. Serão apresentadas as modificações realizadas e as justificativas para tais modificações fossem necessárias.

Classe PartManager

A classe PartManager possui a função de gerenciar os objetos inseridos no ambiente, gerando-os, removendo-os e disponibilizando-os para detecção de colisão. Quatro principais modificações foram feitas para que pudesse ser testado de maneira satisfatória o algoritmo de avaliação de preensão: geração paralelepípedos em toda a mesa, geração de cilindros em toda a mesa, geração de esferas em toda a mesa e adequação das OBBs para que fosse possível a preensão de esferas e cilindros pela frente.

Na versão anterior do simulador (SANTOS et al., 2007b), os objetos eram gerados com o intuito de ser possível a pega deles pelo robô. Desta forma, os objetos eram criados em uma área onde o robô geralmente alcançava. Porém, neste trabalho foram implementados vários testes de exclusão que, para serem testados, necessitavam de objetos gerados fora do alcance do robô virtual. Para que fossem gerados realisticamente, os objetos não poderiam se encontrar perto demais do robô e principalmente sobre a mesa. Foi modificado o algoritmo de geração de objetos para que isto fosse possível.

Cada primitiva necessitava de um tratamento específico para a geração, visto que cada tipo geométrico possui informações diferentes entre si, como, por exemplo, raio, altura e comprimento. Inicialmente, para poder criar os objetos na mesa, precisa-se saber os limites da mesa, para que seja satisfeita as considerações explicadas. A Figura 56 mostra os valores modificados dos limites da mesa no plano XZ.

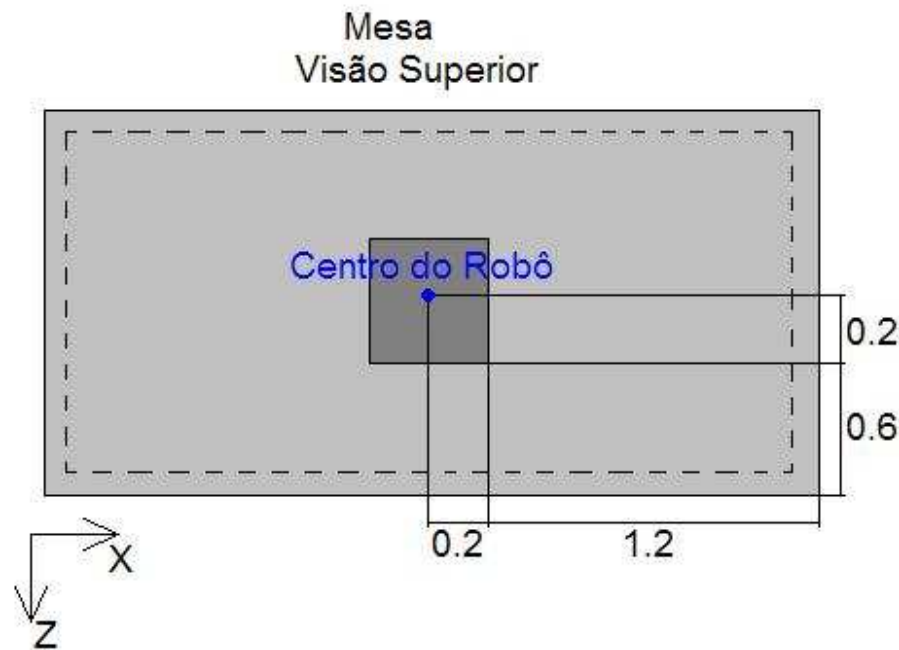


Figura 56 – Limites da mesa

Para evitar que metade dos objetos ficasse para fora da mesa, os limites máximos de posição dos objetos nos eixos X e Z foram reduzidos, enquanto que os limites mínimos não foram alterados. A Figura 56 mostra a área onde o objeto pode ser gerado (área pontilhada) em relação aos limites totais da mesa. A Tabela 8 mostra a variação dos limites nos eixos X e Z para cada tipo de primitiva.

	Limite máximo em X	Limite máximo em Z
Paralelepípedo	$1.2 - \text{maior}(\text{paralel}.d, \text{paralel}.w)$	$0.6 - \text{maior}(\text{paralel}.d, \text{paralel}.w)$
Cilindro	$1.2 - \text{maior}(\text{cilin}.r, \text{cilin}.h)$	$0.6 - \text{maior}(\text{cilin}.r, \text{cilin}.h)$
Esfera	$1.2 - \text{esf}.r$	$0.6 - \text{esf}.r$

Tabela 8 – Limites para geração de Objetos

Com a imposição de limites maiores do que os estipulados anteriormente, cria-se a possibilidade de testar os Raciocínios Geométricos de exclusão que levam em conta o posicionamento do objeto em cena. Houve também pequenas mudanças nos valores limites dimensionais dos objetos para que eles ficassem em tamanhos que possibilitem a preensão.

Mesmo com estas modificações na geração de objetos, ainda foi necessária mais uma mudança que possibilitaria o robô de pegar os objetos pela frente. Todas as OBBs dos objetos, com exceção as OBBs dos paralelepípedos, estavam sendo geradas com seus vetores de

orientação paralelos aos eixos do sistema de coordenadas global. Dependendo da posição do objeto, isto impossibilitava a preensão de um cilindro ou de uma esfera pela frente. A Figura 57 ilustra uma situação que ocorre colisão durante a preensão de maneira errônea, eliminando-a indevidamente.

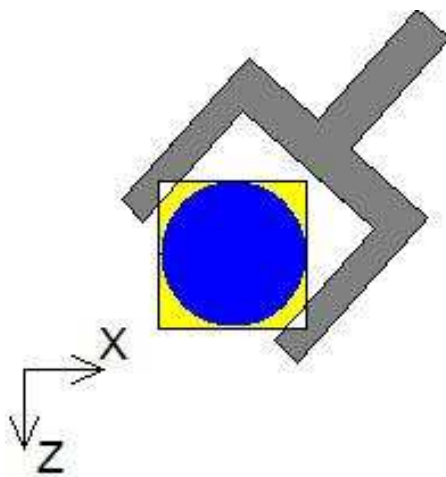


Figura 57 – Colisão errônea ao pegar um objeto

Para que seja possível a preensão do objeto, é necessário rotacionar a OBB de forma que ela fique alinhada com o plano do braço do robô. Para isto, as OBBs dos objetos dos tipos Esfera e Cilindro são rotacionadas em torno de eixo Y um valor em determinado pela Fórmula (5.1), já utilizada anteriormente. Ao rotacionar as OBBs, elas continuam limitando os objetos corretamente, porém agora permitem a preensão deles com a garra vinda pela frente. Ainda assim, o problema não foi sanado por completo. Decorrente do uso de OBBs, percebe-se que a preensão de Esferas e Cilindros por cima também são afetadas. Somente algumas angulações de rolamento da garra possibilitam a pega destes objetos por cima.

Classe PrincipalComandos

A classe PrincipalComandos foi modificada principalmente no que diz respeito à interface. Novos elementos foram adicionados nesta classe que possui herança da classe Java *applet*. Outras modificações também foram feitas, como a inclusão do sistema de avaliação de preensão, este executando juntamente com o sistema de colisões implementado por Santos (2007a).

A interação do sistema de colisões com o sistema de avaliação de preensão se dá somente no momento em que o usuário solicita a preensão de um objeto. Foram adicionadas

algumas linhas de código que impedem a execução dos Raciocínios Geométricos se estiver ocorrendo alguma colisão no ambiente.

Foi necessária uma outra modificação nesta classe para manter coerentes com o sistema de coordenadas do robô os valores de posicionamento dos objetos inseridos no ambiente. Para isto, é feita uma conversão do sistema de coordenadas do VRML para o sistema de coordenadas do robô, o que difere somente em uma translação negativa no eixo Y no sistema de coordenadas do VRML. Foi necessária esta translação para que o robô ficasse sobre a mesa no ambiente virtual.

Classe PrincipalControles

Poucas modificações foram realizadas da classe PrincipalControles. Estas modificações ocorreram basicamente em alguns elementos visuais da interface, capacitados por informar os valores das coordenadas do *PFP* ao usuário. Estes elementos passaram a apresentar ao usuário os valores numéricos com as cores dos eixos de coordenadas ao qual eles estão atrelados.

Outra modificação foi feita buscando manter os valores do *PFP* coerentes com o sistema de coordenadas do robô. A coerência dos valores foi obtida através de uma conversão do sistema de coordenadas do VRML para o sistema de coordenadas do robô, visto que o robô sofreu uma translação negativa no eixo Y do sistema de coordenadas global do VRML.

Classe RoboCollision

As mudanças realizadas nesta classe tinham o intuito de consertar alguns *bugs* encontrados durante primeira tentativa de interação entre o sistema de colisão e o sistema de avaliação da preensão. No momento em que a garra era manipulada e posta em uma posição para ser iniciada a avaliação da preensão, detectava-se uma colisão errônea entre a garra e o objeto a ser pego. Muitas vezes em que um objeto estava entre os dedos da garra, era detectada uma colisão que, graficamente, não ocorria.

Como é possível visualizar na Figura 58, a colisão se dava pelo fato de a OBB criada para o suporte dos dedos da garra estar mais esticada e achatada do que o normal. Havia uma troca nas dimensões da OBB durante a sua criação, fazendo com que seus dados iniciais fossem incorretos. Era acusada a colisão quando havia um objeto ente os dedos da garra, mesmo que esta não ocorresse visualmente. Com a correção dos valores de criação da OBB

de suporte dos dedos, passou a ser possível a interação entre o sistema de colisão e o sistema de avaliação da preensão.

Outro problema foi encontrado na disposição das OBBs da estrutura do robô. Ao atualizar as OBBs (quando o robô é movimentado no ambiente virtual), era atualizada a OBB da base de forma incorreta. Como o robô sofreu uma translação negativa no eixo Y, as suas OBBs também necessitavam sofrer a mesma translação. Todas as OBBs da estrutura robótica eram atualizadas corretamente, porém a OBB da base não estava sendo transladada como as outras. Isto levava a uma colisão sempre que a estrutura do braço do robô estivesse em uma posição mais elevada no eixo Y. Ao transladar a OBB da base para seu respectivo lugar, a colisão com a base passou a ser detectada corretamente.

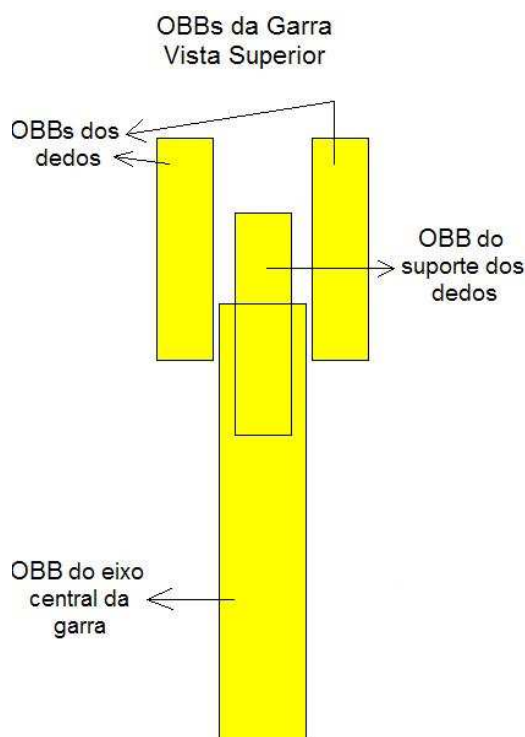


Figura 58 – Bug na criação das OBBs da garra

Ausência de Protótipos de Funções no arquivo “Simulador.wrl”

Ao executar o aplicativo em diferentes computadores, percebeu-se que em algumas máquinas era apresentado um erro ao abrir o arquivo VRML do simulador. Uma mensagem de erro, durante a interpretação realizada pelo compilador VRML, era apresentada ao iniciar o aplicativo no *browser*.

A raiz do problema consistia na ausência de protótipos de certas funções implementadas em Javascript localizadas no arquivo “Robo.wrl”. Estas funções eram chamadas pelo arquivo “Simulador.wrl” no momento que ele abria o arquivo “Robo.wrl”, porém não eram encontradas, pois faltavam os protótipos no arquivo “Robo.wrl”. Ao criar o ambiente, o compilador VRML não encontrava estas funções e retornava uma mensagem de erro. Devido ao fato de este erro ser informado somente em alguns computadores, ele demandou um tempo maior para ser resolvido. A solução do problema foi apenas declarar os protótipos das funções no arquivo “Robo.wrl” para que elas fossem encontradas pelo arquivo “Simulador.wrl”.

5.1.4. Como Utilizar a Avaliação de Preensão

Após terem sido esclarecidas as considerações decorrentes durante a implementação do sistema, será explicado como o simulador pode ser utilizado por um usuário iniciante em robótica. A modelagem comportamental do sistema de avaliação de colisões é mostrada nas Figuras 59 e 60. As relações entre os módulos do sistema são ilustradas no Diagrama de Seqüência de eventos.

Como o elemento que inicia a avaliação da preensão pode sofrer modificações e possui diferentes funções, é necessário especificar pelo menos dois Diagramas de Seqüência (dois estados possíveis) para a funcionalidade de avaliação de preensão. Primeiramente será especificado o Diagrama de Seqüência dos eventos que ocorrem no sistema quando o usuário solicita a avaliação da preensão, este mostrado na Figura 59. Logo abaixo, na Figura 60, é mostrado o Diagrama de Seqüência de eventos quando ocorre a solicitação de liberação o objeto no ambiente.

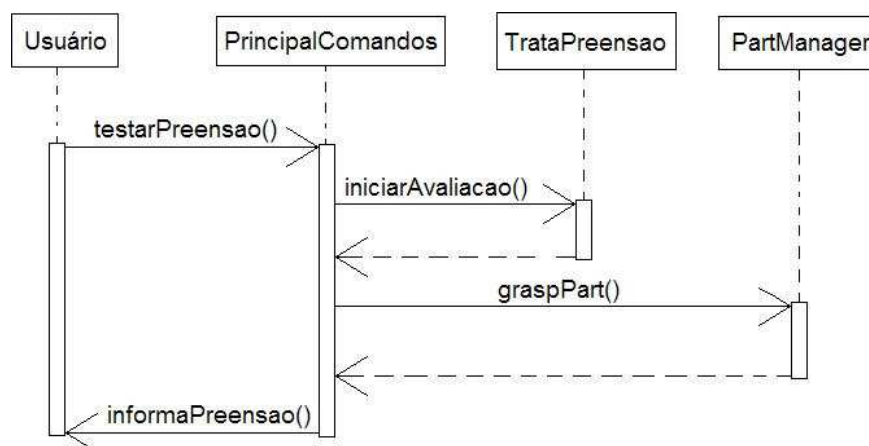


Figura 59 – Diagrama de Seqüência – Pegar Objeto

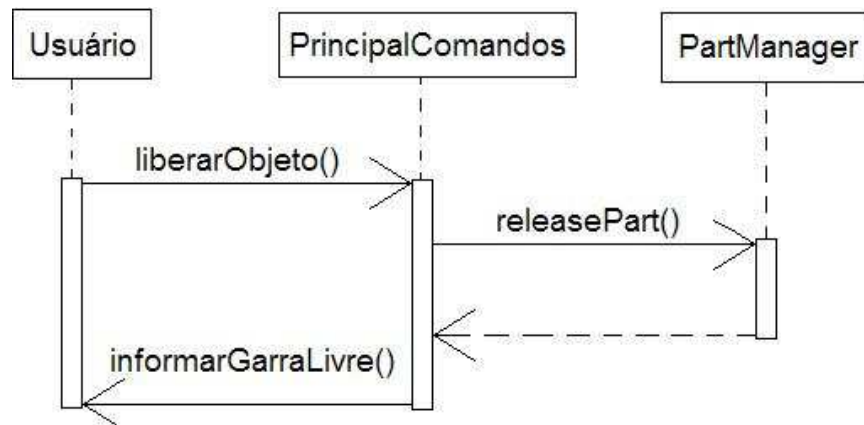


Figura 60 – Diagrama de Seqüência – Liberar Objeto

Fazendo uma análise da Figura 59, quando o usuário solicitar a avaliação da preensão, a *applet* `PrincipalComandos` inicia a execução dos Raciocínios Geométricos implementados na classe `TrataPreensao`. Terminando o processamento, um dos objetos é considerado pego pelo sistema e este é anexado na garra do robô virtual. As informações mostradas ao usuário após a execução dos Raciocínios Geométricos são: o *feedback* textual mostrado em um elemento na interface, a troca do *label* do botão que avalia a preensão, e a inclusão do objeto na garra. A Figura 61 mostra o botão que inicia a avaliação da preensão em seu estado inicial (Figura 61 (a)) e em seu estado quando a preensão foi realizada (Figura 61 (b)). Percebe-se ainda que na área de *feedback*, localizada logo abaixo do botão citado, é mostrada uma mensagem que a preensão foi realizada com sucesso. Vale lembrar que isto só acontecerá segundo a máquina de estados explicada na Figura 51.

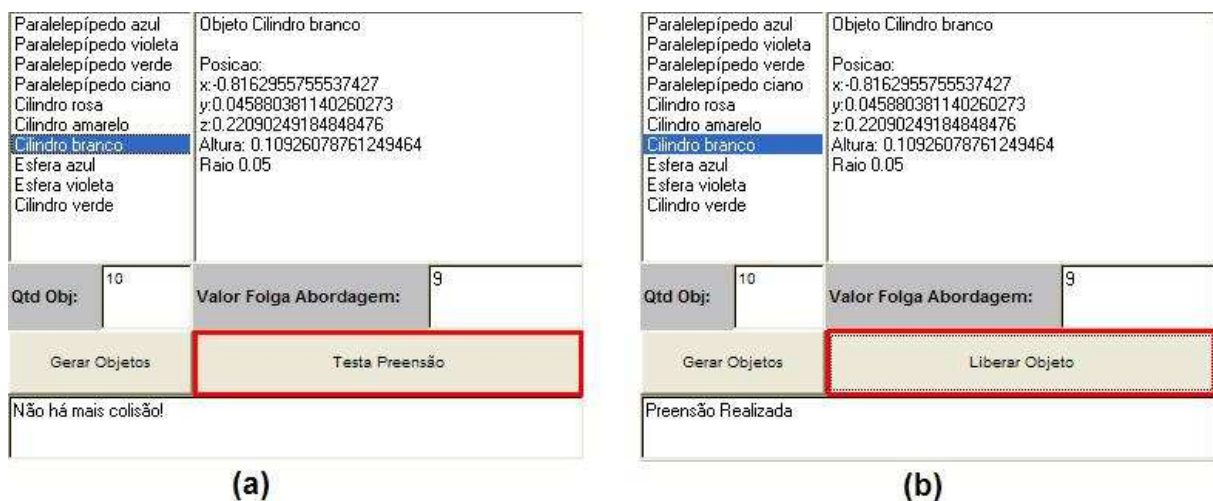


Figura 61 – Diferentes estados do botão de teste de Preensão

A partir do momento que o robô está segurando um objeto, é possível que este seja manuseado e, posteriormente, liberado da garra. Esta é outra função que o botão de preensão desempenha, e segue, na Figura 60, a seqüência de eventos que ocorrentes no sistema quando esta função é solicitada. Basta o usuário pressionar novamente neste botão, estando o objeto segurado pela garra, para que a liberação do objeto seja feita. No momento que o usuário pressionar no botão solicitando soltar o objeto, estando este localizado no ambiente de modo que seja possível liberá-lo, o sistema modifica o estado do botão para o seu estado inicial (Figura 61 (a)), libera o objeto no ambiente e mostra uma mensagem na área de *feedback* de que o objeto foi solto. Com o objeto solto no ambiente, existe a possibilidade de o robô virtual pegar outro objeto, estando o botão de avaliação em seu estado inicial.

A versão do simulador com a funcionalidade de avaliação de preensão está disponível na *web* através do *site* (LARVA, 2007). Neste sítio também se encontra a versão anterior, com detecção de colisão e geração de objetos.

5.2. Testes Funcionais

Os testes funcionais foram realizados procurando colocar em execução cada sub-rotina projetada do algoritmo de Raciocínios Geométricos. Desta forma, primeiramente serão mostrados testes com os procedimentos que fazem parte da etapa de exclusão, depois serão mostrados testes com a etapa de Confirmação e será finalizado com algumas considerações sobre a *performance* do algoritmo.

5.2.1. Testes com a Etapa de Exclusão

Os testes de exclusão implementados foram: *Bounding Box* de trabalho, Fatia Traseira e Proximidade. A seguir serão mostradas situações que reprovam ao serem submetidas aos Raciocínios Geométricos de exclusão.

***Bounding Box* de Trabalho**

Para realizar o teste funcional de *Bounding Box* de trabalho foi incluso no ambiente o *Bounding Box* de trabalho do robô para uma melhor visualização do resultado. A Figura 62 mostra um caso específico que um objeto, ao ser submetido para a análise de preensão, reprovou no teste de *Bounding Box* de trabalho. Como é mostrado na Figura 62, o paralelepípedo amarelo, selecionado em um componente da interface e com suas informações

listadas em outro, estava fora do *Bounding Box* de trabalho do robô virtual. Isto fez com que o algoritmo reprovasse o caso de prensão deste objeto. É possível perceber a reprovação através do componente de *feedback*, mostrado na caixa localizada na Figura 62, que informa a falha de prensão ao usuário.

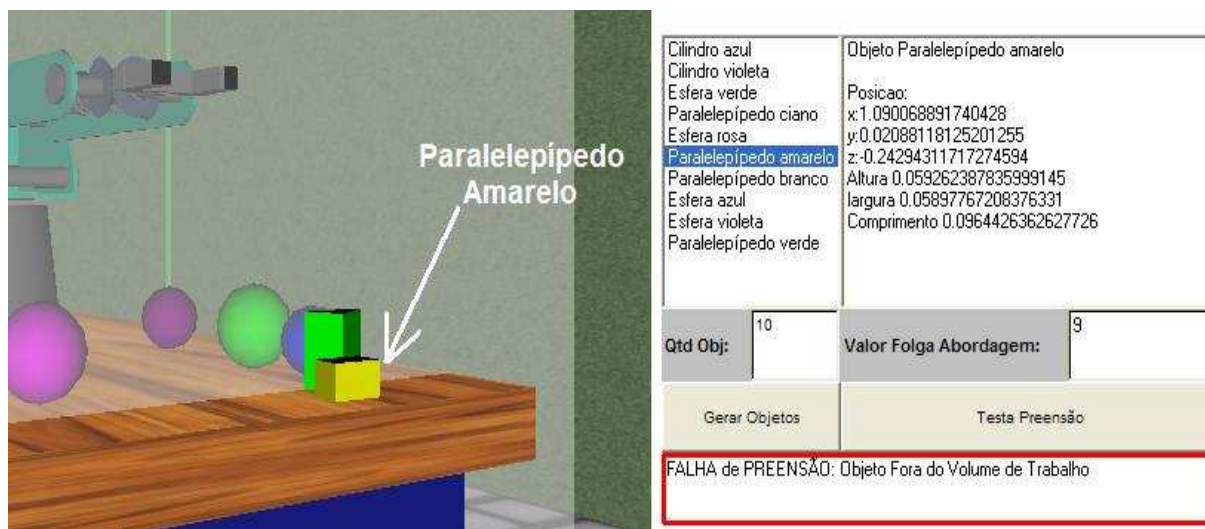


Figura 62 – Teste Funcional – *Bounding Box* de trabalho

Fatia Traseira

A situação submetida para teste consistia em um objeto do tipo Cilindro localizado em uma área onde o robô não alcança com a garra devido ao seu limite de rotação da base. Como mostra a Figura 63, o robô virtual já se encontra com a sua rotação da base no valor máximo, este correspondente a 310°. Mesmo com o objeto se encontrando interno ao *Bounding Box* de trabalho do robô, ele é incapaz de alcançar aquela área especificamente. Este foi o principal motivo para este teste possuir a prioridade menor do que o teste de *Bounding Box* de trabalho.

É possível visualizar o motivo da reprovação através da mensagem que informa a falha de prensão mostrada na área de *feedback*, esta interna à caixa localizada na Figura 63. O objeto testado foi o Cilindro verde, selecionado em um componente da interface e também mostrado na figura.

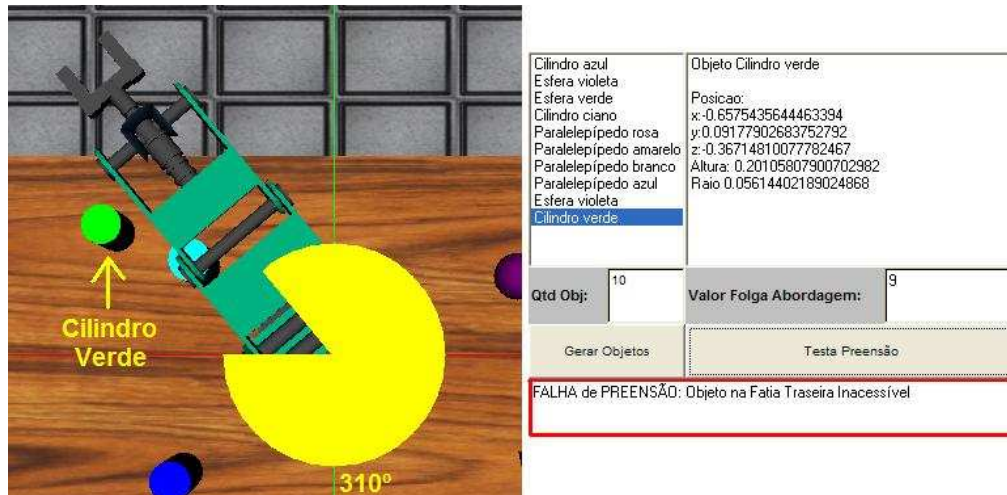


Figura 63 – Teste Funcional – Fatia Traseira

Proximidade

A situação submetida para o teste de proximidade, mostrada na Figura 64, consistia na tentativa de preensão de um paralelepípedo branco. Mesmo com o paralelepípedo estando interno ao *Bounding Box* de trabalho robô e não estando localizado em uma área que o robô não alcance devido o limite de rotação da base, ele reprovou no teste de proximidade ao ser submetido para a avaliação de preensão. Este foi o fator que definiu este teste com a menor prioridade dos testes de exclusão.

Como é possível visualizar na Figura 64, ao ser chamado o algoritmo de avaliação de preensão, a garra do robô se localiza muito afastada do centro do objeto, fazendo com que esta situação reprove no teste de proximidade. A caixa localizada na Figura 64 mostra a região de *feedback* informando ao usuário que o objeto selecionado (paralelepípedo branco) não está próximo da garra o suficiente para continuar as análises de preensão.

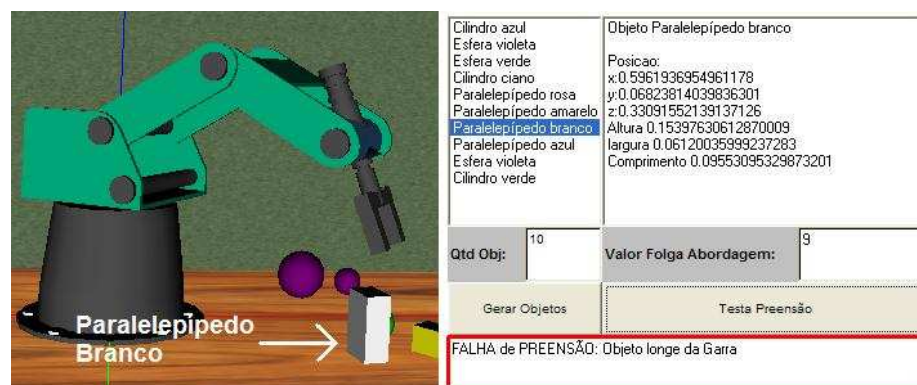


Figura 64 – Teste Funcional – Proximidade

5.2.2. Testes com a etapa de Confirmação

Os Testes Funcionais foram feitos levando em consideração os dois diferentes tratamentos que existem na etapa de Confirmação: o tratamento de Folga de Abordagem e de Folga Geométrica.

5.2.2.1. Testes de Folga de Abordagem

Foram realizados testes que reprovam na etapa de Folga de Abordagem para os diferentes tipos geométricos que podem ser inseridos no ambiente. Como foi visto no projeto do algoritmo de Raciocínios Geométricos, a etapa de Folga de Abordagem é mais específica aos objetos. Desta forma, foram realizados testes funcionais para cada primitiva possível de ser gerada no ambiente. Como a primitiva esfera não possui análises de Folga de Abordagem, não foram realizados testes de Abordagem para esta primitiva.

Paralelepípedos

O teste funcional de Folga de Abordagem para Paralelepípedos tentou produzir uma situação instável durante a preensão. Como é possível visualizar na Figura 65, caso a garra fechasse os dedos envolta do objeto, ele sofreria um arrastamento sobre a mesa. A situação foi considerada instável devido ao fato de o valor de Folga de Abordagem usado para este teste ter limitado o rolamento da garra durante a preensão. A Folga de Abordagem, para paralelepípedos, busca limitar a preensão somente dos paralelepípedos que estão alinhados (ou quase) com a garra.

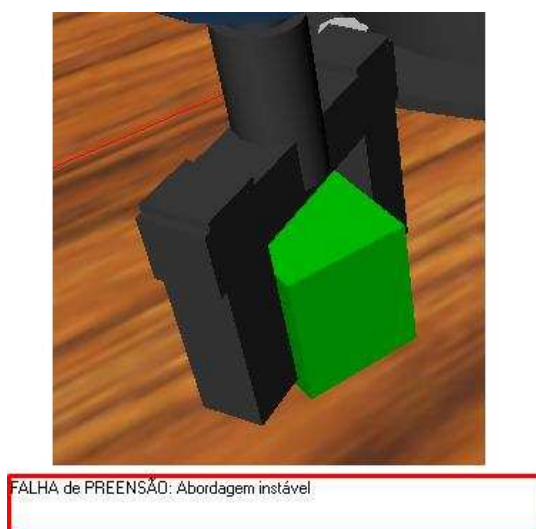


Figura 65 – Teste Funcional – Abordagem para Paralelepípedos

Cilindros

O próximo teste funcional de Folga de Abordagem consistiu em pegar um cilindro pela frente, porém com uma rotação *roll* da garra que levava a uma preensão instável. Como é mostrado na Figura 66, caso a garra fechasse os dedos para realizar a preensão, ocorreria uma torção do cilindro e da garra. Novamente a situação apresentada foi considerada instável pelo fato de o limite de Folga de Abordagem ter sido menor do que o rolamento da garra naquele instante. Como a garra não estava alinhada ao eixo central do cilindro, a situação foi considerada pelo algoritmo de avaliação de preensão como sendo instável.



Figura 66 – Teste Funcional – Abordagem para Cilindros

5.2.2.2. Testes de Folga Geométrica

Para finalizar, serão mostrados os testes funcionais de Folga Geométrica submetidos ao algoritmo de avaliação de preensão. Como foi definido no projeto que a etapa de Folga Geométrica é mais genérica (sendo o mesmo tratamento para todas as primitivas), foram feitos testes onde o *PFP* se localizava fora do volume de folga. Devido ao fato de o volume de Folga ser um Cubo, os testes funcionais apresentavam situações que o *PFP* estava fora do cubo nos seus três eixos. Os eixos do volume de Folga Geométrica são paralelos aos vetores

de orientação da Garra. Então serão mostrados os testes funcionais de acordo com os eixos de orientação da garra.

Eixo paralelo ao $\overrightarrow{Apg}$

O sólido submetido ao primeiro teste de Folga Geométrica foi um paralelepípedo. O teste consistia em criar uma situação onde era impossível a prensão pelo motivo de o ponto *PPF* se localizar fora volume de Folga. Como o volume de folga foi criado abaixo do Centro de Gravidade do objeto (para o caso da Figura 67) a situação mostrada foi reprovada. O principal motivo da reprovação desta situação é o de evitar prensões onde somente uma pequena área da superfície do objeto esteja em contato com os dedos da garra. Reprovando prensões deste tipo, evitam-se oscilações no objeto ao mover o robô e sobrecarga da garra em uma situação real.



Figura 67 – Teste Funcional – Folga Geométrica paralela ao $\overrightarrow{Apg}$

Eixo paralelo ao $\overrightarrow{Upg}$

O segundo teste funcional de Folga Geométrica é semelhante ao primeiro, porém difere no eixo do volume de folga ao qual o é validada a Folga Geométrica. Na tentativa de pegar o paralelepípedo pela frente, o algoritmo de Raciocínios Geométricos reprova a tentativa de prensão mostrada da Figura 68. Pelo fato de garra se localizar muito acima do Centro de Gravidade do paralelepípedo e, sendo que a prensão está sendo realizada pela frente, a situação ilustrada é reprovada. Caso o *PPF* estivesse mais para baixo e estivesse interno ao

volume de folga, a situação seria aprovada. Disposições de objetos e garra que seguem este padrão são reprovadas também pelo motivo de se evitar oscilações no objeto e a sobrecarga da garra em uma situação real.

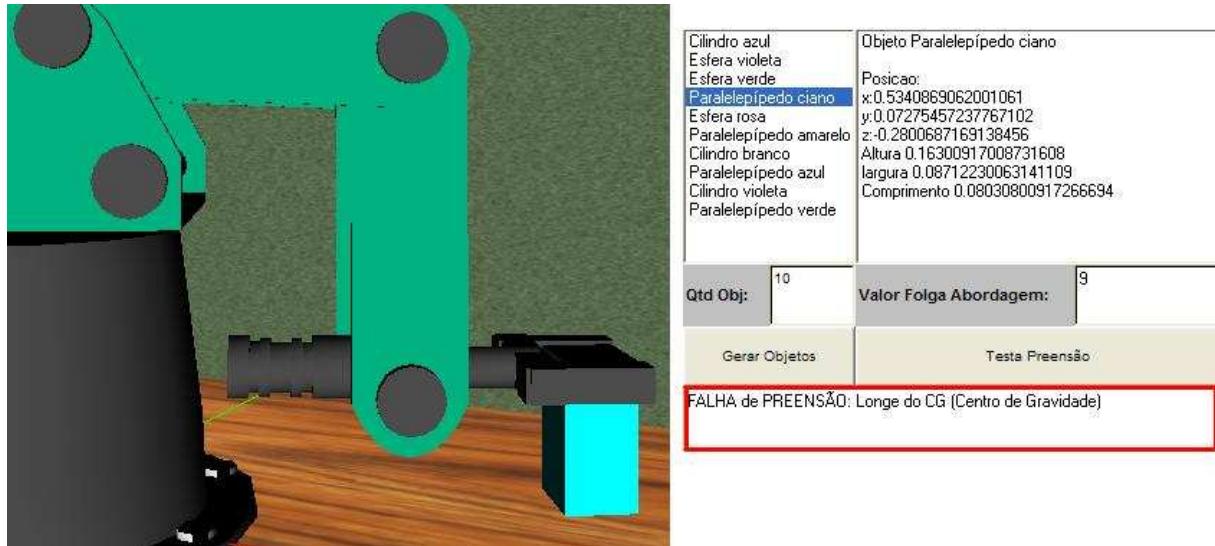


Figura 68 – Teste Funcional – Folga Geométrica paralela ao Upg

Eixo paralelo ao $\overrightarrow{Elg}$

Foram realizados vários testes funcionais buscando uma disposição onde um objeto é reprovado por exceder o limite do volume de folga, especificamente no seu eixo paralelo ao vetor $\overrightarrow{Elg}$. Percebeu-se que situações similares às mostradas na Figura 69 são difíceis de acontecer. Como a abertura máxima entre os dedos da garra é pequena, é exigido que o objeto possua suas dimensões menores ainda. Para ocorrer uma situação em que o volume de folga seja extrapolado no seu eixo paralelo ao vetor $\overrightarrow{Elg}$, as dimensões do objeto precisam ser pequenas o bastante para possibilitar a disposição mostrada na Figura 69.

Para que seja possível ocorrer este tipo de disposição, é necessário diminuir os valores mínimos de dimensão dos objetos. Percebe-se que, ao diminuir estes valores, geometrias muito pequenas são geradas no ambiente. Além disto, a diferença entre os limites superiores e inferiores seriam grandes, possibilitando a geração de objetos que seriam geometricamente “finos” e/ou “compridos”. Desta forma, é realisticamente inviável gerar estes tipos de objetos, visto que objetos que possuem sua base com uma área muito pequena têm a tendência natural de queda em uma situação real. Apesar de ser difícil ocorrer situações semelhantes à mostrada

na Figura 69, esta parte dos testes de Folga Geométrica não foi retirada, visto que em futuras versões do simulador estes tipos de situações poderão ocorrer.

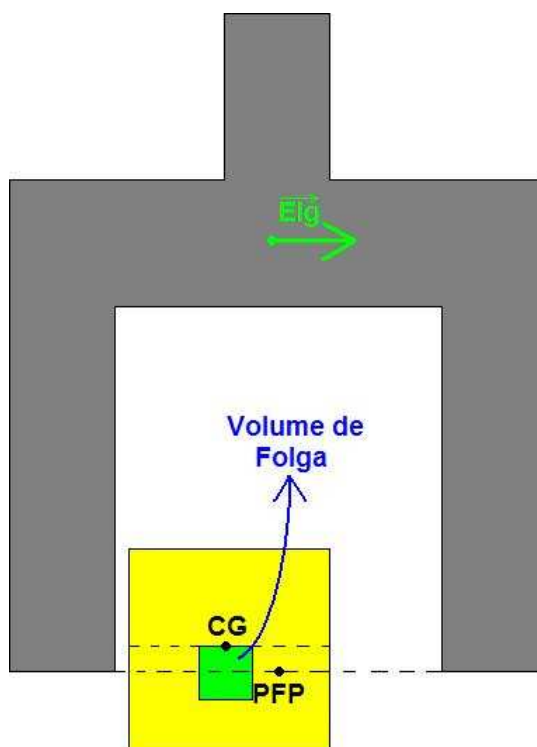


Figura 69 – Teste Funcional – Folga Geométrica paralela ao Elg

Durante a realização dos testes de Folga Geométrica, percebeu-se uma situação em especial onde a prensão é aceitável, porém o algoritmo de Raciocínios Geométricos a reprova. Como mostra a Figura 70, mesmo com a garra abraçando todo o objeto sem que ocorram colisões, caracterizando uma situação de prensão estável, ela é reprovada. A situação mostrada na Figura 70 ilustra um caso de reprovação em um dos testes de exclusão (proximidade), porém isto também pode ocorrer em um caso especial de reprovação por Folga Geométrica.

Como o teste de proximidade é feito através de uma esfera envolvente (R), é possível ocorrer uma situação onde o PPF esteja interno a esta esfera, porém ultrapasse o volume de Folga Geométrica. Este caso também é reprovado, pois excede o limite de Folga Geométrica, mais especificamente no eixo do volume de folga paralelo ao vetor $\overrightarrow{Apq}$. Pelo motivo de o PPF ultrapassar todo o volume de Folga Geométrica e, conseqüentemente localizar-se fora do volume, o caso é reprovado erroneamente.



Figura 70 – Reprovação errônea do algoritmo de Raciocínios Geométricos

Apesar de reprovar certas situações que levam a preensão, foi decidido não modificar os Raciocínios Geométricos para este caso. Esta decisão foi tomada com base em que, futuramente, poderão ocorrer casos onde dois objetos estão um sobre o outro no ambiente, caracterizando situações que não promovem a preensão (dois objetos não podem ser pegos ao mesmo tempo). Estes casos são reprovados devido ao mesmo motivo que reprovou a situação mostrada na Figura 70.

5.2.3. Teste de Performance

Em se tratando de *performance*, os sistemas de detecção de colisão e de avaliação de preensão em momento algum executam ao mesmo tempo. A disputa pelo processador não ocorre, pois o *software* apenas executa o sistema de avaliação de preensão quando o robô está parado para iniciar uma preensão. Quando o ambiente não está em movimento, o sistema de detecção de colisão não faz processamento algum. O mesmo ocorre quando o ambiente está em movimento, onde o sistema de detecção de colisões utiliza o processador enquanto que o sistema de avaliação de preensão não utiliza.

Testes de *performance* foram realizados a fim de obter uma verificação se realmente foi alcançada uma otimização de processamento com o uso dos Raciocínios Geométricos. Após vários testes com um temporizador implementado no próprio código do robô virtual, percebeu-se que todo o algoritmo é percorrido em menos de 10 (dez) milissegundos. Os testes foram realizados em um computador com processador Pentium 4 de 2.4 Ghz e 512 MB de memória RAM que é uma máquina considerada “comum”. O simulador também não demandou um tempo perceptível executando em um computador Pentium 3 de 500 Mhz e 256

MB de memória RAM, que pode ser considerado a máquina com os recursos mínimos para esta implementação.

O tempo necessário demandado por cada parte do algoritmo (testes de Exclusão e Confirmação) não foi calculado. Devido ao fato de a função Java *System.currentTimeMillis()*, que retorna o tempo atual da máquina através de um valor inteiro longo, necessitar de mais de dez milissegundos para atualizar o tempo retornado por ela, foi impossível saber quanto tempo estimado cada parte do algoritmo necessita para a sua execução. Percebeu-se que somente em alguns testes de *performance* o tempo retornado no início e o tempo retornado no fim da execução dos Raciocínios Geométricos foram diferentes. Ou seja, a execução dos Raciocínios Geométricos, na maioria dos casos, era finalizada antes mesmo de a função *System.currentTimeMillis()* atualizar o tempo retornado.

Portanto, conclui-se que o algoritmo de Raciocínios Geométricos possui sua execução satisfatoriamente rápida e que foi atingido o objetivo do trabalho ao se buscar otimização de processamento. Percebe-se, ainda, a possibilidade de empregar o sistema de avaliação de preensão junto ao sistema de detecção de colisões, fazendo os dois executarem ao mesmo tempo na máquina. Viu-se a possibilidade de criar uma versão do simulador com propósitos de produção, podendo ser utilizado em ambientes onde a preensão não é avaliada, mas sim é procurada pelo usuário sem intuítos educativos. Devido à pequena quantidade de tempo demandado para executar os Raciocínios Geométricos, percebe-se que a simulação do ambiente não será visualmente afetada caso o sistema de avaliação de preensão e o sistema de detecção de colisões executarem ao mesmo tempo.

5.3. Considerações Finais

Com principal intuito de testes, o algoritmo de Raciocínios Geométricos foi posto em execução sem que o sistema de detecção de colisões estivesse executando juntamente com ele. Foi identificado que o sistema de avaliação de preensão depende de que o sistema de detecção de colisões esteja executando paralelamente com ele. Alguns testes de exclusão especificados no projeto, não foram implementados. Isso criou uma dependência direta do sistema de detecção de colisões por parte do sistema de avaliação por Raciocínios Geométricos. Testes de exclusão que dizem respeito à acessibilidade da garra ao objeto fazem análises de colisão, porém estes testes não foram implementados justamente pelo fato de o sistema de detecção de colisões já tratar este problema.

Como dito anteriormente, o sistema de detecção de colisões pode ser desativado de maneira que este não interfira no resultado final do sistema de avaliação de preensão. Mesmo possuindo dependência mencionada, um sistema executa satisfatoriamente sem que o outro esteja ativado. Isto se dá pelo fato de o sistema de detecção de colisões possuir uma sub-rotina responsável unicamente por identificar as colisões entre as OBBs existentes no ambiente. Caso esta sub-rotina não seja chamada, o sistema de detecção de colisões apenas atualiza as OBBs, mas não verifica as colisões entre elas. Desta forma, o sistema de detecção de colisões pode ser ativado ou desativado, não deixando de atualizar as OBBs dispostas no ambiente.

Como o sistema de avaliação de preensão faz uso de informações da OBB da garra, basta que a sub-rotina responsável por atualizar as OBBs do ambiente seja chamada para que a funcionalidade de avaliação de preensão funcione corretamente, independente se estão sendo analisadas colisões entre as OBBs. Ainda assim, o sistema de avaliação de preensão possui seu algoritmo encapsulado em uma classe, sendo que esta não modifica qualquer valor existente nas variáveis do sistema de colisões. Isto faz com que o sistema de avaliação de preensão conviva harmoniosamente com o sistema de colisões.

Para que o simulador possa ser executado, é necessário que o aplicativo esteja executando em um *browser*, mais especificamente no *Internet Explorer*. Foram feitas tentativas de executar em outros programas, porém não foi obtido sucesso. O VRML também precisa estar instalado no computador onde vai executar o simulador. Para isto, basta instalar qualquer versão do Cortona® VRML *Client*. Já para o *browser*, somente versões do *Internet Explorer* superiores a 4 e anteriores a 7 executam o simulador. Ainda assim é necessário configurar o *browser* para executar a máquina virtual Java correta. Para isto, é necessário selecionar o menu “Ferramentas” existente no *Internet Explorer* e selecionar a opção “Opções da Internet...”. Ao abrir a janela de Opções da Internet, clique na aba “Avançadas”. É necessário que todas as opções de “Java (Sun)” estejam desmarcadas e que todas as opções de “Microsoft VM” estejam marcadas. Após modificar as configurações, é necessário reiniciar o *browser* antes de iniciar o aplicativo. Caso qualquer *applet* não abra, é necessário excluir os arquivos de *cookies*. Após realizar estas mudanças, o aplicativo está pronto para ser iniciado.

6. CONCLUSÃO

6.1. Considerações Finais

A tarefa de preensão de objetos é importante para simuladores robóticos. Um robô capaz de desempenhar esta função satisfatoriamente, necessita que o seu sistema robótico tenha suporte à preensão de objetos. Sistemas robóticos que têm ausência de preensão são incapazes de interagir com o ambiente ao seu redor de maneira eficaz e precisa. Como sistemas deste tipo geralmente possuem finalidades que necessitam precisão ao interagir com o meio, é indispensável que esta tarefa esteja inclusa no sistema, tanto na parte física, através das garras, como digitalmente, através de *software*.

Apesar de a maioria dos simuladores estudados realizarem esta tarefa buscando identificar situações ótimas de preensão automaticamente, este trabalho solucionou um problema que possui uma abordagem diferente. O problema consiste em avaliar a disposição do ambiente resultante da manipulação direta do usuário, e, após analisar este ambiente, concluir se a preensão é possível de ser realizada ou não. Como se pode perceber, o propósito deste trabalho visou criar uma solução capaz de avaliar a maneira como o usuário da aplicação resolveu pegar um objeto através da estrutura robótica. Desta forma, o usuário da aplicação passa-se como estudante de manipulação da estrutura robótica existente no simulador.

Com a inserção do sistema de avaliação de preensão no simulador, percebeu-se uma contribuição educacional para o ensino de robótica. Além de os Raciocínios Geométricos identificarem situações estáveis e instáveis de preensão, eles também informam o motivo de a preensão não ter ocorrido. Como foi visto, após a requisição de avaliação de preensão o simulador mostra mensagens ao usuário que informam o motivo de a preensão não acontecer, caso não aconteça. Estas mensagens foram estrategicamente criadas para que o usuário possa identificar intuitivamente o que aconteceria de errado caso a garra fosse fechada naquele momento. Diferente dos trabalhos estudados, o motivo de uma preensão ter ocorrido (ou não) não fica somente interno à lógica implementada no *software*, mas sim é mostrado para o usuário que aprende mais a cada tentativa de preensão. Desta forma, não se garante que o sistema ensine a melhor forma de realizar uma preensão, mas sim os requisitos mínimos de como uma preensão não deve ser feita.

Evitando o uso de tratamentos físicos computacionalmente custosos, este trabalho ofereceu uma solução alternativa para o problema da preensão utilizando somente heurísticas com aspectos geométricos da garra e dos objetos. Informações como peso e atrito foram desconsiderados no decorrer da solução, visto que este tipo de análise demandaria um tempo considerável de processamento. O trabalho solucionou o problema especificamente para o robô Scorbót ER-4PC com uma garra do tipo pinça de dois dedos, como proposto no objetivo geral do trabalho. Para atingir este propósito, foram investigadas as condições geométricas através das diferentes situações que podem acontecer e foram criados os Raciocínios Geométricos com base nestas restrições.

Os Raciocínios Geométricos foram divididos em dois blocos principais: Exclusão e Confirmação. A Exclusão contém testes onde são eliminadas as situações que a preensão é impossível de ser realizada. Situações onde o objeto se encontra muito longe da garra, por exemplo, são excluídos inicialmente neste bloco. A Confirmação contém testes mais focados que tratam cada geometria de uma forma específica. Este bloco está dividido de acordo com as geometrias que podem ser inseridas no ambiente e não existe uma priorização, visto que são análises mais específicas ao objeto.

Os Raciocínios Geométricos indicam situações impossíveis e instáveis de preensão. Desta forma, a sua aplicação leva a concluir que a pega não está errada se tratando somente de dados geométricos. Apesar de uma situação ter sido aprovada pelos Raciocínios Geométricos, não necessariamente ela é correta se fosse submetida para análise em um ambiente real. Mas existe a certeza de quando uma situação é dada como instável pelos Raciocínios Geométricos, pois nem restrições geométricas a situação atende. Desta forma, é indispensável recorrer a um teste de *try out* caso uma situação necessite ser avaliada com precisão. Não se pode prescindir de um teste de campo para uma situação real, pois problemas físicos na prática são extremamente complexos e demandam calibração do algoritmo que, nesta versão da implementação, não foram considerados.

Como o algoritmo de Raciocínios Geométricos apresentou um desempenho consideravelmente rápido, o simulador poderia também possuir uma versão onde a avaliação de preensão seja realizada a todo instante que a estrutura robótica é movimentada, juntamente com sistema de detecção de colisões. Uma versão do *software* para estudante (iniciar os Raciocínios Geométricos ao ser sugerido pelo usuário) e uma versão para ambientes fabris

(executar os Raciocínios Geométricos a cada movimentação do robô) são aplicáveis, levando em consideração o desempenho apresentado.

Este trabalho contribuiu na literatura apresentando uma nova abordagem para a tarefa de prensão. Foi apresentada uma solução com a capacidade de avaliar se a prensão é possível de ser realizada ou não, diferente dos trabalhos estudados. Desta maneira, a solução deste trabalho atende a um novo propósito. Percebe-se também que podem se evitar acidentes com um robô real caso aprendizes de robótica façam uso deste simulador antecipadamente. A falta de experiência na programação de robôs por parte do usuário pode causar danos em uma estrutura robótica real durante a sua movimentação. Mesmo com a perda de precisão ao se evitar uso de física, o simulador pode prever muitos casos onde ocorreriam acidentes no ambiente. Outra contribuição deste trabalho foi a criação de um algoritmo que faz somente o uso geometria para avaliar a tarefa da prensão. Isto possibilitou a realização de um balanço entre desempenho computacional e precisão na tarefa.

6.2. Trabalhos Futuros

É proposta como trabalhos futuros a inclusão de algumas funcionalidades no simulador do robô Scorbot ER-4PC. Com mostrado no item 5.1.3. (Figura 57), um dos problemas que acontece no simulador durante a prensão de objetos ocorre devido ao uso de OBB nos objetos como volume limitante para tratamento de colisões. O fato de ocorrer colisão durante a prensão de um cilindro por cima, não identifica necessariamente que está ocorrendo colisão. A colisão das OBBs da garra com a OBB do objeto, faz com que algumas vezes casos de prensão de cilindros sejam inválidos por apontar erroneamente uma colisão. É proposto um trabalho capaz de solucionar este problema que ocorre durante a etapa de detecção de colisão.

Outro trabalho seria implementar a simulação física no ambiente virtual. Ao objeto ser solto da garra, ele fica flutuando onde ele foi solto. O fato de fazer o objeto cair, simulando gravidade, faria com que o simulador apresentasse as situações no ambiente de forma mais semelhante com o que ocorre em um ambiente real. Outro aspecto de simulação física diz respeito à rotação necessária para simular a movimentação do objeto pego. A partir do instante que um objeto é pego, ele não é rotacionado de acordo com a garra. No momento que a garra sofre mudança na rotação de uma de suas juntas (*roll* ou *pitch*), o objeto não é rotacionado em relação à junta rotacionada de maneira a seguir a movimentação garra. Seria

uma proposta de trabalho futuro fazer a simulação física levando em conta estes dois aspectos mencionados.

Outra funcionalidade seria incluir no simulador um sistema de resposta da colisão. O fato de um objeto cair quando a garra colide com ele deixa o ambiente simulado mais realístico. Outra abordagem para a resposta da colisão seria limitar as variáveis de junta do robô caso ocorra uma colisão. A movimentação do robô poderia ser parada caso fosse detectada uma colisão. Isto faz com que seja mais simples de visualizar regiões do ambiente onde o robô não pode acessar devido a grande quantidade de colisões decorrentes durante a sua manipulação. Regiões onde existe um acúmulo de objetos, não seriam acessíveis se esta abordagem de resposta de colisão fosse imposta.

Como a proposta de tratamento de preensão deste trabalho atendia ao objetivo específico de avaliar o usuário, é proposta a inserção de uma funcionalidade que mostre uma resposta ótima ao usuário. Apesar de ser uma abordagem semelhante aos trabalhos estudados, ela também pode ter sua contribuição educacional. A resposta pode ser mostrada através de uma animação da sequência de passos a serem seguidos para a preensão. Isto pode fazer com que o aprendizado do usuário possa seja maximizado, caso o instrutor que esteja ensinando sobre a estrutura robótica esteja ausente.

Outro trabalho seria criar um simulador com a capacidade de troca de garra com a seleção entre garras de sucção, magnéticas, ganchos ou outros tipos. Outros tipos de geometria de objetos a serem pegos seriam necessárias caso a garra mude, porém isto também poderia ser estudado no contexto do trabalho. Como se pode perceber, para um tipo de garra gancho, certas geometrias não poderiam ser pegadas (como uma esfera, por exemplo).

Outra idéia é a extensão deste trabalho para preensão de objetos que possuem geometria não uniforme (geometrias mais complexas do que primitivas geométricas). Fazer com que o robô Scorbots possa pegar objetos com geometrias diferentes do que apenas geometrias primitivas aumenta a robustez da funcionalidade de preensão do simulador. Seguindo esta ótica de raciocínio, é possível criar tanto um trabalho que busca a resposta ótima para preensão de objetos não uniformes, como um trabalho que avalia a preensão destes tipos de objetos (semelhante a este, porém com geometrias mais complexas).

Buscado comparar desempenho, um trabalho com o mesmo objetivo deste seria aconselhável, sendo que a preensão seria avaliada fazendo uso de informações e tratamentos físicos. Com a funcionalidade semelhante e implementada no mesmo simulador, os

Raciocínios Geométricos poderiam ser comparados em desempenho, custo-benefício (mais processamento em troca de mais precisão) e demanda de memória.

Como este trabalho não realiza uma simulação física ao soltar os objetos pegos, estes objetos ficam dispostos exatamente onde eles foram liberados. Isto faz com que o realismo da cena seja consideravelmente afetado. Outro trabalho futuro pode tratar este problema, fazendo com os objetos soltos pela pinça caiam sobre a mesa e realizem uma movimentação semelhante à queda real. Simulação física de quando o objeto cai na mesa, ou de quando o objeto cai sobre outro objeto, representam aspectos que aumentam a imersão no ambiente virtual.

REFERÊNCIAS

CLEMSON, U. **“Robotics Laboratory – Clemson University”**. Disponível em: <<http://www.ece.clemson.edu/crb/labs/RoboticsLab/barrethand.htm>>. Acessado em: 29/06/2008.

DELAURENTIS, K. J. e MAVROIDIS, C. **“Mechanical Design of a Shape Memory Alloy Actuated Prosthetic Hand”**. Submitted to the journal Technology and Health Care. Agosto de 2000. Disponível em: <http://robots.rutgers.edu/papers/rutgers_hand.pdf>. Acessado em: 03/06/2007.

DLR, Institute of Robotics and Mechatronics. **“DLR-Institut für Robotik und Mechatronik”**. Disponível em: <<http://bilddb.rb.kp.dlr.de/deutsch/Bild.asp?qryIDBilder=188>>. Acessado em: 03/06/2007.

FOLEY, J. D.; DAM, A. van; FEINER, S. K. e HUGHES, J. F. **“Computer Graphics Principles and Practice”**. Second Edition. United States of America, Addison-Wesley, 1995.

GRASPIT. **“Andrew Miller – GraspIt!”**. Disponível em: <<http://www1.cs.columbia.edu/~amiller/graspit>>. Acessado em: 30/04/2007.

GROOVER, M. P.; WEISS, M.; NAGEL, R. e ODREY, N. G. **“Robótica Tecnologia e Programação”**. São Paulo, McGraw-Hill, 1989.

LARVA. **“Laboratório de Realidade Virtual Aplicada”**. Disponível em: <http://www.joinville.udesc.br/larva>. Acessado em: 15/03/2007.

LEE, E. **“Parallel-Jaw Grip Design Applet UC Berkeley”**. Disponível em: <http://ford.ieor.berkeley.edu/grip>. Acessado em: 29/06/2008.

LEITE, A. C. **“Controle Híbrido de Força e Visão de um Manipulador Robótico sobre Superfícies Desconhecidas”**. Dissertação de Mestrado, Programa de Pós-graduação de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, p 44-47. Abril, 2005. Disponível em: <http://www.pee.ufrj.br/teses/textocompleto/2005042806.pdf>. Acessado em: 01/06/2007.

MILLER, A. T. **“GraspIt!: A Versatile Simulator for Robotic Grasping”**. Tese de Doutorado, Departamento de Ciência da Computação, Universidade de Columbia, Junho de 2001. Disponível em: <http://www1.cs.columbia.edu/~amiller/thesis.pdf>. Acessado em: 03/06/2007.

MILLER, A. T.; KNOOP, S.; ALLEN, P. K. e CHRISTENSEN H. I. **“Automatic Grasp Planning Using Shape Primitives”**, In *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1824-1829, Setembro de 2003. Disponível em: <http://www1.cs.columbia.edu/~amiller/graspPlanning.pdf>. Acessado em: 17/04/2007.

MÖLLER, T.; HAINES, E. **“Real-Time Rendering”**, Second Edition. Massachusetts, A. K. Peters. 2002.

MORALES, A.; SANZ, P. J.; POBIL, A. del e FAGG, A. H. **“Vision-based three-finger grasp synthesis constrained by hand geometry”**. *Robotics and Autonomus Systems*. 54, p. 496-512, 2006. Disponível em: http://www-symbiotic.cs.ou.edu/papers/2004/morales_grasp_synthesis_ver_1.pdf. Acessado em: 17/04/2007.

MORTENSON, M. E. **“Geometric Modeling”**. Third Edition. New York, Industrial Press Inc., p. 505.

NASA, Johnson Space Center. **“ROBONAUT - Home”**. Disponível em: <http://robonaut.jsc.nasa.gov/status/Mar_Robonaut_Status_02.htm>. Acessado em: 03/06/2007.

PAZOS, F. **“Automação de Sistemas & Robótica”**. Axcel Books. Rio de Janeiro – RJ, p. 384, 2002.

RANCH, R. **“Types of Robots”**. Disponível em: <<http://prime.jsc.nasa.gov/ROV/types.html>>. Acessado em: 03/06/2007.

REDEL, R. e HOUNSELL, M. S. **“Implementação de Simuladores de Robôs com o Uso da Tecnologia de Realidade Virtual”**. In: IV Congresso Brasileiro de Computação, Itajaí – SC, Outubro de 2004. IV CBCOMP, v. 1. p. 398-401, 2004.

REQUICHA, A.A.G. **“Representations for rigid solids Theory, methods and systems”**. In: ACM Computing Surveys, p 437–464, 1980.

ROBOTEC E. **“Scorbot ER-4PC: User’s Manual”**, Rosh Ha’ayin: Israel, 1982.

ROMANO, V. F. **“Robótica Industrial Aplicada na Indústria de Manufatura e de Processos”**. Edgard Blücher Ltda, 2002.

SANTOS, R. G. **“Detecção de Colisão para um Simulador de Robô Manipulador”**. Trabalho de Conclusão do Curso, Curso de Bacharelado em Ciência da Computação, Universidade do Estado de Santa Catarina (UDESC). Novembro, 2007a.

SANTOS, R. G.; HOUNSELL, M. S. e DOROW, A. **“Geração de Cenários Variantes Não Colidentes em um Simulador Robótico”**. Revista Hífen - Volume 31 - Nº 59 - I e II Semestre, ISSN 0103-1155, PUCRS. Uruguaiana, 2007b.

SANTOS, W. E. **“Introdução a Robótica”**. Notas de aula, CEFET. Paraná, Abril de 2006. Disponível em: <<http://www.julio stampa.com.br/~junior/robotica/introducao%20a%20robotica.pdf>>. Acessado em: 10/07/2006.

SCHILLING, R. A. **“Fundamentals of Robotics Analysis and Control”**. Englewood Cliffs Prentice Hall. Cop, p. 425, 1990.

SMITH, G.; LEE, E.; GOLDBERG, K.; BÖHRINGER, K. e CRAIG, J. **“Computing Parallel-Jaw Grips”**. *IEEE International Conference on Robotics and Automation*, Detroit, MI, v. 3, p. 1897-1903, 1999.

SPECK, H. J. **“Avaliação Comparativa das Metodologias Utilizadas em Programas de Modelagem Sólida”**. Dissertação de Mestrado, Programa de Pós Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina. Florianópolis, Junho de 2001. Disponível em: <<http://teses.eps.ufsc.br/defesa/pdf/6704.pdf>>. Acessado em: 01/06/2007.

SUNIL, V.B. e PANDE, S.S. **“WebROBOT: Internet based robotic assembly planning system”**. *Computers in Industry*. Vol. 54 p.191-207. Disponível em: <<http://graco.unb.br/alvares/temp/webassembly.pdf>>. Acessado em: 17/04/2007.

VETORAZZI, C.N. Jr. **“Geração Automática de Programas para um Robô Industrial”**. Dissertação de Mestrado, Departamento de Engenharia de Materiais, Universidade Estadual de Campinas, 1994.

ZWIRTES R. A., **“Cinemática Inversa para controle e abordagem de órgãos terminais de robôs manipuladores”**, Trabalho de Conclusão do Curso, Curso de Bacharelado em Ciência da Computação, UDESC. Novembro de 2004.

ANEXOS

Código da Classe Garra

```
public class Garra {  
  
    public static final double H = 0.25875;  
    public static final double A = 0.099;  
    public static final double L = 0.036;  
    public static final double P = 0.036;  
    public static final double O = 0.1125;  
  
    private double[] pfp;  
    private double[] upg;  
    private double[] apg;  
    private double[] elg;  
  
    public void setPFP(double[] p) {  
        pfp = p;  
    }  
  
    public void setOrientacao(Obb o) {  
        upg = o.getEixoX();  
        apg = o.getEixoY();  
        elg = o.getEixoZ();  
    }  
  
    public void setOrientacao(double[] up, double[] ap, double[] el) {  
        upg = up;  
        apg = ap;  
        elg = el;  
    }  
  
    public double[] getPFP() {  
        return pfp;  
    }  
  
    public double[] getUpg() {  
        return upg;  
    }  
  
    public double[] getApg() {  
        return apg;  
    }  
  
    public double[] getElg() {  
        return elg;  
    }  
}
```

Códigos da Classe TrataPreensao

Constantes:

```
public static final int APROVADO = 1;
public static final int ERRO_BOUNDING_BOX_TRABALHO = -1;
public static final int ERRO_FATIA_TRASEIRA = -2;
public static final int ERRO_TAMANHO_OBJETO_GARRA = -3;
public static final int ERRO_PROXIMIDADE = -4;
public static final int ERRO_ABORDAGEM = -5;
public static final int ERRO_DELTA_APG = -6;
public static final int ERRO_DELTA_UPG = -7;
public static final int ERRO_DELTA_ELG = -8;
```

Método Principal:

```
public static int preensao(ListaQuadrante lista, int id, Garra g, double deltaAb, double
piDeltaGe) {
    char obj;
    //O objeto é uma Esfera
    if(lista.tipoComponente(id).startsWith("class Objetos.E"))
        obj = 'E';
    //O objeto é um Cilindro
    else if (lista.tipoComponente(id).startsWith("class Objetos.C"))
        obj = 'C';
    //O objeto é um Paralelepipedo
    else
        obj = 'P';

    int ret = exclusao(lista, id, obj, g);
    if(ret<0)
        return ret;

    return confirmacao(lista, id, obj, g, deltaAb, piDeltaGe);
}
```

Métodos para a etapa de Exclusão:

```
//Etapa da Exclusão
private static int exclusao(ListaQuadrante lista, int id, char obj, Garra g) {
    if(!boundingBoxTrabalho(lista, id))
        return ERRO_BOUNDING_BOX_TRABALHO;

    if(!fatiaTraseira(lista, id))
        return ERRO_FATIA_TRASEIRA;

    if(!tamanhoObjetoGarra(lista, id, g))
        return ERRO_TAMANHO_OBJETO_GARRA;

    if(!proximidade(lista, id, obj, g))
        return ERRO_PROXIMIDADE;

    return APROVADO;
}

//Teste de Bounding Box de Trabalho
private static boolean boundingBoxTrabalho(ListaQuadrante lista, int id) {
    //Leva-se em consideração a escala do Robô
    if(lista.getObbTreePos(id).getRoot().getCentro()[0]<
13.07*0.075||lista.getObbTreePos(id).getRoot().getCentro()[0]>13.07*0.075)
        return false;
    if(lista.getObbTreePos(id).getRoot().getCentro()[2]<
13.07*0.075||lista.getObbTreePos(id).getRoot().getCentro()[2]>13.07*0.075)
        return false;
    return true;
}

//Teste de Fatia Traseira do Robô
private static boolean fatiaTraseira(ListaQuadrante lista, int id) {
```

```

        double angulo = Math.atan2(lista.getObbTreePos(id).getRoot().getCentro()[2], -
lista.getObbTreePos(id).getRoot().getCentro()[0]);
        //Conversão para ângulos positivos
        if(angulo<0)
            angulo = 2*Math.PI + angulo;
        //Se é necessário um ângulo maior do que o máximo permitido pela base
        if(angulo>(31*Math.PI)/18) //310 graus
            return false;
        return true;
    }

    //Teste de Tamanho do Objeto
    private static boolean tamanhoObjetoGarra(ListaQuadrante lista, int id, Garra g) {

        if(Math.min(lista.getObbTreePos(id).getRoot().getExtensao()[0], Math.min(lista.getObbTre
ePos(id).getRoot().getExtensao()[1], lista.getObbTreePos(id).getRoot().getExtensao()[2]))>Garra
.0)
            return false;
        return true;
    }

    //Proximidade
    private static boolean proximidade(ListaQuadrante lista, int id, char obj, Garra g) {
        double dist;
        double R;

        if(obj == 'E') {
            Esfera e = lista.getComponenteEsfera(id);
            dist = distanciaPontoPonto(e.getObbTree().getRoot().getCentro(), g.getPFP());
            R = e.getRaioEsferaEnv()*e.getRaioEsferaEnv();
        }
        else if(obj == 'C') {
            Cilindro c = lista.getComponenteCilindro(id);
            dist = distanciaPontoPonto(c.getObbTree().getRoot().getCentro(), g.getPFP());
            R = c.getRaioEsferaEnv()*c.getRaioEsferaEnv();
        }
        else {
            Paralelepipedo p = lista.getComponenteParalelepipedo(id);
            dist = distanciaPontoPonto(p.getObbTree().getRoot().getCentro(), g.getPFP());
            R = p.getRaioEsferaEnv()*p.getRaioEsferaEnv();
        }

        //É eliminado caso a distância seja maior do que o raio da esfera envolvente
        if(dist>R)
            return false;

        return true;
    }
}

```

Métodos para a etapa de Confirmação:

```

//Etapa da Confirmação
private static int confirmacao(ListaQuadrante lista, int id, char obj, Garra g, double
deltaAb, double piDeltaGe) {
    int ret = folgaAbordagem(lista, id, obj, g, deltaAb);
    if(ret<0)
        return ret;

    return folgaGeometrica(lista, id, obj, g, piDeltaGe);
}

//Análise de Folga de Abordagem
private static int folgaAbordagem(ListaQuadrante lista, int id, char obj, Garra g, double deltaAb)
{
    //Teste de Folga de Abordagem para a Geometria Cilindro
    if(obj == 'C')
        return folgaAbordagemCilindro(lista.getComponenteCilindro(id), g, deltaAb);
    //Teste de Folga de Abordagem para a Geometria Paralelepípedo
    else if(obj == 'P')
        return
folgaAbordagemParalelepipedo(lista.getComponenteParalelepipedo(id), g, deltaAb);
    //Não é necessário teste de Abordagem para a Geometria Esfera
}

```



```

        return APROVADO;
    }

    //Análise de Folga de Abordagem para Cilindro
    private static int folgaAbordagemCilindro(Cilindro c, Garra g, double deltaAb) {
        //Cálculo do ângulo entre o vetor Up do Cilindro e o vetor de Aproximação da Garra
        double angulo = anguloEntreVetores(c.getObbTree().getRoot().getEixoY(), g.getApg());

        //A garra está pegando o Cilindro por cima
        if(angulo <= deltaAb || Math.PI - angulo <= deltaAb)
            //Não é necessário testes com outros vetores, pois independe o rolamento
            existente na garra para esta situação
            return APROVADO;

        else if(angulo >= Math.PI/2 - deltaAb && angulo <= Math.PI/2 + deltaAb) {
            //Cálculo do ângulo entre o vetor Up do Cilindro e o vetor Up da Garra
            angulo = anguloEntreVetores(g.getUp(), c.getObbTree().getRoot().getEixoY());

            //A garra está segurando o cilindro pelas suas laterais
            if(angulo <= deltaAb || Math.PI - angulo <= deltaAb)
                return APROVADO;
            //A garra está segurando o cilindro pelas suas faces circulares
            else if(angulo >= Math.PI/2 - deltaAb && angulo <= Math.PI/2 + deltaAb)
                return APROVADO;
            //A garra não pode pegar o cilindro
            else
                return ERRO_ABORDAGEM;
        }
        //A garra está pegando o cilindro pelo lado de maneira inclinada
        else {
            //Cálculo do ângulo entre o vetor Up do cilindro e o vetor El da Garra
            angulo = anguloEntreVetores(c.getObbTree().getRoot().getEixoY(), g.getElg());

            //A garra está alinhada com a lateral do cilindro
            if(angulo >= Math.PI/2 - deltaAb && angulo <= Math.PI/2 + deltaAb)
                return APROVADO;
            //A garra não pode pegar o cilindro
            else
                return ERRO_ABORDAGEM;
        }
    }

    //Análise de Folga de Abordagem para Paralelepípedo
    private static int folgaAbordagemParalelepipedo(Parelelepipedo p, Garra g, double deltaAb) {
        double angulo;

        //Verifica se existe paralelismo entre o vetor Eixo Lateral da garra com qualquer vetor
        de orientação do paralelepípedo
        if((angulo =
        anguloEntreVetores(p.getObbTree().getRoot().getEixoX(), g.getElg())) <= deltaAb || Math.PI -
        angulo <= deltaAb)
            return APROVADO;
        else if((angulo =
        anguloEntreVetores(p.getObbTree().getRoot().getEixoY(), g.getElg())) <= deltaAb || Math.PI -
        angulo <= deltaAb)
            return APROVADO;
        else if((angulo =
        anguloEntreVetores(p.getObbTree().getRoot().getEixoZ(), g.getElg())) <= deltaAb || Math.PI -
        angulo <= deltaAb)
            return APROVADO;
        else
            return ERRO_ABORDAGEM;
    }

    //Análise de Folga Geométrica
    private static int folgaGeometrica(ListaQuadrante lista, int id, char obj, Garra g, double
    piDeltaGe) {
        double folgaGeometrica;
        double[] centro;

        if(obj == 'E') {
            Esfera e = lista.getComponenteEsfera(id);

```

```

        folgaGeometrica =
        (Math.min(Garra.A, Math.min(e.getObbTree().getRoot().getExtensao()[0], Math.min(e.getObbTree().g
etRoot().getExtensao()[1], e.getObbTree().getRoot().getExtensao()[2])))/2)*piDeltaGe;
        centro = new double[] {e.getPosicao()[0] +
g.getApp()[0]*(folgaGeometrica/2), e.getPosicao()[1] +
g.getApp()[1]*(folgaGeometrica/2), e.getPosicao()[2] + g.getApp()[2]*(folgaGeometrica/2)};
    }
    else if (obj == 'C') {
        Cilindro c = lista.getComponenteCilindro(id);
        folgaGeometrica =
        (Math.min(Garra.A, Math.min(c.getObbTree().getRoot().getExtensao()[0], Math.min(c.getObbTree().g
etRoot().getExtensao()[1], c.getObbTree().getRoot().getExtensao()[2])))/2)*piDeltaGe;
        centro = new double[] {c.getPosicao()[0] +
g.getApp()[0]*(folgaGeometrica/2), c.getPosicao()[1] +
g.getApp()[1]*(folgaGeometrica/2), c.getPosicao()[2] + g.getApp()[2]*(folgaGeometrica/2)};
    }
    else {
        Paralelepipedo p = lista.getComponenteParalelepipedo(id);
        folgaGeometrica =
        (Math.min(Garra.A, Math.min(p.getObbTree().getRoot().getExtensao()[0], Math.min(p.getObbTree().g
etRoot().getExtensao()[1], p.getObbTree().getRoot().getExtensao()[2])))/2)*piDeltaGe;
        centro = new double[] {p.getPosicao()[0] +
g.getApp()[0]*(folgaGeometrica/2), p.getPosicao()[1] +
g.getApp()[1]*(folgaGeometrica/2), p.getPosicao()[2] + g.getApp()[2]*(folgaGeometrica/2)};
    }
    return testeFolgaGeometrica(centro, g, folgaGeometrica/2);
}

private static int testeFolgaGeometrica(double[] pontoCentralFolga, Garra g, double deltaGe) {
    if(distanciaPontoPlano(pontoCentralFolga, g.getUpg(), g.getPFP())>deltaGe)
        return ERRO_DELTA_UPG;
    if(distanciaPontoPlano(pontoCentralFolga, g.getElg(), g.getPFP())>deltaGe)
        return ERRO_DELTA_ELG;
    if(distanciaPontoPlano(pontoCentralFolga, g.getApp(), g.getPFP())>deltaGe)
        return ERRO_DELTA_APG;
    return APROVADO;
}

```

Métodos auxiliares:

```

//Método que calcula a distância entre dois pontos
private static double distanciaPontoPonto(double[] a, double[] b) {
    return Math.pow(a[0]-b[0],2) + Math.pow(a[1]-b[1],2) + Math.pow(a[2]-b[2],2);
}

//Método que calcula o ângulo entre dois vetores
private static double anguloEntreVetores(double v1[], double v2[]) {
    return Math.acos((v1[0]*v2[0] + v1[1]*v2[1] + v1[2]*v2[2])/(Math.sqrt(v1[0]*v1[0] +
v1[1]*v1[1] + v1[2]*v1[2])*Math.sqrt(v2[0]*v2[0] + v2[1]*v2[1] + v2[2]*v2[2])));
}

//Método que calcula a menor distância entre um ponto e um plano
private static double distanciaPontoPlano(double pontoPlano[], double vetorNormal[], double
ponto[]) {
    double modulo = Math.sqrt(vetorNormal[0]*vetorNormal[0] + vetorNormal[1]*vetorNormal[1]
+ vetorNormal[2]*vetorNormal[2]);
    if(modulo>0)
        return Math.abs(vetorNormal[0]*(ponto[0]-pontoPlano[0]) +
vetorNormal[1]*(ponto[1]-pontoPlano[1]) + vetorNormal[2]*(ponto[2]-pontoPlano[2]))/modulo;
    return 0;
}

```

Plano de Trabalho