MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
Seção de Engenharia de Computação / SE 8

Diego Félix de Almeida
Thiago Arruda Navarro Amaral

Planejamento de Movimento Baseado em Física no Ambiente da *Small Size League*

Rio de Janeiro
2011

INSTITUTO MILITAR DE ENGENHARIA

Praça General Tibúrcio, 80 - Praia Vermelha

Rio de Janeiro - RJ CEP: 22290-270

629.892 Amaral, Thiago A. Navarro
A485 Planejamento de Movimento Baseado em Física no Ambiente da *Small Size League*/
Thiago A. N. Amaral, Diego Félix de Almeida. -
Rio de Janeiro: Instituto Militar de Engenharia, 2011.

74 f.

Projeto de Fim de Curso - Instituto Militar de
Engenharia - Rio de Janeiro, 2011

1 - Engenharia de Computação - Tese e Dissertação
2 - Robôs 3 - Robótica - Simulação Computacional
I - Amaral, Thiago A. Navarro II - Almeida, Diego Félix III - Título.
IV - Instituto Militar de Engenharia.

CDD 629.892

INSTITUTO MILITAR DE ENGENHARIA

DIEGO FÉLIX DE ALMEIDA

THIAGO ARRUDA NAVARRO AMARAL

PLANEJAMENTO DE MOVIMENTO BASEADO EM FÍSICA NO AMBIENTE DA *SMALL SIZE LEAGUE*

Projeto de Fim de Curso sob o título *"Planejamento de Movimento Baseado em Física no Ambiente da Small Size League"*, defendida por Diego Félix de Almeida, Thiago Arruda Navarro Amaral e aprovada em 29 de junho de 2011 no Rio de Janeiro, Estado do Rio de Janeiro, pela banca examinadora constituída pelos professores:

Paulo Fernando Ferreira Rosa - PhD

Julio Cesar Duarte - D. C.

Ricardo Choren - M. C.

Título do projeto:

Planejamento de Movimento Baseado em Física no Ambiente da *Small Size League*

Área de Concentração:

Tecnologias e Sistemas de Computação

Linha de Pesquisa:

Sistemas Autônomos e Inteligentes de Robótica

Sumário

Lista de Figuras

Lista de Algoritmos p. 8

Resumo p. 9

Abstract p. 10

1 Introdução p. 11

1.1 Tema . . . . . p. 11

1.1.1 Planejamento de Movimento . . . . . p. 11

1.1.2 Small Size League . . . . . p. 13

1.2 Objetivo . . . . . p. 14

1.3 Justificativa . . . . . p. 15

1.4 Metodologia . . . . . p. 15

1.5 Estrutura . . . . . p. 17

2 Modelo e Algoritmo Básico de Planejamento Baseado em Física p. 18

2.1 Definição . . . . . p. 18

2.1.1 Corpos Rígidos e Parâmetros . . . . . p. 19

2.1.2 Estados, Ações, Domínios . . . . . p. 20

2.1.3 Transições de Estados . . . . . p. 20

2.1.4 Restrições e Definição do Problema de Planejamento . . . . . p. 21

2.2 Tipos de Corpos Rígidos . . . . . p. 21

2.3 Um Algoritmo Básico de Planejamento Baseado em Física . . . . . p. 22

3 Arquitetura STP p. 24

3.1 Habilidades . . . . . p. 25

3.2 Táticas . . . . . p. 26

4 Planejamento Eficiente Usando os Conceitos de Táticas e Habilidades p. 29

4.1 Habilidades e Táticas em Espaço de Estados e em Domínios . . . . . p. 29

4.2 Exemplo de Planejamento Baseado em Táticas . . . . . p. 30

5 Conclusão p. 32

6 Cronograma p. 33

Referências p. 34

Lista de Figuras

1 Problema semelhante ao "Problema do Movedor de Piano"em que se deseja mover um objeto de um ponto a outro em um ambiente CAD (url11a). . . . . p. 12

2 Cenário do simulador utilizado neste trabalho para previsão futura do sistema. . . . . p. 12

3 Partida da categoria *Small Size Robot League* da equipe RoboIME durante a CBR2011 . . . . . p. 14

4 Diagrama de comunicação entre os módulos do software da equipe RoboIME. . . . . p. 16

5 Diagrama de comunicação entre os módulos do software da equipe RoboIME integrado com o planejador. . . . . p. 16

6 Plataforma da categoria *Small Size League* da equipe RoboIME. . . . . p. 17

7 Esquema que representa o funcionamento do Simulador (AdA11) para determinar o próximo estado. . . . . p. 21

8 Diagrama dos tipos de corpos rígidos. . . . . p. 22

9 Esquema da Figura 7 redefinido para inclusão da caixa preta Controlador. p. 25

10 Exemplo de uma Tática do jogador atacante. . . . . p. 28

11 Aplicação do algoritmo de planejamento para o autômato (a). . . . . p. 30

12 Cronograma do Trabalho . . . . . p. 33

Lista de Algoritmos

1 Planejador Básico . . . . . p. 23

2 Algoritmo da Habilidade *GoTo* . . . . . p. 27

Resumo

Neste trabalho, é apresentada uma solução física para o problema de planejamento de movimento, baseado em (Zic10), no domínio da *Small Size League*. Na robótica, o problema de planejamento de movimento é usado para detalhar uma tarefa em movimentos discretos. Conceitos como *Habilidade, Tática e Jogada* serão abordados para criar uma base formal para o entendimento do algoritmo de planejamento de movimento, *Behavioral Kinodynamic Balanced Growth Trees (BK-BGT)*. A implementação desse algoritmo no ambiente da *Small Size League* será o foco principal.

Abstract

It is presented in this work a physics solution for a motion planning problem, based on (Zic10), in Small Size League environment. The motion planning problem is used in robotics for the process detailing a task into discrete motions. Concepts like Skill, Tactic e Play will be shown to create a formal-based for the understanding of the motion planning algorithms, Behavioral Kinodynamic Balanced Growth Trees (BK-BGT). The algorithms implementation in Small Size League environment will the main focus of this work.

1 Introdução

1.1 Tema

1.1.1 Planejamento de Movimento

Uma problema existente no contexto da robótica é a conversão de ações em alto nível para ações em baixo nível na qual um determinado robô deve cumprir, por exemplo, pode-se desejar que um robô conduza determinado objeto a uma posição e orientação assim como o mesmo atinja uma determinada posição e orientação, porém não se sabe quais ações em baixo nível (velocidades ou acelerações dos atuadores) devem ser tomadas para se atingir essas ações em alto nível. Os termos planejamento de movimento e planejamento de trajetória são termos utilizados para esse tipo de problema. O planejamento de trajetória, normalmente, leva em consideração as limitações mecânicas do robô.

Um problema clássico de planejamento de movimento é o "*Piano Mover's Problem*", de forma semelhante ao da Figura 1, ele consiste em mover um piano em um ambiente CAD (*Computer-aided design*) de um determinado cômodo da casa para outro sem colidir com os obstáculos. Os problemas de planejamento de movimento, normalmente, ignoram a dinâmica dos corpos assim como o "Problema do Movedor de Piano"que foca apenas nos movimentos de rotação e translação para atingir o objetivo. Esse trabalho trata de problemas de planejamento de movimento em ambientes complexos, nos quais existem diversos possíveis objetivos para se atingir, além de corpos passivos que devem ser manipulados e restrições nos estados intermediários dos percursos dos corpos, assim como corpos dinâmicos com diversos graus de controle e previsibilidade.

Podem ser identificados duas classes de problemas existentes no domínio da *Small Size League*, aplicação abordada nesse trabalho. A primeira classe envolve os problemas de navegação que consistem em fazer com que um corpo ativamente controlado (descrito na Figura 8) consiga atingir um determinado estado desejado (posição, orientação, velocidade linear e velocidade angular), um exemplo desses problemas seria a navegação de um

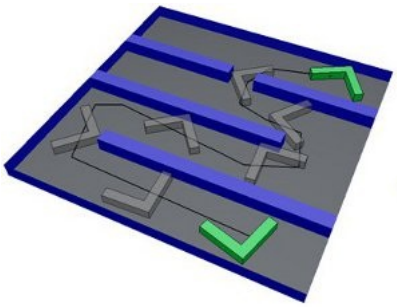

Figura 1: Problema semelhante ao "Problema do Movedor de Piano"em que se deseja mover um objeto de um ponto a outro em um ambiente CAD (url11a).

robô para determinada posição no campo. A segunda classe envolve os problemas de manipulação que consistem em manipular corpos passivos (descrito na Figura 8) com a finalidade de atingir determinado estado, um exemplo desses problemas seria fazer com que um robô manipulasse a bola para dentro do gol.

A solução abordada nesse trabalho para o problema do planejamento de movimento é baseada em física, assim como em (Zic10). Dessa forma, com a utilização do simulador desenvolvido no trabalho (AdA11), é possível ter uma função de transição de estado do sistema que permite uma previsão futura do sistema levando em consideração as complexas interações físicas entre os corpos do sistema. Esse simulador modela corpos rígidos sendo alguns corpos dinâmicos e outros estáticos. Na Figura 2 está representado o cenário do simulador, no qual é possível visualizar os robôs, a bola e o campo.

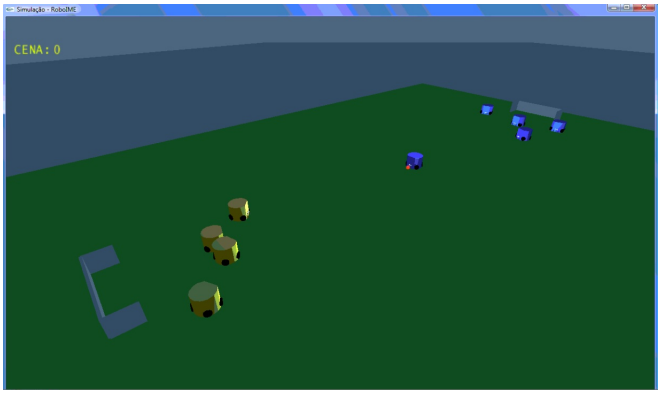

Figura 2: Cenário do simulador utilizado neste trabalho para previsão futura do sistema.

Com a intenção de diminuir o espaço de busca das ações que cada corpo poderia realizar, já que se for considerado todas as possíveis forças e torques que poderiam ser aplicadas em um determinado corpo tornaria a solução inviável, foi adotado a arquitetura STP descrita no capítulo 3. Dessa forma, o espaço de ações é restrito ficando em função de um conjunto de Táticas e um conjunto de Habilidades. É utilizado nesse trabalho o

algoritmo de planejamento *Behavioral Kinodynamic Balanced Growth Trees* (BK-BGT) baseado em Habilidades e Táticas.

1.1.2 Small Size League

Cooperação é uma relação de ajuda mútua entre indivíduos e/ou entidades, no sentido de alcançar objetivos comuns, utilizando métodos mais ou menos consensuais. Projetar robôs para trabalharem juntos não é uma tarefa trivial. A tarefa se torna mais difícil ainda quando os robôs devem ser autônomos. A aplicação escolhida para ser investigada é o futebol de robôs. Nos trabalhos em que o domínio da aplicação é o futebol de robôs, é comum a ideia de se distribuir papéis, dinamicamente, entre os membros da equipe. Este tipo de modelo em um ambiente cooperativo necessita de que todos os membros tenham características em comum, não havendo especialização entre eles. Adicionalmente, é comum a adoção de objetivos globais e locais em trabalhos robóticos cooperativos.

A ideia de robôs jogando futebol foi mencionada pela primeira vez pelo professor Alan Mackworth (*University of British Columbia*, Canadá) em um artigo intitulado "*On Seeing Robots*", apresentado no *Vision Interface 92* e posteriormente publicado em um livro chamado *Computer Vision: System, Theory, and Applications*. Independentemente, um grupo de pesquisadores japoneses organizou um *Workshop* no *Grand Challange in Artificial Intelligence*, em Outubro de 1992, Tóquio, discutindo e propondo problemas que representavam grandes desafios. Esse *Workshop* os levou a sérias discussões sobre usar um jogo de futebol para promover ciência e tecnologia. Estudos foram feitos para analisar a viabilidade prática dessa ideia. Os resultados desse estudo mostraram que a ideia era viável, desejável e englobava diversas aplicações práticas. Em 1993 um grupo de pesquisadores incluindo Minoru Asada, Yasuo Kuniyoshi e Hiroaki Kitano, lançaram uma competição robótica chamada de *Robot J-League* (fazendo uma analogia à *J-League*, que é o nome da Liga Japonesa de Futebol Profissional). Em um mês vários pesquisadores já se pronunciavam dizendo que a iniciativa devia ser estendida ao âmbito internacional. Surgia, então, a *Robot World Cup Initiative* (RoboCup).

RoboCup é uma competição destinada a desenvolver os estudos na área de robótica e inteligência artificial através de uma competição amigável. Além disso, ela tem como objetivo até 2050, desenvolver uma equipe de robôs humanoides totalmente autônomos capazes de derrotar a equipe campeã mundial de futebol humano. A competição possui várias modalidades. Para este trabalho o fator custo e a oportunidade de utilização da plataforma da equipe de futebol de robôs do IME (RoboIME) foram relevantes na escolha

da *Small Size Robot League* mais conhecida como F180, na Figura 3 está representado uma partida dessa modalidade. De acordo com as regras da RoboCup F180, as equipes devem ser compostas por no máximo 5 robôs, sendo um deles o goleiro. O goleiro deve ser designado antes do início do jogo. Durante o jogo nenhuma interferência humana é permitida com o sistema de controle dos robôs. As regras da RoboCup F180 permitem a utilização de visão global e controle centralizado dos robôs. O sistema de controle dos robôs geralmente é externo, recebe os dados da câmera localizada acima do campo, processa os dados, determina qual comando deve ser executado em cada robô e envia este comando através de radiofrequência aos robôs. Embora a maioria das equipes utilize visão e controle centralizado, algumas equipes têm utilizado visão e decisão locais.

Figura 3: Partida da categoria *Small Size Robot League* da equipe RoboIME durante a CBR2011

1.2 Objetivo

Este trabalho tem como objetivo desenvolver um planejador de movimento baseado em física para o ambiente da *Small Size League*. Esse planejador utiliza o algoritmo *Behavioral Kinodynamic Balanced Growth Trees* (BK-BGT) baseado em Habilidades e Táticas, componentes da arquitetura STP que é utilizada para controle reativo de robôs. Também é objetivo do trabalho o desenvolvimento dos componentes Habilidades e Táticas que serão utilizados pelo planejamento seguindo as definições especificadas no capítulo 3. Serão realizados testes sobre o planejador tanto no ambiente de simulação (AdA11) quanto na plataforma da *Small Size League* da equipe RoboIME.

1.3 Justificativa

Este trabalho propõe uma melhoria em relação as soluções tradicionais para o problema de planejamento de movimento, pois leva em consideração a dinâmica dos corpos sendo útil em ambientes complexos, nos quais existem diversos possíveis objetivos para se atingir. Além disso, em ambientes com corpos passivos que devem ser manipulados e restrições nos estados intermediários dos percursos dos corpos, assim como corpos dinâmicos com diversos graus de controle e previsibilidade.

Como justificativa para a escolha da aplicação, temos que o futebol de robôs, problema padrão de investigação internacional, reúne grande parte dos desafios presentes em problemas do mundo real a serem resolvidos em tempo real. As soluções encontradas para o futebol de robôs podem ser estendidas, possibilitando o uso da robótica em locais de difícil acesso para humanos, ambientes insalubres e situações de risco de vida iminente, incluindo a exploração espacial.

Há diversas novas áreas de aplicação da robótica, tais como explorações espacial e submarina, navegação em ambientes inóspitos e perigosos, serviço de assistência médica e cirúrgica, além do setor de entretenimento, podem beneficiar-se do uso de sistemas de multi-robôs. Nestes domínios de aplicação, sistemas de multi-robôs deparam-se sempre com tarefas muito difíceis de serem efetuadas por um único robô. Um time de robôs pode prover redundância e contribuir cooperativamente para resolver o problema em questão. Outrossim, eles podem resolver o problema de uma maneira mais confiável, mais rápida e mais econômica, quando comparado com o desempenho de um único robô.

1.4 Metodologia

Inicialmente, será feito um estudo sobre o problema de planejamento de movimento de uma forma geral baseando-se em (LaV06). Em seguida, será estudado a arquitetura STP descrita em (BBBV04). A tese (Zic10) será a base para o trabalho tanto no estudo da solução baseada em física quanto do algoritmo $BK-BGT$. Tendo sido formada a base teórica para o desenvolvimento, primeiramente, serão feitas implementações e adaptações das Habilidades e Táticas no projeto de inteligência da equipe RoboIME (equipe de futebol de robôs do Instituto Militar de Engenharia), pois já existe uma estrutura que permitirá a realização de testes dos algoritmos desenvolvidos neste trabalho sobre a plataforma da equipe. A estrutura do projeto de software da equipe RoboIME pode ser visto na Figura

4 e descrito em (AAJ11).

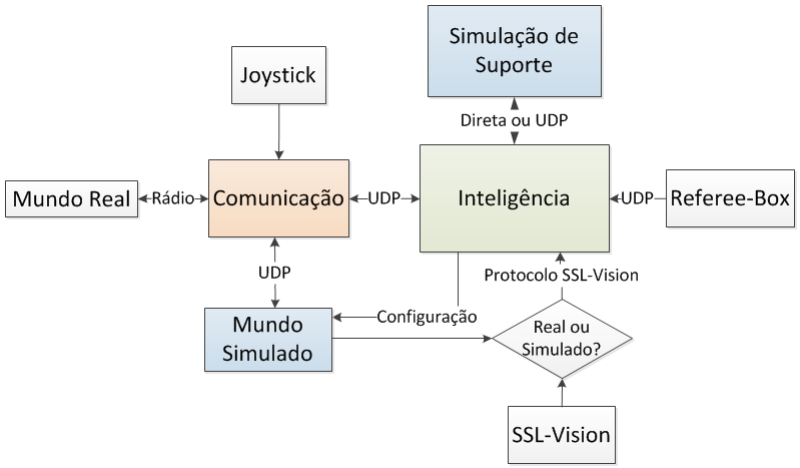

Figura 4: Diagrama de comunicação entre os módulos do software da equipe RoboIME.

O algoritmo de planejamento desenvolvido neste trabalho representará um módulo dentro do software da equipe RoboIME como pode ser visto na Figura 5. Esse módulo de planejamento será utilizado pelo módulo de inteligência na forma de uma biblioteca que permitirá a decisão a cada momento de quais Habilidades e Táticas serão aplicadas sobre os robôs controlados.

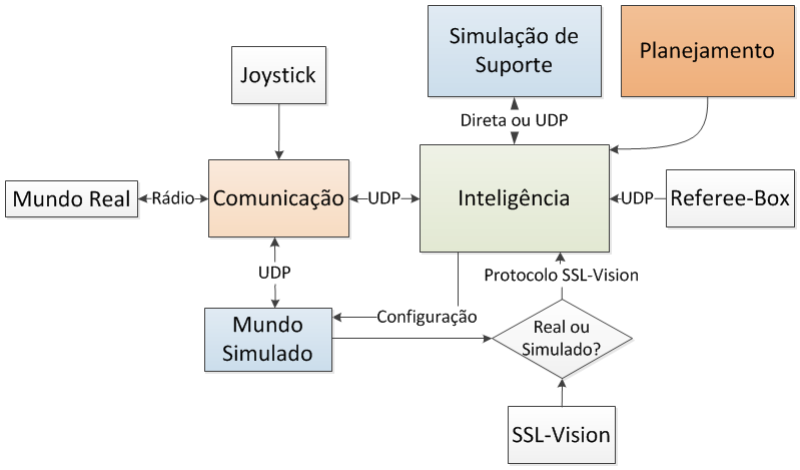

Figura 5: Diagrama de comunicação entre os módulos do software da equipe RoboIME integrado com o planejador.

Para validação do algoritmo de planejamento será utilizado o simulador desenvolvido em (AdA11) que está indicado na Figura 5 como o módulo "Simulação de Suporte"". Serão realizados testes tanto no ambiente de simulação quanto na plataforma da equipe RoboIME que está indicada na Figura 6.

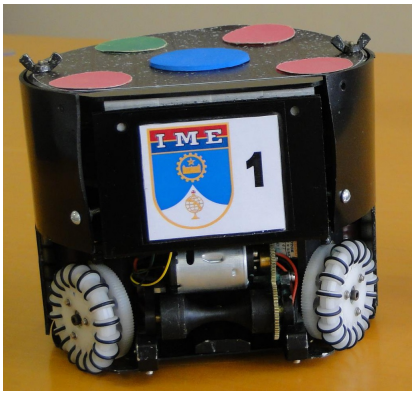

Figura 6: Plataforma da categoria *Small Size League* da equipe RoboIME.

1.5 Estrutura

- **Capítulo 2 - Modelo de Planejamento Baseado em Física e Algoritmo Básico**: Será exposto uma formalização do problema de planejamento baseado em física, que incluem conceitos como Corpo Rígido, Domínio, Espaço de Estados e de Ações, entre outros. Por fim, será exposto o algoritmo básico de planejamento que é a base de todos os algoritmos que será apresentando neste trabalho.

- **Capítulo 3 - Arquitetura STP**: Será dissertado o conceito da arquitetura STP (*Skills, Tactics and Plays*). Além disso, será redefinido o conceito de ação de modo a ficar compatível com a entrada do simulador (AdA11).

- **Capítulo 4 - Planejamento Eficiente Usando os Conceitos de Táticas e Habilidades** : Será abordado como os conceitos de Habilidades e Táticas pode ajudar para resolver o problema de planejamento de movimento. Além disso, um exemplo de planejamento baseado em táticas é apresentado, de modo que se possa entender o contexto de aplicação do algoritmo de planejamento baseado em física, *BK-BGT*.

2 Modelo e Algoritmo Básico de Planejamento Baseado em Física

Este capítulo será definido alguns conceitos importantes que serão usados durante toda a tese, como: estados, ações, função de transição, corpos rígidos. Por fim, será apresento um algoritmo básico de planejamento.

2.1 Definição

O problema geral de planejamento resume-se em encontrar um conjunto de ações partindo de um estado inicial até chegar a um estado final. É possível restringir o problema de modo que o conjunto de estados intermediários para atingir um objetivo (estado final) seja restrito.

Seja X o espaço de estado, $x_{inic} \quad \epsilon \quad X$ o estado inicial, $X_{obj} \subset X$ o conjunto de estados finais, $X_{valid} \subset X$ o conjunto de estados válidos do sistema, A o conjunto de ações. É importante definir também a função de transição e: Dado $a \quad \epsilon \quad A, \quad x \quad \epsilon \quad X, \quad x^{'} \quad \epsilon \quad X \Rightarrow e :< x, a >\rightarrow x^{'}$. Então, em outras palavras, o problema de planejamento é encontrar o conjunto de ações $a_1, a_2, \cdots, a_n$, de modo que o sistema passe pelo conjunto de estados válidos $x_{inic}, x_1, x_2, \cdots, x_n, x_{obj} \quad \epsilon \quad X_{valid}$ e $x_{obj} \quad \epsilon \quad X_{obj}$.

Em um *planejamento de movimento*, os estados representam as configurações físicas dos corpos em um determinado período de tempo, ou seja, representa se o dispositivo de chute ou o sensor de bola estão acionados, se os motores estão em modo de espera (desligados), por exemplo. Já as ações são os possíveis movimentos que podem ser aplicados no espaço de estado definido.

Será definido *planejamento baseado em física*, que é um tipo especial de planejador de movimento que leva em consideração a dinâmica dos corpos para decidir quais os

movimentos necessários para manter o controle do robô e da bola. Segundo (Zic10), é possível responder perguntas que o robô não conseguiria responder, como: quão rápido precisa rotacionar o robô e o dispositivo de drible de modo a não perder o domínio de bola? Qual a força deve ser exercida na bola para a realização de um passe quando ambos os robôs estão em movimento?

2.1.1 Corpos Rígidos e Parâmetros

Seja n corpos rígidos $r_1, \cdots, r_n$ que compõe um sistema de corpos rígidos.

Definition 2.1.1 (Corpo Rígido)**.** Um corpo rígido r é definido por dois conjuntos de parâmetros $r =< \hat{r}, \bar{r} >$, onde:

- $\hat{r}$: parâmetros de estado mutáveis do corpo
- $\bar{r}$: parâmetros de estado imutáveis do corpo

Definition 2.1.2 (Parâmetro Mutáveis do Corpo)**.** Os parâmetros mutáveis do corpo, $\hat{r}$, são parâmetros de um corpo que podem ser manipulados durante o curso do planejamento. $\hat{r} =< \alpha, \beta, \gamma, \omega >$, onde:

- α: posição (vetor 3D),
- β: orientação (matriz de rotação $3x3$)
- γ: velocidade linear (vetor 3D)
- ω: velocidade angular (vetor 3D)

Definition 2.1.3 (Parâmetro Imutáveis do Corpo)**.** Os parâmetros imutáveis do corpo, $\bar{r}$, são parâmetros intrínsecos da natureza do corpo que e permanecem constantes durante o curso do planejamento. $\bar{r} =< \phi_{Shape}, \phi_{Mass}, \phi_{MassC}, \phi_{FricS}, \phi_{FricD}, \phi_{Rest}, \phi_{DampL}, \phi_{DampA} >$, onde:

- ϕ_{Shape}: forma geométrica do corpo,
- ϕ_{Mass}: massa do corpo
- ϕ_{MassC}: vetor posição do centro de massa (referencial do corpo)
- ϕ_{FricS}: coeficiente de atrito estático

- ϕ_{FricD}: coeficiente de atrito dinâmico
- ϕ_{Rest}: coeficiente de elasticidade
- ϕ_{DampL} coeficiente de amortecimento linear
- ϕ_{DampA} coeficiente de amortecimento angular

2.1.2 Estados, Ações, Domínios

Definition 2.1.4 (Estados e Espaço de Estados)**.** O espaço de estados de um planejamento baseado em física X é definido pelos parâmetros mutáveis de n corpos rígidos e tempo t. Se $x \quad \epsilon \quad X$ então $x :=< t, \hat{r}_1, \cdots, \hat{r}_n >$.

Definition 2.1.5 (Ação e Espaço de Ações)**.** O espaço de ações A é um conjunto de ações onde o planejador baseado em física pode pesquisar. Uma ação $a \epsilon A$ é definida como um vetor de ações $a =< a_1, \cdots, a_n >$, onde a_i representa um par $\hat{a}_i =<$ força, torque $>$, onde *força* e *torque* são vetores 3D, aplicados em um corpo rígido r_i.

Definition 2.1.6 (Domínio)**.** Um domínio de planejamento baseado em física d é definido como uma tupla $d =< G, \bar{r}_1 \cdots \bar{r}_n, A >$, onde:

- G: vetor força gravitacional global
- $\bar{r}_1 \cdots \bar{r}_n$: parâmetros imutáveis de n corpos rígidos
- A: conjunto de ações que podem ser aplicadas no sistema

2.1.3 Transições de Estados

A transição de estado é baseada na dinâmica de corpos rígidos. Em (AdA11) foi desenvolvido um Simulador que é utilizado nesse trabalho para simular a dinâmica de um corpo baseado nas leis físicas. Com isso, dado um corpo rígido e um conjunto de ações (*força* e *torque*) e um estado inicial, o Simulador consegue avançar à novos estados em função do tempo. Nesta tese, a simulação é tratada como uma caixa preta (figura 7).

Definition 2.1.7 (Função de Transição de Estado)**.** Seja e a função de transição. Então, $e :< \hat{r}_1, \cdots, \hat{r}_n, a, G, \bar{r}_1, \cdots, \bar{r}_n, \Delta t > \rightarrow < \hat{r}'_1, \cdots, \hat{r}'_n, L >$. Todos os parâmetros já foram definidos anteriormente, exceto L que é uma lista de colisões $L =< l_1, l_2, \cdots >$ que ocorre durante a simulação e Δt que é o período de tempo entre dois passos da simulação.

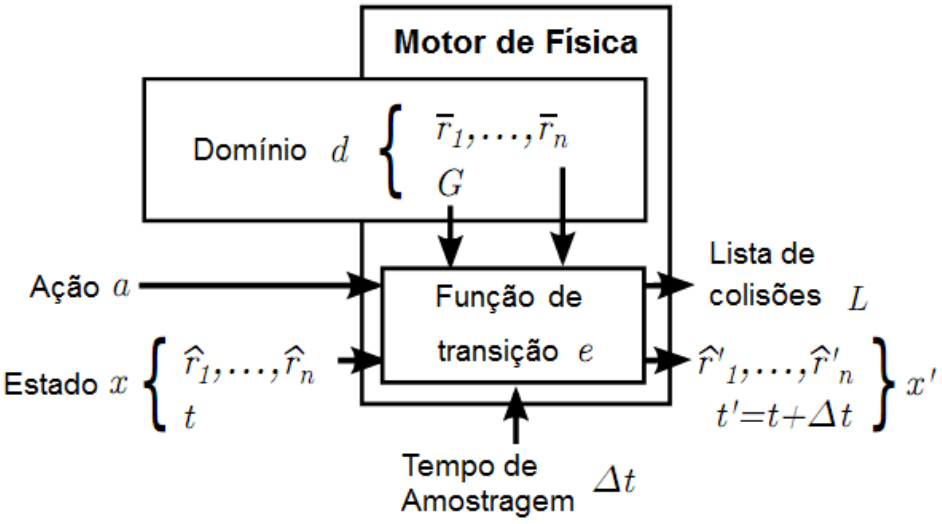

Figura 7: Esquema que representa o funcionamento do Simulador (AdA11) para determinar o próximo estado.

2.1.4 Restrições e Definição do Problema de Planejamento

O objetivo do planejador é encontrar um conjunto de ações, que partindo de um x_{inic} e usando a função de transição, possa chegar em um $x_{obj} \quad \epsilon \quad X_{obj}$. Durante o processo, o sistema passará por alguns estados intermediários. Esses estados, em determinados casos, podem ser inválidos para a aplicação em domínio. Por exemplo, suponha que durante a execução de uma trajetória, um corpo colida com um obstáculo, que não é permitido. Em outras palavras, o conjunto de ações que foi escolhido para realizar aquela trajetória conduziu o sistema para um estado inválido. Portanto, para evitar essa situação, pode-se definir uma função que determina se o próximo estado é válido ou não, a partir da lista de colisão L.

Definition 2.1.8 (Função de Validação)**.** $V(x^{'}, L) = \begin{cases} true & \text{if} \quad x^{'} \text{ is valid,} \\ false & \text{if} \quad x^{'} \text{ is invalid.} \end{cases}$

Definition 2.1.9 (Problema)**.** Com um domínio d, pode-se definir diversos problema de planejamento. Para definir um problema específico é necessário, além de um domínio d, uma função de validação de estados,$V(\cdot)$, um estado inicial, x_{inic}, e um conjunto de estados finais, X_{obj}.

2.2 Tipos de Corpos Rígidos

Observe a figura 8. Todo corpo é definido como corpo rígido, que pode ser dividido em duas subclasses: *dinâmico* e *estático*. Este se caracteriza por não executar nenhum tipo de

movimento, por exemplo, as paredes do sistema ou um corpo de massa grande. Já aquele exibe uma reação ao ser aplicado uma força sobre o mesmo. Dentre os corpos dinâmicos, têm-se os internamente controlados, onde o planejador pode exercer alguma ação sobre eles; os passivos, que são corpos que só se movimentam se existir alguma influência externa; os externamente controlados, que são controlados, mas não pelo planejador. No domínio da *Small Size League*, os corpos ativamente controlados seriam os nossos jogadores, a bola seria um corpo passivo, o gol seria um corpo estático, e os adversários seriam corpos externamente controlados.

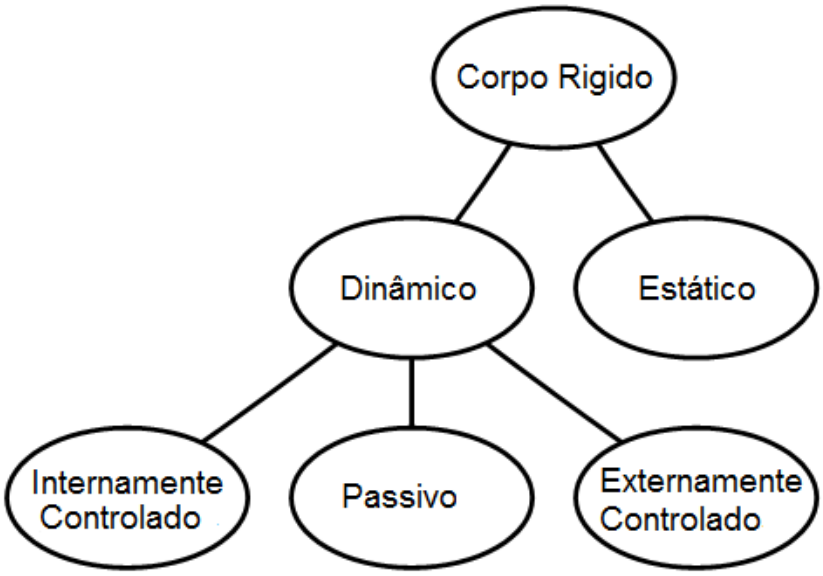

Figura 8: Diagrama dos tipos de corpos rígidos.

2.3 Um Algoritmo Básico de Planejamento Baseado em Física

O algoritmo 1 começa criando uma árvore e adicionando o nó $x_{inic} \quad \epsilon \quad X$ na raiz. Depois é feita iterações até um valor pré-determinado, z. Caso esse valor seja atingido, o planejador falha. Para cada iteração, um nó é selecionado através da função *Select-SourceNode(tree)*. A escolha do nó será abordada no capítulo 4. O mesmo acontece com a função *GenerateAction* que escolhe uma ação no espaço de ações A, que será abordada no capítulo 4. Segundo (Zic10), a escolha de como é gerada uma ação influencia significativamente na performance do planejador. Em seguida, é calculado o próximo estado do sistema com auxílio da função de transição e. Uma verificação é feita antes que novo nó calculado seja inserido na árvore. Para isso, é usada a função de validação, V. Se o novo estado for válido, então é inserido na árvore. Em seguida, é criada uma aresta

entre o nó que foi escolhido pela função $SelecionarNoFonte(arvore)$ e o nó que acabou de ser inserido. Observe que a aresta guarda a informação da ação que foi realizada para chegar nesse novo estado. Por fim, é verificado se o novo nó inserido, que representa um estado do sistema, pertence ao conjunto X_{obj}. Se sim, então foi possível atingir o objetivo pré-determinado e o algoritmo retorna o caminho que deve ser feito para chegar até o estado final.

Algoritmo 1: Planejador Básico

Entrada: Domínio: d, estado inicial: x_{inic}, conjunto de estados finais: X_{obj}, tempo de amostragem: Δt, função de validação: $V(\cdot)$, número máximo de iteração: z

Saída:

```
1  início
2      arvore ← NovaArvore();
3      arvore.AdicionarNo(x_inic);
4      for iter ← 1 to z do
5          x ← SelecionarNoFonte(arvore);
6          a ← GerarAcao(x, d, A);
7          < r̂'_1, ··· , r̂'_n, L > ← e(x. < r̂_1 ··· , r̂_n >, a, d.G, d. < r̄_1, ··· , r̄_n, Δt);
8          t' ← x.t + Δt;
9          x' ← < t', r̂'_1, ··· , r̂'_n >;
10         se V(x', L) então
11             arvore.AdicionarNo(x');
12             arvore.AdicionarAresta(x, x', a);
13             se x ϵ X_obj então
14                 return Caminho(x', tree);
15             fim
16         fim
17     end
18     return Falhou;
19 fim
```

3 Arquitetura STP

STP é uma arquitetura de controle reativa para robôs, a sigla quer dizer *Skills*, *Tactics* e *Plays* que são os componentes da arquitetura, neste trabalho serão utilizados os termos, respectivamente, equivalentes: Habilidades, Táticas e Jogadas. Os componentes Habilidades e Táticas serão descritos a seguir, mas de forma resumida as Habilidades funcionam como rotinas que fornecem capacidades básicas ao robô, como por exemplo atingir uma determinada posição, já as Táticas funcionam de forma semelhante a uma máquina de estados finita na qual os estados seriam as Habilidades, um exemplo de Tática seria a máquina que descreve o comportamento do jogador atacante. O componente Jogada determina quais Táticas serão realizadas por cada robô em uma equipe de robôs. Nesse trabalho o componente Jogada da arquitetura STP não será utilizado.

A seguir será redefinido o conceito de ação já que o simulador utilizado nesse trabalho ao invés de ter como entrada um par força e torque como na figura 7 ele terá como entrada as velocidades que se deseja que o corpo atinja.

Definition 3.0.1 (Redefinição de Ação)**.** Uma ação $a \epsilon A$ é redefinida como um vetor de ações $a =< a_1, \cdots, a_n >$, onde a_i representa a tupla $\hat{a}_i =< \gamma, \quad \omega >$, onde γ e ω são, respectivamente, vetores velocidade linear e angular que se deseja que o corpo rígido r_i atinja.

Dessa forma o esquema da Figura 7 pode ser redefinido para o esquema da Figura 9. Existe uma caixa preta chamada Controlador dentro do simulador que converte uma ação (velocidade linear e velocidade angular) em força e torque que são aplicados no corpo rígido. O Controlador funciona como um controle em malha fechada (AdA11) que conduz a velocidade linear e velocidade angular do corpo rígido para as velocidades da ação (desejado). O simulador possui embutido a ele uma caixa preta Controlador, pois da mesma forma o Robô (real, não simulado) que será utilizado nos experimentos desse trabalho possui um Controlador em seu *Firmware* (AdA11).

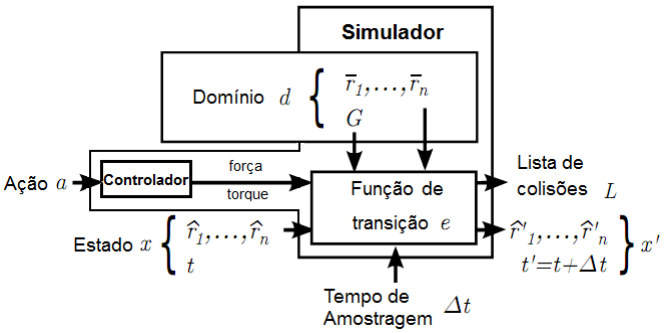

Figura 9: Esquema da Figura 7 redefinido para inclusão da caixa preta Controlador.

3.1 Habilidades

Definition 3.1.1 (Habilidade)**.** Uma Habilidade é defina como a tupla $s =< C, D, f >$, onde:

- C é o conjunto das constantes de configuração da Habilidade s.
- D é um conjunto de amostras de uma distribuição.
- f é o operador da Habilidade s, sendo definido pela função $f :< C, D, V, x, a > \rightarrow < a', V', b >$, onde V e V' são conjuntos de variáveis, a e a' são ações como definido em 3.0.1 e b é um booleano.

Uma Habilidade não determinística possui a característica de sua função f gerar diferentes saídas ($< a', V', b >$) para uma mesma entrada ($< C, V, x, a >$). O conjunto de aleatoriedade D tem a função de prover amostras de uma distribuição, pois em função dessa aleatoriedade a Habilidade será não determinística. Nesse trabalho serão implementadas Habilidades determinísticas e não determinísticas, por exemplo, a Habilidade 2 é não determinística, pois faz uso do conjunto D para gerar a saída.

O conjunto C das constantes de configuração armazena dados específicos de uma Habilidade, como por exemplo, o índice i do corpo rígido na qual aquela Habilidade está sendo aplicada, a velocidade linear máxima que aquele corpo rígido pode atingir a velocidade angular máxima que aquele corpo rígido pode atingir.

Os conjuntos V e V' armazenam dados da mesma forma que C, porém as variáveis em V são armazenadas, externamente, ao escopo da Habilidade, podendo dessa forma manter informações sobre a última iteração do operador f. O booleano b é utilizado para indicar se a Habilidade conseguiu ou não alcançar o objetivo dela, dessa forma quando a

Habilidade já alcançou o objetivo b é falso, caso contrário se a Habilidade estiver ocupada em execução b é verdadeiro.

O algoritmo 2 descreve a Habilidade *GoTo* que permite que um corpo rígido controlado consiga atingir uma posição e orientação amostradas do conjunto D. As rotinas *ControlePosicao* e *ControleOrientacao* implementam, respectivamente, controles proporcionais integrativos para posição e orientação do robô, como descrito em (AdA11). A rotina *PegarAmostra* retorna uma amostra do conjunto passado como parâmetro segundo a distribuição D. As rotinas *EstaPosicionado* e *EstaOrientado* verificam, respectivamente, se o robô conseguiu atingir a posição e a orientação desejadas.

3.2 Táticas

Definition 3.2.1 (Tática)**.** Uma Tática é defina como a tupla $\tau =< S, \Pi, \sigma_{inic} >$, onde:

- S é o conjunto das k Habilidades $< s_1, \ldots, s_k >$ da Tática τ.
- Π é a função $< \sigma, \sigma^{'}, x > \rightarrow p$, onde p é a probabilidade de ocorrer a transição da Habilidade s_σ para a Habilidade $s_{\sigma'}$ na Tática τ estando o sistema no estado x.
- σ_{inic} é a Habilidade, inicialmente, executada na Tática $\tau \in S$.

O conceito de Tática é bem próximo de uma máquina de estados finita não determinística, onde as Habilidades são os estados da máquina. Um diferencia seria que a Tática muda dinamicamente com o tempo, já que a função de probabilidade de transição é dependente do estado do sistema que também é alterado com o tempo.

Definition 3.2.2 (Função de Transição de uma Tática)**.** A Função de Transição de uma Tática é definida como

$$g :< S, \Pi, \sigma, x > \rightarrow \sigma^{'},$$

onde S é o conjunto composto das Habilidades pertencentes a Tática, Π é a função de probabilidade de transição, que fornecerá a probabilidade de transição da Habilidade s_σ para a Habilidade $s_{\sigma'}$.

O autômato da figura 10 representa um exemplo de tática para um atacante. Cada estado significa a habilidade que o jogador pode executar. Para ocorrer uma transição,

Algoritmo 2: Algoritmo da Habilidade *GoTo*

Entrada:
Conjunto D:

- Conjunto das possíveis posições existentes na distribuição: P
- Conjunto das possíveis orientações existentes na distribuição: O

Constantes C:

- Velocidade Linear Máxima do Corpo Rígido: γ_{Max}
- Velocidade Angular Máxima do Corpo Rígido: ω_{Max}
- Índice do corpo rígido controlado: i

Saída: $< a^{'}, V^{'}, b >$

início
  $a^{'} \leftarrow a$;
  $V^{'} \leftarrow V$;
  $i \leftarrow C.i$;
  se $posicao \notin V$ **então**
    $posicao \leftarrow PegarAmostra(D.P)$;
    $V^{'} \leftarrow V^{'} \bigcup \{posicao\}$;
  fim
  se $orientacao \notin V$ **então**
    $orientacao \leftarrow PegarAmostra(D.O)$;
    $V^{'} \leftarrow V^{'} \bigcup \{orientacao\}$;
  fim
  $a^{'}.\hat{a}_i.\gamma \leftarrow ControlePosicao(posicao, C.\gamma_{Max}, x.\hat{r}_i)$;
  $a^{'}.\hat{a}_i.\omega \leftarrow ControleOrientacao(orientacao, C.\omega_{Max}, x.\hat{r}_i)$;
  se $EstaPosicionado(x.\hat{r}_i, posicao)$ **e** $EstaOrientado(x.\hat{r}_i, orientacao)$ **então**
    $b \leftarrow false$ //Objetivo alcançado;
  fim
  senão
    $b \leftarrow true$ //Objetivo não alcançado;
  fim
fim

é preciso satisfazer uma das condições das setas. Os números representados antes das condições servem para indicar a probabilidade de uma transição ocorrer. Por exemplo, o robô começa no estado *Pegar Bola*. Se o jogador estiver com a posse de bola, poderá ir para estado *Girar Bola pro Gol* com probabilidade de 0.8 ou *Girar Bola pro Jogador* com probabilidade de 0.2.

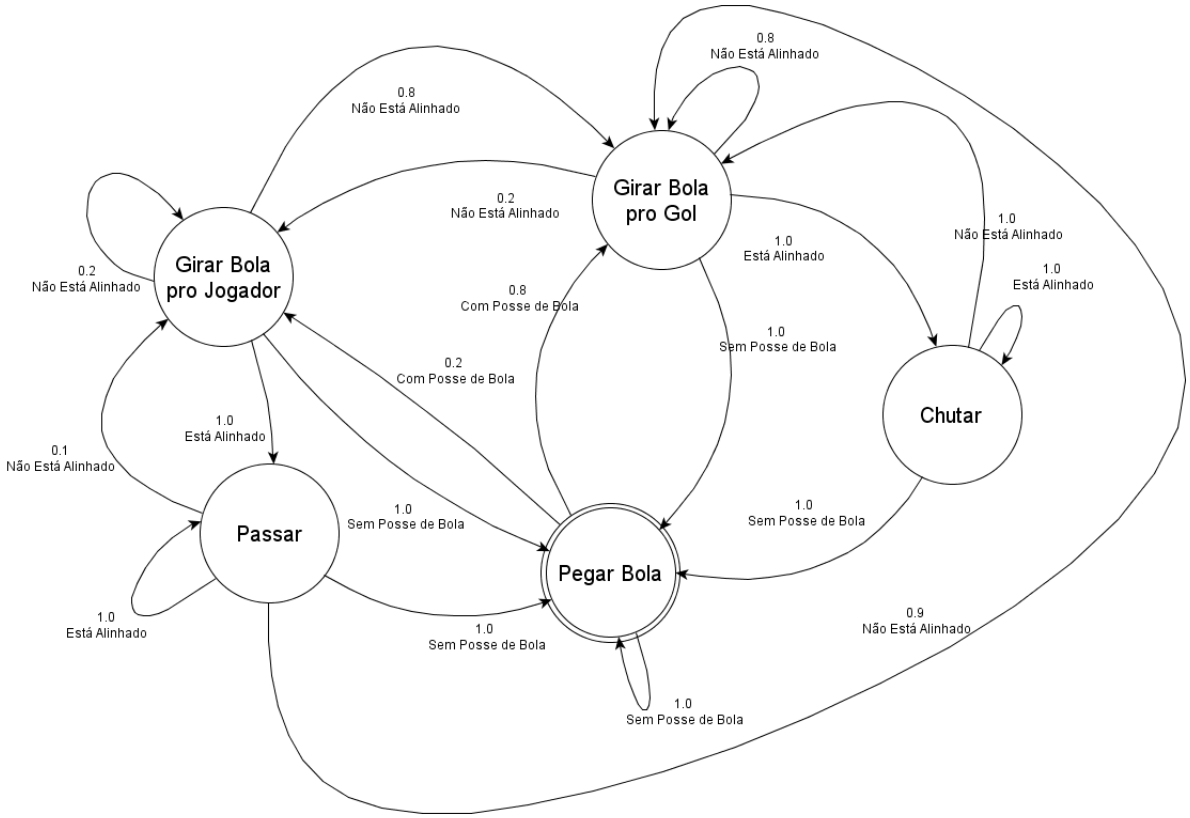

Figura 10: Exemplo de uma Tática do jogador atacante.

4 Planejamento Eficiente Usando os Conceitos de Táticas e Habilidades

No capítulo anterior, foram apresentados alguns conceitos, como o de Tática e de Habilidade. Nesse capítulo abordará algumas definições e mostrar como o esses conceitos são utilizados em um planejamento baseado em táticas.

4.1 Habilidades e Táticas em Espaço de Estados e em Domínios

A definição 2.1.4 embute as variáveis mutáveis em função do tempo. Para utilizar um planejamento baseado em Táticas e Habilidades, é necessário redefinir esse conceito.

Definition 4.1.1 (Espaço de Estados com Táticas)**.** Se $x \quad \epsilon \quad X$, então $x =< t, \hat{r_1}, \cdots, \hat{r_n}, \sigma_1, \cdots, \sigma_h, V_1, \cdots, V_h, b_{AND}, b_1, \cdots, b_h >$ onde:

- σ_i é o índice da i-ésima Habilidade,
- V_i são as variáveis internas da i-ésima Habilidade,
- h é a quantidade de Táticas,
- b_i é uma flag booleana que representa se a i-ésima Habilidade etá endo usada,
- b_{AND} representa a operação lógica AND de todos b_i, que será explicado posteriormente.

Já a definição 2.1.6 havia o espaço de ações, A, que poderia ser aplicado em um corpo rígido. Observe que a informação de qual ação será executada vem da função f da habilidade, que possui uma das entradas $a \quad \epsilon \quad A$. Segundo (Zic10), para aplicar o conceito de Habilidade e Táticas, convém redefinir o conceito de domínio.

Definition 4.1.2 (Domínio Baseado em Táticas)**.** Um domínio baseado em Táticas d é definido: $d =< \hat{r_1}, \cdots, \hat{r_n}, \tau_1, \cdots, \tau_h >$

4.2 Exemplo de Planejamento Baseado em Táticas

Observe a figura 11 (Zic10). O autômato (a) representa a Tática de um jogador, que será usada pelo planejador. Cada estado do autômato representa a Habilidade que o jogador pode executar. Lembre que a função de transição g computa a próxima Habilidade a ser executada.

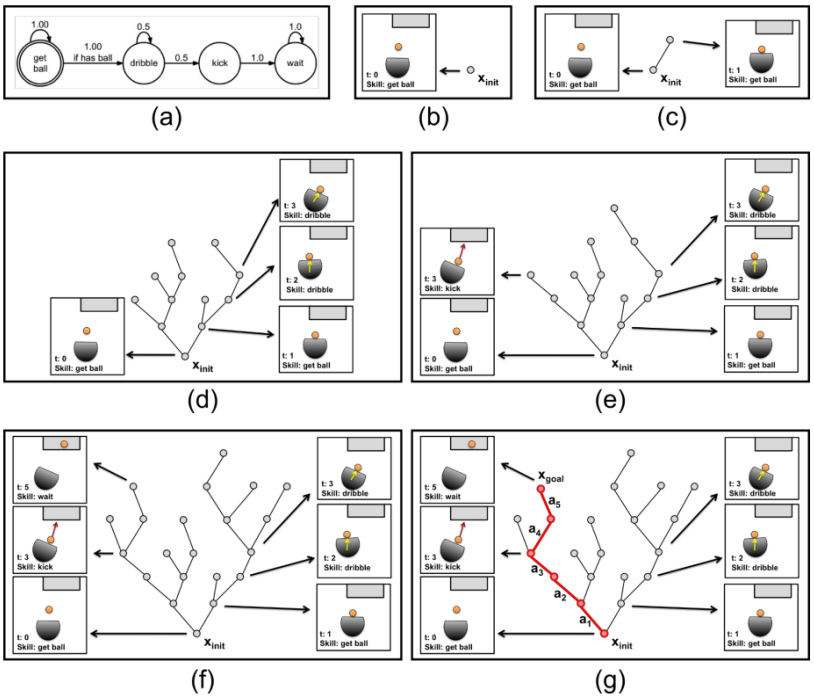

Figura 11: Aplicação do algoritmo de planejamento para o autômato (a).

O algoritmo de planejamento baseado em Táticas é semelhante ao algoritmo 1. Ao invés de escolher uma ação do espaço de ações A, a função f da Habilidade que está ativada é que calcula a próxima ação a ser executada pelo corpo rígido. Vale lembrar também que esse resultado depende do estado atual x do sistema (definição 4.1.1)

O algoritmo começa recebendo um estado inicial x_{inic} como entrada. Esse estado está associado ao tempo zero, $t = 0$. Baseado na Habilidade inicial $s_{\sigma_{init}}$ e no estado inicial x_{inic} é possível calcular a ação que será executada pelo simulador através da função f. A ação calculada é usada no PhysX para computar o estado sucessor ($t = 1$) do sistema com base na função de simulação e. Além disso, do estado inicial x_{inic}, o planejador determina

a próxima Habilidade computada pela função de transição g. Observe que, no exemplo da figura 11 (c), a Tática continua sendo "get ball". Só irá ocorrer uma mudança na Habilidade se o robô tiver a posse de bole (definição da Tática está representada na figura 11 (a)).

O mesmo raciocínio continua, porém é feita outra escolha para o nó fonte. Só ocorre um sucesso quando o planejador consegue encontrar o objetivo x_{obj} (figura 11 (g)). A solução encontrada é o caminho da árvore do estado inicial até o estado objetivo. A maneira de como é escolhido o nó fonte é usando o algoritmo *Behavioral Kinodynamic Balanced Growth Trees (BK-BGT)* que é o foco principal da tese.

5 Conclusão

Foi observado que o problema de planejamento de movimento pode ser resolvido através de uma solução baseada em física aplicada sobre a arquitetura STP. Durante os capítulos, definições como Habilidades, Táticas, Domínios, por exemplo, foram apresentadas com o objetivo de formar uma base formal para o entendimento do algoritmo *BK-BGT*, que será a próxima tarefa a ser realizada. Por fim, foi apresentado um exemplo de planejamento baseado em táticas, que serviu para demonstrar onde o algoritmo *BK-BGT* pode ser usado para melhorar a eficiência do planejador.

6 Cronograma

Para facilitar a compreensão, optou-se por dividir o trabalho em módulos de acordo com o cronograma abaixo (figura 12).

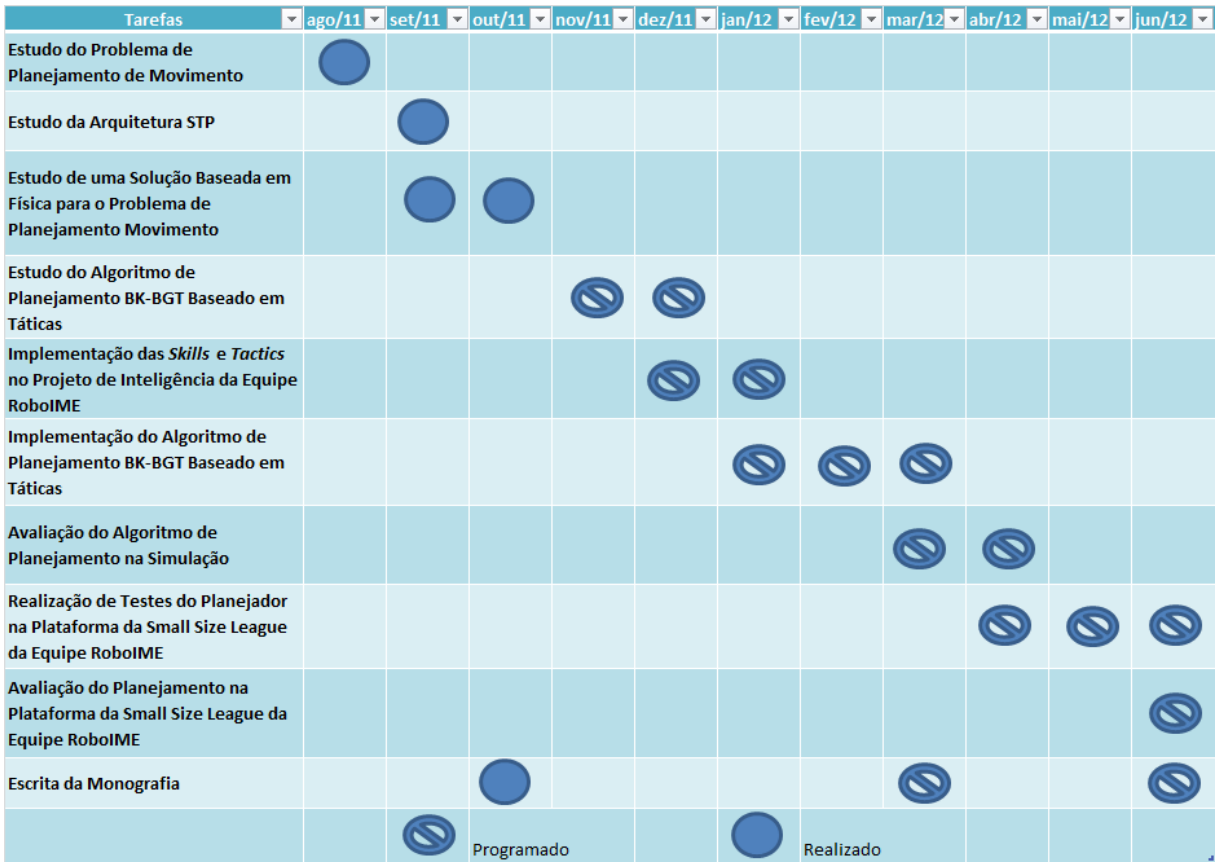

Figura 12: Cronograma do Trabalho

Referências

AAJ11 Diego F. Almeida, Thiago N. A. Amaral, and Erbene C. M. Júnior. *Uma Arquitetura de Hardware e Software para Competições em Robótica Móvel Cooperativa*, Acessado em 11 de Outubro de 2011. `http://www.roboime.com.br/svn/RoboCup/Projeto\%20Computacional/Documentos/CBR2011.pdf`.

AdA11 Thiago A. N. Amaral and Diego F. de Almeida. Robôs autônomos cooperativos: Small size league. IP CDD-629.892-A485, IME, Junho 2011.

BBBV04 B. Browning, J. Bruce, M. Bowling, and M. Veloso. Stp: Skills, tactics and plays for multi-robot control in adversarial environments. Novembro 2004.

LaV06 S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Disponível em http://planning.cs.uiuc.edu/.

url11a *Path planning*, Acessado em 30 de Setembro de 2011. `http://www.v-rep.eu/helpFiles/en/pathPlanningModule.htm`.

url11b *Robocup 2009 - Small Size League*, Acessado em 30 de Setembro de 2011. http://www.robocup2009.org/files/img_08482.jpg.

Zic10 Stefan Zickler. *Physics-Based Robot Motion Planning in Dynamic Multi-Body Environments*. PhD thesis, Maio 2010.