MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

FÁBIO SILVEIRA VIDAL

SISTEMA DE NAVEGAÇÃO PARA DIRIGÍVEIS AÉREOS NÃO-TRIPULADOS BASEADO EM IMAGENS

Rio de Janeiro
2007

INSTITUTO MILITAR DE ENGENHARIA

FÁBIO SILVEIRA VIDAL

SISTEMA DE NAVEGAÇÃO PARA DIRIGÍVEIS AÉREOS NÃO-TRIPULADOS BASEADO EM IMAGENS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Paulo Fernando Ferreira Rosa, Ph.D.

Rio de Janeiro
2007

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

V648s Vidal, F. S.
Sistema de Navegação para Dirigíveis Aéreos Não-Tripulados Baseado em Imagens/ Fábio Silveira Vidal. – Rio de Janeiro: Instituto Militar de Engenharia, 2007.
163 p.: il., tab.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2007.

1. Veículos Aéreos Não-Tripulados. 2. Sistema de Navegação. I. Título. II. Instituto Militar de Engenharia.

CDD 629.892

INSTITUTO MILITAR DE ENGENHARIA

FÁBIO SILVEIRA VIDAL

SISTEMA DE NAVEGAÇÃO PARA DIRIGÍVEIS AÉREOS NÃO-TRIPULADOS BASEADO EM IMAGENS

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Paulo Fernando Ferreira Rosa, Ph.D.

Aprovada em 05 de maio de 2007 pela seguinte Banca Examinadora:

Prof. Paulo Fernando Ferreira Rosa, Ph.D. do IME - Presidente

---

Prof. Maurício Kischinhevsky, D.Sc. da UFF

Profa. Cláudia Marcela Justel, D.Sc. do IME

Rio de Janeiro
2007

Aos meus queridos pais, Elias e Vanda.
À minha amada noiva, Verônica.

AGRADECIMENTOS

Agradeço a todas as pessoas que contribuíram com o desenvolvimento desta dissertação de mestrado, tenha sido por meio de críticas, idéias, apoio, incentivo ou qualquer outra forma de auxílio. Em especial, desejo agradecer às pessoas citadas a seguir.

Aos meus pais, Elias e Vanda, que foram meus pilares para que eu conseguisse concluir esta dissertação e o curso de mestrado.

Ao meu orientador Paulo Fernando Ferreira Rosa, por todo apoio, dedicação e orientações passados durante o período em que estive no Instituto Militar de Engenharia.

Aos companheiros do curso de Mestrado: Alexandre Tadeu Rossini da Silva, Liliana Paola Mamani Sanchez, Marco Antônio Firmino de Sousa, Monael Pinheiro Ribeiro, Rafael Lima de Carvalho, Ricardo Maroquio Bernado e demais colegas que puderam me auxiliar de alguma forma através de debates técnicos.

Por fim, a todos os professores e funcionários da Seção de Engenharia de Sistemas (SE/8) do Instituto Militar de Engenharia.

Fábio Silveira Vidal

É indigno de homens eminentes perder horas como escravos na tarefa desgastante de calcular. Esse trabalho bem poderia ser confiado a pessoas sem qualquer qualificação especial, se máquinas pudessem ser utilizadas

Gottfried Wilhelm Leibniz

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

ABREVIATURAS

ACO	-	*Ant Colony Optimization*
AMACOIA	-	*Approche Multi-Agents pour la Conception d'Installations d'Assemblage*
AMD	-	*Advanced Micro Devices*
AMF	-	*Absolute Map Filter*
ANSI	-	*American National Standard Institute*
API	-	*Application Programming Interface*
AUML	-	*Agent Unified Modeling Language*
AURORA	-	*Autonomous Unmanned Remote Monitoring Robotic Airship*
BCB	-	*Borland C++ Builder*
CenPRA	-	*Centro de Pesquisas Renato Archer*
CG	-	*Centro de Gravidade*
CLSF	-	*Constrained Local Submap Filter*
CV	-	*Centro de Volume*
DANT	-	*Dirigível(is) Aéreo(s) Não-Tripulado(s)*
DGPS	-	*Differencial Global Positioning System*
GPF	-	*Geometric Projection Filter*
GPS	-	*Global Positioning System*
IDE	-	*Integrated Development Environment*
IME	-	*Instituto Militar de Engenharia*
INS	-	*Inertial Navigation Sensor*
MaSE	-	*Multiagent Software Engineering*
MCD	-	*Matriz Cosseno de Direção*
OpenCV	-	*Open Computer Vision Library*
OpenGL	-	*Open Graphics Library*
OplA	-	*Object plus Agent Methodology*
PCM	-	*Planejador e Controlador da Missão*
SDANT	-	*Sistema de Dirigíveis Aéreos Não-Tripulados*
SLAM	-	*Simultaneous localization and mapping*
SMA	-	*Sistemas Multiagentes*

UAV	-	*Unmanned Aerial Vehicles*
UML	-	*Unified Modeling Language*
VANT	-	*Veículo(s) Aéreo(s) Não-Tripulado(s)*
VRML	-	*Virtual Reality Modeling Language*

RESUMO

O uso de VANT (Veículos Aéreos Não-Tripulados) é conveniente para auxiliar a execução de diversas atividades como vigilância, inspeção e reconhecimento de ambientes, devido à posição em que os mesmos atuam e pelo fato de poderem ser de menor porte por não possuírem tripulantes. Sobretudo, a utilização destes veículos torna-se mais interessante se os mesmos possuírem autonomia, ou seja, se forem capazes de realizar tarefas a eles delegadas com o mínimo de intervenção humana. Um dos requisitos para que um VANT seja autônomo é que o mesmo possua um sistema de navegação próprio, o qual possibilite determinar por onde o veículo deverá seguir a fim de alcançar uma determinada posição, previamente estabelecida. Dentre os tipos de VANT, podemos destacar o dirigível como uma opção de baixo custo aquisitivo e operacional, possuindo, ainda, vantagens como dispensar pista de pouso e capacidade de pairar no ar, entre outras. Portanto, é proposto, neste trabalho, o desenvolvimento de um sistema de navegação para dirigíveis aéreos não-tripulados em ambientes fechados baseado em imagens, o qual é feito, essencialmente, em três passos: aquisição das imagens a partir das câmeras embarcadas; mapeamento do ambiente com o uso de visão estereoscópica; e, planejamento de trajetória. Ao longo da dissertação, serão apresentados os conceitos fundamentais da problemática abordada e dos métodos empregados para o desenvolvimento da solução proposta.

ABSTRACT

The UAV (Unmanned Aerial Vehicles) use is convenient for help in the diverse tasks execution as environment monitoring, survey and recognition, due the position that this act and by can to possess smaller size because don't have crew members. Especially, these vehicles use make itself more interesting if the same be autonomous, in other words, if be capable of execute its tasks without human intervention. A requirement for that a UAV be autonomous is to have a navigation system, which make possible to determine the path that the vehicle must to follow to arrive a determined position, previously established. Amongst the some UAV types, can to detach the blimp as a low acquisition and operational cost option, the same have advantages as to excuse landing track and hovering capability, among others. Therefore, propose, in this work, the navigation system development for unmanned aerial blimps based in images, which is made, essentially, in three steps: images acquisition from embedded cameras; environment mapping using stereo vision; and, trajectory planning. In this work, will be presented the problematic boarded and methods used to propose solution development fundamentals concepts.

Os veículos aéreos não-tripulados vêm sendo utilizados em diversas tarefas, tais como: monitoramento, exploração e vigilância de ambientes, e também como apoio a sistemas de comunicação. E podem, ainda, ser utilizados em operações relacionadas a ambientes desconhecidos, ou a locais nos quais encontram-se dificuldades de tramitação ou de acesso (por motivos geográficos ou por se tratar de uma missão militar). Pois, quando devidamente equipados, tais veículos são capazes de extrair dados de um ambiente específico, mesmo estando a uma distância considerável do mesmo. Portanto, são capazes de auxiliar a observação de ocorrências/movimentos, subsidiando a montagem do perfil de um ambiente em questão. Desta forma, VANT (Veículos Aéreos Não-Tripulados) podem ser muito úteis em operações de resgate ou no uso de reconhecimento de ambientes para uma futura missão de alguma operação de uso militar ou civil. E têm o seu potencial aproveitado, de melhor forma, quando são utilizados equipamentos modernos que, no entanto sejam estáveis, duráveis e seguros, pois podem ser utilizados em situações hostis.

Em situações nas quais é necessário que ambientes desconhecidos, ou locais nos quais o homem encontre dificuldades de tramitação ou de acesso, sejam freqüentemente observados ou monitorados, são requeridos meios de observação que consigam registrar as ocorrências/movimentos de forma a gerar dados que venham a subsidiar ou montar perfis de tais ambientes. Este "cenário", a ser monitorado, exige a utilização de equipamentos modernos que, sejam estáveis, duráveis, seguros e, preferivelmente, de baixo custo. Os VANT são uma opção que atende às necessidades levantadas anteriormente, uma vez que os mesmos podem ser utilizados em diversas aplicações como vigilância, monitoramento e inspeção de ambientes e, também, para realizar comunicação de dados em situações de resgate ou em alguma operação de uso militar. Dentre os VANT, o dirigível possui algumas características que o tornam interessante e viável, como:

- baixo custo de aquisição e de manutenção;
- capacidade de pairar no ar;
- ser menos sensível a turbulências;

- possuir um controle menos complexo.
- aterrissar e decolar verticalmente, por isto dispensa pista de pouso e decolagem;
- conseguir voar em baixa velocidade, isto facilita o controle e também as atividades desempenhadas pelo VANT

Além dessas facilidades, como o dirigível é inflado com gás hélio, é bastante difícil que ele venha a cair devido a algum erro no seu controle ou devido a algum evento externo, como colisão ou ataques de forças inimigas. Pois mesmo que o balão de ar do dirigível seja perfurado o gás hélio não vaza do seu recipiente com facilidade; ao contrário do avião e do helicóptero que podem cair facilmente em caso de algum erro de controle ou ataque. Portanto, o dirigível é considerado uma plataforma mais viável para o desenvolvimento deste trabalho.

Como o trabalho não considera apenas um dirigível, mas um grupo de dirigíveis, o ideal é que haja uma cooperação entre os membros do grupo. Faz-se isso com o objetivo de formar um sistema cooperativo de VANT para que haja melhor aproveitamento dos recursos embarcados em cada um deles e diminuir o custo total do sistema. Em um sistema cooperativo, os VANT devem trocar informações. Isto quer dizer que deve existir um sistema de comunicação móvel entre os membros do grupo. Esta comunicação é importante porque, deste modo, é possível que se tenha um conjunto de robôs heterogêneos, ou seja, robôs que possuam equipamentos distintos e, portanto, funcionalidades diferentes. Assim, se um dirigível necessita de dados oriundos de determinado sensor que ele não possua, o mesmo pode solicitar estes dados a um outro membro do grupo que possua o equipamento em questão.

Como, no sistema de navegação baseada em visão, o reconhecimento e a reconstrução do ambiente em 3D devem ser feitos com o uso de duas ou mais imagens que devem ser obtidas de ângulos diferentes, cada dirigível deve possuir duas câmeras embarcadas e ao captar as imagens, a partir das câmeras, deve processá-las e enviar os resultados aos outro(s) membro(s) do grupo. Cada membro, ao receber o resultado do processamento das imagens obtidas por outro(s) membro(s) deve fazer, então, a reconstrução do ambiente em 3D através do uso de visão estereoscópica. Isto possibilita que o veículo tenha uma estimativa de onde estão os obstáculos e a que distância eles se encontram. A partir deste

ponto, é possível realizar o planejamento de trajetória do robô no espaço tridimensional até o destino alvejado.

Considerando-se um amplo ambiente fechado (*indoor*), como um centro de convenções, no qual acontecem vários eventos simultâneos como palestras e *workshops*, onde não seja viável a instalação de muitas câmeras para monitorar o ambiente, podem-se utilizar dirigíveis, que levem câmeras consigo, para captar imagens, esporadicamente, diante da ocorrência de algum evento suspeito; assim, em determinado instante, se necessário, o dirigível deve posicionar-se adequadamente, para que seja possível captar imagens da situação a ser monitorada. Ainda no contexto de um centro de convenções, um conjunto de dirigíveis pode ser utilizado para executar tarefas como fornecer sinal de rede sem fio ou captar informações do ambiente, a partir de um sensor embarcado, deslocando-se para dar cobertura a áreas de maior interesse. Vários fatores podem ser considerados para definir o nível de interesse de cobertura de uma região como a quantidade de pessoas, a realização de um evento específico ou a presença de alguma pessoa importante ou objeto de grande valor. No caso de tarefas de vigilância, pode-se determinar um ponto a ser observado e o dirigível locomover-se em busca de uma posição adequada para captar imagens de tal ponto; esta estratégia pode ser utilizada, sobretudo, em armazéns.

1.1 MOTIVAÇÃO

Os ambientes abertos (*outdoors*) são regiões ao ar livre como florestas, mar aberto ou áreas urbanas. Na maioria dos casos, ambientes semi-estruturados ou desestruturados. Ambientes deste tipo estão sujeitos a interferência de eventos inesperados como chuva, vento, tempestades ou vôo de pássaros, entre outros. Tais interferências dificultam o controle e a navegação de um VANT neste tipo de ambiente.

Os ambientes fechados (*indoors*) são regiões em um espaço físico limitado como armazéns, centros de convenções ou ginásios de esportes, entre outros. Tais ambientes, por serem fechados, estão protegidos da interferência de eventos como chuva ou vento. Assim, o controle e a navegação de um VANT torna-se uma tarefa menos complicada em relação a ambientes abertos. Por este motivo, considera-se um ambiente fechado para o desenvolvimento deste trabalho. Pretende-se que um outro trabalho venha a estender este, por

meio de adaptações do sistema de navegação em ambientes fechados para um sistema de navegação em ambientes abertos.

A maioria dos sistemas de navegação para veículos aéreos, em geral, utiliza, como forma de orientação, GPS (*Global Positioning System*) e INS (*Inertial Navigation Sensor*). O primeiro, é um sistema mantido e controlado pelo Departamento de Defesa dos Estados Unidos da América, e é formado por um sistema de 24 satélites (mais 4 sobressalentes) que circundam o planeta Terra em 6 planos orbitais a uma altitude de, aproximadamente, 20.000 quilômetros; estações terrestres, localizadas em torno da Linha do Equador, que monitoram, continuamente, a constelação de satélites recalculando parâmetros orbitais; e os aparelhos receptores dos usuários, popularmente conhecidos como GPS, que podem ser utilizados em qualquer ponto da terra. O objetivo do sistema consiste em informar a posição, referente às coordenadas geodésicas do sistema WGS84, do aparelho receptor, dada pela latitude, longitude e altitude do mesmo. O cálculo da posição é feito a partir da distância entre o aparelho receptor e 4 satélites, com a precisão de 15m. Com o uso do DGPS (*Differencial Global Positioning System*), um sistema de correção de erros baseado em informações sobre outros satélites, oriundas das estações terrestres, é possível alcançar a precisão de aproximadamente 1m.

O INS é um sistema de navegação inercial que tem, como principal vantagem, o fato de dispensar informações oriundas do meio externo e poder ser utilizado a qualquer instante e altitude. Segundo (SOUSA, 2004) *apud* (MOSTAFA et. al., 2001), o Sistema Inercial fornece a velocidade do avião relativamente ao solo, a sua rota e tempo de vôo e pode ser considerado como o principal sistema de navegação, uma vez que não necessita transmitir nem receber sinal externo e por não ser afetado por perturbações externas. Teoricamente não possui limitações em termos de precisão, fornecendo informações precisas de velocidade, direção e altitude. A navegação aérea baseada no INS é possível para todas as altitudes, mesmo em condições adversas. Podendo ser aplicada para qualquer veículo aéreo e também para navios.

Apesar do amplo, e eficiente, uso do GPS e INS em sistemas de navegação, é vantajoso utilizar um sistema de navegação baseado em imagens, tendo em vista, tornar dispensável o uso do INS ou do GPS e usar câmeras como sensores de navegação. Assim, o sistema

diminui a carga embarcada de cada dirigível. Conseqüentemente, o uso de bateria ou combustível também diminui e o tempo total de trabalho é aumentado. A redução da carga embarcada do dirigível é um fator muito importante, pois o mesmo não possui grande capacidade de embarcação. Como as câmeras também podem ser usadas para realizar tarefas de vigilância, monitoramento e inspeção de ambientes, o uso delas para a navegação consiste em um melhor aproveitamento de recursos do robô e, conseqüentemente, redução de custos.

1.2 OBJETIVOS DO TRABALHO

1.2.1 OBJETIVOS GERAIS

Este trabalho é parte do projeto VANT do Instituto Militar de Engenharia, que tem, como objetivo, construir um sistema de veículos aéreos não-tripulados que possam atuar de forma autônoma em ambientes abertos (*outdoors*) como florestas, mar aberto ou áreas urbanas. Tal sistema deve efetuar tarefas como a vigilância, monitoramento e inspeção de ambientes e comunicação de dados.

Para que o sistema em questão consiga ter autonomia, seus membros devem ter a capacidade de navegar no ambiente em que se encontram, calcular sua própria trajetória e fazer uso adequado de seus equipamentos e sensores para melhor execução de sua missão.

Inicialmente, o projeto VANT foi modelado pelo Ten. Carlos Pinheiro visando um sistema de dirigíveis homogêneos. Nesta modelagem foi utilizada a metodologia, para sistemas multi-agentes, MaSE. Além disso, Pinheiro implementou o sistema de comunicação entre os agentes do sistema.

O projeto VANT visa um sistema cooperativo heterogêneo. Para isso, os membros do grupo devem se comunicar e tomar suas decisões levando em consideração o estado dos outros membros do grupo. Além disso, por tratar-se de um sistema heterogêneo, os membros possuem equipamentos distintos. Assim, um elemento pode usar dados adquiridos por outros em suas tomadas de decisão.

1.2.2 OBJETIVO ESPECÍFICO

Propor um sistema de navegação para dirigíveis aéreos não-tripulados (VANT) em ambiente fechado (*indoors*). Tal sistema deve usar, como parâmetros, imagens captadas a partir de câmeras embarcadas nos veículos para reconhecimento e reconstrução do ambiente em 3D. Além de determinar sua posição e orientação, em relação a um referencial deste ambiente. E que, a partir dos dados obtidos nestas operações, seja capaz de identificar o ponto para onde o veículo deve seguir, ou seja, identificar o seu objetivo. E realizar o planejamento de trajetória do veículo até tal ponto.

1.3 ORGANIZAÇÃO DA DISSERTAÇÃO

Para descrever o trabalho desenvolvido, esta dissertação foi organizada conforme a seqüência dos capítulos definidos a seguir. Após o primeiro capítulo, no qual foi feita uma abordagem introdutória do assunto aqui tratado, temos os seguintes capítulos:

- Capítulo 2 - Revisão Bibliográfica: é feita uma breve descrição de alguns trabalhos relacionados ao tema desta dissertação;
- Capítulo 3 - Fundamentação Teórica: é dada uma abordagem superficial dos conceitos utilizados no desenvolvimento do trabalho como VANT, Sistemas multiagentes, Visão Computacional, SLAM (*Simultaneous Localization and Mapping*) e planejamento de trajetória, dentre outros;
- Capítulo 4 - Modelagem e Implementação do Sistema de Navegação: descreve-se a solução proposta para se atingir os objetivos específicos do trabalho, abordando-se a modelagem do sistema computacional e as metodologias utilizadas da implementação;
- Capítulo 5 - Testes, Simulações e Resultados: neste capítulo são mostrados os resultados obtidos através de testes e simulações aos quais o sistema implementado foi o submetido;
- Capítulo 6 - Considerações Finais: são feitos comentários gerais sobre o desenvolvimento do trabalho e conclusões que relacionam as metodologias utilizadas e os resultados obtidos, além de definir trabalhos futuros.

E, ao final do trabalho, no Capítulo 7, são apresentadas as referências bibliográficas que fundamentaram esta pesquisa.

Neste capítulo, serão apresentados alguns trabalhos relacionados a Veículo Aéreos Não-Tripulados, principalmente no que diz respeito aos sistemas de navegação destes, em especial aos que são baseados no uso de visão computacional para realização do mapeamento e reconstrução tridimensional de ambientes; como é o caso do sistema descrito nesta dissertação.

Um projeto considerado precursor, tratando-se de VANT, é o projeto AURORA (*Autonomous Unmanned Remote Monitoring Robotic Airship*), desenvolvido pelo CenPRA (Centro de Pesquisas Renato Archer) que propôs o uso de um dirigível não-tripulado para realizar, de forma semi-autônoma, tarefas como inspeção, pesquisa e monitoração ambiental, climatológica e de biodiversidade. No âmbito deste projeto foram desenvolvidos diversos trabalhos científicos e a seguir serão citados alguns deles.

Em (ELFES et. al., 1998) encontra-se uma abordagem sobre o projeto AURORA, sendo descritas a plataforma física do projeto, que conta com um dirigível não-tripulado; que é acompanhado, nas operações, por uma estação terrestre móvel; um computador PC 104; e sensores como inclinômetro, câmera e um receptor de sinal de GPS, a bordo. Também é especificada, a arquitetura de softwares e interface homem-máquina; e fornecida uma visão superficial da modelagem dinâmica e sistema de controle, percepção e navegação baseada em visão, e navegação autônoma. Neste trabalho são apresentados argumentos a favor da utilização de dirigíveis, ao invés de aviões ou helicópteros, em tarefas como monitoramento e inspeção ambiental como baixo custo de operação, longa resistência, capacidade de pairar no ar, baixa turbulência, pouso e decolagem verticais e baixo consumo de combustível.

A partir de (GOMES et. al., 1998), pode-se obter um ponto de partida para pesquisas na área de robótica utilizando dirigíveis aéreos; para tanto, (GOMES et. al., 1998) descreve os princípios físicos e o modelo dinâmico de um dirigível YEZ-2A que utiliza um balonete interno para controle da altitude de vôo; descrevendo, também, o vôo do dirigível,

os efeitos da temperatura e as vantagens e desvantagens da operação do dirigível.

Na dissertação de mestrado de (FARIA, 2005) são abordadas três metodologias para identificação do modelo dinâmico do dirigível: a identificação estacionária; a identificação dinâmica e linearizada do veículo a partir de um vôo com entradas de pertubações conhecidas; e a identificação por meio de estratégias evolutivas; tais metodologias foram testadas usando o simulador do dirigível AS800 (FIG. 2.1) do projeto AURORA; e, segundo o autor, as três metodologias, citadas anteriormente, mostraram bons resultados durante os testes. Um outro trabalho, alheio ao projeto AURORA, é o de (HIMA et. al., 2001) que descreve as propriedades cinemáticas e dinâmicas do dirigível AS200, fornecendo equações de força e de momento, usando a abordagem de Newton Euler e considerando o dirigível um corpo rígido, desprezando as deformações sofridas pelo mesmo, durante sua operação; também é descrita uma metodologia de planejamento de trajetória, respeitando os limites físicos do veículo e minimizando o tempo de viagem necessário para executar a trajetória, e a formulação do problema de geração de movimento; ao final são expostos resultados obtidos a partir de simulações.

O trabalho de (RAMOS et. al., 2001) descreve como um dirigível não-tripulado percorre uma trajetória pré-definida, operando o veículo a partir de informações oriundas de sensores como GPS, acelerômetro, sensor de vento e inclinômetro, sem o emprego de técnicas de visão computacional; são descritos testes realizados em 2000 em uma área militar, onde o seguimento da trajetória foi guiado por um sistema embarcado implementado em linguagem C, porém a altitude sendo controlada remotamente de forma manual; além disso, também é definido o modelo dinâmico do dirigível AS800 e descrita toda a plataforma experimental utilizada, na qual consta um computador PC104 embarcado e um simulador desenvolvido em VRML (*Virtual Reality Modeling Language*), em uma estação em terra. Em (RAMOS et. al., 1999) são descritos *softwares* desenvolvidos para suporte ao projeto AURORA; trata-se de um conjunto de ferramentas para projeto e teste de métodos de controle e navegação, visualização do dirigível, compartilhamento de informações via internet e treinamento de pilotos, desenvolvidos com o uso da ferramenta MATLAB/Simulink; além de fazer uma análise de requisitos para ferramentas de suporte ao projeto AURORA, descrever o sistema embarcado instalado na gôndola do dirigível, e ao final é mostrado um estudo de caso que consiste em simular um vôo controlado de

FIG. 2.1: Dirigível AS800 do projeto AURORA (FARIA, 2005)

forma manual e remota, dando um retorno visual para o operador.

O projeto AURORA desenvolveu trabalhos muito relevantes, porém, estes levam em conta apenas um dirigível; no contexto do projeto VANT do IME (Instituto Militar de Engenharia) considera-se um grupo de dirigíveis heterogêneos que compartilham recursos e, com isso, consegue-se diminuir o custo total do sistema; pois, é feita uma distribuição dos equipamentos, permitindo o uso de dirigíveis de menor porte.

O trabalho, aqui descrito, está no contexto do projeto VANT do IME e tem como antecedente a dissertação de mestrado (PINHEIRO, 2006) na qual foi realizada a modelagem de um SDANT (Sistema de Dirigíveis Aéreos Não-Tripulados) usando a abordagem de sistemas multiagentes, definindo os requisitos funcionais e não-funcionais e desenvolvendo os diagramas de casos de uso, seqüências, papéis, tarefas concorrentes, classes, e implantação. Dentre os requisitos do SDANT podemos citar: evitar colisões, corrigir a trajetória e realizar o deslocamento da frota automaticamente. Estas são tarefas inerentes a um sistema de navegação, que é o foco desta dissertação.

Como este trabalho trata de um sistema de navegação para dirigíveis aéreos não-tripulados baseado em imagens, têm-se dois enfoques principais: a percepção e mapeamento do ambiente a partir de imagens digitais; e o planejamento da trajetória a ser percorrida. Os próximos parágrafos tratam de alguns trabalhos que abordam os assuntos supracitados.

O mapeamento do ambiente é parte de um problema conhecido, na bibliografia, como SLAM (*Simultaneous localization and mapping*), o qual consiste em construir um mapa do ambiente de trabalho, e a partir deste mapa, determinar a localização de um robô enquanto o mesmo está em operação; uma abordagem introdutória sobre o problema é apresentada no capítulo 3. Neste parágrafo serão abordados alguns trabalhos sobre este assunto. (DISSANAYAKE et. al., 2001) afirma que é possível um veículo autônomo atuar a partir de uma posição desconhecida em um ambiente desconhecido e apresenta uma solução para o problema em questão que retorna resultados muito precisos. Para tanto, são utilizadas ondas de radar milimétricas e conta com o apoio de marcações geográficas; sua principal contribuição é demonstrar a existência de uma teoria de solução de estimativas não-divergentes para o problema de SLAM, e elucidar a estrutura geral de algoritmos de navegação SLAM. No artigo, é apresentada uma boa abordagem matemática sobre SLAM, porém seria de difícil implementação para uso em sistemas de tempo real embarcados. Já (ELIAZAR et. al., 2003) oferece uma outra solução através de um algoritmo baseado em filtro de partículas sem o uso de marcações geográficas, o qual atende a requisitos de aplicações em tempo real e utiliza uma grade binária para mapear o ambiente, que é uma idéia também utilizada no desenvolvimento desta dissertação. Contudo, na solução descrita em (ELIAZAR et. al., 2003), o sensor utilizado para a detecção de obstáculos baseia-se em laser, que é disparado de uma altura fixa, o que torna o mapeamento bem limitado. O artigo de (MONTEMERLO et. al., 2003) propõe o algoritmo *FastSlam 2.0* para SLAM que também utiliza filtros de partículas. Este algoritmo, por sua vez, é uma modificação do *FastSLAM* (MONTEMERLO et. al., 2002) e traz uma boa abordagem matemática sobre o problema e sua solução também utiliza marcações geográficas; o algoritmo foi testado na navegação aérea de um veículo, comparando os resultados com dados obtidos a partir de um GPS embarcado. A principal contribuição de (MONTEMERLO et. al., 2003) é provar a convergência do *FastSLAM* com uma única

partícula. Apesar de ter bons resultados, os métodos utilizados em (DISSANAYAKE et. al., 2001), (ELIAZAR et. al., 2003), (MONTEMERLO et. al., 2002) e (MONTEMERLO et. al., 2003) não são interessantes para este trabalho, pois, para implementar qualquer um deles, seria necessário o acréscimo de componentes ao sistema, o que acarretaria em aumentar o custo do mesmo e a carga a ser levada a bordo. O artigo de (WILLIAMS et. al., 2002) apresenta uma abordagem original para o SLAM, onde um mapa global do ambiente é feito através de fusões de submapas do ambiente, tais submapas são adquiridos, periodicamente, na vizinhança do veículo através do CLSF (*Constrained Local Submap Filter*), o qual é descrito no trabalho em questão. Os resultados oriundos do CLSF foram comparados com o caso global do SLAM e somente em alguns casos específicos, foram observados melhores resultados, porém, o custo computacional do CLSF é bem mais baixo. A tarefa de mapeamento do ambiente, desta dissertação, é realizada de forma parecida com a de (WILLIAMS et. al., 2002), pois periodicamente serão analisadas imagens adquiridas de câmeras embarcadas, e a partir delas o ambiente será mapeado. Além de realizar o mapeamento de forma iterativa, como tem-se um grupo de dirigíveis, serão utilizadas as imagens captadas de todos os dirigíveis para realizar o mapeamento do espaço de trabalho. Desta forma, a solução para o SLAM é feita de forma cooperativa e possui maior alcance. Um outro sistema de mapeamento que também tem comportamento cooperativo é descrito por (FENWICK et. al, 2002), no qual é apresentado um algoritmo de trabalho cooperativo baseado em uma estimativa estocástica e utiliza uma abordagem baseada em características para extrair marcações geográficas do ambiente; os testes foram feitos de forma simulada e prática, obtendo bons resultados nos dois casos. O diferencial apresentado em (FENWICK et. al, 2002) é a mesclagem entre os dados captados de vários veículos para realizar o mapeamento do ambiente. A tese de doutorado de (NEWMAN, 1999) mostra o desenvolvimento de um algoritmo chamado GPF (*Geometric Projection Filter*) que realiza o mapeamento do ambiente e a localização do robô de forma incremental, porém, para o seu funcionamento, tal algoritmo necessita de marcações geográficas que sejam, previamente, conhecidas; além do GPF, na tese de (NEWMAN, 1999) também foi implementado o algoritmo AMF (*Absolute Map Filter*), um algoritmo convencional e bem conhecido de SLAM. As duas implementações foram submetidas a testes com veículos submarinos e, ao final, o GPF apresentou resultados mais atrativos.

Após a solução do SLAM o próximo passo a ser executado é o planejamento de tra-

jetória. Existem diversos trabalhos, sobre este problema, presentes na bibliografia, dentre eles podemos mencionar os seguintes.

Thrun (THRUN, 1998) desenvolveu um aprendizado de mapeamento híbrido de ambientes fechados, baseado na combinação de grades e mapas topológicos, os quais são obtidos previamente; a grade é obtida por meio da interpretação de sinais de sonar, que por sua vez são interpretados por uma integração de redes neurais artificiais e regras bayesianas; a combinação sugerida por Thrun resultou em ganhos de precisão/consistência bem como eficiência. Em 2001, (SINOPOLI et. al., 2001) estendeu o modelo de (THRUN, 1998) de um ambiente 2D para um ambiente 3D em trabalho de um sistema de navegação, baseado em imagens, para um helicóptero, no qual são utilizados equipamentos como INS e GPS; para a escolha do melhor caminho da trajetória global (SINOPOLI et. al., 2001) usou o algoritmo Djikstra. Em (GOLDBARG et. al., 2005) é encontrada uma abordagem sobre este algoritmo. No trabalho, consegue-se encontrar o caminho ótimo de um ponto ao outro, porém, ele utiliza um algoritmo exato que possui um custo computacional elevado, podendo não ser interessante para aplicações dinâmicas e com restrições de tempo. Em (LEE et. al., 1995) é descrito um método de planejamento de trajetória para dois robôs em um mesmo espaço de trabalho. O método em questão produz trajetórias livres de obstáculos e que não entram em estado de colisão. Leva-se em conta, mudanças na velocidade dos robôs, bem como atrasos no seguimento da trajetória para que seja possível que os dois robôs se movimentem em um mesmo espaço de trabalho, de forma a alcançar seus alvos sem que haja colisões entre eles. O trabalho (JIANG et. al., 2005) descreve um método de navegação híbrida em um ambiente conhecido ou dinâmico para o controle autônomo da miniatura de um robô escalador. Este sistema de navegação calcula uma trajetória ótima a partir de um mapeamento do ambiente, previamente conhecido; durante o seguimento da trajetória, a partir da fusão de dados de multi-sensores, a mesma é adaptada, com uso de um sistema neuro-fuzzy, conforme mudanças no ambiente são detectadas. Testes feitos através de simulações mostram que o método de planejamento de trajetória em questão pode ser utilizado em situações nas quais existam restrições de tempo. Por fim, o trabalho (KIGUCHI et. al., 2001) faz uma abordagem sobre planejamento de trajetórias para robôs móveis usando computação DNA; o ambiente de trabalho é dividido em várias seções ou células; primeiramente, é traçada uma trajetória ótima do ponto de partida até o alvo; após essa etapa, usa-se computação DNA, a qual simula

reações químicas moleculares, para realizar o desvio dos obstáculos que são previamente conhecidos; para testar o método proposto, foram feitas simulações, em um computador convencional, de um ambiente de trabalho bidimensional dividido em 254^2 secções, conseguindo encontrar a trajetória ótima; porém, os algoritmos baseados computação DNA demandam muito tempo para sua execução, tornando-os inviáveis para aplicações de tempo-real.

Este capítulo é dedicado a fazer a descrição dos principais conceitos utilizados no desenvolvimento desta dissertação, para que o leitor consiga melhor entendimento da problemática e da solução que são tratadas neste trabalho.

3.1 VANT

Esta dissertação trata de um sistema de navegação para dirigíveis aéreos não-tripulados, que fazem parte de uma classe de veículos, chamada VANT (Veículos Aéreos Não-Tripulados); nesta classe, também podem estar inseridos outros meios de transporte como aviões ou helicópteros; a TAB. 3.1 fornece uma comparação entre aviões, helicópteros e dirigíveis, considerando-se seus recursos e custos. Estes veículos não necessitam de um piloto a bordo durante a operação, e podem ser autônomos, semi-autônomos ou não-autônomos.

TAB. 3.1: Comparação de veículos aéreos como plataformas para ambiente de pesquisa e monitoramento de missões
(alto = √ √ √, médio = √ √, baixo ou nulo = √) (ELFES et. al., 1998)

Requisitos do Projeto	**Avião**	**Helicóptero**	**Dirigível**
Baixo custo de operação	√ √	√	√ √ √
Alta resistência	√ √	√	√ √ √
Capacidade de pairar no ar	√	√ √ √	√ √ √
Proporção do peso para carga paga	√ √	√	√ √ √
Alta capacidade de manobramento	√ √	√ √ √	√
Baixo ruído e turbulência	√	√	√ √ √
Decolagem e pouso verticais	√	√ √ √	√ √ √
Baixo consumo de combustível	√ √	√	√ √ √
Baixa vibração	√ √	√	√ √ √

Os VANT autônomos são capazes de realizar todo o trabalho a eles atribuídos de forma independente, ou seja, sem a interferência ou comando externo. Devem ter capacidade de percepção do ambiente e de sua localização; para tanto, podem utilizar sensores como sonares, GPS, INS ou câmeras, e um módulo de processamento como o PC 104 ou o SiriuStar embarcados. Também é comum que seja embarcado um rádio, visando

estabelecer a comunicação entre o VANT e uma estação base em terra; assim é possível que a estação base receba informações sobre uma missão enquanto a mesma é executada, bem como enviar mensagens para o VANT sobre mudanças no planejamento da missão. Já os semi-autônomos, são capazes de realizar, somente, uma parte do seu trabalho de forma independente. Por exemplo, um avião que possui autonomia para sobrevoar uma determinada área mas necessita ser tele-operado em suas decolagens e/ou aterrissagens. Por sua vez, os VANT não-autônomos necessitam ser totalmente tele-operados, desde o início até o fim de sua missão, pois não possuem nenhum grau de autonomia.

Tendo em vista as vantagens do dirigível, apresentadas na TAB. 3.1, o veículo citado foi escolhido como plataforma para este trabalho. A seguir é dada uma abordagem sobre o mesmo.

3.1.1 O DIRIGÍVEL

O dirigível é um veículo aéreo formado, basicamente, por um envelope preenchido por gás hélio, o qual é um corpo não-rígido e traz uma gôndola acoplada na parte inferior. O princípio de vôo do dirigível baseia-se no gás hélio contido no envelope, o qual por ser mais leve que o ar, faz com que o dirigível paire no ar. Alguns dirigíveis possuem um balonete no interior do envelope o qual é inflado ou esvaziado durante o vôo a fim de controlar a altitude do veículo. Ao inflar o balonete interno, o dirigível desce, e ao esvaziá-lo, o mesmo sobe. Na gôndola, são acoplados propulsores, os quais podem ser utilizados para deslocamento e rotação, bem como para o controle de altitude. O vôo também pode ser controlado através de lemes, profundores e o motor de popa, localizados na parte traseira. A FIG. 3.1 ilustra os componentes do dirigível. A seguir será descrito o modelo dinâmico de um dirigível, encontrado em (GOMES et. al., 1998).

3.1.1.1 MODELO DINÂMICO

Primeiramente, para facilitar o desenvolvimento de um modelo matemático viável, levou-se em consideração alguns fatores, como as diferenças do modelo de uma aeronave de uso convencional:

- Momento e massa de inércia muito altos;
- A massa do envelope de gás hélio pode aumentar ou diminuir em um curto espaço

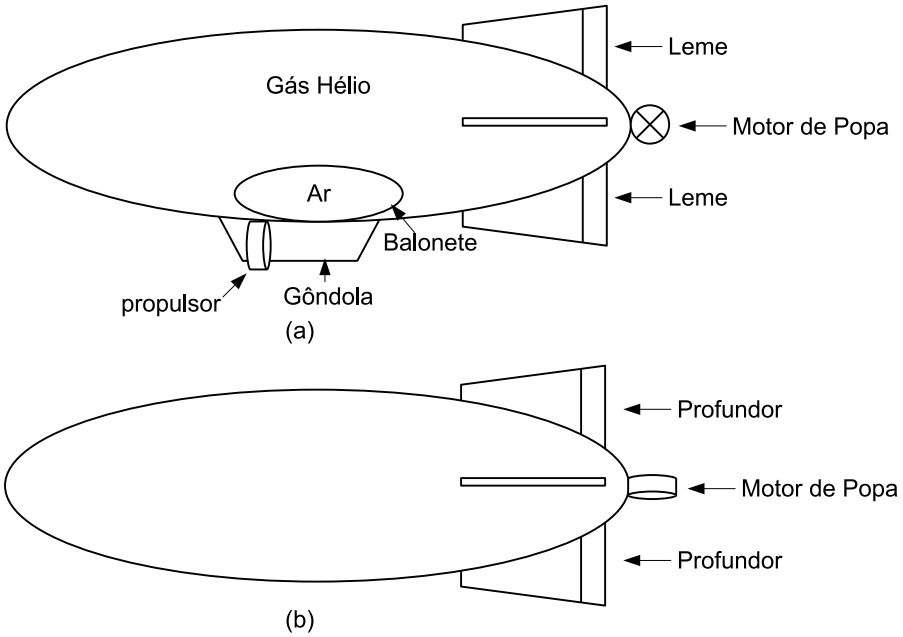


FIG. 3.1: Desenhos do dirigível: (a) visão lateral; (b) visão superior.

de tempo, devido à compressão e descompressão do balonete interno de ar.

- Devido às mudanças constantes da posição do centro de gravidade, uma característica singular do dirigível, os movimentos do mesmo devem ser referenciados a um sistema de coordenadas, fixo ortogonal aos eixos do corpo da aeronave, com origem no centro de volume; como mostra a FIG. 3.2

Por razões práticas, foram feitas duas limitações no desenvolvimento do modelo matemático não linear:

- o dirigível é considerado um corpo rígido, desconsiderando-se os efeitos aeroelásticos;
- a base da aeronave é simétrica sobre o plano XZ.

O modelo dinâmico é dado pela equação:

$$M\dot{x} = F_d(x) + A(x) + G(\lambda_{13}, \lambda_{23}, \lambda_{33}) + P \tag{3.1}$$

onde cada um dos componentes é descrito a seguir.

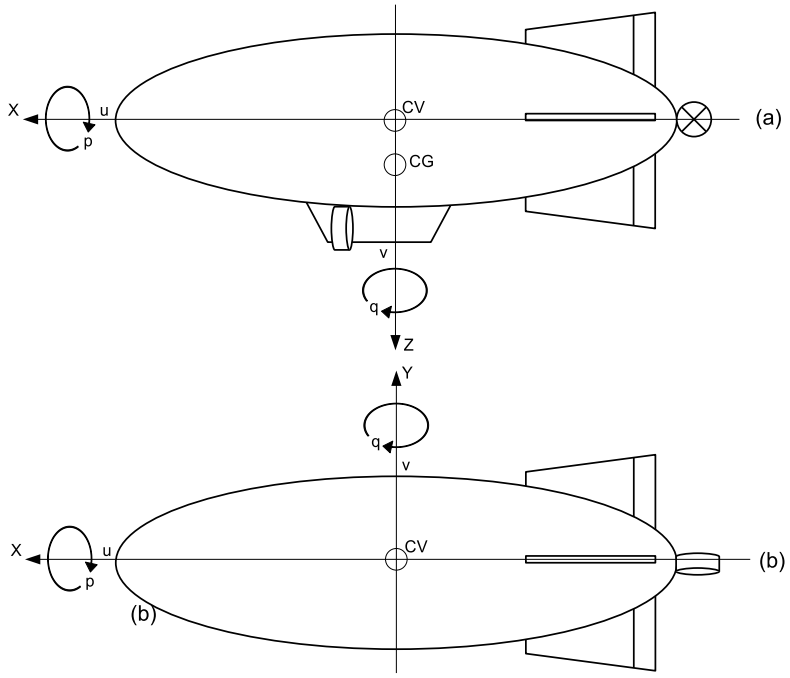


FIG. 3.2: Sistema de coordenadas do dirigível
Mostra-se os centros de volume (CV) e de gravidade (CG); e as velocidades lineares (u, v, w) e angulares (p, q, r) em relação aos eixos X, Y e Z, respectivamente.

O vetor velocidade, x é composto pelas velocidades lineares u, v, w e angulares p, q, r. Para fins de navegação, as mesmas devem ser transformadas para o eixo fixo de referência inercial no mundo (*NORTH-EAST-UP*), por uma MCD (*Matriz Cosseno de Direção*):

$$\begin{bmatrix} V_{NORTH} \\ V_{EAST} \\ V_U P \end{bmatrix} = MCD \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \lambda_{13} \\ \lambda_{21} & \lambda_{22} & \lambda_{23} \\ \lambda_{31} & \lambda_{32} & \lambda_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.2}$$

A matriz M_{6x6} incorpora toda a massa e inércia do dirigível, e é dada por:

$$M = \begin{bmatrix} m-X_{\dot{u}} & 0 & 0 & 0 & ma_z-X_{\dot{q}} & 0 \\ 0 & M-y_{\dot{v}} & 0 & -ma_z-Y_{\dot{p}} & 0 & ma_x-Y_{\dot{r}} \\ 0 & 0 & m-Z_{\dot{w}} & 0 & -ma_x-Z_{\dot{q}} & 0 \\ 0 & -ma_z-L_{\dot{v}} & 0 & I_x-L_{\dot{p}} & 0 & -J_{xz} \\ ma_z-M_{\dot{u}} & 0 & -ma_x-M\dot{w} & 0 & I_y-M_{\dot{q}} & 0 \\ 0 & ma_x-N_{\dot{v}} & 0 & -J_{xz} & 0 & I_z-N_{\dot{r}} \end{bmatrix} \tag{3.3}$$

onde:

- m é a massa do dirigível;
- $m_x = m - X_{\dot{u}}$, $m_y = m - Y_{\dot{v}}$, $m_z = m - Z_{\dot{w}}$;
- $X_{\dot{u}}$, $Y_{\dot{v}}$ e $Z_{\dot{w}}$ são os termos de massa virtual para os eixos X, Y e Z, respectivamente;

- I_x, I_y e I_z são momentos de inércia sobre OX, OY e OZ, respectivamente;
- $L_{\dot{p}}$, $M_{\dot{q}}$ e $N_{\dot{r}}$ são os termos de inércia virtual sobre OX, OY e OZ, respectivamente;
- $J_x = I_x - L_{\dot{p}}$, $J_y = I_y - M_{\dot{q}}$ e $J_z = I_z - N_{\dot{r}}$;
- $J_{xz} = I_{xz} + N_{\dot{p}} = I_{xz} + L_{\dot{r}}$, onde I_{xz} é o produto da inércia sobre OY, e $N_{\dot{p}}$ e $L_{\dot{r}}$ são termos de inércia virtual;
- a_x e a_z são coordenadas do Centro Gravitacional e do Centro de flutuação;
- $X_{\dot{q}}$, $Y_{\dot{p}}$ e $Z_{\dot{r}}$ são os termos de massa virtual;
- $L_{\dot{v}}$, $M_{\dot{u}}$, $M_{\dot{w}}$ e $N_{\dot{v}}$ são os termos de inércia virtual;

A massa e inércia virtuais podem ser estimadas como (GOMES et. al., 1998) *apud* (LAMB, 1918) e (MUNK, 1936):

$$X_{\dot{u}} = -k_1 B/g \tag{3.4}$$

$$Y_{\dot{v}} = -k_2 B/g \tag{3.5}$$

$$Z_{\dot{w}} = Y_{\dot{v}} \tag{3.6}$$

$$M_{\dot{q}} = -k'[B/g][(l^2 + d^2)/20] \tag{3.7}$$

$$N_{\dot{r}} = M_{\dot{q}} \tag{3.8}$$

onde:

- B é a força de flutuação;
- g é a aceleração gravitacional;
- k_1 e k_2 são as taxas inerciais de Lamb para movimentos ao longo do eixo longitudinal (OX) e lateral (OY), respectivamente;
- k' é a taxa inercial de Lamb para rotações sobre o eixo lateral (OY);
- l e d são o comprimento e o diâmetro máximo do dirigível, respectivamente;

O vetor F_d representa as forças dinâmicas e contém os termos de Coriolis e centrífugos do modelo dinâmico, e é dado por:

$$F_d = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 \end{bmatrix}^T \tag{3.9}$$

onde:

$$f_1 = -m_z wq + m_y rv + m[a_x(q^2 + r^2) - a_z rp] \tag{3.10}$$

$$f_2 = -m_x ur + m_z pw + m[-a_x pq - a_z rq] \tag{3.11}$$

$$f_3 = -m_y vp + m_x qu + m[-a_x rp + a_z(q_2 + p_2)] \tag{3.12}$$

$$f_4 = -(J_z - J_y)rq + J_{xz}pq + ma_z(ur - pw) \tag{3.13}$$

$$f_5 = -(J_x - J_z)pr + J_{xz}(r_2 - p_2) + m[a_x(vp - qu) - a_z(wq - rv)] \tag{3.14}$$

$$f_6 = -(J_y - J_x)qp - J_{xz}qr + m[-a_x(ur - pw)] \tag{3.15}$$

O vetor A representa a força aerodinâmica. O mesmo é computado a partir dos coeficientes não-dimensionais de força de elevação (C_L), arrasto (C_D) e lateral (C_Y), bem como os momentos de *pitching* (C_m), *yawing* (C_n) e *rolling* (C_l), como:

$$A = F(C_i) = \begin{bmatrix} A_X & A_Y & A_Z & A_L & A_M & A_N \end{bmatrix}^T \tag{3.16}$$

Os coeficientes não-dimensionais C_i, podem ser obtidos pelos seguintes métodos:

- medição direta em um túnel de vento;
- a partir das características geométricas do veículo (GOMES et. al., 1998) *apud* (RAMOS, 1997);
- derivadas de estabilidade aerodinâmica.

Os coeficientes não-dimensionais de forças e momentos seguem o padrão convencional para dirigíveis:

$$C_D = A_X/(0.5\rho U^2 V^{2/3}) \tag{3.17}$$

$$C_Y = A_Y/(0.5\rho U^2 V^{2/3}) \tag{3.18}$$

$$C_L = A_Z/(0.5\rho U^2 V^{2/3}) \tag{3.19}$$

$$C_l = A_L/(0.5\rho U^2 V) \tag{3.20}$$

$$C_m = A_M/(0.5\rho U^2 V) \tag{3.21}$$

$$C_n = A_N/(0.5\rho U^2 V) \tag{3.22}$$

onde:

- A_X, A_Y e A_Z são as forças laterais, de elevação e de arrasto (em N);
- A_L, A_M e A_N são os momentos de *pitching*, *yawing* e *rolling* (em $N * m$);
- V é o modelo volumétrico (em m^3);
- U é a velocidade do ar em um túnel de vento, durante uma seção de testes (em m/s);
- ρ é a densidade do ar (em Kg/m^2).

G é o vetor de gravidade e flutuação, dado pela diferença entre o peso do dirigível, W, e a ação de flutuação, B; resolvido nos eixos do corpo do dirigível pelos parâmetros cosseno de direção λ_{ij}, e simplificado pela simetria do veículo sobre o plano XZ:

$$G = W - B = \begin{bmatrix} \lambda_{31}(W-B) \\ \lambda_{32}(W-B) \\ \lambda_{33}(W-B) \\ \lambda_{32}a_z W \\ (\lambda_{31}a_z - \lambda_{33}a_x)W \\ (\lambda_{32}a_x)W \end{bmatrix} \tag{3.23}$$

onde:

- λ_{ij} são elementos da MCD (*Matriz Cosseno de Direção*);
- W é o peso do dirigível atuando no centro de gravidade;
- B é a força de flutuação atuando no centro de volume;
- a_x e a_z são coordenadas do centro de gravidade nos eixos X e Z, respectivamente.

O vetor P contém os termos associados com as forças de propulsão e momentos. Para um dirigível com dois propulsores, um em cada lado da gôndola, P é expressado como:

$$P = \begin{bmatrix} X_{prop} & Y_{prop} & Z_{prop} & L_{prop} & M_{prop} & N_{prop} \end{bmatrix} \tag{3.24}$$

onde:

- $X_{prop} = (T_{ds} + T_{dp})cos(\mu)$;
- $Y_{prop} = 0$
- $Z_{prop} = -(T_{ds} + T_{dp})sin(\mu)$
- $L_{prop} = (T_{dp} - T_{ds})sin(\mu)d_y$
- $M_{prop} = (T_{ds} + T_{dp})[d_z cos(\mu) - d_x sin(\mu)]$
- $N_{prop} = (T_{dp} - T_{ds})cos(\mu)d_y$
- T_{ds} é a força propulsora do lado direito;
- T_{dp} é a força propulsora do lado esquerdo;
- μ é o ângulo de vetorização;
- d_x é a distância horizontal do centro de flutuação ao propulsor ao longo do eixo principal do dirigível;
- d_y é a distância horizontal do centro de flutuação ao propulsor perpendicular ao eixo principal do dirigível;
- d_z é a distância vertical do centro de flutuação ao propulsor;

Considerando-se um sistema heterogêneo de dirigíveis, agindo de forma cooperativa a fim de desempenhar uma missão determinada, como é o caso do projeto VANT do Instituto Militar de Engenharia, tem-se um sistema multiagentes, onde cada veículo é dado como um agente. Nas próximas seções, são apresentados os conceitos de agentes e sistemas multiagentes.

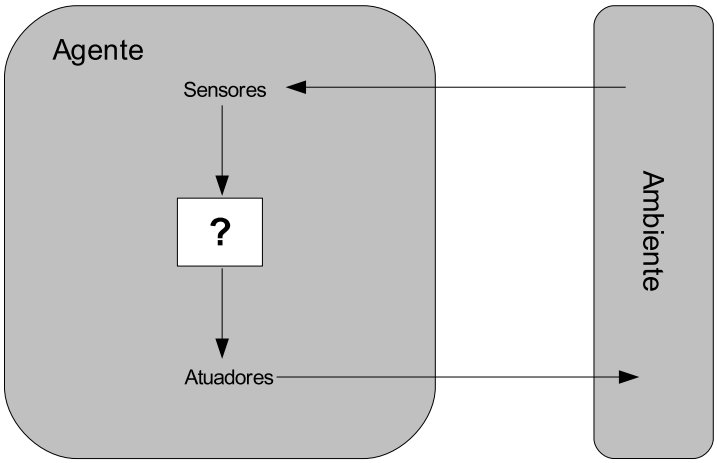


FIG. 3.3: Agentes interagem com o ambiente por meio de sensores e atuadores
Figura reproduzida a partir de (RUSSELL et. al., 2004)

3.2 AGENTES

Em situações reais de operação, os dirigíveis de um SDANT usam sensores para adquirir informações sobre o ambiente que os cercam e utilizam seus motores, equipamentos embarcados, e propriedades aerodinâmicas para locomover-se e executar suas funções em um dado espaço de trabalho. Diante desta situação, cada um dos veículos é considerado um agente. Pois, segundo a definição encontrada em (RUSSELL et. al., 2004), um agente é o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e de agir sobre esse ambiente por intermédio de atuadores. Essa idéia é ilustrada na FIG. 3.3. Um agente humano tem olhos, ouvidos e outros órgãos como sensores, e tem mãos, pernas, boca e outras partes do corpo que servem como atuadores. Um agente robótico poderia ter câmeras e detectores da faixa de infravermelho funcionando como sensores e vários motores como atuadores. Um agente de software recebe seqüências de teclas digitadas, conteúdo de arquivos e pacotes de rede como entradas sensórias e atua sobre o ambiente exibindo algo na tela, gravando arquivos e enviando pacotes de rede. Faremos a suposição geral de que todo agente pode perceber suas próprias ações (mas nem sempre os efeitos).

Outras definições podem ser encontradas em (WOOLDRIDGE et. al., 1998), (DURFEE, 1989), (FRANKLIN et. al, 1996) e (DELOACH et. al., 2001) entre outras obras presentes na bibliografia. Segundo (BOTELHO, 2005) *apud* (MARIETTO, 2000) a dificuldade em desenvolver uma definição consensual sobre agentes deve-se às seguintes

razões:

- A interdisciplinariedade e a abrangência dos campos de pesquisa e comercial, nos quais a teoria de agentes pode ser estudada e aplicada;
- Agentes estão presentes no mundo real. Geralmente, os conceitos envolvidos com ambientes nos quais os agentes estão inseridos são imprecisos e incompletos. Esta característica torna difícil a categorização dos ambientes, e como conseqüência, também a categorização dos agentes que os representam.

Em meio a todas as definições citadas e estudadas, a de (RUSSELL et. al., 2004) foi considerada a mais adequada devido à semelhança entre a situação do dirigível em relação ao agente descrito na bibliografia em questão.

3.3 SMA - SISTEMAS MULTIAGENTES

No contexto do projeto VANT, o paradigma dos sistemas multiagentes se torna muito interessante, pois, a partir do mesmo, é desenvolvida a interação e a comunicação dos dirigíveis (que são modelados como agentes, individualmente) para que a execução de trabalhos cooperativos em busca de objetivos comuns seja viabilizada. A seguir, será dada a definição de sistemas multiagentes encontrada em (HUBNER, 2003).

A área de SMA estuda o comportamento de um grupo *organizado* de agentes *autônomos* que cooperam na resolução de problemas que estão além das capacidades de resolução de cada um individualmente. Duas propriedades, aparentemente contraditórias, são fundamentais para os SMA: a autonomia dos agentes e sua organização (BRIOT, 2002). O atributo autônomo significa, aqui, o fato de que um agente tem sua existência independente dos demais e, até mesmo, do problema sendo solucionado (WEISS, 1999). Por outro lado, a organização estabelece restrições aos comportamentos dos agentes, visando estabelecer um comportamento grupal coeso. Muitas das propriedades desejadas nos SMA advêm do equilíbrio destes dois opostos. Portanto, compreender como estas duas propriedades interagem é uma questão importante (e interessante) no contexto dos SMA.

Tomando um ponto de vista de desenvolvimento de sistemas computacionais, o objetivo da área de SMA passa a ser a definição de modelos genéricos de agentes, interações

e organizações que possam ser instanciados dinamicamente, dado um problema (estas etapas são brevemente descritas na FIG. 3.4). De forma geral, o ciclo de vida de um SMA passa por duas etapas: concepção e resolução (SICHMAN, 1995). Na concepção são definidos modelos de propósito geral para os agentes, para suas interações e para suas formas de organização. Na resolução, um grupo de agentes adota estes modelos para resolver os problemas que lhe são apresentados. Diferentes tipos de problemas demandam dos agentes diferentes escolhas de modelos. A principal característica é a independência entre a concepção dos modelos e o problema, isto é, os modelos não são desenvolvidos para solucionar um problema particular. Por exemplo, os protocolos contratuais de (SMITH, 1980) são um modelo de interação aplicável em vários tipos de problemas. Dado este ideal de metodologia de desenvolvimento de sistemas, esta abordagem apresenta as seguintes características (ALVARES et. al., 1997):

- os agentes são concebidos independentemente de um problema particular;
- a interação entre os agentes não é projetada anteriormente, busca-se definir protocolos que possam ser utilizados em situações genéricas;
- a decomposição de tarefas para solucionar um dado problema pode ser feita pelos próprios agentes;
- não existe um controle centralizado da resolução do problema.

Destas características, decorrem algumas vantagens:

- Viabilizam sistemas adaptativos e evolutivos: o SMA tem capacidade de adaptação a novas situações, tanto pela eliminação e/ou inclusão de novos agentes ao sistema quanto pela mudança da sua organização.
- É uma metáfora natural para a modelagem de sistemas complexos e distribuídos: em muitas situações o conhecimento está distribuído, o controle é distribuído, os recursos estão distribuídos. Quanto à modelagem do sistema, a decomposição de um problema e a atribuição dos sub-problemas a agentes permite um alto nível de abstração e independência entre as partes do sistema (JENNINGS et. al., 1998)
- Tira proveito de ambientes heterogêneos e distribuídos: agentes com arquiteturas diferentes, que funcionam em plataformas diferentes, distribuídas em uma rede de

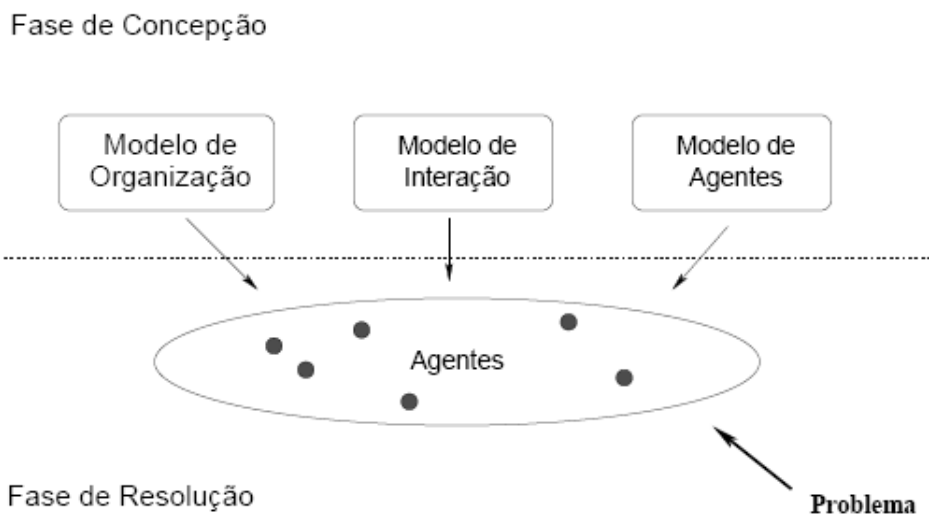


FIG. 3.4: Ciclo de vida de um Sistema Multiagentes
(SICHMAN, 1995)

computadores, podem cooperar na resolução de problemas. Isto permite o uso das potencialidades particulares de cada arquitetura e, pela distribuição, melhora o desempenho do sistema.

- Permite conceber sistemas abertos: os agentes podem migrar entre sociedades, isto é, agentes podem sair e entrar em sociedades, mesmo que desenvolvidos por projetistas e objetivos distintos. Tal abertura permite a evolução e a adaptabilidade do sistema.

Na bibliografia, diversas metodologias para modelagem de sistemas multi-agentes podem ser encontradas; como a OplA (*Object plus Agent Methodology*) (SCHWAMBACH, 2004), Vers (FLOURET et. al., 2002), AMACOIA (*Approche Multi-Agents pour la Conception d'Installations d'Assemblage*) (FRANCOIS et. al., 1997) e AUML (*Agent Unified Modeling Language*) (BAUER et. al., 2001). Contudo, a MaSE (*Multiagent Software Engineering*) (DELOACH et. al., 2001) foi adotada, devido a alguns pontos fortes apresentados pela mesma, como suportar a heterogeneidade entre os agentes, considerar os agentes especializações de objetos e ser independente de arquitetura e linguagem de programação, dentre outras. Na próxima seção é feita uma abordagem sobre a MaSE.

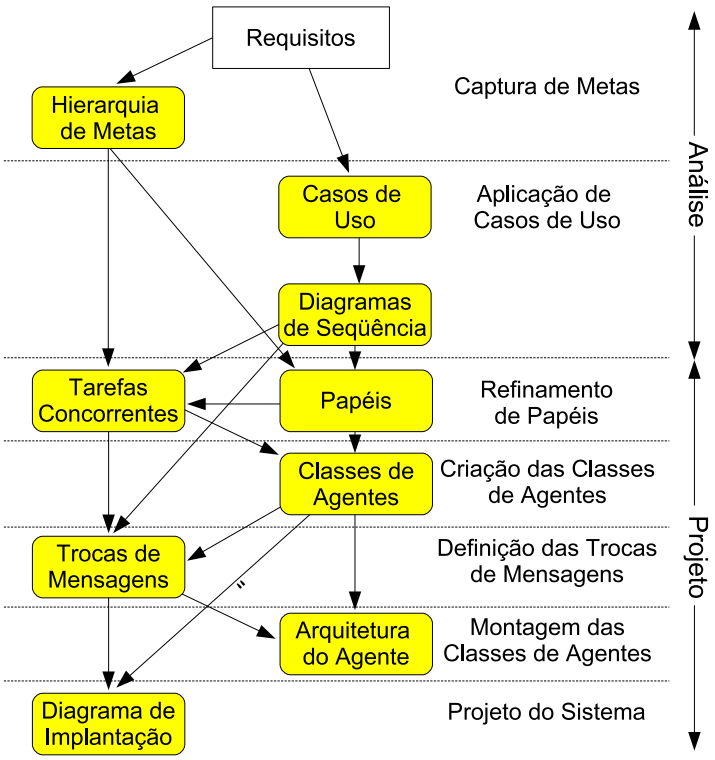


FIG. 3.5: Fases da metodologia MaSE
Figura adaptada de (WOOD et. al., 2001)

3.4 MASE - *MULTIAGENT SOFTWARE ENGINEERING*

A MaSE é uma metodologia para modelagem de sistemas multi-agentes que alveja guiar o projetista através do ciclo de vida do software a ser desenvolvido, sendo independente da arquitetura dos agentes e dos sistemas multi-agentes, linguagens de programação e dos sistemas de troca de mensagens empregados. A MaSE é dividida em fases, as quais possuem saídas que são utilizadas como entradas para as seguintes, e obedecem uma ordem, determinada, de execução (vide fluxograma da FIG. 3.5); São elas:

- captura de metas: é a fase inicial da MaSE; consiste em, a partir dos requisitos do sistema, criar uma hierarquia de objetivos que devem ser alcançados; basicamente, cria-se uma árvore de atividades a serem desempenhadas pelo sistema especificando-se uma relação de dependências e prioridades entre elas, a partir de um diagrama de metas;

- aplicação de casos de uso: a partir dos requisitos do sistema, são criados os casos de uso; cada um destes consiste em uma narrativa dos passos a serem seguidos,

ou executados, para realizar uma dada tarefa; cada caso de uso gera um diagrama de seqüências, que, por sua vez, é usado para determinar, e minimizar, o fluxo de mensagens trocadas entre os agentes participantes de uma tarefa, durante a execução da mesma;

- refinamento de papéis: o refinamento de papéis consiste em atribuir responsabilidades para os agentes; assim, para cada objetivo, traçado no diagrama de metas, há um ou mais agentes que contribuirão para que tais objetivos sejam alcançados; desta forma, atribui-se a cada agente, participação na execução de alguma(s) tarefa(s), e cada uma delas tem pelo menos um agente que deve contribuir para garantir sua execução; finalmente, a partir das relações tarefa/agente, são gerados o diagrama de papéis e os diagramas de tarefas concorrentes;

- criação de classes de agentes: nesta fase da MaSE, essencialmente, identifica-se as classes de agentes oriundos do diagrama de papéis e, então, cria-se um diagrama de classes de agentes, por meio do qual os agentes são representados, bem como as trocas de mensagens entre eles;

- definição das trocas de mensagens: após definir, nas fases anteriores, entre quais agentes será necessário realizar alguma comunicação, ou troca de informações, durante a execução do sistema, a próxima etapa é a definição das trocas de mensagens, que consiste em estabelecer um protocolo que especifica como as mesmas serão executadas; isto é feito através de diagramas de estados finitos;

- montagem de classes de agentes: esta fase possui, como objetivo, chegar ao máximo detalhamento dos agentes; assim são definidas as arquiteturas e o funcionamento dos agentes, conforme suas atribuições dentro do sistema, visando a implementação dos mesmos;

- projeto do sistema: esta é a última fase da MaSE, e é desenvolvida a partir do diagrama de classes de agentes e dos diagramas de estados, nos quais foram especificadas as trocas de mensagens entre os agentes; o objetivo é criar o diagrama de implantação que, por sua vez, mostra quantos, e quais, agentes terão a missão de agir, em cada um dos locais da área de atuação do sistema a ser desenvolvido.

A utilização da MaSE, durante o desenvolvimento de um sistema, pode ser viabilizada pelo uso de uma ferramenta chamada AgentTool (DELOACH & WOOD, 2001)

que oferece uma interface gráfica para o desenvolvimento dos diagramas e especificações requeridos pela metodologia em questão; maiores detalhes sobre a mesma são encontrados em (DELOACH, 2001), (DELOACH et. al., 2001), (MATSON et. al., 2002) e (WOOD et. al., 2001). Sua utilização viabilizou a modelagem do SDANT como um Sistema de Robôs Móveis Autônomos Cooperativos; a seguir, é dada uma abordagem sobre o assunto.

3.5 SISTEMAS DE ROBÔS MÓVEIS AUTÔNOMOS COOPERATIVOS

Podemos definir um robô como um agente físico que atua, efetivamente, no mundo real. Analogamente ao que acontece em tarefas executadas por pessoas, em uma sociedade, um trabalho, se realizado por uma equipe de robôs, pode ser feito de forma mais rápida e com melhor eficiência. Pois as regiões de atuação de cada elemento do grupo e a participação de cada um deles podem ser reduzidas. Assim, a equipe pode possuir membros mais especializados em suas respectivas tarefas e ambientes de atuação. Porém, para que isto, realmente, aconteça, é necessário que todos trabalhem em harmonia, ou seja, é necessário que haja cooperação entre os membros do grupo. Já a mobilidade dos robôs, os tornam mais versáteis, capazes de atuar em diferentes regiões do espaço de trabalho, sem a necessidade de serem levados de um lugar para outro. A seguir serão descritas as características essenciais que um sistema deste tipo deve possuir: mobilidade, autonomia e cooperação.

- Mobilidade: um robô é considerado móvel por ser capaz de locomover-se no seu espaço de trabalho. Para tanto, podem utilizar mecanismos como rodas, pernas, propulsores ou hélices, entre outros. Estes robôs podem ser utilizados em aplicações como transporte de cargas, limpeza doméstica, resgates ou vigilância.

- Autonomia: o grau de autonomia de um robô pode ser medido a partir de sua capacidade de cumprir tarefas sem intervenção ou ordem/informação de terceiros, sejam pessoas ou outros robôs. Assim, um robô, considerado autônomo, deve realizar seu trabalho utilizando seus atuadores e as informações oriundas de seus sensores.

- Cooperação: um sistema robótico cooperativo é caracterizado pelo fato de os membros da equipe cooperarem individualmente, cada um desempenhando um papel particular, no contexto geral da execução da tarefa, para que o objetivo do grupo seja alcançado em menor tempo, com menor custo, ou de melhor forma.

O sistema de navegação tratado nesta dissertação tem, como principal meio de orientação, imagens digitais capturadas a partir de câmeras embarcadas nos dirigíveis do SDANT (Sistema de Dirigíveis Aéreos Não-Tripulados). Para que tais imagens tenham utilidade real, é necessário extrair, delas, informações pontuais, como o formato, tamanho e localização dos obstáculos presentes no espaço de trabalho do veículo. Esta tarefa é executada por meio de técnicas de processamento de imagens e de visão computacional, as quais serão abordadas nos próximos parágrafos.

Uma imagem consiste de um suporte bidimensional no qual, para cada ponto associamos uma informação de cor. Assim, podemos utilizar como modelo matemático de imagens uma função $f : U \subset \Re^2 \rightarrow C$. O conjunto U é o *suporte de imagem* e o conjunto de valores de f é chamado de *gamute de imagem*. Nesse modelo, o domínio da função imagem, normalmente, é um retângulo $U = [a, b] \times [c, d]$, e o contra-domínio é um espaço tricromático $C = \Re^3$, como por exemplo o espaço RGB. Para poder representar uma imagem no computador temos que discretizar tanto o domínio quanto o contra-domínio da função imagem. Chamamos de *amostragem* a discretização do suporte geométrico e de *quantização* a discretização do espaço de cor (vide FIG. 3.6) (VELHO et. al., 2001).

Santos (SANTOS, 1995), descreve processamento de imagens e visão computacional da seguinte forma:

Entende-se por processamento de imagens, o conjunto de técnicas que permitem melhorar o aspecto da imagem. Estas técnicas têm como característica receberem como entrada uma imagem, e gerarem como saída uma outra imagem, que resulta da primeira mas que sofreu algumas alterações.

Visão computacional designa a ciência que extrai, de forma automática, informações da imagem para percepção autônoma da máquina. Exemplos de aplicações neste campo são o reconhecimento de caracteres, controle de robôs industriais, localização de objetos em movimento, etc. As técnicas de visão computacional recebem como entrada uma imagem e/ou estruturas de dados e geram outras estruturas de dados que representam

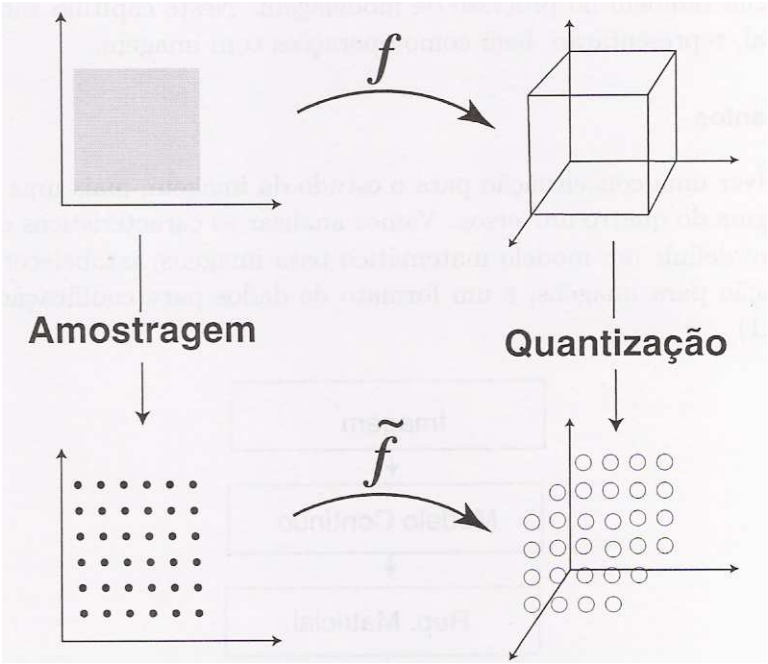


FIG. 3.6: Discretização de Imagens
(VELHO et. al., 2001)

alguma(s) característica(s) da imagem. Os algoritmos de processamento de imagens lidam com toda esta informação e, podem ser computacionalmente caros. Os algoritmos de visão computacional lidam com estruturas de dados abstratas que representam características da imagem (localização de objetos, contornos de objetos, texturas, etc.). A quantidade de informação a processar é bastante menor.

Os métodos que lidam com toda a imagem e geram outra imagem (como remoção de ruídos, por exemplo) são classificados como de baixo nível. Os métodos que recebem como entrada uma imagem e geram uma estrutura de dados que representa alguma característica dessa imagem (como a descrição do contorno de um objeto) são classificados como de nível intermediário. Os métodos que lidam apenas com estruturas de dados abstratas e geram outras estruturas de dados ou tomam decisões baseados na sua entrada são classificados como de alto nível. À medida que aumenta o nível, diminui a quantidade de dados a processar mas aumenta a complexidade do algoritmo (FIG. 3.7).

Para (GONZALES et. al., 1992), o processamento de imagens digitais abrange uma ampla escala de hardware, software e fundamentos teóricos e pode ser dividido nos se-

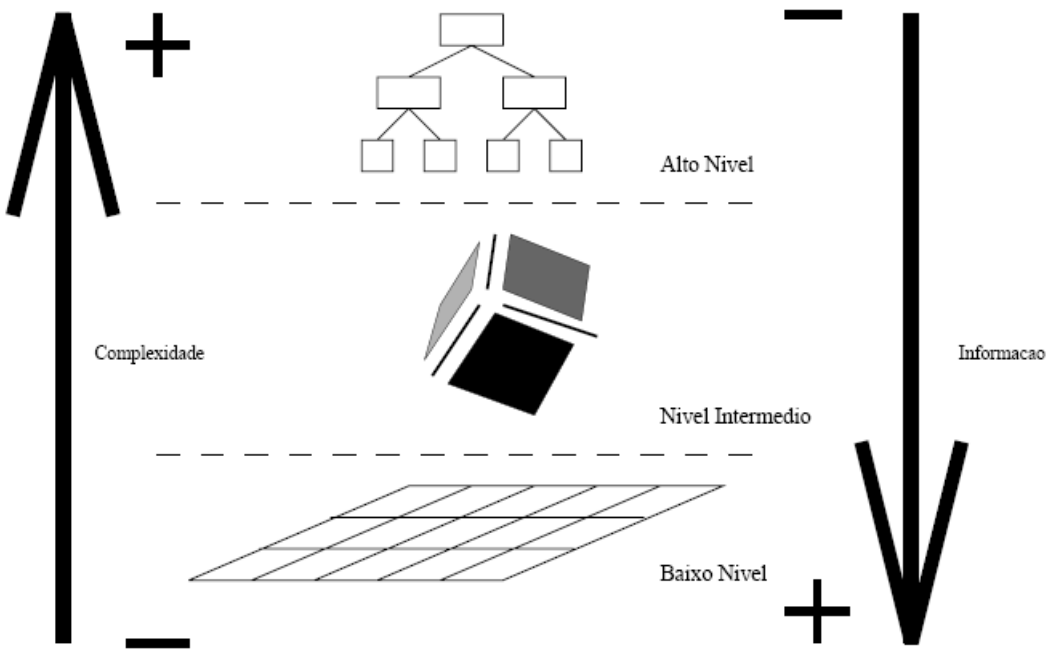


FIG. 3.7: Níveis de processamento
(SANTOS, 1995)

guintes passos, conforme FIG. 3.8:

- Aquisição da imagem: adquirir uma imagem digital a partir de algum sensor de imageamento;
- Pré-processamento: envolve técnicas de realce de contrastes, remoção de ruído e isolamento de regiões com o intuito de melhorar a imagem de forma a aumentar as chances para o sucesso dos processos seguintes;
- Segmentação: consiste em dividir uma imagem de entrada em partes ou objetos constituintes. Esta etapa geralmente é uma das mais difíceis no processamento de imagens digitais;
- Representação e descrição: consiste em converter os dados oriundos da segmentação para uma estrutura de dados adequada ao processamento computacional.
- Reconhecimento e interpretação: basicamente, corresponde ao processo de reconhecer um objeto e atribuir ao mesmo algum significado.

Santos (SANTOS, 1995) define a arquitetura para um sistema de visão computacional que é ilustrada pelo diagrama da FIG. 3.9. No qual uma placa digitalizadora (*frame*

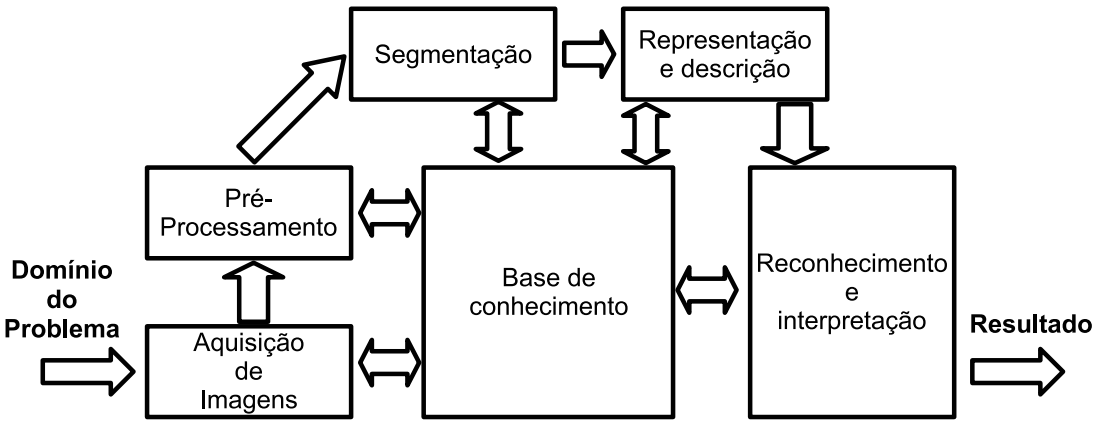


FIG. 3.8: Passos fundamentais em processamento de imagens (GONZALES et. al., 1992)

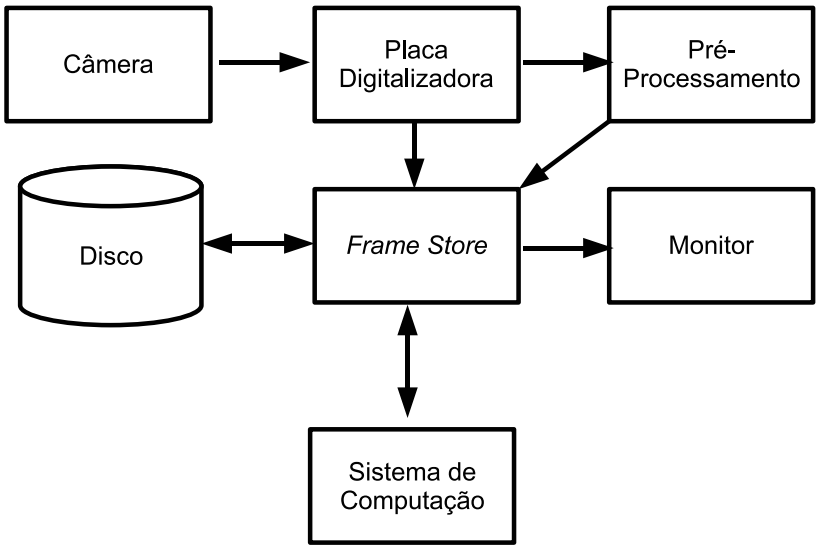


FIG. 3.9: Arquitetura de um sistema de visão computacional Figura adaptada de (SANTOS, 1995)

grabber) é responsável por converter a imagem analógica para digital. A *frame store* é uma área de memória onde a imagem é armazenada e pode ser acessada pelos outros módulos do sistema. E o pré-processador é responsável pelas operações de baixo nível. Já o sistema de computação é encarregado de executar as operações de alto nível.

3.6.1 SEGMENTAÇÃO DE IMAGENS

A segmentação de imagens é um problema de decomposição semântica, exigindo portanto soluções específicas para determinados problemas que requerem uma intensa intervenção humana. A escolha da técnica apropriada para determinada aplicação deve levar em conta a natureza da base de dados a ser utilizada e os resultados esperados do algoritmo, sendo os resultados analisados em função de sua forma, contornos, regiões e texturas dos objetos localizados (HUGUET, 2003).

Geralmente, os métodos direcionados a solução do problema de segmentação de imagens são baseados, basicamente, em duas características da imagem: descontinuidade ou similaridade entre os pixels de uma vizinhança. E costumam depender de parâmetros, como limiares ou sementes, que servem de ponto de partida para o algoritmo a ser executado. Simões (SIMOES, 2000), afirma que na segmentação baseada na descontinuidade a partição é baseada em alterações bruscas nos níveis da função imagem e que a segmentação baseada em similaridade busca agrupar regiões com características semelhantes. Dentre os métodos de segmentação de imagens presentes na literatura podemos destacar: crescimento e fusão de regiões, limiarização, watershed, transformada de Hough, *quadtrees* e a segmentação baseada em bordas.

3.6.2 VISÃO ESTEREOSCÓPICA

Tendo em vista as vantagens de um sistema de navegação baseado em visão, e o fato de os dirigíveis atuarem em ambientes dinâmicos e desestruturados, surge a necessidade de realizar o mapeamento do espaço de trabalho para detecção de obstáculos e, conseqüente prevenção de colisões com os mesmos. Como as câmeras utilizadas pelo sistema de visão são embarcadas, uma forma de localizar os obstáculos é calcular a distância entre estes e o veículo. Estes fatos tornam a visão estereoscópica uma metodologia adequada a ser usada como solução para esta problemática. Nos próximos parágrafos serão descritos alguns conceitos sobre estereoscopia.

A estereoscopia visa determinar a localização de um determinado ponto no espaço, a partir de suas projeções em duas imagens bidimensionais, capturadas de pontos de vista distintos. E baseia-se no mesmo princípio empregado pelo sistema de visão humano, o qual utiliza duas imagens planas, capturadas de pontos de vista distintos, para então obter informações em três dimensões. Para a aplicação deste método, em sistemas de visão computacional, é necessário determinar os parâmetros intrínsecos e extrínsecos do par de câmeras utilizadas, a disparidade entre elas, e um conjunto de pares de pontos correspondentes entre as imagens das mesmas. A partir destes dados, é possível extrair a localização de cada par de pontos correspondentes em relação a um referencial, que fica entre as duas câmeras.

A distância focal (f) é a distância entre o centro da lente da câmera e o plano da imagem, vide FIG. 3.10. Considerando-se um par de câmeras estereoscópicas fixadas com os eixos, de seus respectivos sistemas de coordenadas paralelos, e separados por uma distância (b), chamada de *baseline*, ao longo do eixo X; é possível calcular as coordenadas x, y e z, de um ponto qualquer, por meio de triangulação, em relação ao sistema de origem localizado entre as duas câmeras (vide FIG. 3.10), quando tem-se as projeções do mesmo nos planos de imagem das duas câmeras. Assim, tendo-se as coordenadas da projeção de um ponto no plano da imagem 1, $(x'l, y'l)$, e no plano da imagem 2, $(x'r, y'r)$; temos as seguintes equações (HORN, 1986):

$$\frac{x'l}{f} = \frac{x + b/2}{z} \tag{3.25}$$

$$\frac{x'r}{f} = \frac{x - b/2}{z} \tag{3.26}$$

$$y'l = y'r = \frac{y}{z} \tag{3.27}$$

Logo:

$$\frac{x'l - x'r}{f} = \frac{b}{z} \tag{3.28}$$

$$x = b\frac{(x'l + x'r)/2}{x'l - x'r} \tag{3.29}$$

$$y = b\frac{(y'l + y'r)/2}{x'l - x'r} \tag{3.30}$$

$$z = b\frac{f}{x'l - x'r} \tag{3.31}$$

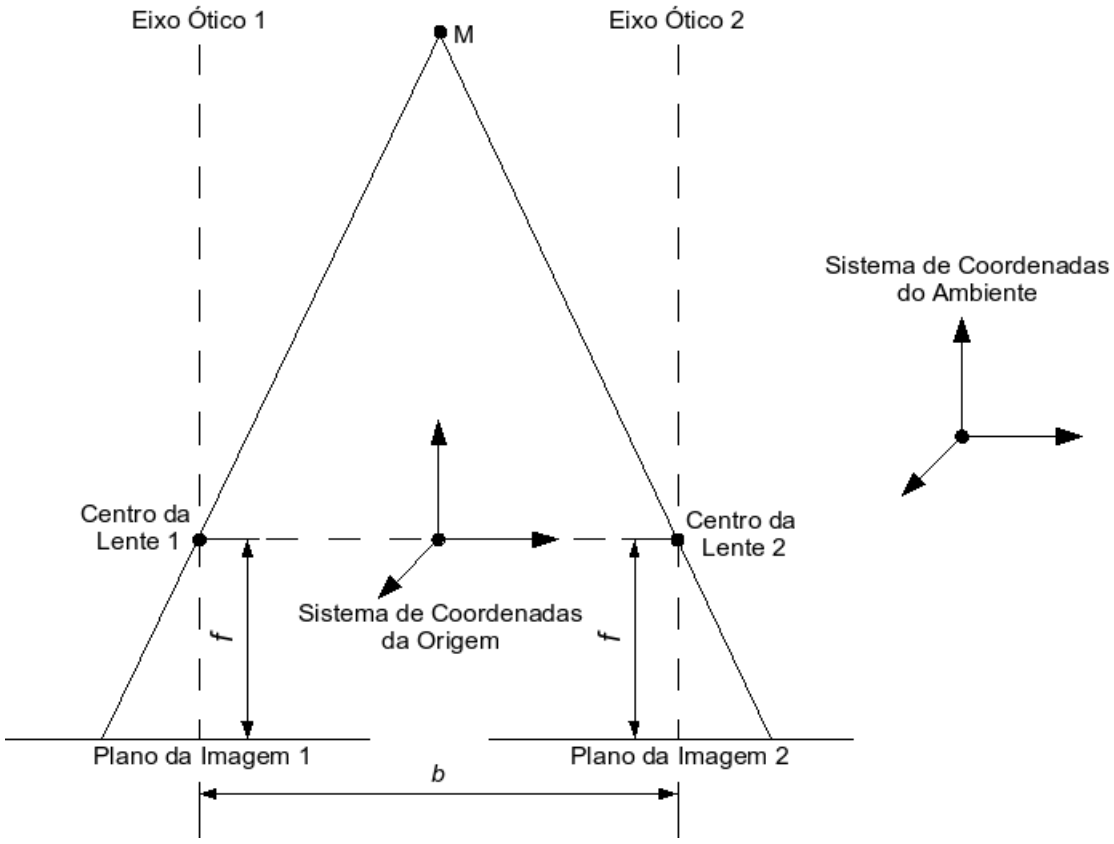


FIG. 3.10: Projeção de um ponto nos planos das duas imagens de um par estéreo

3.6.3 RECONSTRUÇÃO

Após a aplicação de algoritmos de visão estereoscópica nas imagens adquiridas a partir das câmeras embarcadas, é interessante que seja feita uma visualização do mapeamento para ter-se idéia de quão bom e preciso está sendo o método empregado. Para tanto, são usadas técnicas de reconstrução tridimensional.

Segundo (GOMES et. al., 2003b), em diversas situações é conveniente trabalhar com o modelo contínuo do objeto no universo matemático. A operação de obter o objeto no universo matemático a partir de sua representação discreta é chamado de *reconstrução*. A FIG. 3.11 ilustra as operações de representação e reconstrução. Em seguida são apresentados quatro fatores que ressaltam a importância da reconstrução:

a) Quando desejamos obter uma outra representação de um objeto podemos reconstruir o objeto e representar o objeto reconstruído.

b) A reconstrução é útil quando precisamos trabalhar no domínio contínuo de forma a minimizar erros de cálculo.

c) A reconstrução é fundamental no processo final de visualização de um objeto. Como exemplo, para visualizar uma imagem precisamos reconstruí-la no monitor.

d) Em geral o usuário especifica um objeto gráfico determinando uma representação, uma vez que a interface de especificação permite apenas a especificação de um número finito de parâmetros. O objeto deve então ser reconstruído a partir da especificação.

A operação de reconstrução é totalmente dependente da representação utilizada e em geral é muito difícil de ser calculada. Quando um determinado método de representação possui um método de reconstrução que permite obter o objeto original a partir de sua representação, dizemos que a representação é *exata* caso contrário a representação é dita *aproximada*. Representações exatas são muito difíceis de serem obtidas. Em geral os métodos de reconstrução fornecem aproximações dos objetos originais.

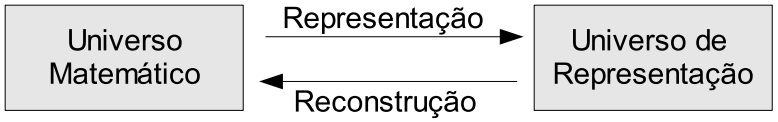


FIG. 3.11: Representação e reconstrução
Figura reproduzida de (GOMES et. al., 2003b)

3.7 SLAM - *SIMULTANEOUS LOCALIZATION AND MAPPING*

Durante a navegação robótica em ambientes desconhecidos, onde existam obstáculos móveis, não é possível que o robô movimente-se com segurança, a não ser que seja feito um mapeamento contínuo do espaço de trabalho, de modo que possam ser captadas mudanças na configuração do ambiente. Além disso, por conseqüência dos movimentos do robô, a posição do mesmo é modificada freqüentemente. Portanto, também é necessário calcular sua localização continuamente, pois é de fundamental importância conhecer a sua posição corrente em relação ao ponto objetivo e aos obstáculos, para que seja feito o planejamento e o seguimento de trajetórias. Desta forma, surge o problema de Localização e Mapeamento Simultâneos, conhecido na bibliografia como SLAM.

O mapeamento do ambiente deve ser feito de forma rápida, pois trata-se de uma aplicação com restrição de tempo, e pode ser calculado a partir de dados obtidos de sensores embarcados, como câmeras, sonares ou lasers, em um ou vários robôs. Diversas metodologias são utilizadas para a resolução do problema como derivação probabilística, filtro de Kalman, filtro de partículas e fusão de submapas. Já a localização pode ser obtida a partir de sensores como GPS e INS ou ainda, a partir de características extraídas do ambiente pelos sensores embarcados.

3.8 PLANEJAMENTO DE TRAJETÓRIA

O problema de planejamento de trajetória pode ser definido como a tarefa de conduzir um sistema móvel de uma posição corrente a uma posição de destino (PIO et. al., 2003). E é a atividade básica da robótica móvel. No planejamento de trajetória global traça-se uma trajetória da posição inicial até a posição objetivo. Tal trajetória é calculada a partir de um mapa do ambiente armazenado previamente, supondo que este seja estático. É usado, principalmente, em navegação de longa distância onde os sensores do robô não conseguem

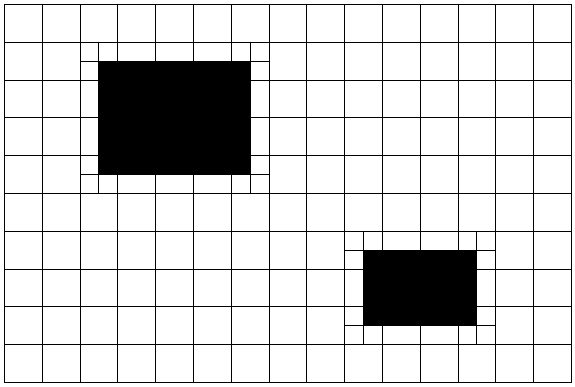


FIG. 3.12: Exemplo de decomposição em células por método exato.

captar informações sobre todo o percurso a partir da posição inicial. Caso apareçam novos obstáculos, na trajetória traçada inicialmente, esta é modificada por meio de um planejamento local para que não aconteça colisão do atuador com eventuais obstáculos. Entre os métodos de planejamento de trajetória podemos citar os *roadmaps*, diagramas de voronoi, decomposição em células, campo potencial e o campo de força virtual que serão brevemente descritos a seguir.

3.8.1 DECOMPOSIÇÃO EM CÉLULAS

O planejamento de trajetória, a partir do método de Decomposição em Células, consiste em dividir o espaço de trabalho em um conjunto de subregiões, onde cada uma delas é tratada como uma célula; as conexões entre estas células podem ser representadas por um grafo não dirigido. Assim, o problema de planejamento de trajetória se resume a encontrar um caminho, entre a célula de partida e a célula destino. O método de Decomposição em Células é abordado na bibliografia, classicamente, de duas maneiras: Decomposição em Células por métodos exatos e Decomposição em Células por métodos aproximados.

Segundo Latombe (LATOMBE, 1991), os métodos exatos de decomposição em células dividem o espaço livre em células, cuja a união é exatamente o espaço livre. O limite de uma célula corresponde a algum tipo de mudança crítica, isto é, uma mudança crítica as restrições aplicadas ao movimento do robô. Vide FIG. 3.12.

Ainda seguindo a definição de Latombe (LATOMBE, 1991), os métodos aproximados de decomposição em células produzem células de formatos pré-definidos (por exemplo,

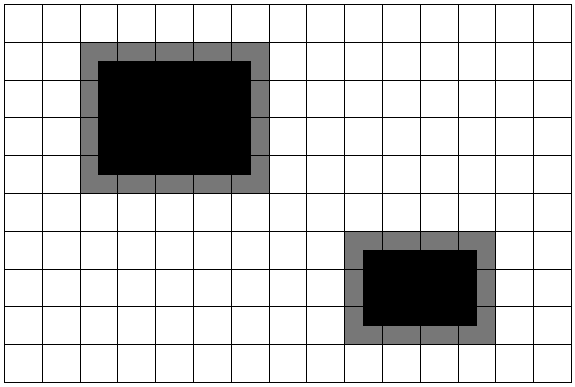


FIG. 3.13: Exemplos de decomposição em células por método aproximado.

retangulares), das quais a união é estritamente incluída no espaço livre. O limite de uma célula não caracteriza algum tipo de descontinuidade e não possui significado físico. Vide FIG. 3.13.

3.8.2 *ROADMAP*

A abordagem de *roadmap* para o planejamento de trajetória consiste em capturar a conectividade do espaço livre do robô em uma rede de curvas unidimensionais, chamadas de roadmap. Uma vez que um roadmap R é construído, ele pode ser usado como um conjunto de caminhos padronizados. Assim, o planejamento de trajetória é reduzido à conexão dos pontos inicial e objetivo em R e buscando em R um caminho entre estes pontos. Se algum caminho for construído, este corresponde a concatenação de três subcaminhos: o primeiro, conectando o ponto inicial ao roadmap; o segundo, um subcaminho contido no *roadmap*; e o último, um subcaminho conectando o roadmap ao ponto objetivo (LATOMBE, 1991). Dentre os métodos que já foram propostos para a construção de *roadmaps*, podemos citar o Grafo de Visibilidade (vide FIG. 3.14) e o Diagrama de Voronoi (vide FIG. 3.15).

3.8.3 CAMPO POTENCIAL

Segundo Menezes (MENEZES, 1999), este método baseia-se na colocação de cargas repulsivas, geradoras de um campo, nos obstáculos e uma carga atrativa na posição objetivo. O robô escolhe a trajetória do gradiente do potencial gerado em direção ao seu mínimo. O controle da plataforma pode ser feito tanto em nível cinemático como dinâmico usando

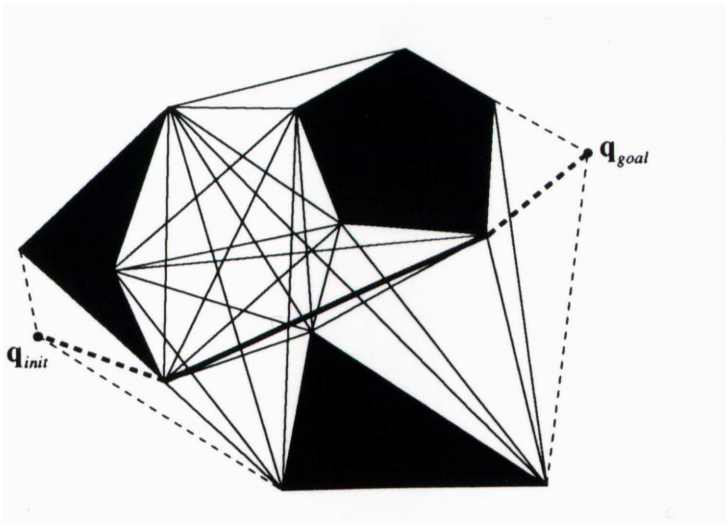


FIG. 3.14: *Roadmap* construído com Grafo de Visibilidade (LATOMBE, 1991)

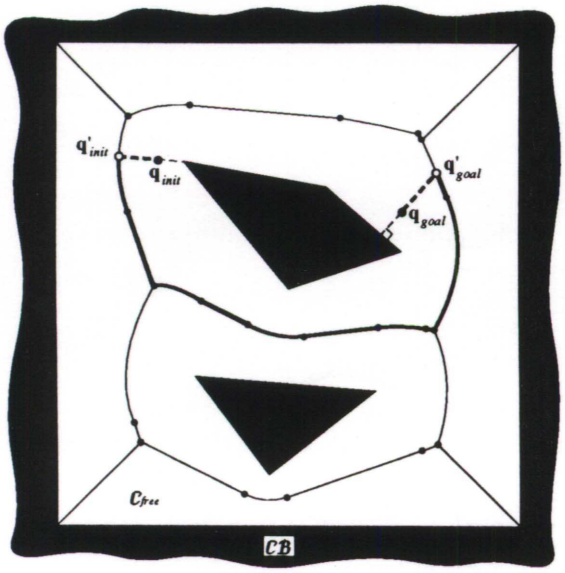


FIG. 3.15: *Roadmap* construído com Diagrama de Voronoi (LATOMBE, 1991)

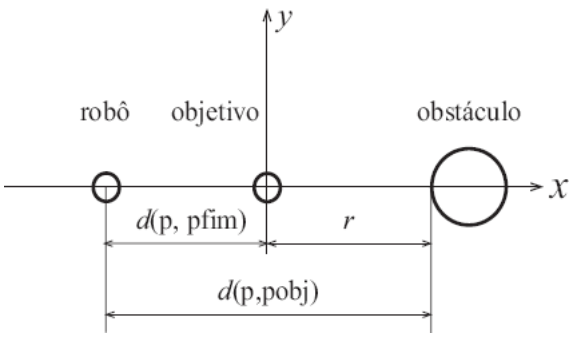


FIG. 3.16: Exemplo de um mínimo local do Campo Potencial
(PIO et. al., 2003)

o equivalente à força aplicada no robô por este campo para induzir valores de velocidade e aceleração. Este método tem sido largamente utilizado devido à sua simplicidade e capacidade de reação. No entanto, a sua maior falha é a possibilidade de obter mínimos locais para determinadas configurações de obstáculos e posição objetivo.

A idéia básica desse método é representar o robô no espaço F e tratar o robô como sendo uma partícula sob influência de um campo potencial artificial U. O campo potencial U é construído para refletir localmente a estrutura do espaço livre (PIO et. al., 2003).

O problema dos mínimos locais no método do Campo Potencial pode acontecer devido à anulação das forças de repulsão e atração exercidas sobre o robô, vide FIG. 3.16.

3.8.4 CAMPO DE FORÇA VIRTUAL

Segundo (PIO et. al., 2003), O Campo de Força Virtual utiliza os conceitos da Grade de Ocupação com os conceitos dos Campos Potenciais, como pode ser percebido através da atuação das forças e da área de influência projetada para o robô mostrada na FIG. 3.17. O espaço de configuração C é discretizado em uma grade regular e sobre essa grade se aplica um campo potencial. O espaço C passa a ser representado por uma grade C_{ij}, um histograma bidimensional conhecido também como Histograma de Campo Vetorial, onde cada célula dessa grade contém um valor que representa a certeza de existência de um obstáculo. O histograma passa então a ser atualizado a medida que o robô vai navegando a partir das informações fornecidas pelos sensores, que verificam a certeza ou não da existência de obstáculo em cada célula. Assim, as células que representarem os obstáculos

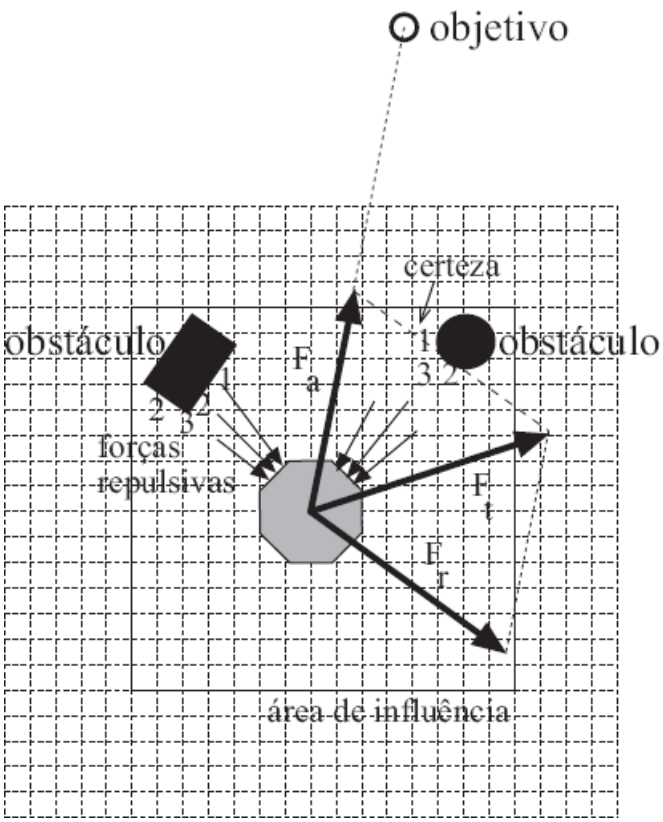


FIG. 3.17: Forças repulsivas provenientes dos obstáculos que atuam sobre o robô em sua área de influência

(PIO et. al., 2003)

terão valores maiores de certeza do que as que não representarem.

3.9 METAHEURÍSTICAS

As metaheurísticas, geralmente, são utilizadas em situações, nas quais a solução do problema demande um grande período de tempo para ser encontrada, como pode acontecer com a execução de algoritmos que utilizam busca exaustiva, por exemplo. Pois, se por um lado não retornam uma solução ótima, por outro, conseguem gerar uma boa solução em tempo hábil. Outro problema das metaheurísticas são os mínimos locais, neste caso, o projetista deve analisar as características do problema e do método utilizado para tentar sair dos mínimos locais ou não permitir que se chegue nos mesmos.

Segundo (GOMES, 2003a) *apud* (REEVES, 1993), uma metaheurística é uma técnica que procura soluções boas (próximas da ótima) a um custo computacional razoável sem

se tornar hábil para garantir a viabilidade ou o grau de otimalidade dessa solução, ou até em muitos casos a metaheurística pode dizer o quão próxima uma solução viável está da solução ótima. Isso para casos em que a solução ótima já seja conhecida para algumas instâncias do problema em análise. Outra forma de medir a qualidade da solução é compará-la com métodos que geram limites inferiores e/ou superiores para as instâncias dos problemas em estudo que não se conhece seus valores ótimos.

A seguir, será descrita a metaheurística colônia de formigas, a qual foi utilizada neste trabalho para realizar o planejamento de trajetória.

3.9.1 COLÔNIA DE FORMIGAS

A metaheurística Colônia de Formigas, também conhecida como ACO (*Ant Colony Optimization*) é uma técnica de computação evolutiva proposta por Marco Dorigo (DORIGO et. al., 1999) e trata-se de uma técnica de resolução de problemas de otimização inspirada no comportamento de ninhos de formigas. Estas, apesar de não serem racionais, possuem mecanismos naturais de otimização para encontrar o melhor caminho através de experimentos aleatórios. Na busca por comida, para descobrirem um caminho ótimo, do ninho à fonte de alimento, diversas formigas saem a procura de alimentos. Durante o percurso, por onde passam, depositam uma substância química chamada feromônio. As formigas seguintes devem seguir o caminho pelo qual sentirem que exista maior quantidade de feromônio. Desta forma, estabelece-se uma comunicação indireta, entre as formigas de uma colônia, baseada em trilhas de feromônio.

Segundo (COELHO, ET. AL., 2004), as formigas reais são capazes de encontrar o caminho mais curto para uma fonte de alimento do formigueiro sem a utilização de dados visuais. Enquanto caminham, as formigas depositam feromônio no solo, e tem seu deslocamento baseado em trilhas de feromônios previamente depositados por outras formigas. Estas trilhas de feromônios podem ser observadas por outras formigas e motivar elas em seguir determinado caminho, isto é, um movimento aleatório das formigas segue com maior probabilidade uma trilha de feromônio. Esta é uma maneira de como as trilhas são reforçadas e, cada vez mais, formigas tendem a seguir aquela trilha. Uma formiga trabalha da seguinte forma: Primeiro, quando as formigas chegam a um ponto de decisão em que elas precisam tomar a decisão de mover-se à direita ou à esquerda, as formigas

selecionam aleatoriamente o próximo caminho e depositam feromônio no solo, sem ter a noção de qual é a melhor escolha. Depois de um pequeno período de tempo, a diferença entre a quantidade de feromônio entre dois caminhos é suficientemente grande para influenciar a decisão de novas formigas que estiverem indecisas por qual caminho seguir. Neste caso, as novas formigas escolhem o caminho com maior quantidade de feromônio. Conseqüentemente, as formigas podem cheirar o feromônio e escolher os caminhos marcados com concentrações mais acentuadas de feromônios.

A metaheurística Colônia de Formigas é de inteligência coletiva baseada em uma população de formigas que possui as seguintes características (COELHO, ET. AL., 2004):

- é um algoritmo não-determinístico baseado em mecanismos presentes na natureza, isto é, ele é baseado no comportamento de formigas para a determinação de caminhos através de suas colônias para procura eficiente de fontes de comida;
- é um algoritmo paralelo e adaptativo, pois uma população de agentes move-se simultaneamente, de forma independente e sem um supervisor (não há um controle ou supervisão central);
- é um algoritmo cooperativo, pois cada formiga escolhe um caminho com base na informação (trilhas de feromônios) depositadas por outros agentes que tenham selecionado previamente o mesmo caminho. Este comportamento cooperativo tem ingredientes de autocatálise (catálise provocada por feromônios que se formam no próprio sistema reativo), isto é, a metaheurística Colônia de Formigas providencia uma realimentação positiva, desde que a probabilidade de um agente escolher o caminho aumente com o número de agentes que escolheu previamente aquele caminho.

Este capítulo é destinado a apresentar a solução proposta para o desenvolvimento de um sistema de navegação para dirigíveis aéreos não-tripulados baseado em imagens. A qual, é um sistema computacional que deverá ser embarcado em dirigíveis, e possui 4 (quatro) módulos responsáveis por executar as tarefas inerentes ao mesmo. As próximas seções abordam a modelagem do sistema, com a descrição dos seus módulos, e as tarefas a serem realizadas por eles.

4.1 MODELAGEM DO SISTEMA DE NAVEGAÇÃO

Primeiramente, será dada uma abordagem sobre a modelagem do projeto VANT do IME; seguindo o que foi exposto em (PINHEIRO, 2006), em especial no que compreende as tarefas relacionadas com a navegação dos dirigíveis. Faz-se isso, para facilitar o entendimento da inserção do sistema de navegação proposto, no projeto supracitado. A modelagem do projeto VANT foi desenvolvida utilizando a técnica de engenharia de sistemas multiagentes MaSE, e será descrita, parcialmente, nos seguintes níveis: diagrama de metas, casos de uso, diagrama de papéis, diagrama de tarefas concorrentes e diagrama de classes de agentes.

Nos próximos parágrafos serão mostrados alguns diagramas, ou fragmentos dos mesmos, da modelagem feita por (PINHEIRO, 2006), a começar com um fragmento do diagrama de metas (vide FIG. 4.1) que ilustra as metas diretamente ligadas, ao sistema de navegação, dentre elas podemos destacar:

- 1.4 Posicionar SDANT;
- 1.4.2 Calcular Trajetória;
- 1.6 Decolar SDANT;
- 1.7 Pousar SDANT;

Seguindo o diagrama de metas, destacam-se os seguintes casos de uso:

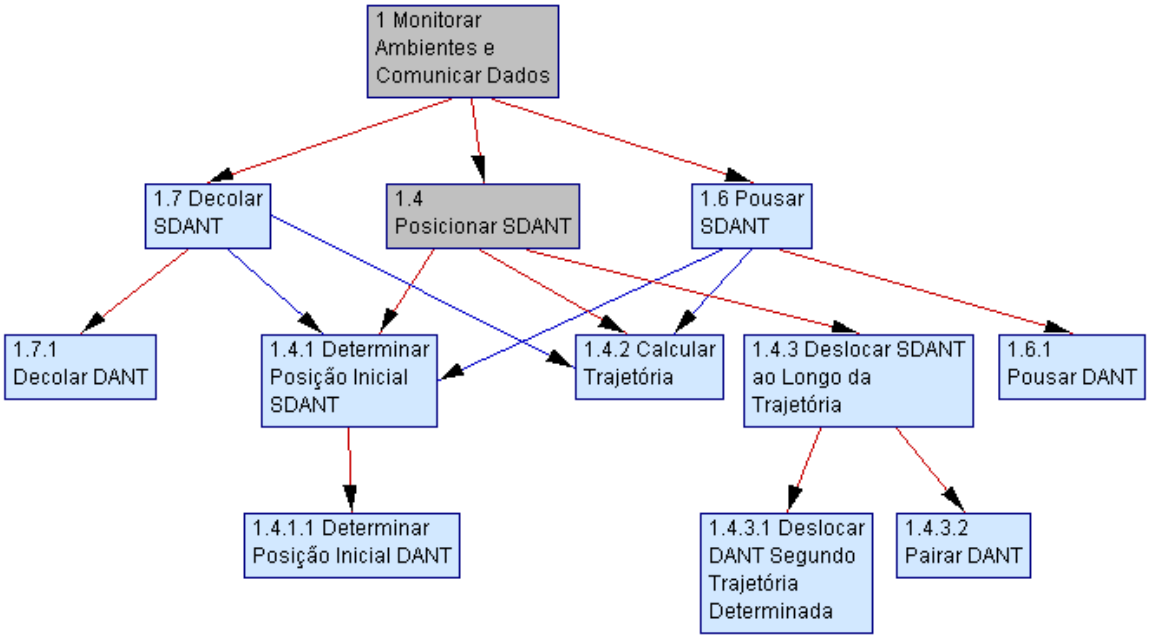


FIG. 4.1: Fragmento do diagrama de metas

- Posicionar SDANT: comanda o posicionamento dos DANT (Dirigíveis Aéreos Não-Tripulados). Este posicionamento segue os seguintes passos: determinar a posição inicial do SDANT; calcular a trajetória; deslocar o SDANT ao longo da trajetória determinada; e pairar SDANT. Pré-Condição: SDANT deve pairar.

- Decolar SDANT: comanda a decolagem dos DANTs. Esta decolagem segue os seguintes passos: decolagem do DANT Âncora (o DANT que abstrai o SDANT); em seguida, o DANT que lhe é adjacente; e assim sucessivamente. Após decolarem os DANTs pairam a 100 metros de altura. O SDANT define também, ao comandar a decolagem de cada DANT, a posição que cada DANT deve ocupar imediatamente após a decolagem. Os DANTs não podem mover-se para uma eventual posição de destino sem que respeitem a restrição da intercomunicação deles. Pré-Condição: SDANT deve estar ativo;

- Pousar SDANT: comanda o pouso dos DANTs. O pouso segue os seguintes passos: determinar a posição inicial do SDANT; calcular a trajetória; deslocar o SDANT ao longo da trajetória determinada; e pousar SDANT. Pré-Condição: SDANT deve pairar.

O diagrama de papéis (vide FIG. 4.2) mostra quais agentes atuam e como eles in-

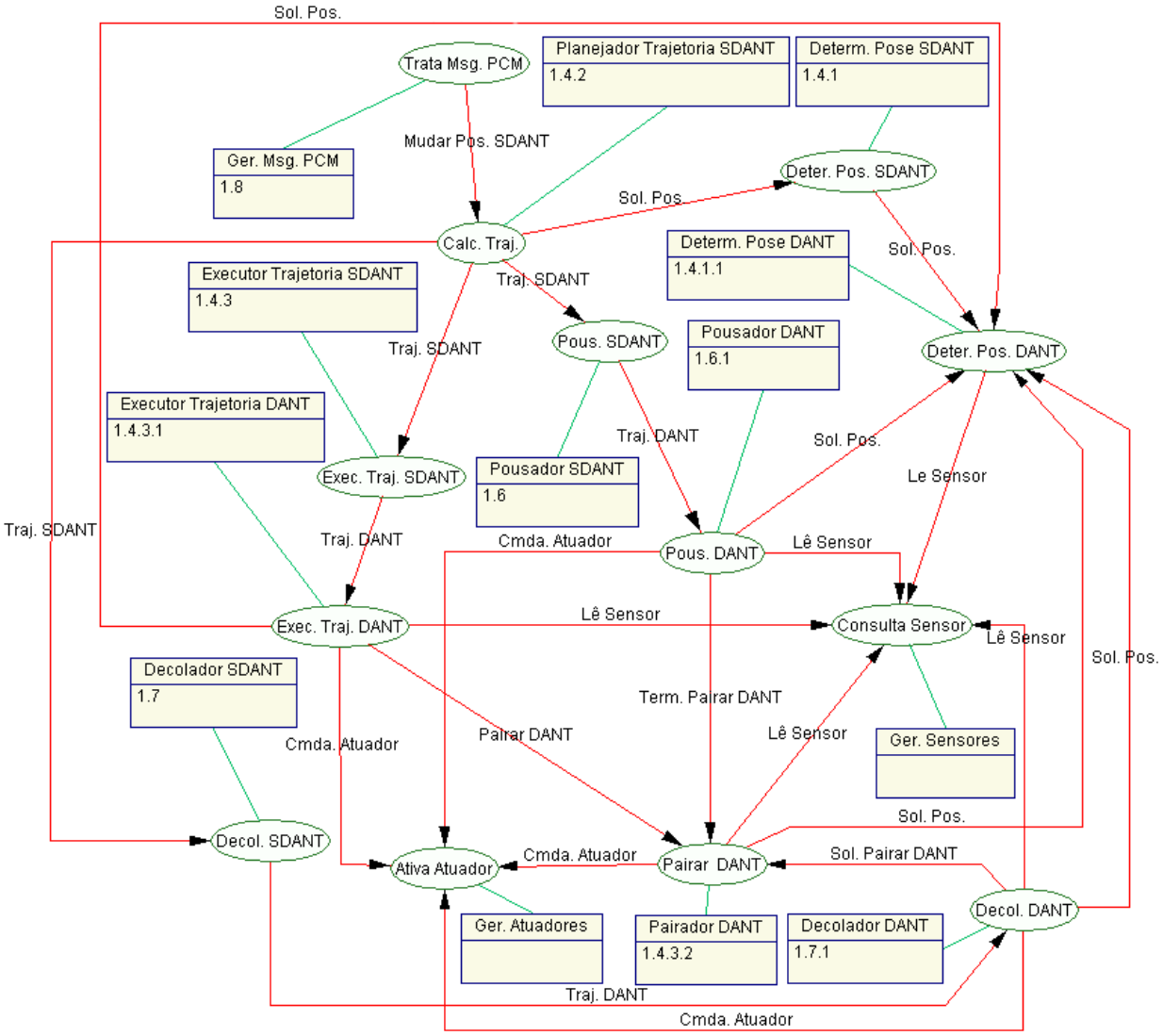


FIG. 4.2: Fragmento do diagrama de papeis

teragem para executar as tarefas inerentes ao sistema de navegação, além de mostrar as dependências das tarefas dentre as mesmas.

Como o diagrama de papéis mostra cada tarefa de forma bem superficial, é interessante apresentá-las com mais detalhes. Para tanto, são mostrados os seguintes diagramas de tarefas concorrentes:

- Calc. Traj. (calcular trajetória, vide FIG. 4.3);

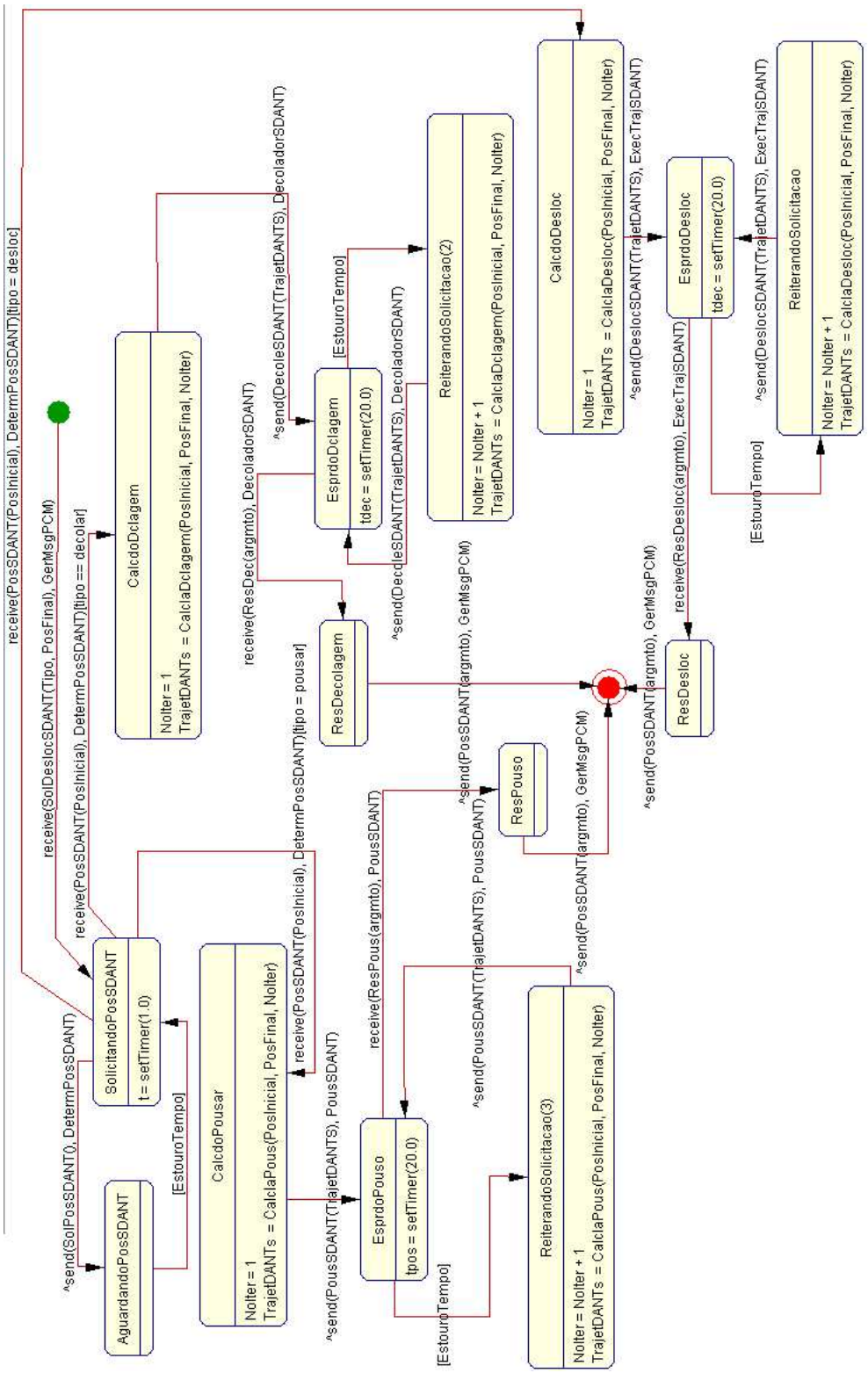


FIG. 4.3: Diagrama de Tarefas Concorrentes - Calcular Trajetória

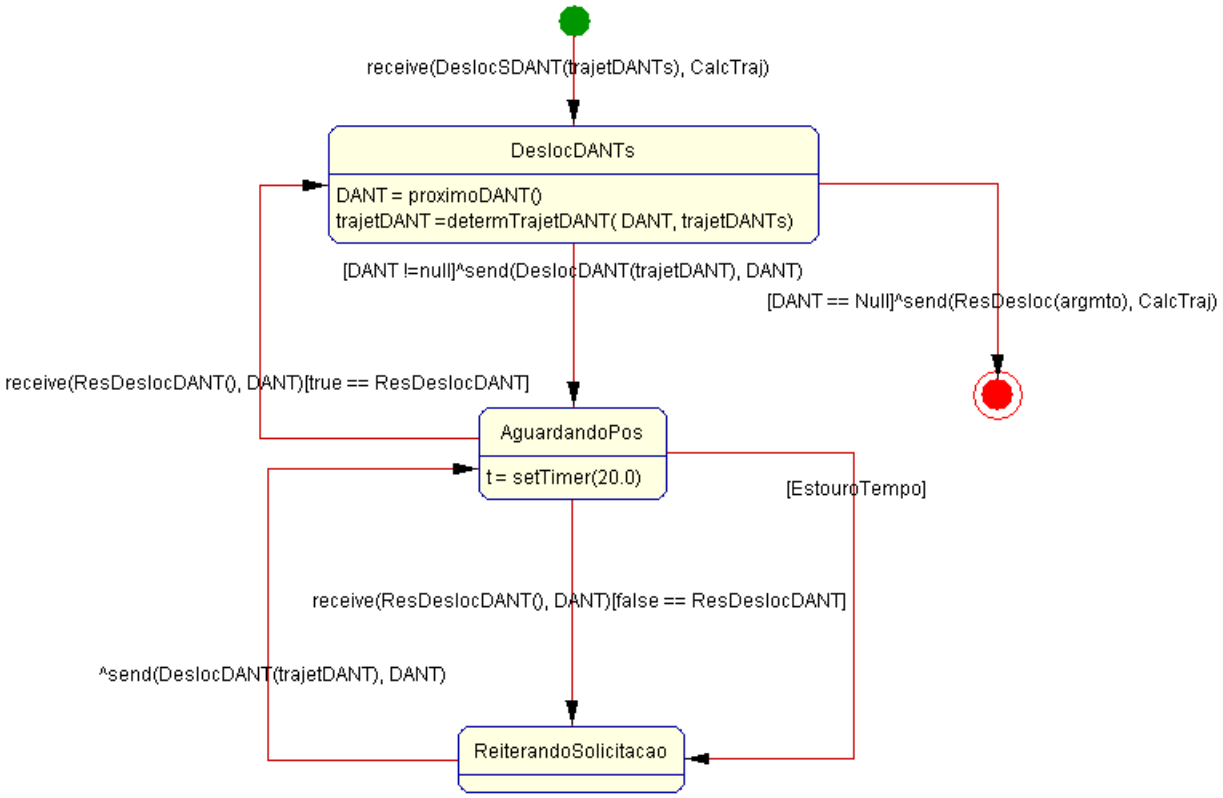


FIG. 4.4: Diagrama de Tarefas Concorrentes - Exec. Trajetória do SDANT

- Exec. Traj. SDANT (Executar Trajetória do SDANT, vide FIG. 4.4);
- Exec. Traj. DANT (Executar Trajetória do DANT, vide FIG. 4.5);

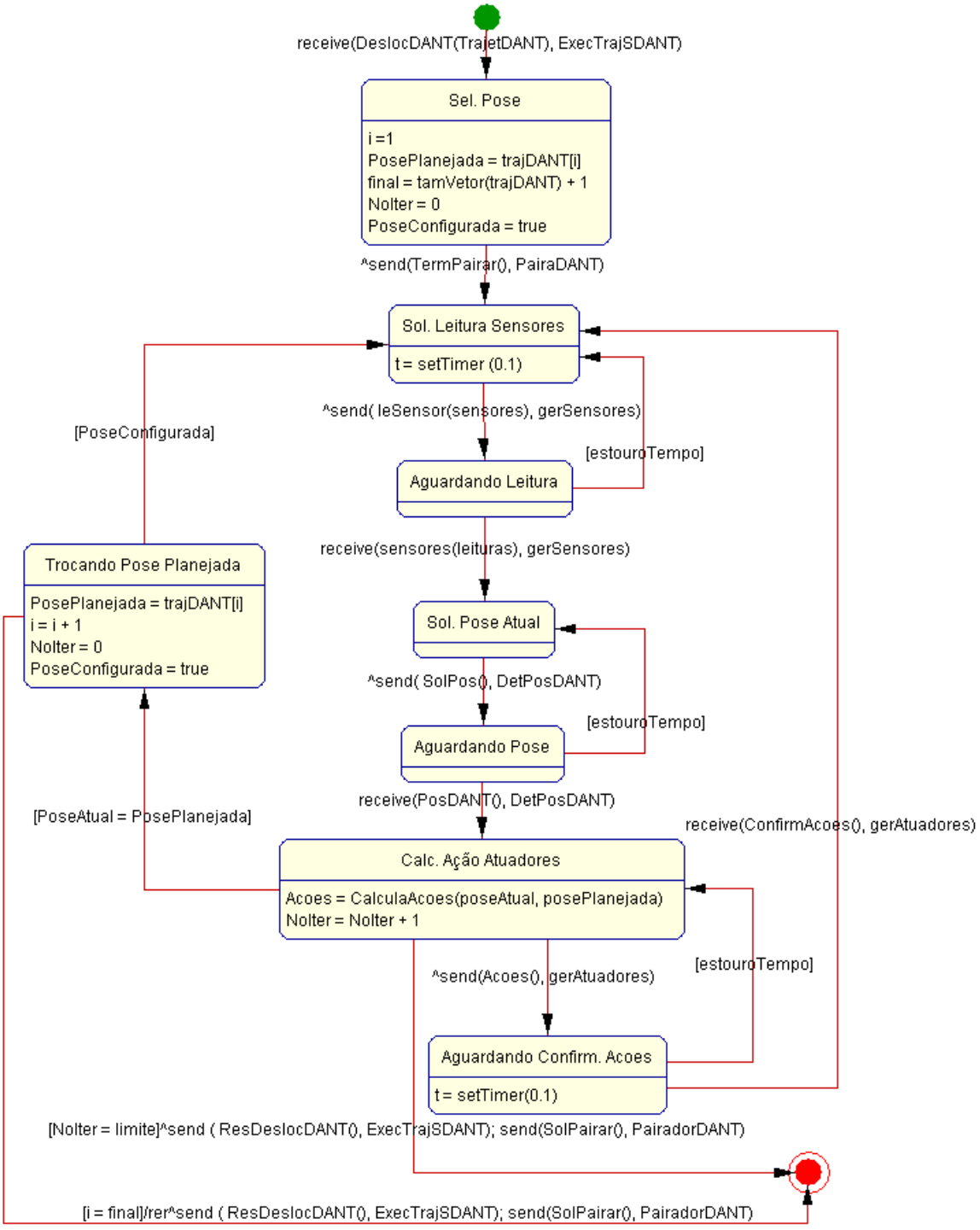


FIG. 4.5: Diagrama de Tarefas Concorrentes - Executar Trajetória do DANT

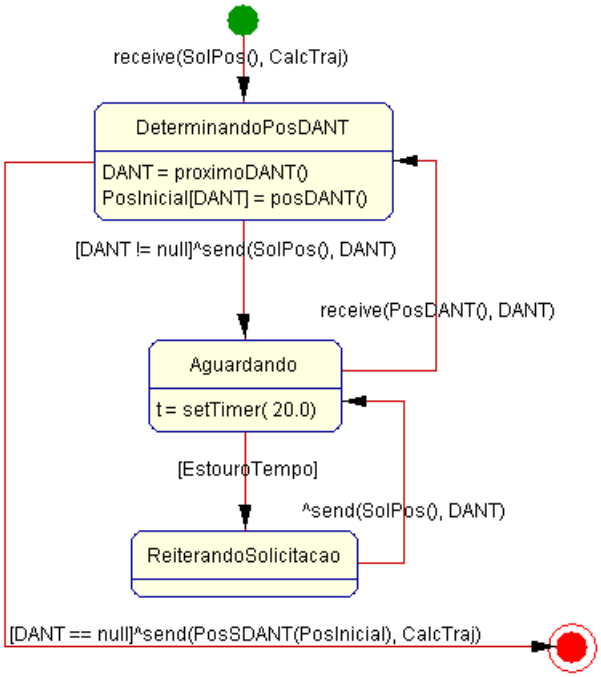


FIG. 4.6: Diagrama de Tarefas Concorrentes - Determinar Pose do SDANT

- Deter. Pos. SDANT (Determinar Pose do SDANT, vide FIG. 4.6);

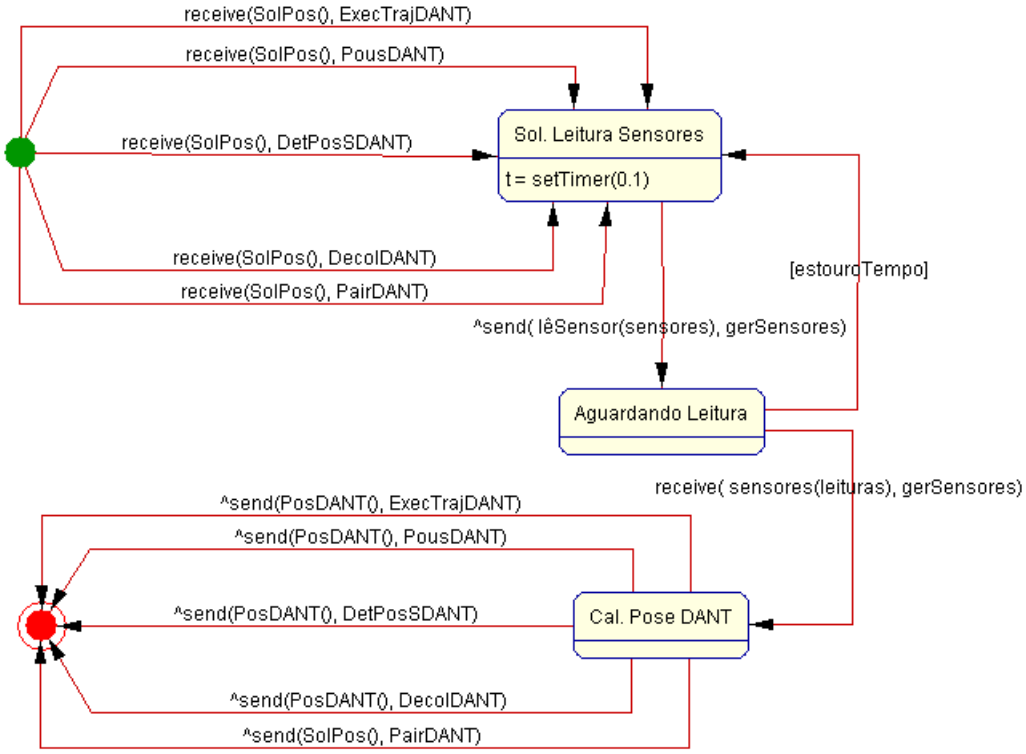


FIG. 4.7: Diagrama de Tarefas Concorrentes - Determinar Pose do DANT

- Deter. Pos. DANT (Determinar Pose DANT, vide FIG. 4.7);

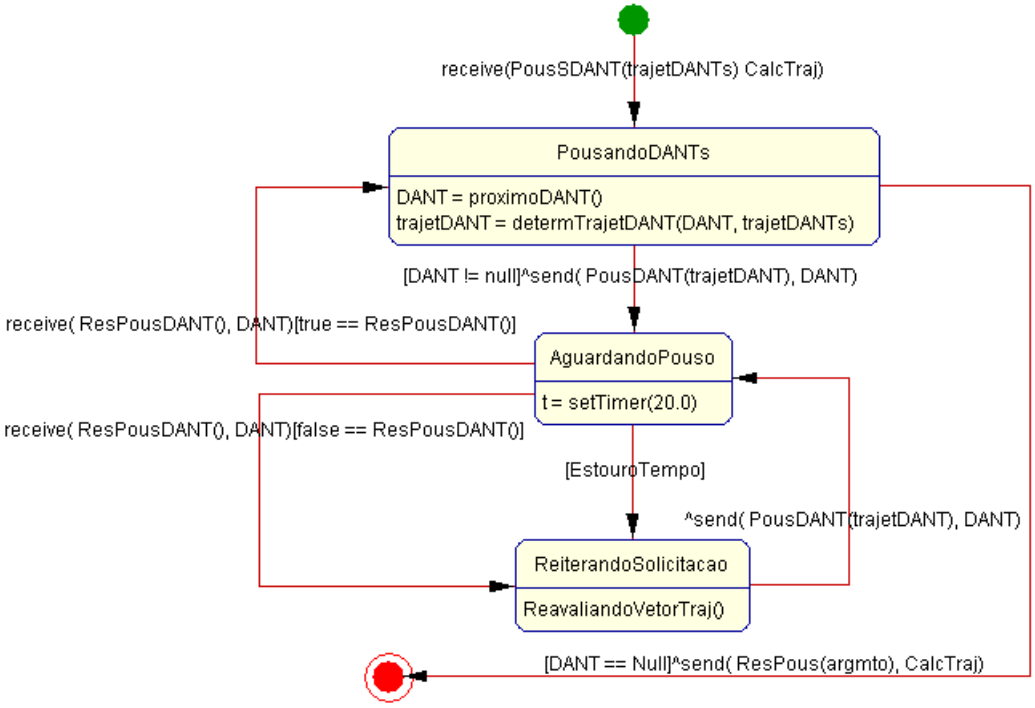


FIG. 4.8: Diagrama de Tarefas Concorrentes - Pousar SDANT

- Pous. SDANT (Pousar SDANT, vide FIG. 4.8);
- Pous. DANT (Pousar DANT, vide FIG. 4.9);

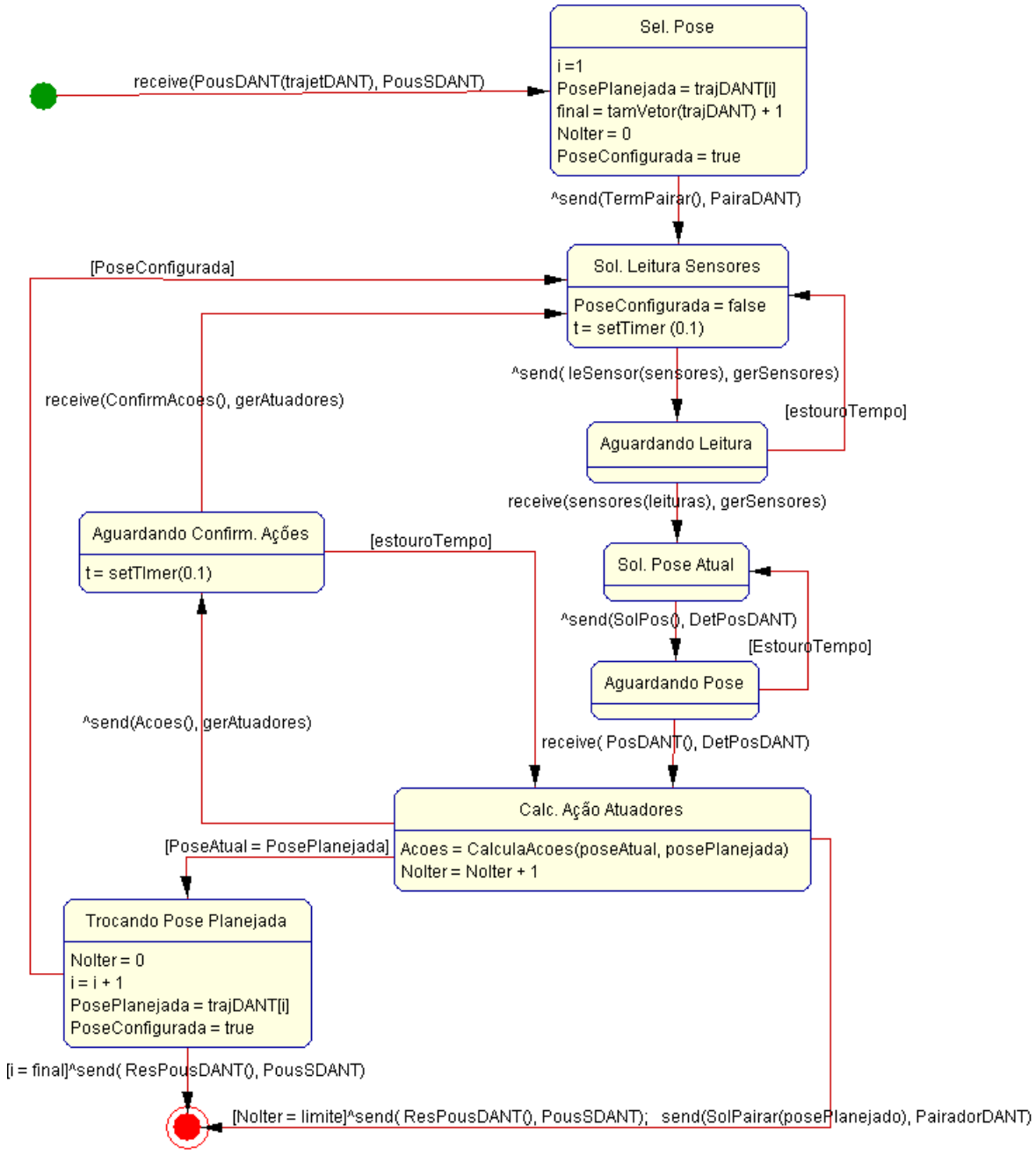


FIG. 4.9: Diagrama de Tarefas Concorrentes - Pousar DANT

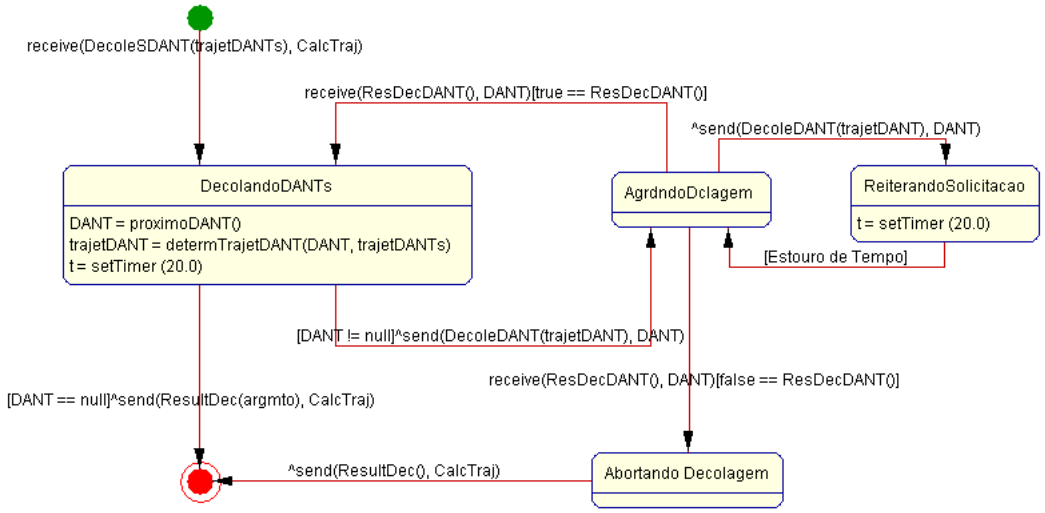


FIG. 4.10: Diagrama de Tarefas Concorrentes - Decolar SDANT

- Decol. SDANT (Decolar SDANT, vide FIG. 4.10);
- Decol. DANT (Decolar DANT, vide FIG. 4.11);

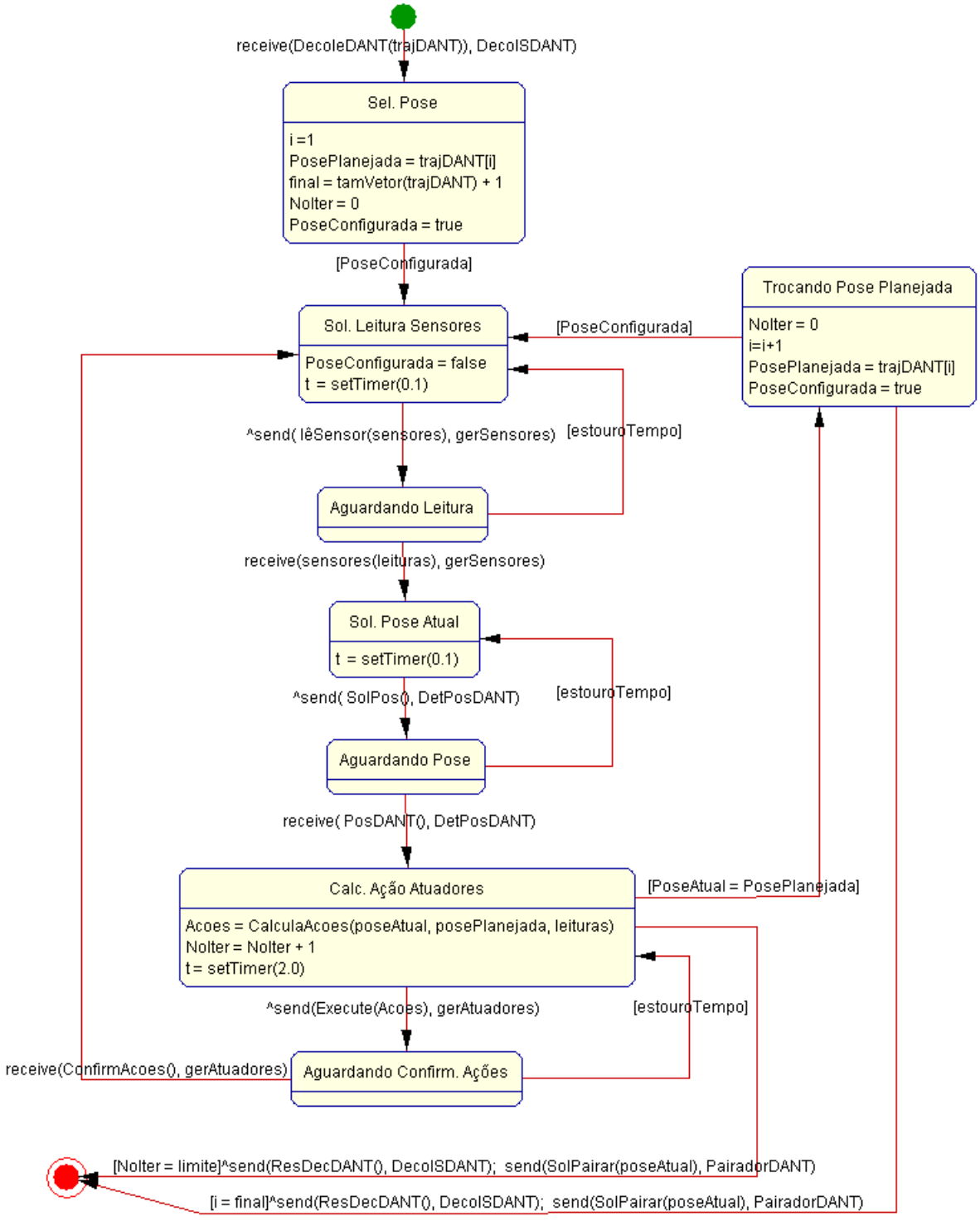


FIG. 4.11: Diagrama de Tarefas Concorrentes - Decolar DANT

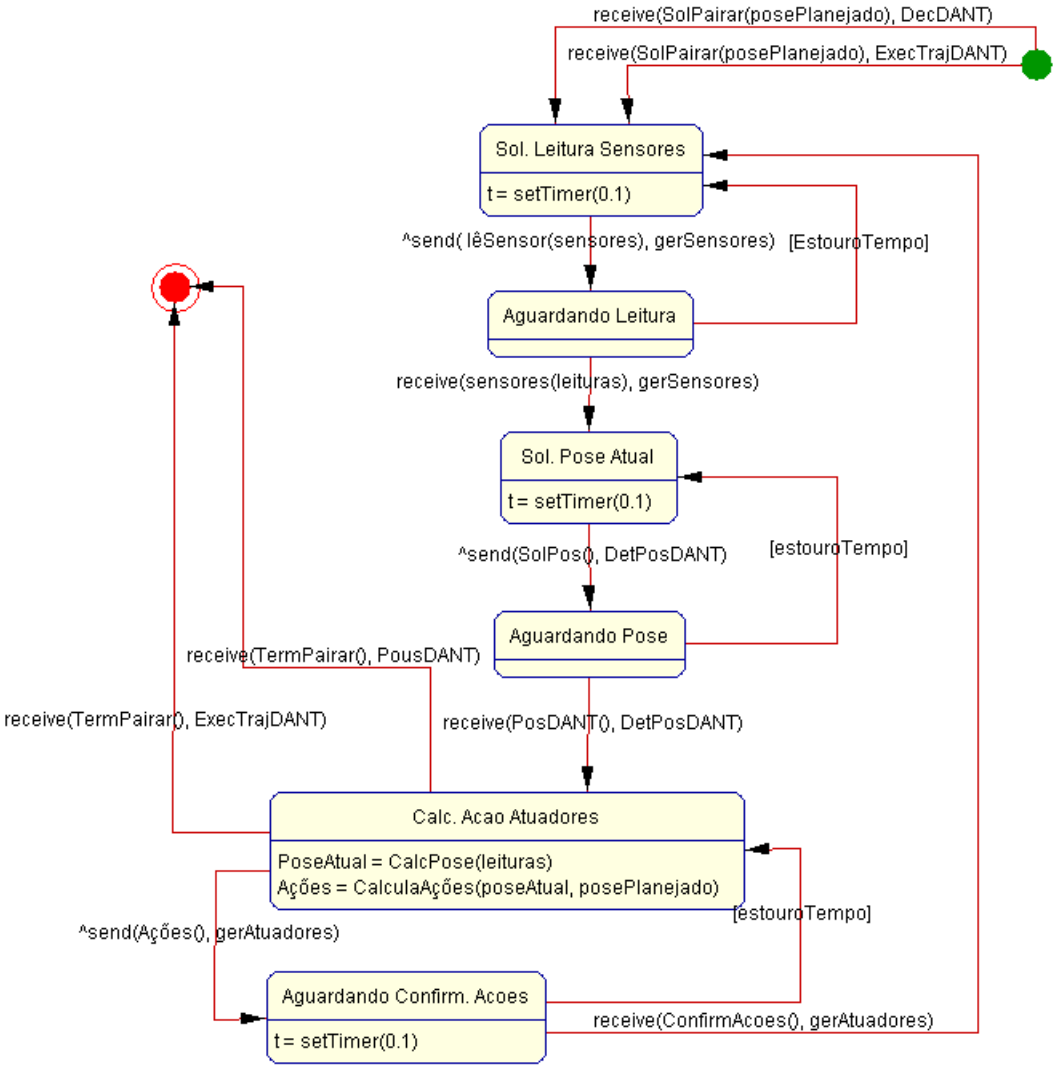


FIG. 4.12: Diagrama de Tarefas Concorrentes - Pairar DANT

- Pairar DANT (vide FIG. 4.12).

Por fim, é mostrado um fragmento do diagrama de classes de agentes que exibe, de forma mais detalhada, os agentes que participam do sistema de navegação do VANT (vide FIG. 4.13).

A modelagem, parcialmente apresentada anteriormente, não contém um detalhamento do funcionamento do sistema de navegação empregado no projeto VANT do IME que, por sua vez, é fundamental para a viabilização do atendimento de alguns requisitos não-funcionais do SDANT como evitar colisões, corrigir a trajetória e realizar o deslocamento

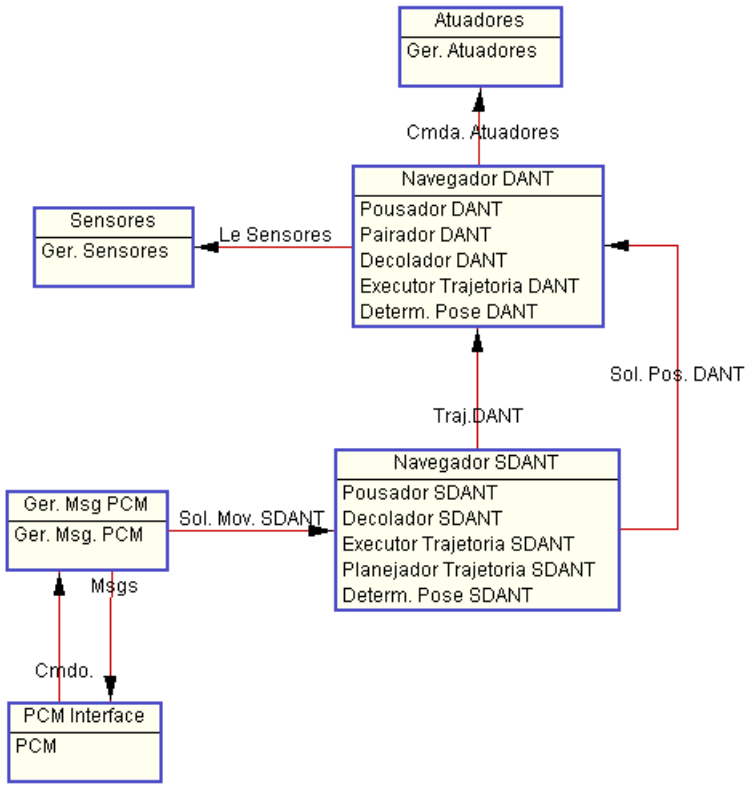


FIG. 4.13: Fragmento do Diagrama de Classes de Agentes

da frota automaticamente. Por este motivo, foi necessário desenvolver um complemento para a mesma. Assim, após análise do funcionamento do sistema em questão, optou-se por dividi-lo nos seguintes módulos:

- Módulo de visão: responsável por processar as imagens oriundas dos sensores;
- Módulo de comunicação: responsável por enviar e receber mensagens para os outros membros da equipe;
- Módulo de localização: responsável por determinar a localização do dirigível e dos objetos detectados pelo módulo de visão;
- Módulo de planejamento de trajetória: responsável por definir a trajetória de um dirigível, a partir de dados gerados pelo módulo de localização;
- Módulo de controle: responsável por determinar as ações dos atuadores do dirigível para que o mesmo movimente-se seguindo determinada trajetória.

O agente Sensores detém a responsabilidade pelos módulos de visão e de localização; este agente deve capturar imagens das câmeras embarcadas e, a partir delas, enviar o mapeamento do ambiente ao módulo de planejamento de trajetória. Já a execução das tarefas, pertinentes ao módulo de planejamento de trajetória, cabem ao agente NavegadorDANT. Por fim, o módulo de comunicação é implementado pelo agente Gerenciador de Mensagens PCM (Planejador e Controlador da Missão) - Ger.Msg.PCM.

Uma visão geral da interação entre os módulos do sistema de navegação pode ser vista através da ilustração da FIG. 4.14. E um maior detalhamento será dado a seguir.

O módulo de visão é responsável pelo processamento das imagens obtidas a partir das câmeras embarcadas. Isto é feito conforme o processo ilustrado no fluxograma da FIG. 4.15. O primeiro passo é a aquisição de duas imagens (imagens da câmera 1 e câmera 2); em seguida, a imagem da câmera 1 é segmentada, dividindo-a em um conjunto de partes, onde cada uma delas corresponde a um objeto ou a alguma parte dele; e por fim, para cada segmento, são estabelecidos pontos de controle; e, para cada um deles, é calculado um ponto da imagem da câmera 2, que seja seu correspondente. Após a conclusão do processamento das imagens capturadas, o módulo de visão envia os resultados ao módulo

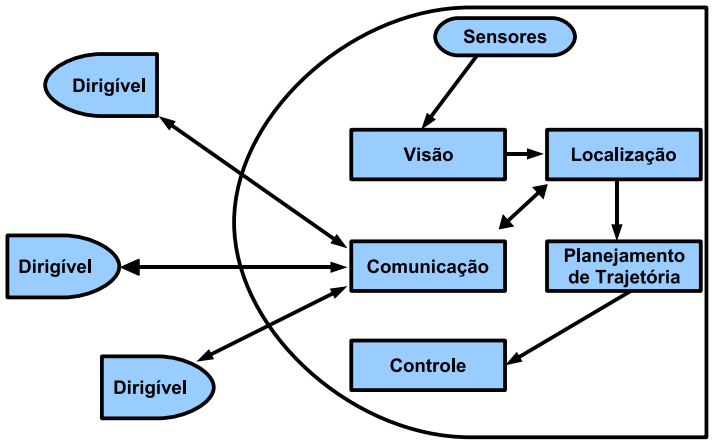


FIG. 4.14: Interação entre os módulos do sistema de navegação

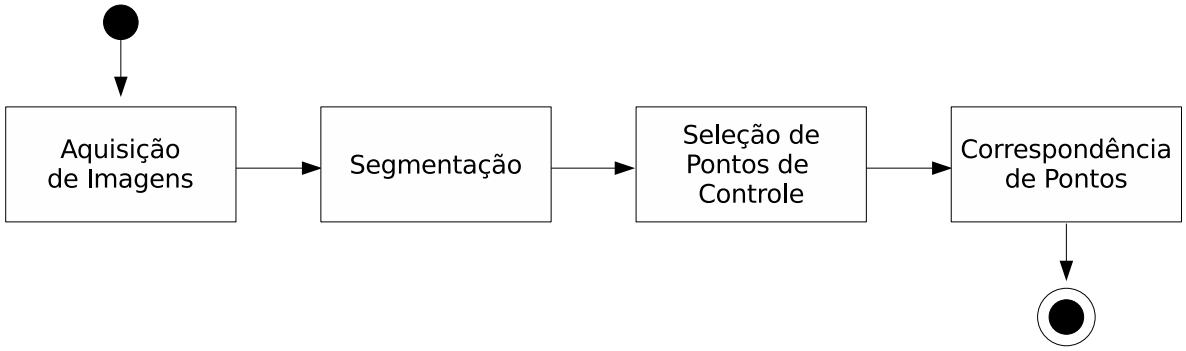


FIG. 4.15: Fluxograma do processamento das imagens

de localização. Este, por sua vez, deve calcular a localização do dirigível e o mapeamento do ambiente, através do método de decomposição em células; a partir dos pontos de interesse e segmentos de imagem informados pelo módulo de visão. Para tanto, o módulo de localização também recebe dados sobre a localização dos outros dirigíveis da equipe através do módulo de comunicação; que é responsável pela troca de mensagens entre os dirigíveis. Feito o cálculo da localização do dirigível e dos obstáculos, são enviados dados aos módulos de comunicação e de planejamento de trajetória. A FIG. 4.16 mostra o diagrama de classes dos módulos de visão e localização.

O módulo de planejamento de trajetória é o responsável por determinar qual caminho o dirigível deve seguir para conseguir chegar a um determinado ponto, para tanto é utilizada a metaheurística colônia de formigas; e deve atuar de forma cooperativa e dinâmica. Para trabalhar de forma cooperativa deve-se considerar não só dados oriundos

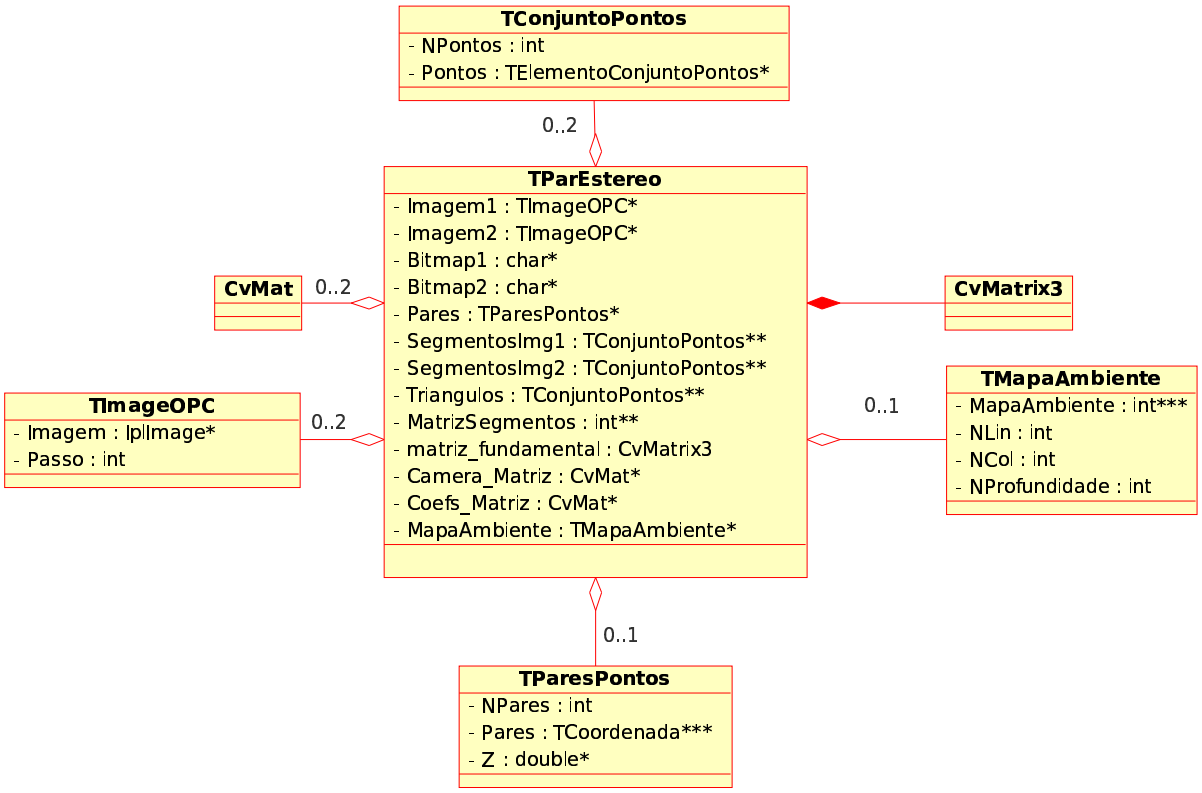


FIG. 4.16: Diagrama de classes dos módulos de visão e localização

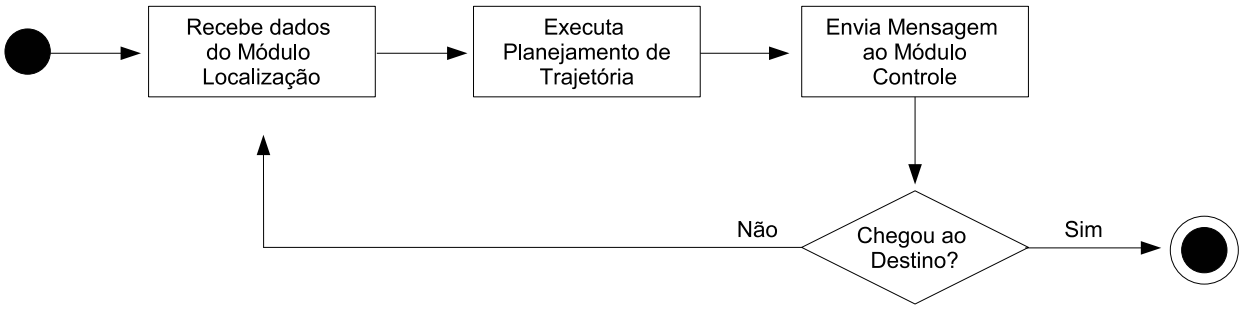


FIG. 4.17: Fluxograma do planejamento de trajetória

dos sensores do dirigível, mas também dados que possam ser adquiridos a partir de outros membros da equipe. Para tanto, os dirigíveis trocam mensagens a fim de comunicarem, uns aos outros, suas localizações, bem como a dos objetos que foram localizados através do processamento feito pelos módulos de visão e localização de cada um deles. Assim, pode-se evitar a colisão de dirigíveis e o mapeamento do ambiente torna-se mais completo. A partir de tais informações, o módulo de planejamento de trajetória executa o cálculo do caminho a ser percorrido pelo dirigível. Feito este cálculo, os dados devem ser enviados a um módulo de controle, que comandará os atuadores a fim de fazer com que o dirigível consiga movimentar-se conforme a trajetória especificada. Como os obstáculos contidos no ambiente onde o dirigível se encontra são móveis, o planejamento de trajetória deve ser feito continuamente. Portanto, enquanto o dirigível executa a trajetória planejada inicialmente, é necessário, simultaneamente, verificar o ambiente e ao longo do caminho replanejar a trajetória. Como é mostrado no fluxograma da FIG. 4.17. A FIG. 4.18 mostra o diagrama de classes do módulo de planejamento de trajetória.

O módulo de comunicação foi desenvolvido por (PINHEIRO, 2006) e é responsável pelo envio e recebimento de dados aos demais dirigíveis. Já o módulo de controle será concluído em uma etapa posterior e será responsável por comandar os atuadores do dirigível para que o mesmo movimente-se conforme uma determinada trajetória.

4.2 LINGUAGEM E FERRAMENTAS UTILIZADAS

Tendo em vista a idéia de utilizar o sistema de navegação em sistemas embarcados, este deve funcionar em tempo real, ou seja, com restrição de tempo; portanto, para a escolha da linguagem para implementação computacional, levou-se em conta os recursos

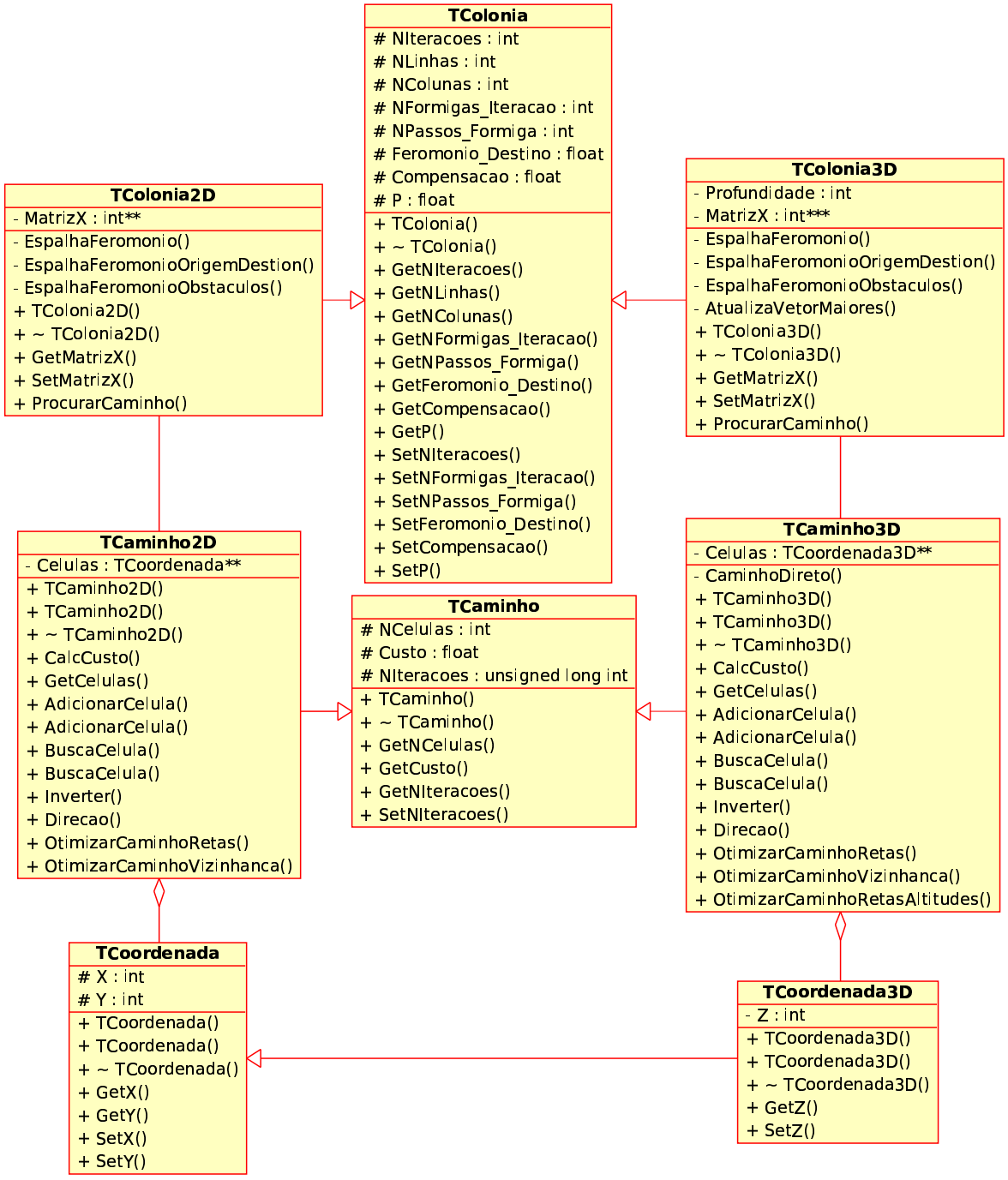


FIG. 4.18: Diagrama de classes do módulo de planejamento de trajetória

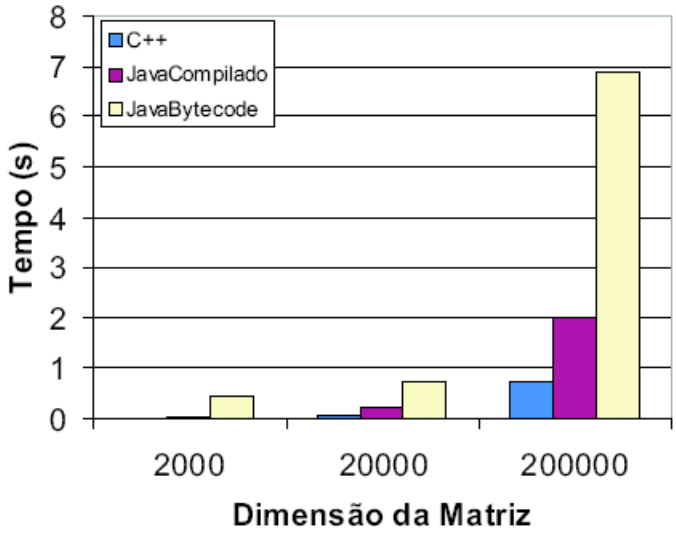


FIG. 4.19: Comparação entre os tempos de execução da resolução da equação de Laplace através do método iterativo do Gradiente Conjugado
(SCHEPKE et. al., 2004)

computacionais a serem alocados em tempo de execução. Então, tomou-se como ponto de partida o relatório técnico de (PRECHELT, 2000), no qual é feita uma comparação entre as linguagens C, C++, Java, Perl, Python, Rexx, e TCL em programas de busca e processamento de strings, com a utilização de algoritmos implementados por diversos programadores, em aspectos como o tempo de processamento e o uso de memória. O relatório aponta que programas escritos nas linguagens C e C++ usam uma quantidade menor de memória e gastam menor tempo de processamento quando estão em execução.

Um outro trabalho que aponta bom desempenho dos programas escritos em C++ é o de (SCHEPKE et. al., 2004), no qual é feita uma comparação entre as linguagens C++ e Java na computação numérica, testando-se programas para resolução da equação de Laplace através do método iterativo do Gradiente Conjugado, ressaltando as vantagens e desvantagens dessas linguagens no contexto da computação científica e de alto desempenho. Na fase de testes de desempenho, foram feitas três implementações: a primeira utiliza linguagem C++; a segunda, Java compilado; e a última, Java Bytecode. A comparação de desempenho entre as três implementações citadas é apresentada no gráfico da FIG. 4.19 e na TAB. 4.1, e mostra que a utilização da linguagem C++ é mais vantajosa para esta aplicação, e portanto, tal linguagem de programação foi adotada.

TAB. 4.1: Tempos de execução da resolução da equação de Laplace através do método iterativo do Gradiente Conjugado

(SCHEPKE et. al., 2004)

Implementação/Dimensão da Matriz	**2.000**	**20.000**	**200.000**
C++	0,06196s	0,76465s	8,14305s
Java Compilado	0,0908s	12,1857s	1.823,1154s
Java Bytecode	0,4898s	17,6046s	1.907,2011s

Segundo (LEE et. al., 2001), a linguagem C++ possui as seguintes capacidades técnicas:

- é uma linguagem portável, uma vez que existe o padrão ANSI (*American National Standard Institute*) para C++;
- produz programas rápidos, por não gastar tempo de execução com atividades do tipo "verificação e coleta de lixo", encontradas na maioria das linguagens "orientadas a objeto puras";
- não requer ambiente gráfico e tem um custo relativamente baixo de aquisição;
- permite, ao desenvolvedor, escrever código no nível apropriado para modelar uma solução particular, pois a linguagem é um casamento entre a linguagem Assembly, de baixo nível, e construções orientadas a objeto, de alto nível;
- por ser uma linguagem multiparadigmas, oferece, ao programador, uma gama de opções relativas ao projeto e codificação de uma solução.

Como plataforma de desenvolvimento do trabalho, foi escolhida a IDE (*Integrated Development Environment*) Borland C++ Builder 6 Personal Edition. Pois, a mesma possui interface gráfica bastante amigável e um compilador C++ de alto desempenho; é capaz de interagir com outras ferramentas e dispositivos; e é uma versão disponível para uso em ambientes acadêmicos.

Apesar de todos os recursos oferecidos pela IDE BCB (*Borland C++ Builder*) e linguagem C++, as mesmas não oferecem recursos específicos para gerar visualização gráfica em três dimensões. Como é necessário realizar um mapeamento tridimensional do ambiente e, conseqüentemente, a visualização do mesmo, foi necessário recorrer a outros meios. A solução encontra foi utilizar a OpenGL (*Open Graphics Library*), uma API (*Application*

Programming Interface) desenvolvida pela empresa Silicon Graphics em 1972.

OpenGL é uma interface de software para os dispositivos gráficos. A interface consiste em um conjunto de diversos procedimentos e funções que permitem, ao programador, especificar objetos e operações na produção de imagens gráficas de alta qualidade, especificamente imagens coloridas de objetos tridimensionais (SEGAL et. al., 2006).

Além das operações de computação gráfica para visualização tridimensional, outro ponto chave da implementação são os métodos de visão computacional. A IDE BCB oferece recursos para captura de imagens e processamento destas; contudo, estão disponíveis outras bibliotecas específicas de visão computacional para linguagem C++. Tendo em vista a escolha de um método que utilize poucos recursos computacionais, alvejando atender as restrições de tempo da problemática abordada, foi feito um teste simples de três soluções possíveis para definir qual delas seria adotada:

- implementação utilizando somente recursos da IDE BCB;
- implementação utilizando BCB e Vigra (KOTHE, 2000), uma biblioteca de visão computacional que obedece aos padrões da programação genérica;
- implementação utilizando BCB e OpenCV (*Open Computer Vision Library*) (INTEL, 2001), uma biblioteca para visão computacional, multiplataforma desenvolvida pela Intel.

O teste de desempenho realizado teve, como objetivo, medir a velocidade de processamento das metodologias citadas; para auxiliar a escolha da mais rápida entre as mesmas. Como a biblioteca desenvolvida pela Intel promete ter melhores resultados em máquinas com computadores Intel, foram eleitas duas plataformas de testes: um computador equipado com processador Intel e um outro equipado com um computador de outra marca, no caso, um AMD (*Advanced Micro Devices*); denominadas "*Plataforma Intel*" e "*Plataforma AMD*", respectivamente; vide especificações na TAB. 4.2.

Em cada plataforma de teste *vs.* metodologia, executou-se a operação de inversão de uma imagem RGB de 512x512 pixels, 300 vezes; medindo o tempo gasto em cada operação em ms. O resumo dos testes nas plataformas Intel e AMD são apresentados pelas TAB.

TAB. 4.2: Plataformas de testes de desempenho das metodologias de desenvolvimento de aplicações de visão computacional

	Processador	**Memória**	**Disco Rígido**
Plataforma AMD	AMD Atlhon XP 3000+ 2GHz	512MB	80GB 7200rpm
Plataforma Intel	Intel Pentium IV 2.8GHz	512MB	80GB 7200rpm

TAB. 4.3: Resumo dos testes de desempenho com a "*Plataforma Intel*"

	Maior Tempo	**Menor Tempo**	**Tempo Médio**	**Tempo Total**
BCB	123,359539ms	116,298326ms	117,599012ms	35279,599012ms
Vigra	171,430521ms	134,492746ms	163,184185ms	48955,255538ms
OpenCV	12,093570ms	9,091268ms	9,202234ms	2760,670292ms

4.3 e TAB. 4.4. A FIG. 4.20 mostra o resultado da operação de processamento de imagens implementada para o teste.

Como pode ser constatado, nos testes realizados, a biblioteca OpenCV teve melhor desempenho que as outras metodologias testadas e, como prometido, na plataforma Intel obteve melhores resultados. Portanto, a mesma foi escolhida para ser utilizada na implementação dos algoritmos relacionados a parte de visão computacional do sistema de navegação.

A seguir será explicado como as tarefas desempenhadas pelos módulos do sistema são executadas.

4.3 MAPEAMENTO DO AMBIENTE

O problema do mapeamento do ambiente consiste em determinar onde se encontram os obstáculos dentro do espaço de trabalho, para que a trajetória possa ser calculada; evitando colisões com os mesmos. Como a etapa de planejamento de trajetória é dividida em duas abordagens, o mesmo é feito com o mapeamento do ambiente. Assim, serão apresentadas duas soluções para esta parte do problema. Uma delas é o mapeamento bidimensional, e será utilizada para navegação em situações que a altitude do veículo seja praticamente constante. E a outra é o mapeamento tridimensional, e será utilizada nos casos onde se deseje que o dirigível tenha altitude de vôo variável.

TAB. 4.4: Resumo dos testes de desempenho com a "*Plataforma AMD*"

	Maior Tempo	**Menor Tempo**	**Tempo Médio**	**Tempo Total**
BCB	75,552641ms	58,772810ms	59,072390ms	17721,716864ms
Vigra	178,199323ms	128,689142ms	172,587563ms	51776,268867ms
OpenCV	27,981742ms	12,277798ms	12,493973ms	3748,3191938ms

FIG. 4.20: (a)Imagem original utilizada no teste; (b)Imagem original após operação de inversão

Ambas abordagens possuem pontos fortes e pontos fracos. O mapeamento bidimensional é mais simples de ser implementado e necessita de menor tempo de processamento para ser calculado. Porém, limita a navegação do veículo em um nível único de altitude de vôo; deste modo, podem-se obter resultados que não possibilitem que o sistema escolha uma trajetória que possua baixo custo de execução; além de limitar a área de atuação do veículo. Já o mapeamento tridimensional, por considerar as três dimensões do ambiente, possibilita que qualquer trajetória, possível de ser executada, seja utilizada. Porém, possui maior custo computacional na etapa de mapeamento; e também na etapa de planejamento de trajetória. Além disso, as trajetórias, oriundas de tal mapeamento, podem gerar maiores dificuldades, durante sua execução pelo veículo, por necessitarem de variações de altitudes.

Como este sistema de navegação se baseia em imagens, a tarefa de mapeamento deve ser feita através de visão computacional, a partir de imagens capturadas de câmeras em-

barcadas nos dirigíveis do sistema, durante o vôo. Para tanto, os dirigíveis devem possuir duas câmeras embarcadas. A partir das quais, serão executados algoritmos de visão estereoscópica; para que seja possível calcular a distância entre o veículo e os obstáculos detectados.

O mapeamento do ambiente é realizado pelos módulos de visão e localização; nesta seção serão abordadas as tarefas de calibração do par de câmeras estéreo, segmentação das imagens, triangulação de Delaunay, mapeamento de pontos no espaço e determinação do mapa do ambiente.

4.3.1 CALIBRAÇÃO DO PAR DE CÂMERAS ESTÉREO

A etapa de calibração das câmeras consiste em dois passos: o primeiro trata da correção da distorção radial, a qual é uma característica pertinente a qualquer lente de câmera, e faz com que a imagem capturada seja deformada em suas extremidades. O segundo se destina a calcular a matriz fundamental do par de câmeras estéreo.

A correção da distorção radial é realizada com o auxílio de métodos da biblioteca OpenCV, através dos quais é possível calcular os coeficientes k_1, k_2, p_1 e p_2 que determinam a distorção radial, bem como corrigí-la; obtendo-se uma nova imagem. Estes coeficientes são parâmetros intrínsecos da câmera; ou seja, uma vez adquiridos, podem ser utilizados em qualquer outra cena capturada pela mesma. Assim, todas as imagens capturadas e utilizadas no sistema de visão computacional sofrerão a correção da distorção radial.

Para realizar o cálculo dos coeficientes, é necessário, primeiramente, adquirir uma imagem que contenha quadrados brancos e pretos de tamanhos iguais, dispostos como em um tabuleiro de xadrez (vide FIG. 4.21). A partir da imagem em questão, são localizados os cantos internos do tabuleiro de xadrez utilizando-se os métodos cvFindChessboardCorners e cvFindCornerSubPix. Conhecendo-se a localização dos cantos internos na imagem, e também em relação a um referencial, localizado no canto superior esquerdo do tabuleiro de xadrez, e o tamanho dos quadros no mundo real, calculam-se os coeficientes k_1, k_2, p_1 e p_2 com o uso do método CalibrateCamera2 da biblioteca OpenCV.

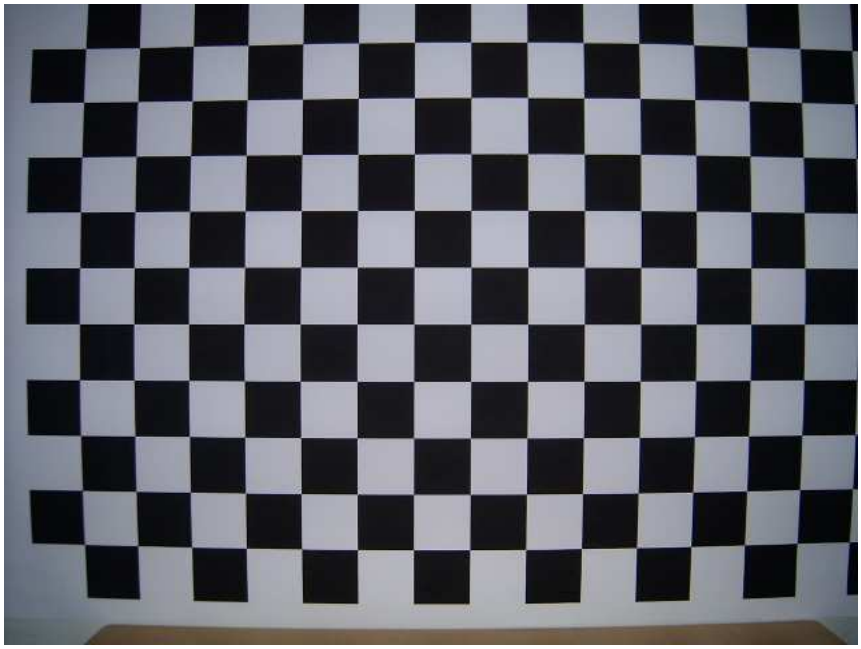


FIG. 4.21: Imagem utilizada para determinação dos coeficientes de distorção radial

A matriz fundamental, a exemplo dos coeficientes de distorção radial, também é calculada através de um método da biblioteca OpenCV. Para tanto, basta adquirir duas imagens de uma mesma cena a partir de pontos de vista distintos (no caso, as imagens utilizadas são oriundas de cada uma das câmeras do par estéreo) e localizar, pelo menos, 8 (oito) pontos em comum em cada uma das duas imagens. Feito isto, aplica-se o método FindFundamentalMat que calcula a matriz fundamental de um par estéreo.

4.3.2 SEGMENTAÇÃO DAS IMAGENS

Na operação de mapeamento do ambiente, o processo de segmentação é feito objetivando dividir a imagem em várias partes, onde cada uma delas seja um objeto ou parte de algum. O método empregado, para efetuar esta tarefa, foi o de crescimento de regiões, o qual consiste em, a partir de um *pixel* (usado como semente), verificar, na sua vizinhança, os *pixels* que são semelhantes ou não; os que forem semelhantes são considerados como pertencentes ao mesmo segmento da semente; e, partindo de cada um deles, o processo é repetido até que não se encontre nenhum pixel que tenha relação de semelhança. A relação de semelhança entre dois *pixels* é dada pela EQ. 4.1. O método de crescimento de regiões é aplicado levando-se em consideração várias sementes, espalhadas pela imagem

a ser segmentada. O número de sementes deve variar de acordo com a capacidade de processamento do *hardware* embarcado.

$$S(P_1, P_2, L) = \begin{cases} 0, \text{ se } |I(P_1) - I(P_2)| > L \\ \\ 1, \text{ se } |I(P_1) - I(P_2)| \leq L \end{cases} \tag{4.1}$$

onde:

- P_1 e P_2 são os dois *pixels* em questão;
- L é um limiar de tolerância;
- I é uma função que retorna o tom de cinza de um dado pixel.

Porém, devido a grande quantidade de ruído existente nas imagens e de variações de brilho na mesma, antes da aplicação do método de crescimento de regiões, é feito um pré-processamento da imagem; o qual consiste em uma operação de suavização que substitui o valor do pixel pela média encontrada em sua vizinhança, de forma a gerar uma imagem resultante com menor variação de cores. Para implementação deste filtro, foi utilizado o método cvSmooth da biblioteca OpenCV.

Após dividir a imagem em grupos de *pixels*, o próximo passo é determinar, para cada um deles, 8 (oito) pontos de controle, os quais são úteis para fazer uma aproximação do contorno de cada um dos segmentos da imagem e, também, para realizar o mapeamento do ambiente. Os pontos de controle são determinados conforme o pseudocódigo da FIG. 4.22.

4.3.3 TRIANGULAÇÃO DE DELAUNAY

A partir dos oito pontos de controle, estabelecidos no contorno dos segmentos, é preciso determinar um fecho convexo deste conjunto de pontos, para que o segmento da imagem fique realmente definido. Além disso, é necessário estabelecer triângulos dentro de tal fecho convexo, de modo viabilizar a determinação do mapa do ambiente. Cada triângulo interno define uma face de um dado segmento, e, através destas faces, é feita uma aproximação do formato do mesmo.

```
algoritmo pontos_controle(XMin,YMin, XMax,YMax)
        pontos[1][1] ← XMin
        pontos[1][2] ← YMin
        enquanto(pertence_ao_contorno(pontos[1][1],  pontos[1][2]) = false)
                pontos[1][1]++
                pontos[1][2]++
        fim_enquanto

        pontos[2][1] ← media(XMin, Xmax)
        pontos[2][2] ← XMin
        enquanto(pertence_ao_contorno(pontos[2][1] , pontos[2][2] ) = false)
                pontos[2][2]++
        fim_enquanto

        pontos[3][1] ← media(XMin, Xmax)
        pontos[3][2] ← XMax
        enquanto(pertence_ao_contorno(pontos[3][1], pontos[3][2]) = false)
                pontos[3][1]--
                pontos[3][2]++
        fim_enquanto

        pontos[4][1] ← XMin
        pontos[4][2] ← media(YMin, YMax)
        enquanto(pertence_ao_contorno(pontos[4][1], pontos[4][2]) = false)
                pontos[4][1]++
        fim_enquanto

        pontos[5][1] ← XMax
        pontos[5][2] ← media(YMin, YMax)
        enquanto(pertence_ao_contorno(pontos[5][1], pontos[5][2]) = false)
                pontos[5][1]--
        fim_enquanto

        pontos[6][1] ← XMin
        pontos[6][2] ← YMax
        enquanto(pertence_ao_contorno(pontos[6][1], pontos[6][2]) = false)
                pontos[6][1]++
                pontos[6][2]--
        fim_enquanto

        pontos[7][1] ← media(XMin, YMin)
        pontos[7][2] ← YMax
        enquanto(pertence_ao_contorno(pontos[7][1], pontos[7][2]) = false)
                pontos[7][2]--
        fim_enquanto

        pontos[8][1] ← XMax
        pontos[8][2] ← YMax
        enquanto(pertence_ao_contorno(pontos[8][1], pontos[8][2]) = false)
                pontos[8][1]--
                pontos[8][2]--
        fim_enquanto

        retornar(pontos)
fim algoritmo
```

FIG. 4.22: Pseudocódigo para determinação dos pontos de controle de um segmento de imagem

```
algoritmo triangulacao(Matriz_Arestas, Vertices, NVertices)
        Ntriangulos ← 0
        para (k = 1 até Nvertices – 2)
                para (l = k + 1 até Nvertices - 1)
                        se (Matriz_Arestas[k][l] = 1)
                                para (m = l+ 1 até  NVertices)
                                        se (Matriz_Arestas[k][m] = 1 e Matriz_Arestas[l][m] = 1)
                                                NTriangulos++
                                                Triangulos[NTriangulos][1] = Vertices[k]
                                                Triangulos[NTriangulos][2] = Vertices[l]
                                                Triangulos[NTriangulos][3] = Vertices[m]
                                        fim_se
                                fim_para
                        fim_se
                fim_para
        fim_para

        retorna(Triangulos);
fim algoritmo
```

FIG. 4.23: Pseudocódigo para determinação de triângulos que formam um fecho convexo, dado um conjunto de vértices e de arestas

Para a determinação dos triângulos internos, é utilizada a Triangulação de Delaunay. Este método determina um conjunto de triângulos que formam um fecho convexo, dado um conjunto de pontos em um plano. Cada triângulo de Delaunay determina um círculo que não contém nenhum outro ponto em seu interior, além disso suas respectivas arestas não se cruzam com nenhum outro triângulo do conjunto. Em (LEE et. al., 1980) encontram-se dois algoritmos para implementação do método de Delaunay. Porém, devido ao bom desempenho que a biblioteca OpenCV mostrou nos testes, e para facilitar a implementação, optou-se por utilizar seus algoritmos. Através dos métodos CreateSubdivDelaunay2D e SubdivDelaunay2DInsert, é possível calcular quais são as arestas que formam um conjunto de triângulos internos de um fecho convexo, dado um conjunto de pontos. Entretanto, os vértices dos triângulos internos não são determinados.

Assim, foi necessário implementar um algoritmo que, dado um conjunto de arestas, determine quais os vértices dos triângulos. Tal algoritmo funciona conforme o pseudocódigo da FIG. 4.23.

4.3.4 MAPEAMENTO DE PONTOS NO ESPAÇO

O mapeamento de pontos no espaço consiste em determinar as coordenadas dos pontos de controle, dos contornos dos segmentos das imagens, em relação a um referencial; que,

no caso, é o sistema de origem do par de câmeras estéreo, que localiza-se no centro da *baseline*. Entretanto, as coordenadas adquiridas pelo sistema de visão são dadas em *pixels*, e o cálculo das coordenadas, por meio de visão estereoscópica, deve ser feito com as coordenadas dos pontos no plano da imagem. Portanto, faz-se necessário efetuar uma transformação oriunda do Teorema Fundmental da Geometria Projetiva, a qual determina a localização de um *pixel* no plano da imagem. Tal transformação projetiva é dada da seguinte forma.

Dada a transformação projetiva T:

$$T(u,v,1) = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} ax+by+c & dx+ey+f & gx+hy+1 \end{bmatrix} \quad (4.2)$$

onde:

- (u, v) corresponde a um ponto da imagem dado em unidade de *pixel*;
- a, b, c, d, e, f, g e h são dados pela resolução do sistema linear da EQ. 4.3.

$$\begin{bmatrix} u_0 & v_0 & 1 & 0 & 0 & 0 & -u_0x_0 & -v_0x_0 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -u_1x_1 & -v_1x_1 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -u_2x_2 & -v_2x_2 \\ u_3 & v_3 & 1 & 0 & 0 & 0 & -u_3x_3 & -v_3x_3 \\ 0 & 0 & 0 & u_0 & v_0 & 1 & -u_0y_0 & -v_0y_0 \\ 0 & 0 & 0 & u_1 & v_1 & 1 & -u_1y_1 & -v_1y_1 \\ 0 & 0 & 0 & u_2 & v_2 & 1 & -u_2y_2 & -v_2y_2 \\ 0 & 0 & 0 & u_3 & v_3 & 1 & -u_3y_3 & -v_3y_3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad (4.3)$$

onde

- (u_i, v_i) é um ponto da imagem (dado em unidade de *pixel*) $\forall\ i$, tal que $1 \leqslant i \leqslant 4$ e $i \in \aleph$;
- (x_i, y_i) é um ponto no plano da imagem (dado em mm) $\forall\ i$, tal que $1 \leqslant i \leqslant 4$ e $i \in \aleph$.

O ponto $(x, y, 1)$ (dado em mm) no plano da imagem, correspondente ao ponto $(x, y, 1)$ (dado em unidade de *pixel*) no plano da imagem, é dado pela EQ. 4.4.

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} ax+by+c & dx+ey+f & gx+hy+1 \end{bmatrix} / (gx+hy+1) \quad (4.4)$$

Assim, para determinar as coordenadas x, y e z de um ponto no espaço, tendo como sistema de referência a origem do sistema do par de câmeras estéreo; calcula-se a projeção de tal ponto no plano de imagem de cada uma das câmeras, e, posteriormente, aplicam-se EQ. 3.29, EQ. 3.30 e EQ. 3.31, respectivamente. Em seguida é apresentado o método proposto para a determinação do mapa do ambiente.

4.3.5 DETERMINAÇÃO DO MAPA DO AMBIENTE

Esta é a última etapa a ser concluída, antes de se executar o cálculo do planejamento de trajetória, e deve ser feita a partir dos triângulos obtidos com o uso da triangulação de Delaunay.

Para determinar o mapa do ambiente, é utilizada a técnica de decomposição em células aproximada; levando-se em consideração o espaço tridimensional, divide-se o mesmo em cubos, cujas dimensões são idênticas e, parametrizadas de acordo com a necessidade da aplicação, e a capacidade de processamento do hardware embarcado. Desta forma, para cada cubo, atribui-se um valor binário que pode ser "ocupado" ou "não-ocupado".

Assim, para cada triângulo encontrado, durante a triangulação de Delaunay, são feitos os seguintes passos:

a) Determinar as coordenadas 3D de cada vértice, como descrito na seção 4.3.4;

b) Determinar uma caixa tridimensional fechada, que envolva o triângulo. Tal caixa possui, como diagonal interna, o segmento de reta entre os pontos $P1$ e $P2$, onde:

$$P1_x = min(V1_x, V2_x, V3_x) \tag{4.5}$$

$$P1_y = min(V1_y, V2_y, V3_y) \tag{4.6}$$

$$P1_z = min(V1_z, V2_z, V3_z) \tag{4.7}$$

$$P2_x = max(V1_x, V2_x, V3_x) \tag{4.8}$$

$$P2_y = max(V1_y, V2_y, V3_y) \tag{4.9}$$

$$P3_z = max(V1_z, V2_z, V3_z) \tag{4.10}$$

onde, $V1$, $V2$ e $V3$ são os vértices do triângulo;

```
algoritmo processa_triangulo(P1x, P1y, P1z,  P2x, P2y, P2z, NLin, NCol, Profundidade,
                                                        tam_aresta_cubo, Matriz_Cubos)
    para i = P1y div tam_aresta_cubo até i <= P2y div tam_aresta_cubo
        A1y ← i * tam_aresta_cubo
        A2y ← A1y + tam_aresta_cubo
        para j = P1x div tam_aresta_cubo até j <= P2x div tam_aresta_cubo
            A1x ← i * tam_aresta_cubo
            A2x ← A1x + tam_aresta_cubo
            para k = P1z div tam_aresta_cubo até k <= P2z div tam_aresta_cubo
                A1z ← i * tam_aresta_cubo
                A2z ← A1z + tam_aresta_cubo
                se ( A1x <= P1x e (A2x >= P2x ou
                    (A2x >= P1x e A2 <= P2))) ou
                    ((A1x >= P1x) e A2x >= P2x) ou
                    ( A1y <= P1y e (A2y >= P2y ou
                    (A2y >= P1y e A2 <= P2))) ou
                    ((A1y >= P1y) e A2y >= P2y) ou
                    ( A1z <= P1z e (A2z >= P2z ou
                    (A2z >= P1z e A2 <= P2))) ou
                    ((A1z >= P1z) e A2z >= P2z)
                        Matriz_Cubos[i][j][k] = 1
fim algoritmo
```

FIG. 4.24: Pseudocódigo para determinação do mapa do ambiente

c) Atribuir valor "ocupado" para os cubos que possuem alguma interseção com a caixa tridimensional fechada que envolve o triângulo. Isto é feito seguindo o pseudocódigo mostrado na FIG. 4.24.

4.4 PLANEJAMENTO DE TRAJETÓRIA

O planejamento de trajetória do sistema de navegação, proposto neste trabalho, tenta otimizar uma função objetivo que, dependendo da situação em que o veículo se encontra, pode ser: (a) o tempo gasto para percorrer o caminho; (b) o comprimento do caminho; ou (c) o consumo de bateria ou combustível pelo veículo. Devido às possibilidades de existirem obstáculos móveis e de surgirem novos obstáculos durante a execução da trajetória, a solução para este problema deve ser dividida em duas etapas: (a) planejamento de trajetória global; e (b) planejamento de trajetória local. No primeiro caso, um caminho é determinado partindo do ponto de origem e chegando ao ponto objetivo. Já no segundo caso, o cálculo da trajetória é realizado, continuamente, enquanto o veículo percorre o caminho determinado na primeira etapa do processo; determinando o caminho a partir

da posição atual do veículo até a posição objetivo. Assim, o caminho obtido inicialmente pode ser modificado conforme as mudanças ocorridas na configuração do espaço de trabalho, para que sejam evitadas colisões do veículo com obstáculos que tenham se movido ou que tenham surgido no ambiente. São seguidas duas abordagens para o planejamento de trajetória deste sistema de navegação: a primeira é destinada a espaços de trabalho bidimensional e a segunda é voltada a espaços de trabalhos tridimensionais. Os próximos itens desta seção irão tratar estas abordagens.

4.4.1 PLANEJAMENTO DE TRAJETÓRIA NO ESPAÇO DE TRABALHO BIDIMENSIONAL

Esta primeira abordagem foi desenvolvida devido à existência de situações nas quais seja preferível que os dirigíveis atuem em altitude praticamente constante, como nos casos de vigilância e monitoração de ambientes. Assim, foi considerado um espaço de trabalho bidimensional, o qual é divido, usando o método de decomposição em células; uma abordagem sobre este método é encontrada no capítulo 3 ou em (LATOMBE, 1991). Desta forma, divide-se o ambiente de trabalho em células retangulares e de tamanhos iguais. As células são distribuídas em n colunas (coordenadas do eixo x) e n linhas (coordenadas do eixo y) formando um mapa geográfico de forma matricial. Este mapa é representado matematicamente por uma matriz binária quadrada. As células, onde se encontra alguma parte de um obstáculo qualquer, são representadas pelo valor 1 e as demais representadas pelo valor 0, conforme FIG. 4.25. Após realizar a representação do ambiente através de uma matriz binária, o próximo passo é determinar qual caminho a ser percorrido; onde um caminho é uma seqüência de células adjacentes não-ocupadas por obstáculos. Este processo é descrito nos próximos parágrafos.

Como um dos requisitos deste trabalho é atuar em um ambiente dinâmico e, para que isso seja possível, é necessário que o planejamento de trajetória seja feito de forma contínua; de modo que sejam evitadas possíveis colisões do dirigível, com obstáculos móveis ou com novos obstáculos que venham a aparecer durante a execução da trajetória. Surge, a partir desta necessidade, uma restrição de tempo. Portanto, utiliza-se um processo heurístico para determinar o caminho a ser seguido, pois tentar encontrar sempre a melhor solução, de forma a esgotar todas a possibilidades possíveis, pode requerer um grande custo computacional e tempo de processamento inviável, de acordo com as di-

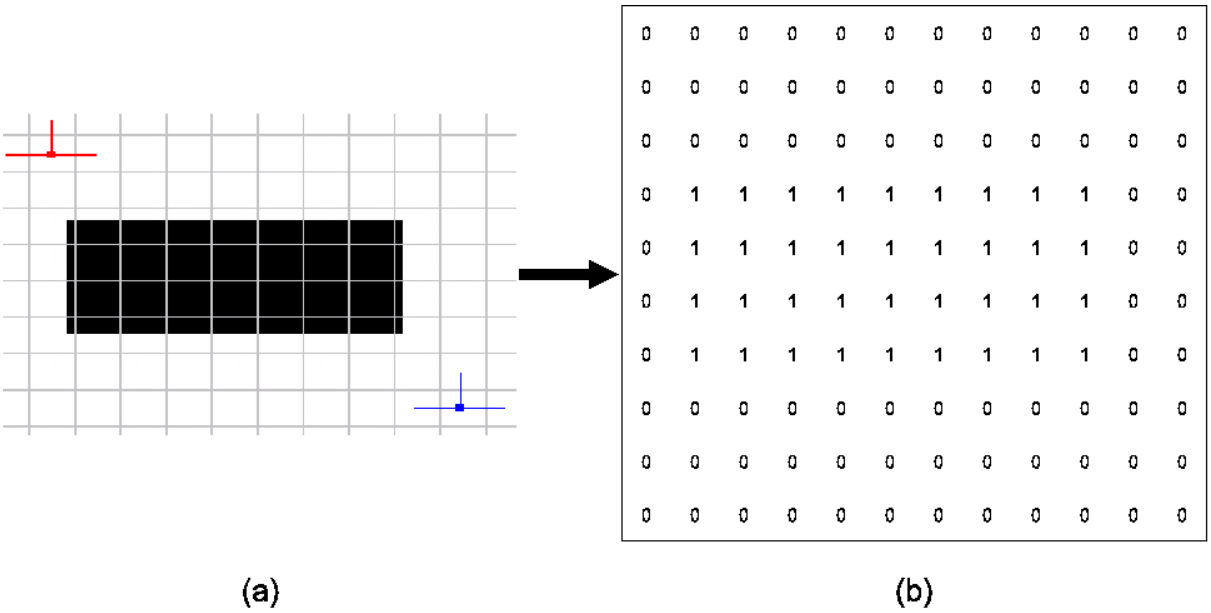


FIG. 4.25: (a) Mapeamento bidimensional do ambiente. (b) Representação matemática através de uma matriz binária

mensões do mapa do ambiente. Assim, usa-se a metaheurística Colônia de Formigas (DORIGO et. al., 1999), com a particularidade de depositar o feromônio em cada célula do mapa do espaço de trabalho. O objetivo é fazer com que as formigas artificiais sigam na direção das células mais próximas do objetivo.

Para que o algoritmo encontre uma solução mais rapidamente, antes da primeira iteração, são feitas as seguintes trilhas iniciais de feromônio (vide FIG. 4.26 e FIG. 4.27) que correspondem às células que estão:

- nas linhas do ponto de origem;
- nas linhas do ponto de destino;
- nas colunas do ponto de origem;
- nas colunas do ponto de destino;
- nas diagonais do ponto de origem;
- nas diagonais do ponto de destino;

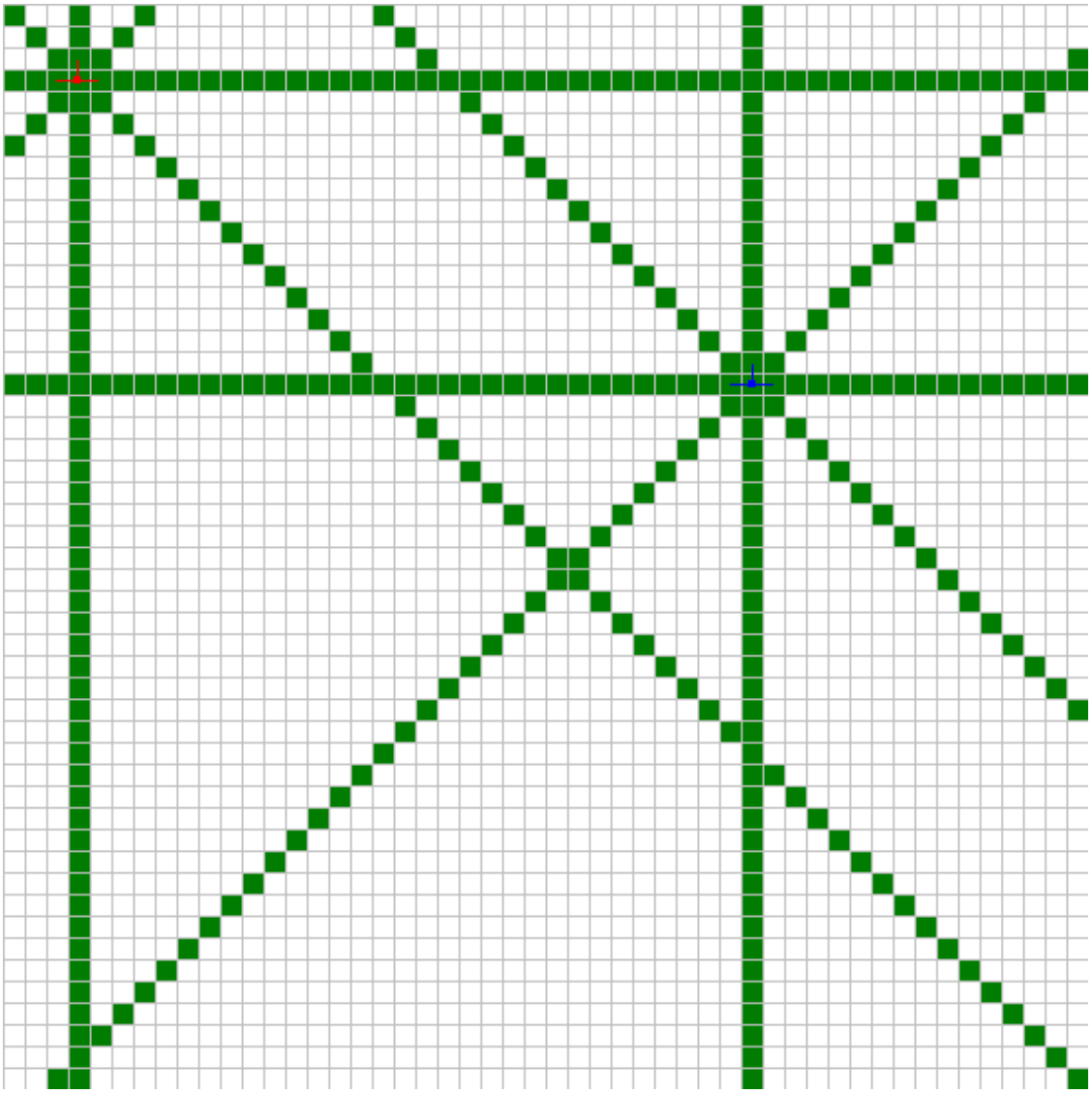


FIG. 4.26: Trilhas iniciais de feromônio em um exemplo sem obstáculos

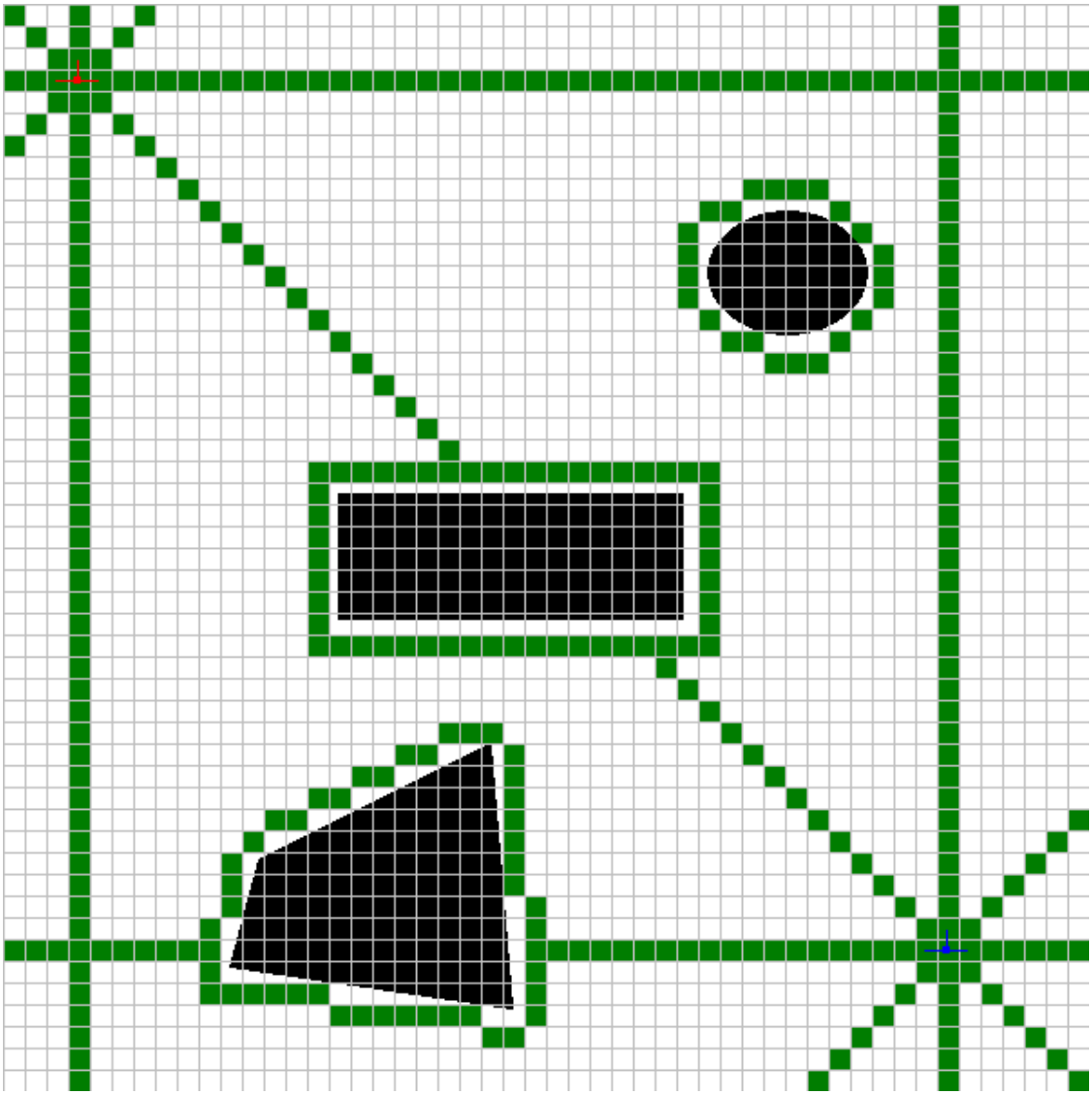


FIG. 4.27: Trilhas iniciais de feromônio em um exemplo com obstáculos

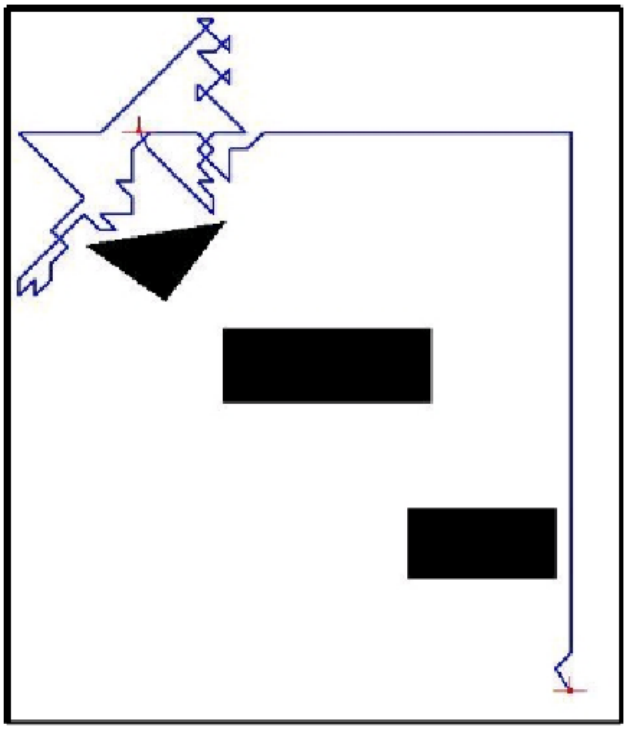


FIG. 4.28: Caminho gerado pela metaheurística Colônia de Formigas.

- na vizinhança dos obstáculos.
- em um caminho ótimo, o qual é traçado previamente desconsiderando a presença de obstáculos;

Em seguida, o processo da metaheurística Colônia de Formigas é disparado. A cada iteração, de forma alternada, metade das formigas tenta ir do ponto de saída ao ponto de chegada e a outra metade tenta fazer o caminho inverso. É estabelecido um número máximo de passos que uma formiga pode efetuar. Caso ela chegue a este número máximo, a mesma é desconsiderada. Uma outra restrição é que as formigas não podem passar por uma mesma célula mais de uma vez, ou seja, o caminho não pode ter células repetidas. Infelizmente, os caminhos gerados pela aplicação da metaheurística Colônia de Formigas, geralmente não são uma boa solução, principalmente quando existem obstáculos entre os pontos de origem e destino. Vide ilustração da FIG. 4.28. Porém, observou-se que, por meio de otimizações, as soluções poderiam ser melhoradas. Então, após a obtenção de cada caminho gerado no processo, são feitas duas otimizações: (a) otimização na vizinhança; (b) otimização nas retas. A seguir, estas otimizações serão descritas.

Nos caminhos gerados inicialmente, notou-se que uma mesma célula possuíra mais de um vizinho pertencente ao caminho. Assim, ao passar por uma célula qualquer, pode-se seguir para um de seus vizinhos, passar por outras células, e depois chegar a um outro vizinho da mesma célula. Portanto, o caminho possui percursos desnecessários. Para resolver este problema, foi feita a otimização na vizinhança, que consiste em verificar se cada célula possui mais de um vizinho pertencente ao caminho. Em caso afirmativo, escolhe-se o vizinho correspondente ao ponto mais adiante do caminho como próxima célula. Assim, os outros vizinhos e todas as outras células por onde o caminho iria passar, de forma desnecessária, são eliminados.

Após a execução da otimização na vizinhança, nota-se que o caminho gerado ainda pode ser melhorado. O motivo é que ele tende a dar voltas. A otimização nas retas tenta eliminar estas voltas deixadas pela otimização na vizinhança. Esta operação é feita da seguinte forma: cria-se um novo caminho, no qual a primeira célula coincide com a primeira célula do caminho original; e, a célula seguinte é a vizinha na direção em que se encontra a célula mais adiante pertencente ao caminho original, e que seja alcançável em linha reta (vertical, horizontal ou diagonal), ou seja, que esteja na mesma linha, coluna ou diagonal e com o percurso livre de obstáculo(s). Após o cálculo da segunda célula, determinam-se as células seguintes, com o mesmo processo, até encontrar a última célula do caminho. A FIG. 4.29 mostra um caminho gerado e otimizado pelos dois processos já citados. Note que ele possui uma trajetória bem mais "comportada", ou seja, possui menos curvas. Como citado anteriormente, o algoritmo deve funcionar de forma dinâmica; assim o dirigível, em sua trajetória, deve desviar-se de obstáculos móveis, bem como de novos obstáculos que venham a surgir durante a execução da mesma. A FIG. 4.30 mostra uma seqüência de adaptações do caminho durante o seguimento da trajetória.

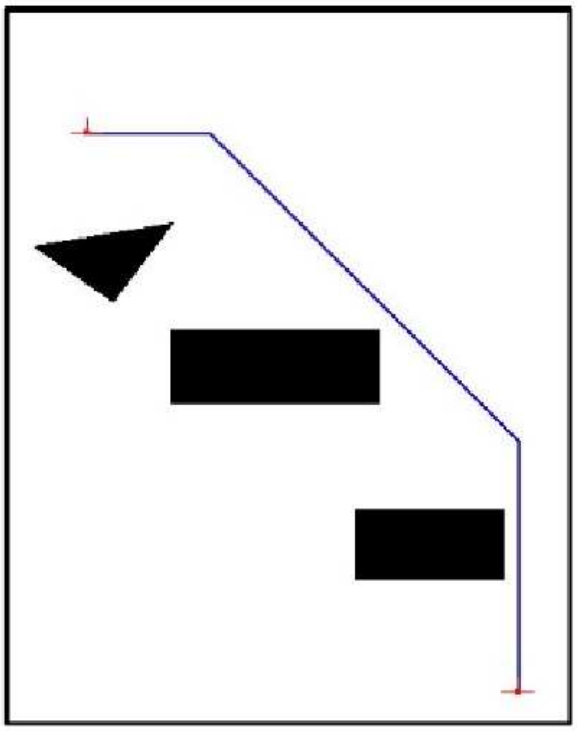


FIG. 4.29: Caminho da FIG. 4.28 após as otimizações na vizinhança e nas retas

Seguindo a abordagem dada nos parágrafos anteriores, observa-se que pode-se utilizar uma matriz (com valores 0 ou 1) para representar o mapa do ambiente, e uma outra (com valores no intervalo $[0.. + \infty]$) para representar a quantidade de feromônio depositada em cada célula. Porém, em se tratando de um sistema embarcado, com restrição de tempo, e para melhor aproveitamento dos recursos computacionais (mas especificamente da memória principal), foi usada somente uma matriz. Nesta, as células que estão ocupadas são iniciadas com o valor -1 e não são modificadas durante todo o processo; já as células que estão livres de obstáculos são iniciadas com o valor 0 e podem ser modificadas no início (construção das trilhas iniciais de feromônio) e/ou no decorrer do processo (feromônio depositado pelas formigas artificiais). Assim, como as formigas sempre selecionam como próxima célula a ser seguida, uma célula escolhida (através de um sorteio) entre suas vizinhas com maior quantidade de feromônio, a restrição de que um caminho não pode possuir uma célula ocupada é respeitada. Através da FIG. 4.31, é mostrado o pseudocódigo do algoritmo de determinação de um caminho no espaço de trabalho com duas dimensões.

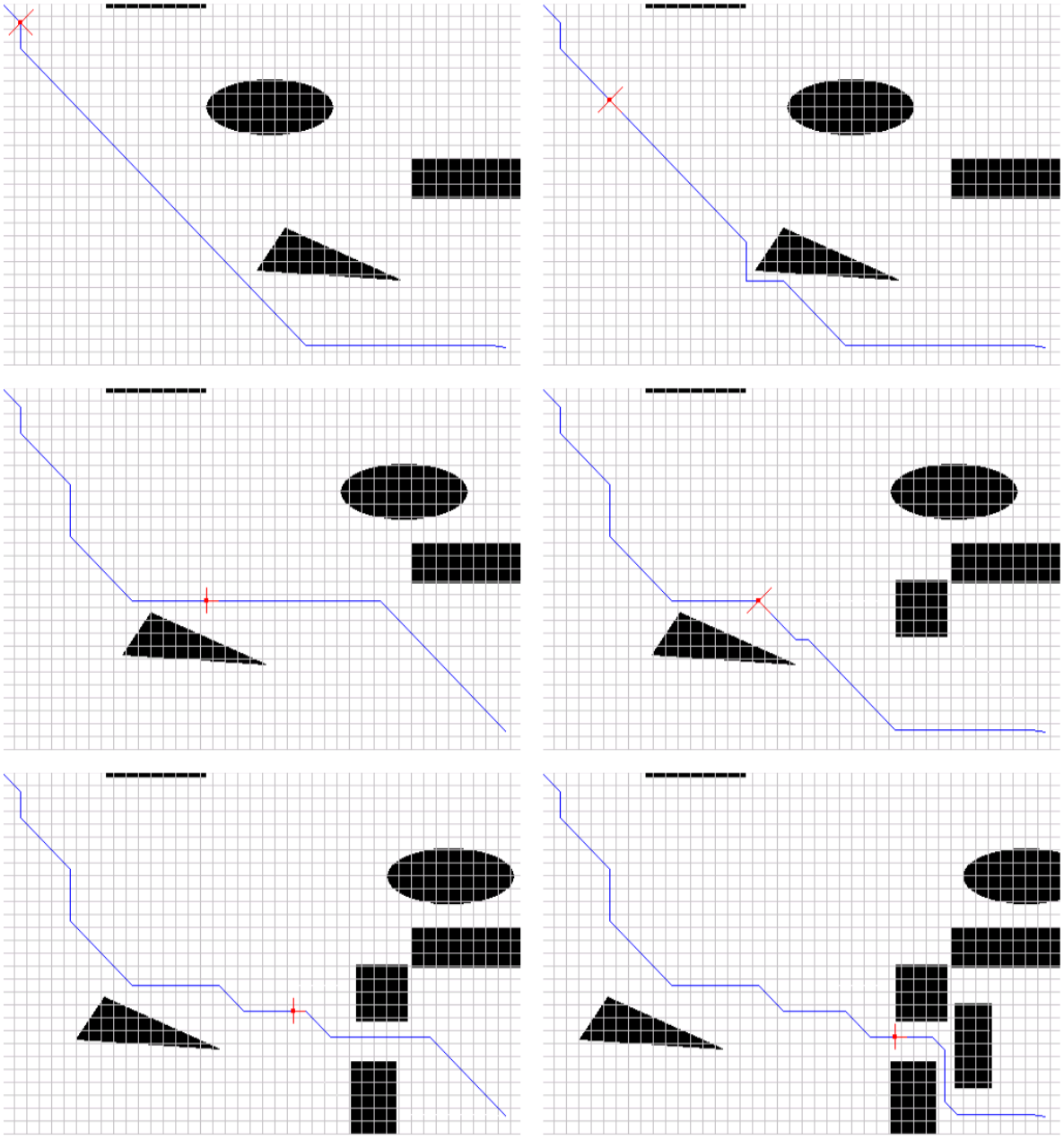


FIG. 4.30: Seqüência de adaptações da trajetória

```
algoritmo procurar_caminho2D(XOrigem, YOrigem, XDestino, YDestino, NIteracoes, NPassosFormiga, NFormigasIteracao,
                                                                                    MatrizMapaF, CustoMaximo)
fazer_trilhas_feromonios(XOrigem, YOrigem,  XDestino, YDestino, MatrizMapaFT)
sair ← 0
n_celulas_caminho ← 0
n_caminhos ← 0
enquanto(i < NIteracoes e sair = 0)
i++
                                    MatrizMapaF ← MatrizMapaFT;
para j = 0 até j < NFormigas_Iteracao
se sair = 1
parar
fim se
se (resto(j, 2) <> 0)
XObjetivo ← XOrigem
                                    YObjetivo ← YOrigem
                                                                x ← XDestino
                                                                y ← YDestino
fim se
senao
x ← XOrigem
y ← YOrigem
XObjetivo ← XDestino
YObjetivo ← YDestino
fim senao
n_celulas_caminho ← 0
enquanto ((x <> x_objetivo ou y <> y_objetivo) e n_celulas_caminho < NPassosFormiga)
escolhe_proxima_celula_caminho(x, y, MatrizMapaF)
MatrizMapaFT[x][y] ←  MatrizMapaFT[x][y] + 0.5
caminho[j][0] ← x
caminho[j][1] ← y
n_celulas_caminho ← n_celulas_caminho + 1;
fim enquanto
se  (x = XObjetivo e y = YObjetivo)
n_caminhos ← n_caminhos + 1
otimizacao_retas(caminho, n_celulas_caminho, MatrizMapaF)
otimizacao_vizinhanca(caminho, n_celulas_caminho)
se n_caminhos = 1
                                                                melhor_caminho ← caminho
                                                                n_celulas_melhor_caminho ← n_celulas_caminho
fim se
senao se custo(caminho, n_celulas_caminho) < custo(melhor_caminho,
                                    n_celulas_melhor_caminho)
                                                                melhor_caminho ← caminho
                                                                n_celulas_melhor_caminho ← n_celulas_caminho
fim senao se
depositar_feromonio(caminho, MatrizMapaFT)
se custo(caminho, n_celulas_caminho) <= CustoMaximo
sair ← 1
fim se
fim se
fim para
evaporar_feromonio(MatrizMapaFT)
fim enquanto
retornar(melhor_caminho)
fim algoritmo
```

FIG. 4.31: Pseudocódigo do algoritmo aproximado para determinação de um caminho próximo do ótimo no espaço de trabalho bidimensional

Utilizando a metaheurística Colônia de Formigas, são encontrados diversos caminhos distintos, do ponto de origem ao ponto de destino. De posse dessa informação, o caminho, a ser seguido, é escolhido de maneira tal que otimize uma função objetivo; que define o custo do caminho, definido a seguir.

Em um sistema de navegação, dependendo da operação ou das condições do veículo pode-se tentar otimizar vários fatores pertinentes ao mesmo, como consumo de bateria ou combustível, distância a ser percorrida ou o tempo de viagem. Neste caso específico, tenta-se minimizar o tempo gasto para percorrer a trajetória. Portanto, a função objetivo, aqui utilizada, retorna o custo de um caminho. Este é o valor correspondente à estimativa do tempo que o dirigível gastaria para sair do ponto de partida, em estado de repouso, e chegar ao ponto destino, de forma a voltar ao repouso. Chamaremos esse tempo de "*tempo de viagem*". Para estimar o *tempo de viagem*, são considerados os seguintes parâmetros relativos à cinemática do dirigível:

- tamanho das células;
- velocidade de cruzeiro (velocidade máxima que pode ser alcançada pelo dirigível);
- aceleração de cruzeiro;
- tempos de rotação (tempos médios gastos pelo dirigível para realizar as rotações de 45°, 90°, 135° e 180°).

O caminho gerado é formado por uma seqüência de segmentos de retas, unidos por suas extremidades. Para calcular o custo de um caminho, primeiro determina-se o tempo que o dirigível levaria para ir de uma extremidade a outra de cada segmento de reta que forma o caminho. Em seguida é feito o somatório dos tempos gastos para fazer a rotação de cada transição de um segmento de reta para o próximo, no trajeto traçado. Para finalizar, são adicionados os tempos gastos para rotacionar o dirigível de sua pose inicial para a direção do primeiro segmento de reta, chamada de rotação inicial; e para rotacionar o dirigível da direção do último segmento de reta para a orientação de sua pose final. O custo total do percurso é a soma dos tempos de viagem de cada segmento de reta adicionados ao somatório do tempo total necessário para efetuar todas as rotações durante o percurso, incluindo as rotações inicial e final. Assim, podemos calcular o custo de um caminho através da EQ. 4.13 que, apesar de não retornar, precisamente, o tempo a ser

gasto pelo dirigível para percorrer um dado caminho, satisfaz as condições do problema. Pois, para resolvê-lo, não é necessário mensurar o custo exato dos caminhos, mas apenas classificá-los.

$$C_{2D} = [\sum_{i=1}^{ns} Ts(v, a, S_i) + \sum_{i=1}^{nr} G(R_i)] + G(ri) + G(rf) \tag{4.11}$$

onde:

- ns: número de segmentos de reta que formam o caminho;
- Ts: função que retorna o tempo necessário para percorrer um segmento de reta, definida na EQ. 4.12;
- v: velocidade de cruzeiro;
- a: aceleração de cruzeiro;
- S: vetor que contém os segmentos de reta que formam o caminho;
- nr: número de rotações entre os segmentos de reta que compõem o caminho;
- G: função que retorna o tempo gasto em determinada rotação;
- R: vetor que contém o número de graus referentes às rotações realizadas durante o percurso;
- ri: número de graus referente à rotação inicial;
- rf: número de graus referente a rotação final, após chegar ao objetivo.

$$Ts(v, a, c) = \begin{cases} 2 \times \dfrac{\sqrt{\dfrac{v^2}{2a}}}{a} + c - \dfrac{v}{a}, \text{se } c > \dfrac{v^2}{2a} \\ 2 \times \sqrt{\dfrac{c}{a}}, \text{se } c \leq \dfrac{v^2}{2a} \end{cases} \tag{4.12}$$

onde:

- v: velocidade de cruzeiro;
- a: aceleração de cruzeiro;
- c: é o comprimento do segmento.

4.4.2 PLANEJAMENTO DE TRAJETÓRIA NO ESPAÇO DE TRABALHO TRIDIMENSIONAL

Visando situações mais gerais, é feita esta abordagem, a ser aplicada em casos nos quais os dirigíveis necessitem fazer, durante suas trajetórias, mudanças de altitude, a fim de alcançar seus objetivos. Portanto, é proposto um método de planejamento de trajetória para o espaço tridimensional. A exemplo do planejamento de trajetória no espaço bidimensional, também foram utilizados o método de decomposição em células para efetuar o mapeamento do ambiente, e a metaheurística Colônia de Formigas para gerar a trajetória a ser seguida.

Para realizar o mapeamento do ambiente, divide-se o mesmo em um conjunto de células em formato cúbico e de dimensões iguais. Tais células são distribuídas em n colunas (coordenadas do eixo x), n linhas (coordenadas do eixo y), e n níveis de profundidade (coordenadas do eixo z); e atribui-se a cada célula um valor binário (0 ou 1). Assim, matematicamente, forma-se um mapa do ambiente representado por uma matriz binária cúbica, onde as células com valor igual a 1 são dadas como ocupadas e as células com valor igual 0 são dadas como livres. Desta forma, tem-se um mapa aproximado do ambiente, e a partir do mesmo é realizado o planejamento de trajetória, o qual é descrito nos próximos parágrafos.

A aplicação da metaheurística Colônia de Formigas, ao problema de planejamento de trajetória para o espaço de trabalho tridimensional, possui a finalidade de encontrar uma seqüência de células livres e adjacentes, de um ponto de partida a um determinado ponto objetivo. Faz-se isso, visando minimizar uma função custo que, neste caso, corresponde ao *tempo de viagem* gasto para percorrer o caminho. A seqüência de células gerada deve corresponder a uma trajetória factível de ser executada por um dirigível, o qual é o tipo de veículo a que se destina o sistema de navegação tratado nesta dissertação. Devido a esta restrição, nem todas as transições entre células são consideradas válidas. Desta forma, é permitida qualquer transição entre células que estão em uma mesma altitude (coordenada do eixo y) no mapa do espaço de trabalho; porém, as transições entre células que estão em altitudes diferentes são permitidas somente se as mesmas estiverem em uma mesma coluna (coordenada do eixo x) e em um mesmo nível de profundidade (coordenada do eixo z); esta restrição é imposta para facilitar a tarefa do módulo de controle durante as

operações de seguimento de trajetória.

No planejamento de trajetória tridimensional, devido à adição de mais uma dimensão, em relação ao bidimensional, há um crescimento considerável no espaço de busca por um caminho. Devido a isto, as trilhas iniciais de feromônio são feitas de modo diferenciado. Assim, as trilhas iniciais de feromônio correspondem às células que estão:

- na coluna do ponto de origem;
- na coluna do ponto de destino;
- na linha do ponto de origem;
- na linha do ponto de destino;
- no nível de profundidade do ponto de origem;
- no nível de profundidade do ponto de destino;
- nas diagonais do ponto de origem, considerando o plano xz;
- nas diagonais do ponto de destino, considerando o plano xz;
- na vizinhança dos obstáculos;
- em um caminho ótimo, traçado previamente, desconsiderando-se a presença de obstáculos, partindo do ponto de origem até o ponto de destino (a FIG. 4.32 mostra o pseudocódigo do algoritmo desenvolvido para gerar um caminho ótimo entre dois pontos);
- na coluna de alguma célula pertencente ao caminho ótimo, traçado previamente desconsiderando-se a presença de obstáculos, do ponto de origem ao ponto de destino;
- em um caminho ótimo, traçado previamente, desconsiderando-se a presença de obstáculos, partindo do ponto de destino até o ponto de origem (a FIG. 4.32 mostra o pseudocódigo do algoritmo desenvolvido para gerar um caminho ótimo entre dois pontos);

```
algoritmo caminho_otimo3D(XOrigem, YOrigem, ZOrigem, XDestino, Ydestino, ZDestino)
        caminho[0][0] = Xorigem
        caminho[0][1] = Yorigem
        caminho[0][2] = Zorigem
        n_celulas = 1;
        enquanto (YOrigem <> YDestino)
                caminho[n_celulas][0] ← XOrigem
                if (caminho[n_celulas - 1][1] < YDestino)
                        caminho[n_celulas][1] ← caminho[n_celulas – 1][1] + 1
                fim if
                else
                        caminho[n_celulas][1] ← caminho[n_celulas – 1][1] - 1
                fim else
                caminho[n_celulas][2] ← ZOrigem
                n_celulas = n_celulas + 1;
        fim enquanto

        enquanto (XOrigem <> XDestino || ZOrigem <> Zdestino)
                if (caminho[n_celulas - 1][0] < XDestino)
                        caminho[n_celulas][0] ← caminho[n_celulas – 1][0] + 1
                fim if
                else if (caminho[n_celulas - 1][0] > XDestino)
                        caminho[n_celulas][0] ← caminho[n_celulas – 1][0] - 1
                fim else if
                else
                        caminho[n_celulas][0] = XDestino
                fim else
                caminho[n_celulas][1] ← YDestino

                if (caminho[n_celulas - 1][2] < YDestino)
                        caminho[n_celulas][2] ← caminho[n_celulas – 1][2] + 1
                fim if
                else if (caminho[n_celulas - 1][2] > YDestino)
                        caminho[n_celulas][2] ← caminho[n_celulas – 1][2] - 1
                fim else if
                else
                        caminho[n_celulas][2] = YDestino
                fim else
                n_celulas = n_celulas + 1;
        fim enquanto
        retorna caminho
fim algoritmo
```

FIG. 4.32: Pseudocódigo do algoritmo de determinação de um caminho ótimo no espaço tridimensional, desconsiderando-se obstáculos

- na coluna de alguma célula pertencente ao caminho ótimo, traçado previamente desconsiderando a presença de obstáculos, do ponto de destino ao ponto de origem.

Após o processo de construção das trilhas iniciais de feromônio, o próximo passo é iniciar o processo da metaheurística Colônia de Formigas. A adaptação de tal metaheurística para o caso tridimensional foi similar ao caso bidimensional. Primeiramente, é estipulado um número máximo de iterações e um número máximo de formigas por iterações. De forma alternada, metade das formigas tenta encontrar um caminho partindo do ponto de origem, direcionando-se ao ponto de destino, e a outra metade tenta encontrar o caminho contrário. Cada formiga possui um número máximo, estipulado, de passos; caso uma formiga alcance este máximo, a mesma é desconsiderada. Os caminhos não podem conter células repetidas. Mais uma vez, os resultados obtidos apenas com o uso da metaheurística colônia de formigas não foram considerados bons em muitos casos, sobretudo quando existiam obstáculos entre a origem e o destino.

Os problemas observados foram semelhantes aos do caso bidimensional: células com mais de uma vizinha pertencente ao caminho, e tendência a fazer voltas. Portanto, as otimizações na vizinhança e nas retas também foram aplicadas; entretanto, a otimização nas retas é feita referenciando-se o plano xz ao invés do plano xy. Porém, isto não foi suficiente, pois mesmo após a execução das duas otimizações citadas, ainda notava-se mais um problema: havia uma tendência a fazer voltas em hiperplanos perpendiculares ao plano xz, como é o caso ilustrado na FIG. 4.33, a qual mostra um caminho gerado sem otimização. Então, foi desenvolvida a otimização retas/altitudes; a qual é uma extensão da otimização nas retas. Esta otimização consiste em verificar não apenas as células do caminho que estiverem nas retas (linha, profundidade e diagonais no plano xz) e na altura do eixo y de uma dada célula; mas sim, se nas retas da mesma (em toda a extensão do eixo y) é possível traçar um caminho ótimo sem passar por células ocupadas, conforme o algoritmo da FIG. 4.32. Em caso afirmativo, o caminho segue direto para a célula mais adiante no caminho, que satisfaça esta condição; vide FIG. 4.34.

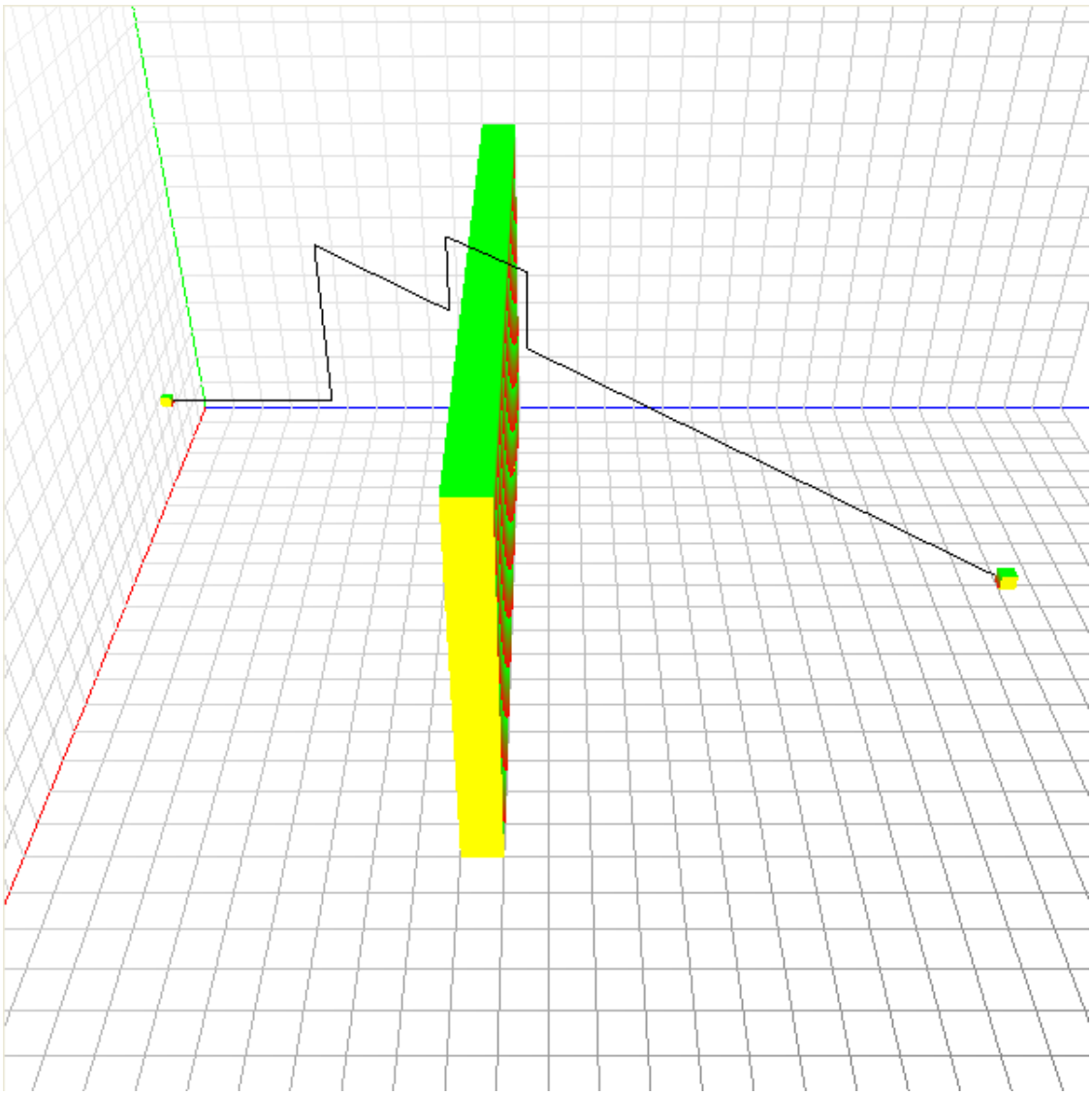


FIG. 4.33: Exemplo de um caminho, não otimizado, no espaço tridimensional com obstáculos

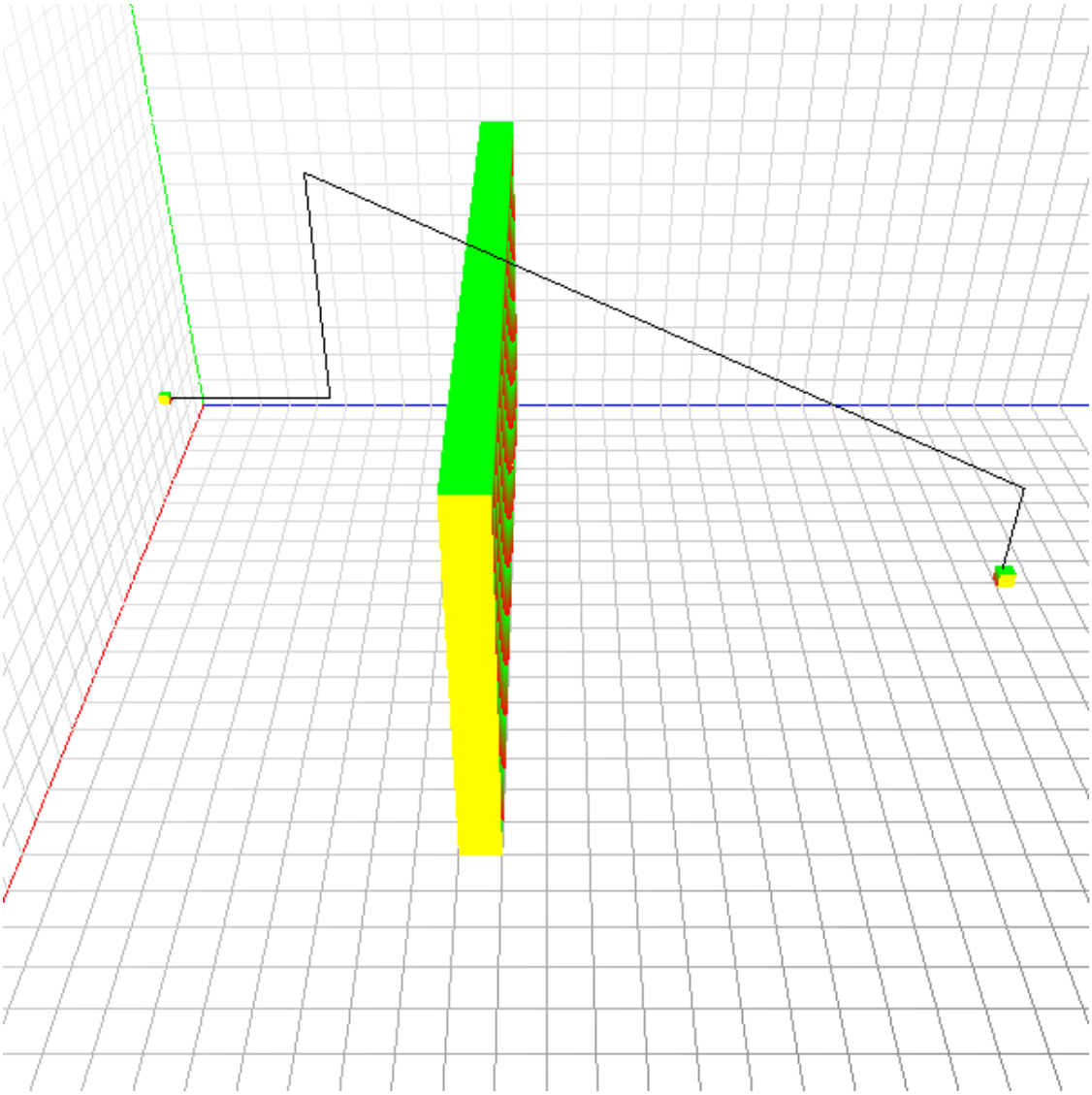


FIG. 4.34: Exemplo de um caminho, otimizado, no espaço tridimensional com obstáculos

A exemplo do caso bidimensional, o planejamento de trajetória no espaço de trabalho tridimensional também utiliza apenas uma matriz para representar o mapa do ambiente e a quantidade de feromônio depositada em cada uma das células. Um pseudocódigo do algoritmo de planejamento de trajetória no espaço de trabalho tridimensional é mostrado na FIG. 4.35. A seguir, é descrito como é feito o cálculo do custo de um caminho, o qual corresponde a estimativa do *tempo de viagem* do mesmo.

Os caminhos produzidos, para o espaço de trabalho tridimensional, são compostos de uma seqüência de segmentos de retas, unidas por suas extremidades. Portanto, o cálculo do custo dos mesmos é feito de forma análoga a do caso bidimensional. Desta forma, o custo de um caminho é dado pela soma da estimativa dos tempos gastos para percorrer todos segmentos de reta; dos tempos gastos para realizar cada rotação intermediária, entre cada par de segmentos de reta seqüenciais; e dos tempos gastos na rotação inicial e rotação final. Porém, existe uma diferença, ao contrário do caso bidimensional, existem segmentos que implicam em mudança de altitude. Portanto, ao se determinar o custo de um dado segmento de reta, deve-se levar em conta este fator; assim, o custo de um caminho é dado pela seguinte equação:

$$C_{3D} = [\sum_{i=1}^{ns} T(vc, vs, vd, a, S_i, D_i) + \sum_{i=1}^{nr} G(R_i)] + G(ri) + G(rf) \tag{4.13}$$

onde:

- ns: número de segmentos de reta que formam o caminho;
- T: função que retorna o tempo necessário para percorrer um segmento de reta, definida na EQ. 4.14;
- vc: velocidade de cruzeiro;
- vs: velocidade de subida;
- vd: velocidade de descida;
- a: aceleração de cruzeiro;
- S: vetor que contém os segmentos de reta que formam o caminho;

```
algoritmo procurar_caminho3D(XOrigem,YOrigem, ZOrigem, XDestino, YDestino, ZDestino, NIteracoes, NPassosFormiga,
                                                                    NFormigasIteracao, MatrizMapa, CustoMaximo)
        fazer_trilhas_feromonios(XOrigem, YOrigem,  ZOrigem, XDestino, YDestino, ZDestino, MatrizMapa, MatrizFeromonioT)
        sair ← 0
        n_celulas_caminho ← 0
        n_caminhos ← 0
        enquanto(i < NIteracoes e sair = 0)
                i++
                MatrizFeromonio ← MatrizFeromonioT;
                para j = 0 até j < NFormigas_Iteracao
                        se sair = 1
                                parar
                        fim se
                        se (resto(j, 2) <> 0)
                                XObjetivo ← XOrigem
                                YObjetivo ← YOrigem
                                ZObjetivo ← ZOrigem
                                x ← XDestino
                                y ← YDestino
                                z ← YDestino
                        fim se
                        senao
                                x ← XOrigem
                                y ← YOrigem
                                z ← ZOrigem
                                XObjetivo ← XDestino
                                YObjetivo ← YDestino
                                ZObjetivo ← ZDestino
                        fim senao
                        n_celulas_caminho ← 0
                        enquanto ((x <> x_objetivo ou y <> y_objetivo ou z <> z_objetivo) e n_celulas_caminho <
                                                                                        NPassosFormiga)
                                escolhe_proxima_celula_caminho(x, y, z, MatrizMapa, MatrizFeromonio)
                                MatrizFeromonioT[x][y][z] ←  MatrizFeromonioT[x][y][z] + 0.5
                                caminho[j][0] ← x
                                caminho[j][1] ← y
                                caminho[j][2] ← z
                                n_celulas_caminho ← n_celulas_caminho + 1;
                        fim enquanto
                        se  (x = XObjetivo e y = YObjetivo e z = ZObjetivo)
                                n_caminhos ← n_caminhos + 1
                                otimizacao_retas(caminho, n_celulas_caminho, MatrizMapa)
                                otimizacao_vizinhanca(caminho, n_celulas_caminho)
                                otimizacao_retas_altitudes(caminho, n_celulas_caminhos, MatrizMapa)
                                se n_caminhos = 1
                                        melhor_caminho ← caminho
                                        n_celulas_melhor_caminho ← n_celulas_caminho
                                fim se
                                senao se custo(caminho, n_celulas_caminho) < custo(melhor_caminho,
                                                                                n_celulas_melhor_caminho)
                                        melhor_caminho ← caminho
                                        n_celulas_melhor_caminho ← n_celulas_caminho
                                fim senao se
                                depositar_feromonio(caminho, MatrizFeromonioT)
                                se custo(caminho, n_celulas_caminho) <= CustoMaximo
                                        sair ← 1
                                fim se
                        fim se
                fim para
                evaporar_feromonio(MatrizFeromonioT)
        fim enquanto
        retornar(melhor_caminho)
fim algoritmo
```

FIG. 4.35: Pseudocódigo do algoritmo de determinação de um caminho no espaço de trabalho tridimensional

- D: vetor que contém a diferença das coordenadas y (altitude) da primeira e última células dos segmentos de reta que formam o caminho, cada elemento desse vetor é igual a y_1 - y_2, onde y_1 e y_2 correspondem, respectivamente, às coordenadas y da primeira e última células de um caminho;
- nr: número de rotações entre os segmentos de reta que compõem o caminho;
- G: função que retorna o tempo gasto em determinada rotação;
- R: vetor que contém o número de graus referentes às rotações realizadas durante o percurso;
- ri: número de graus referente à rotação inicial;
- rf: número de graus referente a rotação final.

$$T(vc, vs, vd, a, c, d) = \begin{cases} \frac{c}{vs}, \text{se } d < 0 \\ Ts(vc, a, c), \text{se } d = 0 \\ \frac{c}{vd}, \text{se } d > 0 \end{cases} \tag{4.14}$$

onde:

- vc: velocidade de cruzeiro;
- vs: velocidade de subida;
- vd: velocidade de descida;
- a: aceleração de cruzeiro;
- c: é o comprimento do segmento.
- d: y_1 - y_2, onde y_1 e y_2 correspondem, respectivamente, às coordenadas y da primeira e última células do caminho;
- Ts: função que retorna o tempo necessário para percorrer um segmento de reta, definida na EQ. 4.12.

Este capítulo trata da descrição dos resultados obtidos ao final deste trabalho e dos testes realizados através de simulações. Como resultado, é apresentado um programa para processamento de imagens e planejamento de trajetória denominado NavDANTI. Já nos testes, são apresentadas algumas simulações do planejamento de trajetória.

5.1 O PROGRAMA NAVDANTI

Esta seção destina-se a mostrar as funcionalidades do programa NavDANTI, desenvolvido durante a realização deste trabalho. O NavDANTI é dividido em duas partes: (a) processamento de imagens; (b) planejamento de trajetória.

A parte de processamento de imagens permite que o usuário execute operações de suavização, segmentação e calibração de um par de câmeras estéreo. A segmentação e a suavização da imagem é feita, diretamente pelo usuário, através da tela inicial do NavDANTI, vide FIG. 5.1. A calibração do par de câmeras estéreo é feita em dois passos:

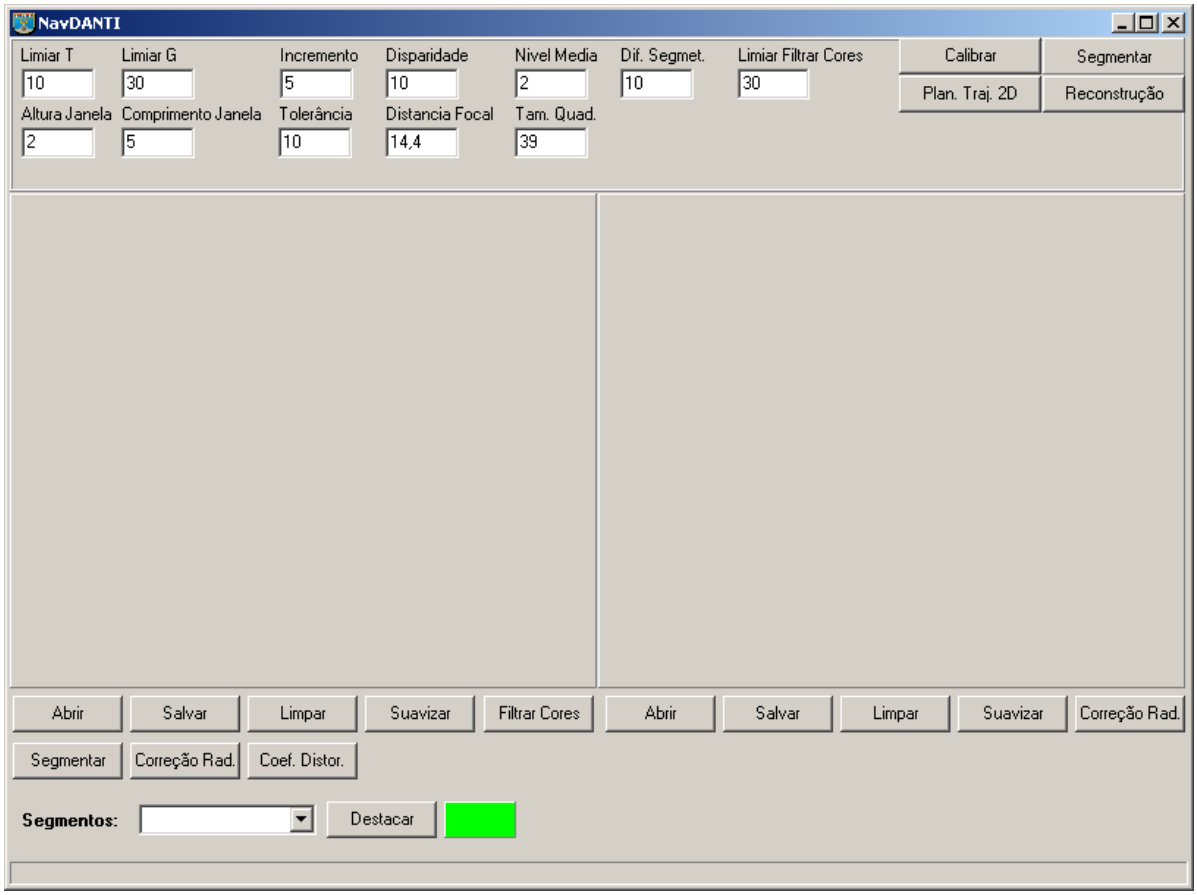


FIG. 5.1: Tela inicial do programa NavDANTI

- cálculo dos coeficientes k_1, k_2, p_1 e p_2 para a tarefa de correção da distorção radial. Assim, é necessário fornecer ao programa, uma imagem que contenham quadrados brancos e pretos, dispostos em 11 (onze) linhas e 15 (quinze) colunas, como em um tabuleiro de xadrez, vide FIG. 5.2 e FIG. 5.3.

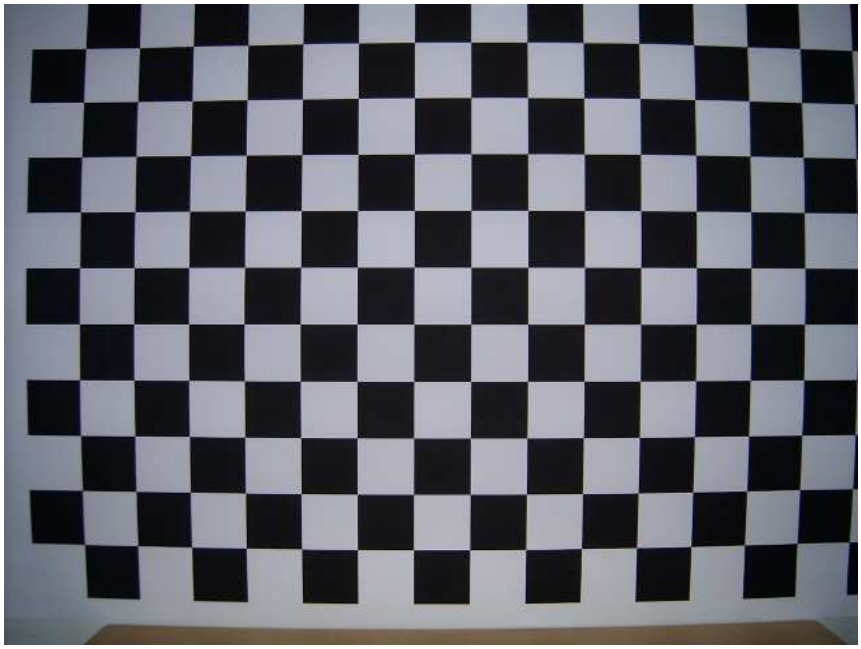


FIG. 5.2: Figura para cálculo dos coeficientes de distorção radial

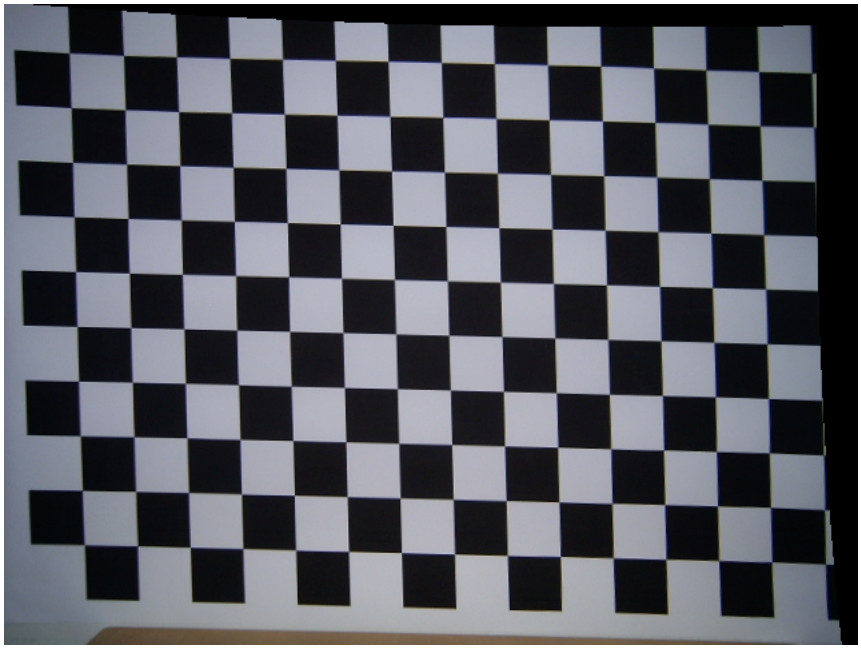


FIG. 5.3: FIG. 5.2, após o processo de correção da distorção radial

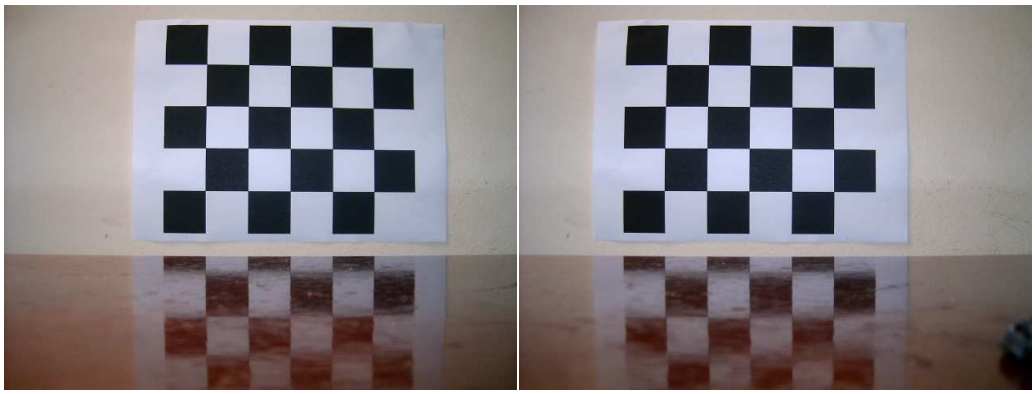


FIG. 5.4: Imagens para calibração do par de câmeras estéreo
(a) Imagem obtida da câmera esquerda. (b) Imagem obtida da câmera direita.

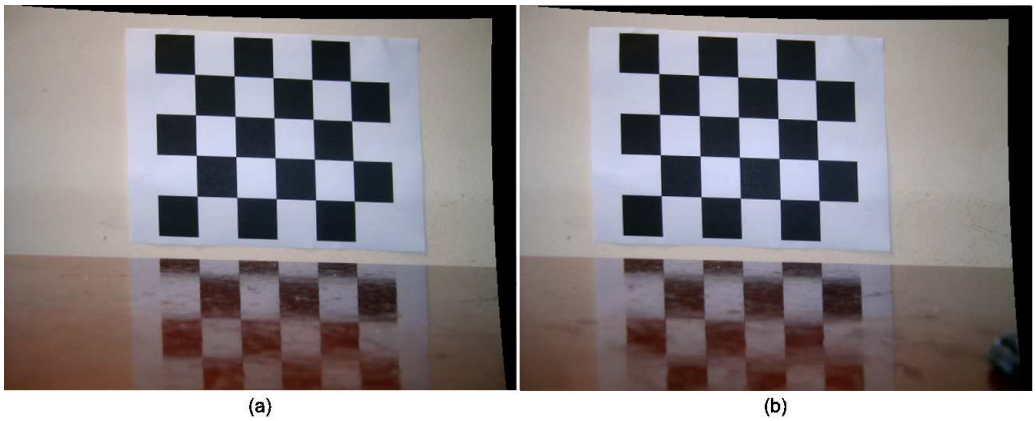


FIG. 5.5: (a) FIG. 5.4(a), após correção da distorção radial. (b) FIG. 5.4(b), após correção da distorção radial.

- cálculo da matriz fundamental: para a calibração do par de câmeras estéreo, que consiste em calcular a matriz fundamental do mesmo, é necessário fornecer ao sistema duas imagens obtidas de pontos de vistas diferentes, vide FIG. 5.4 e FIG. 5.5. As duas imagens devem:
  - conter quadrados brancos e pretos, dispostos em 6 (seis) linhas e 5 (cinco) colunas, como em um tabuleiro de xadrez;
  - ser obtidas de câmeras idênticas (mesma marca e modelo), cujas posições devem estar alinhadas nos eixos X e Z, havendo diferença apenas no eixo Y.

Feitas as operações de calibração, suavização e segmentação; o próximo passo é determinar o mapa do ambiente. Tal mapa é visualizado através da tela de planejamento

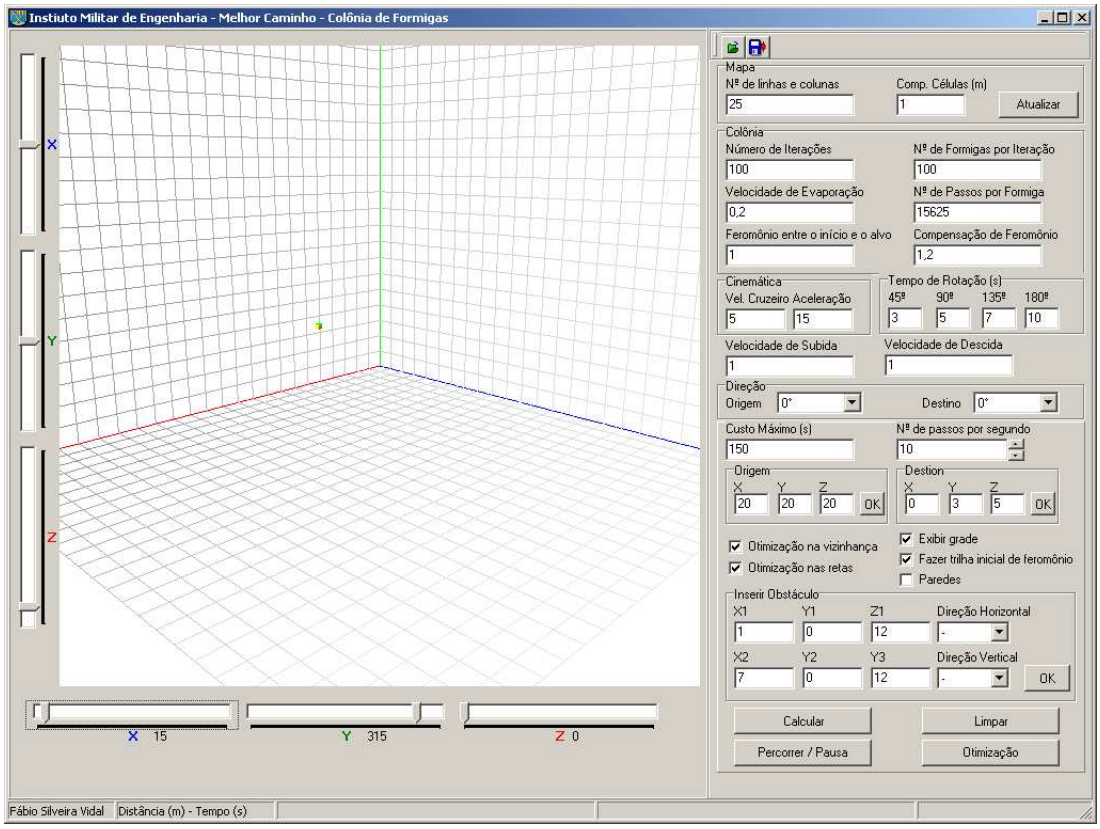


FIG. 5.6: Tela para simulação do planejamento de trajetória no espaço de trabalho tridimensional

de trajetória no espaço de trabalho tridimensional, na qual as áreas ocupadas por algum obstáculo são representadas por cubos, e é possível acrescentar obstáculos estáticos ou móveis, bem como armazenar mapas para uso posterior e observar a execução da trajetória gerada dinamicamente, conforme a movimentação dos obstáculos; vide FIG. 5.6. A FIG. 5.7 mostra a tela para simulação de planejamento de trajetória no espaço de trabalho bidimensional, a qual possui as mesmas funcionalidades da tela para simulação de planejamento de trajetória no espaço de trabalho tridimensional.

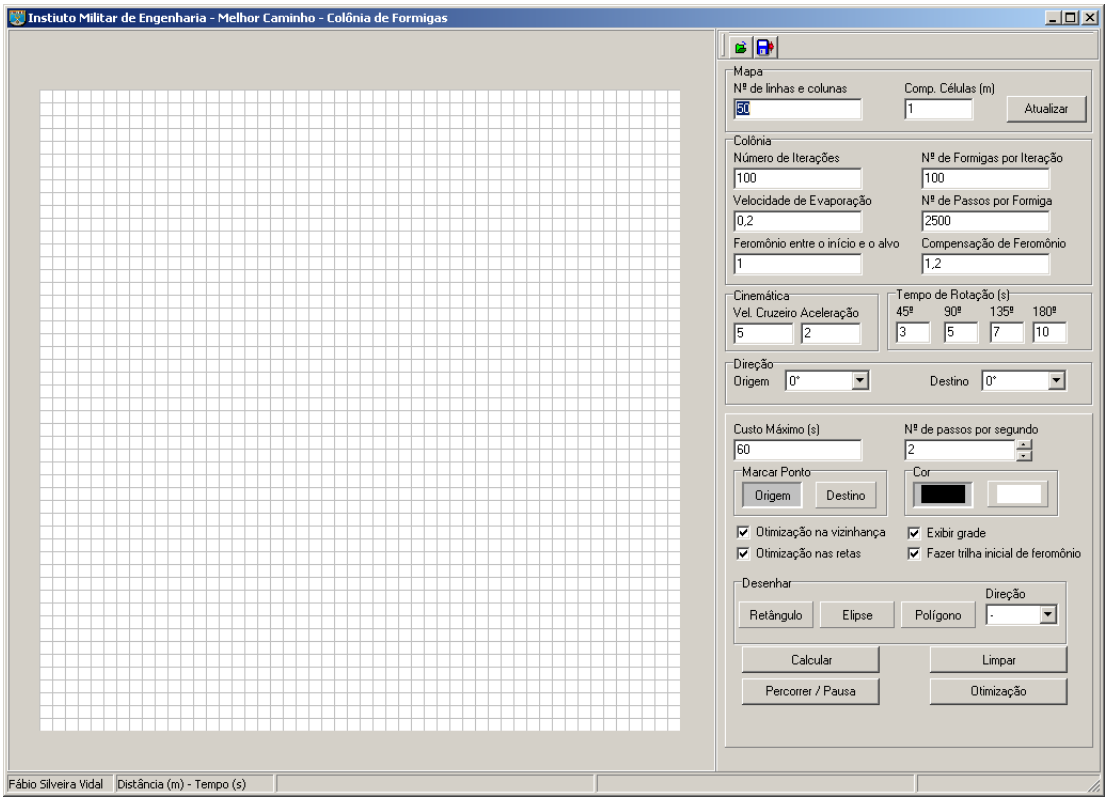
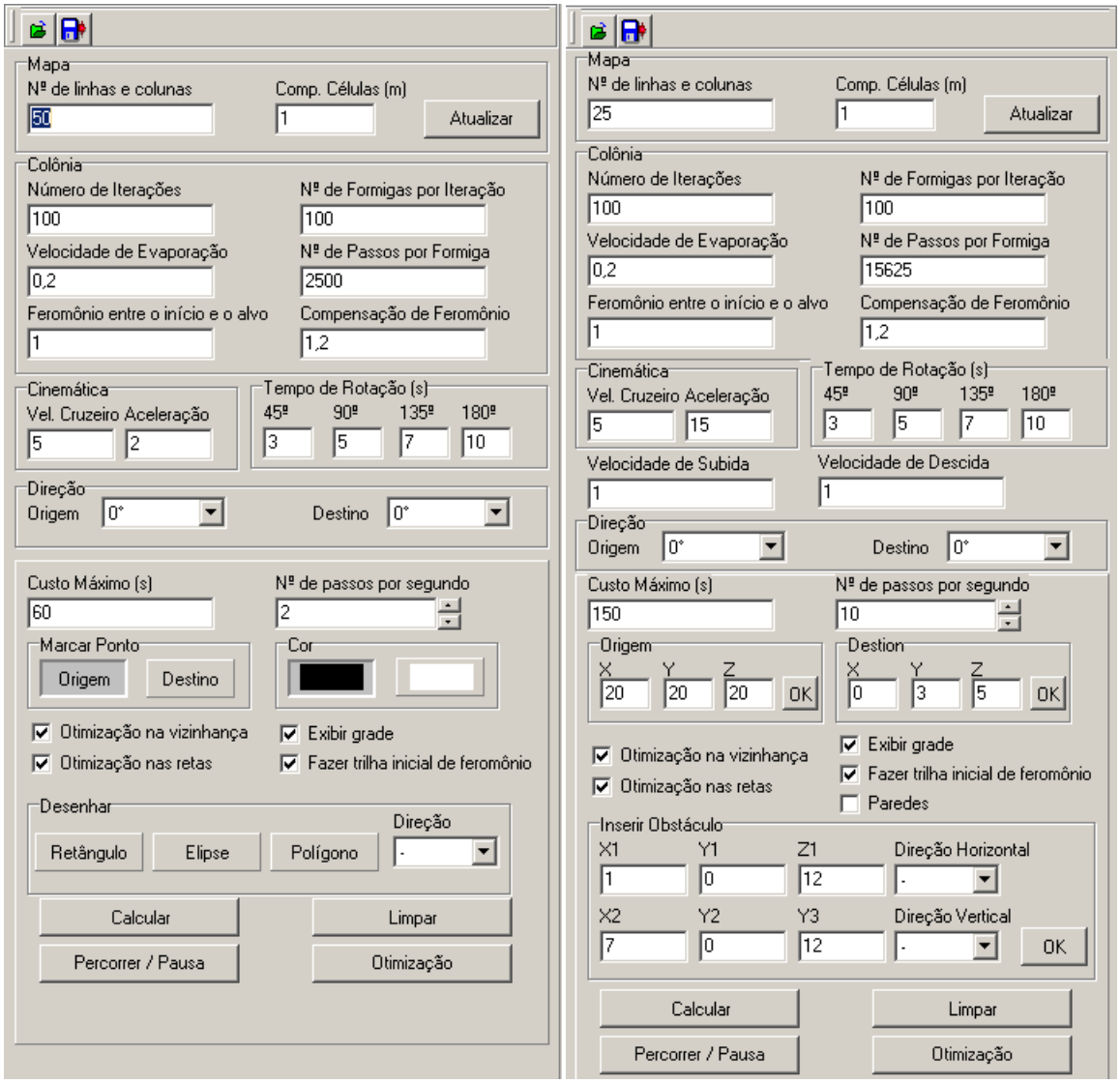


FIG. 5.7: Tela para simulação do planejamento de trajetória no espaço de trabalho bidimensional

A FIG. 5.8 mostra detalhes das telas de simulação de planejamento de trajetória. A seguir, serão apresentados alguns testes realizados com o NavDANTI.



(a) (b)

FIG. 5.8: Detalhe das telas para simulação de planejamento de trajetória (a) Espaço de trabalho bidimensional. (b) Espaço de trabalho tridimensional.

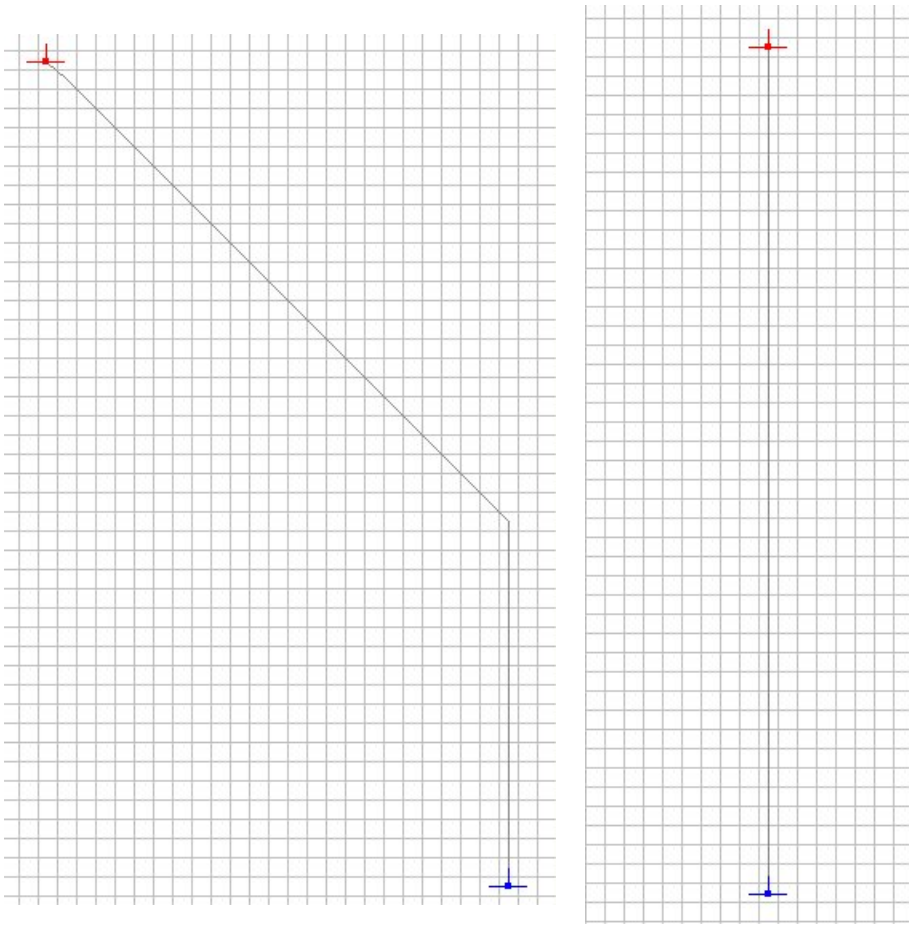


FIG. 5.9: Trajetória para uma situação sem obstáculos em um espaço de trabalho bidimensional.

5.2 TESTES E SIMULAÇÕES

Para os dois tipos de espaços de trabalhos tratados no desenvolvimento desta dissertação, foram feitos testes para determinar o caminho entre um ponto e outro em ambientes sem obstáculos, com obstáculos estáticos e com obstáculos móveis.

Para o espaço de trabalho bidimensional, foram testadas as seguintes situações:

- sem obstáculos, vide FIG. 5.9;

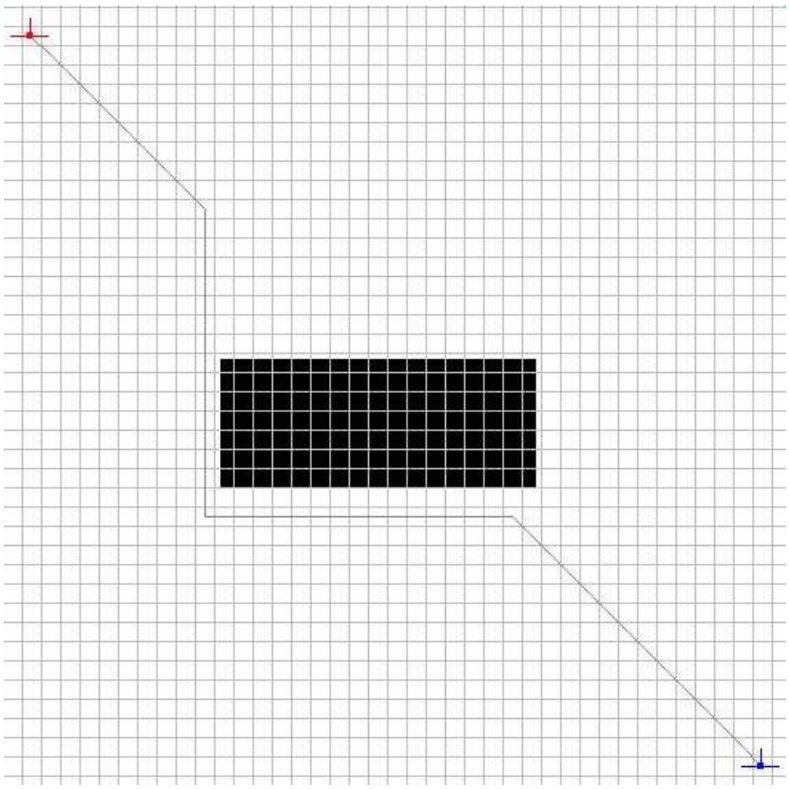


FIG. 5.10: Trajetória para uma situação com obstáculos estáticos em um espaço de trabalho bidimensional.

- com obstáculos estáticos, vide FIG. 5.10, FIG. 5.11 e FIG. 5.12;

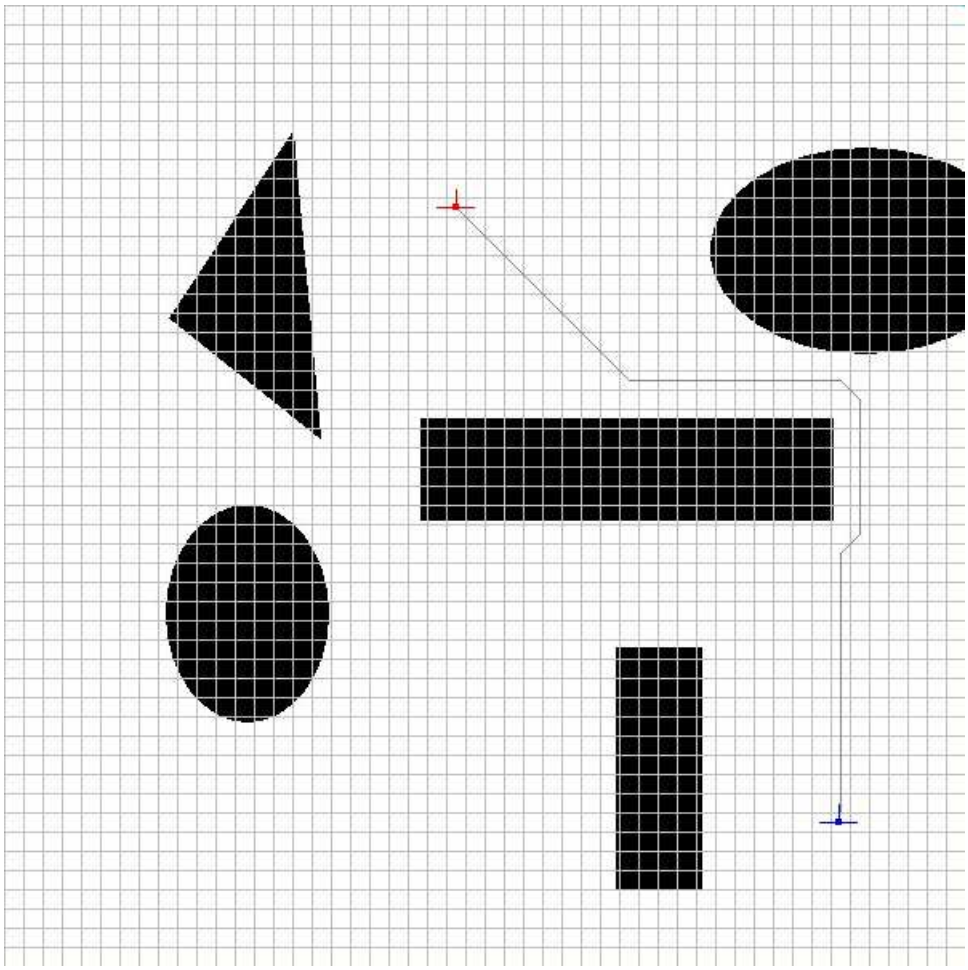


FIG. 5.11: Trajetória para uma situação com obstáculos estáticos em um espaço de trabalho bidimensional.

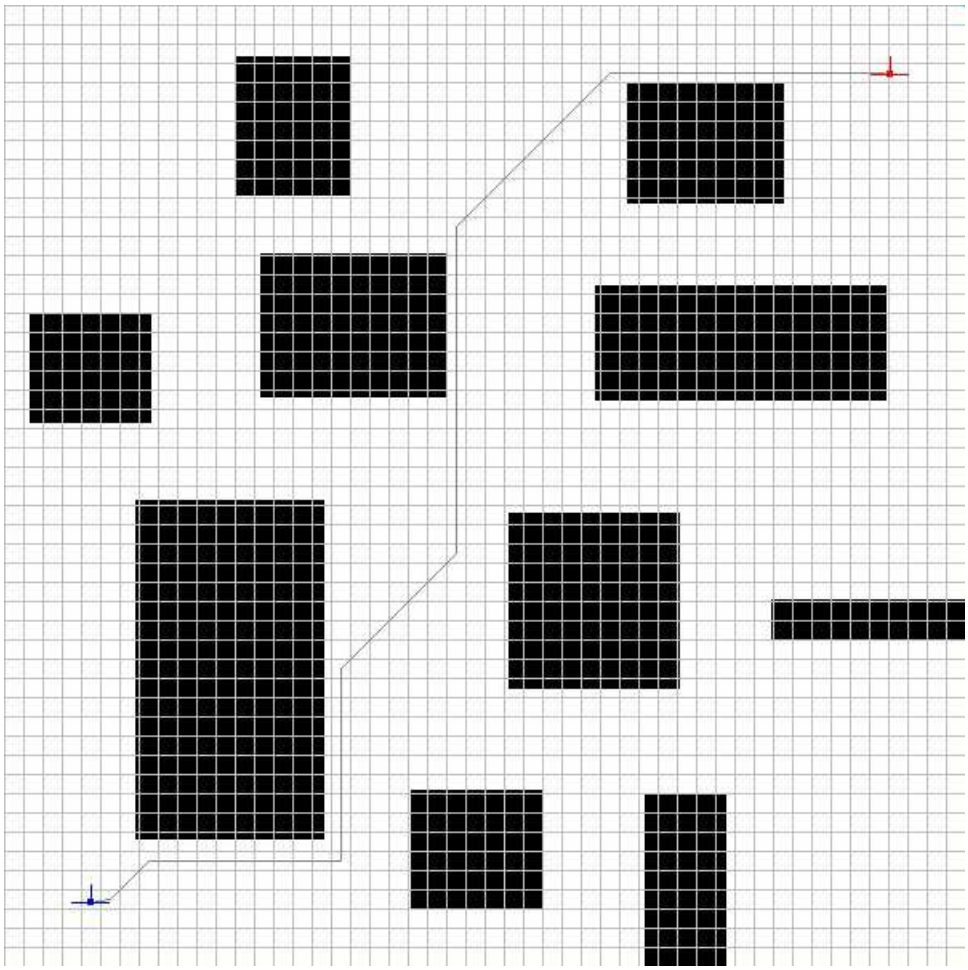


FIG. 5.12: Trajetória para uma situação com obstáculos estáticos em um espaço de trabalho bidimensional.

- com obstáculos móveis: nesta situação, a trajetória é adaptada, enquanto é percorrida, conforme a movimentação dos obstáculos. Assim, são mostradas duas seqüências de imagens que mostram as adaptações efetuadas durante a execução da trajetória. A primeira é constituída pelas FIG. 5.13, FIG. 5.14, FIG. 5.15 e FIG. 5.16; e, a segunda seqüência é constituída pelas FIG. 5.17, FIG. 5.18, FIG. 5.19, FIG. 5.20, FIG. 5.21, FIG. 5.22 e FIG. 5.23.

Nos resultados do planejamento de trajetória, em espaço de trabalho bidimensional, os caminhos gerados são diretos e com o mínimo de curvas possíveis, quando não existem obstáculos. Na presença de obstáculos, as trajetórias possuem poucas curvas e tendem a contorná-los; devido ao depósito de feromônio ao redor deles, feito no início do algoritmo. Em situações com obstáculos móveis, não foram encontrados problemas para executar a adaptação da trajetória. Porém, tais adaptações podem obrigar o dirigível a fazer mudanças de direção bruscamente. Isto acontece devido ao fato do sistema de planejamento de trajetória não ser adaptado às propriedades físicas do dirigível.

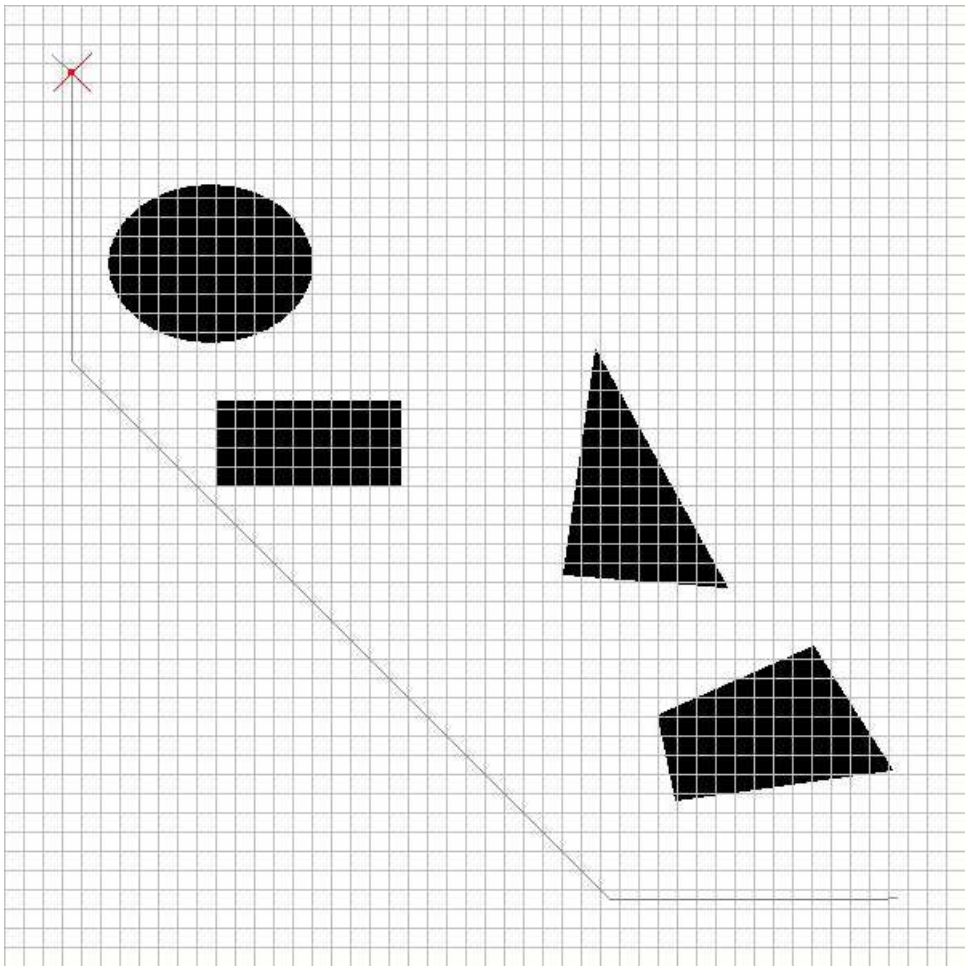


FIG. 5.13: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 1/4.

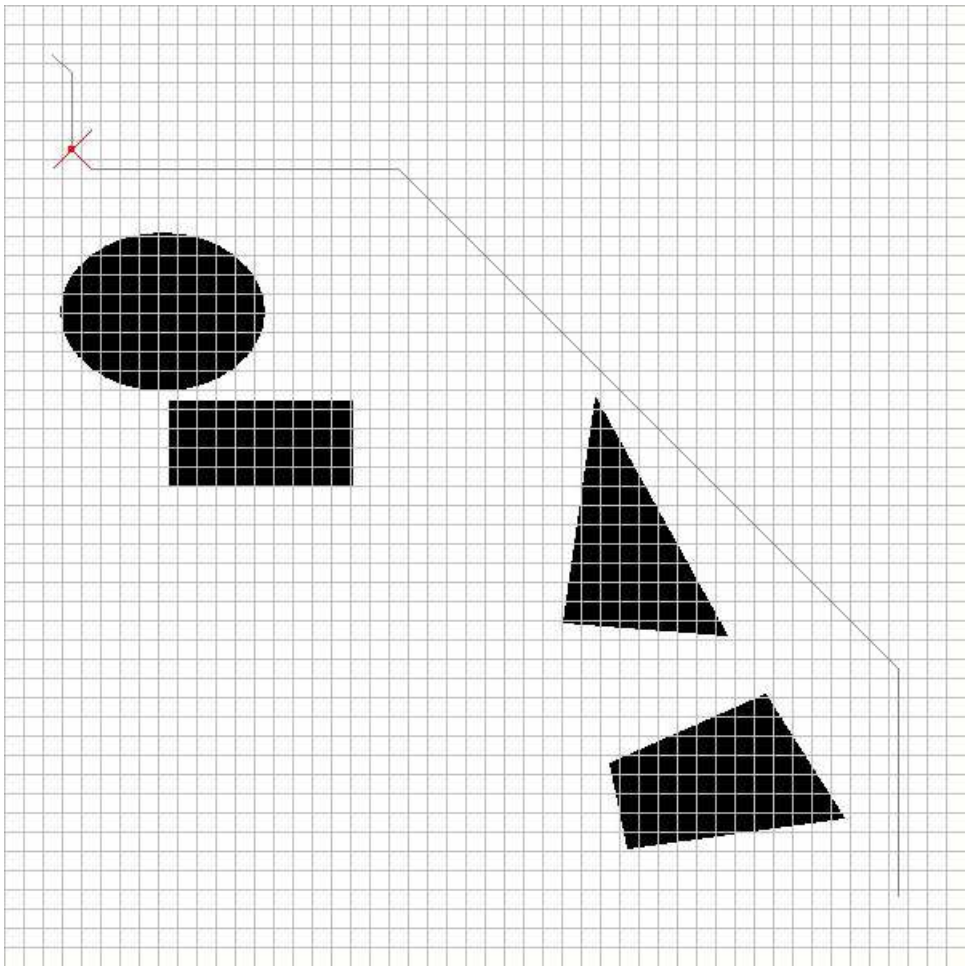


FIG. 5.14: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 2/4.

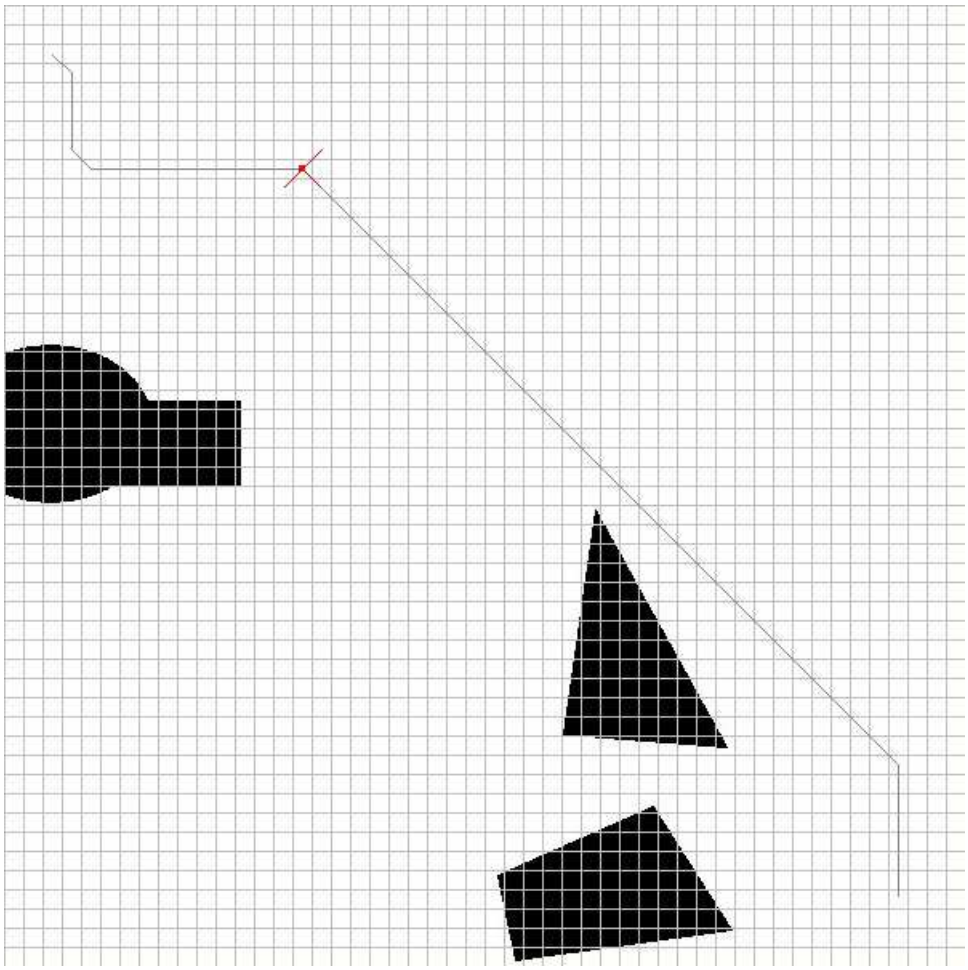


FIG. 5.15: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 3/4.

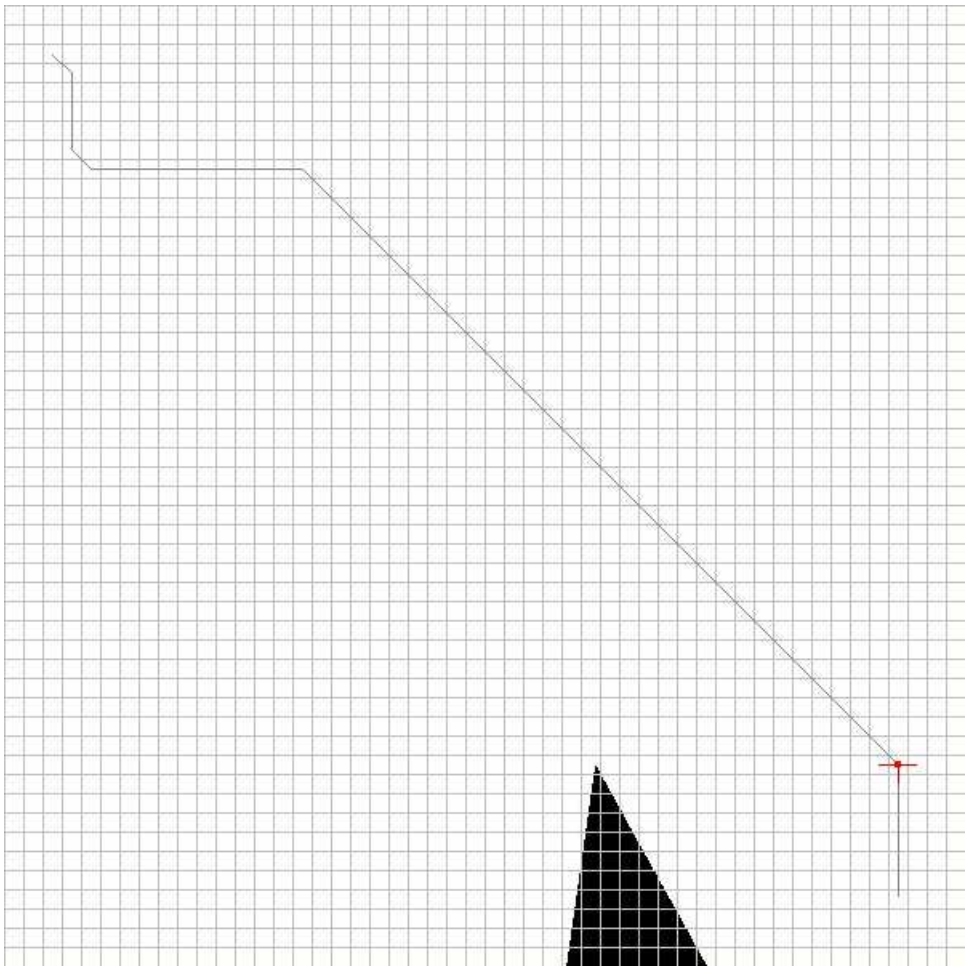


FIG. 5.16: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 4/4.

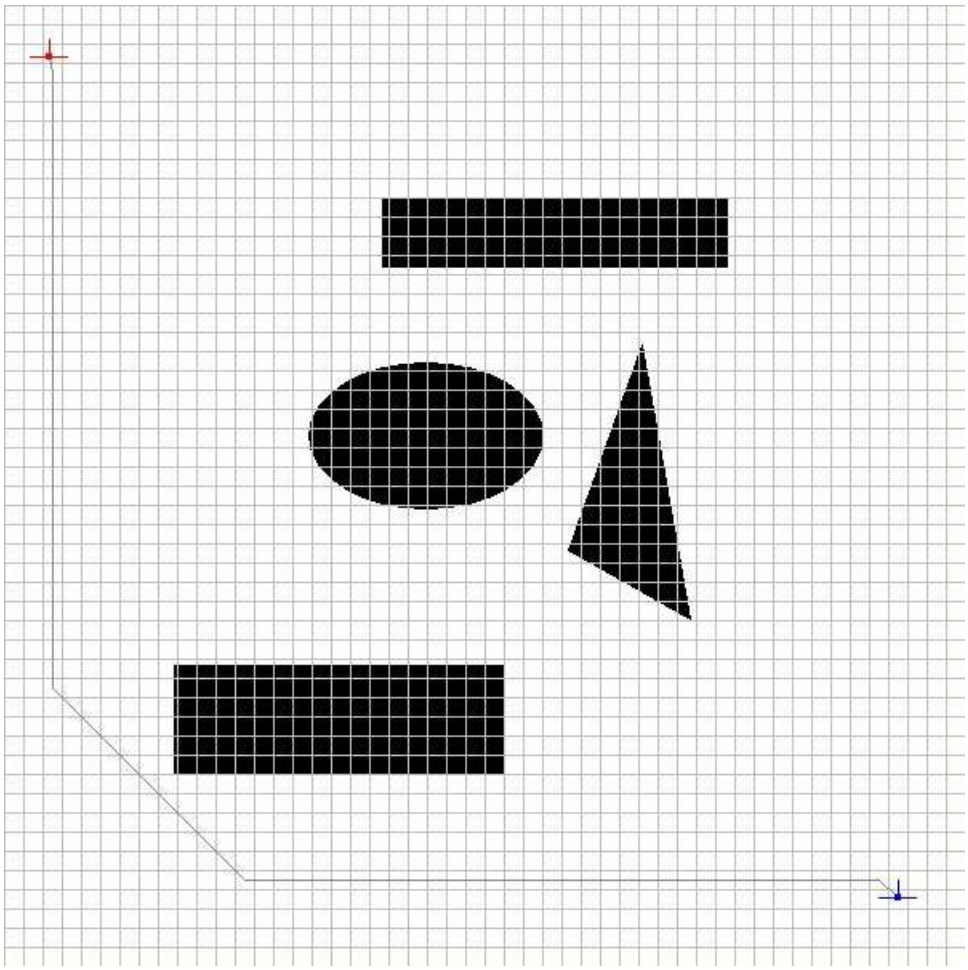


FIG. 5.17: Segunda seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 1/7.

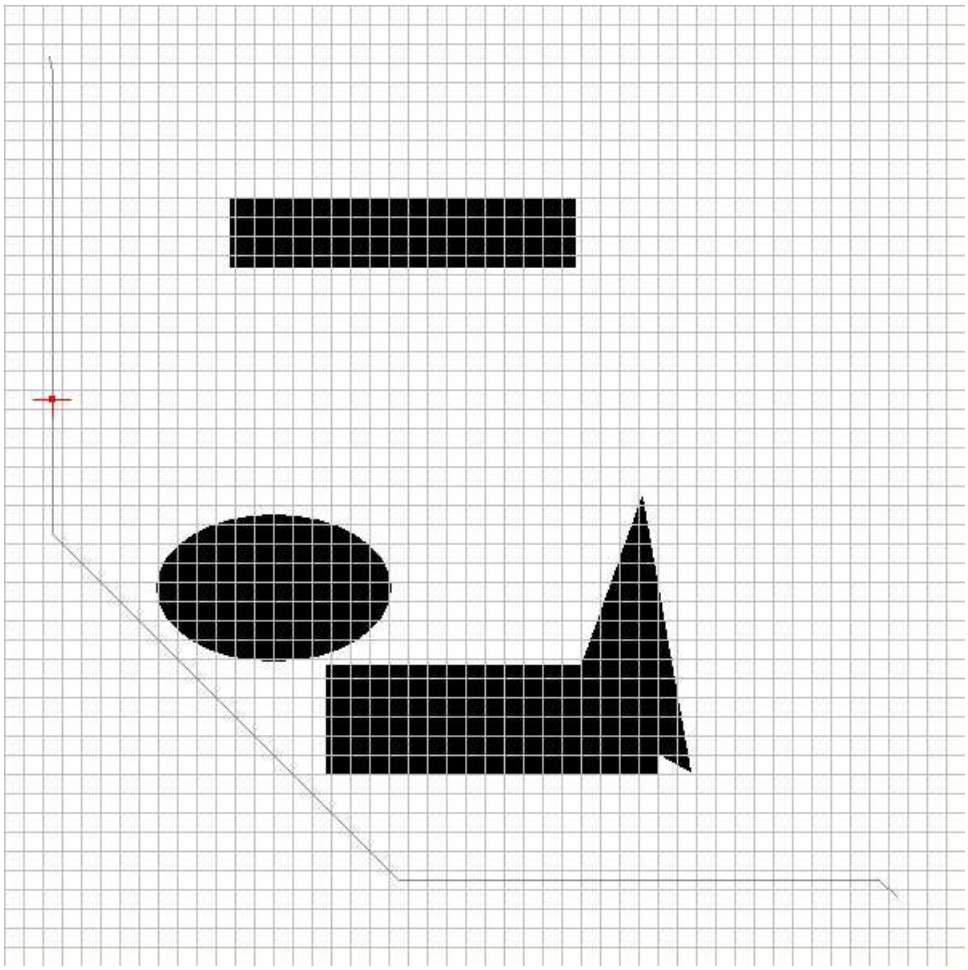


FIG. 5.18: Segunda seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 2/7.

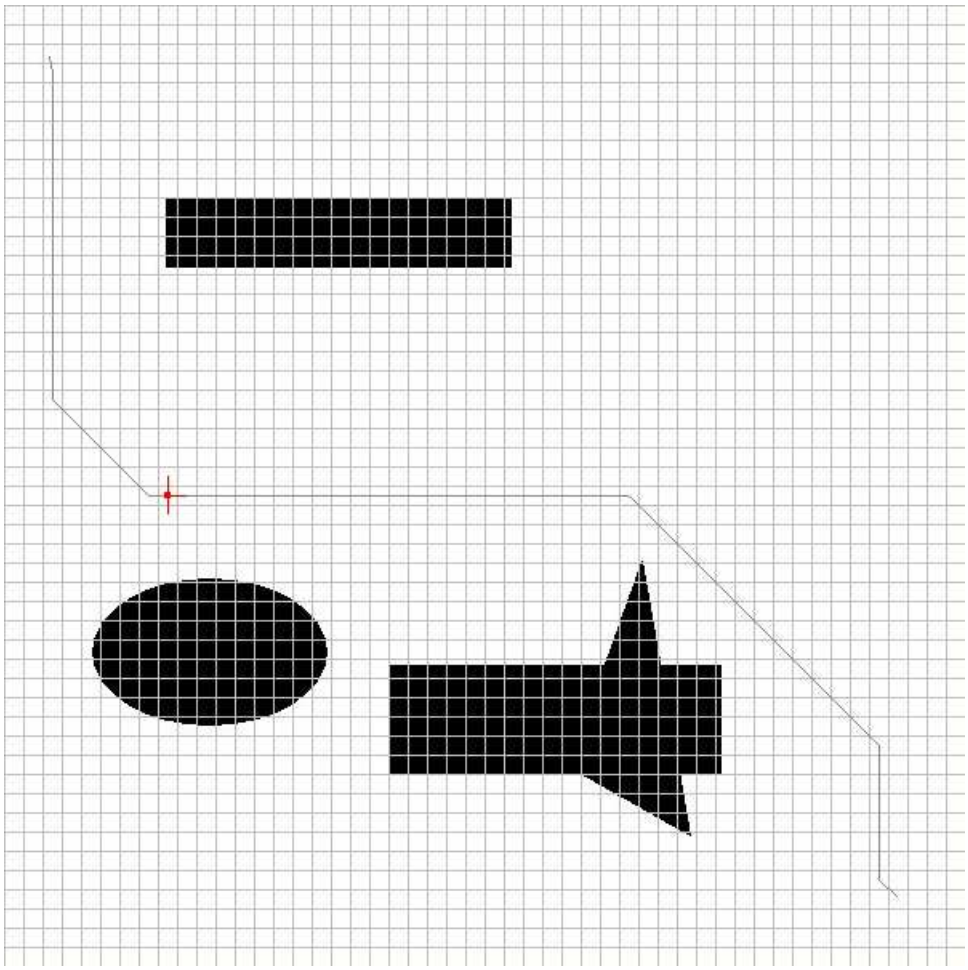


FIG. 5.19: Segunda seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 3/7.

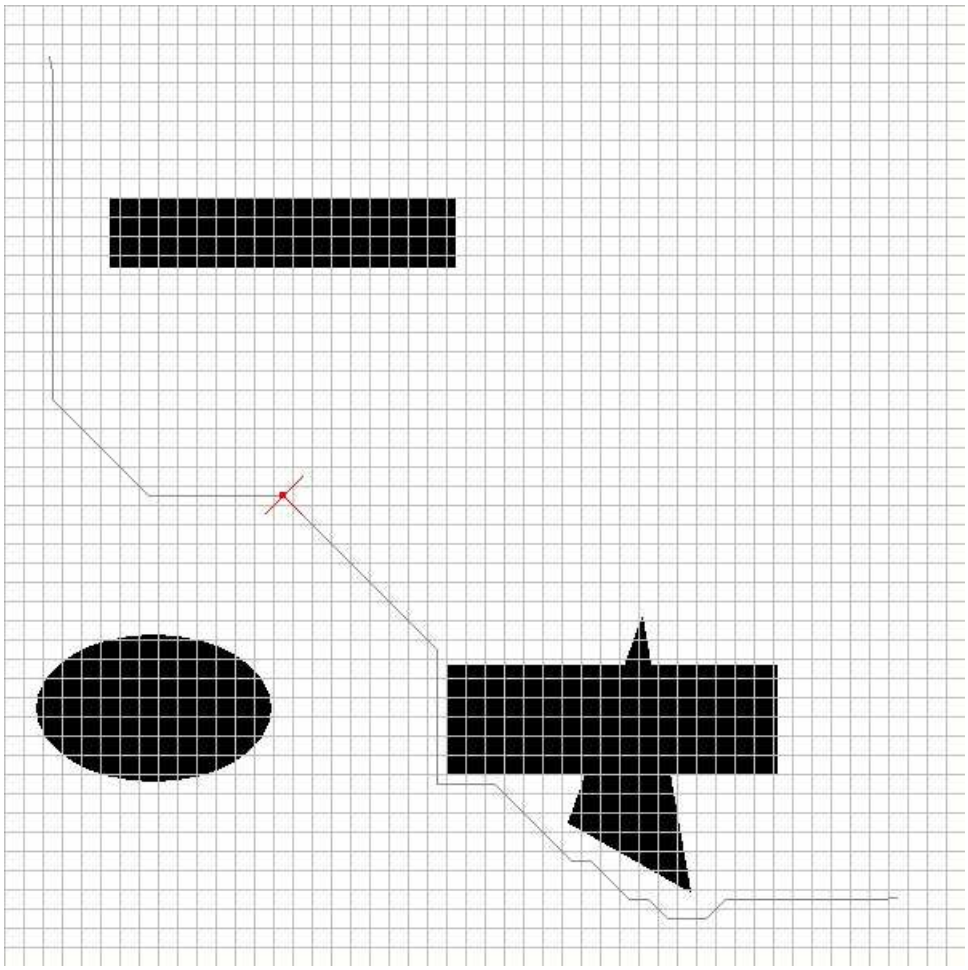


FIG. 5.20: Segunda seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 4/7.

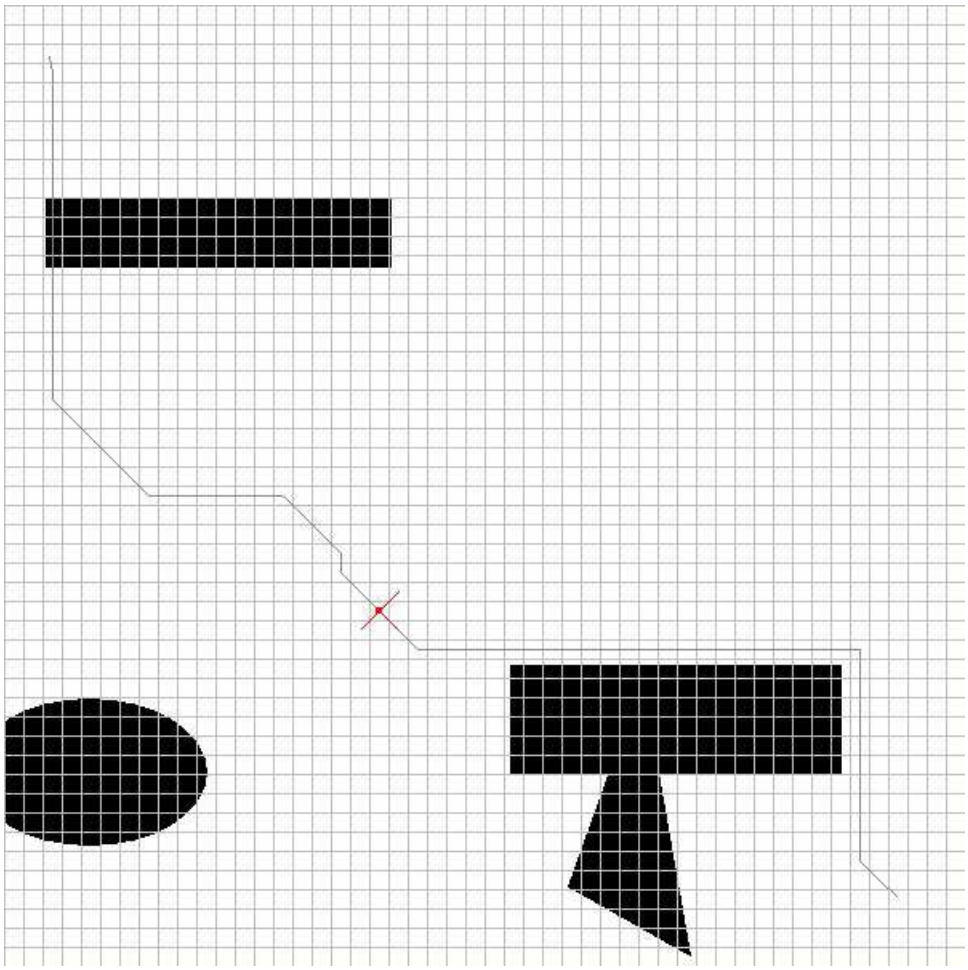


FIG. 5.21: Segunda seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 5/7.

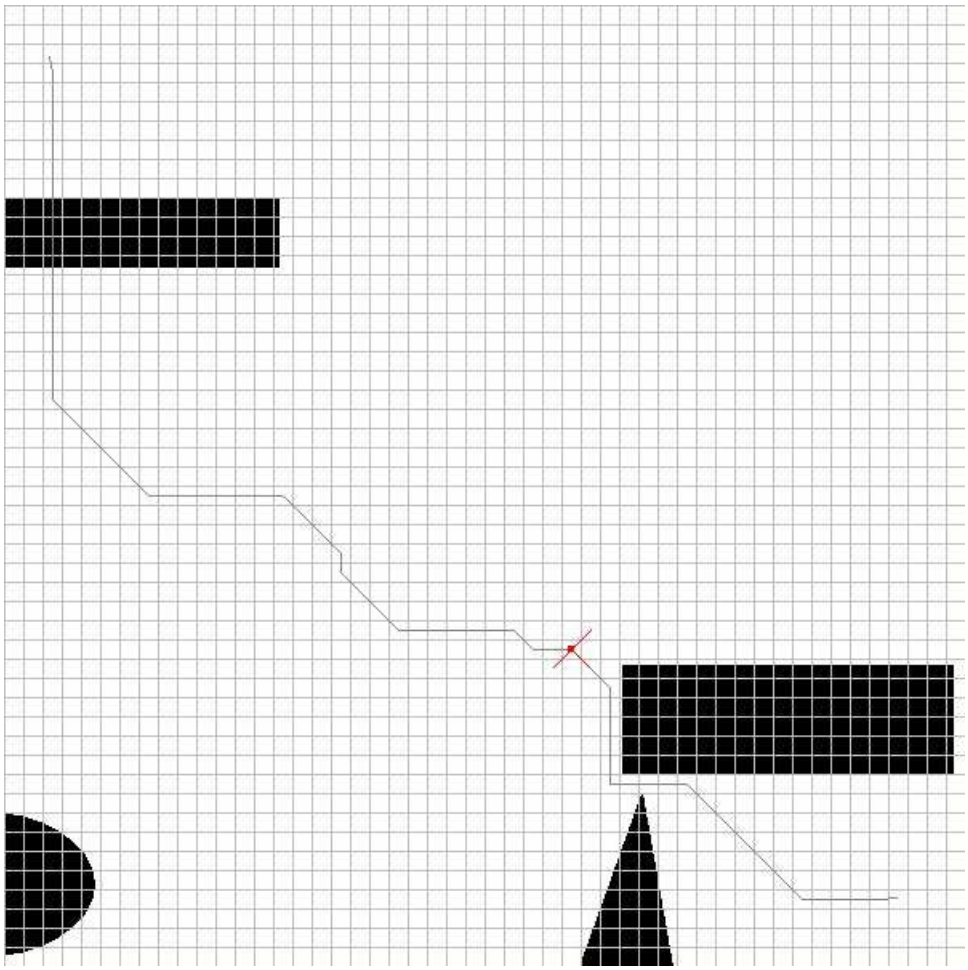


FIG. 5.22: Segunda seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 6/7.

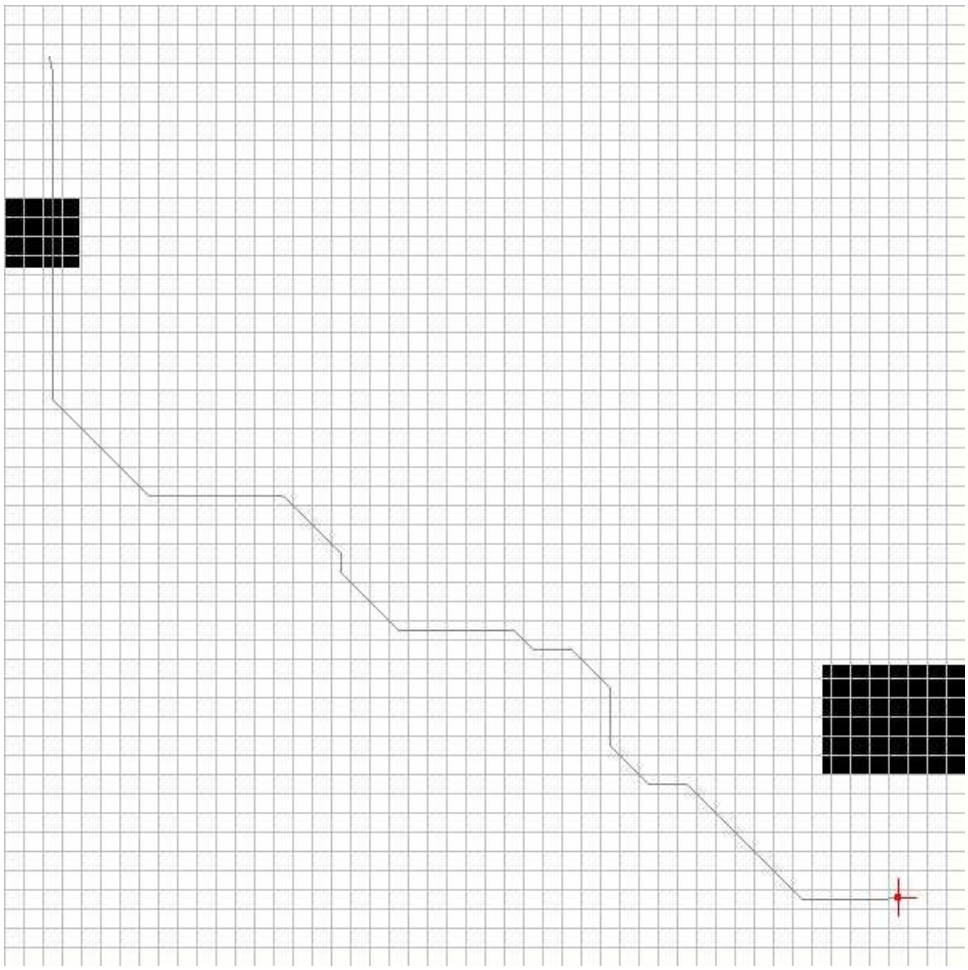


FIG. 5.23: Segunda seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho bidimensional 7/7.

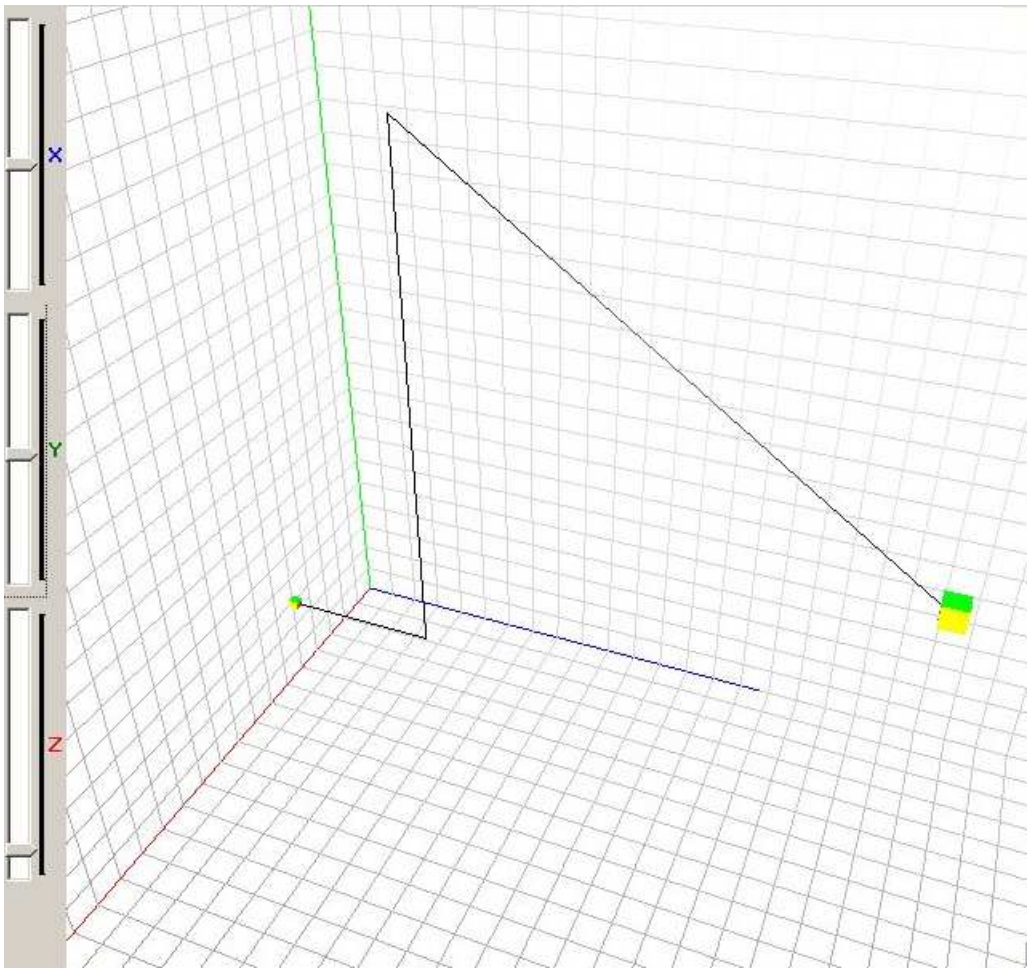


FIG. 5.24: Trajetória para uma situação sem obstáculos em um espaço de trabalho tridimensional.

As mesmas situações foram testadas para o espaço de trabalho tridimensional:

- sem obstáculos, vide FIG. 5.24, FIG. 5.25 e FIG. 5.26;

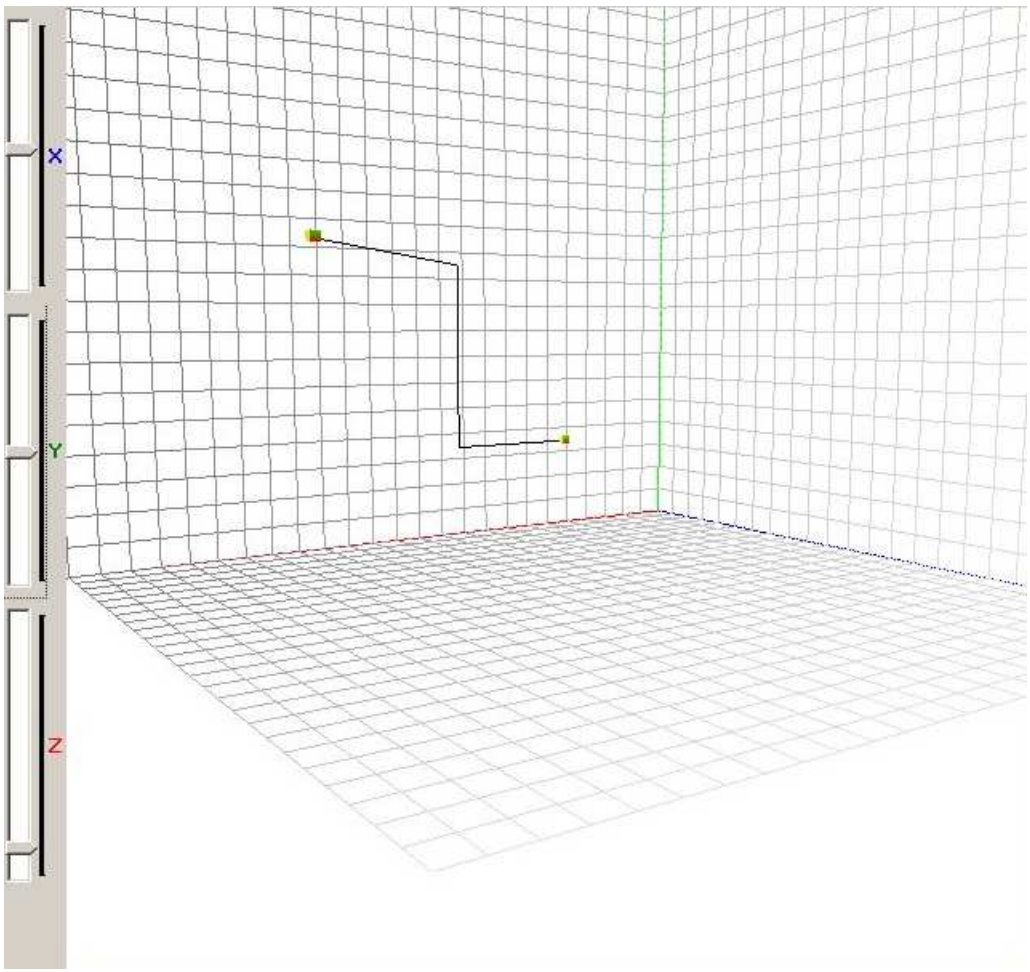


FIG. 5.25: Trajetória para uma situação sem obstáculos em um espaço de trabalho tridimensional.

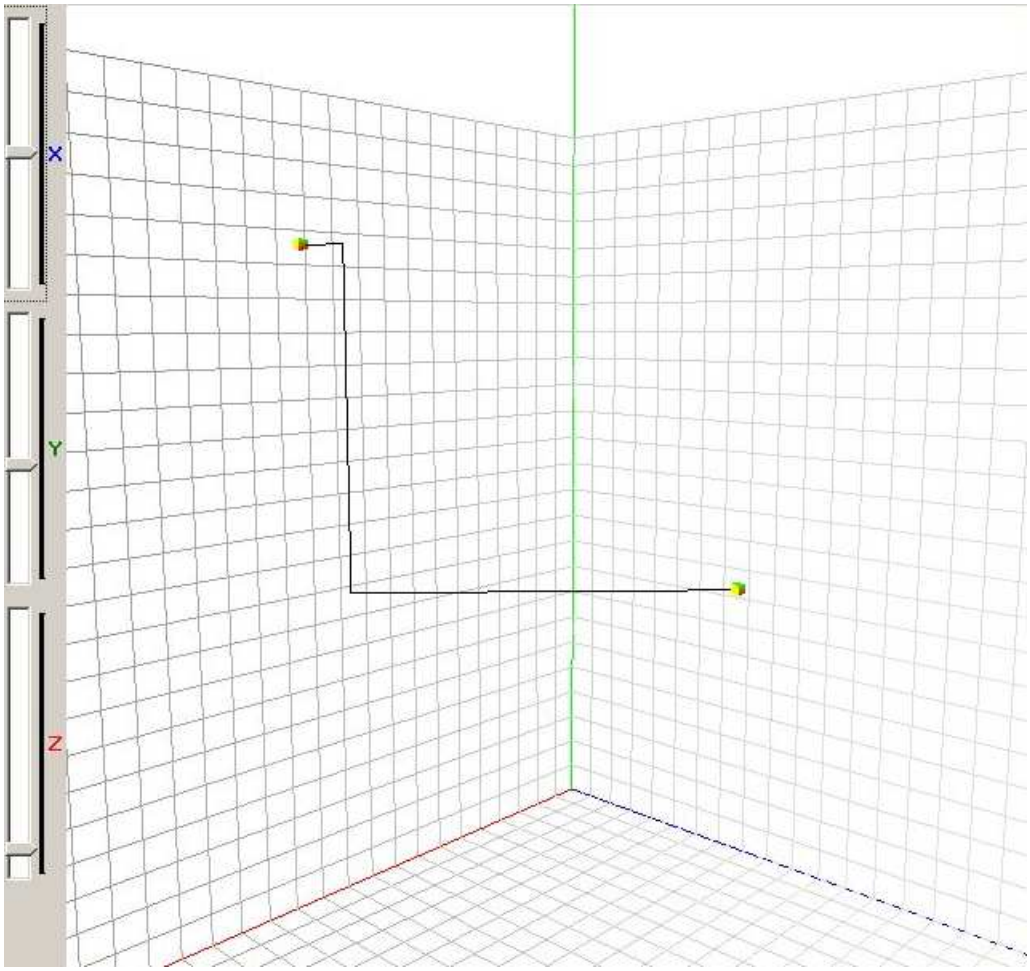


FIG. 5.26: Trajetória para uma situação sem obstáculos em um espaço de trabalho tridimensional.

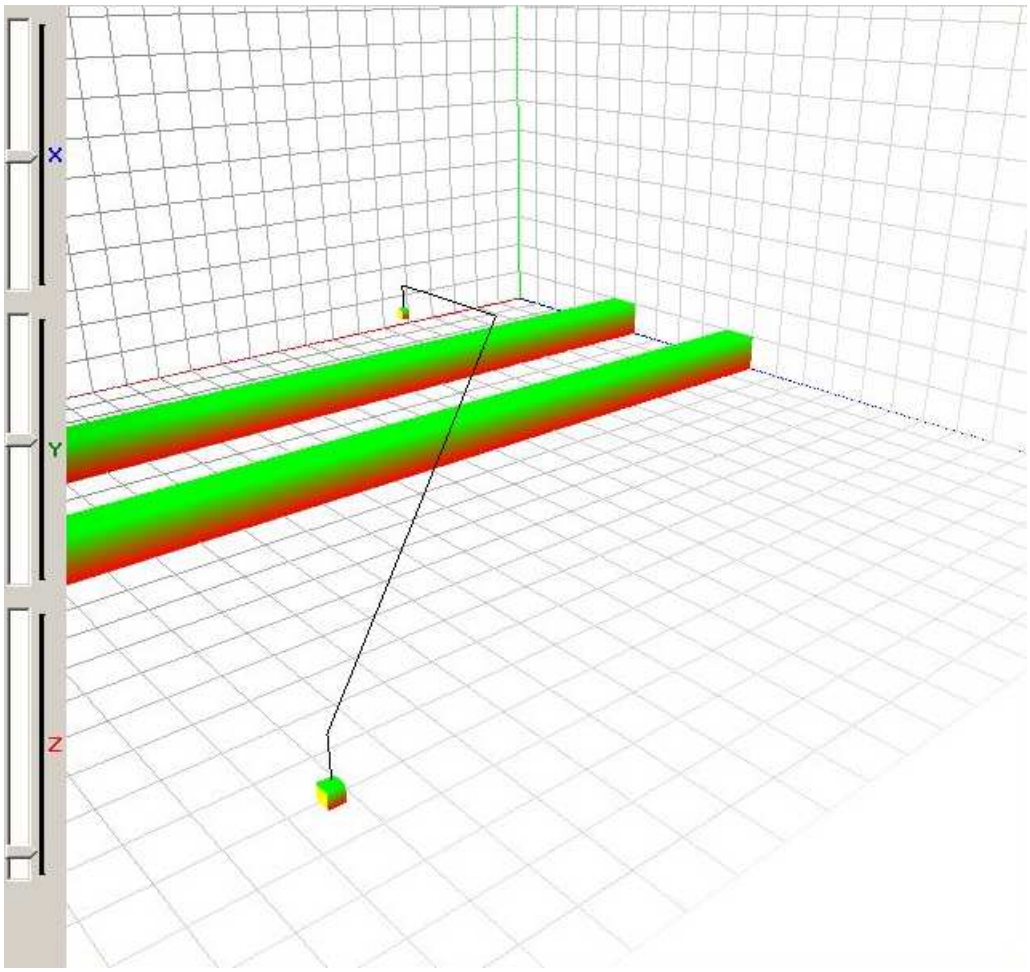


FIG. 5.27: Trajetória para uma situação com obstáculos estáticos em um espaço de trabalho tridimensional.

- com obstáculos estáticos, vide FIG. 5.27, FIG. 5.28 e FIG. 5.29;

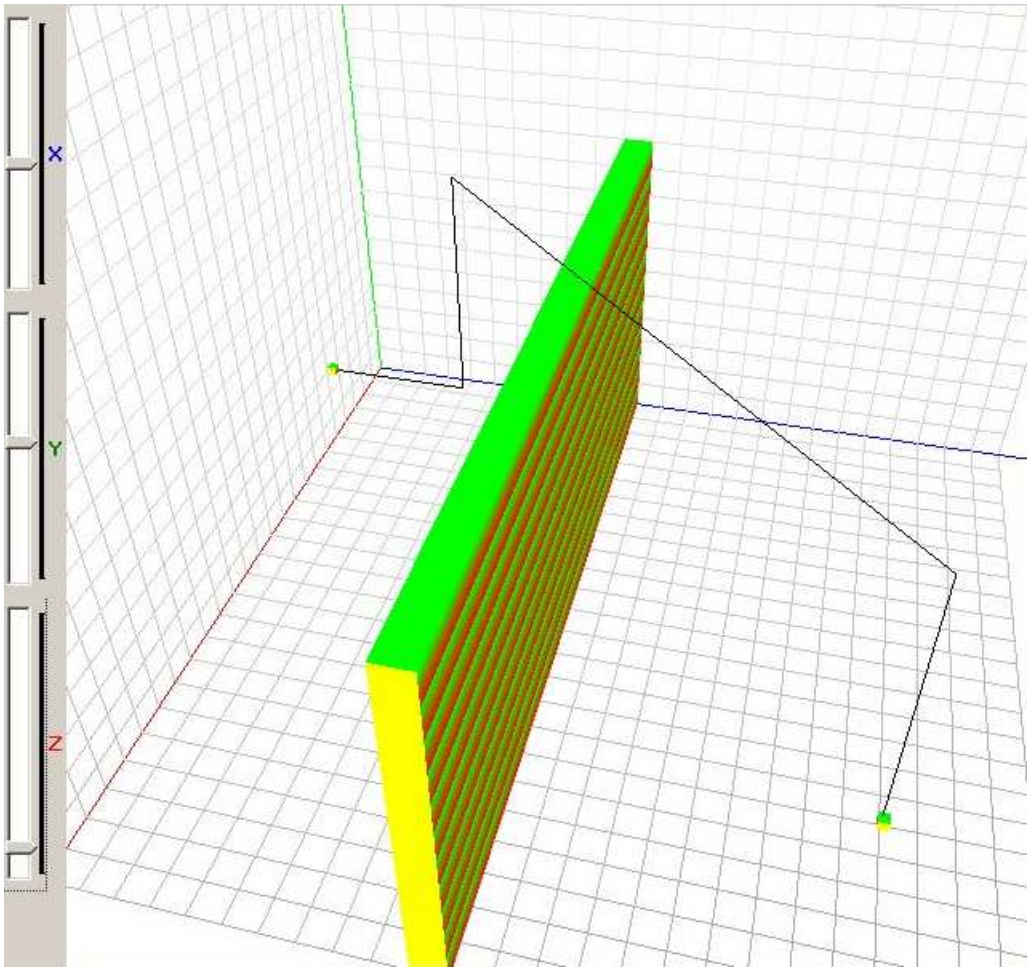


FIG. 5.28: Trajetória para uma situação com obstáculos estáticos em um espaço de trabalho tridimensional.

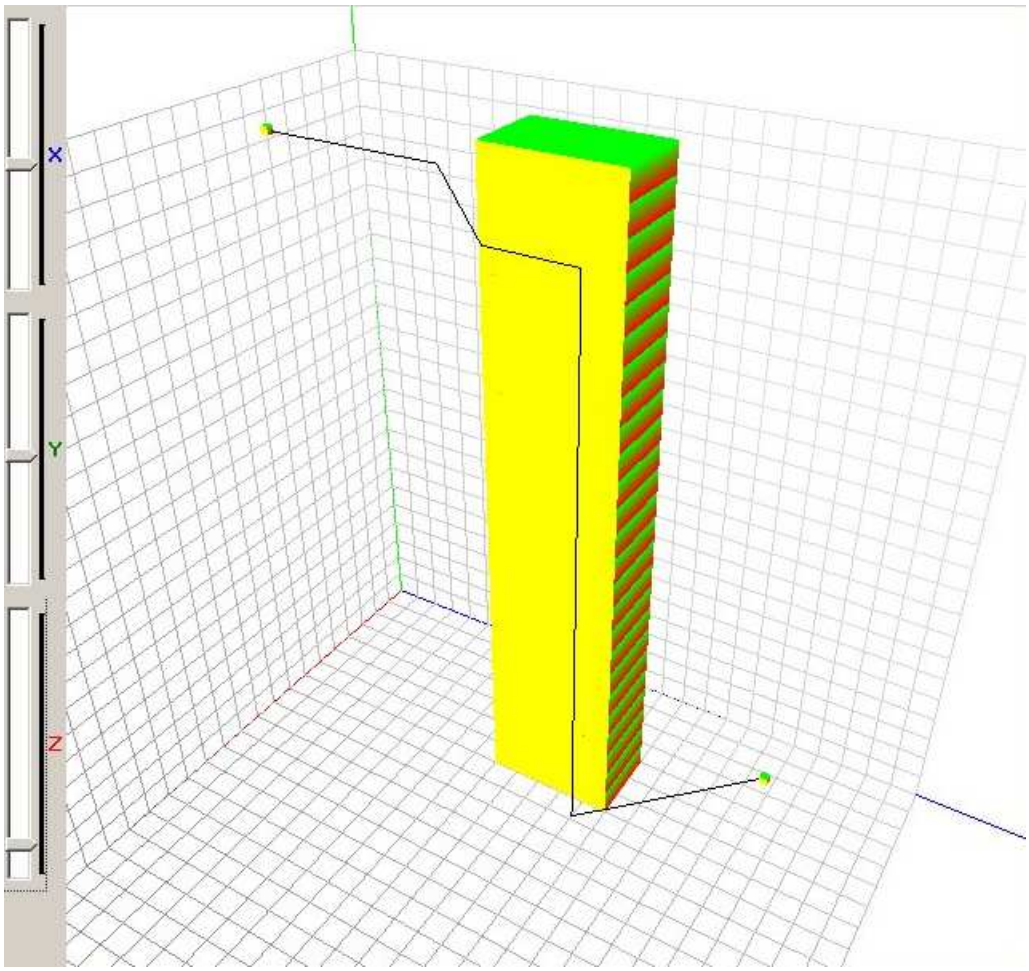


FIG. 5.29: Trajetória para uma situação com obstáculos estáticos em um espaço de trabalho tridimensional.

- com obstáculos móveis: são mostradas duas seqüências de imagens que mostram as adaptações efetuadas durante a execução da trajetória. A primeira seqüência é constituída pelas FIG. 5.30, FIG. 5.31, FIG. 5.32 e FIG. 5.33; e, a segunda seqüência é constituída pelas FIG. 5.34, FIG. 5.35, FIG. 5.36, FIG. 5.37 e FIG. 5.38.

Os resultados do planejamento de trajetória, em espaço de trabalho tridimensional, possuem características semelhantes aos do caso bidimensional. Na ausência de obstáculos, tenta-se minimizar o número de variações de altitudes e as trajetórias geradas são diretas e com o mínimo de curvas possíveis. Já quando existem obstáculos, as trajetórias possuem poucas curvas e, também, tendem a contornar os obstáculos. Com obstáculos móveis, as adaptações foram feitas sem dificuldades mas o problema das mudanças bruscas de direção permaneceu.

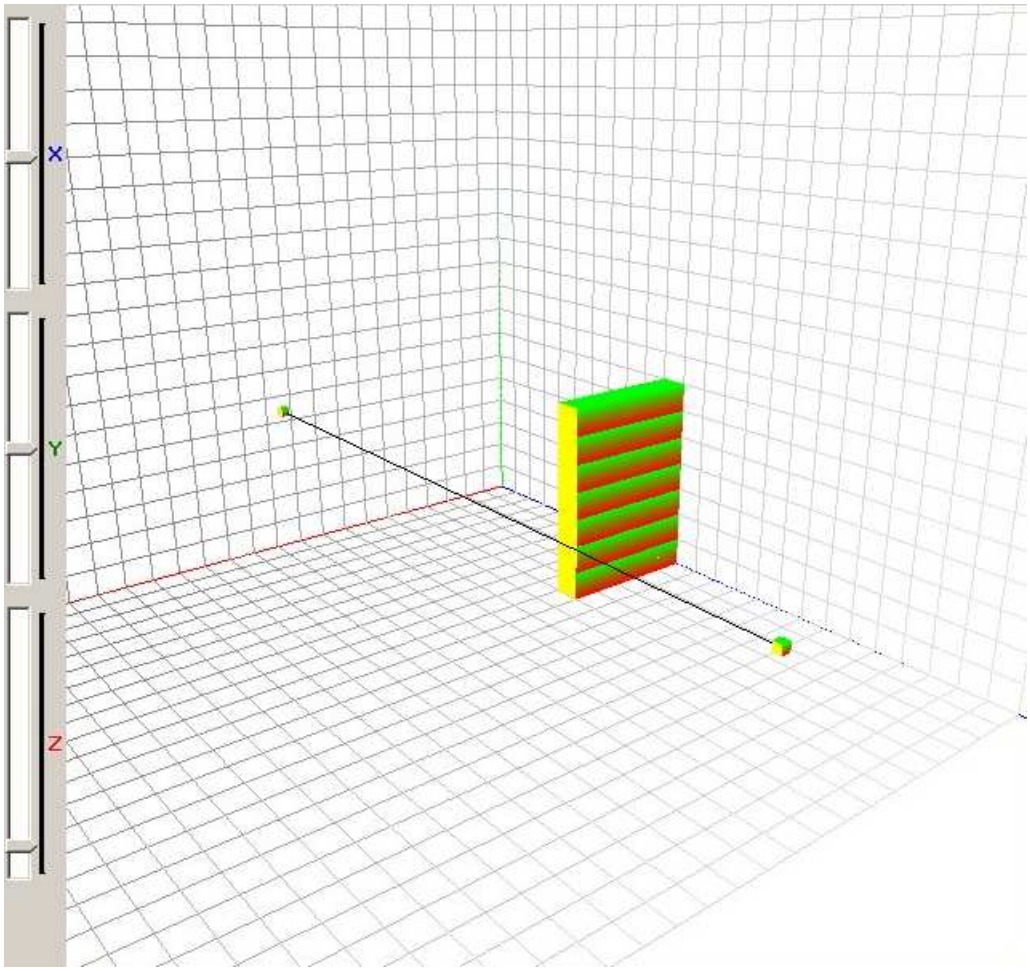


FIG. 5.30: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 1/4.

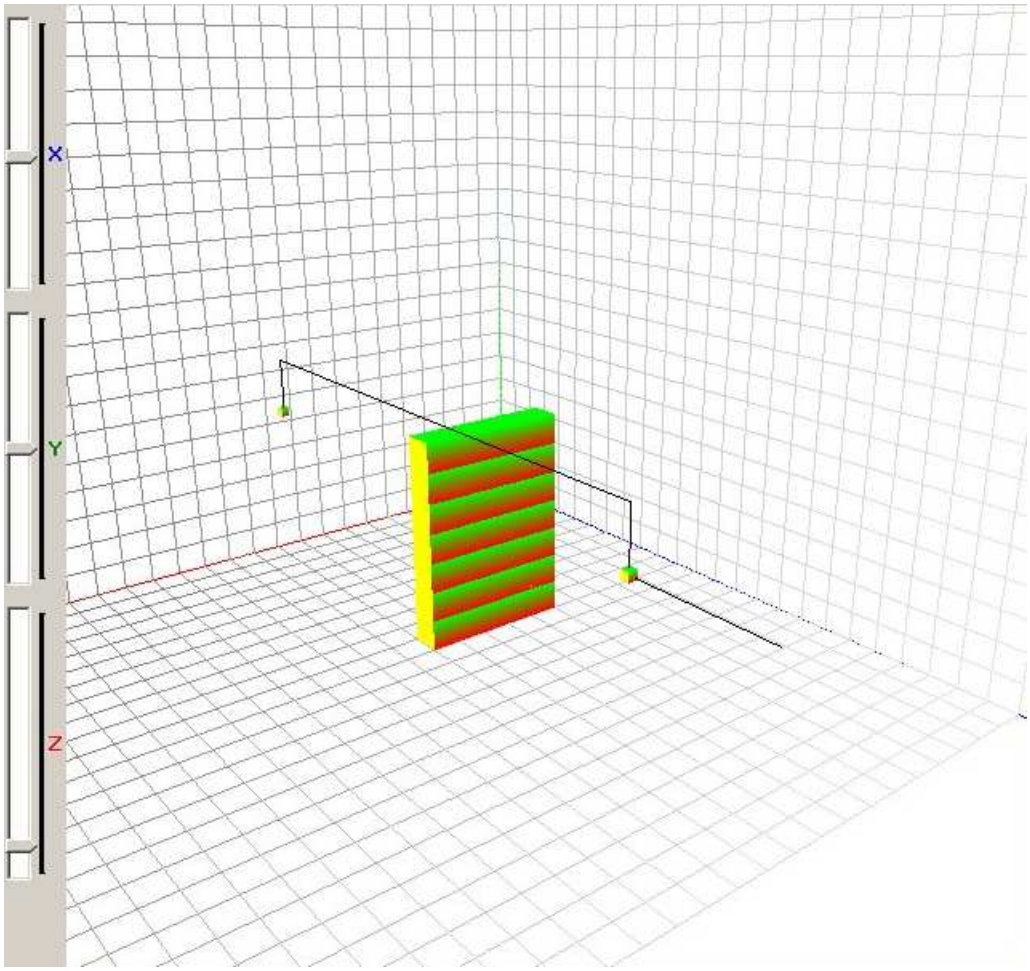


FIG. 5.31: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 2/4.

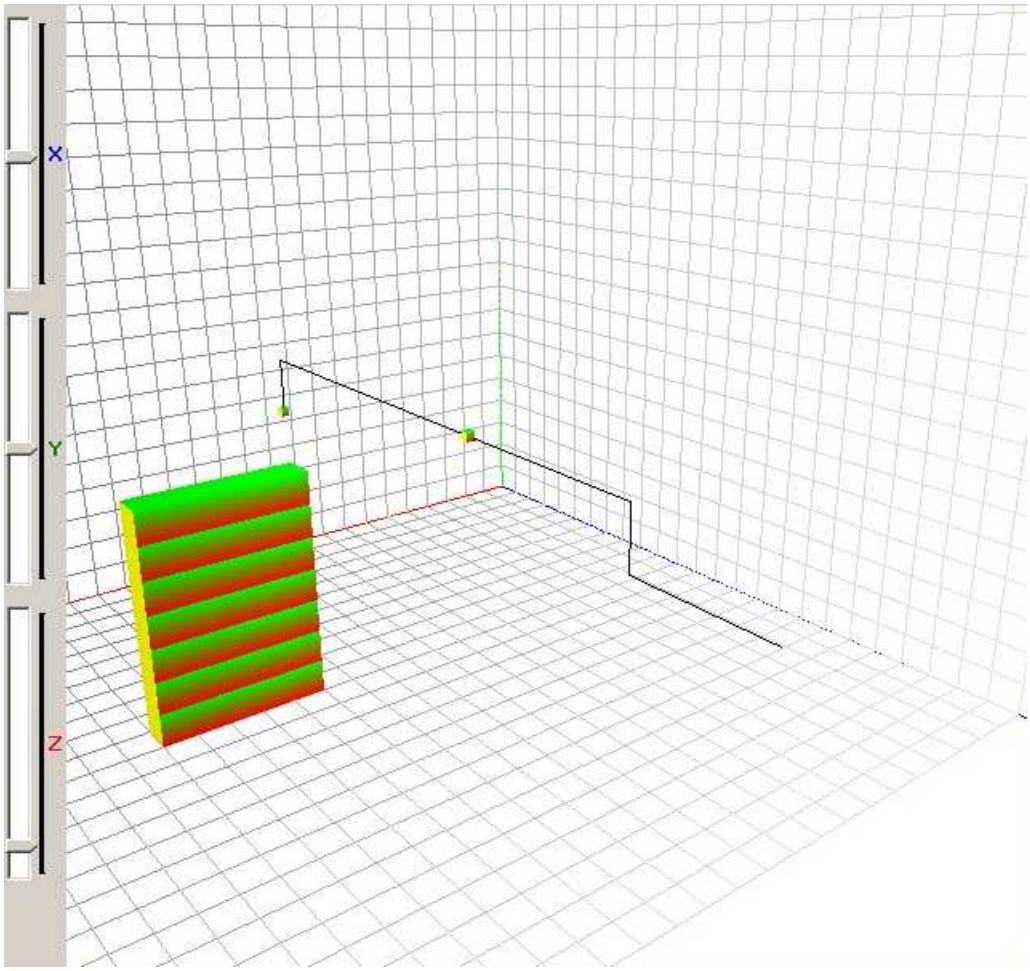


FIG. 5.32: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 3/4.

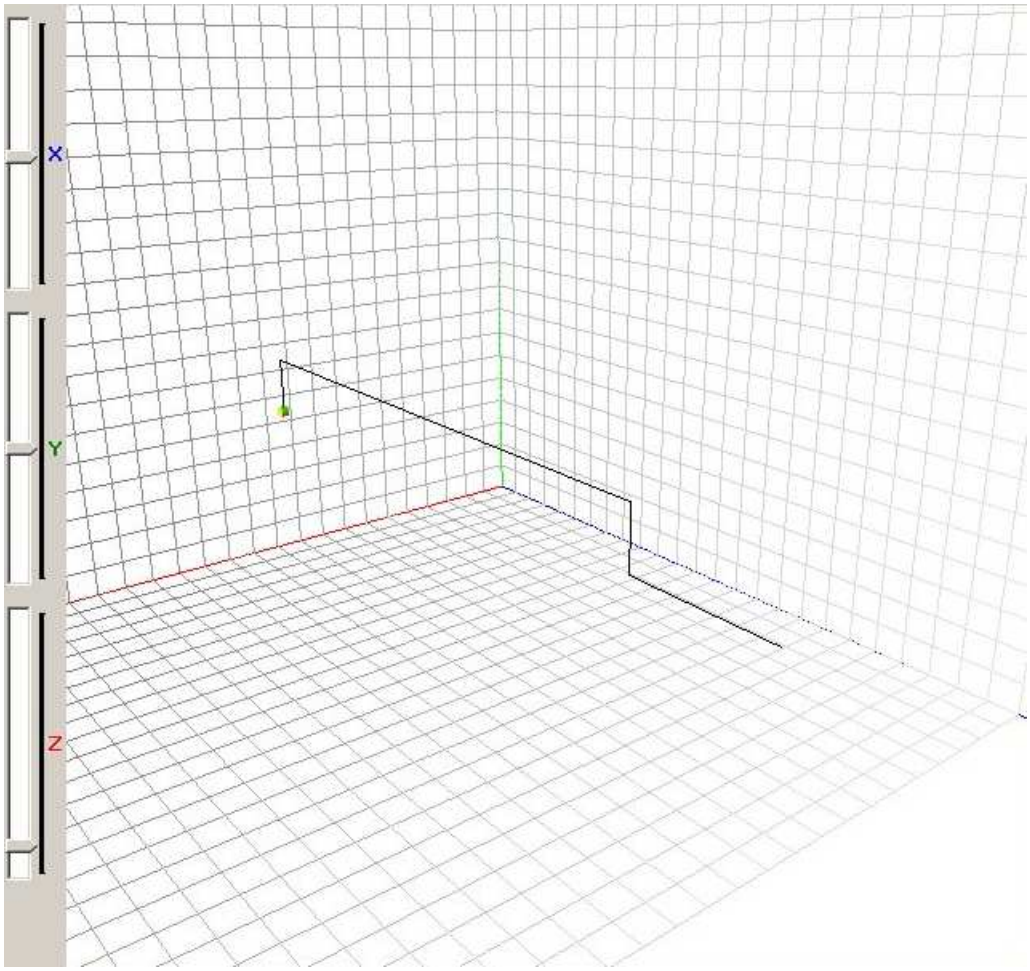


FIG. 5.33: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 4/4.

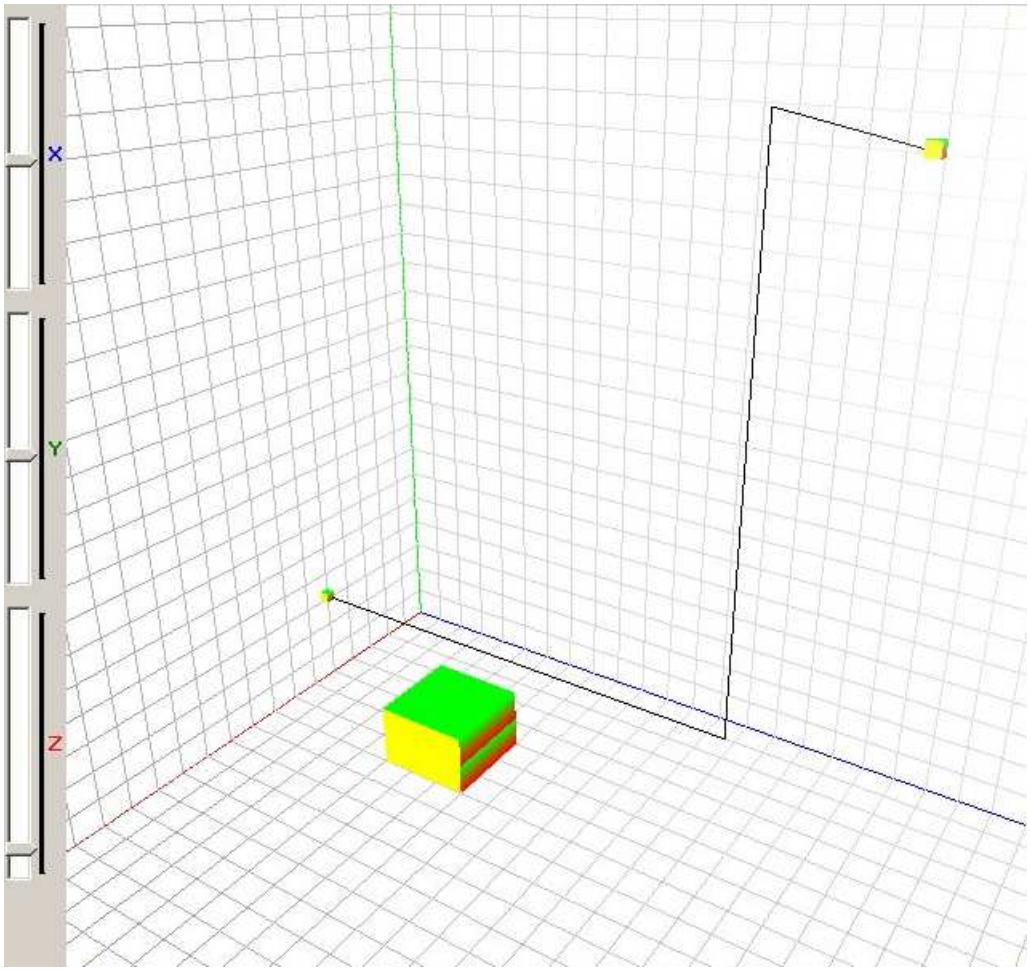


FIG. 5.34: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 1/5.

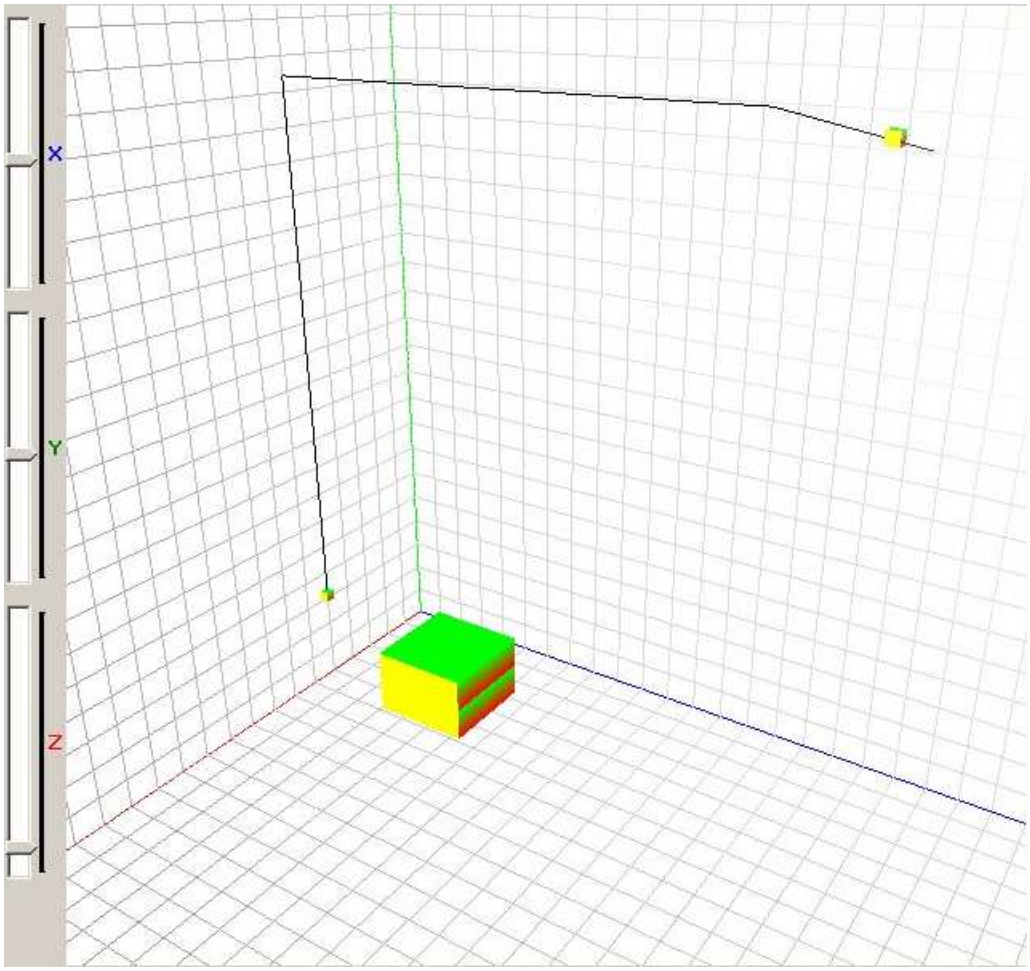


FIG. 5.35: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 2/5.

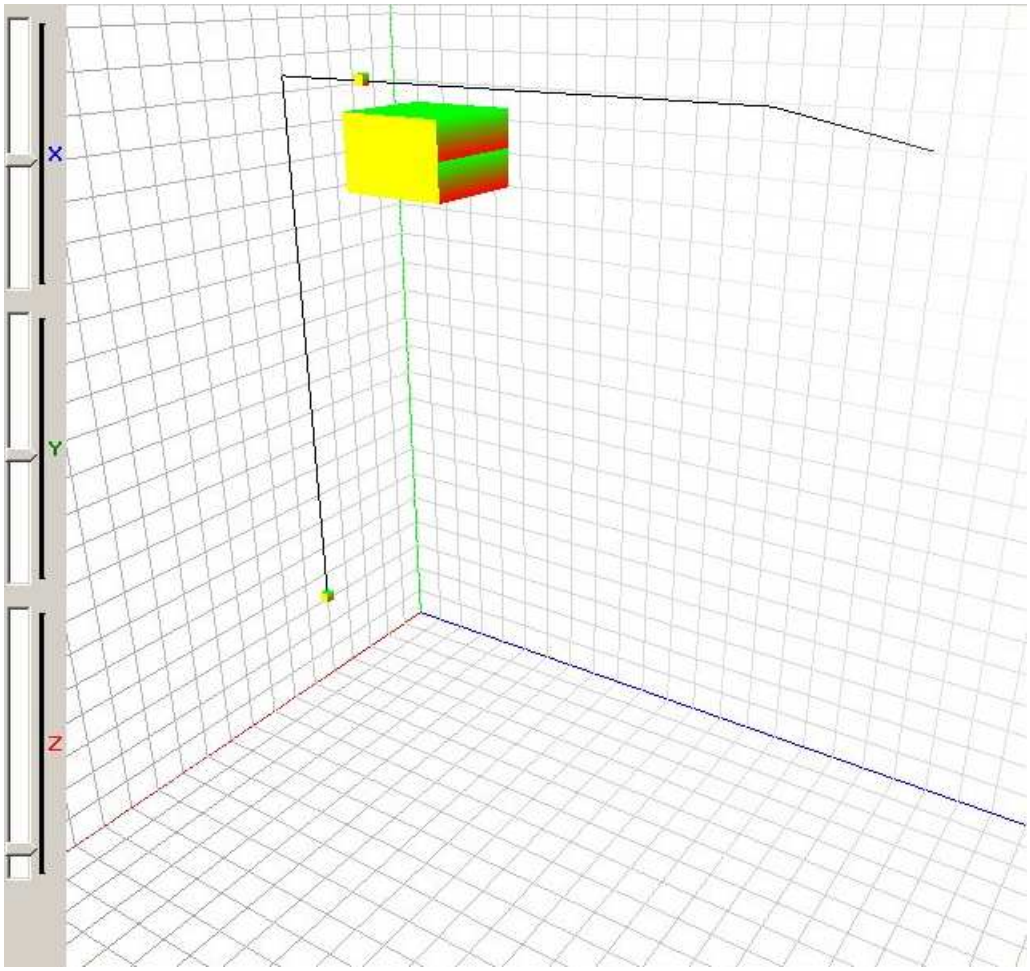


FIG. 5.36: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 3/5.

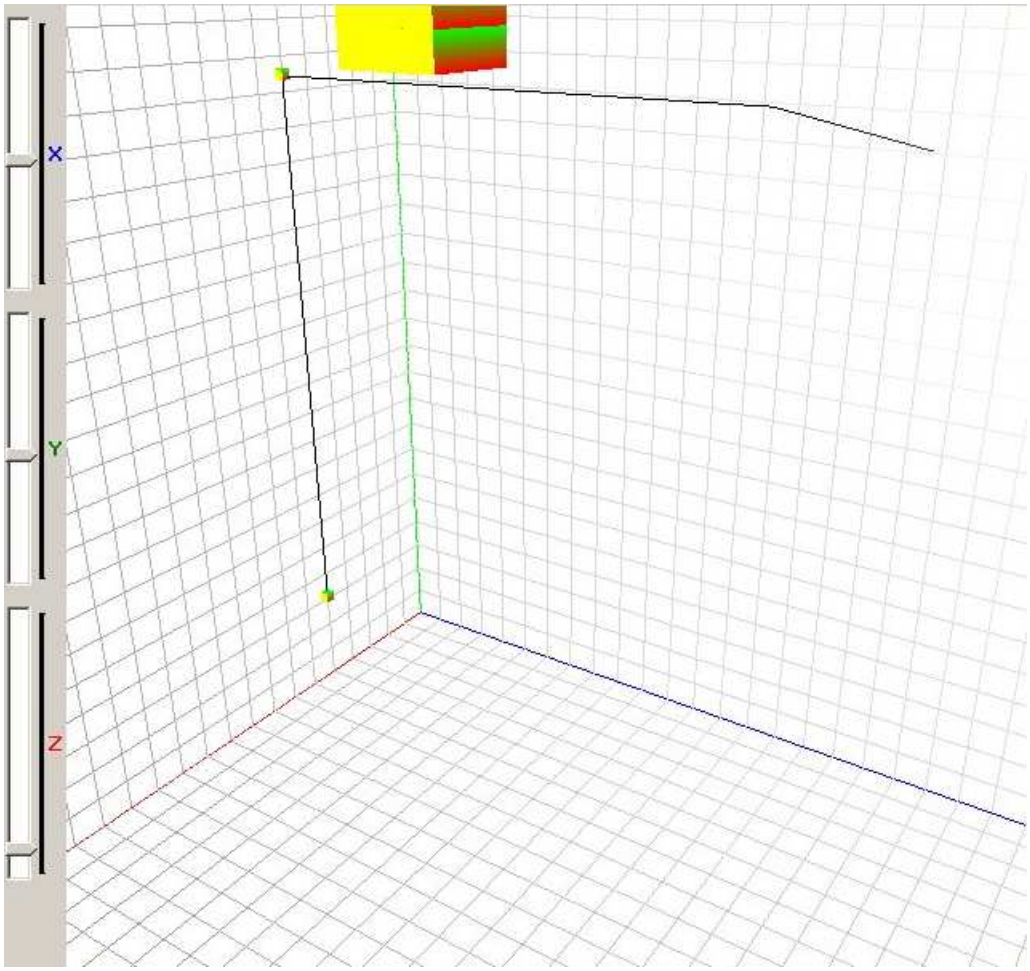


FIG. 5.37: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 4/5.

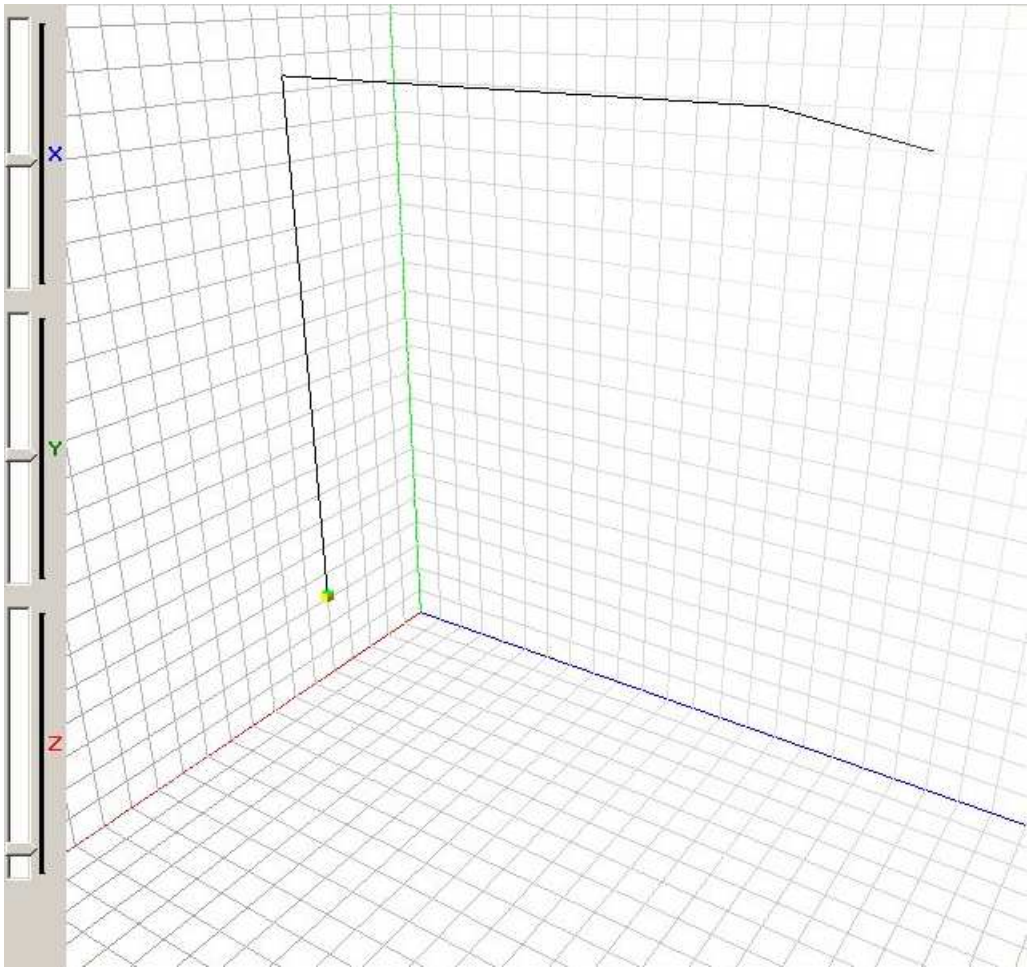


FIG. 5.38: Primeira seqüência de adaptações da trajetória com obstáculos móveis em um espaço de trabalho tridimensional 5/5.

6.1 CONCLUSÃO

Os veículos aéreos não-tripulados podem ser muito úteis para instituições civis ou militares em tarefas de vigilância e monitoramento, e reconhecimento de ambientes, entre outras; sobretudo, se os mesmos forem autônomos. Pois, desta forma, pode-se minimizar a intervenção humana durante a execução de alguma missão; reduzindo custos e riscos. Um dos requisitos, para que um VANT seja autônomo, é possuir seu próprio sistema de navegação. Este, deve obter dados, oriundos de sensores embarcados no veículo, e a partir deles conseguir calcular uma trajetória até um dado alvo, o que vem a ser uma tarefa bastante complexa.

Este trabalho tomou, como estudo de caso, um sistema de navegação para dirigíveis aéreos não-tripulados baseado em imagens. Portanto, foi proposta uma estratégia para o desenvolvimento do mesmo, abordando-se a problemática como um todo, atacando os pontos de mapeamento do ambiente e planejamento de trajetória.

No mapeamento do ambiente, foram utilizadas técnicas de visão computacional como visão estereoscópica, segmentação de imagens e calibração de câmeras. Os algoritmos de visão estereoscópica e calibração de câmeras são conhecidos e a implementação deles pode ser encontrada, parcialmente, em bibliotecas que tratam de algoritmos de visão computacional, como a OpenCV, que foi utilizada no desenvolvimento deste trabalho. Porém, a tarefa de segmentação de imagens, apesar de possuir soluções diversas, ser considerada como um problema particular. Pois, os métodos de segmentação são desenvolvidos de acordo com a situação do conjunto de imagens a ser segmentadas; levando em consideração fatores como condições de iluminação, formato dos objetos, e o contraste entre eles, entre outros. Portanto, será necessário um estudo mais aprofundado, e detalhado, do problema de segmentação de imagens deste trabalho.

A parte do trabalho relacionada ao planejamento de trajetória, depende diretamente do

mapeamento do ambiente. Como o ambiente foi mapeado com o método de decomposição em células, chegou-se a um problema de otimização. O uso da metaheurística Colônia de Formigas foi de grande importância no processo e mostrou bons resultados. Porém, as trajetórias obtidas ainda devem ser refinadas para melhor adaptação às limitações do veículo em questão.

Cada um dos pontos citados possuem uma complexidade considerável; este fato pode ser observado através de trabalhos científicos que tratam de cada um deles, especificamente. O que impediu que se chegasse a um aprofundamento mais detalhado em cada um dos tópicos inerentes ao problema. Contudo, os resultados obtidos foram promissores.

A seguir, são sugeridos alguns trabalhos futuros para continuidade e melhoramento do sistema de navegação, aqui proposto.

6.2 TRABALHOS FUTUROS

Apesar de apresentar resultados promissores, como alguns pontos não foram aprofundados como deveriam ser, são sugeridos alguns trabalhos futuros para que se tenha um sistema de navegação mais confiável e que possa, realmente, ser embarcado. São eles:

- Fazer o modelo dinâmico próprio, para um dirigível da instituição;
- Adaptação da trajetória planejada às propriedades físicas da aeronave;
- Migrar o sistema de navegação para uma plataforma móvel, a qual possa ser embarcada no dirigível;
- Desenvolver um método particular de segmentação de imagens, para obtenção de resultados mais seguros;
- Utilizar, no mapeamento do ambiente, dados oriundos de outros dirigíveis, tornando o sistema cooperativo.

6.3 AGRADECIMENTOS

Agradeço ao Instituto Militar de Engenharia pela oportunidade de fazer o curso de mestrado em Sistemas e Computação, pelo qual desenvolvi o presente trabalho, e por

toda a estrutura disponibilizada.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES por viabilizar este estudo através de financiamento de bolsa de estudo e aquisição de equipamentos.

Ao professor Júlio César Silva Neves pela orientação na parte de otimização do algoritmo de planejamento de trajetória.

Ao 1º Ten Bruno Madeira pelos conhecimentos compartilhados sobre conceitos de computação gráfica e visão computacional.

ALVARES, L. O., SICHMAN, J. S. **Introdução aos Sistemas Multiagentes.** Publicado em: Jornadas de Atualização de Informática - XVI JAI. XVII Congresso da Sociedade Brasileira de Computação, Brasília, 2-8 de Agosto, 1997

BAUER, B., MLLER, J. P., ODELL, J. **Agent UML: A Formalism for Specifyng Multiagent Interation.** Agent-Oriented Software Engineering. Paolo Ciancarini and Michael Wooldridge eds., Springer-Verlag, Berlin, pags: 91-103, 2001

BOTELHO, W. T. **Um Sistema de Identficação e Adaptação Pervasivo para a Casa Inteligente Utilizando Sistemas Multiagentes.** Dissertação de Mestrado. Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 2005.

BRIOT, J. P., DEMAZEU, Y. **Principes et Architecture des Systèmes Multiagents.** Paris. Hermes, 2002

COELHO, L. S., NETO, R. F. T., **Colônia de Formigas: Uma Abordagem Promissora para Aplicações de Atribuição Quadrática e Projeto de Layout** Publicado em XXIV ENEGEP. Florianópolis, SC, Brasil, 03 a 05 de novembro de 2004.

DELOACH, S. A., **Analysis and Design Using MaSE and AgentTool.** Proceedings of the 12th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS 2001). Miami University, Oxford, Ohio, March 31, Abril 2001a

DELOACH, S. A., WOOD, M. F. **Developing Multiagent Systems with AgentTool.** Proceedings of the Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2001b.

DELOACH, S. A., WOOD, M. F., SPARKMAN, C. H. **Multiagent Systems Engineering.** International Journal of Software Engineering and Knowledge Engineering. Vol. 11, N 3, pags: 231-258. World Scientifc Publishing Company, 2001c

DISSANAYAKE, M. W. M. G., NEWMAN, P., CLARK, S., DURRANT-WHYTE, H., CSORBA, M. **A Solution to the Simultaneous Localization and Map Building (SLAM) Problem.** Proceedings of the IEEE Transactions on Robotics and Automation, Vol. 17, N 3. June 2001.

DORIGO M.; DI CARO, G., CAMBARDELLA L. M. **Ant Algorithms for Discrete Optimization.** Proceedings in Artificial Life, 1999. Vol.5, N 2, p. 137-172.

DURFEE, E. H., LESSER, V. R., CORKILL, V. R. **Trends in cooperative distrubuted problem solving.** IEEE Transactions on Knowledge and Data Engineering, Vol. 1, N. 1, pags: 63-83, 1989

ELFES, A., BUENO, S. S., BERGERMAN, MARCEL, RAMOS, J. J. G. **A Semi-Autonomous Robotic Airship for Environmental Monitoring Missions.** Proccedings of the 1998 IEEE. International Conference on Robotics & Automation. Leuven, Belgium. May 1998.

ELIAZAR, A., PARR, R. **DP-SLAM: Fast, Robust Simultaneous Localization and Mapping Without Predetermined Landmarks.** Proceedings of the 18th International Joint Conference on Artificial Intelligence, pages 1135-1142. Acapulco, México. August, 2003.

ESPINHOSA, D. R. S. **Influência da Injunção da Base na Fototrinagulação de Imagens Obtidas com uma Unidade de Mapeamento Móvel.** Dissertação de Mestrado. Universidade Estadual Paulista - Faculdade de Ciências e Tecnologia. Presidente Prudente, 2006

FARIA, B. G. **Identificação Dinâmica Longitudinal de um Dirigível Robótico Autônomo.** Dissertação de Mestrado. Universidade Estadual de Campinas, Campinas, Brasil, 2005.

FENWICK, J. W., LEONARD, J. J. **Cooperative concurrent mapping and localization.** Proceedings of the IEEE International Conference on Robotics & Automation, vol. 2, pages 1810-1817. Washington, USA. May 2002.

FLOURET, M., SIMON, G., MERMET, B. **Vers une méthodologie de développement de SMA adaptés aux problèmes d' optimisation** Journées Francophones IAD et SMA. Lille, France, 2002.

FRANCOIS, S., MLLER, J. **AMACOIA: A Multiagen System for Designing Flexible Assembly Lines.** Applied Artificial Intelligence. Vol. 11, N 6, pags: 573-589, 1997

FRANKLIN, S., GRAESSER, A. **Is it an agente, or just a program?: A taxonomy for autonomous agentes.** Proccedings of the Springer-Verlag, ECAI'96. Thrid International Workshop on Agente Theoris, Architectures, and Languages: Intelligent Agents III, pages 21-36, August 1996. LNAI, Vol. 1193

GOLDBARG, M. C., LUNA, H. P. L. **Otimização Combinatória e Programação Linear: Modelos e Algoritmos.** Editora Campos, 2 Edição, 2005.

GOMES, H. A. S. **Utilização da Metaheurística *Simulated Annealing* no Problema de Alocação de Pessoal em Empresas de Transporte Coletivo por Ônibus.** Dissertação de Mestrado. Universidade Federal do Ceará Fortaleza, Brasil, 2003

GOMES, J., VELHO, L. **Fundamentos da Computação Gráfica.** Instituto Nacional de Matemática Pura e Aplicada- IMPA. Série de Computação e Matemática. Rio de Janeiro, 2003

GOMES, S. B. V., RAMOS, J. J. G. **Airship Dynamic Modeling for Autonomous Operation.** Proccedings of the 1998 IEEE. International Conference on Robotics & Automation. Leuven, Belgium. May 1998.

GONZALES, R. C., WOODS, R. E. **Processamentos de Imagens Digitais.** Editora Edgar Blcher Ltda. São Paulo, 1992

HIMA, S., BESTAQUI, Y. **Motion Generation on Trim Trajectories for an Autonomous Underactuated Airship.** Proceedings in 4th International Airship Convention and Exhibition. Jul 28-31, 2002. Cambridge, England.

HORN, B. K. P. **Robot Vision.** The MIT Press, 1986

HBNER, J. F, SICHMAN, J. S. **Um Modelo de Reorganização de Sistemas Multiagentes.** Tese de Doutorado. Universidade de São Paulo. São Paulo, Brasil, 2003

HUGUET, A. B. **Reconstrução de Cenas Urbanas Baseada em Estereoscopia e Segmentação por Watershed.** Dissertação de Mestrado. Universidade Federal de Minas Gerais - UFMG. Belo Horizonte, 2003

Intel Corporation. **Open Source Computer Vision Library - Reference Manual.** Copyright Intel Corporation, 1999-2001

JENNINGS, N. R., WOOLDRIDGE, M. J. **Agent Technology: Foundations, Applications, and Markets.** Springer-Verlag. London, 1998

JIANG, Y., ZHAO, M., WANG, H., FANG, L. **Hibrid Navigation for a Climbing Robot by Fuzzy Neural Netowrk and Trajectory Planning.** Proceedings of the Fourth International Conference on Machine Learning and Cybernetics. Guangzhou, 2005.

KIGUCHI, K, WATANABE, K., FUKUDA, T. **Trajectory Planning of Mobile Robots Using DNA Compunting.** Proceedings of IEEE International Symposium on Computacional Intelligence in Robotics and Automation. Banff, Alberta, Canada, 2001.

KÖTHE, U. **Generishe Programmierung fr die Bildverarbeitung.** Dissertation, Fachbereich Informatik, Universität Hamburg, Hamburg, 2000

LAMB, H **The Inertia Coefficients of an Ellipsoid Moving in Fluid.** British Aeronautical Research Commitee Report and Memoranda N 623, 1918

LATOMBE, J. C. **Robot Motion Planning.** Kluwe Academic Publishers, Boston, 1991.

LEE, D. T., SCHACHTER, B. J. **Two Algorithms for constructing a Delaunay Triangulation** International Journal ofParallel Programming, Volume 9, N 3. Springer Netherlands, 1980

LEE, J., NAN, H. S., LYOU, J. **A practical collision-free trajectory planning for two robot systems.** Proceedings of the IEEE International Conference on Volume 3, Issue, 1995, p. 2439-2444.

LEE, R. C., TEPFENHART, W. M. **UML e C++ - Guia Prático de Desenvolvimento Orientado a Objeto.** Editora Makron Books Ltda. São Paulo, 2001

MARIETTO, M. G. G. **Definição Dinâmica de Estratégias Instrucionais em Sistemas de Tutoria Inteligente: Uma Abordagem Multiagentes na WWW.** Tese de Doutorado, Instituto Tecnológico de Aeronáutica - ITA, 2000

MATSON, E., DELOACH, S. **Organization Model for Cooperative and Sustaining Robotic Ecologies.** Proceedings of the Robosphere, 2002

MENEZES, P. J. C. **Estudos em Navegação de Robôs Móveis.** Dissertação de Mestrado, Universidade de Coimbra, 1999

MONTEMERLO, M., THRUN, S., KOLLER, D., WEGBREIT, B. **FastSlam 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges.** Proceedings of the 18th International Joint Conference on Artificial Intelligence. Acapulco, México. August, 2003.

MONTEMERLO, M., THRUN, S., WEGBREIT, B. **A Factored Solution to the Simultaneous Localization and Mapping Problem.** Proceedings of the Eighteenth National Conference on Artificial Intelligence, pages 593-598. Edmonton, Alberta, Canada. Published by The AAAI Press, Menlo Park, California. July, 2002.

MOSTAFA, M. M. R., SCHWARZ, K. P. **Digital Image Georeferencing From a Multiple Camera System by GPS/INS.** ISPRS Journal of Photogrammetry & Remote Sensing, 56, p. 1-12.

MUNK, M. M. **Aerodynamics of Airships** Aerodynamics Theory, Vol. 6, Editora Julius Springer, Berlin, 1936

NEWMAN, P. M. **On the Structure and Solution of the Simultaneous Localization and Map Building Problem.** PhD. Thesis. Australian Center for Field Robotics, The University of Sydney. 1999.

PIO, J. L. S., CAMPOS, M. F. M. **Navegação Robótica.** Publicado em: Jornada de Atualização em Informática - Livro Texto. 1 ed. Campinas: SBC - Sociedade Brasileira de Computação, v. II, p.389-438.

PINHEIRO, C. A. P. **Veículos Aéreoes Autônomos Não-Tripulados para Monitoramento de Ambientes Desestruturados e Comunicação de Dados.** Dissertação de Mestrado. Insitituto Militar de Engenharia, Rio de Janeiro, Brasil, 2006.

PRECHELT, L. **An Empirical Comparision of C, C++, Java, Perl, Python, Rexx and TCL for a Search/String-processing Program.** Technical Report 2000-5. Universität Karlsruhe. Karlsruhe, Germany, 2000

RAMOS, J. G., BRUCIAPAGLIA, A., BUENO, S. **An Internet Based Airship Simulator.** Technical Report LRV-1997-13, Automation Institute-CTI

RAMOS, J. J. G., MAETA, S. M., MIRISOLA, L. G. B., BERGERMAN, M., BUENO, S. S., PAVANI, G. S., BRUCIAPAGIA, A. **A Software Environment for an Autonomous Unmanned Airship.** Proceedings of the 1999 IEEE/ASME. International Conference on Advanced Intelligent Mechatronics. September 19-23. Atlanta, USA.

RAMOS, J. J. G., PAIVA, E. C., BUENO, S. S., MAETA, S. M., MIRISOLA, L. G. B., BERGMAN, M., FARIA, B. G. **Autonomous Flight Experiment With a Robotic Unmanned Airship.** Proceedings of the 2001 IEEE. Seoul, Korea. May 21-26, 2001 International Conference on Robotics & Automation.

REEVES, C. R. **Modern Heuristic Techiniques for Combinatorial Problems.** Blackwell Scientific Publications, Osney Mead, Oxford, 1993

RUSSELL, S.,NORVIG, P. **Inteligência Artificial.** Editora Campus. Rio de Janeiro, 2004.

SANTOS, L. P. P. **Visão por Computador.** Universidade do Minho, Escola de Engenharia - Departamento de Informática. Braga, 1995

SEGA, M., AKELEY, K. **The OpenGL Graphics System: A Specification.** Silicon Graphics, Inc., 2006

SCHEPKE, C., CHARÃO, A. S. **Comparação entre Java e C++ na Computação Numérica.** Quinto Workshop em Sistemas Computacionais de Alto Desempenho. Foz do Iguaçú, Brasil, 2004

SCHWAMBACH, M. M. **OplA: Uma Metodologia para Desenvolvimento de Sistemas Baseados em Agentes e Objetos.** Dissertação de Mestrado. Universidade Federal do Espírito Santo - UFES. Vitória, Brasil, 2004

SICHMAN, J. S. **Du Raisonnment Social Chez Lez Agents: Une Approche Fondeé Sur La Théorie de la Dépendance.** Tese de Doutorado. Institut National Polytechnique de Grenoble, 1995

SIMÕES, A. S. **Segmentação de Imagens por Classificação de Cores: Uma abordagem Neural.** Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo. São Paulo, 2000

SINOPOLI, B., MICHELI, M., DONATO, G., KOO, T. J. **Vision Based Navigation for an Unmanned Aerial Vehicle.** Proceedings of the IEEE International Conference on Robotics an Automation. Seoul, Korea. May, 2001.

SMITH, R. G. **The Contract Net Protocol: High-Level communication and Control in a Distributed Problem Solver.** IEEE Transaction on Computers. Vol. 29, N 12, pags: 1104-1113

SOUSA, C. M. F. **A Integração do Sistema GPS/INS para a Monitorização da Linha de Costa do Litoral do Algarve.** Dissertação de Mestrado. Faculdade de Ciências, Universidade de Lisboa. Portugal, Lisboa, 2004

THRUN, S. **Learning metric-topological maps for indoor mobile navigation.** Proceedings of the Artificial Intelligence, 99(1):21-71, February 1998.

VELHO, L, GOMES, J **Sistemas Gráficos 3D.** Instituto de Matemática Pura e Aplicada - IMPA. Série de Computação e Matemática. Rio de Janeiro, 2001

WEISS, G. **Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence.** London, MIT Press, 1999. ISBN 0-262-23203-0

WILLIAMS, S. B., DISSANAYAKE, G., DURRANT-WHYTE, H. **An Efficient Approach to the Simultaneous Localization and Mapping Problem.** Proceedings of the 2002 IEEE. International Conference on Robotics & Automation. Washington, USA. May 2002.

WOOD, M. F., DELOCACH, S. A. **An Overview of The Multiagent Systems Engineering Methodology.** Lecture Notes in Computer Science. Vol. 1957, Springer Verlag, Berlin, January 2001

WOOLDRIDGE, M., JENNINGS, N. R. **Pitfalls of agent-oriented development.** Proceedings of the 2nd International Conference on Autonomous Agents (Agents'98), pages 385-391, New York, 9-13 1998. ACM Press.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo