EXPLORAÇÃO E TRANSPORTE COOPERATIVO DE OBJETOS EM AMBIENTES DESCONHECIDOS

ELIZABETH DUANE SANTOS DA COSTA

EXPLORAÇÃO E TRANSPORTE COOPERATIVO DE OBJETOS EM AMBIENTES DESCONHECIDOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Mario Fernando Montenegro Campos
Co-orientador: Prof. Pedro Mitsuo Shiroma

Belo Horizonte
29 de março de 2012

Costa, Elizabeth Duane Santos da
C837e Exploração e transporte cooperativo de objetos em ambientes desconhecidos / Elizabeth Duane Santos da Costa. — Belo Horizonte, 2012
xxvi, 90 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais- Departamento de Ciência da Computação.

Orientador: Prof. Mario Fernando Montenegro Campos
Co-orientador: Prof. Pedro Mitsuo Shiroma

1. Computação - Teses. 2. Robótica - Teses. I. Orientador. II. Título.

CDU 519.6*82.9(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Exploração e transporte cooperativo de objetos em ambientes desconhecidos

ELIZABETH DUANE SANTOS DA COSTA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. MÁRIO FERNANDO MONTENEGRO CAMPOS - Orientador
Departamento de Ciência da Computação - UFMG

PROF. PEDRO MITSUO SHIROMA - Co-orientador
Departamento de Ciência da Computação - UFSJ

PROF. LUIZ CHAIMOWICZ
Departamento de Ciência da Computação - UFMG

PROF. MARCOS AUGUSTO MENEZES VIEIRA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 29 de março de 2012.

Aos meus pais, Elias e Ladyr.

Agradecimentos

Ao chegar no fim dessa etapa e olhar para todo o processo, sei que não seria possível concluí-lo sem a ajuda de muitos.

Agradeço aos meus pais, por serem meus grandes incentivadores e pela estrutura familiar que me proporcionaram. Um simples agradecimento não seria suficiente para expressar a importância de vocês neste processo. Aos meus irmãos, avós e demais familiares, agradeço pelo carinho, preocupação e compreensão nesse período de grande ausência. Agradeço ao meu querido Junior, pela paciência, compreensão, carinho e incentivo que foi fundamental em vários momentos.

Agradeço ao meu orientador, professor Mario Campos, por todo o aprendizado adquirido nesse período, pela paciência e conversas de incentivos nos momentos necessários. Agradeço ao meu co-orientador professor Pedro Shiroma pela disponibilidade e pelas valiosas dicas principalmente, nos processos de implementação.

Aos colegas do VeRLab, com quem pude aprender muito nesse período, pelas discussões e momentos de descontração. Em especial ao Samuel e Gabriel pela grande disponibilidade e ajuda em diversas situações. Yuri, Erickson, Douglas, Wilson, Armando e Vinícius, agradeço pelas discussões e pelas dicas. Armando, pelas revisões do texto. Renato, por todas as dicas com o sistema *Silvver*. Ao Wolmar por toda ajuda na montagem dos robôs.

Seja nas questões acadêmicas, seja uma conversa informal ou mesmo uma palavra de incentivo. Agradeço a todos que muito ou pouco contribuíram para eu concluir esta etapa.

Agradeço a Deus, por sua grande fidelidade e por me capacitar e permitir chegar até aqui. Sem Ele nenhum agradecimento poderia ser feito.

"Pensava que nós seguíamos caminhos já feitos,
mas parece que não os há.
O nosso ir faz o caminho."
(C. S. Lewis)

Resumo

Este trabalho apresenta um arcabouço para identificação e transporte de objetos realizado por um grupo de robôs móveis em um ambiente desconhecido. Tais objetos possuem forma, tamanho e peso variados e podem ser transportados de duas maneiras: carregados com o auxílio de um braço manipulador acoplado ao robô (*grasping*) ou empurrados por um ou mais robôs (*pushing*). De acordo com a capacidade de atuação dos robôs e as características do objeto, o que leva ao questionamento de qual modo deve ser invocado e quais robôs devem ser alocados para determinada tarefa de transporte.

A maioria dos trabalhos relatados na literatura abordam apenas o transporte, e partem do pressuposto que já conhecem a priori todas as informações necessárias dos objetos a serem transportados. Neste trabalho propõe-se que o robô descubra as características do objeto necessárias e tais informações serão usadas no processo de decisão do método de transporte. Para tal, o robô faz o mapeamento 2D de um determinado objeto utilizando um *laser* e estima o esforço necessário para transportá-lo por meio de medições da potência elétrica aplicadas ao motor. Com essas informações, o robô consegue decidir como o objeto será transportado e, caso seja necessário, solicita ajuda de outro(s) robô(s) para executar a tarefa.

Resultados dos experimentos simulados e em ambiente real foram utilizados para avaliar a eficácia e algumas limitações do arcabouço. Foi possível avaliar que o arcabouço proporciona a cooperação entre os robôs para a realização do transporte, embora algumas suposições na implementação possam vir tornar o sistema um pouco restrito.

Palavras-chave: Sistema de múltiplos robôs, robótica cooperativa, exploração de objetos, transporte cooperativo, recrutamento.

Abstract

This dissertation presents a framework for identification and transportation of objects arranged in an unknown environment. The objects' shape, size and weight can be varied and transported in one of ways: grasped by a manipulator coupled to the robot base or pushed. According to the robots' capacity and the object's characteristics, one transporting mode may be more adequate than the other. This leads determine which transporting mode should be performed and which robots should be allocated for the task.

Most of the approaches that address the cooperative transport consider that robots have prior information of the object characteristics. In this work the robot estimates the object's characteristics for the transportation decision process. For that, the robot uses a laser to map a 2D object. Furthermore, it estimates the effort required to push the object through estimates of the power applied to the motor. With this information the robot can decide how the object will be manipulated and if it will need to request help from other robots to perform the transportation.

We present simulation and experimental results to evaluate the framework efficacy and limitations. Although some of assumptions may limit the system, the framework enable cooperation among the robots to carry out the proposed task.

Keywords: Multi-robot system, cooperative robotics, multi-robot transport, object exploration, recruitment.

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

MRTA	*Multi-Robot Task Allocation* (Alocação de Tarefas para Multiplos Robôs)
RANSAC	RANdom SAmple Consensus
MRS	*Multi-Robot System* (Sistema Multi-Robô)
VCS	Vacancy Chain Scheduling
MAS	*Multi-Agent System* (Sistema Multi-Agente)
OAP	*Optimal Assignment Problem* (Problema de Atribuição Ótima)
VeRLab	Laboratório de Visão e Robótica
OpenCV	*Open Source Computer Vision*
DoF	*Degrees-of-Free* (Graus de Liberdade)
OAP	*Optimal Assignment Problem* (Problema de Atribuição Ótima)
GAP	*General Assignment Problem* (Problema de Atribuição Geral)
DCOP	*Distributed Constraint Optimization Problem* (Problema Otimização de Restrição Distribuída)
E-GAP	*Extended Generalized Assignment Problem* (Problema de Atribuição Geral Extendida)

Sumário

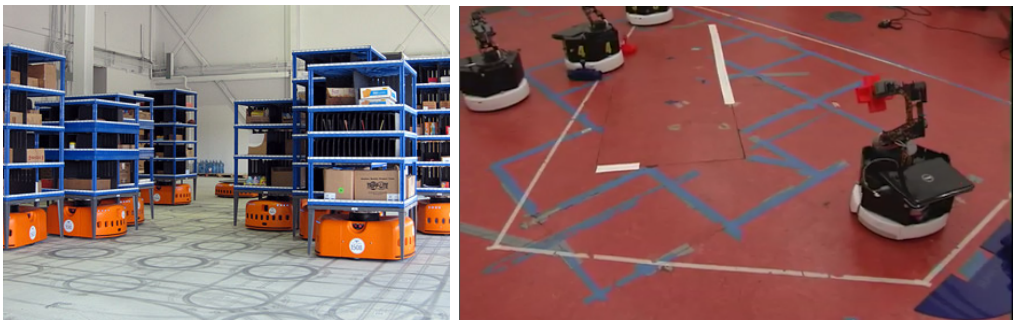
Capítulo 1

Introdução

1.1 Motivação

Transporte de objetos é uma tarefa comum e bem aplicável para o campo da robótica móvel. Os sistemas de transporte podem ser úteis em fábricas, grandes depósitos e até mesmo em casas. O transporte de objetos realizado por robôs móveis autônomos é uma tarefa de suporte para várias tarefas mais complexas, como construção cooperativa [Bolger et al., 2010].

Estima-se que futuramente tarefas de transporte serão cada vez mais executadas por robôs autônomos em ambientes diversos [Wawerla & Vaughan, 2010].



(a) Sistema *Kiva* [Guizzo, 2008]. (b) Construção cooperativa [Bolger et al., 2010].

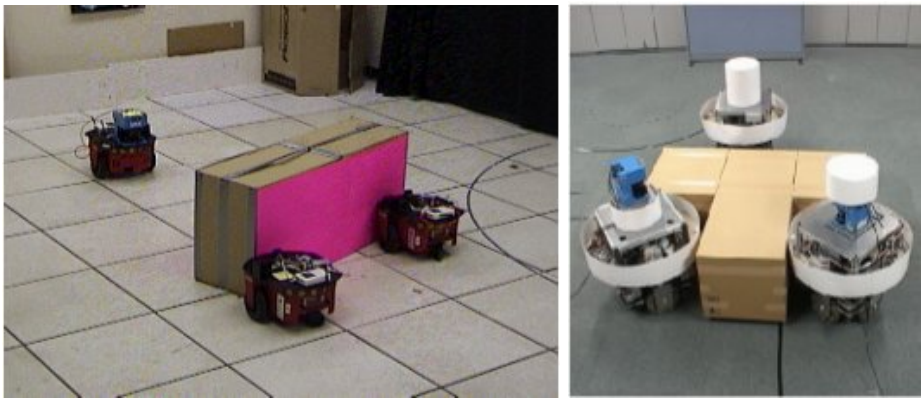
Figura 1.1: Transporte realizado em cenários diferentes. Em 1.1a os robôs fazem o transporte de produtos em um armazém e em 1.1b os robôs fazem o transporte de peças usadas para construção cooperativa.

Em um ambiente com objetos distintos, pode-se ter situações em que o objeto será transportado por apenas um robô. Em outros casos um robô sozinho não será capaz de manipular objetos pesados e/ou grandes. Nesse caso será necessário o

transporte cooperativo, feito por um grupo de robôs (Sistema Multi-Robô (*Multi-Robot System*, ou MRS)).

MRS têm sido foco de muitas pesquisas em robótica móvel por apresentarem várias vantagens em diversas aplicações. Apresentam também grandes desafios, tais como coordenação entre os robôs, falha de comunicação, incertezas de leitura de sensores, entre outros [Gerkey & Mataric, 2004; Verret, 2005; Parker, 2008]. Em alguns cenários, as tarefas só podem ser realizadas por meio de múltiplos robôs. Além disso MRS pode garantir, entre outros, maior eficiência na execução das tarefas e robustez em relação a falhas.

Em um MRS para transporte cooperativo de objetos (Figura 1.2), vários robôs autônomos navegam de forma coordenada em um determinado ambiente que pode ser interno/externo, estático/dinâmico, com obstáculos ou não, etc. de forma a transportar um objeto de um local de origem para um local de destino [Siriwardana, 2009].



(a) Sistema MURDOCH [Gerkey & Mataric, 2002a] (b) [Wang et al., 2005].

Figura 1.2: Transporte cooperativo.

Os objetos são transportados por um ou mais robôs sendo empurrados (Figura 1.2) ou levantados por algum dispositivo de auxílio (Figura 1.1). De acordo com a capacidade de atuação do time de robôs e as características do objeto, um modo de transporte é mais adequado que outro.

Quando os robôs não têm conhecimento das características do objeto, não há como determinar o melhor modo de transporte a ser realizado. O robô precisa explorar o objeto para estimar as informações necessárias para o transporte verificando, desta forma, se precisa de ajuda para transportar o objeto.

Semelhante processo acontece nos insetos sociais quando estão procurando

alimento [Beekman & Dussutour, 2009]. Dentre as diferenças das muitas espécies de insetos, o transporte de alimentos é realizado por meio do processo de recrutamento. Por exemplo, em algumas espécies de formigas, quando uma formiga descobre um alimento, inicialmente ela tenta fazer o transporte sozinha, ao verificar que não consegue chama ajuda para o transporte.

O processo de transporte de objetos por múltiplos robôs apresenta um cenário desafiador. Nessa dissertação propõe-se uma abordagem para o transporte cooperativo de objetos em um ambiente desconhecido, inspirado no processo de recrutamento de insetos sociais. O foco principal consiste na exploração do objeto para estimar informações importantes para o processo de decisão do tipo de transporte e quais robôs serão envolvidos no transporte.

1.2 Definição do Problema

Dado um ambiente E, um conjunto de objetos desconhecidos $O = \{o_1, ..., o_i\}$ com requisitos mínimos $X = \{x_1, ..x_p\}$ para serem transportados, um conjunto de robôs heterogêneos $R = \{r_1, ..., r_k\}$ com capacidades $C = \{c_1, ..., c_n\}$ e dado um conjunto de tipos de transporte $T = \{t_1, ..., t_m\}$ que podem ser executados, descobrir e transportar os elementos de O para um determinado destino em E utilizando um subconjunto de R por meio de um tipo de transporte t_m, de forma que as capacidades c_n dos robôs satisfaçam os requisitos mínimos x_p dos objetos.

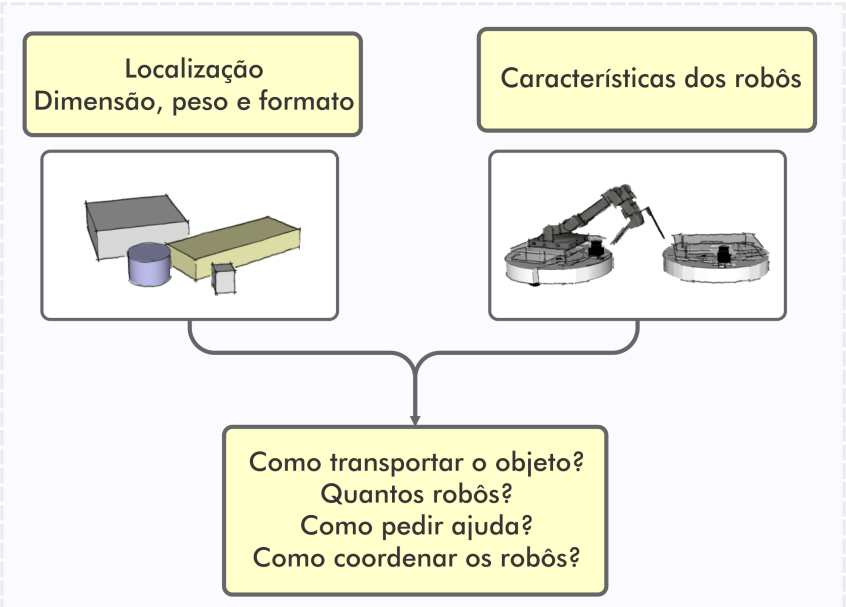


Figura 1.3: Esboço do problema a ser resolvido.

A partir da descoberta do objeto no ambiente, conforme representado na Figura 1.3, a pergunta que se pode fazer é: Qual o tipo de transporte escolher e quais robôs devem executar as tarefas?

O domínio do problema abordado é o transporte de objetos em um ambiente interno. O cenário é constituído de objetos de tamanho, forma e peso diferentes (Figura 1.4). Os robôs devem encontrar esses objetos e transportá-los para uma determinada área. Alguns objetos requerem mais de um robô para o transporte. As formas de transporte consideradas são empurrando (`PUSH`) ou levantando os objetos por meio de um braço robótico (`GRASP`).

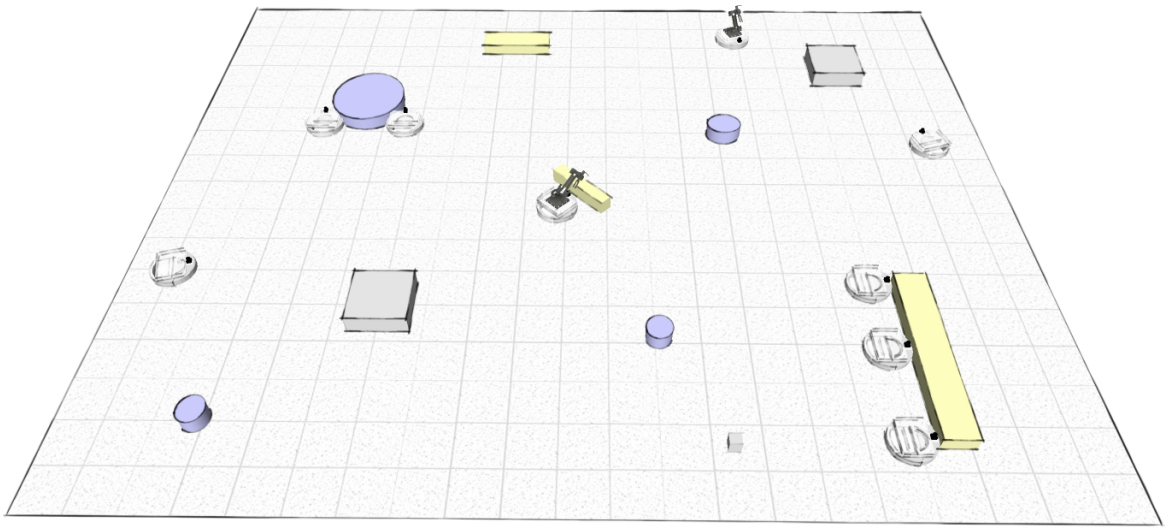


Figura 1.4: Esboço do cenário do problema. Os robôs devem encontrar e transportar os objetos presentes no ambiente, que podem ter tamanho, forma e peso distintos.

Dada a descrição do cenário, pode-se dividir o problema em três subproblemas:

- Exploração do ambiente para encontrar os objetos;
- Exploração do objeto para estimar as características necessárias para a escolha do modo de transporte;
- Alocação de Tarefas/Recrutamento para executar o transporte cooperativamente.

1.3 Contribuições

As principais contribuições obtidas nessa dissertação são:

- Um método para estimar as características de forma e dimensão dos objetos por meio de sensoreamento $2D$ de um *laser*.
- Um método para estimar o esforço necessário para transportar o objeto, seja individual ou cooperativo.
- Um arcabouço para o processo de transporte de objetos desde a exploração do ambiente até o transporte efetivo, seja cooperativo ou não.

1.4 Organização do Trabalho

Neste capítulo foi apresentada uma introdução sobre o problema de transporte de objetos. No próximo capítulo serão apresentados alguns trabalhos encontrados na literatura sobre o tema tratado. Serão abordados trabalhos sobre sistemas que permitem a cooperação e coordenação entre times de robôs e métodos de transporte de objetos. O Capítulo 3 apresenta os detalhes da metodologia desenvolvida como solução para o problema proposto.

No Capítulo 4 são discutidos os resultados alcançados e cada parte da metodologia é validada. Os resultados em ambiente real e simulado mostram a eficácia da metodologia em resolver o problema proposto. Por fim, o Capítulo 5 apresenta as conclusões dos resultados obtidos e as possíveis extensões do trabalho.

Capítulo 2

Trabalhos Relacionados

Neste capítulo, serão discutidos alguns dos principais trabalhos sobre o transporte cooperativo e a coordenação de múltiplos robôs. O transporte é realizado utilizando algoritmos de manipulação de objetos. A coordenação do time pode-se dar através de algoritmos de coordenação e também pode ser tratado como um problema de alocação de tarefas que segundo Gerkey & Mataric [2004], consiste basicamente em definir quais tarefas cada robô deve executar para atingir um objetivo global.

2.1 Transporte Cooperativo de Objetos

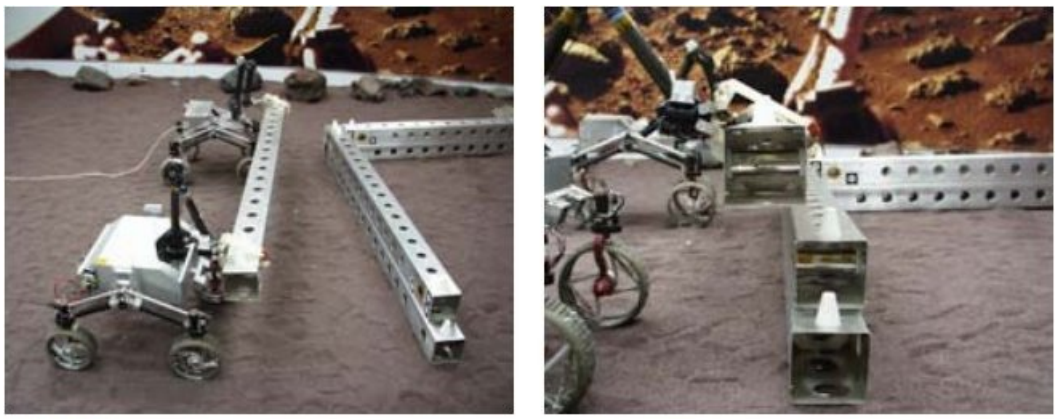


Figura 2.1: Transporte cooperativo de objetos aplicado em um cenário de construção [Thomas et al., 2005].

Em um processo de transporte cooperativo de objetos (Figura 2.1), vários robôs autônomos navegam de forma coordenada em um determinado ambiente

(*indoor/outdoor*, estático/dinâmico, com obstáculos ou não, etc.) de forma a transportar um objeto de um local de origem para um local de destino [Siriwardana, 2009].

2.1.1 Manipulação de Objetos

Para que o objeto seja transportado, pode-se citar três maneiras mais utilizadas para se manipular o objeto, sendo elas: *grasping*, *caging* e *pushing* [Wang & Kumar, 2002; Fink et al., 2007; Pereira et al., 2003] representados na Figura 2.2.

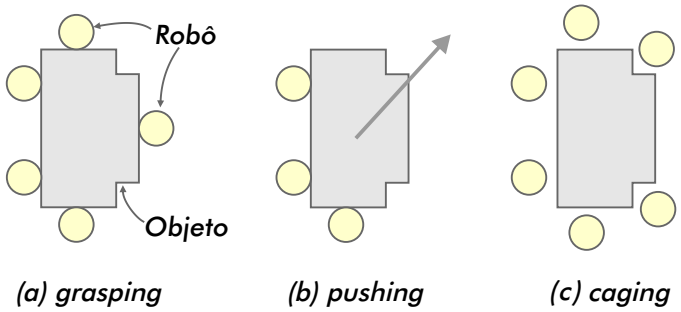


Figura 2.2: Três tipos de manipulação de objetos por múltiplos robôs [Wang & Kumar, 2002].

Na manipulação por *grasping* (Figura 2.2-(a) e Figura 2.3), todos os robôs aplicam uma força ao objeto com o objetivo de agarrá-lo. As condições *Form Closure* ou *Force Closure* devem ser estritamente respeitadas como pode ser visto em [Zhang & Gruver, 1996; Vahrenkamp et al., 2010]. *Force Closure* é uma condição que implica que as forças aplicadas para agarrar/segurar o objeto resistem à outras forças externas. *Form Closure* é uma condição que garante *Force Closure* sem a necessidade de contato atritivo. Tarefas de manipulação de objetos via *pushing* (Figura 2.2-(b)) requerem a condição de *Conditional Closure*. Normalmente os robôs são a única fonte de força aplicada ao objeto, porém, quando forças externas tal como gravidade e atrito interferem no sistema tem-se a condição de *Conditional Closure*. A tarefa de *box-pushing* vista nos trabalhos de Mataric et al. [1995] e Rus et al. [1995] é um exemplo de *Conditional Closure*. Na manipulação por *caging* (Figura 2.2-(c)), a ideia básica é introduzir uma região limitada para o posicionamento do objeto. Por meio da condição de *Object Closure*, o objeto é preso entre os robôs e é transportado conforme o deslocamento do grupo de robôs. Essa condição é menos restrita em relação a força exercida sobre o objeto pois não é necessário que todos os robôs

estejam a todo instante em contato com o objeto [Wang & Kumar, 2002; Pereira et al., 2003; Fink et al., 2007, 2008].

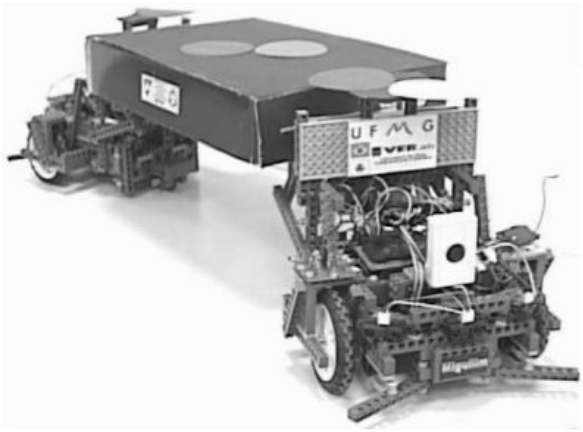


Figura 2.3: Robôs transportando cooperativamente um objeto por meio de *grasping* [Pereira et al., 2002]

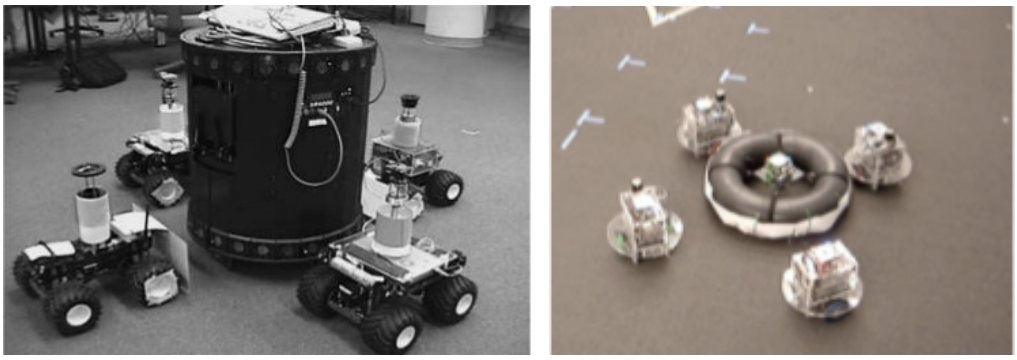
Em um cenário de organização de móveis de um quarto, Rus et al. [1995] apresentam quatro protocolos de coordenação para manipular o objeto cooperativamente. Além da posição (x, y) de destino, o objeto deve ficar em determinada orientação (θ) desejada. São explorados nos protocolos se o planejamento para a execução da tarefa é de fato necessário, controle local e global e comunicação explícita.

Mataric et al. [1995] apresentam uma solução para o problema de *box-pushing* com dois robôs. A estratégia utiliza cooperação em três níveis: sensorial, de atuação e de controle. Cada robô toma uma decisão por vez. Os robôs revezam para executar uma ação levando a uma coordenação implícita. Os robôs trocam informações dos sensores para reduzir a incerteza e ruídos das medições.

Emery & Balch [2001] apresentam um controlador para robôs não holonômicos que permite aos mesmos transportar objetos. É também incorporado um módulo de desvio de obstáculos utilizando campos vetoriais. O controle é baseado nos três comportamentos que o robô assume: *Search*, quando o robô está procurando pelo objeto para ser transportado, *Aquire*, em que o robô deve navegar para uma posição específica de acordo com a posição do objeto e o destino do mesmo e *Deliver*, quando o robô propriamente empurra o objeto levando-o a mover-se para o destino. As tarefas são de cooperação fraca, ou seja, cada robô executa uma tarefa sozinho com o fim de contribuir para o objetivo final. Não há o compartilhamento de uma mesma tarefas.

Wang & de Silva [2006] propuseram uma estratégia para o transporte de objetos via *pushing* com múltiplos robôs utilizando aprendizado de máquina. O algoritmo de aprendizado por reforço *Q-learning* foi estendido para o domínio de multi-robôs e cada robô aprende individualmente em qual ponto de contato do objeto que deve empurrar. A ação de cada robô produz uma força resultante, que empurra o objeto ao destino.

Na abordagem de transporte por *caging* apresentada em [Fink et al., 2007] cada robô segue um conjunto de regras que compõem os comportamentos *Approach*, *Surround*, e *Transport* a fim de enclausurar geometricamente o objeto e movê-lo de uma maneira previsível. Utilizando a composição de campos potenciais artificiais, o time de robôs forma um padrão de movimento que resulta na manipulação do objeto. Posteriormente Fink et al. [2008] estenderam o trabalho para ambientes com obstáculos. As Figuras 2.4a e **??** mostram um grupo de robôs manipulando o objeto por meio de *caging*.



(a) [Pereira, 2003]. (b) [Fink et al., 2007].

Figura 2.4: Manipulação de objetos via caging.

Algumas abordagens se inspiram nas sociedades de insetos sociais, que são sistemas descentralizados escaláveis e adaptáveis por natureza. Kube & Bonabeau [2000] apresentam uma abordagem descentralizada para *box pushing* por múltiplos robôs inspirada no comportamento de algumas espécies de formigas. A cooperação é obtida baseada no mecanismo de recrutamento que as formigas usam quando precisam de ajuda para transportar algum alimento. A principal contribuição do trabalho foi o modelo formalizado descrevendo o transporte cooperativo das formigas. Aplicado ao grupo de robôs, mostrou-se que os robôs podem se auto-organizar de forma descentralizada para executar tarefas de transporte. Porém, a abordagem puramente baseada em formiga não se mostrou eficiente no que diz respeito ao tempo de execução.

Berman et al. [2011] formularam um modelo com controle descentralizado para manipulação cooperativa de objetos baseado no fenômeno de captura de presa das formigas. Através de experimentos com a espécie de formigas *A. cockerelli's*, extraiu-se regras que governam suas ações e forças individuais aplicadas para transportar uma presa para o ninho. O modelo apresentado é escalável e não requer conhecimento prévio sobre o peso do objeto.

2.2 Coordenação

Botelho & Alami [1999] combinam planejamento local e negociação para a alocação das tarefas. O trabalho proposto é um protocolo descentralizado dividido em três camadas: um módulo de alocação de tarefas baseada em Contract Net Protocol, módulo de tolerância à falhas e um módulo de execução, responsável por coordenar os robôs. A descrição de uma tarefa é compartilhada entre todos os robôs e é necessário conhecer o ambiente previamente.

A abordagem mais utilizada para controlar e coordenar um time de robôs na execução de tarefas é a baseada em comportamentos. [Brooks, 1986]. Cada robô do time segue uma lei de comportamento que contribui para o comportamento global esperado. Através da percepção (sensoreamento) do ambiente, cada robô toma a sua decisão e o comportamento final é dado pela contribuição de cada robô.

ALLIANCE [Parker, 1998] é um sistema distribuído e modela times de robôs heterogêneos. Utiliza o paradigma baseado em comportamento. Possui um mecanismo de tolerância a falhas que permite aos robôs envolvidos em determinada tarefa detectar falhas em outros robôs do time e adaptar o seu comportamento para finalizar a execução da tarefa. Cada robô explicitamente estima seu desempenho e o desempenho dos outros robôs e seleciona sua tarefa através dos atributos de impaciência e aquiescência.

De acordo com a tarefa, o time pode se coordenar de maneira que cada robô assuma um papel diferente. Cada robô do time segue uma lei de comportamento referente ao seu papel. Chaimowicz et al. [2002] utiliza a alocação dinâmica de papéis na coordenação do time de robôs. As tarefas cooperativas e a alocação dinâmica são modelados como um autômato híbrido, que pode ser visto como uma composição de estados discretos e equações contínuas. O comportamento de cada robô é representado por um autômato híbrido e a composição de vários autômatos leva à modelagem e execução da tarefa cooperativamente.

Em [Pereira, 2003], a coordenação pode ser obtida utilizando-se dos concei-

tos da teoria de grafos. A metodologia é baseada na conectividade de grafos que fornece uma única solução para várias tarefas cooperativas. Os robôs são representados pelos vértices e as arestas representam as restrições de movimento em relação aos outros robôs do grupo. Essas restrições variam de acordo com a tarefa. Para que o time seja capaz de executar tarefas cooperativas variadas é necessário mapeá-las em múltiplos problemas individuais de planejamento de movimento restrito.

Martinson & Arkin [2003] utiliza aprendizado de máquina para coordenar um grupo de robôs em uma tarefa de *foraging*. Os robôs aprendem individualmente qual comportamento devem adotar e quando devem trocar de papel para completar a tarefa em execução. Embora o aprendizado seja individual, o comportamento global do time converge para a execução cooperativa das tarefas.

Outra abordagem utilizada para promover a cooperação entre o time de robôs é a abordagem que utiliza o conceito de coalizão *(coalition)* tratado em Sistema Multi-Agente (*Multi-Agent System*, ou MAS). A coalizão consiste em uma organização temporária de agentes com o objetivo de executar uma tarefa conjuntamente. Exemplos de trabalhos que usaram coalizão são: ASyMTRe [Parker & Tang, 2006] e CoMutaR [Shiroma & Campos, 2009].

CoMutaR lida ao mesmo tempo com problema de coordenação e alocação de tarefas como uma única camada. Um algoritmo de leilão proporciona a formação das coalizões e a coordenação é obtida por meio de restrições compartilhadas.

2.2.1 Alocação de Tarefas

A coordenação do time de robôs também pode ser obtida por meio de Alocação de Tarefas para Multiplos Robôs (*Multi-Robot Task Allocation*, ou MRTA). O problema consiste em achar uma distribuição ótima das tarefas para um grupo de agentes/robôs que estão trabalhando para o mesmo objetivo global. Dado um conjunto de tarefas definidas, determinar quais são as atribuições de tarefas que otimizam uma função objetivo do sistema [Gerkey & Mataric, 2004; Ferreira Jr et al., 2010]. Esse é um problema $\mathcal{NP}$-difícil e as soluções encontradas são algoritmos aproximados ou heurísticas.

Além da completude da tarefa, deseja-se que esta seja executada de maneira ótima considerando algumas restrições, tais como: energia do robô consumida, capacidade individual dos robôs, tempo para execução da tarefa, entre outras. A dificuldade desse problema será conforme o tipo de tarefa. Algumas tarefas podem ser executadas de forma independente, outras requerem ajuda ou tem alguma res-

trição de precedência [Gerkey & Mataric, 2004]. Essas particularidades aumentam o nível e dificuldade do problema de alocação.

As abordagens de alocação podem ser centralizadas, distribuídas ou híbridas. No contexto de MRS, deseja-se que a alocação seja distribuída. Essa abordagem possibilita a escalabilidade do sistema. Quando se tem uma entidade centralizada para resolver o problema a solução pode ser computacionalmente inviável. Outra característica importante para o sistema é a possibilidade de resolver as tarefas em paralelo, diminuindo a complexidade do problema em relação ao tempo [Parker, 2008].

Algoritmos bio-inspirados, algoritmos de leilão que são baseados no mercado econômico *(market-based)* e aprendizado de máquina são algumas das abordagens utilizadas para solucionar o problema de MRTA [Farinelli et al., 2004; Parker, 2008]. A escolha por um ou outro método depende da aplicação em si e da maneira que o sistema pode ser modelado.

De acordo com as restrições do problema de alocação, pode-se reduzi-lo à algum problema conhecido de otimização. Gerkey & Mataric [2004] reduziram o problema de alocação de tarefas para uma instância de um Problema de Atribuição Ótima (*Optimal Assignment Problem*, ou OAP). Já no trabalho de Ferreira Jr et al. [2010], a abordagem para resolver o problema de alocação foi baseada em sistemas multi-agente e os formalismos utilizados foram: Problema de Atribuição Geral (*General Assignment Problem*, ou GAP) e Problema Otimização de Restrição Distribuída (*Distributed Constraint Optimization Problem*, ou DCOP).

Também no contexto de multi-agente, dos Santos & Bazzan [2009] apresentam o algoritmo aproximado *eXtreme-Ants*. Esse resolve o problema de alocação como um Problema de Atribuição Geral Extendida (*Extended Generalized Assignment Problem*, ou E-GAP) que incorpora ambiente dinâmico e inter-relação de tarefa. O algoritmo proposto é inspirado na divisão de trabalho dos insetos sociais e no processo de recrutamento. Na divisão do trabalho cada agente tem uma capacidade específica para executar determinada tarefa e o recrutamento envolve pedir ajuda aos outros agentes do time quando for necessário.

Embora apresentem bons resultados, as soluções obtidas para os MAS não podem ser aplicadas diretamente para os MRSs pois não consideram as restrições do mundo real que o time de robôs sofre [Vig & Adams, 2006].

Uma abordagem de grande importância na alocação de tarefas para múltiplos robôs são os algoritmos de leilão. Nessa abordagem, uma tarefa é lançada por um leiloeiro *(auctioner)* e os robôs explicitamente comunicam-se para se oferecerem *(to bid)* à uma tarefa de acordo com sua capacidade. Normalmente a tarefa é associada

ao robô com a maior oferta. Viguria et al. [2008] apresenta o algoritmo S+T que utiliza uma abordagem *market-based* distribuída para realizar a alocação de tarefas. Para propiciar a cooperação entre os agentes, foi introduzido o conceito de serviço. A ideia básica é que o robô pode solicitar por serviços quando o mesmo não puder executar a tarefa sozinho. O custo da tarefa será a soma da tarefa e de todos os serviços requisitados.

Em Dias et al. [2006] é apresentado um estudo abrangente e os vários desafios da abordagem *market-based*.

Muitos trabalhos têm buscado a adaptação e flexibilidade do sistema através de métodos de aprendizado [Dahl et al., 2002; Martinson & Arkin, 2003; Dahl et al., 2009; Sun et al., 2009]. O time de robôs aprende, a partir das interações com o ambiente, qual a melhor distribuição de tarefas no instante atual do sistema e como reagir com novos estados do ambiente [Dahl et al., 2002]. O grande problema do aprendizado é o tempo de aprendizagem e o espaço de representação dos estados do ambiente. A maioria dos trabalhos citados utiliza o aprendizado juntamente com outros métodos ou heurísticas.

Dahl et al. [2009] usa o aprendizado por reforço para estimar a utilidade das tarefas. Utiliza um modelo formal chamando Vacancy Chain Scheduling (VCS). Nesse formalismo o desempenho global do sistema é divido entre contribuições individuais dos robôs, diminuindo a comunicação explicita para atingir o objetivo geral. Com a atualização contínua da tabela de utilidade de cada indivíduo a solução ótima emerge para uma solução global.

Uma visão mais abrangente para os vários sistemas de MRTA pode ser encontrada nos trabalhos de Gerkey & Mataric [2004], Verret [2005] e Parker [2008].

2.3 Resumo

Nos trabalhos apresentados nesta seção observa-se três grupos classificados pelo foco do trabalho: transporte do objeto para um destino, manipulação do objeto e coordenação do time de robôs.

Em todos os trabalhos observa-se que os objetos são escolhidos de tal forma que os robôs são capazes de manipulá-los. O peso e o tamanho do objeto não influenciam nas tarefas de transporte.

Os trabalhos que tem o foco na coordenação, seja por meio da alocação de tarefas ou não, utilizam-se do processo de transporte apenas como uma aplicação. O modo que o transporte é realizado e as características dos objetos não são

previamente conhecidas ou não interferem na realização da tarefa.

Capítulo 3

Metodologia

Este capítulo descreve a metodologia proposta para o transporte de objetos por múltiplos robôs. Inicialmente, apresenta-se uma visão geral da solução proposta. Posteriormente, é feita uma descrição detalhada de cada módulo.

3.1 Visão Geral

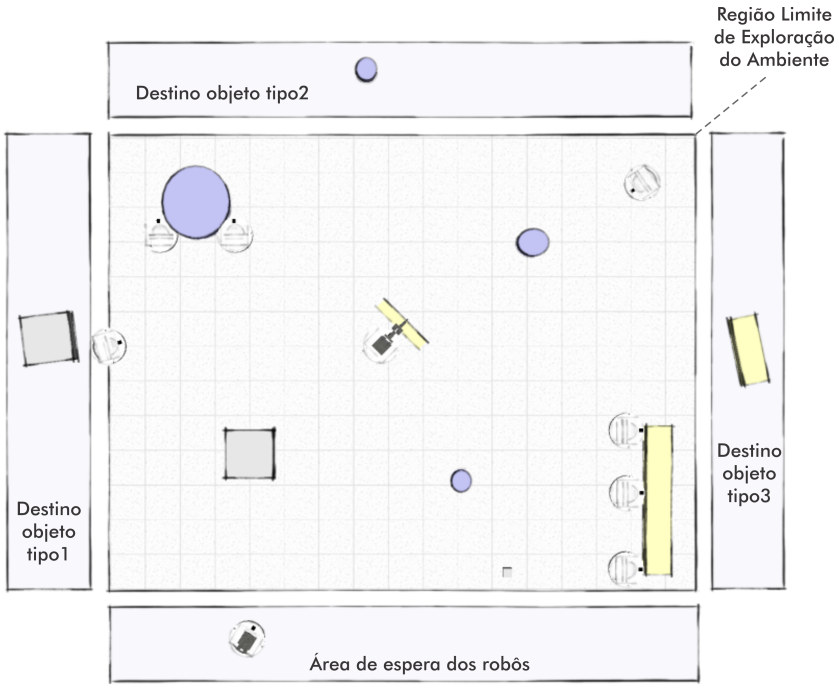


Figura 3.1: Os robôs exploram uma determinada região em busca de objetos para serem transportados. Tanto os robôs quantos os objetos apresentam características heterogêneas. O destino dos objetos é escolhido de acordo com suas características. O transporte pode ser realizado cooperativamente ou não.

Neste trabalho, adota-se as seguintes suposições:

Suposição 1. *Os robôs têm conhecimento da localização global de todo o time.*

Suposição 2. *A rede é totalmente conectada e não ocorrem falhas de comunicação e nem atrasos.*

Suposição 3. *Tamanho máximo* $(l \times w \times h)$ *dos objetos é conhecido.*

Suposição 4. *Altura h dos objetos é previamente conhecida.*

Na abordagem proposta, o robô não tem conhecimento do ambiente e nem dos objetos presentes; tanto os robôs quanto os objetos têm características distintas e o destino dos objetos é decidido conforme suas características (Figura 3.1).

O time de robôs é composto por dois tipos: o `PUSHER` que transporta um objeto empurrando-o e o `GRASPER` que pode transportar um objeto utilizando um manipulador robótico (Figura 3.2).

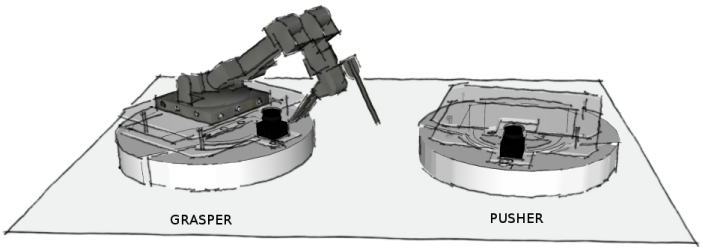


Figura 3.2: Tipos de robô.

Para que o transporte seja realizado, a abordagem proposta divide o problema em: (1) explorar o ambiente a fim de descobrir os objetos, (2) explorar os objetos para extrair informações necessárias ao transporte, (3) recrutar cooperadores caso seja necessário e por fim, (4) transportar o objeto. Portanto, pode-se definir inicialmente um conjunto de ações para a realização do transporte: *explorar o ambiente, explorar o objeto, recrutar cooperadores e transportar o objeto*.

Tendo em vista as necessidades discutidas, o problema tratado pode ser dividido em quatro módulos: `ENVIRONMENT_EXPLORING`, `OBJECT_EXPLORING`, `TRANSPORTING` e `RECRUITING`. A Figura 3.3 apresenta as transições entre os estados do robô correspondentes aos quatro módulos citados, além do estado de espera `WAITING`.

Os robôs podem assumir dois papéis: robô explorador ou cooperador. O robô explorador inicia no estado `ENVIRONMENT_EXPLORING`. Os outros robôs inicialmente,

encontram-se no estado `WAITING` (ver Figura 3.3). Nesse estado, o robô permanece na área de espera aguardando pedidos de ajuda para participar de uma tarefa de transporte. Se o robô receber um pedido de ajuda e for alocado para a tarefa, é feita uma transição para o estado `TRANSPORTING`.

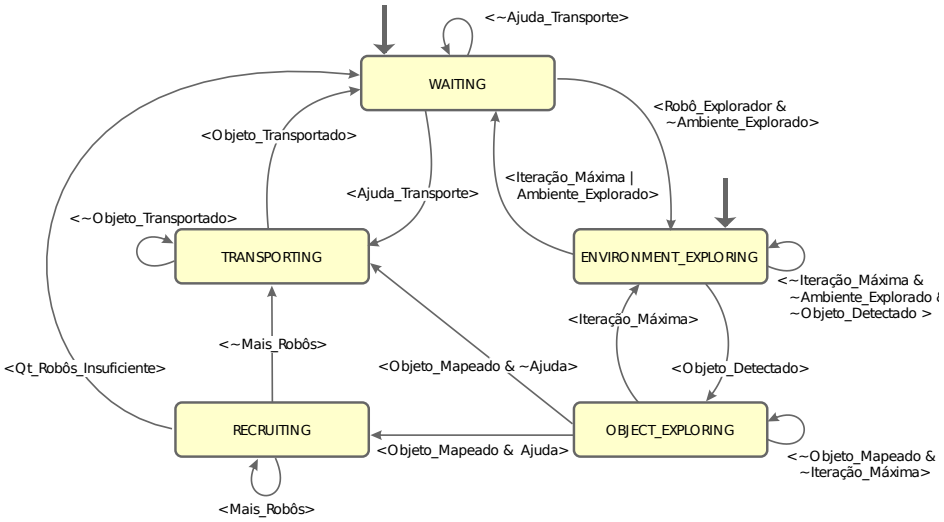


Figura 3.3: Máquina de estados representando o comportamento dos robôs.

No estado `ENVIRONMENT_EXPLORING`, o robô explorador navega pelo ambiente em que está inserido em busca de novos objetos para serem transportados. Enquanto navega, o robô cria um mapa do ambiente explorado utilizando o método Grade de Ocupação *(Occupancy Grid)* [Elfes, 1989]. Com o mapa do ambiente, o robô tem condições de saber quais lugares ele já visitou. O robô tende a evitar a navegação em locais já explorados anteriormente, maximizando assim a chance de encontrar novos objetos.

Quando o robô explorador encontra um novo objeto, inicia-se o processo de exploração do objeto no estado `OBJECT_EXPLORING`. O objetivo dessa exploração é inferir as características do objeto. O robô tenta estimar informações que sejam úteis à tarefa de transporte, tais como pose (x, y, θ), forma geométrica do objeto e dimensões da largura (w), profundidade (l) e altura (h). Para estimar essas informações, o robô cria um mapa local do objeto utilizando o método de Grade de Ocupação. Uma vez que o robô tem a imagem do mapa local do objeto, ele tenta extrair as informações necessárias utilizando operações de processamento de imagem (Seção 3.4.1). Além disso, o robô estima o esforço necessário para o transporte do objeto com o fim de avaliar se consegue transportá-lo ou se é necessária a ajuda de outros robôs para a execução da tarefa.

Há um tempo máximo para permanecer nesse estado. Se o tempo exceder o limite, o robô volta para o estado `ENVIRONMENT_EXPLORING` e o objeto não será transportado. Quando a exploração do objeto é bem sucedida, o robô tem condições de saber se transporta o objeto sozinho ou se precisa de ajuda.

Caso seja necessária ajuda de outros robôs, inicia-se ao processo de recrutamento no estado `RECRUITING`. O robô explorador torna-se o recrutador. Por um determinado período, ele solicita ajuda por meio de envio de mensagens. O robô pede ajuda até que tenha um número suficiente de ajudantes para realizar o transporte. Se dentro do período determinado não houver robôs suficientes para a execução da tarefa o recrutamento é abortado.

No estado `TRANSPORTING` os robôs interagem com o objeto de acordo com as leis de controle do algoritmo de transporte, até que ele seja transportado para o seu destino. Quando o transporte é bem sucedido, o robô explorador volta ao estado `ENVIRONMENT_EXPLORING` e os outros voltam ao estado `WAITING`.

3.2 Exploração do Ambiente

Uma vez que o ambiente é desconhecido, faz-se necessária sua exploração com o objetivo de encontrar os objetos eventualmente presentes. Para garantir que todos os objetos sejam descobertos, os robôs devem explorar ao máximo a região determinada.

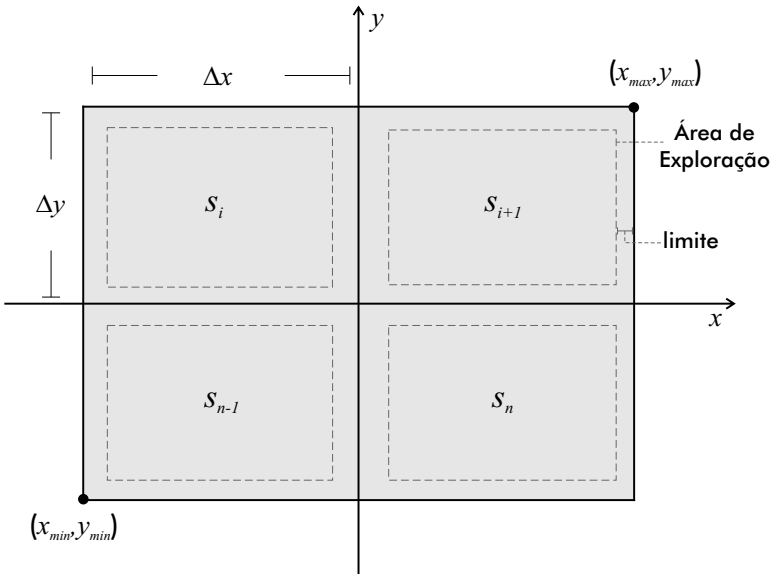


Figura 3.4: Divisão da área de exploração.

No processo de exploração do ambiente pode haver vários robôs explorando ao mesmo tempo. Uma das estratégias propostas para exploração cooperativa é a segmentação do ambiente [Schneider-Fontán & Mataric, 1998; Wurm et al., 2008].

A segmentação do ambiente foi realizada por divisão territorial [Schneider-Fontán & Mataric, 1998]. A região de exploração total $(\mathcal{S})$ é uma área retangular definida pelas coordenadas (x_{min}, y_{min}) e (x_{max}, y_{max}), conforme representação na Figura 3.4.

A estratégia de exploração adotada neste trabalho consiste em dividir a região $\mathcal{S}$ em N_A áreas retangulares. A região será particionada como uma matriz de l linhas e c colunas. Os valores de N_A, c e l são determinados empiricamente.

Cada subárea terá a dimensão $\Delta x \times \Delta y$, conforme a Equação 3.1.

$$\begin{aligned} \Delta x &= dimX/c, \\ \Delta y &= dimY/l. \end{aligned} \tag{3.1}$$

Para cada subárea s_i, $0 \leq i \leq N_A - 1$, é alocado um robô r_i, dado o número total de robôs N_R, $1 \leq N_A \leq N_R$.

A alocação dos robôs às respectivas subáreas é feita de acordo com o identificador ID_r do robô. O índice (i, j) referente à subárea de exploração é definido pela Equação 3.2.

$$\begin{aligned} i = \quad & \text{mod } (ID_r, c), \\ j = & \left\lfloor \frac{ID_r}{c} \right\rfloor. \end{aligned} \tag{3.2}$$

Cada subárea tem um limite b_{lim} para navegação do robô. A região para navegação será definida pelas coordenadas (xp_{min}, yp_{min}) e (xp_{max}, yp_{max}), conforme a Equação 3.3.

$$\begin{aligned} xp_{min} &= x_{min} + i\Delta x + b_{lim}, \\ yp_{max} &= y_{max} - j\Delta y - b_{lim}, \\ xp_{max} &= xp_{min} + \Delta x - b_{lim}, \\ yp_{min} &= yp_{max} - \Delta y + b_{lim}. \end{aligned} \tag{3.3}$$

Definida a área de exploração, é necessário estabelecer uma estratégia de navegação. Neste trabalho, adotou-se a estratégia de geração aleatória de posições que o robô deve explorar. Dentro da respectiva área de exploração, são geradas aleatoriamente $RAND\_POS$ posições para o robô explorar. Para cada posição é calculado o ganho que o robô terá caso escolha esse destino.

Pelo método de mapeamento adotado (Seção 3.3), cada célula do mapa contém a probabilidade de estar ocupada (obstáculo), livre ou desconhecida. Dada a probabilidade de cada célula, utilizou-se a medida de Entropia [Alpaydin, 2010] para calcular o ganho de informação (Equação 3.4). Quanto maior a entropia, menos informação se tem sobre a região.

Algoritmo 1: EXPLORARAMBIENTE (N_A, ID)

```
1  S = s_0, ..., s_{N_A - 1}        //Determinar as subáreas
2  i = ID_r                         //Associar a subárea ao ID do robô
3  it ← 0
4  while (it < MAX_IT_EXP) do
5      max ← 0
6      for i ← 0 to RAND_POS do
7          x = rand()               // Sortear x de forma que x ∈ s_i
8          y = rand()               // Sortear y de forma que y ∈ s_i
9          entropia ← CalcEntropia(x, y)    //Calcular a entropia da região
10                                          //de (x, y)
11         if (entropia ≥ max) then
12             max ← entropia       //Utilizar a região (x, y)
13             xG ← x               //de maior entropia
14             yG ← y
15     while !chegouDestino do
16         areaExplorada ← VaiPara(xG, yG)    //Navegar pelo ambiente
17                                            //e atualizar o mapa
18         if (objetoDetectado) then
19             return 1;
20         if (areaExplorada) then
21             return 0;
22     it ← it + 1
```

$$H = -p\log(p) - (1-p)\log(1-p), \tag{3.4}$$

onde log é o logaritmo na base 2 e p é a probabilidade de uma célula (x, y) estar ocupada.

O cálculo do ganho é feito pela média da informação da região em torno da posição (x, y) inclusive. Essa região é definida por uma janela de tamanho $W \times W$, em que W é ímpar. A região cuja média for a maior, corresponde à região onde se tem menos informação. Portanto, é uma boa região para ser explorada.

O Algoritmo 1 demostra o processo de exploração do ambiente. A cada iteração, o robô escolhe uma posição de destino (xG, yG) dentro das limitações da sua subárea de exploração. Por um período MAX_IT_EXP definido, o robô permanece explorando o ambiente até encontrar um novo objeto ou até que a sua respectiva subárea seja completamente explorada.

3.3 Mapeamento

O mapeamento do ambiente e do objeto são feitos utilizando-se o método de Grade de Ocupação [Elfes, 1989]. Nesta abordagem, o ambiente é discretizado e representado por uma matriz (m). Cada célula ($m_{i,j}$) constitui um local do ambiente que pode estar vazio (região livre), ocupado (obstáculo detectado), ou ainda ser um local desconhecido, ou seja, não há informação sobre ele [Elfes, 1989; Siegwart & Nourbakhsh, 2004].

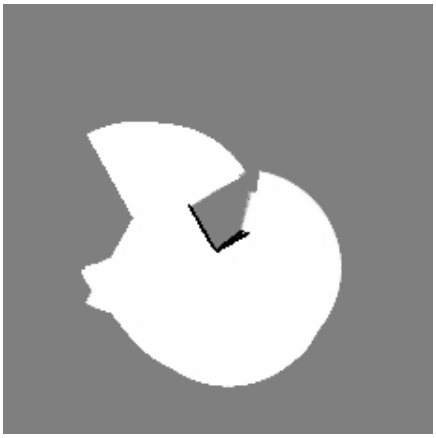


Figura 3.5: Mapa obtido pelo método de grade de ocupação. As cores do mapa estão relacionadas com a probabilidade de ocupação de cada célula. Áreas mais escuras têm maior probabilidade de serem obstáculo. Já as áreas mais claras têm maior probabilidade de serem regiões livres de obstáculos. As áreas de cor cinza são regiões não mapeadas.

O mapa é construído a partir das informações de um sensor de distância. Neste trabalho, um *laser* fixo ao robô realiza leituras 2D do ambiente. Dada a localização do robô e as medições obtidas pelo *laser*, cada célula é atualizada com uma probabilidade $p(m_{i,j})$ de estar ocupada. A Figura 3.5 apresenta um mapa obtido pelo método de Grade de Ocupação.

3.4 Exploração do Objeto

Dado o cenário descrito na Seção 1.1, o robô detecta um obstáculo quando encontra um objeto ou um outro robô. Uma vez que a posição de todos os robôs é conhecida (Suposição 1), pode-se verificar se o que foi encontrado é um robô ou um objeto. Se for um objeto, o robô começa o processo de exploração. Esse processo é compreendido por duas fases, sendo a primeira a estimação da pose, tipo geométrico e dimensões (Figura 3.6) do objeto e a segunda envolve a avaliação do esforço necessário para transportá-lo (Figura 3.13).

Na **Fase 1**, o robô fica circundando o objeto até que o mesmo seja completamente mapeado ou até que se atinja o tempo máximo de iterações MAX_IT_MAP definido para o processo de mapeamento. O robô termina esse mapeamento quando identifica que o contorno do objeto foi fechado. A detecção do fechamento do contorno é feita por meio de algoritmos de processamento de imagem aplicados ao mapa local do objeto, que serão descritos na Subseção 3.4.1.

As dimensões e pose do objeto são estimadas a cada iteração do mapeamento (Seção 3.4.1). Ao término, o tipo do objeto é definido por meio de correspondência de imagens (Seção 3.4.3) e são estimados o número mínimo N_{R_MIN} e máximo N_{R_MAX} de robôs necessários para executar o transporte do objeto (Seção 3.4.4), finalizando a **Fase1**.

O próximo passo é definir se o objeto pode ser transportado por um ou mais robôs. Portanto, na **Fase2**, o robô verifica o esforço necessário para transportar o objeto (Seção 3.4.5), a fim de saber se precisará ou não de ajuda para executar a tarefa.

3.4.1 Mapeamento do Objeto

O robô verifica se o objeto foi totalmente mapeado analisando o contorno da imagem encontrada no mapa local do objeto. A cada intervalo de tempo pré-definido, o robô verifica se há um contorno fechado na imagem por meio do Algoritmo 2.

O algoritmo consiste basicamente em identificar o contorno do objeto na imagem e verificar se esse é um contorno fechado. A decisão pelo término do mapeamento é feita pela área do contorno encontrado. Enquanto o contorno não é fechado, a área retornada é nula.

Entende-se por contorno uma sequência de pontos conectados em torno de uma região contínua da imagem. Dada uma imagem binarizada, a função *AcharContorno* retorna o número de contornos encontrados e os pontos (x, y) per-

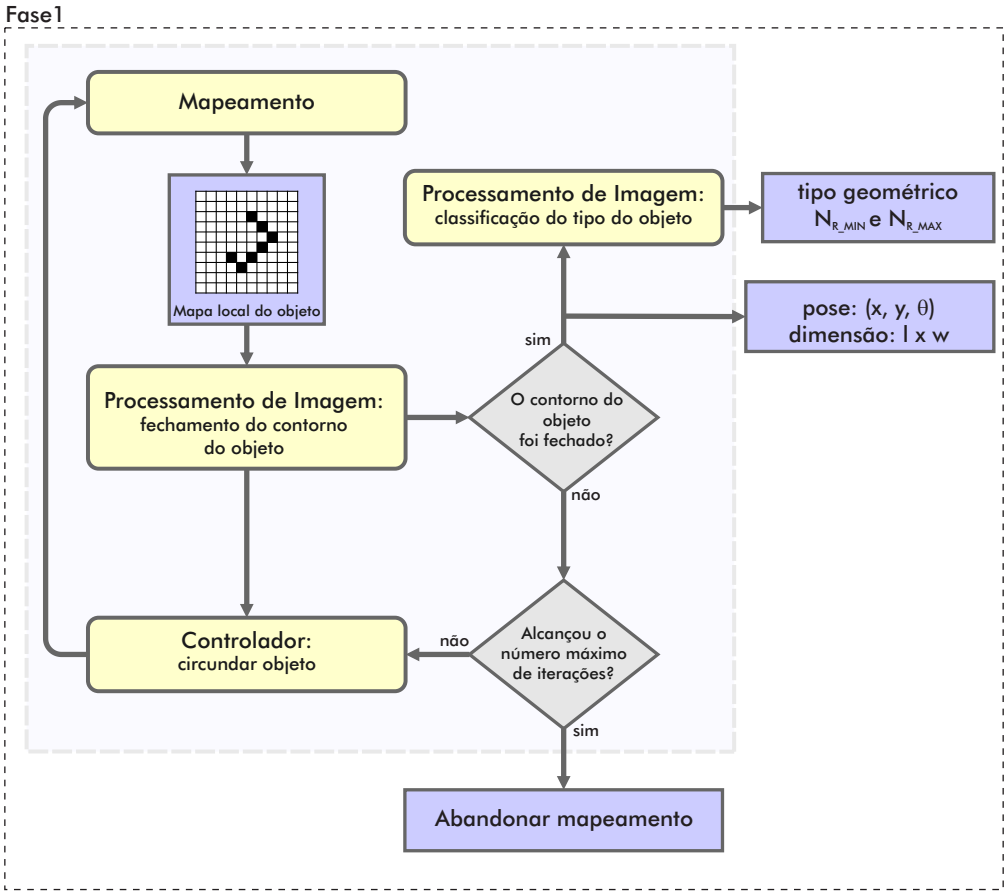


Figura 3.6: Fase 1: Processo de mapeamento do objeto.

tencentes à cada contorno.

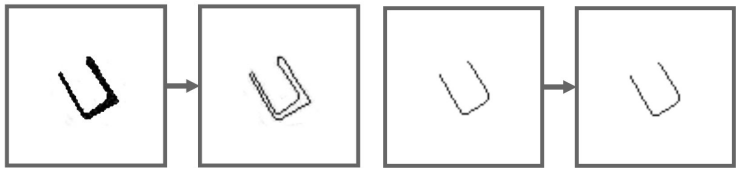
A imagem recebida como entrada para o Algoritmo 2 é uma região do mapa obtido pelo método de Grade de Ocupação. A única informação importante para o processamento é a região que o *laser* detectou como obstáculo (cor preta) (Figura 3.5). Para que o contorno seja melhor identificado, a imagem inicialmente é pré-processada.

O objetivos desse pré-processamento é: reduzir ruídos e possíveis falhas de continuidade do contorno do objeto. Inicialmente, a imagem é binarizada de forma a classificar os *pixels* em pertencentes ao contorno ou não. Os filtros gaussiano e da média são utilizado na imagem com o objetivo de reduzir ruídos. Para reduzir possíveis falhas de continuidade no contorno do objeto, realizam-se as operações morfológicas de abertura e fechamento [Gonzalez & Woods, 2006].

Após esse pré-processamento, a imagem tem um contorno bem definido (Figura 3.7a). Entretanto, devido à ruídos nos sensores e atuadores, o contorno obtido pode possuir uma espessura considerável. Como a decisão é feita pela área

Algoritmo 2: FECHOUCONTORNO (*img*)

```
1  img ← Binarizar(img)
2  img ← Filtros(img)                    //filtros gaussiano e da média
3  img ← AberturaFechamento(img)         //operações morfológicas
4  img ← Esqueletonizacao(img)           //afinamento do contorno
5  N_c ← AcharContorno(img)              //quantidade de contornos encontrados em img
6  if (N_c = 1) then
7  |  pts ← pontos (x, y) pertencentes ao contorno encontrado
8  |  area ← área de pts                 //calcular a área do contorno encontrado
9  |  CalcularPose(pts)
10 |  CalcularDimensao(pts)
11 |  if (area > 0) then
12 |  |  return verdadeiro               //contorno encontrado está fechado
13 |  else
14 |  |  return falso
15 else
16 |  return falso                       //mais de um contorno encontrado
```



(a) Contorno sem Afinamento (b) Contorno com Afinamento

Figura 3.7: Processamento do contorno do objeto.

determinada pelo contorno, a área dessa região seria maior que zero e o algoritmo erroneamente retornaria que o mapeamento foi concluído.

Portanto, faz-se necessário que o contorno seja definido por apenas um *pixel* de largura. Para isso, a imagem passa por um processo de afinamento do contorno (esqueletonização - Algoritmo de Zhang-Suen descrito em Parker [1996]). Como pode ser visto na Figura 3.7b, após esse processamento, os pontos retornados como contorno serão os próprios pontos da imagem e a área do contorno será zero.

Dado o número de contornos encontrado N_c e os pontos (x, y) pertencentes ao mesmo, se o N_c for maior que um, significa que há falhas no contorno do objeto e não se pode inferir se o objeto foi totalmente mapeado. Se o número de contornos for apenas um, calcula-se a área atual do contorno. Como é garantido que o contorno na imagem é de apenas um pixel, se a área dos pontos encontrados for

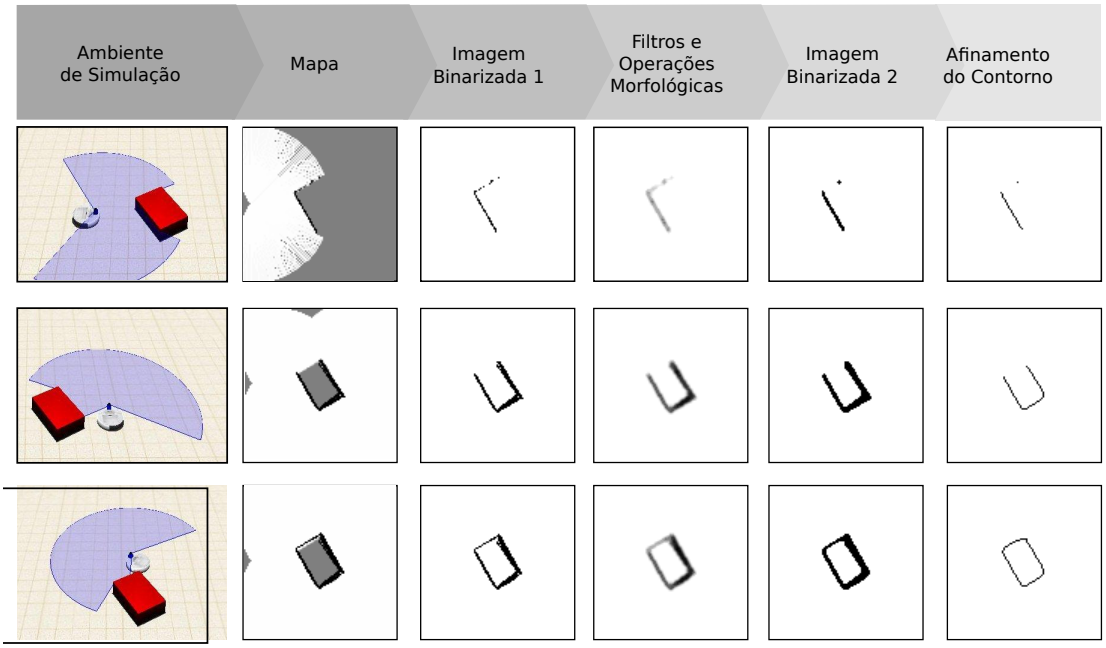


Figura 3.8: Processo de mapeamento e identificação do contorno do objeto.

maior que zero, o contorno foi fechado, caso contrário o mapeamento continua. A Figura 3.8 apresenta três momentos diferentes, durante o processo de mapeamento até que o contorno seja fechado.

A cada iteração, os pontos do contorno encontrado são enquadrados em uma estrutura retangular (Figura 3.9) e a partir desta, são obtidas a pose (x, y, θ) e as dimensões (w, l) do contorno atual. No decorrer do processo, a única informação importante é a pose, que irá guiar o controlador do robô (Seção 3.4.2).

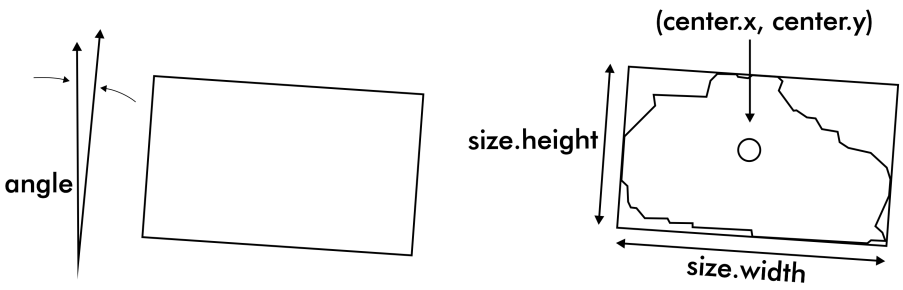


Figura 3.9: Enquadramento do contorno em uma estrutura retangular [Bradski & Kaehler, 2008].

3.4.2 Controlador de Movimento do Robô

O controle de navegação é feito utilizando o método de Campo Potencial Artificial [Khatib, 1986], amplamente utilizado na literatura. O método considera o robô como um ponto $q = (x, y)$ sob influência de um campo potencial artificial $U(q)$ [Siegwart & Nourbakhsh, 2004]. O robô segue essa função considerando que há um campo atrativo U_{att} criado a partir do destino e um campo repulsivo U_{rep} criado a partir da posição dos obstáculos (Equação 3.5).

$$U(q) = U_{att}(q) + U_{rep}(q). \tag{3.5}$$

Considerando que U é uma função diferenciável, pode-se relacionar a força artificial que age na posição q como:

$$F(q) = -\nabla U(q), \tag{3.6}$$

onde $\nabla U(q)$ é o vetor gradiente de U no ponto q.

Neste trabalho, o campo potencial atrativo U_{att} é definido como uma função quadrática (Equação 3.7).

$$U_{att} = \frac{1}{2} k_{att} \cdot \rho^2_{destino}(q), \tag{3.7}$$

onde k_{att} é um fator de escala e $\rho(q)$ é a distância Euclidiana $\parallel q - q_{destino} \parallel$, onde $q_{destino}$ é a posição de destino. Esse campo conduz à força de atração F_{att} representada na Equação 3.8.

$$\begin{aligned} F_{att}(q) &= -\nabla U_{att}(q), \\ &= -k_{att} \cdot (q - q_{destino}). \end{aligned} \tag{3.8}$$

A repulsão é feita utilizando o método *Vortex Field* [Luca & Oriolo, 1994]. A ideia é substituir o campo repulsivo contrário ao gradiente atrativo por um fluxo de rotação em torno do objeto. O campo repulsivo U_{rep} e a força repulsiva estão representados nas Equações 3.9 e 3.10, respectivamente.

$$U_{rep}(q) = \begin{cases} \frac{1}{2} k_{rep} \left(\frac{1}{\rho(q)} - \frac{1}{\rho_0} \right)^2 & \text{se } \rho(q) < \rho_0, \\ 0 & \text{se } \rho(q) \geq \rho_0, \end{cases} \tag{3.9}$$

onde k_{rep} é um fator de escala, $\rho(q)$ é a distância mínima entre a posição q e o obstáculo detectado e ρ_0 é a distância de influência do obstáculo.

$$F_{rep}(q) = \pm \begin{bmatrix} \frac{\partial U_{rep}(q)}{\partial y} \\ -\frac{\partial U_{rep}(q)}{\partial x} \end{bmatrix}. \tag{3.10}$$

O sinal $\pm$ na Equação 3.10 indica o sentido no qual o robô irá circundar o obstáculo. Com o sinal $+$ o sentido será anti-horário.

A força que atua sobre a posição q será composta por:

$$F(q) = F_{att}(q) + F_{rep}(q), \tag{3.11}$$

onde $F_{att}(q)$ é força de atração para uma posição de destino, $F_{rep}(q)$ é a força de repulsão dos obstáculos.

Quando o obstáculo detectado for um objeto, uma força atrativa conduz o robô para o centro atual deste. Ou seja, o destino será o centro do objeto. De acordo com a informação de pose do objeto obtida a cada iteração do mapeamento (Seção 3.4), é criado um campo atrativo para o centro do objeto e um campo repulsivo tangente ao mesmo. A soma desses dois campos cria um fluxo de rotação em torno desse objeto.

Se o obstáculo detectado não for um objeto para mapeamento, o destino do robô é uma posição diferente da posição do obstáculo detectado. Portanto o campo de atração será criado para essa posição de destino e o robô irá circundar o objeto somente para desviar do mesmo e será atraído para a posição de destino enquanto desvia do obstáculo.

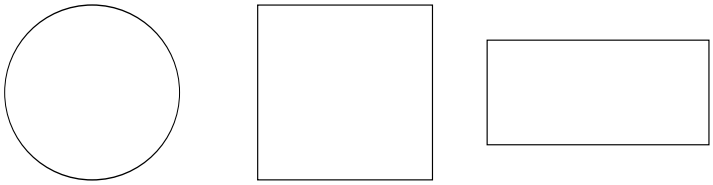
3.4.3 Tipo do Objeto

Assume-se neste trabalho que os tipos de objetos presentes no ambiente variam de acordo com sua forma geométrica (2D), podendo ser em forma: circular (`TYPE_CIRCLE`) ou retangular (`TYPE_RECT`). A determinação do tipo do objeto é realizado por meio de correspondência de imagens (*matching*).

A correspondência consiste em comparar duas imagens e determinar um valor de similaridade entre elas.

Para cada forma geométrica determinada, existe uma imagem de referência (Figura 3.10), que é o modelo para comparação. O tipo `TYPE_RECT` tem dois modelos para comparação uma com todos os lados iguais e outra com os lados diferentes. Foi utilizado dois modelos devido à variação que o tipo retangular pode ter. Um objeto de forma quadrática também tem a forma retangular.

Para se determinar o tipo do objeto, compara-se o contorno 2D da imagem



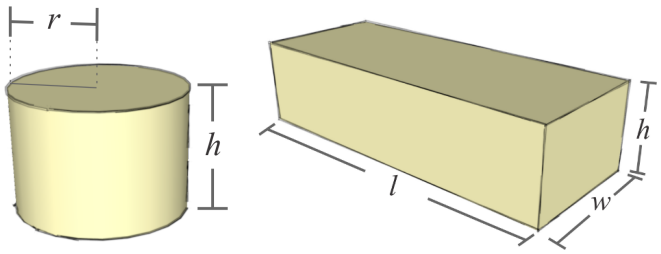
(a) TYPE_CIRCLE (b) TYPE_RECT (c) TYPE_RECT

Figura 3.10: Modelos das formas geométricas consideradas para a correspondência entre as imagens 2D dos objetos.

obtida no mapeamento com o contorno da imagem de todos os modelos. A comparação que retornar o menor valor corresponde à maior similaridade entre os objetos. O objeto será classificado com o mesmo tipo do modelo com o qual obteve maior similaridade. Apenas os valores menores que um limite th_m são considerados. Sendo assim, após a comparação com todos os modelos, o objeto pode ficar sem um tipo determinado. Se isso acontecer, não será possível transportá-lo.

3.4.4 Dimensão

De acordo com as dimensões do robô e do objeto são estimados o número mínimo N_{R_MIN} e máximo N_{R_MAX} de robôs necessários para executar o transporte. Os tipos e dimensões dos objetos considerados neste trabalho são apresentados na Figura 3.11.



(a) TYPE_CIRCLE (b) TYPE_RECT

Figura 3.11: Tipos dos objetos considerados e suas dimensões.

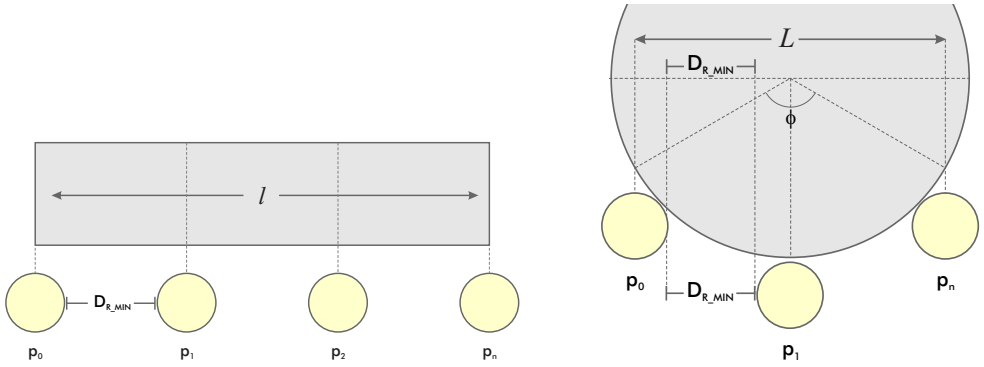
Adotou-se como critério para calcular o número de robôs a Equação 3.12. Se o objeto tiver a forma circular, o cálculo será feito considerando-se o diâmetro do mesmo. Caso o objeto seja em forma retangular, o cálculo será feito considerando o comprimento l do objeto. Quando o robô termina o mapeamento do objeto, ele atribui a l o maior lado, portanto $l > w$ sempre.

$$L = \begin{cases} 2r\sin(\frac{\phi}{2}) & \text{se objeto é TYPE\_CIRCLE,} \\ l & \text{se objeto é TYPE\_RECT} \end{cases} \tag{3.12}$$

onde, l é a largura dos objetos retangulares e ϕ é o ângulo considerado para medir a largura nos objetos circulares (Figura 3.12).

Se L for maior que o tamanho limite L_{lim}, o robô explorador precisa de no mínimo mais um robô para ajudá-lo na tarefa. O número mínimo de robôs para executar o transporte será dado pela Equação 3.13:

$$N_{R\_MIN} = \begin{cases} 1 & \text{se } L \leq L_{lim}, \\ 2 & \text{se } L > L_{lim}. \end{cases} \tag{3.13}$$



(a) TYPE_RECT (b) TYPE_CIRCLE

Figura 3.12: No transporte cooperativo, cada robô envolvido tem uma posição p_i específica em relação ao objeto. Os robôs devem ficar à uma distância mínima D_{R_MIN} uns dos outros.

Para determinar o número máximo de robôs que podem fazer parte do transporte, leva-se em consideração, além do raio r_r do tamanho do robô, a distância D_{R_MIN} mínima que eles devem estar uns dos outros, conforme representação na Figura 3.12. O número máximo de robôs para executar o transporte será dado pela Equação 3.14:

$$N_{R\_MAX} = \begin{cases} 1 & \text{se } L \leq L_{lim}, \\ 2 + \frac{L-(2r_r+D_{R\_MIN})}{2r_r+D_{R\_MIN}} & \text{se } L > L_{lim} \end{cases} \tag{3.14}$$

Considerando a quantidade mínima de robôs para a execução do transporte, o robô explorador consegue verificar se precisará de ajuda para realizar o transporte do objeto. Porém, só essa informação não é suficiente. Se o peso do objeto for

considerado, pode haver casos de objetos de dimensões menores apresentarem peso além da capacidade de transporte do robô. Portanto, o robô ainda precisa verificar se é capaz de transportar o objeto considerando esse peso. O processo citado será descrito na próxima seção.

3.4.5 Estimação do Esforço

Determinadas as dimensões e o tipo do objeto, a etapa de exploração do objeto é concluída pela estimação do esforço necessário para que o robô execute o transporte. Neste trabalho, entende-se por esforço, a potência elétrica do robô despendida para empurrar um determinado objeto ou o torque das juntas do manipulador robótico para levantar um objeto.

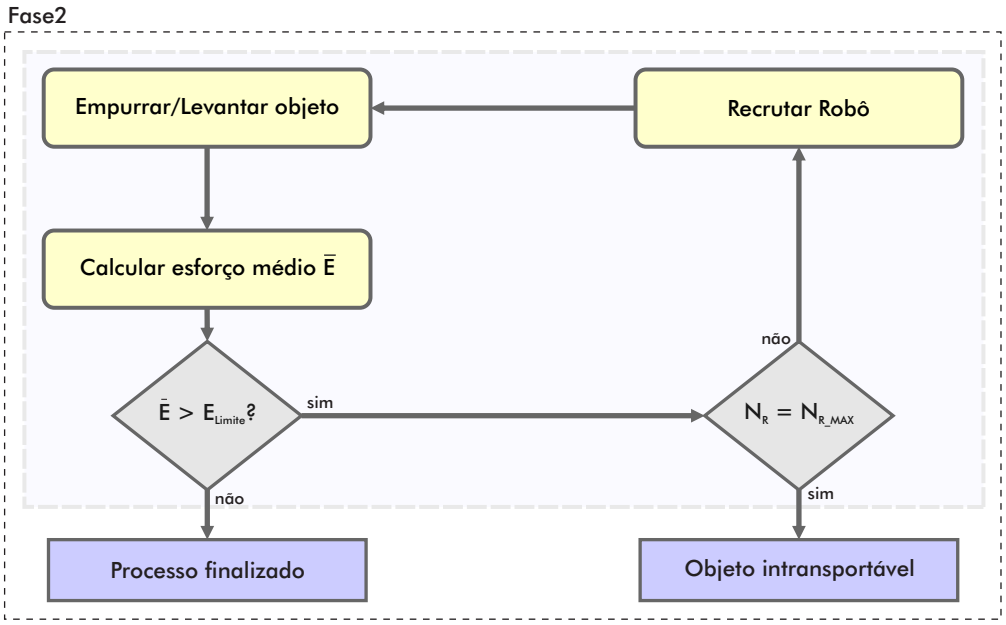


Figura 3.13: Fase 2: Processo de estimação do esforço para transporte do objeto.

Considerando os dois tipos de robôs existentes no time: `PUSHER` e `GRASPER`, os robôs com capacidade de empurrar estimam o esforço pela potência elétrica dos motores de tração. Os robôs que executam o transporte com o manipulador estimam o esforço pelo valor do torque das juntas do manipulador.

3.4.5.1 Esforço para transporte do tipo `PUSH`

Como será mostrado a diante, pode-se relacionar a potência elétrica do robô com o peso do objeto que ele está tentando empurrar. Primeiramente, serão definidas

as forças que atuam sobre o objeto para que esse entre em movimento. Posteriormente, será mostrado como a força e a potência se relacionam.

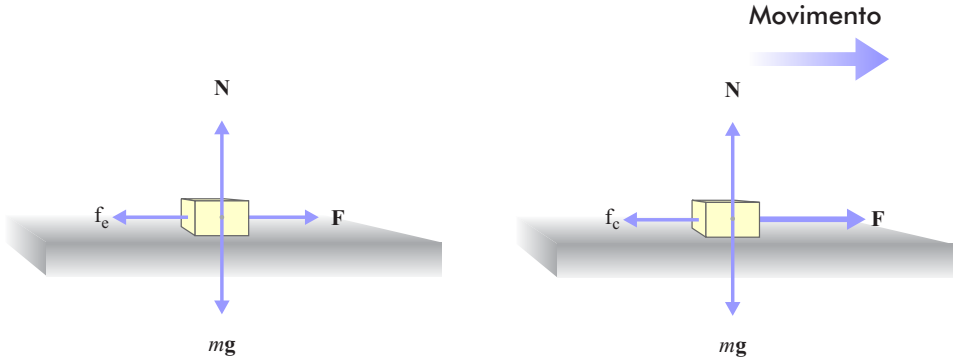
Um objeto sobre uma superfície está sujeito à ação da força de atrito (f). As forças de atrito sempre se opõem ao movimento. Para que um objeto se movimente é necessário que a resultante das forças aplicadas sobre ele seja diferente de zero. A Equação 3.15 mostra que o módulo da força de atrito f é dada pelo produto do coeficiente de atrito μ e do módulo da força normal N. A força normal tem o módulo do peso, que é dado pelo produto da massa m pela gravidade g.

$$\begin{aligned} f &= \mu N, \\ f &= \mu \cdot m.g. \end{aligned} \tag{3.15}$$

A Figura 3.14 mostra que enquanto o objeto não entra em movimento, $f = f_e = \mu_s N$, onde f_e é a força do atrito estático e μ_s corresponde ao coeficiente de atrito estático. A força f_e age sobre o objeto até que a força F aplicada no objeto vença a força de atrito estático. A partir desse momento o objeto entra em movimento e $f = f_c = \mu_c N$, onde f_c é a força do atrito cinético e μ_c é o coeficiente de atrito cinético [Resnick & Halliday, 1983].

Considerando a Equação 3.15 e que os objetos a serem transportados estão no mesmo ambiente, μ e g serão constantes. Logo, a força necessária para o transporte dependerá apenas do valor de m.

Na metodologia proposta não se tem informação das forças aplicadas sobre o objeto, mas sim, da potência elétrica aplicada ao motor do robô. Pode-se relacionar essa potência com a força **F** e a velocidade **v** conforme a Equação 3.16.



(a) Atrito Estático (b) Atrito Cinético

Figura 3.14: Forças que atuam no objeto. Adaptação de Resnick & Halliday [1983].

$$P = \frac{W}{t} = \frac{\mathbf{F} \cdot \mathbf{d}}{t} = \mathbf{F} \cdot \mathbf{v}. \tag{3.16}$$

Quando o objeto está parado, a força $F = f_e$. Quando a força F supera f_e, inicialmente tem-se uma aceleração que provoca o movimento do objeto e posteriormente, mantendo-se a mesma força F, o objeto se move com velocidade constante e a força $F = f_c$, de tal forma que $\sum F = 0$. Substituindo-se a força F na Equação 3.16, pela força de atrito (f_e ou f_c), pode-se dizer então que

$$P = \mu.m.g.v. \tag{3.17}$$

Observa-se que Equação 3.17 só é válida quando o objeto está parado ou quando está em movimento retilíneo uniforme. Pois, na transição "parado" para "em movimento", $f_c < F \leq f_e$. Portanto, considerando-se o período em que o robô está empurrando o objeto com velocidade constante, pode-se dizer que a potência dependerá apenas do valor da massa m.

Considerando a Equação 3.17, dado que μ e g são constantes durante o processo de estimação do esforço, se o robô empurrar um objeto a uma velocidade constante (v_{cte}), a potência variará de acordo com o peso do objeto. Sendo assim, o processo de estimação do esforço do robô consiste em:

1. o robô empurra o objeto a uma velocidade v_{cte} por um período de tempo;
2. realiza a leitura do valor da potência atual (P) a cada instante;
3. estima o valor médio da potência ($\bar{P}$) medida nesse intervalo de tempo.

Considerando que o robô tem uma potência limite P_{lim}, a decisão de empurrar o objeto sozinho ou pedir ajuda é feita de acordo com a média da potência. Se $\bar{P} > P_{lim}$ o robô solicitará ajuda de até N_{R_MAX} robôs, estimado na fase anterior (Seção 3.4.4). O limite P_{lim} é a potência necessária para transportar o objeto de maior peso dentre os que o robô consegue empurrar sozinho.

3.4.5.2 Esforço para transporte do tipo `GRASP`

Para o transporte utilizando o manipulador, o esforço deve ser calculado em relação à capacidade desse em carregar um objeto.

O manipulador utilizado possui 4 Graus de Liberdade (*Degrees-of-Free*, ou DoF) e é controlado por servo motores (Figura 3.16). A capacidade de carga do

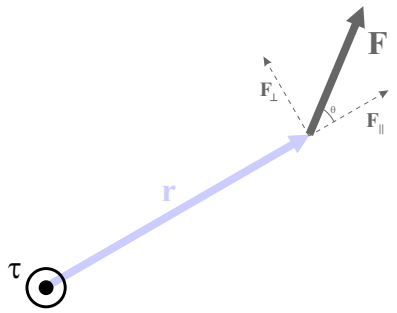
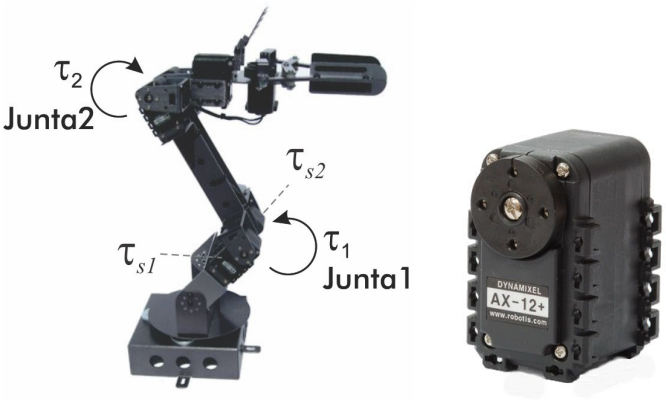


Figura 3.15: Relação entre o torque τ, a força **F** aplicada e a distância **r**. Adaptação de Resnick & Halliday [1983]

manipulador será dada pelo torque τ dos motores. O torque é uma a força que produz rotação ou torção, cujo módulo é dado pela relação:

$$\tau = rF \text{ sen } \theta. \tag{3.18}$$

onde r é a distância em relação ao ponto de rotação, F é a força aplicada e θ é o ângulo formado entre **r** e **F**. Como os objetos estão suspensos do solo, $F = mg$, então pode-se relacionar o torque ao peso do objeto a ser transportado.



(a) *Crust Crawler AX-12 SmartArm* (b) Servo motor AX-12+

Figura 3.16: Manipulador e servo motor.

As juntas envolvidas no processo de levantar o objeto são as Junta1 e Junta2 (Figura 3.16). Cada uma dessas juntas é controlada por dois servos. O esforço de cada junta será dado pela soma do torque torque dos dois servos.

$$\tau_i = \tau_{s1} + \tau_{s2}. \tag{3.19}$$

Como a Junta1 é responsável por levantar todo o braço, o esforço será cal-

culado em relação ao torque nessa junta. O processo de estimação do esforço do manipulador consiste em:

1. o robô tenta levantar o objeto a uma certa altura pré-definida, por um período de tempo;

2. realiza a leitura de τ_{si} dos servos e calcula o esforço da junta τ_1 a cada instante;

3. calcula o esforço médio $(\bar{\tau_1})$ da junta nesse intervalo de tempo.

Da mesma forma que para potência, os servos têm um torque limite τ_{lim} que pode ser aplicado para o manipulador carregar um objeto. Se a junta 1 tiver o esforço maior que τ_{lim}, o objeto não pode ser transportado. O limite τ_{lim} é o esforço necessário para se transportar o objeto de maior peso dentre os que o robô consegue carregar.

3.5 Recrutamento

O processo de recrutamento é bastante estudado em sociedades de insetos sociais (tais como formigas e abelhas). A maneira que um integrante pede ajuda aos companheiros pode ser, por exemplo, por meio de feromônio, como acontece com as formigas. Outro exemplo são algumas espécie de abelhas, que utilizam dança para solicitar ajuda [Beekman & Dussutour, 2009].

Dentre as diferenças das espécies, o processo de recrutamento das formigas para o transporte cooperativo é realizado basicamente em três passos [Kube & Bonabeau, 2000; dos Santos & Bazzan, 2009]:

1. A formiga que descobriu o alimento inicialmente tenta transportá-lo sozinha. Se não conseguir, inicia-se o processo de recrutamento. A comunicação é feita utilizando feromônios;

2. As formigas que aceitarem o recrutamento movem-se para perto do alimento;

3. A quantidade de cooperadores no transporte varia de acordo com as características do alimento a ser transportado.

Inspirado nos processos biológicos, a ajuda para o transporte é feita por meio do recrutamento. Esta etapa está diretamente ligada à fase de estimação do esforço para transportar o objeto (Seção 3.4.5).

Quando o robô explorador encontra um objeto, primeiramente ele verifica se consegue transportá-lo sozinho, observando o tipo do objeto e o esforço despendido (Seção 3.4). Somente depois, caso não consiga transportar o objeto, ele começa o processo de recrutamento. Esse processo é feito recrutando-se um robô por vez. Para cada robô que aceita a tarefa, o grupo verifica se a quantidade atual de robôs permite o transporte. Caso se atinja um número máximo de robôs e o transporte não puder ser realizado, o processo de recrutamento é abortado e o objeto é abandonado.

Neste trabalho utilizou-se a comunicação explícita para recrutar os robôs. O recrutamento é feito por meio de troca de mensagens. Pela Suposição 2, todos os robôs do grupo recebem a mensagem de pedido de ajuda.

Quando o mapeamento do objeto é finalizado, o robô decide se tem capacidade para executar o tipo de transporte definido para o objeto (empurrar ou pegar com a garra). Se o robô não tem capacidade para executar o transporte, ele inicia o processo de recrutamento, porém apenas o robô recrutado executará o transporte. Se o robô tiver capacidade para executar o transporte mas precisa de ajuda (Seção 3.4), ele começa o processo de recrutamento a fim de obter ajuda de outros robôs.

Portanto, o recrutamento ocorre em duas situações: (1^o) quando o robô explorador não tem capacidade de transportar o objeto devido à restrição de atuadores (Seção 3.6) e (2^o) quando o robô explorador é capaz de realizar o transporte, mas precisa de ajuda.

Quando o robô que estava explorando o objeto decide pedir ajuda, ele se torna o recrutador e líder momentâneo dos robôs cooperadores. O robô recrutador tem papel de líder apenas para coordenar o time quanto à posição que os mesmos devem ter em relação ao objeto e verificar se o transporte pode começar efetivamente. A partir desse momento, cada robô interage com o objeto para realizar o transporte.

Para o recrutamento, o robô recrutador calcula a posição que o outro robô deve assumir, conforme o número de robôs envolvidos no transporte e as dimensões do objeto.

No primeiro recrutamento, o robô recrutador calcula a posição que ele e o robô recrutado podem assumir. Então, escolhe para si a posição mais próxima e envia uma mensagem recrutando um robô para a outra posição. Inicialmente são alocadas as posições extremas em relação ao objeto. Esses robôs das extremidades nunca mudam de posição até o fim do processo de transporte. Somente os robôs das posições internas necessitam de uma realocação de posição em certos casos. À partir do segundo recrutamento, o robô recrutador recalcula as posições internas

para o novo número de robôs e verifica qual posição que deve ser preenchida.

O recrutamento é baseado em um Algoritmo de Leilão [Gerkey & Mataric, 2002b] e consiste em cinco etapas:

1. **Anúncio da tarefa:** o robô que realizou a exploração do objeto se torna o "leiloeiro" enviando uma mensagem solicitando ajuda para todos os robôs. A mensagem enviada contém informações da dimensão, tipo e pose do objeto e também do tipo de transporte a ser realizado;

2. **Análise:** cada robô analisa os ganhos e custos para executar a tarefa e calcula o seu "lance". É necessário uma métrica para escolher o melhor robô naquele momento para executar a tarefa. Neste caso, usou-se o tipo do robô e a distância Euclidiana dele até o objeto;

3. **Lance:** cada robô publica seu "lance" para executar a tarefa enviando uma mensagem para o "leiloeiro" com um valor numérico que representa o valor para executar a tarefa;

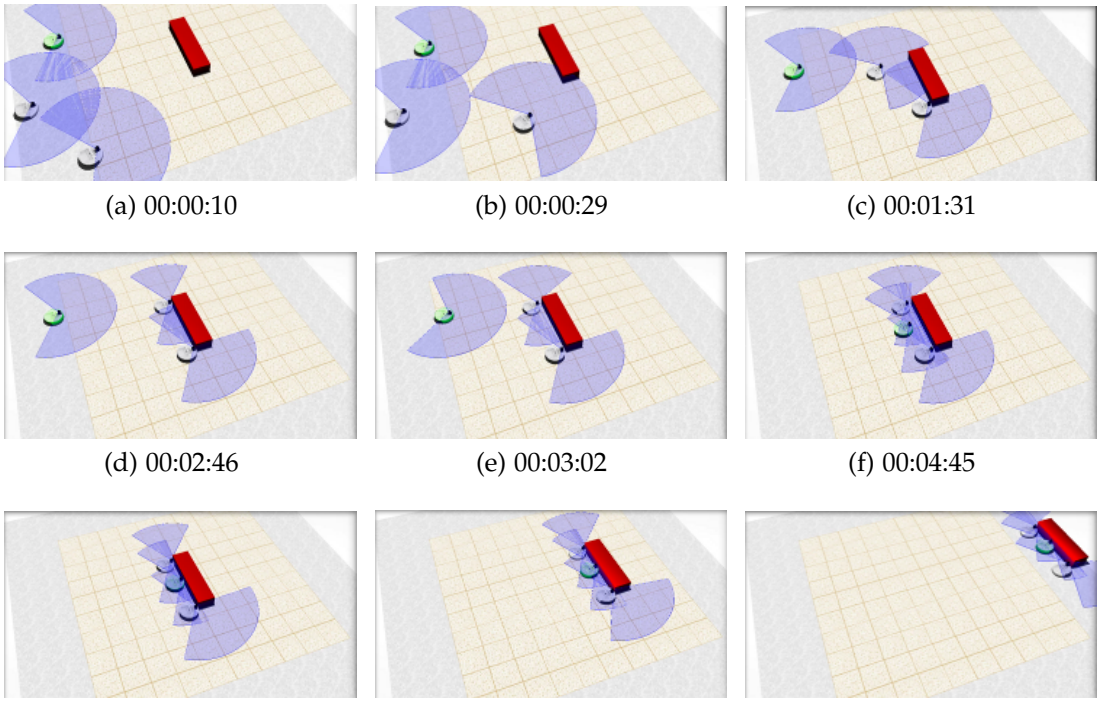
4. **Fim do leilão:** após um período de tempo definido, o robô "leiloeiro" processa os "lances" dos robôs e envia uma mensagem final recrutando o vencedor. Os outros robôs recebem uma mensagem liberando-os da tarefa;

5. **Acompanhamento da tarefa:** Quando o recrutador não participa do transporte, aguarda uma mensagem de término do robô que está executando a tarefa.

Os robôs que recebem o pedido de ajuda calculam o seu lance para a tarefa de acordo com a distância Euclidiana até a posição de destino, o tipo do transporte e o tipo do robô. Quanto menor a distância, maior será o lance. Para o transporte do tipo `GRASP` (Seção 3.6), somente os robôs com manipulador respondem ao pedido de ajuda considerando apenas a distância Euclidiana. Já no transporte tipo `PUSH`, todos os robôs respondem ao pedido de ajuda, sendo que o robô sem o manipulador terá uma tendência maior para executar essa tarefa. O valor do lance do robô é calculado conforme a Equação 3.20:

$$l_p = \frac{1}{d_p(1-\beta)}, \tag{3.20}$$

onde d_p é distância Euclidiana do robô até a posição de destino e β é a tendência do robô para executar a tarefa e pode assumir os valores entre $[0, 1[$.



(a) 00:00:10 (b) 00:00:29 (c) 00:01:31

(d) 00:02:46 (e) 00:03:02 (f) 00:04:45

(g) 5 : 00*min* (h) 5 : 14*min* (i) 5 : 47*min*

Figura 3.17: Recrutamento e transporte cooperativo: (3.17a) inicialmente o robô do canto inferior é o robô explorador e os outros robôs se encontram no estado de espera. (3.17b) O robô explorador encontra um objeto, estima as informações de dimensão e forma e verifica o tipo de transporte a ser executado. O robô verifica que tem capacidade para executar o tipo de transporte, porém precisa de ajuda. (3.17c) O robô explorador recruta um ajudante, aguarda a sua chegada para (3.17d) cooperativamente estimarem o esforço necessário para o transporte. Como o esforço é maior que o limite determinado, (3.17e) o robô recrutador solicita ajuda de mais um robô e (3.17f) aguarda sua chegada. (3.17g) Ao estimarem o esforço (3.17h)verificam que podem transportar o objeto até o (3.17i) destino. O robô explorador volta a explorar o ambiente e o robô recrutado volta ao estado de espera.

A Figura 3.17 apresenta um exemplo de recrutamento. O robô do tipo `GRASPER` está representado pela cor verde. Quando o robô explorador encontra um objeto e verifica que precisa de ajuda, inicia o processo de recrutamento. O primeiro robô recrutado é do tipo `PUSHER` (Figura 3.17c) devido à sua maior tendência para executar a tarefa. Os dois robôs não conseguem realizar o transporte então, o recrutador solicita ajuda de mais um robô (Figura 3.17e). Ao fazer o novo recrutamento, recalcula as posições para a nova quantidade de robôs e verifica que o novo robô deve ocupar a posição do meio. Como o robô tipo `GRASP` é o único robô disponível e tem condições de realizar o transporte, ele se aproxima do objeto enquanto os outros aguardam sua chegada para dar continuidade ao processo de

transporte.

Para a coordenação entre os robôs, o processo de estimação do esforço só acontece quando todos os robôs estão prontos para executá-lo. O robô recrutador é quem coordena. Cada robô que chega à sua posição de destino envia uma mensagem ao robô recrutador e quando este receber confirmação de todos os robôs envolvidos, envia uma mensagem de volta confirmando o início do processo de estimação do esforço.

3.6 Transporte

No cenário proposto tem-se objetos de tamanho, forma e peso variados e também robôs com capacidades distintas. Alguns dos objetos são pequenos e leves, podendo ser transportados com o auxílio de um manipulador robótico. Já outros objetos, por serem grandes ou pesados, obrigatoriamente serão empurrados.

Os robôs podem ser classificados em dois tipos dependendo da capacidade de transporte: `GRASPER`, que possui um braço robótico acoplado à sua base, e `PUSHER`. Os robôs do primeiro tipo têm a capacidade de pegar com a garra ou empurrar os objetos, já os do segundo só têm a capacidade de empurrar os objetos. Para executar a tarefa de transporte, o robô pode fazê-la de três maneiras:

- `PUSH_ALONE`: um robô transporta o objeto empurrando-o até o destino.
- `PUSH_COOPERATION`: dois ou mais robôs transportam o objeto empurrando-o até o destino
- `GRASP_ALONE`: um robô transporta o objeto segurando-o com a ajuda de um braço robótico com uma garra.

Pode-se definir então, os tipos de transporte que cada tipo de robô pode executar, conforme a Tabela 3.1.

Tipo	Capacidade de Transporte		
	`PUSH_ALONE`	`PUSH_COOPERATION`	`GRASP_ALONE`
`GRASPER`	√	√	√
`PUSHER`	√	√	

Tabela 3.1: Tipos de transporte que o robô pode realizar de acordo com sua capacidade de atuação.

Dada as dimensões do objeto, a decisão entre o tipo de trasporte `PUSH` ou `GRASP` será tomada conforme a capacidade de abertura da garra do braço robótico.

Conforme a Equação 3.21, os objetos de dimensão w_o menores que a abertura limite T_{lim} da garra podem ser transportados utilizando o manipulador robótico.

$$\texttt{TIPO\_TRANSPORTE} = \begin{cases} \texttt{GRASP} & \text{se } w_o \leq T_{lim}, \\ \texttt{PUSH} & \text{se } w_o > T_{lim} \ . \end{cases} \tag{3.21}$$

Se o transporte for do tipo `PUSH`, o robô ainda tem de decidir se irá transportar o objeto sozinho (`PUSH_ALONE`), ou se redirá ajuda (`PUSH_COOPERATION`) para executar o transporte, conforme os parâmetros descritos na Seção 3.4.

3.6.1 *Pushing*

No transporte do tipo `PUSH`, baseado no trabalho de Gerkey & Mataric [2002a], o controle dos robôs é realizado calculando-se as velocidades linear (v_{obj}) e angular (w_{obj}) desejados para o objeto e posteriormente as velocidade são distribuídas entre os robôs transportadores.

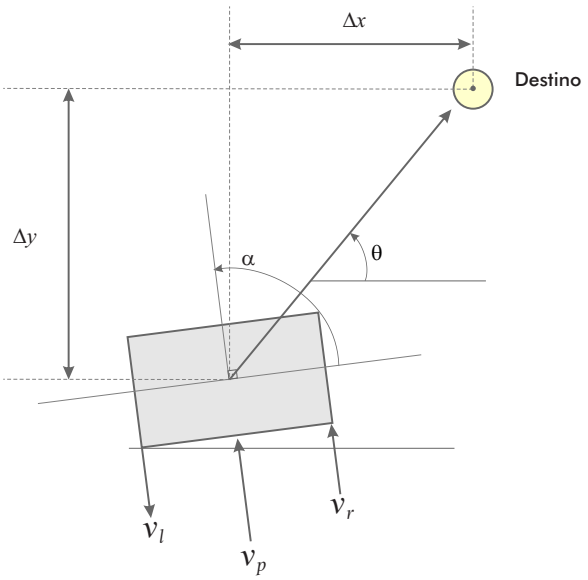


Figura 3.18: Δx e Δy são as distâncias do objeto no eixo X e Y, respectivamente, em relação ao destino. θ é o ângulo formado entre a posição atual do objeto e o destino. α é a orientação do objeto. v_p é a velocidade calculada para o objeto e v_l e v_r são as velocidades referentes à v_p distribuídas nas extremidades do objeto

Considerando o destino do objeto (Figura 3.18), as velocidades v_{obj} e w_{obj} são calculadas conforme a Equação 3.22. A velocidade v_{obj} é obtida pela distância Euclidiana do objeto até o destino, multiplicada por um ganho k_v.

Para calcular a velocidade w_{obj}, determina-se o ângulo formado entre o centro do objeto e a posição de destino, dado por θ. A cada instante o objeto pode sofrer uma rotação. Para que objeto atinja o destino, a velocidade w_{obj} será dada pela diferença entre θ e α multiplicada por um ganho k_w. O valor de α é a soma do ângulo de orientação atual do objeto mais 90º.

$$\begin{aligned} v_{obj} &= k_v.\sqrt{\Delta x^2 + \Delta y^2} \\ w_{obj} &= k_w.(\theta - \alpha). \end{aligned} \tag{3.22}$$

Após achar v_{obj} e w_{obj}, as velocidades são distribuídas diferencialmente entre dois pontos extremos do objeto: v_l à esquerda e v_r à direita (Equação 3.23).

$$\begin{aligned} v_l &= v_{obj} + \frac{1}{2} w_{obj} \\ v_r &= v_{obj} - \frac{1}{2} w_{obj} \end{aligned} \tag{3.23}$$

Conforme a quantidade de robôs envolvidos no transporte, a velocidade para cada posição é calculada por meio da ponderação entre as duas velocidades extremas (v_l e v_r). A velocidade final no ponto em que o robô deve empurrar o objeto é dada pela Equação 3.24.

$$v_p = v_l + (v_r - v_l).p.k_p, \tag{3.24}$$

onde,

$$k_p = \begin{cases} 1 & \text{se } N_t = 1, \\ \frac{1}{N_t - 1} & \text{se } N_t > 1 \end{cases}$$

é a fração de espaço em que o objeto foi dividido, conforme o número N_t de robôs participantes do transporte. A posição p do robô em relação ao objeto é calculada e enviada pelo robô recrutador.

3.6.1.1 Estimação da Posição do Objeto

Durante o transporte do tipo `PUSH`, a posição do objeto é estimada a cada instante por meio das leituras realizadas pelo *laser*. Essa leitura retorna um vetor de pontos no formato (ρ_i, d_i), em que ρ representa a angulação onde determinado valor foi obtido e d_i a distância (em metros) do *laser* até o obstáculo detectado e na i-ésima posição do vetor. Para evitar interferência de leituras de outros obstáculos, apenas as leituras menores que um limite L_{lim} são consideradas.

O *laser* está deslocado no eixo X do robô à distância $desloc_l$ do centro do

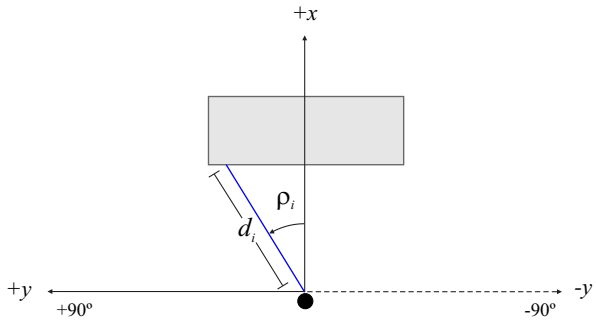


Figura 3.19: Variáveis envolvidas na leitura do *laser*.

robô. Para transformar as medidas em relação ao referencial do robô é feita a transformação:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = d_i \begin{bmatrix} \cos\rho_i \\ \sin\rho_i \end{bmatrix} + \begin{bmatrix} desloc_l \\ 0 \end{bmatrix}$$

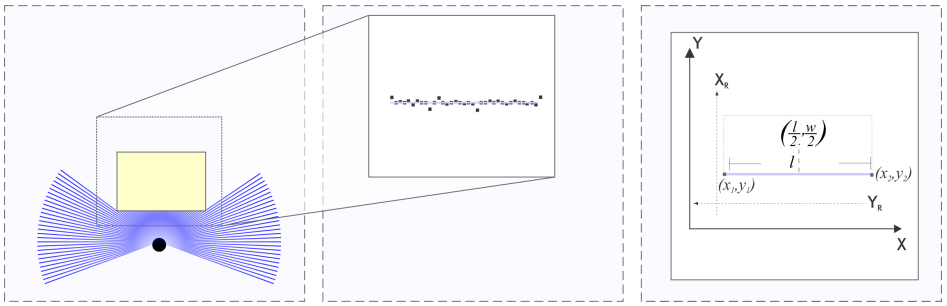


Figura 3.20: Leitura do *laser* para estimação da posição de objetos do tipo `TYPE_RECT`.

Para os objetos do `TYPE_RECT`, as leituras do *laser* são processadas por um método de estimação de retas (usou-se quadrados mínimos e RANdom SAmple Consensus (RANSAC) Fischler & Bolles [1981]). Por meio desse processamento são obtidos os pontos extremos (x_1, y_1) e (x_2, y_2) e o ângulo θ_s dessa reta em relação à posição do robô.

Quando o transporte for realizado por um robô apenas, o *laser* consegue atingir as duas extremidades do objeto correspondestes aos pontos (x_1, y_1) e (x_2, y_2). Se o transporte for cooperativo, apenas o robô recrutador estima a posição e envia aos outros. Este deverá ocupar uma das posições extremas do objeto para estimar a posição.

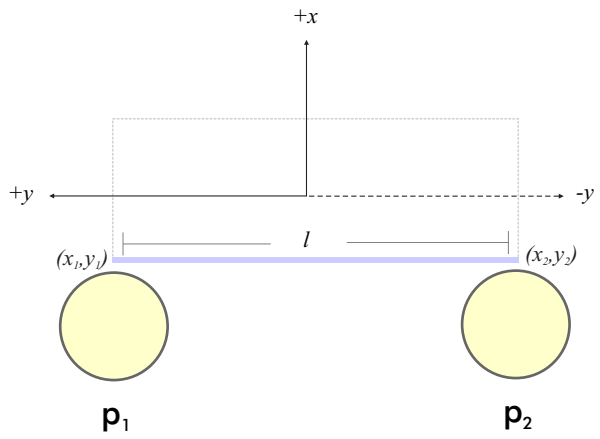


Figura 3.21: Posições que os robôs podem assumir para estimar a posição do objeto.

Na posição 1, o valor de $y2$ será usado como base e

$$y_1 = y_2 - l,$$

na posição 2, o valor de $y1$ será usado como base e

$$y_2 = y_1 + l.$$

A posição do centro do objeto será dada por:

$$\begin{aligned} x_m &= \frac{x_1 + x_2}{2} + \frac{w}{2}, \\ y_m &= \frac{y_1 + y_2}{2}. \end{aligned} \tag{3.25}$$

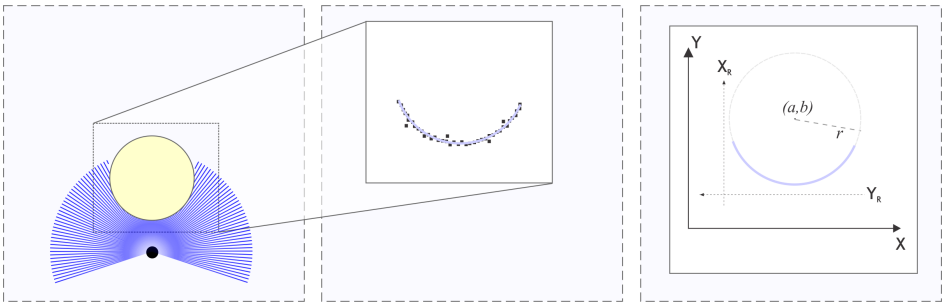


Figura 3.22: Leitura do *laser* para estimação da posição de objetos do tipo `TYPE_CIRCLE`.

Para objetos do tipo `TYPE_CIRCLE`, as leituras do *laser* são processadas por

um método de estimação de círculos (usou-se ajuste de curva implícita). Como resultado tem-se o centro (a, b) do círculo e a posição do centro do objeto será:

$$\begin{aligned} x_m &= a \\ x_y &= b. \end{aligned} \tag{3.26}$$

Os valores (x_m, y_m) encontrados são em relação ao robô. Dada a posição global do robô (x_r, y_r, θ), a posição global do objeto (x_o, y_o) é descrita por:

$$\begin{bmatrix} x_o \\ y_o \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_m \\ y_m \end{bmatrix} + \begin{bmatrix} x_r \\ y_r \end{bmatrix}$$

A orientação do objeto (θ_o) será dada conforme a Equação 3.27. A orientação do objeto do tipo `TYPE_CIRCLE` será considerado nula em todos os casos e a orientação do objeto do tipo `TYPE_RECT` será dada pela soma do ângulo da reta estimada (θ_s) e da orientação do robô (θ_r).

$$\theta_o = \begin{cases} \theta_r + \theta_s & \text{se } \texttt{TYPE\_RECT}, \\ 0 & \text{se } \texttt{TYPE\_CIRCLE}. \end{cases} \tag{3.27}$$

3.6.2 *Grasping*

No transporte do tipo `GRASP_ALONE`, o robô se posiciona perpendicularmente à dimensão l_o, próximo ao centro do objeto para facilitar que a garra prenda-o pelo seu lado w_o (Figura 3.23).

O transporte do tipo `GRASP_ALONE` envolve quatro passos básicos:

1. aproximação do objeto;
2. captura do objeto;
3. navegação até o destino;
4. deposição do objeto.

Os passos que envolvem controlar o manipulador são: (2) pegar e (4) deixar o objeto. Os outros passos são apenas de navegação do robô.

Dada a posição do objeto e a posição de destino, os passos 1 e 3 são realizados pelo controlador de navegação (Seção 3.4.2).

Devido a possíveis erros de atuação, a posição do objeto é novamente estimada quando o robô se aproxima do objeto. Essa é feita por meio da leitura do

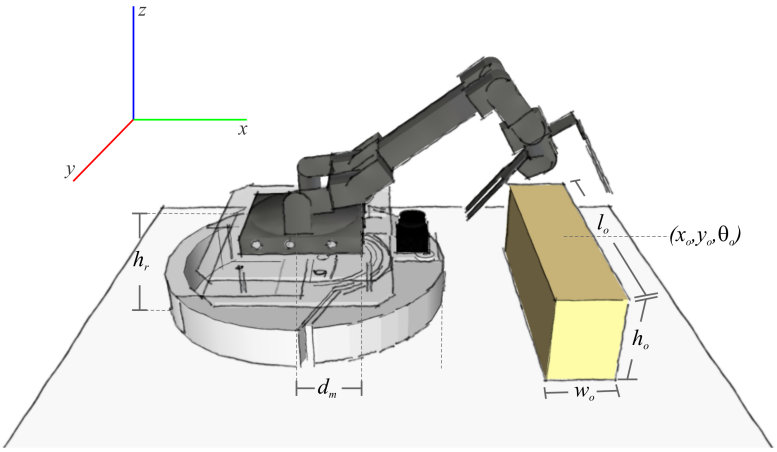


Figura 3.23: Posicionamento do robô e do manipulador em relação ao objeto a ser transportado.

laser. A posição local do objeto, em relação ao referencial do robô, é estimada conforme as Equações 3.25 e 3.26 da Seção 3.6.1.1.

Conforme representado na Figura 3.23, o deslocamento do manipulador nos eixos X e Y leva em consideração a distância do centro do robô ao objeto. O deslocamento no eixo Z deve levar em consideração a altura h_r da plataforma robótica e a altura h_o do objeto. Pela Suposição 4, h_o é conhecido previamente.

A posição central do manipulador está deslocada no eixo X do robô a uma distância d_m, portanto deve ser considerada no cálculo da posição de destino no eixo X. Dada a posição do robô (x_r, y_r, θ_r) e a posição do objeto (x_o, y_o, θ_o) em relação ao referencial do robô, a Equação 3.28 apresenta o cálculo dos valores de destino x, y e z para o manipulador.

$$\begin{aligned} x &= x_o - x_r - d_m, \\ y &= y_o - y_r, \\ z &= h_o - h_r. \end{aligned} \tag{3.28}$$

Como o robô se posiciona perpendicularmente à dimensão l_o do objeto, a orientação da garra deve ser de 90º para prendê-lo pela seu lado w_o.

3.6.3 Exemplo da Metodologia

Para exemplificar todo o processo da metodologia descrito nesta seção, apresenta-se na Figura 3.24 o processo do transporte de três objetos.

O time de robôs é composto por dois robôs do tipo `PUSHER` e um robô do tipo `GRASPER`. (1) Um robô do tipo `PUSHER` é o explorador. Os outros robôs permanecem

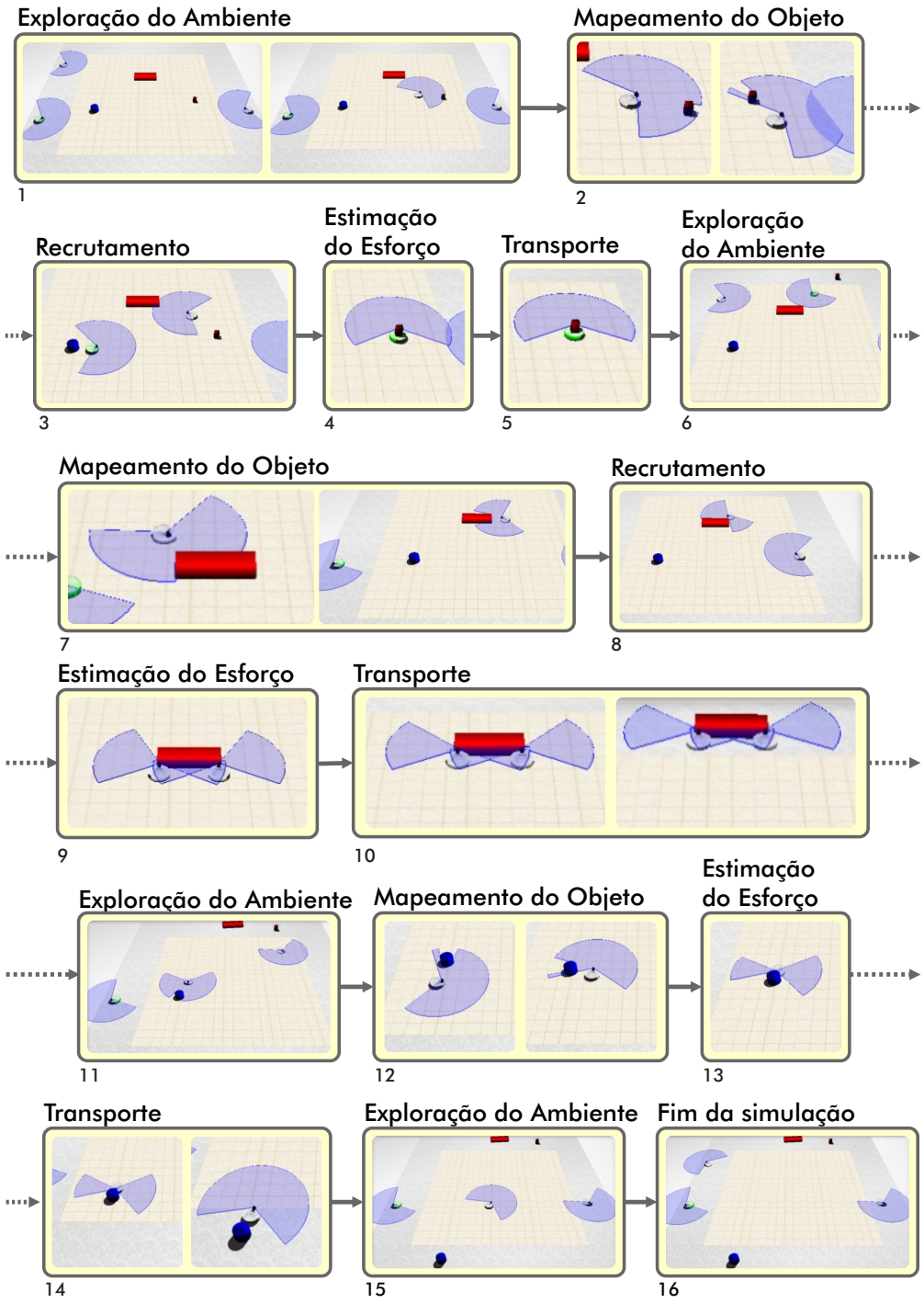


Figura 3.24: Processo de transporte de três objetos.

no estado de espera até receberem um pedido de ajuda.

(2) Após mapear o objeto 2, o robô verifica que não pode executar o tipo de transporte escolhido e faz o recrutamento do robô tipo `GRASPER`, que após estimar o esforço, faz o transporte do objeto e vai à área de espera.

O robô explorador volta a navegar pelo ambiente e (6) encontra um objeto, que deve ser transportado cooperativamente. Pela dimensão do objeto, o robô que precisa de no mínimo mais um robô, (8) então faz o recrutamento. (9) Após estimarem cooperativamente o esforço, (10) os dois robôs realizam o transporte. (11) O robô explorador volta à navegar pelo ambiente e o outro vai para o estado de espera.

Por fim, (12) o robô explorador encontra um objeto que pode ser transportado por ele apenas. (13) Após estimar o esforço, (14) realiza o transporte do objeto e (15) volta a explorar o ambiente até que o mesmo seja completamente mapeado ou se atinja o tempo máximo de exploração.

Capítulo 4

Experimentos

Neste capítulo serão apresentados e discutidos os resultados obtidos nos experimentos realizados utilizando a metodologia apresentada no Capítulo 3. Os experimentos foram realizados tanto em ambiente simulado quanto em ambiente real.

Visando explorar a eficácia de cada módulo apresentado na Seção 3.1, primeiramente foram realizados experimentos individuais para cada um dos módulos e posteriormente experimentos abrangendo o sistema completo.

4.1 Arcabouço Experimental

4.1.1 Programação

Para o controle dos robôs, foi utilizado o software *Player* do projeto *Player/Stage/Gazebo*[Gerkey et al., 2003]. O projeto oferece uma ferramenta de código aberto para controle simplificado de robôs e sensores.

Player é uma plataforma cliente/servidor que fornece uma interface, limpa e simples, de sensores do robô e atuadores. *Player* provê uma camada de abstração do hardware que permite executar algoritmos para os robôs tanto em ambiente simulado quanto em ambiente real. Utilizou-se a versão *Player 3.0.2*.

4.1.2 Ambiente de Simulação

Os algoritmos foram testado utilizando o simulador 3D *Gazebo*. Por meio da biblioteca ODE [1], *Gazebo* incorpora interações dinâmicas entre os modelos do ambiente. Utilizou-se a versão *Gazebo 0.10.0*.

[1]Open Dynamics Engine - http://www.ode.org/

Na simulação, a comunicação foi realizada por meio de troca de mensagem entre *threads*.

As simulações foram executadas em um notebook Intel Core i5 2.30GHz, 4GB com o sistema operacional Ubuntu.

4.1.3 Ambiente Real

Os experimentos em ambiente real foram realizados no Laboratório de Visão e Robótica (VeRLab) em uma área de aproximadamente $2,5m \times 2,5m$.

Os robôs utilizados consistem na plataforma robótica móvel *iRobot Create*, robô diferencial de dimensões $33cm \times 12cm$ (diâmetro x altura). Foram acoplados às plataformas robóticas um *laser* 2D *Hokuyo URG Ranger*, além disso, alguns robôs do time possuem um braço manipulador *Crust Crawler AX-12 SmartArm* conforme apresentado na Figura 4.1. Maiores detalhes da construção e implementação da plataforma estão disponíveis na página do Projeto Roomba desenvolvido no VeRLab [Verlab, 2010].

Cada robô foi equipado com um netbook Asus 1.60GHz, 1GB com o sistema operacional Ubuntu. Cada netbook executa um cliente do *Player*.

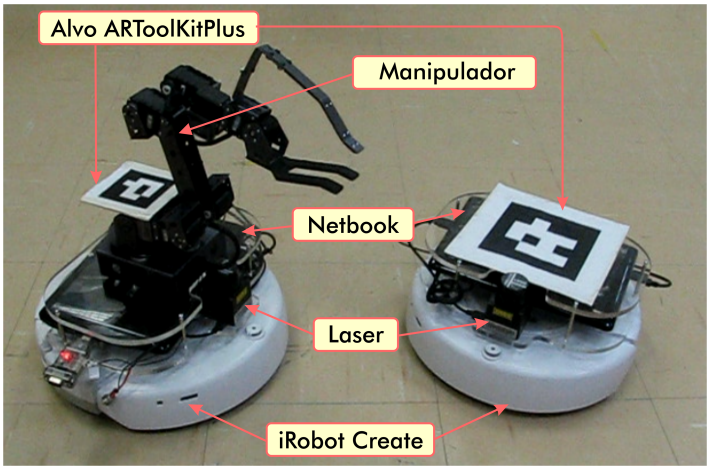


Figura 4.1: Plataforma iRobot Create acoplada com o *laser Hokuyo* e o braço manipulador *Crust Crawler*.

Os robôs possuem informação da sua posição global que é dada por meio do sistema cliente/servidor *Silvver* [Garcia et al., 2007] desenvolvido no VeRLab. O sistema consiste em rastrear alvos *ARToolKitPlus* e estimar a posição global. O servidor do *Silvver* envia pacotes contendo a posição de todos os alvos mapeados a cada instante.

A comunicação entre os robôs é feita por meio de troca de mensagens entre *threads* e na rede Wi-Fi do VeRLab. Todos os robôs recebem as mensagens que são enviadas pela rede (Suposição 2).

4.1.4 Objetos

Os objetos para o transporte são compostos pelos dois tipos descritos na metodologia, `TYPE_RECT` e `TYPE_CIRCLE`, e têm dimensões e pesos variados, sendo que:

1. Uma vez que o *laser* foi acoplado na parte superior do robô (Subseção 4.1.3), os objetos dever ter altura de no mínimo $18cm$ para serem identificados.

2. O tamanho mínimo do objeto (Seção 3.4) deve ser determinado conforme a resolução do *laser* e do mapa do objeto.

3. Quanto à largura ou diâmetro máximo não há restrição, porém esse valor deve ser dado ao algoritmo (Suposição 3);

4.2 Exploração do Ambiente

Nesse experimento, procurou-se validar a eficácia dos robôs para explorar a área determinada em busca de novos objetos.

A região de exploração possui uma área de $20m \times 20m$ e o mapa gerado pelo método de Grade de Ocupação tem resolução de $0,10m$. Foram realizadas simulações variando-se o número de robôs e o número de subáreas em 2, 3 e 4;

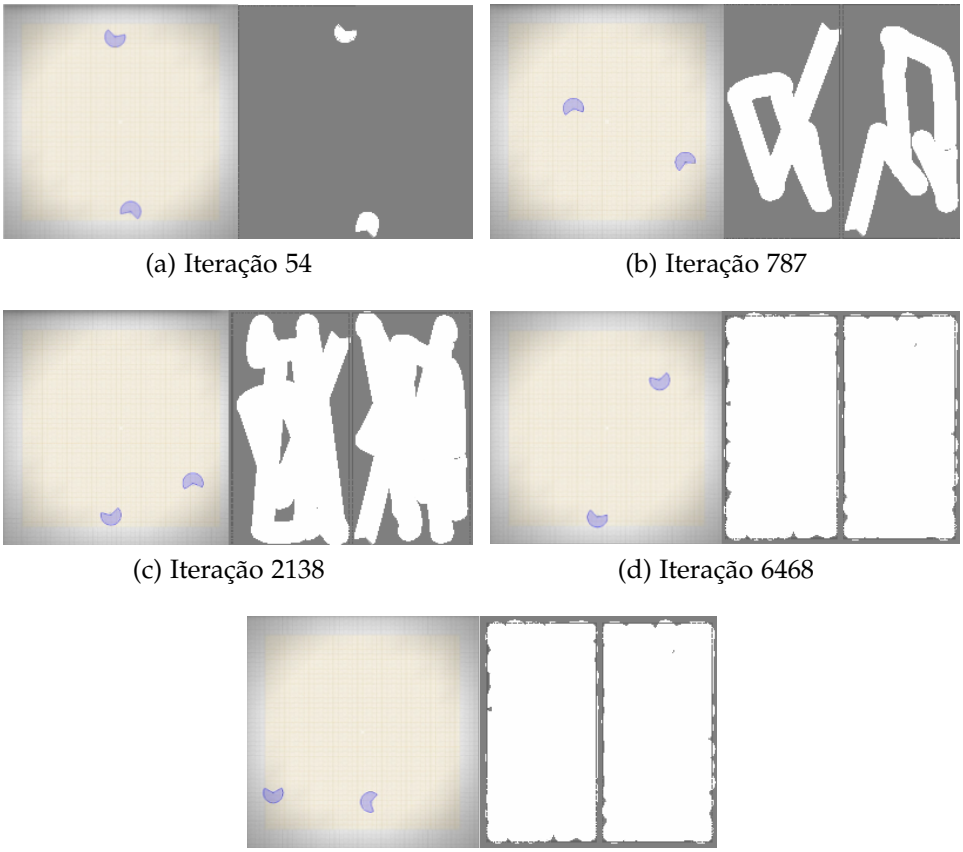
Conforme a Seção 3.2, a escolha da melhor região a ser explorada é feita pelo cálculo da entropia da região. Para esse cálculo, o tamanho da janela W (Equação 3.4) foi escolhido levando-se em consideração o alcance do *laser* para atualização do mapa, que foi determinado em $1,10m$. Como a resolução do mapa é $0,10m$, foi definido $W = 21$.

Para a implementação do mapeamento do ambiente e do objeto (Seção 3.3), utilizou-se a implementação `COccupancyGridMap2D` da biblioteca MRPT (*The Mobile Robot Programming Toolkit*)[2].

Os robôs têm conhecimento apenas do seu mapa, cada um sabe que terminou o mapeamento pois tem conhecimento da parte total do ambiente que deve mapear.

O mapa da região não precisa ser necessariamente completo. O importante é a detecção dos objetos. Portanto, foi considerado que o robô termina o mapeamento

[2]http://www.mrpt.org



(a) Iteração 54 (b) Iteração 787

(c) Iteração 2138 (d) Iteração 6468

(e) Iteração 7721

Figura 4.2: Instantes da exploração do ambiente com dois robôs. No lado esquerdo de cada imagem, apresenta-se os robôs em ambiente de simulação e à direita o respectivo mapa criado até o momento.

quando 99% da sua subárea foi mapeada. Embora 1% represente $4m^2$, como pode ser visto nas Figuras 4.2e, 4.3e e 4.4e, o espaço não mapeado ficou diluído em pequenas regiões.

Robô	Iterações	Tempo	Distância Percorrida(m)
1	7722	20,09	443,87
2	6469	18,79	412,09
Média	7095,5 $\pm$ 886	19,44 $\pm$ 0,92	427,98 $\pm$ 22,47

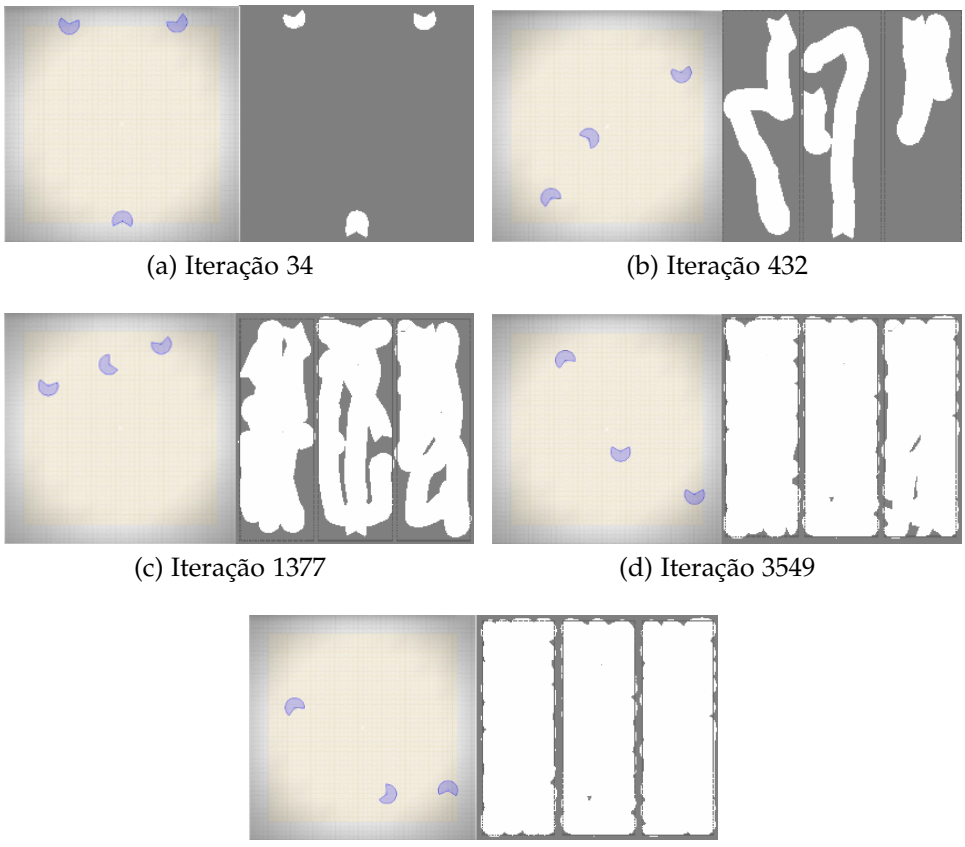
Tabela 4.1: Resultados da exploração do ambiente com dois robôs.

A Figura 4.2 apresenta o processo de mapeamento para dois robôs exploradores. Cada subárea tem dimensão $10m \times 20m$. Conforme apresentado na Tabela 4.1, considerando o maior tempo de para exploração, os robôs levaram cerca de 20 minutos para mapear toda a região.

Robô	Iterações	Tempo	Distância Percorrida(m)
1	5170	18,07	318,59
2	3550	13,71	203,58
3	4738	17,66	282,20
Média	4486 $\pm$ 838,89	16,48 $\pm$ 2,40	268,12 $\pm$ 58,78

Tabela 4.2: Resultados da exploração do ambiente com três robôs.

A região delimitada no mapa refere-se à subárea que o robô deve mapear.



(a) Iteração 34 (b) Iteração 432

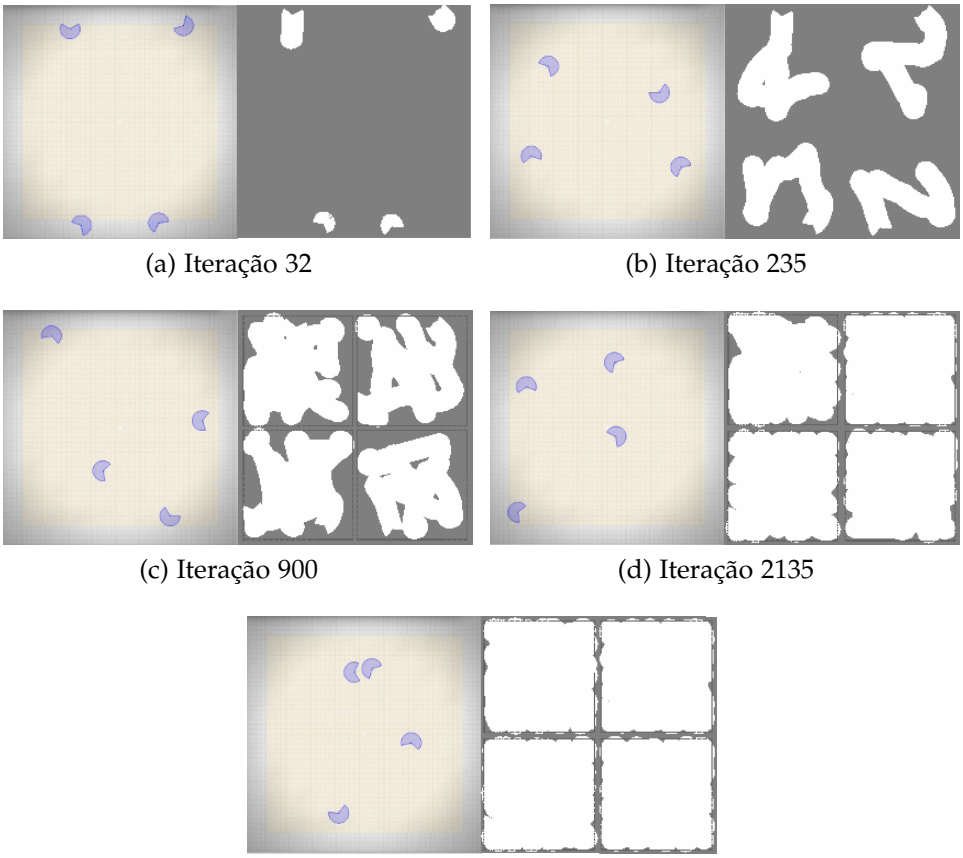
(c) Iteração 1377 (d) Iteração 3549

(e) Iteração 4737

Figura 4.3: Instantes da exploração do ambiente com três robôs. No lado esquerdo de cada imagem, apresenta-se os robôs em ambiente de simulação e à direita o respectivo mapa criado até o momento.

A Figura 4.3 apresenta o processo de mapeamento para três robôs exploradores. Cada subárea tem dimensão $6,67m \times 20,00m$. Conforme apresentado na Tabela 4.2, os robôs levaram cerca de 18 minutos para mapear toda a região.

A Figura 4.4 apresenta o processo de mapeamento para quatro robôs exploradores. Cada subárea tem dimensão $10m \times 10m$. Conforme apresentado na Tabela



(a) Iteração 32 (b) Iteração 235

(c) Iteração 900 (d) Iteração 2135

(e) Iteração 3114

Figura 4.4: Instantes da exploração do ambiente com quatro robôs. No lado esquerdo de cada imagem, apresenta-se os robôs em ambiente de simulação e à direita o respectivo mapa criado até o momento.

Robô	Iterações	Tempo	Distância Percorrida(m)
1	3115	13,71	178,61
2	2136	9,39	118,21
3	3451	13,80	211,58
4	3211	12,84	191,05
Média	2978,25 $\pm$ 579	12,45 $\pm$ 2,08	174,86 $\pm$ 40,14

Tabela 4.3: Resultados da exploração do ambiente com quatro robôs.

4.3, os robôs levaram cerca de 14 minutos para mapear toda a região.

Pelos resultados apresentados, verificou-se que por meio dessa exploração e mapeamento os robôs conseguem abranger quase que totalmente a região determinada. Além disso, a cooperação produz uma redução no tempo total de mapeamento, como era de se esperar (Tabelas 4.1, 4.2 e 4.3). Observa-se que enquanto o tempo simulado diminui, o tempo real aumenta. Isso é devido ao processamento

necessário para simular grupos com mais robôs. Quanto mais robôs no ambiente de simulação, mais lento é o processamento do programa.

Nas imagens do mapa final, nota-se que o robô mapeia pequenas regiões fora da área determinada. Os robôs finalizaram o processo exploração, pois inferiram que a área que ele deveria explorar já estava mapeada. Porém, essas pequenas áreas fora da região influenciaram no cálculo da área mapeada. Logo, pequenas áreas do ambiente não foram mapeadas.

Considerando o tamanho mínimo dos objetos e os espaços não mapeados, de acordo com a posição dos objetos, há a possibilidade de alguns deles não serem identificados. Isso pode ocorrer principalmente nas regiões extremas do ambiente, a maior parte do ambiente foi corretamente mapeada.

O mais importante não é o ambiente ser totalmente mapeado, e sim que os objetos sejam encontrados. Porém, é óbvio que para garantir que todos os objetos sejam encontrados, deve-se explorar ao máximo o ambiente. Mas, sabendo-se o tamanho mínimo dos objetos presentes, pode-se verificar se as áreas não mapeadas são maiores ou menores que o tamanho mínimo deles. Sendo assim, mesmo o ambiente não sendo totalmente mapeado, pode-se garantir que todos os objetos são encontrados.

4.3 Exploração do Objeto

Para a exploração do objeto, os experimentos foram divididos em duas partes. A primeira teve como objetivo a validação do mapeamento 2D e da estimação das informações de dimensão e forma do objeto. No segundo experimento, o foco esteve na estimação do esforço necessário para transportar o objeto.

4.3.1 Mapeamento 2D

O mapeamento do objeto foi realizado tanto em ambiente de simulação como em ambiente real. Os objetos foram variados em relação ao tipo e tamanho. A resolução do mapa do objeto foi de $0,02m$. Para cada objeto, o experimento foi executado 10 vezes.

O processamento de imagem foi realizado utilizando as funções da biblioteca *Open Source Computer Vision* (OpenCV) [3], que contém implementadas várias funções utilizadas nos algoritmos de visão computacional.

[3] http://opencv.willowgarage.com/documentation/

As principais funções utilizadas da biblioteca foram *cvFindContours*, *cvMatchShapes*, *cvBoundingRect*, *cvSmooth*, *cvDilate*, *cvErode*.

4.3.1.1 Experimentos em Ambiente de Simulação

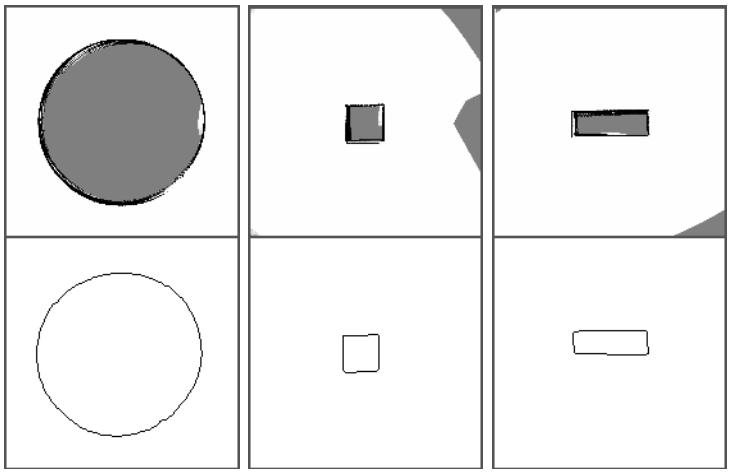
No ambiente de simulação foram considerados nove objetos com características distintas em relação ao tipo e tamanho. A Tabela 4.4 apresenta as características desses objetos. As dimensões do ambiente foram de $5m \times 5m$.

Simulação	Dimensões $(l \times w)$ do objeto(cm)	Tipo
1	10×10	TYPE_CIRCLE
2	30×30	
3	120×120	
4	30×10	TYPE_RECT
5	60×20	
6	120×30	
7	10×10	
8	30×30	
9	120×120	

Tabela 4.4: Características dos objetos considerados para simulação.

A Figura 4.5 apresenta o resultado do mapeamento de três objetos e o processamento final antes do objeto ser classificado em um tipo.

Considerando o maior objeto ($120cm \times 120cm$), inicialmente verificou-se que o número máximo de iterações para realizar o mapeamento dele era de aproxi-



(a) $120 \times 120cm$ (b) $30 \times 30cm$ (c) $60 \times 20cm$

Figura 4.5: Resultado do mapeamento dos objetos e contornos encontrados após o processamento dos mapas.

madamente 200 iterações. Portanto, para o ambiente de simulação, considerou-se $IT_{map} = 500$ (Algoritmo 2 da Seção 3.4.1). Para o maior objeto do ambiente o robô ainda teria a chance de realizar, no mínimo, duas voltas em torno do mesmo para tentar mapeá-lo. Dentro desse tempo estipulado, em apenas um experimento o mapeamento não foi concluído (Tabela 4.5).

Simulação	Acerto da Forma	Tempo Simulado (s)	Tempo Real (s)	Iterações
1	8/10	30,84 $\pm$ 1,73	40,93 $\pm$ 2,80	92 $\pm$ 6
2	10/10	39,19 $\pm$ 1,03	52,92 $\pm$ 2,07	119 $\pm$ 5
3	10/10	60,45 $\pm$ 2,22	84,34 $\pm$ 2,84	192 $\pm$ 7
4	10/10	33,51 $\pm$ 3,91	45,08 $\pm$ 5,22	103 $\pm$ 13
5	10/10	42,05 $\pm$ 6,00	57,39 $\pm$ 9,36	131 $\pm$ 24
6	9/9	65,35 $\pm$ 27,03	98,79 $\pm$ 41,14	207 $\pm$ 88
7	10/10	33,50 $\pm$ 0,98	43,76 $\pm$ 1,54	99 $\pm$ 3
8	10/10	35,74 $\pm$ 2,26	48,42 $\pm$ 2,67	110 $\pm$ 6
9	10/10	65,80 $\pm$ 1,03	90,98 $\pm$ 2,49	205 $\pm$ 7

Tabela 4.5: Resultado do mapeamento dos objetos em ambiente de simulação. Para cada objeto foram realizadas 10 execuções.

Durante os primeiros testes do mapeamento, constatou-se que só a verificação do mapa atual do objeto não era suficiente para determinar se o contorno havia fechado. Devido à atualização constante do método de Grade de Ocupação, as novas leituras do *laser* podem alterar para livres as células que anteriormente foram mapeadas como ocupadas.(Figuras 4.6a e 4.6b). Portanto, fez-se necessário, verificar durante o período de mapeamento, a sobreposição dos mapas processados em cada iteração (Figura 4.6c).

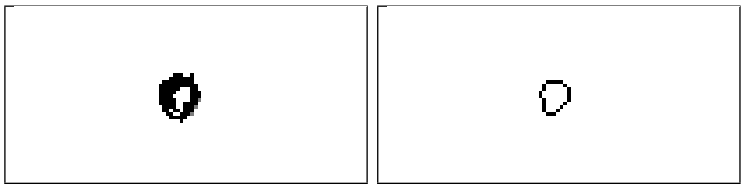
(a) Iteração 50 (b) Iteração 70 (c) Sobreposição

Figura 4.6: (4.6b) A atualização do *laser* pode apagar (4.6a) regiões anteriormente mapeadas. (4.6c) A sobreposição dos mapas evita que isso ocorra.

Analisando o mapa sobreposto, o mapeamento foi realizado com o robô executando apenas uma volta em torno do objeto na maioria dos experimentos. Porém mesmo com essa análise, nem todos os experimentos foram concluídos, como

foi o caso do objeto 6. O mapeamento dos objetos foi finalizado em 98% do casos. Os objetos foram classificados corretamente em 97% dos casos. Os outros 3% correspondem aos objetos do tipo `TYPE_CIRCLE` que foram mapeados como tipo `TYPE_RECT`.

Esse erro é proveniente do próprio mapa do objeto, que apresenta um erro de precisão, devido à resolução. A sobreposição dos mapas também influencia tanto na estimação da forma, quanto nas dimensões do objeto. Por exemplo, na Figura 4.7, após o processamento, o contorno do objeto foi considerado do `TYPE_RECT`, sendo do tipo `TYPE_CIRCLE`.



(a) Contorno sobreposto (b) Afinamento do Contorno

Figura 4.7: Sobreposição e afinamento do contorno.

A Figura 4.8, apresenta o erro médio das dimensões estimadas. O erro apresentado é considerável. Embora pode-se ver que os objetos maiores tiveram um erro maior, para o processo de transporte não tem grande impacto.

A maior influência do erro é no processo de decisão entre o tipo de transporte e a estimação do número mínimo de robôs para o transporte. No caso, os objetos

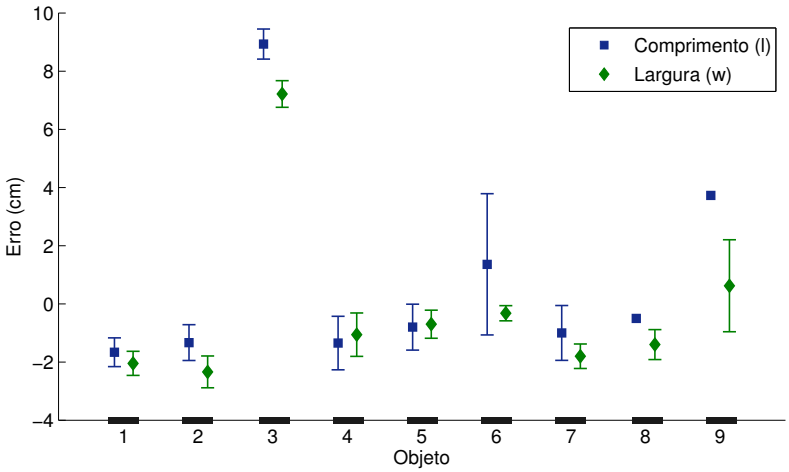
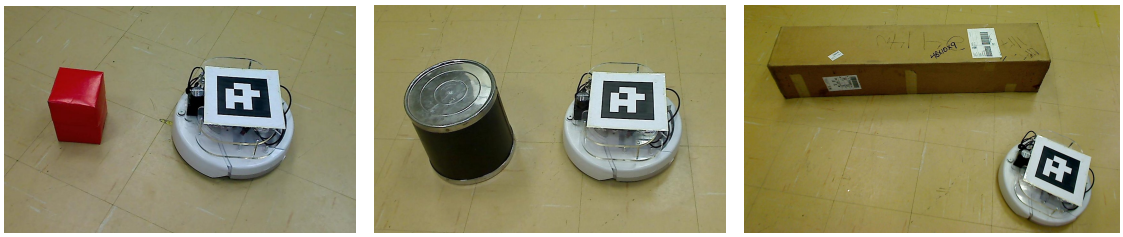


Figura 4.8: Erro médio das dimensões $(l \times w)$ estimadas no processo de mapeamento dos objetos em ambiente simulado.

que apresentarem tamanho próximo do valor limiar de decisão poderiam acarretar em alguma inferência errada.

4.3.1.2 Experimentos com Robôs Reais

No ambiente real foram considerados três objetos com características distintas em relação ao tipo e tamanho. A Figura 4.9 apresenta os objetos inseridos no ambiente real e a Tabela 4.6 apresenta as características desses objetos. As dimensões do ambiente foram de $2,5m \times 2,5m$.



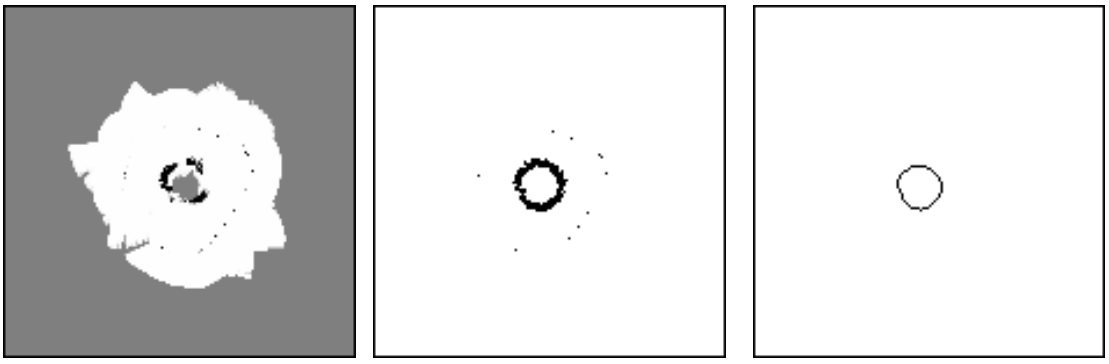
(a) Experimento 1 (b) Experimento 2 (c) Experimento 3

Figura 4.9: Objetos mapeados em ambiente real.

Experimento	Dimensões $(l \times w)$ do objeto(cm)	Tipo
1	12×12	`TYPE_RECT`
2	24×24	`TYPE_CIRCLE`
3	122×25	`TYPE_RECT`

Tabela 4.6: Dimensão e tipos dos objetos considerados para o experimento real

A Figura 4.10 apresenta os mapas obtidos ao final do processo de mapeamento.



(a) Mapa atual (b) Mapa sobreposto (c) Afinamento do contorno

Figura 4.10: Mapas obtidos no processo de estimação da dimensão e tipo do objeto.

Da mesma forma que na simulação foi considerado o tempo máximo de iteração igual a 500. Conforme apresentado na Tabela 4.7, nesse tempo estipulado todos os objetos foram mapeados.

Experimento	Acerto da Forma	Tempo (s)	Iterações
1	10/10	57,84 $\pm$ 12,65	184 $\pm$ 39
2	8/10	44,79 $\pm$ 7,49	143 $\pm$ 26
3	10/10	104,50 $\pm$ 13,17	347$\pm$ 45

Tabela 4.7: Resultado do mapeamento dos objetos em ambiente real.

No caso do ambiente real, devido ao ruído do *laser*, a imprecisão do mapa é maior. Os objetos foram classificados corretamente em 93% dos casos.

Na identificação da forma do objeto, a dificuldade foi encontrada para objetos de dimensões menores. Como foi o caso do objeto 2 que era do `TYPE_CIRCLE` e foi mapeado como `TYPE_RECT` no dois casos de erro.

O erro médio da estimação das dimensões é apresentado na Figura 4.11. Conclui-se que o maior problema na estimação da dimensão foi devido à sobreposição dos mapas. Em alguns casos, o contorno ficou muito espesso, levando à uma estimação das dimensões bem maior do que realmente são. Sabendo-se que esse erro sempre acontece, pode-se corrigi-lo para as próximas estimações, por meio dos valores encontrados neste experimento.

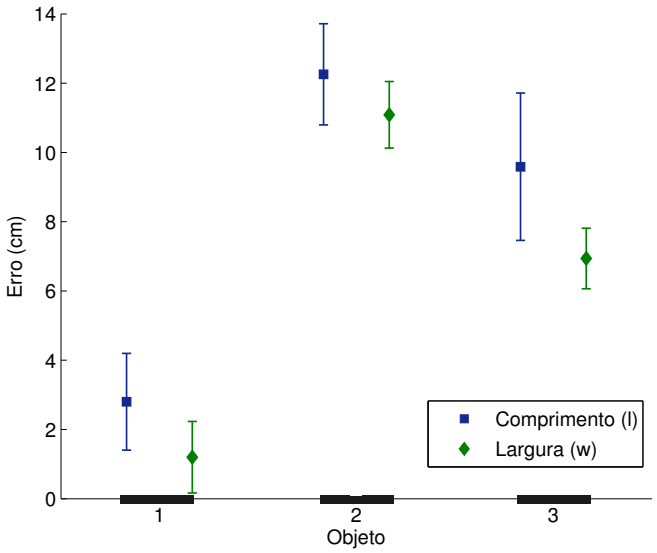


Figura 4.11: Erro médio das dimensões $(l \times w)$ estimadas no processo de mapeamento dos objetos em ambiente real.

4.3.2 Estimação do Esforço

Os experimentos de estimação do esforço foram realizados em ambiente real para as duas situações possíveis: estimação pela potência elétrica no motor do robô *iRobot Create* e pelo torque nos servos motor do manipulador *Crust Crawler AX-12 SmartArm*.

4.3.2.1 Transporte do tipo `PUSH`

Nos experimentos da potência elétrica, procurou-se verificar empiricamente a relação entre a potência, peso do objeto e velocidade do robô. Essa relação pode ser verificada nas Figuras 4.12, 4.13 e 4.15.

A leitura da potência elétrica aplicada ao motor foi realizada por meio da interface *Power* do *Player*.

A relação entre a potência e a velocidade foi analisada por meio do seguinte experimento. O robô navegou livremente por uma área do VeRLab e a cada iteração foi realizada a leitura da potência elétrica atual do motor. A cada intervalo de 5 iterações a velocidade do robô foi aumentada em $0,05m/s$. Variou-se a velocidade do robô entre $0,05m/s$ até $0,50m/s$. Esse procedimento foi realizado 10 vezes.

No gráfico da Figura 4.12 é apresentado o comportamento da potência elétrica no motor do robô *iRobot Create* de acordo com a velocidade, em que pode ser verificado que $P \propto v$.

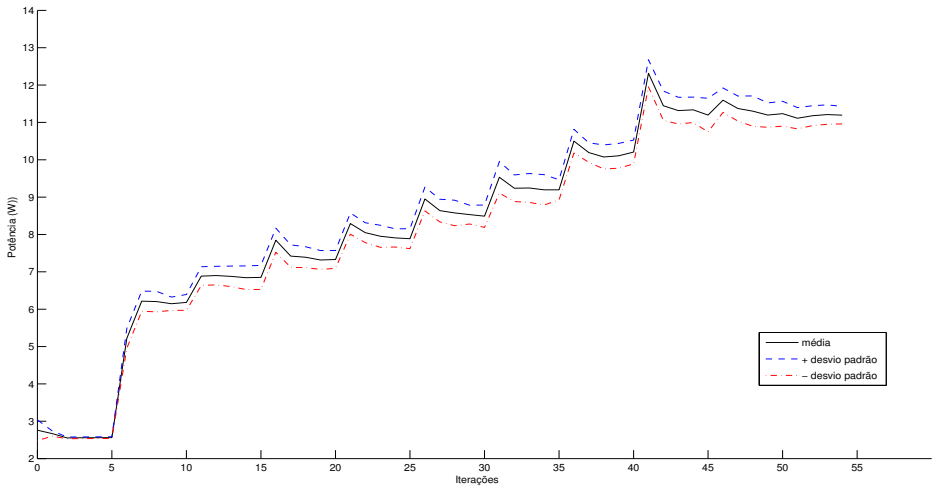


Figura 4.12: Medições da potência elétrica no motor do robô *iRobot Create*. A cada intervalo de 5 iterações a velocidade do robô foi acrescida de $0.05m/s$ com variação de $0,0m/s$ à $0,5m/s$. Média e desvio padrão de 10 leituras.

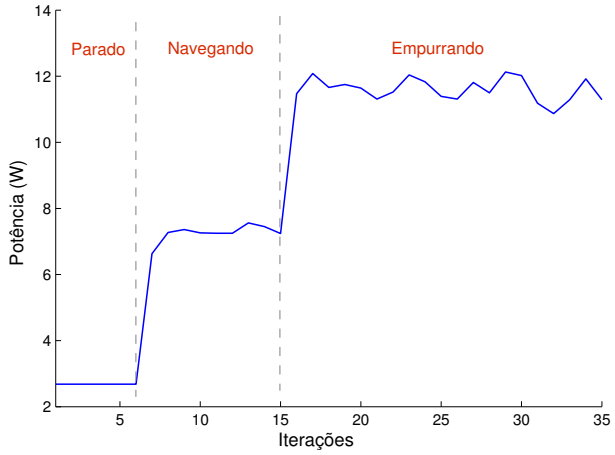


Figura 4.13: Potência elétrica do motor do robô *iRobot Create* conforme os estados: parado, navegando e empurrando o objeto. O robô se move à velocidade constante de $0,10m/s$ e o peso do objeto é $2Kg$.

Outra relação que pode-se verificar é que quando o robô está empurrando um objeto, o seu gasto de potência é maior que quando está apenas navegando livremente por um ambiente. Na Figura 4.13 é apresentado o perfil da potência elétrica no motor do robô quando o mesmo se encontra parado, navegando à uma velocidade de $0,10m/s$ e empurrando um objeto de aproximadamente $2Kg$.

Uma vez que a potência é proporcional à velocidade e ao peso do objeto, o experimento de estimação de esforço foi realizado em relação a essas duas variáveis. Conforme descrito na Subseção 3.4.5, o robô tenta empurrar um determinado objeto por uma distância de aproximadamente $40cm$ com velocidade constante (Figura 4.14).

(a) 00:00:00 (b) 00:00:03 (c) 00:00:05

Figura 4.14: Processo de estimação do esforço pela potência elétrica no motor: o robô navega pelo ambiente, empurra o objeto por um determinado período enquanto faz a leitura da potência.

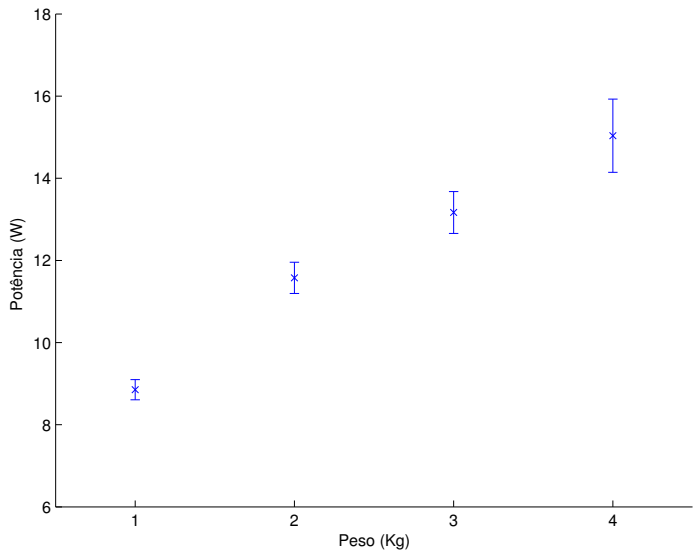


Figura 4.15: Medições da potência elétrica no motor do robô iRobot Create com velocidade constante de $0.10m/s$ e variação do peso do objeto de $1Kg$ à $4Kg$. Média e desvio padrão de 10 leituras.

A distância foi medida utilizando a informação de posição do sistema *Silvver* e dados de odometria. A odometria foi usada para os casos em que o robô não consegue empurrar o objeto e fica parado. Nesse caso, devido à derrapagem das rodas, os dados odométricos informam uma distância percorrida.

Embora a velocidade máxima do *iRobot Create* seja $0,50m/s$, empiricamente constatou-se que para velocidades acima de $0,30m/s$ é difícil controlar o robô e o sistema de localização não captura bem a posição. Portanto, analisou-se a potência do robô para velocidades entre $0,05m/s$ e $0,30m/s$. Para cada velocidade e peso do objeto foram realizadas 10 experimentos. O gráfico da Figura 4.15 apresenta essa variação no motor do robô *iRobot Create* a uma velocidade constante de $0,10m/s$ e a Tabela 4.8 apresenta o resultado de todas as medições realizadas.

Peso	Velocidade (m/s)					
(Kg)	0.05	0.10	0.15	0.20	0.25	0.30
1,00	7,97 ± 0,26	8,85 ± 0,25	10,00 ± 0,33	11,41 ± 0,37	12,24 ± 0,37	13,25 ± 0,33
2,00	9,73 ± 0,38	11,58 ± 0,38	12,94 ± 0,59	13,87 ± 0,34	15,56 ± 0,34	16,38 ± 0,72
3,00	11,95 ± 0,71	13,17 ± 0,51	14,66 ± 0,44	16,42 ± 0,66	16,16 ± 0,66	17,04 ± 1,31
4,00	13,81 ± 0,88	15,04 ± 0,89	16,48 ± 0,93	18,54 ± 0,95	19,20 ± 0,95	19,12 ± 1,50

Tabela 4.8: Média e desvio padrão de 10 leitura da potência elétrica no motor do robô *iRobot Create*.

Pelos resultados apresentados na Tabela 4.8 percebe-se que à medida que a velocidade e o peso do objeto aumentam, menor é precisão das leitura da potência. Isso ocorre devido à aproximação do limite de saturação da potência do motor e consequentemente as leituras ficam mais imprecisas. Portanto para a estimação do esforço é melhor que esse seja feito com velocidades menores.

Dos experimentos realizados, verificou-se que o robô apresenta dificuldade para empurrar objetos com mais de $3Kg$. Para se garantir a realização do transporte, considerou-se o peso máximo do objeto de $2.5Kg$ e potência limite (P_{lim}) de $11W$.

4.3.2.2 Transporte do tipo GRASP

Para estimar o esforço com a informação do torque (Subseção 3.4.5.2), utilizou-se objetos de pesos diferentes e para cada objeto, o robô tenta levantá-lo à uma altura de aproximadamente $3cm$ e posteriormente foi realizada a leitura do torque em cada junta por um determinado período. A velocidade dos motores foi de $4rpm$. A Figura 4.16 apresenta o processo de estimação descrito.

(a) 00:00:00 (b) 00:00:03 (c) 00:00:09

Figura 4.16: Processo de estimação do esforço do manipulador: o robô levanta o objeto à certa altura e faz a leitura do torque nas juntas do manipulador.

Os pesos dos objetos utilizados foram de aproximadamente $50g$, $100g$, $300g$ e $500g$. No processo de estimação, foram realizadas 50 leituras do torque de cada servo motor. Para cada peso do objeto, o experimento foi realizado por 10 vezes.

Na Figura 4.17 é apresentado o resultado das medidas realizadas para as Juntas 1 e 2 conforme o peso do objeto. Como descrito na metodologia, o esforço para levantar o objeto foi consideravelmente maior na Junta1.

Os servo motores utilizados para os experimentos retornam a informação de porcentagem atual τ_p do torque máximo. Utilizando essa informação, o esforço de

cada junta foi estimado por meio da média entre a porcentagem de torque dos dois servos (Equação 4.1).

$$\tau_i = \frac{\tau_{p1} + \tau_{p2}}{2}. \tag{4.1}$$

Pelo gráfico apresentado na Figura 4.17, pode-se perceber que é possível diferenciar o tipo de objeto que o robô pode carregar.

Dos experimentos realizados, o robô não conseguiu levantar os objetos com peso a partir de $300g$. Portanto, para os outros experimentos definiu-se um peso limite considerando a média entre o $100g$ e $300g$, no caso $200g$ e o esforço de 60%.

4.4 Recrutamento e Transporte

O foco desses experimentos esteve no recrutamento de robôs e no transporte de objetos nas diversas situações que podem ocorrer no sistema. Para esse fim, o processo de mapeamento do objeto foi considerado como resolvido, portanto, nesses experimentos ao detectar o objeto, o robô já sabe as dimensões, pose e tipo do mesmo.

Conforme Equação 3.21 da Subseção 3.6.2, a escolha do tipo de transporte é feita considerando o limite de abertura da garra T_{lim}. Nas simulações seguintes foi definido $T_{lim} = 15cm$. Para o processo de estimação do número mínimo N_{R_MIN} e máximo N_{R_MAX} de robôs envolvidos no transporte (Subseção 3.4.4), o raio do robô r_r foi considerado de $16,5cm$ e para o transporte cooperativo, a distância mínima entre os robôs $D_{R\_MIN} = 16,5cm$ (Ver Figura 3.12).

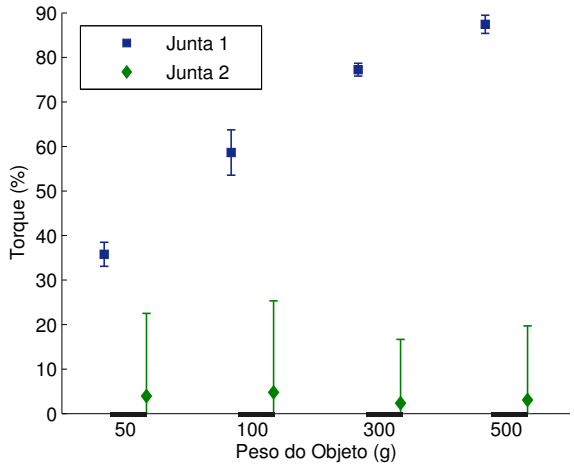


Figura 4.17: Resultado do esforço encontrado conforme o peso do objeto.

4.4.1 Experimentos em Ambiente de Simulação

No ambiente simulado as dimensões da área de exploração foram $5m \times 5m$. Em cada simulação um objeto se encontra na posição $(x, y, \theta) = (0, 0, 0)$ e deve ser transportado para região contrária à área de espera dos robôs (Ver Figura 4.18a). O objetivo é avaliar as situações em que o robô transporta o objeto e/ou recruta outros robôs.

Chama-se atenção para o fato de que no ambiente de simulação a estimação do esforço e o transporte do tipo `GRASP_ALONE` não foram efetivamente realizados devido a limitações encontradas no *software* de simulação utilizado. Porém, para fins de compreensão do processo de transporte, os objetos foram aclopados ao robô para representar o transporte do tipo `GRASP_ALONE` e o esforço foi dado ao robô. Para fins de diferenciação, o robô de cor verde corresponde ao robô do tipo `GRASPER` e os outros correspondem ao robô do tipo `PUSHER`.

A Tabela 4.9 apresenta uma síntese dos experimentos realizados em simulação e apresentados nas respectivas figuras.

Simulação	Recrutamento	Transporte	Tempo Real (*min*)	Tempo de Simulação (*min*)
Figura 4.18	√		3,04	0,28
Figura 4.19		√	4,12	0,33
Figura 4.20		√	4,20	0,39
Figura 4.21			2,28	0,16
Figura 4.22	√	√	4,62	0,47
Figura 4.23	√		7,83	0,69

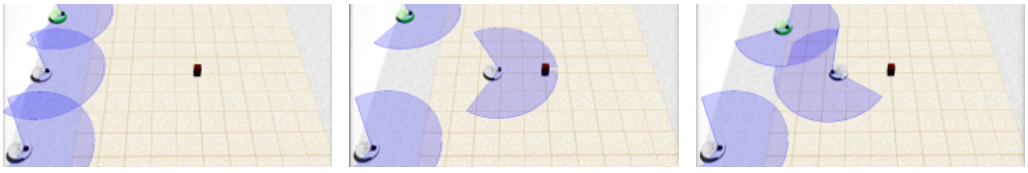
Tabela 4.9: Resultado do processo de recrutamento e transporte.

No primeiro ambiente, representado nas Figuras 4.18 e 4.19, o objeto tem dimensões $12cm \times 12cm \times 20cm$ e é do tipo `TYPE_RECT`. O transporte a ser realizado é do tipo `GRASP_ALONE`.

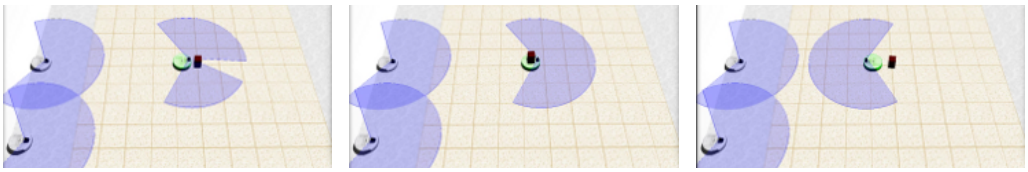
No tipo de transporte `GRASP_ALONE`, o recrutamento só acontece quando um robô do tipo `PUSHER` encontra um objeto de dimensões pequenas. Como representado na Figura 4.18, o robô explorador ao estimar as informações do objeto, verificou que não podia transportá-lo, então realizou o recrutamento. Enquanto o robô recrutado realiza o transporte, o robô recrutador volta ao local de espera e . O robô explorador aguarda uma mensagem do robô recrutado informando o fim do processo de transporte (realizado ou não) para voltar a explorar o ambiente.

Na simulação representada na Figura 4.19, o robô explorador é um robô do tipo `GRASPER`. Ao encontrar o objeto e estimar as informações necessárias para o processo de decisão, o robô verifica que é capaz de executar o tipo de transporte.

Logo, o robô realiza o transporte do objeto até o seu destino.

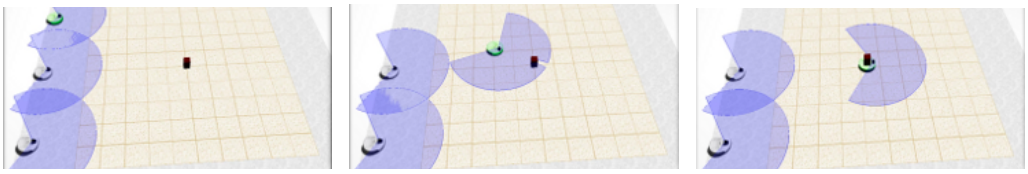


(a) 00:00:00 (b) 00:00:26 (c) 00:00:53



(d) 00:02:26 (e) 00:02:36 (f) 00:03:05

Figura 4.18: Recrutamento e transporte do tipo `GRASP_ALONE`: (4.18b) Ao encontrar um objeto e obter suas informações, o robô verifica o tipo de transporte a ser executado. Como não tem capacidade para executá-lo, (4.18c) recruta um robô do tipo `GRASPER`. (4.18d) O robô recrutado aproxima do objeto e (4.18e) estima o esforço necessário para o transporte. (4.18f) Como o esforço é maior que o limite do robô, o objeto é abandonado.



(a) 00:00:00 (b) 00:00:26 (c) 00:01:52

(d) 00:02:22 (e) 00:02:57 (f) 00:03:53

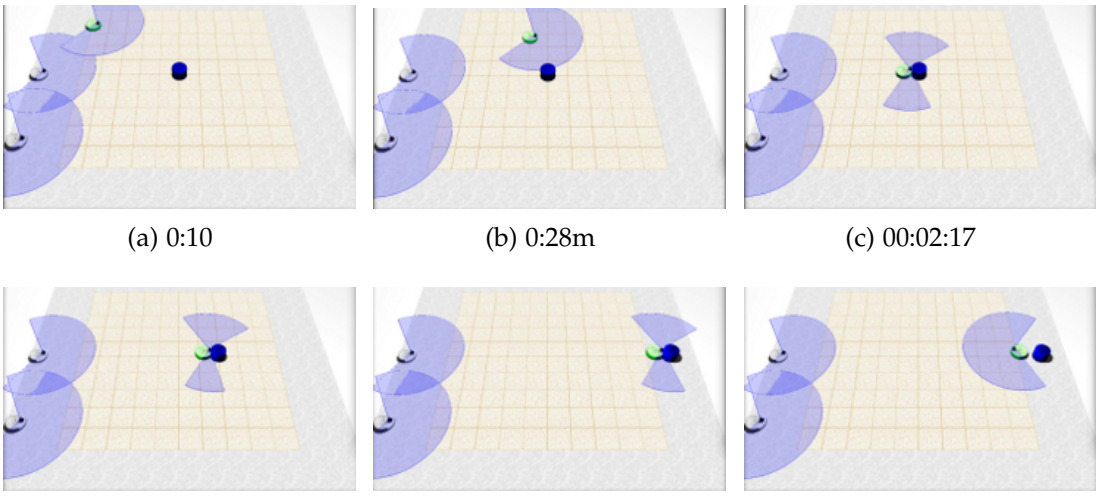
Figura 4.19: Transporte do tipo `GRASP_ALONE`: (4.19b) Ao encontrar um novo objeto e obter suas informações, o robô explorador verifica o tipo de transporte a ser executado. O robô tem condições de executar o tipo de transporte, portanto, (4.19c) ele mesmo estima o esforço necessário. Nesse caso, o robô verificou que o esforço é menor que o limite determinado, portanto, (4.19e) ele realiza o transporte até o destino determinado.

Nas Figuras 4.20 e 4.21, o objeto explorado é do tipo `TYPE_CIRCLE` com dimensões $30cm \times 30cm \times 20cm$. Devido às dimensões do objeto, o tipo de transporte escolhido é o tipo `PUSH`.

Pelas Equações 3.13 e 3.14, o número mínimo (N_{R_MIN}) e máximo (N_{R_MAX}) de robôs para executar o transporte são ambos iguais a um. Portanto, o robô tenta transportar o objeto sozinho (`PUSH_ALONE`).

A primeira simulação desse ambiente é representada na Figura 4.20 em que pode-se verificar um robô do tipo `GRASPER` explorando o ambiente e o objeto. Como o robô do tipo em questão pode realizar os dois tipos de transporte, o mesmo estima o esforço necessário para o transporte e realiza a tarefa.

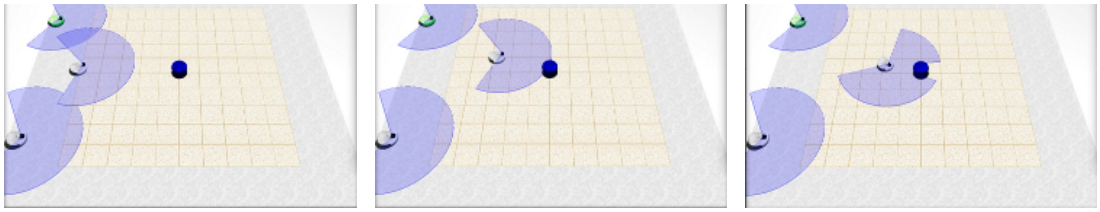
No mesmo cenário que o anterior, outra situação que pode acontecer é quando o esforço necessário para o transporte é maior que a capacidade do robô. Esse é o caso da simulação representada na Figura 4.21. Como o número máximo de robôs para executar esse transporte é igual a um, mesmo havendo robôs para ajudar, não foi possível realizar o transporte desse objeto.



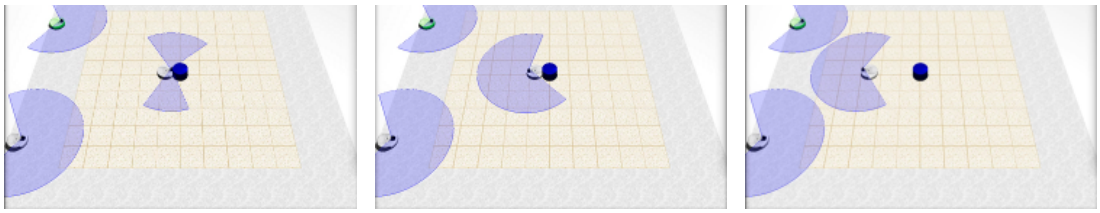
(a) 0:10 (b) 0:28m (c) 00:02:17

(d) 00:02:38 (e) 00:03:27 (f) 00:03:55

Figura 4.20: Transporte do tipo `PUSH_ALONE`: (4.20a) O robô do tipo `GRASPER` explora o ambiente e (4.20b) ao encontrar um objeto, estima sua informações e verifica o tipo de transporte a ser executado. Os robôs do tipo `GRASPER` podem realizar os dois tipos de transporte, logo, (4.20c) ele estima o esforço e (4.20d - 4.20e) realiza o transporte até o destino do objeto.



(a) 00:00:10 (b) 00:00:31 (c) 00:00:51



(d) 00:02:00 (e) 00:02:21 (f) 00:02:46

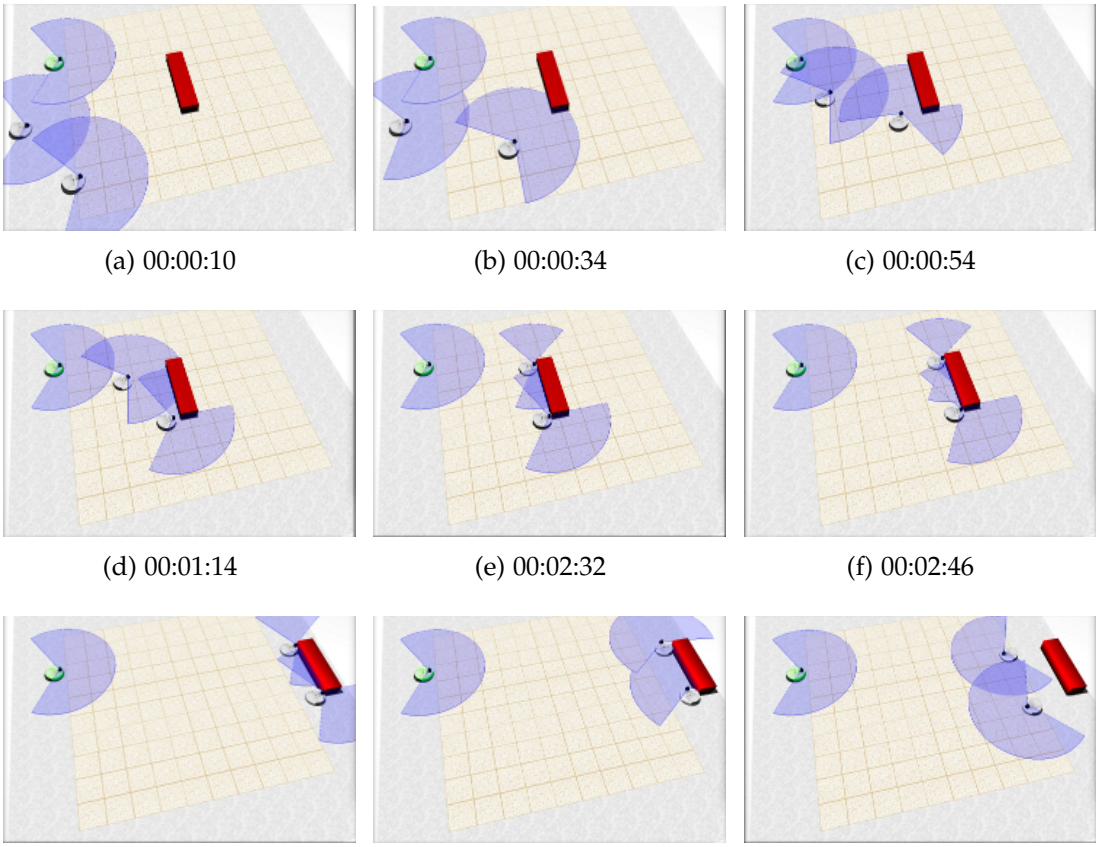
Figura 4.21: Tentativa de transporte do tipo `PUSH_ALONE`: (4.21b) O robô explorador encontra um objeto, estima suas informações e verifica o tipo de transporte a ser executado. O robô tem capacidade para executá-lo, (4.21c) porém, ao estimar o esforço (4.21d) verifica que não pode realizá-lo. (4.21e) O objeto é abandonado.

As Figuras 4.22 e 4.23 apresentam as simulações quando se faz necessária a cooperação entre os robôs para o transporte dos objetos (`PUSH_COOPERATION`). No ambiente tem-se um objeto do tipo `TYPE_RECT` de dimensões $120cm \times 30cm \times 20cm$. O número de robôs para a execução do transporte são $N_{R\_MIN} = 2$ e $N_{R\_MAX} = 3$.

Na Figura 4.22, após estimar as informações do objeto, o robô explorador calcula as possíveis posições que os robôs devem assumir em relação ao objeto. Nesse momento são duas: as posições extremas da direita e esquerda. Uma vez que o robô explorador está mais próximo da posição esquerda (Figura 4.22c), ele solicita ajuda para um robô empurrar na posição extrema direita. Os dois robôs que estão na área de espera enviam o lance para executarem a tarefa. Embora o robô tipo `PUSHER` esteja mais longe da posição de destino que o robô tipo `GRASPER`, o seu lance foi maior devido à sua tendência para esse tipo de transporte (Equação 3.20 da Seção 3.5).

Outra situação que pode acontecer nesse cenário é quando se atinge o número máximo de robôs que podem executar o transporte e mesmo assim o time de robôs não consegue executar a tarefa. Como representado na Figura 4.23, o robô explorador recruta dois robôs, um por vez, e depois de estimar o esforço com os três robôs, verifica que não é possível realizar o transporte, pois se atingiu o número máximo de robôs. No caso da simulação não haveria mais robôs para ajudar, porém mesmo

se tivesse não seria possível realizá-lo, pois $N_{R\_MAX} = 3$.



(a) 00:00:10 (b) 00:00:34 (c) 00:00:54

(d) 00:01:14 (e) 00:02:32 (f) 00:02:46

(g) 00:03:28 (h) 00:03:44 (i) 00:04:18

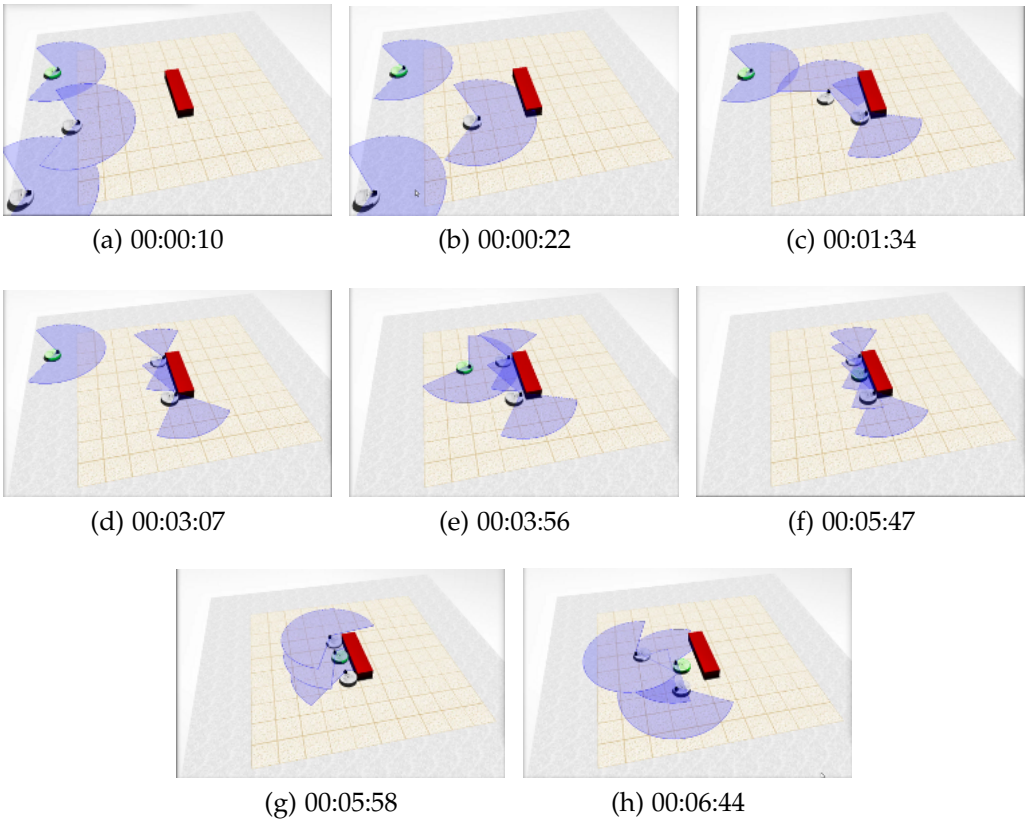
Figura 4.22: Recrutamento e transporte do tipo `PUSH_COOPERATION`: (4.22b) O robô explorador encontra um objeto, estima suas informações e verifica o tipo de transporte a ser executado. O robô tem capacidade para executar o tipo de transporte porém, precisa de ajuda. (4.22c) O robô explorador recruta um ajudante, (4.22d) aguarda a sua chegada para (4.22e) cooperativamente estimarem o esforço necessário para o transporte. (4.22e) Uma vez verificado que conseguem empurrar o objeto, (4.22g) realizam o transporte até o destino do objeto.

Conforme apresentado na Tabela 4.9, em relação ao recrutamento e transporte foi demonstrado que para haver o transporte nem sempre é necessário o recrutamento e em algumas situações mesmo recrutando-se mais robôs o transporte não foi realizado.

A distância percorrida no transporte do objeto foi de aproximadamente $2,5m$. Os tempos apresentados correspondem ao momento de detecção do objeto até o fim do transporte.

O tempo de execução da tarefa é influenciado pelos fatores: disponibilidade dos robôs, distância dos robôs até o objeto e quantidade de robôs recrutados.

Os processos de recrutamento e transporte adotados mostraram-se eficazes porém, podem ser um pouco demorados conforme o número de robôs necessários para o transporte.



(a) 00:00:10 (b) 00:00:22 (c) 00:01:34

(d) 00:03:07 (e) 00:03:56 (f) 00:05:47

(g) 00:05:58 (h) 00:06:44

Figura 4.23: Recrutamento para o transporte do tipo`PUSH_COOPERATION`: (4.23b) O robô explorador encontra um objeto, estima suas informações e verifica o tipo de transporte a ser executado. O tem capacidade para executar o tipo de transporte, porém, precisa de ajuda. (4.23c) Então, ele recruta um ajudante, aguarda a sua chegada para (4.23d) cooperativamente estimarem o esforço necessário para o transporte. Como o esforço é maior que o limite determinado, (4.23e) o robô solicita ajuda de mais um robô e aguarda sua chegada. (4.23f) Ao estimarem o esforço verificam que mesmo com mais um robô ajudando não podem realizar o transporte. Como se atingiu o número máximo de robôs para ajudar, (4.23g) o objeto é abandonado.

A coordenação foi realizada por meio de troca de mensagens. O problema desta solução é a possível sobrecarga na rede. Um robô pode demorar a receber uma mensagem, atrasando o processo de transporte ou se o tempo de espera for muito grande, pode desistir da tarefa.

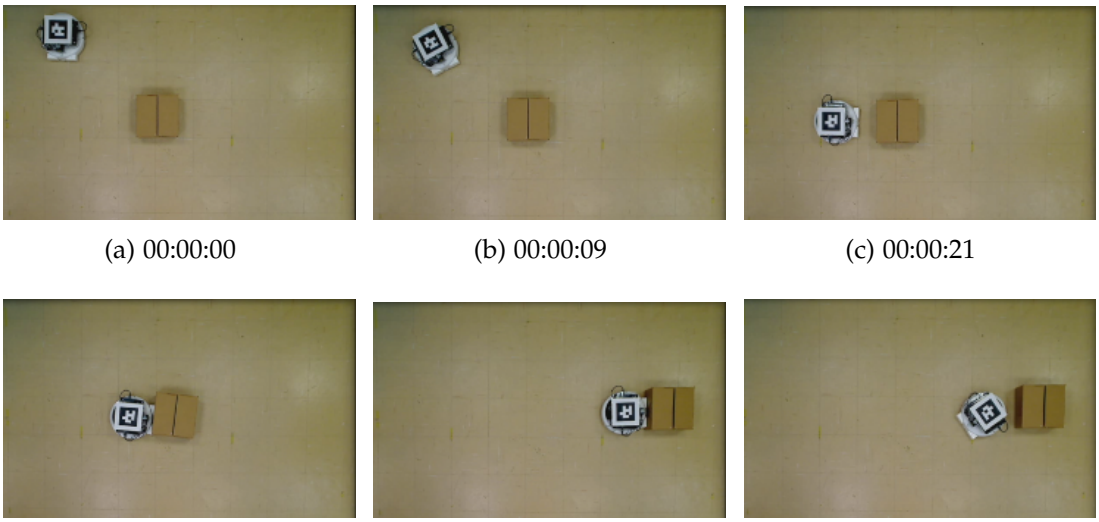
4.4.2 Experimentos com Robôs Reais

Para completar o processo de exploração do objeto, nesses experimentos procurou-se validar juntamente com o recrutamento e o transporte, a estimação do esforço do robô.

O experimento foi realizado no VeRLab em uma área de aproximadamente $2,5m \times 2,5m$. A posição inicial do objeto foi obtida pelo sistema de localização. A partir do início do transporte, a posição do objeto foi estimada pelo robô (Subseção 3.6.1.1). Nos casos em que o transporte foi realizado, a distância percorrida foi de aproximadamente $1m$.

Os comandos necessários para controlar o manipulador foram enviados por meio da biblioteca *libax12arm* desenvolvida no VeRLab [Verlab, 2010].

Na primeira configuração do ambiente (Figuras 4.24 e 4.25), o objeto tem dimensões $32cm \times 32cm \times 20cm$ e é do tipo `TYPE_RECT`. O tipo de transporte é `PUSH_ALONE` e a velocidade para estimação do esforço foi $0,05m/s$. Considerando as medições apresentadas na Subseção 4.3.2 a potência limite para o transporte foi $11W$.



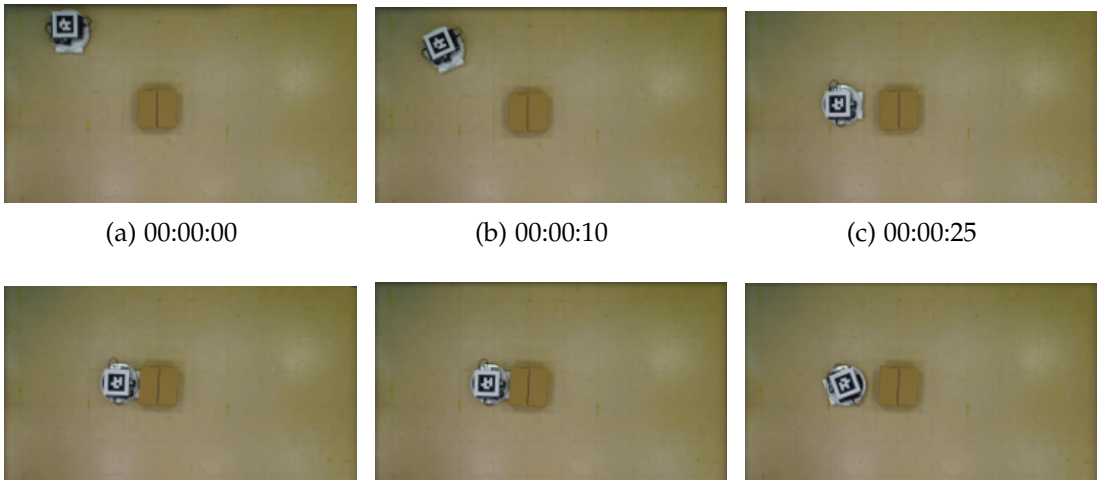
(a) 00:00:00 (b) 00:00:09 (c) 00:00:21

(d) 00:00:29 (e) 00:00:39 (f) 00:01:00

Figura 4.24: Objeto de peso igual a $2kg$ sendo transportado via `PUSH_ALONE`: (4.24a) o robô do tipo `PUSHER` é o robô explorador e (4.24b) ao encontrar um objeto, estima suas informações de dimensão e forma e verificar o tipo de transporte a ser executado. (4.24c) O robô aproxima do objeto para estimar o esforço (4.24d) e uma vez que é capaz de transportar objeto, (4.24e) realiza o transporte até o destino.

Na Figura 4.24, o objeto tem peso aproximado $2Kg$. A potência média estimada foi de $9,19W$. Como a potência é menor que o limite determinado, o robô transporta o objeto até o destino.

Na Figura 4.25, o objeto tem peso aproximado de $7Kg$. A potência média calculada foi de $14,80W$. Como a potência é maior que o limite determinado, o robô precisa de ajuda. Porém, o número máximo de robôs N_{R_MAX} para executar o transporte é 1, portanto o transporte não foi realizado.



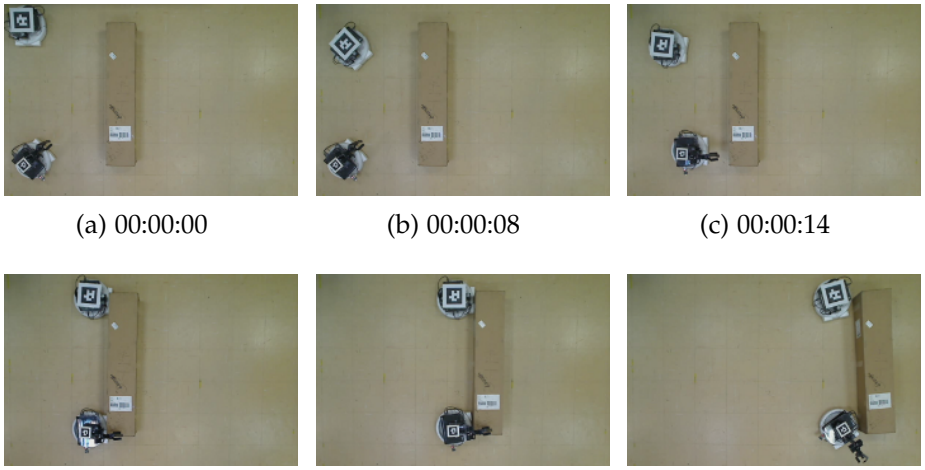
(a) 00:00:00 (b) 00:00:10 (c) 00:00:25

(d) 00:00:27 (e) 00:00:29 (f) 00:00:51

Figura 4.25: Objeto de peso igual a $7kg$. Tentativa de transporte via `PUSH_ALONE`: (4.25a) o robô explorador é do tipo `PUSHER`. (4.25b) Ao encontrar um objeto, estima suas informações de dimensão e forma e verificar o tipo de transporte a ser executado. (4.25c) O robô aproxima do objeto (4.25d) para estimar o esforço. (4.25e) Como o esforço é maior que o limite e o robô não pode solicitar ajuda, o objeto é abandonado.

Nos experimentos referentes às Figuras 4.26 e 4.27, procurou-se explorar a estimação do esforço cooperativamente. O objeto tem dimensões $122cm \times 25cm \times 20cm$ e é do tipo `TYPE_RECT`.

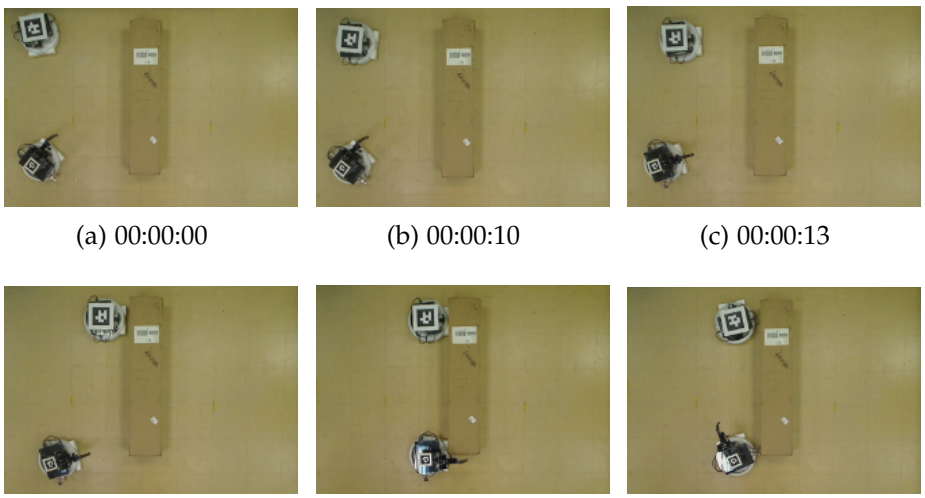
Na Figura 4.26 o objeto tem peso aproximado de $4Kg$. Devido às suas dimensões, o número mínimo de robôs N_{R_MIN} para o transporte é igual a 2. Ao estimar o esforço necessário para o transporte, o robô recrutador obteve como potência média o valor de $7,38W$ e o robô recrutado obteve média de $6,83W$. Sendo assim o transporte foi realizado cooperativamente.



(a) 00:00:00 (b) 00:00:08 (c) 00:00:14

(d) 00:00:26 (e) 00:00:31 (f) 00:00:45

Figura 4.26: Objeto de peso igual a $4kg$. Transporte via `PUSH_COOPERATION`: (4.26a) Robô explorador é do tipo `PUSHER`. (4.26b) Ao encontrar um objeto e estimar suas informações, (4.26c) o robô solicita ajuda, (4.26d - 4.26e) estima o esforço cooperativamente e (4.26f) realiza o transporte.



(a) 00:00:00 (b) 00:00:10 (c) 00:00:13

(d) 00:00:21 (e) 00:00:42 (f) 00:00:55

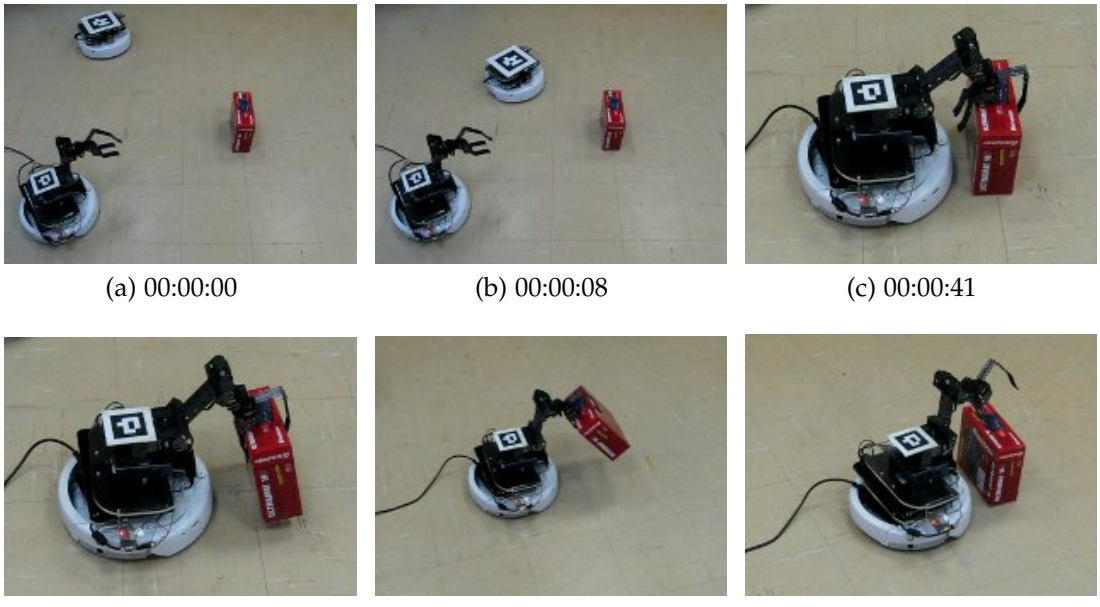
Figura 4.27: Objeto de peso igual a $21kg$. Transporte via `PUSH_COOPERATION`: (4.27a) Robô explorador é do tipo `PUSHER`. (4.27b) Ao encontrar um objeto e estimar suas informações, (4.27c) o robô solicita ajuda. (4.27d - 4.27e) Os robôs estimam o esforço cooperativamente. (4.27f) No caso o objeto é muito pesado e o transporte não é realizado.

Na Figura 4.27 o objeto tem peso aproximado de $21Kg$. Ao estimar o esforço, o robô recrutador obteve como potência média o valor de $11,23W$ e o robô recrutado obteve média de $11,83W$. Nesse caso o transporte não foi realizado.

No processo de estimação do esforço cooperativo, observou-se a estimação dos robôs pode variar muito. Os dados da potência são bem ruidosos e têm uma variação de acordo com o robô. Esta variação não foi analisada, porém, percebeu-se que isso ocorre. Um robô pode iniciar o transporte sozinho, por estimar um esforço menor que o limite enquanto o outro estima um esforço maior e não transporta.

Para o processo de estimação do esforço utilizando o torque, na configuração do ambiente (Figuras 4.28 e 4.29), o objeto tem dimensões $20cm \times 10cm \times 20cm$ e é do tipo `TYPE_RECT`. Considerando as medições apresentadas na Subseção 4.3.2.2, a porcentagem limite do torque é 60%.

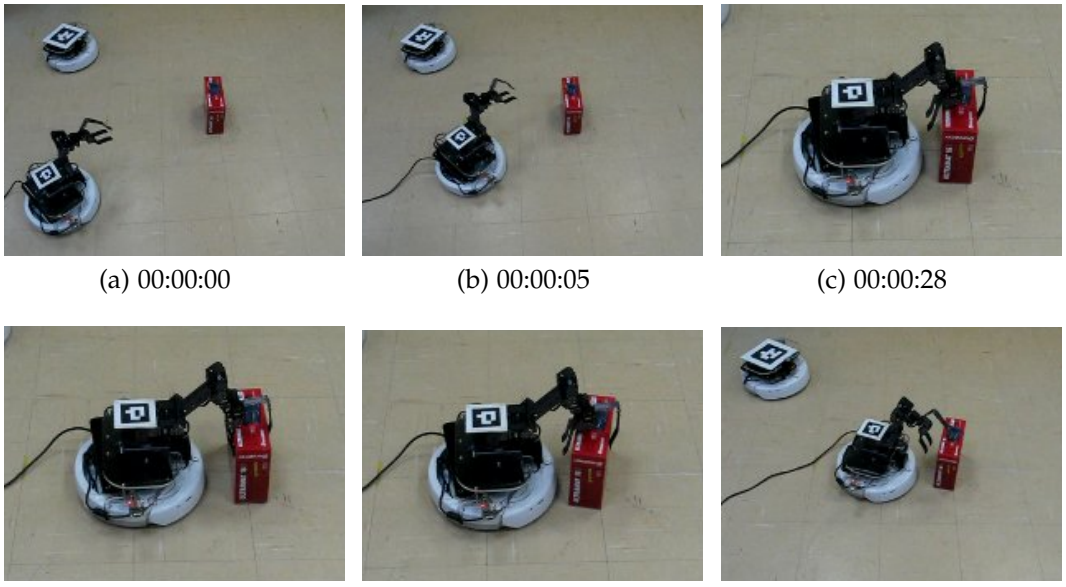
No experimento apresentado na Figura 4.28, o robô estimou um esforço de $41,61\%$ para um objeto de $50g$, portanto o objeto foi transportado. No experimento apresentado na Figura 4.29, o robô tentou levantar um objeto de $300g$ e estimou um esforço de $71,15\%$ e o objeto não foi transportado.



(a) 00:00:00 (b) 00:00:08 (c) 00:00:41

(d) 00:00:45 (e) 00:00:53 (f) 00:01:15

Figura 4.28: Objeto de peso igual a $50g$. Transporte via `GRASP_ALONE`: (4.28a) Robô explorador é do tipo `PUSHER`. (4.28b) Ao encontrar o objeto e estimar suas informações, (4.28c) recruta o robô do tipo `GRASPER` (4.28d - 4.28e) que estima o esforço necessário para realizar o transporte e (4.28f) realizar a tarefa.



(a) 00:00:00 (b) 00:00:05 (c) 00:00:28

(d) 00:00:33 (e) 00:00:41 (f) 00:00:57

Figura 4.29: Objeto de peso igual a 300g. Tentativa de transporte via `GRASP_ALONE`: (4.29a) Robô explorador é do tipo `GRASPER`. (4.29b) Ao encontrar o objeto e estimar suas informações, (4.29c - 4.29e) estima ainda o esforço necessário para realizar o transporte, (4.29f) que nesse caso não foi realizado.

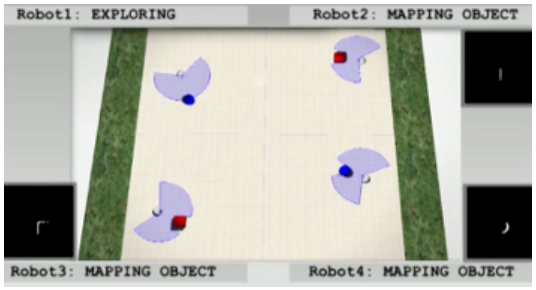
Embora as medições do torque sejam bem ruidosas, foi possível demonstrar que pela metodologia apresentada, o robô pode decidir sobre o transporte do objeto.

4.5 Sistema Integrado

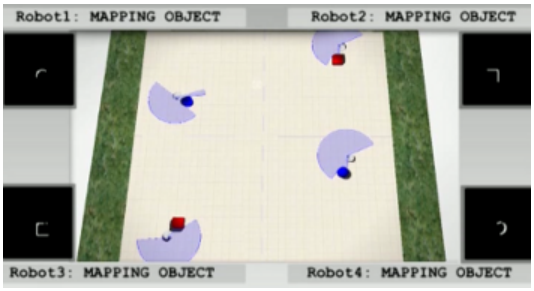
Nesses experimentos, o objetivo foi mostrar o funcionamento do sistema integrando-se todas as partes que o compõe. Para tal foram realizadas duas simulações: uma focando na exploração simultânea de objetos por vários robôs exploradores e outra mostrando o transporte de vários objetos, sendo necessário o recrutamento em algumas situações.

A primeira simulação, representada na Figura 4.30 apresenta quatro robôs exploradores do tipo `PUSHER`, dois objetos do tipo `TYPE_RECT` e dois objetos do tipo `TYPE_CIRCLE`, todos com dimensões de $30cm \times 30cm$. A área de exploração tem dimensões $15m \times 15m$ e foi dividida em quatro regiões, cada uma explorada por um robô específico. Os objetos têm destinos diferentes de acordo com o seu tipo. Nessa simulação não foi considerado o esforço do robô, apenas a capacidade de mapear e transportar o objeto para o destino correto.

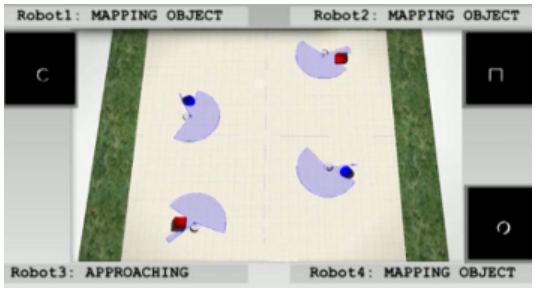
(a) 00:00:10



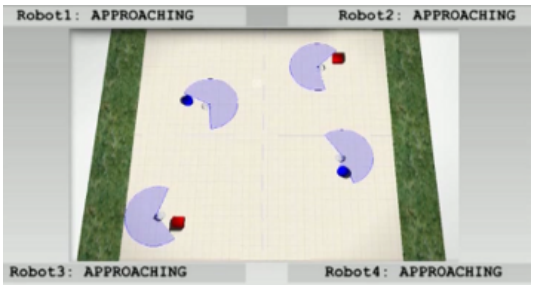
(b) 00:01:00



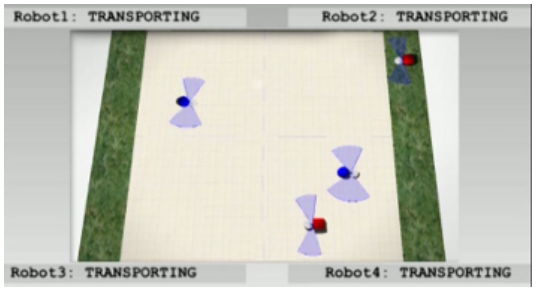
(c) 00:02:30



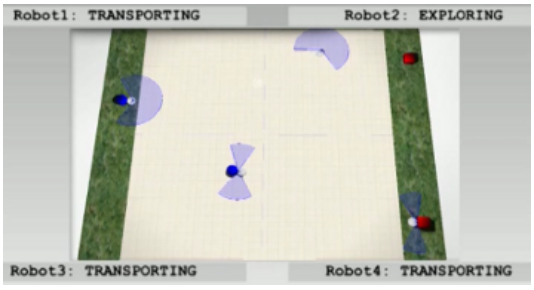
(d) 00:04:45



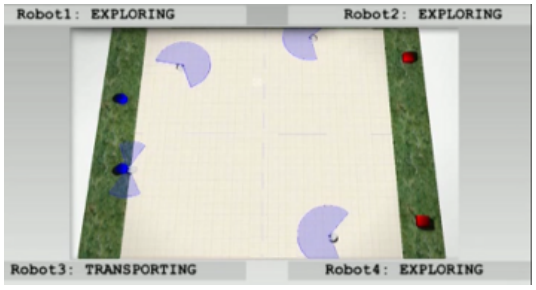
(e) 00:06:04



(f) 00:09:37



(g) 00:11:03



(h) 00:12:50

Figura 4.30: (4.30a) Cada robô explora sua subárea e (4.30b - 4.30c) ao encontrar um novo objeto inicia o processo de mapeamento do mesmo. (4.30d - 4.30e) Os robôs se aproximam do objeto e (4.30f) realizam o transporte até o (4.30g) destino determinado para cada tipo de objeto. (4.30h) A medida que os robôs terminam a tarefa, voltam ao estado de exploração.

Como pode ser visto na Figura 4.30, cada objeto foi mapeado por um robô, e o transporte realizado foi do tipo `PUSH_ALONE`. O transporte dos quatro objetos foi realizado corretamente, mesmo para os objetos com destino opostos à região de mapeamento dos robôs.

A Figura 4.31 apresenta o percentual do tempo que cada robô ficou em um determinado estado considerando o tempo total da simulação. Onde pode ser visto que o processo de mapeamento e transporte são os que demandam mais tempo. O mapeamento é um processo lento, devido à velocidade que deve ser realizado. Nos testes preliminares, observou-se que quando o robô realizou o mapeamento do objeto à uma velocidade baixa, obteve-se um mapa visivelmente melhor. Além da velocidade, o mapeamento será proporcional ao tamanho do objeto explorado. O transporte também depende da velocidade dos robôs, mas não há necessidade de ser baixa. O tempo de realização do transporte dependerá da velocidade de atuação do robô e da distância do objeto ao seu destino.

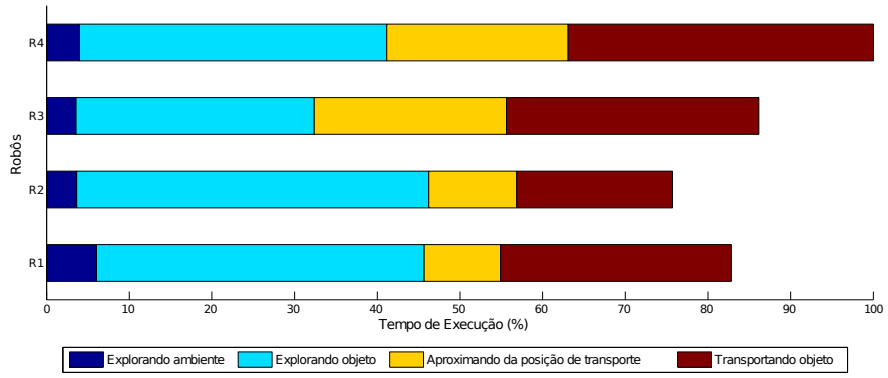


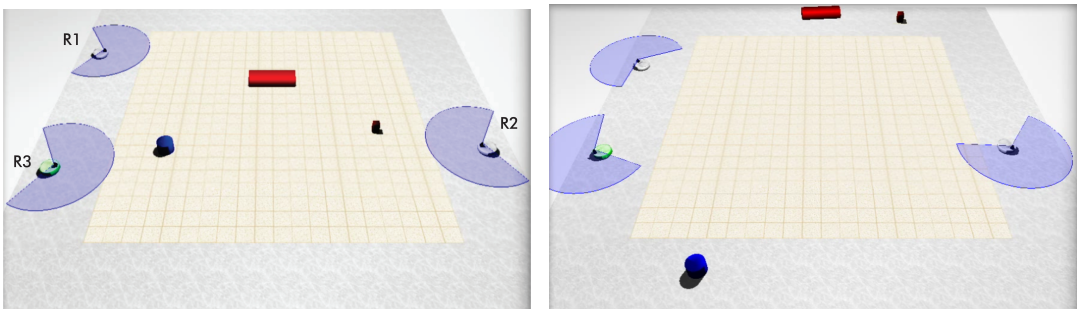
Figura 4.31: Porcentagem do tempo que cada robô permaneceu em um estado.

Na segunda simulação, representada nas Figuras 4.32 e 4.33, tem-se um time composto por três robôs: dois do tipo `PUSHER` e um do tipo `GRASPER`. A área de exploração tem dimensões de $10 \times 10m$ e três objetos para o transporte descritos na Tabela 4.10. Os objetos têm destinos distintos de acordo com o seu tipo. Apenas um robô do tipo `PUSHER` explora o ambiente (Figura 4.32a).

Objeto	Dimensão (cm)	Tipo do objeto	Tipo de transporte
1	12×12	`TYPE_RECT`	`GRASP_ALONE`
2	100×30	`TYPE_RECT`	`PUSH_COOPERATION`
3	30×30	`TYPE_CIRCLE`	`PUSH_ALONE`

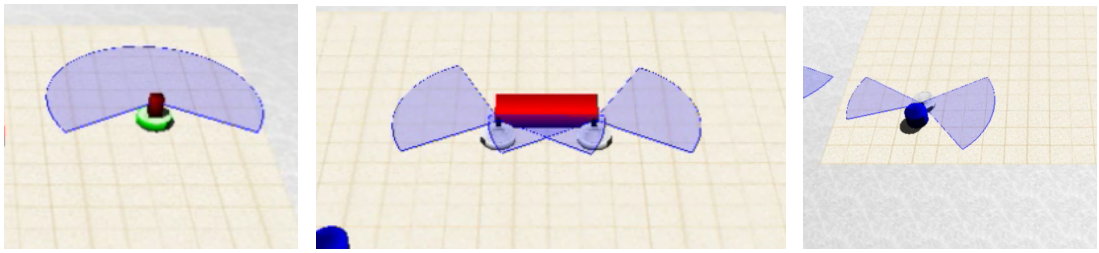
Tabela 4.10: Característica dos objetos presentes no ambiente.

A Figura 4.32b apresenta o instante final da simulação, onde pode ser visto que o transporte foi realizado corretamente de acordo com o tipo do objeto. No processo do transporte o robô explorador recrutou um robô do tipo `GRASPER` para realizar o transporte do objeto 1 (Figura 4.33a) e recrutou um robô para cooperativamente transportarem o objeto 2 (Figura 4.33b). A Figura 4.33 apresenta três instantes nos quais os objetos estavam sendo transportado.



(a) Configuração Inicial (b) Configuração Final

Figura 4.32: Configuração inicial e final do ambiente.



(a) `GRASPER_ALONE` (b) `PUSH_COOPERATION` (c) `PUSH_ALONE`

Figura 4.33: Instantes em que os robôs estavam transportando um objeto.

A Figura 4.34 apresenta a relação do percentual do tempo em que cada robô permaneceu em um estado. Apenas o robô R1, que era o explorador, passou pelos estados de exploração do ambiente e mapeamento do objeto. Consequentemente, os robôs R2 e R3 passaram grande parte do tempo esperando um pedido de ajuda. Como esses robôs dependem do robô explorador para serem alocados em alguma tarefa, eles passam grande parte do tempo ociosos.

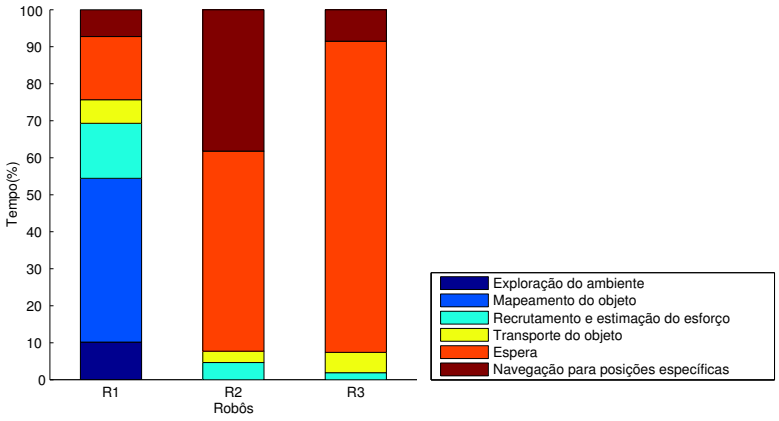


Figura 4.34: Distribuição do tempo que cada robô permaneceu em um estado

4.6 Limitações da Metodologia

A metodologia proposta neste trabalho apresenta algumas restrições que serão discutidas nesta seção. Iniciando-se pelo processo de exploração do ambiente (Seção 4.2), os robôs não compartilham o mapa que é criado. Com isso, o robô explorador deve ser o mesmo durante todo o processo. Essa restrição impede o melhor aproveitamento da capacidade do time.

A divisão da área de exploração proposta permite que a exploração seja executada em paralelo mas, para cada área, deve ser o mesmo robô explorando. Essa divisão poderia ser dinâmica, de acordo com a capacidade atual do time. Os outros robôs ficam no estado de espera enquanto poderiam ajudar na exploração do ambiente e quando fosse necessário, ajudar no transporte.

Quanto ao mapeamento dos objetos (Subseção 4.3.1), o robô só consegue identificar um objeto por vez no mapa. Portanto, deve ser garantida uma distância mínima entre eles suficiente para o *laser* sensorear apenas um objeto.

Em um cenário real os objetos ficam espalhados aleatoriamente no ambiente, sendo difícil garantir essa restrição. Uma proposta de solução seria a identificação do objeto no mapa global de acordo com a posição atual do robô. Outra solução poderia considerar os objetos como entidades inteligentes que poderiam filtrar as leituras do *laser* e atualizar no mapa apenas as leituras que ele considerar suas.

Durante o transporte (Seção 4.4), deve-se garantir que pelo percurso do que o objeto irá passar não há obstáculos à frente. Essa restrição pode ser resolvida com a implementação de um módulo de desvio de obstáculo.

Capítulo 5

Conclusão

Neste trabalho foi apresentada uma metodologia para a realização do transporte de objetos com forma e dimensão distintas dispostos em um ambiente desconhecido. Os trabalhos relatados na literatura referentes ao processo de transporte, assumem que as informações dos objetos são conhecidas à priori.

Na metodologia proposta, o processo de transporte abordou desde a descoberta até o transporte efetivo do objeto. A metodologia envolveu quatro etapas sendo elas: *explorar o ambiente, explorar o objeto, recrutar cooperadores e transportar o objeto*. Para cada etapa descrita foi apresentada uma abordagem que solucionasse o problema em questão.

A exploração do ambiente foi resolvida por meio da divisão territorial. Os experimentos apresentados demostraram que a abordagem proporciona o mapeamento da região de interesse e uma redução no tempo de exploração. A limitação encontrada, discutida na Seção 4.6, foi em relação ao compartilhamento do mapa. Os robôs só têm conhecimento do seu próprio mapa, e como não há o compartilhamento, o robô explorador deve ser o mesmo durante todo o processo. Essa restrição diminui a capacidade de cooperação entre o time, já que qualquer robô poderia assumir o papel de explorador em determinado momento.

Um dos módulos mais importantes no processo de transporte, foi a exploração do objeto. A partir dos resultados obtidos nesse módulo, várias decisões foram tomadas. A exploração do objeto consistiu em um mapeamento $2D$ utilizando um sensor *laser* para a estimação de informações de dimensão e do formato geométrico do objeto, além da estimação do esforço necessário para a realização do transporte. Essa estimação foi realizada por meio da potência elétrica aplicada ao motor do robô e do torque nas juntas do manipulador.

Nos experimentos realizados foi apresentado que o mapeamento proposto na metodologia permite que o robô estime as dimensões e forma do objeto. Porém, apresenta alguns erros devido aos ruídos, inerentes ao ambiente real. Outra razão para os erros encontrados, foi a adição dos mapas dos objetos para a identificação do fechamento do contorno do objeto. Essa adição foi fundamental para a realização da tarefa de mapeamento, porém, interferiu na estimação da dimensão e forma. Como a adição foi realizada com a sobreposição dos mapas, em alguns casos o contorno do objeto ficou muito espesso. Devido à resolução do mapa do objeto, essa espessura trouxe um impacto muito grande na estimação da dimensão.

Quanto à estimação do esforço, demonstrou-se que os robôs conseguem decidir se transportam ou não o objeto conforme as estimativas da potência elétrica e do torque. As medidas são bastante ruidosas, o que dificulta o processo de decisão. Porém, a metodologia se mostrou eficaz uma vez que se tenha sensores mais precisos.

Foi apresentado um método, inspirado em insetos sociais, para recrutamento de robôs quando necessária a ajuda para um determinado transporte. Como o recrutamento é feito de um em um, dependo do tamanho do objeto, esse pode ser um processo muito demorado.

Uma grande deficiência encontrada na solução é o processo de coordenação. Embora a simulação apresentada da Figura 4.30 tenha sido realizada paralelamente, há muitas dificuldades para que o sistema todo efetivamente funcione, tais como distância mínima entre os objetos, interferência da leitura do *laser* no mapeamento dos objetos e percurso do objeto sem obstáculos.

Neste trabalho, a coordenação é feita por meio de troca de mensagens. Foi assumido comunicação ilimitada e sem falhas de comunicação, Porém, em um ambiente real, sempre ocorrem atrasos e isso comprometeria principalmente o processo de estimação de esforço cooperativo.

A metodologia proposta apresentou-se eficaz para o problema, tendo como maior contribuição o processo de exploração do objeto. Por meio das informações estimadas nesse módulo, os robôs puderam decidir qual o tipo de transporte mais adequado para o objeto mapeado e realizá-lo conforme suas capacidades.

Os desafios encontrados para que um sistema como o apresentado tenha menos restrições de execução e seja totalmente autônomo são parte dos problemas encontrados em MRS, tais como sistema totalmente distribuído, localização e mapeamento simultâneo, restrições de comunicação entre os robôs e desvio de obstáculos de grupos de robôs.

Trabalhos futuros podem abordar as limitações discutidas anteriormente e na Seção 4.6. Para a exploração do ambiente, o compartilhamento do mapa entre os robôs resolveria, em partes, o problema de se ter apenas um robô explorador por subárea. Qualquer robô poderia, em um determinado momento, assumir o papel de explorador; assim a ociosidade dos robôs seria reduzida.

A estratégia de exploração poderia ser direcionada de acordo com o destino do objeto. Poderia se incluir uma ação de detecção do caminho do objeto até o destino como: bloqueado ou livre. À medida que os objetos forem mapeados, eles seriam ordenados antes de se começar o transporte. Por meio de uma lista de precedência, o robô saberia quais objetos devem ser transportados primeiro, assim seria garantido o caminho livre.

A tarefa de mapeamento do objeto também poderia ser cooperativa. Para isso, o objeto seria uma entidade inteligente responsável por filtrar as leituras do *laser* dos vários robôs.

Referências Bibliográficas

Alpaydin, E. (2010). *Introduction to Machine Learning*, Decision Trees, pp. 188–192. MIT, 2 edição.

Beekman, M. & Dussutour, A. (2009). *Food Exploitation By Social Insects: Ecological, Behavioral, and Theoretical Approaches*, How to Tell Your Mates: Costs and Benefits of Different Recruitment Mechanisms, pp. 115–134. In Beekman & Dussutour [2009].

Berman, S.; Lindsey, Q.; Sakar, M.; Kumar, V. & Pratt, S. (2011). Experimental study and modeling of group retrieval in ants as an approach to collective transport in swarm robotic systems. *Proceedings of the IEEE*, 99(9):1470 –1481.

Bolger, A.; Faulkner, M.; Stein, D.; White, L.; kook Yun, S. & Rus, D. (2010). Experiments in decentralized robot construction with tool delivery and assembly robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 5085 –5092.

Botelho, S. & Alami, R. (1999). M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pp. 1234–1239.

Bradski, G. & Kaehler, A. (2008). *Learning OpenCV*. O'Reilly Media Inc.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14 – 23.

Chaimowicz, L.; Campos, M. & Kumar, V. (2002). Dynamic role assignment for cooperative robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pp. 293 – 298 vol.1.

Dahl, T.; Mataric, M. & Sukhatme, G. (2002). Adaptive spatio-temporal organization in groups of robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pp. 1044–1049.

Dahl, T. S.; Mataric, M. J. & Sukhatme, G. S. (2009). Multi-robot task allocation through vacancy chain scheduling. *Robotics and Autonomous Systems*, 57(4):674–687.

Dias, M. B.; Zlot, R.; Kalra, N. & Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257 – 1270.

dos Santos, F. & Bazzan, A. L. C. (2009). An ant based algorithm for task allocation in large-scale and dynamic multiagent scenarios. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pp. 73--80, New York, NY, USA. ACM.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57.

Emery, R. & Balch, T. (2001). Behavior-based control of a non-holonomic robot in pushing tasks. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 3, pp. 2381 – 2388 vol.3.

Farinelli, A.; Iocchi, L. & Nardi, D. (2004). Multirobot systems: a classification focused on coordination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2015 –2028.

Ferreira Jr, P. R.; dos Santos, F.; Bazzan, A. L.; Epstein, D. & Waskow, S. J. (2010). Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies. In *International Conference on Autonomous Agents and Multiagent Systems*, volume 20, pp. 421–443.

Fink, J.; Hsieh, M. & Kumar, V. (2008). Multi-robot manipulation via caging in environments with obstacles. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pp. 1471 –1476.

Fink, J.; Michael, N. & Kumar, V. (2007). Composition of vector fields for multi-robot manipulation via caging. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA.

Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Garcia, R. F.; Shiroma, P. M.; Chaimowicz, L. & Campos, M. F. M. (2007). Um arcabouço para a localização de enxames de robôs. In *Simpósio Brasileiro de Automação Inteligente*.

Gerkey, B. & Mataric, M. (2002a). Pusher-watcher: an approach to fault-tolerant tightly-coupled robot coordination. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pp. 464 – 469 vol.1.

Gerkey, B. & Mataric, M. (2002b). Sold!: auction methods for multirobot coordination. *Robotics and Automation, IEEE Transactions on*, 18(5):758 – 768.

Gerkey, B. P. & Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954.

Gerkey, B. P.; Vaughan, R. T. & Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *In Proceedings of the 11th International Conference on Advanced Robotics*, pp. 317--323.

Gonzalez, R. C. & Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Guizzo, E. (2008). Three engineers, hundreds of robots, one warehouse. *Spectrum, IEEE*, 45(7):26 –34.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, volume 2, pp. 500 – 505.

Kube, C. R. & Bonabeau, E. (2000). Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30(1-2):85--101.

Luca, A. D. & Oriolo, G. (1994). Local incremental planning for nonholonomic mobile robots. In *In Proceedings of 1994 International Conference on Robotics and Automation*, pp. 104--110.

Martinson, E. & Arkin, R. (2003). Learning to role-switch in multi-robot systems. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pp. 2727 – 2734 vol.2.

Mataric, M.; Nilsson, M. & Simsarin, K. (1995). Cooperative multi-robot box-pushing. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 3, pp. 556 –561 vol.3.

Parker, J. R. (1996). *Algorithms for Image Processing and Computer Vision*. Wiley.

Parker, L. (1998). Alliance: an architecture for fault tolerant multirobot cooperation. *Robotics and Automation, IEEE Transactions on*, 14(2):220 –240.

Parker, L. & Tang, F. (2006). Building multirobot coalitions through automated task solution synthesis. *Proceedings of the IEEE*, 94(7):1289 –1305.

Parker, L. E. (2008). Distributed intelligence: Overview of the field and its application in multi-robot systems. *Journal of Physical Agents*, 2(1).

Pereira, G.; Pimentel, B.; Chaimowicz, L. & Campos, M. (2002). Coordination of multiple mobile robots in an object carrying task using implicit communication. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pp. 281–286.

Pereira, G. A. S. (2003). *Motion Planning and Control of Cooperating Mobile Robots: A Graph Connectivity Approach*. Tese de doutorado, Universidade Federal de Minas Gerais.

Pereira, G. A. S.; Kumar, V. & Campos, M. F. M. (2003). Decentralized algorithms for multirobot manipulation via caging. *International Journal of Robotics Research*, 23:783–795.

Resnick, R. & Halliday, D. (1983). *Física*. Volume 4 of Resnick & Halliday [1983].

Rus, D.; Donald, B. & Jennings, J. (1995). Moving furniture with teams of autonomous robots. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pp. 235 –242 vol.1.

Schneider-Fontán, M. & Mataric, M. J. (1998). Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14:815–822.

Shiroma, P. M. & Campos, M. F. M. (2009). Comutar: A framework for multi-robot coordination and task allocation. In *International Conference on Intelligent Robots and Systems*.

Siegwart, R. & Nourbakhsh, I. R. (2004). *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA.

Siriwardana, P. G. D. (2009). Machine leaning-based multi-robot cooperative transportation of object. , Mechanical Engineering - University of British Columbia.

Sun, X.; Mao, T.; Kralik, J. & Ray, L. (2009). Cooperative multi-robot reinforcement learning: A framework in hybrid state space. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 1190 –1196.

Thomas, G.; Howard, A.; Williams, A. & Moore-Alston, A. (2005). Multirobot task allocation in lunar mission construction scenarios. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 1, pp. 518 – 523 Vol. 1.

Vahrenkamp, N.; Kuhn, E.; Asfour, T. & Dillmann, R. (2010). Planning multi-robot grasping motions. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pp. 593 –600.

Verlab (2010). Projeto roomba. Disponível em http://www.verlab.dcc.ufmg.br/projetos/roomba/index.

Verret, S. (2005). Current state of the art in multirobot systems. Technical memorandum, Defence R&D Canada â Suffield.

Vig, L. & Adams, J. (2006). Multi-robot coalition formation. *Robotics, IEEE Transactions on*, 22(4):637 –649.

Viguria, A.; Maza, I. & Ollero, A. (2008). S+t: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation. In *ICRA*, pp. 3163–3168. IEEE.

Wang, Y. & de Silva, C. (2006). Multi-robot box-pushing: Single-agent q-learning vs. team q-learning. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pp. 3694 –3699.

Wang, Z.; Hirata, Y. & Kosuge, K. (2005). An algorithm for testing object caging condition by multiple mobile robots. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 3022 – 3027.

Wang, Z. & Kumar, V. (2002). Object closure and manipulation by multiple cooperating mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 1, pp. 394 – 399 vol.1.

Wawerla, J. & Vaughan, R. (2010). A fast and frugal method for team-task allocation in a multi-robot transportation system. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1432 –1437.

Wurm, K.; Stachniss, C. & Burgard, W. (2008). Coordinated multi-robot exploration using a segmentation of the environment. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pp. 1160 –1165.

Zhang, Y. & Gruver, W. (1996). Classification of grasps by multifingered robot hands. In *Intelligent Robots and Systems '96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on*, volume 3, pp. 1052 –1059 vol.3.