2 Embasamento Teórico

Este capítulo apresenta o embasamento teórico para entendimento desta dissertação, e contempla os principais tipos de controles autônomos existentes, arquiteturas usadas no controle baseado em comportamento, e a ferramenta de simulação usada.

2.1. Robôs Móveis Autônomos

Grande parte dos robôs móveis e fixos foram idealizados para trabalhar em ambientes bem diferentes. Os dispositivos fixos tipicamente são especificados para ambientes estruturados, pré-definidos e relativamente estáticos. Já os robôs móveis pretendem atuar em ambientes altamente dinâmicos e desconhecidos.

Ambientes estruturados possuem objetos em posições definidas antes que qualquer processo seja iniciado, objetos irrelevantes para as tarefas são tipicamente excluídos destes ambientes pela possível dificuldade de modelá-los matematicamente de uma forma simplificada. O chão de fábrica de indústrias automobilísticas é altamente estruturado. Salas de escritórios comerciais e residências têm uma definição intermediária da natureza dos objetos incluídos no ambiente, enquanto que florestas e desertos não mapeados, por exemplo, são totalmente aleatórios em termos de pré-definição do ambiente.

Um ambiente dinâmico é um ambiente que sofre alterações constantemente. Os objetos que fazem parte do local podem alterar sua posição e orientação independentemente do robô. Robôs fixos aceitam trabalhar com ambiente com pouca dinâmica como, por exemplo, objetos que variam sua posição e orientação em uma esteira industrial, contanto que o robô seja capaz de identificar essas pequenas variações. Robôs móveis projetados para ambientes dinâmicos não possuem *a priori* o conhecimento da localização e orientação dos objetos e sua variação durante o processo.

Outra grande diferença entre ambos é a auto-localização: robôs fixos sabem exatamente onde estão devido a não sofrerem alteração de sua base instalada e

possuírem *encoders* de posição precisos (Figura 8) em suas juntas, enquanto os móveis raramente sabem, pois mesmo que possuam *encoders*, sofrem com as variações do ambiente e do solo que geram erros significativos (Figura 8).

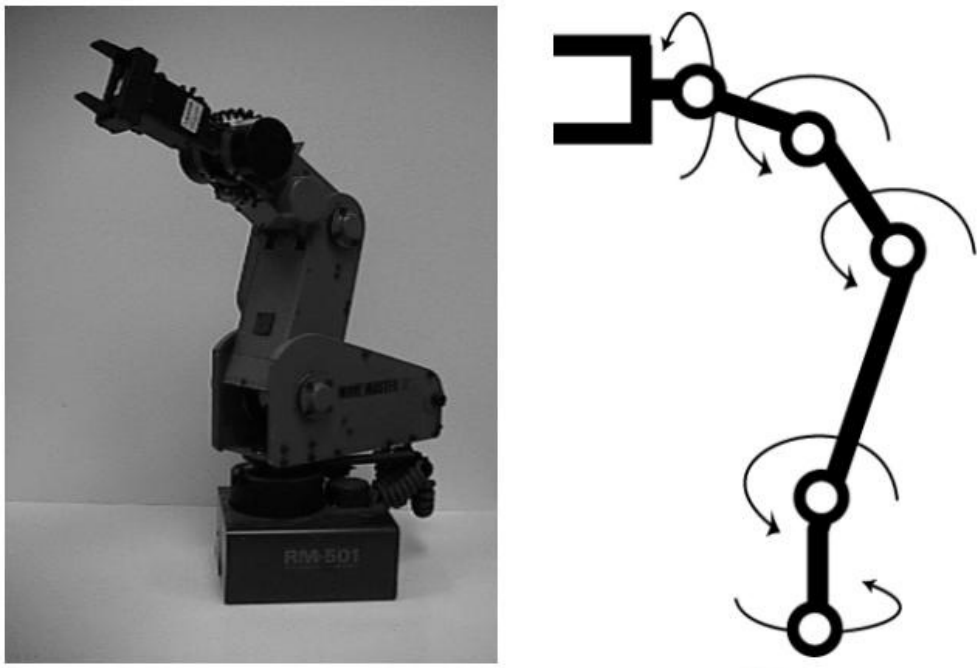

Figura 8 - Robô *MOVEMASTER* a) Robô Fixo b) Juntas associadas (MURPHY, 2000) [26]

O grande desafio que os robôs móveis autônomos trouxeram para a atualidade é dotar estes sistemas de uma capacidade de raciocínio inteligente e de interação com o meio em que estão inseridos, e fazê-los "sentir" e "reagir" ao ambiente em que estão inseridos através da leitura de seus sensores (e.g. sensores infravermelho, lasers, *bumpers*, câmeras de vídeo, etc.). É através desta percepção sensorial que podem executar melhor as suas ações, (MEDEIROS 1998) [24], (HEINEN 1999) [25].

Existem robôs móveis atuando em diferentes áreas, como por exemplo: robôs desarmadores de bombas, robôs usados para a exploração de ambientes hostis, e a condução de veículos (carros) robotizados. (Figura 9).

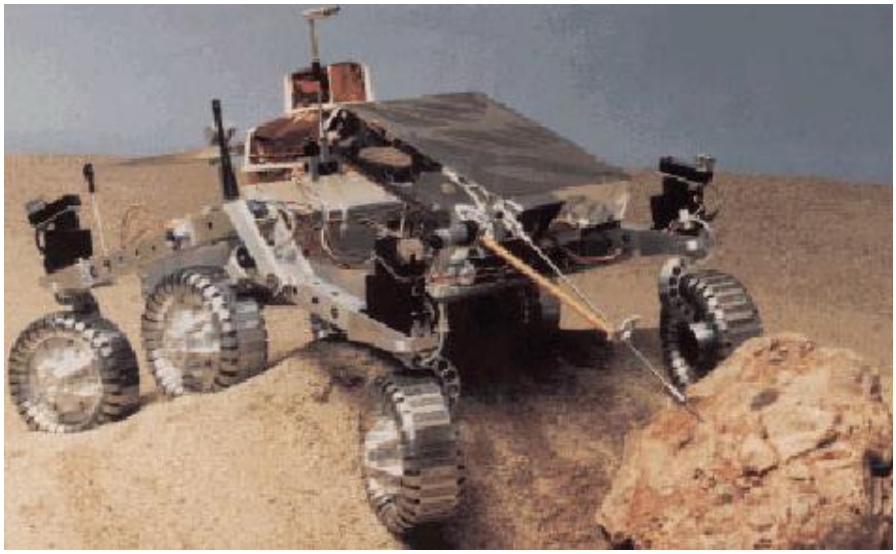

Figura 9 - Robô *Soujorner* - Robô para exploração de ambientes dinâmicos e desconhecidos (MURPHY, 2000) [26].

Exemplos de robôs móveis autônomos que obtiveram destaque internacional são: o sistema desenvolvido pelo NavLab da CMU (POMERLEAU, 1998) [27], (BATAVIA, 1996) [29] capaz de conduzir uma caminhonete pelas estradas americanas; o robô do tipo exploratório enviado para Marte pela NASA (SCHENKER, 2003) [14], o robô Dante que explora vulcões (LEMONICK ,1994) [28] e o sistema de controle de um veículo *Ligier* elétrico desenvolvido pelos pesquisadores do INRIA na França (PAROMTCHIK, 1996) [30], (SCHEUER, 1998) [31]. Os robôs exploratórios enviados para Marte chamam atenção pela alta tecnologia autônoma empregada, em 2003 a NASA enviou os robôs *Spirit* e *Opportunity* (NASA, 2003) [56] e o robô *Phoenix* (NASA, 2007) [56]. Aplicações militares como o BIGDOG (BOSTON D., 2008) [57], robô de carga para terrenos acidentados. Destaque também para o entretenimento pessoal como o robô AIBO da empresa *SONY* (STONE, 2007) [46] que simula um cachorro e suas emoções, e para robôs de limpeza doméstica como o *Roomba 500* da empresa *iRobot* (iROBOT, 2007) [55]. Todos estes sistemas possuem em comum a capacidade de receber leituras de sensores que lhes dão informações sobre o ambiente em que estão inseridos e de modo parcialmente ou completamente autônomo. Geram os comandos de deslocamento para navegação em um ambiente de modo seguro, sem se chocar contra obstáculos ou colocar em risco sua integridade ou a dos elementos no ambiente.

2.2. Sistemas de Controle

Para executar uma tarefa, um robô normalmente deve agregar tipos diferentes de objetivos. Buscar um determinado objeto num ambiente desconhecido é uma tarefa de alto nível que se utiliza de vários objetivos enquanto que uma tarefa do tipo "andar em linha reta com velocidade constante" é denominada uma tarefa de baixo nível, pois utiliza um ou poucos objetivos agregados.

Precisa-se estabelecer quais são os elementos que interagem com o sistema de controle para poder defini-lo. Ligado diretamente ao sistema de controle está o robô, que é o sistema que deseja-se controlar. O ambiente não é ou não pode ser totalmente controlado diretamente pelo sistema de controle, mas interfere de maneira significativa em seu funcionamento.

O sistema de controle tem a tarefa de fazer com que todo o sistema alcance um determinado estado. Alcançar este estado pode envolver ou depender de mudanças que ocorrem no ambiente, no sistema controlado ou devido à interação entre os dois. O sistema de controle trabalha adquirindo informações captadas do ambiente através de sensores, e a partir destas leituras toma decisões, atua e interage com o ambiente através de motores, garras, manipuladores, dentre outros. Logo, um sistema de controle é um processo que pode utilizar seus sensores para obter informações sobre o sistema controlado e sobre o ambiente. Ele pode utilizar este conhecimento para controlar seus atuadores fazendo com que todo o sistema alcance um determinado estado.

A fim de exemplificar a função do sistema de controle, considere o seguinte exemplo. Um robô se locomove dentro de uma arena cercada tentando evitar a colisão com muros. Neste caso, o sistema controlado é o robô, e os muros são parte do ambiente. A interação do robô com o ambiente se dá com o contato das rodas com o terreno ou caso o robô acidentalmente colida com um muro. O sistema de controle obtém informações sobre o próprio robô e sobre o ambiente através dos sensores, as processa, e então utiliza a resposta gerada por este processamento para controlar os atuadores e manter o robô distante dos muros, que seria o estado desejado do sistema.

2.3. Técnicas de Controle

As técnicas de controle mais difundidas e utilizadas por um sistema de controle são denominadas controle em malha aberta (*open loo*p) e malha fechada (*closed loop*). Os sistemas *open loop* não necessitam de sensores. Admite-se por exemplo um sistema de controle que deve fazer um robô se mover a uma velocidade constante. Após os cálculos preliminares com base nos dados físicos do robô, pode-se utilizar um modelo que pode prever quanta energia deve ser fornecida aos motores do robô para que ele atinja a velocidade desejada. Esta técnica não garante que o robô vá manter sua velocidade constante, pois ao se iniciar o processo não há mais como alterar seus parâmetros caso haja alguma modificação no ambiente.

Os sistemas *closed loop* podem ser divididos em sistemas de malha fechada *feedforward* ou *feedback*. Os sistemas que utilizam o f*eedforward* utilizam sensores somente para perceber o ambiente. Neste tipo de técnica de controle, medições do ambiente são utilizadas para atualizar variáveis no modelo do sistema. A partir do exemplo anterior, pode-se adicionar um sensor que informa a inclinação do terreno, pois esta informação pode ser utilizada para recalcular a força necessária para que o robô atinja a velocidade desejada.

As técnicas *open loop* e malha fechada *feedforward* podem ser utilizadas preferencialmente quando o ambiente é praticamente estático e previsível, o que não ocorre na maioria dos casos de controle robótico. Ambientes dinâmicos necessitam de um controle mais fino, e a técnica mais utilizada em sistemas robóticos é de malha fechada *feedback*. Esta monitora continuamente a resposta dos sensores e ajusta seus atuadores de acordo com a necessidade programada. O exemplo citado pode ser alterado substituindo o sensor de inclinação por um *encoder* que irá determinar a velocidade atual das rodas do veículo e com esta informação atuar para que, assumindo que não haja deslizamento, o mesmo mantenha sempre a mesma velocidade, independentemente de inclinações na pista que possam variar.

2.4. Arquiteturas de Controle

A definição de arquitetura de controle robótico segundo Russel (RUSSELL 2003) [32] é:

"A arquitetura de um robô define como é organizada a tarefa de gerar ações através da percepção."

As principais arquiteturas são descritas a seguir.

2.4.1. Arquitetura Horizontal

Tradicionalmente, para o desenvolvimento de sistemas de controle para robôs móveis autônomos, utiliza-se a arquitetura horizontal, que tem como premissa dividir o controle em unidades de função. Cada uma destas unidades está intimamente ligada e dependente às unidades vizinhas, de tal forma que todo o sistema deve ser projetado de uma forma completa, pois um nível sozinho não é capaz de executar nenhuma ação. Para se adicionar uma nova atividade, é preciso alterar todo o projeto em sua concepção.

Neste tipo de arquitetura, as tarefas do sistema de controle são divididas em várias tarefas mais simples baseadas em suas funcionalidades. Uma abordagem comum é dividir as tarefas como mostra a figura 10.

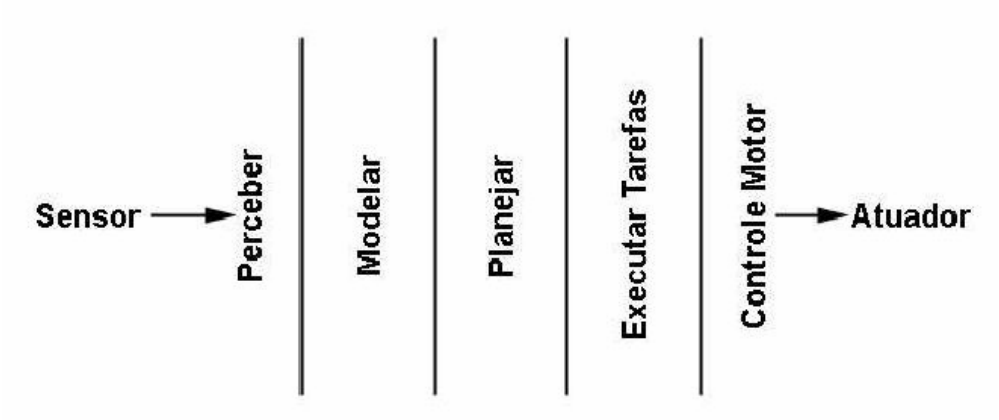

Figura 10 – Arquitetura Horizontal de Controle. Modelo SMPA *(Sense, Model, Plan, Act)*

Os sistemas de controle baseados nesta arquitetura executam suas tarefas em várias etapas. As entradas sensoriais são utilizadas para modificar a representação interna do ambiente e logo após e baseado nesta representação, um plano a longo prazo é elaborado. Esta sequência resulta em uma série de ações que o robô deve executar para alcançar o seu objetivo. Esta série de ações é utilizada para comandar os atuadores do robô e encerrar o ciclo de controle. A partir deste ponto o sistema é reiniciado para atingir novos objetivos.

Obter as informações relevantes do ambiente e construir um modelo o mais completo possível é a parte fundamental desta arquitetura. Feito este modelo estático do mundo real, os algoritmos planejam da maneira mais eficiente que podem as ações necessárias para alcançar os objetivos. Problemas diversos surgem com esta abordagem. Armazenar um modelo, na maioria dos casos, é complexo, devido às limitações e imperfeições dos sensores. Um ambiente dinâmico inviabiliza o planejamento gerado, assim como o fato de que enquanto o sistema de controle está planejando a próxima seqüência de ações, ele não é capaz de perceber as mudanças no ambiente. Caso algo diferente ocorra no ambiente enquanto o sistema está planejando, estas diferenças não serão consideradas no plano, o que resulta, no mínimo, em um plano defasado em relação à realidade atual, mas possivelmente em um plano perigoso para o robô e para o ambiente em questão.

2.4.2. Arquitetura Vertical

A arquitetura vertical, utilizada neste trabalho, foi idealizada pela primeira vez por Rodney Brooks em 1986 (BROOKS 1986) [16], que em sua definição, denominou de arquitetura "*subsumption*". Será utilizada a arquitetura "*subsumption*" como exemplo de arquitetura vertical, e as descrições serão baseadas em uma arquitetura vertical mais genérica. O termo tem tradução literária para subsunção, é sinônimo de supressão ou ato de suprimir.

O desenvolvimento desta arquitetura difere notavelmente da maneira tradicional de dividir o controle em módulos de função, como percepção, planejamento, modelagem, etc.

Brooks propôs que, ao invés de as tarefas serem divididas em função da funcionalidade, a divisão deveria ser feita baseando-se em comportamentos que executam tarefas, organizados em camadas. A organização do controle se dá na criação de um conjunto de camadas que representam uma tarefa ou comportamento complexo. (Figura 11).

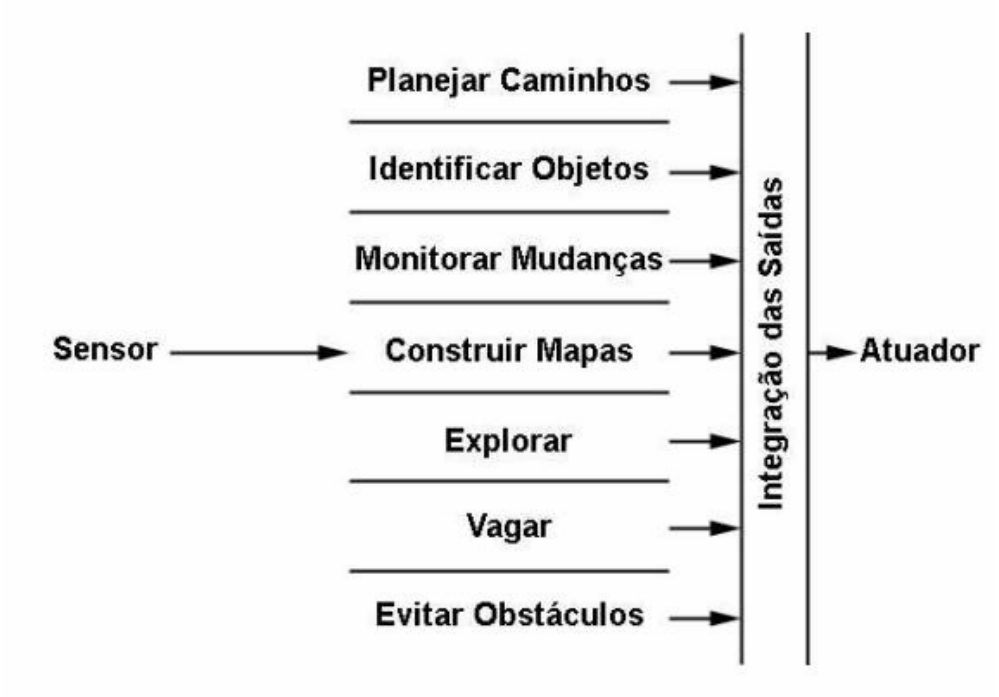

Figura 11 – Arquitetura Vertical de Controle.

A divisão em camadas de atividades permite acrescentar um comportamento quando este for necessário, construindo a nova camada e relacionando-a com o sistema. A idéia é construir um sistema autônomo completo, muito simples, e testá-lo no mundo real. (Figura 12).

Figura 12 – Arquitetura *Subsumption*.

Subsumption, em inglês, vem de subsunção ou supressão, e encaixa perfeitamente neste contexto, pois a camada mais elevada suprime a função das camadas inferiores quando uma determinada configuração dos sensores indica uma situação favorável para sua atuação. O nível acima suprime o fluxo de dados da camada inferior. É possível interromper o sistema em qualquer camada, pois as

camadas inferiores continuarão formando um sistema completo e funcional, porém com menos “inteligência”.

Um sistema de controle que utiliza a arquitetura *subsumption* é constituído de diversos comportamentos executados em paralelo, e cada um destes apresenta uma resposta de orientação diretamente para os atuadores utilizando as entradas sensoriais. As saídas sugeridas pelo comportamento com a mais alta prioridade suprimem as saídas de baixa prioridade, e são então utilizadas para controlar os atuadores do robô. Não existe relação combinatória entre as saídas de camadas diferentes. As saídas são sequenciadas conforme a prioridade da hierarquia.

A principal novidade desta arquitetura foi que uma entrada sensorial não precisa mais passar por uma série de camadas de processamento antes de se transformar em uma saída para os atuadores. Somente as entradas sensoriais relevantes para a tarefa executada por aquele determinado comportamento são utilizadas, não para atualizar uma representação interna, mas para gerar diretamente as saídas para os atuadores. Isto torna a ligação entre os sensores e os atuadores mais forte e direta que resulta em alta velocidade de resposta.

A técnica "*open loop*" utilizada na arquitetura horizontal é abandonada e os sistemas baseados na arquitetura vertical são na maioria construídos utilizando-se uma técnica de controle do tipo "*feedback*".

Atividades simples dos níveis inferiores provocam reações rápidas devido à não utilização de representações complexas do mundo por parte das camadas. O sistema está baseado no principio de percepção-ação. A idéia é sentir o ambiente frequentemente e assim ter uma idéia atualizada e em tempo real do que acontece.

Cada camada pode ser imaginada como tendo seu propósito implícito. A idéia principal é adaptar cada objetivo às condições atuais do ambiente real.

2.4.3. Arquitetura Híbrida

A integração dos componentes da arquitetura horizontal para o planejamento a longo prazo, e resolução de problemas com os componentes da arquitetura vertical para o controle em tempo real, é denominada arquitetura híbrida.

Esta arquitetura pode ser classificada entre a vertical e a horizontal por combinar aspectos de ambas. A abordagem híbrida não traz nenhuma estratégia

isolada adequada para a execução de todas as tarefas relevantes para um sistema de controle robótico, o que traria complicações na elaboração do planejamento e execuções em tempo real das tarefas. Apenas problemas muito específicos teriam necessidade do uso deste tipo de arquitetura.

2.5. Estratégias de Controle

"A estratégia de controle fornece os princípios para a organização dos sistemas de controle. Além de fornecer a estrutura, que impõe restrições sobre a forma como problema de controle pode ser resolvido." (MATARIC 1997) [33]

Mataric também resumiu a maneira como as estratégias mais utilizadas se comportam:

Controle Deliberativo – "Pense, depois aja"
Controle Reativo – "Não pense, reaja"
Controle Híbrido – "Pense e atue ao mesmo tempo"
Controle Baseado em Comportamento – "Pense na maneira de atuar"

As estratégias de controle são descritas a seguir.

2.5.1. Controle Deliberativo

Pensar e depois agir resume a idéia do controle deliberativo, pois ele contém explicitamente representado um modelo simbólico do mundo, e as decisões são planejadas previamente para só após serem colocadas em prática.

O robô usa todas as informações sensoriais e todo o conhecimento armazenado internamente para construir um modelo mais completo possível do ambiente, eventualmente usando fusão de sensores. A partir deste modelo, o robô gera um plano para conseguir atingir seus objetivos, e finalmente executa este plano.

O sistema de controle normalmente é organizado por uma decomposição em módulos do processo de decisão, composto por um módulo de processamento sensorial, um para modelagem, um de planejamento, um módulo de tomada de decisões, e um módulo de execução. Esta fragmentação funcional permite que

operações complexas sejam executadas, mas implica em forte interdependência sequencial entre os módulos de tomada de decisão.

Embora essa técnica tenha demonstrado alta capacidade de complexidade, suas limitações logo se tornaram evidentes. O módulo de planejamento, um dos principais componentes da inteligência artificial, é um processo que exige muito poder computacional se comparado aos outros módulos. Requer que o robô construa uma sequência de sentir, modelar, planejar e agir, ou seja, combinar os dados sensoriais em um mapa do mundo, planejar uma trajetória ótima e então enviar as saídas geradas para as rodas do robô. Brooks (BROOKS 1986) [16] se refere a esta estratégia de controle como o modelo SMPA (*Sense - Model - Plan - Act*). Este módulo deve avaliar potencialmente todos os planos possíveis até encontrar um que lhe permita atingir seu objetivo, resolver a tarefa, ou decidir sobre a trajetória a ser executada.

O controle deliberativo tem dificuldade de operar em um ambiente dinâmico, pois ruídos ou imprecisões dos sensores causam grandes erros acumulados e o modelo de mundo torna-se ineficaz, o que gera a necessidade de uma estratégia de controle diferenciada.

2.5.2. Controle Reativo

O controle reativo é fundamentado na biologia, e utiliza a concepção do estímulo e resposta, que não exige o conhecimento prévio do ambiente a ser explorado e não depende dos processos complexos utilizados no controle deliberativo. É um método de controle poderoso e eficaz que se inspira na natureza. Insetos, que superam amplamente os vertebrados em número, são altamente reativos.

Esta estratégia utiliza como base a arquitetura vertical para dividir o controle em uma série de regras que concorrem entre si. Estas regras envolvem uma quantidade mínima de computação, e não utilizam representações internas ou conhecimento global do mundo.

Sistemas reativos podem alcançar rápidas respostas em tempo real pela incorporação no controlador do robô de uma coleção de regras pré-programadas, com estados simples definidos como, por exemplo, uma regra que define que em caso de qualquer colisão os motores devem ser parados, e outra regra que em caso

de parada dos motores deve-se recuar. Cada um destes processos é conectado com suas próprias entradas sensoriais, com a possibilidade de inibir as entradas ou saídas de outros comportamentos, dependendo das prioridades de cada um. Isso faz o controle reativo especialmente adequado para mundos dinâmicos e não-estruturados, em que ter acesso a um modelo do mundo não é uma opção viável. Além disso, a quantidade mínima de computação embutida nos sistemas reativos torna a resposta rápida às mudanças no ambiente.

Braitenberg (BRAITENBERG, 1984) [43], utilizando o ponto de vista da psicologia, estendeu os princípios do comportamento de circuitos analógicos para uma série de veículos reativos. Estes sistemas utilizam inibidores e excitadores diretamente acoplados entre os sensores e os motores. Braitenberg criou veículos que simulam de forma primitiva reações animais como medo, agressividade (Figura 13) e até mesmo amor. (Figura 14).

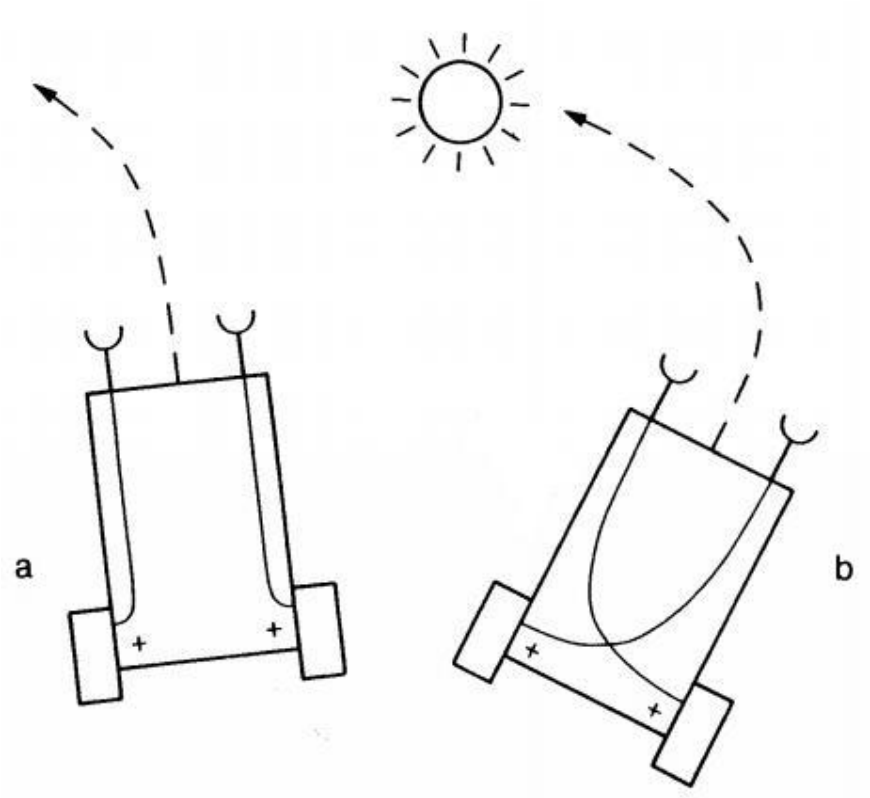

Figura 13 – Robôs reativos. a) Expressa medo. b) Expressa agressividade.

Os robôs apresentados na figura 13 possuem dois motores e dois sensores foto-elétricos ligados diretamente aos motores, ou seja, quanto maior a luminosidade, maior é a tensão aplicada ao respectivo motor. A figura 13(a)

expressa um comportamento de medo da luz devido à proximidade com a fonte de luz, e o robô se locomove para longe dela. Já na figura 13(b), os sensores são ligados inversamente aos motores, o que traduz um comportamento de agressão, pois quanto mais próximo da luz, mais rápido o robô irá se aproximar até colidir.

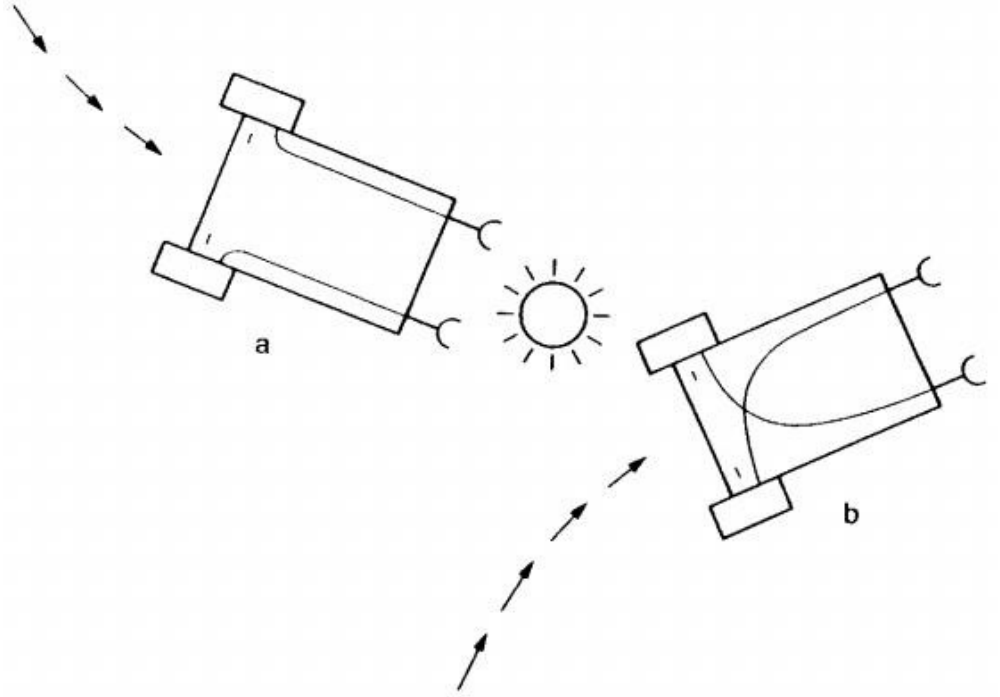

Figura 14 – Robôs reativos. a) Expressa amor. b) Expressa curiosidade.

Na figura 14, o mesmo esquema de ligação é utilizado, porém inibidores são usados entre os sensores e os motores, ou seja, quanto maior a luminosidade, menor é a velocidade transmitida para as rodas, e a ausência de luminosidade faz com que o robô pare. Este fato traduz na figura 14(a) um comportamento denominado de "amor" pela fonte luminosa, pois o robô ao detectar a fonte luminosa se aproxima até parar, e ficará parado indefinidamente, como se contemplasse a fonte. Já na figura 14(b), com as ligações invertidas o robô desvia lentamente ao se aproximar da fonte luminosa até não detectá-la mais, o que demonstra curiosidade por estar sempre próximo à fonte luminosa, mas sempre virado para outro lado aguardando alguma outra fonte, um comportamento denominado de "curiosidade" ou de "exploração".

Em geral, um sistema reativo não possui nenhuma forma de representação interna do modelo do ambiente, e não utiliza um raciocínio simbólico complexo. Ele é baseado no princípio da reatividade, ou seja, na suposição de que

comportamentos inteligentes podem ser gerados sem nenhuma representação simbólica explícita e de que a inteligência é uma propriedade que emerge de certos sistemas complexos.

No entanto, as limitações à reatividade pura incluem a incapacidade de armazenar as representações internas do mundo e, portanto, não pode otimizar sua trajetória. O controle reativo abdica da complexidade de raciocínio para valorizar a velocidade de resposta. Em outros tipos de ambientes e tarefas, onde os modelos internos, memória ou aprendizagem são necessários, o controle reativo não é suficiente.

2.5.3. Controle Híbrido - Reativo/Deliberativo

O controle híbrido tem por objetivo combinar os melhores aspectos dos controles reativo e deliberativo. A resposta em tempo real da reatividade e a racionalidade e otimização da deliberação trazem resultados que contêm os dois componentes combinados, o reativo, condição simultânea de regras de ação, e o deliberativo, que deve interagir para produzir uma saída coerente.

Combinar estas estratégias é um desafio porque o componente reativo lida com as necessidades imediatas do robô, tais como deslocamento, evitando obstáculos e, portanto, opera em uma escala de tempo muito rápida e usa diretamente dados sensoriais. Já o componente deliberativo utiliza uma representação altamente abstrata e simbólica do mundo, e funciona numa escala de tempo mais longa. Enquanto os resultados das duas componentes não estão em conflito, o sistema não exige uma coordenação maior.

As duas partes do sistema devem interagir para que possam beneficiar o sistema como um todo, e consequentemente o sistema reativo deve suprimir o deliberativo se o mundo apresentar algum desafio inesperado e imediato. Analogamente, a componente deliberativa deve fornecer informações à reativa, a fim de guiar o robô em direção a trajetórias mais eficientes e melhores até seu objetivo. A interação das duas partes do sistema exige um componente intermediário, que concilia as diferentes representações utilizadas pelos outros dois e os conflitos entre as suas saídas. A construção deste componente intermediário é geralmente o maior desafio do projeto do sistema híbrido.

O controle híbrido é referenciado como sendo uma estratégia de três camadas, devido à sua estrutura, que consiste na execução, coordenação e planejamento, que são respectivamente as camadas reativa, intermediária e deliberativa.

Duas arquiteturas clássicas que utilizaram a estratégia híbrida em sua concepção foram a *AuRA* (ARKIN 1986) [1], e a arquitetura *Atlantis* (GAT 1991) [35]. Ambas utilizam o controle em três camadas.

Arquiteturas de três camadas têm o objetivo de aproveitar o melhor do controle reativo no que diz respeito à dinâmica do ambiente e velocidade de resposta, com o melhor do controle deliberativo, com ações globalmente eficientes sobre uma escala de tempo. No entanto, existem questões complexas envolvidas na integração dessas estratégias fundamentalmente diferentes, e na maneira em que a sua funcionalidade deve ser dividida.

2.5.4. Controle Baseado em Comportamento

O controle baseado em comportamento tem sua inspiração obtida da biologia, e tenta modelar o cérebro dos animais para lidar com problemas complexos, tanto de planejamento quanto de execução de ações. É constituído de um conjunto de módulos de interação, chamados de comportamentos, que coletivamente tentam atingir um nível complexo de comportamento. Para um observador externo, os comportamentos são os padrões de atividade do robô emergentes das interações entre o robô e seu ambiente. Um programador vê os comportamentos como módulos de controle que definem um conjunto de ferramentas para alcançar um objetivo.

Cada comportamento recebe entradas de sensores, de outros comportamentos do sistema, ou de ambas, e oferece saídas para os atuadores do robô ou para outros comportamentos. Assim, um controle baseado em comportamento é uma rede estruturada de interações entre comportamentos, sem uma representação centralizada do mundo ou foco de controle. Em vez disso, os comportamentos individuais e redes de comportamentos garantem quaisquer informações sobre o estado e modelos.

"O controle comportamental permite relacionar, de maneira coerente, todos os elementos que um controle robótico exige." (JONES, 1998) [20]

Sistemas baseados em comportamento bem projetados aproveitam a dinâmica de interação entre os comportamentos e entre os comportamentos e o ambiente. A funcionalidade destes sistemas emerge dessas interações e, portanto, não são propriedades do robô ou do ambiente, e sim um resultado da interação entre eles.

Ao contrário do controle reativo, que utiliza conjuntos de regras reativas com pouco ou nenhum controle do estado e sem representação, o controle baseado em comportamento utiliza conjuntos de comportamentos que não possuem essas restrições. Comportamentos têm estados e podem ser usados para construir representações, permitindo o raciocínio, planejamento e aprendizagem.

Relacionado ao controle híbrido que precisa de um modelo do ambiente e trabalha em uma escala de tempo mais lenta o controle baseado em comportamento ganha velocidade de processamento por não ter esta representação armazenada. Outro diferencial é que no controle por comportamento não existe um processo que precisa coordenar partes reativas com deliberativas, que é uma tarefa complexa e possui dependências dos módulos de função do robô, que não permite adições de outros comportamentos de maneira modular.

O intuito deste trabalho é utilizar o controle baseado em comportamento para explorar ambientes dinâmicos e desconhecidos sem um modelo pré-definido do mesmo.

O controle baseado em comportamento possui algumas características bem definidas, entre elas:

- Comportamentos são implementados como as leis de controle (por vezes semelhantes às usadas na teoria de controle), seja em software ou hardware, como um elemento de processamento ou como um processo.
- Cada comportamento pode receber entradas de sensores do robô (sensores de proximidade, detectores de ultrassom, sensores de contato, câmera) e de outros módulos. As saídas são enviadas para os atuadores do robô (rodas, garras, braços, fala) e outros módulos.
- Comportamentos diferentes podem receber contribuições de forma independente do mesmo sensor e comandar de ações de saída para o mesmo atuador.

- Os comportamentos são codificados para serem de relativa simplicidade, e são adicionados ao sistema de forma incremental.
- Comportamentos (ou seus subgrupos) são executados simultaneamente e não sequencialmente, a fim de explorar o paralelismo e velocidade de computação, bem como a dinâmica de interação entre os comportamentos e entre os comportamentos e o ambiente.

A robótica baseada em comportamento foi desenvolvida para permitir ao robô adaptar-se à dinâmica do ambiente do mundo real sem trabalhar sobre abstrações da realidade, mas também dar-lhe uma capacidade computacional maior da que está presente nos robôs reativos. Sistemas baseados em comportamento possuem uma forte ligação entre o sentir e o agir devido aos comportamentos. É raro para um comportamento realizar cálculos extensos confiando em um modelo de ambiente tradicional, a menos que tal cálculo possa ser feito em tempo hábil para uma resposta dinâmica que exija rapidez na saída.

Os comportamentos são projetados em diferentes níveis de abstração, facilitando a construção destes sistemas. Novos comportamentos são introduzidos no sistema de forma incremental e modular, desde o simples ao mais complexo, até que a interação entre eles resulte na capacidade total desejada para o robô. (Figura 15).

Comportamento para desviar de obstáculos estáticos assim como desviar especificamente de “predadores” e o comportamento que gera um ruído aleatório criado para escapar situações singulares, quando executados em paralelo gera um comportamento mais completo como o comportamento explorar. (Figura 15).

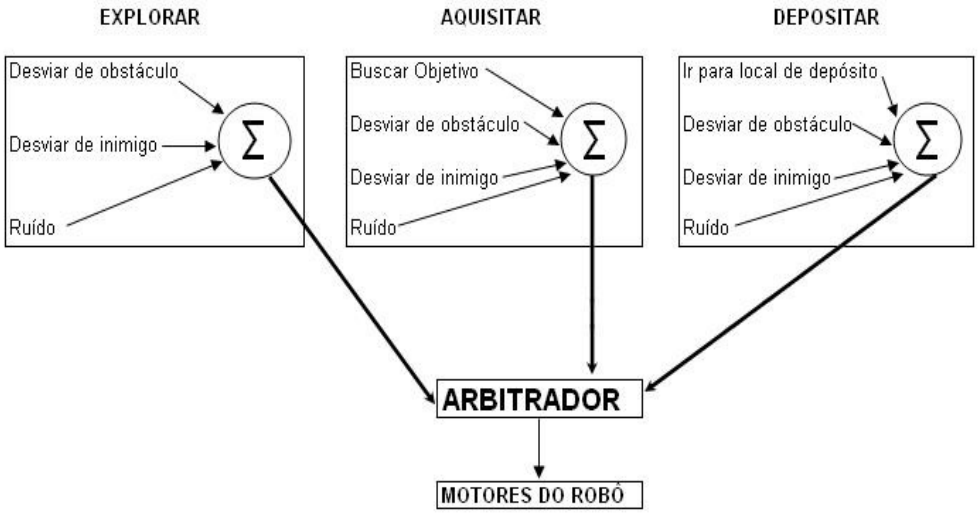

Figura 15 – Comportamentos complexos gerados a partir da interação entre comportamentos simples.

O primeiro passo é implementar os comportamentos primários de sobrevivência, como evitar obstáculos e colisões iminentes. Estes comportamentos são muitas vezes de natureza reativa, uma vez que as regras reativas são componentes de comportamentos simples. Nota-se que existe uma distinção entre as condições de ativação, que permitem ao comportamento gerar estímulos, a partir dos quais as ações são geradas.

Posteriormente, novos comportamentos são adicionados para fornecer capacidades mais complexas, como seguir paredes, perseguir um alvo, encontrar um objeto, recarregar uma bateria, evitar luz, juntar-se a um grupo, pegar um objeto, encontrar um marco definido, fugir de um "predador".

A interação e integração dos efeitos temporais e espaciais são de fundamental importância no comportamento destes sistemas. O efeito combinado dos comportamentos executados paralelamente ao longo do tempo, orientados pela percepção e estados internos, cria o comportamento relevante à dinâmica de um sistema de controle baseado em comportamento.

Os sistemas baseados em comportamento devem resolver o problema da escolha de uma determinada ação ou comportamento a partir de várias opções. Este é um dos desafios centrais da concepção baseada em comportamento. Uma abordagem para seleção de ação é o uso de uma hierarquia de comportamento pré-definido, em que os comandos do mais alto nível ao comportamento ativo são enviados para o atuador e todos os outros de nível inferior são ignorados. Existem

várias outras metodologias baseadas em outros princípios para solucionar o problema da seleção de ação, como a arquitetura de esquemas motores. Estas metodologias têm como objetivo proporcionar uma maior flexibilidade mas, em alguns casos, podem fazê-lo com o custo de reduzir a eficiência ou a capacidade de análise do sistema de controle resultante.

2.5.4.1. *Motor-Schema* e Campos Potenciais

Dentro do controle baseado em comportamento existem diversas arquiteturas possíveis para organização dos comportamentos, dentre elas destacam-se:

- "S*ubsumption*" por Rodney Brooks em 1986 [16]
- "*Motor-Schema*" por Ronald Arkin em 1987 [38]
- "*Action-selection*" por Pattie Maes em 1990 [49]
- "*Colony*" por John Connell em 1990 [50]
- "*Animated agent*" por R.James Firby em 1989 [54]
- "*DAMN (Distributed Architecture for Mobile Navigation)*" por Julio Rosenblatt em 1997 [51]
- "*Skill Network*" por David Zeltzer em 1991 [53]
- "*Circuit*" por L. Kaelbing e S. Rosenschein em 1991 [52]

Cada método possui características que se adequam melhor a diferentes tipos de finalidades para as quais o robô será programado. A arquitetura usada neste trabalho é a de esquemas motores (*Motor-Schema*) desenvolvida por Ronald Arkin (ARKIN 1987) [36], a mais difundida no controle baseado por comportamento.

A teoria dos esquemas motores fornece os seguintes recursos para a especificação e concepção de sistemas baseados em comportamento (ARBIB 1992) [37]:

- Explica o comportamento em termos de controle simultâneo de muitas atividades diferentes.
- Um esquema armazena como reagir e a forma que a reação deve ser realizada.

- É um modelo distribuído de computação.
- Fornece uma linguagem para conectar ação e percepção.
- Os níveis de ativação são associados com os esquemas que determinam a sua disponibilidade ou a aplicabilidade de atuação.
- É útil para explicar o funcionamento do cérebro, bem como aplicações distribuídas de inteligência artificial (como a robótica baseada em comportamento).

A teoria dos esquemas motores na robótica móvel autônoma, segundo Arkin (ARKIN 1987) [36], fornece grande modularidade computacional ao sistema para expressar a relação entre controle motor e percepção, em contraste com os modelos de redes neurais. Os esquemas motores atuam individualmente, porém todos em paralelo, ou seja, são agentes distribuídos em uma competição cooperativa que os torna facilmente mapeáveis em arquiteturas distribuídas. Comportamentos primitivos gerados pelos esquemas motores, quando agrupados, podem gerar comportamentos mais complexos.

O método de esquemas motores aplicado na robótica móvel fornece comportamentos reativos que, ao trabalharem de forma simultânea, produzem resultados inteligentes e comportamentos complexos em resposta a estímulos ambientais.

As diferenças e principais vantagens desta arquitetura para as demais existentes são:

- As saídas geradas pelos comportamentos são representadas em um modelo único de vetores contendo velocidade e ângulo do robô, gerados a partir de uma abordagem de campos potenciais, também discutida neste trabalho.
- A adição dos vetores, ou seja, a soma de todas saídas, gera a melhor coordenação para o destino do robô.
- Não existe hierarquia pré-definida para a coordenação, em vez disso os comportamentos são configurados em tempo real a partir das interações do robô com o ambiente. Os esquemas podem ser instanciados ou não a qualquer momento, com base nos eventos perceptivos; essa estrutura se parece mais com uma rede dinâmica do que uma arquitetura em camadas.

- Arbitrar simplesmente está fora do escopo desta arquitetura, ao invés disso cada comportamento pode contribuir em graus variados para a resposta global do robô através dos ganhos relativos de cada comportamento.
- As incertezas perceptivas podem ser refletidas na resposta do comportamento, permitindo que estas sejam usadas como entrada para o cálculo comportamental.

Esta teoria teve como base de desenvolvimento os componentes reativos do projeto *AuRA* (ARKIN 1986) [1] e tem seu principal método de desenvolvimento baseado na etologia , ou seja, o estudo do comportamento animal.

Esquemas motores têm grande usabilidade em várias circunstâncias. Muitos dos comportamentos têm parâmetros internos que proporcionam maior flexibilidade na sua implantação. Os comportamentos geralmente são análogos aos comportamentos animais, pelo menos aqueles úteis para tarefas de navegação.

Um esquema de percepção está inserido dentro de cada esquema motor. Esses esquemas perceptivos fornecem as informações ambientais específicas para um comportamento específico. O princípio básico do esquema perceptivo contempla o algoritmo que fornece as informações necessárias para um determinado comportamento responder conforme desejado.

Cada esquema motor possui um processo perceptivo capaz de oferecer estímulos adequados o mais rapidamente possível. Esquemas perceptivos são definidos de forma recursiva, isto é, sub-esquemas perceptivos podem extrair pedaços de informações, que são posteriormente processados por um esquema perceptivo em uma unidade comportamental mais significativa. Um exemplo desta aplicação é o reconhecimento de uma pessoa com mais de um sensor. Sensores infravermelhos podem fornecer uma assinatura de calor enquanto que a visão computacional pode fornecer uma forma humana.

As informações geradas em cada um desses processos de nível inferior da percepção são fundidas em uma interpretação de nível mais alto antes de atuar nos motores do robô. Isso permite o uso de múltiplos sensores dentro do contexto de um único comportamento.

Cada esquema motor tem como saída um vetor de ação (que neste trabalho consiste em componentes de orientação e magnitude da velocidade) que define a

forma como o robô deve se mover em resposta aos estímulos recebidos. (Figura 16).

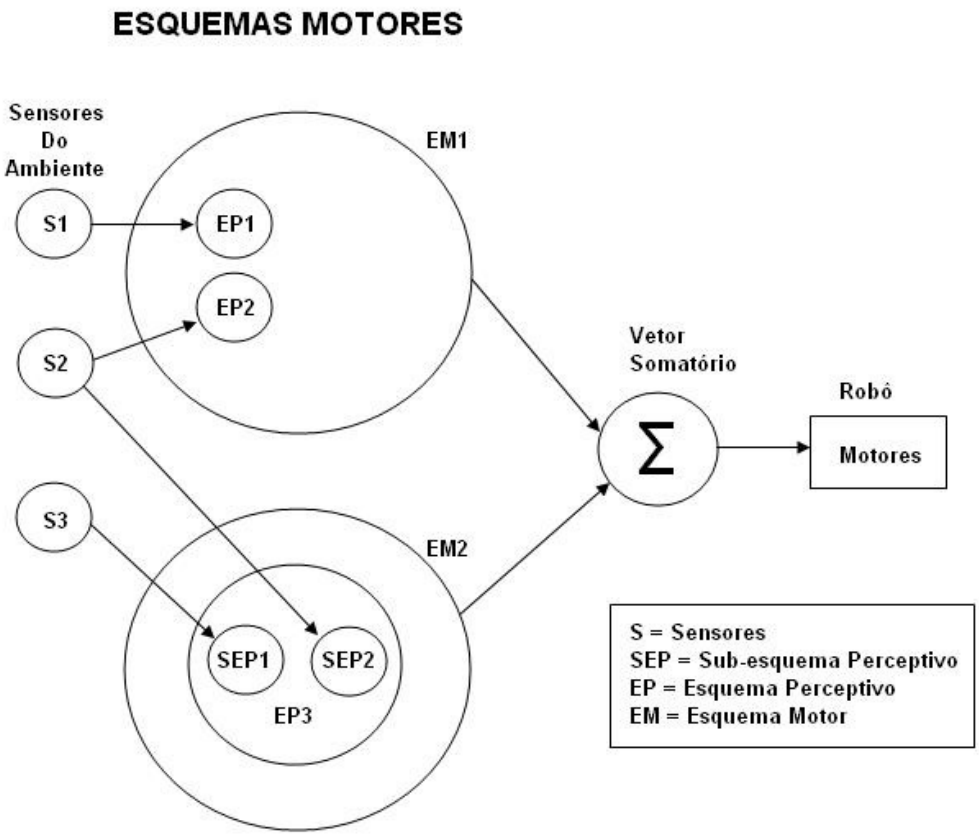

Figura 16 – Diagrama representativo da arquitetura *Motor Schema.*

Esta abordagem tem sido utilizada para navegação em terrenos planos, onde cada vetor é bidimensional (ARKIN 1989) [38], para a geração de três vetores tridimensionais para uso em voo ou navegação submarina (ARKIN, 1992) [39], e para uso em manipuladores móveis com muitos graus de liberdade redundantes (CAMERON 1993) [40].

Existem diversos esquemas clássicos já definidos, por exemplo:

- *Move-ahead*: mover-se em uma direção determinada.
- *Move-to-goal*: em direção a um objeto detectado.
- *Avoid-static-obstacle*: se afastar de objetos estáticos.
- *Dodge*: desviar de um objeto balístico que não altera trajetória.
- *Escape*: desviar de um ponto pré-determinado ou de um possível encontro com um predador que esteja continuamente em perseguição.

- *Stay-on-path*: manter-se no sentido do caminho como estradas e corredores. Para navegação tridimensional torna-se um esquema de *stay-in-channel*.
- *Noise*: move-se em uma direção aleatória.
- *Follow-the-leader*: move-se para um ponto determinado próximo a algum objeto em movimento. Comporta-se como se estivesse vinculado ao objeto.
- *Probe*: move-se para áreas abertas.
- *Dock*: abordagem de um objeto a partir de uma determinada direção.
- *Avoid-past*: afastar-se das áreas visitadas recentemente.
- *Move-up, move-down, maintain-level*: mover-se para cima, para baixo ou manter altitude.
- *Teleautonomy*: permite que o operador humano interaja e forneça dados ao sistema de controle no mesmo nível que outro esquema motor.

O método de coordenação utilizado é o do somatório e normalização dos vetores de saída, e o método de codificação da resposta dos esquemas motores é o uso contínuo de campos potenciais.

Uma partícula carregada atravessando um campo magnético ou uma bola rolando de uma colina são visualizações de campos potenciais reais. A idéia básica dos campos potenciais, apresentada por Khatib (KHATIB, 1985) [41] e Krogh (KROGH, 1984) [42], é a utilização de funções potenciais para criação de campos de forças que representam a direção e angulação que o robô deve navegar, representados por vetores com as componentes de magnitude da velocidade e direção. Objetivos são associados a campos de atração enquanto que obstáculos a campos de repulsão. O comportamento exibido pelo robô dependerá da combinação e da forma dos campos.

Uma característica marcante deste método é a codificação da ação de forma contínua, ou seja, em todo o plano haverá vetores de força associados ao objeto, não importando tamanho ou forma.

Os campos potenciais mais utilizados no controle baseado em comportamento que utiliza a arquitetura de esquemas motores são o uniforme, o

perpendicular, o atrativo, o repulsivo e o tangencial como mostrado na figura 17: (MURPHY 2000) [26]

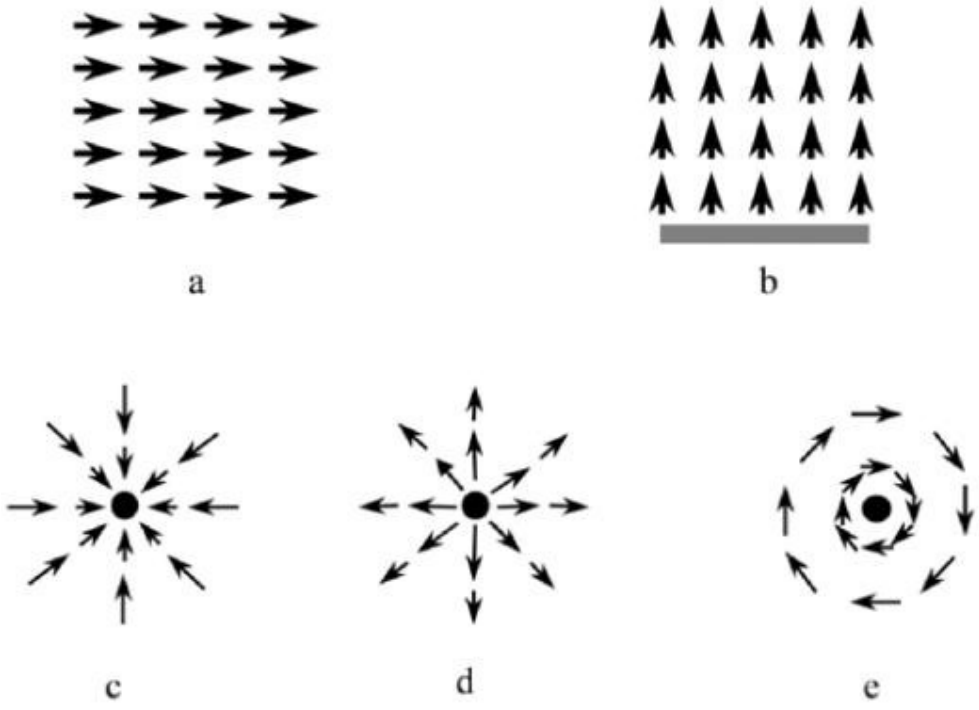

Figura 17 – Campos potenciais mais comuns. a) Uniforme b) Perpendicular c) Atrativo d) Repulsivo e) Tangencial

- Campo uniforme – Um robô em um campo uniforme sofre as mesmas forças em magnitude e direção em qualquer lugar que ele esteja. Qualquer movimento que for realizado pelo robô será guiado pelo campo, e acompanhará a direção do campo com a velocidade constante associada a ele. Este tipo de campo é usado quando se deseja que o robô tenha forças que o conduzam sempre em uma determinada direção.
- Campo perpendicular – Neste campo o robô é orientado a se afastar de um determinado obstáculo ou parede. Este campo pode estar associado a uma função decrescente, fazendo com o que o robô sofra uma força de repulsão que decresce conforme se afasta da parede.
- Campo atrativo – Um objeto de destino gera um campo radial de atração para o robô. Onde quer que o robô esteja, ele sofrerá uma força de atração na direção do objeto em questão. São campos muito

utilizados para representar objetivos como comida, luz, destinos finais.

- Campo repulsivo – É o oposto ao campo atrativo, exerce uma força de repulsão sobre o robô, muito utilizado para obstáculo e predadores. Quanto mais perto o robô estiver, maior será a magnitude da repulsão, e na direção de afastamento.
- Campo tangencial – Este campo gera uma força tangencial ao objeto desejado. São forças perpendiculares as linhas radiais que se estendem para fora do objeto. São utilizados quando se deseja fazer o robô investigar um objeto ou apenas manter-se próximo a ele, mas em movimento constante tanto em sentido horário como anti-horário.
- Campo aleatório – Gera vetores aleatórios, com valores pequenos de magnitude e angulação variada (Figura 18). Muito utilizado para solucionar problemas de singularidade quando os somatórios de diferentes campos anulam a resposta desejada para uma situação causando uma colisão, por exemplo.

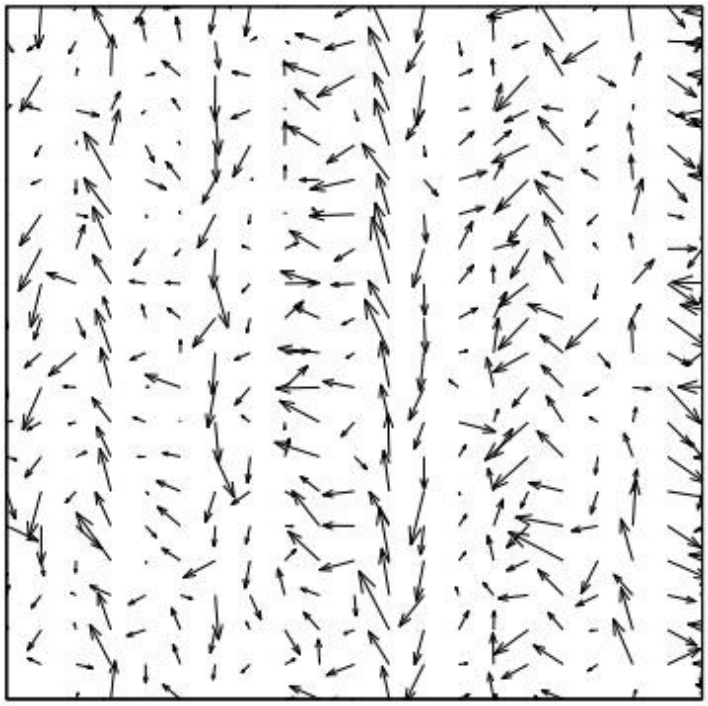

Figura 18 – Campo aleatório

Os esquemas motores clássicos utilizam os campos potenciais para a atuação motora no robô, e só calculam a contribuição de cada campo

Ao contrário dos campos onde a topologia é externamente especificada pelas condições ambientais, a topologia dos campos potenciais que um robô experimenta é determinada pelo programador. Mais especificamente, o programador atribui a cada comportamento criado uma determinada tarefa ou função, representa cada um destes comportamentos como um campo potencial, e combina todos os comportamentos para produzir o movimento do robô através da combinação dos campos potenciais mediante a superposição vetorial.

Uma representação de um campo repulsivo em três dimensões utilizada para navegação submarina de um robô móvel autônomo é apresentada na figura 19.

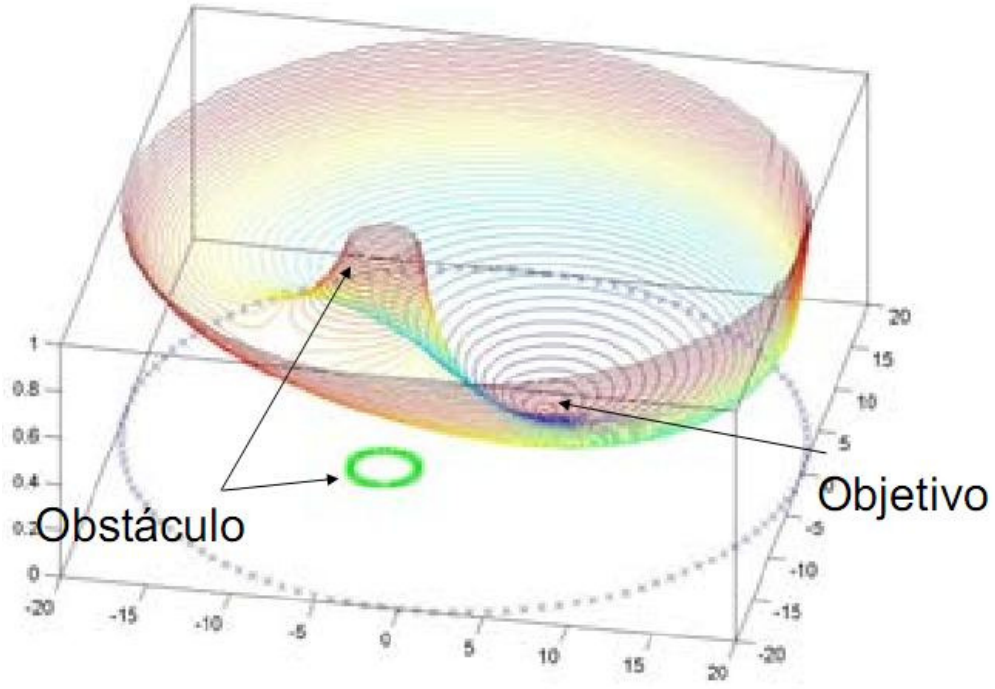

Figura 19 – Campos repulsivo 3D para navegação submarina. (MURPHY 2000) [26]

As codificações mais usadas dos campos potenciais utilizados nos esquemas motores, onde M denota a magnitude do vetor resposta, são apresentadas a seguir.

O esquema clássico *move-to-goal* (Figura 20) que define um ponto ou objeto de destino pode possuir muitas codificações de saída, como funções exponenciais decrescentes, mas descrevem-se a seguir funções simples que atendem bem às necessidades de robôs autônomos simples.

$$M = \begin{cases} \infty & \text{para } d > S \\ \dfrac{(d - R)^2}{S - R} \bullet G & \text{para } R < d \leq S \\ 0 & \text{para } d \leq R \end{cases}$$

onde a direção do vetor é radial ao redor do centro do obstáculo mas, direcionada para dentro. (Figura 20). e;

d = distância do robô até o centro do objeto.

G = ganho.

R = raio do objeto.

S = raio de interesse onde a função irá atuar.

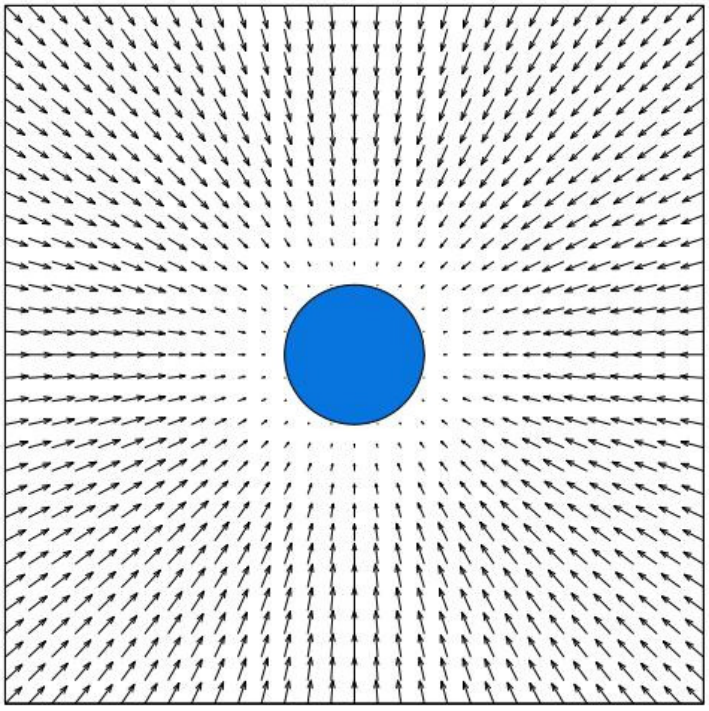

Figura 20 – Campo potencial *Move-to-goal.*

Neste esquema utilizou-se uma função simples que determina o decaimento da magnitude da velocidade do robô proporcionalmente ao quadrado da distância ao objeto. As simulações e experimentos deste trabalho utilizam robôs de pequeno porte, portando os valores para os parâmetros são expressos em cm. Os parâmetros d e S assumem valores máximos em torno de 200cm. O ganho de cada esquema pode ser atrelado a uma função que altere a capacidade de contribuição de suas saídas ao longo do processo. Entretanto, neste trabalho os

esquemas perceptivos possuem implicitamente a capacidade de contribuir com alta ou até nenhuma relevância durante o processo, por este motivo não é necessário utilizar funções que alterem os ganhos. Estes foram definidos como funções binárias onde os esquemas perceptivos são ativados e contribuem para o somatório vetorial ou simplesmente permanecem inertes.

Todos os valores referenciados de distâncias e velocidades podem ser analisados no código fonte das simulações. (Apêndice II)

Devido às limitações do ambiente de simulação foram desconsiderados os valores inerciais dos robôs. Uma vez que o foco do trabalho são robôs de pequeno porte, esta aproximação torna-se satisfatória.

O esquema *avoid-static-obstacle* é similar ao *move-to-goal*, porém gera forças de repulsão crescentes proporcionais ao quadrado da distância. (Figura 21).

$$M = \begin{cases} 0 & \text{para } d > S \\ \dfrac{S - R}{(d - R)^2} \bullet G & \text{para } R < d \leq S \\ \infty & \text{para } d \leq R \end{cases}$$

onde a direção do vetor é radial ao redor do centro do obstáculo mas, direcionada para fora. (Figura 21).

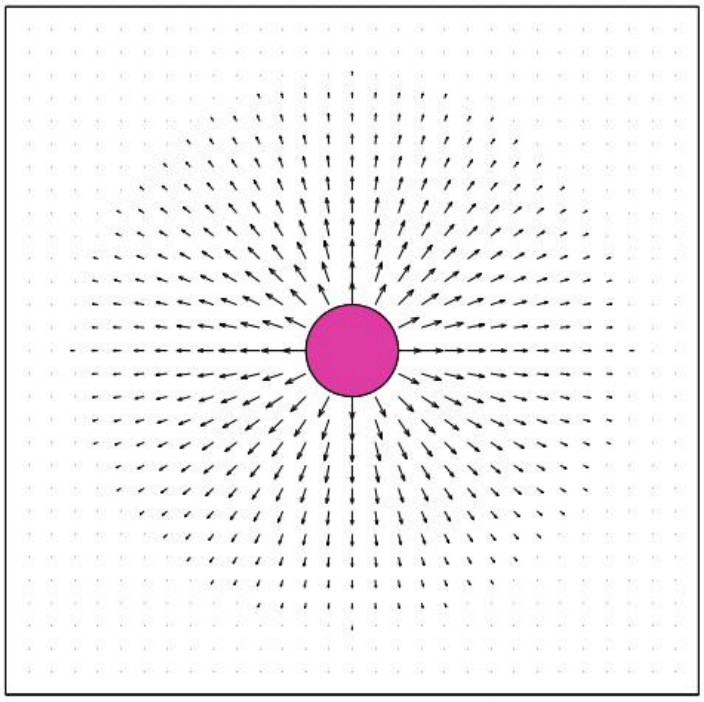

Figura 21 – Campo potencial *Avoid-static-obstacle.*

Quando se colocam ambos os objetos em um mesmo ambiente, os campos potenciais são sobrepostos e todas as forças resultantes geram o campo no qual o robô navegará.

Dentro do obstáculo, o campo de potencial repulsivo é infinito, e aponta para fora do centro do obstáculo. Fora do círculo de influência, o campo de potencial repulsivo é zero. Dentro do círculo de influência, mas fora do raio do obstáculo, a magnitude do vetor cresce proporcionalmente ao quadrado da distância do robô ao objeto.

Uma vez que haverá interações entre dois ou mais objetivos e obstáculos um campo onde todos os campos gerados se sobrepõem deve ser criado. (Figura 22).

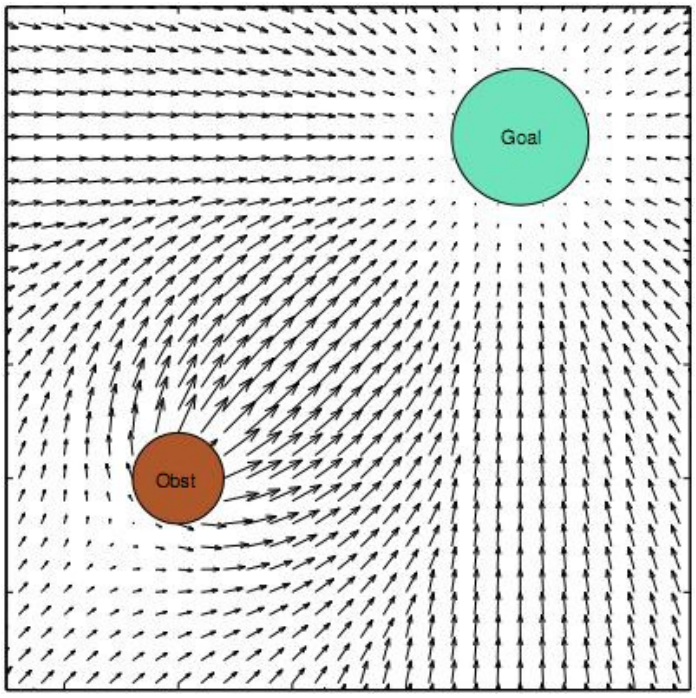

Figura 22 – Campos potenciais *Move-to-goal e Avoid-static-obstacle* sobrepostos.

A partir da criação do campo resultante final, pode-se analisar como um robô se comporta nesta situação, ou seja, qual será sua trajetória com base nos vetores resultantes da soma vetorial, vide figura 23.

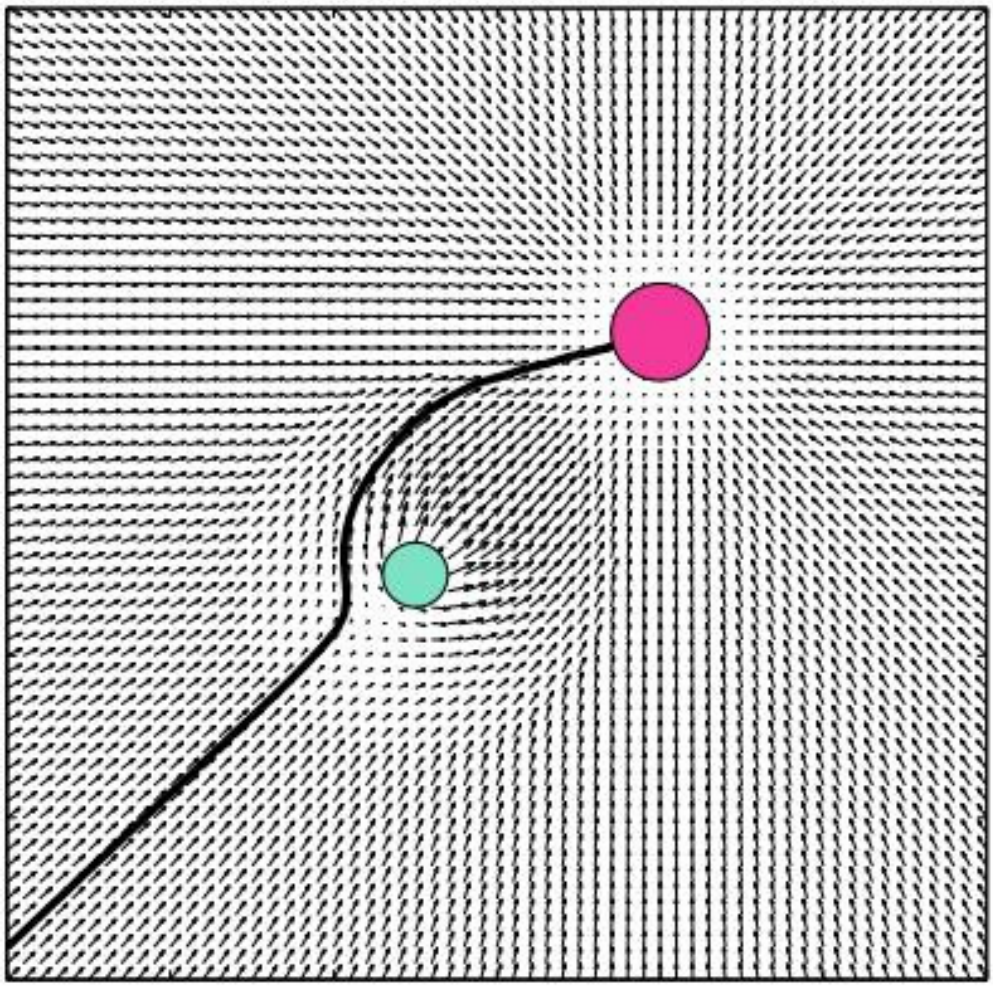

Figura 23 – Trajetória de um robô no ambiente com campos potenciais *Move-to-goal* e *Avoid-static-obstacle* sobrepostos.

Observa-se que o robô executa inicialmente uma trajetória uniforme que foi gerada pelo campo potencial do objetivo em todo o ambiente. Quando se aproxima de um obstáculo, as forças de repulsão fazem com que este seja desviado, e o somatório das forças de repulsão com as de atração do objetivo criam a trajetória com a qual o robô o alcança, desacelerando conforme a proximidade até parar completamente.

Note que existem outros tipos comuns de campos que podem ser utilizados no processo de criação do campo potencial resultante do ambiente.

2.6. Ambiente de Simulação *Player / Stage*

O *Player* é um ambiente computacional de desenvolvimento de sistemas de controle de robôs móveis amplamente utilizado por universidades, institutos de pesquisa e empresas em diversos países. Este projeto foi iniciado em 2000 por pesquisadores da *University of Southern Califórnia* para suprir a demanda de um controlador e simulador de robôs móveis, que fosse eficiente, de grande compatibilidade, e independente de arquitetura de programação do robô.

O desenvolvimento do *Player* conta com a colaboração de diversos pesquisadores das mais diversas instituições e, por ser um sistema de código aberto, de livre distribuição, e por rodar em Linux, está em constante desenvolvimento para se adequar a um número cada vez maior de plataformas robóticas e sensores comerciais. Assim como um sistema operacional cria uma interface de alto nível para facilitar o acesso de usuários e programadores ao hardware de um computador, o *Player* apresenta uma interface de acesso ao hardware de robôs móveis e sensores, facilitando sua programação e utilização. (Figura 24).

A estrutura do *Player* é baseada no modelo cliente/servidor. O servidor faz a interface com o robô e com outros sensores, obtendo dados, enviando-os para o cliente e recebendo instruções do cliente para o controle do robô e dos sensores. O cliente é o programa que controla efetivamente o robô, ou seja, a aplicação. O cliente é responsável por obter os dados do servidor, interpretá-los, e enviar instruções para o servidor, no caso o robô, para a execução de determinada tarefa.

A arquitetura cliente/servidor permite grande versatilidade no controle de robôs e sensores, de forma que um cliente pode controlar diversos servidores e diferentes clientes podem controlar diferentes sensores de um mesmo robô. Toda a comunicação entre cliente e servidor é realizada através de TCP/IP.

O cliente Player foi projetado para ser compatível com diversas linguagens e existem bibliotecas disponíveis para clientes em C, C++, Java, Python, Tcl, entre outras. Para que o cliente possa se comunicar com o servidor, é necessário que o código fonte do programa de controle inclua uma biblioteca fornecida com o Player.

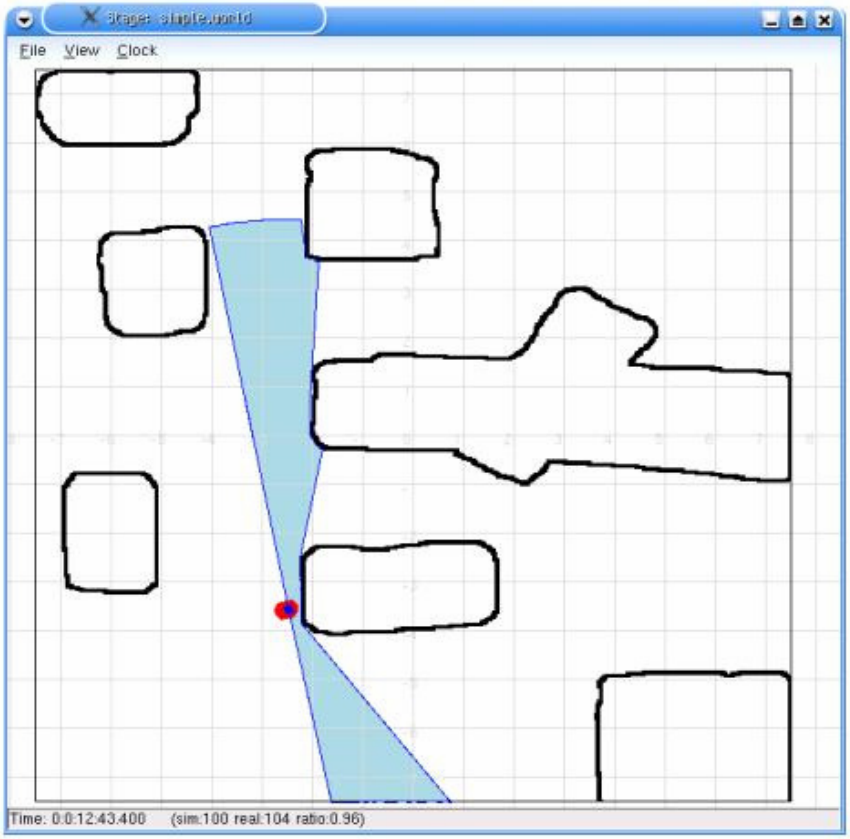

Figura 24 – Ambiente de simulação do *STAGE*.

O Stage é um módulo adicional ao Player que traz um simulador de robôs e sensores para ambientes bidimensionais. Múltiplos robôs e sensores podem ser simulados simultaneamente, controlados por um ou mais clientes. (Figura 25).

Figura 25 – Capacidade de simulação do *Stage*. (2000 robôs *Pioneer* 2-DX).

O Stage normalmente é usado para o desenvolvimento inicial de código, até que o mesmo se encontre confiável o suficiente para ser testado em robôs reais. Outras aplicações do Stage envolvem a utilização de sensores e robôs em quantidade não disponível nos experimentos, mas cujo código possa ser validado através de simulação. É também um simulador cinemático, que não leva em conta a dinâmica dos objetos simulados, contudo é possível obter grande proximidade com a realidade pelo fato de a simulação objetivar robôs de pequeno porte e rápida aceleração até atingir sua velocidade máxima. Como toda a comunicação cliente e servidor é realizada através da rede TCP/IP, é totalmente transparente para o programa cliente se o mesmo está conectado a um robô simulado ou a um robô real. Normalmente, a única modificação necessária para se executar um código desenvolvido para controlar um robô simulado em um código de um robô real é a mudança do IP do robô simulado, que é o computador que executa o simulador, para o IP do robô real.

Os projetos iniciais simulados pelo Stage estavam mais associados aos robôs *Pioneer* da empresa *ActivMedia*. Desde o início de sua implementação, a característica que mais se destaca neste ambiente de simulação é a capacidade de simular um controle de robôs, utilizando múltiplos robôs. (Figura 25).

O projeto Player/Stage avançou em relação aos projetos iniciais no sentido de permitir uma separação maior entre o hardware e o ambiente de programação dos robôs, permitindo utilizar diferentes tipos de robôs. (Figura 26).

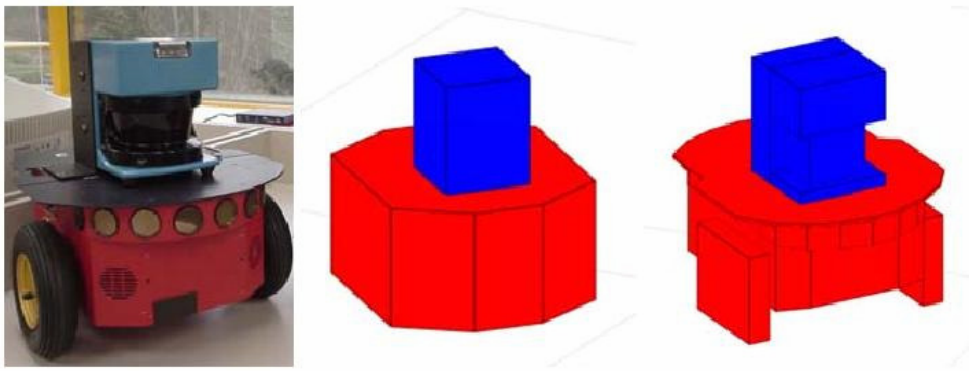

Figura 26 – Robô *Pioneer* 2-DX real, simulado simplificado e simulado de forma completa.

O projeto suporta uma série de robôs e sensores, além de uma independência de linguagem de programação, e pode simular o movimento dos

robôs e os dados coletados dos sensores a partir da utilização de um mapa em forma de uma imagem binária.

O processo de instalação e configuração do software se torna complicada em algumas ocasiões, para solucionar alguns problemas utilizou-se um guia desenvolvido por Denis Wolf (WOLF, 2009) [58].

Além do *Stage*, o *Player* também é compatível com o simulador *Gazebo*. O *Gazebo* é um simulador de robôs móveis em três dimensões que permite a simulação realista de ambientes complexos. Através do uso de bibliotecas de modelagem física, o *Gazebo* permite uma simulação extremamente fiel do comportamento e da interação física de robôs e objetos do ambiente. Por ser computacionalmente mais complexo que o *Stage*, o *Gazebo* requer mais recursos computacionais.

O *Gazebo* normalmente é utilizado em situações onde a simulação bidimensional do Stage não é fiel o suficiente. Por exemplo, em ambientes externos onde o solo é irregular e isso compromete substancialmente o funcionamento dos robôs e sensores, ou quando o robô deve mapear ambientes com objetos 3D complexos. Nesse caso, não é possível modelar esses objetos no Stage. (Figura 27).

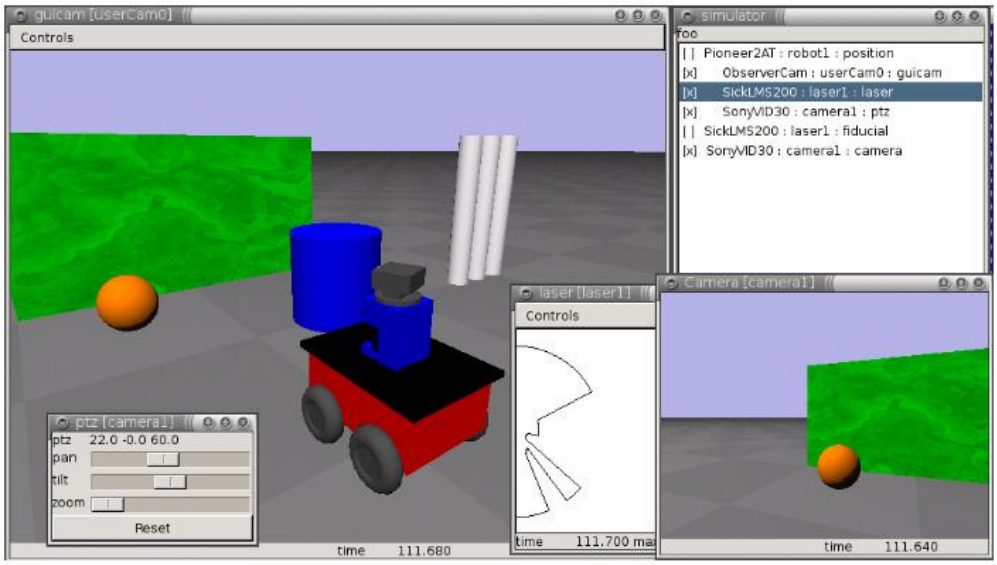

Figura 27 – Ambiente de simulação *Gazebo.*

No próximo capítulo, os sensores utilizados neste trabalho são apresentados.