**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO**
**CENTRO TECNOLÓGICO**

**PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA**

MICHAEL ANDRÉ GONÇALVES

# ALGORITMO A-ESTRELA DE ESTADO HÍBRIDO APLICADO À NAVEGAÇÃO AUTÔNOMA DE VEÍCULOS

VITÓRIA

2013

MICHAEL ANDRÉ GONÇALVES

# ALGORITMO A-ESTRELA DE ESTADO HÍBRIDO APLICADO À NAVEGAÇÃO AUTÔNOMA DE VEÍCULOS

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

VITÓRIA

2013

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

---

Gonçalves, Michael André, 1987-

G635a Algoritmo A-Estrela de estado híbrido aplicado à navegação autônoma de veículos / Michael André Gonçalves. – 2013.

64 f. : il.

Orientador: Alberto Ferreira de Souza.

Coorientador: Thiago Oliveira dos Santos.

Dissertação (Mestrado em Informática) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Robótica. 2. Veículos Autônomos. 3. Navegação. 4. Navegação de robôs móveis. I. Souza, Alberto Ferreira de. II. Santos, Thiago Oliveira dos. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

---

MICHAEL ANDRÉ GONÇALVES

# ALGORITMO A-ESTRELA DE ESTADO HÍBRIDO APLICADO À NAVEGAÇÃO AUTÔNOMA DE VEÍCULOS

COMISSÃO EXAMINADORA

---

Prof. Dr. Alberto Ferreira De Souza
Universidade Federal do Espírito Santo
Orientador

---

Prof. Dr. Thiago Oliveira dos Santos
Universidade Federal do Espírito Santo
Coorientador

---

Profa. Dra. Claudine Badue
Universidade Federal do Espírito Santo

---

Prof. Dr. Edilson de Aguiar
Universidade Federal do Espírito Santo

---

Prof. Dr. Luiz Chaimowicz
Universidade Federal de Minas Gerais

Vitória, 28 de Agosto de 2013.

Dedico esse trabalho à minha família e aos meus amigos, pelo incentivo e motivação que serviram como luz para o meu caminho.

# AGRADECIMENTOS

Primeiramente a Deus, pela força e iluminação. Ao meu orientador Prof. Dr. Alberto Ferreira De Souza pelo incentivo, ajuda e ensinamento que serviram como portas para o meu trabalho. Agradeço também ao meu coorientador Prof. Dr. Thiago Oliveira dos Santos pela revisão rigorosa deste trabalho. À minha família, especialmente minha mãe Eliana Salvador Ribeiro e meu pai Marcos Antônio Gonçalves pela compreensão de minha ausência durante os meus estudos. À minha namorada Isabele Vianna por me ajudar na escrita, mesmo sem compreender o assunto. Aos meus amigos LCAD que deram braço e força para o apoio técnico. E a todos os meus amigos que considero como irmãos pela força e motivação.

# RESUMO

Nesse trabalho nós investigamos o emprego do algoritmo A-estrela (A*) com estado híbrido na navegação autônoma de veículos em um espaço tridimensional. Nós modelamos a posição do veículo (origem), o destino e outros pontos de interesse no mundo (estados) como vértices de um grafo. O custo de navegar entre estes vértices foram modelados como arestas do grafo, e uma variante do algoritmo A* (A* com estado híbrido) foi utilizada para escolher o melhor caminho entre a origem e o destino. Para alcançar resultados mais rapidamente evitando obstáculos, utilizamos duas heurísticas combinadas para estimar o custo do vértice atual para o vértice destino: uma sem a limitação de rotação do veículo e considerando apenas os obstáculos, e a sua dual com limitação cinemática $R^3$ desconsiderando os obstáculos.

Nós implementamos a solução de navegação proposta e a incorporamos ao *framework* de robótica CARMEN como um módulo de navegação para veículos autônomos. Nosso módulo interage com outros módulos existentes (módulos de interface com sensores, de mapeamento, de localização, etc.) via troca de mensagens, possibilitando a utilização prática do algoritmo. Resultados de experimentos realizados no IARA (*Intelligent Autonomous Robotic Automobile* – carro de passeio autônomo desenvolvido na UFES) mostraram a viabilidade de utilização do algoritmo tanto na navegação em ambientes estruturados e simples como estradas, quanto em ambientes não estruturados e complexos como estacionamentos e regiões sem pavimento.

# ABSTRACT

In this work, we investigated the use of A-star algorithms (A*) with hybrid state in autonomous navigation of vehicles in a three-dimensional space. We have modeled the vehicle position (origin), the goal and other points of interest in the world (states) as nodes of a graph. The cost of navigating between these nodes were modeled as edges of the graph, and a variant of the A* algorithm was used to choose the best path between origin and goal. In order to be able to avoid obstacles and achieve fast algorithm, we used a combination of two heuristics to estimate the cost of the current node to the goal node: one considering only the obstacles and without the limitation of rotation of the vehicle, and its dual disregarding the obstacles and with limited cinematic $R^3$.

We implemented the proposed navigation solution and incorporated it to the framework of robotics CARMEN as a navigation module for autonomous vehicles. Our module interacts with other existing modules (interface modules with sensors, mapping, localization, etc.) by means of message exchanging. It enables practical use of the algorithm. Results of experiments performed on IARA (*Intelligent Robotic Autonomous Automobile* - autonomous drive car developed in UFES) showed the viability of using the algorithm in simple and structured environments, such as roads, as well as in unstructured and complex environments, such as parking lots and unpaved areas.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# 1 INTRODUÇÃO

Os veículos são de essencial importância para a sociedade. Sua utilização é uma necessidade para a maior parte da população. Essa necessidade proporcionou um aumento demasiado no número de veículos que trafegam nas ruas e, como resultado, a ocorrência de engarrafamentos quilométricos nas grandes cidades. Esse tráfego faz com que os motoristas passem boa parte do dia ao volante, mesmo trafegando em pequenos trechos de estrada. Além disso, o homem, quando ao volante, está sujeito a distrações, ocasionando ineficiência na condução e riscos de acidentes. Mesmo quando o motorista está em pleno comando da situação, ele pode não ser capaz de agir instantaneamente e consequentemente não ser capaz de evitar acidentes.

A preocupação com problemas relacionados ao aumento do número de veículos e a sua má utilização não é recente e foi primeiramente apresentada em [1]. Nesse trabalho o autor enfatiza a importância da existência de veículos autônomos com a evolução da tecnologia, onde estes veículos poderiam ser a chave para um controle desses crescentes problemas. Nos últimos anos, o estudo de veículos autônomos tem recebido muita atenção da comunidade robótica [2] [3] [4], motivados principalmente pela ocorrência de dois grandes eventos: o *Defense Advanced Research Projects Agency* (DARPA) *Grand Challenge* (DGC) [5] e o *Urban Challenge* (DUC) [6] [7].

Como resultados, surgiram diversas soluções para a navegação autônoma, utilizando o modelo cinemático de 4 rodas, denominado *Ackerman* [8]. Essa modelagem possui limitações na movimentação, impossibilitando o veículo que gire em torno do próprio eixo. Consequentemente, essa limitação torna o cálculo da navegação autônoma mais complexo impedindo que algoritmos mais simples sejam utilizados.

Um método muito utilizado na navegação de veículos Ackerman é o algoritmo A* [9] [10]. Esse algoritmo recebe um mapa, a pose inicial do robô e o destino desejado, e gera como saída o percurso da origem ao destino considerando os obstáculos existentes no mapa. O A* tem sido utilizado tanto para a navegação global [11] com o objetivo de achar o menor percurso possível, quanto para a navegação local [2] com o objetivo de desviar de obstáculos. Também é possível a sua utilização no estacionamento [12], quando é considerado o ângulo final do trajeto.

## 1.1 Trabalhos Correlatos

A utilização do algoritmo A* [9] [10] e suas variantes no contexto de planejamento de movimento de veículos Ackerman tem sido objeto de estudos recentes, dado a motivação criada por competições robóticas, como por exemplo a de Veículos Autônomos proposta pela DARPA [5] [6] [7]. Nos próximos parágrafos, é apresentado um resumo dos trabalhos que estão de alguma forma relacionados a essa dissertação.

Começando a versão mais simples do A*, pode-se ver em [13], A* utilização para criar destinos intermediários em um mapa local de pista. Esses destinos são identificados constantemente por sensores, principalmente por meio de visão. O algoritmo planeja destinos intermediários, onde um algoritmo auxiliar reativo se encarrega de cumpri-los. O A* atualiza os destinos intermediários a todo momento com o objetivo de evitar futuras colisões de um obstáculo novo que possa aparecer e manter uma lista de objetivos a ser cumprido para alcançar o objetivo global. A heurística utilizada é baseada somente no mapa de custo, o qual não traz estimativas de custo da limitação cinemática do veículo.

Em [11], é apresentada uma variante do A* denominado Field D*. Este algoritmo permite a geração de caminhos suaves criando caminhos que podem entrar e sair de células em posições arbitrárias. Basicamente, ele muda a maneira com que os estados são extraídos do *grid,* permitindo assim caminhos mais naturais em ambientes discretizados [13]. No entanto, o Field D* é limitado a caminhos lineares entre as células ocasionando muitas restrições no espaço de soluções.

Na dissertação [2], é proposta uma navegação utilizando A* com uma heurística que considera a soma de arcos e segmentos de retas como uma aproximação do movimento Ackerman até o destino. Essa heurística não utiliza informações de obstáculos na estimativa, portanto o seu uso em ambientes complexos ou com destino distante pode ser inviável. Os estados explorados nesse trabalho são podados a partir de um cálculo de distância entre um novo estado e os já explorados.

Em [14], é feito um estudo de um planejamento de movimento utilizando o RRT no contexto do robô IARA. Os autores utilizam *Gradient Field* e o conhecimento sobre a

região de pista para guiar o planejamento. O RRT deste trabalho considera a aceleração do volante e do carro para o planejamento, proporcionando uma navegação mais suave na pista. Porém, o algoritmo RRT não possui limitação dos estados explorados, podendo rodar indefinidamente ou nunca achar uma solução para o problema. O RRT não é algoritmo ótimo, portanto, não há garantia de que um melhor caminho seja encontrado. Contudo, o algoritmo pode proporcionar soluções mais suaves em um tempo viável mesmo em casos que há uma maior dimensionalidade na complexidade do controle.

Como solução para alguns desses problemas, duas variantes mais complexas foram apresentadas, o *anytime* D* [15] e o algoritmo A* de Estado Híbrido [12]. Elas utilizam heurísticas Não-Holonômica sem Obstáculos e Holonômica com Obstáculos para estimar o custo h(.) do A*. Em [15] por exemplo, é utilizado o *anytime* D*, com o objetivo de planejar manobras complexas de estacionamento considerando movimentos *Ackerman.* Esse algoritmo utiliza inicialmente um caminho sub-ótimo que é posteriormente convergido até a solução ótima considerando um espaço de solução com discretização variável. Já em [12], é demonstrado que é possível utilizar o algoritmo A* de Estado Híbrido para o planejamento de movimentos Ackerman. Os estados são associados a um mapa em grid 2D de resolução fixa, que foi a principal base dessa dissertação. O desempenho desses algoritmos ficou comprovado na competição DARPA *Urban* Challenge, dado que os dois primeiros colocados tiveram suas soluções baseadas nelas.

## 1.2 Motivação

Este trabalho está inserido em uma linha de pesquisa do Departamento de Informática da UFES (Linha de Pesquisa em Ciência da Cognição) cujo objetivo de longo prazo é implementar uma arquitetura neural artificial com cognição espacial [16] semelhante, ou até mesmo equivalente, à humana. O grupo de pesquisa envolvido (Grupo de Pesquisa em Ciência da Cognição – GPCG) iniciou seus trabalhos pelo estudo do sentido da visão, por ser o sentido que fornece mais informações sobre o mundo externo. No cérebro de primatas a visão está associada/influencia o padrão de ativação de células

do hipocampo que foram identificadas como células de lugar (*place cells*), isto é, células que ficam ativadas quando o animal está ou olha para um lugar específico do ambiente [17]. Células de outras regiões do cérebro estão ligadas ao planejamento de ações [18]. Assim, a modelagem matemático-computacional dos circuitos neurais dos quais estas células fazem parte pode viabilizar a implementação de sistemas de navegação neurais artificiais para robôs autônomos. Contudo, para sua avaliação, é importante compará-los ao estado da arte em navegação de robôs. Portanto, este trabalho tem como motivação principal, trazer o grupo de robótica da UFES para o estado atual da arte em navegação de veículos Ackerman e possibilitar a comparação entre o avanço de futuros trabalhos considerando a arquitetura neural artificial com o estado da arte.

## 1.3 Objetivo

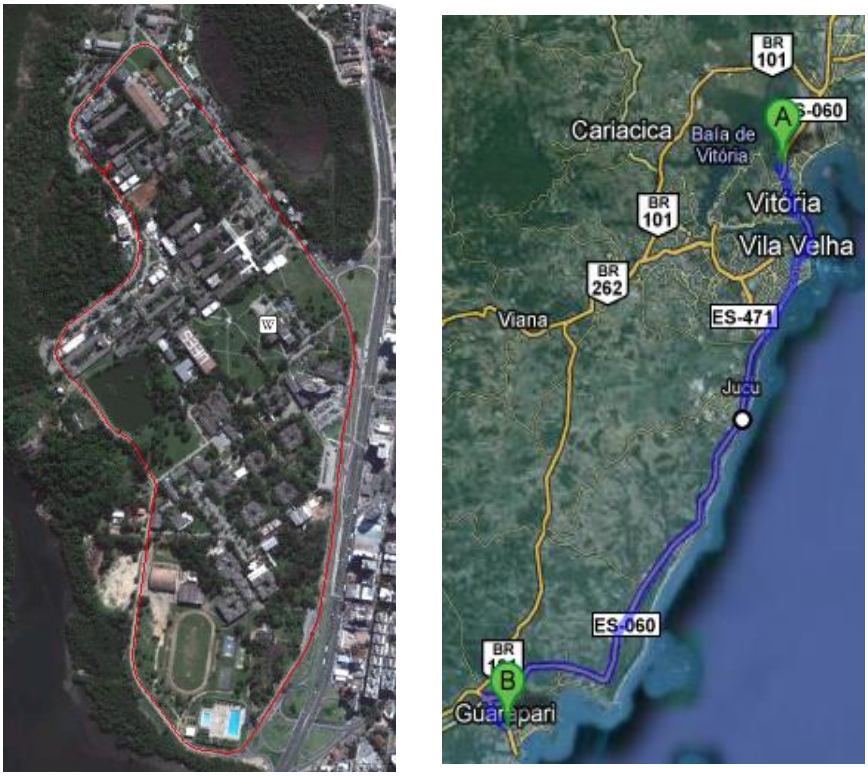
O projeto no qual esse trabalho está inserido possui dois grandes objetivos (realizar uma volta na UFES e realizar uma ida a Guarapari) que definiram os objetivos gerais e específicos dessa dissertação e serão descritos a seguir.

**Objetivos geral:** Investigar de métodos computacionais baseados no algoritmo A* para permitir a navegação autônoma de veículos convencionais.

**Objetivos específicos:**

1. Implementar um método computacional capaz de ser utilizado em ambientes reais de pista e de estacionamento.
2. Contextualizar o algoritmo para o marco da volta da Ufes.

O campus principal da UFES (campus de Goiabeiras) possui um anel viário que o circunda que possui 3.570 metros (Figura 1-a). Na Volta da UFES, o objetivo é desenvolver pesquisas que viabilizem a transformação de um automóvel de passeio em um veículo capaz de realizar a volta pela universidade autonomamente. No caminho da Volta da UFES, as restrições impostas pelas leis de trânsito podem ser, em larga medida, desconsideradas, já que será possível realizá-la em um domingo ou feriado, situações em que o trânsito no anel viário é praticamente nulo. A distância da UFES à cidade de Guarapari, por outro lado, é de 58,5 Km (Figura 1-b).




(a) (b)

**3. Figura 1 - (a) Caminho da Volta da UFES. (b) Caminho da Ida a Guarapari.**

## 1.4 Contribuições

As principais contribuições são:

- Investigação de um algoritmo de navegação para veículos Ackerman que seja capaz de realizar manobras de direção em pista e em estacionamento em tempo real evitando obstáculos.
- Investigação de heurísticas para estimativa de custo de caminho para veículos Ackerman.
- Integração do algoritmo de navegação no framework CARMEN possibilitando o funcionamento em conjunto aos demais módulos de robótica existentes.
- Análise e testes do algoritmo em um ambiente real com um carro híbrido.

## 1.5 Organização do Texto

No Capítulo 2, é apresentado o problema de planejamento de movimento e suas principais definições, assim como as estruturas necessárias para solucioná-lo. No Capítulo 3, é discutida a solução proposta para a o planejamento de movimento e no 4, a metodologia utilizada para a solução e para a avaliação da mesma. No Capítulo 5, os experimentos propostos são apresentados e no capítulo 6 são discutidos os trabalhos correlatos e feita uma análise crítica do trabalho. Finalmente no Capítulo 7, são apresentadas as conclusões e as propostas de trabalhos futuros.

# 2 PLANEJAMENTO DE CAMINHO

Nesse capítulo, será apresentado o problema de planejamento de caminho de veículos. Primeiramente, será feita a definição do problema seguida pela sua formulação matemática e também será apresentado tanto o modelo de movimento do robô quanto o mapa utilizado.

## 2.1 Definição do Problema de Planejamento de Caminho

Uma necessidade fundamental na área da robótica é ter algoritmos que convertam comandos de alto nível de tarefas humanas em descrições de baixo nível de tarefas de movimentação. Os termos *Motion Planning* (Planejamento de Movimento) e *Trajectory Planning* (*Path Planning* - Planejamento de Caminho) são muitas vezes utilizados para esses tipos de problemas [8].

A versão clássica do planejamento de caminho é geralmente mencionado como o *Piano Mover's Problem* (Problema do Movimento do Piano). Suponha que temos um modelo preciso de uma casa e de um piano como entrada para um algoritmo. O algoritmo deve determinar a trajetória do piano de um cômodo para outro da casa sem acertar nenhum obstáculo. Problemas desse tipo são comuns, como por exemplo movimentar um sofá ou um colchão nas escadas de um prédio. Planejamento de caminho geralmente ignora a dinâmica e outras restrições diferenciais e se concentra principalmente nas translações e rotações necessárias para a movimentação.

Na literatura recente, planejamento de caminho, muitas vezes, refere-se à construção de entradas para um sistema que, partindo de um estado inicial, o leve até um estado final especificado. No nosso contexto de Planejamento de caminho, desejamos encontrar um percurso de custo viável partindo da posição atual do robô até um destino final escolhido, isto é, encontrar um caminho obedecendo todas as restrições cinemáticas do robô e evitando colisões com obstáculos.

## 2.2 Modelo Cinemático Ackerman

Para realizar um planejamento de caminho, é necessário conhecer as limitações físicas do deslocamento do robô. Uma modelagem mais simples, portanto mais livre, de um robô é conhecida como diferencial, permitindo que um robô possa girar em torno de seu próprio eixo e seguir qualquer direção sem nenhuma equação que restrinja o seu movimento. Contudo, movimentações Ackerman são mais complexas por exigirem uma equação de movimentação restringindo as posições alcançáveis e aumentando a dimensão da complexidade do cálculo do caminho a ser seguido.

A movimentação de um veículo de 4 rodas é baseado na geometria Ackerman. Embora puramente um modelo cinemático, é uma boa representação para baixas velocidades [8]. A Figura 2 mostra o esquema do veículo e suas principais variáveis. A pose é definida pela posição $(x, y)$ e a rotação $\theta$. Os atuadores da pose são a velocidade $v$, o intervalo de tempo $\Delta t$ e o ângulo $\phi$, que é dado pela média dos ângulos das duas rodas dianteiras do veículo. O único parâmetro é a distância entre os eixos dianteiro e traseiro $L$. Com isso, o modelo cinemático de movimentação do veículo é dado por:

$$x_{t+1} = x_t + \Delta t_t * v_t * \cos(\theta_t) \quad (1)$$

$$y_{t+1} = y_t + \Delta t_t * v_t * \sin(\theta_t) \quad (2)$$

$$\theta_{t+1} = \theta_t + \Delta t_t * v_t * \frac{\tan(\phi_t)}{L} \quad (3)$$

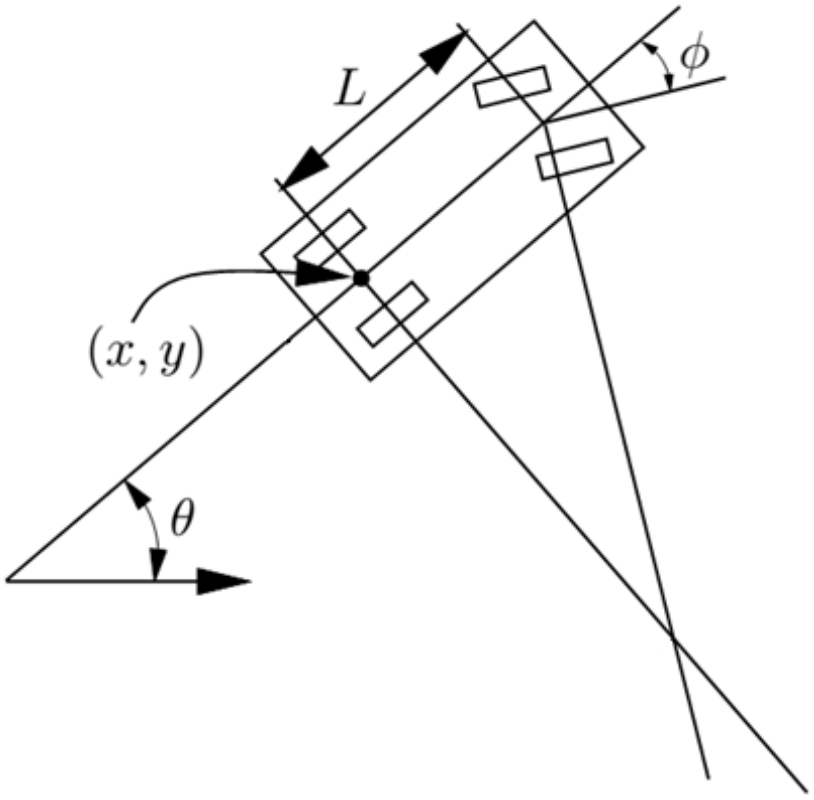



**Figura 2 - Modelo Ackerman, adaptado de [8].**

Com isso, seguindo a notação padrão [19] [15] [12], podemos definir o vetor de estado do veículo como:

$$\boldsymbol{s} = [x, y, \theta]^T \tag{4}$$

Também podemos definir o vetor de controles, que são os atuadores que fazem a transição de um estado para outro:

$$\boldsymbol{u} = [v, \phi]^T \tag{5}$$

A equações discretas (1), (2) e (3) possuem as informações necessárias para permitir a realização da estimativa de movimentação do veículo de um estado inicial $\boldsymbol{s_0}$ a um final $\boldsymbol{s_f}$.

## 2.3 Mapas do Tipo Grid

Para realizar o planejamento de caminho, é também necessária uma representação do mundo. Esta representação é utilizada pelo algoritmo para descobrir onde o robô pode trafegar com segurança e o que é obstáculo. Neste trabalho, o mundo é representado a partir de um mapa de ocupação de *grid* bidimensional.

Para possibilitar a tarefa de planejar o caminho de um robô, é necessário ter conhecimento do ambiente a ser trafegado. Portanto, informações de onde é trafegável (área livre) e onde não é trafegável (obstáculos) são necessárias para escolher um caminho viável para o robô.

Existem diversas soluções para representar um mapa de obstáculos, podendo ser por exemplo um grafo de visibilidade [20], diagrama de *Voronoi* [21] e *grid* de ocupação [22]. A representação do mundo por um *grid* de ocupação é uma solução já utilizada em outros projetos de planejamento que utilizam o algoritmo A* [11] [12] e também é utilizado no projeto do GPCG por isso, é a solução utilizada neste trabalho.

Um *grid* de ocupação representa o mapa *m* como um conjunto de células $m_i$, no qual cada célula $m_i$ representa um espaço do mundo real com uma resolução fixa. Resoluções maiores trazem mais detalhes sobre o mapa, porém podem prejudicar o armazenamento e o custo computacional da navegação.

Cada célula do mapa possui uma probabilidade de ocupação $p(m_i) \in [0, 1]$. Quando uma célula do mapa é desconhecida utilizamos o valor -1 para a sua representação. Dependendo da aplicação, regiões inexploradas podem ser utilizadas como trafegáveis durante o planejamento de caminho, mas a navegação utilizando somente regiões conhecidas é mais segura.

Por ser um mapa probabilístico, um limiar é utilizado para julgar se uma célula $m_i$ será considerada livre ou ocupada. Quanto maior o limiar, maior o risco de passar por um obstáculo, porém valores muito baixos podem levar a ignorar boas soluções.

O mapa em *grid* pode ser gerado a partir de algoritmos de Localização e Mapeamento Simultâneos (*Simultaneous Localization and Mapping* - SLAM) utilizando os dados do sensores do robô.

Um exemplo de representação do mapa de ocupação utilizando *grid* pode ser observado na Figura 3. Probabilidades de ocupação $p(m_i)$ são representadas por escalas de cinza, com cores mais claras representando menores probabilidades e cores mais escuras maiores probabilidades de ocupação da célula. A cor azul representa uma região inexplorada, ou seja, não se tem conhecimento da sua ocupação.

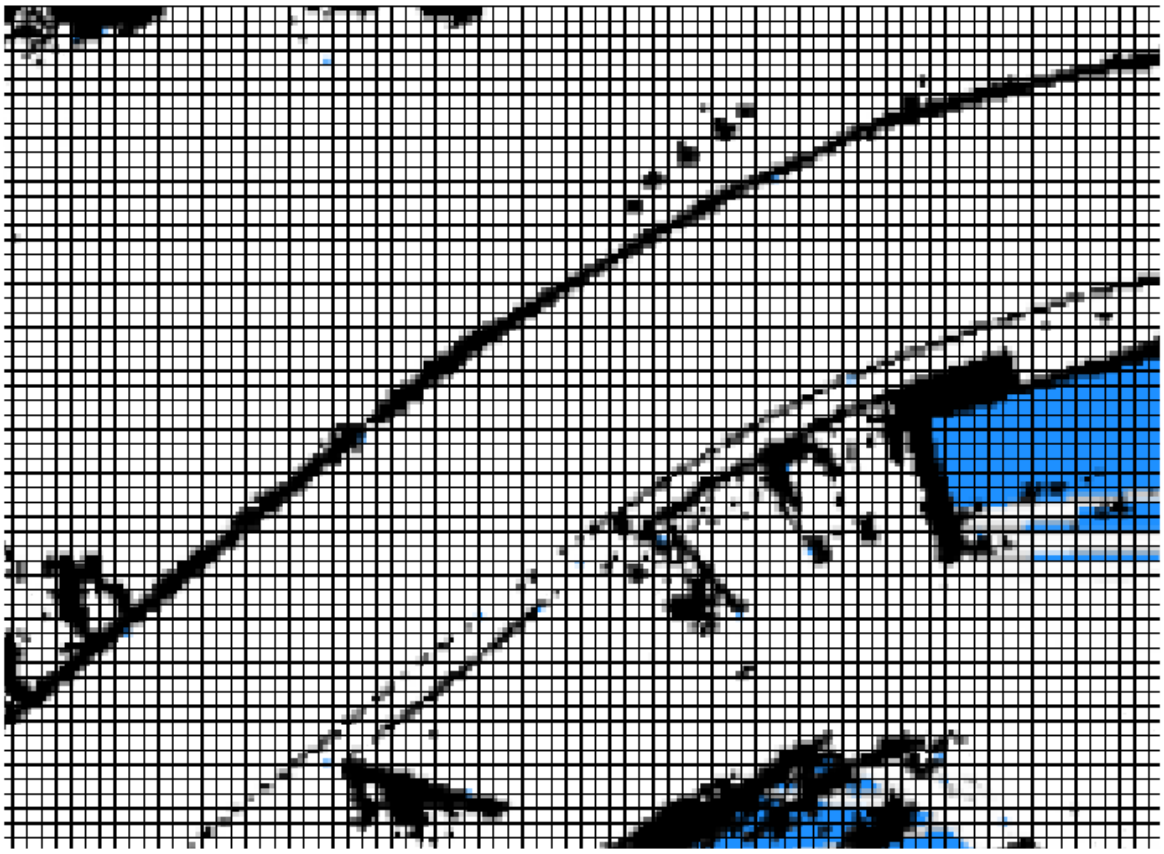


**Figura 3 – Grid de ocupação.**

## 2.4 Algoritmo A*

Em grafos com custos crescentes com a distância, o algoritmo de busca em largura produz um caminho mais curto de um vértice origem a todos os outros. Porém, ele necessita visitar todos os vértices do problema, o que pode ser custoso e inviável em casos de maior complexidade, como por exemplo um grafo muito esparso. Em uma busca em grafo, é desejável otimizar de modo que seja visitado o menor número possível de vértices, tornando a busca mais eficiente.

Busca em largura e busca em profundidade são algoritmos em que a busca se move através do grafo sem qualquer preferência ou influência de onde o objetivo está localizado. Mas se coordenadas do objetivo em um grafo em malha são conhecidas, então essa informação pode ser utilizada para ajudar a decidir qual vértice será visitado para chegar ao objetivo.

Embora possamos ter alguma informação, o melhor que se pode fazer é definir uma heurística para estimar o custo para o vértice objetivo [23]. Por exemplo, um algoritmo de busca pode escolher como seu próximo vértice aquele que tem uma menor distância euclidiana para o destino, pois tal vértice tem possibilidade maior, com base em informações locais, de ficar mais próximo do objetivo. No entanto, não há nenhuma garantia de que esse vértice leve ao caminho mais curto, é apenas uma boa estimativa. Estas boas estimativas são baseadas na melhor informação disponível para a busca.

Uma forma possível de se fazer uso dessa informação é utilizar o algoritmo A* (A-estrela) [9] [10], que permite fazer uma busca num grafo de forma eficiente seguindo uma heurística escolhida. Se a heurística é boa, então a busca é eficiente. Se a heurística é ruim, embora o caminho possa ser encontrado, sua pesquisa provavelmente vai demorar mais tempo do que necessário e possivelmente retornar um caminho subótimo. O A* irá produzir um caminho ideal se a sua heurística é otimista. Uma heurística otimista, ou admissível, sempre retorna um valor inferior ou igual ao custo do caminho mais curto a partir do vértice atual para um vértice destino do grafo. Por exemplo, em um grafo em malha, uma heurística admissível poderia ser a distância euclidiana para o objetivo, pois esta distância sempre é menor ou igual à menor distância possível em uma malha. Uma breve descrição desse algoritmo é apresentada a seguir.

O A* tem uma fila de prioridades. Esta prioridade é determinada pela soma da distância a partir do vértice de início para o vértice atual e a estimativa do atual para o destino. Tomando a Figura 4 como exemplo, o primeiro vértice a ser colocado na fila de prioridade é naturalmente o vértice INICIO (*Start*). Em seguida, ele é expandido colocando todos os vértices adjacentes na fila de prioridade de forma ordenada, Figura 5 (esquerda). Uma vez que o vértice B tem a maior prioridade, pois tem o menor custo, ele é expandido na próxima iteração, ou seja, é retirado da fila e seus vizinhos são

adicionados como pode ser visto na Figura 5 (centro). Note que apenas vértices que não foram visitados são adicionados à fila de prioridade, ou seja, não é adicionado novamente o vértice de início.

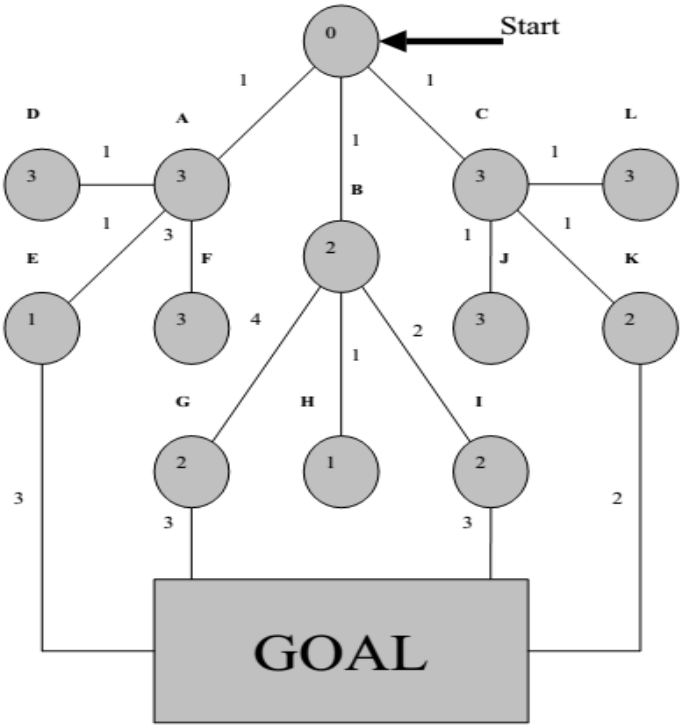



**Figura 4 – grafo de exemplo do A*. Os valores nas arestas representam o custo de caminhar entre os vértices adjacentes e os valores nos vértices representam a estimativa de custo até o destino [23].**

Em seguida o vértice com maior prioridade é expandido, isto é, vértice H. Ele é retirado da fila e todos os seus vizinhos são adicionados. No entanto, H não tem vizinhos, portanto, nada é adicionado à fila. Uma vez que não são acrescentados novos vértices, nenhuma ação ou expansão é associada com o nó H, Figura 5 (direita). Depois, o vértice com maior prioridade é retirado, ou seja, o nó A, e é expandido, adicionando todos os seus vizinhos adjacentes para a fila de prioridade. Por fim, o vértice E é expandido, o que nos dá um caminho para o objetivo com um custo total de 5. Note que esse custo é o custo real, isto é, a soma dos custos do início para o destino. Neste ponto, há vértices na fila de prioridade, que têm um valor de prioridade maior ou igual ao custo para o destino. Uma vez que estes valores de prioridade são limites inferiores de custo de caminho para o destino, todos os caminhos através desses vértices terão um maior custo que o caminho já encontrado. Portanto, esses vértices podem ser descartados.

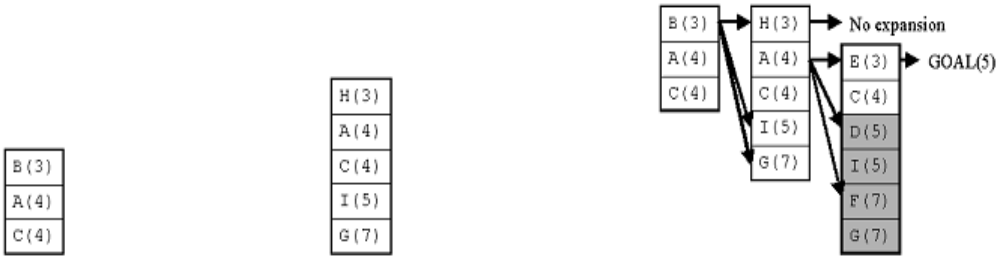



**Figura 5 – Na esquerda Lista de prioridade após Start ter sido expandido, no meio Lista de prioridade após B ter sido expandido e na esquerda três iterações do exemplo e a organização da lista de prioridade [23].**

O caminho explícito através do grafo é representado por uma série de arestas de retorno. Eles representam a história imediata do processo de expansão. Assim, as arestas de retorno de A, B e C, apontam para o INICIO. Da mesma forma que as arestas de D, E, e F apontam para A. Por fim, a aresta de retorno do vértice de destino leva à E. Portanto, o caminho do INICIO ao DESTINO (*Goal*) pelas arestas de retorno é: A, E, e DESTINO. As setas na Figura 6 apontam no sentido inverso das arestas de retorno.

Mesmo que um caminho para o objetivo tenha sido determinado, o A* não está acabado porque pode haver um melhor caminho. No A* isso ainda é possível porque na fila de prioridade ainda há vértices cujos os valores são menores do que o do estado DESTINO encontrado. A fila de prioridade neste ponto apenas contém o nó C com custo menor que 5 que é então expandido adicionando nós J, K e L para a fila de prioridade (Figura 6). Nós podemos remover imediatamente J e L, porque seus valores de prioridade são maiores ou igual ao custo do caminho mais curto encontrado até o momento. O vértice K é então expandido encontrando o DESTINO com um custo inferior ao caminho previamente encontrado através do nó E. O caminho INICIO, C, K e DESTINO torna-se o melhor caminho atual. Uma vez que neste ponto a fila de prioridade não possui quaisquer elementos, cujos valores são menores do que o do nó objetivo, este caminho resulta no ótimo.

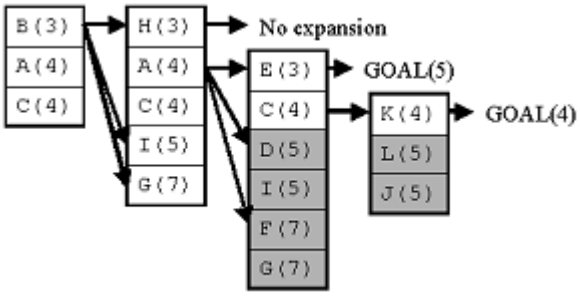



**Figura 6 – Lista de prioridade de quatro iterações do exemplo [23].**

Definindo formalmente o algoritmo A*, temos como entrada o grafo, onde os vértices podem ser associados com os espaços livres de um mapa em *grid* do robô. Arestas correspondem aos vértices adjacentes, células vizinhas no caso de *grid*, e têm valores correspondentes ao custo necessário para percorrer entre eles. A saída do algoritmo A* é uma sequência de vértices a partir do destino até o início.

Para a definição formal são utilizadas duas estruturas de dados adicionais, um conjunto de vértices abertos $O$ e um conjunto de vértices fechados $C$. O conjunto de vértices aberto $O$ é a fila de prioridade e o conjunto de vértices fechado $C$ contém todos os vértices já processados. Também podemos definir que:

- $Star(n)$ representa o conjunto de vértices adjacentes a $n$.
- $c(n_1, n_2)$ é o custo da aresta que liga $n_1$ a $n_2$.
- $g(n)$ é o custo total conhecido do vértice inicial $q_{start}$ até o $n$.
- $h(n)$ é a função heurística de custo do vértice $n$ até o vértice de destino $q_{goal}$.
- $f(n) = g(n) + h(n)$ é o custo estimado do caminho mais curto de $q_{start}$ até $q_{goal}$ passando por $n$.

Como pode ser visto no algoritmo A* da Figura 7, o algoritmo recebe como entrada o grafo onde se deseja achar o caminho mínimo ponto-a-ponto. A lista $O$ de vértices abertos inicialmente possui apenas um vértice inicial de onde a exploração é iniciada e a lista de vértices fechados $C$ é vazia.

**Algoritmo A***

**Entrada:** Um grafo
**Saída:** Um caminho do nó inicial ao final

1: **repita**
2: Pegue $n_{best}$ de $O$ em que $f(n_{best}) \leq f(n), \forall n \in O$.
3: Remova $n_{best}$ de O e adicione em C.
4: Se $n_{best} = q_{goal}$, SAIR.
5: Expandir $n_{best}$: para todo $x \in Star(n_{best})$ que não está em $C$.
6: **Se** $x \notin O$ **então**
7: add $x$ to $O$.
8: **senão se** $g(n_{best}) + c(n_{best}, x) < g(x)$ **então**
9: Atualize o backpointer de $x$ para apontar para $n_{best}$
10: $g(x) = g(n_{best}) + c(n_{best}, x)$
11: **fim-se**
12: **Até que** $O$ seja vazio

**Figura 7 – Pseudo-código do algoritmo A*, adaptado de [23].**

Uma iteração do algoritmo inicia extraindo o vértice $n_{best}$ com a melhor função objetivo $f(n)$ da lista $O$ e o adiciona em $C$. Caso o vértice extraído seja $q_{goal}$, o algoritmo é encerrado retornando o caminho até ele; caso contrário, continua. Os vizinhos $x$ de $n_{best}$ que estão na lista $C$ são descartados, pois já foram explorados. Os vizinhos que não estão em $C$ e nem em $O$ são adicionados em $O$, pois é um caminho ainda desconhecido. Caso contrário eles já existem na lista $O$. Neste caso, é feita a verificação $g(n_{best}) + c(n_{best}, x) < g(x)$. Caso essa afirmação seja verdadeira, o caminho até $x$ passando por $n_{best}$ tem um custo menor que o contido na lista, $O$, e o vértice $x$ é atualizado para passar por $g(n_{best})$. O seu valor total de custo também é atualizado com $g(n_{best}) + c(n_{best}, x)$. Os demais vizinhos de $g(n_{best})$ que não atendem a esse critério são descartados.

## 2.5 Limitações do Algoritmo A*

O algoritmo A* pode ser utilizado em diversas aplicações de planejamento de caminho, sendo estas tanto para modelos holonômicos quanto não-holonômicos. O algoritmo, quando implementado em um robô holonômico, possibilita utilizar o próprio mapa em *grid* como um conjunto finito de soluções, pois se adotado o centro de uma célula como vértice é possível alcançar qualquer vértice da vizinhança realizando movimentos de rotação e translação até o seu centro.

Porém, movimentações Ackerman são classificadas como não-holonômicas e, portanto, possuem restrições nas movimentações, impedindo a garantia de que o centro de uma célula do mapa em *grid* seja alcançável pelos estados do algoritmo A* convencional. Os estados poderiam simplesmente não serem discretizados, porém, sem uma modelagem discreta dos estados, a movimentação passaria a ter infinitas soluções. A falta de discretização dos estados do A* prejudica o seu desempenho já que passa a não ter um critério que indique se um estado já foi visitado anteriormente, se foi explorado e já está na lista ordenada ou se ainda é um caminho inexplorado.

Outro problema é o consumo de memória, já que para cada iteração do algoritmo todos os estados vão ser adicionados na lista de abertos, criando uma árvore de estados que pode crescer indefinidamente sem haver nenhuma poda. Com infinitas possibilidades de estados, o critério de parada se torna impossível de ser atingido, fazendo com que o algoritmo entre em *loop* nesses casos.

# 3 PLANEJAMENTO DE CAMINHO DO ROBÔ IARA UTILIZANDO O A* DE ESTADO HÍBRIDO

Neste capítulo, abordaremos os detalhes da solução utilizada no planejamento de caminho do carro robô denominado IARA (*Intelligent and Autonomous Robotic Automobile* – carro de passeio autônomo desenvolvido na UFES). Na Seção 3.1 serão apresentados os aspectos gerais do funcionamento do algoritmo A* de Estado Híbrido do IARA. As heurísticas utilizadas são descritas em 3.2 e em 3.3 é descrito um método auxiliar utilizado para interligar o último estado explorado ao objetivo em áreas livres.

## 3.1 Algoritmo A* de Estado Híbrido

Uma variante do algoritmo A* denominado A* de Estado Híbrido (*Hibrid-state* A*) [12] aplica ao espaço cinemático do veículo uma regra modificada de atualização de estados. Nesse modelo, o espaço de busca é discretizado e um grafo é associado ao mapa de *grid*. Enquanto o A* tradicional armazena os estados explorados numa dimensão equivalente à movimentação do robô, este algoritmo associa cada vértice do grafo de exploração à um espaço cinemático contínuo do veículo. No quadro esquerdo da Figura 8 pode ser visto um exemplo de exploração do A* tradicional onde cada estado explorado possui apenas uma possibilidade de solução numa célula do A*. Contudo no A* de Estado Híbrido representado no quadro direito da Figura 8, cada vértices do algoritmo pode representar diversas soluções, inclusive numa dimensão superior do que a utilizada na exploração, no nosso caso o estado $R^3$ do robô é associado à vértices de exploração 2D.

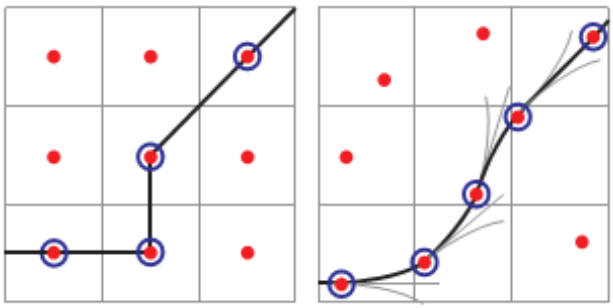


**Figura 8 - Comparação gráfica de algoritmos de busca. Esquerda: A* com custo associados aos centros da célula. Direita: A* de estado híbrido associa um estado contínuo, com cada célula, adaptado de [12].**

A exploração do algoritmo utiliza o modelo *Ackerman* descrito no tópico 2.2, com a diferença que a relação $v * \Delta t$ é simplificada para uma distância fixa $d$, sendo assim o comando $\boldsymbol{u}$ descrito como $[d, \phi]^T$.Com o intuito de obter uma navegação mais suave, um custo maior é definido para estados que se mantenham no sentido de ré, que invertam o sinal da velocidade ou mudem o ângulo do volante. A descrição detalhada da busca pelo caminho é feita a seguir.

Na Figura 9, está representado o funcionamento de cada iteração do algoritmo, que, inicialmente, recebe como entrada a posição inicial de onde vai ser efetuada a busca, o destino a ser alcançado, o mapa de *grid* e os mapas de custos das heurísticas utilizadas. O estado $\boldsymbol{s}$ inicial está associado com o vértice de pesquisa inicial do algoritmo. Sua expansão (exploração da vizinhança) é feita com o modelo cinemático Ackerman, utilizando o comando $\boldsymbol{u} = [d, \phi]^T$, onde $\phi$ é a direção do volante e $d = v * \Delta t$ é um deslocamento de movimento com velocidade positiva (frente) quanto negativa (ré). A cada expansão temos 6 novos movimentos a serem avaliados: 3 posições diferentes para a roda (centrada, máximo para a direita e máximo para a esquerda) combinadas com as velocidades positivas e negativas, isto é, frente e ré, esta exploração é exemplificada na Figura 10, onde o carro ao centro representa o estado de origem e os 6 outros carros representam os destinos gerados pelos movimentos. Em seguida, os estados explorados são avaliados, obtendo a função de custo $f = g + \mathrm{h}$, onde o custo $g$ é calculado pelo deslocamento $d$ e a estimativa $\mathrm{h}$ é obtida a partir de duas heurísticas sendo elas Não-Holonômica sem Obstáculos e Holonômica com Obstáculos. Para cada um dos estados avaliados, é feita a sua associação à uma célula do *grid*. São descartados os estados associados ao vértice contido na lista $C$ ou na lista $O$ que possuem um custo $g(.)$ menor ou igual do que o encontrado no estado avaliado. Os estados avaliados que não foram descartados são atualizados como pertencentes aos vértices que foram associados. Em seguida a lista $O$ é atualizada com estes. Após a lista $O$ ser atualizada é extraída dela o estado $\boldsymbol{s}$ com o melhor valor $f(.)$, onde se inicia uma nova iteração do algoritmo. Esse ciclo se repete até que o objetivo seja alcançado.

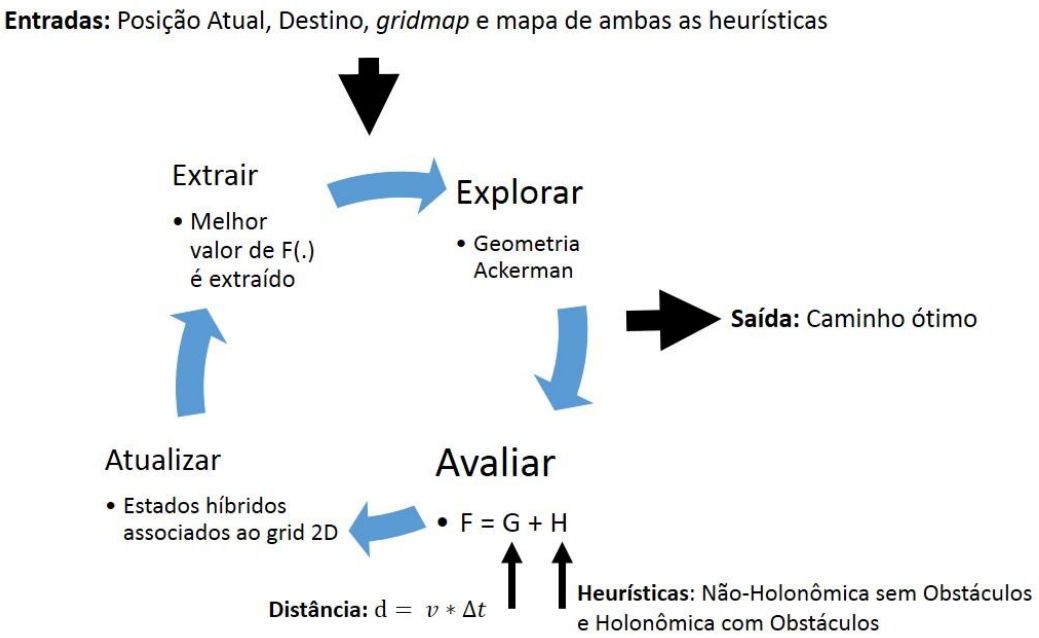



**Figura 9 - Esquematização do Algoritmo A* de Estado Híbrido.**

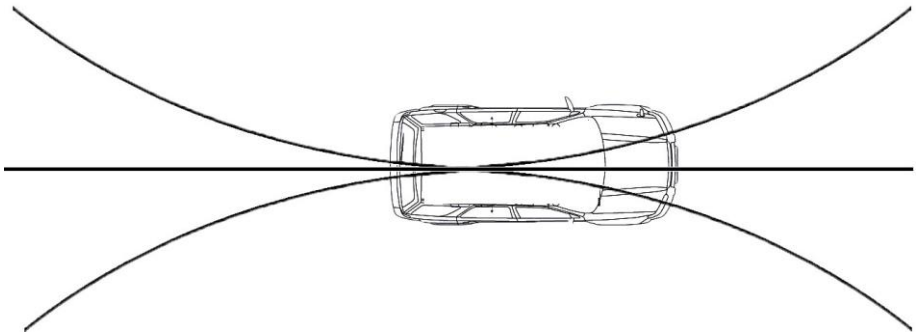


**Figura 10 – Modelo de exploração dos estados do Veículo.**

Uma demonstração do funcionamento da exploração do algoritmo pode ser observada na Figura 11. Para facilitar o entendimento, o peso das heurística não foi considerado, fazendo com que o algoritmo se comporte semelhante à busca em largura. As arestas representam a possível transição de um estado através do movimento Ackerman. Os carros representam os estados fechados e que, consequentemente, tiveram a sua vizinhança explorada. A malha ao fundo da figura representa os estados Híbridos que estão limitados a células 2D da figura. Cada espaço 2D representado pode ter apenas um estado associado. Caso haja mais de um movimento que seja incidente a um mesmo espaço, como acontece nos espaços representados em amarelo, apenas o que contenha o menor custo $f(.)$ é considerado, descartando estados com solução

inferior que tente explorar o mesmo espaço mesmo que possua uma orientação diferente do melhor estado incidente na célula. É evidente que a hibridização dos estados limita o número de possíveis soluções melhorando o desempenho do algoritmo e evitando loops, pois cada célula do espaço tem conhecimento de qual estado está associado a ela.

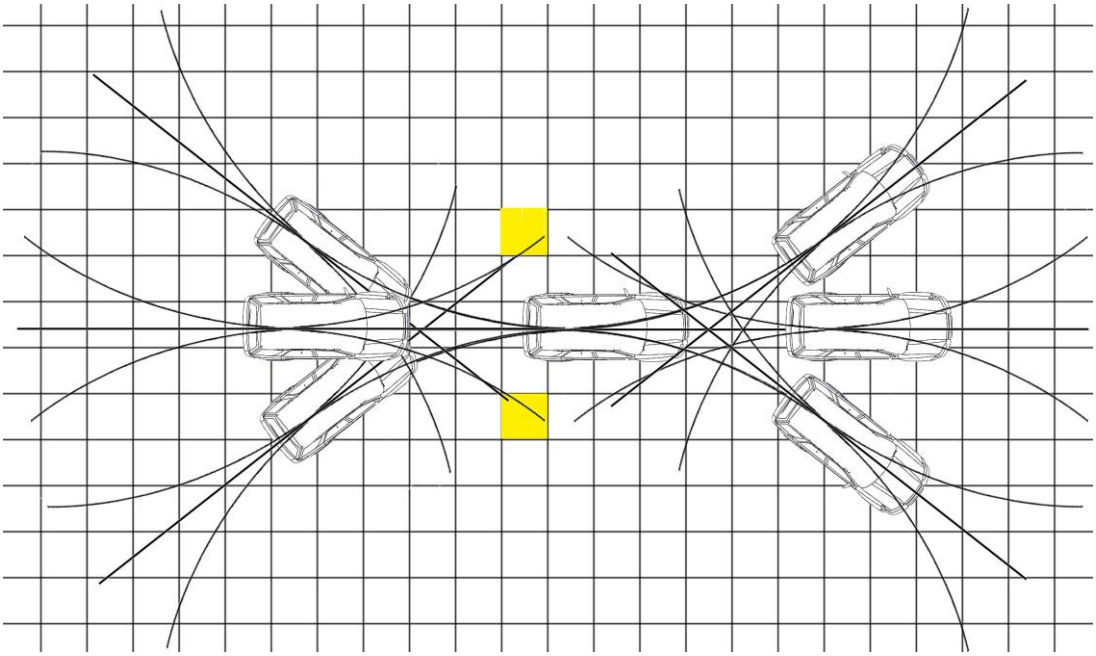


**Figura 11 - Exemplo de exploração do algoritmo A* de Estado Híbrido sem a utilização de heurísticas.**

Devido à discretização dos controles e do tempo e à simplificação das possíveis soluções dos estados, o A* de estado híbrido não garante que encontrará a solução ótima real para o problema. No entanto, o caminho resultante é garantido ser cinematicamente viável. Em [12] o autor descreve que em resultados empíricos o algoritmo sempre encontra uma solução em ambientes realistas e este resultado encontra-se tipicamente na vizinhança da ótima global.

O algoritmo A* de Estado Híbrido foi escolhido como solução do planejamento de caminho do IARA por se adaptar ao tipo de mapa de obstáculos utilizado (mapa em grid) e representar o estado da arte [12], obtendo bons resultados na competição DARPA *Urban Challenge* [6] [7].

## 3.2 Heurísticas do A*

A utilização de uma boa função de estimativa de custo $f(.)$ tem um papel fundamental para a eficiência do algoritmo A*. Nós utilizamos duas heurísticas para formar essa função, uma denominada Heurística Não-Holonômica sem Obstáculos e outra denominada Heurística Holonômica com Obstáculos. É importante mencionar que essas heurísticas não dependem de quaisquer propriedades do A* de estado híbrido e são portanto aplicáveis a outros métodos, como por exemplo, o A* convencional. A descrição detalhada de cada uma delas é apresentada nas subseções a seguir.

### 3.2.1 Heurística Não-Holonômica sem Obstáculos

A heurística denominada não-holonômica sem obstáculos, ignora obstáculos, mas leva em conta a natureza não-holonômica do carro. Para criar esta heurística, calculamos o caminho mais curto para o destino de todos os pontos no espaço $R^3$ $(x, y, \theta)$ assumindo um ambiente livre de obstáculos. O mapeamento é feito com um algoritmo Dijkstra em um grafo de malha $R^3$ representando posição em x, posição em y e o ângulo de orientação do veículo $\theta$. Os mapas foram computados utilizando uma resolução de 0,1 metros para o espaço euclidiano 2D e $5^o$ para a orientação do veículo.

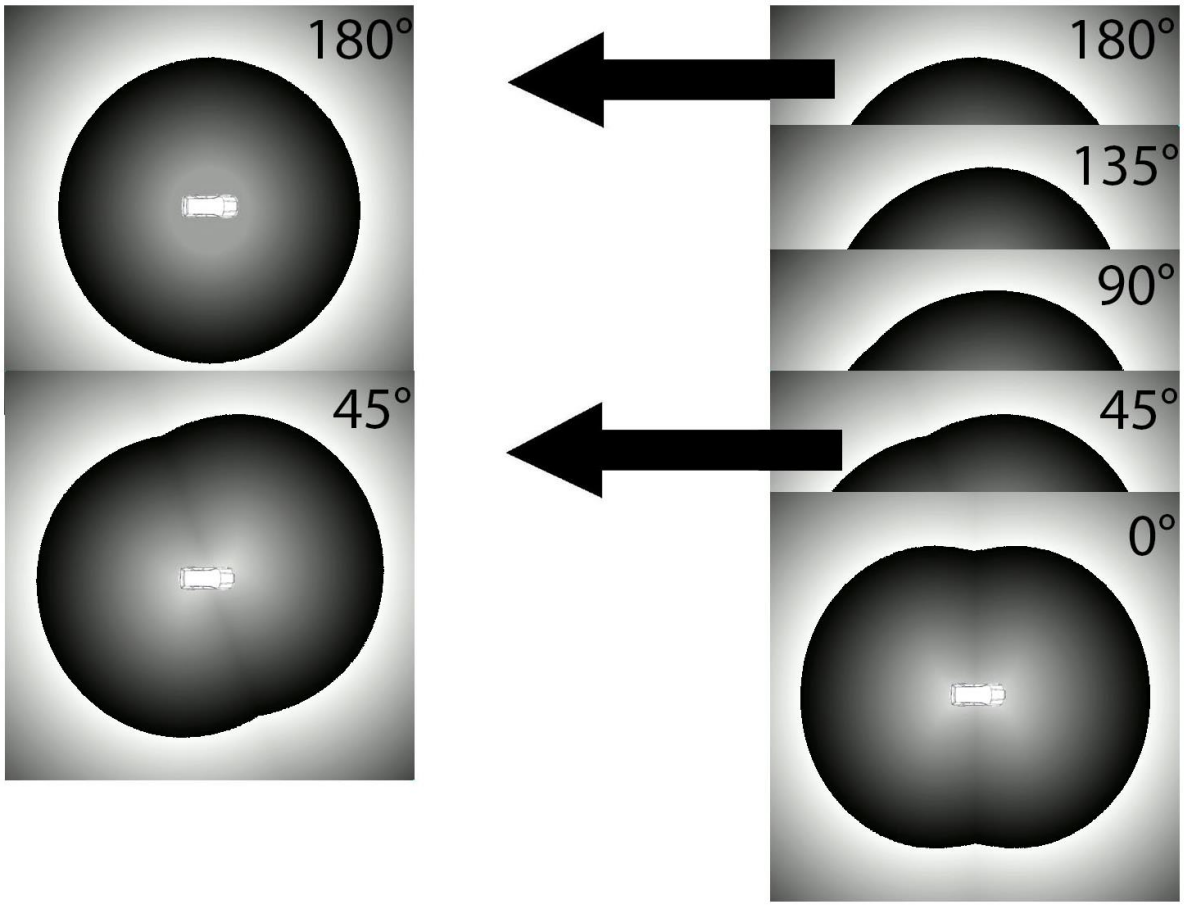



**Figura 12 – Mapas de custo da heurística não-holonômica sem obstáculos. Os tons de cinza de claro ao escuro representam respectivamente um valor de custo mais baixo ao mais alto de sair do estado $s = (\mathbf{0}, \mathbf{0}, \mathbf{0})$ até a qualquer estado computado.**

Um exemplo visual do mapa de custo do algoritmo é apresentado na Figura 12 onde cada mapa 2D representa o custo de sair da posição representado pelo carro dentro do mapa, de valor $s = (0,0,0)$ para qualquer posição 2D com a orientação $\theta$ indicada no canto superior direito do mapa. Os tons de cinza mais claros aos mais escuros representam respectivamente os de menor custo aos maiores. É possível notar por exemplo que a dificuldade de alcançar uma determinada posição no espaço 2D varia de acordo com a orientação final que se deseja alcançar. Também é possível observar que o custo só é equivalente à distância euclidiana quando o objetivo tem a mesma orientação e a posição coincide com o movimento reto do robô como pode ser visto na Figura 13 à esquerda, em outros casos há necessidade de movimentos mais complexos para posições próximas como uma pose sob a mesma orientação na lateral como pode ser visto na Figura 13 à direita, onde o movimento necessário e, consequentemente o custo, são maiores mesmo para uma distância aparentemente menor que a do quadro esquerdo.

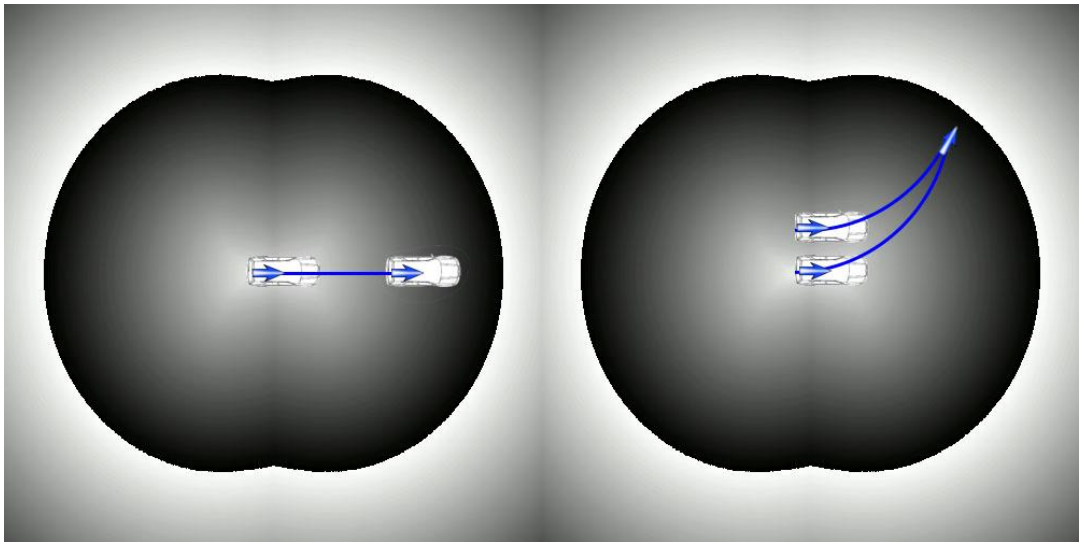


**Figura 13 - Exemplo de custo estimado para posições com orientação $\theta = 0$. Quadro a esquerda: custo para uma posição à frente, quadro a direita: custo para uma posição lateral.**

A principal vantagem desta heurística é a atribuição de altos custos para caminhos que se aproximam do objetivo com a orientação $\theta$ errada o que não ocorreria simplesmente coma distância euclidiana. Contudo, a heurística possui alto custo computacional, pois necessita que seja computado o caminho mínimo para todos os vértices de um grafo demasiadamente grande. Pelo fato da heurística não depender de informações geradas em tempo de execução, ela pode ser totalmente pré-computada off-line, viabilizando a sua utilização. Esta heurística proporciona melhorias significativas no número de nós expandidos em relação ao custo de uma heurística baseada simplesmente em distância euclidiana, pois a distância euclidiana não traz nenhuma informação das limitações de orientação existentes no modelo Ackerman. A heurística não-holonômica sem obstáculos é certamente uma heurística admissível, pois aproxima-se do custo real do caminho ótimo do veículo em um espaço livre sem fazer super estimativas.

### 3.2.1 Heurística Holonômica com Obstáculos

A segunda heurística, holonômica com obstáculos, ignora a natureza não holonômica do carro mas usa o mapa de obstáculos para calcular a menor distância para a meta, realizando o caminho mínimo 2D do mapa em *grid*, sendo assim complementar da primeira apresentada. Uma representação do mapa de custo dessa heurística pode ser vista na Figura 14 onde cores do tom de cinza do escuro ao claro representam respetivamente os custos mais altos e mais baixos para chegar ao destino.

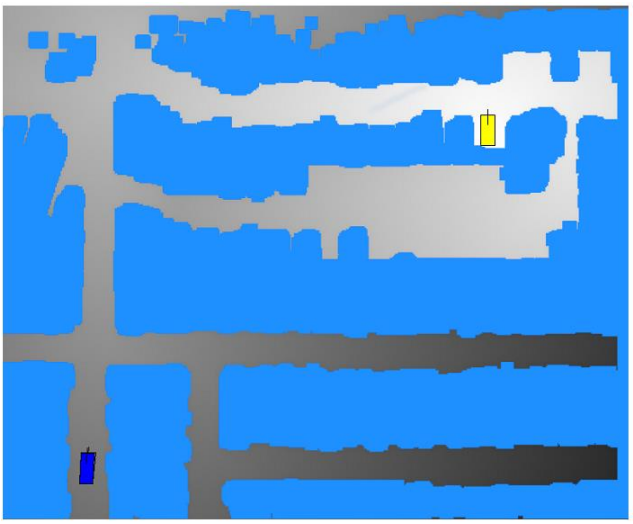


**Figura 14 - Mapa de Custo gerado pelo *Gradient Field.* Os tons de cinza de claro ao escuro representam respectivamente um custo menor ao maior para alcançar o objetivo. Regiões em azul representam ambiente não trafegável.**

O Mapa de Custo de Gradiente é gerado a partir do algoritmo *Gradient Field* que faz parte do planejador de caminho *Gradient Method* [24]. Ele apresenta um planejador de caminho em tempo real para robôs diferenciais. O mapa de custos do gradiente é construído baseado na seguinte equação:

$$F(P) = \sum_i I(p_i) + \sum_i A(p_i, p_{i+1}) \tag{6}$$

onde, I e A são funções arbitrárias do caminho discreto. Tipicamente, $I$ irá representar o custo de percorrer através de uma determinada posição, e será definido de acordo com as características do domínio. No nosso caso ele é associado a um custo proporcional à proximidade com um obstáculo. Outras possibilidades é ter custos mais elevados para regiões desconhecidas, regiões escorregadias e etc. O comprimento do percurso pode ser tomado em conta através da atribuição de $A$ que é proporcional à distância euclidiana que o robô viaja entre os dois pontos. A soma de $A$ então dá um custo proporcional ao comprimento do percurso.

A construção do mapa de custos dessa heurística funciona da seguinte forma: inicialmente, é atribuído ao destino o custo $C$ igual a 0, e a todos os outros vértices um custo $C$ infinito. O destino é então colocado numa lista de ativos. A cada iteração do algoritmo, é realizada uma atuação em cada ponto da lista ativa, retirando este ponto e atualizando o custo $C$ de seus vizinhos. Considere um ponto $p$ arbitrário que acabamos

de atribuir um valor. Este ponto é rodeado por oito vizinhos sobre uma malha regular, ilustrado na Figura 15. Para qualquer um dos vizinhos $q$, podemos calcular o custo para estender o caminho de $p$ para este ponto, usando a métrica de custo $F(P)$. Se o novo custo em $q$ é menor do que o seu próprio custo, então ele é substituído, e adicionado na lista de ativos. Por exemplo, na Figura 15, o novo valor de custo $C$ para o vizinho diagonal $q$ é o custo $C$ de $p$ mais $I$ e $A$, ou seja, $20 + 14 + 5 = 39$, que é inferior ao seu custo de $q$ anterior $40$. Desse modo, o seu custo é reduzido e ele é colocado na lista de ativos. O processo se repete até que a lista de ativos torna-se vazia.

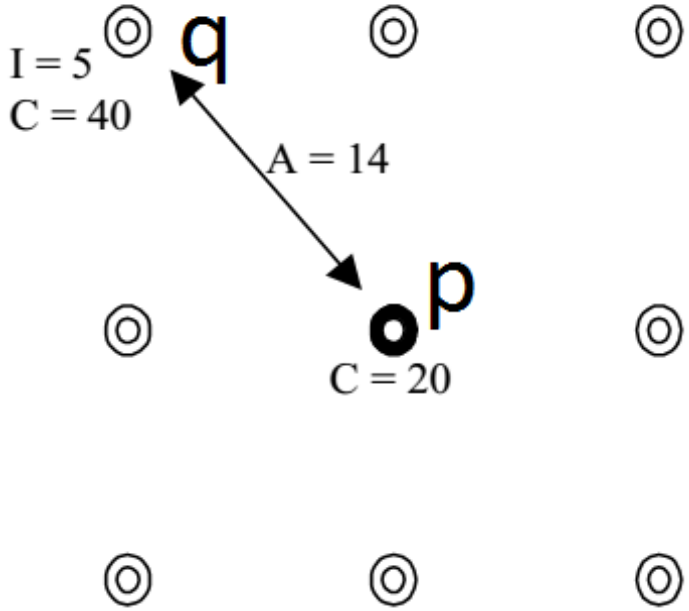



**Figura 15 – Exemplo de atualização de um vizinho.**

### 3.2.2 Utilização das heurísticas

Por ambas as heurísticas serem admissíveis, elas garantem ser sempre ou iguais ou inferiores ao custo real. Utilizando o valor máximo entre as duas temos a possibilidade de melhorar o desempenho do algoritmo pois, quando o destino estimado está muito longe a heurística que considera os obstáculos, holonômica com obstáculos, provavelmente terá um custo mais próximo do real, principalmente em regiões com características de labirinto. Por outro lado, quando o destino a ser calculado está próximo e

com orientação diferente, a heurística não-holonômica sem obstáculos pode representar um custo mais próximo do real, pois, apesar de não considerar os obstáculos ao seu redor, ela considera o custo da manobra necessária para alcançar o objetivo com a orientação correta.

## 3.3 Conexão de um estado ao destino em regiões livres

Em situações em que não há obstáculos de um determinado estado ao destino, um caminho pode ser obtido de maneira menos custosa utilizando curvas *Reeds-Shepp* [25]. Este método é composto por no máximo 5 movimentações, e cada movimentação possui seis possíveis configurações, sendo elas: RETO, DIRETA e ESQUERDA, tanto para frente quanto para a ré. Os movimentos para a direita e para esquerda nesse método utilizam o ângulo máximo possível das rodas. O método possui um total de 48 possíveis caminhos para uma dada configuração de origem e destino, conseguindo achar sempre um caminho quando não há obstáculos no trajeto.

O método por apresentar um baixo custo computacional, pode ser testado frequentemente num estado atual do A* de Estado Híbrido, principalmente quando há proximidade com o objetivo. Devido a discretização do espaço de soluções, a conectividade entre o estado final do ambiente discreto e o estado final real sempre se dará por meio desse método, caso contrário o planejador de caminho não poderá garantir que o objetivo final exato será alcançado.

# 4 METODOLOGIA

Neste capítulo, abordaremos os detalhes da metodologia utilizada. Na Seção 4.1, será apresentado o *framework* de robótica onde a contribuição foi implementada. A arquitetura de todo sistema e suas funções são descritas em 4.2. Na Seção 4.3 serão apresentados os detalhes do veículo robô IARA. Em 4.4 são apresentados os parâmetros do algoritmo. Em 4.5 e 4.6 os métodos de comparação e o percurso utilizado, respectivamente e em 4.7 as métricas para os testes do algoritmo.

## 4.1 Framework CARMEN

Desenvolvido em 2002, CARMEN é um sistema de controle robótico aberto GPL (http://carmen.sourceforge.net), que fornece uma interface consistente e um conjunto básico de primitivas para a pesquisa em robótica, permitindo a abstração de boa parte da implementação necessária, principalmente ao que tange a comunicação e cooperação entre diversos módulos. Os principais objetivos do CARMEN são a diminuição da barreira para a criação de novos algoritmos robóticos e a facilitação do compartilhamento de pesquisas e algoritmos entre diferentes instituições [26]. Esse sistema foi projetado sobre uma arquitetura modular, orientada a serviços (SOA - *Service Oriented Architecture*). Ele fornece algoritmos básicos para construção de mapas, localização e navegação em uma mesma interface [27]. Além disso, ele foi desenvolvido atendendo às seguintes características:

**Usabilidade** – A quantidade de tempo necessária para aprender a utilização do sistema e a sua modificação para atender novos requisitos é pequena e viável.

**Extensibilidade** – Novos módulos implementados devem ser fáceis de escrever, tendo os módulos centrais como uma base sólida para o desenvolvimento de funcionalidades de nível superior. Portanto também é facilitada a substituição de módulos já existentes.

**Robustez** – Os programas devem ser robustos à uma série de falhas, incluindo problemas com comunicação e falhas de outros programas ou módulos.

O CARMEN foi desenvolvido com uma arquitetura de software modular onde cada nova funcionalidade é criada como um módulo separado. Essa modularidade traz vários benefícios importantes, como a flexibilidade, que podem ser combinadas diversas bases para diferentes tipos de robôs e sensores simplesmente executando os módulos apropriados. Isso elimina a necessidade de acomodar todas as configurações possíveis de *hardware* num único processo [26]. Outra vantagem é que um módulo pode ser executado em processadores e em computadores diferentes que se comunicam por uma rede. Isso possibilita uma maior confiabilidade, já que esse nível de independência não causa falha nos demais módulos caso algum deles falhe. Por último, a modularidade proporciona usabilidade e extensibilidade, pois diferentes módulos que executam a mesma função convergem para uma única interface abstrata.

No CARMEN, a comunicação é feita a partir do software *Inter-Process Communication* (IPC), que foi criado em 1994 por Reid Simmons em *Carnegie Mellon*. Esse tipo de comunicação foi projetada para facilitar a interação entre processos de controle heterogêneos em grandes sistemas de engenharia [28]. O IPC se baseia nos paradigmas de comunicação *Publish-Subscribe* e *Request-Reply* [29], que permitem enviar e receber estruturas de dados complexas de uma forma flexível e eficaz, incluindo listas e vetores de tamanho variável. Além da capacidade de conexão de alto nível entre processos, a biblioteca do IPC oferece também facilidades para definição (*Register*), serialização (*Marshall*), envio e recebimento de mensagens, abstraindo os detalhes de comunicação via *sockets* sobre o protocolo TCP/IP.

O IPC fornece suporte de alto nível para a conexão de processos usando pacotes TCP/ IP e envio de dados entre os processos, abstraindo os detalhes de comunicação. Nele é tratado a abertura dos pacotes, e registro, envio e recebimento, de mensagens sendo tanto do tipo *publish*/*subscribe* quanto cliente/servidor. Essa biblioteca contém funções para o desempacotamento onde são realizadas as chamadas das funções de *callback*, também conhecidas como *handlers,* definidas pelos usuários em intervalos definidos. Uma única mensagem enviada pode ser entregue para diversos destinos que estejam inscritos nela.

Para realizar a troca de mensagens é utilizado um módulo servidor central, aplicação independente utilizada para manter as informações de todo o sistema e para rotear o tráfego de mensagens e de *log*. Antes de iniciar os módulos, esse programa central é

iniciado. O principal serviço que o módulo central fornece é a passagem de mensagens. Uma mensagem enviada a partir de qualquer módulo será encaminhada pelo módulo servidor central para o módulo que contém o manipulador para a mensagem. Uma representação da comunicação pode ser visto na Figura 16.

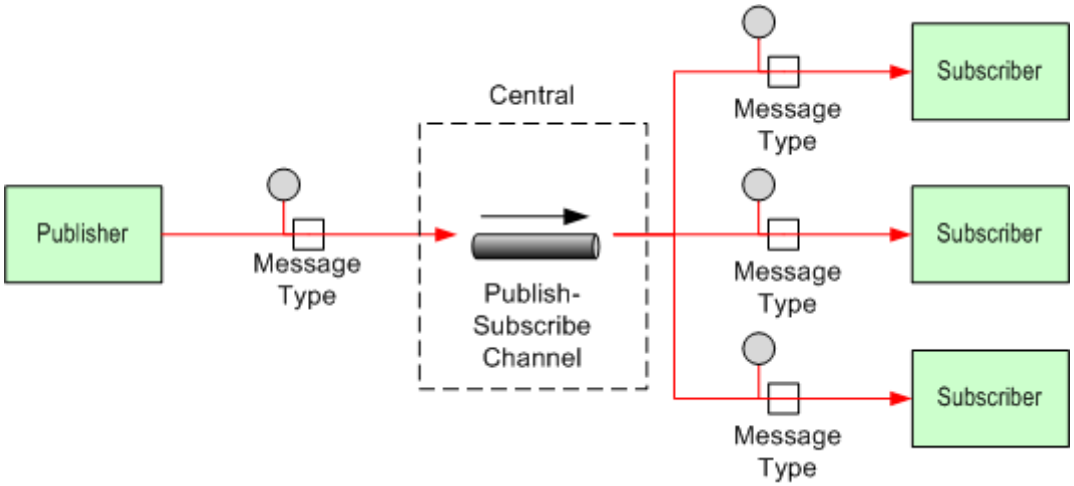



**Figura 16 - Modelo de comunicação Publish-Subscribe [29].**

Mais de um módulo servidor central pode ser executado na mesma máquina, usando portas de comunicação diferentes para cada servidor. Ter múltiplos servidores é especialmente útil para o desenvolvimento de *software*.

## 4.2 Arquitetura do Sistema

CARMEN está preparado para controlar vários robôs comerciais como *Nomad* XR4000 [30], *Scout* [31] e *ActivMedia Pioneers* [32]. Entretanto o foco desse trabalho é o desenvolvimento de um planejador de caminho para veículos de passeio. Por isso, CARMEN foi estendido adicionando as funcionalidades necessárias. Os módulos relevantes para este trabalho são mostrados na Figura 17. Nessa figura, cada módulo é representado pelos retângulos e os fluxos de mensagens são representados por setas pontilhadas. Segue abaixo uma breve descrição do papel de cada módulo.

- ***Mapper*:** módulo responsável por construir o mapa do ambiente próximo ao veículo autônomo, a partir de dados obtidos dos sensores e da localização do veículo.

- ***Map Server*:** módulo responsável por publicar os mapas construídos do ambiente em torno da posição atual do veículo.

- ***Localize*:** módulo responsável por estimar a localização do veículo a partir de dados obtidos dos sensores e do mapa enviado pelo módulo *Map Server.*

- ***Behavior Selector*:** módulo responsável por manter a lista de objetivos que o veículo deve seguir. Além disso, este módulo deve controlar o comportamento do planejador de acordo com o ambiente (pista ou estacionamento).

- ***Motion Planner*:** módulo responsável por implementar o planejamento de caminho (foco deste trabalho).

- ***Obstacle Avoider*:** módulo responsável por receber o planejamento do *motion planner* e, caso necessário (caso haja algum obstáculo novo no trajeto), interferir reduzindo as velocidades para que o veículo não colida.

- ***Base*:** módulo responsável pela comunicação de baixo nível com o robô. Recebe uma lista de comandos gerados pelo planejador de caminho e os traduz em esforço de freio, acelerador e giro da roda dianteira.

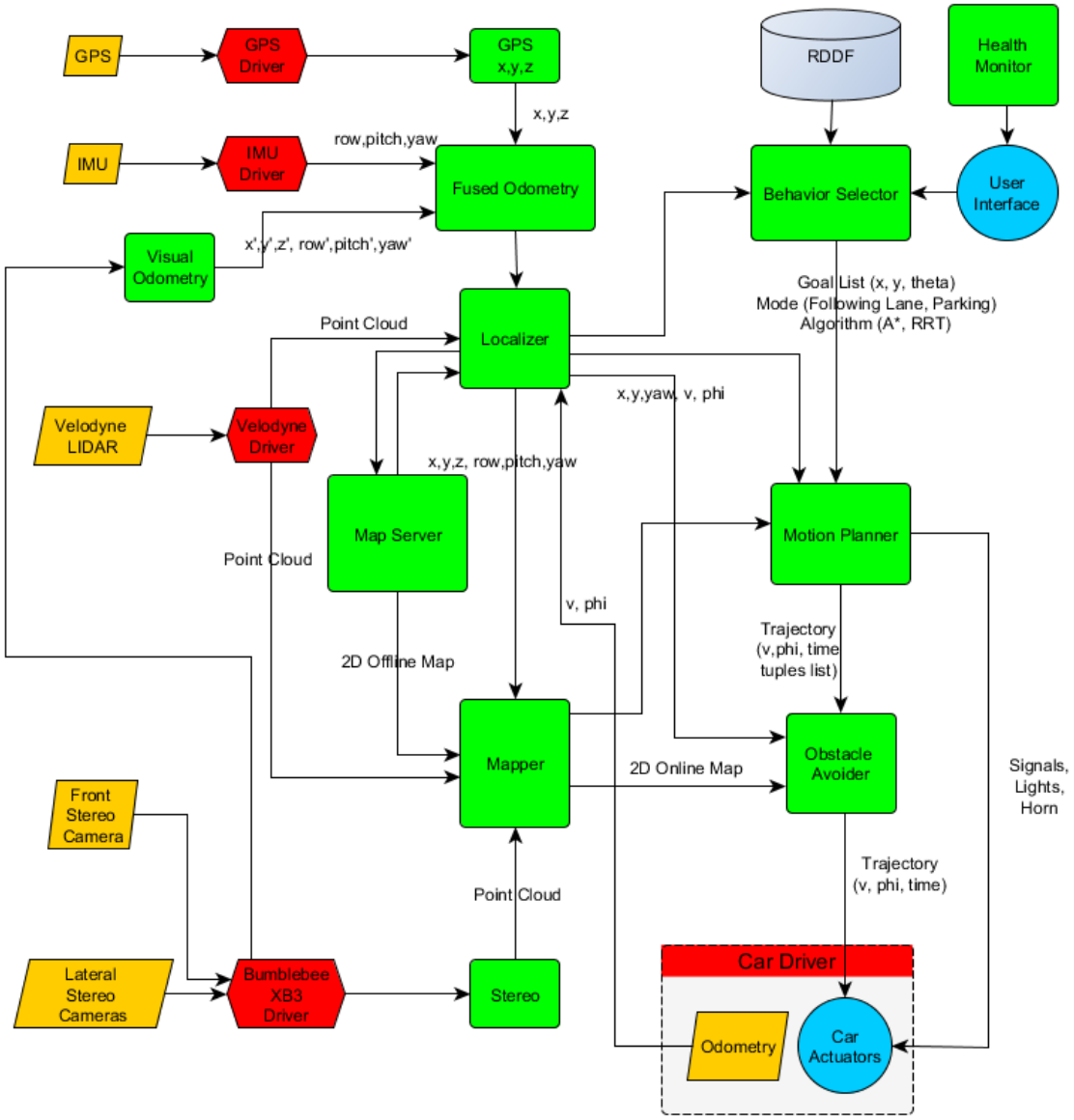



**Figura 17 - Módulos relevantes utilizados no CARMEN.**

## 4.3 O Veículo Autônomo IARA

Para os testes do planejador de caminho desenvolvido neste trabalho foi utilizado um *Ford Escape Hybrid*, apresentado na Figura 18.


**Figura 18 - Ford Escape Hybrid.**

A tecnologia eletrônica de acionamento dos atuadores (volante, acelerador, freio, entre outros) do automóvel foi desenvolvida pela empresa *Torc Robotics*, Virginia, Estados Unidos [33] para que seja apto a receber comandos de um computador. O veículo foi equipado com diversos sensores para perceber o mundo (Figura 19). Os sensores disponíveis na plataforma robótica incluem:

**Câmeras estéreo Bumblebee XB3 e XB2 (Point Grey, Vancouver, Canadá)** [34] – Responsáveis pela captura de imagens estéreo são utilizadas para gerar uma nuvem de pontos 3D do ambiente.

***Light Detection And Ranging*** **(LIDAR) HDL-32E (*Velodyne,* Califórnia, Estados Unidos)** [35] **–** Um robusto sensor *laser* que apresenta 32 lasers na vertical também proporcionando uma visibilidade de 360° do ambiente na horizontal a partir de rotação em seu próprio eixo, gerando uma nuvem de 700 mil pontos por segundo com um alcance de 70 metros e precisão em torno de 2cm.

***Altitude and Heading Reference System*** **(AHRS) MTi-G (Xsens, Enschede, Holanda)** [36] **–** O MTi-G é um GPS auxiliado por uma unidade de medição Inercial (*based Inertial Measurement* - IMU) e sensor de pressão estática. Ele proporciona um desempenho sem precedentes para o seu tamanho, peso, custo e baixa complexidade em uso. O seu uso auxilia principalmente a correção da localização.

Os sensores citados são utilizados para realizar o mapeamento do ambiente, assim como também a localização no mesmo. Portanto, é a partir deles que são obtidos os

dados para estimar a localização do robô e os obstáculos presentes ao seu redor, ambos utilizados durante o planejamento de caminho.


**Figura 19 – Sensores do robô.**

O porta malas do carro robô pode ser observado na Figura 20. Foi equipado com um sistema para extrair energia da bateria do veículo e alimentar computadores responsáveis por todo o processamento necessário. Para processar todos os dados dos sensores, IARA também conta com um computador Dell Precision R5500 (2 Processadores Intel Xeon 2.13 GHZ, 12 GB de memória DDR3 1333MHZ, 2 HDs SSD 120GB em RAID0, 2 Placas de Rede 1GB, Placa de Vídeo Quadro 600, Placa de Vídeo Tesla C2050).


**Figura 20 – Recursos computacionais do robô.**

## 4.4 Parâmetros do Algoritmo

Nesse tópico são definidos alguns parâmetros utilizados durante os testes do algoritmo. Estes parâmetros foram obtidos empiricamente ou representam limitações físicas do veículo ou da implementação do algoritmo. Os valores são mostrados na Tabela 1.

| Parâmetro | Valor |
|---|---|
| Velocidade Máxima em manobra de pista | **2,5 m/s** |
| Velocidade Máxima em manobra de estacionamento | **1,5 m/s** |
| Ângulo da Roda Dianteira Máximo | **26,356 graus** |
| Aceleração máxima | **2,7 m/s$^2$** |
| Desaceleração máxima | **9 m/s$^2$** |
| Taxa Máxima de Curvatura da Roda Dianteira | **27,699 graus/s** |
| Massa do Veículo | **1500 kg** |
| Distância entre Vértices | **1 m** |
| Resolução do mapa em *grid* | **0.2 m** |
| Resolução (x,y) da heurística pré-computada | **0.2 m** |
| Resolução (theta) da heurística pré-computada | **5°** |
| Dimensão (x,y) da heurística pré-computada | **100m** |

Tabela 1 - Parâmetros do A* de Estado Híbrido.

O parâmetro de velocidade em manobras de pista e de estacionamento são estipulados de forma que permita ao algoritmo conseguir obedecer o trajeto definido. Manobras de estacionamento tiveram uma velocidade máxima reduzida por exigirem caminhos mais complexos e alternações entre frente e ré. Esses limites também servem como uma margem de segurança para que o algoritmo não apresente grandes riscos em caso de falha em testes no IARA.

A taxa máxima de curvatura, aceleração e desaceleração e a massa do veículo são valores da arquitetura do robô utilizado. A velocidade máxima em pista e em estacionamento são limites atribuídos ao algoritmo para oferecer segurança nos testes no mundo real. A distância entre vértices indica a distância percorrida entre 2 estados durante a exploração. A resolução do mapa em *grid* indica o nível de detalhes que os obstáculos serão tratados. Os valores de distância entre vértices e resolução do mapa, quando possuem valores muito altos tornam difícil a busca em ambiente que exija

manobras mais complexas, porém valores muito baixos tornam o algoritmo mais lento. A resolução da heurística pré-computada indica o nível de discretização do mapa pré-computado e sua dimensão indica até qual distância do veículo o mapa irá ser computado. Valores muito baixos tornam a estimativa menos precisa, porém valores muito altos tornam a etapa de pré-computação mais custosa e exige maior memória para armazena-la.

## 4.5 Método de comparação

Neste tópico são apresentados os métodos para avaliar o algoritmo implementado. Foram realizados testes medindo a energia gasta e a distância percorrida e esses resultados foram comparados com os de motoristas humanos. Há interesse em minimizar esses valores, pois eles podem reduzir o consumo de combustível de um automóvel. Avaliações do tempo necessário para o algoritmo achar o caminho até o destino também foram realizadas para testar a viabilidade do algoritmo em ambientes reais. O algoritmo foi avaliado tanto em ambiente simulado quanto no robô IARA dentro da universidade, realizando tanto testes de manobras de estacionamento quanto de pista. Foram utilizados dois tipos de motoristas humanos para o teste, sendo eles:

- **Experiente**: Um motorista habilitado a bastante tempo.
- **Recém habilitado**: Um motorista que ainda possuí a carteira provisória.

Foram realizados 20 experimentos do algoritmo por meio de simulação a fim de obter uma análise estatística confiável a partir da média dos resultados obtidos. Os testes físicos no robô IARA foram executados apenas uma única vez dando uma simples indicação do desempenho em caso real.

## 4.6 Trajeto dos testes

Os testes foram realizados num estacionamento da UFES, mostrado em amarelo ao centro da foto de satélite na Figura 21. Esse lugar foi escolhido por ter baixa rotativi-

dade de carros e pessoas e por haver espaço suficiente para realizar percursos consideráveis e para explorar as principais manobras de curva que um carro pode realizar, tanto para pista quanto para estacionamento.

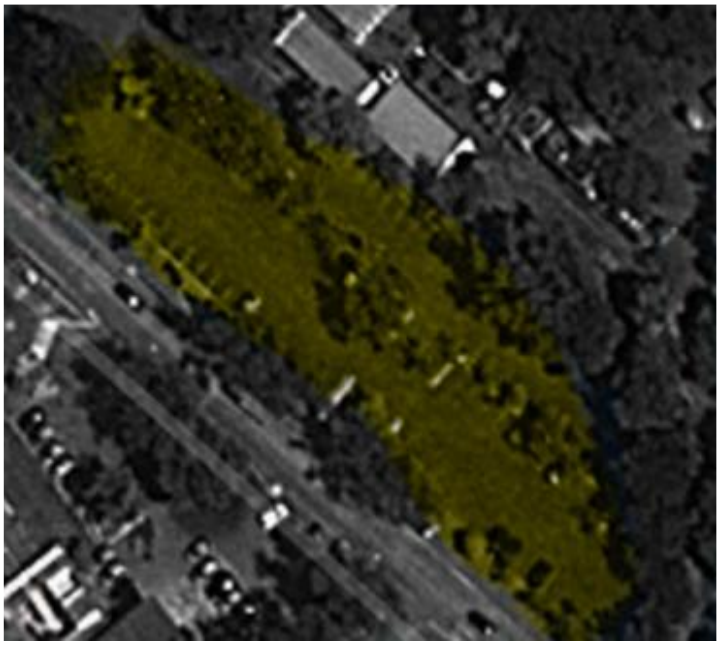


**Figura 21 - Foto de satélite do estacionamento. A parte amarelada representa o ambiente de estacionamento utilizado.**

O trajeto para testar o planejador em pista constitui de uma sequência de destinos enviados pelo módulo *Behavior Selector*. A sequência de destinos foi formulada para que tenha um aspecto de circuito. Como pode ser visto na Figura 22, o trajeto inicia no ponto verde indicado no mapa e segue um caminho de vários *waypoints* representados em vermelho sugeridos pelo *Behavior Selector* em forma de circuito tendo como objetivo final o ponto de partida indicado em verde.

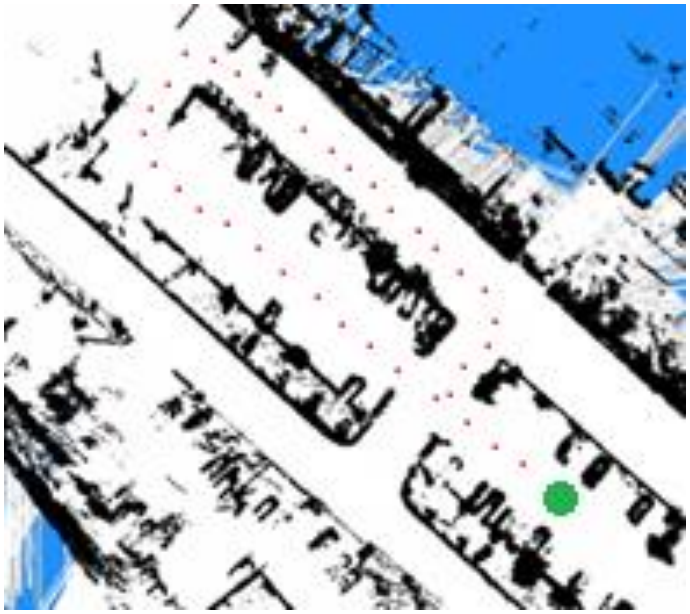


**Figura 22 - Percurso do circuito de pista. A cor verde representa o ponto inicial de onde o carro deve partir e retornar ao final. Os pontos vermelhos representam os destinos intermediários indicados pelo módulo *Behavior Selector*.**

Para testar o algoritmo em estacionamento foram utilizados um ponto de origem e de destino fixos que levaram a uma manobra de estacionamento vertical. A origem e o destino podem ser vistos na Figura 23, onde estão representados pela cor azul-escuro e amarela, respectivamente.

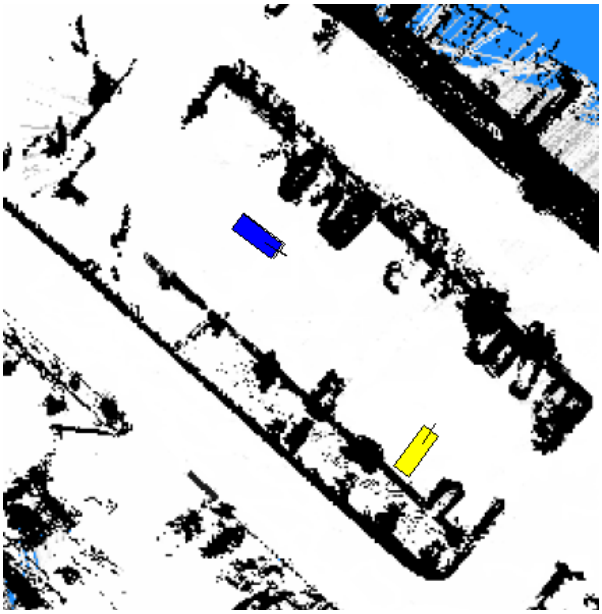


**Figura 23 – Origem e destino da manobra de estacionamento. Carro de cor azul-escuro representando a posição atual do carro e cor amarela o destino desejado. As cores pretas representam obstáculos e brancas as áreas trafegáveis. Azul claro representa um ambiente desconhecido.**

## 4.7 Métricas Comparativas

Neste trabalho serão utilizadas as seguintes métricas:

- Energia: Para descobrir a energia aproximada gasta em um percurso, a equação de energia cinética dada por [37] é utilizada:

$$E = \sum \frac{m\left({v_i}^2 - v_{i-1}^2\right)}{2} \tag{7}$$

onde $v_i$ é a velocidade do veículo no instante $i$ e $m$ é a massa do veículo.

- Distância percorrida:

$$DP = \sum v_i * \Delta t_i \tag{8}$$

Onde $v_i$ e $\Delta t_i$ são a velocidade e o intervalo de tempo medido no instante $i$.

Os dados de velocidade e intervalo de tempo são coletados diretamente da odometria do carro com uma frequência de 20 hertz.

Essas métricas possuem a vantagem de não serem dependentes de planejadores de caminho, possibilitando realizar comparações entre diferentes tipos de planejadores e até mesmo com motoristas humanos, desde que o percurso utilizado para medição seja o mesmo.

Tanto a odometria quanto os controles do carro possuem imprecisão. Essa imprecisão gera oscilações na velocidade, o que dificulta a comparação da energia gasta, principalmente com casos simulados onde não há tais oscilações como pode ser visto na Figura 24. Para que a comparação seja mais justa, foi aplicado o filtro *Buttherworth* [38].

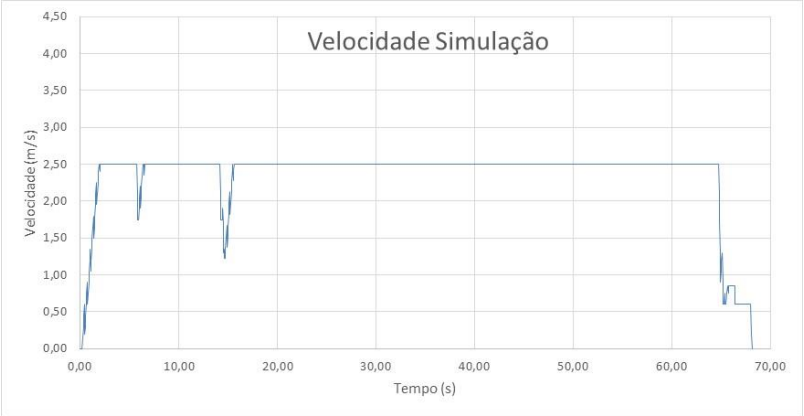



**Figura 24 - Velocidade do Algoritmo na Simulação.**

Por exemplo, podemos ver um gráfico da velocidade (m/s) em função do tempo (s) de um motorista experiente sem a velocidade filtrada e com a velocidade filtrada nas Figura 25 e Figura 26, onde fica clara a redução das oscilações com a aplicação do filtro, diminuindo assim a energia gerada a partir dos erros.

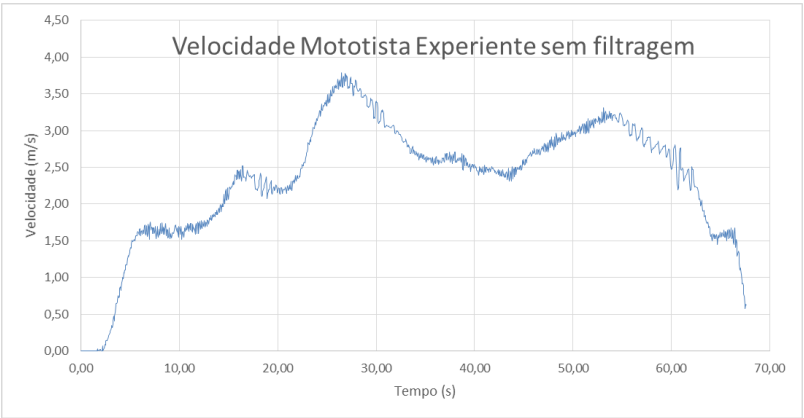



**Figura 25 - Velocidade de um motorista Experiente sem filtragem.**


**Figura 26 - Velocidade de um motorista Experiente com filtragem.**

# 5 EXPERIMENTOS E RESULTADOS

Neste capítulo, são apresentados os resultados dos experimentos realizados para avaliar o planejador de caminho implementado.

No tópico 5.1, são apresentados os resultados dos experimentos realizados para avaliar o desempenho do planejador em manobras de estacionamento e no tópico 5.2 são apresentados os resultados dos testes de manobras de pista. Com o intuito de comparar o desempenho do algoritmo com o desempenho de um motorista, foram feitos gráficos que plotam os percursos realizados assim como a energia gasta, a distância percorrida e o tempo gasto para realizar o percurso.

## 5.1 Resultados de Manobras de Estacionamento

Nesta seção, são apresentados os resultados do desempenho do algoritmo no planejamento de manobras de estacionamento. Estacionar geralmente exige manobras mais complexas e portanto demanda que o robô obedeça os comandos do planejamento com mais perfeição. Portanto, os passos são executados com velocidade reduzida. Além disso, o robô para totalmente a cada manobra, assim como um uma pessoa faz na maioria das balizas.

A plotagem do percurso realizado pelos motoristas e pelo algoritmo, ambos extraídos a partir dos dados de localização do robô, podem ser vistos na Figura 27, onde é possível ver que os dois tipos de motoristas se comportaram de maneira muito semelhante durante a manobra de estacionamento: eles executaram manobras mais simples, porém que necessitam de um espaço grande para a sua execução. Um motorista Recém habilitado poderia ter dificuldades ou não conseguir estacionar num ambiente que não possua um espaço semelhante.

Já nos testes do algoritmo foi observado um caminho com um número maior de manobras. Contudo, o caminho ficou visivelmente menor, necessitando menos espaço para o estacionamento, o que indica que o algoritmo é capaz de achar rotas que dificilmente uma pessoa conseguiria encontrar. Contudo, o algoritmo executado no robô se comportou diferente do encontrado na simulação, pois, devido às incertezas da

localização e do controle, o algoritmo teve que refazer o plano antes da primeira curva onde houve um pequeno desvio, e nesse novo plano o algoritmo achou um caminho diferente para a nova posição que ele se encontrava.

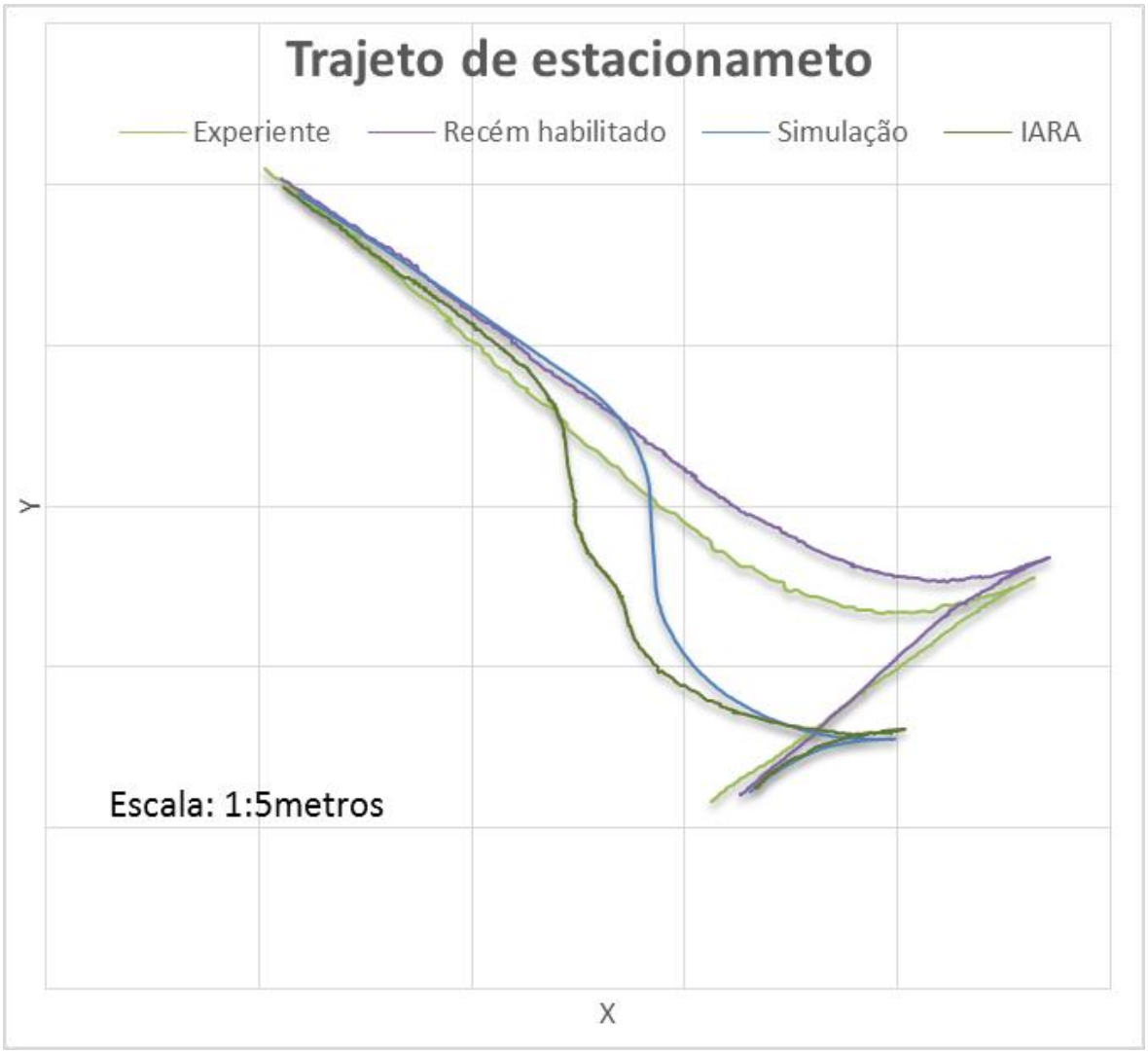



**Figura 27 –Representação da trajetória realizada durante o estacionamento no espaço 2D x e y. O trajeto foi realizado por dois tipos de motoristas, vinte simulações e um teste real com o robô IARA.**

Tanto o algoritmo Simulado quanto o implementado no robô apresentaram uma distância percorrida menor que os motoristas avaliados, como pode ser visto na Figura 28. Esse gráfico mostra que os motoristas percorreram uma distância em torno de 37 metros, enquanto o algoritmo A* de Estado Híbrido apresentou uma distância menor em torno de 29 metros. Uma distância um pouco maior foi observada no teste no IARA, porque, devido às imprecisões, o robô se desviou do seu caminho e fez um replanejamento, que resultou em um trajeto maior.

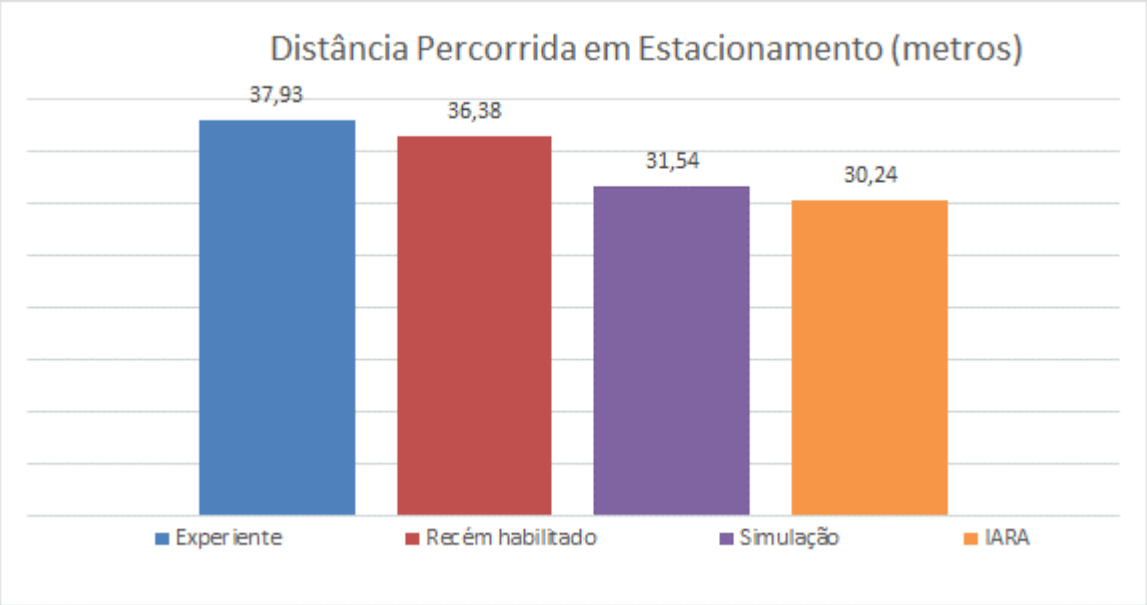



**Figura 28 - Distância percorrida durante o estacionamento em metros. Teste realizado com dois motoristas, vinte experimentos simulados e um experimento no robô IARA.**

Na Figura 29 é possível observar a energia total gasta durante o trajeto dos testes. O motorista Experiente teve um gasto exagerado de energia quando comparado ao motorista recém habilitado. O algoritmo teve os menores consumos de energia na simulação, contudo, no robô apresentou um gasto de energia menor apenas que o motorista experiente, que teve um gasto exagerado de energia.

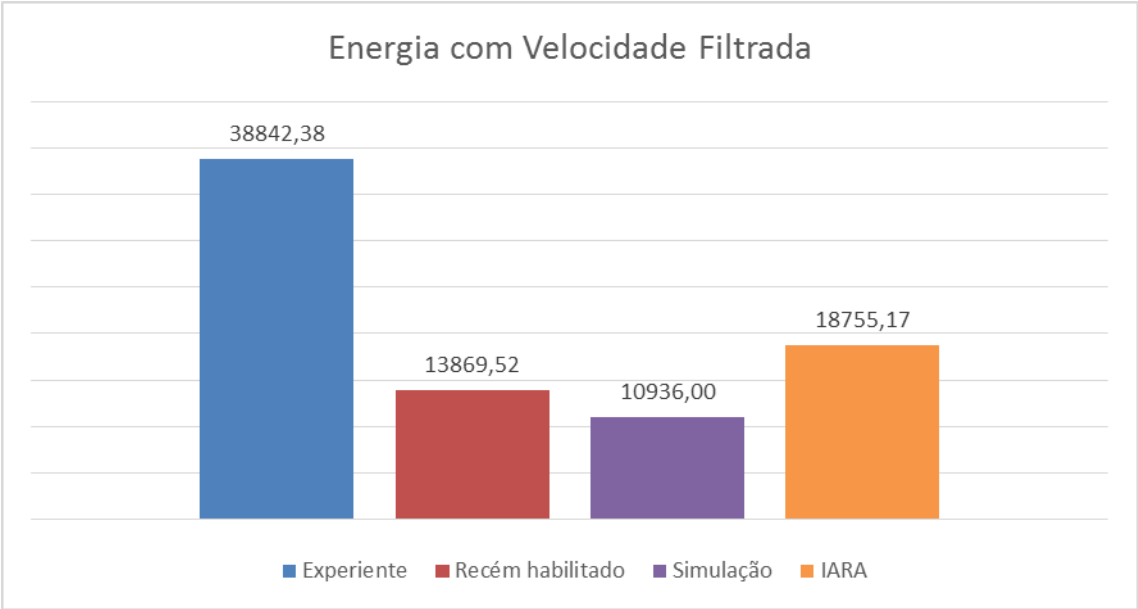



**Figura 29 - Energia do teste de estacionamento com velocidade filtrada. Teste realizado com dois motoristas, vinte experimentos simulados e um experimento no robô IARA.**

O alto consumo de energia do motorista experiente ocorreu devido ao exagero de aceleração e frenagem durante a trajetória com o objetivo de chegar mais rápido ao

destino, como pode ser visto na Figura 30, onde podemos observar que o motorista chega a uma velocidade de quase 5 m/s e depois reduz abruptamente a velocidade para realizar a ré. Na Figura 31, o A* de Estado Híbrido executado não teve excessos de velocidade. Porém, além do algoritmo parar diversas vezes para executar manobras precisas no volante, também teve que seguir uma nova trajetória, devido aos erros de localização e controle.

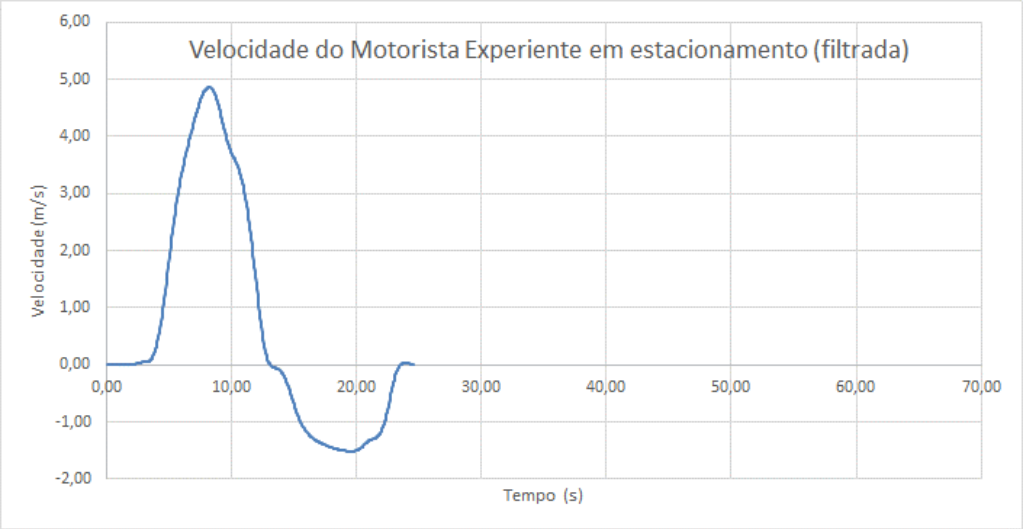



**Figura 30 - Velocidade do motorista experiente com filtragem em função do tempo no teste de estacionamento.**

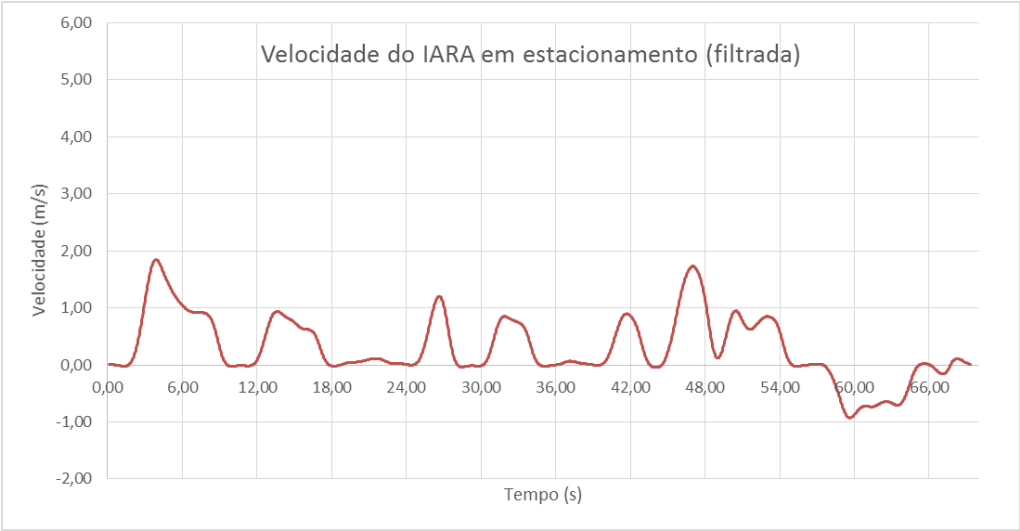



**Figura 31 - Velocidade do robô IARA com filtragem em função do tempo no teste de estacionamento.**

Na Figura 32 é apresentado o tempo gasto para realizar o trajeto de estacionamento de cada um dos testes realizados. É possível ver que o motorista experiente conseguiu um tempo visivelmente menor, pois possuía segurança do caminho realizado e o

fez utilizando uma maior velocidade, sem se preocupar com o gasto de energia. O motorista recém habilitado não possui muita confiança no trajeto percorrido e por isso demorou mais tempo para realizá-lo. O Algoritmo nos testes de simulação mostrou tempos superiores ao motorista menos experiente, pois, apesar de parar várias vezes para realizar manobras no volante, utilizou trajetos mais curtos e não necessitam de muito tempo para planejar o trajeto a ser executado. Contudo, A* de Estado Híbrido executado no robô IARA gastou mais tempo para executar a trajetória, pois, como foi dito anteriormente, ele teve um maior número de paradas.

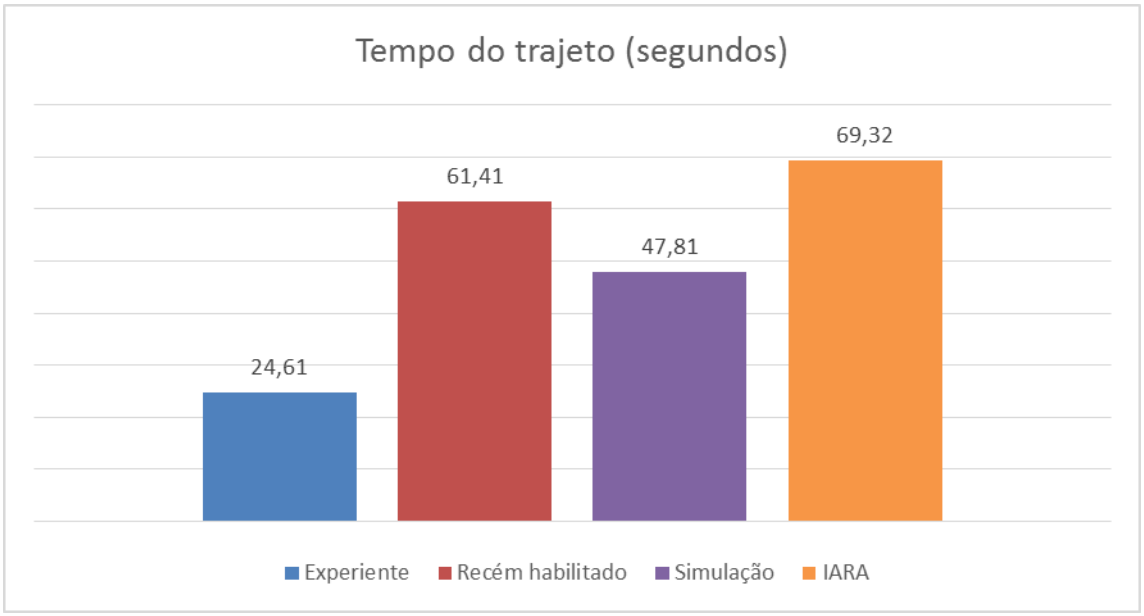



**Figura 32 - Tempo de planejamento do trajeto no Estacionamento em segundos. Teste realizado com dois motoristas, vinte experimentos simulados e um experimento no robô IARA.**

## 5.2 Resultados de Manobras de Pista

Nesta seção, são apresentados os resultados do desempenho do algoritmo no planejamento de manobras em pista. Os testes em pista geralmente não exigem manobras muito complexas, porém necessitam de respostas rápidas de planejamento, principalmente em virtude de mudanças inesperadas do ambiente em que o carro trafega e em virtude de imprecisões da localização e do controle.

A plotagem do percurso realizado pelos motoristas e pelo algoritmo, ambos extraídos a partir dos dados de localização do robô, podem ser vistos na Figura 33. É possível observar que o algoritmo se comportou de maneira um pouco diferenciada dos motoristas. Na última curva por exemplo, o motorista experiente a realiza de maneira mais

aberta para só depois seguir o caminho reto, enquanto o motorista recém habilitado fecha demais a curva. Isso provavelmente se dá, pelo fato da falta de experiência com a condução, tendo que fazer logo após uma correção da rota. Nesse intervalo, o algoritmo simulado seguiu uma rota entre os dois motoristas nem abrindo e nem fechando demais a curva final, preferindo finalizar a curva num ângulo que incentive o prosseguimento pelo caminho mais curto.


**Figura 33 - Representação da trajetória realizada durante o circuito em pista no espaço 2D x e y. O trajeto foi realizado por dois motoristas e vinte simulações.**

Não foram realizados testes práticos no robô IARA devido a imprecisão na execução dos controles, inviabilizando uma navegação suave em velocidades que são desejadas em uma pista.

# 6 DISCUSSÃO

Neste capítulo são discutidos os principais trabalhos correlatos em 1.1 bem como a análise crítica desse trabalho em 6.1.

## 6.1 Análise Crítica

A implementação feita do algoritmo de navegação apresentou algumas limitações que foram vistas principalmente nos resultados apresentados de manobras de pista, em que o controle tem dificuldades de obedecer o comando enviado pelo planejamento de caminho, pois os comandos não obedecem algumas velocidades, principalmente na velocidade de rotação do ângulo volante. Isto implica no replanejamento constante durante a navegação em pista, a fim de evitar que o robô siga um caminho incorreto. Este constante planejamento acaba fazendo com que haja uma constante mudança de trajetória, ocasionando uma navegação imperfeita e pouco suave. Durante a navegação em estacionamento, essa limitação do algoritmo é compensada parando o veículo a cada manobra de volante.

A incapacidade de compreender o que é mão ou contramão é outra limitação presente no algoritmo. Portanto, se a contramão representar um menor caminho para o algoritmo, ele provavelmente seguirá por ela. Porém os objetivos intermediários enviados pelo módulo *Behavior Selector* incentivam o algoritmo a se manter na mão certa durante a navegação em pista.

# 7 CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo apresenta um breve sumário e as conclusões alcançadas em 7.1, e em 7.2 aponta novas direções para trabalhos futuros.

## 7.1 Conclusão

Este trabalho abordou o estudo de um algoritmo de planejamento de caminho para automóveis robôs, tendo como objetivo a investigação e implementação de métodos computacionais baseados no algoritmo A* para permitir a navegação autônoma de veículos convencionais no contexto da Volta da Ufes. O planejador foi utilizado em dois ambientes: estacionamento e pista. O algoritmo foi comparado com dois tipos de motoristas, sendo um experiente e outro recém habilitado. Foram realizados testes comparativos medindo a energia gasta no caminho, a distância percorrida e o tempo necessário para realizar o percurso.

Em estacionamento, o planejador se mostrou capaz de gerar manobras mais curtas que um motorista comum realizaria, sem desperdiçar energia mantendo um tempo para manobrar compatível com o tempo gasto pelo motorista. Em pista foi possível realizar um percurso na simulação com velocidades de até 2.5 m/s, compatível com o que um motorista comum andaria em ambientes dentro da universidade.

O algoritmo se portou bem em testes que exijam manobras para estacionamento. Porém testes que exigiram manobras de pista não foram realizados com o IARA pela falta de suavidade dos comandos do A* de Estado Híbrido, ou seja, os controles não foram capazes de obedecer os comandos exigidos pelo algoritmo, causando desvios da rota.

## 7.2 Trabalhos Futuros

Uma direção para trabalhos futuros seria a implementação de uma pós-otimização do algoritmo, com o intuito de tornar os caminhos mais suaves e facilitar a sua execução

pelo controle do robô. Outra direção para trabalho futuro será um controle mais robusto, que minimize a imprecisão da velocidade e do volante e obedeça mais fielmente os comandos enviados pelo planejamento de caminho. Outra importante tarefa para trabalhos futuros seria incorporar o conhecimento de faixas de trânsito ao algoritmo, diminuindo o risco de acidentes em ambientes com transito. Finalmente, seria também importante incorporar a detecção e acompanhamento de obstáculos móveis, minimizando os riscos do planejamento de caminho gerar colisão com estes obstáculos ou gerar processamento desnecessário em função da incerteza do ambiente.

# 8 REFERÊNCIAS


[1] N. Geddes, Magic motorways, Random house, 1940.

[2] L. M. Goncalves, Sistema de Navegação de Veículo Autônomo Inteligente, Itajubá: Universidade Federal de Itajubá - Programa de Pós Graduação em Engenharia Eletrica, 2010.

[3] "The self-driving car logs more miles on new wheels," 2012. [Online]. Available: http://googleblog.blogspot.hu/2012/08/the-self-driving-car-logs-more-miles-on.html. [Acesso em 22 Setembro 2013].

[4] "AutoNOMOS Labs," [Online]. Available: http://www.autonomos.inf.fu-berlin.de/. [Acesso em 22 Setembro 2013].

[5] M. Buehler, K. Iagnemma e S. Singh, The 2005 DARPA Grand Challenge: The Great Robot Race, 1st ed., Springer Publishing Company, Incorporated, 2007.

[6] M. Buehler, K. Iagnemma e S. Singh, "Special Issue on the 2007 DARPA Urban Challenge, Part I," *Journal of Field Robotics*, vol. 25, nº 8, pp. 423-424, 2008a.

[7] M. Buehler, K. Iagnemma e S. Singh, "Special Issue on the 2007 DARPA Urban Challenge, Part II," *Journal of Field Robotics*, vol. 25, nº 9, pp. 567-568, 2008.

[8] S. LaValle, Planning Algorithms, Cambridge University Press, 2006.

[9] P. Hart, N. Nilsson e B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, nº 2, pp. 100-107, 1968.

[10] P. E. Hart, N. J. Nilsson e B. Raphael, "Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths"," *SIGART Bull.*, nº 37, pp. 28-29, 1972.

[11] D. Ferguson e A. Stentz, "Field D*: An Interpolation-based Path Planner and Replanner," pp. 239-253, October 2005.

[12] D. Dolgov, S. Thrun, M. Montemerlo e J. Diebel, "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments," *The International Journal of Robotics Research*, vol. 29, nº 5, pp. 485-501, 2010.

[13] B. Leedy, J. Putney, C. Bauman, S. Cacciola, J. Michael Webster e C. Reinholtz, "Virginia Tech's Twin Contenders: A Comparative Study of Reactive and Deliberative Navigation," em *The 2005 DARPA Grand Challenge*, vol. 36, M. Buehler, K. Iagnemma e S. Singh, Eds., Springer Berlin Heidelberg, 2007, pp. 155-182.

[14] R. R. Radaelli, Planejamento De Movimento Para Veículos Convencionais Usando Rapidly-Exploring Random Tree, Vitória: Universidade Federal do Espirito Santo - Departamento de Informática, Data prevista para defesa: 26/08/2013.

[15] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer e others, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, nº 8, pp. 425-466, 2008.

[16] G. Wyeth e M. Milford, "Spatial cognition for robots," *Robotics Automation Magazine, IEEE*, vol. 16, nº 3, pp. 24-32, 2009.

[17] A. D. Ekstrom, M. J. Kahana, J. B. Caplan, T. A. Fields, E. A. Isham, E. L. Newman e I. Fried, "Cellular networks underlying human spatial navigation," *Nature*, vol. 425, nº 6954, pp. 184-188, 2003.

[18] A. S. Etienne e K. J. Jeffery, "Path integration in mammals," *Hippocampus*, vol. 14, nº 2, pp. 180-192, 2004.

[19] Q. Chen e Ü. Özgüner, "Intelligent off-road navigation algorithms and strategies of Team Desert Buckeyes in the DARPA Grand Challenge 2005," *Journal of Field Robotics*, vol. 23, nº 9, pp. 729-743, 2006.

[20] R. Brooks, "Visual map making for a mobile robot," em *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*, 1985.

[21] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, nº 3, pp. 345-405, 1991.

[22] A. Elfes, "Occupancy grids: A stochastic spatial representation for active robot perception," *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 136-146, 1990.

[23] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki e S. Thrun, Principles of Robot Motion: Theory, Algorithms, and Implementations, Cambridge, MA: MIT Press, 2005.

[24] K. Konolige, "A gradient method for realtime robot control," *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, vol. 1, pp. 639-646, 2000.

[25] J. A. REEDS e L. A. SHEPP, "Optimal Paths for a Car tha goes both Forwards and Backwards," *PACIFIC JOURNAL OF MATHEMATICS*, pp. 367-393, 1990.

[26] M. Montemerlo, N. Roy e S. Thrun, "Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit," pp. 2436-2441, 2003.

[27] N. A. Melchior e W. D. Smart, "A framework for robust mobile robot systems," pp. 145-154, 2004.

[28] R. S. D. James, "Inter Process Comunication - A Reference Manual," 2011.

[29] G. Hohpe e B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions, 1 ed., Addison-Wesley Professional, 2003.

[30] Nomadic, "Nomadic," [Online]. Available: http://cbcis.ttu.edu/ep/old_netra_site/about/xr4000/XR4000.htm. [Acesso em 07 08 2013].

[31] Reconrobotics, "Reconrobotics," [Online]. Available: http://www.reconrobotics.com/. [Acesso em 07 08 2013].

[32] mobile Robots, "Mobile Robots," [Online]. Available: http://www.mobilerobots.com/. [Acesso em 7 08 2013].

[33] [Online]. Available: http://www.torcrobotics.com/. [Acesso em 01 07 2013].

[34] [Online]. Available: http://ww2.ptgrey.com/. [Acesso em 01 07 2013].

[35] [Online]. Available: http://www.velodynelidar.com. [Acesso em 2013 07 01].

[36] [Online]. Available: http://www.xsens.com. [Acesso em 03 07 2013].

[37] YOUNG, FREEDMAN, SEARS e ZEMANSKY, Física 1: Mecânica, 12ª ed., São Paulo: Pearson Addison Wesley.

[38] G. Matthaei, E. Jones e L. Young, Microwave Filters, Impedance-Matching Networks, and Coupling Structures, McGraw-Hill, 1964.

[39] E. Rimon e D. Koditschek, "Exact Robot Navigation Using Artificial Potential Fields," *IEEE Transactions on Robotics and Automation,* vol. 8, nº 5, pp. 501-518, 1992.

[40] K. Bekris e L. Kavraki, "Greedy but Safe Replanning under Kinodynamic Constraints," em *Robotics and Automation, 2007 IEEE International Conference on*, 2007.

[41] A. Weinstein e K. Moore, "Pose estimation of Ackerman steering vehicles for outdoors autonomous navigation," em *Industrial Technology (ICIT), 2010 IEEE International Conference on,* 2010.

[42] "ES TV 1ª Edição," 2013. [Online]. Available: http://g1.globo.com/videos/espirito-santo/estv-1edicao/t/edicoes/v/departamento-de-informatica-da-universidade-federal-do-es-desenvolve-carro-autonomo/2500413/. [Acesso em 14 07 2013].

[43] "G1," 2013. [Online]. Available: http://g1.globo.com/espirito-santo/noticia/2013/04/alunos-e-professores-do-es-desenvolvem-carro-que-anda-sozinho.html. [Acesso em 14 07 2013].

[44] "Jornal Hoje," 2013. [Online]. Available: http://g1.globo.com/jornal-hoje/videos/t/edicoes/v/professores-e-alunos-de-universidade-do-es-criam-carro-que-anda-sozinho/2515111/. [Acesso em 14 07 2013].

[45] "G1," 2013. [Online]. Available: http://g1.globo.com/espirito-santo/noticia/2013/04/g1-faz-test-drive-em-carro-que-atropelou-ana-maria-braga.html. [Acesso em 14 07 2013].

[46] "Tv Capixaba - Espaço Sustentável," 2013. [Online]. Available: http://www.youtube.com/watch?feature=player_embedded&v=xYnfNkCHeb4. [Acesso em 14 07 2013].

[47] "Mais Você," 2013. [Online]. Available: http://tvg.globo.com/programas/mais-voce/videos/t/programas/v/ana-maria-chega-ao-mais-voce-em-carro-que-anda-sem-motorista/2530185/. [Acesso em 14 07 2013].

[48] S. B. W. F. D. THRUN, Probabilistic Robotics, MIT Press, 2005.

[49] L. H. O. Rios e L. Chaimowicz, "A survey and classification of A* based best-first heuristic search algorithms," em *Proceedings of the 20th Brazilian conference on Advances in artificial intelligence*, Berlin, Heidelberg, 2010.

[50] T. Fraichard e A. Scheuer, "From Reeds and Shepp's to continuous-curvature paths," *Trans. Rob.,* vol. 20, nº 6, pp. 1025-1035, 2004.

[51] G. Oriolo, G. Ulivi e M. Vendittelli, "Real-time map building and navigation for autonomous robots in unknown environments," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 28, nº 3, pp. 316-333, 1998.

[52] J. R. Souza, D. O. Sales, P. Y. Shinzato, F. S. Osorio e D. F. Wolf, "Template-based autonomous navigation and obstacle avoidance in urban environments," *SIGAPP Appl. Comput. Rev.,* vol. 11, nº 4, pp. 49-59, 2011.