UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
INSTITUTO DE MATEMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RAFAEL FONTELLA KATOPODIS
VINICÍUS GARCIA SILVA DA COSTA

# ELABORAÇÃO DE UM SISTEMA DE CONTROLE DE UM ROBÔ MÓVEL AUTÔNOMO EM AMBIENTE CONHECIDO A PRIORI

RIO DE JANEIRO
2019

RAFAEL FONTELLA KATOPODIS
VINICÍUS GARCIA SILVA DA COSTA

# ELABORAÇÃO DE UM SISTEMA DE CONTROLE DE UM ROBÔ MÓVEL AUTÔNOMO EM AMBIENTE CONHECIDO A PRIORI

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Claudio Miceli de Farias

RIO DE JANEIRO
2019

Katopodis, Rafael Fontella
K19e Elaboração de um sistema de controle de um robô móvel autônomo em ambiente conhecido a priori / Rafael Fontella Katopodis, Vinícius Garcia Silva da Costa. – 2019.

63 f.

Orientador: Claudio Miceli de Farias.

Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Federal do Rio de Janeiro, Instituto de Matemática, Bacharel em Ciência da Computação, 2019.

1. Filtro de Kalman. 2. Robótica móvel. 3. Auto-localização. I. Costa, Vinícius Garcia Silva da. II. Farias, Claudio Miceli de (Orient.). III. Universidade Federal do Rio de Janeiro, Instituto de Matemática. IV. Título.

RAFAEL FONTELLA KATOPODIS
VINICÍUS GARCIA SILVA DA COSTA

# ELABORAÇÃO DE UM SISTEMA DE CONTROLE DE UM ROBÔ MÓVEL AUTÔNOMO EM AMBIENTE CONHECIDO A PRIORI

Trabalho de conclusão de curso de graduação apresentado ao Departamento de Ciência da Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 26 de julho de 2019

BANCA EXAMINADORA:

Prof. Claudio Miceli de Farias,
Orientador
D.Sc. (PPGI-UFRJ)

Profa. Silvana Rossetto
D.Sc. (UFRJ)

Prof. Adriano Joaquim de Oliveira Cruz
Ph.D. (UFRJ)

## RESUMO

Uma das questões mais fundamentais da robótica móvel é o problema da localização: permitir que o robô determine sua posição a partir dos sensores disponíveis. Dado que tais sensores frequentemente produzem leituras com ruído, são utilizados algoritmos de fusão de informação para se obter estimativas precisas da posição do robô. Neste trabalho, buscamos solucionar uma instância do problema da localização proposta pela modalidade "Trekking" da competição Winter Challenge: nela, um robô autônomo deve passar por três cones de trânsito num campo de futebol em posições conhecidas a priori, sinalizando quando estiver a menos de 1 metro de cada um deles. Na solução, foi explorado o uso de um filtro de Kalman estendido, capaz de fundir leituras de um par de encoders rotativos, uma UMI, um GPS e um processo de odometria visual baseado na cascata de Haar. Também foram elaborados dois modelos distintos de atuação para o robô, que se diferenciam em seu uso das informações visuais. Experimentos foram realizados em um ambiente simulado com o software Gazebo. Os resultados obtidos indicam que um dos modelos de atuação é incapaz de completar a prova com qualquer grau de consistência, enquanto o outro a completa na maior parte dos casos.



**Palavras-chave**: Filtro de Kalman. Robótica Móvel. Auto-localização.

## ABSTRACT

One of the most fundamental issues in mobile robotics is the localization problem: for the robot to determine its position based on its available sensors. Given that such sensors frequently produce noisy readings, information fusion algorithms are used to obtain precise estimates of the robot's pose. In the present work, we strive to solve an instance of the localization problem proposed by the "Trekking" modality of the Winter Challenge robotics competition: in it, an autonomous robot must reach three traffic cones in a soccer field placed in positions known a priori, signaling when closer than 1 meter to each of the marks. In the solution, an Extended Kalman Filter (EKF) was used, capable of fusing readings from a pair of rotary encoders, an IMU, a GPS and a visual odometry process based on Haar cascade detection. Further, two distinct actuation models were developed, differing in regard to their use of visual information. Experiments were made in a simulated environment with the Gazebo software. The obtained results show that one of the actuation models is incapable of consistently completing the test trial, while the other succeeds most of the time.



**Keywords**: Kalman Filtering. Mobile Robotics. Self-localization.

# LISTA DE ILUSTRAÇÕES

**SUMÁRIO**

# 1 INTRODUÇÃO

Nas últimas duas décadas, os avanços na robótica móvel fizeram com que o campo expandisse suas aplicações práticas em nossa sociedade. Dentre as áreas afetadas, podemos destacar o ambiente doméstico (como aspiradores de pó, cortadores de grama e limpadores de piscina), militar (drones de reconhecimento e exploração em regiões de conflito), industrial (caminhões de carga autônomos), científico (exploração espacial e marítima) e urbano (carros autônomos). A ubiquidade da robótica móvel cresce, assim como o interesse em trazê-la para novos contextos. Para atingir este fim, torna-se necessária uma compreensão aprofundada dos princípios e desafios do campo (WOLF et al., 2009).

Um dos problemas mais desafiadores e fundamentais da robótica móvel é o da *localização*: permitir que o robô responda com um alto grau de confiança a pergunta "onde estou?". De fato, a importância desta pergunta é consequência direta da dimensão de "mobilidade" que diferencia a robótica móvel da robótica convencional (NEGENBORN, 2003). Caso o robô não conheça sua posição, a tarefa básica de navegação se torna difícil. Consequentemente, os comportamentos mais complexos que dependem desta mobilidade podem ficar comprometidos, e o robô incapaz de executar as tarefas para as quais foi construído em primeiro lugar. Por esta razão, este problema é ocasionalmente referenciado como "o problema mais fundamental no fornecimento de capacidades autônomas a um robô móvel" (COX, 1989).

Porém, apesar do problema ser tão fundamental, ele não admite uma solução única e elegante. A diversidade das aplicações em robótica móvel acarreta em uma diversidade de limitações, tal que a definição de "solução viável" usualmente varia de um contexto para outro. Assim sendo, ao longo das últimas décadas foi desenvolvida uma enorme gama de sensores e técnicas de fusão de dados diferentes para solucionar o problema da localização (BORENSTEIN et al., 1997; KAM; ZHU; KALATA, 1997). Para desenvolver uma aplicação bem-sucedida em robótica móvel, é necessário entender como as soluções pré-existentes interagem com as particularidades físicas e situacionais da aplicação, possivelmente desenvolvendo novas técnicas. Um robô que transita dentro de um ambiente fechado com luzes no teto — como uma casa ou hospital — poderá utilizar estes marcos imóveis facilmente distinguíveis para se localizar (DULIMART; JAIN, 1997). Em contrapartida, um robô que se move em um ambiente desconhecido ao ar livre — como uma planície ou estrada — terá de recorrer a outros métodos de localização, dado que não encontrará marcos deste tipo.

Usualmente, soluções para o problema da localização são inicialmente testadas em simuladores, por providenciarem um ambiente de testes com diversas vantagens sobre a realidade. Uma destas vantagens é a produtividade. Parte fundamental da construção de um sistema de controle é o ciclo constante de testes e ajustes, conforme os parâmetros do

filtro são alterados e otimizados; com um robô físico, este ciclo significa modificar o software do robô e realizar novos testes, constituindo um processo demorado, inconveniente e que desgasta partes do robô e dos sensores. O uso de uma simulação busca mitigar estes problemas, dado que parâmetros temporais podem ser acelerados, dificuldades logísticas são eliminadas e o robô físico permanece intacto. Além da produtividade, um ambiente simulado também permite testes que seriam impossíveis ou inviáveis na realidade. Exemplos clássicos são variações nas condições climáticas (chuva ou neve) e na luminosidade (dia ou noite). Testes como esses não só asseguram o funcionamento adequado do sistema em situações críticas, como também ajudam a estabelecer os limites físicos dentro dos quais o sistema se comporta de forma esperada. Sobretudo, o uso de simulações frequentemente permite responder a pergunta "e se?" de modo muito mais conveniente (ŽLAJPAH, 2008).

## 1.1 OBJETIVOS

A abordagem apresentada neste trabalho busca solucionar o problema proposto pela competição Winter Challenge, promovida pela RoboCore, mais especificamente na modalidade denominada "Trekking" (ROBOCORE, 2018). Nesta modalidade, o robô deve partir de uma posição conhecida em um campo de futebol e realizar um percurso de forma autônoma, passando por três marcos pré-estabelecidos que estão sinalizados com cones de trânsito (a distância máxima entre o robô e o marco deve ser de 1 metro). Adicionalmente, o robô deve anunciar de forma visual ou sonora quando reconhecer que passou por cada marco. O objetivo principal da modalidade é a conclusão do percurso, e o critério de desempate inclui o tempo gasto.

## 1.2 METODOLOGIA

Para atingir o êxito nesta modalidade, neste trabalho visamos a desenvolver o sistema de controle de um robô móvel autônomo na *framework* ROS, capaz de completar a prova no menor tempo possível, dentro dos limites da competição.

O sistema de controle é dividido em duas partes: a localização ("onde o robô está") e a atuação ("o que ele deve fazer, dado que sabe onde está"). Para o problema da localização, utilizamos um filtro de Kalman estendido para estimar a posição e a orientação do robô a partir de quatro sensores distintos: uma unidade de medição inercial (acelerômetro e giroscópio), encoders rotativos, um sistema de posicionamento global e uma câmera capaz de detectar os cones. Para atuar em cima destas estimativas, elaboramos dois modelos de atuação que utilizam as informações visuais de formas diferentes.

Avaliamos a viabilidade do sistema desenvolvido em um conjunto de simulações feitas com a ferramenta Gazebo. Representamos digitalmente o ambiente da prova e realiza-

mos experimentos para analisar o comportamento do robô quando utilizados diferentes conjuntos de sensores, assim como diferentes velocidades e modelos de atuação.

## 1.3 ORGANIZAÇÃO DO TEXTO

O restante deste trabalho está organizado em quatro capítulos. No capítulo 2, são apresentados os conceitos básicos necessários para a compreensão do trabalho, assim como os trabalhos relacionados. No capítulo 3, discutimos a proposta do sistema de controle e a implementação do filtro de Kalman estendido. No capítulo 4, são detalhados os experimentos realizados e a análise dos resultados obtidos. Por fim, no capítulo 5, apresentamos as conclusões finais e as propostas de trabalhos futuros.

# 2 CONCEITOS BÁSICOS

Neste capítulo são apresentados os conceitos básicos que fundamentam os temas abordados e portanto são necessários para o entendimento do trabalho proposto. Discutiremos o problema fundamental do trabalho na seção 2.1. Veremos os tipos de informação aos quais o robô tem acesso na seção 2.2, e na seção 2.3 discutiremos como essa informação é utilizada na resolução do problema. A seção 2.4 aborda a técnica de detecção de objetos utilizada por um dos sensores, e a seção 2.5 trata do modelo de locomoção do robô. Por fim, o capítulo se encerra com uma análise dos trabalhos relacionados na seção 2.6.

## 2.1 PROBLEMA DA LOCALIZAÇÃO

O problema da localização na robótica móvel consiste em determinar a *pose* de um robô (sua posição e orientação) relativa ao ambiente em que ele se encontra. Sendo este um problema tão fundamental, é importante compreendê-lo mais a fundo. Para isso, é útil analisar como as particularidades do problema variam de uma aplicação para outra.

Uma das divisões do problema é entre *localização local* e *localização global*. No problema da localização local, também chamado de problema do rastreamento, o robô conhece sua posição inicial e deve atualizá-la conforme navega pelo ambiente. Já no problema da localização global, o robô não conhece sua posição inicial e deve detectá-la a partir da estaca zero. Alguns subtipos deste problema o complicam adicionalmente, como no caso do *problema do robô raptado*, no qual a qualquer momento o robô pode ser transportado para outra posição no ambiente sem que seja avisado (THRUN; BURGARD; FOX, 2005).

Uma outra dimensão do problema da localização envolve o ambiente, que pode ser *estático* ou *dinâmico*. Em um ambiente estático, apenas o robô se movimenta: a única variável no ambiente é a sua posição. Um ambiente dinâmico, por outro lado, se reconfigura: objetos além do robô podem se movimentar, alguns podem ser removidos e outros adicionados. A grande maioria dos ambientes reais são dinâmicos, devido a elementos como a movimentação de pessoas ao redor do robô, portas que podem se abrir e fechar, e até mesmo variações na luz solar. Contudo, em muitos casos, é possível obter resultados satisfatórios modelando um ambiente dinâmico como se fosse estático. Por exemplo, um robô ao ar livre que permanece ativo durante poucos minutos provavelmente poderá ignorar as variações na luz solar (THRUN; BURGARD; FOX, 2005).

Uma terceira dimensão relevante do problema da localização diz respeito ao mapeamento do ambiente. Em alguns casos, o robô é inicializado com um mapa do ambiente, tendo todas as informações necessárias para planejar suas rotas e desviar dos obstáculos. Em outros casos, o robô não conhece nada a priori e deve se preocupar em obter um mapa do ambiente (THRUN et al., 2002). Técnicas utilizadas para solucionar este pro-

blema envolvem uma fase inicial de exploração (no qual o robô busca conhecer o ambiente desconhecido e criar um mapa do mesmo) ou uma simultaneidade entre o mapeamento e as funções normais do robô (conhecido como *Simultaneous Localization and Mapping*, ou SLAM) (NEGENBORN, 2003; DISSANAYAKE et al., 2001).

Neste trabalho, abordamos o problema do rastreamento em um ambiente modelado de maneira estática. O mapa é conhecido a priori, assim como a posição inicial do robô (Figura 1).

Figura 1 – Mapa da região na qual o robô deverá se locomover. A partir da posição inicial I, o robô deverá percorrer os marcos 1, 2 e 3, nesta ordem.


Fonte: (ROBOCORE, 2018)

## 2.2 SENSORES

Para determinarmos a *pose* do robô, é necessário ter um conjunto de fontes que forneçam as informações de posição e orientação periodicamente, de forma direta ou indireta. Estas fontes são os sensores do robô, e normalmente são divididas de acordo com a natureza do sensoriamento da pose: *relativa* ou *absoluta*.

Sensores *proprioceptivos* baseiam suas medições somente no próprio corpo do robô, ou seja, eles não utilizam nenhuma informação do ambiente em seu sensoriamento. Por conseguinte, todos os sinais fornecidos por um sensor proprioceptivo são relativos às condições iniciais da medição, e estão suscetíveis ao acúmulo de erros. Uma analogia comum é a de fechar os olhos e andar cinco passos à frente: apesar de não termos certeza absoluta de onde estamos, o percurso foi curto o suficiente para termos uma boa ideia. Porém, caso déssemos cem passos ao invés de cinco, estaríamos consideravelmente mais incertos de nossa posição, apesar de continuarmos tendo certeza de quantos passos foram dados. Isto é, na ausência de informação externa, cada movimento incerto adiciona um pouco de

incerteza à nossa posição, e consequentemente esta incerteza cresce com o tempo. Este processo de deduzir a posição atual a partir de posições passadas e movimentos detectados é chamado de *navegação inercial*. Caso não houvesse nenhuma incerteza na medição dos movimentos (por exemplo, se soubéssemos exatamente o comprimento de cada passo dado), a navegação inercial produziria estimativas perfeitas da pose. Contudo, este é raramente o caso, dado que sensores possuem vieses e ruídos pelas mais diversas razões, muitas vezes até mesmo de fábrica. Apesar disso, sensores proprioceptivos são frequentemente utilizados para estimação de pose na robótica móvel (BORENSTEIN et al., 1997; NEGENBORN, 2003).

Em contrapartida, os sensores *exteroceptivos* baseiam suas medições apenas no ambiente. Eles não acumulam incerteza, pois a localização atual é independente das medições anteriores e portanto não é afetada pela incerteza das mesmas. Exemplos de sensores que podem ser utilizados como exteroceptivos de *pose* são câmeras, magnetômetros e sistemas de posicionamento global (NEGENBORN, 2003).

À primeira vista pode parecer que os sensores exteroceptivos são superiores aos proprioceptivos em todos os aspectos, e que portanto devem ser utilizados em todos os casos. Porém, uma análise mais cautelosa revela as nuances da situação. Um sistema de posicionamento global, por exemplo, possui uma incerteza de dezenas de metros, e portanto é inviável de ser utilizado em diversos contextos, como em ambientes domésticos. Uma câmera pode não ser muito útil em um ambiente sem pontos de referência claros, ou de baixa iluminação. A leitura de um magnetômetro é gravemente comprometida quando próxima de ímãs ou correntes elétricas (BACHMANN; YUN; PETERSON, 2004). Nestes três casos, um acelerômetro e um giroscópio (ambos proprioceptivos) poderiam ser utilizados sem maiores limitações. O acúmulo de incertezas pode parecer problemático, mas além de sensores mais precisos, há também diversas técnicas de correção. O ideal, contudo, é uma fusão de sensores dos dois tipos.

### 2.2.1 Unidade de Medição Inercial

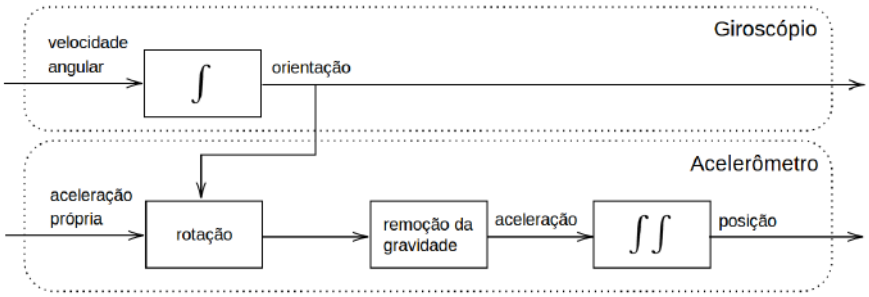
O termo "sensor inercial" denomina um conjunto normalmente composto por um acelerômetro tri-axial e um giroscópio tri-axial. Normalmente, um dispositivo que contém um ou mais sensores inerciais é chamado de Unidade de Medição Inercial (UMI). O objetivo de um sensor inercial, e consequentemente de uma UMI, é providenciar dados sobre a velocidade angular de um corpo (medida pelo giroscópio) e sobre a aceleração própria dele (medida pelo acelerômetro). Ocasionalmente, um magnetômetro tri-axial também é considerado parte de um sensor inercial, e portanto uma UMI poderia oferecer dados a respeito dos campos magnéticos aplicados sobre um corpo. Acelerômetros e giroscópios, por medirem dados internos ao sensor, são classificados proprioceptivos; em contrapartida, o magnetômetro mede uma informação externa ao corpo, e é um exemplo de sensor

exteroceptivo (KOK; HOL; SCHÖN, 2017) (NEGENBORN, 2003). Isto significa que uma UMI pode ser composta por uma mistura de sensores proprioceptivos e exteroceptivos.

UMIs estão presentes em diversos dispositivos do nosso dia-a-dia, notavelmente em *smartphones*. Quando um smartphone detecta que a tela está na horizontal e altera sua perspectiva de "retrato" para "paisagem", é devido a um sensoriamento de seu giroscópio. Outros dispositivos como controles do console Nintendo Wii e headsets de realidade virtual incluem UMIs para detecção de movimentos dos jogadores (KOK; HOL; SCHÖN, 2017).

No campo da robótica móvel, um dos principais usos de UMIs é na área de localização, para estimar a posição e a orientação do robô. Embora uma UMI não produza essas informações diretamente, é possível obtê-las com algumas manipulações algébricas. Integrando a velocidade angular do giroscópio obtemos a orientação do robô, e a partir dela somos capazes de eliminar a influência da gravidade na aceleração própria medida pelo acelerômetro. Assim, obtemos a aceleração linear, e uma dupla integração nos dá a posição. Este processo está ilustrado na Figura 2. O ocasional magnetômetro, ao medir os campos magnéticos aplicados sobre o sensor, pode ser utilizado como fonte adicional de orientação (KOK; HOL; SCHÖN, 2017).

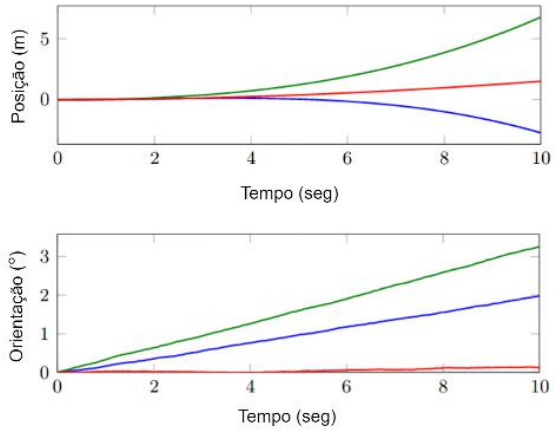
Figura 2 – Esquemática do processo de estimação de *pose* a partir de uma UMI com acelerômetro e giroscópio.




Fonte: (KOK; HOL; SCHÖN, 2017), traduzida pelos autores

Por serem proprioceptivos, estimativas feitas apenas com base no acelerômetro e no giroscópio são suscetíveis ao acúmulo de incertezas. Um exemplo claro pode ser visto na Figura 3: devido ao ruído dos sensores, uma UMI parada numa superfície plana detecta que está se movimentando no espaço. Esta incerteza acumulada é particularmente incipiente sobre a estimativa de posição, dado que ela é afetada tanto pela incerteza da estimativa de orientação (durante a remoção da força da gravidade) quanto por uma integração dupla (ao contrário da orientação, que é integrada unicamente). Portanto, uma UMI é usualmente uma boa fonte de orientação, mas não tanto de posição (BARSHAN; DURRANT-WHYTE, 1995). A incerteza da orientação pode ser adicionalmente mitigada caso um magnetômetro confiável esteja presente, pois se possibilita a fusão dos sinais de orientação do magnetômetro e do giroscópio (KOK; HOL; SCHÖN, 2017).

Figura 3 – Estimativas de posição e orientação a partir da UMI de um Sony Xperia Z5 Compact em repouso na superfície de uma mesa. As estimativas são referentes aos três eixos: $x$ (azul), $y$ (verde) e $z$ (vermelho).




Fonte: (KOK; HOL; SCHÖN, 2017), traduzida pelos autores

### 2.2.2 Encoders Rotativos

Um encoder rotativo é um transdutor que converte as revoluções de um eixo em sinais elétricos. Quando afixado no eixo de uma roda, este dispositivo é chamado de encoder de roda. Na robótica móvel, este tipo de sensor é utilizado para medir a distância percorrida por um robô, caso se saiba a dinâmica do robô e suas características físicas (diâmetro das rodas e distância entre elas). Como nenhuma informação do ambiente é extraída, um encoder de roda é considerado um sensor proprioceptivo, de tal forma que o processo de estimar a posição do robô a partir dos sinais dos encoders é um tipo de navegação inercial normalmente chamado de *odometria* (SIEGWART et al., 2011).

Ao longo de vastas distâncias, a odometria pode apresentar erros de diversos metros, devido ao acúmulo de incertezas resultante de sua natureza proprioceptiva: a relação entre as revoluções das rodas e o deslocamento causado por elas pode mudar de forma imprevisível. Isto pode ser provocado por fatores físicos, como o deslizamento das rodas e irregularidades do terreno, ou por erros na modelagem, como diferenças milimétricas no diâmetro das rodas (SIEGWART et al., 2011). Apesar destas questões, encoders rotativos apresentam boa precisão no curto-prazo, baixo custo e altas taxas de amostragem. Por isso, são amplamente utilizados como fontes de informação para solucionar o problema da localização (NEGENBORN, 2003).

## 2.3 FUSÃO DE SENSORES

Se todo sensor possui um grau de incerteza, é natural questionar se a utilização de mais sensores geraria mais incerteza. Isto nem sempre é verdade. No caso em que os sensores são *redundantes*, isto é, providenciam o mesmo tipo de informação sobre o ambiente,

intuitivamente há uma vantagem estatística em aumentar sua quantidade, dado que um crescimento do número de amostras aumenta o grau de confiança no resultado. Já no caso oposto, em que os sensores são não-redundantes, ou *complementares*, também se pode imaginar uma possível vantagem: quando uma informação estiver oculta para um sensor, pode estar disponível para outro (p.ex.: uma câmera possui dificuldade em detectar objetos muito próximos, mas em contrapartida este é o único caso em que um sensor de proximidade funciona). Uma outra perspectiva não diz respeito à fonte das informações, mas sim às desvantagens de cada sensor: um sensor pode ser mais preciso e operar com baixa frequência, enquanto outro é menos preciso e produz leituras mais frequentes. Sob esta análise, a combinação ideal dos dois sensores uniria suas vantagens, fornecendo sinais precisos e frequentes (HALL; LLINAS, 1997).

Sob que circunstâncias é possível combinar múltiplos sensores incertos e obter sinais mais precisos, e como? A *fusão de sensores* é a disciplina que trata destes problemas. Suas aplicações atravessam áreas como a biomédica (p.ex.: monitoramento de pacientes via múltiplos sinais biológicos), a agrícola (p.ex.: monitoramento do solo e do clima), a militar (p.ex.: vigilância de áreas hostis e rastreamento de alvos aéreos e terrestres) e a industrial (p.ex.: monitoramento da saúde de maquinários e de transportes de carga) (LUO; YIH; SU, 2002). Sua utilidade na robótica é evidente, de tal modo que o problema da localização pode ser interpretado como um problema de fusão de sensores: como melhor estimar a posição do robô dado seus diversos sensores. Portanto, podemos buscar a solução deste problema nas técnicas desenvolvidas na área de fusão de sensores, mais especificamente na categoria dos métodos de *estimativa* (KAM; ZHU; KALATA, 1997).

Métodos de estimativa buscam estimar o estado de um sistema a partir de um ou mais vetores de medição. Exemplos populares incluem Máxima Verossimilhança, Mínimos Quadrados, Média Móvel, Filtros de Kalman e Filtros de Partículas. Destes, os dois últimos são os mais adequados para o caso em que o estado do ambiente muda aproximadamente de acordo com um modelo conhecido (como a dinâmica do corpo), e portanto são os mais utilizados na robótica móvel (NAKAMURA; LOUREIRO; FRERY, 2007).

Diversos trabalhos já foram realizados comparando a precisão destes dois métodos (ARULAMPALAM et al., 2002) (THRUN, 2002). Filtros de Kalman são os mais precisos quando o ruído dos sensores pode ser aproximado como uma distribuição gaussiana; quando este não é o caso, os Filtros de Partícula obtêm maior precisão, embora a um custo computacional maior (FOX et al., 2001).

### 2.3.1 Filtro de Bayes

Dentre a pletora de técnicas de estimação de estados existentes, um conjunto, do qual o filtro de Kalman e o filtro de Partículas são membros, se destaca por se apoiar de forma bastante elegante em conceitos da estatística Bayesiana. O processo de estimar estados pode ser descrito informalmente como a operação de extrair quantias de interesse

de medições sensoriais, que individualmente carregam apenas informações parciais e são corrompidas por ruído. Essa atividade pode ser vista à luz da teoria de probabilidade: leituras de sensores são interpretadas não como resultados de uma função determinística, mas sim como amostras de uma distribuição estocástica. No contexto de robótica móvel estamos interessados em três distribuições em todo instante de tempo $t$: a de observações feitas pelos sensores, $Z_t$, a de ações realizáveis pelo robô, $U_t$, e a de *poses* possíveis, $X_t$.

Nesse enquadramento, podemos fazer uso de leis probabilísticas para modelar os processos associados a essas distribuições. Para a evolução da pose do robô ao longo do tempo, é razoável esperar, em um primeiro momento, que a pose em um dado instante, $x_t$, esteja condicionada a todas as poses, medições e ações passadas e também à ação atual. No entanto, se assumirmos que nosso estado é completo, isso é, que a pose passada é o melhor preditor para a próxima pose, $x_t$ torna-se dependente apenas da pose que a precede e da ação sendo realizada. A probabilidade resultante é chamada de probabilidade de transição de estado ou modelo de processo (equação 2.1). De forma similar, podemos modelar o processo que dá origem às observações. A mesma suposição de independência torna a próxima medição condicionada apenas no estado atual. Essa, é chamada de probabilidade ou modelo de medição (equação 2.2).

$$p(x_t|x_{0:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t|x_{t-1}, u_t) \tag{2.1}$$

$$p(z_t|x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t|x_t) \tag{2.2}$$

A linguagem da teoria da probabilidade ainda pode ser usada para formalizarmos o que o robô deve computar para estimar seu estado. Mesmo que sejamos incapazes de medir a pose real do robô, podemos ter maior ou menor confiança em uma determinada pose com base no que observou e nas ações que realizou. Essa intuição sugere a definição do conceito de *crença*: a distribuição probabilística de estados mantida internamente pelo robô e condicionada em todas as informações disponíveis (definição 2.3). Com isso, o problema de estimação da pose de um robô se resume a computar essa crença. Isso naturalmente leva à pergunta: como calcular essa crença?

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}) \tag{2.3}$$

O *Filtro de Bayes* provê um método para realizar esse cálculo. Ele é obtido por indução na definição de crença e realizando algumas suposições de independência e completude de estado. Pode ser entendido como realizando dois passos: primeiro uma *predição* é realizada com base na crença anterior, no modelo de processo e na ação realizada e, em seguida, uma

*atualização*, com base na predição, probabilidade de medição e na mais recente observação. O método também pode ser visto como um pseudo-algoritmo (Algoritmo 1).

$$\overline{bel}(x_t) = \int p(x_t|x_{t-1}, u_t) bel(x_{t-1}) dx_{t-1} \tag{2.4}$$

$$bel(x_t) = \eta p(z_t|x_t) \overline{bel}(x_t) \tag{2.5}$$

**Algoritmo 1:** Filtro de Bayes para estimação de estados

1 função $bayes\_filter$ $(\overline{bel}(x_{t-1}), u_t, z_t)$;
  **Entrada:** Distribuição de crença corrente, sinal de comando e medição atuais
  **Saída :** Nova distribuição de crença
2 **foreach** $x_t$ **do**
3 | $\overline{bel}(x_t) \leftarrow \int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$;
4 | $bel(x_t) \leftarrow \eta p(z_t|x_t) \overline{bel}(x_t)$;
5 **end**
6 return $bel(x_t)$

Infelizmente, o algoritmo acima não pode ser implementado diretamente. Computadores não podem realizar integrais sem recorrer a alguma técnica numérica e o algoritmo não faz menção a quais são as distribuições probabilísticas de transição de estado, medição e crença inicial e como essas podem ser representadas em memória. Implementações concretas, como o filtro de Kalman e suas variantes e o filtro de partículas existem exatamente para endereçar esses problemas.

### 2.3.2 Filtro de Kalman

Dentre as técnicas de estimação de estado que implementam o filtro Bayesiano, o filtro de Kalman é, de longe, a mais popular (FOX et al., 2003). Formulado em 1960 (KALMAN, 1960), esse algoritmo tem sido amplamente usado desde então em uma gama variada de problemas, desde a modelagem de sistemas biológicos (SäRKKä, 2013) até sistemas de navegação de foguetes (GREWAL; ANDREWS, 2010). Como mais uma confirmação de sua grande generalidade, a técnica também encontrou, nos últimos anos, aplicações no campo de robótica móvel.

Uma das principais razões para essa popularidade decorre de sua simplicidade: o método pode ser resumido a algumas poucas operações matriciais. Isso por conta de algumas suposições feitas sobre o problema que tornam o filtro não apenas computacionalmente viável como também bastante eficiente. A primeira suposição feita é a de que crenças podem ser apropriadamente modeladas por gaussianas. Com isso, todas as distribuições envolvidas no filtro podem ser representadas e manipuladas através de dois parâmetros apenas: um vetor de médias e uma matriz de covariâncias.

A segunda suposição é a de que o processo a ser estimado pode ser modelado pela seguinte relação gaussiana linear:

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t$$

Aqui, $x_i$ é o estado do sistema no instante $i$, $u_i$ é um sinal de controle nesse instante, $A_i$ é uma matriz que descreve como o sistema evolui no tempo sem influência de um sinal de controle, $B_i$ uma matriz que descreve como um sinal de controle afeta o estado do sistema e, por fim, $\varepsilon_i$ é um ruído gaussiano branco, respondendo pela incerteza no processo.

Como exemplo, imagine que estamos acompanhando um carro em movimento ao longo de um segmento reto de estrada com uso de um filtro de Kalman. Nesse contexto, nosso estado de interesse, $x_i$, pode ser a posição e velocidade do carro. $A_i$ descreve o que acontecerá com esse carro se, imprudentemente, nos abstivermos de comandá-lo. Como ele já está em movimento, é esperado que seu estado mude, mesmo na ausência de qualquer sinal de controle. $B_i$ descreve como comandos de aceleração e mudança de direção, através dos pedais e volante, afetam o sistema. Por último, um sistema complexo como um carro em movimento no mundo real está sujeito a um infindável número de variáveis, grande demais para que todas sejam contabilizadas em uma modelagem. Nós lidamos com isso abstraindo o efeito dessas variáveis não modeladas como ruído no sistema, e o termo $\varepsilon$ cumpre esse papel.

A terceira e última suposição de Kalman é a de que medições podem ser modeladas pela relação gaussiana linear:

$$z_t = C_t x_t + \delta_t$$

Aqui, $C_i$ é uma matriz que permite o mapeamento do espaço de estado para o espaço de medições. Esses dois espaços não são necessariamente os mesmos. Retornando a nosso exemplo do carro, nosso espaço de estado tem dimensões de posição e de velocidade. Do ponto de vista de medição, no entanto, poderíamos estar acompanhando acelerações. Portanto, temos dois espaços diferentes. Não obstante, eles estão linearmente relacionados pelas equações de movimento de Newton. $\delta_i$ é um ruído branco que nos permite lidar com a incerteza na medição.

Podemos agora associar essas suposições ao filtro de Bayes e observar suas implicações nos cálculos da estimação de estado:

$$\overline{bel}(x_t) \;=\; \int \underbrace{p(x_t|u_t, x_{t-1})}_{\sim N(x_t;\, A_t x_{t-1} + B_t u_t,\; Q_t)} \quad \underbrace{bel(x_{t-1})}_{\sim N(x_{t-1};\, \mu_{t-1},\; \Sigma_{t-1})} dx_{t-1}$$

A etapa de predição, torna-se essencialmente uma soma de gaussianas, que resulta também em uma gaussiana com parâmetros:

$$\overline{bel}(x_t) = \begin{cases} \overline{\mu_t} = A_t\mu_{t-1} + B_t u_t \\ \overline{\Sigma_t} = A_t\Sigma_{t-1}A_t^T + Q_t \end{cases} \tag{2.6}$$

Já a etapa de correção torna-se um produto de gaussianas:

$$\begin{aligned} bel(x_t) \;&=\; \eta \underbrace{p(z_t|x_t)}_{\Big|\;\sim N(z_t; C_t x_t,\; R_t)} \underbrace{\overline{bel}(x_t)}_{\sim N\left(x_t; \overline{\mu}_t,\; \overline{\Sigma}_t\right)} \end{aligned}$$

Que também resulta em uma gaussiana de parâmetros:

$$bel(x_t) = \begin{cases} \mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t) \\ \sigma_t = (I - K_t C_t)\overline{\Sigma}_t \end{cases} \tag{2.7}$$

Colocando tudo isso em conjunto, chegamos no Algoritmo 2.

**Algoritmo 2:** Filtro de Kalman para estimação de estados

1 função kf $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$;
**Entrada:** Média e covariância correntes da distribuição e sinal de comando e medição atuais
**Saída :** Média e covariância da nova distribuição de crença
2 $\overline{\mu}_t \leftarrow A_t\ \mu_{t-1} + B_t\ u_t$;
3 $\overline{\Sigma}_t \leftarrow A_t\ \Sigma_{t-1}\ A_t^T + R_t$;
4 $K_t \leftarrow \overline{\Sigma}_t\ C_t^T\ (C_t\ \overline{\Sigma}_t\ C_t^T + Q_t\ )^{-1}$;
5 $\mu_t \leftarrow \overline{\mu}_t + K_t\ (z_t\text{- }C_t\ \overline{\mu}_t)$;
6 $\Sigma_t \leftarrow (\text{I - }K_t\ C_t)\overline{\Sigma}_t$;
7 return $\mu_t$, $\Sigma_t$

### 2.3.3 Filtro de Kalman Estendido

As suposições feitas pelo filtro de Kalman o tornam uma solução ótima e extremamente eficiente para o filtro Bayesiano. Supomos que nossa crença inicial no estado pode ser representada por uma gaussiana e que predições e novas observações que atualizem essa crença não tiram essa distribuição da normalidade. Infelizmente, essas mesmas suposições também tornam o filtro impossível de ser aplicado em todos, exceto os mais simples problemas de estimação de estados.

Mais especificamente, a suposição de preservação da normalidade é feita impondo que os modelos do processo e de medição sejam lineares. No entanto, vários problemas de

estimação de estado, incluindo muitos de robótica móvel, envolvem dinâmicas não-lineares. Retornando a nosso exemplo corrente de um carro em movimento, seu estado poderia ser satisfatoriamente acompanhado por um filtro de Kalman padrão enquanto navegasse em uma trajetória retilínea, mas assim que tomasse uma saída à direita e descrevesse uma trajetória curva, as estimativas produzidas se tornariam progressivamente desconexas da realidade. Isso acontece pois curvas são inerentemente não-lineares.

O filtro de Kalman, entretanto, ainda pode ser aplicado a dinâmicas mais complexas, mediante uma modificação de seu algoritmo: a adição de uma etapa de linearização, que garante que as distribuições envolvidas permaneçam gaussianas. O filtro de Kalman Estendido realiza isso aproximando as funções não-lineares envolvidas como seus respectivos polinômios de Taylor de 1º grau. Como exemplo, o polinômio de Taylor de i-ésima ordem, centrado em um ponto $a$, de uma função arbitrária pode ser determinado por:

$$f(a) = \sum_{n=0}^{i} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{2.8}$$

Assim, as condições do filtro podem ser relaxadas. Agora, a suposição é de que a probabilidade de transição de estado e as probabilidades de medições são governadas por funções não-lineares $g$ e $h$, respectivamente:

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \tag{2.9}$$

$$z_t = h(x_t) + \delta_t \tag{2.10}$$

A cada passo discreto do filtro, retas que melhor aproximam essas funções são determinadas. A linearização é realizada nas redondezas de um ponto. Para o problema de estimação de estados, é apenas natural que escolhamos como ponto central aquele que temos maior confiança de representar o estado real — a média de nossa distribuição de crença. Assim, o algoritmo aproxima as funções $g$ e $h$ como:

$$\begin{aligned} g(u_t, x_{t-1}) &\approx g(u_t, \mu_{t-1}) + g'(u_t, \mu_{t-1})(x_{t-1} - \mu_{t-1}) \\ &= g(u_t, \mu_{t-1}) + G_t(x_{t-1} - \mu_{t-1}) \end{aligned} \tag{2.11}$$

$$\begin{aligned} h(x_t) &\approx h(\overline{\mu}_t) + h'(\overline{\mu}_t)(x_t - \overline{\mu}_t) \\ &= h(\overline{\mu}_t) + H_t(x_t - \overline{\mu}_t) \end{aligned} \tag{2.12}$$

Ao realizar essas linearizações, voltamos a trabalhar apenas com transformações que preservam a normalidade da distribuição de crença. Isso, por sua vez, nos permite implementar o filtro de forma muito similar a vista anteriormente. (Algoritmo 3).

**Algoritmo 3:** Filtro de Kalman estendido para estimação de estados

1 função ekf $(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$;
**Entrada:** Média e covariância correntes da distribuição e sinal de comando e medição atuais
**Saída :** Média e covariância da nova distribuição de crença
2 $\overline{\mu}_t \leftarrow \mathrm{g}(u_t, \mu_{t-1})$;
3 $\overline{\Sigma}_t \leftarrow G_t \; \Sigma_{t-1} \; G_t^T + R_t$;
4 $K_t \leftarrow \overline{\Sigma}_t \; H_t^T \; (H_t \; \overline{\Sigma}_t \; H_t^T + Q_t \;)^{-1}$;
5 $\mu_t \leftarrow \overline{\mu}_t + K_t \; (z_t\text{- } \mathrm{h}(\overline{\mu}_t))$;
6 $\Sigma_t \leftarrow (\mathrm{I} - K_t \; H_t)\overline{\Sigma}_t$;
7 return $\mu_t$, $\Sigma_t$

## 2.4 CASCATA DE HAAR

Em (VIOLA; JONES et al., 2001), os autores apresentam uma *framework* de detecção de objetos em imagens que se popularizou sob o nome de "Cascata de Haar", ou "algoritmo Viola-Jones de detecção de faces". O trabalho original é demonstrado no problema de detecção de faces humanas, porém o método é genérico e funciona para outros objetos, como semáforos e cones de trânsito.

O treinamento de um detector de objetos deste algoritmo funciona em quatro etapas: (I) seleção de atributos, (II) criação de imagens integrais, (III) classificação de atributos e (IV) cascata de classificadores.

Na etapa (I), são selecionados os atributos disponíveis aos classificadores. Os atributos são representados por retângulos brancos e pretos (Figura 4) e seguem um modelo pseudo-Haar. O artigo original utiliza por volta de 180.000 atributos distintos, e embora uma implementação da *framework* possa usar seus próprios atributos, em geral o conjunto original é mantido. Os atributos são fundamentais ao funcionamento do algoritmo, visto que "detectar uma face em uma imagem" é "detectar uma sub-área da imagem na qual boa parte dos atributos de uma face estão presentes". A presença de um atributo é representado por um valor escalar, calculado como a soma da intensidade de cor dos *pixels* sob a área branca, subtraída da soma da intensidade de cor dos *pixels* sob a área preta.

Na etapa (II), são criadas as imagens integrais. Uma imagem integral é uma estrutura de dados criada para reduzir a carga computacional do algoritmo, por armazenar valores que terão de ser utilizados diversas vezes ao longo do processo de detecção. Estas imagens são construídas atribuindo a cada *pixel* um valor que corresponde à soma dos valores na imagem original de todos os *pixels* acima e à sua esquerda, incluindo a si próprio.

Na etapa (III), é empregado o algoritmo AdaBoost de aprendizado de máquina junto com um conjunto de imagens *positivas* (em que o objeto a ser detectado está presente) e *negativas* (em que o objeto a ser detectado está ausente) para que se selecione os atributos

Figura 4 – Dois atributos utilizados pela técnica da cascata de Haar, aplicadas na figura de uma face.

Fonte: (VIOLA; JONES et al., 2001)

que melhor descrevem o objeto a ser detectado. Nesta etapa, também é utilizado o AdaBoost para treinar os classificadores.

Na etapa (IV), é feita a suposição de que para a maior parte das sub-áreas de uma imagem, não há faces a serem detectadas. Portanto, ao invés de testar todos os atributos em cada sub-área da imagem, é inicialmente testado apenas o classificador do melhor atributo (aquela que melhor separa os exemplos positivos dos negativos). Caso uma sub-área falhe neste classificador, ela é abandonada e nenhum outro classificador passa por ela; porém caso o teste seja bem-sucedido, a sub-área será redirecionada para o segundo melhor classificador, e o mesmo processo se repetirá. Esta é a chamada cascata de classificadores, e é responsável por uma redução significativa no custo computacional do algoritmo.

As etapas (I) e (III) são empregadas somente no treinamento dos classificadores, enquanto as etapas (II) e (IV) são executadas tanto no treinamento quanto na inferência.

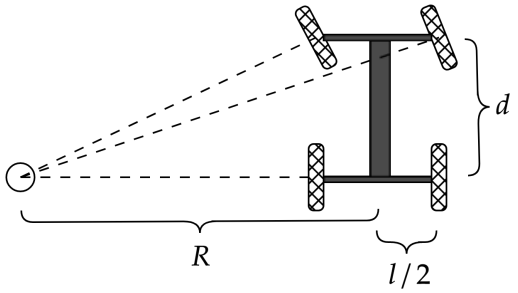
## 2.5 GEOMETRIA E CINEMÁTICA DE ACKERMANN

Compreender a geometria e cinemática de um robô móvel é necessário se quisermos ser capazes de relacionar o movimento de seus atuadores ao deslocamento do veículo como um todo. Digamos que sabemos, através de encoders, a velocidade angular das rodas de um robô similar a um carro. Como determino seu movimento geral? De modo inverso, digamos que queremos realizar um determinado movimento com esse robô, como estabelecer uma velocidade linear e angular. Quais devem ser os sinais de controle para as rodas? Essas duas perguntas constituem a base de duas tarefas essenciais em robótica autônoma — odometria e atuação — e ressaltam, ambas, a necessidade de se conhecer o mapeamento entre o movimento do robô como um todo e a ação de suas partes móveis.

A geometria de Ackermann é o mecanismo típico de direção usado em veículos de quatro rodas. Nesse arranjo, as rodas dianteiras são acopladas de tal maneira que formam ângulos ligeiramente diferentes quando realizando uma curva (Figura 5). O propósito é

que os eixos de todas as rodas apontem para o mesmo centro de curvatura. Se esse não fosse o caso e as rodas dianteiras formassem o mesmo ângulo, por exemplo, seria inevitável que algum conjunto delas derrapasse lateralmente.

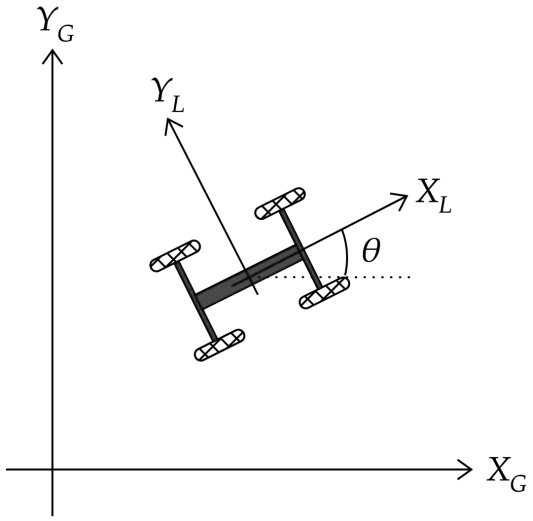
Figura 5 – Geometria de Ackermann




Antes de discutir a cinemática de um veículo que usa a geometria de Ackermann, é importante definir dois referenciais: um global, de origem arbitrária, e outro local, rigidamente ligado ao corpo do robô (Figura 6). Do ponto de vista do referencial local, a velocidade linear do carro sempre será observada ao longo do eixo $X_l$ e a velocidade angular, ao redor do eixo $Z_l$. Ambas essas velocidades podem ser determinadas a partir das velocidades lineares das rodas traseiras do veículo, como ilustrado na equação 2.13.

$$v = \frac{v_r + v_l}{2} \qquad\qquad w = \frac{v_r - v_l}{l} \tag{2.13}$$

Figura 6 – Referenciais local e global




Determinar o deslocamento do ponto de vista do referencial global, então, se resume

a rotacionar (no sentido horário) o deslocamento observado pelo referencial local (2.14).

$$\begin{vmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{vmatrix} = \begin{vmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} v \\ 0 \\ w \end{vmatrix} \tag{2.14}$$

Esse vetor, por sua vez, pode ser integrado para obter uma estimativa de posição.

## 2.6 TRABALHOS RELACIONADOS

Dado que o nosso trabalho envolve um problema específico com limitações específicas que atravessa diversas áreas, é apenas natural que não tenha nenhum trabalho similar ao nosso em todos os aspectos. Porém, há diversos trabalhos que, individualmente, estão relacionados com problemas que nós abordamos.

Em (KHATIB et al., 2015), é apresentado um método de localização e controle para um robô móvel diferencial. Os autores utilizam um filtro de Kalman estendido para fundir as informações provenientes de quatro sensores: encoders rotativos, GPS, UMI e magnetômetro. É feita uma simulação e são comparadas três diferentes combinações dos sensores, e é concluído que a combinação que produz o melhor resultado é a que funde todos os sensores, apoiando a máxima de que mais sensores trazem uma estimativa mais precisa. Nossa proposta é similar a (KHATIB et al., 2015), pois também pretendemos apresentar diferentes combinações de sensores com um filtro de Kalman estendido. Porém, além de termos acesso a um outro conjunto de sensores (não temos magnetômetro mas temos odometria visual), a distância percorrida pelo nosso robô é muito maior, e portanto o efeito de deriva é mais preocupante.

Em (SUMANARATHNA et al., 2014), também é feita uma simulação que utiliza um filtro de Kalman estendido para fundir sensores diferentes (encoder, GPS e acelerômetro). Todos os sensores têm suas leituras convertidas para a posição $x$ e $y$ do robô no plano cartesiano, ou seja, o estado do sistema é modelado como um vetor de duas variáveis. Um diferencial deste trabalho é que ao invés de utilizar um único filtro para fundir todas as medições, como (KHATIB et al., 2015), são testadas três abordagens. A primeira delas é utilizar um filtro de Kalman que funde a leitura dos encoders com a do GPS. A segunda envolve utilizar um filtro de Kalman para fundir o resultado do filtro da primeira abordagem com a leitura do acelerômetro. A terceira e última abordagem envolve um algoritmo que troca entre a primeira e a segunda abordagem, dependendo da discrepância entre a leitura dos encoders e os sinais de controle enviados pelo sistema do robô. Caso os encoders estejam muito discrepantes, é porque o robô está em terreno escorregadio e/ou falho, então é melhor utilizar a segunda abordagem (o acelerômetro). Porém, caso esse não seja o caso e o terreno esteja liso, o acelerômetro pioraria as estimativas, então a primeira abordagem seria preferível. Os autores concluem que o melhor resultado não é utilizar

todos os sensores, diferente da conclusão de (KHATIB et al., 2015). A diferença deste trabalho para o nosso é que possuímos mais sensores, e o ruído dos que temos em comum é significativamente diferente (nossos encoders possuem, por exemplo, uma resolução bem menor que os deles).

Em (GANGANATH; LEUNG, 2012), começamos a ver exemplos de odometria visual sendo considerada na fusão de sensores. Os autores apresentam um modelo de localização de um robô móvel que funde informação de encoders com informação de odometria visual, extraída a partir de um Kinect. São afixados cinco círculos de cores diferentes em um mapa em cinco posições conhecidas a priori, e os círculos são diferenciados a partir de um processamento de imagem que utiliza transformada de Hough. Assim que um marco é detectado por Hough, o Kinect é utilizado para detectar a distância do robô ao marco, assim como a orientação entre os dois. Deste modo, se obtém a posição do robô no mapa a partir da odometria visual, e esta informação é fundida com a informação dos encoders por um filtro de Kalman estendido e por um filtro de partículas. Os autores concluem que o filtro de partículas possui um erro médio menor do que o de Kalman. Em termos de odometria visual, o diferencial do nosso trabalho é que estamos utilizando uma câmera monocular ao invés de um sensor de profundidade. Portanto, não é tão simples obter a distância ao marco. Além disso, o Kinect utilizado pelos autores tem alcance de no máximo 3.5 metros, sendo que idealmente desejaríamos um alcance maior.

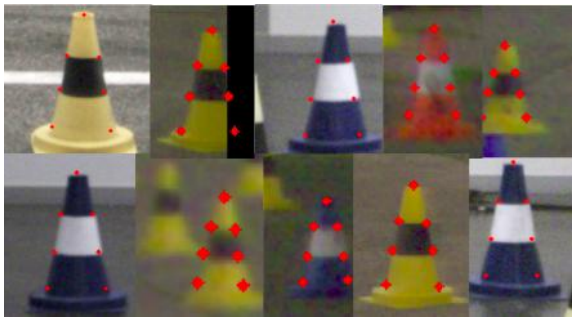
Em (WU; JOHNSON; PROCTOR, 2005), os autores exploram o uso de um filtro de Kalman estendido para fundir informações de UMI, magnetômetro e odometria visual em um veículo aéreo não-tripulado. A odometria visual extrai informações de posição do veículo em relação a um marco artificial (uma janela de cor preta num fundo branco). São realizados testes numa simulação e numa situação real, com o veículo voando ao redor do marco em um loop fechado. Os autores obtiveram um bom desempenho, e notaram uma dependência forte da orientação providenciada pelo magnetômetro, já que uma detecção na imagem pode ter sido gerada por diversas poses diferentes, e a orientação é essencial para determinar com precisão quais dessas poses possíveis é a correta. A abordagem de odometria visual utilizada por (WU; JOHNSON; PROCTOR, 2005) é muito próxima da nossa, embora tenhamos mais sensores e tenhamos de nos locomover por distâncias maiores, portanto tendo de passar certo intervalo de tempo sem a odometria visual. Nosso diferencial é elaborar uma solução mais apropriada para o conjunto de sensores em mãos, assim como estimar a posição global do robô, ao invés da relativa.

Em (KIM; LYOU, 2009), os autores utilizam apenas uma câmera monocular e um giroscópio para estimarem a posição de um robô com um filtro de Kalman estendido. A câmera é apontada para o teto, onde estão afixados retângulos pretos que contrastam com o fundo branco e servem como marcos. A partir do momento em que um marco é detectado, a posição do robô pode ser obtida por relações trigonométricas, dado que a distância entre o marco e o chão é conhecida (a distância do chão ao teto). A orientação

do robô é obtida se aproveitando do formato retangular dos marcos — o ângulo entre os vértices opostos do retângulo é utilizado para deduzir a orientação da câmera, e portanto do robô. O diferencial do nosso trabalho é que não possuímos a simplificação de saber a distância entre o marco e o chão, e portanto as relações trigonométricas precisam ser redefinidas. Além disso, não podemos saber a orientação do robô a partir apenas de uma imagem, dado que o cone é radial e não apresenta nenhuma diferença visual dependendo do ângulo de que for visto, se os ângulos estiverem no mesmo plano paralelo ao chão.

Especificamente sobre a detecção de cones de trânsito em veículos autônomos, podemos destacar (DHALL; DAI; Van Gool, 2019). Nele, os autores empregam uma pipeline para detectar cones a partir de uma câmera monocular, com três etapas distintas. Na primeira etapa, os cones são detectados na imagem e uma *bounding box* é desenhada ao seu redor, utilizando a rede neural convolucional YOLO (REDMON et al., 2016). Na segunda etapa, em cada cone detectado é aplicada uma outra rede neural convolucional para extrair as coordenadas de 7 pontos-chave na imagem (Figura 7). Na terceira etapa, é solucionado o problema do *Perspective-n-Point* para estimar a pose tridimensional de cada um dos cones detectados, pois todas as suas informações físicas são conhecidas (altura, largura, diâmetro etc). No contexto específico dos autores, também há uma competição, embora os cones estabeleçam os limites da prova e não marcos a serem percorridos. A abordagem utilizada lhes garante um bom desempenho. O diferencial do nosso trabalho é que estamos lidando com dispositivos computacionais significativamente menos poderosos, que são incapazes de executar grande parte dos algoritmos utilizados por (DHALL; DAI; Van Gool, 2019), como o YOLO. Além disso, não conhecemos todas as características físicas dos cones a priori, apenas o seu visual.

Figura 7 – Ilustração da detecção de 7 pontos-chave em cada cone, para estimar sua posição no ambiente tridimensional.



Fonte: (DHALL; DAI; Van Gool, 2019)

# 3 PROJETO DE UM SISTEMA DE CONTROLE DE ROBÔ

Neste capítulo detalhamos o projeto do sistema de controle proposto. Na seção 3.1, apresentamos as bibliotecas de software utilizadas na implementação do trabalho. Na seção 3.2, detalhamos as características físicas relevantes do robô que será simulado. Na seção 3.3, apresentamos o sistema de controle, assim como os diferentes modelos de atuação. Na seção 3.4, é apresentado o funcionamento da odometria visual, que extrai informações de pose a partir das imagens filmadas pela câmera. Por fim, na seção 3.5, detalhamos a implementação do filtro de Kalman estendido no contexto das estimativas do robô.

## 3.1 BIBLIOTECAS AUXILIARES

Nesta seção, apresentamos as bibliotecas auxiliares que foram utilizadas para a implementação deste trabalho. Na subseção 3.1.1, introduzimos o conjunto de bibliotecas ROS, utilizado como base para a arquitetura de todo o sistema de controle do robô. Na subseção 3.1.2, abordamos a ferramenta Gazebo, na qual foi desenvolvido o ambiente de simulação utilizados nos experimentos.

### 3.1.1 ROS

Embora cada aplicação em robótica tenha suas particularidades, certas necessidades aparecem com frequência independente da aplicação. Dentre elas, podemos destacar a comunicação entre dispositivos computacionais diferentes, a passagem de dados entre processos, e a implementação recorrente das mesmas funcionalidades básicas. Uma das ferramentas utilizadas para atender a estas necessidades é o Robot Operating System (ROS) (QUIGLEY et al., 2009): um conjunto de bibliotecas de código aberto destinadas a auxiliar o desenvolvimento de software na área de robótica.

O ROS funciona com base num modelo *publish-subscribe*, em que *nós* trocam *mensagens* via *tópicos*. Um nó é um processo que realiza computação, e pode ser entendido como um "módulo de software". Em um exemplo genérico, um *nó* poderia representar um sensor coletando informações de temperatura, e outro nó poderia ser o computador central ao qual ele envia os dados coletados. Os dados coletados em si seriam uma *mensagem*, que nada mais é que uma estrutura de dados estritamente tipada. Esta mensagem seria publicada pelo sensor num *tópico* — uma cadeia de caracteres, por exemplo *"/temperatura/"* — e o computador central poderia recebê-la caso estivesse inscrito neste tópico.

O uso do ROS traz diversas vantagens. Uma delas é a facilidade proporcionada pela abstração de tópicos quanto à organização de troca de dados. As preocupações de gerenciamento de fila e comunicação entre dispositivos computacionais diferentes são abstraídas com o uso do ROS, permitindo ao desenvolvedor pensar no sistema como uma rede *peer-*

*to-peer* em que os nós trocam informações entre si. Outra vantagem se dá devido à popularidade do ROS — por ter uma comunidade ativa, há diversos pacotes de código aberto disponíveis para a solução de rotinas corriqueiras e essenciais, como por exemplo transformações entre sistemas de coordenadas e algoritmos de planejamento de rota. A instalação destes pacotes permite ao desenvolvedor se focar nas particularidades alto-nível do problema que tem em mãos, aumentando sua eficiência (STARANOWICZ; MARIOTTINI, 2011). Uma outra vantagem é a facilidade de integração do ROS com diferentes simuladores, como Player, OpenRAVE e Gazebo (HARRIS; CONRAD, 2011).

### 3.1.2 Gazebo

O Gazebo (KOENIG; HOWARD, 2004) é uma *engine* de simulação de código aberto voltada à área de robótica, capaz de representar múltiplos robôs em ambientes tridimensionais (Figura 8). O Gazebo não faz parte do projeto ROS, porém ambos são altamente integrados, de tal forma que o software executado num robô simulado pelo Gazebo está muito próximo do software em ROS que deve ser rodado num robô físico. Esta integração contínua é uma das principais vantagens de utilizar o Gazebo em projetos feitos em ROS, visto que reduz a complexidade de transitar entre um ambiente simulado e sua contraparte real.

O Gazebo é uma ferramenta robusta de simulação, possuindo uma vasta gama de funcionalidades que o torna adequado para aplicações razoavelmente distintas, como braços robóticos industriais e carros autônomos de diversas geometrias (como a de Ackermann). A ferramenta é capaz de simular tanto ambientes cobertos quanto exteriores, e conta com uma extensa biblioteca de modelos 3D para aumentar a fidelidade dos ambientes representados. É possível montar rapidamente um escritório ou um campo de futebol com os modelos oferecidos pela ferramenta. Ela também possui uma biblioteca de diversos sensores que podem ser adicionados ao corpo do robô virtual, gerando sinais com distribuições de incerteza parametrizáveis pelo usuário. Esta biblioteca conta por exemplo com UMIs, GPSs, e LIDARs, e também é oferecido ao usuário uma interface extensível para o desenvolvimento de novos sensores, como encoders rotativos. Também é possível acoplar uma câmera ao robô virtual e processar as imagens gravadas, permitindo a simulação adequada de módulos de visão computacional.

## 3.2 ARQUITETURA DO ROBÔ

Apesar de avaliarmos as soluções desenvolvidas neste trabalho apenas em um ambiente simulado, o objetivo principal é solucionar um problema no mundo real, com um robô real (Figura 9). Portanto, algumas características físicas deste robô devem ser respeitadas para que as soluções desenvolvidas sejam consideradas viáveis.

Figura 8 – Ilustração do ambiente Gazebo.

Fonte: (CABRITA; MADHAVAN; MARQUES, 2015)

Dentre as características físicas que são relevantes para este trabalho, destacamos três. Em primeiro lugar, os sensores disponíveis. O robô real possui um conjunto limitado de sensores, e estes que devem ser utilizados para solucionar o problema. Em segundo lugar, os dispositivos computacionais. O robô real, por ser móvel, apresenta limitações de poder computacional que devem ser consideradas no desenvolvimento da solução. Em terceiro e último lugar, a dinâmica do carro. Para que a simulação seja fidedigna e tenha valor, é preciso que o carro simulado se locomova de forma similar ao real, e interaja com o mundo de forma similar a como o real interagiria.

Quanto à dinâmica do carro, é suficiente dizer que a plataforma usada é um carro RC 1:8 com geometria de Ackermann (detalhada na seção 2.5), e que portanto o veículo simulado também deve se adequar a estas características. Já os demais aspectos relevantes da dimensão física estão detalhados nas subseções seguintes.

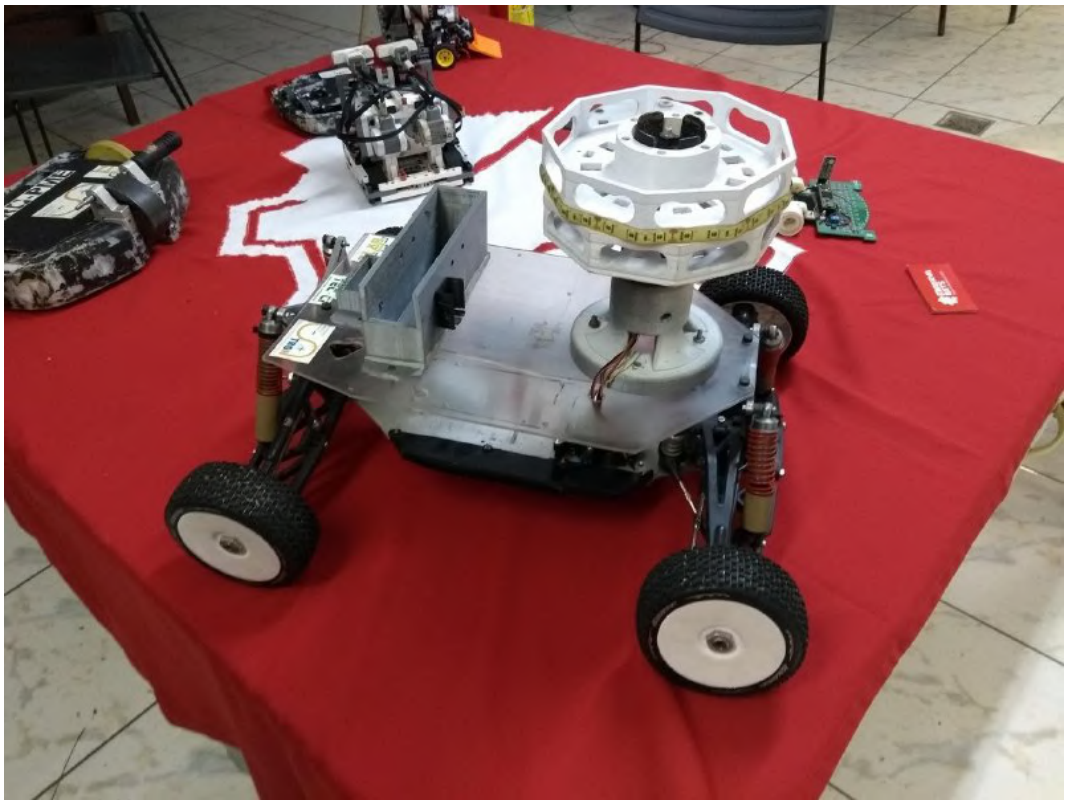
### 3.2.1 Sensores

O robô real apresenta quatro sensores distintos: (i) uma UMI (com acelerômetro e giroscópio tri-axiais), (ii) um GPS, (iii) uma câmera monocular, e (iv) dois encoders rotativos (um em cada roda traseira). Os sensores (i), (ii) e (iii) se encontram embarcados no mesmo dispositivo: um smartphone Motorola G5 cujos recursos computacionais também são utilizados. Os encoders rotativos são acoplados separadamente nas rodas e utilizam foto-interruptores para detectar as revoluções.

### 3.2.2 Dispositivos Computacionais

O robô possui três principais dispositivos computacionais: um Arduino, um Raspberry Pi e um smartphone.

Figura 9 – Fotografia do robô móvel físico a ser simulado.



O Raspberry Pi é responsável por integrar a comunicação entre sensores e atuadores, pois executa todas as rotinas essenciais do núcleo ROS. Também neste dispositivo, ocorre a estimação da pose do robô pelo filtro de Kalman, assim como a decisão de como atuar a partir desta estimativa. Por ter essas responsabilidades, é justificado entendê-lo como sendo o "cérebro" do robô.

O Arduino, por sua vez, é utilizado para estabelecer a comunicação entre o Raspberry Pi e alguns dispositivos de baixo-nível, como por exemplo os encoders rotativos e os dois atuadores do robô (locomoção e direção).

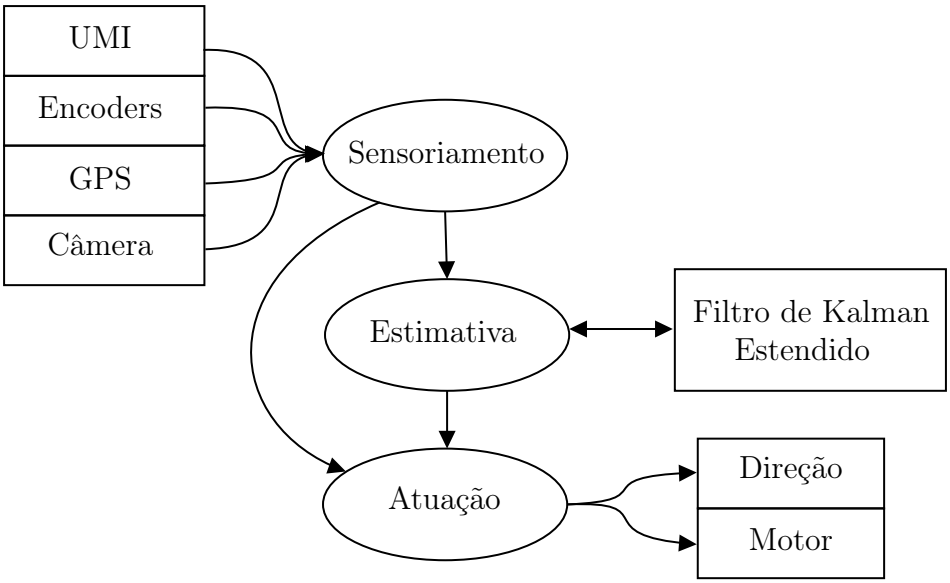
Por fim, o smartphone está encarregado da odometria visual, e portanto carrega o maior esforço computacional dos três dispositivos. Suas tarefas envolvem coletar as imagens de sua própria câmera, executar os algoritmos de detecção de objetos e por fim extrair dados de pose a partir destas informações. Seus resultados são enviados ao Raspberry Pi.

## 3.3 ARQUITETURA DO SOFTWARE

O objetivo desta seção é apresentar uma visão geral de como os diferentes módulos do sistema de controle do robô interagem entre si, e que tipo de informação é trocada entre eles. Para isso, podemos separar o sistema em três módulos distintos: sensoriamento, estimativa e a atuação.

O módulo de *sensoriamento* é responsável por coletar as informações dos sensores e

Figura 10 – Ilustração da arquitetura do software.




processá-las de modo a poderem ser introduzidas no filtro de Kalman. Ele não apenas serve como uma interface com o hardware de baixo-nível (como UMI e encoders), como também executa rotinas de processamento (por exemplo, toda a extração de posição da odometria visual). Cada sensor possui uma rotina de processamento distinta. Toda vez que um sensor produz uma nova medição, ela é tratada apropriadamente e publicada no respectivo tópico ROS.

O módulo de *estimativa* é o responsável por executar o filtro de Kalman e por publicar a estimativa de pose em um tópico ROS. Este módulo está inscrito em todos os tópicos do módulo de sensoriamento, e toda vez que um novo dado é publicado, ele atualiza o filtro adequadamente. Caso nenhum dado seja publicado em um determinado intervalo de tempo, o módulo continua atualizando seu estado baseado no modelo dinâmico do robô. Isto garante que a estimativa continue sendo produzida mesmo quando houver problemas nos sensores, ou quando é desejada uma série de estimativas mais granular do que os sinais coletados pelos sensores.

O módulo de *atuação* é responsável por utilizar tanto a pose do módulo de *estimativa* quanto outras informações do módulo de *sensoriamento* para decidir quais ações o robô deve realizar para corrigir seu trajeto. Isto não significa necessariamente atuar toda vez que uma nova estimativa é publicada — usualmente, este módulo terá de atuar em uma frequência diferente do módulo de estimativa. Esta frequência será determinada primariamente pelos requisitos do hardware dos atuadores.

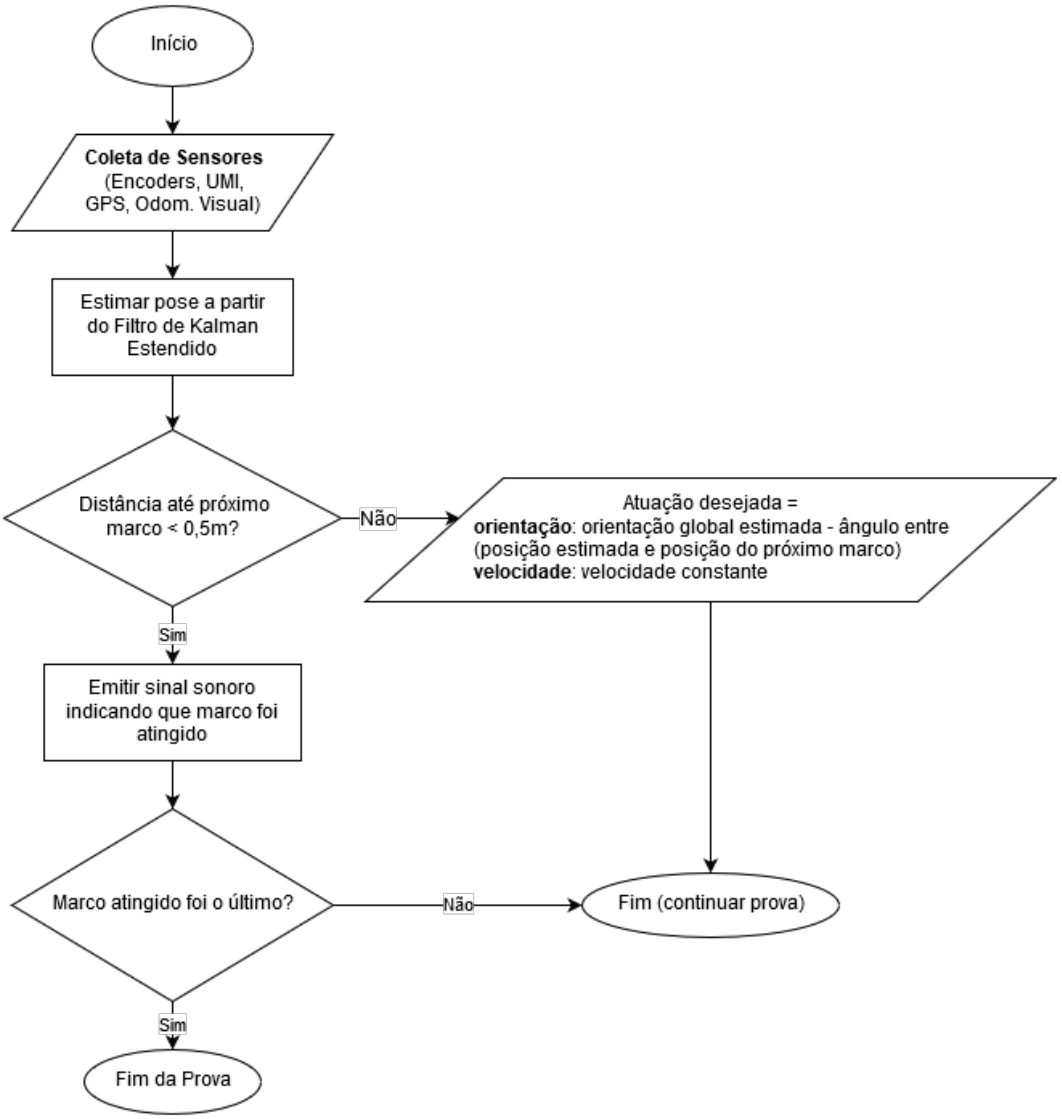
### 3.3.1 Modelos de Atuação

Há diversas formas de integrar as informações de estimativa e sensoriamento para gerar sinais de atuação. A escolha desta integração pode ser a diferença entre o sucesso

e o fracasso do robô, mesmo quando consideradas as mesmas informações de entrada. Neste trabalho, apresentamos dois modelos de atuação alternativos: um *generalista* e um *especialista*. Os dois diferem principalmente quanto ao uso da odometria visual, e ambos são avaliados experimentalmente no capítulo 4.

O modelo de atuação generalista está ilustrado no fluxograma da Figura 11. Neste modelo, o papel da odometria visual é fornecer uma estimativa de pose global (como apresentado na seção 3.4.4), sendo fundida pelo filtro de Kalman com os outros três sensores para se obter uma melhor estimativa de pose global. A atuação é realizada somente com base nessa estimativa: a velocidade de locomoção é constante, e a direção do robô é calculada de modo que o robô sempre aponte para o próximo marco. O modelo é chamado ”generalista” pois ele utiliza poucas informações específicas da prova, e sua arquitetura poderia ser facilmente reutilizada em qualquer situação com sensores e marcos a serem percorridos.

Figura 11 – Fluxograma detalhando o funcionamento do modelo de atuação generalista.



Por outro lado, o modelo de atuação especialista (Figura 12) usufrui das particularidades da prova em questão e fornece maior destaque à odometria visual. Ele se baseia na hipótese de que a partir do momento que a câmera visualiza um cone, guiar-se pelas estimativas do filtro de Kalman provoca mais prejuízo do que ganho, visto que as incertezas dos outros sensores podem reduzir a qualidade da informação providenciada pela câmera. Quando nenhum cone estiver visível, o robô se locomove como no modelo generalista. Porém, caso um cone esteja sendo detectado, o modelo especialista busca se guiar apenas pela odometria visual: o robô se orienta de modo a manter o cone no centro da tela e avança com velocidade constante. Eventualmente, o robô se encontrará tão próximo do cone que o cone deixará de ser detectado — neste instante, o robô emite o sinal sonoro sinalizando que encontrou o marco.

Em teoria, somente este processo já deveria ser satisfatório para o modelo especialista. Contudo, a detecção de objetos possui suas próprias incertezas, de tal modo que falsos positivos podem ocorrer e resultar no robô sinalizando fora dos marcos. Para mitigar essa possibilidade, antes de realizar qualquer sinalização, o robô valida sua posição com as estimativas do filtro de Kalman: um marco é apenas detectado como alcançado quando a fusão de sensores indica que o robô está próximo do cone.

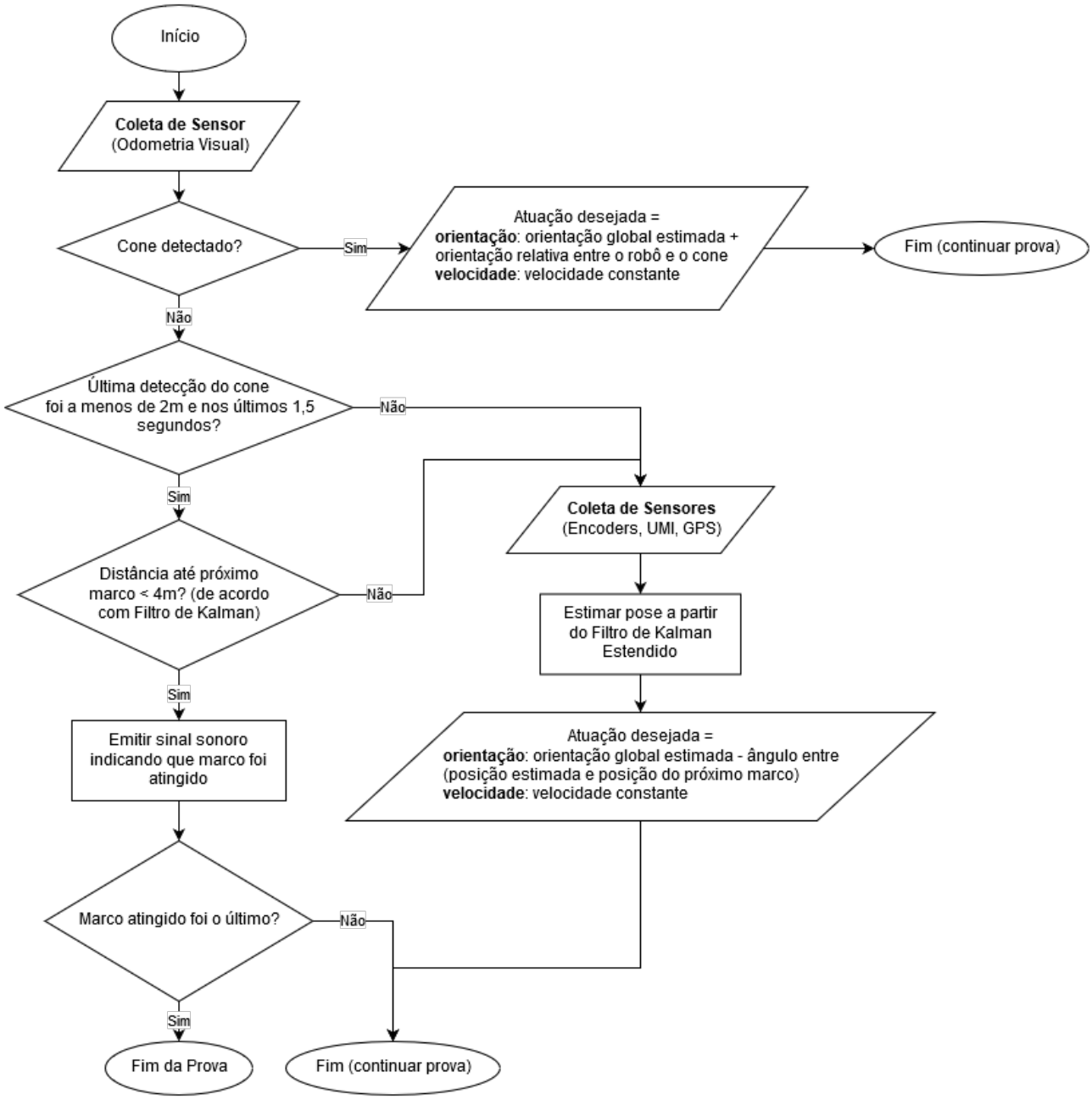
## 3.4 ODOMETRIA VISUAL

Tanto a UMI quanto os encoders rotativos são sensores proprioceptivos, e portanto, suscetíveis à deriva. Como a prova envolve percorrer uma larga extensão de terreno e realizar diversas curvas, há um perigo real de erros serem acumulados no longo prazo, fazendo com que estes sensores desviem insatisfatoriamente da posição absoluta. Portanto, há uma necessidade por sensores exteroceptivos que não acumulem erros. Um destes sensores é o GPS, porém ele possui suas próprias desvantagens: frequência de atualização baixa (cerca de 1 Hz) e raio de erro demasiado alto para as particularidades da prova. Seria útil ter um sensor exteroceptivo com uma frequência de atualização mais elevada e um raio de erro menor.

Um dos sensores ao qual o robô tem acesso é uma câmera monocular. Como todos os marcos são sinalizados com cones de trânsito (um objeto particularmente visível e contrastante), a câmera foi amplamente explorada como valiosa fonte de informação. A partir da detecção de um cone em um frame, é possível estimar tanto a distância entre o robô e o cone quanto a orientação do robô relativa ao cone. Estas informações, por sua vez, podem ser fundidas para se obter a posição global do robô.

O processo geral de obtenção da pose de um robô a partir de informações visuais é usualmente denominado *odometria visual*. Neste trabalho, ele ocorre em quatro passos: (i) detecção de objetos, (ii) estimativa de orientação relativa, (iii) estimativa de distância, e (iv) estimativa de posição. As próximas subseções detalham cada um destes passos.

Figura 12 – Fluxograma detalhando o funcionamento do modelo de atuação especialista.




### 3.4.1 Detecção de Objetos

Para extrair qualquer informação útil a partir de uma imagem, é necessário detectar que objetos estão presentes nela. No contexto do presente trabalho, isso significa detectar os cones. É necessário saber não apenas se há cones na imagem (reconhecimento de objetos), como também onde estão os cones na imagem, isto é, as coordenadas em pixels de cada cone (detecção de objetos).

Há diversas tecnologias que buscam solucionar o problema da detecção de objetos. Uma alternativa que se popularizou bastante nos anos recentes é a de redes neurais convolucionais, como SSD (LIU et al., 2016) e YOLO (REDMON et al., 2016). A literatura

apresenta casos em que essa abordagem produziu resultados satisfatórios, inclusive para detecção de cones de trânsito em robôs autônomos (DHALL; DAI; GOOL, 2019). Porém, uma das desvantagens do uso de redes neurais é o seu custo computacional. Apesar de existirem arquiteturas de redes neurais focadas em obter bom desempenho em aplicações móveis, como o Mobilenet (HOWARD et al., 2017), a etapa de inferência ainda requisita um poder de processamento que muitos dispositivos embarcados mal possuem.

Em testes preliminares, foi observado que dos três dispositivos computacionais disponíveis (Arduino, Raspberry Pi e smartphone), apenas o smartphone atingiu os requisitos mínimos da rede neural de menor custo computacional (Mobilenet SSD) (HOWARD et al., 2017); os outros dois dispositivos não dispunham dos recursos de memória e de processamento necessários. Ainda assim, o smartphone apresentou uma baixa taxa de quadros por segundo (5,54 FPS). O modelo utilizado foi o divulgado pelo projeto TensorFlow Lite (GOOGLE LLC, 2017), e havia sido treinado com as 80 classes do dataset COCO (LIN et al., 2014). Usualmente, questões de baixo desempenho deste tipo são evitadas com o uso de dispositivos computacionalmente mais poderosos — como placas GPU da linha Nvidia Jetson, que oferecem o poder computacional de placas gráficas para aplicações móveis. Porém, estas soluções fogem do escopo deste trabalho pois não figuravam entre os dispositivos disponíveis. Também foi cogitada a possibilidade de reduzir o tempo de inferência da rede neural reconfigurando-a para detectar apenas 1 classe — a do cone de trânsito. Porém, priorizamos buscar outras alternativas devido ao tempo e recursos necessários para um treinamento completo da rede neural.

Feita esta análise preliminar, a viabilidade do uso de redes neurais convolucionais depende da velocidade desejada para o robô. Caso o robô esteja se locomovendo a uma velocidade plausível de 60 km/h, uma detecção de objetos a 5.54 FPS estará analisando aproximadamente 1 frame a cada 3 metros. Isto significa que a detecção de objetos estará sempre com um atraso de 3 metros. Tendo em vista que a distância máxima entre o robô e um marco deve ser de 1 metro, este erro é considerável. A eliminação desse atraso requisitaria a redução da velocidade do robô — uma troca entre corretude e velocidade. Como o segundo critério de vitória da prova é o tempo de trajeto, foram exploradas alternativas computacionalmente menos custosas.

A alternativa selecionada foi a *framework* Viola–Jones de detecção de objetos a partir de características pseudo-Haar (VIOLA; JONES et al., 2001), popularmente conhecida como "cascata de Haar". Esta abordagem possui vantagens e desvantagens em relação às redes neurais discutidas anteriormente. A principal vantagem é o tempo de execução: nos nossos testes, a cascata de Haar obteve uma detecção média de 12,6 quadros por segundo, mais do que o dobro do obtido pela Mobilenet SSD. Dentre as desvantagens, podemos destacar que (i) a cascata de Haar é treinada para detectar apenas um objeto (como faces humanas), enquanto os modelos atuais de redes neurais convolucionais são capazes de detectar múltiplas classes de objetos simultaneamente, e (ii) a cascata de Haar

possui comparativamente menor precisão na detecção. A limitação a uma única classe não é um problema, pois na modalidade de competição em questão é necessário detectar apenas cones de trânsito. A menor precisão, por sua vez, também não se apresentou como um problema; em testes preliminares em situações reais, o cone se apresentou um objeto distinto o suficiente para ser detectado por uma cascata de Haar treinada com cones de trânsito. No contexto dos experimentos realizados, este algoritmo se apresentou satisfatório.

### 3.4.2 Estimativa de Orientação Relativa

A partir de uma fotografia com um cone e as coordenadas em pixels deste cone (delimitadas por um retângulo), é possível obter diversas informações a respeito da relação entre a câmera e o cone. Para nossos cálculos, faremos duas pressuposições importantes: (i) um quadro obtido pela câmera nunca incluirá mais de um cone detectável (uma pressuposição sensata, dada a larga distância entre os marcos quando comparada com o campo de visão da câmera) e (ii) se um cone foi detectado pela câmera, este cone corresponde ao próximo marco a ser alcançado.

A primeira informação a se extrair de uma imagem com um cone detectado é a orientação do robô em relação ao cone. Uma das formas mais simples de realizar essa estimativa é a partir da modelagem de uma câmera estenopeica (ou *pinhole*) (HARTLEY; ZISSERMAN, 2003). Esta modelagem é uma simplificação, pois não considera nenhum tipo de distorção (como barril ou olho de peixe), porém isto não é um problema no nosso caso pois o processo de calibração da câmera envolve remover estas distorções. A orientação relativa é então obtida partir da seguinte equação:

$$\alpha = arctan\left(\frac{k}{f}\right) \tag{3.1}$$

Sendo $k$ a distância em *pixels* entre a imagem do cone e o centro da fotografia, e $f$ a distância focal da lente. A relação entre as propriedades está ilustrada na Figura 13.

### 3.4.3 Estimativa de Distância

Também é necessário saber a distância $d$ do robô ao cone. A Figura 14 nos fornece as relações necessárias para obter $d$ em função de outras variáveis conhecidas:

$$\frac{w_{real}}{w_{foto}} = \frac{d}{f/cos(\alpha)} \tag{3.2}$$

Sendo $f$ a distância focal da lente, $\alpha$ a orientação do robô em relação ao cone, e $w_{real}$ e $w_{foto}$ a largura do cone real e na foto respectivamente.

Figura 13 – Relação entre as dimensões de orientação no mundo físico e na fotografia tirada pelo robô, como modelado por uma câmera estenopeica.

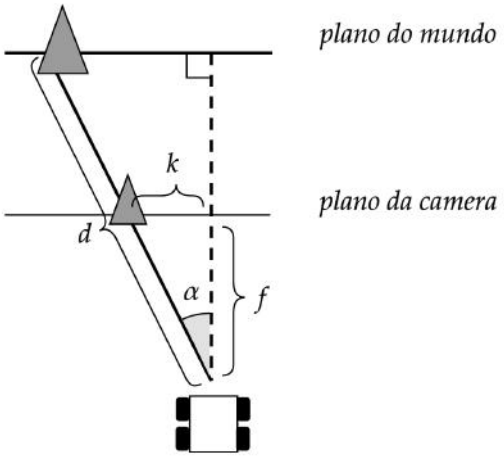
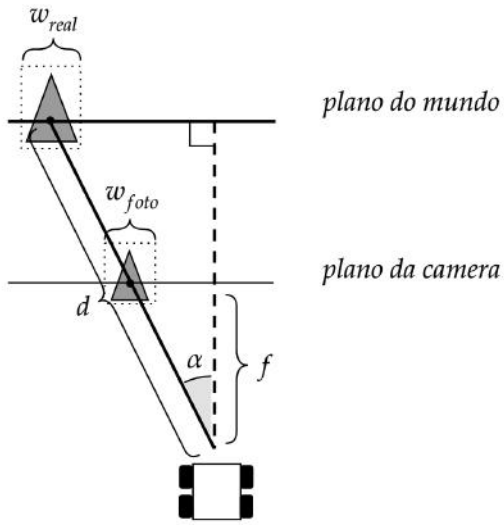



Figura 14 – Relação entre as dimensões de distância no mundo físico e na fotografia tirada pelo robô, como modelado por uma câmera estenopeica.




### 3.4.4 Estimativa de Posição Global

Para estimar, enfim, a posição do robô no mapa, são necessárias outras duas informações cuja obtenção não é trivial: a orientação global do robô $\theta_g$ e a posição global do cone $(x_c, y_c)$. Veremos em seguida como obter estas duas informações.

A orientação global do robô pode ser obtida a partir das próprias estimativas do filtro de Kalman. Embora a princípio possa parecer contraintuitivo utilizar informações estimadas para produzir outras estimativas, não é um processo sem precedentes. Por exemplo, a estimativa de aceleração do acelerômetro depende da retirada do vetor da gravidade a partir da estimativa da orientação global. A inclusão da orientação global $\theta_g$ adicionará mais incerteza na odometria visual, porém será necessário para estimar a posição.

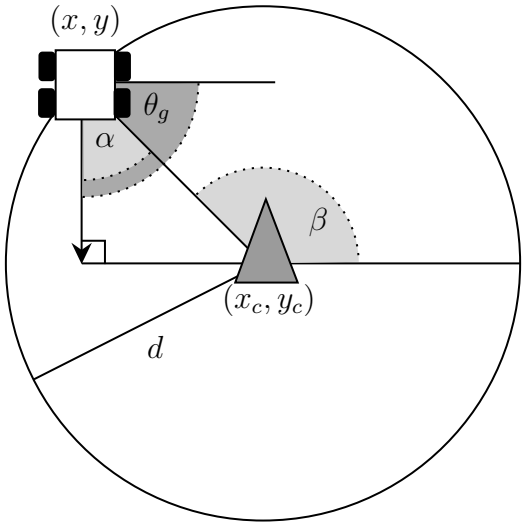
Por sua vez, a posição global de cada um dos três cones é conhecida a priori. Portanto, para saber a posição global do cone que está na fotografia, basta saber qual dos cones está sendo visualizado. Isto é solucionado com nossas duas pressuposições, pois as coordenadas deste cone sempre são as coordenadas do próximo marco a ser alcançado.

Com estas duas informações e as estimativas obtidas a partir das imagens da câmera, podemos calcular a posição global do robô de acordo com as equações abaixo, ilustradas na Figura 15.

$$\beta = 180^\circ - \theta_g + \alpha \tag{3.3}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x_c + d.cos(\beta) \\ y_c + d.sin(\beta) \end{pmatrix} \tag{3.4}$$

Figura 15 – Ilustração da obtenção da posição global do robô a partir da odometria visual.




## 3.5 IMPLEMENTAÇÃO DO FILTRO DE KALMAN ESTENDIDO PARA LOCALIZAÇÃO

Nesta seção, é detalhada a implementação do filtro de Kalman estendido usado para estimar a pose do robô. A subseção 3.5.1 delimita quais são os elementos do filtro que precisam ser definidos para que uma implementação concreta seja feita. A subseção 3.5.2 discorre sobre o modelo de locomoção, usado na etapa de predição do estimador. Finalmente, a subseção 3.5.3 desenvolve os modelos de medição para a câmera, receptor GPS e UMI, usados nas atualizações das estimativas.

### 3.5.1 Componentes do Filtro

Quando o filtro de Kalman estendido foi apresentado na seção 2.3.3, foram especificados os elementos que caracterizam um estimador: a variável de estado e sua média e

covariância iniciais; o modelo de locomoção e sua covariância associada; e o modelo de medição e covariância associada. No entanto, ainda é uma pergunta em aberto o que esses elementos são em termos concretos e no contexto do sistema sendo desenvolvido.

Essa pergunta, e sua resposta, são, de fato, o que projetar um filtro envolve: a formulação de Kalman define apenas um arcabouço. São as propriedades dos sensores a serem colocados em uso e a dinâmica do robô que determinam os parâmetros dessa estrutura e que irão ser explorados a seguir.

### 3.5.2 Modelo de Locomoção

O deslocamento descrito pelo robô é modelado como um movimento determinístico, função das velocidades linear e angular determinadas pela odometria, adicionalmente perturbado por um termo estocástico, de forma a levar em conta a incerteza nas velocidades executadas e efeitos externos não modelados. Se partirmos de uma pose $s_{t-1} = (x_{t-1}, y_{t-1}, \theta_{t-1})^T$ e mantivermos um sinal de controle $u_t = (v, \omega)^T$ (velocidades linear e angular, respectivamente) por um intervalo de tempo $\Delta t$, o robô irá descrever um arco cujo centro é determinado pelas equações 3.5 e 3.6. Com isso, é possível projetar a próxima pose ao final do intervalo através da equação 3.7. Essa é a função de locomoção usada pelo estimador de localização.

$$x_{\text{centro}} = x_{t-1} - \frac{v}{\omega} sen\theta_{t-1} \tag{3.5}$$

$$y_{\text{centro}} = y_{t-1} - \frac{v}{\omega} cos\theta_{t-1} \tag{3.6}$$

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = g(s_{t-1}, u_t) = \begin{bmatrix} x_{t-1} - \frac{v}{\omega} sen\theta_{t-1} + \frac{v}{\omega} sen(\theta_{t-1} + \omega\Delta t) \\ y_{t-1} + \frac{v}{\omega} cos\theta_{t-1} - \frac{v}{\omega} cos(\theta_{t-1} + \omega\Delta t) \\ \theta_{t-1} + \omega\Delta t \end{bmatrix} \tag{3.7}$$

É possível determinar a matriz jacobiana dessa função, tomando a derivada parcial de cada uma das linhas da função relativo às variáveis de estado:

$$G_t(\theta_{t-1}) = \begin{bmatrix} 1 & 0 & -\frac{v}{\omega} cos\theta_{t-1} + \frac{v}{\omega} cos(\theta_{t-1} + \omega\Delta t) \\ 0 & 1 & -\frac{v}{\omega} sen\theta_{t-1} + \frac{v}{\omega} sen(\theta_{t-1} + \omega\Delta t) \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

### 3.5.3 Modelos de Medições

O modelo de medição nos informa que observações esperaríamos ser feitas pelos sensores dado que estamos em um estado conhecido. Com isso, somos capazes de contrastar a expectativa de medição com a realidade e atualizar nossa crença na pose do robô. Aqui é onde a fusão sensorial efetivamente ocorre: como as medidas de todos os sensores (não

importando seus diferentes princípios físicos) são relacionados ao mesmo estado, por meio de suas funções de medição, podemos fazer uso de todos os instrumentos disponíveis para aprimorar a estimativa de pose. É necessário, então, definir quais são essas funções para cada um dos sensores.

#### 3.5.3.1 Função de Medição da Câmera

Como explorado na seção 3.4, o emprego de um detector de objetos às imagens produzidas pela câmera a torna efetivamente um sensor de distância e direção de cones. Se partimos do pressuposto que conhecemos a pose corrente do robô, $s_t$, e a posição do cone avistado, $P_{\text{cone}}$, determinar o que o sensor *deveria* perceber é uma simples questão de geometria: a distância esperada pode ser computada pelo comprimento do segmento de reta que liga o robô ao cone e sua direção relativada, pelo ângulo de inclinação dessa reta subtraído do ângulo de direção do robô (equação 3.9). A equação 3.10 mostra sua respectiva matriz Jacobiana.

$$\begin{bmatrix} d \\ \alpha \end{bmatrix} = h_{\text{câmera}}(s_{t-1}, P_{\text{cone}}) = \begin{bmatrix} \sqrt{(y_{t-1} - y_{\text{cone}})^2 + (x_{t_1} - x_{\text{cone}})^2} \\ \arctan(\frac{y_{\text{cone}} - y_{t-1}}{x_{\text{cone}} - x_{t-1}}) - \theta_{t-1} \end{bmatrix} \tag{3.9}$$

$$H_{\text{câmera}}(s_{t-1}, P_{\text{cone}}) = \begin{bmatrix} \frac{x_{t-1} - x_{\text{cone}}}{h(x_{t-1}, P_{\text{cone}})} & \frac{y_{t-1} - y_{\text{cone}}}{h(x_{t-1}, P_{\text{cone}})} & 0 \\ \frac{y_{\text{cone}} - y_{t-1}}{(x_{cone} - x_{t-1})^2 + (y_{\text{cone}} - y_{t-1})^2} & \frac{x_{t-1} - x_{\text{cone}}}{(y_{\text{cone}} - y_{t-1})^2 + (x_{\text{cone}} - x_{t-1})^2} & -1 \end{bmatrix} \tag{3.10}$$

#### 3.5.3.2 Função de Medição do GPS

Dentre todos os sensores empregados no projeto, o receptor GPS é o que tem a função de medição mais trivial. Isso acontece pois as medidas que toma são variáveis do vetor de estado, sem qualquer conversão. Portando, se assumimos que o robô está em uma determinada posição do campo, essa posição é exatamente a medição que esperamos. Essa função e sua matriz Jacobiana associada são mostradas nas equações 3.11 e 3.12.

$$\begin{bmatrix} x \\ y \end{bmatrix} = h_{GPS}(s_{t-1}) = \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} \tag{3.11}$$

$$H_{\text{GPS}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.12}$$

#### 3.5.3.3 Função de Medição da UMI

Unidades inerciais usualmente são empregadas como fonte de odometria. Sistemas de navegação automotiva, por exemplo, fazem uso de UMIs para efetuar predições de trajetória e leituras de GPS para corrigí-las. No entanto, o par de encoders já responde por essa informação odométrica. No sistema proposto, portanto, esse sensor é posto

em uso em etapas de atualização, em conjunto com a câmera e o receptor GPS. Essa aplicação impõe à função de medição (equação 3.13) e sua jacobiana (equação 3.14) uma certa dificuldade que as destaca das demais: elas envolvem derivadas que não podem ser calculadas analiticamente a priori. É necessário, então, recorrer a um método de diferenciação numérica, como o de diferenças finitas, para computar essas funções durante a execução do filtro.

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \dot{\theta} \end{bmatrix} = h_{\text{UMI}}(s_{t-1}) = \begin{bmatrix} \frac{d^2 x_{t-1}}{dt^2} \\ \frac{d^2 y_{t-1}}{dt^2} \\ \frac{d\theta}{dt} \end{bmatrix} \tag{3.13}$$

$$H_{\text{UMI}} = \begin{bmatrix} \frac{\partial}{\partial x_{t-1}} \frac{d^2 x_{t-1}}{dt^2} & 0 & 0 \\ 0 & \frac{\partial}{\partial y_{t-1}} \frac{d^2 y_{t-1}}{dt^2} & 0 \\ 0 & 0 & \frac{\partial}{\partial \theta_{t-1}} \frac{d\theta}{dt} \end{bmatrix} \tag{3.14}$$

# 4 EXPERIMENTOS E ANÁLISES DOS RESULTADOS

Neste capítulo são descritos cinco experimentos que visam avaliar o impacto de cada sensor nas estimativas do filtro de Kalman, assim como comparar o desempenho dos dois modelos de atuação e avaliar a relação entre as estimativas do filtro e a velocidade do robô. Todos os experimentos foram realizados por meio de simulações. Na seção 4.1, é descrito o ambiente de software utilizado nos experimentos. Na seção 4.2, é detalhado o cenário físico dos experimentos, assim como o objetivo do robô neste cenário. Na seção 4.3, discutimos as diferenças entre o ambiente simulado e a realidade. A seção 4.4 apresenta as métricas utilizadas nos experimentos. Por fim, a seção 4.5 detalha os resultados obtidos em cada experimento, assim como a análise de tais resultados.

## 4.1 DESCRIÇÃO DO AMBIENTE

Os experimentos foram implementados utilizando o arcabouço de robótica ROS, na versão Kinetic (OPEN SOURCE ROBOTICS FOUNDATION, 2016). Para o desenvolvimento dos nós que compõem o sistema de controle, foram utilizadas as linguagens de programação C++ e Python. As simulações foram feitas com o Gazebo, versão 7.0.0. As detecções produzidas pela cascata de Haar foram mantidas a cerca de 10 quadros por segundo, para reproduzir a capacidade de processamento do hardware embarcado.

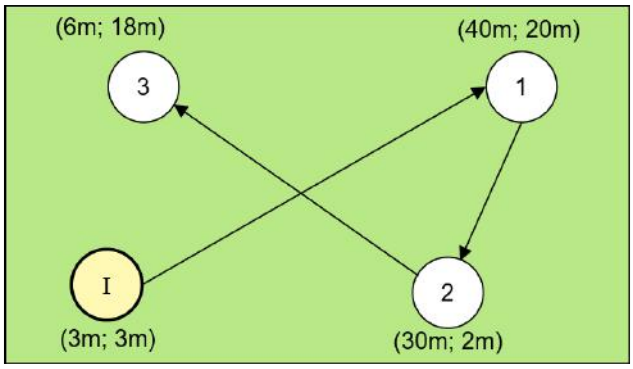
## 4.2 DESCRIÇÃO DO CENÁRIO

O mesmo cenário é empregado em todos os experimentos. Neste cenário, o objetivo do robô é partir da posição $(3m; 3m)$ e passar por três marcos sinalizados por cones de trânsito, nas coordenadas $(40m; 20m)$, $(30m; 2m)$ e $(6m; 18m)$. Os marcos devem ser percorridos nesta ordem. Para que seja reconhecido que o robô passou por um marco, ele deve sinalizar quando estiver a menos de 1m do cone; adicionalmente, o robô deve sinalizar apenas três vezes por "prova". O cenário está ilustrado na Figura 16.

## 4.3 DISCUSSÃO DAS DIFERENÇAS ENTRE A SIMULAÇÃO E A REALIDADE

Apesar do sistema proposto buscar soluções para um problema no mundo real, todos os experimentos são realizados em um ambiente simulado. A proximidade entre esses dois ambientes é essencial para que a solução desenvolvida tenha aplicação concreta. Na seção 3.2, observamos como realizar esta aproximação no que diz respeito a algumas características físicas do robô. Contudo, existem diversas outras dimensões nas quais a simulação pode diferir da realidade, e inevitavelmente algumas delas sofrem simplificações.

Figura 16 – Trajeto que deve ser realizado pelo robô: partir do ponto I e passar pelos outros três marcos em ordem crescente.




Identificar estas dimensões é essencial para interpretar os resultados obtidos e levantar pontos de ajustes para a simulação.

Algumas das simplificações dizem respeito ao terreno, e impactam primariamente os encoders. Por exemplo, o terreno real é irregular (gramado de um campo de futebol), enquanto o simulado é um plano perfeito. A irregularidade de um terreno afetaria as leituras dos encoders, de modo que caso um robô percorresse uma linha reta num terreno irregular, seus encoders acusariam rotações diferentes para cada roda, mesmo se tais encoders fossem perfeitos. Isto ocorre pois de fato as rodas não percorreriam o mesmo terreno, cada uma atravessaria seus próprios aclives e declives. Mitigamos esta simplificação adicionando ruído artificial a cada um dos encoders simulados, reproduzindo este mesmo comportamento.

Outras simplificações são feitas no contexto das características físicas do robô. As rodas do robô real possuem o mesmo diâmetro, porém desgastes e erros de medição poderiam resultar em uma assimetria do veículo. Similarmente, os eixos internos conectados às rodas certamente apresentam diferentes resistências, cuja medição é inviável. Ambas estas possibilidades poderiam fazer o robô desviar-se para um dos lados, mesmo quando comandado para executar um trajeto retilíneo num terreno plano. Este tipo de situação não é em si problemática, desde que fosse notada pelos sensores e corrigida pelo sistema de atuação. Portanto, devido ao misto de baixa gravidade e difícil modelagem, optamos pela simplificação.

No que diz respeito ao GPS, não é representada na simulação a possibilidade dele perder sinal durante a prova. Alguns dos experimentos realizados não incluem o GPS, porém não há casos em que o sensor é retirado enquanto uma prova está em andamento.

Uma dimensão importante que está modelada na simulação é a derrapagem. Como comentado na subseção 2.2.2, uma das principais fontes de ruído dos encoders rotativos é a derrapagem, que ocorre quando uma ou mais rodas escorregam e perdem contato com o chão, girando mais (ou menos) do que deveria. Dado que superfícies irregulares são particularmente suscetíveis à derrapagem, este é um problema real durante a prova. Nos

experimentos, é simulado o atrito responsável por fazer o carro derrapar em mudanças bruscas de velocidade. Por conta disso, foi necessário escolher conjuntos de aceleração e arrancada que suavizassem estas mudanças bruscas, tal qual ocorreria na realidade.

Talvez a dimensão mais importante que está modelada é referente ao ruído nos sensores. Inicialmente numa simulação, nenhum dos sensores possui ruído - todos são determinísticos, já que o estado real do sistema é conhecido (com exceção da câmera). Porém, como isto acarretaria em estimativas de pose praticamente perfeitas, mitigamos esta simplificação adicionando ruído artificial gaussiano em todos os sensores. Avaliamos experimentalmente os sensores reais para obter medidas adequadas de média e variância para os seus ruídos, nos permitindo assim representá-los com maior precisão na simulação. Esta ainda é uma simplificação pois não temos garantia de que o ruído real segue uma distribuição gaussiana; porém, na ausência de modelos mais precisos, esta foi a escolha que nos pareceu mais adequada.

## 4.4 DESCRIÇÃO DOS EXPERIMENTOS E MÉTRICAS

Todos os experimentos (I, II, III, IV e V) foram realizados 50 (cinquenta) vezes, e utilizaram um intervalo de confiança de 95% nas amostras obtidas. As métricas de avaliação adotadas no trabalho foram: (i) métricas relativas à acurácia da estimativa do filtro de Kalman; (ii) métricas relativas ao tempo de conclusão da prova e (iii) métricas relativas à completude da prova. As métricas usadas para a acurácia da estimativa do filtro são a média e a variância da distância euclidiana entre a posição estimada e a posição real. A métrica relativa ao tempo de conclusão da prova é o total de segundos entre a partida do robô do ponto inicial e a sinalização da passagem pelo terceiro cone. As métricas relativas à completude da prova são a quantidade de cones sinalizados corretamente pelo robô e a distância euclidiana entre uma sinalização e o cone correspondente. Uma ilustração das métricas (ii) e (iii) pode ser vista na Figura 17: no exemplo, o tempo total da prova é $50seg$, o número de marcos corretamente alcançados é 1, e a distância euclidiana média entre uma sinalização e o cone correspondente é $(5 + 10 + 0)/3 = 5m$.

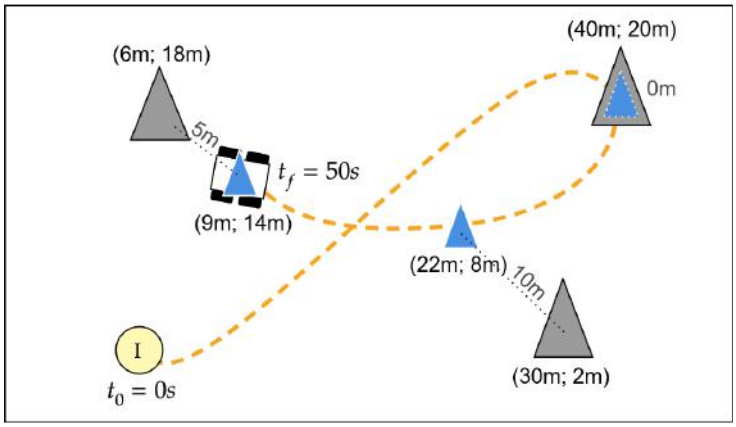
Os experimentos buscam responder as seguintes perguntas: (i) qual o impacto de cada sensor na estimativa do filtro; (ii) quais as melhores combinações de sensores; (iii) quais as melhores velocidades para o robô e (iv) qual modelo de atuação resulta no melhor desempenho na prova. Para responder a essas perguntas, cada experimento analisa combinações diferentes de sensores, velocidade e modelo de atuação.

Os experimentos podem ser divididos em dois grandes grupos. No grupo A (experimentos I, II e III), o robô estima sua posição com o filtro de Kalman, mas executa o percurso baseando-se na sua posição real. Neste grupo, são utilizadas métricas relativas à acurácia da estimativa do filtro. No grupo B (experimentos IV e V), a estimativa do filtro é utilizada para corrigir o percurso, e são utilizadas métricas relativas à comple-

tude e tempo de conclusão da prova. Esta distinção é feita pois embora o segundo grupo esteja mais próximo da situação real que se deseja solucionar, muitos experimentos não podem ser analisados adequadamente segundo as métricas do grupo B. Isto ocorre pois para algumas das combinações de sensores experimentadas, o robô não é capaz de atingir nenhum dos três cones, e portanto haveria uma perda de informações — afinal, sob as métricas de completude todas estas combinações seriam iguais. Portanto, tais métricas são aplicadas apenas sobre as alternativas que tenham capacidade de completar parte da prova; as alternativas restantes são testadas sob uma situação simplificada.

Outra razão para esta divisão de grupos diz respeito à odometria visual: quando nenhum dos marcos é alcançado, a câmera é incapaz de detectar cones e a alternativa se comporta como se a câmera não estivesse presente. Portanto, é interessante realizar o percurso com base na posição real para comparar o impacto deste sensor.

Figura 17 – Exemplo ilustrativo das métricas do grupo de experimentos B. Em cinza, a posição real dos marcos. Em azul, os pontos do trajeto em que o robô sinalizou que encontrou um marco. Imagem fora de escala.




A Tabela 1 resume os objetivos e métricas dos experimentos realizados.

Tabela 1 – Síntese dos experimentos realizados.

| Experimento | Grupo | Objetivo | Métricas |
|---|---|---|---|
| I | A | Analisar Sensores (Encoders e UMI) | Acurácia do Filtro |
| II | A | Analisar Sensores (GPS) | Acurácia do Filtro |
| III | A | Analisar Sensores (Odometria Visual) | Acurácia do Filtro |
| IV | B | Analisar Atuação Baseada em Estimativa | Tempo e Completude da Prova |
| V | B | Comparar Modelos de Atuação | Tempo e Completude da Prova |

## 4.5 ANÁLISE DE RESULTADOS

Nesta seção, detalhamos os resultados obtidos nos experimentos e é feita a análise dos respectivos resultados.

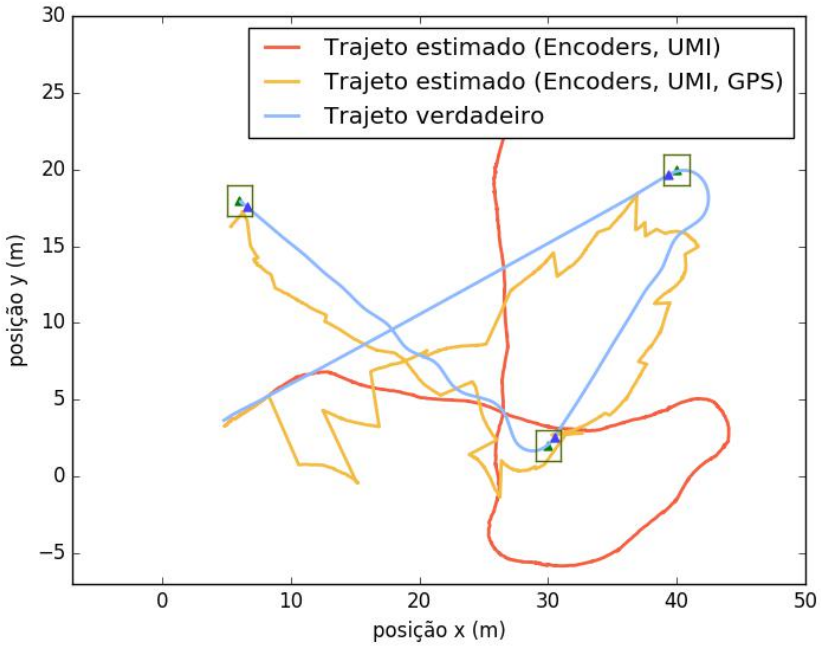
### 4.5.1 Experimento I: Análise dos Encoders e UMI

O objetivo do experimento I é observar como são as estimativas de posição feitas apenas com base nos dois sensores proprioceptivos (encoders e UMI), assim como o impacto da alteração da velocidade nestes sensores. Para isso, o carro percorre o percurso previamente mencionado com base na posição real. Foram testadas velocidades no intervalo $(1m/s; 5m/s)$. Os resultados estão apresentados na Tabela 2, e uma das simulações pode ser vista na Figura 18.

Tabela 2 – Resultados do experimento I (amostras independentes).

| Velocidade constante | I.C. da média da distância euclidiana entre a posição estimada e real (m) | I.C. da variância da distância euclidiana entre a posição estimada e real ($m^2$) |
|---|---|---|
| 1m/s | [7,549; 10,270] | [26,853; 45,424] |
| 2m/s | [8,620; 11,631] | [36,002; 60,795] |
| 3m/s | [11,146; 14,093] | [49,645; 80,728] |
| 4m/s | [10,969; 14,181] | [59,006; 101,036] |
| 5m/s | [12,011; 15,726] | [59,941; 113,779] |

Figura 18 – Uma das simulações executadas nos experimentos do grupo A.



Duas conclusões podem ser tiradas deste experimento. A primeira delas é a relação entre a velocidade e o desempenho da estimativa: um decréscimo na velocidade implica num decréscimo da média dos intervalos, isto é, quanto mais devagar o robô se movimenta, mais sua estimativa se aproxima do valor real. Este resultado é esperado, pois como discutido na seção 4.3, o efeito negativo mais incipiente sobre a leitura dos encoders é a derrapagem, que ocorre de modo mais acentuado em altas velocidades. Uma redução na velocidade reduz a derrapagem e melhora as estimativas dos encoders, ao passo que não afeta as da UMI. No fim das contas, a estimativa melhora.

A segunda conclusão corresponde à baixa qualidade de uma estimativa baseada apenas nos sensores proprioceptivos. Isto já havia sido aludido no capítulo 2, e agora se apresenta em termos numéricos. Na mais baixa das velocidades testadas ($1m/s$), a distância entre a posição estimada e a real ainda beira $10m$, com uma alta variância decorrente sem dúvida do fato de que esta distância aumenta conforme o tempo passa e mais incertezas se acumulam (como visto na Figura 18). Dado o objetivo de sinalizar na vizinhança de $1m$ do marco, o uso de outros sensores se mostra imprescindível.

### 4.5.2 Experimento II: Análise do GPS

O objetivo do experimento II é entender o impacto da inserção do GPS na fusão de sensores. Este experimento foi feito pareado com o experimento I, no que tange à velocidade do robô. Portanto, foram também testadas cinco velocidades diferentes no intervalo $(1m/s; 5m/s)$, e a Figura 18 apresenta uma das simulações que foram feitas. Os resultados deste experimento estão reunidos na Tabela 3.

Tabela 3 – Resultados do experimento II (amostras independentes).

| Velocidade constante | I.C. da média da distância euclidiana entre a posição estimada e real (m) | I.C. da variância da distância euclidiana entre a posição estimada e real ($m^2$) |
|---|---|---|
| 1m/s | [1,972; 2,266] | [0,840; 0,953] |
| 2m/s | [1,772; 2,873] | [0,830; 1,037] |
| 3m/s | [1,831; 2,189] | [0,894; 1,074] |
| 4m/s | [1,931; 2,966] | [1,001; 1,196] |
| 5m/s | [1,987; 2,963] | [1,208; 1,578] |

Comparando aos valores do experimento I, podemos observar que a inclusão do GPS resultou em significativa melhoria nas principais métricas. A média da distância euclidiana, que antes beirava os $10m$, reduziu para por volta de $2,5m$. A variância, por sua vez, decresceu para abaixo da média. Erros desta ordem estão muito mais próximos da distância desejada de $1m$, indicando que a adição do GPS é um passo na direção certa. Acreditamos que isto se deve majoritariamente à natureza exteroceptiva do sensor, que impede o acúmulo de incertezas, tal qual ocorre com os encoders e a UMI.

Outras conclusões podem ser obtidas a partir destes resultados. Ao contrário do experimento anterior, a redução da velocidade não provocou mudanças significativas na média da distância, e apenas leve alteração na variância. Pode-se interpretar tal resultado sob a linha de que as leituras do GPS não são afetadas por mudanças na velocidade, e portanto a redução apenas influenciaria os encoders. Dado que o GPS é o principal responsável pelo baixo valor das estimativas, a melhora dos encoders passa despercebida nos resultados finais, e nenhuma alteração significativa se torna aparente.

Uma terceira conclusão deste experimento diz respeito à natureza do trajeto estimado e não pode ser vista pelos dados da tabela. Ao analisarmos o percurso executado em uma das simulações (Figura 18), há uma evidente alteração nas estimativas feitas com o GPS, quando comparadas às estimativas feitas apenas com os encoders e a UMI. Os trajetos do GPS, embora mais próximos do trajeto verdadeiro, apresentam "saltos" que estão ausentes em sua contraparte. Este fenômeno ocorre devido à natureza exteroceptiva do GPS: como ele providencia posição absoluta ao filtro de Kalman, é inevitável que em algum momento ele forneça o sensoriamento de uma posição que diverge bastante da posição estimada anteriormente. Neste caso, o filtro incorporará esta medição em sua estimativa, e a posição estimada sofrerá um "salto". Isto está ausente nos outros sensores pois eles não medem informações absolutas, isto é, são proprioceptivos — saltos na medição da velocidade continuam provocando leituras contínuas de posição. A característica do GPS que o torna indispensável para a melhora das estimativas é a mesma que gera estes saltos.

Embora pareçam inofensivos, estes saltos introduzem um problema que deve ser analisado: como atuar baseado em estimativas que podem ter saltos significativos? Esta questão será retomada no Experimento IV.

### 4.5.3 Experimento III: Análise da Odometria Visual como Estimador de Pose Global

No Experimento III, busca-se analisar o impacto da odometria visual nas estimativas de pose do filtro de Kalman. Para isso, o processo de odometria visual é inserido a dois conjuntos de sensores diferentes: (i) um conjunto com encoders e UMI, e (ii) um conjunto com encoders, UMI e GPS. A velocidade é variada no intervalo $(1m/s; 5m/s)$ como nos experimentos anteriores. Os resultados estão apresentados na Tabela 4, e podem ser comparados com os resultados dos experimentos I e II.

Quando fundida apenas com os sensores proprioceptivos, a odometria visual oferece melhora perceptível na distância euclidiana média. A natureza dessa melhora pode ser visualizada na Figura 20(a): os dois trajetos estimados são idênticos até o instante em que a câmera detecta o cone; neste momento, a odometria visual corrige a posição absoluta do robô e a aproxima consideravelmente do trajeto verdadeiro. Assim que a detecção se encerra, o robô volta a se guiar apenas pelos sensores proprioceptivos, o que explica a relação entre menores velocidades e melhores estimativas (como visto no experimento I).

Tabela 4 – Resultados do experimento III (amostras independentes).

| Sensores | Velocidade | I.C. da média da distância euclidiana entre a posição estimada e real (m) | I.C. da variância da distância euclidiana entre a posição estimada e real ($m^2$) |
|---|---|---|---|
| Encoders, UMI, Odometria Visual | 1m/s | [2,976; 3,978] | [14,063; 24,132] |
| | 2m/s | [3,373; 4,505] | [13,495; 23,272] |
| | 3m/s | [3,623; 5,288] | [16,396; 39,702] |
| | 4m/s | [4,173; 5,218] | [19,127; 32,100] |
| | 5m/s | [4,676; 6,074] | [27,282; 47,193] |
| Encoders, UMI, GPS, Odometria Visual | 1m/s | [1,874; 2,200] | [1,003; 1,188] |
| | 2m/s | [1,983; 2,294] | [1,001; 1,226] |
| | 3m/s | [1,937; 2,259] | [1,064; 1,306] |
| | 4m/s | [1,998; 2,336] | [1,191; 1,465] |
| | 5m/s | [1,913; 2,119] | [1,216; 1,489] |

Embora a ocasionalidade desta correção possa fazê-la parecer pouco significativa, é visível nas figuras que esta correção causa um impacto duradouro — afinal, todas as estimativas estão mais próximas do trajeto verdadeiro, não apenas as produzidas nos arredores dos marcos.

Ainda assim, ao compararmos os resultados deste experimento com os do experimento II, se evidencia que esta correção dos sensores proprioceptivos provocada pela inclusão da odometria visual não supera a correção oferecida pelo GPS. Este resultado é natural, pois não importa quão duradoura seja a correção da odometria visual, ela não se compara à do GPS — que funciona similarmente, porém é contínua sobre todo o trajeto.

Relativo à fusão dos quatro sensores — encoders, UMI, GPS e odometria visual — podemos comparar os resultados deste experimento com os do experimento II para observar que quando o GPS também está presente, a inclusão da odometria visual não oferece ganho algum, independente da velocidade. Tal efeito indica que as estimativas da odometria visual não apresentam um misto de frequência e qualidade que justifique sua presença como estimador de pose global, frente ao GPS. Levando em consideração que este uso da odometria visual forma a base do modelo de atuação generalista, resta analisar se tal inadequação permanece presente nos testes do modelo especialista (experimento V).

Outras observações se fazem pertinentes. Na Figura 20, podemos notar uma das incertezas da odometria visual: a dependência da estimativa de orientação global. Analisando os trajetos estimados nas proximidades do terceiro marco, é perceptível que a estimativa que inclui a odometria visual ofereceu uma correção não coincidente com o trajeto verdadeiro, ao contrário do ocorrido nas proximidades do segundo marco. Além disso, a partir do instante de detecção do terceiro cone, os trajetos estimados são visivelmente paralelos entre si. Estas observações são explicadas pela dependência que a odometria visual possui em relação às estimativas de orientação global, como referenciado na seção 3.4.4: caso a orientação global esteja estimada incorretamente pelo filtro, a posição global

Figura 19 – Exemplos de detecção do cone pela cascata de Haar, como vistos pela câmera do robô. Os quadros são consecutivos.

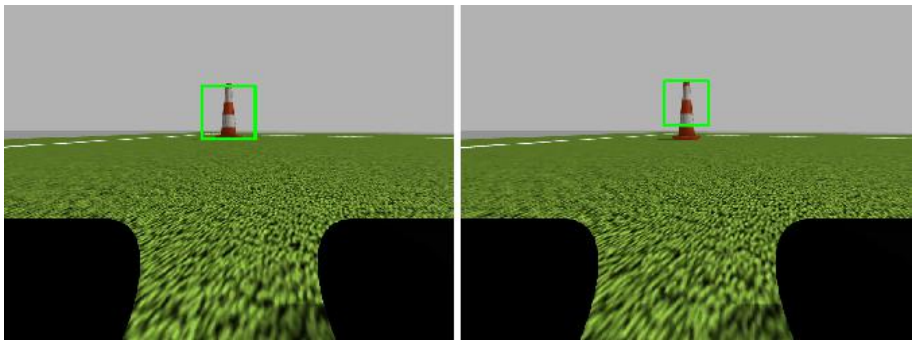
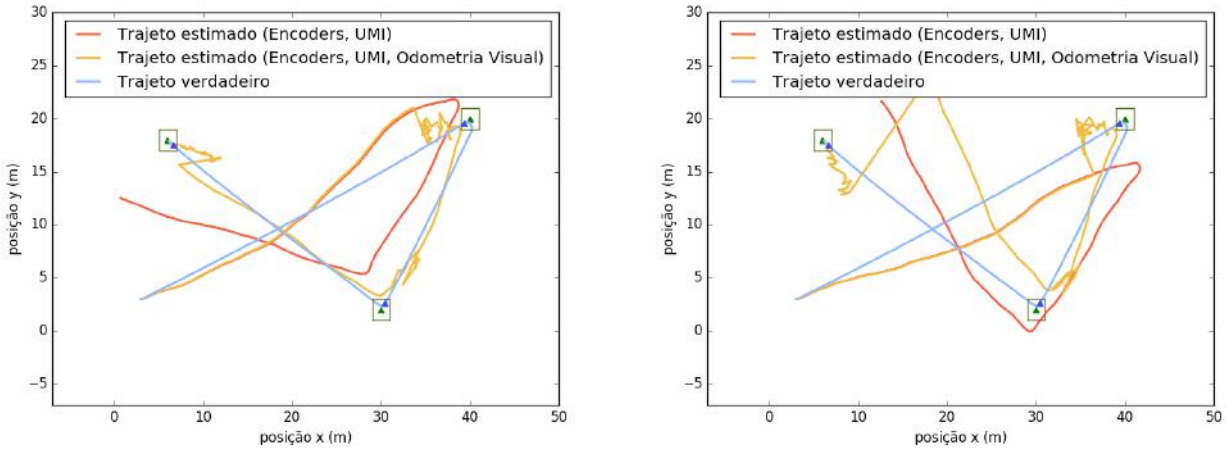


Figura 20 – Duas simulações executadas no Experimento III.




(a) (b)

também será estimada incorretamente pela odometria visual. Como os trajetos estimados na figura diferem apenas na presença da odometria visual, ambos possuem as mesmas estimativas de orientação global, e portanto ambos são paralelos a partir dos instantes de detecção dos cones. Esta incerteza significa que mesmo com uma detecção perfeita, o trajeto estimado ainda conterá erros. Este fenômeno pode ser visualizado com ainda mais clareza na Figura 20(b), entre os últimos dois marcos.

Ainda outra incerteza da odometria visual que pode ser percebida na figura diz respeito ao "ziguezague" do trajeto estimado nas proximidades dos marcos. Este efeito é provocado por variações na etapa de detecção de objetos, pois dependendo da distância entre o robô e o marco, seções distintas do cone podem ser detectadas pela cascata de Haar (Figura 19). Este é um exemplo de incerteza interna à odometria visual que a afasta do ideal. Para remediá-la, bastaria que as seções detectadas fossem consistentes de um quadro para o outro, não sendo necessário que o algoritmo sempre detectasse o cone por inteiro.

### 4.5.4 Experimento IV: Análise de Atuação Generalista Baseada em Estimativa

O objetivo do experimento IV é analisar como as melhores configurações do grupo A se comportam numa situação mais próxima da real, em que o robô deve se orientar de acordo com as estimativas do filtro de Kalman. Quando a odometria visual se faz presente, é empregado o modelo de atuação generalista. Conjuntos diferentes de combinações foram testados, e é importante destacar a motivação por trás destas escolhas. Para as configurações "Encoders, UMI" e "Encoders, UMI, Odometria Visual", foram utilizadas as melhores velocidades como indicadas pelos experimentos I e III, respectivamente. Para a configuração composta pelos sensores proprioceptivos e GPS, o experimento II não indicou diferença entre as velocidades, portanto utilizou-se tanto a velocidade mais baixa quanto a mais alta, para verificar se esta relação se manifestaria também neste novo experimento. O mesmo foi feito para a fusão dos quatro sensores. Os resultados estão reunidos na Tabela 5.

Tabela 5 – Resultados do experimento IV (amostras independentes).

| Sensores | Velocidade | I.C. da média da distância euclidiana entre uma sinalização e o marco correspondente (m) | I.C. da média de marcos sinalizados corretamente | I.C. da média do tempo total (seg) |
|---|---|---|---|---|
| Encoders, UMI | 1m/s | [8,508; 11,370] | [0,014; 0,189] | [92,355; 93,313] |
| Encoders, UMI, GPS | 1m/s | [1,696; 2,799] | [0,667; 1,210] | [104,432; 143,537] |
| | 5m/s | [1,875; 3,284] | [0,526; 0,953] | [44,48; 50,88] |
| Encoders, UMI, Odometria Visual | 1m/s | [10,542; 14,648] | [0,137; 0,515] | [82,975; 87,735] |
| Encoders, UMI, GPS, Odometria Visual | 1m/s | [1,743; 2,180] | [0,588; 1,125] | [99,218; 117,202] |
| | 5m/s | [2,058; 2,488] | [0,480; 0,906] | [33,240; 36,703] |

Em relação à fusão apenas dos sensores proprioceptivos (Encoders e UMI), podemos notar que esta teve o pior desempenho. Tal configuração raramente acertou um dos marcos, e a distância euclidiana média obtida é comparável à do experimento I — este resultado está de acordo com o observado nos experimentos anteriores, e também ao já discutido impacto do acúmulo de incertezas. As métricas de tempo estão de acordo com a resposta intuitiva: considerando que o trajeto tem por volta de 90m e o robô se move a 1m/s, realizar um percurso retilíneo significaria terminar a prova em cerca de 90 segundos,

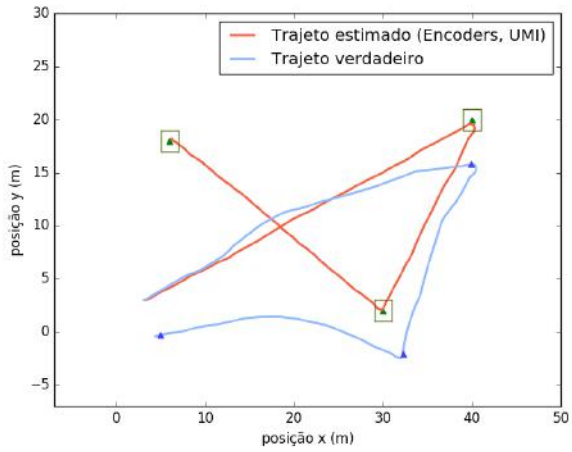
que foi o obtido.

Em relação à combinação dos sensores proprioceptivos com o GPS, podemos observar resultados próximos aos do experimento II: tanto há grande semelhança na distância euclidiana média, quanto na relação pouco significativa entre alterações na velocidade e melhoras na estimativa. Contudo, quanto às métricas de tempo, os resultados obtidos soam contraintuitivos — não só a velocidade de 1m/s difere levemente dos 90 segundos esperados, como também a velocidade de 5m/s não produz tempos de prova cinco vezes menores. Para explicar estes resultados, é preciso analisar os saltos da estimativa de posição aludidos no experimento II.

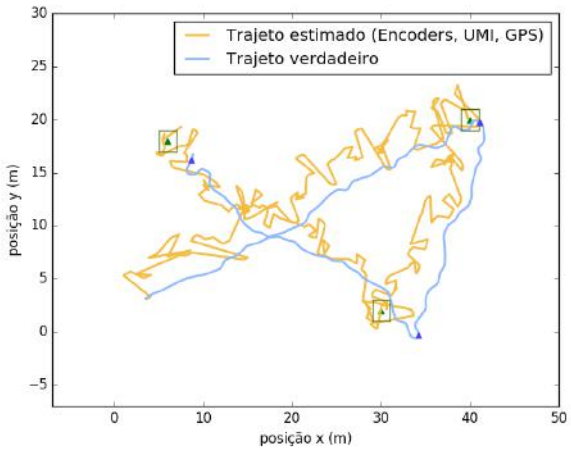
Quando o robô atua com base em uma estimativa de sensores proprioceptivos, toda leitura discrepante impacta apenas o acúmulo de incertezas a longo prazo, produzindo correções graduais e um trajeto suave (Figura 21(a)). Em contrapartida, quando a estimativa é suscetível a saltos, um sensoriamento de posição global discrepante se reflete em uma correção instantânea da orientação, causando uma trajetória ondulada. Isto está ilustrado na Figura 21(b), na qual o robô se move a 1m/s. A velocidade é relevante nesta discussão pois quanto mais alta a velocidade, maior é o trajeto que o robô percorre entre medições, e consequentemente maior é o percurso que deve ser corrigido a cada leitura discrepante. Isto provoca ondulações ainda mais evidentes, como visto na Figura 21(c), na qual o robô se move a 5m/s. As ondulações e sua relação com a velocidade explicam os resultados obtidos referentes ao tempo, dado que o robô não está se movendo em linha reta entre os marcos.

Em relação à fusão dos sensores proprioceptivos com a odometria visual, as métricas temporais estão dentro do esperado; contudo, o mesmo não pode ser dito sobre as outras duas métricas. Tanto foi obtida uma média de distância euclidiana significativamente mais elevada do que a observada no experimento III — se equiparando à combinação dos sensores proprioceptivos — quanto uma média de marcos sinalizados corretamente maior do que a vista em tal combinação. A razão deste resultado se deve a dois eventos. O primeiro evento ocorre quando as estimativas dos sensores proprioceptivos não são boas o suficiente para colocarem os cones no campo de visão do robô. Nestes casos, nenhum cone é detectado, a odometria visual nunca é ativada, e a trajetória se comporta como se apenas sensores proprioceptivos estivessem presentes (Figura 22(a)). O segundo evento ocorre quando as estimativas dos sensores proprioceptivos levam o robô a se desorientar gravemente quanto à sua posição global e provocam a violação do pressuposto de que "o cone sendo visto pela câmera corresponde ao próximo marco". Nestes casos, a odometria visual leva o robô a acreditar que está nas proximidades do marco errado, gerando vastos saltos na posição global e reforçando ainda mais sua desorientação (Figura 22(b)). Ambos os eventos fazem a trajetória estimada pelo robô ser igual ou pior do que a vista no grupo A, quando a odometria visual sempre atuava corretamente nas proximidades dos marcos. Todavia, apesar desses empecilhos, há raros casos em que a fusão proprioceptiva se apro-

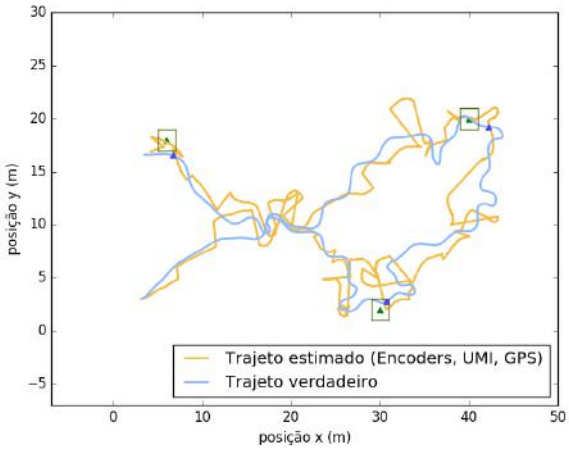
Figura 21 – Simulações executadas no Experimento IV com estimativas baseadas em encoders e UMI (a), e GPS (b, c).




(a)




(b)



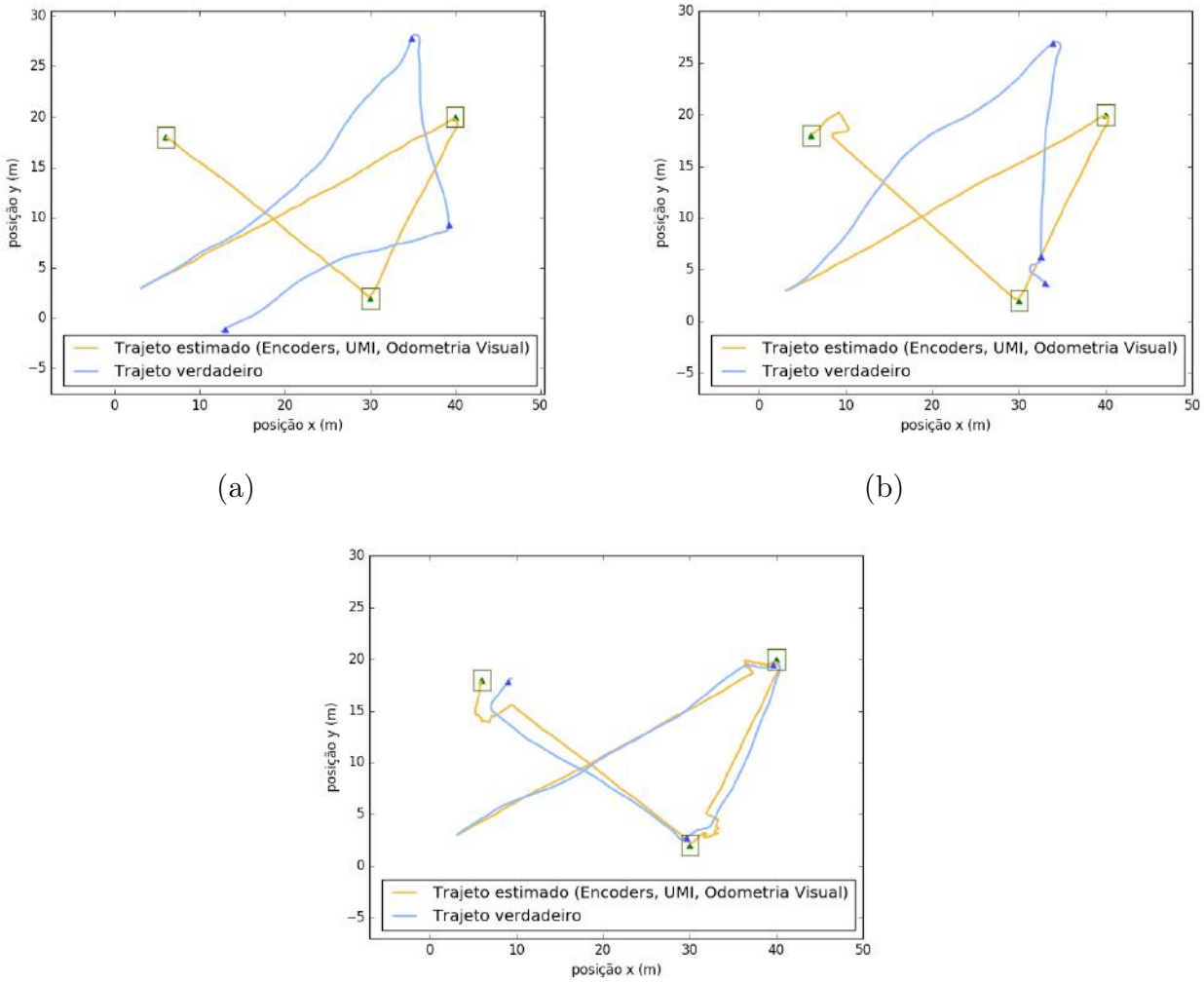

(c)

xima do trajeto verdadeiro — nestes casos, a odometria visual apresenta funcionamento ideal, auxiliando o robô a identificar corretamente quando está próximo de um marco, e permitindo sinalizações corretas em sucessão (Figura 22(c)). Isto explicaria a maior média de marcos sinalizados corretamente desta configuração, quando comparada à fusão apenas dos sensores proprioceptivos.

Relativo à fusão dos quatro sensores, podemos observar que os resultados são semelhantes aos observados na fusão dos sensores proprioceptivos com o GPS, isto é, a inclusão da odometria visual não acarretou em grandes mudanças. Este resultado também foi observado no experimento III.

Tendo em vista todas as configurações testadas, a conclusão mais provocante é que nenhuma das configurações oferece estimativas boas o suficiente para completar o percurso com qualquer grau de confiança. O critério de sucesso é quando os três marcos são

Figura 22 – Simulações executadas no Experimento IV com estimativas baseadas em encoders, UMI e odometria visual.




(a) (b)

(c)

sinalizados corretamente, e neste experimento, em média o robô atinge bem menos de dois. Apesar de algumas das configurações consistentemente produzirem estimativas com erro inferior a 2,5m, o robô usualmente sinaliza fora da área válida por uma distância mínima (Figuras 21(b) e 21(c)). Tal resultado demonstra a necessidade por sensores de menor incerteza e possivelmente a ineficiência do modelo de atuação generalista.

### 4.5.5 Experimento V: Análise do Modelo de Atuação Especialista

No experimento V, se objetiva observar o desempenho do modelo de atuação especialista, em comparação com os resultados obtidos pelo generalista nos experimentos anteriores. Tendo em vista que no experimento IV foi observada a baixa qualidade de uma atuação baseada na fusão dos sensores proprioceptivos com a odometria visual, neste experimento o GPS é sempre empregado. A velocidade foi variada, e as escolhas de variação

merecem ser discutidas com mais detalhes.

De início, foi testada a velocidade constante de 5m/s, contudo esta possibilidade se mostrou incompatível com a taxa de processamento de 10 quadros por segundo da odometria visual. O modelo especialista requer que o robô atue, quando nas proximidades dos cones, apenas com base na câmera; porém, o intervalo natural entre a fotografia da câmera e o processamento da imagem se mostrou problemático, pois quando a odometria visual detecta um cone, o robô já está em outra pose razoavelmente diferente, e os sinais de correção de orientação já não são mais adequados. Assim, o robô se mostrou incapaz de convergir no cone.

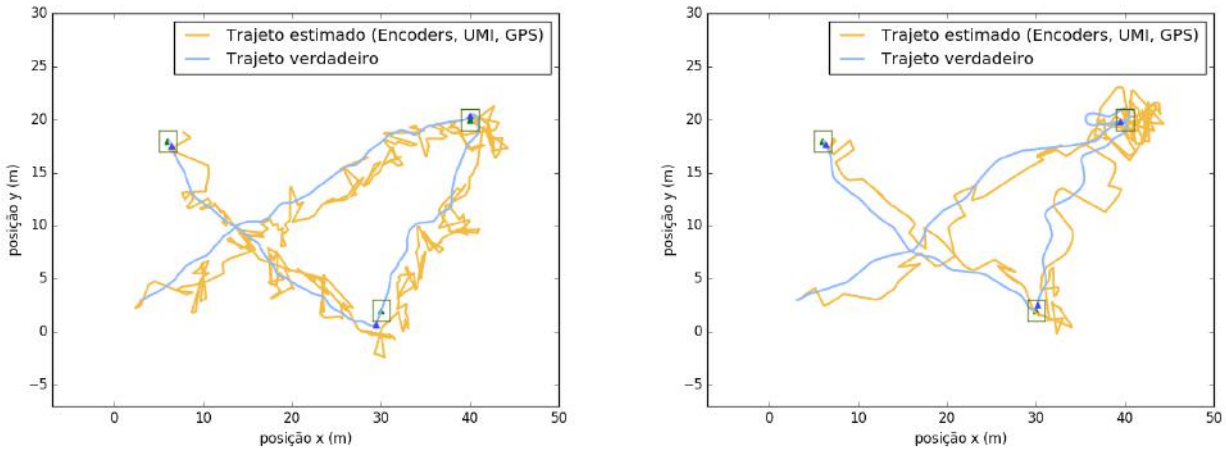
Em vista desta questão, foi empregada a estratégia de uma velocidade de deslocamento $v_d$ para o percurso entre marcos, e uma velocidade de aproximação $v_a$ para quando um cone fosse detectado. Tal $v_a$ teria de ser baixa pelos motivos delineados acima, porém este limitante não se aplicaria à $v_d$. Uma combinação adequada de $(v_d; v_a)$ que se apresentou para nós foi a de (5m/s; 1m/s), porém testes preliminares indicaram que ela não eliminava a não-convergência do robô, devido ao atraso de processamento no momento da transição de $v_d$ para $v_a$. Portanto, foi utilizada o par (3m/s; 1m/s), no qual esse problema não ocorre, e também o par (1m/s; 1m/s), para checar se a mudança de velocidade impacta as estimativas do robô. Os resultados estão reunidos na Tabela 6.

Tabela 6 – Resultados do experimento V (amostras independentes).

| Velocidade | I.C. da média da distância euclidiana entre uma sinalização e o marco correspondente (m) | I.C. da média de marcos sinalizados corretamente | I.C. da média do tempo total (seg) |
|---|---|---|---|
| $v_d$: 3m/s<br>$v_a$: 1m/s | [0,531; 0,703] | [2,739; 2,974] | [68,496; 178,748] |
| $v_d$: 1m/s<br>$v_a$: 1m/s | [0,397; 0,759] | [2,738; 3,016] | [196,225; 236,847] |

Em relação às métricas de conclusão da prova, é evidente a superioridade do modelo de atuação especialista. Se as abordagens dos experimentos anteriores raramente atingiam dois cones, no experimento V há raros casos em que os três marcos não são corretamente sinalizados. Ambas as velocidades apresentam este resultado. Isto se explica pois foi eliminada a possibilidade do robô se aproximar do marco e errá-lo por uma distância mínima: caso o robô esteja próximo do marco, então o cone está dentro do campo de visão da câmera, e consequentemente o robô é capaz de se guiar pela câmera e alcançar o cone. O modelo de atuação não performaria tão bem se as estimativas dos outros sensores fossem insuficientes para aproximar o robô dos marcos, porém isto não ocorre, como foi observado no experimento IV.

Figura 23 – Duas das simulações executadas no experimento V.




(a) (b)

Neste experimento, os raros casos em que não há sinalização correta dos três cones ocorrem quando a sinalização é feita um pouco mais tarde do que deveria (Figura 23(a)). Isto acontece quando, no momento em que a odometria visual identifica que está passando pelo cone, o filtro de Kalman estima que o robô está a mais de 4 metros de distância do marco: portanto, as condições de sinalização não são atingidas e a sinalização não é feita. Porém, após menos de 1,5 segundos, o filtro de Kalman estima que o robô está nesse raio de 4 metros do marco, e possivelmente ele pode não estar sobre o marco. O modelo de atuação especialista deve ser refinado para solucionar este caso, possivelmente alterando a distância (atualmente de 4 metros) e/ou o intervalo de tempo (atualmente de 1,5 segundos).

Há casos em que este mesmo cenário ocorre, porém a sinalização não ocorre fora de hora pois o intervalo de tempo se esgota (Figura 23(b)). Nestes casos, o robô simplesmente faz uma volta de modo a passar pelo cone novamente. Às vezes, isto ocorre mais de uma vez na mesma simulação, e é por conta de tais voltas que a variação nas métricas temporais excede o esperado, superando algumas das vistas no experimento IV. Porém, dado que a conclusão bem-sucedida da prova é mais prioritária do que oc critérios de tempo, a abordagem vista neste experimento ainda pode ser considerada como a de melhor desempenho.

# 5 CONCLUSÃO

Neste trabalho, elaboramos o sistema de controle de um robô móvel autônomo capaz de solucionar o problema denominado "Trekking". Neste problema, o robô deve partir de uma posição conhecida em um campo de futebol e realizar um percurso passando por três marcos pré-estabelecidos sinalizados por cones de trânsito. Para passar por um marco, o robô deve emitir um sinal quando estiver a menos de 1 metro do cone. Devido a restrições de hardware, o custo computacional deve ser limitado e apenas quatro sensores estão disponíveis (encoders rotativos, UMI, GPS e câmera). Na solução, empregamos um filtro de Kalman estendido capaz de fundir os sensores distintos para produzir estimativas acerca da pose global do robô. Estas estimativas foram integradas com dois modelos de atuação diferentes, cujos resultados foram comparados. Todos os experimentos foram realizados em ambiente simulado. O melhor desempenho alcançado pertence ao modelo de atuação especialista, fundindo encoders, UMI e GPS — neste modelo, a odometria visual é utilizada como único estimador quando o cone é detectado pela câmera do robô. Para este caso, o robô atingiu uma média de 2,918 cones alcançados corretamente, muito próximo do maior valor possível. Com o modelo de atuação generalista (em que a odometria visual é tratada como um estimador de pose global), a maior média de acertos foi de 1,285 cones.

Outros resultados menores podem ser destacados. Avaliamos que o filtro de Kalman estendido foi capaz de fornecer estimativas dentro de um erro de 2,5m, insuficiente para as condições da prova mas possivelmente adequado para outras situações. Além disso, o GPS demonstrou ser um sensor indispensável, capaz de levar o robô perto o suficiente do marco para o usufruto da odometria visual. Por sua vez, a cascata de Haar também demonstrou ser suficiente na tarefa de detecção, especialmente no modelo especialista onde a precisão da estimativa de distância não se mostra tão essencial.

Foi identificada também a insuficiência dos encoders e da UMI como únicos estimadores de pose, devido a sua natureza proprioceptiva resultar no acúmulo de muitas incertezas. A fusão destes sensores resultou no acerto de uma média de 0,101 cones.

## 5.1 TRABALHOS FUTUROS

Em perspectiva, podem ser investigadas outras formas de melhorar os resultados obtidos, como por exemplo, comparar o desempenho do filtro de Kalman estendido com o filtro de partículas. A literatura aponta que há diversas situações em que o filtro de partículas produz estimativas melhores que Kalman, e seu uso não foi empregado neste trabalho apenas devido ao seu custo computacional superior, que poderia superar os limites dos dispositivos disponíveis. O emprego deste filtro alternativo melhoraria o desempenho do modelo generalista, embora possivelmente não tanto o especialista. Analogamente, um

filtro de Kalman *unscented* poderia ser avaliado.

Outra linha de pesquisa envolveria a investigação do uso de técnicas de detecção de objetos alternativas à cascata de Haar. Embora tenha sido suficiente para o bom desempenho do modelo de atuação especialista, julgamos que há espaço para melhorias, devido ao impacto que pequenas variações na detecção provocam na estimativa de distância da odometria visual do modelo generalista. Técnicas de detecção mais robustas mitigariam esse efeito, embora tivessem de enfrentar o desafio do baixo poderio computacional.

Finalmente, está sendo pesquisado pelos autores a análise do desempenho do sistema de controle em um robô físico. Num ambiente real, sem dúvida surgiriam variáveis não-modelados na simulação, como assimetrias do veículo e dificuldades de controle do robô. Cabe avaliar se as soluções desenvolvidas permanecem adequadas perante estes novos desafios.

# REFERÊNCIAS


ARULAMPALAM, M. S. et al. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. **IEEE Transactions on signal processing**, Ieee, v. 50, n. 2, p. 174–188, 2002.

BACHMANN, E. R.; YUN, X.; PETERSON, C. W. An investigation of the effects of magnetic variations on inertial/magnetic orientation sensors. In: IEEE. **IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004**. [S.l.], 2004. v. 2, p. 1115–1122.

BARSHAN, B.; DURRANT-WHYTE, H. F. Inertial navigation systems for mobile robots. **IEEE Transactions on Robotics and Automation**, Institute of Electrical and Electronics Engineers, v. 11, n. 3, p. 328–342, 1995.

BORENSTEIN, J. et al. Mobile robot positioning: Sensors and techniques. **Journal of robotic systems**, Wiley Online Library, v. 14, n. 4, p. 231–249, 1997.

CABRITA, G.; MADHAVAN, R.; MARQUES, L. A framework for remote field robotics competitions. In: IEEE. **2015 IEEE International Conference on Autonomous Robot Systems and Competitions**. [S.l.], 2015. p. 192–197.

COX, L. Blanche: Position estimation for an autonomous robot vehicle. In: IEEE. **Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems'.(IROS'89)'The Autonomous Mobile Robots and Its Applications**. [S.l.], 1989. p. 432–439.

DHALL, A.; DAI, D.; GOOL, L. V. Real-time 3d traffic cone detection for autonomous driving. **arXiv preprint arXiv:1902.02394**, 2019.

DHALL, A.; DAI, D.; Van Gool, L. Real-time 3d traffic cone detection for autonomous driving. In: **IEEE Intelligent Vehicles Symposium (IV)**. [S.l.: s.n.], 2019.

DISSANAYAKE, M. G. et al. A solution to the simultaneous localization and map building (slam) problem. **IEEE Transactions on robotics and automation**, IEEE, v. 17, n. 3, p. 229–241, 2001.

DULIMART, H. S.; JAIN, A. K. Mobile robot localization in indoor environment. **Pattern Recognition**, Elsevier, v. 30, n. 1, p. 99–111, 1997.

FOX, D. et al. Bayesian filters for location estimation. **IEEE Pervasive Computing**, v. 3, p. 24–33, 2003.

FOX, D. et al. Particle filters for mobile robot localization. In: **Sequential Monte Carlo methods in practice**. [S.l.]: Springer, 2001. p. 401–428.

GANGANATH, N.; LEUNG, H. Mobile robot localization using odometry and kinect sensor. In: IEEE. **2012 IEEE International Conference on Emerging Signal Processing Applications**. [S.l.], 2012. p. 91–94.

GOOGLE LLC. **TensorFlow Lite**. 2017. Acessado em 19 de jul. de 2019. Disponível em: <https://www.tensorflow.org/lite>.

GREWAL, M. S.; ANDREWS, A. P. Applications of kalman filtering in aerospace 1960 to the present. **IEEE Control Systems Magazine**, v. 30, p. 69–78, 2010.

HALL, D. L.; LLINAS, J. An introduction to multisensor data fusion. **Proceedings of the IEEE**, IEEE, v. 85, n. 1, p. 6–23, 1997.

HARRIS, A.; CONRAD, J. M. Survey of popular robotics simulators, frameworks, and toolkits. In: IEEE. **2011 Proceedings of IEEE Southeastcon**. [S.l.], 2011. p. 243–249.

HARTLEY, R.; ZISSERMAN, A. **Multiple view geometry in computer vision**. [S.l.]: Cambridge university press, 2003.

HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Transactions of the ASME–Journal of Basic Engineering**, v. 82, n. Series D, p. 35–45, 1960.

KAM, M.; ZHU, X.; KALATA, P. Sensor fusion for mobile robot navigation. **Proceedings of the IEEE**, IEEE, v. 85, n. 1, p. 108–119, 1997.

KHATIB, E. I. A. et al. Multiple sensor fusion for mobile robot localization and navigation using the extended kalman filter. In: IEEE. **2015 10th International Symposium on Mechatronics and its Applications (ISMA)**. [S.l.], 2015. p. 1–5.

KIM, T.; LYOU, J. Indoor navigation of skid steering mobile robot using ceiling landmarks. In: IEEE. **2009 IEEE International Symposium on Industrial Electronics**. [S.l.], 2009. p. 1743–1748.

KOENIG, N.; HOWARD, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IEEE. **2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)**. [S.l.], 2004. v. 3, p. 2149–2154.

KOK, M.; HOL, J. D.; SCHÖN, T. B. Using inertial sensors for position and orientation estimation. **Foundations and Trends in Signal Processing**, Now Publishers Inc., v. 11, n. 1-2, p. 1–153, 2017.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 740–755.

LIU, W. et al. Ssd: Single shot multibox detector. In: SPRINGER. **European conference on computer vision**. [S.l.], 2016. p. 21–37.

LUO, R. C.; YIH, C.-C.; SU, K. L. Multisensor fusion and integration: approaches, applications, and future research directions. **IEEE Sensors journal**, IEEE, v. 2, n. 2, p. 107–119, 2002.

NAKAMURA, E. F.; LOUREIRO, A. A.; FRERY, A. C. Information fusion for wireless sensor networks: Methods, models, and classifications. **ACM Computing Surveys (CSUR)**, ACM, v. 39, n. 3, p. 9, 2007.

NEGENBORN, R. Robot localization and kalman filters on finding your position in a noisy world. **ser. Utrecht University, Utrecht, Netherlands**, 2003.

OPEN SOURCE ROBOTICS FOUNDATION. **ROS Kinetic Kame**. 2016. Acessado em 19 de jul. de 2019. Disponível em: <http://wiki.ros.org/kinetic>.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE, JAPAN. **ICRA workshop on open source software**. [S.l.], 2009. v. 3, n. 3.2, p. 5.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 779–788.

ROBOCORE. **Regras - Robô Trekking**. 2018. Acessado em 19 de jul. de 2019. Disponível em: <https://www.robocore.net/upload/attachments/robocore__regras_robo_trekking_118.pdf>.

SäRKKä, S. **Bayesian Filtering and Smoothing**. [S.l.]: Cambridge University Press, 2013.

SIEGWART, R. et al. **Introduction to autonomous mobile robots**. [S.l.]: MIT press, 2011.

STARANOWICZ, A.; MARIOTTINI, G. L. A survey and comparison of commercial and open-source robotic simulator software. In: ACM. **Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments**. [S.l.], 2011. p. 56.

SUMANARATHNA, D. et al. Simulation of mobile robot navigation with sensor fusion on an uneven path. In: IEEE. **2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]**. [S.l.], 2014. p. 388–393.

THRUN, S. Particle filters in robotics. In: MORGAN KAUFMANN PUBLISHERS INC. **Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence**. [S.l.], 2002. p. 511–518.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic robotics**. [S.l.]: MIT press, 2005.

THRUN, S. et al. Robotic mapping: A survey. **Exploring artificial intelligence in the new millennium**, v. 1, n. 1-35, p. 1, 2002.

VIOLA, P.; JONES, M. et al. Rapid object detection using a boosted cascade of simple features. **CVPR (1)**, v. 1, p. 511–518, 2001.

WOLF, D. F. et al. Robótica móvel inteligente: Da simulação às aplicações no mundo real. In: SN. **Mini-Curso: Jornada de Atualização em Informática (JAI), Congresso da SBC**. [S.l.], 2009. p. 13.

WU, A. D.; JOHNSON, E. N.; PROCTOR, A. A. Vision-aided inertial navigation for flight control. **Journal of Aerospace Computing, Information, and Communication**, v. 2, n. 9, p. 348–360, 2005.

ŽLAJPAH, L. Simulation in robotics. **Mathematics and Computers in Simulation**, Elsevier, v. 79, n. 4, p. 879–897, 2008.