Veículo Guiado Automatizado (AGV) - Warehouse

Gabriel Guimarães A.Castro
Matrícula: 15/0126425
gabriel11997.castro@gmail.com

Stephanie Cardoso Guimarães
Matrícula:15/0021950
stephanieguimaraes7@gmail.com

Abstract

Esse projeto desenvolve uma aplicação de um sistema de tempo real implementado através de um software de simulação, com o propósito de aprofundar e complementar os conhecimentos adquiridos durante o curso. A aplicação elaborada é um Automated Guided Vehicle AGV (Veículo Guiado Automatizado), esse robô portátil e automatizado possui a função de transportar pacotes, auxiliando nos processos de organização e gerenciamento de armazenagem.

1. Introdução

Um AGV (Automated Guided Vehicle) ou veículo guiado automaticamente para Warehouse (figura 1) típicamente nessecita de um modelo de guia para se locomover, possui a habilidade "Pick and Place" de transporte automatizado de material, não necessita da interação humana para executar suas tarefas, é melhor empregado em tarefas repetitivas, o uso das unidades dos AGV's pode ser facilmente escalável e entre outros.

Figure 1. Exemplo AGV para Warehouse da Amazon.

1.1. História e Mercado

Atualmente, a maior área de aplicação da robótica é a robótica industrial. A história dos primeiros robôs móveis que se tem notícia foram criados em 1968 no MIT (Massachusetts Institute of Technology), onde podiam mover-se em determinado ambiente de forma remota, e em 1966 e 1972 foi desenvolvido o que hoje é considerado o primeiro rôbo autônomo móvel do mundo "Shakey" no Stanford Research Institute (SRI).

Com a evolução da robótica, as constantes transformações e demandas do meio industrial e dos modelos de manufatura, o crescimento do mercado online (E-commerce) e as mudanças na forma de consumo, a robótica desde de suas primeiras aplicações vêm crescendo suas raízes nos mais diversos campos de produção e consumo. Assim, a utilização dos AGV's de transporte em Warehouse em ambientes industriais ou como solução inteligente em armazéns, têm ganhado cada vez mais espaço na econômia e mercado industrial, figura 2.

Figure 2. AGV para Warehouse em uso.

1.2. Conceitos Básicos de Tempo Real

Uma aplicação de tempo real esta submetida a requisitos de natureza temporal, esses requisitos temporais são definidos e demandados pelo meio externo ou ambiente ao qual pertence e podem acontecer de forma simultânea e variada. A aplicação de tempo real deve por tanto, ser proje-

tada e desenvolvida com o objetivo de se cumprir com esses requisitos temporais de forma segura, ”fazer o trabalho no tempo disponível e não no tempo necessário”.

O estudo do tempo real também define uma característica denominada ”criticalidade”, classificando o nível da dependência temporal do sistema, como ”Hard Real Time” cuja incapacidade do sistema em cumprir com os requisitos temporais exigidos pode causar falhas catastróficas e o nível de criticalidade ”Soft Real Time”, em que os requisitos temporais define simplesmente o comportamento desejado.

1.2.1 Tarefas

As tarefas podem ser definidas como um segmento de código com funções específicas, possui seu próprio requisito temporal a ser cumprido, é executada de forma repetitiva com determinado período e com um intervalo de tempo rígido (deadline).

1.2.2 Deadlines

O deadline (literalmente, fim da linha) é o instante ou intervalo de tempo máximo que uma tarefa possui disponível para sua execução, o principal requisito de tempo possível e pode ou não ser definido em relação ao início da tarefa (relativo ou absoluto).

Os tipos de deadline também podem ser classificados em níveis como, ”Deadline Hard” é o nível com consequências mais graves causados pelo seu descumprimento (figura 3), o ”Deadline Firm” não possui consequências graves e caso a tarefa não seja executada dentro do limite desse deadline ela pode ser descartada, o ”Deadline Soft” também não possui consequências graves mas a tarefa deve ser executada mesmo que o deadline tenha sido descumprido (figura 4).

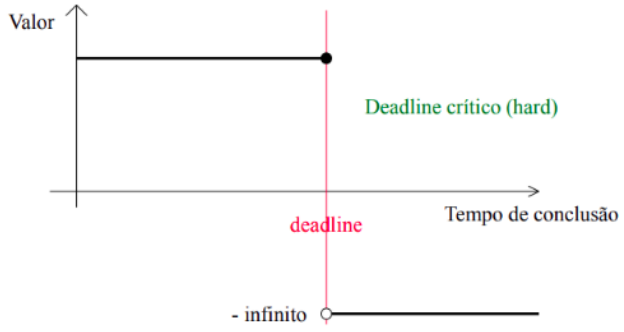

Figure 3. Representação gráfica do Deadline Hard.

1.2.3 WCET - Tempo de Execução no pior Caso

O tempo de execução de uma tarefa equivale ao tempo em que a tarefa precisa de processador para ser executada, e

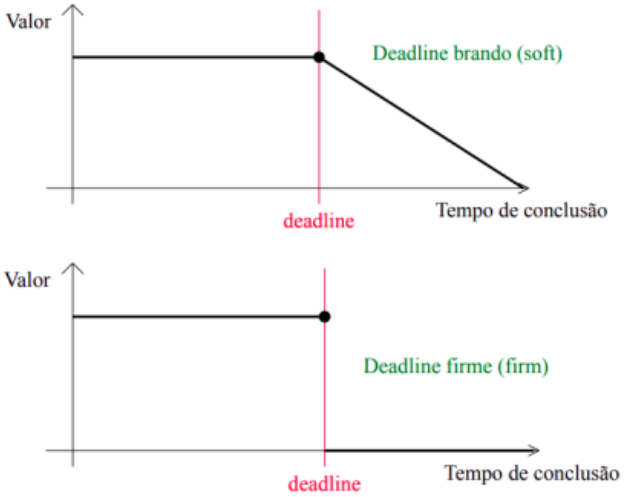

Figure 4. Representação gráfica do Deadline Hard.

diversos fatores (com origem de software ou hardware) podem causar variações nessa variável. Existem diversas abordagens para se realizar uma estimativa do WCET, sendo sua precisão dependente da técnica de análise usada, e é preciso supor os piores casos possíveis do comportamento de todo o sistema.

Idependente se o sistema de tempo real é crítico ou não as informações a respeito do WCET são relevantes. O método mais usado na prática para análise do WCET é a medição do tempo de execução de uma tarefa isolada, considerando diversas situações possíveis, obtendo-se dessa forma o ”máximo tempo de execução observado (High Water Mark - HWM)”, que normalmente possui um valor menor que o valor verdadeiro do WCET (figura **??**).

1.2.4 WCRT - Tempo de Resposta no Pior Caso

O tempo de resposta pode ser interpretado como o intervalo de tempo em que uma entrada é modificada até que o momento em o sistema reage a essas modificações ou simplesmente como o inicio e conclusão de uma tarefa, seu valor depende da forma como a tarefa é implementada e pode variar por diversas causas.

Para se realizar uma estimação estatística do WCRT é preciso levar em consideração as características, organização e a forma de implementação das tarefas do sistema. O WCRT também pode ser estimado através da realização de diversas medições e análise estatística dos resultados desses testes, obtendo-se um valor observado ou HWM para o pior tempo de resposta.

2. Materiais e Métodos

Esse projeto foi inteiramente implementado respeitando as medidas de distânciamento social, por tanto, para facil-

itar o trabalho em grupo considerando essas circunstâncias foram adotadas diversas ferramentas para logistica de trabalho online. Os principais aplicativos usados para gestão do projeto foram, o GitHub para organização, compartilhamento e atualização do código feito por cada membro, Notion para oraganização de tarefas e o drawio.

2.1. Simulador

Esse projeto foi implementado através do simulador de robô CoppeliaSim (V-REP) com ambiente de desenvolvimento integrado e com uma arquitetura de controle distribuída, permitindo o uso de uma API remota e linguagem python (usada nesse projeto). O coppeliaSim possui diversos modelos, objetos e cenas prontos, e além de permitir a modificação desses modelos já prontos de acordo com a necessidade do usuário, é também possível criar diversos tipos de modelos.

Os principais objetos usados para a realização da simulação foram:

- A figura 5 mostra a cena completa usada para a simulação, criada para representar um pequeno galpão. Essa cena possui, uma câmera localizada no ”teto” permitindo a vista de todo o galpão, paredes para limitação da área, blocos para ilustração visual das prateleiras, o AGV e um alvo (quadrado verde).

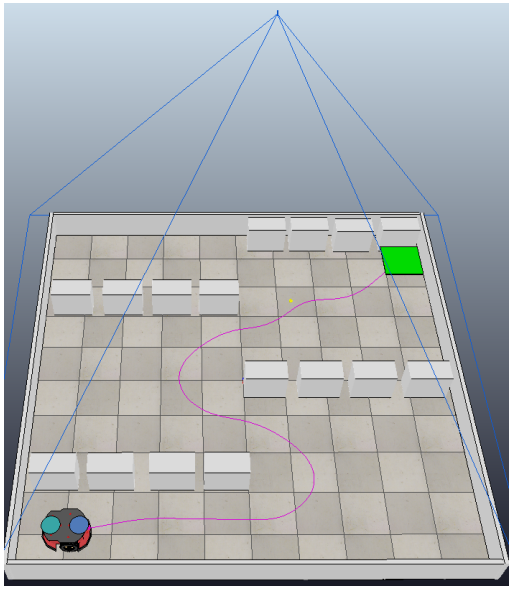

Figure 5. Cena completa usada para simulação.

- Um robô móvel ”pioneer p3dx.ttm” com diversas modificações personalizadas para atender a essa aplicação (figura 6), alguns de seus componentes são, os dezesseis sensores ultrassônicos e os discos localizados na vista superior para localização da posição pela câmera e cálculo da orientação do robô.

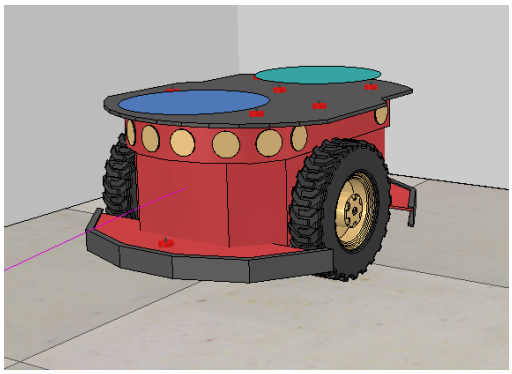

Figure 6. Modelo do AGV usado na simulação.

2.2. Aplicação de Tempo Real

O tipo de aplicação implementada nesse projeto busca através da tecnologia melhorar o desempenho de sistemas logísticos e oganizacionais de armazens, para estoque e entrega de pacotes e a função do AGV é transportar esses pacotes de um ponto a outro dentro do galpão em segurança. Esse tipo de serviço demanda que o AGV execute uma quantidade de transportes de pacotes dentro de um determinado intervalo de tempo, para que os pacotes possam ser entregues ao cliente dentro do prazo previsto e para ajudar a manter o nível de estoque sincronizado e organizado de acordo com a demanda.

Esse projeto foi desenvolvido da forma mais simplificada possível com apenas um AGV em funcionamento, mas esse tipo de solução pode ser e é geralmente usada em larga escala para empresas com grande demanda de estoque e entrega de produtos, ”Amazon já tem mais de 100 mil robôs autômatos em seus galpões” [3]. Assim, esse projeto busca tornar possível que o AGV possa completar sua função de transporte de pacote dentro de um intervalo de tempo esperado (deadline).

Para se realizar a implementação do projeto, os testes e as análises temporais, a aplicação foi dividida em tarefas (listadas abaixo), figura 7.

1. Determinação do caminho traçado do AGV até o alvo.

2. Percorrer do ponto A (localização do AGV) ao B (localização alvo).

3. Verificar obstáculos com os sensores ultrassônicos.

Devido as características das tarefas definidas, a tarefa 3 só pode ocorrer durante a execução da tarefa 2 de forma concorrente, porque só é possível identificar um obstáculo

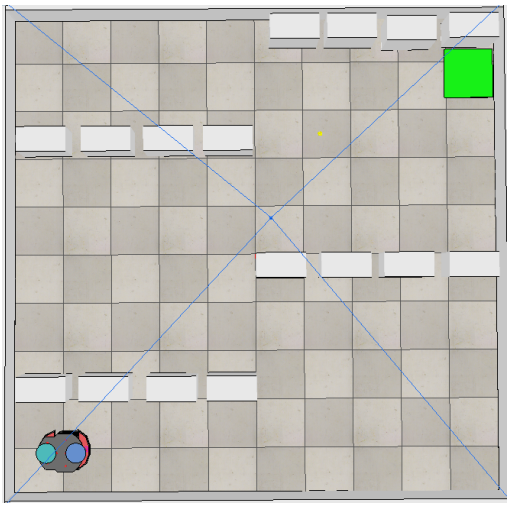

Figure 7. Vista superior da cena.

quando o robô inicia a ação de se locomover. Por tanto, durante todo o processo de execução da tarefa 2 é preciso que a tarefa 3 fique verificando se exite um obstáculo a frente parando o AGV caso seja encontrado.

Para realização dos testes e na tentativa de simular uma situação real de gerenciamento de armazem, o deadline para essa aplicação foi estipulado de acordo com a capacidade do próprio AGV em executar sua função de transportar uma determinada quantidade de pacotes durante um determinado período de tempo (média do tempo obtido nas simulações, exposto nos resultados), considerando o tempo que ele leva para chegar no ponto mais distante do armazem, ou seja, uma demanda que o AGV é normalmente capaz de atender mesmo considerando as eventuais interferências ”aleatórias” (obstáculo surpresa que impede o carrinho de andar durante um determinado intervalo de tempo). Assim, quando ocorre muitas interferências (obstáculos que param o carrinho) o AGV tenta compensar aumentando sua velocidade na tentativa de conseguir cumprir com esse deadline estipulado.

2.3. Programação

A linguagem utilizada para o desenvolvimento desse projeto foi o python e alguns poucos scripts na linguagem lua. Para uma descrição geral do código é realizado primeiramente o ”path tracking”, onde o robot aponta para uma posição a sua frente buscando identificar o caminho calculado e traçado e o robô recebe o controle das velocidades linear e rotacional para alcançar o ponto a frente identificado no trajeto.

2.3.1 Diagramas

Para facilitar a compreensão do projeto os diagramas podem ser divididos em estágios. A figura 8 mostra uma representação em diagrama do algoritmo desenvolvido para executar o processo de planejamento e suavização do caminho, usando como referência as imagens oferecidas pela câmera para a localização dos objetos de interesse.

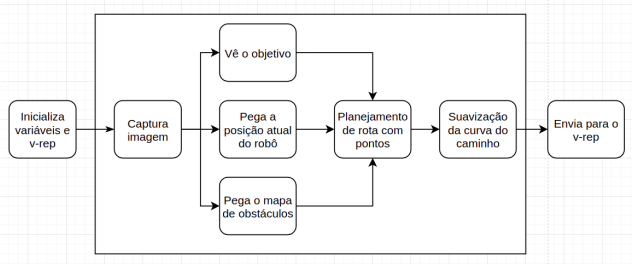

Figure 8. Diagrama do planejamento e suavização do caminho.

Na figura 9, o diagrama representa a lógica do algoritmo que faz o AGV de fato seguir o caminho trilhado para cada iteração.

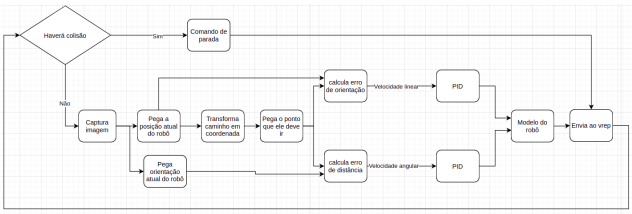

Figure 9. Diagrama da ação de percorrer o caminho pelo AGV.

O terceiro diagrama figura 10, mostra a representação lógica do algoritmo de funcionamento do sensor ultrassônico para a identificação e ações ao se encontrar obstáculos.

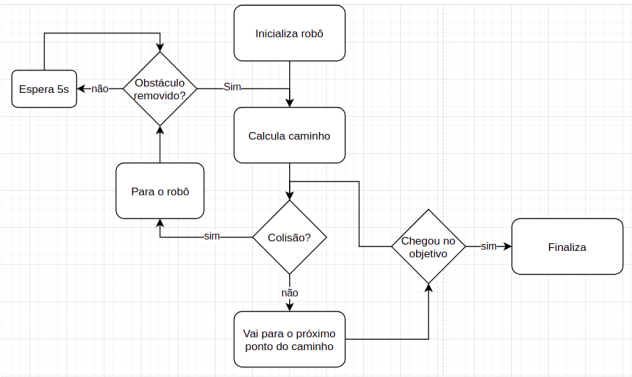

Figure 10. Diagrama da lógica do AGV ao encontrar um obstáculo.

3. Resultados

Após o desenvolvimento de todo o projeto foram efetuadas diversas simulações para a obtenção dos dados de interesse de forma automática (adicionada no código), o AGV faz um loop percorrendo o mesmo percurso medindo o tempo total de cada ciclo da simulação (tempo que leva para calcular a rota mais o a ação do AGV de percorrer o trajeto), salvando esse valores em uma array para a execução das análises, confecção dos gráficos e da tabela.

Na tabela mostrada na figura 11 foram obtidos, o tempo total de cada rodada ou ciclo percorrido (tempo que o AGV leva para calcular a rota e chegar no alvo), os tempos ”aleatórios” de parada que são gerados para simular as situações de obstáculo encontrado e finalmente obtendo os valores do Kp, sendo os valores do Kp responsável por aumentar a velocidade do AGV quando necessário.

	Tempo total (s)	Tempo parado (s)	Kp
	39,750	3	0,5
	39,550	3	0,5
	39,700	3	0,5
	42,850	4	0,5
	43,200	2	0,5
	46,750	5	0,6
	39,750	2	0,7
	41,950	1	0,8
	58,350	5	0,9
	49,100	4	0,9
	39,750	2	0,9
	39,600	4	0,9
	45,050	5	0,9
	39,900	5	0,9
	43,050	3	0,9
	41,600	2	0,9
	39,950	1	0,9
	40,850	3	0,9
	42,900	1	0,9
	39,850	4	0,9
	40,750	1	0,9
	43,950	1	0,9
	40,800	0	0,9
	39,750	1	0,9
	44,950	4	0,9
	44,000	4	0,9
	44,000	4	0,9
	46,000	5	0,9
	41,500	1	0,9
	39,900	2	0,9
Total	1279,050	85	

Figure 11. Tabela com os dados obtidos.

A figura 12 mostra os valores do tempo que o AGV ficou parado esperando o obstáculo ser removido, para depois de ser removido poder recalcular a rota e voltar a andar.

A figura 13 mostra o histograma com os valores obtidos do tempo total decorrido durante cada ciclo na simulação.

3.1. Requisitos

Os primeiro requisito para o projeto é que o robô não colida em nada durante o seu trajeto, por isso, a cada iteração dentro do laço principal que roda muito frequentemente, o robô verifica se precisa parar. A distância segura para o robô foi estabelecida a partir de uma margem de segurança de 30% da distância que ele precisava para parar em sua velocidade máxima. Como resultado, arredondando para um valor maior, o robô para em uma distância frontal de 30cm.

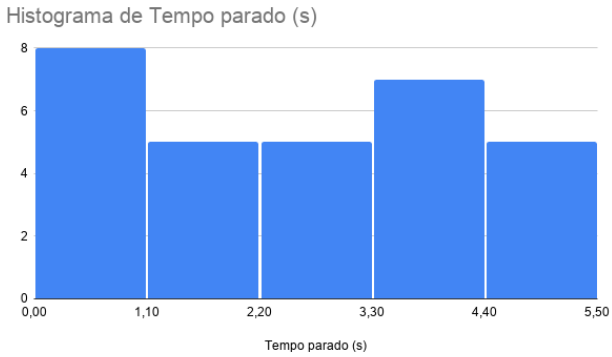

Figure 12. Histograma de tempo parado.

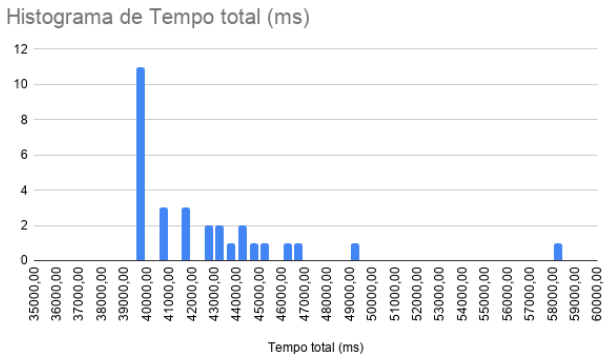

Figure 13. Histograma do tempo total.

O segundo requisito é cumprir um deadline de levar em média 45s entre os dois pontos mais distantes da *warehouse* simulada em 30 rodadas. Para que haja uma certa margem de segurança, o robô tentará ter uma valor limiar em código entre 40s e 41s ao invés de 45s. Sendo assim, teremos um tempo total para as 30 rodadas, de 1350s ou 22m e 30s.

Ele verifica a cada vez que faz o percurso o tempo gasto para percorrer e se for menor 40 ele compensa aumentando o ganho proporcional do PID que controla a sua velocidade linear. Caso for maior que 41 ele diminui a velocidade da mesma forma para a próxima rodada. Na simulação h

4. Discussão e Conclusões

Para o desenvolvimento da aplicação proposta nesse projeto foram necessários diversos conhecimentos adquiridos ao longo do curso, principalmente robótica, computação e os conhecimentos de tempo real adquiridos na disciplina, e devido as circunstâncias atuais de isolamento social o projeto teve que ser implementado em um simulador. Apesar de inicialmente o simulador escolhido ser o Gazebo, mas essa opção teve que ser excluida devido a incapacidade dos computadores usados de suportar o software, sendo

este substituído posteriormente pelo CoppeliaSim mais adequado para as ferramentas disponíveis.

Após a definição das ferramentas de simulaçao e programação mais adequadas, o principal desafio era conciliar todos os problemas e desafios de origem robótica inerentes à aplicação escolhida, com as características de um sistema de tempo real e com toda a implementação feita de forma virtual.

Apesar de alguns percalços encontrados o resultado final do projeto se mostrou satisfatório, possibilitando grandes aprendizados e experiência prática e teórica para a equipe, no campo da disciplina e interdisciplinar, incluindo a experiência com o uso de ferramentas para gestão de projetos.

5. Bibliografia

[1] MURPHY, Andrew. Título: AGV Deep Dive: How Amazon's 2012 Acquisition Sparked a $10B Market. LOUPVENTURES, 2017. Disponível em: (Link). Acesso em: (17/05/2021).

[2] ARTI. Título: Uma Breve História Da Robótica. MoreThanDigital, 2021. Disponível em: (Link). Acesso em: (17/05/2021).

[3] MULLER, Léo.Título: Amazon já tem mais de 100 mil robôs autônomos em seus galpões. tecmundo, 2018. Disponível em: (Link). Acesso em: (17/05/2021).

[4] Creators of V-REP. Título: CoppeliaSim User Manual. CoppeliaSim, version 4.2.0. Disponível em (Link). Acesso em: (18/05/2021).

[5] FAKHARANY, Ahmed. Título: Path tracking tutorial for pioneer robot in vrep. Medium, 2018. Disponível em: (Link). Acesso em: (18/05/2021).

[6] OLIVEIRA, Rômulo. Título: Fundamentos dos Sistemas de Tempo Real. Edição do Autor, 1ª edição, Florianópolis-SC, 2018.