
Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

# Estudo e implementação de técnicas para identificação e controle de trajetória para um AGV usando Arduino

João Roberto Cavalcanti de Araújo

Campina Grande, 2017

João Roberto Cavalcanti de Araújo

# Estudo e implementação de técnicas para identificação e controle de trajetória para um AGV usando Arduino

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG
Centro de Engenharia Elétrica e Informática - CEEI
Departamento de Engenharia Elétrica - DEE

Orientador:
**Péricles Rezende Barros, PH.D**

Campina Grande, de 2017

João Roberto Cavalcanti de Araújo

# Estudo e implementação de técnicas para identificação e controle de trajetória para um AGV usando Arduino

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Banca examinadora.

---

**Prof. Dr. Péricles Rezende Barros, PH.D**
Orientador

---

**Professor**

Prof. Dr. Rafael Bezerra Correia Lima

Campina Grande, 8 de setembro de 2017

# Resumo

Neste trabalho são relatados os procedimentos realizados no processo de implementação de um sistema de identificação de faixa para controle de trajetória de um AGV deslizante. Foram utilizados fotodiodos como sensores para a obtenção de dados para identificação da trajetória. Cinco algoritmos de estimação de posição da faixa foram testados e analisados, sendo dentre eles quatro algoritmos geométricos e um baseado em Redes Neurais. Uma plataforma simuladora de faixa foi usada para adquirir uma base de dados para o teste dos algoritmos e o treinamento da rede neural. Foi feita a modelagem cinemática e mecânica do AGV e a partir dela foi construído um simulador em Matlab. Por fim foi feita a modelagem do erro de trajetória para ser usada futuramente na implementação do controle de trajetória.

**Palavras-chave:** AGV, Identificação de trajetória, Controle de trajetória.

# Abstract

<In this work is reported the procedures performed in the process of implementing a track identification system to control the trajectory of a sliding AGV. Photodiodes were used as sensors to obtain data to identify the trajectory. Five strip position estimation algorithms were tested and analyzed, among them four geometric algorithms and one based on Neural Networks. A strip simulator platform is used to acquire a database for algorithm testing and neural network training. A kinematic and mechanical model of the AGV was made and from it a creation of simulator in Matlab. Finally, a trajectory error modeling was made in order to be to be used in the implementation of the trajectory control in the future.>

**Key-words**: AGV, Trajectory identification, Trajectory control.

# Lista de figuras

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| AGV | *Auto Guided Vehicle* (Veículo Auto Guiado) |
| NN | *Neural Networks* (Redes Neurais) |
| ICR | *Instant Center of Rotation* (Centro Instantâneo de Rotação) |

# Lista de símbolos

| | |
|---|---|
| $\phi(v)$ | Função de ativação |
| $w_i$ | Peso sináptico |
| $E(n)$ | Função de custo |
| $\beta$ | Taxa de aprendizado |
| $\varepsilon$ | Tolerância para erro de treinamento |
| $\Gamma$ | Erro translacional de trajetória |
| $\theta$ | Erro rotacional de trajetória |
| $q$ | Vetor de posição inercial do AGV |
| $v_{cx}$ | Projeção da velocidade do AGV em seu eixo $x$ |
| $v_{cy}$ | Projeção da velocidade do AGV em seu eixo $y$ |
| $\omega_c$ | Velocidade angular do AGV |
| $x_{ICR}$ | Projeção do ICR no eixo $x$ do AGV |
| $F_i$ | Força devido ao troque |
| $F_l$ | Força devido ao atrito lateral |
| $\mu_l$ | Coeficiente de atrito lateral |
| $N$ | Força normal gravitacional |
| $m$ | Massa do AGV |
| $\mathbf{M}$ | Matriz de massa e momento de inércia |
| $\mathbf{F}$ | Matriz de forças de atrito |
| $\mathbf{B}$ | Matriz de transmissão de torque |
| $\mathbf{S}$ | Matriz de mudança de coordenadas inerciais para coordenadas do AGV |
| $\eta$ | Vetor de velocidades lineares e angular do AGV com relação a ele mesmo |

# Sumário

# 1 Introdução

O ambiente industrial exige muitas vezes atividades de transporte de material, identificação e reconhecimento de itens e ambientes. Essas atividades frequentemente são perigosas, simples ou inconvenientes demais para um humano fazer. Por esse motivo é de grande interesse o uso de um AGV (Veículo Auto-Guiado, do inglês *Auto Guided Vehicle*) para a realização dessas tarefas (Automatic...).

Os AGVs são veículos auto guiados aos quais se podem acoplar diversos sensores para a realização de diversas tarefas, tornando-se bastante versáteis. O controle de trajetória usado em um AGV pode variar dependendo da aplicação do mesmo. Algumas estratégias são:

- Identificação de uma trajetória fixa destacada no chão;
- Reconhecimento de imagem, identificação do robô e do trajeto alvo por câmera;
- Determinação de posição por triangulação (GPS).

Na proposta deste trabalho está presente estudo e implementação de um sistema de identificação de uma trajetória fixa destacada no chão, a modelagem do AGV e do erro de trajetória.

## 1.1 Objetivos

Este Trabalho de Conclusão de Curso teve por objetivo o estudo, implementação e avaliação de técnicas de identificação de uma trajetória pintada no chão. Essas técnicas incluem a utilização de fotodiodos para aquisição de dados e cinco algoritmos de estimação de parâmetros para a identificação da faixa, sendo quatro desses algoritmos baseados em geometria e um baseado em redes neurais. Este trabalho também teve como objetivo a modelagem do AGV, a simulação de seu sistema em Matlab e a modelagem do erro de trajetória.

## 1.2 Estrutura do trabalho

Este relatório segue no Capítulo 2 com os materiais, métodos e avaliações utilizados para a obtenção do sistema de reconhecimento de faixa. É feita uma breve explicação sobre Redes Neurais e sobre os algoritmos de estimação utilizados. Ao final do Capítulo 2 são apresentados e discutidos os resultados obtidos com cada algoritmo.

No Capítulo 3 é apresentado a modelagem do AGV usada na construção do simulador, assim como a descrição e resultados deste. Posteriormente é mostrado a modelagem do erro de trajetória, que será usado como parâmetros do controlador.

# 2 Identificador de Faixa

É preciso que a trajetória a qual o veículo deverá seguir seja identificada para que o sistema de controle possa atuar com base na posição da mesma. A identificação da faixa é feita por meio de aquisição de dados, tratamento dos mesmos e a utilização de um algoritmo que estime a posição da faixa com base nos dados tratados. Os dados correspondem a sinais luminosos obtidos embaixo do veículo.

Como um dos algoritmos usados para estimar a posição da faixa é baseado em Redes Neurais Artificiais, este assunto será abordado brevemente. Em Simon (2008) o assunto de Redes Neurais é tratado de forma completa e aprofundada. Entretanto neste trabalho será abordado apenas o necessário para o leitor ter uma ideia do que são redes neurais, como elas funcionam e como elas foram aplicadas.

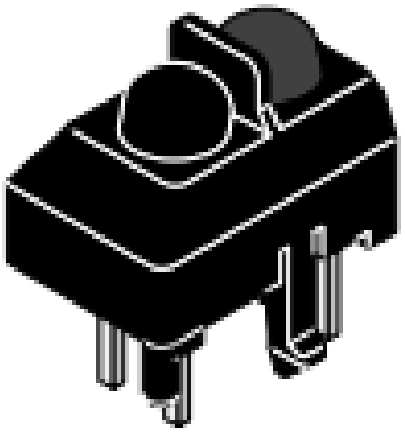
Este capítulo apresenta os equipamentos e métodos para obter bases de dados usados para a identificação da faixa. Em seguida, são descritos e testados cinco algoritmos de estimação da faixa.

## 2.1 Sensores

Os sensores de identificação da trajetória usados são fotodiodos TCRT5000(L), como é ilustrado na Figura 1. Os mesmos se encontram em um total de 12 e dispostos em duas filas de 6 espaçadas cerca de 4 cm uma da outra e os fotodiodos de uma mesma fila espaçados de cerca de 1 cm uns dos outros, como é mostrado na Figura 1.

Essa estrutura de sensores é controlada por um Arduino MEGA 5600 e foi montada de forma que uma faixa no chão possa ser detectada por meio de sua posição e angulação.

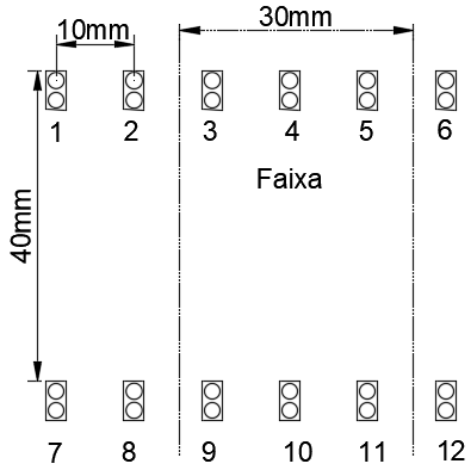
Figura 1 – Sensor TCRT5000(L)



Fonte: *datasheet* TCRT5000(L)

O sinal do sensor é interpretado em 10 bits variando de 0 a 1023, de forma que 0 implica reflexão máxima do sinal enviado e 1023 implica reflexão mínima do sinal enviado. Ou seja, valores altos indicam a provável presença da faixa. Ensaios experimentais realizados anteriormente sugeriram que uma faixa de largura de 3 cm seria ideal. A disposição dos sensores e da faixa são ilustrados na Figura 2.

Figura 2 – Estrutura de Sensores



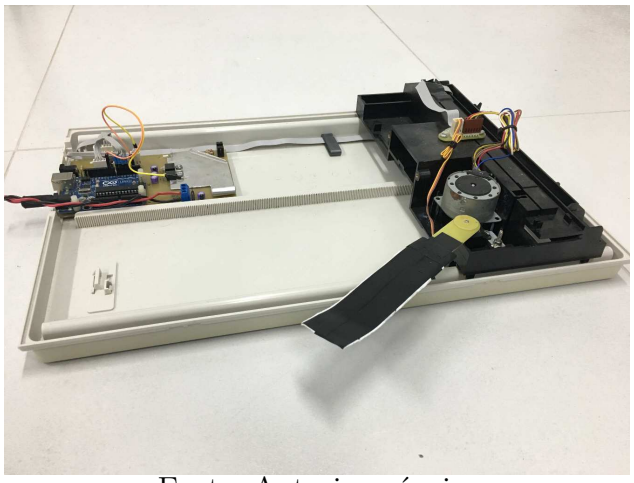

Fonte: Autoria própria

## 2.2 Plataforma simuladora de faixa

Controlada por um Arduino UNO, a plataforma é composta por uma base fixa, uma parte móvel, uma haste preta de 3cm de largura e dois motores (Figura 3). Os motores são presos na parte móvel de forma que um deles é responsável por transladar a mesma sobre a parte fixa e o outro tem como função a rotação da haste. O deslocamento da plataforma

e o giro da haste são controlados pelo Arduino. Dessa forma a plataforma simula uma faixa preta no chão de modo que as informações de posição e angulação são conhecidas.

Figura 3 – Plataforma simuladora de faixa



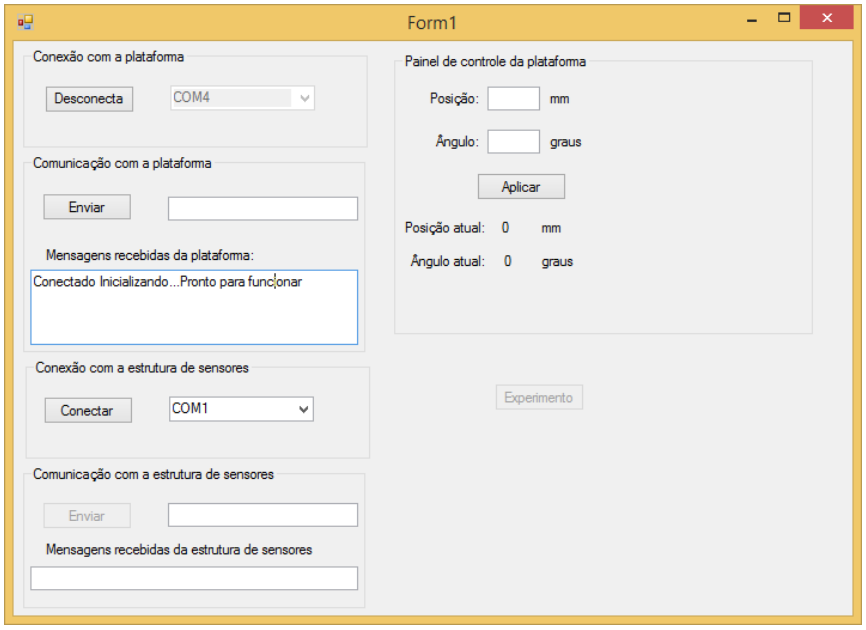
Fonte: Autoria própria

## 2.3 Obtenção de base de dados para análise

A comunicação entre a plataforma simuladora de faixa e a estrutura de sensores de forma a se obter dados foi feita por meio de uma interface em C#, onde um PC se comunica serialmente com ambas.

A interface foi desenvolvida com o auxílio de Clarisse P. B. Barros com o objetivo de obter uma base de dados que possua a informação acerca da posição e angulação real da faixa simulada e seus respectivos valores obtidos com os fotodiodos (Figura 4).

Figura 4 – Interface do controle da plataforma e dos sensores




Fonte: *print screen* da interface em C#

Inicialmente o usuário faz a conexão serial com o Arduino UNO, que controla a estrutura de sensores, e com o Arduino MEGA 5600, que controla a plataforma simuladora de faixa, por meio dos blocos "Conexão com a plataforma" e "Conexão com a estrutura de sensores". Em seguida o usuário zera a posição e o ângulo da plataforma por meio do bloco "Painel de controle da plataforma" e manualmente posiciona a plataforma de forma que a haste se encontre logo abaixo dos sensores 6 e 12 (estes são os sensores que se encontram na extremidade direita da estrutura). Esse passo é necessário pois os valores de angulação e posição interpretados pela plataforma são relativos e não absolutos.

Uma vez que os dois Arduinos estão conectados serialmente com o PC, o botão experimento poderá ser pressionado. Este inicia o processo de obtenção de dados.

O experimento consiste em fazer com que a faixa se encontre abaixo das duas filas de sensores e que varie entre os ângulos de -51º a 51º a passos de aproximadamente 10º, e que varie entre as posições de 0mm a 50mm a passos de 5mm. Tais valores foram escolhidos pois os mesmos são os valores limites que podem ser identificados pelos sensores. Os valores dos sensores para cada uma das posições da faixa são salvos em um arquivo texto.

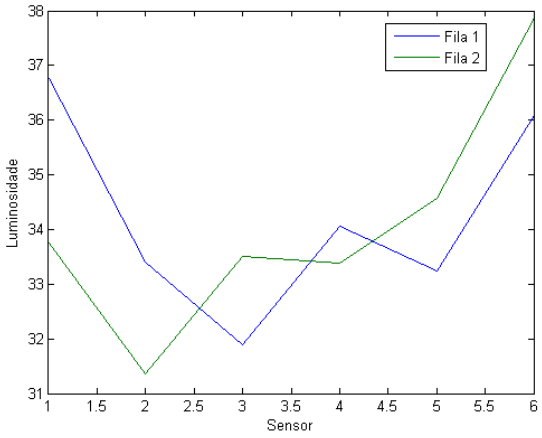
## 2.4 Tratamento de dados

Uma vez que se obteve o arquivo com o banco de dados do experimento, um programa em Matlab foi usado para tratar os dados e testar alguns algoritmos de estimação do posicionamento da trajetória.

O tratamento de dados é necessário pois o dado de posição real da plataforma possui uma referência diferente da referência usada pelos algoritmos propostos. O dado real da plataforma considera o sensor 6 como ponto zero e toma o sentido do sensor 6 ao sensor 1 como sentido positivo. Já para os algoritmos propostos o zero da posição se encontra no ponto médio da estrutura de sensores e toma o sentido sensor 1 ao sensor 6 como sentido positivo.

Também é necessário fazer a normalização dos dados enviados pelos sensores. Foi observado que os valores de reflexão infravermelha enviados pelo sensores são bem diferentes em média, principalmente dos sensores que se encontram nos extremos das filas. Isso se dá porque os sinais emitidos influenciam nos sinais recebidos pelos sensores adjacentes. Dessa forma, os sensores nas extremidades sofrem menor influência que os demais, pois só possuem um sensor adjacente cada. É mostrado na Figura 5 os sinais recebidos por cada um dos sensores para as duas filas de sensores no caso em que a estrutura de sensores se encontra acima de uma superfície plana branca.

Figura 5 – Sinais dos sensores em um plano branco




Fonte: Gráfico gerado pelo autor em Matlab

Antes de realizar a normalização é necessário se obter uma base de dados para o caso em que não existe faixa abaixo dos sensores, apenas uma superfície branca. Em seguida calcula-se a média de valores para cada sensor. A normalização é feita dividindo o valor enviado de cada sensor por seu respectivo valor médio.

## 2.5 Redes Neurais (NN)

As redes neurais são modelos matemáticos que tentam simular o sistema nervoso central biológico, principalmente o cérebro. Composto pela interconexão de até bilhões de neurônios, o cérebro de animais é bastante poderoso no que diz respeito à capacidade de aprendizado e reconhecimento de padrões.

Neurônios são células nervosas cuja função corresponde à transmissão de impulsos elétricos e são compostos basicamente por três estruturas; dendritos, corpo celular e axônio. Os dendritos são estruturas bastante ramificadas e sensíveis a estímulos nervosos, é onde ocorre a recepção dos sinais nervosos. O corpo celular contém o núcleo e as organelas do neurônio. O axônio é uma estrutura alongada responsável pela transmissão do impulso nervoso para outras células.

Os neurônios só transmitem impulsos nervoso quando os dendritos são estimulados acima de um certo limiar, caso contrário não há transmissão da informação. A característica das células nervosas que permitem a variação desse limiar é chamada de neuroplasticidade. A neuroplasticidade corresponde à capacidade que sistemas nervosos tem que se adaptar à presença de certos estímulos. Em outras palavras, quanto mais se estimula uma célula nervosa, menor é o limiar necessário para que a mesma transmita o estímulo.

A seção segue com a definição matemática de *neurônio*, algumas funções de ativação usadas, algumas formas de se arquitetar redes neurais, regras e paradigmas de aprendizagem e conclui com um exemplo de um algoritmo de aprendizagem.

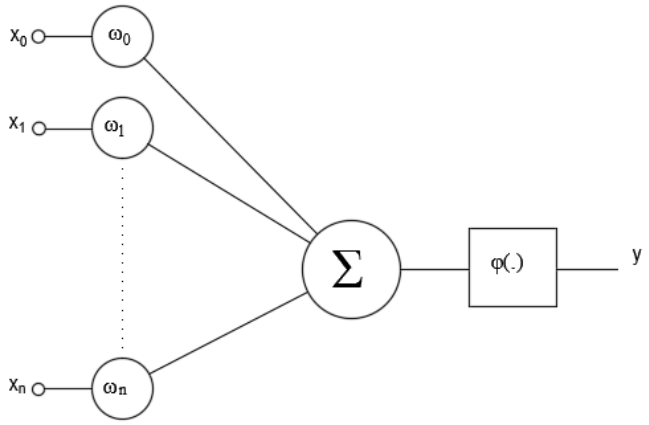
As redes neurais consistem em redes de "neurônios", que apresentam uma função de transferência ou função de ativação, de forma que cada neurônio obedece ao seguinte modelo matemático

$$y = \varphi(v)$$

$$v = w_0 + \sum_{i=1}^{j} w_i x_i$$

onde $y$ é a saída do neurônio e $v$ é o nível de ativação, que corresponde à uma soma de produtos dos sinais de entrada $x_i$ por um peso $w_i$ somado de uma constante de polarização $w_0$ (Figura 6).

Figura 6 – Modelo do neurônio




Fonte: Autoria própria

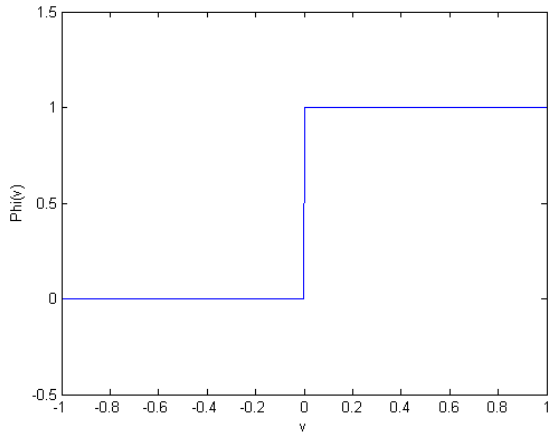
### 2.5.1 Tipos de Funções de Ativação

Existem diversos tipos de funções de ativação, algumas delas são:

*Função de Heaviside* (Figura 7). Definida como

$$\varphi(v) = \begin{cases} 1, & \text{se } v \geq 0 \\ 0, & \text{se } v < 0 \end{cases}$$

Figura 7 – Função Heaviside

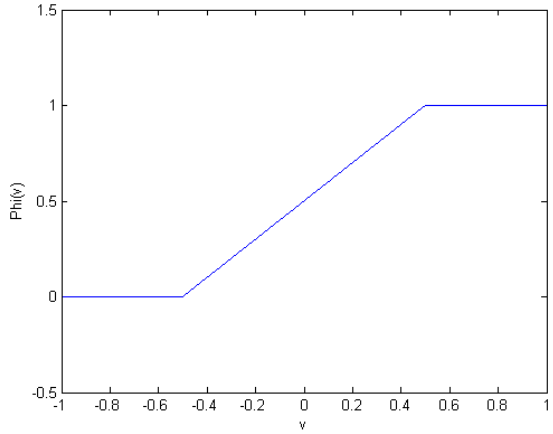



Fonte: Gráfico gerado pelo autor em Matlab

*Função Linear por Partes* (Figura 8). Definida como

$$\varphi(v) = \begin{cases} 1, & v \geq \frac{1}{2} \\ 0, & \frac{1}{2} > v > -\frac{1}{2} \\ \text{-1}, & v \leq -\frac{1}{2} \end{cases}$$

Figura 8 – Função Linear



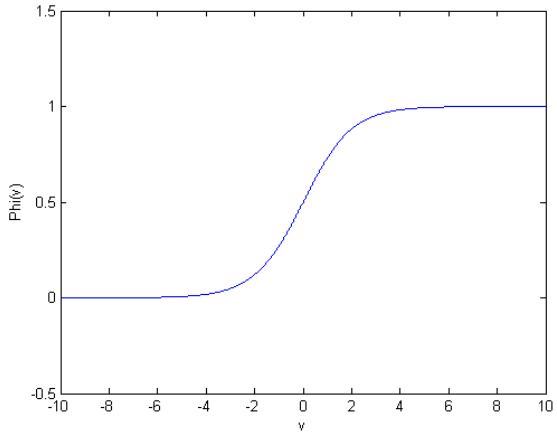

Fonte: Gráfico gerado pelo autor em Matlab

*Função Sigmoide* (Figura 9). Definida como

$$\varphi(v) = \frac{1}{1 + exp^{-av}}$$

onde $a$ é o parâmetro de inclinação da função sigmoide.

Figura 9 – Função Sigmoide




Fonte: Gráfico gerado pelo autor em Matlab

### 2.5.2 Arquiteturas de Redes Neurais

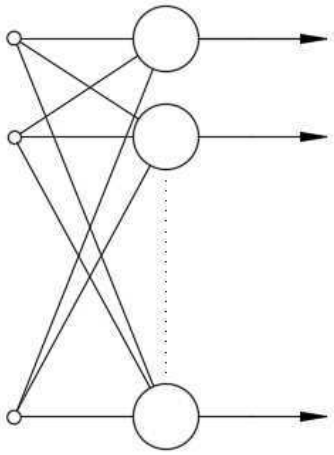
A estrutura de uma rede neural está relacionada com o propósito da mesma e corresponde à disposição dos neurônios na rede. Quanto a sua topologia, pode-se classificar as redes neurais em dois tipos, *redes feed-forward* e *redes recorrentes*.

***Redes Feed-forward***

As *redes feed-forward* podem ser subdivididas em dois grupos, *monocamada* e *multicamada* com alimentação unidirecional. Como o nome sugere as *redes feed-forward* não possuem realimentação, o sinal é sempre enviado para os neurônios a frente quando possível.

As *redes monocamada* com alimentação unidirecional só possuem uma camada de neurônios. Os mesmos neurônios que recebem os sinais de entrada fornecem os sinais de saída (Figura 10).

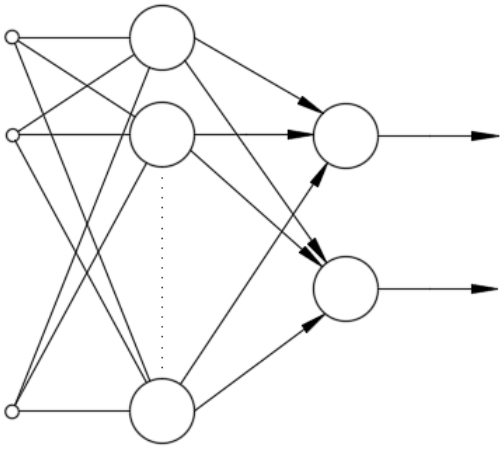
Figura 10 – Rede monocamada



Fonte: Autoria própria

As *redes multicamada* com alimentação unidirecional possuem ao menos duas camadas, uma camada de saída e uma camada escondida (Figura 11).

Figura 11 – Rede multicamada



Fonte: Autoria própria

***Redes Recorrentes***

As redes neurais são classificadas como *Redes Recorrentes* se tiverem ao menos um laço de realimentação (Figura 13). Uma realimentação implica dizer que a saída de um neurônio depende dela mesma.

Figura 12 – Rede Recorrente

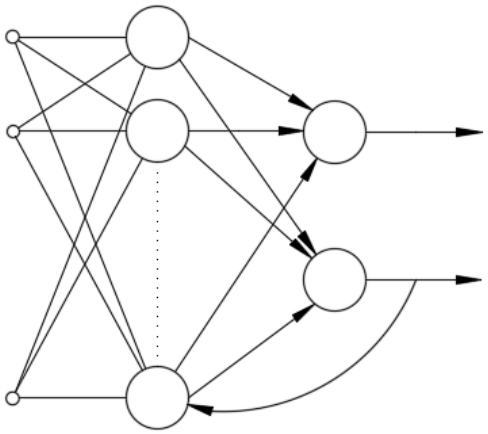


Figura 13 – Rede Recorrente

### 2.5.3 Aprendizagem

A característica mais importante de uma rede neural é a sua capacidade de *aprender*. A aprendizagem de uma rede neural está diretamente ligada com sua eficiência. Ela consiste na modificação de seus parâmetros devido à estímulos do ambiente no qual ela está inserida. Esse processo ocorre de forma iterativa, ou seja, a cada iteração os pesos sinápticos e de polarização mudam de forma que a rede fique cada vez mais *ciente* do seu ambiente.

Os dados usados para fazer com que uma rede neural *aprenda* são chamados de dados de treinamento. Eles são formados por informações sobre entradas próximas o bastante das que provavelmente ocorrerão quando a rede estiver em uso e suas respectivas saídas desejadas. Os dados de treinamento passam para a rede informações sobre o seu ambiente e sobre seu comportamento desejado. A forma com a qual a rede modifica seus parâmetros define o algoritmo de treinamento.

No processo de aprendizagem exitem as chamadas *Regras de aprendizagem* e os *Paradigmas de aprendizagem*. A seguir serão abordadas três regras de aprendizagem; *aprendizagem por correção de erro*, *aprendizagem baseada em memória* e *aprendizagem Hebbiana*. Logo após serão abordados dois paradigmas de aprendizagem; *supervisionada* e *não-supervisionada*.

### 2.5.4 Aprendizagem por correção de erro

Esse tipo de aprendizagem se baseia na diferença entre a saída real da rede e a saída desejada para uma entrada específica. Tomando como exemplo o caso simples de apenas um neurônio $k$. Considere que esse neurônio recebe um vetor de entrada $x(n)$ ocasionando em uma saída $y_k(n)$. Esta saída é comparada com o sinal de saída desejado $d_k(n)$, gerando

assim um sinal de erro definido como

$$e_k(n) = d_k(n) - y_k(n)$$

O sinal de erro é usado como entrada para um sistema de controle que modifica os pesos das sinapses de forma que a resposta real do neurônio se aproxime da resposta desejada. Isso é feito minimizando-se a *função de custo*, que é definida como

$$E(n) = \frac{1}{2}e_k^2(n)$$

Essa função de custo é interpretada como a energia instantânea do erro. Quando sua derivada é zero, quer dizer que ela foi minimizada, ou seja, foi encontrado o menor valor de erro possível. O objetivo da aprendizagem por correção de erro é definir os pesos sinápticos necessários para se obter um erro menor ou igual ao erro mínimo tolerável $e(n) \leq \varepsilon$. Mais adiante será mostrado detalhadamente um algoritmo chamado *Gradiente Descendente*, que usa a regra de aprendizagem por correção de erro.

### 2.5.5 Aprendizagem baseada em memória

Na aprendizagem baseada em memória, exemplos de experiências que contém informações corretas sobre entradas e saídas são armazenadas em memórias. A classificação de um vetor $x$ que não tenha sido visto antes é feita analisando-se a *vizinhança local* de $x$.

Os algoritmos que fazem uso da aprendizagem baseada em memória são diferenciados entre si pela forma com a qual fazem uso de duas escolhas necessárias. Como definir a *vizinhança local* de $x$ e a regra de aprendizagem aplicada aos exemplos de treinamento na *vizinhança local* de $x$

Uma regra bastante simples usada em algoritmos de aprendizagem baseada em memória é a chamada *regra do vizinho mais próximo*. Ela consiste em classificar um vetor de teste $x_t$ no vetor de treinamento $x_n$ que possuir a menor distância euclidiana com relação ao vetor de teste. Ou seja, caso

$$\min_i d(x_i, x_t) = d(x_n, x_t)$$

em que

$$i = 1, 2, ..., N$$

e $d(x_i, x_t)$ é a distância euclidiana entre o vetor de teste e o vetor de treinamento $x_i$. Então $x_t$ será classificado como $x_n$.

### 2.5.6 Aprendizagem Hebbiana

Baseada nos estudos do neuropsicólogo Hebb(1949) a aprendizagem Hebbiana consiste no fortalecimento ou enfraquecimento de sinapses como função da correlação temporal entre atividades *pré-sinápticas* e *pós-sinápticas*. A regra de Hebb dita que

- "Se dois neurônios em ambos os lados de uma sinapse são ativados simultaneamente, então a força daquela sinapse é seletivamente aumentada"
- "Se dois neurônios em ambos os lados de um sinapse são ativados assincronamente, então aquela sinapse é seletivamente enfraquecida ou eliminada"

Uma forma bastante simples de aprendizagem Hebbiana é apresentada pelo seguinte modelo matemático

$$\Delta\omega(n) = \beta \times y(n)x(n)$$

onde $y(n)$ e $x(n)$ representam os sinais pós-sinápticos e pré-sinápticos respectivamente, e $\beta$ é uma constante positiva definida como *taxa de aprendizagem*.

### 2.5.7 Aprendizagem supervisionada

No que diz respeito aos paradigmas de aprendizagem, a aprendizagem supervisionada consiste naquela quando existe um indicador da resposta desejada que é apresentada a rede, ou um *professor*. Nela tanto o professor quanto a rede são apresentados à vetores de treinamento característicos de um certo ambiente. O professor é capaz de informar à rede qual comportamento da mesma é o desejado.

Supondo que inicialmente a rede não possui qualquer conhecimento acerca do ambiente, é de se esperar que as primeiras saídas obtidas dos vetores de treinamento estejam completamente incoerentes com as saídas desejadas, de forma que se tem sinal de erro considerável. O erro é usado em *funções de custo* como o sinal de referência que o algoritmo de aprendizado possui para fazer os ajustes dos parâmetros da rede.

### 2.5.8 Aprendizagem não supervisionada

Neste tipo de aprendizagem não há a presença de um *professor*, a rede usa apenas as entradas como parâmetros de classificação. A rede aprende a classificar padrões semelhantes e diferenciá-los uns dos outros.

### 2.5.9 Gradiente Descendente

Para que uma rede neuronal funcione é preciso, após a fase de arquitetura, passar por uma etapa de treinamento. Para o caso de treinamento supervisionado, a etapa de treinamento consiste em informar a rede valores de entrada e seus respectivos valores desejados de saída, para que os pesos $w_i$ sejam calculados de forma a se obter o menor erro tolerável. Uma vez treinada, a rede está pronta para ser testada.

Um algoritmo bastante simples usado no treinamento supervisionado de NN é o chamado **gradiente descendente**. Ele é implementado da seguinte forma, considere o

seguinte sinal de saída de uma unidade linear

$$o = \sum_{i=1}^{j} w_i x_i$$

e a seguinte função de custo

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

onde

- $D$ é o conjunto de treinamento
- $t_d$ é a saída desejada para o dado de treinamento $d$
- $od$ é a saída da unidade linear para o dado de treinamento $d$

Deseja-se encontrar o vetor $\vec{w}$ que minimiza a função $E$. O gradiente da função de custo é dado por

$$\nabla E(\vec{w}) = \left[ \frac{\partial E}{\partial w_0} \frac{\partial E}{\partial w_1} \cdots \frac{\partial E}{\partial w_j} \right]$$

A atualização dos pesos é feita da seguinte forma

$$\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$$

em que

$$\Delta \vec{w} = -\beta \nabla E(\vec{w})$$

e $\beta$ é chamado de taxa de aprendizagem. Assim, temos

$$\begin{aligned} \frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i} \frac{1}{2} \sum_d (t_d - o_d)^2 \\ &= \frac{1}{2} \sum_d \frac{\partial}{\partial w_i} (t_d - o_d)^2 \\ &= \sum_d (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d) \\ \frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{id}) \end{aligned}$$

Logo, a atualização dos pesos se torna

$$w_i \leftarrow w_i + \beta \sum_d (t_d - o_d) x_{id}$$

Os pesos são modificados até que o erro seja menor ou igual à um limiar de tolerância, $e \leq \varepsilon$

É necessário ter em mente que o desempenho de uma rede neural varia bastante tanto com a arquitetura escolhida, quanto com os elementos usados na fase de treinamento. Uma vez que uma rede foi treinada ela já pode ser imersa no ambiente para o qual foi projetada e ser usada.

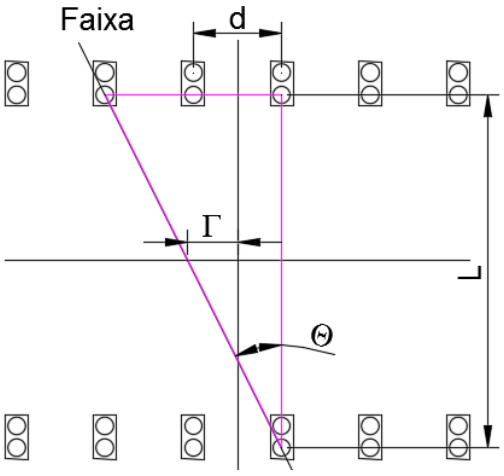
## 2.6 Algoritmos de estimação da faixa

Os algoritmos de estimação de posição da faixa fazem uso das variáveis $\Gamma$ e $\theta$ para realizar a estimação. O $\Gamma$ significa a distância, paralela ao eixo dos motores, da trajetória para o centro da estrutura de sensores. O $\Theta$ significa o ângulo com relação ao eixo perpendicular ao eixo dos motores da tangente da trajetória no ponto de cálculo do $\Gamma$ (Figura 14).

Para a realização do controle de trajetória se faz necessário estimar a posição da faixa com base nas únicas informações externas que o Arduino pode obter, neste caso são as informações obtidas pelos fotodiodos.

A seguir são descritos cinco algoritmos de estimação para as variáveis $\Gamma$ e $\Theta$ a partir dos valores obtidos com os sensores. Desses, quatro são geométricos e um é neural. Como o nome sugere, os algoritmos geométricos se baseiam na disposição geométrica dos sensores e em relações de triângulos para determinar as variáveis. Já o algoritmo neural consiste em uma rede neural treinada com dados provenientes dos sensores e da plataforma simuladora de faixa.

Figura 14 – Algoritmos Geométricos




Fonte: Autoria própria

Todos os algoritmos citados foram testados com uma base de dados de teste de 1240 amostras para cada sensor. Uma vez que se tem os valores reais de $\Gamma$ e $\Theta$, pode-se verificar os resultados obtidos com os algoritmos e compará-los.

### 2.6.1 Algoritmo Geométrico A

Algoritmo proposto por Geovanny (1998) se baseia nos índices dos sensores de cada fila que acusarem menor luminosidade para o cálculo do $\Gamma$ e do $\Theta$.

Considerando $i_1$ o índice do sensor da fila 1 que retorna valor de menor luminosidade e $i_2$ o índice do sensor da fila 2 que retorna valor de menor luminosidade, a distância de

cada um desses sensores para o centro de sua fila é dada por

$$d_n = i_n d - 35$$

Os valores de $\Gamma$ e $\Theta$ são calculados da seguinte forma

$$\Gamma = \frac{d_1 + d_2}{2}$$

$$\Theta = atan(\frac{d_1 - d_2}{L})$$

### 2.6.2 Algoritmo Geométrico B

Algoritmo proposto por Geovanny (1998) faz uma ponderação, com seus respectivos valores de luminosidade, dos dois índices dos sensores de cada uma das filas que apresentam menor luminosidade para o cálculo das variáveis, de forma que se tem

$$p_1 = \frac{d_{1a}S_{1a} + d_{1b}S_{1b}}{S_{1a} + S_{1b}}$$
$$p_2 = \frac{d_{2a}S_{2a} + d_{2b}S_{2b}}{S_{2a} + S_{2b}}$$

$$\Gamma = \frac{p_1 + p_2}{2}$$
$$\Theta = atan(\frac{p_1 - p_2}{L})$$

onde

- $p_1$ é a posição onde a faixa intercede a primeira fila de sensores;
- $p_2$ é a posição onde a faixa intercede a segunda fila de sensores;
- $d_{1a}$ é a distância para o centro de sua fila do sensor da primeira fila que acusa menor luminosidade
- $d_{1b}$ é a distância para o centro de sua fila do sensor da primeira fila que acusa segunda menor luminosidade
- $d_{2a}$ é a distância para o centro de sua fila do sensor da segunda fila que acusa menor luminosidade
- $d_{2b}$ é a distância para o centro de sua fila do sensor da segunda fila que acusa segunda menor luminosidade
- $S_{1a}$ é o valor retornado pelo sensor da primeira fila que acusa menor luminosidade;

- $S_{1b}$ é o valor retornado pelo sensor da primeira fila que acusa segunda menor luminosidade;
- $S_{2a}$ é o valor retornado pelo sensor da segunda fila que acusa menor luminosidade;
- $S_{2b}$ é o valor retornado pelo sensor da segunda fila que acusa segunda menor luminosidade;

### 2.6.3 Algoritmo Geométrico C

Algoritmo proposto por Balaji et al. (2015) faz uma ponderação, com seus respectivos valores de luminosidade, de todos os índices de sensores, de forma que se tem

$$p_1 = \frac{\sum_n d_{1n} S_{1n}}{\sum_n S_{1n}}$$
$$p_2 = \frac{\sum_n d_{2n} S_{2n}}{\sum_n S_{2n}}$$

$$\Gamma = \frac{p_1 + p_2}{2}$$
$$\Theta = atan(\frac{p_1 - p_2}{L})$$

### 2.6.4 Algoritmo Geométrico D

Algoritmo proposto por Clarisse P. B. Barros considera $\Gamma$ como sendo a distância em que a trajetória intercepta a primeira fila de sensores para o centro da mesma, de forma que se tem

$$\Gamma = p_1$$
$$\Theta = atan(\frac{p_1 - p_2}{L})$$

em que $p_1$ e $p_2$ calculados de forma semelhante ao algoritmo geométrico B.

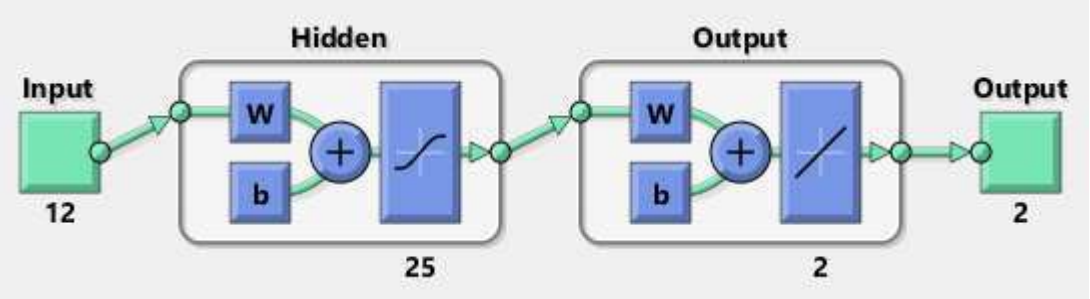
### 2.6.5 Algoritmo Neuronal (NN)

Foi criada uma rede neural *feed-forward* usando a ferramenta do Matlab nftool (Figura 15). A nftool pertence a caixa de ferramentas *NN start* do Matlab e é uma ferramenta usada na construção e treinamento de redes neurais artificiais *feed-forward* com aprendizagem baseada em erro. Ela possui como configuração padrão a utilização de 15% dos dados totais para validação e 5% para teste. As rede neurais criadas com essa ferramenta possuem apenas duas camadas. A camada escondida possui neurônios ativados

por uma função sigmoide, já a camada de saída possui neurônios ativados por uma função linear.

A rede neutral criada possui duas camadas com 25 neurônios em sua camada escondida. A rede foi treinada com uma base de dados de 992 valores para cada sensor, sendo 793 amostras para treinamento, 149 para validação e 50 para teste. O número de neurônios e a quantidade de elementos foram escolhidos empiricamente com base no desempenho da rede.

Figura 15 – Estrutura da Rede Neural utilizada

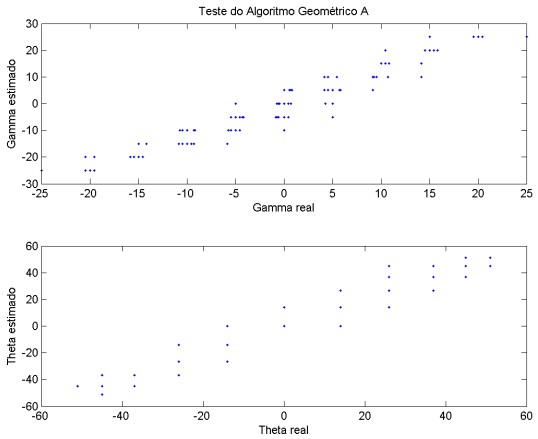



Fonte: *print screen* da extrutura darede neura em Matlab

## 2.7 Apresentação e Análise dos Resultados

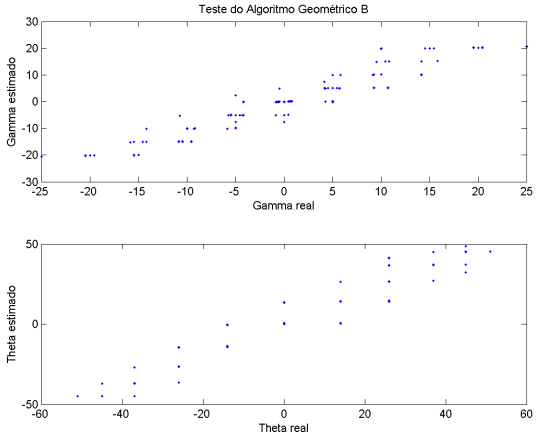
Os algoritmos foram testados utilizando uma base de dados obtida pela plataforma de simulação de faixa com a estrutura de sensores. Após o teste de cada algoritmo foi construído um gráfico com os valores reais das variáveis $\Gamma$ e $\Theta$ no eixo $x$ e os valores estimados pelo algoritmo no eixo $y$ (Figuras 16 - 20).

Figura 16 – Resultados Algoritmo A



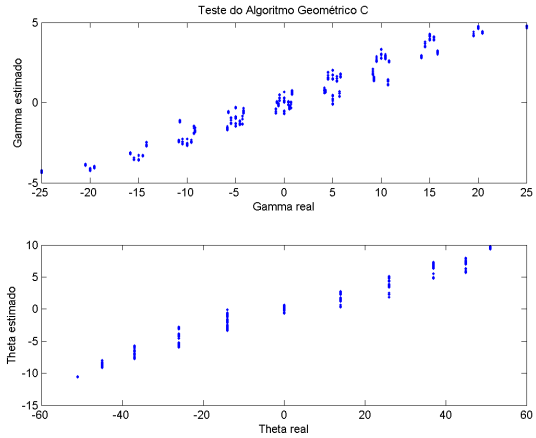

Fonte: Gráfico gerado pelo autor em Matlab

Figura 17 – Resultados Algoritmo B



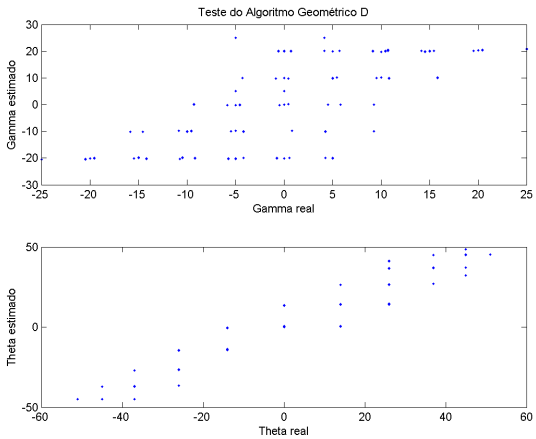

Fonte: Gráfico gerado pelo autor em Matlab

Figura 18 – Resultados Algoritmo C



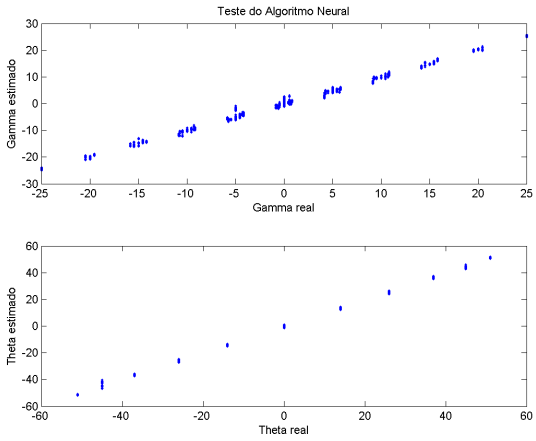

Fonte: Gráfico gerado pelo autor em Matlab

Figura 19 – Resultados Algoritmo D




Fonte: Gráfico gerado pelo autor em Matlab

Figura 20 – Resultados Algoritmo Neural




Fonte: Gráfico gerado pelo autor em Matlab

Para cada algoritmo foi calculado o erro médio, máximo e a variância para cada variável. Os erros e a variância foram calculados da seguinte forma

$$Err_{max} = max(\mid \vec{v_r} - \vec{v_e} \mid)$$

$$Err_{med} = \frac{1}{N} \sum_{n=1}^{N} \mid v_{rn} - v_{en} \mid$$

$$Var = \frac{1}{N} \sum_{n=1}^{N} (v_{rn} - v_{en})^2$$

onde

- $N$ é o número de amostras de teste
- $\vec{v_r}$ é o vetor de valores reais do $\Gamma$ ou $\theta$
- $\vec{v_e}$ é o vetor de valores do $\Gamma$ ou $\theta$ estimados pelo algoritmo

Como forma de decidir qual dos algoritmos de estimação da faixa é o mais eficiente, foram montadas tabelas que mostram os erros médios, máximos e variância de cada variável para os algoritmos previamente descritos (Tabela 1 e Tabela 2 ).

Tabela 1 – Erros de estimação de $\Gamma$

| **Erros** $\mid \Gamma_{real} - \Gamma_{estimado} \mid$ | $\Delta\Gamma_{med}$**(mm)** | $\Delta\Gamma_{max}$**(mm)** | $Var(\Gamma)$ |
|---|---|---|---|
| Algoritmo A | 3.27 | 10 | 17.23 |
| Algoritmo B | 2.39 | 9.84 | 10.42 |
| Algoritmo C | 7.42 | 20.79 | 85.25 |
| Algoritmo D | 3.74 | 11.1 | 106.7 |
| Algoritmo neural | 0.52 | 4.18 | 0.543 |

Tabela 2 – Erros de estimação de $\Theta$

| **Erros** $\mid \Theta_{real} - \Theta_{estimado} \mid$ | $\Delta\Theta_{med}(^\circ)$ | $\Delta\Theta_{max}(^\circ)$ | $Var(\theta)$ |
|---|---|---|---|
| Algoritmo A | 4.12 | 19 | 48.8 |
| Algoritmo B | 4.14 | 15.26 | 47.77 |
| Algoritmo C | 18.21 | 41.66 | 482.93 |
| Algoritmo D | 4.14 | 15.26 | 47.77 |
| Algoritmo neural | 0.54 | 4.74 | 0.54 |

Com base nas tabelas é possível verificar que o algoritmo neural obteve o melhor resultado, pois obteve o menores erros e variância para $\Gamma$ e $\Theta$ dentre todos os algoritmos.

Com relação aos algoritmos geométricos, o algoritmo B obteve os menores erros se comparados com os dos outros algoritmos geométricos, com exceção do $\Delta\Theta_{med}$ que foi $0.02^\circ$ maior que o do algoritmo geométrico A. Entretanto, como o $\Delta\Theta_{max}$ do algoritmo B foi quase $4^\circ$ menor que o do algoritmo A, o algoritmo B foi considerado o mais eficiente dentre os algoritmos geométricos.

Um erro de até cerca de 10mm em $\Gamma$ é compreensível, tendo em vista a possível presença de perturbações vindas do ambiente e que a distância entre sensores adjacentes é de 10mm. Esse foi o caso para os algoritmos A, B e D.

O algoritmo A não é uma opção interessante a princípio, pois é bastante impreciso devido a ser capaz de estimar apenas 36 posições diferentes para a faixa. Já os algoritmos B, C e D são capazes de estimar a faixa em um intervalo contínuo entre dois sensores. Isso ocorre pois eles ponderam com seus respectivos valores de luminosidade os dois sensores que acusam menor luminosidade. Isso justifica os algoritmos B e D possuírem em geral erros menores que o algoritmo A.

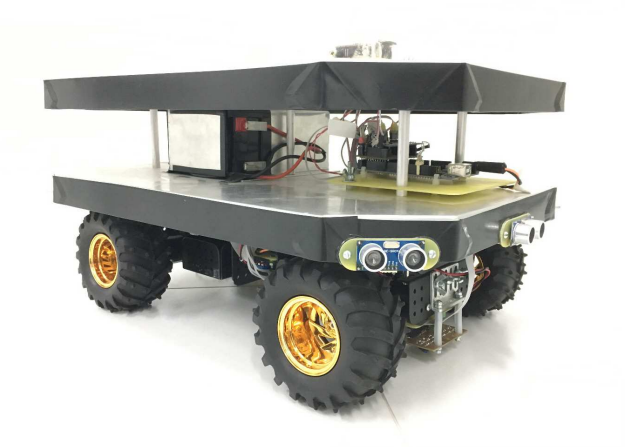
O problema do algoritmo C está no fato de que, mesmo quando não há faixa, os sensores retornam valores. Isso implica que, pela ponderação de todos os valores, o algoritmo C tende a estimar o $\Gamma$ mais ao centro da estrutura de sensores do que ele realmente está. O fato de levar em conta todos os valores de sensores também faz com que o algoritmo seja mais susceptível ao erro, pois será contabilizado qualquer ruído que afetar qualquer sensor.

Uma justificativa para parte dos erros encontrados nos algoritmos que fazem o uso da ponderação está no fato que a relação entre $\Gamma$ e luminosidade não é linear. Ou seja, mesmo que a normalização dos valores enviados pelos sensores diminua o erro devido à influência dos sinais dos sensores adjacentes, a ponderação não representa uma relação fiel entre $\Gamma$ e a luminosidade.

# 3 AGV Conceitos e Modelagem

O veículo estudado neste trabalho é um AVG deslizante (Figura 21). Ele possui dois pares paralelos de rodas paralelas de forma que para realizar um trajeto curvo um deslizamento paralelo ao eixo das rodas é necessário, isso ocorre porque as rodas acabam por não tangenciar a curvatura da trajetória. O controle de um veículo com essas características é desafiador pois dependendo da projeção do ICR (Centro de Rotação Instantânea, do inglês *Instant Center of Rotation*) no eixo longitudinal do veículo pode ocorrer perda de estabilidade de movimento.

Figura 21 – AGV



Fonte: Autoria própria

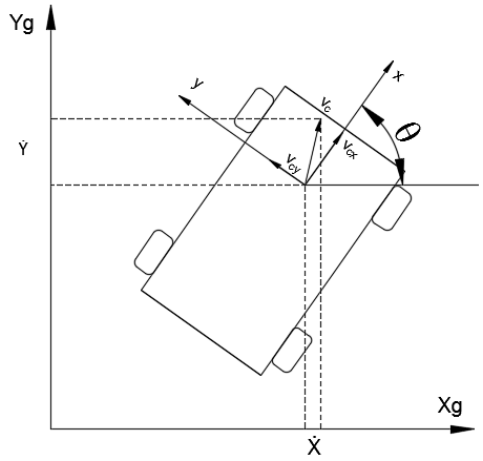
## 3.1 Modelagem do AGV deslizante

É realizada a modelagem deste sistema para o desenvolvimento de um simulador e um controlador robusto eficiente. A modelagem desenvolvida a seguir considera modelos cinemáticos e dinâmicos do veículo e é feita tomando como base Pazderski, Kozlowski e W.E (2004) e Pazderski e Kozlowski (2006).

### 3.1.1 Modelo Cinemático

O modelo cinemático é feito com os modelos cinemáticos do corpo livre do veículo com o modelo cinemático das rodas. Na Figura 22 o diagrama de corpo livre do veículo

Figura 22 – Diagrama de corpo livre




Fonte: Autoria prórpria

Por simplicidade, é considerado que o veículo vai percorrer uma superfície plana. Definimos que $Xg$ e $Yg$ são as coordenadas do referencial inercial do AGV, $x$ e $y$ são as coordenadas do referencial do carro e $v_c$ é o vetor velocidade do veículo, $v_{cx}$ e $v_{cy}$ representam a projeção do vetor $v_c$ nos eixos $x$ e $y$, respectivamente. O vetor de velocidade linear a angular do veículo é definido como

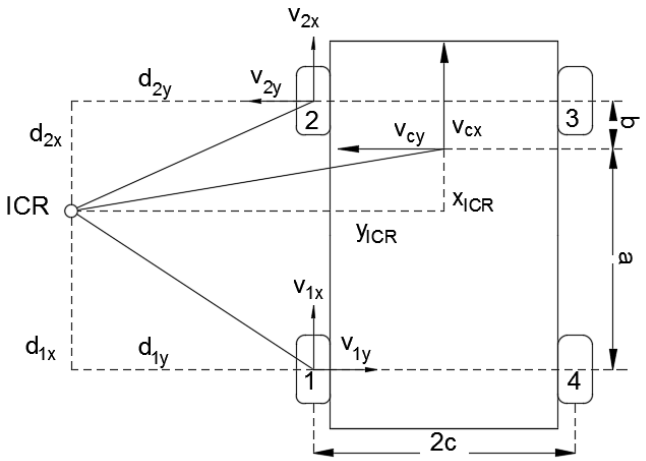
$$\dot{q} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix}, \tag{3.1}$$

onde $\dot{X}$ e $\dot{Y}$ representam as coordenadas de velocidade linear do carro no eixo de coordenadas inerciais e $\dot{\theta}$ representa a velocidade angular do sistema de coordenadas do carro com relação ao sistema inercial. É possível observar pela Figura 22 a seguinte relação entre o sistema de coordenadas do carro e o inercial

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{bmatrix} \times \begin{bmatrix} v_{cx} \\ v_{cy} \end{bmatrix}. \tag{3.2}$$

Para a cinemática das rodas, definimos $\omega_i$ para $i = 1, 2, 3, 4$ como sendo a velocidade angular da roda $i$, e $\omega_c$ como sendo a velocidade angular do veículo. É preciso explicitar que a tração diferencial do AGV implica que as rodas da esquerda possuem a mesma velocidade e as rodas da direita também, ou seja, $\omega_e = \omega_1 = \omega_2$ e $\omega_d = \omega_3 = \omega_4$, onde $\omega_e$ e $\omega_d$ correspondem às velocidades angulares das rodas da esquerda e da direita, respectivamente.

Figura 23 – Relação entre distâncias e velocidades




Fonte: Autoria própria

É ilustrado na Figura 23 as velocidades do carro e das rodas 1 e 2 assim como suas distâncias correspondentes para o ICR. As velocidades e distâncias das rodas 3 e 4 foram desconsideradas da Figura 23 por motivos de facilidade de visualização, entretanto as relações desenvolvidas a seguir valem para todas as rodas. É possível observar pela geometria na Figura 23 as seguintes relações

$$\omega_c = \frac{v_{ix}}{d_{iy}} = \frac{v_{iy}}{d_{ix}} = \frac{v_{cx}}{y_{ICR}} = \frac{v_{cy}}{-x_{ICR}}, \tag{3.3}$$

onde

- $\omega_c$ é a velocidade angular do AGV
- $ICR$ é o centro instantâneo de rotação
- $v_{ix}$ é a velocidade da roda $i$ na direção $x$
- $v_{iy}$ é a velocidade da roda $i$ na direção $y$
- $d_{ix}$ é a distância na direção $x$ da roda $i$ para o ICR
- $d_{iy}$ é a distância na direção $y$ da roda $i$ para o ICR
- $x_{ICR}$ é a coordenada $x$ da posição do ICR
- $y_{ICR}$ é a coordenada $y$ da posição do ICR

A partir equação (3.3) é possível observar a seguinte restrição

$$\omega_c \times x_{ICR} + v_{cy} = 0 \tag{3.4}$$

A Figura 23 também fornece informação sobre as distâncias de cada roda para o ICR. Pode-se observar que

$$\begin{aligned} d_{1y} &= d_{2y} = d_{cy} - c \\ d_{3y} &= d_{4y} = d_{cy} + c \\ d_{1x} &= d_{4x} = b + d_{cx} \\ d_{2x} &= d_{3x} = a - d_{cx} \end{aligned} \tag{3.5}$$

As distâncias $a$, $b$ são as distâncias do centro de massa do veículo para os eixos das rodas na direção $x$ e $c$ é a distância do centro demassa do veículo para os eixos na direção $y$.

Como assumimos que a velocidade angular das rodas são iguais para cada um dos lados, temos

$$\begin{bmatrix} \omega_L = \omega_1 = \omega_2 \\ \omega_R = \omega_3 = \omega_4 \end{bmatrix} = \begin{bmatrix} v_{1x} \\ v_{3x} \end{bmatrix} \frac{1}{r}, \tag{3.6}$$

onde $r$ é o raio das rodas.

Por meio das equações de (3.3) a (3.6) é possível encontrar

$$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \begin{bmatrix} \omega_c d_{1y} \\ \omega_c d_{3y} \end{bmatrix} \frac{1}{r} = \begin{bmatrix} \omega_c (d_{cy} - c) \\ \omega_c (d_{cy} + c) \end{bmatrix} \frac{1}{r} = \begin{bmatrix} \omega_c d_{cy} + v_{cy} \frac{c}{x_{ICR}} \\ \omega_c d_{cy} - v_{cy} \frac{c}{x_{ICR}} \end{bmatrix} \frac{1}{r},$$

$$= \frac{1}{r} \begin{bmatrix} 1 & \frac{c}{x_{ICR}} \\ 1 & -\frac{c}{x_{ICR}} \end{bmatrix} \begin{bmatrix} v_{cx} \\ v_{cy} \end{bmatrix},$$

$$\begin{bmatrix} \omega_L \\ \omega_R \end{bmatrix} = \frac{1}{r} \begin{bmatrix} 1 & \text{c} \\ 1 & \text{-c} \end{bmatrix} \begin{bmatrix} v_{cx} \\ \omega_c \end{bmatrix}. \tag{3.7}$$

De forma análoga, para $v_{1y}$ e $v_{2y}$ temos

$$\begin{bmatrix} v_{1y} = v_{4y} \\ v_{2y} = v_{3y} \end{bmatrix} \begin{bmatrix} \omega_c d_{1x} \\ \omega_c d_{2x} \end{bmatrix} = \begin{bmatrix} \omega_c (a - d_{cx}) \\ \omega_c (b + d_{cx}) \end{bmatrix} = \begin{bmatrix} -\omega_c d_{cx} - v_{cy} \frac{a}{x_{ICR}} \\ \omega_c d_{cx} - v_{cy} \frac{b}{x_{ICR}} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 - \frac{a}{x_{ICR}} \\ 0 & 1 - \frac{b}{x_{ICR}} \end{bmatrix} \begin{bmatrix} v_{cx} \\ v_{cy} \end{bmatrix}$$

$$\begin{bmatrix} v_{1y} \\ v_{2y} \end{bmatrix} = \begin{bmatrix} 0 & -x_{ICR} - a \\ 0 & -x_{ICR} + b \end{bmatrix} \begin{bmatrix} v_{cx} \\ \omega_c \end{bmatrix} \tag{3.8}$$

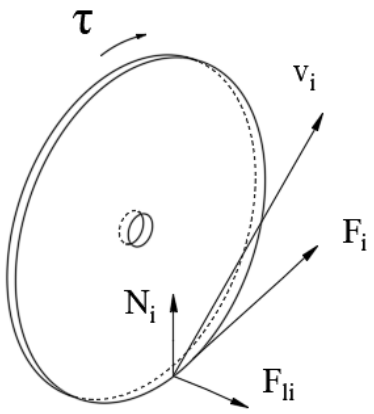
A equação (3.8) mostra que para se fazer o controle lateral do deslizamento é necessário o conhecimento de $x_{ICR}$.

### 3.1.2 Modelo Dinâmico

O modelo dinâmico das rodas leva em consideração forças de atrito devido o deslizamento, força normal gravitacional e torque. O atrito viscoso foi desconsiderado por simplicidade. É ilustrado na Figura 24 o diagrama de forças em uma roda em que,

- $F_i$ é a força resultante do torque
- $N_i$ é a força normal gravitacional
- $F_{li}$ é a força lateral relacionada ao atrito

Figura 24 – Diagrama de forças em uma roda




Fonte: Autoria própria

Definimos as forças como

$$F_i = \tau \times r, \tag{3.9}$$

$$F_{li} = sgn(v_{yi}) N_i \mu_l, \tag{3.10}$$

onde

- $sgn()$ é a função *signal*
- $\mu_l$ é o coeficiente de atrito estático lateral

As equações para as forças normais podem ser encontradas igualando-se os torques que as mesmas realizam tomando como referência o centro de massa do carro. Como o veículo é considerado simétrico com relação ao eixo $x$, temos que

$$N_1 = N_4 = \frac{1}{2}\frac{b}{a+b}mg \tag{3.11}$$

$$N_2 = N_3 = \frac{1}{2}\frac{a}{a+b}mg \tag{3.12}$$

O modelo dinâmico do AGV até agora descrito pode ser determinado por meio de uma equação que considera as forças envolvidas, de torque e de atrito, junto com o movimento resultante do corpo. Em outras palavras temos que o torque executado pelos motores acarreta no movimento do veículo somado às forças de atrito. Essa equação é descrita matematicamente a seguir

$$\mathbf{M}\ddot{q} + \mathbf{F} = \mathbf{B}\tau \tag{3.13}$$

onde

- $\mathbf{M}$ é a matriz diagonal de massa e momento de inércia,

$$\mathbf{M} = \begin{bmatrix} \mathrm{m} & 0 & 0 \\ 0 & \mathrm{m} & 0 \\ 0 & 0 & \mathrm{I} \end{bmatrix}$$

- $\mathbf{F}$ correponde às forças de atrito,

$$\mathbf{F} = \begin{bmatrix} 0 \\ F_s sen(\theta) + F_l cos(\theta) \\ M_r \end{bmatrix}, \tag{3.14}$$

$$F_l = \sum_i F_{li},$$

$$M_r = b[F_{l2} + F_{l3}] - a[F_{l1} + F_{l4}].$$

  As componentes $\mathbf{F}(1)$ e $\mathbf{F}(2)$ do vetor de forças corresponde à projeção da forças de atrito nos eixos $X_g$ e $Y_g$ respectivamente. Como não é considerado deslizamento longitudinal, apenas lateral, $\mathbf{F}(1) = 0$. A componente $\mathbf{F}(3)$ corresponde aos momentos realizados pelas forças de atrito com relação ao centro de massa do AVG.

- $\mathbf{B}$ corresponde à matriz de transmissão do torque realizado pelos motores, e $\tau$ ao vetor de torques,

$$\mathbf{B} = \frac{1}{r}\begin{bmatrix} cos(\theta) & cos(\theta) \\ sen(\theta) & sen(\theta) \\ -c & c \end{bmatrix} \tag{3.15}$$

$$\tau = [\tau_R, \tau_L]^T$$

É possível reescrever a equação do modelo dinâmico em termos de $\eta$. $\eta$ corresponde às velocidades lineares e angular do AGV com relação a ele mesmo, de forma que

$$\dot{q} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = \mathbf{S} \times \eta = \begin{bmatrix} cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{cx} \\ v_{cy} \\ \omega_c \end{bmatrix}, \tag{3.16}$$

em que $\mathbf{S}$ pode ser vista como a matriz de mudança de coordenadas de $\dot{q}$ para $\eta$. A equação do modelo dinâmico escrita em termos de $\eta$ fica

$$\mathbf{S}^T\mathbf{B}[\dot{\mathbf{S}\eta}] + \mathbf{S}^T\mathbf{F} = \mathbf{S}^T\mathbf{B}\tau \tag{3.17}$$

$$\mathbf{S}^T\mathbf{M}\mathbf{S}\dot{\eta} + \mathbf{S}^T\mathbf{M}\dot{S}\eta + S^T\mathbf{F} = \mathbf{S}^T\mathbf{B}\tau \tag{3.18}$$

em que, calculando-se o produto entre as matrizes, temos

$$\mathbf{S}^T\mathbf{M}\mathbf{S} = \mathbf{M} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix},$$

$$\mathbf{S}^T\mathbf{M}\dot{\mathbf{S}} = \begin{bmatrix} 0 & -m \times \dot{\theta} & 0 \\ m \times \dot{\theta} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{S}^T\mathbf{F} = \begin{bmatrix} 0 \\ F_l \\ M_r \end{bmatrix},$$

$$\mathbf{S}^T\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ \text{-c} & \text{c} \end{bmatrix}.$$

A equação (3.18) foi obtida substituindo $\dot{q}$ por $\mathbf{S} \times \eta$ como está presente em (3.16) e multiplicando $\mathbf{S}^T$ pela esquerda para que o resultado esteja em termos de $[v_{cx}, v_{cy}, w_c]^T$. Logo temos que a equação que carcteriza o sistema é dada por

$$\begin{bmatrix} m \times \dot{v_{cx}} \\ m \times \dot{v_{cy}} \\ I \times \dot{w}_c \end{bmatrix} + \begin{bmatrix} -m \times v_{cy} w_c \\ m \times v_{cx} w_c \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ F_l \\ M_r \end{bmatrix} = \begin{bmatrix} \frac{\tau_e + \tau_d}{r} \\ 0 \\ \frac{(\tau_d - \tau_e)c}{r} \end{bmatrix} \tag{3.19}$$

Utilizando a equação da segunda linha de (3.19) e substituindo $v_{cy}$ por $-x_{ICR}w_c$ como em (3.4) tem-se

$$\frac{d}{dt}[-x_{ICR}w_c] + v_{cx}w_c + \frac{F_l}{m} = 0$$

$$\dot{x_{ICR}}w_c = x_{ICR}\dot{w_c} - v_{cx}w_c - \frac{F_l}{m}$$

Assumindo que a velocidade angular é constante ($\dot{w_c} = 0$)

$$\dot{x_{ICR}} = -v_{cx} - \frac{F_l}{w_c m}$$

Pode-se assim estimar o valor de $x_{ICR}$. Em Pazderski e Kozlowski (2006) é explicitado que, para que não haja perda de estabilidade, $x_{ICR}$ deve estar contido no intervalo $-a \leq x_{ICR} \leq b$.

## 3.2 Simulador

Com base no modelo obtido, foi desenvolvido um simulador para a dinâmica longitudinal e lateral do AGV (Figura 25). O simulador foi desenvolvido utilizando a interface gráfica do Matlab. Ele inicia com os valores do AGV de massa, momento de inércia, dimensões, raio das rodas, posição do centro de massa e coeficiente de atrito com o chão utilizados por Pazderski et al. (2017) em sua simulação. Entretanto é possível para o usuário modificar qualquer um desses parâmetros antes de iniciar a simulação.

Um sistema de controle proporcional foi implementado para que o veículo não acelere continuamente durante o experimento. A variável de controle usada foi o torque das rodas. A velocidade e torques instantâneos são mostrados na interface. A equação de controle é descrita a seguir

$$T(t) = K \times e(t),$$
$$e(t) = v_{ref} - v_{cx}(t)$$

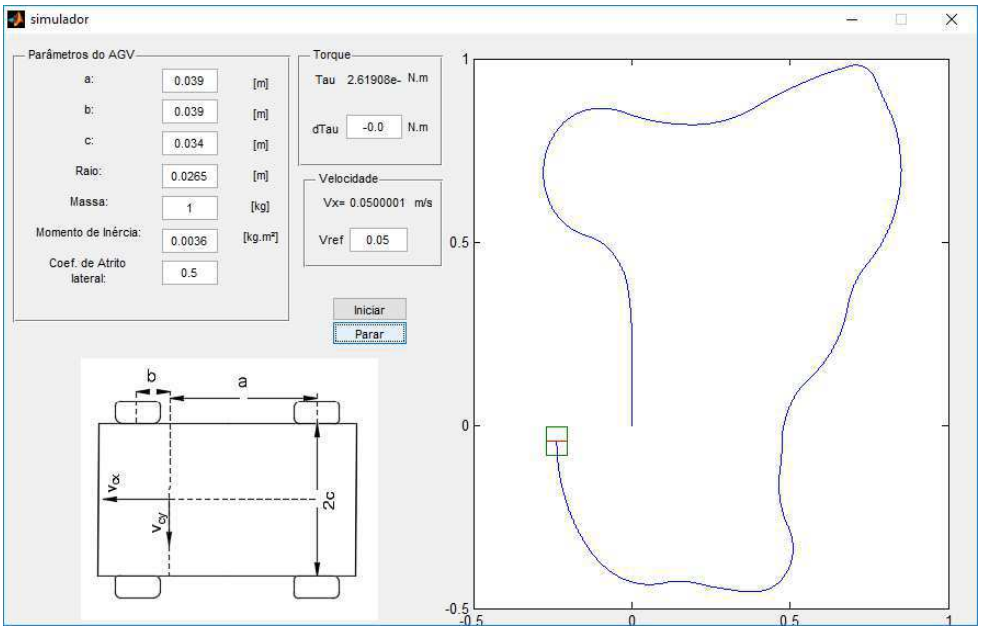
onde

- $T(t)$ é o torque das rodas
- $K$ é o ganho do controlador
- $v_{ref}$ é a velocidade de referência
- $v_{cx}(t)$ é a velocidade do veículo em seu eixo $x$
- $e(t)$ é o erro de velocidade

A velocidade de referência pode ser escolhida pelo usuário antes da simulação. O simulador também permite que o usuário escolha o torque diferencial das rodas. Dessa forma é possível observar o AGV realizando trajetórias curvas.

A interface possui dois botões, para iniciar e pausar a simulação. Ao selecionar o botão Iniciar, o simulador carrega todas informações das caixas de texto editáveis e inicia a simulação. A única caixa de texto editável cujo valor é atualizado constantemente é a caixa referente ao torque diferencial. Assim, pode-se variar o raio e o sentido das curvas realizadas durante a simulação.

Figura 25 – Simulador de AGV deslizante




Fonte: *print screen* do simulador

O gráfico mostrado à direita da janela do simulador (Figura 25) corresponde ao veículo e a sua trajetória percorrida. As curvas em azul são a trajetória e o retângulo verde é o AGV. A trajetória percorrida foi determinada arbitrariamente alterando-se o valor do torque diferencial.

A simulação se mostrou coerente com o que se espera da realidade. Quão maior, em módulo, é o torque diferencial, menor se torna o raio da curva realizada pelo veículo. Torques diferenciais positivos implicam curva realizada para a esquerda e torques diferanciais negativos implicam curva realizada para a direita.

## 3.3 Modelagem cinemática do erro

O sistema de controle de trajetória que se pretende fazer utiliza o modelo do AGV junto com o modelo do erro de trajetória dado pelas variáveis $\Gamma$ e $\theta$ (Figura 26). A modelagem do erro é feita por meio de equações que caracterizam o $\Gamma$ e o $\Theta$.

A modelagem do sistema é feita de forma que a velocidade linear do AGV permaneça constante por todo o percurso e a variável de controle é a variação da velocidade angular das rodas. Implicando nas seguintes equações

$$v = \frac{v_e + v_d}{2}$$
$$v_e = v - \Delta\omega r$$
$$v_d = v + \Delta\omega r$$
$$\Delta\omega = \frac{v_d - v_e}{2r}$$

onde

- $v$ é a velocidade linear do AGV;
- $v_e$ é a velocidade linear da roda esquerda;
- $v_d$ é a velocidade linear da roda direita;
- $\Delta\omega$ é a variação das velocidades angulares das rodas;
- $r$ é o raio das rodas.

Para cálculo de $\Gamma$ é possível observar na Figura 20 que $\Gamma$ pode ser determinado da seguinte forma

$$\Gamma = tan(\Theta)a$$

para simplificar o sistema tornando-o linear, supõe-se que para pequenos valores de $\Theta$ a seguinte aproximação é válida

$$tan(\Theta) \approx \Theta$$

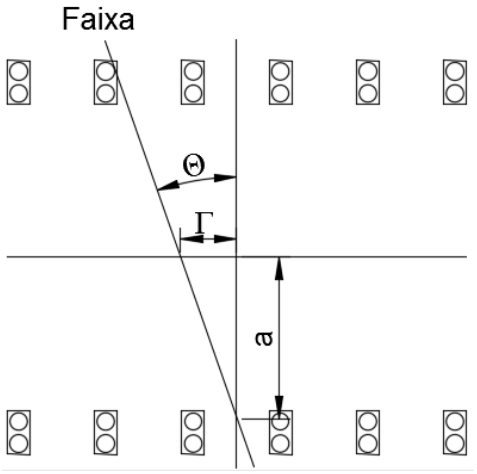
para o caso de a trajetória ser uma reta, ou seja, $\Theta$ é constante, temos que

$$\dot{\Gamma} = \Theta v$$

Para o cálculo de $\Theta$ é possível perceber que a derivada do mesmo corresponde a variação da velocidade linear das rodas dividida pela distância $D$ entre as rodas, de forma que temos

$$\dot{\Theta} = \frac{\Delta\omega r}{D}$$

Figura 26 – Variáveis de Estado




Fonte: Autoria própria

Dessa forma se obtém a seguinte equação de representação do sistema em variáveis de estado

$$\begin{bmatrix} \dot{\Gamma} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Gamma \\ \Theta \end{bmatrix} + \begin{bmatrix} 0 \\ r/D \end{bmatrix} \Delta\omega \tag{3.20}$$

# 4 Conclusões e Trabalhos Futuros

Foi possível concluir que o algoritmo neural é o melhor, dentre os estudados, para estimar o posicionamento da trajetória abaixo do AGV, apresentando baixos índices de erros e tornando-o provável de ser implementado no sistema de identificação de trajetória.

O simulador desenvolvido em Matlab a partir dos modelos cinemático e dinâmico do veículo apresentou o comportamento desejado, podendo ser usado para quaisquer simulações que envolvam a utilização de um AGV deslizante.

Devido à questões de tempo, não foi possível desenvolver a lei de controle de trajetória para o AGV. Entretanto, com o simulador construído e a modelagem do erro de trajetória feita, fica mais fácil o desenvolvimento da lei de controle e a simulação do sistema controlado.

Como proposta para trabalhos futuros é sugerido que se continue o estudo e a implementação do controle de trajetória, desenvolvendo-se a lei de controle com base no modelo construído do AGV e do erro de trajetória e aplicando o controlador de trajetória no simulador desenvolvido. Uma vez que o sistema de controle apresente resultado satisfatório no simulador, parte-se para a aplicação do sistema no modelo real do AGV.

# Referências

AUTOMATIC Guided Vehicle Manufacturers. 2017. <http://www.automaticguidedvehicles.com/>. Acessado em: 27-08-2017. Citado na página 14.

BALAJI, V. et al. Optimization of pid control for high speed line tracking robots. *2015 IEEE International Symposium on Robotics and Intelligent Sensors*, 2015. Citado na página 31.

GEOVANNY, A. B. *Um sistema Óptico de Reconheicmento de Trajetórias para Veículos Automáticos*. Dissertação (Mestrado) — Pós-Graduação em Engenharia Elétrica, Universidade Federal de Campina Grande, 1998. Citado 2 vezes nas páginas 29 e 30.

PAZDERSKI, D.; KOZLOWSKI, k. Practical Stabilization of a Skid-steering Mobile Robot: A kinematic-based approach. *Mechatronics, 2006 IEEE International Conference on*, 2006. Citado 2 vezes nas páginas 36 e 43.

PAZDERSKI, D. et al. Modeling and control of a 4-wheel skid-steering mobile robot: from theory to practice. 09 2017. Citado na página 43.

PAZDERSKI, D.; KOZLOWSKI, k.; W.E, D. *Tracking and Regulation Control of a Skid Steering Vehicle*. Gainesville, Florida: [s.n.], 2004. Citado na página 36.

SIMON, H. *Redes Neurais: Princípios e Prática*. [S.l.]: Bookman, 2008. ISBN 0132733501. Citado na página 16.