Universidade Federal de Minas Gerais
Escola de Engenharia
Departamento de Engenharia Eletrônica
Engenharia de Controle e Automação

Relatório 01 - Trabalho de Simulação

Alunos:

Hugo Ferreira Marques - 2018014573

Thiago Palhares Assis - 2018112036

Matheus Bento Araujo de Moraes - 2019021581

Professores:

Armando Alves Neto

Leonardo A. Mozelli

Belo Horizonte
2022

Sumário

Lista de Figuras

1 Questão 1

O objetivo central desta questão é determinar os parâmetros do modelo longitudinal de um veículo em um cenário de simulação de pista reta sem inclinação. Para a coleta de dados o veículo foi acelerado até uma velocidade v e então cessou o torque do motor de forma a manter seu deslocamento inercial até sua parada.

Cálculo de β

O parâmetro "Beta"no código de simulação é calculado com base nas medições de velocidade do veículo e tempo durante a desaceleração na simulação. Dessa forma, o código inicia um loop que coleta dados enquanto a simulação está em execução. Isso inclui a leitura da velocidade atual do veículo e o tempo atual em cada iteração do loop. Além disso, é colocado algumas condições iniciais para o código executar:

1. O volante do veículo é mantido alinhado com o veículo, o que faz com que o veículo se mova em linha reta. Isso é alcançado através da configuração car.setSteer(0.0364)
2. O código verifica se o tempo na simulação (car.t) for menor ou igual a 8 segundos, é então aplicado um torque no motor(car.u) causando um movimento acelerado à frente do veículo.
3. Se o tempo decorrido na simulação (car.t) é maior que 8 segundos, o veículo não recebe nenhum torque do motor, resultando em nenhuma componente de força à frente em seu deslocamento até a parada do veículo.

Assim, após o loop, os dados coletados, incluindo o tempo, a velocidade instantânea e a velocidade inicial, são salvos em um arquivo CSV.

Posteriormente o cálculo do Beta, é realizado mediante o código do Matlab desenvolvido, nele é criado o cálculo do Beta.

O cálculo do parâmetro β envolve a análise da relação entre a velocidade instantânea do veículo (representada por v_s e a velocidade inicial (representada por v_0 em um intervalo de tempo entre a máxima velocidade alcançada e a parada total do veículo. Esse intervalo de tempo começa após 8 segundos de simulação. O objetivo é entender como a velocidade do veículo varia durante esse período. Após realizar o tratamento dos dados para encontrar o β é criado um gráfico que representa a relação entre Δv e Δt normalizados para diferentes valores de β como referência. Isso permite estimar a que curva de referência, o β da simulação se aproxima. .

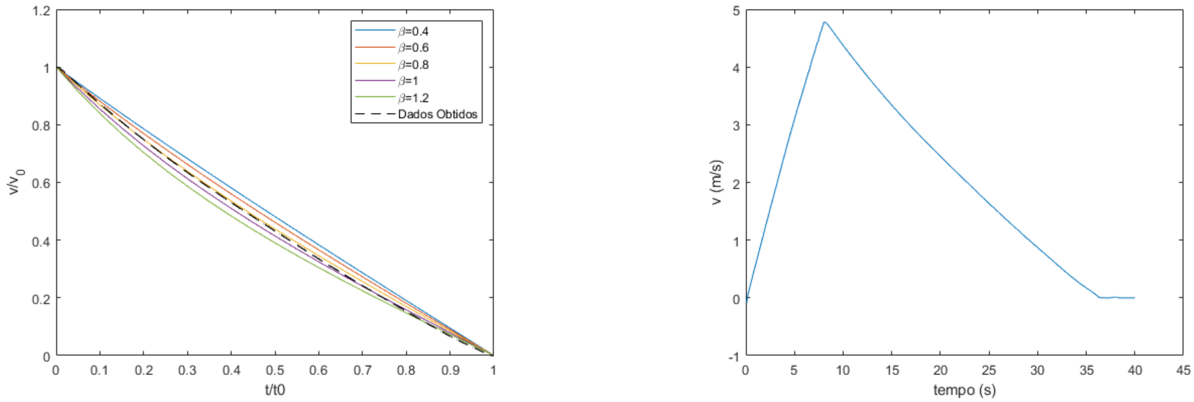

Figura 1: Dados coletados e relação v/v_0 e t/t_0

Percebemos que a aproximação da área frontal pela equação $A_f = 1.6 + 0.00056(m - 765)$ retornava uma área desproporcional às medidas do veículo. Desta forma, fizemos a aproximação da área por 81% da largura vezes a altura do veículo.

$$\begin{aligned} A_f &= 0.81 \times l \times a \\ A_f &= 0.81 \times (0.4) \times (0.2) \\ A_f &= 0.0648m^2 \end{aligned} \tag{1}$$

Para encontrar o coeficiente de arrasto e a resistência de rolagem do modelo longitudinal do veículo foi utilizado as seguintes fórmulas:

$$C_d = \frac{2m\beta \tan^{-1}(\beta)}{V_0 T_p A_f} \tag{2}$$

$$R_x = \frac{V_0 m \tan^{-1}(\beta)}{\beta T} \tag{3}$$

Por fim, os resultados encontrados para o modelo longitudinal do veículo foram:

1. $\beta = 0.8$;
2. $C_d = 2.0239$;
3. $R_x = 0.7908$;

Como não temos o raio da roda nem a razão de conversão da velocidade angular do motor para a velocidade angular das rodas (razão da transmissão) não calculamos a força de tração do veículo. Assim não foi possível criar um modelo caixa-branca.

2 Questão 2

Para essa questão, a partir do modelo em alto nível da dinâmica do carro, o mesmo possui a seguinte função de transferência:

$$A(s) = \frac{K}{s(\tau s + 1)} \tag{4}$$

Com isso, é possível observar que a própria planta já possui um integrador, portanto, ao calcular a resposta ao degrau dessa planta, o estado estacionário nunca é atingido e o seu valor só cresce.

Para estimar um modelo com integrador, foi-se utilizado um método descrito pelo artigo "Sobre a identificação de processos integradores com atraso e tempo-morto"do Luís Gustavo Soares. Nesse estudo, é possível obter os valores de τ a partir do cruzamento da reta tangencial no eixo x, ou seja, o tempo no qual o sistema começa a responder ao degrau aplicado. Além disso, é possível obter o valor de K, quando t=0 a partir da expressão $y = -k/(\tau)$. Assumindo que o sistema não possui tempo morto.

Assim, realizou-se um teste no simulador, em que o carrinho executava um degrau de 0.2 em seu torque por 5 segundos. Ao obter a velocidade do carro, foi necessário trasladar o ponto de início do degrau para os 0 segundos, obtendo o seguinte resultado:

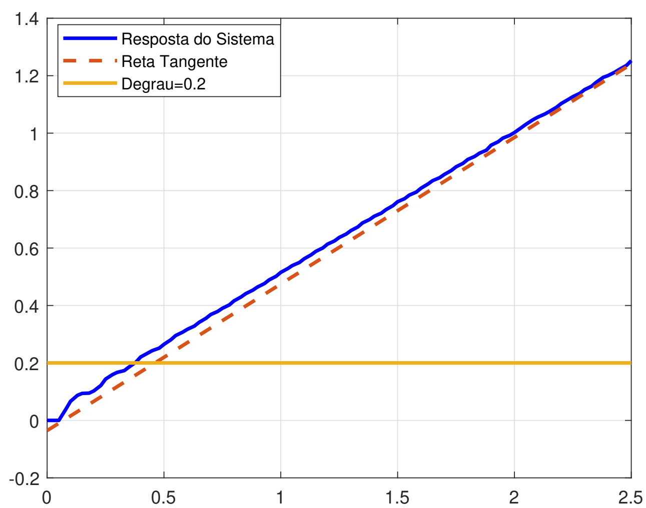

Figura 2: Resposta obtida pelo degrau aplicado e a tangente calculada

A partir disso, foi possível obter os parâmetros da planta, chegando à seguinte função de transferência:

$$A(s) = \frac{0.4946}{s(0.0389s + 1)} \tag{5}$$

Ao utilizar o método do lugar das raízes, obteve-se um controlador com os seguintes ganhos:

$$Kp = 28.8 \qquad Kd = 0.0 \qquad Ki = 16.7 \tag{6}$$

Mas devido às vibrações do modelo na simulação que geravam um bug nos sensores (que descobrimos só posteriormente), reduzimos gradativamente os ganhos até que não sofresse mais esse problema.

```
#Parâmetros do controlador
Vref = 1.0
Kp = 0.8
Ki = 0.6
Kd = 0.1
delta_t = 0.05
```

3 Questão 3

O controlador então foi testado em uma simulação na qual a velocidade de referência é igual a 1m/s sujeito à perturbação de uma rampa inclinado no trajeto.

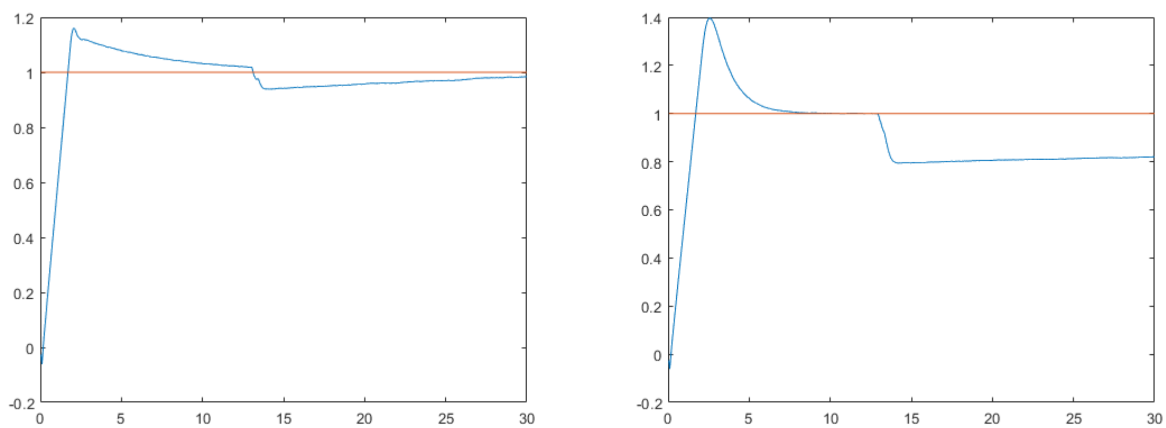

Figura 3: Controle PID e PI sujeitos a pertubação

Pelos dados do experimento vemos no controle PID um sobresinal de menos de 20% e ele tentando rejeitar a perturbação da rampa.

4 Questão 4

Para implementação do controle lateral optamos pelo uso da estratégia de controle Stanley dada por:

$$\delta(t) = \psi(t) + \arctan \frac{k \cdot e(t)}{v(t)} \tag{7}$$

onde:

- $\delta \rightarrow$ ação de guinada
- $\psi \rightarrow$ erro entre orientação do carro e orientação da trajetória
- $e \rightarrow$ erro lateral da posição do carro à posição da trajetória

Como nesta etapa do trabalho ainda estamos trabalhando variáveis de posicionamento e orientação fornecidas pela classe do modelo do carro, desenvolvemos o script deste controlador primeiro gerando a trajetória desejada que são vetores sequenciais (matriz) de pontos (x,y) e θ, que é a orientação da trajetória em cada ponto.

A partir dessa trajetória desenhada $\psi = \theta_{curva} - \theta_{carro}$.

```
##################################
## Cálculo PSI
#################################
        if  np.abs (car.th - ptheta[idx]) > np.pi: ##menor angulo
                psi = (car.th -ptheta[idx])
                if (car.th < ptheta[idx]):
                        psi = psi + 2*np.pi
                else:
                psi = psi - 2*np.pi
        else:
                psi = (car.th - ptheta[idx])
```

E para cálculo de e, realizamos o produto escalar do vetor lateral ao carro ($\theta_c arro + \pi/2$) com o vetor do carro até o ponto mais próximo da trajetória.

```
##################################
## Cálculo erro lateral
#################################
        vetor_lateral_carro = [ -np.cos(car.th + np.pi / 2),
                                -np.sin(car.th + np.pi / 2)]

        vetor_curv_carro = [px[idx]-car.p[0],
                                                py[idx]-car.p[1]]

        erro = np.dot(vetor_curv_carro, vetor_lateral_carro)
```

Posteriormente essas variáveis são aplicadas dentro do loop com o código do controle lateral:

```
###############################
###Lei de controle lateral
##################################
    acao_guinada = psi + np.arctan2(( k * erro ), car.v + min)
        if np.abs(acao_guinada) < delta_max:
                delta = acao_guinada
        else:
                if acao_guinada >= delta_max:#saturou positivamente
                        delta = delta_max
                else: # saturou negativamente acao_guinada <= -
                   delta_max:
                        delta = -delta_max
```

5 Questão 5

A fim de analisar o desempenho do controlador lateral em um veículo autônomo foi desenvolvido o código em questão que se destina a geração de trajetórias retas, circulares e em formato de 8. O objetivo é que durante a simulação o veículo receba informação de qual é o ponto da trajetória mais perto e sua orientação. A partir destes dados é possível calcular o ângulo ψ (menor diferença entre a orientação do carro e a orientação da trajetória) e também o erro lateral que é o produto escalar do vetor lateral ao carro e o vetor até o ponto de referência. Calculado esses erros, eles então são submetidos à lei de controle projetada de forma que assim ele possa tomar as ações corretivas de guinada necessárias.

Como as trajetórias são pontos sequenciais, a evolução é feita quando alcançado novo ponto mais perto da trajetória.

As figuras abaixo apresentam os resultados obtidos utilizando o controlador lateral implementado:

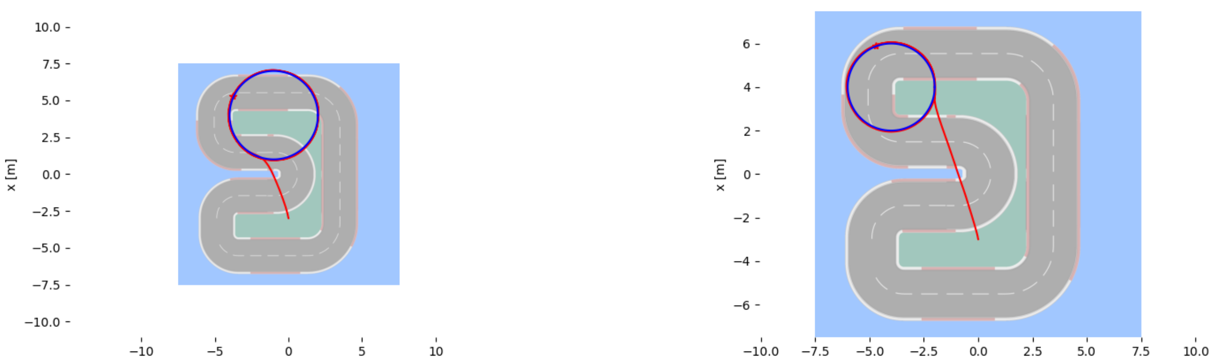

Figura 4: Simulação em trajetórias circulares

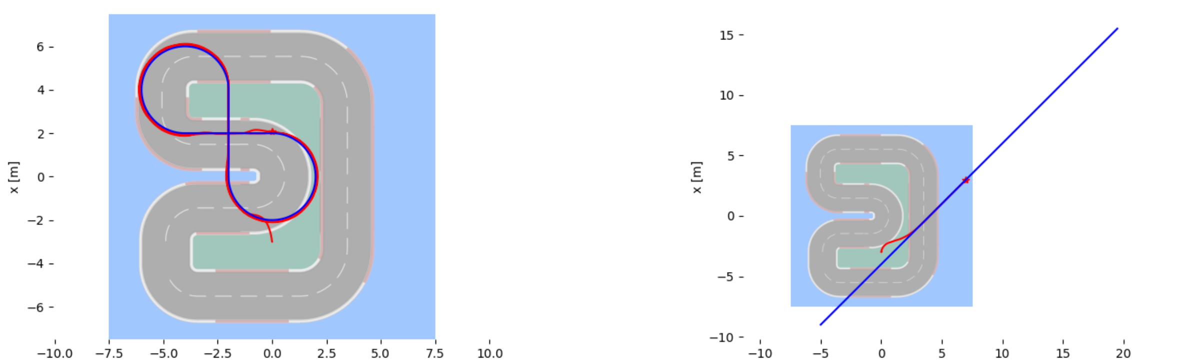

Figura 5: Simulação em trajetória 8 e em reta

6 Questão 6

Pra esta questão, na qual analisamos o desempenho do controlador longitudinal e o controlador lateral atuando simultaneamente é válido ressaltar que nos deparamos com um problema que se origina nos atributos da classe *car*. Quando há alguma inclinação da plataforma do modelo do carro durante a simulação por algum movimento mais acentuada, as variáveis car.v e car.th também variavam agressivamente de forma a tornar o controle instável. Acreditamos que isso ocorre porque o movimento oscilatório da plataforma é mais rápido que a velocidade de deslocamento do carro e como a frequência de amostragem é alta, gerava esse ruído na leitura de velocidade e psi. O veículo está deslocando para a frente e a velocidade é medida pela distância do ponto anterior até o ponto atual divido por dt. No entanto, a plataforma inclinou-se pra trás suficientemente para que a distância entre o ponto anterior e o ponto atual (medido a partir da plataforma) seja quase nula e consequentemente a velocidade medida é também nula, apesar do veículo ainda estar se deslocando a frente.

Apesar de o controle lateral sofrer também os efeitos desse problema, o controlador longitudinal é mais sensível a ele, pois com as oscilações bruscas de v, resultam em erros mais significativos, provocando variações mais acentuadas no torque aplicado ao motor, tornando-se praticamente um controlador ON-OFF instável.

Ainda assim foi possível aplicar os controladores e observar uma conversão nos resultados de forma satisfatória quando não nos deparávamos com esse problema.

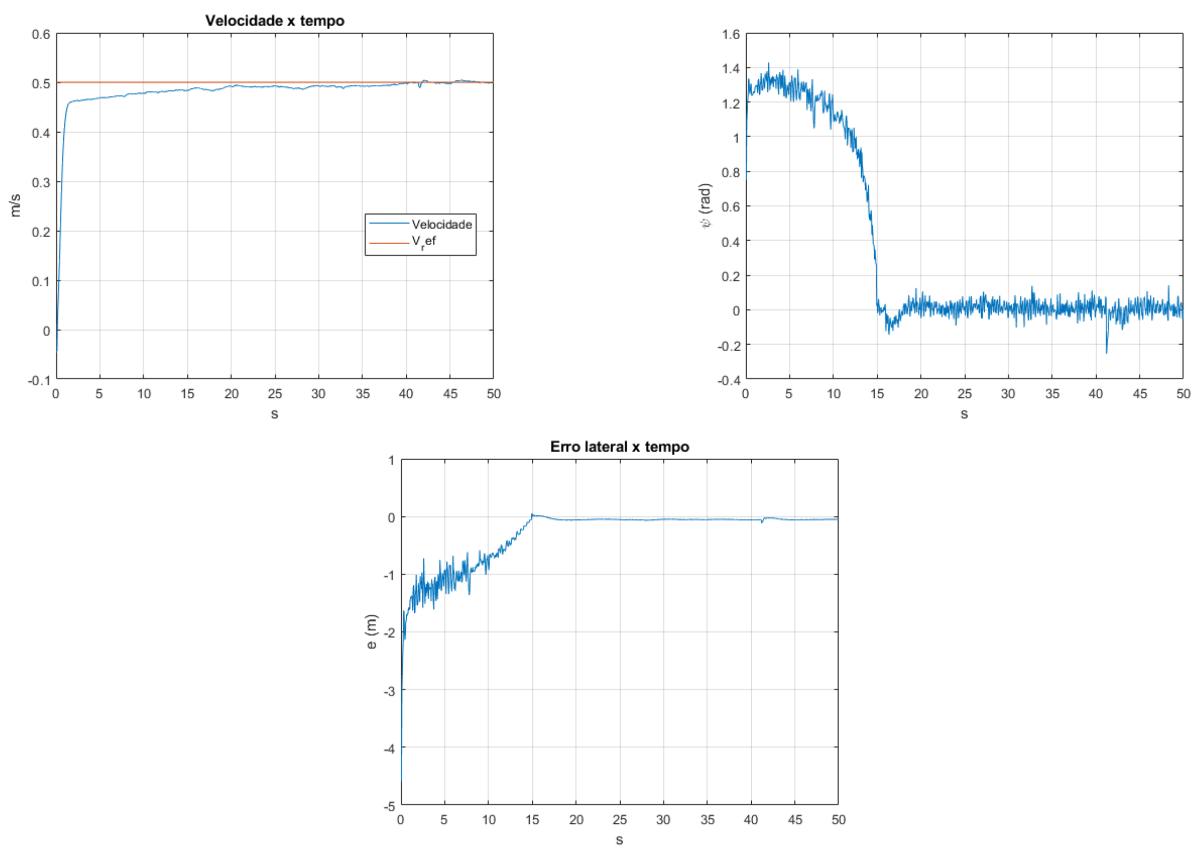

Figura 6: Dados coletados da simulação com atuação de ambos controladores

7 Questão 7

Essa questão tem como objetivo implementar um sistema de visão computacional capaz de medir a orientação do veículo com relação a pista.

Para implementar um sistema de controle para um carro autônomo, é essencial que o veículo seja capaz de identificar a linha de referência na pista, neste caso, uma linha tracejada localizada ao centro da pista. Dessa forma, a função getDashedLine(image) tem como objetivo realizar essa detecção.

A função recebe como entrada uma imagem, uma captura da câmera do carro. Inicialmente, a imagem é convertida para escala de cinza com a função cv2.cvtColor(image, cv2.COLOR.BGR2GRAY). Isso simplifica o processamento subsequente, tornando a imagem em tons de cinza. Em seguida, é aplicada uma técnica de limiarização com a função cv2.threshold() para destacar a linha tracejada, transformando os pixels mais claros em branco e os mais escuros em preto. Além disso, ocorre uma dilatação da imagem para conectar regiões interrompidas e inversão da imagem para obter a linha em branco sobre fundo preto. Em seguida é feita a detecção de contornos e filtragem dos contornos com base na área, mantendo apenas aqueles abaixo de um determinado tamanho.

Além disso, é utilizado a transformada de Hough para detecção de linhas tem como finalidade aplicação do detector de bordas de Canny para encontrar as bordas da linha tracejada. Posteriormente, é realizado o cálculo da inclinação (theta) e erro de posição (rho) da linha, assim é possível filtrar as linhas detectadas com base na inclinação para excluir as linhas não relevantes. Assim, é possível obter o desenho da linha tracejada detectada na imagem original com base nos valores calculados de theta e rho.

Após realizar o processamento das linhas tracejadas desejadas, é implementado um controlador de guinada lateral, onde é calculado o erro de posição e inclinação entre o carro e a linha.

O controlador Stanley desenvolvido calcula o ângulo de guinada(delta) com base nos erros coletados, e garante que o ângulo de guinada não ultrapasse o valor máximo definido. É importante destacar que é necessário um fator de correção em relação ao erro entre a posição do carrinho e da linha, uma vez que esse erro está definido em pixels e para o controle funcionar, este valor tem que estar em metros. Para isso, foi definido um valor de escala de 0.00392 para realizar essa conversão

8 Questão 8

Neste experimento, foram utilizados os controles Longitudinal e Stanley, mencionados nas questões 3 e 4, juntamente com a detecção de placas para determinar a velocidade de um carrinho. Para isso, adicionaram-se duas placas coloridas, de verde e vermelho, em posições específicas: uma em y = 5, indicando 1.0 m/s, e outra em y = 35, indicando 1.5 m/s. Para detectar essas placas, foi empregada uma função que filtrava as cores verde e vermelha na imagem, seguida pelo uso da função *approxPolyDP* para determinar o número de lados do contorno encontrado. Se fosse um octógono com área suficientemente grande, próximo ao carrinho, o programa interpretava a velocidade indicada.

Observou-se que o controle lateral era afetado significativamente pelo aumento de velocidade. Para solucionar esse problema, os ganhos do controle foram ajustados de acordo com a velocidade, garantindo uma resposta mais estável. Assim, definiu-se os ganhos em $K = 0.2$, $K = 0.8$ e $K = 1.0$ para as velocidades de $v = 0.5m/s$, $V = 1.0m/s$ e $v = 1.5m/s$

Como resultado, fez-se uma simulação entre 0 a 65 segundos, obteve-se os seguintes gráficos, de velocidade e posição lateral:

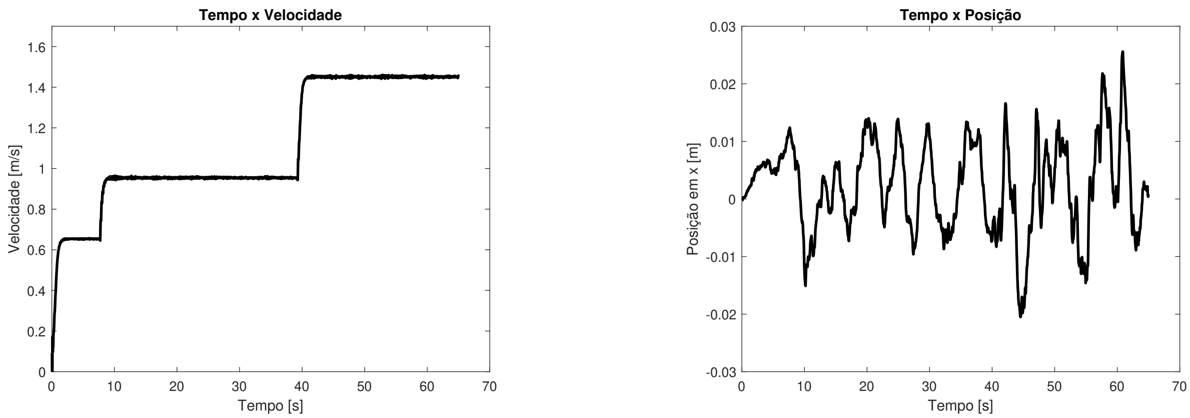

Figura 7: Resultados obtidos da questão 8

As imagens abaixo mostram também o momento de detecção das placas, em que elas são contornadas na cor ciano:

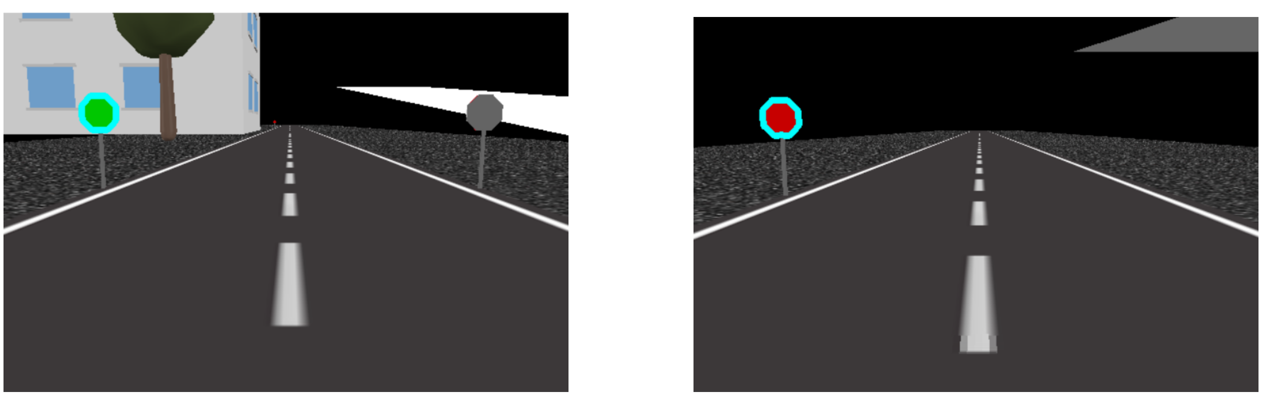

Figura 8: Detecção das placas em verde e vermelho

Como análise, observa-se que a placa foi detectada com precisão e o controle longitudinal conseguiu manter a velocidade e mudá-la rapidamente, sem nenhum overshoot. Para o controle lateral, apesar de com velocidades maiores o carro oscila mais, como esperado, essa oscilação máxima foi de 3 centímetros, o que é bastante baixo, fazendo com que o carro se mantivesse na faixa central.

9 Questão 9

A questão 9 basicamente é uma junção do controle longitudinal da Questão 2, da orientação em relação a pista e controle lateral utilizando visão computacional da Questão 7 e da detecção de placas e definição da velocidade da Questão 8.

Porém, o melhor valor encontrado para o ganho do controlador Stanley foi $K = 0.1$. Este valor possivelmente baixo se dá pelo fator de conversão, observado na Questão 8, que pode estar errado, uma vez que este fator é multiplicado pelo erro.

De forma semelhante à questão 8, fez-se uma simulação entre 0 a 65 segundos, obteve-se os seguintes gráficos, de velocidade e posição lateral:

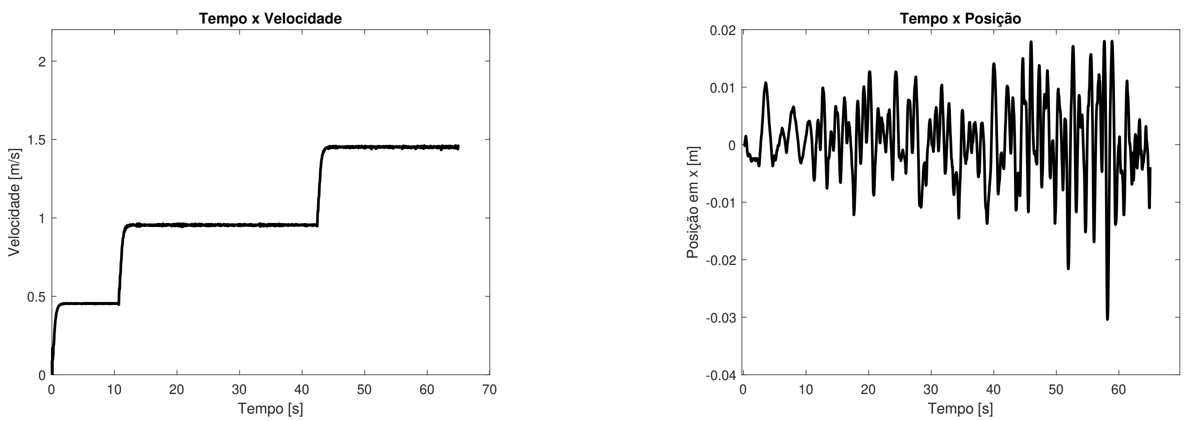

Figura 9: Resultados obtidos da questão 9, para K=0.1

De forma análoga a Questão 8, o carrinho conseguiu se manter com precisão tanto em relação a velocidade quanto em relação à sua posição lateral. Observa-se que para a posição lateral, houve maior oscilação e isso possivelmente se dá pela visão computacional, que em alguns momentos calculava a linha de referência errada, porém se ajustava rapidamente. Estes picos de erros faziam com que o carrinho virasse de maneira errada, porém não prejudicou o controle de forma geral, uma vez que o pico máximo de erro foi de 0.03 m.

10 Questão 10

Para a questão 10, foi necessário fazer algumas modificações na simulação:

1. Mover o carrinho para uma posição central, de modo que ele começasse no centro traçado da pista;
2. Mover a câmera de forma que ela filmasse mais diretamente voltado para o chão.

Com essas alterações, inicialmente, foi desenvolvido um algoritmo para detectar a faixa central. A detecção apenas no espaço de cores HSV não foi suficiente, uma vez que a linha lateral da pista e a linha central tinham as mesmas cores. Para resolver esse problema, foi concebida uma solução que envolvia a detecção dos contornos dessas linhas. Em seguida, aplicava-se uma operação de erosão para expandir esses contornos e, posteriormente, calculava-se a média no espaço de cores HSV desses contornos expandidos. Como as linhas tracejadas desejadas apresentavam uma área em seu entorno mais escura (referente a estrada), foi necessário aplicar um filtro para selecionar apenas os contornos que possuíam a média desejada após a expansão.

Ao obter somente os contornos desejados, foi necessário fazer um novo tratamento em que de todas as linhas obtidas através da função *HoughLinesP*, fosse retornada apenas aquela que era mais próxima do carrinho, ou seja, do ponto inferior central. Após esse procedimento, tornou-se possível aplicar a mesma estratégia mencionada na Questão 8.

Definindo a constante de controle longitudinal como $K = 1.6$, obteve-se o seguinte resultado:

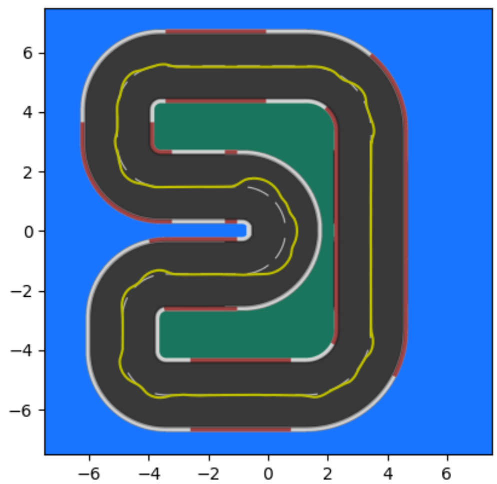

Figura 10: Trajetória realizada pelo carrinho, em amarelo

Observa-se que o controle da posição do carrinho foi mantido ao longo de todo o percurso, permitindo que o carro completasse uma volta inteira. Esse resultado indica o sucesso na abordagem proposta pela questão.

11 Códigos

A UFMG possui uma restrição de tamanho para envio de 20mb. Como os códigos, incluindo os modelos da pista ultrapassam esse limite, todo o código realizado pode ser encontrado no Github através do link: https://github.com/HugoFM2/FundVeiculosAutonomos

Referências

[1] CASTRUCCI, Plinio, Anselmo Bitar., Controle Automático,5.ed. São Paulo: Pearson Prentice Hall, 2007.

[2]SAKAI,Atsushi .PythonRobotics PathTracking Disponível em : https://github.com/AtsushiSakai/PythonRobotics/tree/master/PathTracking/stanley_controller Acesso em out/2023.

[3] LONGHI, Luis. Sobre Identificação de Processsos Integragradores com Atraso e TEmpo Morto. Controle & Instrumentação, vol.183, jan., 2012.

[4]SNIDER, Jarrod M. et al. Automatic steering methods for autonomous automobile path tracking. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08, 2009.