



**UNIVERSIDADE FEDERAL DA BAHIA  
ENGG64 – VISÃO COMPUTACIONAL**

**RUAN OLIVEIRA  
WALISSON SANTOS OLIVEIRA**

**RELATÓRIO: PROJETO DE ESTIMAÇÃO DA POSE DE UM ROBÔ MÓVEL**

**SALVADOR - BA  
2025**

**RUAN OLIVEIRA  
WALISSON SANTOS OLIVEIRA**

**RELATÓRIO: PROJETO DE ESTIMAÇÃO DA POSE DE UM ROBÔ MÓVEL**

Relatório apresentada ao componente curricular ENGG64 – Visão Computacional como requisito parcial para a aprovação no curso de Engenharia de Computação da Universidade Federal da Bahia.

Professor: André Gustavo Scolari Conceição

**SALVADOR - BA  
2025**

# RELATÓRIO

## I. FLUXOGRAMA DO PROJETO:

Conforme a figura 1, o projeto de estimação da pose de um robô seguiu a seguinte estrutura:

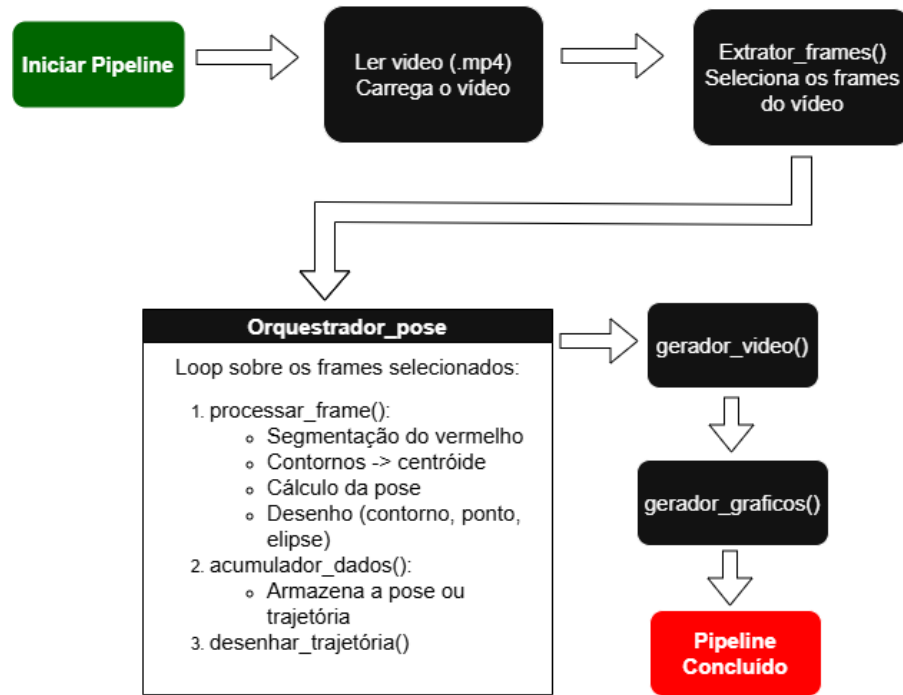


Figura 1: Pipeline de desenvolvimento do projeto

- O pipeline é iniciado. O primeiro passo é fazer a leitura do vídeo a partir da função `ler_video()`, que retornará o vídeo carregado.
- Após a leitura do vídeo, é realizado a extração de alguns frames para facilitar o trabalho. A função `extrator_frames()` é a encarregada de fazer esse processo. Primeiro ela obtém o total de frames do vídeo. Desse total, ela pega 20 índices igualmente espaçados. Logo depois, itera sobre esses índices definindo a posição do frame atual a partir dos índices. Cada frame correspondente é colocado em uma lista chamada “frames\_selecionados”. A função `extrator_frames()` retorna essa lista.
- Essa lista de frames é passada para a função `orquestrador_pose()`. Basicamente, essa função orquestra o processamento de cada frame para extrair a trajetória do robô:
  - Realiza um loop sobre os frames da lista.
  - Para cada frame, chama a função `processar_frame()`, que tem o objetivo de processar um único frame. O processamento do frame é feito da seguinte forma:
    - O frame é convertido para o padrão float. Logo depois, é encontrado as coordenadas *r* e *g* de cromaticidade, onde *r* e *g* representam a proporção da componente vermelha e verde na imagem.
    - Feito isso, constrói-se uma máscara a partir da segmentação do vermelho. Os limiares foram sendo testados gradualmente, de modo a ter o melhor resultado.

3. Encontrou-se, então, os contornos máscara, selecionando aquele que possui maior área.
  4. Ademais, foram calculados os momentos do contorno, centroide, matriz de inercia, autovalores, autovetores e orientação.
  5. No entanto, é preciso sair da imagem e ir para o mundo. Para obter os valores do centroide no mundo, a função `pixel_para_mundo()` se encarregou de fazer esse processo. Ela pega a matriz de homografia da câmera e calcula sua inversa. Assim, consegue multiplicar essa matriz inversa pela matriz coluna do ponto na imagem. O resultado após normalização é uma matriz com o valor do ponto da imagem no mundo real.
  6. É feito um dicionário “dados\_robo” onde reúne todos os dados correspondentes as coordenadas  $u$  e  $v$  da imagem,  $x_{real}$  e  $y_{real}$  do mundo e a orientação.
  7. Por consequente, é plotado também o centroide na imagem, o contorno, texto da posição e a elipse do contorno.
  8. A função `processar_frame()` retorna o dicionário `dados_robo` e o `frame_out`, que é uma cópia do frame que recebeu como entrada, porém com todos os plots do item 7.
- iii. Para cada frame processado, os dados são acumulados em uma nova estrutura chamada “pose”, que um dicionário. Quem realiza esse processo é uma função chamada `acumulador_dados()`.
  - iv. Para cada frame processado, é desenhado a trajetória, por meio da função `desenhar_trajetoria()`. Ela recebe o frame atual processado e duas listas, que correspondem as coordenadas  $u$  e  $v$  na imagem. Essas listas são convertidas em uma matriz de pontos e depois essa matriz é passada para o método “polylines”, que desenha a linha da trajetória na imagem processada.
  - v. Todos os frames processados são colocados dentro de uma lista chamada de “frames\_final”
  - vi. É feito um plot de cada frame, de modo a permitir o rastreamento em tempo real.
  - vii. Em seguida, a função `gerador_video()` se encarrega de fazer um vídeo de todos os frames processados. Recebe como parâmetro a lista “frames\_final” e as dimensões de largura e altura necessárias para fazer o vídeo.
  - viii. Por final, a função `gerador_graficos()` gera os gráficos da trajetória do robô e do seu ângulo ao longo do frames
- d. Pipeline é concluído.

## II. ANÁLISE DOS RESULTADOS:

Os resultados gerados foram: dois vídeos com o plot da trajetória, contorno e elipse da região de interesse; duas imagens do rastreamento em tempo real; e quatro gráficos correspondentes a posição e orientação do robô.

Nas figuras 2 e 3 é observado as imagens de rastreamento em tempo real. Cada vértice da trajetória (linha na cor ciano) corresponde a um frame processado. É possível visualizar também o centroide, caracterizado como um pontinho vermelho dentro da região de interesse; o contorno da região de interesse, em verde; e a elipse, na cor azul.



Figura 2: Imagem da trajetória do robô em tempo real (video1)



Figura 3: Imagem da trajetória do robô em tempo real (video2)

Seguindo a análise, as figuras 4 e 5 mostram a trajetória através de um gráfico. Os valores são apresentados em milímetros. Observe na figura 4, correspondente a trajetória do robô no vídeo 1, que o robô avança a sua posição no eixo x, logo depois avança a sua posição no eixo y, por conseguinte reduz a posição no eixo x, reduz a posição no eixo y e volta a avançar a sua posição no eixo x, de modo a fazer a sua trajetória semelhante a um espiral. Por outro lado, na figura 5, que mostra a trajetória do robô no vídeo 2, a forma do caminho percorrido é semelhante a uma circunferência.

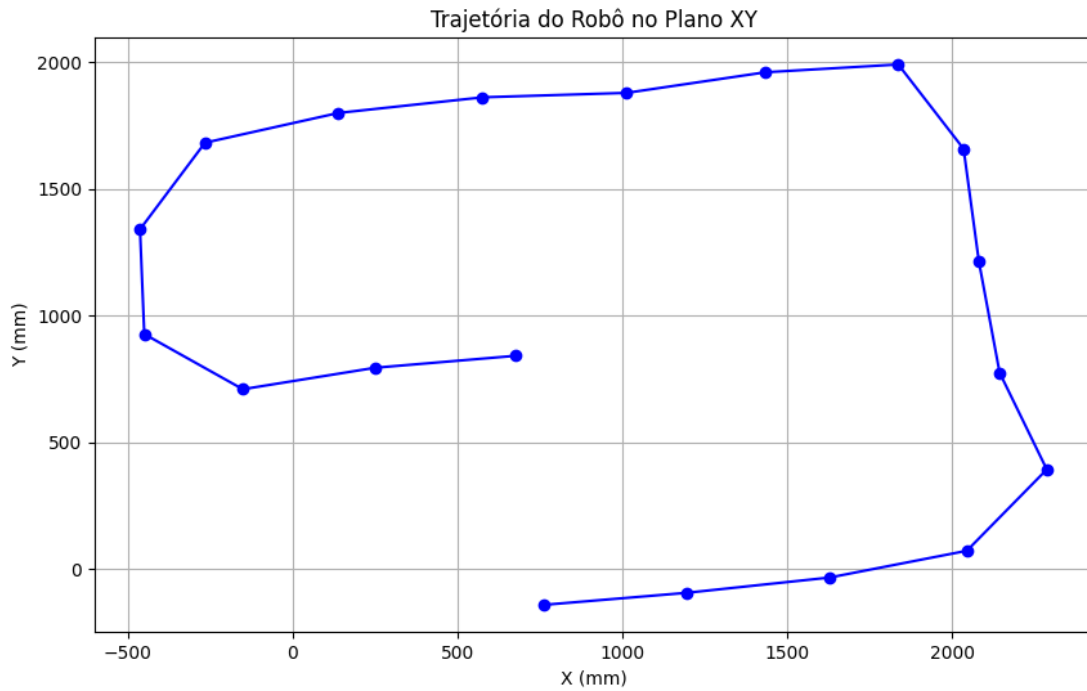


Figura 4: Gráfico da trajetória do robô (video1)

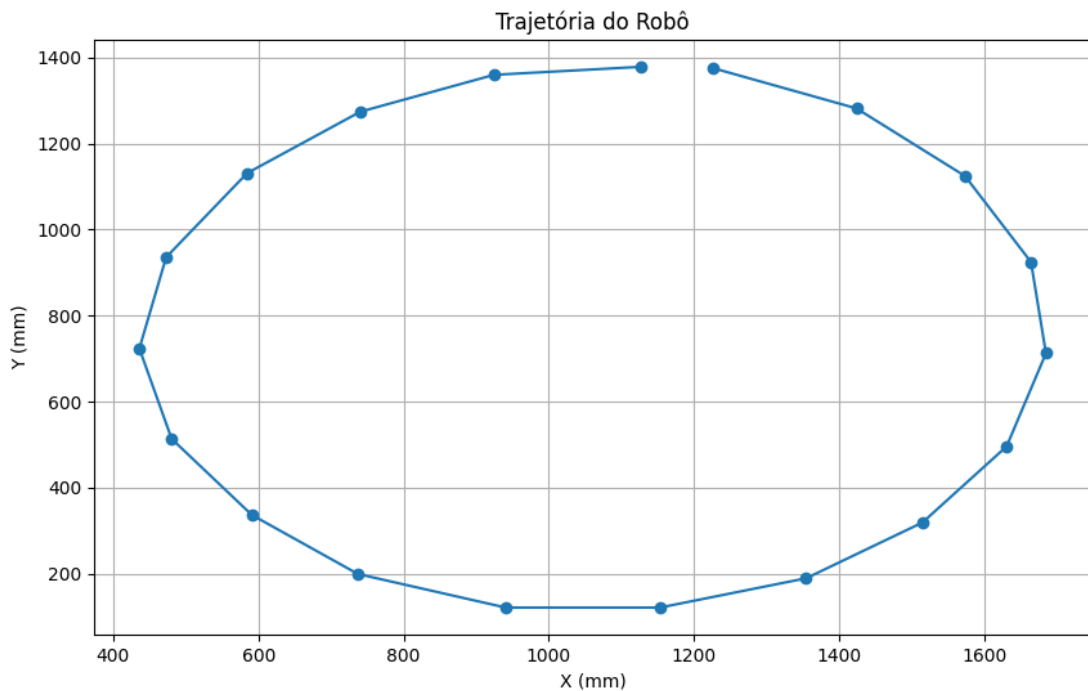


Figura 5: Gráfico da trajetória do robô (video2)

Por conseguinte, as figuras 6 e 7, elucidam o comportamento da orientação do robô ao longo dos frames processados. Observe que o comportamento do gráfico dos gráficos há uma estrutura um pouco repetitiva. Na figura 7, o robô começa alinhado em  $0^\circ$  e sobe progressivamente até mais ou menos  $160^\circ$ . Logo depois, tem uma queda brusca para mais ou menos  $-170^\circ$  e volta a subir novamente para ao redor de  $160^\circ$ , repetindo o ciclo. No entanto, na figura 6, a orientação se comporta de uma forma um pouco diferente. É possível perceber que o robô se mantém alinhado às proximidades de

0° por alguns frames e depois apresenta uma queda para o entorno de -160° e novamente uma subida para 170°. Por conseguinte, há uma queda gradual da orientação e novamente se mantém em torno de 0°. Esse ciclo tende a se repetir no restante da trajetória.

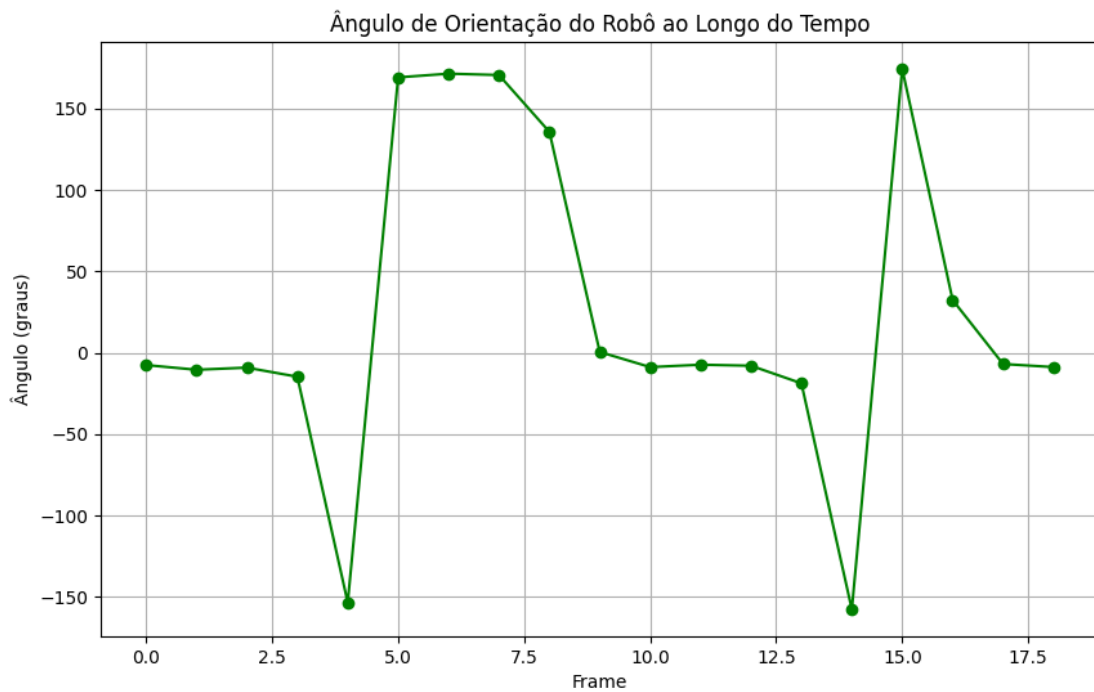


Figura 6: Gráfico da orientação do robô (video1)

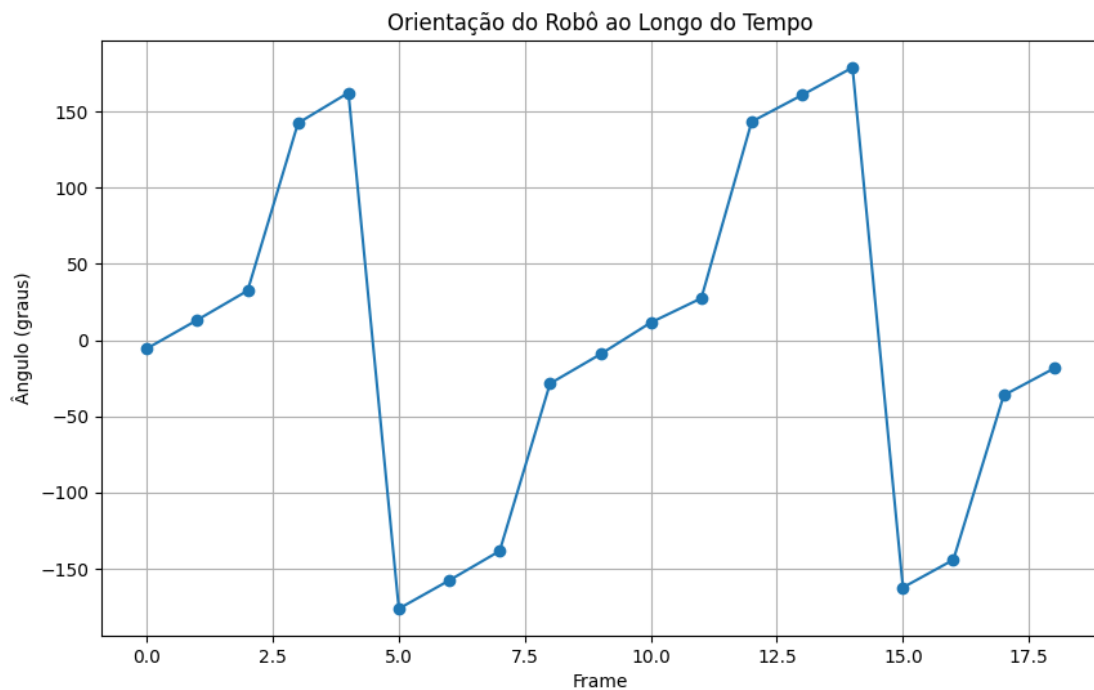


Figura 7: Gráfico da orientação do robô (video2)