Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Aeroespacial

Desenvolvimento de Ferramenta Utilizando Aprendizagem de Máquina para o Estudo de Combustível Sólido em Motor de Foguete

Autor: Gabriel Metre Resende
Orientador: Prof. Dr. Artur De Morais Elias Bertoldi

Brasília, DF
2024

Gabriel Metre Resende

Desenvolvimento de Ferramenta Utilizando Aprendizagem de Máquina para o Estudo de Combustível Sólido em Motor de Foguete

Monografia submetida ao curso de graduação em Engenharia Aeroespacial da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Aeroespacial.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Artur De Morais Elias Bertoldi

Brasília, DF

2024

Gabriel Metre Resende

Desenvolvimento de Ferramenta Utilizando Aprendizagem de Máquina para o Estudo de Combustível Sólido em Motor de Foguete

Monografia submetida ao curso de graduação em Engenharia Aeroespacial da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia Aeroespacial.

Trabalho aprovado. Brasília, DF, 23 de setembro de 2024 – Data da aprovação do trabalho:

Prof. Dr. Artur De Morais Elias Bertoldi
Orientador

Prof. Dr. Felipe Chagas Storti
Convidado 1

Prof. Dr. Olexiy Shynkarenko
Convidado 2

Brasília, DF
2024

Resumo

Este trabalho aborda a estimativa da taxa de regressão do combustível sólido em motores de foguete híbrido, utilizando abordagens clássicas e modelos de IA pré-treinados. O foco principal é segmentar a região do combustível sólido e prever a taxa de regressão com base em imagens da câmara de combustão de um queimador plano. A metodologia utiliza um conjunto de dados experimentais de testes em um motor de foguete híbrido. Os resultados indicam que o modelo empregado oferece previsões promissoras para a segmentação do combustível sólido, mesmo sem treinamento específico para essa finalidade. As conclusões têm implicações significativas para o projeto e otimização de motores de foguete híbridos, capacitando engenheiros a tomar decisões mais fundamentadas sobre a composição do combustível. A etapa de treinamento personalizado e refinamentos superou com sucesso o desafio de segmentar o combustível sólido em meio às complexas dinâmicas da combustão, resultando em um modelo capaz de distinguir o combustível sólido com alta confiabilidade. A aplicação de técnicas avançadas de visão computacional e modelos de inteligência artificial na análise da combustão de propelentes sólidos é fundamental para obter a taxa de regressão por meio de um banco de dados existente. A criação de uma ferramenta para extrair a taxa de regressão de vídeos de alta velocidade melhora o processo de obtenção da taxa de regressão, contribuindo para o desenvolvimento efetivo desses combustíveis e avançando nessa área específica.

Palavras-chave: foguete híbrido, taxa de regressão, queimador plano, visão computacional, combustível sólido, segmentação de imagem.

Abstract

This work addresses the estimation of solid fuel regression rates in hybrid rocket engines using both classical approaches and state-of-the-art pre-trained AI models. The primary focus is on segmenting the solid fuel region and predicting the regression rate based on images from the combustion chamber of a flat slab burner. The methodology utilizes an experimental dataset from tests on a hybrid rocket engine. Results indicate that the employed model provides promising predictions for solid fuel segmentation, even without specific training for this purpose. The findings have significant implications for the design and optimization of hybrid rocket engines, empowering engineers to make more informed decisions about fuel composition. The phase of customized training and refinements successfully overcame the challenge of segmenting solid fuel amidst the complex dynamics of combustion, resulting in a model capable of distinguishing solid fuel with high reliability. The application of advanced computer vision techniques and pre-trained AI models in the analysis of solid propellant combustion is crucial for obtaining the regression rate through an existing database. The development of a tool to extract the regression rate from high-speed videos enhances the regression rate acquisition process, effectively contributing to the advancement of solid fuel development in this specific field.

Keywords: hybrid rocket, regression rate, slab-burner, computer vision, solid fuel, image segmentation.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AP	Precisão Média (*Average Precision*)
CAD	Desenho Assistido por Computador (*Computer-Aided Design*)
CV	Visão Computacional (*Computer Vision*)
CV2	OpenCV (Biblioteca de Visão Computacional de Código Aberto)
CVPR	Conferência de Visão por Computador e Reconhecimento de Padrões (*Conference on Computer Vision and Pattern Recognition*)
DAQ	Aquisição de Dados (*Data Acquisition*)
HRE	Motor de Foguete Híbrido (*Hybrid Rocket Engine*)
IA	Inteligência Artificial (*Artificial Intelligence*)
ML	Aprendizagem de Máquina (*Machine Learning*)
MgB2	Diboreto de Magnésio
MOUETTE	Moteur OptiqUe pour ÉTudier et Tester Ergols hybrides (Motor Óptico para Estudo e Teste de Combustíveis Híbridos)
NI	National Instruments

Lista de símbolos

A_f, A_i	Área final e inicial da seção transversal
A_{pf}	Área final da porta de combustão
A_{pi}	Área inicial da porta de combustão
A_{port}	Área da porta de combustão
A_t	Área da garganta
a, n, m	Constantes de regressão
c_D	Coeficiente de descarga
c_{*exp}	Velocidade característica de expansão
D	Diâmetro da porta de combustão
G	Taxa de fluxo de massa do oxidante
G_{ox}	Taxa de fluxo de massa do oxidante por unidade de área
H_i, H_f	Alturas da porta de combustão nas posições inicial e final
L_f	Comprimento
O/F	Razão oxidante-combustível
P_c	Pressão média da câmara de combustão
W	Comprimento axial da porta de combustão
W_f	Largura da porta de combustão
X	Distância axial a partir da porta
$\dot{m}$	Taxa de fluxo de massa
$\dot{m}_{ox}$	Taxa de fluxo de massa do oxidante
$\dot{m}_f$	Taxa de fluxo de massa do combustível sólido
$\dot{r}$	Taxa de regressão do combustível sólido
$\dot{r}_s$	Taxa de regressão média

Δm_f	Variação de massa do combustível sólido
η^*	Eficiência de combustão
γ	Razão específica de calor
ρ_0	Massa específica do oxigênio
t_b	Tempo nominal do teste
r_s	Taxa de regressão

Sumário

1 INTRODUÇÃO

1.1 Contextualização e Motivação

O crescente interesse na pesquisa de foguetes com propelente híbrido, especialmente após problemas com motores de propelente sólido no programa espacial americano, tem estimulado estudos mais amplos sobre a aplicação dessa tecnologia em sistemas espaciais. Dois eventos importantes na propulsão híbrida dão suporte a essas expectativas: (i) o sucesso do voo da nave SpaceshipOne, cujo sistema propulsor do segundo estágio é totalmente baseado em tecnologia híbrida, e (ii) a descoberta da parafina como um combustível eficiente em motores de foguetes com propelente híbrido (KARABEYOGLU et al., 2005). Nesses motores, é comum injetar oxidante líquido na câmara de combustão, seguido por vaporização e mistura com o combustível sólido.

Os métodos utilizados no presente trabalho envolvem o emprego de técnicas avançadas de visão computacional, utilizando modelos de inteligência artificial previamente treinados. A biblioteca *Open Source Computer Vision Library*, conhecida por suas amplas funcionalidades de processamento e análise de imagens, foi utilizada com intuito de extrair características relevantes das imagens capturadas durante os testes de combustão. Como uma abordagem mais sofisticada, dois modelos pré-treinados foram utilizados, o Grounding DIno e Segment Anything Model, modelos estes que estão no estado da arte atual da detecção, anotação e segmentação de imagens.

Este trabalho utilizou o banco de dados de pesquisas realizadas pelo Departamento de Aero-Termo-Mecânica (ATM) da Universidade Livre de Bruxelas em colaboração com o Laboratório de Propulsão Química (CPL) da Universidade de Brasília no contexto do projeto *Experimental characterisation with firing tests of regression rate in hybrid rocket engines using paraffin fuel doped with metallic particles* desenvolvido pelos professores doutores Artur Elias M. Bertoldi (CPL/UnB) e Patrick Hendrick (ATM/ULB).

A importância deste trabalho reside na aplicação de técnicas avançadas de visão computacional e modelos de inteligência artificial para a análise da combustão de propelentes sólidos em motores de foguete híbrido, com foco na estimativa da taxa de regressão utilizando o banco de dados fornecido. Essa taxa é um parâmetro fundamental para o estudo da queima do combustível, pois influencia diretamente o desempenho e a eficiência do motor. A partir da criação de uma ferramenta para a obtenção da taxa de regressão por meio da análise de vídeos capturados por uma câmera de alta velocidade, é possível aprimorar o processo de determinação desse parâmetro e, consequentemente, contribuir para o desenvolvimento de combustíveis sólidos mais eficientes.

1.2 Objetivos

1.2.1 Objetivo Geral

Este trabalho tem como objetivo desenvolver uma ferramenta computacional para a análise de imagem de amostras de combustível sólido em um queimador plano, a partir da aplicação de técnicas de visão computacional utilizando inteligências artificiais previamente treinadas.

1.2.2 Objetivo Específicos

Este estudo visa atingir objetivos específicos que direcionam a análise dos vídeos de testes de motores de foguete híbrido:

1. **Avaliação das Abordagens Clássicas:** Utilizar técnicas tradicionais de visão computacional para analisar os vídeos, buscando obter informações detalhadas sobre o comportamento do combustível sólido e da chama.
2. **Análise Avançada com Modelos de IA Pré-Treinados:** Empregar modelos de inteligência artificial pré-treinados, considerados no estado da arte, especializados em processamento de imagens.
3. **Treinamento Personalizado:** Realizar treinamento específico para os modelos, visando o reconhecimento e a distinção precisa entre o combustível sólido e a chama nos vídeos dos testes.
4. **Obtenção da Taxa de Regressão:** Extrair quantitativamente a taxa de regressão a partir da análise dos vídeos dos testes selecionados.
5. **Criação de uma interface de análise:** Criação de uma ferramenta de fácil uso para processamento de videos de testes, gerando a taxa de regressão a partir das imagens.

2 REFERENCIAL TEÓRICO

2.0.1 Foguetes Híbridos

A propulsão híbrida tem despertado interesse na indústria de foguetes comerciais e em aplicações espaciais devido a diversas vantagens. Esses sistemas combinam elementos dos motores sólidos e líquidos, sendo comumente configurados com um oxidante armazenado em fase líquida ou vapor, e um grão combustível sólido na câmara de empuxo.A figura 1 apresenta a estrutura interna de um foguete de motor de combustão híbrida.

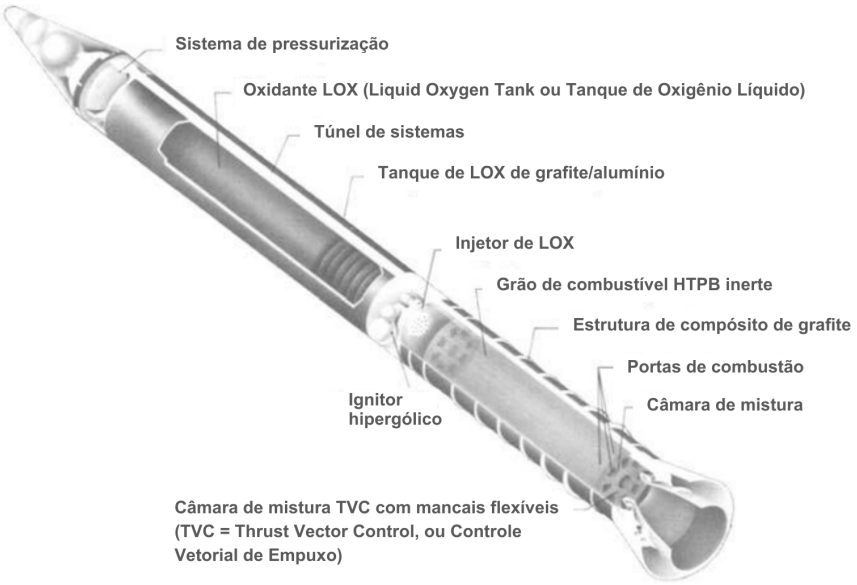


Figura 1 – Estrutura Interna de um foguete de combustão híbrida (SUTTON; BIBLARZ, 2010)

O funcionamento do sistema envolve a utilização de combustível e oxidante em fases diferentes, como, por exemplo, óxido nitroso como oxidante líquido e parafina como combustível sólido. O empuxo é gerado pela ejeção dos gases resultantes da combustão. Em motores híbridos, o combustível sólido é frequentemente alocado na câmara de combustão em forma de cilindro com um canal denominado porta de combustão, enquanto o oxidante permanece em um recipiente pressurizado conectado ao sistema por dutos e válvulas de controle.

Durante a ignição, uma chama de difusão turbulenta se forma ao longo da superfície do combustível. A combustão é sustentada pela transferência de calor da chama para o combustível sólido, resultando na vaporização contínua do combustível até que o fluxo de oxidante seja interrompido.

Uma análise importante desses sistemas considera a taxa de regressão, fornecendo dados sobre eficiência energética, fluxo de massa no motor e vazão mássica dos gases.

2.0.2 Vantagens e desvantagens da propulsão híbrida

As destacadas vantagens dos sistemas de propulsão híbrida são as seguintes, conforme retirado de (SUTTON; BIBLARZ, 2010):

1. Melhor segurança contra explosões ou detonações durante a fabricação, armazenamento e operação.
2. Capacidade de iniciar, parar e reiniciar.
3. Relativa simplicidade, que pode se traduzir em custo total do sistema menor em comparação com líquidos.
4. Impulso específico mais elevado do que motores de foguete sólido e impulso específico de densidade mais elevada do que motores líquidos bipropelentes.
5. Capacidade de variar suavemente o impulso ao longo de uma ampla faixa, conforme necessário.

Por outro lado, existem algumas desvantagens associadas aos sistemas de propulsão híbrida (SUTTON; BIBLARZ, 2010):

1. Variação da razão de mistura e, consequentemente, do impulso específico durante a operação em estado estacionário (bem como durante o controle de aceleração).
2. Geometrias de combustível relativamente complicadas com resíduos de combustível significativos no final da queima, o que reduz um pouco a fração de massa e pode variar se houver controle de aceleração aleatório.
3. Propensão a grandes flutuações de pressão de baixa frequência e amplitude (denominadas *chugging*).
4. Balística interna relativamente complicada do motor, resultando em descrições incompletas tanto das taxas de regressão do combustível quanto dos efeitos de dimensionamento do motor, afetando o projeto de sistemas híbridos de grande porte.

Uma grande desvantagem adicional dos sistemas de propulsão híbrida é a baixa taxa de regressão. No entanto, parte desses problemas tem sido atenuada com o emprego de parafina como combustível. O uso da parafina aumentou a taxa de regressão do grão sólido de 3 a 4 vezes mais em comparação com os combustíveis clássicos utilizados no

passado, como o HDPE (polietileno de alta densidade) e o HTPB (polibutadieno terminado em hidroxila) (BERTOLDI; GELAIN; HENDRICK, 2022). Pela figura 2 e tabela 1, é possível observar a alta taxa de regressão dos combustíveis sólidos compostos com parafina, como PR100, PR95PE05 e PR90PE10, em relação aos demais, como HTPB, HDPE, LDPE e SP-1a.

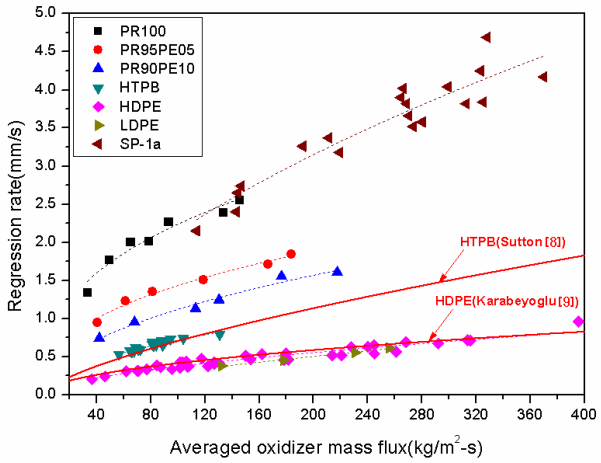


Figura 2 – Taxa de regressão versus fluxo de massa do oxidante médio para cada combustível. (KIM et al., 2010)

Tipo do Combustível	a	n	Taxa de aumento relativo ao HDPE
PR100	0.410	0.37	6.4
SP-1a	0.117	0.62	5.7
PR95PE05	0.234	0.39	3.9
PR90PE10	0.120	0.49	3.0
HTPB	0.072	0.50	2.0
HDPE/LDPE	0.026	0.58	1

Tabela 1 – Comparação das correlações das taxas de regressão para cada tipo de combustível da figura 2. Adaptado de (KIM et al., 2010)

Os parâmetros a e n, apresentados na tabela 1, são coeficientes da regressão determinados experimentalmente para cada combustível.

2.0.3 Formulação Geral Taxa de Regressão

A taxa de regressão ($\dot{r}$), que representa a velocidade de queima do propelente, é determinada por diversos fatores, incluindo o fluxo mássico de oxidante (G), o diâmetro da porta de combustão (D), entre outros parâmetros. A equação fundamental para a taxa de regressão, conforme a 2.1, revela a dependência crucial desses parâmetros no comportamento do projeto. Esta equação, amplamente reconhecida na literatura de propulsão híbrida, estabelece uma relação única entre o fluxo mássico do oxidante e a taxa

de regressão do combustível sólido, com a, n, e m sendo constantes experimentais (KARABEYOGLU; CANTWELL; ZILLIAC, 2007).

$$\dot{r} = aG_{ox}^{n}X^{m} \tag{2.1}$$

Onde,

- $\dot{r}$ é a taxa de regressão do combustível sólido (mm/s),
- a, n, m são constantes de regressão determinadas experimentalmente,
- G_{ox} é o fluxo de massa do oxidante por unidade de área,
- X é a distância axial da porta do grão.

Calcula-se o fluxo de massa do oxidante por unidade de área (G_{ox}) pela equação 2.2:

$$G_{ox} \equiv \frac{\dot{m}_{ox}}{A_{port}} \tag{2.2}$$

Onde, A_{port} representa a área da porta do grão, e $\dot{m}_{ox}$ é o fluxo de massa do oxidante.

Para um grão de combustível cilíndrico, a taxa de regressão média pode ser calculada pela 2.5 (KIM et al., 2010):

$$\Delta m_f = (A_{pf} - A_{pi})L_f\rho_f \tag{2.3}$$

$$A_{pf} = \frac{\pi}{4}D_{pf}^{2} \tag{2.4}$$

$$\bar{\dot{r}}_p = \frac{D_{pf} - D_{pi}}{2t_b} \quad \text{(para grão de combustível cilíndrico)} \tag{2.5}$$

- Δm_f: Variação de massa do combustível sólido.
- A_{pf}: Área final do porta do grão.
- A_{pi}: Área inicial do porta do grão.
- D_{pf}: Diâmetro final do porta do grão.
- D_{pi}: Diâmetro inicial do porta do grão.
- $\bar{\dot{r}}_p$: Taxa de regressão média para grão de combustível cilíndrico.
- t_b: Tempo nominal do teste.

2.0.4 Formulações para o caso do Queimador Plano

Para o caso do queimador plano, a equação 2.5 não pode ser utilizada. A taxa de regressão média para o queimador plano deve ser calculada por 2.8 (KIM et al., 2010)

$$\Delta m_f = (A_f - A_i) L_f \rho_f \tag{2.6}$$

$$A_f = A_i + (H_i - H_f) W_f \tag{2.7}$$

$$\overline{\dot{r}}_s = \frac{H_i - H_f}{t_b} \quad \text{(para grão de combustível em forma de queimador plano)} \tag{2.8}$$

- Δm_f: Variação de massa do combustível sólido.
- A_f: Área transversal final da câmara.
- A_i: Área transversal inicial da câmara.
- H_i: Altura inicial do grão de combustível em forma de queimador plano.
- H_f: Altura final do grão de combustível em forma de queimador plano.
- L_f: Comprimento do grão de combustível.
- ρ_f: Densidade do grão combustível.
- W_f: Largura do grão de combustível.
- $\overline{\dot{r}}_s$: Taxa de regressão média para grão de combustível para o queimador plano.

De maneira alternativa, a taxa de regressão média para o queimador plano também pode ser calculada pela equação 2.9 (BERTOLDI; GELAIN; HENDRICK, 2022):

$$\overline{\dot{r}} = \frac{\Delta m_f}{A_f t_b \rho_f} \tag{2.9}$$

- $\overline{\dot{r}}$: Taxa de regressão média do combustível.
- Δm_f: Variação da massa do combustível sólido.
- A_f: Area média da superfície do queimador.
- t_b: Tempo nominal do teste da queima.
- ρ_f: Densidade do combústivel.

Na queima do queimador plano, um orifício de estrangulamento é empregado na linha de oxigênio gasoso. Esse dispositivo desempenha a função crucial de controlar a injeção do oxidante, assegurando um fluxo de massa constante e proporcionando isolamento às válvulas em relação às oscilações de pressão na câmara de combustão (BERTOLDI; GELAIN; HENDRICK, 2022). Essa medida contribui para a estabilidade operacional do sistema, ao mesmo tempo em que preserva o controle preciso sobre a introdução do oxidante durante o processo de queima.

A taxa média de fluxo de massa do oxidante é calculada dividindo a massa total injetada de oxidante pelo tempo de queima. Esses dados, juntamente com a massa medida do combustível, são usados para avaliar a taxa total de fluxo de massa do propelente. A taxa do fluxo de massa que passa por ele pode ser calculada de acordo com a equação 2.10 (BERTOLDI; GELAIN; HENDRICK, 2022):

$$\dot{m} = c_D A \sqrt{\gamma \rho_0 P_0 \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}} \tag{2.10}$$

Onde:

- $\dot{m}$: Taxa de fluxo de massa de oxigênio.
- γ: Razão de capacidade térmica do oxigênio.
- ρ_0: Massa específica do oxigênio a montante do orifício (ambas funções da pressão e temperatura a montante).
- P_0: Pressão do fluxo a montante do orifício.
- A: Área do orifício.
- c_D: Coeficiente de descarga.

A razão média oxidante-combustível ($\overline{O/F}$) é calculada pela equação 2.11, sendo definida como a razão entre o fluxo de massa médio de oxidante ($\dot{m}_{ox}$) e o fluxo de massa médio do combustível sólido ($\dot{m}_f$) (BERTOLDI; GELAIN; HENDRICK, 2022):

$$\overline{O/F} = \frac{\dot{m}_{ox}}{\dot{m}_f} \tag{2.11}$$

Onde:

- $\dot{m}_{ox}$: Taxa de fluxo de massa do oxidante
- $\dot{m}_f$: Taxa de fluxo de massa do combustível sólido

O fluxo de massa inicial de oxidante é definido como a taxa de fluxo de massa instantânea de oxidante sobre a área transversal do canal do grão (SUTTON; BIBLARZ, 2010). A eficiência de combustão (η^*) é calculada pela equação 2.12 e é a razão entre a velocidade característica (c^*_{exp}), calculada pela equação 2.13, e a velocidade característica teórica (c^*_{th}), obtida usando um software (BERTOLDI; GELAIN; HENDRICK, 2022).

$$\eta^* = \frac{c^*_{\text{exp}}}{c^*_{\text{th}}} \tag{2.12}$$

$$c^*_{\text{exp}} = \frac{\overline{P_c} \cdot A_t}{\overline{\dot{m}}} \tag{2.13}$$

Onde:

- c^*_{exp}: velocidade característica
- $\overline{P_c}$: pressão média da câmara de combustão
- A_t: área da garganta
- $\overline{\dot{m}}$: taxa mássica média de fluxo de massa

No entanto, devido ao recorte escolhido para o presente trabalho, a taxa de regressão será determinada em pixels por segundo e na variação percentual entre as alturas final e inicial em pixels.

3 Fundamentação Métodos de Visão Computacional em Python

Visão computacional é uma área multidisciplinar que estuda métodos de processamento de imagens com a finalidade de extrair informações significativas de dados visuais. Esse capítulo apresenta as bibliotecas utilizadas no desenvolvimento deste trabalho..

3.1 Bibliotecas

3.1.1 NumPy

A biblioteca **NumPy** é fundamental para a computação científica em Python, oferecendo suporte para arrays e matrizes de grandes dimensões, com funções de alto desempenho (NumPy, 2024).

- **np.array**
  - Serve para criar um array NumPy a partir de qualquer estrutura de dados como listas ou tuplas.
  - **Inputs:**
    - **object**: Objeto de entrada (lista, tupla, etc.) a ser convertido em array.
    - **dtype** (opcional): Tipo de dados desejado para os elementos do array.
- **np.ceil**
  - Retorna o menor inteiro maior ou igual a um número dado.
  - **Inputs:**
    - **x**: Valor numérico ou array de entrada.
- **np.sqrt**
  - Calcula a raiz quadrada de cada elemento no array de entrada.
  - **Inputs:**
    - **x**: Valor numérico ou array de entrada.
- **np.frombuffer**
  - Interpreta um buffer como um array unidimensional.

  - **Inputs:**
    - **buffer**: Objeto de buffer de entrada.
    - **dtype** (opcional): Tipo de dados desejado.

- **np.reshape**
  - Dá a um array uma nova forma sem alterar seus dados.
  - **Inputs:**
    - **a**: Array a ser remodelado.
    - **newshape**: Nova forma desejada.

- **np.zeros**
  - Retorna um novo array preenchido com zeros.
  - **Inputs:**
    - **shape**: Forma do novo array.
    - **dtype** (opcional): Tipo de dados desejado.

- **np.nonzero**
  - Retorna os índices dos elementos não nulos.
  - **Inputs:**
    - **a**: Array de entrada.

3.1.2 OpenCV

A biblioteca **OpenCV** (Open Source Computer Vision Library) é uma biblioteca de programação voltada principalmente para a visão computacional em tempo real (OpenCV, 2024).

- **cv2.VideoCapture**
  - Usada para capturar vídeos de um arquivo ou câmera.
  - **Inputs:**
    - **filename**: Nome do arquivo de vídeo ou ID da câmera.

- **cap.read**
  - Faz a leitura do frame do vídeo.
  - **Retorna:**
    - **ret**: Valor booleano que indica se o quadro foi lido com sucesso.

    - **frame**: O quadro capturado.

- **cap.set**
  - Define um atributo da captura de vídeo.
  - **Inputs:**
    - **propId**: ID da propriedade a ser ajustada (por exemplo, posição do frame).
    - **value**: Novo valor para a propriedade.

- **cap.get**
  - Obtém uma propriedade da captura de vídeo.
  - **Inputs:**
    - **propId**: ID da propriedade a ser obtida (por exemplo, taxa de quadros).

- **cv2.cvtColor**
  - Converte uma imagem de um espaço de cores para outro.
  - **Inputs:**
    - **src**: Imagem de entrada.
    - **code**: Código que indica o tipo de conversão (por exemplo, `cv2.COLOR_BGR2RGB`).

- **cv2.threshold**
  - Aplica uma limiarização à imagem.
  - **Inputs:**
    - **src**: Imagem de entrada em escala de cinza.
    - **thresh**: Valor do limiar.
    - **maxval**: Valor máximo atribuído aos pixels acima do limiar.
    - **type**: Tipo de limiarização (por exemplo, `cv2.THRESH_BINARY`).

- **cv2.findContours**
  - Detecta contornos em uma imagem binária.
  - **Inputs:**
    - **image**: A imagem binária de entrada.
    - **mode**: Modo de recuperação dos contornos (por exemplo, `cv2.RETR_EXTERNAL`).
    - **method**: Método de aproximação de contorno (por exemplo, `cv2.CHAIN_APPROX_SIMP`

- **cv2.imwrite**
  - Salva uma imagem em um arquivo.

  - **Inputs:**
    - **filename**: Nome do arquivo de saída.
    - **img**: Imagem a ser salva.

3.1.3 Matplotlib

A biblioteca **Matplotlib** é usada para a criação de visualizações estáticas, animadas e interativas em Python (Matplotlib Developers, 2024).

- **plt.figure**
  - Cria uma nova figura.
  - **Inputs:**
    - **figsize** (opcional): Tamanho da figura em polegadas (largura, altura).
- **plt.imshow**
  - Exibe uma imagem em uma figura.
  - **Inputs:**
    - **X**: Imagem de entrada.
    - **cmap** (opcional): Mapa de cores (por exemplo, `'gray'`).
- **plt.axis**
  - Controla a exibição dos eixos.
  - **Inputs:**
    - **option**: Configuração dos eixos (`'on'`, `'off'`).
- **plt.title**
  - Define um título para a figura.
  - **Inputs:**
    - **label**: Texto do título.
- **plt.show**
  - Exibe a figura.

3.1.4 *ipywidgets*

Com esta biblioteca, é possível criar widgets interativos dentro de um Jupyter Notebook (ipywidgets, 2024).

- **widgets.FloatSlider**
  - Constrói um controle deslizante para valores de ponto flutuante.
  - **Inputs:**
    - **value** (opcional): Valor inicial do controle deslizante.
    - **min**: Valor mínimo.
    - **max**: Valor máximo.
    - **step** (opcional): Incremento.
    - **description** (opcional): Descrição do widget.
- **widgets.fixed**
  - Define um valor fixo para um parâmetro em uma função interativa.
  - **Inputs:**
    - **value**: Valor a ser fixado.
- **interact**
  - Renderiza uma interface interativa para uma função.
  - **Inputs:**
    - **function**: Função a ser chamada interativamente.
    - **kwargs**: Argumentos da função, associados aos widgets.

3.1.5 IPython.display

A biblioteca **IPython.display** fornece funções de exibição para renderizar objetos ricos em um Jupyter Notebook (IPython, 2024).

- **display**
  - Exibe um objeto na saída do notebook.
  - **Inputs:**
    - **obj**: Objeto a ser exibido.
- **clear_output**
  - Limpa a saída da célula.

  - **Inputs:**
    - **wait** (opcional): Se `True`, espera antes de limpar.

3.1.6 PIL

A biblioteca **PIL** (Python Imaging Library) é usada para carregar, editar e salvar vários formatos de arquivos de imagem (Python Imaging Library (PIL), 2024).

- **Image.fromarray**
  - Constrói uma imagem a partir de um array NumPy.
  - **Inputs:**
    - **obj**: Array de entrada.
    - **mode** (opcional): Modo de cor.
- **Image.new**
  - Cria uma nova imagem em branco.
  - **Inputs:**
    - **mode**: Modo de cor (por exemplo, `'RGB'`).
    - **size**: Tamanho da imagem (largura, altura).
    - **color** (opcional): Cor de preenchimento.

3.1.7 contextlib

A biblioteca **contextlib** fornece utilitários para lidar com gerenciadores de contexto (Python Software Foundation, 2024a).

- **contextlib.redirect_stdout**
  - Redireciona a saída padrão para outro objeto.
  - **Inputs:**
    - **new_target**: Novo destino para a saída padrão (por exemplo, `None` para suprimir a saída).

3.1.8 io

A biblioteca **io** fornece as ferramentas necessárias para manipular fluxos de entrada e saída (Python Software Foundation, 2024b).

- **io.StringIO**
  - Cria um objeto em memória que funciona como um arquivo.

3.1.9 PyTorch

A biblioteca **PyTorch** é usada para aprendizado profundo utilizada para modelagem de redes neurais (PyTorch, 2024).

- **torch.is_tensor**
  - Verifica se o objeto é um tensor.
- **tensor.element_size**
  - Retorna o tamanho em bytes de um elemento no tensor.
- **tensor.nelement**
  - Retorna o número de elementos no tensor.
- **tensor.cpu**
  - Move o tensor para a CPU.
- **tensor.numpy**
  - Retorna o tensor como um array NumPy.

3.1.10 pandas

A biblioteca **pandas** é usada para manipulação e análise de dados (Pandas, 2024).

- **pd.DataFrame**
  - Cria um DataFrame a partir de dados.
- **df.append**
  - Adiciona linhas ao DataFrame.
  - **Inputs:**
    - **other**: DataFrame ou Series a ser adicionado.
    - **ignore_index** (opcional): Se `True`, ignora os índices.
- **df.sort_values**
  - Ordena um DataFrame por coluna.
  - **Inputs:**
    - **by**: Coluna(s) para ordenar.

- **df.to_excel**
  - Exporta um DataFrame para Excel.
  - **Inputs:**
    - **excel_writer**: Nome do arquivo.
    - **index** (opcional): Se `False`, não exporta o índice.

3.1.11 os

A biblioteca **os** fornece uma maneira de usar funcionalidades dependentes do sistema operacional (Python Software Foundation, 2024c).

- **os.makedirs**
  - Cria diretórios recursivamente.
  - **Inputs:**
    - **name**: Caminho do diretório.
    - **exist_ok** (opcional): Se `True`, não gera erro se o diretório existir.
- **os.path.join**
  - Une partes de um caminho de forma inteligente.
- **os.listdir**
  - Lista as entradas em um diretório.
- **os.path.exists**
  - Verifica se um determinado caminho existe.

3.1.12 shutil

A biblioteca **shutil** oferece várias operações de alto nível em arquivos e coleções de arquivos (Python Software Foundation, 2024d).

- **shutil.copy**
  - Copia um arquivo de origem para o destino.
- **shutil.copytree**
  - Copia recursivamente uma árvore de diretórios.
- **shutil.move**

  - Move um arquivo ou diretório para outro local.

- **shutil.rmtree**

  - Remove recursivamente um diretório.

3.1.13 scikit-learn

A biblioteca **scikit-learn** fornece ferramentas para aprendizado de máquina em Python (Scikit-learn, 2024).

- **train_test_split**

  - Divide matrizes ou listas em conjuntos de treinamento e teste.
  - **Inputs:**
    - **arrays**: Sequência de arrays ou listas a serem divididas.
    - **test_size** (opcional): Proporção do conjunto de teste.
    - **train_size** (opcional): Proporção do conjunto de treinamento.
    - **random_state** (opcional): Semente para geração aleatória.

3.1.14 Estado da Arte da Visão Computacional

A visão computacional evoluiu bastante, permitindo a implementação de técnicas avançadas de processamento de imagem e aprendizado de máquina para a extração de informações de dados visuais.

- **Machine Learning vs. Inteligência Artificial**

  - **Inteligência Artificial (IA)**: Campo abrangente que visa construir sistemas capazes de realizar tarefas que normalmente exigiriam inteligência humana.
  - **Aprendizado de Máquina (ML)**: Subconjunto da IA que se concentra em capacitar máquinas a aprender a partir de dados, identificando padrões e tomando decisões com o mínimo de intervenção humana.

As bibliotecas e métodos apresentados neste capítulo são ferramentas essenciais para o desenvolvimento de aplicações que utilizam técnicas de visão computacional e aprendizado de máquina, contribuindo para o avanço do estado da arte na área.

3.1.15 YOLOv8: Detecção de Objetos Rápida e Precisa em Tempo Real

O **YOLOv8** é a versão mais recente da família de modelos YOLO (*You Only Look Once*) para detecção de objetos. Ele traz melhorias interessantes para quem precisa de soluções rápidas e precisas, mantendo o equilíbrio entre precisão e velocidade - ótimo para dispositivos com recursos limitados (Ultralytics, 2024).

3.1.15.1 Sobre o Yolov8

Algumas razões destacam o YOLOv8 (Ultralytics, 2024):

- **Arquitetura Melhorada**: O modelo foi ajustado para ser mais eficiente, melhorando a extração de características importantes sem sacrificar a rapidez.
- **Precisão Aprimorada**: Técnicas avançadas de regularização e ajustes na função de perda resultaram em desempenho superior em testes de benchmark.
- **Flexibilidade em Diferentes Escalas**: Disponível em tamanhos variados (Nano, Small, Medium, Large), tornando-o versátil para usos que vão de dispositivos móveis a servidores robustos.

3.1.15.2 Aplicações do YOLOv8

Essas melhorias permitiram que o YOLOv8 fosse utilizado em diversas áreas (Ultralytics, 2024):

- **Veículos Autônomos**: Detecção precisa de pedestres, sinais de trânsito e outros veículos.
- **Sistemas de Vigilância**: Monitoramento em tempo real com foco em identificar atividades suspeitas.
- **Agricultura de Precisão**: Monitoramento de safras e detecção de pragas.
- **Aplicativos Móveis**: Utilizado em apps de realidade aumentada e outras aplicações que precisam de rápida detecção de objetos.

3.1.15.3 Desempenho e Resultados

O YOLOv8 não só apresenta excelente desempenho em benchmarks como o COCO dataset, superando outros modelos em termos de precisão média, como também é otimizado para ser extremamente rápido. Isso significa que pode ser usado em tempo real, mesmo em hardware mais modesto. Com o código-fonte disponível no GitHub e uma comunidade ativa de desenvolvedores, o modelo está em constante evolução (Ultralytics, 2024).

3.1.15.4 Desvantagens e Limitações da Ferramenta

Apesar das vantagens, ainda há áreas em que o YOLOv8 pode melhorar (Ultralytics, 2024):

- **Detecção de Objetos Pequenos**: O modelo ainda pode ter dificuldades com objetos muito pequenos.
- **Condições Adversas**: Em situações de pouca luz ou clima extremo, o desempenho pode não ser o ideal.
- **Necessidade de Dados Anotados**: Treinar o modelo para tarefas específicas requer conjuntos de dados extensos e bem anotados, o que pode ser um desafio.

3.1.15.5 Métricas YoloV8

O **YOLOv8** fornece diversas métricas para avaliação de desempenho, permitindo a análise da precisão e da eficiência do modelo em diferentes cenários (Ultralytics, 2024).

- **Train/Box Loss**: Essa métrica mede o erro na previsão das caixas delimitadoras (bounding boxes) durante o treinamento. A ideia é que, conforme o modelo vai aprendendo, esse erro deve diminuir.
- **Train/Seg Loss**: Avalia o quão precisa é a segmentação dos objetos durante o treinamento. Uma perda menor aqui significa que o modelo está fazendo um trabalho melhor em separar e identificar as diferentes partes da imagem.
- **Train/Cls Loss**: Essa métrica se refere ao erro na classificação dos objetos ao longo do treinamento. Um valor menor indica que o modelo está ficando mais eficiente em dizer o que é cada objeto.
- **Train/DFL Loss**: Distribution Focal Loss é usada para melhorar a precisão na localização de objetos. Uma perda que vai diminuindo ao longo do treinamento mostra que o modelo está se tornando mais robusto.
- **Val/Box Loss**: Funciona de forma parecida com o *train/box_loss*, mas é calculada em dados de validação. Se houver uma diferença grande entre as perdas de treino e validação, pode ser um sinal de overfitting.
- **Val/Seg Loss**: Mede a precisão da segmentação nos dados de validação. Se essa perda for muito alta, é um indicativo de que o modelo talvez esteja sobreajustando (overfitting) aos dados de treino.
- **Val/Cls Loss**: Avalia o quão bem o modelo classifica objetos nos dados de validação. Se a perda for alta, pode indicar que o modelo não está generalizando bem.

- **Val/DFL Loss**: Calcula o erro de localização para os dados de validação. Diferenças notáveis em relação ao *train/dfl_loss* podem apontar problemas na capacidade do modelo de generalizar para novos dados.
- **Metrics/Precision(B) e Metrics/Precision(M)**: Essas métricas medem a precisão das previsões positivas. Se os valores são altos, isso indica que o modelo tem poucas previsões falsas positivas.
- **Metrics/Recall(B) e Metrics/Recall(M)**: Aqui, o foco é na capacidade do modelo de capturar todos os casos positivos reais. Valores altos significam que o modelo está capturando bem os positivos.
- **Metrics/mAP50(B) e Metrics/mAP50(M)**: A média de precisão com IoU de 50% (Intersection over Union). Quando os valores são altos, isso é um bom sinal de que o modelo está conseguindo detectar e localizar os objetos corretamente.
- **Metrics/mAP50-95(B) e Metrics/mAP50-95(M)**: Essa métrica calcula a média de precisão em diferentes limiares de IoU. Valores mais altos aqui indicam que o modelo é robusto em uma variedade de condições de detecção e localização.

3.1.16 Overfitting e Underfitting

O **YOLOv8** utiliza técnicas avançadas para mitigar problemas de overfitting e underfitting, garantindo melhor generalização dos modelos (Ultralytics, 2024).

- **Overfitting:** Ocorre quando o modelo aprende demais os detalhes dos dados de treinamento, prejudicando a generalização. Indicado por uma perda de treinamento baixa e perda de validação alta. Soluções incluem regularização, dropout e data augmentation.
- **Underfitting:** Acontece quando o modelo é muito simples ou não foi treinado o suficiente para capturar os padrões dos dados. Refletido em perdas altas tanto no treinamento quanto na validação. Mitigado aumentando a complexidade do modelo ou o tempo de treinamento.

4 Base de Dados Experimental

Este trabalho fez uso do banco de dados de pesquisas conduzidas pelo Departamento de Aero-Termo-Mecânica (ATM) da Universidade Livre de Bruxelas, em parceria com o Laboratório de Propulsão Química (CPL) da Universidade de Brasília, no âmbito do projeto *Experimental characterisation with firing tests of regression rate in hybrid rocket engines using paraffin fuel doped with metallic particles*, desenvolvido pelos professores doutores Artur Elias M. Bertoldi (CPL/UnB) e Patrick Hendrick (ATM/ULB).

Na seção 4.0.1 é descrito o aparato experimental, que contém as informações de (R. GELAIN et al., 2022), uma das pesquisas do projeto mencionado acima; entre outras informações relevantes para obtenção dos dados que serão analisados neste trabalho de conclusão de curso.

4.0.1 Aparato Experimental:

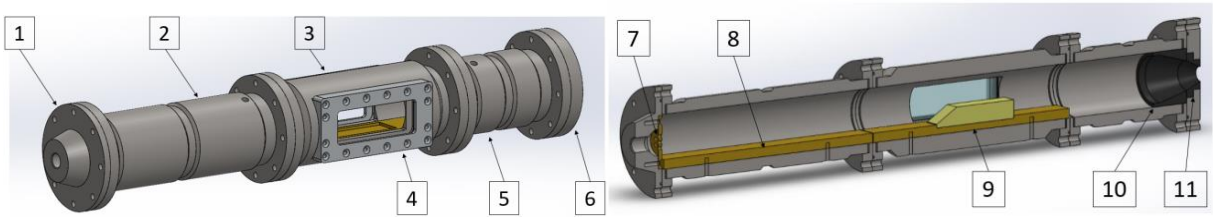
Todas informações descritas nesta seção, foram retiradas do artigo (R. GELAIN et al., 2022).

4.0.1.1 Contextualização:

No âmbito do desenvolvimento de sistemas de propulsão espacial mais eficientes e economicamente viáveis, a Université Libre de Bruxelles (ULB), especificamente o Departamento Aero-Termo-Mecânico (ATM) em colaboração com o Laboratório de Propulsão Química da Universidade de Brasília, realiza pesquisas essenciais na caracterização de propelentes. O foco principal recai sobre os motores de foguete híbridos (HREs), que oferecem flexibilidade na escolha de combustíveis devido à capacidade de misturar grãos sólidos com diferentes aditivos para otimizar propriedades como eficiência de combustão e resistência estrutural. Atualmente, as investigações incluem o uso de um queimador plano (*slab-burner*) baseado em tecnologia de motor foguete híbrido em colaboração com a Royal Military Academy of Belgium (RMA). O aprimoramento do queimador plano RMA resultou na concepção do motor MOUETTE (Moteur OptiqUe pour ÉTudier et Tester Ergols hybrides). O motor MOUETTE apresenta modificações significativas em relação ao design original, incluindo uma forma cilíndrica e aberturas ópticas para medições visuais. Esse novo sistema oferece versatilidade, podendo ser operado tanto como um queimador de bancada convencional quanto como um HRE. Este tipo de queimador plano tem como objetivo fornecer dados essenciais como a taxa de regressão de novas misturas de combustíveis, assim como investigar fenômenos particulares, como o arrastamento de gotículas de cera de parafina e a instabilidade de camadas líquidas.

4.0.1.1.1 Queimador Plano MOUETTE:

A iteração final de design resultou no conceito ilustrado na figura 3, que mostram um modelo CAD dos componentes que constituem o queimador plano MOUETTE. A tabela 2 apresenta as descrições e materiais das etiquetas de números da figura 3.



(a) (b)

Figura 3 – (a): Vista Lateral e (b): Seção Longitudinal do Modelo CAD 3D do MOUETTE (R. GELAIN et al., 2022)

Nº	Componente	Descrição	Material
1	Cabeça do Injetor	Coletor de oxigênio, suporta a placa do injetor	Aço Inoxidável
2	Pré-câmara	Condiciona o fluxo de oxidante proveniente do injetor	Aço Inoxidável
3	Câmara Principal	Seção de teste do MOUETTE	Aço Inoxidável
4	Moldura da Janela	Suporte estrutural para a janela de vidro de quartzo	Alumínio
5	Pós-câmara	Parte traseira do queimador	Aço Inoxidável
6	Suporte da Tubeira	Mantém o inserto da garganta	Aço Inoxidável
7	Placa Injetora	Injeta oxigênio no queimador plano	Latão
8	Base da Pré-câmara	Evita turbulências induzidas pela mudança de seção transversal	Latão
9	Suporte do Grão de Combustível	Suporta o grão de combustível sólido	Latão
10	Inserto da seção convergente da Tubeira	Primeira parte da Tubeira	Grafite
11	Garganta da Tubeira	Segunda parte da Tubeira, com diferentes diâmetros de garganta	Grafite

Tabela 2 – Componentes do queimador plano MOUETTE enumerados de acordo com a figura 3. (R. GELAIN et al., 2022)

4.0.1.2 Grão de Combustível:

Para a campanha de comissionamento, uma série de grãos de combustível foi fabricada usando parafina Tudamelt 52/54. Os grãos têm uma massa de cerca de 90 gramas, formato com borda chanfrada e foram moldados usando parafina fornecida em pellets, apresentado na figura 4 . A ignição é realizada por um dispositivo pirotécnico simples, composto por 40% de açúcar e 60% de nitrato de potássio.

(a) (b)

Figura 4 – (a): Grão combustível de parafina e (b): Molde do grão combustível(R. GELAIN et al., 2022)

4.0.1.3 Sistema de Aquisição de Dados e Controle:

O sistema de aquisição de dados escolhido para o MOUETTE é um National Instruments (NI) DAQmx USB 6218, gerenciado por um programa LabVIEW dedicado. Esse sistema permite a coleta de dados de sensores, controle da ignição e abertura/fechamento de válvulas, com um sequenciamento automático.

4.0.1.4 Configuração do Teste:

A campanha de testes de comissionamento envolveu instrumentação do sistema com transdutores de pressão Heim 100 bar e termopares tipo K. A câmera de alta velocidade Photron FASTCAM SA4 foi usada para gravar vídeos da combustão, e o teste ocorreu em uma instalação da Base da Força Aérea belga localizada em Beauvechain. A figura 5 apresenta a configuração dos equipamentos e da bancada de testes do MOUETTE.

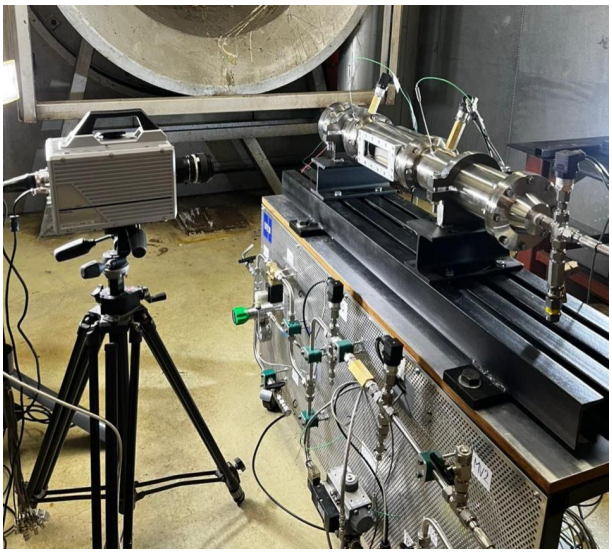


Figura 5 – Configuração dos equipamentos e da bancada de testes do MOUETTE (R. GELAIN et al., 2022)

4.0.1.5 Visão Geral da Campanha de Testes:

A matriz de testes foi planejada para fornecer resultados em diferentes pressões e taxas de fluxo, abrangendo diversos pontos operacionais. Os testes foram conduzidos com sucesso, variando a taxa de fluxo de oxigênio e a pressão da câmara de combustão. Diferentes formulações e geometrias de grãos de combustível foram exploradas, e os resultados foram documentados. A figura 6 apresenta o MOUTTE durante o um teste.

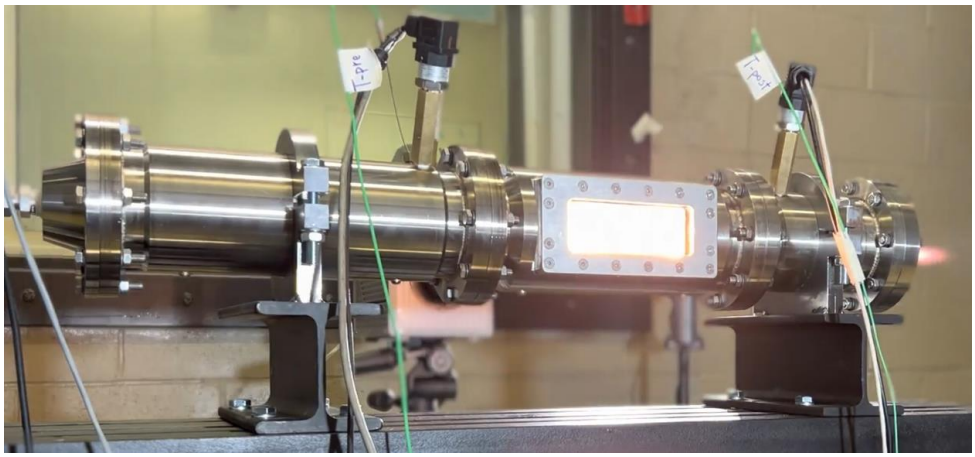


Figura 6 – Fotografia do MOUETTE durante um teste de disparo (R. GELAIN et al., 2022)

Nº	Aditivo	Mistura
1	LiAlH	0,5%
2	LiAlH	0,5%
3	LiAlH	1,0%
4	LiAlH	1,0%
5	LiAlH	1,5%
6	LiAlH	1,5%
7	MgB2	0,5%
8	MgB2	1,5%
9	MgB2	1,0%
10	MgB2	0,5%
11	MgB2	1,0%
12	MgB2	1,5%

Tabela 3 – Visão Geral dos testes (R. GELAIN et al., 2022)

5 METODOLOGIA

Este capítulo descreve detalhadamente a metodologia empregada neste trabalho, que envolve a preparação dos dados, o treinamento de um modelo de segmentação utilizando o YOLOv8 e a aplicação deste modelo na análise de vídeos de combustão de propelentes sólidos.

5.1 Visão Geral da Metodologia

A figura 7 apresenta o fluxograma da metodologia adotada, detalhando suas etapas principais.

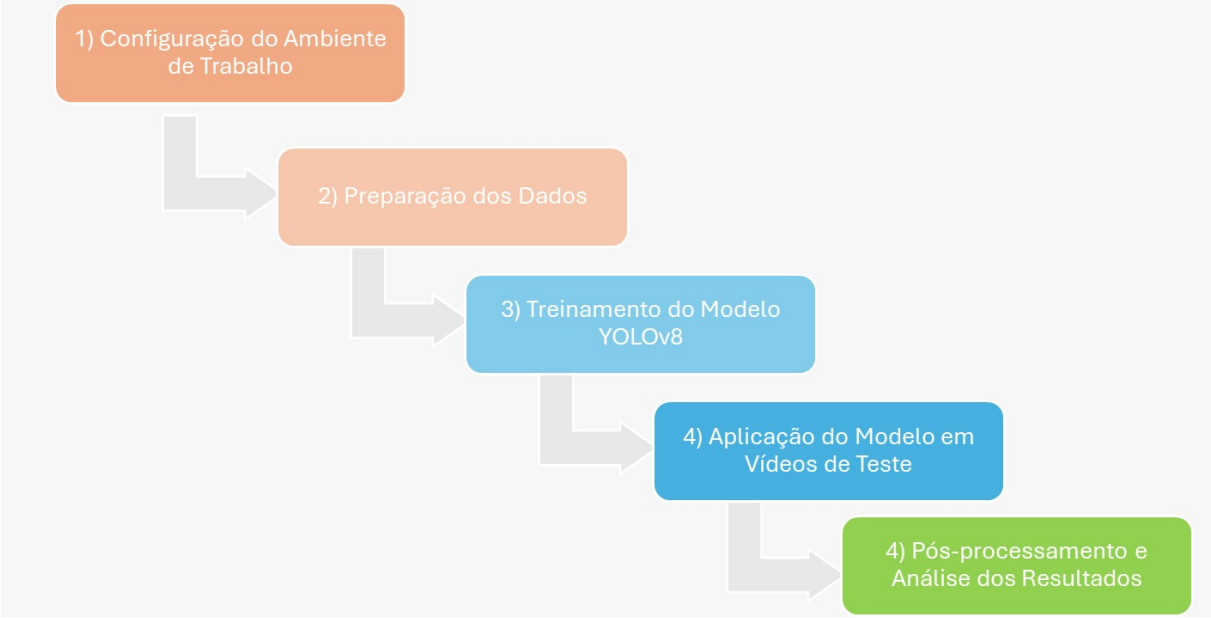


Figura 7 – Fluxograma da metolodologia adotada.

5.2 Configuração do Ambiente de Trabalho

Para a execução dos scripts e manipulação dos dados, utilizou-se o Google Colab devido à sua facilidade de integração com o Google Drive e disponibilização de recursos computacionais adequados. O ambiente foi configurado conforme os seguintes passos:

- **Montagem do Google Drive**: Estabeleceu-se uma conexão entre o Google Colab e o Google Drive para acesso aos vídeos e armazenamento dos resultados.
- **Importação de Bibliotecas**: Foram importadas bibliotecas essenciais, como `numpy`, `pandas`, `cv2` (OpenCV), `matplotlib`, `ipywidgets` e `ultralytics`.

5.3 Preparação dos Dados

5.3.1 Extração de Frames dos Vídeos

Inicialmente, foram selecionados os vídeos dos testes de combustão a partir do Google Drive. Utilizando a biblioteca OpenCV, desenvolveu-se um script que automatiza a leitura dos vídeos e a extração de frames específicos. Foram extraídos 20 frames por teste, os quais foram salvos em pastas organizadas por teste dentro do diretório `Outputs` (ver Figura 8).

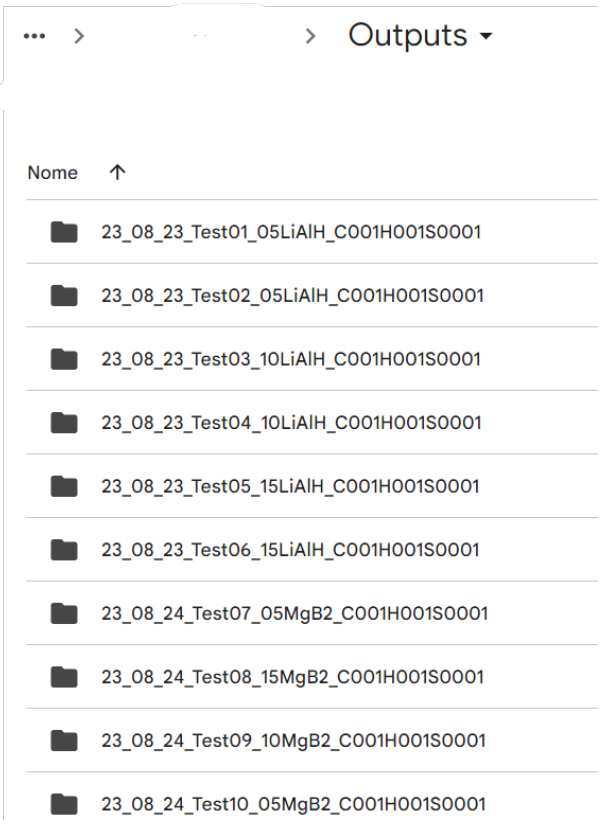


Figura 8 – Estrutura de diretórios na pasta Outputs.

5.3.2 Anotação das Imagens com CVAT.AI

Para criar o dataset de treinamento, as imagens extraídas foram anotadas manualmente utilizando a ferramenta CVAT.AI:

- **Upload das Imagens**: As imagens foram carregadas na plataforma CVAT.AI.
- **Anotação Manual**: Utilizou-se o modo *polygon shape* para delimitar a área do combustível sólido em cada imagem (Figura 9).
- **Exportação das Anotações**: As anotações foram exportadas no formato *Segmentation Object* e salvas em arquivos ZIP.

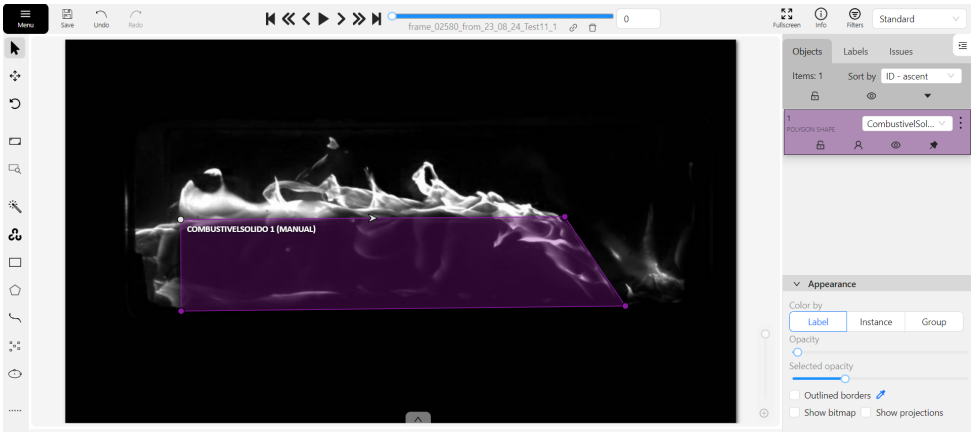


Figura 9 – Anotação das imagens utilizando o CVAT.AI.

5.3.3 Processamento Automatizado das Máscaras de Segmentação

O processamento das máscaras de segmentação foi completamente automatizado através de scripts desenvolvidos especificamente para este fim:

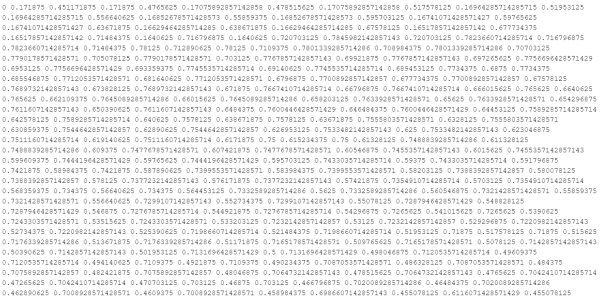
- **Organização das Anotações**:
  - Criou-se uma pasta chamada `Annotations`, onde o usuário faz o upload de todos os arquivos ZIP exportados dos projetos do CVAT.AI.
  - Cada arquivo ZIP corresponde a um projeto de anotação diferente e permanece na pasta `Annotations`.
- **Descompactação Automática das Anotações**:
  - Um script foi desenvolvido para percorrer automaticamente a pasta `Annotations`.
  - Para cada arquivo ZIP encontrado, o script:
    - Cria uma pasta com o mesmo nome do ZIP dentro de `Annotations`.
    - Descompacta o conteúdo do ZIP na pasta criada, preservando a estrutura de diretórios.
  - Este processo assegura que todas as anotações sejam organizadas em pastas separadas, facilitando o acesso aos dados.
- **Extração das Máscaras de Segmentação**:
  - Outro script automatizado acessa cada uma das pastas recém-criadas dentro de `Annotations`.
  - O script procura a subpasta `SegmentationObject` em cada pasta (todas as anotações exportadas do CVAT.AI possuem essa subpasta).
  - As máscaras de segmentação presentes em `SegmentationObject` são copiadas para uma pasta única chamada `Teste_Input_Masks`.

  - Este processo centraliza todas as máscaras de segmentação em um único local para processamento posterior.

- **Conversão Automática para Formato YOLO**:

  - Desenvolveu-se um script que lê todas as máscaras de segmentação na pasta `Teste_Input_Masks`.
  - Para cada máscara, o script:
    * Carrega a imagem da máscara e converte-a para escala de cinza.
    * Extrai os contornos das áreas segmentadas utilizando técnicas de processamento de imagens.
    * Normaliza as coordenadas dos polígonos extraídos com base nas dimensões da imagem.
    * Gera um arquivo `.txt` para cada máscara, contendo as coordenadas normalizadas no formato exigido pelo YOLOv8.
  - Este processo é totalmente automatizado, permitindo a rápida conversão de todas as máscaras para o formato adequado.

(a)



(b)

Figura 10 – (a) Exemplo de máscara de segmentação; (b) Coordenadas extraídas automaticamente da máscara.

Este processamento automatizado garante eficiência e consistência na preparação dos dados, reduzindo a possibilidade de erros manuais e agilizando o fluxo de trabalho.

5.3.4 Criação e Organização do Dataset

Para o treinamento do modelo, o dataset foi organizado de forma automatizada pelos scripts desenvolvidos:

- **Estrutura de Pastas**:

  - Criou-se a pasta `Dataset` com as subpastas `train`, `val` e `test`, cada uma contendo as pastas `images` e `labels`.

- **Divisão dos Dados**:
  - Utilizando scripts em Python e a biblioteca `scikit-learn`, os dados foram automaticamente divididos em 70% para treinamento, 20% para validação e 10% para teste.
  - A divisão assegurou que cada imagem tivesse seu respectivo rótulo, mantendo a consistência entre as pastas `images` e `labels`.

- **Verificação do Dataset**:
  - Implementou-se uma função para verificar o número de arquivos em cada conjunto, assegurando a correta distribuição dos dados (Figura 11).

- **Arquivo de Configuração**:
  - O script gerou automaticamente um arquivo `dataset.yaml` contendo as especificações do dataset para o YOLOv8, incluindo o caminho das imagens e labels para cada conjunto e o número de classes.

```
Train images: 159
Train labels: 159
Validation images: 47
Validation labels: 47
Test images: 25
Test labels: 25
```

Figura 11 – Verificação automatizada do número de arquivos em cada conjunto do dataset.

Essa automação na criação e organização do dataset facilita a replicação do processo por outros usuários e garante que os dados estejam prontos para o treinamento do modelo sem intervenções manuais.

5.4 Treinamento do Modelo

5.4.1 Configuração do Ambiente para o YOLOv8

Instalou-se a versão 8.0.196 do pacote `ultralytics` utilizando o comando `pip`. Importaram-se os módulos necessários e configurou-se o ambiente para o treinamento de forma automatizada.

5.4.2 Treinamento do Modelo de Segmentação

O modelo pré-treinado `yolov8s-seg.pt` foi utilizado como ponto de partida. O treinamento foi configurado e executado através de scripts que automatizam o processo com os seguintes parâmetros:

- **Número de epochs**: 113.
- **Dataset**: Especificado no arquivo `dataset.yaml` gerado automaticamente.

Os resultados do treinamento, incluindo pesos atualizados e métricas de desempenho, foram salvos na pasta `runs` para posterior análise.

5.5 Aplicação do Modelo em Vídeos de Teste

5.5.1 Processamento e Segmentação dos Vídeos

Utilizando scripts automatizados, o modelo treinado foi aplicado nos vídeos de teste:

- **Carregamento do Modelo**:
  - O modelo treinado é carregado automaticamente pelos scripts para realizar as predições.
- **Segmentação Quadro a Quadro**:
  - Os vídeos são processados frame a frame, e a segmentação do combustível sólido é realizada de forma automatizada.
- **Extração de Características**:
  - Scripts específicos extraem os pontos da borda superior do combustível sólido e calculam a posição média ao longo do tempo.

5.5.2 Geração de Resultados e Visualizações

Os resultados são gerados e organizados automaticamente:

- **Criação de Vídeos**:
  - Os frames com as segmentações sobrepostas são compilados em vídeos no formato MP4.

- **Geração de Gráficos**:
  - Gráficos são plotados para mostrar a evolução da posição da borda superior do combustível sólido, sendo salvos em formatos adequados para análise.
- **Salvamento dos Dados**:
  - Tabelas com os dados extraídos são salvas em arquivos Excel (`.xlsx`) de forma automatizada.

5.6 Pós-processamento e Análise dos Resultados

Após a aplicação do modelo, foi realizado um pós-processamento dos resultados:

- **Aplicação de Média Móvel**:
  - Uma média móvel é aplicada aos dados de posição para suavizar flutuações e reduzir ruídos.
- **Filtragem de Valores**:
  - Valores de posição fora do intervalo de interesse (250 a 300 pixels) são filtrados automaticamente.
- **Alinhamento Temporal**:
  - O ponto inicial e final de cada teste dos gráficos da média móvel em pós-processamento foi ajustado para corresponder ao início real da combustão, garantindo precisão na análise temporal.

6 RESULTADOS

6.1 Resultados da Ferramenta Desenvolvida

Neste capítulo, apresentamos os resultados obtidos com a ferramenta desenvolvida para a segmentação e análise de vídeos da queima de combustível sólido. A ferramenta foi implementada em Python utilizando o Google Colab e está estruturada de forma modular, permitindo que qualquer usuário possa utilizá-la futuramente com facilidade, bastando ter acesso às pastas no Google Drive e ao ambiente do Colab.

6.1.1 Descrição da Ferramenta

A ferramenta consiste em um conjunto de scripts que automatizam todo o processo de preparação dos dados, treinamento do modelo de segmentação e aplicação do modelo em novos vídeos. As principais funcionalidades incluem:

- **Integração com Google Drive**: Permite o acesso direto aos arquivos armazenados no Drive, facilitando a organização dos dados e a colaboração.
- **Extração de Frames**: Automatiza a extração de frames dos vídeos selecionados, salvando-os em pastas organizadas por teste.
- **Anotação e Pré-processamento**: Auxilia na preparação dos dados para anotação e converte as máscaras de segmentação para o formato compatível com o YOLOv8.
- **Treinamento do Modelo**: Configura e treina o modelo de segmentação YOLOv8 com os dados preparados.
- **Segmentação de Novos Vídeos**: Aplica o modelo treinado em novos vídeos, gerando predições e extraindo informações relevantes para análise.
- **Visualização e Exportação de Resultados**: Gera visualizações dos resultados e exporta dados e gráficos para posterior análise.

6.1.2 Resultados do Treinamento

O modelo YOLOv8 foi treinado utilizando o dataset preparado pela ferramenta. As métricas obtidas durante o treinamento indicam um desempenho satisfatório na tarefa de segmentação do combustível sólido nos vídeos de combustão.

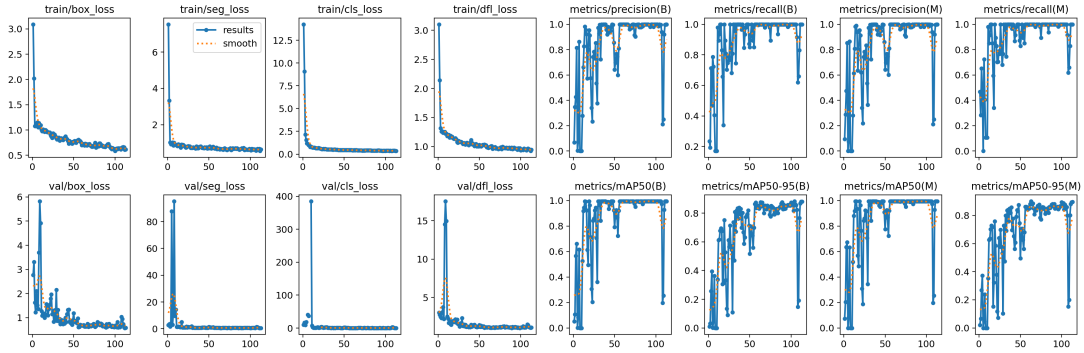


Figura 12 – Métricas obtidas durante o treinamento do modelo YOLOv8.

A Figura 12 apresenta as métricas de desempenho do modelo ao longo das épocas de treinamento, incluindo a precisão (Precision), a revocação (Recall), o F1-score e a média da média de precisão (mAP).

6.1.3 Aplicação da Ferramenta em Vídeos de Teste

Após o treinamento, a ferramenta foi aplicada em vídeos de teste para segmentar a chama e extrair informações relevantes. A interface desenvolvida permite ao usuário selecionar facilmente o vídeo desejado e visualizar os resultados das predições quadro a quadro.

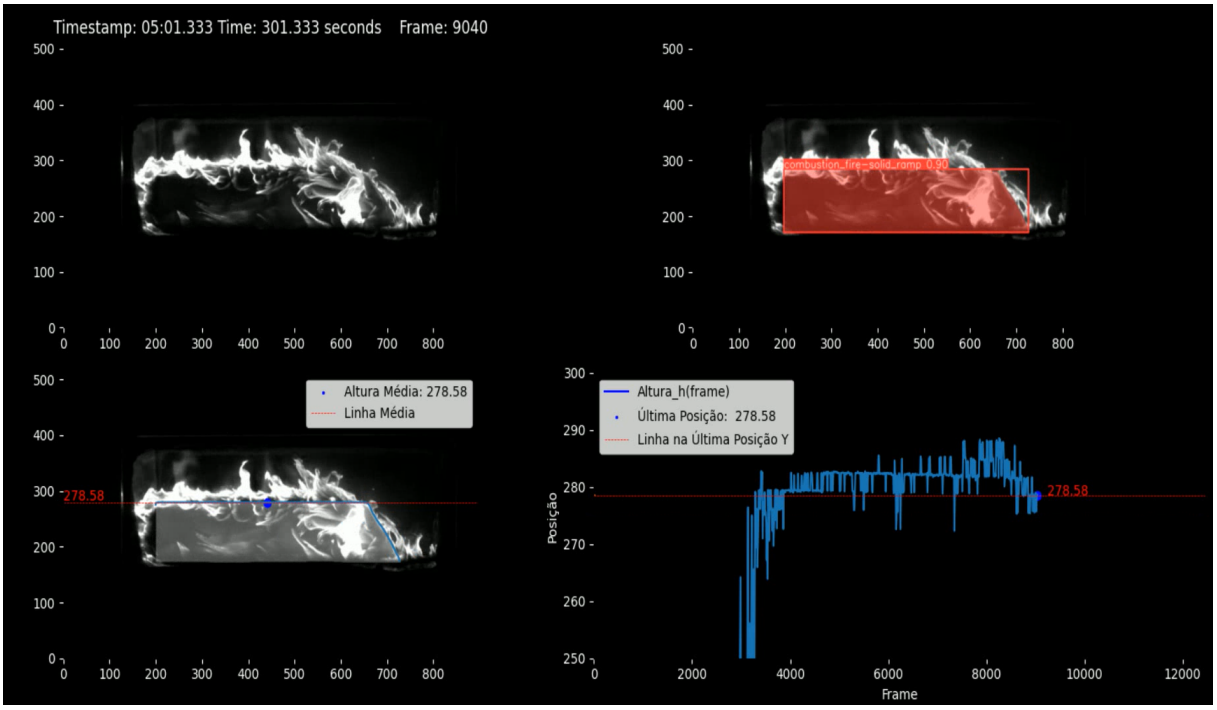


Figura 13 – Exemplo de aplicação da ferramenta em um frame do vídeo de teste, mostrando a segmentação do combustível sólido e a posição da borda superior.

A Figura 13 ilustra a interface da ferramenta durante a segmentação de um frame

específico. A segmentação do combustível sólido é destacada, e a posição da borda superior é calculada e plotada ao longo do tempo.

6.1.4 Organização dos Resultados

Os resultados gerados pela ferramenta são organizados em pastas no Google Drive, facilitando o acesso e a análise posterior. Para cada vídeo processado, são salvos:

- **Vídeo com Segmentações**: Um vídeo no formato MP4 contendo os frames originais com as segmentações sobrepostas.
- **Tabelas de Dados**: Arquivos Excel com informações sobre a posição da borda superior do combustível sólido em cada frame, permitindo análises quantitativas.
- **Gráficos Gerados**: Imagens dos gráficos de evolução da posição da borda superior ao longo do tempo ou do número de frames.

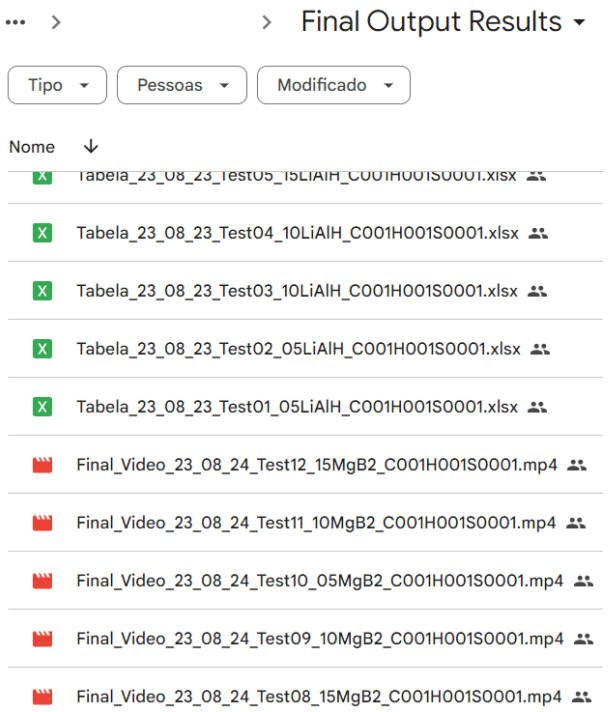


Figura 14 – Estrutura de pastas contendo os resultados finais gerados pela ferramenta.

A Figura 14 mostra a organização das pastas com os resultados finais no Google Drive. Essa estrutura permite que outros pesquisadores acessem facilmente os dados processados.

6.1.5 Exemplo de Resultado Tabular

A ferramenta gera tabelas contendo informações detalhadas sobre cada frame processado. Um exemplo de tabela é apresentado na Figura 15, onde são listados o timestamp, o tempo em segundos, o número do frame e a posição média da borda superior do combustível sólido.

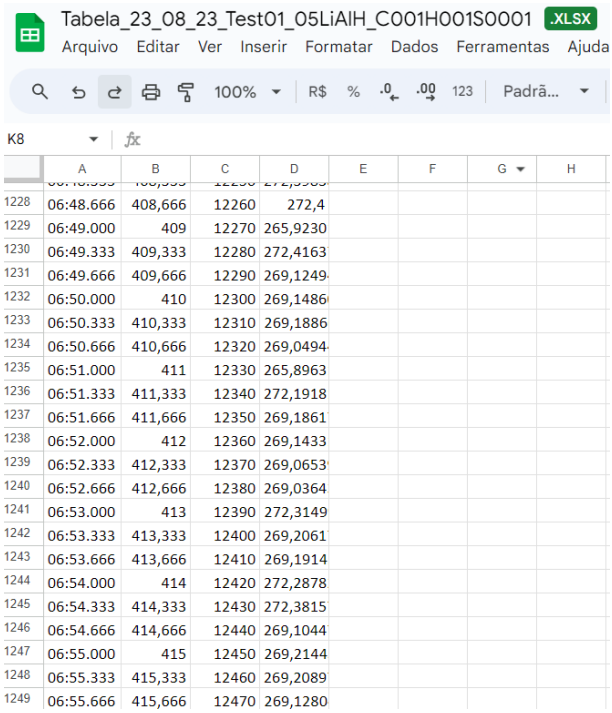


Tabela_23_08_23_Test01_05LiAlH_C001H001S0001 .XLSX

Arquivo Editar Ver Inserir Formatar Dados Ferramentas Ajuda

100% R$ % .0 .00 123 Padrã...

K8

	A	B	C	D	E	F	G	H
1228	06:48.666	408,666	12260	272,4				
1229	06:49.000	409	12270	265,9230				
1230	06:49.333	409,333	12280	272,4163				
1231	06:49.666	409,666	12290	269,1249				
1232	06:50.000	410	12300	269,1486				
1233	06:50.333	410,333	12310	269,1886				
1234	06:50.666	410,666	12320	269,0494				
1235	06:51.000	411	12330	265,8963				
1236	06:51.333	411,333	12340	272,1918				
1237	06:51.666	411,666	12350	269,1861				
1238	06:52.000	412	12360	269,1433				
1239	06:52.333	412,333	12370	269,0653				
1240	06:52.666	412,666	12380	269,0364				
1241	06:53.000	413	12390	272,3149				
1242	06:53.333	413,333	12400	269,2061				
1243	06:53.666	413,666	12410	269,1914				
1244	06:54.000	414	12420	272,2878				
1245	06:54.333	414,333	12430	272,3815				
1246	06:54.666	414,666	12440	269,1044				
1247	06:55.000	415	12450	269,2144				
1248	06:55.333	415,333	12460	269,2089				
1249	06:55.666	415,666	12470	269,1280				

Figura 15 – Exemplo de tabela gerada pela ferramenta com os resultados de um teste.

6.1.6 Disponibilidade e Utilização Futura da Ferramenta

A ferramenta desenvolvida possui uma estrutura modular e é altamente configurável, permitindo que qualquer pessoa possa utilizá-la em novos conjuntos de dados ou adaptá-la para diferentes aplicações. As principais vantagens para utilização futura incluem:

- **Facilidade de Uso**: O usuário precisa apenas organizar os dados em pastas no Google Drive e executar os scripts no Google Colab, sem a necessidade de configurações complexas.
- **Reprodutibilidade**: A utilização de notebooks no Google Colab garante que todo o processo seja documentado e reprodutível por outros pesquisadores.
- **Flexibilidade**: A ferramenta pode ser adaptada para diferentes tipos de vídeos ou tarefas de segmentação, bastando ajustar os parâmetros e dados de entrada.

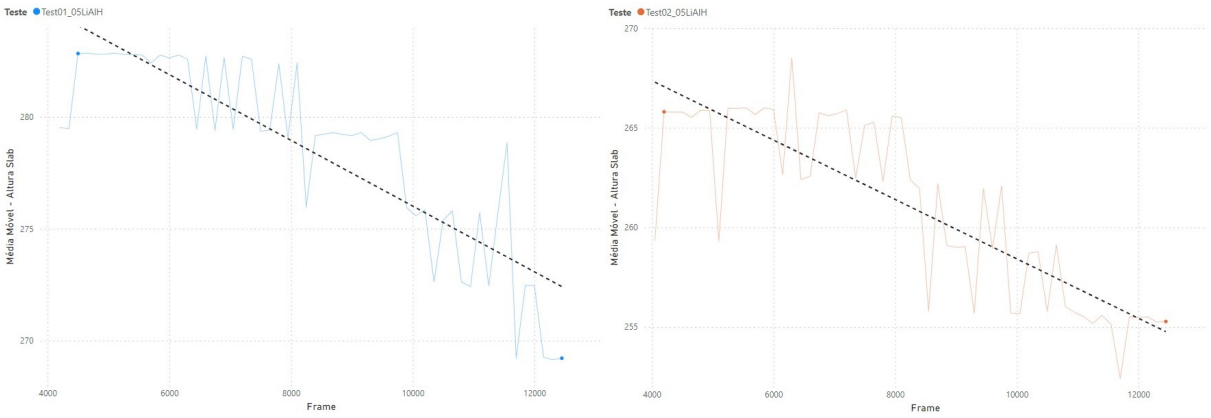
6.2 Resultados Experimentais

Nesta seção serão analisados testes com diferentes aditivos e concentrações. Os gráficos foram obtidos a partir dos dados obtidos.

6.2.1 Análise dos testes dos combústiveis sólidos com aditivo de Hidreto de Alumínio e Lítio

6.2.1.1 Parafina dopada com 0.5% de Hidreto de Alumínio e Lítio

Pela Figura 16a, o Teste 1 começa em 282,84 pixels e termina em 269,21 pixels, com uma taxa de regressão média de 3,43 px/s e uma variação percentual em módulo de 4,82%. Já pela Figura 16b, o Teste 2 inicia em 265,80 pixels e finaliza em 255,27 pixels.



(a) (b)

Figura 16 – (a): Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 1 com 0,5% de LiAlH e (b):Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 2 com 0,5% de LiAlH

A Figura 17 apresenta a média móvel da altura em pixels do combustível sólido ao longo dos testes 1 e 2 para LiAlH (Lithium aluminium hydride) a 0,5%. A taxa de regressão média é de 2,55 px/s, correspondendo a uma variação percentual em módulo de 3,96%.

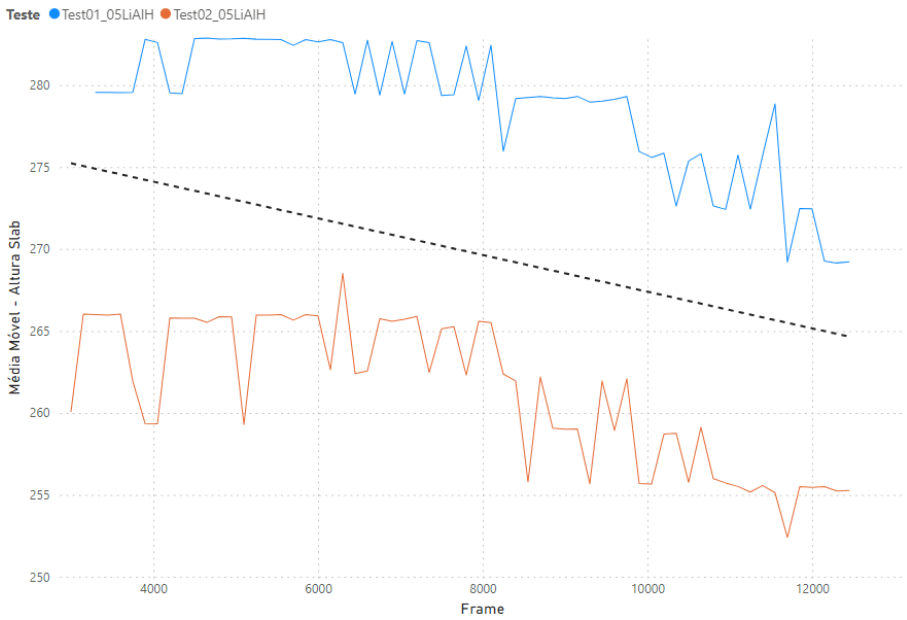
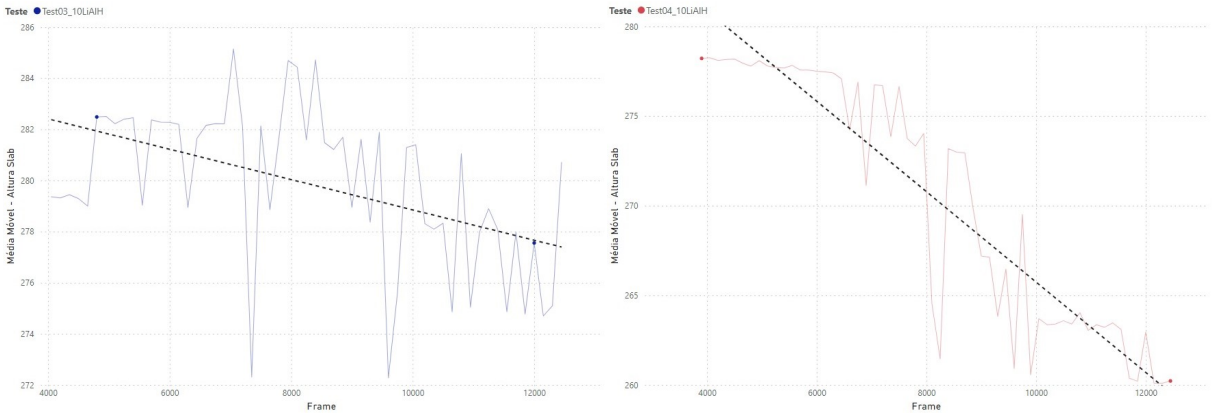


Figura 17 – Comparação da Média Móvel da Posição em Pixels x Frame do Combustível Sólido para Testes com 0,5% de LiAlH

6.2.1.2 Parafina dopada com 1% Hidreto de Alumínio e Lítio

Pela Figura 18a, o Teste 3 começa em 282,48 pixels e termina em 277,56 pixels, com uma taxa de regressão média de 1,37 px/s e uma variação percentual em módulo de 1,74%. Já pela Figura 18b, o Teste 4 inicia em 278,22 pixels e finaliza em 260,22 pixels.



(a) (b)

Figura 18 – (a): Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 3 com 1% de LiAlH e (b):Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 4 com 1% de LiAlH

A Figura 19 apresenta a média móvel da altura em pixels do combustível sólido ao longo dos testes 3 e 4 para LiAlH (Lithium aluminium hydride) a 1,0%. A taxa de regressão média é de 4,21 px/s, correspondendo a uma variação percentual em módulo de 6,47%.

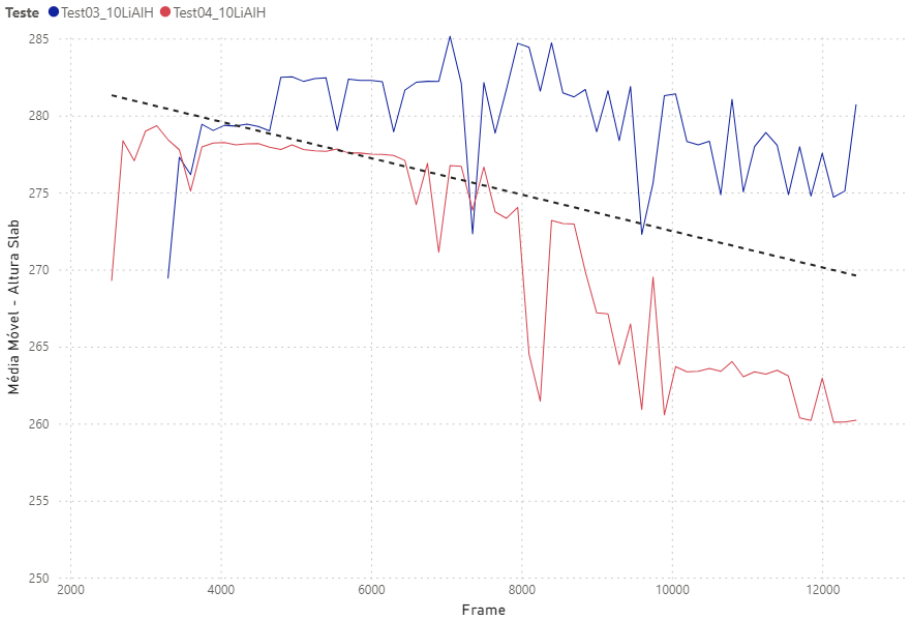
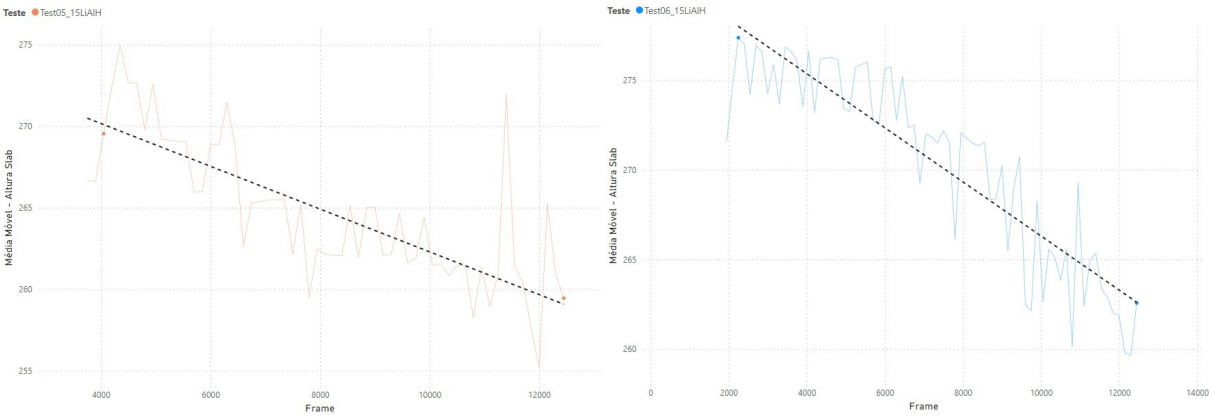


Figura 19 – Comparação da Média Móvel da Posição em Pixels x Frame do Combustível Sólido para Testes com 1% de LiAlH

6.2.1.3 Parafina dopada com 1.5% Hidreto de Alumínio e Lítio

Pela Figura 20a, o Teste 5 começa em 269,54 pixels e termina em 259,45 pixels, com uma taxa de regressão média de 2,40 px/s e uma variação percentual em módulo de 3,74%. Pela Figura 20b, o Teste 6 inicia em 277,37 pixels e finaliza em 262,57 pixels.



(a) (b)

Figura 20 – (a): Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 5 com 1,5% de LiAlH e (b):Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 6 com 1,5% de LiAlH

A Figura 21 apresenta a média móvel da altura em pixels do combustível sólido ao longo dos testes 5 e 6 para LiAlH (Lithium aluminium hydride) a 1,5%. A taxa de regressão média é de 2,90 px/s, correspondendo a uma variação percentual em módulo de 5,34%.

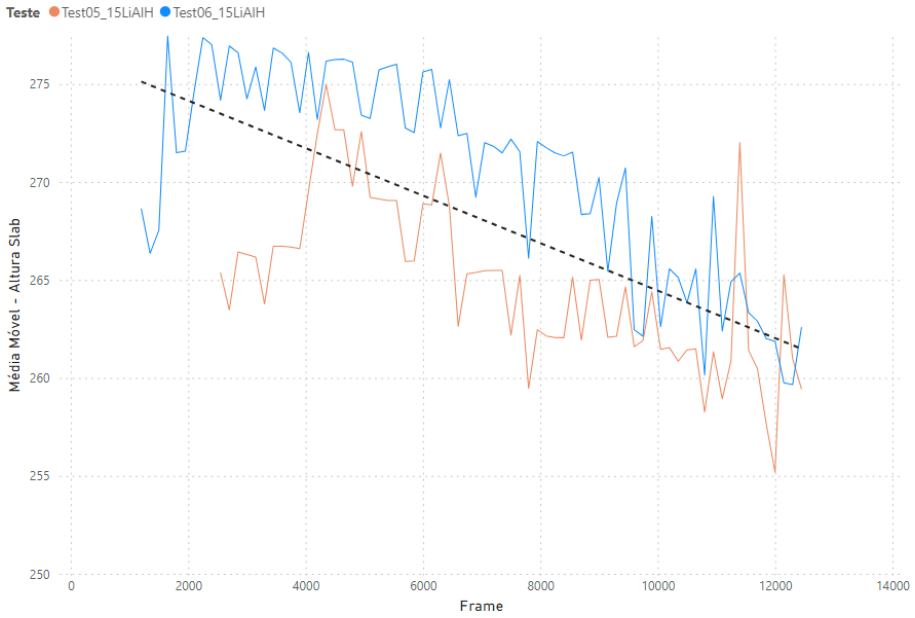


Figura 21 – Comparação da Média Móvel da Posição em Pixels x Frame do Combustível Sólido para Testes com 1,5% de LiAlH

6.2.2 Análise dos testes dos combústiveis sólidos de Parafina com aditivo de diboreto de magnésio (MgB2)

6.2.2.1 Parafina dopada com 0.5% de MgB2

Pela Figura 22a, o Teste 7 começa em 275,59 pixels e termina em 264,29 pixels, com uma taxa de regressão média de 2,22 px/s e uma variação percentual em módulo de 4,10%. Já pela Figura 22b, o Teste 10 inicia em 274,69 pixels e finaliza em 260,11 pixels.

(a) (b)

Figura 22 – (a): Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 7 com 0,5% de MgB2 e (b):Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 10 com 0,5% de MgB2

A Figura 23 apresenta a média móvel da altura em pixels do combustível sólido

ao longo dos testes 7 e 10 para MgB2 a 0,5%. A taxa de regressão média é de 2,66 px/s, correspondendo a uma variação percentual em módulo de 5,31%.

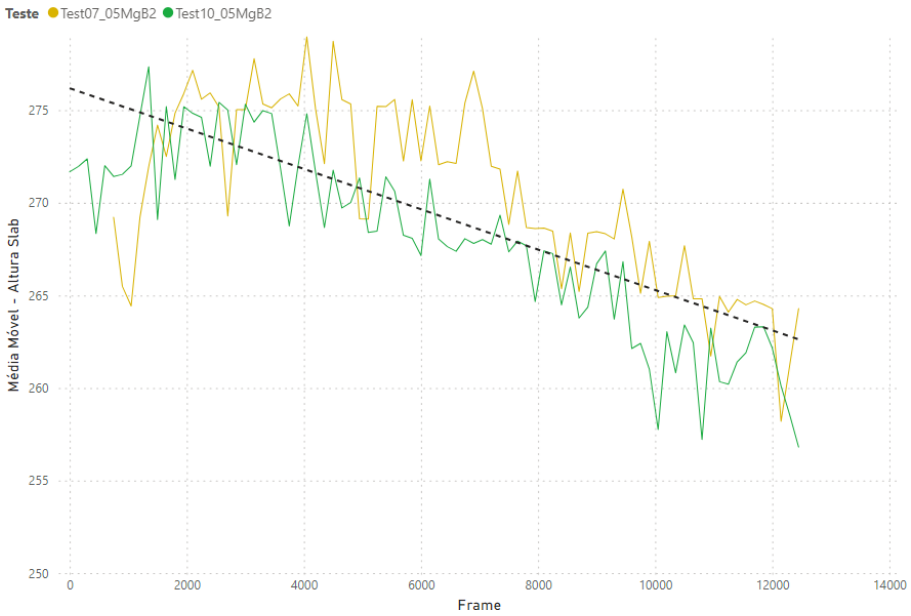
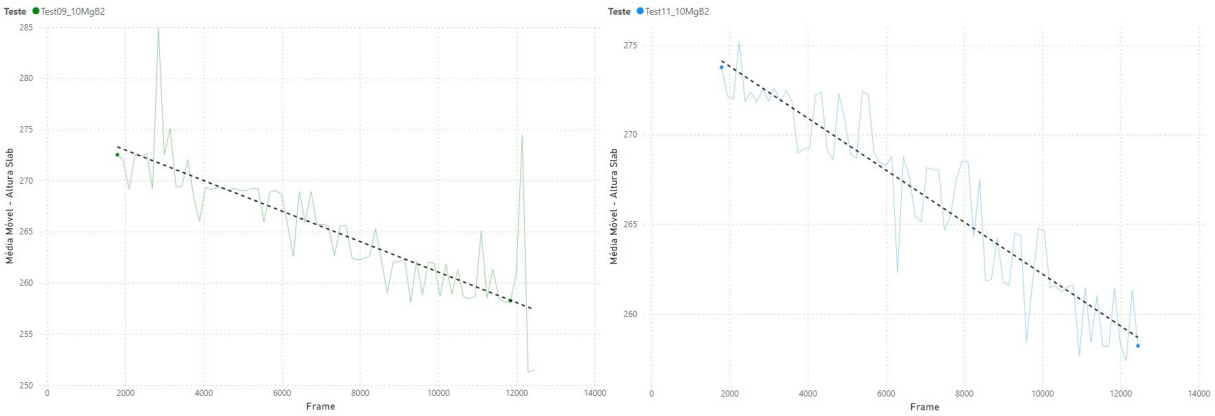


Figura 23 – Comparação da Média Móvel da Posição em Pixels x Frame do Combustível Sólido para Testes com 0.5% de MgB2

6.2.2.2 Parafina dopada com 1% de MgB2

Pela Figura 24a, o Teste 9 começa em 272,52 pixels e termina em 258,24 pixels, com uma taxa de regressão média de 2,84 px/s e uma variação percentual em módulo de 5,24%. Já pela Figura 24b, o Teste 11 inicia em 273,76 pixels e finaliza em 258,22 pixels.



(a) (b)

Figura 24 – (a): Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 9 com 1% de MgB2 e (b):Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 11 com 1% de MgB2

A Figura 25 apresenta a média móvel da altura em pixels do combustível sólido ao longo dos testes 9 e 11 para MgB2 a 1,0%. A taxa de regressão média é de 2,92 px/s,

correspondendo a uma variação percentual em módulo de 5,68%.

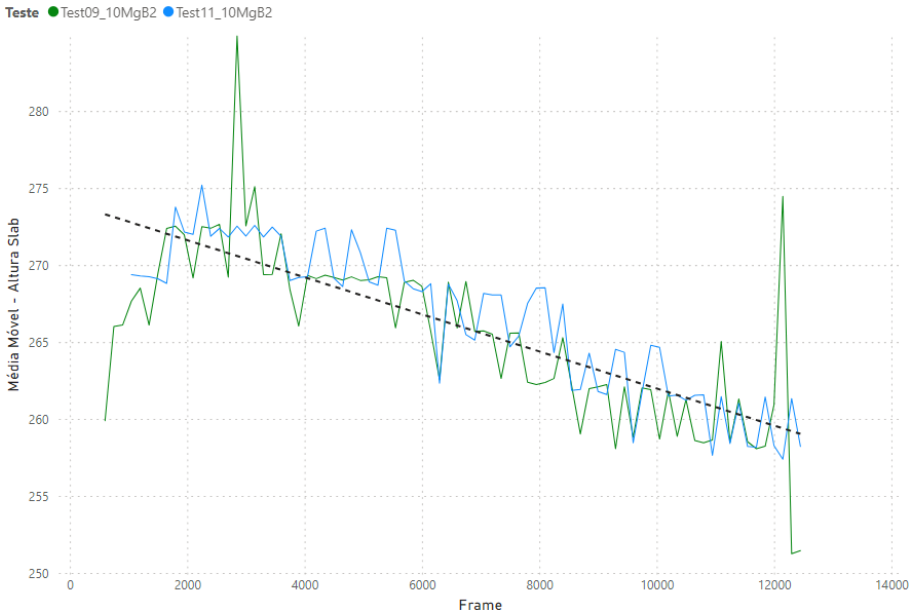
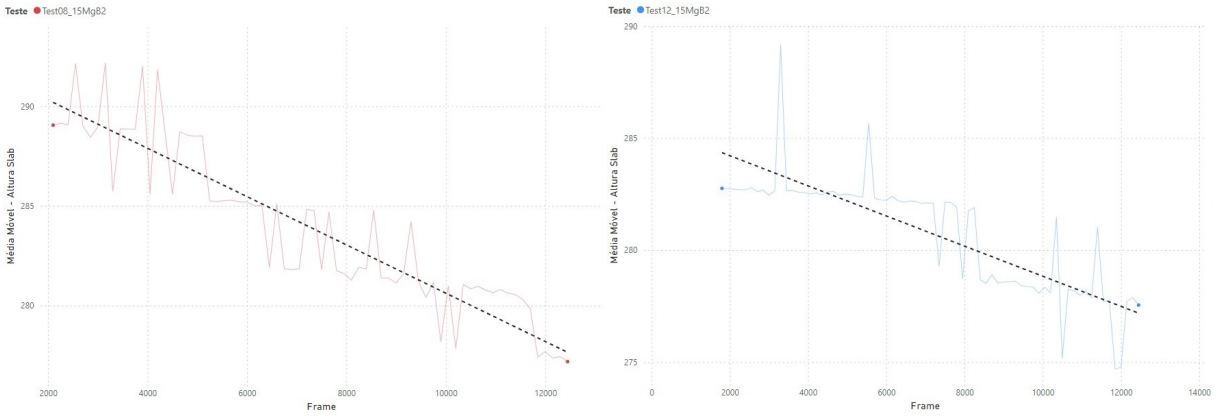


Figura 25 – Comparação da Média Móvel da Posição em Pixels x Frame do Combustível Sólido para Testes com 1% de MgB2

6.2.2.3 Parafina dopada com 1.5% de MgB2

Pela Figura 26a, o Teste 8 começa em 289,06 pixels e termina em 277,19 pixels, com uma taxa de regressão média de 2,29 px/s e uma variação percentual em módulo de 4,11%. Pela Figura 26b, o Teste 12 inicia em 282,76 pixels e finaliza em 277,55 pixels.



(a) (b)

Figura 26 – (a): Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 8 com 1,5% de MgB2 e (b):Média Móvel da Posição em Pixels x Frame do Combustível Sólido para o teste 12 com 1,5% de MgB2

A Figura 27 apresenta a média móvel da altura em pixels do combustível sólido ao longo dos testes 8 e 12 para MgB2 a 1,5%. A taxa de regressão média é de 0,98 px/s, correspondendo a uma variação percentual em módulo de 1,84%.

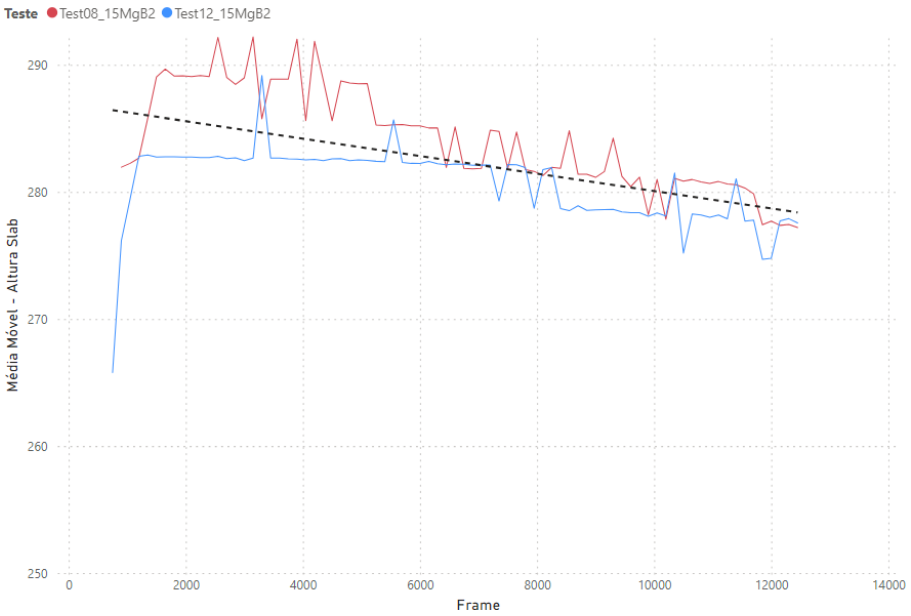


Figura 27 – Comparação da Média Móvel da Posição em Pixels x Frame do Combustível Sólido para Testes com 1,5% de MgB2

6.2.3 Comparação dos testes

A tabela 4 apresenta a compilação dos dados.

Aditivo	Mistura	Nº	Altura Inicial	Altura Final	Frame Inicial	Frame Final	Δ px	px/frame	px/s	Variação %
LiAlH	0,5%	1	282,84	269,21	4500	12450	13,63	1,71E-03	3,43	-4,82%
	0,5%	2	265,8	255,27	4200	12450	10,53	1,28E-03	2,55	-3,96%
	1,0%	3	282,48	277,56	4800	12000	4,92	6,83E-04	1,37	-1,74%
	1,0%	4	278,22	260,22	3900	12450	18	1,12E-03	4,21	-6,47%
	1,5%	5	269,54	259,45	4050	12450	10,09	1,20E-03	2,40	-3,74%
	1,5%	6	277,37	262,57	2250	12450	14,8	1,45E-03	2,90	-5,34%
MgB2	0,5%	7	275,59	264,29	2000	12450	11,3	1,11E-03	2,22	-4,10%
	0,5%	10	274,69	260,11	1200	12450	14,58	1,33E-03	2,66	-5,31%
	1,0%	9	272,52	258,24	1800	11850	14,28	1,42E-03	2,85	-5,24%
	1,0%	11	273,6	258,22	2250	12450	15,37	1,15E-03	2,93	-5,69%
	1,5%	8	289,06	279,22	4000	12450	10,84	8,79E-04	1,76	-3,79%
	1,5%	12	282,76	277,55	1800	12450	5,21	4,89E-04	0,98	-1,84%

Tabela 4 – Tabela de dados de diferentes misturas e tipos de aditivos.

A tabela 5 apresenta a compilação das médias de porcentagens de aditivos versus tipo de aditivo.

Concentração	**Média px/s**	
	LiAlH	**MgB2**
0,5%	2,99	2,44
1,0%	2,79	2,88
1,5%	2,65	1,64

Tabela 5 – Média dos valores de regressão (px/s) para diferentes concentrações de LiAlH e MgB2.

A Figura 28 apresenta uma comparação gráfica dos resultados da tabela 5, mostrando a velocidade média de regressão (px/s) para diferentes concentrações dos aditivos LiAlH e MgB2.

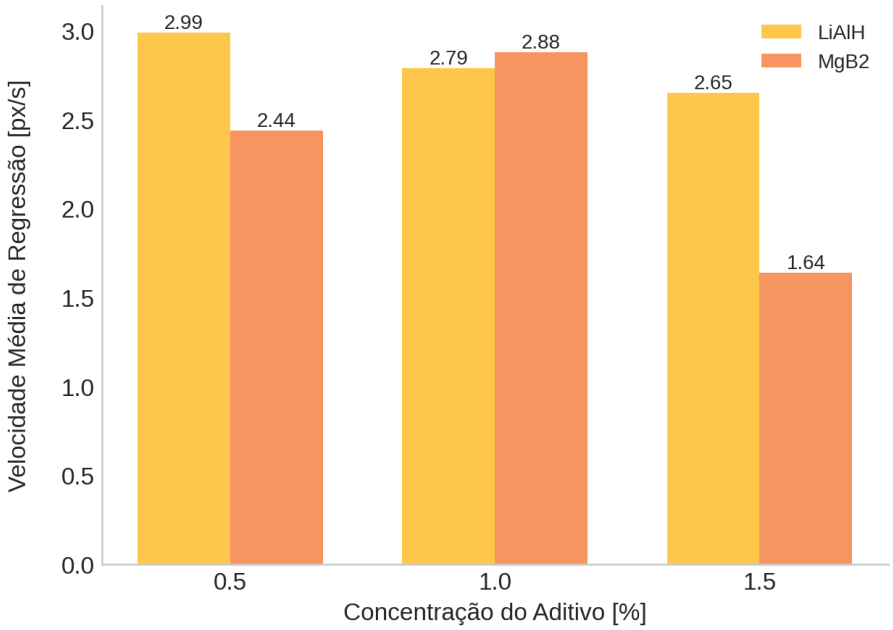


Figura 28 – Gráfico da Velocidade Média de Regressão versus Concentração do Aditivo para LiAlH e MgB2

7 Conclusão

A figura 12 mostra que com um dataset relativamente pequeno, composto por 159 imagens de treinamento, 47 imagens de validação e 25 imagens de teste, o modelo YOLOv8 conseguiu alcançar altas métricas de desempenho após 113 épocas de treinamento. Isso reforça ainda mais a eficácia do modelo em aprender padrões relevantes e generalizar bem, apesar da limitação no número de dados. A capacidade do YOLOv8 de manter altas métricas de precisão, recall e mAP em diferentes limiares, mesmo com poucos dados, sugere que ele está aproveitando bem o conjunto de dados disponível sem sofrer de overfitting, ao evitar uma grande divergência entre as métricas de treinamento e validação. Ao mesmo tempo, a estabilização das perdas em valores baixos indica que o modelo também não está subajustando, conseguindo capturar bem os padrões presentes no pequeno conjunto de dados. Para trabalhos futuros, podem tentar obter a conversão dessa velocidade de px/s para unidades físicas, como milímetros ou centímetros, e estudar a influência de outros aditivos e concentrações na busca por um desempenho ainda melhor dos combustíveis sólidos.

Neste capítulo, são apresentadas também as conclusões dos resultados experimentais sobre a velocidade de regressão dos combustíveis sólidos, medida em pixels por segundo (px/s), com diferentes concentrações dos aditivos LiAlH e MgB_2. Observando as Figuras 17, 19, 21 e 28, nota-se que a velocidade média de regressão para o LiAlH diminui de forma não linear à medida que a concentração aumenta: 2,99 px/s para 0,5%, 2,79 px/s para 1,0% e 2,65 px/s para 1,5%, conforme mostrado na Tabela 5 e representado graficamente na Figura 28. Para o MgB_2, conforme ilustrado nas Figuras 23, 25, 27 e 28, observa-se que a velocidade de regressão aumenta para 2,88 px/s com uma concentração de 1,0%, mas depois cai para 1,64 px/s com 1,5%. Esses dados sugerem que, com o aumento da concentração de LiAlH, há uma redução na velocidade de regressão, o que pode ser resultado de mudanças na estrutura do combustível ou nas suas propriedades de combustão. Em contraste, no caso do MgB_2, uma concentração de 1,0% parece otimizar a velocidade de regressão, como evidenciado nas Figuras 25 e 28. Ao analisar as variações na altura do combustível sólido, apresentadas na Tabela 4, observa-se que essas variações reforçam os resultados das velocidades em px/s. Foram registradas reduções percentuais que variaram de -1,74% a -6,47% para o LiAlH, e de -1,84% a -5,69% para o MgB_2. Diante disso, conclui-se que tanto o tipo quanto a concentração dos aditivos têm um impacto significativo na velocidade de regressão dos combustíveis sólidos.

Referências

R. GELAIN et al. Design and commissioning of the mouette hybrid rocket slab burner. Proceedings of the 9th European Conference for Aerospace Sciences. Lille, France, 27 June - 1 July, 2022, 2022. Disponível em: <https://www.eucass.eu/doi/EUCASS2022-6055.pdf>. Citado 6 vezes nas páginas 7, 11, 43, 44, 45 e 46.

AI, F. *Segment Anything: Generate Segmented Images from Textural Descriptions*. 2023. Disponível em: <https://github.com/facebookresearch/segment-anything?tab=readme-ov-file>. Citado 2 vezes nas páginas 9 e 89.

BERTOLDI, A.; GELAIN, R.; HENDRICK, P. Characterization of hybrid rocket paraffin-based fuels. In: . [S.l.: s.n.], 2022. Citado 4 vezes nas páginas 25, 27, 28 e 29.

GROUNDED SAM Google Colab. <https://colab.research.google.com/github/betogaona7/Grounded-Segment-Anything/blob/main/grounded_sam_colab_demo.ipynb>. [Accessed 09-12-2023]. Citado na página 98.

IDEA-RESEARCH. *Grounded-Segment-Anything README*. 2023. Disponível em: <https://github.com/IDEA-Research/Grounded-Segment-Anything/blob/main/README.md>. Citado 2 vezes nas páginas 9 e 89.

IDEA-RESEARCH. *GroundingDINO: Marrying Grounding-DINO with Segment Anything & Stable Diffusion & Recognize Anything*. 2023. Disponível em: <https://github.com/IDEA-Research/GroundingDINO>. Citado 2 vezes nas páginas 9 e 88.

IPython. *IPython.display Documentation*. 2024. Disponível em: <https://ipython.readthedocs.io/en/stable/api/generated/IPython.display.html>. Citado na página 35.

ipywidgets. *ipywidgets Documentation*. 2024. Disponível em: <https://ipywidgets.readthedocs.io/>. Citado na página 35.

KARABEYOGLU, A. et al. Modeling of hybrid rocket low frequency instabilities. *Journal of Propulsion and Power - J PROPUL POWER*, v. 21, p. 1107–1116, 11 2005. Citado na página 21.

KARABEYOGLU, M. A.; CANTWELL, B. J.; ZILLIAC, G. Development of scalable space-time averaged regression rate expressions for hybrid rockets. *Journal of Propulsion and Power*, American Institute of Aeronautics and Astronautics (AIAA), v. 23, n. 4, p. 737–747, jul. 2007. ISSN 1533-3876. Disponível em: <http://dx.doi.org/10.2514/1.19226>. Citado na página 26.

KIM, S. et al. Effect of paraffin-ldpe blended fuel on the hybrid rocket motor. In: . [S.l.: s.n.], 2010. ISBN 978-1-60086-958-7. Citado 5 vezes nas páginas 7, 11, 25, 26 e 27.

KIRILLOV, A. et al. Segment anything. *arXiv:2304.02643*, 2023. Citado na página 88.

LIU, S. et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023. Citado na página 87.

Matplotlib Developers. *Matplotlib Documentation*. 2024. Disponível em: <https://matplotlib.org/>. Citado na página 34.

NumPy. *NumPy Documentation*. 2024. Disponível em: <https://numpy.org/>. Citado na página 31.

OpenCV. *OpenCV Documentation*. 2024. Disponível em: <https://opencv.org/>. Citado na página 32.

Pandas. *pandas Documentation*. 2024. Disponível em: <https://pandas.pydata.org/>. Citado na página 37.

Python Imaging Library (PIL). *PIL Documentation*. 2024. Disponível em: <https://pillow.readthedocs.io/en/stable/>. Citado na página 36.

Python Software Foundation. *contextlib — Utilities for with-statements*. 2024. Disponível em: <https://docs.python.org/3/library/contextlib.html>. Citado na página 36.

Python Software Foundation. *io — Core tools for working with streams*. 2024. Disponível em: <https://docs.python.org/3/library/io.html>. Citado na página 36.

Python Software Foundation. *os — Miscellaneous operating system interfaces*. 2024. Disponível em: <https://docs.python.org/3/library/os.html>. Citado na página 38.

Python Software Foundation. *shutil — High-level file operations*. 2024. Disponível em: <https://docs.python.org/3/library/shutil.html>. Citado na página 38.

PyTorch. *PyTorch Documentation*. 2024. Disponível em: <https://pytorch.org/>. Citado na página 37.

Scikit-learn. *scikit-learn: Machine Learning in Python*. 2024. Disponível em: <https://scikit-learn.org/>. Citado na página 39.

SUTTON, G. P.; BIBLARZ, O. *Rocket Propulsion Elements*. [S.l.]: John Wiley & Sons, 2010. Citado 4 vezes nas páginas 7, 23, 24 e 29.

Ultralytics. *YOLOv8 Documentation*. 2024. Disponível em: <https://docs.ultralytics.com/>. Citado 3 vezes nas páginas 40, 41 e 42.

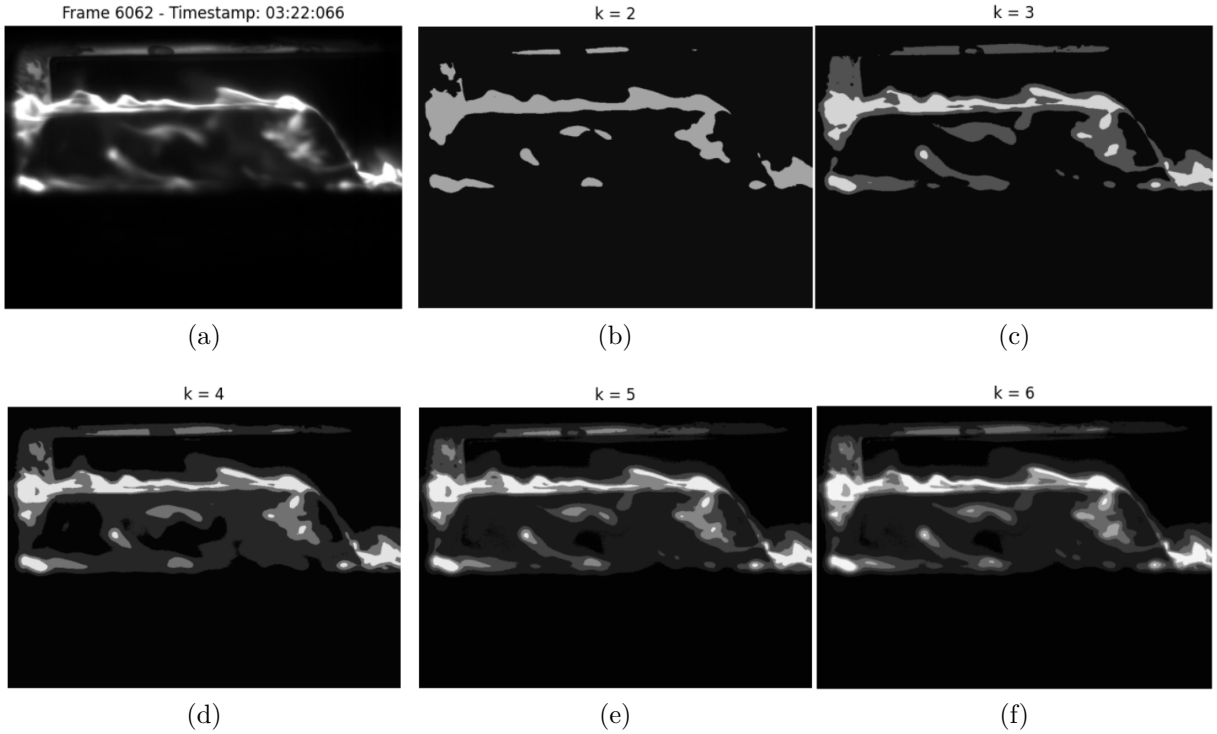
Apêndices

APÊNDICE A – Primeiro Apêndice

A.1 Resultados a partir da abordagem clássicas

A.1.1 Clusterização com cv2.Kmeans

A figura 29 exibe os resultados da clusterização a partir do vídeo do teste 8. Para esta análise, foi escolhido o frame 6062, equivalente ao minuto 03:22. Na figura 29a, a primeira imagem representa o frame original, enquanto as imagens subsequentes apresentam o mesmo frame porém dividido em um número variável de clusters, denotado por "k".

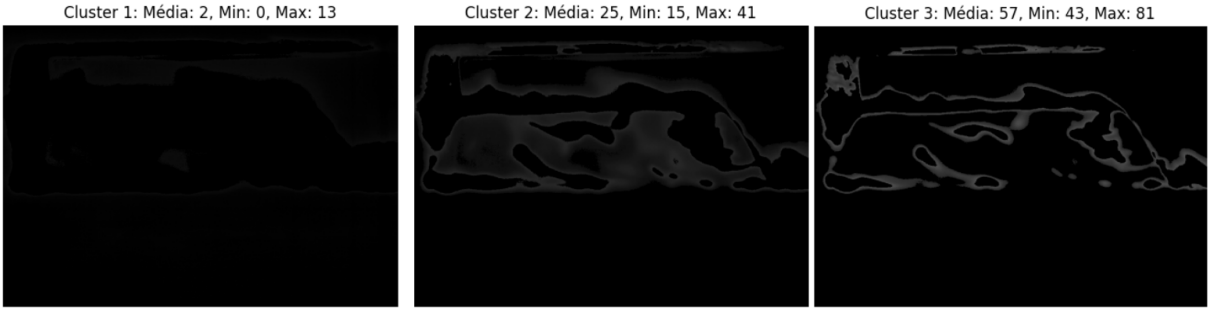


(a) (b) (c)

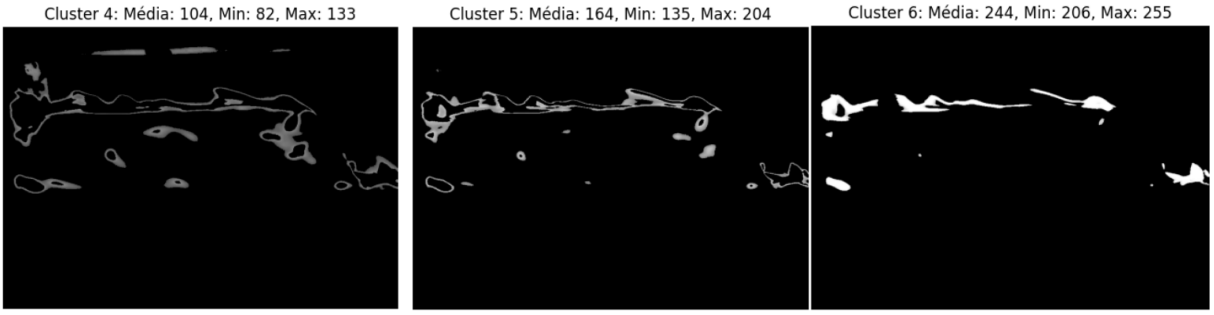
(d) (e) (f)

Figura 29 – Variações do frame 6062 do vídeo do teste 8 após incremento no número de clusters

A partir, dos clusters identificados na figura 29 para o frame 6062 do teste 8, selecionou-se o conjunto de dados para o número de clusters "k=6", apresentado na figura 29f . A figura 30, apresenta a segmentação e apresentação de cada um dos 6 clusters com suas respectivas intensidades médias, máximas e mínimas de cada cluster.

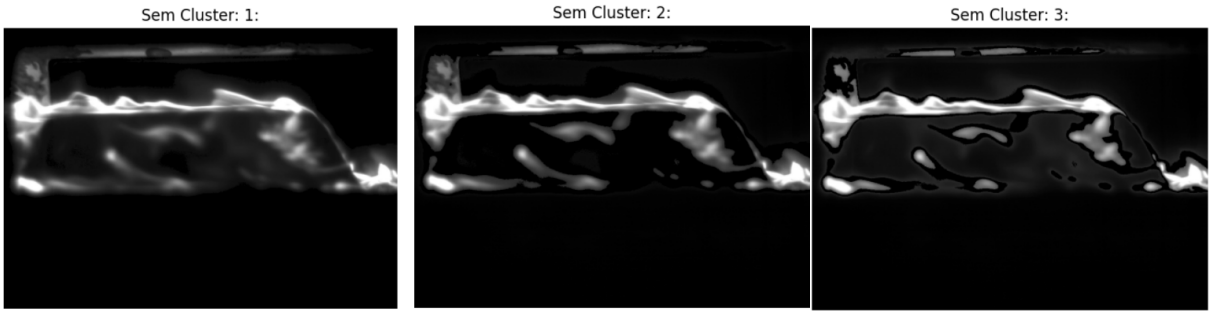


(a) (b) (c)

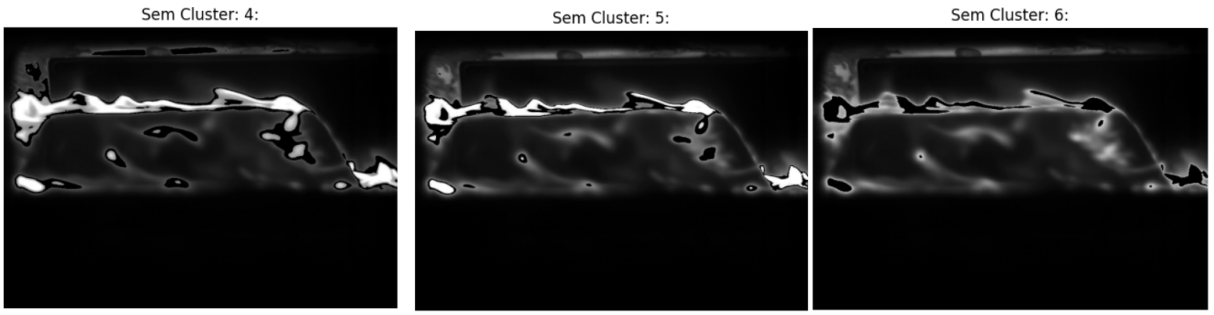


(d) (e) (f)

Figura 30 – Extração e apresentação individual dos clusters identificados na 29f, assim como suas respectivas intensidades médias, máximas e mínimas



(a) (b) (c)



(d) (e) (f)

Figura 31 – Resultados da remoção de cada um dos clusters da figura 30 removidos individualmente do frame 6062 original do teste 8

Com o intuito de analisar a possibilidade de aumentar a precisão da segmentação

do contorno combustível sólido através da remoção de clusters da chama, foi testado a remoção individual de cada um dos clusters, da figura 30, removidos do frame 6062 original do teste 8. A figura 31 apresenta os resultados desta tentativa.

A.2 Resultados da Abordagem Utilizando o Grounded Sam

Nesta seção, foram analisados dois testes utilizando o Grounded SAM para cada um dos tipos de combústivel sólido: parafina pura; e parafina adição de diboreto de magnésio (MgB2). Para cada teste, foram utilizados o primeiro e o último frame do vídeo correspondente como seus momentos inicial e final, respectivamente.

É importante relembrar-se que o Grounded SAM não foi treinado até o presente momento, todos são resultados "Zero-shot", isto é, resultados de classes de objetos detectados que não tiveram treinamento prévio e foram identificados somente pela semelhança em outras classes. Portanto, os resultados dos testes são de caráter preliminar.

A.2.1 Análise dos testes dos combústiveis sólidos de Parafina Pura

A.2.1.1 Teste 8

A figura 32 apresenta resultado do primeiro frame do vídeo do teste número 8, que possui composição apenas de parafina pura, analisado pelo modelo Grounded SAM:

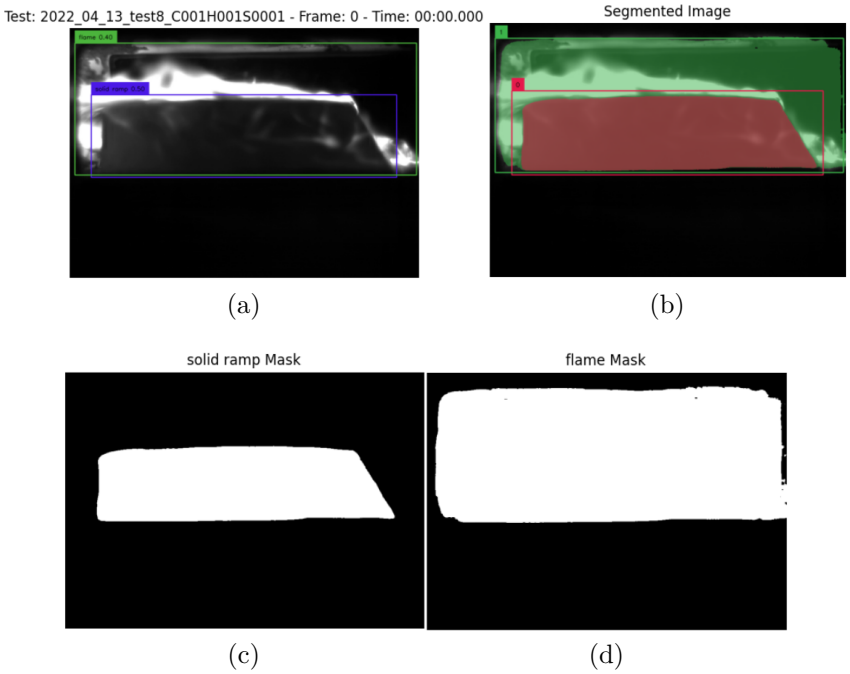
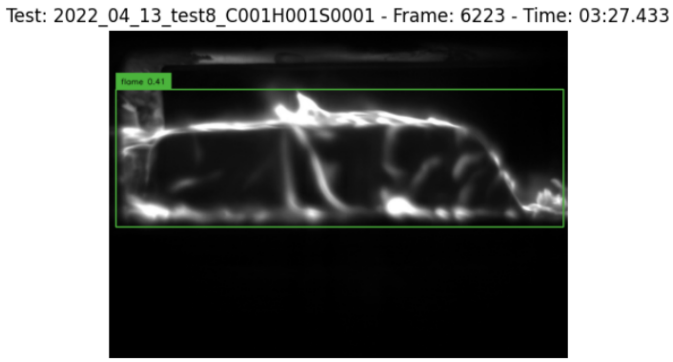


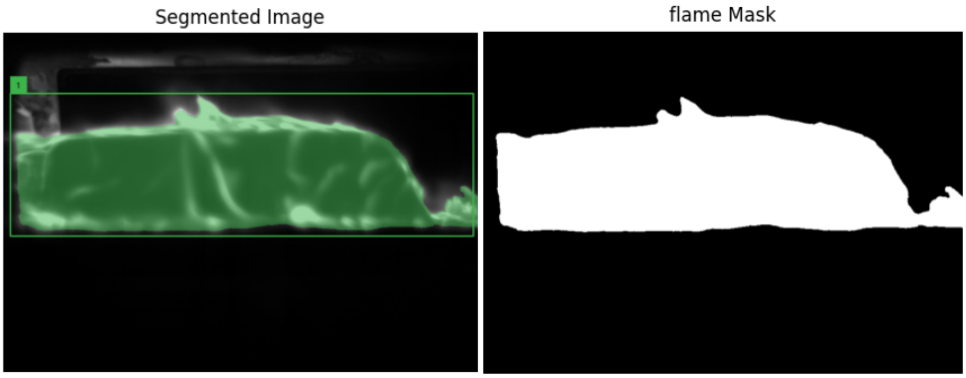
Figura 32 – Resultados do Grounded SAM para o frame 0 do teste 8

A figura 33 apresenta resultado do último frame (frame 6223) do vídeo do teste nú-

mero 8, que possui composição apenas de parafina pura, analisado pelo modelo Grounded SAM:



(a)



(b) (c)

Figura 33 – Resultados do Grounded SAM para o frame 6223 do teste 8

No primeiro frame do teste 8, pela figura 32c, é possível analisar qualitativamente que o combustível sólido (rotulado como "solid ramp") foi identificado com uma razoavelmente boa precisão, por outro lado, a chama não foi encontrada como uma boa acurácia.

No último frame do teste 8, pela figura 33c , no entanto, apesar de ter conseguido contornar com precisão o conjunto do combustível sólido e chama, não conseguiu distinguir ambos.

A.2.1.2 Teste 12

A figura 34 apresenta resultado do primeiro frame do vídeo do teste número 12, que possui composição apenas de parafina pura, analisado pelo modelo Grounded SAM:

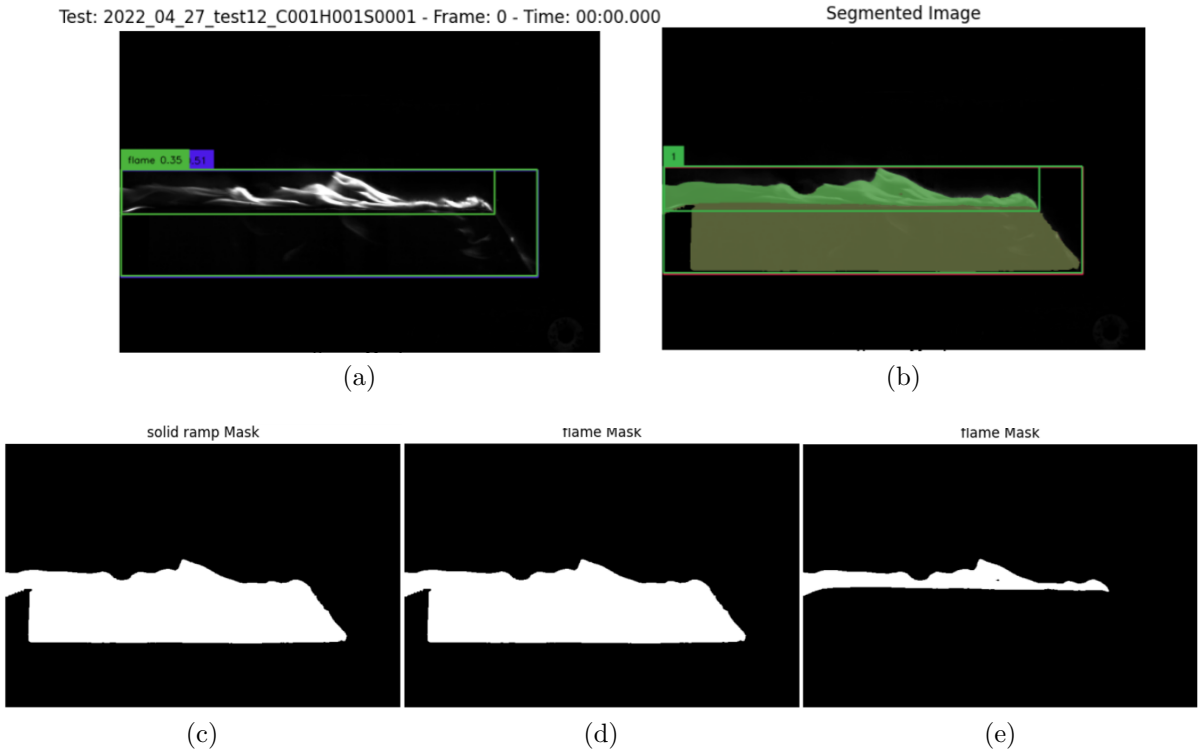


Figura 34 – Resultados do Grounded SAM para o frame 0 do teste 12

A figura 35 apresenta resultado do último frame (frame 14336) do vídeo do teste número 8, que possui composição apenas de parafina pura, analisado pelo modelo Grounded SAM:

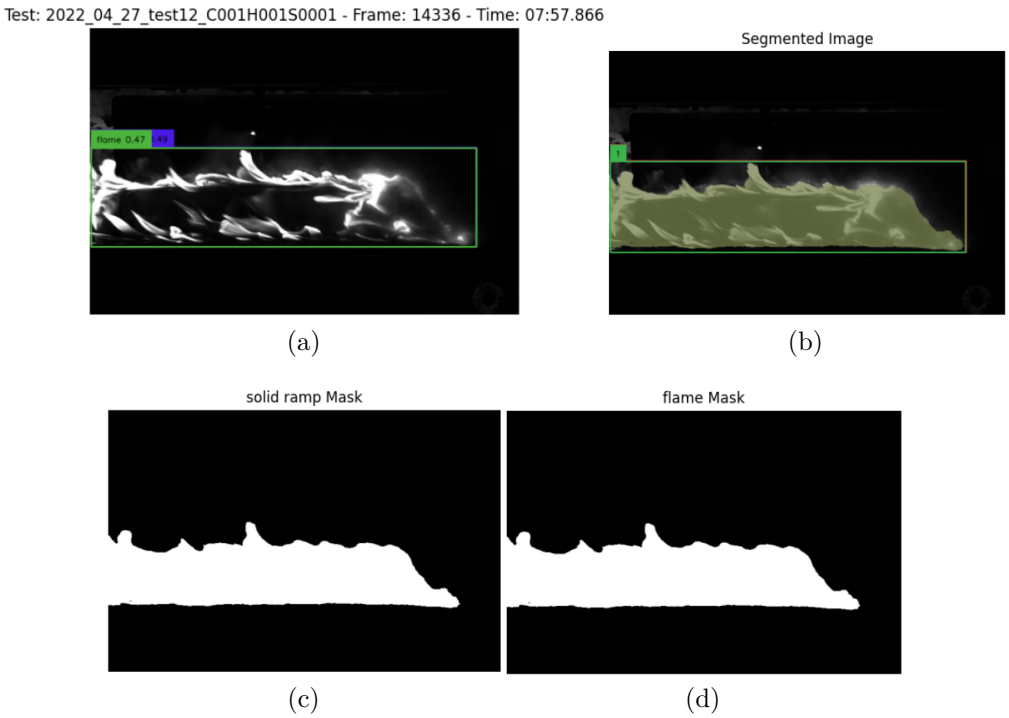


Figura 35 – Resultados do Grounded SAM para o frame 14336 do teste 12

No primeiro frame do teste 12, pela figura 34b, é possível analisar qualitativamente que a chama foi devidamente separada. No entanto, as outras detecções não conseguiram distinguir adequadamente as classes dentro do conjunto detectado.

No último frame do teste 12, conforme ilustrado na Figura 35c, o modelo não conseguiu distinguir ambas classes.

A.2.2 Análise dos testes dos combústiveis sólidos de Parafina com aditivo de diboreto de magnésio (MgB2)

A.2.2.1 Teste 17 (5%) aditivo de diboreto de magnésio (MgB2)

A figura 36 apresenta resultado do primeiro frame do vídeo do teste número 17, que possui composição apenas de parafina com aditivo 5% de diboreto de magnésio (MgB2), analisado pelo modelo Grounded SAM:

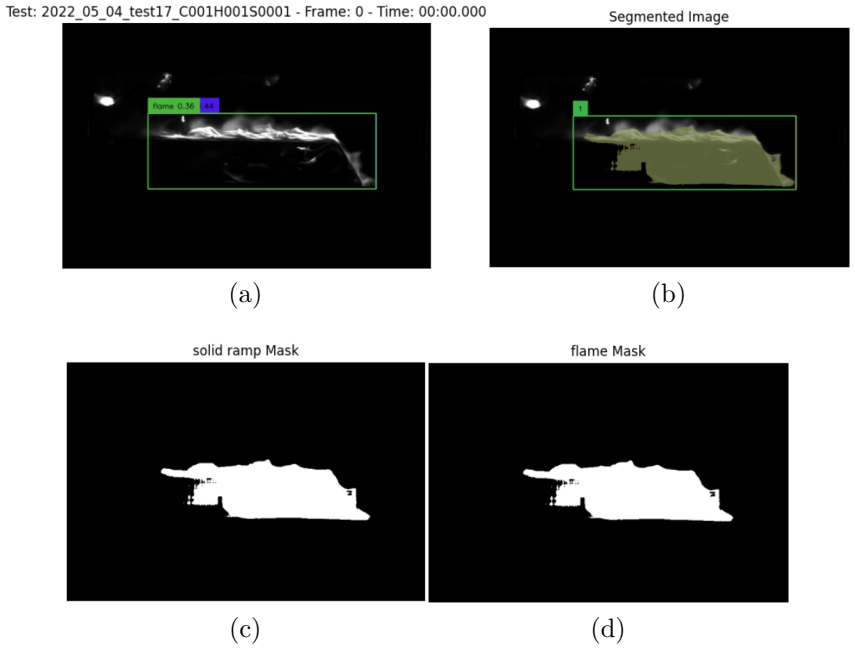
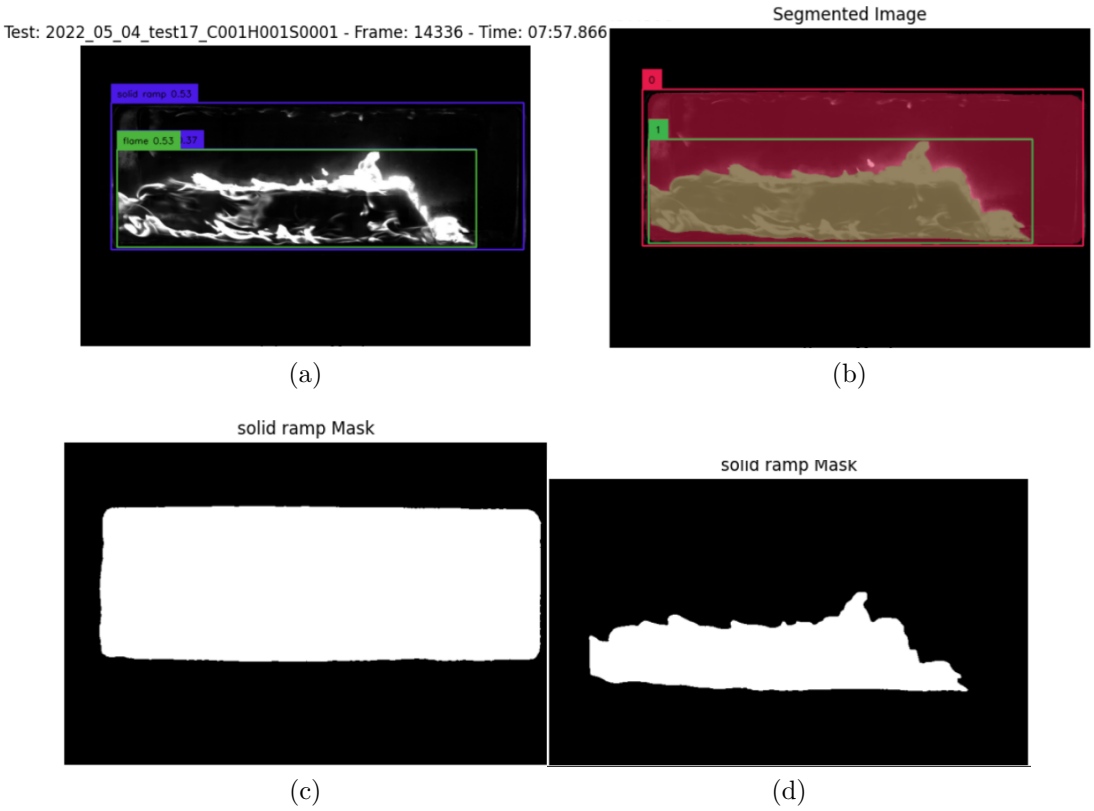


Figura 36 – Resultados do Grounded SAM para o frame 0 do teste 17

A figura 37 apresenta resultado do último frame (frame 14336) do vídeo do teste número 17, que possui composição apenas de parafina com aditivo 5% de diboreto de magnésio (MgB2), analisado pelo modelo Grounded SAM:



(a) (b)

(c) (d)

Figura 37 – Resultados do Grounded SAM para o frame 14336 do teste 17

No primeiro frame do teste 17, pela figura 36c, é possível analisar qualitativamente que nenhuma classe foi propriamente separada.

No último frame do teste 17, pela figura 37c , no entanto, apesar de ter conseguido contornar com precisão o conjunto do combustível sólido e chama, mas assim como resultado o último frame do teste 8, não conseguiu distinguir o combustível sólido e a chama.

A.2.2.2 Teste 13 - Parafina com (10%) aditivo de diboreto de magnésio (MgB2)

Para este teste especificamente, o primeiro frame analisado foi o frame 549 ao invés do frame 0. A justificativa para tal é devido ao fato que no frame 0 teste, a chama não alcançou o fim da geometria do combustível sólido, o que deixa a segmentação do primeiro instante limitada à chama.

A figura 38 apresenta resultado do frame 549 do vídeo do teste número 13, que possui composição apenas de parafina com aditivo 10% de diboreto de magnésio (MgB2), analisado pelo modelo Grounded SAM:

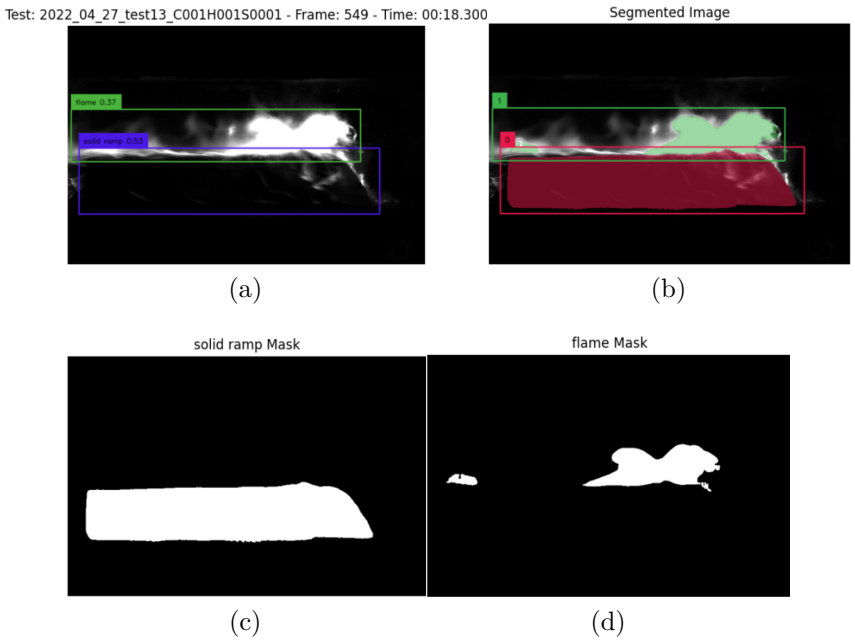


Figura 38 – Resultados do Grounded SAM para o frame 549 do teste 13

A figura 39 apresenta resultado do último frame (frame 14336) do vídeo do teste número 13, que possui composição apenas de parafina com aditivo 10% de diboreto de magnésio (MgB2), analisado pelo modelo Grounded SAM:

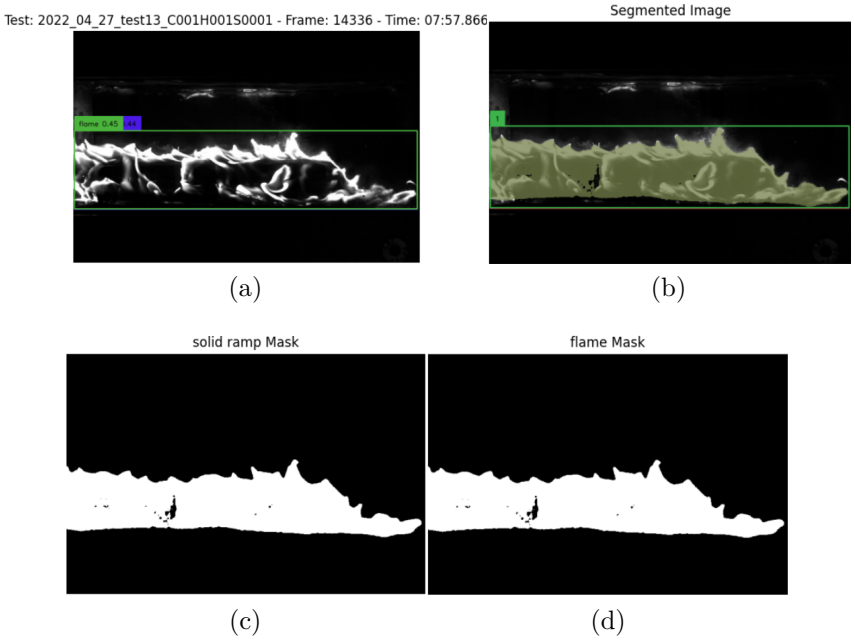


Figura 39 – Resultados do Grounded SAM para o frame 14336 do teste 13

No primeiro frame do teste 13, pelas figuras 38c e 38d, é possível analisar qualitativamente que respectivamente as classes "solid ramp"e "flame"foram identificadas com

uma excelente precisão levando em conta que o modelo ainda não foi treinado para a detecção especifica das classes analisadas no presente trabalho.

No último frame do teste 13, pela figura 39c, no entanto, apesar de ter conseguido contornar com precisão o conjunto do combustível sólido e chama, mas assim como o resultados de outros testes, ainda não conseguiu distinguir o combustível sólido e a chama.

Com o intuito de aprimorar a capacidade do modelo em distinguir entre o combustível sólido e a chama durante o treinamento, nas próximas etapas deste trabalho de conclusão de curso, será conduzido um estudo e implementado um treinamento personalizado. Essa abordagem envolverá o uso de imagens anotadas para cada uma das classes, ou seja, combustível sólido e chama. O objetivo principal é melhorar a capacidade do modelo em detectar e segmentar de maneira mais eficaz o combustível sólido e a chama, levando em consideração suas características específicas associadas ao fenômeno da combustão.

A.3 Conclusão dos Resultados Preliminares

A abordagem de clusterização, evidenciada nas figuras 29, 30 e 31 , apresentou bons resultados qualitativos para a segmentação de diferentes partes da chama. No entanto, a aplicação sistemática dessa abordagem para a segmentação da região do combustível sólido não é considerada interessante pelo autor e opta por não prosseguir com o uso exclusivo da abordagem de clusterização para a segmentação do combustível sólido. A justificativa para tal se deve ao fato de que a presença de chamas na área lateral do combustível sólido introduz um desafio significativo. Esse fenômeno contamina as métricas de contorno e dificulta a identificação clara e distinta da segmentação máscara do combustível sólido, prejudicando a confiabilidade dos resultados obtidos.

No entanto, esta abordagem pode ser interessante para a segmentação das diferentes regiões das chamas e seus centros, potencialmente para obtenção de uma estimativa da pressão ao longo da camara de combustão a partir da correlação da temperatura com o nível de intensidade dos pixels.

Por outro lado, a abordagem utilizando o modelo do Grounded SAM, evidenciada nas figuras 32, 33,34,35,36,37, 38, 39, apresenta resultados promissores até o presente momento. O frame 0 do teste 8, pela figura 33c apresentou uma boa segmentação para o combustível sólido. O frame 0 do teste 12, apresentou uma boa segmentação para a chama especificamente. O frame 549 do teste 13, pela figura 38c, apresentou uma excelente precisão tanto para a segmentação do combustível sólido quando para a chama. O autor gostaria de ressaltar que todos resultados foram obtidos do modelo Grounded SAM, que, até o presente momento, não foi submetido a um treinamento especifico para ambas classes. Este treinamento especifico está planejado no escopo da segunda parte deste trabalho de conclusão de curso.

APÊNDICE B – Segundo Apêndice

A visão computacional, um campo interdisciplinar entre ciência da computação e processamento de imagens, desempenha um papel vital na extração de informações significativas a partir de dados visuais. Este capítulo tem como objetivo apresentar uma breve contextualização dos métodos utilizados no presente trabalho.

B.1 Open CV

O OpenCV (Open Source Computer Vision) é uma biblioteca robusta e de código aberto que se tornou um pilar fundamental na caixa de ferramentas de profissionais e entusiastas da visão computacional. Esta seção explora os principais recursos do OpenCV e seu papel essencial na implementação de técnicas avançadas de processamento de imagem.

B.1.1 cv2.Threshold

A função threshold no OpenCV é usada para aplicar uma operação de limiarização a uma imagem. A limiarização é um processo no qual os pixels de uma imagem são categorizados como brancos ou pretos, com base em se seus valores de intensidade estão acima ou abaixo de um determinado limiar.

Listing B.1 – cv2.Threshold

```
retval, dst = cv2.threshold(gray_image, thresh, maxval, type)
```

- **gray_image**: A imagem de entrada, que deve ser uma imagem de escala de cinza.
- **thresh**: O valor do limiar. Pixels com intensidades abaixo desse valor serão atribuídos a 0 (preto), e pixels com intensidades acima desse valor serão atribuídos a *maxval* (branco).
- **maxval**: O valor atribuído aos pixels que ultrapassam o limiar.
- **type**: O tipo de operação de limiarização a ser realizada. Existem vários tipos, como `cv2.THRESH_BINARY`, `cv2.THRESH_BINARY_INV`, `cv2.THRESH_TRUNC`, `cv2.THRESH_TOZERO`, e `cv2.THRESH_TOZERO_INV`. Cada tipo realiza uma operação de limiarização diferente.

B.1.1.1 Método de Otsu

A função `cv2.threshold()` é usada para segmentar uma imagem convertendo-a em uma imagem binária com base em um valor de limiar. A utilização do método de otsu em python costuma ter a seguinte forma:

Listing B.2 – cv2.Threshold

```
valor_otsu, thresholdedImagem = cv2.threshold(gray_image, 0,
    255, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
```

- `gray_image`: A imagem de entrada já convertida em tons de cinza.
- `0`: Este parametro valor de limiar inicial que não é usado no método de Otsu.
- `255`: O valor máximo que será atribuído aos pixels na imagem binária resultante. Pixels com intensidade acima do valor de limiar serão definidos como este valor.
- `cv2.THRESH_BINARY`: Um dos possíveis métodos de limiarização. Ao utilizar-lo a limiarização será do tipo binário, ou seja, pixels com intensidade acima do valor de limiar serão definidos para o valor máximo (255 neste caso), enquanto os pixels abaixo do valor de limiar serão definidos para zero.
- `cv2.THRESH_OTSU`: Este é um sinalizador adicional que indica que o valor de limiar deve ser calculado automaticamente usando o método de Otsu. Quando este sinalizador é usado, o valor de limiar passado como segundo parâmetro (`0` neste caso) é ignorado.

A função retorna dois valores:

valorOtsu: O valor de limiar ótimo do método de otsu.
thresholdedImagem: A imagem após a segmentação pela aplicação da limiarização do método de otsu.

B.1.2 cv2.findContours

A função findContours da biblioteca OpenCV é uma ferramenta para detecção e extração de contornos em imagens.

Listing B.3 – cv2.findContours

```
contours, hierarchy = cv2.findContours(gray_image, mode, method)
```

gray_image: A imagem de entrada já em tons de cinza.

mode: Define a forma como os contornos são organizados. Existem quatro opções principais:

- `cv2.RETR_EXTERNAL`: Apenas contornos externos são retornados.
- `cv2.RETR_LIST`: Todos os contornos são retornados, mas sem hierarquia.
- `cv2.RETR_CCOMP`: Todos os contornos são retornados e organizados em duas camadas hierárquicas.
- `cv2.RETR_TREE`: Todos os contornos e suas hierarquias completas são retornados.

method: Define como os contornos são aproximados e armazenados. Duas opções principais são:

- `cv2.CHAIN_APPROX_SIMPLE`: Simplificação simples. Armazena apenas os pontos finais
- `cv2.CHAIN_APPROX_TC89_L1`: Aproximação Tangent-Chord com critério Taubin
- `cv2.CHAIN_APPROX_TC89_KCOS`: Aproximação Tangent-Chord com critério Karni-Cohen-Or
- `cv2.CHAIN_APPROX_NONE`: Armazena todos os pontos do contorno

A função retorna dois valores:

contours: Uma lista de contornos, onde cada contorno é uma matriz de coorndenadas.

hierarchy: Uma matriz que descreve a relação de hierarquia entre os contornos.

B.1.3 cv2.kmeans

Listing B.4 – cv2.kmeans

```
compactness, labels, centers = cv2.kmeans(dadosImagem, k, None,
    criteria, n, center_methods)
```

- **`dadosImagem`**: Dados da imagem de entrada. Cada linha representa um ponto de dados.
- **`k`**: O número de clusters escolhido. Deve ser um número inteiro positivo.
- **`centers`**: Inicialização dos centros dos clusters. Pode ser `None` para uma inicialização aleatória ou uma matriz inicializada manualmente.
- **`criteria`**: Critérios de término que determinam quando o algoritmo deve parar. Pode ser uma tupla do tipo `(type, max_iter, epsilon)`:

  - `type`: Tipo de critério, que pode ser `cv2.TERM_CRITERIA_EPS` para parar quando a precisão (epsilon) for atingida ou `cv2.TERM_CRITERIA_MAX_ITER` para parar após um número máximo de iterações.
  - `max_iter`: Número máximo de iterações.
  - `epsilon`: Precisão desejada.

- **`n`**: Número de vezes que o algoritmo é executado usando diferentes centros iniciais. O resultado com a menor compactação é escolhido.

- **`center_methods`**: Método para inicialização dos centros dos clusters. Pode ser um dos seguintes:

  - `cv2.KMEANS_RANDOM_CENTERS`: Inicializa os centros de maneira aleatória a partir dos pontos de dados.
  - `cv2.KMEANS_PP_CENTERS`: Utiliza o método do k-means++ para inicializar os centros de forma mais eficiente e robusta.

A função retorna 3 valores:

- **`compactness`**: Representa a soma das distâncias quadradas de cada ponto de dados ao seu centro de cluster atribuído. Quanto menor o valor, mais compactos são os clusters.

- **`labels`**: Uma matriz que contém as atribuições de cluster para cada ponto de dados. Cada elemento representa a qual cluster o ponto pertence.

- **`centers`**: Uma matriz que contém as coordenadas finais dos centros dos clusters. Cada linha representa as coordenadas de um centro de cluster.

B.2 Estado da Arte da Visão Computacional

B.2.1 Machine Learning x Inteligência Artificial

Inteligência artificial (IA) e aprendizado de máquina (ML) são dois campos da ciência da computação intimamente relacionados, mas com diferenças importantes [1]:

- Escopo: IA é um campo mais abrangente, enquanto ML é uma subárea específica focada em aprendizagem computacional.

- Foco: IA visa imitar a inteligência humana em diversos aspectos, enquanto ML se concentra na capacidade de aprender com dados.

[1] Fonte: `https://cloud.google.com/learn/artificial-intelligence-vs-machine-learning?hl=pt-br`

- Técnicas: IA utiliza diversas técnicas, incluindo heurísticas, lógica simbólica e aprendizado de máquina. Já ML se baseia principalmente em algoritmos de aprendizado.
- Aplicações: IA tem aplicações em áreas como robótica, jogos, sistemas de diagnóstico médico e financeiros. ML é amplamente utilizado em áreas como recomendação de produtos, detecção de fraudes e análise de imagens.

B.2.2 Modelos de Inteligência Artificial para Anotação e Segmentação de Imagens

B.2.2.1 Grounding DINO

Grounding DINO (LIU et al., 2023) é um novo modelo de detecção de objetos que é capaz de detectar objetos que não foram explicitamente treinados no modelo, uma capacidade conhecida como "Zero-Shot". Possui arquitetura composta por duas redes neurais, uma para imagens e outra para texto. As duas redes são treinadas juntas para aprender a relacionar imagens e texto. Além disso, Grounding DINO pode distinguir entre objetos conhecidos e desconhecidos, uma capacidade conhecida como detecção de objetos em aberto.

B.2.2.1.1 Vantagens de Grounding DINO

Grounding DINO oferece várias vantagens em relação aos modelos de detecção de objetos tradicionais, incluindo:

- Zero-Shot: Grounding DINO pode detectar objetos que não foram explicitamente treinados no modelo. Isso significa que ele pode ser usado para identificar objetos novos ou desconhecidos.
- Detecção de objetos em aberto: Grounding DINO pode distinguir entre objetos conhecidos e desconhecidos. Isso é importante para aplicações que exigem que o modelo seja capaz de lidar com objetos que não foram vistos antes.
- Alta precisão: Grounding DINO alcança estado da arte em vários benchmarks de detecção de objetos.

Grounding DINO tem o potencial de ser aplicado em uma ampla gama de domínios, incluindo:

- Visão Computacional em Geral: pode ser usado para melhorar a precisão da detecção de objetos em cenários com objetos não treinado.

- Robótica: pode ser usado para permitir que robôs identifiquem objetos novos ou desconhecidos em seu ambiente.

- Análise de imagens médicas: pode ser usado para identificar anomalias e anormalidades em imagens médicas, mesmo quando elas não foram vistas antes.

- Busca visual e recuperação: pode ser usado para pesquisar imagens com base em descrições textuais.

A figura 40 apresenta um exemplo de utilização do Grounding DINO. Neste exemplo, foi solicitado ao modelo a detecção dos objetos correspondentes às classes "dog"(cachorro) e "cake"(bolo).

Figura 40 – Exemplo de uso do Grounding Dino: solicitado ao modelo dectecção das classes "dog"(cachorro) e "cake"(bolo). . Imagem adaptada de (IDEA-RESEARCH, 2023b)

B.2.2.2 Segment Anything Model

O *Segment-Anything Model (SAM)* (KIRILLOV et al., 2023) é um modelo de inteligência artificial capaz de segmentar imagens com dados de qualquer tipo, independentemente de seu formato ou conteúdo. O SAM é baseado na ideia de usar um único modelo para aprender como segmentar qualquer tipo de dado.

A figura 41 apresenta um exemplo de utilização do Segment-Anything-Model. Neste exemplo, uma imagem de um trem de linha foi fornecida ao modelo, resultando na geração de uma nova imagem com todas as máscaras geradas sobrepostas à imagem original.

Figura 41 – Exemplo de uso do Segment Anything Model: imagem orginal de um trem de linha em uma via pública, com mascaras geradas e sobrepostas à iagem original. Imagem de (AI, 2023)

B.2.2.3 Grounded SAM

O Grounded SAM (Grounded Segment Anything) (IDEA-RESEARCH, 2023a) é um modelo que combina as vantagens dos seguintes modelos:

a. Grounding-DINO: Alta precisão para detecção e rotulação de objetos em uma imagem.

b. Segment Anything (SAM): Alta capacidade de segmentação e geração de máscaras para os objetos já detectados previamente.

A figura 42 ilustra um exemplo de aplicação do Grounded SAM. Neste caso específico, solicitou-se ao modelo a detecção da classe 'bears' (ursos), e tendo como resultado imagem original com as classes anotadas, além das máscaras correspondentes para cada uma das instâncias detectadas.

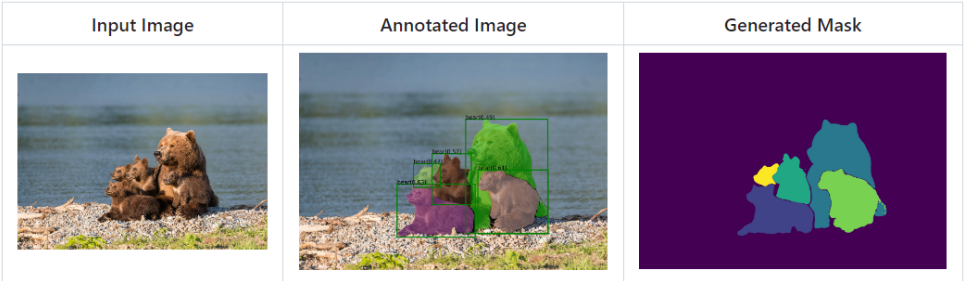


Figura 42 – Exemplo de Uso do Grounded SAM: Anotação e geração das mascaras para cada instância da classe detectada "bears"(ursos). Imagem de (IDEA-RESEARCH, 2023a)

B.2.2.3.1 Características e Realizações do Grounded SAM

- **Alta Precisão:** Na competição Segmentation in the Wild (CVPR 2023), o Grounded SAM alcançou uma média impressionante de 46.0 AP na categoria de aprendizado sem supervisão.
- **Versatilidade:** O Grounded SAM pode ser aplicado a diversas tarefas, incluindo anotação de imagens, reconhecimento de objetos, geração de imagens e mais.
- **Código Aberto:** O código-fonte do Grounded SAM está disponível no GitHub, permitindo que pesquisadores e desenvolvedores explorem e ampliem suas capacidades.

Anexos

ANEXO A – Códigos TCC1

Listing A.1 – FunÃğÃţes Principais

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from ipywidgets import interact, widgets
from IPython.display import display

def mostrar(imagem, imagem_cinza=False):
  fig = plt.gcf()
  fig.set_size_inches(18,6)
  if imagem_cinza:
    plt.imshow(imagem, cmap='gray')
  else:
    plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB), cmap='
       gray')
  plt.axis('off')
  plt.show()

def convert_milliseconds_to_time(milliseconds):
    total_seconds = milliseconds / 1000.0
    minutes, seconds = divmod(total_seconds, 60)
    return f"{int(minutes):02}:{int(seconds):02}:{int((
       total_seconds - int(total_seconds)) * 1000):03}"

def display_frame_from_video(video_path, frame_number):
    cap = cv2.VideoCapture(video_path)

    if frame_number == -1:
        frame_number = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) -
           1

    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)

    ret, frame = cap.read()
```

```
    timestamp_ms = cap.get(cv2.CAP_PROP_POS_MSEC)
    timestamp_str = convert_milliseconds_to_time(timestamp_ms)

    frame_label_time=f'Frame {frame_number} - Timestamp: {
       timestamp_str}'


    return frame, frame_number,timestamp_str,frame_label_time

def plot_images_dynamic(images, titles=None, cmap=None):
    if titles is None:
        titles = ['Image {}'.format(i + 1) for i in range(len(
           images))]

    if cmap is None:
        cmap = "gray"

    num_images = len(images)
    num_cols = int(np.ceil(np.sqrt(num_images)))
    num_rows = int(np.ceil(num_images / num_cols))

    plt.figure(figsize=(15, 7))

    for i, (image, title) in enumerate(zip(images, titles), 1):
        plt.subplot(num_rows, num_cols, i)
        if cmap is not None:
            plt.imshow(image, cmap=cmap)
        else:
            plt.imshow(image)
        plt.title(title)
        plt.axis('off')

    plt.tight_layout()
    plt.show()

def apply_blur(frame, method=None, ksize=5, sigmaX=0, d=0,
   sigmaColor=0, sigmaSpace=0):
    image = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
```

```
    if method is None:
        return image

    if method == 'average':
        blurred = cv2.blur(image, (ksize, ksize))
    elif method == 'gaussian':
        blurred = cv2.GaussianBlur(image, (ksize, ksize), sigmaX
            )
    elif method == 'median':
        blurred = cv2.medianBlur(image, ksize)
    elif method == 'bilateral':
        blurred = cv2.bilateralFilter(image, d, sigmaColor,
            sigmaSpace)
    elif method == 'box':
        blurred = cv2.boxFilter(image, -1, (ksize, ksize))
    elif method == '':
        blurred = image
    else:
        raise ValueError("Invalid blur method. Supported methods
            : 'average', 'gaussian', 'median', 'bilateral', 'box
            '")

    return blurred

from google.colab import drive
drive.mount('/content/drive')
```

Listing A.2 – CÃşdigo utilizado para ClusterizaÃğÃčo

```
\# @title SegmentaÃğÃčo em Clusters { run: "auto", vertical-
    output: true }
Select_Test = "2022_04_13_test8_C001H001S0001" # @param ["2022
    _04_13_test8_C001H001S0001", "2022_04_13_test9_C001H001S0001
    ", "2022_04_27_test11_C001H001S0001", "2022
    _04_27_test12_C001H001S0001", "2022
    _04_27_test13_C001H001S0001", "2022
    _05_04_test16_C001H001S0001", "2022
    _05_04_test17_C001H001S0001", "2022
```

```
    _05_04_test18_C001H001S0001"]


def segmentacao_k_cluster(img, k):
    img_vetorizada = np.float32(img).reshape(-1, 3)
    criterio = (cv2.TERM_CRITERIA_EPS + cv2.
       TERM_CRITERIA_MAX_ITER, 20, 1.0)
    ret, label, centros = cv2.kmeans(img_vetorizada, k, None,
       criterio, 10, cv2.KMEANS_RANDOM_CENTERS)
    centros = np.uint8(centros)
    img_final = centros[label.flatten()]
    img_final = img_final.reshape(img.shape)

    masks = []
    contours = []
    for i in range(k):
        mask = np.uint8(label == i).reshape(img.shape[:2])
        masks.append(mask)

        contour, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL,
           cv2.CHAIN_APPROX_SIMPLE)
        contours.append(contour)

    segmented_images = []
    for i, mask in enumerate(masks):
        segmented_img = cv2.bitwise_and(img, img, mask=mask)
        segmented_images.append(segmented_img)

    statistics = [{'avg': int(np.mean(img[mask > 0])), 'min': np
       .min(img[mask > 0]), 'max': np.max(img[mask > 0])} for
       mask in masks]

    sorted_clusters = sorted(range(len(statistics)), key=lambda
       k: statistics[k]['avg'])
    segmented_images = [cv2.bitwise_and(img, img, mask=masks[i])
        for i in sorted_clusters]
    masks = [masks[i] for i in sorted_clusters]

    statistics= [statistics[i] for i in sorted_clusters]
```

```
    segmented_images_inverse = [np.zeros_like(img) for _ in
        range(k)]
    for i, mask in enumerate(masks):
        segmented_images_inverse[i][mask == 0] = img[mask == 0]


    return img_final, masks, contours, segmented_images,
        segmented_images_inverse, statistics

def clustering(img,frame_label):
  titulos = [frame_label]
  imagens = [img]
  masks=  [img]
  contours=[img]

  cluster_title=[]

  clusters_removed_title=[]

  segmentacoes = 6
  for k in range(2, segmentacoes + 1):
    titulo = 'k = ' + str(k)
    titulos.append(titulo)
    segmented_img, mask, contour, segmented_images,
        segmented_images_inverse, statistics =
        segmentacao_k_cluster(img, k)
    imagens.append(segmented_img)
    masks.append(mask)
    contours.append(contour)

  for i, stats in enumerate(statistics):
    cluster_title.append(f'Cluster {i + 1}: MÃ©dia: {stats["avg
        "]}, Min: {stats["min"]}, Max: {stats["max"]}')
    clusters_removed_title.append(f'Sem Cluster: {i + 1}:')


  plot_images_dynamic(imagens, titulos)
```

```
    plot_images_dynamic(segmented_images, cluster_title)

    plot_images_dynamic(segmented_images_inverse,
        clusters_removed_title)



def process_frame(video_path,frame_number):

    frame, frame_number,timestamp_str,frame_label_time=
        display_frame_from_video(video_path,frame_number)

    clustering(frame,frame_label_time)


video_path = f'/content/drive/MyDrive/Videos_Tests_TCC1/{
    Select_Test}.mp4'
frame_number_default = 0
threshold_value_default = 127

cap = cv2.VideoCapture(video_path)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
cap.release()

frame_slider = widgets.IntSlider(value=frame_number_default, min
    =0, max=total_frames-1, step=1, description='Frame:')


display(
    interact(
        process_frame,
        video_path=widgets.fixed(video_path),
        frame_number=frame_slider

    )
)
```

O cÃşdigo A.3 e A.4 foram adaptados de (GROUNDED...,).

Listing A.3 – Setup Inicial do Grounded SAM

```
!nvidia-smi
import os
HOME = os.getcwd()
print("HOME:", HOME)
%cd {HOME}
!git clone https://github.com/IDEA-Research/GroundingDINO.git
%cd {HOME}/GroundingDINO
!git checkout -q 57535c5a79791cb76e36fdb64975271354f10251
!pip install -q -e .
from google.colab import drive
drive.mount('/content/drive')
%cd {HOME}

import sys
!{sys.executable} -m pip install 'git+https://github.com/
    facebookresearch/segment-anything.git'
!pip uninstall -y supervision
!pip install -q supervision==0.6.0

import supervision as sv
print(sv.__version__)

import os

GROUNDING_DINO_CONFIG_PATH = os.path.join(HOME, "GroundingDINO/
    groundingdino/config/GroundingDINO_SwinT_OGC.py")
print(GROUNDING_DINO_CONFIG_PATH, "; exist:", os.path.isfile(
    GROUNDING_DINO_CONFIG_PATH))
%cd {HOME}
!mkdir -p {HOME}/weights
%cd {HOME}/weights

!wget -q https://github.com/IDEA-Research/GroundingDINO/releases
    /download/v0.1.0-alpha/groundingdino_swint_ogc.pth
import os

GROUNDING_DINO_CHECKPOINT_PATH = os.path.join(HOME, "weights", "
    groundingdino_swint_ogc.pth")
```

```
print(GROUNDING_DINO_CHECKPOINT_PATH, "; exist:", os.path.isfile
    (GROUNDING_DINO_CHECKPOINT_PATH))
%cd {HOME}
!mkdir -p {HOME}/weights
%cd {HOME}/weights

!wget -q https://dl.fbaipublicfiles.com/segment_anything/
    sam_vit_h_4b8939.pth
import os

SAM_CHECKPOINT_PATH = os.path.join(HOME, "weights", "
    sam_vit_h_4b8939.pth")
print(SAM_CHECKPOINT_PATH, "; exist:", os.path.isfile(
    SAM_CHECKPOINT_PATH))
import torch

DEVICE = torch.device('cuda' if torch.cuda.is_available() else '
    cpu')
%cd {HOME}/GroundingDINO

from groundingdino.util.inference import Model

grounding_dino_model = Model(model_config_path=
    GROUNDING_DINO_CONFIG_PATH, model_checkpoint_path=
    GROUNDING_DINO_CHECKPOINT_PATH)

SAM_ENCODER_VERSION = "vit_h"

from segment_anything import sam_model_registry, SamPredictor

sam = sam_model_registry[SAM_ENCODER_VERSION](checkpoint=
    SAM_CHECKPOINT_PATH).to(device=DEVICE)
sam_predictor = SamPredictor(sam)

import numpy as np
from segment_anything import SamPredictor


def segment(sam_predictor: SamPredictor, image: np.ndarray, xyxy
```

```
    : np.ndarray) -> np.ndarray:
     sam_predictor.set_image(image)
     result_masks = []
     for box in xyxy:
         masks, scores, logits = sam_predictor.predict(
             box=box,
             multimask_output=True
         )
         index = np.argmax(scores)
         result_masks.append(masks[index])
     return np.array(result_masks)
```

Listing A.4 – Grounded SAM

```
# @title Resultados - Utilizando Grounding Dino e SAM { run: "
    auto", form-width: "0.5px" }
Test_Number = "2022_04_27_test12_C001H001S0001" # @param ["2022
    _04_13_test8_C001H001S0001", "2022_04_13_test9_C001H001S0001
    ", "2022_04_27_test11_C001H001S0001", "2022
    _04_27_test12_C001H001S0001", "2022
    _04_27_test13_C001H001S0001", "2022
    _05_04_test16_C001H001S0001", "2022
    _05_04_test17_C001H001S0001", "2022
    _05_04_test18_C001H001S0001"]
Class1 = "solid ramp" # @param {type:"string"}
Class2 = "flame" # @param {type:"string"}
VIDEO_PATH = f'/content/drive/MyDrive/ "_________" 1/
    Videos_Tests_TCC1/{Test_Number}.mp4'

from segment_anything import sam_model_registry, SamPredictor

sam = sam_model_registry[SAM_ENCODER_VERSION](checkpoint=
    SAM_CHECKPOINT_PATH).to(device=DEVICE)
sam_predictor = SamPredictor(sam)

import cv2
import numpy as np
from matplotlib import pyplot as plt
from ipywidgets import interact, widgets
```

```
from IPython.display import display

import math


import numpy as np
from segment_anything import SamPredictor


CLASSES = [Class1, Class2]

def enhance_class_name(class_names):
    return [f"all {class_name}s" for class_name in class_names]

def convert_milliseconds_to_time(milliseconds):
    total_seconds = milliseconds / 1000.0
    minutes, seconds = divmod(total_seconds, 60)
    return f"{int(minutes):02}:{int(seconds):02}.{int((
       total_seconds - int(total_seconds)) * 1000):03}"

def display_frame_and_segment(frame_number, box_threshold,
   text_threshold):
    cap = cv2.VideoCapture(VIDEO_PATH)

    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)

    ret, frame = cap.read()

    if not ret:
        raise Exception(f"Failed to read frame {frame_number}
           from the video.")

    timestamp_ms = cap.get(cv2.CAP_PROP_POS_MSEC)
    timestamp_str = convert_milliseconds_to_time(timestamp_ms)

    detections = grounding_dino_model.predict_with_classes(
        image=frame,
        classes=enhance_class_name(class_names=CLASSES),
        box_threshold=box_threshold,
```

```
    text_threshold=text_threshold
)

box_annotator = sv.BoxAnnotator()
labels = [f"{CLASSES[class_id]} {confidence:0.2f}" for _, _,
    confidence, class_id, _ in detections]
annotated_frame = box_annotator.annotate(scene=frame.copy(),
    detections=detections, labels=labels)



detections.mask = segment(
    sam_predictor=sam_predictor,
    image=cv2.cvtColor(frame, cv2.COLOR_BGR2RGB),
    xyxy=detections.xyxy
)

box_annotator = sv.BoxAnnotator()
mask_annotator = sv.MaskAnnotator()
labels = [
    f"{CLASSES[class_id]} {confidence:0.2f}"
    for _, _, confidence, class_id, _
    in detections]
annotated_image = mask_annotator.annotate(scene=frame.copy()
   , detections=detections)
annotated_image = box_annotator.annotate(scene=
   annotated_image, detections=detections, labels=labels)



plt.figure(figsize=(16, 8))

plt.subplot(1, 3, 1)
plt.imshow(annotated_frame)
plt.title(f'Test: {Test_Number} - Frame: {frame_number} -
   Time: {timestamp_str}')

plt.subplot(1, 3, 2)
plt.imshow(annotated_image)
plt.title('Segmented Image')
```

```
    grid_size_dimension = math.ceil(math.sqrt(len(detections.
        mask)))
    titles = [
        CLASSES[class_id]
        for class_id
        in detections.class_id
    ]
    sv.plot_images_grid(
        images=detections.mask,
        titles=titles,
        grid_size=(grid_size_dimension, grid_size_dimension),
        size=(16, 16)
    )

    plt.show()

    cap.release()


cap = cv2.VideoCapture(VIDEO_PATH)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
cap.release()

frame_slider = widgets.IntSlider(value=0, min=0, max=
    total_frames-1, step=1, description='Frame:')
box_threshold_slider = widgets.FloatSlider(value=0.35, min=0,
    max=1, step=0.01, description='Box Threshold:')
text_threshold_slider = widgets.FloatSlider(value=0.25, min=0,
    max=1, step=0.01, description='Text Threshold:')

interact(
    display_frame_and_segment,
    frame_number=frame_slider,
    box_threshold=box_threshold_slider,


    text_threshold=text_threshold_slider
)
```

ANEXO B – CÃșdigos TCC2

Listing B.1 – Configurar e importar bibliotecas

```
# @title Configurar e importar bibliotecas
from google.colab import drive
drive.mount('/content/drive')
import numpy as np

import cv2
import matplotlib.pyplot as plt
from ipywidgets import interact, widgets
from IPython.display import display


def convert_milliseconds_to_time(milliseconds):
    seconds = milliseconds // 1000
    minutes = seconds // 60
    seconds = seconds % 60
    milliseconds = milliseconds % 1000
    return f"{int(minutes):02}:{int(seconds):02}.{int(
        milliseconds):03}"


def get_frame(video_path, time_or_frame, type_input):
    cap = cv2.VideoCapture(video_path)

    if type_input == "Frame":
        frame_number = int(time_or_frame)
    elif type_input == "Time":
        frame_number = int(cap.get(cv2.CAP_PROP_FPS) *
            time_or_frame)

    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
    ret, frame = cap.read()

    return cap, ret, frame, frame_number
```

```
def plot_frame(video_path, time_or_frame,type_input):

    cap,ret,frame,frame_number=get_frame(video_path,
       time_or_frame,type_input)


    if type_input == "Frame":

      frame_label_time=f"Frame: {frame_number}"

    elif type_input == "Time":
      timestamp_ms = cap.get(cv2.CAP_PROP_POS_MSEC)
      timestamp_str = convert_milliseconds_to_time(timestamp_ms)

      frame_label_time=f"Timestamp: {timestamp_str} Time: {
         time_or_frame} seconds     Frame: {frame_number}"

    if ret:
        plt.figure(figsize=(10, 5))
        plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        plt.axis('off')
        plt.title(frame_label_time)
        plt.show()
    else:
        return


    return cap,ret,frame,frame_number



def get_video_path_from_selected_test(Select_Test):
    return f'/content/drive/MyDrive/ "_________"/Inputs/{
       Select_Test}.avi'


def convert_timestamp_to_seconds(timestamp):
```

```
    parts = timestamp.split(':')

    minutes = int(parts[0])
    seconds = int(parts[1])

    total_seconds = minutes * 60 + seconds
    return total_seconds

def process_table_data(table_data):

  n_frames = 20

  print(table_data)

  for row in table_data:
    selected_test = row[0]
    initial_string = row[1]
    final_string = row[2]
    crop_video_initial_to_end_with_N_Frames(selected_test,
       initial_string, final_string, n_frames)


def plot_images_dynamic(images, titles=None, cmap=None):
    if titles is None:
        titles = ['Image {}'.format(i + 1) for i in range(len(
           images))]

    if cmap is None:
        cmap = "gray"

    num_images = len(images)
    num_cols = int(np.ceil(np.sqrt(num_images)))
    num_rows = int(np.ceil(num_images / num_cols))

    plt.figure(figsize=(15, 7))

    for i, (image, title) in enumerate(zip(images, titles), 1):
        plt.subplot(num_rows, num_cols, i)
        if cmap is not None:
```

```
            plt.imshow(image, cmap=cmap)
        else:
            plt.imshow(image)
        plt.title(title)
        plt.axis('on')

    plt.tight_layout()
    plt.show()



# @title Selecionar vÃŋdeo { run: "auto", vertical-output: true
    }
Select_Test = "23_08_23_Test06_15LiAlH_C001H001S0001" # @param
    ["23_08_23_Test01_05LiAlH_C001H001S0001","23
    _08_23_Test02_05LiAlH_C001H001S0001","23
    _08_23_Test03_10LiAlH_C001H001S0001","23
    _08_23_Test04_15LiAlH_C001H001S0001","23
    _08_23_Test05_15LiAlH_C001H001S0001","23
    _08_23_Test06_15LiAlH_C001H001S0001","23
    _08_24_Test07_05MgB2_C001H001S0001","23
    _08_24_Test08_15MgB2_C001H001S0001","23
    _08_24_Test09_10MgB2_C001H001S0001","23
    _08_24_Test10_05MgB2_C001H001S0001","23
    _08_24_Test11_10MgB2_C001H001S0001","23
    _08_24_Test12_15MgB2_C001H001S0001"]



video_path = get_video_path_from_selected_test(Select_Test)

cap = cv2.VideoCapture(video_path)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
fps = cap.get(cv2.CAP_PROP_FPS)
total_time_seconds = total_frames / fps
cap.release()

time_slider = widgets.FloatSlider(value=0, min=0, max=
    total_time_seconds, step=0.1, description='Time (s):')
```

```
interact(
    plot_frame,
    video_path=widgets.fixed(video_path),
    time_or_frame=time_slider,
    type_input=widgets.fixed("Time")
)

# @title V2 Recortar testes
selected_tests = video_files



import cv2
import os



FRAME_STRIDE = 1000

for Select_Test in selected_tests:
    HOME = os.path.expanduser("~")
    VIDEO_PATH = f'{PATH_DRIVE}/Inputs/{Select_Test}'
    FRAME_DIR_PATH = f"{PATH_DRIVE}/Outputs/{Select_Test}/Frames
        "


    os.makedirs(FRAME_DIR_PATH, exist_ok=True)

    cap = cv2.VideoCapture(VIDEO_PATH)

    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

    for frame_number in range(0, total_frames, FRAME_STRIDE):
        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)

        ret, frame = cap.read()

        if ret:
            image_path = os.path.join(FRAME_DIR_PATH, f"frame_{
```

```
                frame_number:05d}_from_{Select_Test}.png")

            cv2.imwrite(image_path, frame)
            print(f"Frame {frame_number} saved as {image_path}")
        else:
            print(f"Failed to read frame {frame_number}")

    cap.release()

# @title Texto de tÃŋtulo padrÃčo
import os
import shutil
from sklearn.model_selection import train_test_split



PATH_DRIVE = "/content/drive/My Drive/"
input_images_dir = f"{PATH_DRIVE}Images"
input_labels_dir = f"{PATH_DRIVE}Teste_Input_Masks"
output_dir = f"{PATH_DRIVE}Teste_Output_Masks"

def create_directories(base_dir):
    os.makedirs(base_dir, exist_ok=True)
    os.makedirs(f"{base_dir}/images", exist_ok=True)
    os.makedirs(f"{base_dir}/labels", exist_ok=True)

create_directories(f"{output_dir}/train")
create_directories(f"{output_dir}/val")
create_directories(f"{output_dir}/test")

all_image_files = [f for f in os.listdir(input_images_dir) if os
    .path.isfile(os.path.join(input_images_dir, f))]

all_label_files = [f.replace('.jpg', '.txt').replace('.png', '.
    txt') for f in all_image_files]

train_images, temp_images, train_labels, temp_labels =
    train_test_split(all_image_files, all_label_files, test_size
    =0.3, random_state=42)
```

```
val_images, test_images, val_labels, test_labels =
    train_test_split(temp_images, temp_labels, test_size=1/3,
    random_state=42)

def copy_files(file_list, src_dir, dst_dir):
    for file_name in file_list:
        src_file_path = os.path.join(src_dir, file_name)
        dst_file_path = os.path.join(dst_dir, file_name)
        if os.path.exists(src_file_path):
            shutil.copy(src_file_path, dst_file_path)

copy_files(train_images, input_images_dir, f"{output_dir}/train/
    images")
copy_files(train_labels, input_labels_dir, f"{output_dir}/train/
    labels")

copy_files(val_images, input_images_dir, f"{output_dir}/val/
    images")
copy_files(val_labels, input_labels_dir, f"{output_dir}/val/
    labels")

copy_files(test_images, input_images_dir, f"{output_dir}/test/
    images")
copy_files(test_labels, input_labels_dir, f"{output_dir}/test/
    labels")

# @title Acessar Outputs e copiar todas imagens de cada subpasta
    para uma pasta Ãžnica Images
import os
import shutil


source_dir = '/content/drive/MyDrive/ "_________"/Outputs'
destination_dir = f"{PATH_DRIVE}Images"

if not os.path.exists(destination_dir):
    os.makedirs(destination_dir)

for root, dirs, files in os.walk(source_dir):
```

```
    for file in files:
        if file.lower().endswith(('.png', '.jpg', '.jpeg', '.gif
           ', '.bmp', '.tiff')):
            file_path = os.path.join(root, file)
            shutil.copy(file_path, destination_dir)

# @title Transformar mascaras de segmentaÃğÃčo
import os
import cv2


input_dir=f"{PATH_DRIVE}Teste_Input_Masks"
output_dir=f"{PATH_DRIVE}Teste_Output_Masks"


os.makedirs(output_dir, exist_ok=True)

for j in os.listdir(input_dir):
    image_path = os.path.join(input_dir, j)
    print(image_path)

    mask = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    _, mask = cv2.threshold(mask, 1, 255, cv2.THRESH_BINARY)

    H, W = mask.shape
    contours, hierarchy = cv2.findContours(mask, cv2.
       RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    polygons = []
    for cnt in contours:
        if cv2.contourArea(cnt) > 200:
            polygon = []
            for point in cnt:
                x, y = point[0]
                polygon.append(x / W)
                polygon.append(y / H)
            polygons.append(polygon)

    output_path = os.path.join(output_dir, f"{os.path.splitext(j
```

```
        )[0]}.txt")
    with open(output_path, 'w') as f:
        for polygon in polygons:
            for p_, p in enumerate(polygon):
                if p_ == len(polygon) - 1:
                    f.write('{}\n'.format(p))
                elif p_ == 0:
                    f.write('0 {} '.format(p))
                else:
                    f.write('{} '.format(p))

# @title A partir dos zips das Annotations de CVAT.IA, criar um
    folder para cada uma
import os
import zipfile

folder_path = '/content/drive/MyDrive/ "_________"/Annotations'

zip_files = [f for f in os.listdir(folder_path) if f.endswith('.
    zip')]

for zip_file in zip_files:
    zip_path = os.path.join(folder_path, zip_file)
    extract_folder = os.path.join(folder_path, os.path.splitext(
        zip_file)[0])
    os.makedirs(extract_folder, exist_ok=True)
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_folder)
        print(f'Unzipped {zip_file} into {extract_folder}')
# @title A partir das pastas des-zipadas, obter todas imagens de
    SegmentationObject
import os
import shutil



source_dir = '/content/drive/MyDrive/ "_________"/Annotations'
destination_dir = '/content/drive/MyDrive/ "_________"/
    Teste_Input_Masks'
```

```
if not os.path.exists(destination_dir):
    os.makedirs(destination_dir)

for root, dirs, files in os.walk(source_dir):
    if 'SegmentationObject' in dirs:
        seg_obj_dir = os.path.join(root, 'SegmentationObject')
        for seg_root, seg_dirs, seg_files in os.walk(seg_obj_dir
            ):
            for file in seg_files:
                if file.lower().endswith(('.png', '.jpg', '.jpeg
                    ', '.gif', '.bmp', '.tiff')):
                    file_path = os.path.join(seg_root, file)
                    shutil.copy(file_path, destination_dir)

# @title Criar Dataset: Acessar Labels, obter imagens
    correspondentes e
import os
import shutil
from sklearn.model_selection import train_test_split


input_images_dir = f"{PATH_DRIVE}Images"
input_labels_dir = f"{PATH_DRIVE}Teste_Output_Masks"
output_dir = f"{PATH_DRIVE}Dataset"

def create_directories(base_dir):
    os.makedirs(base_dir, exist_ok=True)
    os.makedirs(f"{base_dir}/images", exist_ok=True)
    os.makedirs(f"{base_dir}/labels", exist_ok=True)

create_directories(f"{output_dir}/train")
create_directories(f"{output_dir}/val")
create_directories(f"{output_dir}/test")

all_label_files = [f for f in os.listdir(input_labels_dir) if os
    .path.isfile(os.path.join(input_labels_dir, f))]

image_extensions = ['.jpg', '.png']
```

```
all_image_files = []
corresponding_label_files = []

for label_file in all_label_files:
    image_file_base = label_file.replace('.txt', '')
    for ext in image_extensions:
        if os.path.exists(os.path.join(input_images_dir,
           image_file_base + ext)):
            all_image_files.append(image_file_base + ext)
            corresponding_label_files.append(label_file)
            break

print(all_image_files)
print(corresponding_label_files)


train_images, temp_images, train_labels, temp_labels =
   train_test_split(all_image_files, corresponding_label_files,
   test_size=0.3, random_state=42)
val_images, test_images, val_labels, test_labels =
   train_test_split(temp_images, temp_labels, test_size=1/3,
   random_state=42)

def copy_files(file_list, src_dir, dst_dir):
    for file_name in file_list:
        src_file_path = os.path.join(src_dir, file_name)
        dst_file_path = os.path.join(dst_dir, file_name)
        if os.path.exists(src_file_path):
            shutil.copy(src_file_path, dst_file_path)

copy_files(train_images, input_images_dir, f"{output_dir}/train/
   images")
copy_files(train_labels, input_labels_dir, f"{output_dir}/train/
   labels")

copy_files(val_images, input_images_dir, f"{output_dir}/val/
   images")
copy_files(val_labels, input_labels_dir, f"{output_dir}/val/
   labels")
```

```
copy_files(test_images, input_images_dir, f"{output_dir}/test/
    images")
copy_files(test_labels, input_labels_dir, f"{output_dir}/test/
    labels")


# @title Verificar tamanho dataset
print("Train images:", len(os.listdir(f"{output_dir}/train/
    images")))
print("Train labels:", len(os.listdir(f"{output_dir}/train/
    labels")))
print("Validation images:", len(os.listdir(f"{output_dir}/val/
    images")))
print("Validation labels:", len(os.listdir(f"{output_dir}/val/
    labels")))
print("Test images:", len(os.listdir(f"{output_dir}/test/images
    ")))
print("Test labels:", len(os.listdir(f"{output_dir}/test/labels
    ")))

# @title Baixar Ultralytics YoloV8

!pip install ultralytics==8.0.196

from IPython import display
display.clear_output()

import ultralytics
#ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image

# @title Montar arquivo Yaml
import os
import yaml
```

```
output_dir = f"{PATH_DRIVE}Dataset"

destination_folder = '/content/Dataset'


data = {
    'path': destination_folder,
    'names': ['combustion_fire-solid_ramp'],
    'nc': 1,
    'test': 'test/images',
    'train': 'train/images',
    'val': 'val/images'
}


yaml_file_path = os.path.join(output_dir, 'dataset.yaml')

with open(yaml_file_path, 'w') as file:
    yaml.dump(data, file, default_flow_style=False)

print(f"YAML file created at: {yaml_file_path}")

# @title Copiar Dataset do Google Drive e fazer uma cÃşpia local
import shutil
import os

source_folder = '/content/drive/MyDrive/ "_________"/Dataset'
destination_folder = '/content/Dataset'

if os.path.exists(destination_folder):
    shutil.rmtree(destination_folder)

shutil.copytree(source_folder, destination_folder)

# @title Definir local dataset, definir parÃćmetros e Treinar
   yoloV8
yaml_file_path = os.path.join(destination_folder, 'dataset.yaml
   ')
```

```
!yolo task=segment mode=train model=yolov8s-seg.pt data={
    yaml_file_path} epochs=500 imgsz=1024 plots=True


!scp -r /content/runs '/content/drive/MyDrive/ "_________"'

# @title Copiar runs antigas
import shutil
import os

source_folder = '/content/drive/MyDrive/__________'
destination_folder = '/content/loaded_runs'

if os.path.exists(destination_folder):
    shutil.rmtree(destination_folder)

shutil.copytree(source_folder, destination_folder)


# @title Selecionar vÃŋdeo { run: "auto", vertical-output: true
    }
Select_Test = "23_08_24_Test11_10MgB2_C001H001S0001" # @param
    ["23_08_23_Test01_05LiAlH_C001H001S0001","23
    _08_23_Test02_05LiAlH_C001H001S0001","23
    _08_23_Test03_10LiAlH_C001H001S0001","23
    _08_23_Test04_10LiAlH_C001H001S0001","23
    _08_23_Test05_15LiAlH_C001H001S0001","23
    _08_23_Test06_15LiAlH_C001H001S0001","23
    _08_24_Test07_05MgB2_C001H001S0001","23
    _08_24_Test08_15MgB2_C001H001S0001","23
    _08_24_Test09_10MgB2_C001H001S0001","23
    _08_24_Test10_05MgB2_C001H001S0001","23
    _08_24_Test11_10MgB2_C001H001S0001","23
    _08_24_Test12_15MgB2_C001H001S0001"]
from PIL import Image
import matplotlib.pyplot as plt
import contextlib
import io
import torch
```

```
import pandas as pd
from IPython.display import display, clear_output


import numpy as np

def get_size(obj):
    if isinstance(obj, np.ndarray):
        return obj.nbytes
    elif torch.is_tensor(obj):
        return obj.element_size() * obj.nelement()
    else:
        return 0

def plot_frame_and_prediction(video_path, time_or_frame,
   type_input,loop):

    global df_results
    global total_frames
    print(total_frames)

    cap,ret,frame,frame_number=get_frame(video_path,
       time_or_frame,type_input)

    #clear_output()

    #plt.close('all')


    with contextlib.redirect_stdout(io.StringIO()):
        results = best_model(frame)


    print(frame_number)

    for i, r in enumerate(results):
        im_bgr = r.plot()
        im_rgb = Image.fromarray(im_bgr[..., ::-1])
```

```
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
pil_frame = Image.fromarray(frame_rgb)

width, height = pil_frame.size
black_image = Image.new('RGB', (width, height), (0, 0, 0))


timestamp_ms = cap.get(cv2.CAP_PROP_POS_MSEC)
timestamp_str = convert_milliseconds_to_time(timestamp_ms)

timeseconds=  int(timestamp_ms)/1000

if(type_input=="Frame"):
  frame_label_time=f"Timestamp: {timestamp_str} Time: {
     timeseconds:.3f} seconds     Frame: {frame_number}"
else:
  frame_label_time=f"Timestamp: {timestamp_str} Time: {
     time_or_frame} seconds     Frame: {frame_number}"

#frame_label_time=f"Timestamp: {timestamp_str} Time: {
   time_or_frame} seconds     Frame: {frame_number}"



if hasattr(results[0].masks, 'data'):

    masks_array = results[0].masks.data.cpu().numpy()
    final_mask=masks_array[0]


    props_df, average_top_x, average_top_y, mask, orig_img =
        extract_top_edge_points(results)

    if props_df is not None:
        masks_array = results[0].masks.data.cpu().numpy()
```

```
            mask = masks_array[0]
            fig = plot_top_edge_points(frame,mask, props_df,
                average_top_x, average_top_y)

            #plt.show()


            x=[time_or_frame]
            y=[average_top_y]


        tamanho_em_bytes = results[0].orig_img.nbytes


        boxes_size = 0
        masks_size = 0

        if hasattr(results[0].boxes, 'data'):
            boxes_size = get_size(results[0].boxes.data)

        if hasattr(results[0].masks, 'data'):
            masks_size = get_size(results[0].masks.data)



    else:
        final_mask=black_image
        x=[time_or_frame]
        y=[0]

    new_row = pd.DataFrame({'Min':timestamp_str,'Time(s)': int(
        timestamp_ms)/1000,'Frame': [frame_number], 'Position': y
        })
    df_results = pd.concat([df_results, new_row], ignore_index=
        True)
    df_results = df_results.sort_values(by='Frame')

    x_values = df_results['Frame'].tolist()
    y_values = df_results['Position'].tolist()
```

```
    plots = [
        lambda: example_image_plot(frame, None),
        lambda: example_image_plot(im_rgb, None),
        lambda: fig if hasattr(results[0].masks, 'data') else
           example_image_plot(black_image, None),

        lambda: create_line_plot(frame_number,x_values, y_values
           ,'Frame','PosiÃğÃčo'," Title",512)
    ]
    layout_matrix = [
        [0, 1],
        [2, 3]
    ]


    titles=[frame_label_time]

    fig = plot_plot_dynamic(plots, layout_matrix, titles)


    global Select_Test

    #if loop == True:
     # clear_output(wait=True)
     # return fig

df_results = pd.DataFrame(columns=[])

video_path = get_video_path_from_selected_test(Select_Test)

cap = cv2.VideoCapture(video_path)
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
fps = cap.get(cv2.CAP_PROP_FPS)
total_time_seconds = total_frames / fps
```

```
print(fps)

best_model = YOLO('/content/drive/MyDrive/ "_________"/runs/
    segment/train/weights/best.pt')


cap.release()

#time_slider = widgets.FloatSlider(value=0, min=0, max=
    total_time_seconds, step=0.1, description='Time (s):')
time_slider = widgets.FloatSlider(value=0, min=0, max=
    total_frames-1, step=1, description='Frame:')



interact(
    plot_frame_and_prediction,
    video_path=widgets.fixed(video_path),
    time_or_frame=2555,
    type_input=widgets.fixed("Frame"),
    loop=widgets.fixed(False)
)

import matplotlib.pyplot as plt

def create_line_plot(frame_atual, x, y, xlabel='X-axis', ylabel
    ='Y-axis', title='Line Plot', ylim=None, xlim=None):
    global total_frames
    fig, ax = plt.subplots()

    non_zero_y = [value for value in y if value != 0]
    non_zero_x = [x[i] for i in range(len(y)) if y[i] != 0]

    if len(x) == 1 and len(y) == 1:
        ax.scatter(x, y, label='Data Point', color='blue')
        ax.set_xlim(x[0] - 1, x[0] + 1)
        ax.set_ylim(y[0] - 1, y[0] + 1)
        atual_y = y[0]
```

```
else:
    ax.plot(non_zero_x, non_zero_y, label='Altura_h(frame)',
        color='blue')

    index = x.index(frame_atual)
    atual_x = x[index]
    atual_y = y[index]

    ax.scatter([atual_x], [atual_y], color='blue', s=2,
       label=f"ÃŽltima PosiÃğÃčo: {atual_y:.2f}")
    ax.axhline(y=atual_y, color='red', linestyle='--',
       linewidth=0.5, label='Linha na ÃŽltima PosiÃğÃčo Y')
    ax.annotate(f'{atual_y:.2f}', xy=(atual_x, atual_y),
       xytext=(atual_x + 200, atual_y),
                textcoords='data', ha='right', color='red')

if len(non_zero_y) == 0 or atual_y == 0:
    ax.set_ylim(250, 300)
else:
    y_min = 250 if min(non_zero_y) < 270 else 270



    #atual_y
    #y_min = atual_y-10 if atual_y-10 < 250 else 250
    y_max = 300
    ax.set_ylim(y_min, y_max)

if xlim:
    ax.set_xlim(xlim)
else:
    ax.set_xlim(0, total_frames)

ax.set_xlabel(xlabel)
ax.set_ylabel(ylabel)
ax.set_title(title)
ax.legend()

return fig
```

```
# @title Em Loop
import cv2
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
from IPython.display import clear_output
import shutil

tests = [
    #"23_08_23_Test01_05LiAlH_C001H001S0001",
    #"23_08_23_Test02_05LiAlH_C001H001S0001",
    #"23_08_23_Test03_10LiAlH_C001H001S0001",
    #"23_08_23_Test04_10LiAlH_C001H001S0001",
    "23_08_23_Test05_15LiAlH_C001H001S0001",
    #"23_08_23_Test06_15LiAlH_C001H001S0001",
    #"23_08_24_Test07_05MgB2_C001H001S0001",
    #"23_08_24_Test08_15MgB2_C001H001S0001",
    #"23_08_24_Test09_10MgB2_C001H001S0001",
    #"23_08_24_Test10_05MgB2_C001H001S0001",
    #"23_08_24_Test11_10MgB2_C001H001S0001",
    "23_08_24_Test12_15MgB2_C001H001S0001"
]

for Select_Test in tests:

    df_results = pd.DataFrame(columns=['Min', 'Time(s)', 'Frame
        ', 'Position']) # Reset the DataFrame
    plt.close('all')
    video_path = get_video_path_from_selected_test(Select_Test)

    cap = cv2.VideoCapture(video_path)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    fps = cap.get(cv2.CAP_PROP_FPS)
    total_time_seconds = total_frames / fps


    folder_path = '/content/drive/MyDrive/ "_________"/Final
```

```
    Output Results'
os.makedirs(folder_path, exist_ok=True)

output_video_path = f'{folder_path}/Final_Video_{Select_Test
    }.mp4'
fourcc = cv2.VideoWriter_fourcc(*'mp4v')
fps = 30

start_frame = 0
end_frame = total_frames - 1
step = 10

frame_width = 1920
frame_height = 1080

video_writer = cv2.VideoWriter(output_video_path, fourcc,
    fps, (frame_width, frame_height))

for frame_number in range(start_frame, end_frame, step):
    clear_output(wait=True)
    print(f"Processing frame {frame_number} of {Select_Test
        }...")
    fig = plot_frame_and_prediction(video_path, frame_number
        , "Frame", True)
    #plt.show()

    fig.canvas.draw()
    image = np.frombuffer(fig.canvas.tostring_rgb(), dtype='
        uint8')
    image = image.reshape(fig.canvas.get_width_height()
        [::-1] + (3,))

    image = cv2.resize(image, (frame_width, frame_height))

    video_writer.write(cv2.cvtColor(image, cv2.COLOR_RGB2BGR
        ))

video_writer.release()
```

```
file_name = f"Tabela_{Select_Test}.xlsx"
df_results.to_excel(file_name, index=False)

destination_path = os.path.join(folder_path, file_name)

shutil.move(file_name, destination_path)
```