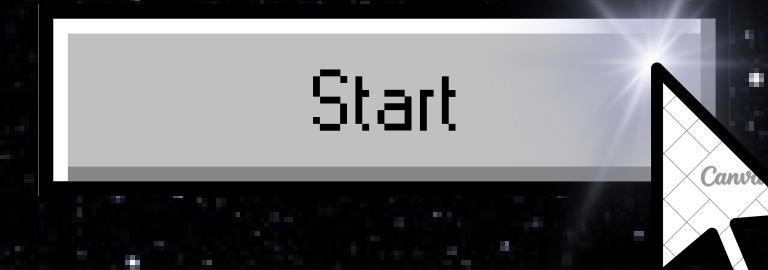
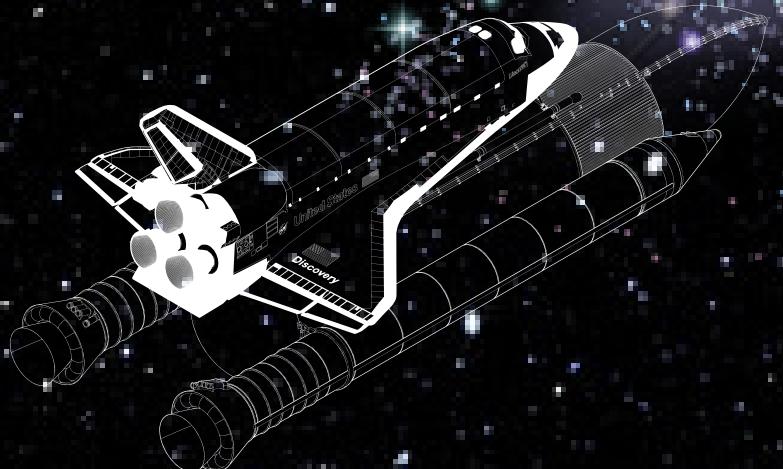


CONTAGEM DE CORPOS CELESTES



Integrantes:

Helena Carvalho Leal
Rafael Takeguma Goto

Orientador:

Leonardo Nogueira Matos

ÍNDICE

- 1. INTRODUÇÃO**
- 2. DATASET**
- 3. DESENVOLVIMENTO**
- 4. RESULTADOS**

INTRODUÇÃO

TEMA: CONTAGEM DE CORPOS CELESTES;

REPOSITÓRIO: [HTTPS://GITHUB.COM/RAFASETO/CONTAGEM-DE-CORPOS-CELESTES](https://github.com/rafaseto/contagem-de-corpos-celestes)

DATASET: [HTTPS://WWW.KAGGLE.COM/DATASETS/JAIMETRICKZ/GALAXY-ZOO-2-IMAGES](https://www.kaggle.com/datasets/jaimetrickz/galaxy-zoo-2-images);

Descrição: Considerando o interesse de muitos em observar astros, criamos um programa que ajuda a tornar o usuário um melhor astrônomo. Logo, são disponibilizadas diversas imagens de galáxias e o usuário precisa identificar quantas delas estão mais adequadas visualmente para serem analisadas posteriormente. Para isso, utilizamos métodos de processamento digital de imagens para contar o número de corpos celestes visíveis, para que assim, o usuário possa comparar a sua própria contagem com a do programa e verificar seus acertos.

Sinta-se a vontade para contar/jogar também!



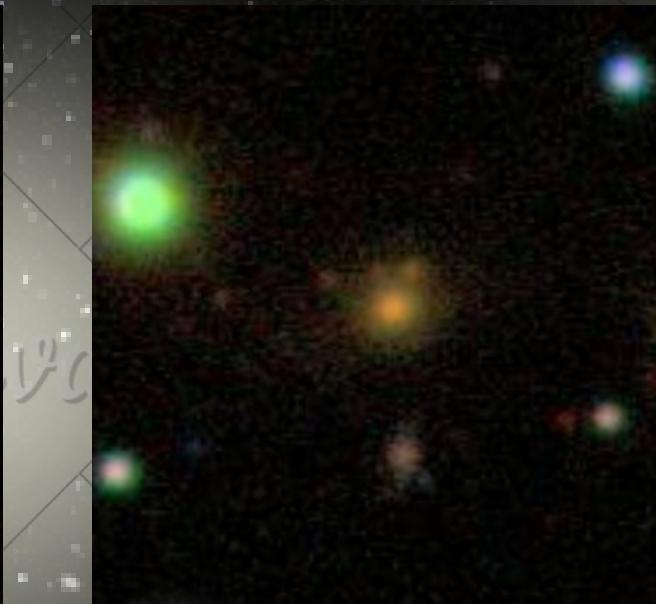
dataset



1 IMAGEM-01



2 IMAGEM-02



3 IMAGEM-03



4 IMAGEM-04



5 IMAGEM-05



6 IMAGEM-06



7 IMAGEM-07



8 IMAGEM-08



9 IMAGEM-09



10 IMAGEM-10

DESENVOLVIMENTO (PASSO A PASSO)

1. CONverte as imagens em escala de cinza
2. binariza as imagens
3. Aplica abertura e fechamento
4. Aplica Sobel
5. Aplica transformada de Hough
6. Cria função 'DESENHA.FIGURA()'
7. Aplica 'DESENHA.FIGURA()'

DESENVOLVIMENTO (PASSO A PASSO)

1. CONverte as imagens em escala de cinza

```
# Lista que armazena as imagens em escala de cinza
imagens_cinza_list = [color.rgb2gray(imagem) for imagem in imagens_list]
```

✓ 0.0s

DESENVOLVIMENTO (PASSO A PASSO)

2. BINARIZA AS IMAGENS

```
# Lista que armazena as imagens binárias
imagens_binarias_list = []

for imagem in imagens_cinza_list:
    imagem_binaria = imagem.copy()

    limiar = imagem.max()*(110/256)
    imagem_binaria[imagem_binaria<=limiar]=0
    imagem_binaria[imagem_binaria>0]=1

    imagens_binarias_list.append(imagem_binaria)
```

DESENVOLVIMENTO (PASSO A PASSO)

3. APLICA ABERTURA E FECHAMENTO

```
# Lista que armazena as imagens refinadas com as operações morfológicas
imagens_refinadas_list = []

for imagem in imagens_binarias_list:
    imagem_refinada = imagem.copy()

    imagem_refinada = binary_opening(imagem_refinada)
    imagem_refinada = binary_closing(imagem_refinada)

    imagens_refinadas_list.append(imagem_refinada)
```

DESENVOLVIMENTO (PASSO A PASSO)

4. APLICA SOBEL

```
# Lista que armazena bordas das imagens
bordas_list = [sobel(imagem) for imagem in imagens_refinadas_list]
```

✓ 0.1s

DESENVOLVIMENTO (PASSO A PASSO)

5. APLICA TRANSFORMADA DE HOUGH

```
# Define que iremos detectar círculos com raio variando entre 5 e 50
raios = np.arange (5,50,2)

# Lista que armazena a acumulação dos parâmetros
grades_list = []

for imagem in bordas_list:
    hough_grade = hough_circle(imagem, raios)
    grades_list.append(hough_grade)

✓ 0.2s
```

DESENVOLVIMENTO (PASSO A PASSO)

6. CRIA FUNÇÃO 'DESENHA-FIGURA()'

```
caminho_da_pasta = 'Resultados'

# Desenha a figura com os círculos detectados em verde
def desenha_figura(imagem, index):
    # As coordenadas dos centros dos círculos detectados são armazenadas em 'a' e 'b'
    acumuladores, a, b, raio = hough_circle_peaks(grades_list[index], raios, total_num_peaks=10)

    fig, ax = plt.subplots(ncols=1, nrows=1, figsize=(10, 6))
    imagem_final = imagem

    for centro_y, centro_x, raio_ in zip(b, a, raio):
        circy, circx = circle_perimeter(centro_y, centro_x, raio_, shape=imagem.shape)
        imagem_final[circy, circx] = (20, 220, 20)

    ax.imshow(imagem_final, cmap=plt.cm.gray)

    # Nome do arquivo a ser salvo
    nome_do_arquivo = f"resultado_{index+1}.png"
```

DESENVOLVIMENTO (PASSO A PASSO)

6. CRIA FUNÇÃO 'DESENHA FIGURA()'

```
for centro_y, centro_x, raio_ in zip(b, a, raio):
    circy, circx = circle_perimeter(centro_y, centro_x, raio_, shape=imagem.shape)
    imagem_final[circy, circx] = (20, 220, 20)

ax.imshow(imagem_final, cmap=plt.cm.gray)

# Nome do arquivo a ser salvo
nome_do_arquivo = f"resultado_{index+1}.png"

# Caso o diretório não exista, cria ele
if not os.path.exists(caminho_da_pasta):
    os.makedirs(caminho_da_pasta)

# Salva a imagem
plt.savefig(os.path.join(caminho_da_pasta, nome_do_arquivo))

# Mostra a imagem
plt.show()
```

DESENVOLVIMENTO (PASSO A PASSO)

P. APLICA FUNÇÃO 'DESENHA.FIGURA()'

```
# index para uma imagem de 'imagens_list' poder referenciar o item respectivo em 'grades_list'  
i = 0  
  
for imagem in imagens_list:  
    desenha_figura(imagem, i)  
    i += 1
```

✓ 6.6s

Python

RESULTADOS

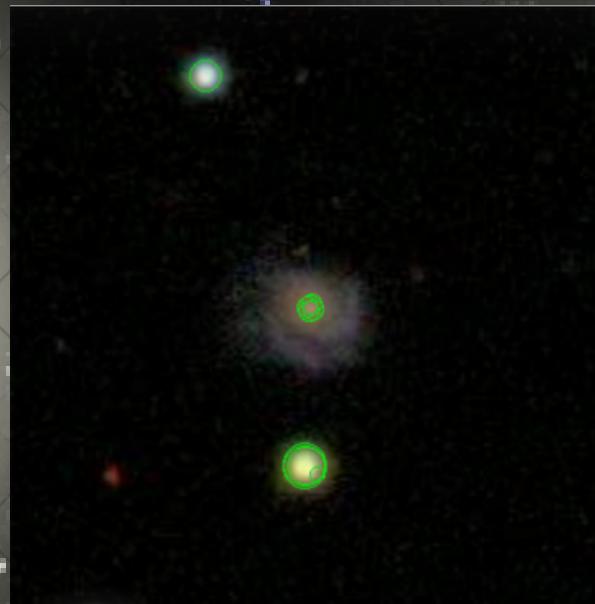
APÓS TODO O PROCESSO QUE SE ENCONTRA NO NOTEBOOK APP.IPYNB TEMOS OS RESULTADOS DA CONTAGEM NA PASTA 'RESULTADOS'.

OS CORPOS CELESTES QUE A APLICAÇÃO DETECTOU ESTÃO CIRCULADOS EM VERDE.

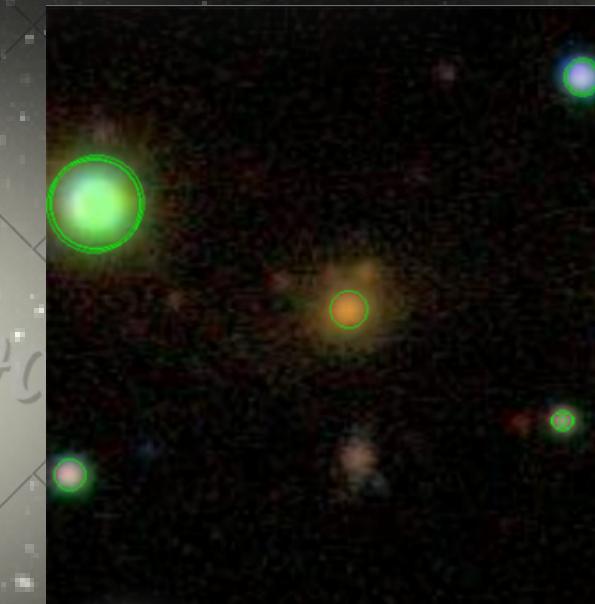
RESULTADOS



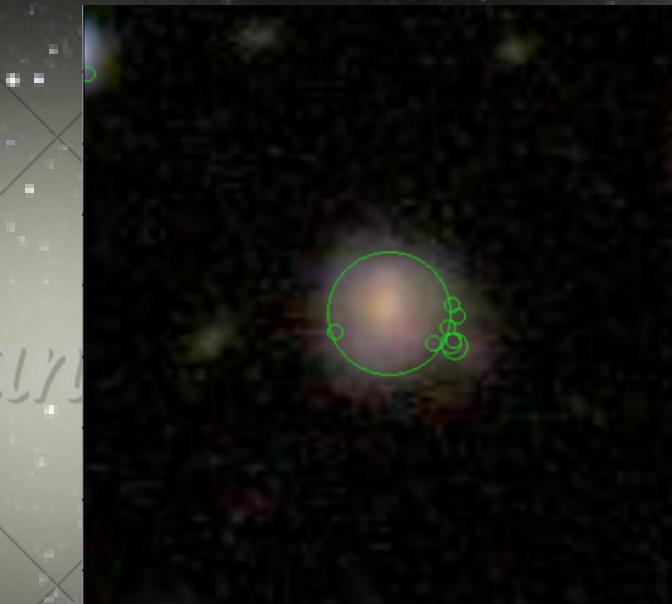
1 IMAGEM-01



2 IMAGEM-02



3 IMAGEM-03



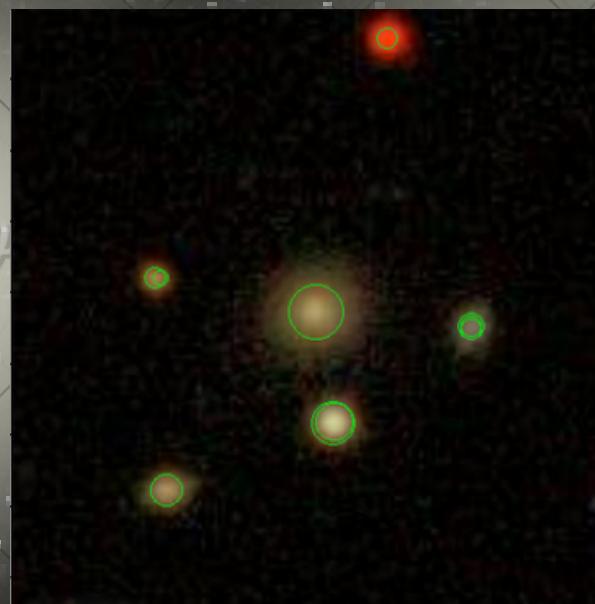
4 IMAGEM-04



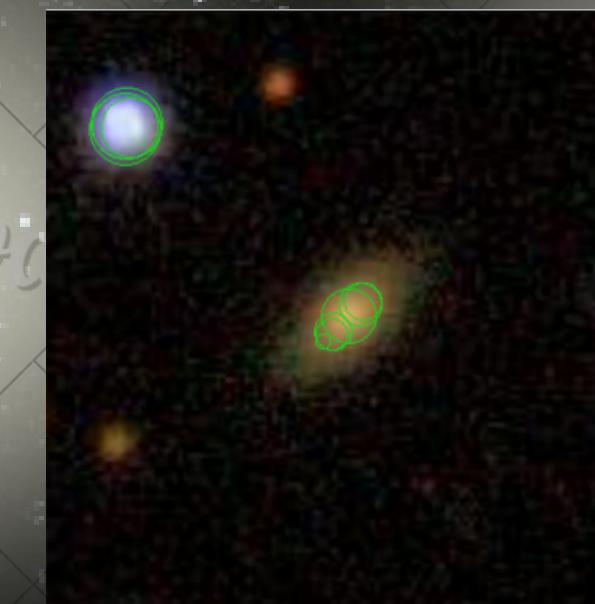
5 IMAGEM-05



6 IMAGEM-06



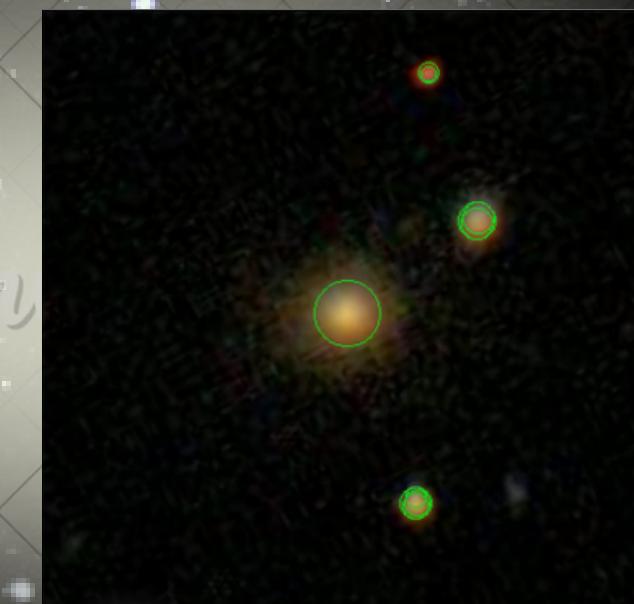
7 IMAGEM-07



8 IMAGEM-08



9 IMAGEM-09

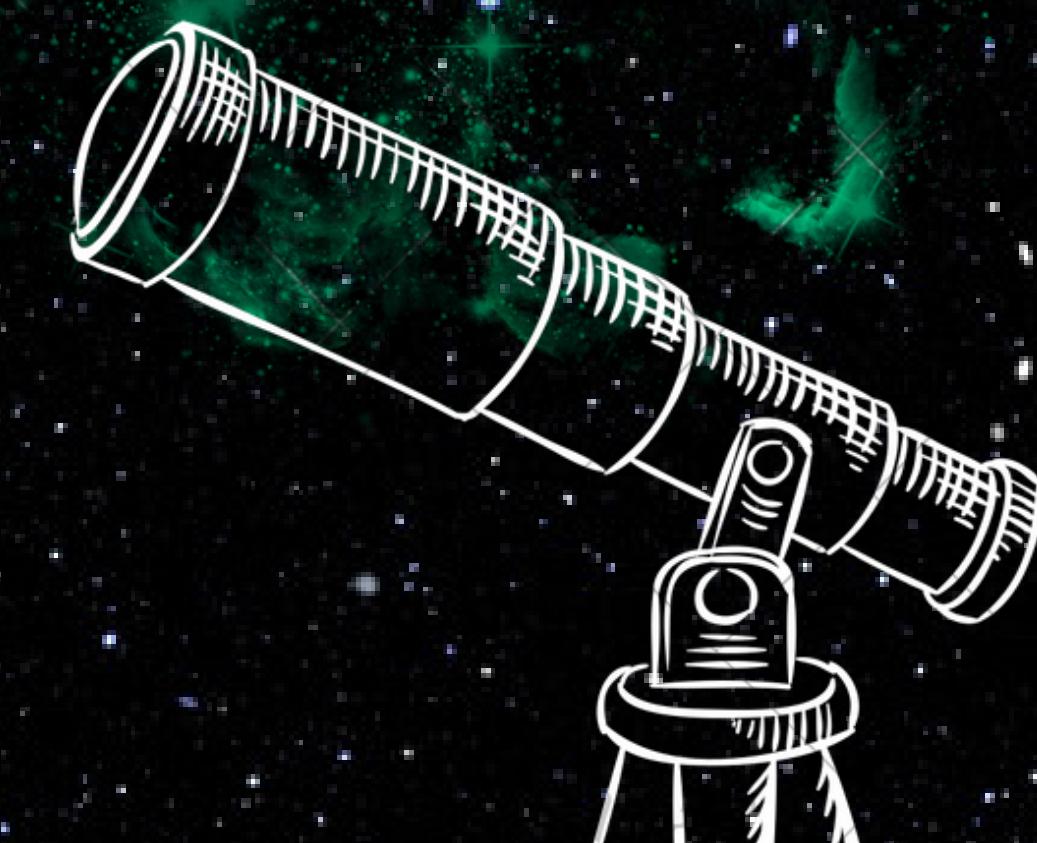


10 IMAGEM-10

RESULTADOS

IMAGENS	CONT.HELENA	CONT.RAFAEL	CONT.PROG	RESULTADO
IMAGEM-01	5	5	3	X
IMAGEM-02	4	3	3	RAFA
IMAGEM-03	6	5	5	RAFA
IMAGEM-04	2	1	1	RAFA
IMAGEM-05	10	7	7	RAFA
IMAGEM-06	9	8	4	X
IMAGEM-07	6	6	6	DOIS
IMAGEM-08	5	5	2	HELENA
IMAGEM-09	8	8	3	X
IMAGEM-10	5	4	4	RAFA

RAFAEL
GANHOU!



obrigado!!!

