UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIENCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELETRICA

EDUARDO FELIPE CAMPANINI

DETECÇÃO E RASTREAMENTO DE METEOROS:

Abordagem em operações morfológicas

Trabalho de conclusão de curso apresentado ao curso de bacharelado em Engenharia Elétrica como parte dos requisitos necessários à obtenção do título de Bacharel em Engenharia Elétrica

Orientador(a): Prof.Dr Robson Barcellos

Coorientadora: Prof.ª Dra. Daniela Cardozo Mourão

São Carlos

2023

DETECÇÃO E RASTREAMENTO DE METEOROS

Abordagem em operações morfológica

BANCA EXAMINADORA

_____

Prof.Dr. Robson Barcellos

Universidade Federal De São Carlos

_____

Profª.Dra. Daniela Cardozo Mourão

Universidade Estadual Paulista

_____

Prof. Dr. Rafael Sfair

Universidade Estadual Paulista

_____

Prof. Dr. Gabriel Borderes Motta

Universidad Carlos III de Madrid

AGRADECIMENTOS

Dedico aos meus pais e os agradeço por todo apoio e amor incondicional durante os percalços da graduação. Gostaria também de agradecer meus orientadores Prof.Dr Robson Barcellos e Prof.ª Drª Daniela Cardozo Mourão, por toda a paciência e ensinamentos transmitidos com dedicação e excelência.

RESUMO

O sistema de detecção e rastreamento de objetos em vídeo implementado neste projeto consiste em uma série de etapas que envolvem a captura e pré-processamento de frames de vídeo, detecção de objetos por meio de segmentação de imagens, rastreamento de objetos usando um algoritmo de correlação de regiões, e finalmente a saída dos resultados em arquivos de Excel e XML.

A implementação faz uso de bibliotecas de processamento de imagens e visão computacional, como OpenCV e Pandas. A detecção de objetos é realizada por meio de thresholding adaptativo e segmentação de imagens, enquanto que o rastreamento é feito por meio do algoritmo de distância euclidiana que usa correlação de regiões para rastrear os objetos ao longo dos frames do vídeo. Os resultados do rastreamento são armazenados em um dataframe, que é posteriormente ordenado e salvo em arquivos de Excel e XML. O arquivo de Excel contém informações como o identificador do objeto, as coordenadas de centroide, e o número do frame em que o objeto foi detectado. O arquivo XML contém informações semelhantes, mas em um formato mais estruturado.

O sistema apresentou bons resultados em diferentes cenários de teste, conseguindo detectar e rastrear objetos com alta precisão e confiabilidade. O código é altamente configurável, permitindo ajustes para diferentes tipos de objetos, tamanhos e velocidades de movimento, bem como para diferentes ambientes de gravação de vídeo.

Palavras-chave: visão computacional, análise de vídeo, meteoro, BRAMON, processamento de imagem, OpenCV, rastreamento de meteoro, rastreamento de objetos, detecção de objetos.

ABSTRACT

The object detection and tracking system implemented in this project consists of a series of steps that involve capturing and preprocessing video frames, object detection through image segmentation, object tracking using a region correlation algorithm, and finally outputting the results in Excel and XML files.

The implementation makes use of image processing and computer vision libraries such as OpenCV and Pandas. Object detection is performed through adaptive thresholding and image segmentation, while tracking is done through the KCF algorithm, which uses region correlation to track objects over video frames. The tracking results are stored in a dataframe, which is later sorted and saved to Excel and XML files. The Excel file contains information such as the object identifier, centroid coordinates, and the frame number in which the object was detected. The XML file contains similar information, but in a more structured format.

The system showed good results in different test scenarios, being able to detect and track objects with high precision and reliability. The code is highly configurable, allowing adjustments for different types of objects, sizes and speeds of movement, as well as for different video recording environments.

keywords: computer vision, video analysis, meteor, BRAMON, image processing, opencv,meteor tracking, object tracking, object detection.

Lista de Tabelas

Lista de Figuras

Sumário

1. INTRODUÇÃO

"A primeira rede de patrulhamento fotográfico do céu noturno visando o estudo de meteoros data de 1936, construída pela universidade de Harvard. Posteriormente, outras redes foram instaladas na Alemanha, República Checa, Eslováquia, Suíça, Áustria, Bélgica, mas com especial relevância nos Estados Unidos, no Canadá, na Espanha, na Holanda e no Japão. No Brasil, a primeira câmera all-sky de patrulha de meteoros foi montada pela Profa. Dra. Elizabeth Zucolotto, da Universidade Federal do Rio de Janeiro, o que inspirou demais propostas nacionais, como a da Rede de Astronomia Observacional (REA), sobre a qual fundamentamos o presente projeto (IZECSON, COELHO e JACQUES, 2008). Até recentemente, porém, não havia nenhum patrulhamento efetivo em funcionamento no Brasil, a não ser pelo crescente empenho da BRAMON (Brazilian Meteor Observation Network) desde 2014" [1].

As soluções de software para análise de imagens de meteoros atualmente disponíveis no mercado brasileiro consistem em códigos fechados e privados, impossibilitando assim que pesquisadores da área tenham acesso ao código-fonte para sua melhor compreensão e análise, dificultando severamente a publicação de artigos relacionados aos estudos que fazem uso destes softwares de análise de meteoros, uma vez que não é possível explicar e justificar os processos internos do programa.

O software mais utilizado, o UFOCaptureV2, foi desenvolvido pela SonotaCo em 2009 sob uma licença de aproximadamente R$1.000,00 (reais) com distribuições apenas em inglês e japonês [2]. É um software relativamente antigo onde pesquisadores não possuem acesso ao código-fonte, os custos de licenças a nível nacional podem ser muito caros e colaboradores podem não ter conhecimento básico de inglês para operar o programa visto que de acordo com a pesquisa "Demandas de Aprendizagem de Inglês no Brasil" (British Council, 2013), apenas 5,1% da população com 16 anos ou mais afirma possuir algum conhecimento da língua inglesa [3]".

O processo de análise de meteoros pode ser basicamente divido em duas etapas. A primeira diz respeito à detecção do evento e rastreio de sua trajetória na imagem do vídeo. Já a segunda etapa, foca nas transformações destas coordenadas de imagem em coordenadas do mundo real como ascensão reta e declinação, um processo chamado de astrometria e conversão de coordenadas. Este estudo foca na primeira etapa e, realiza um estudo preliminar da segunda, onde acessibilidade de conhecimento e operação do programa foram o norte buscado no desenvolvimento do projeto.

1.1.Objetivos

Os objetivos do trabalho envolvem detectar meteoros em um vídeo, rastrear esses objetos ao longo do tempo armazenando suas posições em um formato conveniente para análise posterior. O objetivo geral do trabalho é fornecer uma solução automatizada, em contrapartida a um manual, para detectar e rastrear objetos em um vídeo, permitindo que os pesquisadores ou analistas de dados coletem informações valiosas sobre o movimento e o comportamento de meteoros.

Em um prazo maior, após a implementação da astrometria e conversão de coordenadas, o objetivo é de que esses possam ser utilizados no software PIM (Programa de Integração de Meteoros), que está sendo desenvolvido pela BRAMON (Brazilian Meteor Observation Network). O objetivo do PIM é determinar a trajetória do meteoro antes e depois do evento capturado, incluindo sua trajetória pelo sistema solar e possível local de queda com uma linha de dispersão. Este projeto é parte da BRAMON, uma organização sem fins lucrativos que tem como missão desenvolver e operar uma rede para o monitoramento de meteoros e fornecer dados científicos à comunidade através da análise de capturas realizadas por estações de monitoramento mantidas por seus membros.

2. REVISÃO BIBLIOGRAFICA

Os trabalhos anteriores nesta área têm se concentrado na detecção de eventos de meteoros ao invés do rastreamento de suas trajetórias. Segundo o trabalho de Thiago C. L. Marsola e Ana C. Lorena[4] a melhor abordagem para detecção de meteoros a partir de vídeos de captura de movimento no céu noturno, onde meteoros podem ou não estar presentes, são Deep Convolutional Neural Network (CNN). Para o treinamento da rede foi utilizado um conjunto pequeno de dados expandido artificialmente através de “transfer learning”. Com estas técnicas os resultados obtidos chegaram a uma precisão de 84,35% de acerto.

Já no estudo “Detecting the presence of meteors in images:new collection and results [5] uma abordagem diferente foi utilizada para detecção de meteoros. Segundo os autores, analises baseadas no empilhamento das imagens obteve os melhores desempenhos. Ao empilhar as imagens do evento de meteoros linhas características de seu rastro são obtidas.

3. CONCEITOS PRELIMINARES

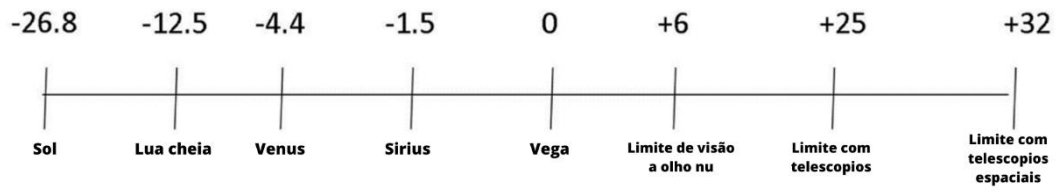
3.1.O que são meteoros e meteoritos

Meteoros são fenômenos astronômicos caracterizados pela entrada de fragmentos de rocha espacial na atmosfera planetária. Quando existe resistência à passagem de um fragmento pela atmosfera e ele atinge o solo, é então designado por meteorito. É estimado que aproximadamente 44 toneladas de materiais meteóricos atingem atmosfera terrestre a cada dia, sendo que a maioria deste material é vaporizado antes de atingir o solo[6]. Aproximadamente 90% dos meteoros possuem menos de 5,4 kg, e 50% pesam menos que 283 g [7].

Em uma noite comum, dada uma região qualquer do céu, é possível observar dezenas de meteoros, porém quando a incidência de meteoros aumenta drasticamente, este fenômeno é então designado chuva de meteoros. Tal fenômeno ocorre anualmente devido à passagem do planeta terra pelo rastro de poeira deixado por cometas, e geralmente a chuva de meteoros recebe o nome da constelação ou estrela mais próxima do seu radiante, sendo o radiante a região no céu de onde os meteoros parecem estar se originando. Por exemplo, a chuva de meteoros Oriónidas possui seu radiante na constelação de Órion, tem sua atividade entre 4 de outubro e 14 novembro sendo originada pelos restos de poeira do cometa 1P/Halley

Em astronomia, magnitude é uma medida sem unidade que mensura a luminosidade de um objeto, como estrelas, planetas e meteoros. A escala de magnitude é de logaritmo inverso, quanto mais brilhante um objeto, menor é seu número de magnitude. O gráfico abaixo ilustra um grande espectro de magnitude.

Figura 1:Escala de magnitude



Fonte: autoria própria

A intensidade luminosa de eventos de meteoro varia dramaticamente. Passando por magnitudes de +6 corresponde a objetos marginalmente visíveis a olho nu, até intensidades maiores do que -4, onde são classificados como bólidos, também conhecidos como bola de fogo. Eventos de meteoro não costumam ter duração maior que 2 segundos[8].

Figura 2:Fenômeno de meteoro

Fonte: autoria própria

Existem diversas classificações para meteoritos quanto a sua composição, porém as três principais são meteoritos rochosos, meteoritos ferrosos, e meteoritos ferro-rochosos. Os meteoritos rochosos são compostos principalmente por minerais silicatos. Meteoritos ferrosos são principalmente compostos por ligas metálicas de ferro e níquel. Já os meteoritos ferro-rochosos são compostos por uma mistura de ambos tipos já citados.

Figura 3:Meteorito palasito, do tipo ferro-rochoso

Fonte: Natural history museum [9]

Alguns meteoritos possuem em seu interior a composição inalterada de matéria de 4,6 biliões de anos atrás, fornecendo a cientistas informações sobre a formação do sistema solar, planetas e a vida.

3.2. O que são pixels

Um pixel é a menor unidade de uma imagem digital e é representado por uma grade de pontos retangulares em que cada ponto representa um valor numérico. Em uma imagem colorida, cada pixel é composto por três valores numéricos que representam as intensidades dos componentes vermelho, verde e azul (RGB, do inglês Red, Green, Blue), que juntos formam a cor do pixel. O modelo de cor RGB é baseado na adição de luz, ou seja, quanto maior a intensidade de um componente, mais brilhante será a cor correspondente. Os valores RGB variam de 0 a 255, com 0 representando a ausência completa da cor e 255 representando a intensidade máxima. A combinação dos valores RGB de um pixel determina sua cor final. Por exemplo, um pixel com intensidades de R=255, G=0 e B=0 seria vermelho, enquanto um pixel com intensidades de R=0, G=255 e B=0 seria verde. Para criar outras cores, basta variar as intensidades dos componentes RGB. A intensidade de um pixel em uma imagem em escala de cinza representa a quantidade de luz refletida pelo objeto naquele ponto da imagem. Essa intensidade é representada por um único valor numérico, variando de 0 a 255, em que 0 representa preto absoluto e 255 representa branco absoluto.

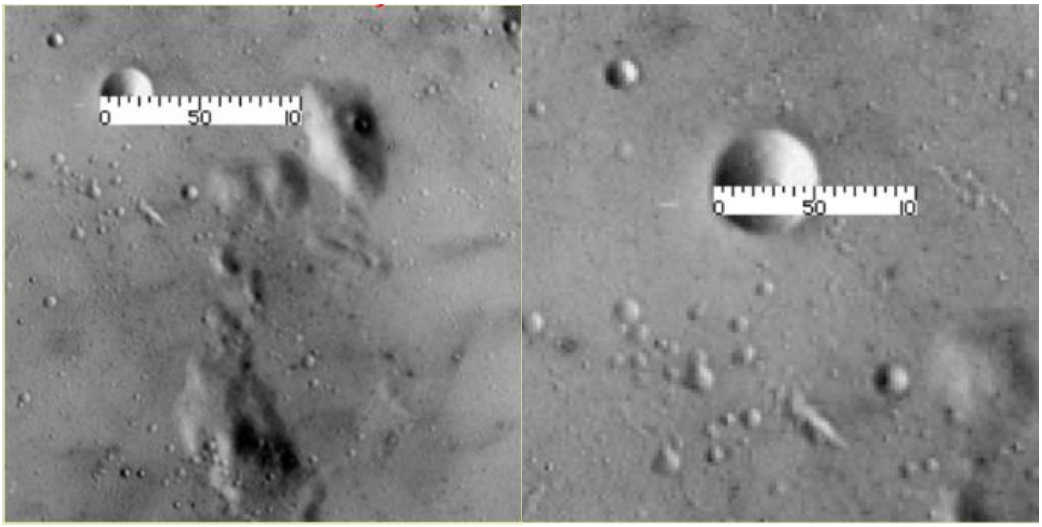
Os conceitos importantes relacionados a pixels no processamento de imagens deste estudo são os seguintes:

Resolução: A resolução de uma imagem é a medida da quantidade de detalhes ou informações que podem ser representadas na imagem. Em outras palavras, a resolução determina a qualidade da imagem em termos de nitidez, clareza e detalhes. A resolução é geralmente expressa em termos de pixels, que são os pontos coloridos que compõem a imagem digital. A resolução de uma imagem é definida pelo número de pixels na largura e altura da imagem, e é geralmente representada pela fórmula "largura x altura". Por exemplo, uma imagem com uma resolução de 1920 x 1080 significa que a imagem tem 1920 pixels de largura e 1080 pixels de altura. Quanto maior a resolução de uma imagem, mais detalhes podem ser representados e, portanto, a imagem terá uma maior qualidade visual. No entanto, é importante lembrar que a resolução não é o único fator que determina a qualidade de uma imagem. Outros fatores, como a qualidade da

lente da câmera, a iluminação e a exposição também têm um papel importante na qualidade final da imagem.

Resolução espacial: O tamanho de um pixel varia de acordo com a resolução da tela, ou seja, o número de pixels que compõem a tela. A imagem abaixo ilustra dois casos

Figura 4: Comparação de resoluções



Fonte: Aula 2, Fundamentos de imagens digitais Professor Dr. Robson Barcellos

Na primeira imagem, a cratera possui 27 pixels de diâmetro, já na segunda 55 pixels. Isso significa que se a cratera tiver 550Km de diâmetro, na primeira imagem isso representa 20,37 Km/píxel e na segunda 10Km/píxel. Ou seja, é possível medir as características da cratera com maior precisão na segunda imagem.

Distância entre pixels: As distâncias D4 e D8 referem-se a diferentes métricas de distância entre pixels em imagens binárias, que são aquelas em que cada pixel é representado por apenas um bit de informação, indicando se ele está ligado (1) ou desligado (0). A distância D4, também conhecida como distância de Manhattan ou distância do tabuleiro de xadrez, mede a distância entre dois pixels apenas nos sentidos horizontal e vertical, ou seja, apenas nas quatro direções possíveis (norte, sul, leste e oeste). Essa distância é calculada como a soma das diferenças em cada dimensão (horizontal e vertical). Por outro lado, a distância D8, também conhecida como distância de Chebyshev ou distância diagonal, mede a distância entre dois pixels em todas as oito direções possíveis (norte, sul, leste, oeste e as diagonais).

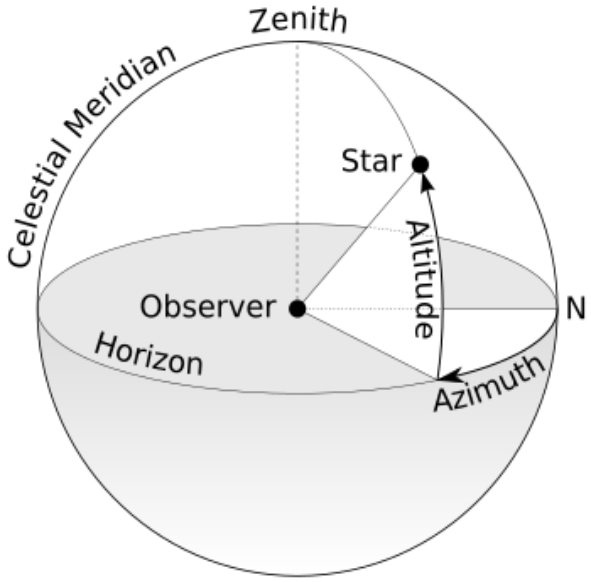
Segmentação: A segmentação é a separação de uma imagem em regiões ou objetos distintos, com base em suas características. Ela é geralmente usada para identificar objetos em uma imagem ou para separar o fundo da imagem do objeto de interesse.

3.3.Coordenadas Celestes

A posição de um objeto no céu pode ser descrita por um conjunto de coordenadas. Existem diversos tipos de sistemas de coordenadas celestes, porém os dois sistemas utilizados neste estudo foram os sistemas Horizontal e Equatorial de coordenadas.

O sistema horizontal de coordenadas divide o céu em dois hemisférios, sendo o ponto central de observação o próprio observador, os objetos visíveis estão localizados no hemisfério superior. O círculo que separa os dois hemisférios é chamado de horizonte celeste, e seu ponto mais alto é denominado Zênite. A primeira coordenada deste sistema, a altitude, informa qual o angulo de um objeto em relação ao horizonte celeste, por exemplo, um objeto localizado no horizonte celeste possui uma altitude de 0°, sendo que um objeto no ponto Zênite possui uma altitude máxima de 90°. A segunda coordenada, denominada Azimute, informa qual o angulo que um objeto apresenta em relação ao polo norte geográfico na linha do horizonte celeste, ou seja, um objeto localizado na direção norte tem um Azimute de 0° e um objeto localizado no Oeste tem uma posição de 270°.

Figura 5:Ilustração do sistema de coordenadas horizontal



Fonte: Fundação tal [n°](fonte)

O sistema equatorial de coordenadas é o mais utilizado entre astrônomos e amadores, visto que seu ponto central é fixo e localizado no centro do planeta terra, ou seja, um sistema geocêntrico. O plano fundamental deste sistema é formado pela projeção da linha do equador na esfera celeste, e com a projeção dos polos norte e sul formam a base para suas medidas. A primeira coordenada deste sistema é denominada de Declinação, e informa qual a medida

angular entre um objeto até o equador celeste. Por exemplo, um objeto no polo norte celeste possui uma declinação de 90º, já um objeto no polo sul celeste possui uma declinação de -90°. A segunda coordenada deste sistema, Ascensão Reta, mede a distância angular entre um objeto e a localização do centro do Sol no equinócio de março (ponto vernal).

Tabela 1. Comparação entre sistemas de coordenadas

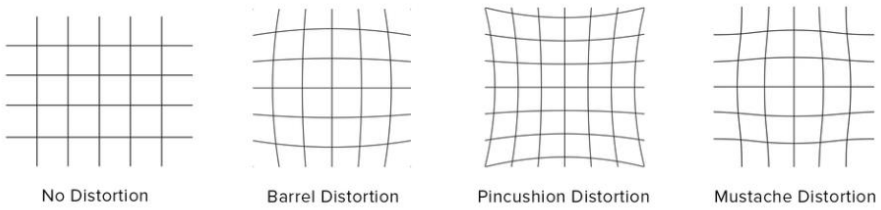
				Coordenadas	
Sistema de coordenadas	**Ponto central**	**Latitude 0°**	**Polos**	**Latitude**	**Longitude**
Horizontal (Alt-Az)	Observador	Horizonte	Zênite, Nadir	Altitude(a)	Azimute(A)
Equatorial	Centro da Terra ou Sol	Equador Celeste	Polos celestiais	Declinação(δ)	Ascensão reta (α)

Fonte: Autoria própria

3.4.Distorção de imagens

O objetivo de uma calibração astrométrica é projetar elementos da esfera celeste em um plano, uma vez que o sensor da câmera pode ser considerado um plano. Os dados iniciais para a realização da calibração são o de apontamento da câmera descrito pela ascensão reta e declinação do centro do campo de visão, rotação e escala. O problema é que a maioria das lentes possui distorções que precisam ser corrigidas via manipulação algébrica de coordenadas em um software.

Figura 6: Representação dos tipos de distorção em lentes. (a) Sem distorção. (b)Distorção de barril. (c)Distorção almofada de alfinete. (d)Distorção de bigode



Fonte: image engineering

No caso de lentes sem distorções, a projeção no plano é representada por linhas retas. Em lentes que apresentam distorção de barril a amplificação da imagem diminui com a distância do eixo óptico. O efeito aparente é de uma imagem mapeada em torno de uma esfera. Um exemplo deste tipo de lente são as lentes olho de peixe que captam vistas hemisféricas. A distorção “almofada de alfinetes” a ampliação da imagem aumenta conforme a distância do

eixo óptico. Já a “distorção de bigode” começa como uma distorção de barril no centro e gradualmente se transforma em uma distorção de almofada de alfinetes na periferia da imagem

3.5.Astrometria

Astrometria é o estudo da posição, movimento e distância de objetos celestes como estrelas, planetas e galáxias. Em estudos de meteoros, a astrometria se refere à medida precisa da trajetória, velocidade e origem dos meteoros a partir da observação da sua posição no céu. Isso permite aos cientistas determinar as características físicas e químicas dos meteoros e fornecer informações sobre a composição e evolução do sistema solar.

A astrometria pode ser realizada usando uma variedade de técnicas, incluindo observações com telescópios, câmeras e análise de dados coletados por naves espaciais e satélites. No escopo deste estudo a astrometria será o processo pelo qual a posição do meteoro será medida e obtida a partir dos frames de vídeo, levando-se em consideração fatores como distorção atmosférica, paralaxe, distorção de lentes e refração atmosférica.

3.6.Brazilian Meteor Network - BRAMON

A Rede Brasileira de Monitoramento de Meteoros é uma organização aberta e colaborativa, mantida por voluntários e colaboradores e sem fins lucrativos. A BRAMON tem como missão desenvolver e operar uma rede para o monitoramento de meteoros, produzindo e fornecendo dados científicos à comunidade através da análise de suas capturas, que são realizadas por estações de monitoramento mantidas por seus membros. O logotipo da BRAMON representa uma chuva de meteoros estilizada com o formato do Brasil como radiante, assim como mostra a figura.

Figura 7:Logotipo da BRAMON

Fonte: website da BRAMON

Entre os objetivos da BRAMON estão:

- Realizar a captura de meteoros em imagens para astrometria e cálculo de suas órbitas;
- Colaborar na catalogação de novos radiantes de chuvas de meteoros;
- Realizar análises espectrográficas de meteoros;
- Triangular possíveis locais de quedas de meteoritos;
- Coletar dados referentes a outros fenômenos que possam ser captados pelas estações; e
- Agregar a colaboração pública e de instituições de ensino básico e superior de forma a tornar a rede uma ferramenta de ensino e divulgação científica.

4. METODOLOGIA E MATERIAIS

4.1.Python/OpenCv2

A utilização do OpenCV2 em conjunto com a linguagem de programação Python é uma técnica bastante popular para processamento de imagens. A biblioteca OpenCV2 fornece uma ampla gama de funções e métodos que podem ser utilizados para análise e manipulação de imagens digitais. Para a implementação de um projeto de processamento de imagens com OpenCV2 e Python, é importante ter uma compreensão básica da linguagem de programação Python e sua sintaxe. É necessário também ter uma boa compreensão dos conceitos fundamentais de processamento de imagens, tais como a estrutura de uma imagem digital, operações de convolução, filtragem, etc. Além disso, é importante ter acesso à documentação oficial do OpenCV2 e Python. A documentação fornece uma visão geral das funções e métodos disponíveis e sua sintaxe, além de exemplos de uso. O processo de aprendizagem utilizando

OpenCV2 e Python envolve a aplicação prática da metodologia baseada em tentativas e erros, guiada pelo conhecimento prévio em processamento de imagens. Isso significa que é necessário experimentar diferentes abordagens e ajustar parâmetros até encontrar uma solução adequada para o problema em questão.

Durante o desenvolvimento do projeto, a documentação do OpenCV2 e Python foi constantemente consultada para obter informações sobre funções específicas, sua sintaxe e exemplos de uso. Essa abordagem permitiu a descoberta de novos recursos e técnicas, além de ajudar a encontrar soluções para os desafios encontrados no processo de desenvolvimento.

5. DESENVOLVIMENTO DO PROJETO

O desenvolvimento deste estudo seguiu três frentes, detecção do meteoro, rastreamento de objetos e geração de rastro de meteoro. Os modelos de detecção de meteoro serão explorados a seguir em conjunto com o rastreamento do meteoro a qual foi aplicado apenas um método.

5.1. Código de detecção de meteoro

O modulo de detecção de meteoro é responsável por receber um vídeo do céu noturno onde há evento de meteoro e realizar manipulações morfológicas na imagem afim de isolar o evento em cada frame do vídeo para que seja possível rastreá-lo. O objetivo final é gerar um arquivo XML/XLSX que contém as informações dos centroides de cada objeto em cada frame do vídeo. Para alcançar esse objetivo, o algoritmo segue os seguintes passos

A. Carregar o vídeo
B. Criar um loop para processar cada frame do vídeo
    a. Ler frames consecutivos
    b. Converter frames para escala de cinza
    c. Calcular a diferença entre os dois frames para expor objetos em movimento
    d. Aplicar detecção de contornos
    e. Calcular o centroide do contorno detectado.
    f. Armazenar informações de posição e frame do meteoro
C. Classificar e salvar as informações dos objetos detectados por ID e frame

A seguir encontra-se o código completo, seguido de uma explicação detalhada de seu funcionamento.

```
import cv2
import pandas as pd
import numpy as np
import xml.etree.ElementTree as ET
from tracker import EuclideanDistTracker
import os

# Get the name of the video file
video_file = 'v3.avi'
video_name = os.path.splitext(video_file)[0]

# Create xlsx file name based on video title
xlsx_file = f"{video_name}_object_data.xlsx"

# Open the video file
cap = cv2.VideoCapture(video_file)

# Initialize variables
frame_index = 0

# Initialize tracker
tracker = EuclideanDistTracker()

# Initialize dataframe to store detections
df = pd.DataFrame(columns=['id', 'cx', 'cy', 'frame'])

# Get the first frame in grayscale
ret, previous_frame = cap.read()
previous_gray = cv2.cvtColor(previous_frame, cv2.COLOR_BGR2GRAY)

# Loop through the frames
while True:
    # Get the next frame in grayscale
    ret, current_frame = cap.read()
    if not ret:
        break
    current_gray = cv2.cvtColor(current_frame, cv2.COLOR_BGR2GRAY)

    # Subtract the previous frame from the current frame
    difference = cv2.absdiff(current_gray, previous_gray)
    cv2.imshow('difference', difference)


    # Apply the Canny edge detector
    edges = cv2.Canny(difference, 100, 200)

    # Dilate the edges
    kernel = np.ones((5, 5), np.uint8)
    dilated = cv2.dilate(edges, kernel, iterations=1)

     # Create a black rectangle at the bottom of the frame to remove
noise
    height = 30
        cv2.rectangle(dilated, (0, current_frame.shape[0] - height),
(current_gray.shape[1], current_frame.shape[0]), (0, 0, 0), -1)

    # Find contours in the dilated image
       contours, hierarchy = cv2.findContours(dilated, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

```
57
58       # Initialize list to store detections
59       detections = []
60
61       # Loop through all contours
62       for cnt in contours:
63           # Calculate contour area
64           area = cv2.contourArea(cnt)
65
66           # If contour area is large enough
67           if area > 10:
68               # Calculate moments of the contour
69               M = cv2.moments(cnt)
70
71               # If moment value is not zero
72               if M["m00"] != 0:
73                   # Calculate centroid coordinates
74                   cx = int(M["m10"] / M["m00"])
75                   cy = int(M["m01"] / M["m00"])
76                   # Draw centroid on the frame
77                    image = cv2.circle(current_frame, (cx, cy), radius=3,
color=(0, 0, 255), thickness=-1)
78                   detections.append([cx, cy])
79
80       # Update tracker with the detections
81       boxes_ids = tracker.update(detections)
82
83       # Loop through detected object boxes
84       for box_id in boxes_ids:
85           # Get object data
86           cx, cy, id = box_id
87
88           # Draw object id on the frame
89         cv2.putText(image, str(id), (cx, cy - 15), cv2.FONT_HERSHEY_PLAIN,
1, (0, 0, 255), 2)
90
91           # Add object data to the dataframe
92           data = [id, cx, cy, frame_index]
93           df.loc[len(df.index)] = data
94           # Show the frame with detections
95           cv2.imshow("Detection", image)
96
97       # Update frame index
98       frame_index += 1
99
100       # Set the current frame and gray frame as the previous frame and
gray frame for the next iteration
101      previous_gray = current_gray
102
103      # Break the loop if the 'q' key is pressed
104      if cv2.waitKey(0) & 0xFF == ord('q'):
105          break
106
107  # Sort dataframe by object id and frame index
108  df_sorted = df.sort_values(by=['id', 'frame'])
109
110  # Save sorted dataframe to Excel file
111  df_sorted.to_excel(xlsx_file, sheet_name='Sheet1', index=False)
112
113  # Create root element for XML file
114  root = ET.Element('Detection')
```

```
115
116  # Iterate through sorted dataframe rows to create XML elements
117  for _, row in df_sorted.iterrows():
118      # Create element for each object in each frame
119      element = ET.SubElement(root, 'object')
120
121      # Set element attributes
122      element.set('id', str(row['id']))
123      element.set('frame', str(row['frame']))
124      element.set('cx', str(row['cx']))
125      element.set('cy', str(row['cy']))
126
127      # Add new line character at the end of each frame element
128      element.tail = '\n'
129
130  # Write XML file
131  tree = ET.ElementTree(root)
132  xml_file = f"{video_name}_object_data.xml"
133  tree.write(xml_file, encoding='utf-8', xml_declaration=True)
134
135  # Release video capture and close windows
136  cap.release()
137  cv2.destroyAllWindows()
```

O primeiro passo, é a realização da importação de bibliotecas necessárias.

- **cv2:** biblioteca OpenCV utilizada para processamento de imagem e vídeo.
- **tracker:** um script Python que contém a classe EuclideanDistTracker, responsável por rastrear os objetos ao longo do tempo.
- **pandas:** biblioteca utilizada para armazenar e manipular dados em forma de tabelas (dataframes).
- **numpy:** biblioteca utilizada para cálculos matemáticos em arrays multidimensionais.
- **xml.etree.ElementTree:** biblioteca utilizada para criar e manipular árvores XML.

Em seguida, na linha 9, é definido o nome do arquivo de vídeo que será analisado. Na linha 10, a extensão do arquivo é removida do nome do arquivo para ser usado posteriormente para nomear o arquivo de saída. Na linha 13, é definido o nome do arquivo xlsx para armazenar os dados de detecção.

Na linha 16, o arquivo de vídeo é aberto e lido usando o objeto "cv2.VideoCapture" e é atribuído a uma variável "cap". Na linha 19, a variável "frame_index" é inicializada como zero, e será usada para acompanhar o índice do frame atual.

Na linha 22, o objeto "EuclideanDistTracker" é inicializado e atribuído a tracker. O rastreador de distância Euclidiana é um algoritmo usado para rastrear objetos entre quadros sucessivos e atribuir um ID unico a cada objeto rastreado.

Na linha 25, um "pandas DataFrame" é inicializado com as colunas 'id', 'cx', 'cy' e 'frame' que serão usados para armazenar as coordenadas x e y do centroide do objeto detectado e o quadro em que o objeto aparece.

Na linha 28, a função "cap.read()" é usada para ler o primeiro quadro do vídeo e é armazenada em "previous_frame". O valor de retorno da função é uma tupla que contém um valor booleano indicando se o quadro foi lido com sucesso e o próprio quadro. A imagem é convertida para escala de cinza na linha 29 usando a função "cv2.cvtColor" e é atribuída a "previous_gray".

Na linha 31 começa o loop de processamento dos frames do vídeo. A partir da linha 33, a variável current_frame recebe o próximo frame do vídeo em escala de cinza na variável current_gray (linha 37), assim como foi feito para o frame anterior na linha 29. Na linha 40, a diferença absoluta entre o frame atual e o frame anterior é calculada pela função cv2.absdiff, produzindo uma imagem binária que mostra as diferenças entre os pixels dos dois quadros. A imagem binária é mostrada na linha 41 com a função cv2.imshow. Na linha 45, é aplicado o detector de bordas Canny na imagem binária para encontrar objetos no frame. Na linha 48, a função cv2.dilate é usada para unir elementos próximos na imagem caso a detecção de borda produza um objeto com muito ruído.

Na linha 53, um retângulo preto é criado na parte inferior da imagem para remover o barra de título presente em vídeos da BRAMON afim de evitar ruídos.

A função cv2.findContours (linha 56) é usada para encontrar contornos na imagem processada, retornando uma lista de contornos na variável contours e uma hierarquia de contornos na variável hierarchy. Esta função possui a seguinte sintaxe : cv2.findContours(image, mode, method[, contours[, hierarchy[, offset]]]). Ela recebe os seguintes parâmetros:

- **image**: a imagem binária na qual os contornos serão encontrados.
- **mode**: especifica se os contornos serão encontrados apenas na imagem ou se também serão levados em conta buracos dentro de objetos. Existem várias opções de modo, mas

o mais comum é cv2.RETR_TREE, que encontra todos os contornos e organiza-os em uma árvore hierárquica.

- **method**: especifica a forma como os pontos do contorno são encontrados. Existem várias opções de método, mas o mais comum é cv2.CHAIN_APPROX_SIMPLE, que comprime segmentos de linha horizontais, verticais e diagonais e deixa apenas seus pontos finais.
- **contours**: (opcional) uma lista vazia que será preenchida com os contornos encontrados.
- **hierarchy**: (opcional) uma matriz que armazena informações sobre a hierarquia dos contornos.
- **offset**: (opcional) um deslocamento que pode ser adicionado a todos os pontos do contorno.

A função cv2.findContours retorna dois valores:

- **contours**: uma lista contendo todos os contornos encontrados na imagem. Cada contorno é uma matriz numpy de pontos (x, y) que formam o contorno.
- **hierarchy**: uma matriz que armazena informações sobre a hierarquia dos contornos. Essas informações incluem o índice do contorno pai, do próximo contorno irmão e do primeiro e último filho. Essas informações podem ser usadas para analisar a estrutura hierárquica dos objetos na imagem.

No código deste estudo, a função cv2.findContours é usada para encontrar contornos dos objetos na imagem binária produzida pela detecção de bordas Canny na linha 45. A lista de contornos é armazenada na variável contours e a hierarquia de contornos é armazenada na variável hierarchy. Essas informações serão usadas para analisar os objetos na imagem e extrair informações úteis sobre eles, como posição, tamanho e forma.

Na linha 59, uma lista vazia chamada detections é inicializada para armazenar as detecções. Essa lista será preenchida com informações dos objetos detectados no frame nos passos seguintes do código.

Em seguida, o código faz um loop(linha 62) através de cada contorno na lista contours, calculando a área do contorno usando a função cv2.contourArea na linha 64. Se a área do contorno for grande o suficiente (maior do que 10 pixels ao quadrado, ou estipulado pelo usuário). Na linha 69, é utilizada a função cv2.moments para calcular os momentos da área delimitada pelo contorno. Os momentos são cálculos estatísticos de uma distribuição de pixels

que podem ser usados para calcular características como a área, centroide e orientação de um objeto na imagem. No contexto desse código, a função cv2.moments é usada para calcular as coordenadas do centroide do objeto delimitado pelo contorno. O resultado é armazenado na variável M, que contém uma série de valores estatísticos calculados para o contorno, incluindo o momento de ordem 0 (m00), que é a área do contorno. As coordenadas do centroide são então calculadas dividindo-se os momentos de primeira ordem (m10 e m01) pelo momento de ordem 0 (m00). O resultado é armazenado nas variáveis cx e cy, respectivamente. Essas coordenadas representam o centro de massa do objeto na imagem e são usadas posteriormente para desenhar um círculo que representa o objeto em questão. O código desenha um círculo na linha 77, ao redor dos centroides calculados, usando a função cv2.circle. As coordenadas do centroide são adicionadas à lista de detecções na linha 78. Por fim, o código atualiza o rastreador com as detecções usando a função tracker.update(detections) na linha 81.

Na linha 84, é iniciado um loop para percorrer as caixas delimitadoras dos objetos detectados. A variável boxes_ids contém uma lista com as informações de cada objeto detectado: a coordenada x do centroide do objeto, a coordenada y do centroide e um identificador único do objeto. As informações de cada objeto são extraídas da lista boxes_ids na linha 86. Na linha 89, o identificador do objeto (um número por ordem de detecção) é desenhado na imagem para o usuário, usando a função cv2.putText. A função recebe como parâmetros a imagem, o texto a ser exibido, a posição do texto na imagem, a fonte, o tamanho da fonte, a cor do texto e a espessura da linha. Na linha 92, as informações do objeto são adicionadas ao dataframe df. Na linha 93, um novo registro é adicionado ao dataframe 'df'. O registro contém informações sobre um objeto detectado, incluindo seu ID, coordenadas x e y do centroide e o índice do quadro atual. A função 'loc' é usada para adicionar o novo registro à última linha do dataframe. A sintaxe 'len(df.index)' retorna o número de linhas existentes no dataframe 'df', que é usado como índice para o novo registro. Na linha 95, a imagem com as detecções é exibida na tela usando a função cv2.imshow.

O índice contador de frames é atualizado na linha 98. Em seguida é atualizado o valor da variável previous_gray para a imagem em escala de cinza do frame atual, para que ela seja usada como imagem anterior na próxima iteração do loop, garantindo que os frames sejam processados em sequência. Na linha 104, é verificado se a tecla "q" foi pressionada e, se for o caso, o loop é interrompido usando a função cv2.waitKey, terminando o loop principal de processamento de imagem do programa.

Na linha 108, o dataframe df contendo todas as detecções é ordenado por 'id' e 'frame' em ordem crescente, usando a função sort_values() do pandas. Na linha 111, o dataframe ordenado é salvo em um arquivo do Excel usando a função to_excel() do pandas.

Na linha 114, é criado um elemento raiz para o arquivo XML usando a classe Element do módulo ElementTree. Na linha 117, um loop é iterado através das linhas do dataframe ordenado para criar um elemento XML para cada objeto em cada quadro. Na linha 119, um novo elemento 'object' é criado usando a classe SubElement do módulo ElementTree. Na linha 122-125, os atributos do elemento 'object' são definidos usando a função set(). Na linha 128, um caractere de nova linha é adicionado ao final de cada elemento 'frame' usando a variável tail afim de melhorar a legibilidade do arquivo. Na linha 131-133, o arquivo XML é gravado usando a função write() da classe ElementTree.

Finalmente na linha 136-137, a captura de vídeo é liberada e as janelas são fechadas usando as funções release() e destroyAllWindows() do OpenCV, respectivamente.

5.2. Código de rastreamento

Este modulo é uma classe python “EuclideanDistTracker” que implementa um algoritmo simples de rastreamento de objetos baseado na distância euclidiana. A finalidade do algoritmo é acompanhar os objetos em um fluxo de vídeo associando cada objeto a um identificador único. O código implementado pode ser visto abaixo.

```
1    import math
2    class EuclideanDistTracker:
3        def __init__(self):
4            # Store the center positions of the objects
5            self.center_points = {}
6            # Keep the count of the IDs
7             # each time a new object id detected, the count will increase
by one
8            self.id_count = 0
9
10
11       def update(self, objects_rect):
12           # Objects boxes and ids
13           objects_bbs_ids = []
14
15           # Get center point of new object
16           for rect in objects_rect:
17               x, y = rect
18               cx = x
19               cy = y
20
21               # Find out if that object was detected already
```

```
22                same_object_detected = False
23                for id, pt in self.center_points.items():
24                    dist = math.hypot(cx - pt[0], cy - pt[1])
25
26                    if dist <25:
27                        self.center_points[id] = (cx, cy)
28                        print(self.center_points)
29                        objects_bbs_ids.append([cx,cy,id])
30                        same_object_detected = True
31                        break
32
33                # New object is detected we assign the ID to that object
34                if same_object_detected is False:
35                    self.center_points[self.id_count] = (cx, cy)
36                    objects_bbs_ids.append([cx, cy, self.id_count])
37                    self.id_count += 1
38
39            # Clean the dictionary by center points to remove IDS not used
anymore
40            new_center_points = {}
41            for obj_bb_id in objects_bbs_ids:
42                #_, _, _, _, object_id = obj_bb_id
43                cx, cy, object_id = obj_bb_id
44                center = self.center_points[object_id]
45                new_center_points[object_id] = center
46
47            # Update dictionary with IDs not used removed
48            self.center_points = new_center_points.copy()
49            return objects_bbs_ids
```

A classe possui duas estruturas de dados principais:

- **‘center_points’,** um dicionário que armazena as posições centrais dos objetos. Cada objeto é identificado por um ID exclusivo.
- **‘id_coun't’**, uma variável que controla o número de objetos que foram detectados até o momento.

A classe possui dois métodos: "init()" e "update()".

- **‘__init__’**: O método "init()" é o construtor da classe e é responsável por inicializar as variáveis necessárias para rastrear os objetos. O método cria um dicionário vazio chamado "center_points" para armazenar os pontos centrais dos objetos rastreados e uma variável "id_count" para contar o número de objetos detectados.
- **‘Update’**: Método de atualização que recebe como entrada uma lista de objetos representados como retângulos. Para cada retângulo na entrada, ele calcula o ponto central do retângulo e então verifica se o ponto central corresponde a um

objeto que já foi detectado anteriormente. Isso é feito calculando a distância euclidiana entre o novo ponto central e os pontos centrais de todos os objetos armazenados no dicionário ‘center_points’. Se a distância euclidiana entre o novo ponto central e um dos centros armazenados for menor que 25, assume-se que o novo objeto corresponde ao mesmo objeto associado ao ponto central armazenado e seu ID é atualizado na lista ‘objects_bbs_ids’. Se nenhum ponto central armazenado for encontrado, então um novo objeto é considerado detectado e um novo ID é atribuído a ele na lista ‘objects_bbs_ids’. Depois que todos os retângulos foram processados, o método remove os IDs de objetos que não estão mais sendo rastreados, atualizando o dicionário "center_points" com os IDs restantes. Finalmente, o método retorna a lista "objects_bbs_ids" contendo os centros dos objetos e seus IDs associados para uso posterior no rastreamento de objetos.

Na linha 2, a classe EuclideanDistTracker é definida e, dentro de seu construtor (__init__), dois dicionários são criados: center_points e id_count. O primeiro dicionário, center_points, armazena os pontos centrais dos objetos detectados em quadros anteriores, juntamente com o ID atribuído a cada objeto. O segundo dicionário, id_count, é utilizado para controlar a numeração dos IDs atribuídos a novos objetos detectados.

O método update, que começa na linha 5, é chamado a cada quadro do vídeo e recebe como entrada os retângulos de contorno dos objetos detectados. A variável objects_bbs_ids, inicializada na linha 8, será preenchida com os IDs e coordenadas dos centroides dos objetos detectados em cada quadro.

A partir da linha 11, o código percorre todos os retângulos de contorno (objects_rect) para verificar se cada objeto já foi detectado em quadros anteriores. Isso é feito comparando as coordenadas do centroide do objeto atual com os centroides armazenados no dicionário center_points.

Se um objeto com coordenadas semelhantes já tiver sido detectado anteriormente, seu ID é recuperado do dicionário center_points e adicionado à lista objects_bbs_ids na linha 25. Caso contrário, um novo ID é atribuído ao objeto (linha 29) e o novo centroide é adicionado ao dicionário center_points (linha 30), que armazenará as informações dos objetos detectados em quadros anteriores.

Finalmente, na linha 35, o dicionário center_points é atualizado com as informações dos objetos detectados no quadro atual, removendo os IDs que não foram detectados neste quadro. Dessa forma, ao final da execução do método update, a lista objects_bbs_ids conterá as informações dos objetos detectados no quadro atual, juntamente com seus IDs. Esse método é chamado a cada quadro do vídeo, permitindo que o algoritmo rastreie os objetos em movimento em cada quadro consecutivo.

5.3. Rastro de meteoro

Este código lê um arquivo de vídeo onde há evento de meteoro, encontra os valores máximos de pixel em cada frame do vídeo e, em seguida, salva a imagem final como um arquivo PNG. A ideia é que este arquivo PNG mostre o rastro de meteoro durante toda sua duração. Ele usa a biblioteca OpenCV para processar o vídeo e a biblioteca numpy para manipular matrizes.

```
1    import cv2
2    import numpy as np
3    import os
4
5    # Open the video file
6    video_file_name = 'v6.avi'
7    cap = cv2.VideoCapture(video_file_name)
8
9    # Get the first frame to set up dimensions
10   ret, frame = cap.read()
11   height, width = frame.shape[:2]
12
13   # Create an array to store the maximum pixel values
14   max_values = 0 * np.ones((height, width), dtype=np.uint8)
15
16   while True:
17       # Read a new frame from the video
18       ret, frame = cap.read()
19
20       # If we didn't get a frame, we've reached the end of the video
21       if not ret:
22           break
23
24       # Convert the frame to grayscale
25       gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
26
27       # Update the maximum pixel values
28       max_values = np.maximum(max_values, gray)
29
30       # Display the resulting image
31       cv2.imshow('Meteor Video', max_values)
32
33       # Wait for a key press and exit if 'q' is pressed
34       if cv2.waitKey(1) & 0xFF == ord('q'):
35           break
36
37   # Release the video file and close the window
38   cap.release()
39   cv2.destroyAllWindows()
40
```

```
41    # Save the final image as a PNG file with the same name as the video
file
42    filename = os.path.splitext(video_file_name)[0]
43    cv2.imwrite(f'{filename}_max_values.png', max_values)
```

A biblioteca OpenCV é importada com o nome cv2 (linha 1). A biblioteca numpy é importada com o nome np (linha 2). A biblioteca os é importada (linha 3).

O nome do arquivo de vídeo é definido como 'v6.avi' (linha 6). O arquivo é aberto com a função cv2.VideoCapture(), que retorna um objeto que pode ser usado para ler quadros individuais do vídeo. O objeto é armazenado em uma variável chamada 'cap' (linha 7). O primeiro quadro do vídeo é lido com a função cap.read() (linha 10). O valor de retorno da função é uma tupla que contém um valor booleano indicando se o quadro foi lido com sucesso e o próprio quadro. A altura e a largura do quadro são obtidas usando a função shape() (linha 11) da biblioteca numpy e armazenadas nas variáveis height e width, respectivamente. Uma matriz 2D de zeros é criada com as mesmas dimensões do quadro usando a função ones() (linha 14) da biblioteca numpy e multiplicando por 0 para que todos os valores iniciais sejam zero. Essa matriz é usada para armazenar os valores máximos de pixel em cada quadro do vídeo e é armazenada na variável max_values (linha 14)

O processamento do vídeo começa com um loop while (linha 16). Dentro do loop, um novo quadro é lido com a função cap.read() (linha 18). Se a leitura do quadro falhar, a execução do loop é interrompida com o comando 'break' (linha 21). O quadro lido é convertido em escala de cinza usando a função cv2.cvtColor() (linha 25) para que seja possível comparar a intensidade luminosa em uma escala de 0 a 255. A matriz resultante é armazenada em uma variável chamada 'gray' (linha 25).

Os valores máximos de pixel para cada posição na matriz 'max_values' são atualizados usando a função np.maximum(), que recebe as matrizes 'max_values' e 'gray' como argumentos (linha 28). A imagem resultante é exibida em uma janela usando a função cv2.imshow() (linha 31). O nome da janela é definido como 'Meteor Video'.

A execução do loop aguarda uma tecla ser pressionada com a função cv2.waitKey() (linha 34) para que cada frame do vídeo avance. Se a tecla 'q' for pressionada, a execução do loop é interrompida com o comando 'break' (linha 35). O arquivo de vídeo é liberado com a função cap.release() (linha 38). A janela exibindo a imagem é fechada com a função cv2.destroyAllWindows() (linha 39).

Finalmente, a imagem final é salva em um arquivo PNG usando a função cv2.imwrite() (linha 43). O nome do arquivo é gerado com a função os.splitext(), que retorna uma tupla com o nome do arquivo e sua extensão. O nome do arquivo é salvo na variável 'filename' e é usado para definir o nome do arquivo PNG, baseado no nome do arquivo de vídeo.

5.4. Seleção manual de meteoros

O objetivo deste código é permitir que o usuário assista a um vídeo e clique em um ponto na imagem em cada quadro do vídeo. O ponto clicado e o número do quadro são armazenados em um arquivo do Excel (.xlsx) que é salvo após o término do vídeo.

```
1    import os
2    import cv2
3    import openpyxl
4
5    # Create a workbook and add a sheet
6    workbook = openpyxl.Workbook()
7    sheet = workbook.active
8
9    # Set the column headings
10   sheet.cell(row=1, column=1).value = 'Frame'
11   sheet.cell(row=1, column=2).value = 'mx'
12   sheet.cell(row=1, column=3).value = 'my'
13
14   # Create a mouse callback function
15   def on_mouse_click(event, x, y, flags, params):
16       if event == cv2.EVENT_LBUTTONDOWN:
17           # Mark the clicked position with a red circle
18           cv2.circle(frame, (x, y), 1, (0, 0, 255), -1)
19           # Update the displayed frame
20           cv2.imshow("Video", frame)
21           # Store the clicked position and frame number in the .xlsx file
22           sheet.append([params['frame'], x, y])
23
24   # Get the name of the video file
25   video_file = 'v3.avi'
26   video_name = os.path.splitext(video_file)[0]
27   # Open the video file
28   video_capture = cv2.VideoCapture(video_file)
29
30
31   frame_number = 0
32   while True:
33       # Read a frame from the video
34       ret, frame = video_capture.read()
35
36       # Check if the video has ended
37       if not ret:
38           break
39
40       # Display the frame number in the top left corner of the frame
41       text = 'Frame: {}'.format(frame_number)
42       cv2.putText(frame, text, (10, 25), cv2.FONT_HERSHEY_SIMPLEX,
43                   1, (0, 255, 255), 2, cv2.LINE_AA)
44
45       # Display the frame
```

```
46        cv2.namedWindow("Video", cv2.WINDOW_NORMAL)
47                  cv2.setWindowProperty('Video',  cv2.WND_PROP_FULLSCREEN,
cv2.WINDOW_NORMAL)
48        cv2.imshow("Video", frame)
49
50        # Set the mouse callback function to handle clicks
51                cv2.setMouseCallback("Video",  on_mouse_click,  {'frame':
frame_number})
52
53        # Wait for the user to press a key
54        if cv2.waitKey(0) & 0xFF == ord('q'):
55            break
56
57        frame_number += 1
58
59    # Release the video capture and destroy the window
60    video_capture.release()
61    cv2.destroyAllWindows()
62
63    # Create xlsx file name based on video title
64    xlsx_file = f"{video_name}_clicked.xlsx"
65    # Save the .xlsx file
66    workbook.save(xlsx_file)
```

Na linha 1, a biblioteca "os" é importada para permitir que o nome do arquivo do Excel seja criado com base no nome do arquivo de vídeo. Na linha 2, a biblioteca "cv2" é importada para permitir o uso das funções OpenCV para processamento de imagem e vídeo. Na linha 3, a biblioteca "openpyxl" é importada para permitir a criação e manipulação de arquivos do Excel.

Na linha 6, uma nova pasta de trabalho do Excel é criada usando a função openpyxl.Workbook(). Na linha 7, a planilha ativa é armazenada em uma variável chamada "sheet". As colunas da planilha do Excel são definidas nas linhas 10 a 12. A primeira coluna é intitulada "Frame" para armazenar o número do quadro. A segunda coluna é intitulada "mx" para armazenar a posição x do ponto clicado. A terceira coluna é intitulada "my" para armazenar a posição y do ponto clicado.

A função "on_mouse_click()" é definida nas linhas 15 a 23 como um manipulador de eventos para o mouse. Se um botão do mouse for pressionado e solto (evento "LBUTTONDOWN"), um círculo vermelho é desenhado na posição do clique na imagem (linha 18). O número do quadro atual e a posição do clique são adicionados à planilha do Excel (linha 22). O nome do arquivo de vídeo é obtido na linha 25 e armazenado em "video_file". Na linha 26, o nome do arquivo é modificado para remover a extensão usando a função os.path.splitext(). O objeto de captura de vídeo é criado na linha 28 usando a função cv2.VideoCapture().

O número do quadro é definido como 0 na linha 31. O loop "while True" começa na linha 32 e continua até que o vídeo tenha sido completamente lido (linha 37). A cada iteração do loop, um quadro do vídeo é lido usando a função cap.read() na linha 34. Se a leitura do quadro falhar, o loop é encerrado (linha 37). O número do quadro atual é adicionado ao canto superior esquerdo da imagem usando a função cv2.putText() na linha 42. Isto é feito para dar ao usuário melhor controle e noção de localização no video. A imagem é exibida em uma janela com o nome "Video" na linha 46 usando a função cv2.imshow(). A função cv2.namedWindow() na linha 46 define as propriedades da janela, permitindo que a imagem seja exibida em tela cheia. A função cv2.setMouseCallback() na linha 51 define a função de retorno de chamada do mouse para lidar com eventos de clique. O loop aguarda a entrada do usuário na linha 54 usando a função cv2.waitKey(). Se o usuário pressionar a tecla 'q', o loop é encerrado (linha 55). O número do quadro atual é incrementado na linha 57.

Após o término do vídeo, o objeto de captura de vídeo é liberado e a janela é destruída usando as funções cv. Na linha 64 o nome do arquivo Excel é criado baseado no nome do arquivo de vídeo, e na linha seguinte o arquivo é salvo no diretório de execução.

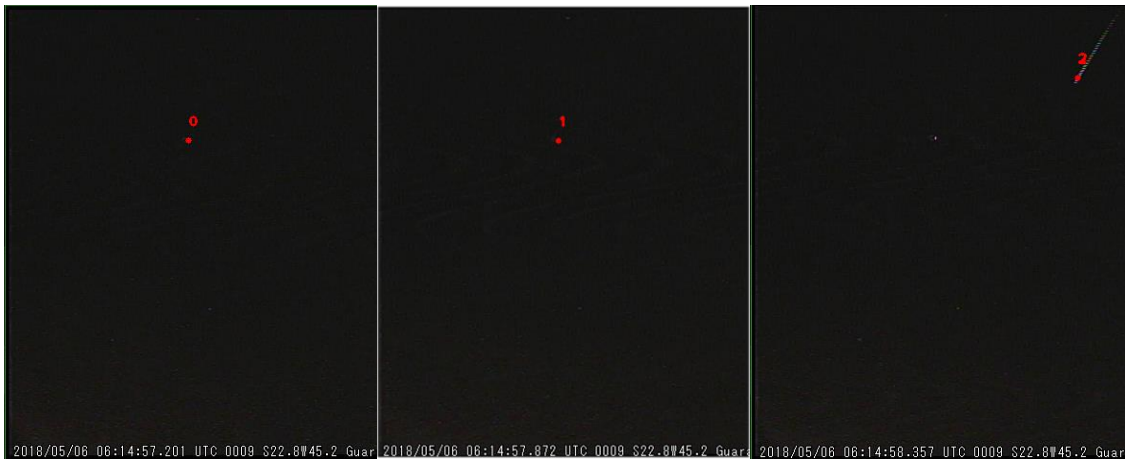
6. ANÁLISE DOS RESULTADO

Esta secção é dedicada a avaliar se as saídas dos programas geraram os resultados esperados que possibilitem a posterior comparação de dados entre a seleção manual da posição do meteoro e as posições obtidas automaticamente.

6.1. Detecção e rastreamento

Os resultados da análise de detecção e rastreamento de meteoros mostram que o programa desenvolvido foi capaz de realizar o rastreamento do meteoro ao longo de todo o vídeo, atribuindo um ID único a cada objeto detectado. Adicionalmente, as posições de centroide foram armazenadas com sucesso. Abaixo está a imagem de saída do programa, que mostra o meteoro identificado como objeto 2, logo após duas outras detecções.

Figura 8:Resultadovisual de rastreamento de meteoro com ID



Fonte: autoria própria.

Para finalizar, os resultados armazenados na tabela Excel confirmam o resultado visual.

Tabela 2:Tabela de resultado de rastreamento e atribuição de ID objetos

id	cx	cy	frame
0	193	142	3
1	193	143	22
1	193	143	23
2	386	6	29
2	382	12	30
2	377	20	31
2	371	29	32
2	366	37	33
2	361	46	34
2	358	51	35
2	352	59	36
2	347	68	37
2	341	79	38
2	336	87	39
2	331	95	40

Os objetos 0 e 1 ocorrem, pois, sua área é maior do que a área estipulada de 10 pixels como critério para tamanho mínimo de meteoro. Além disso, segundo as coordenadas destes objetos é depreendido que na verdade se tratam do mesmo objeto pois não há diferença entre suas coordenadas (x,y) , porém são tratados como distintos pelo programa pois há uma grande diferença de frames entre sua detecção novamente.

6.2.Rastro de meteoro

Os resultados de produção de rastro de meteoros produzem saídas como a mostrada abaixo.

A imagem mostra o rastro completo de um bólido, satisfatório para uma análise preliminar. Por hora, o objetivo deste modulo de rastro foi atingido, e será confirmado na comparação de vários casos de meteoro na próxima secção.

6.3.Comparação

Neste estudo, foram realizadas análises em seis vídeos que registraram eventos de meteoro sob diferentes condições disponibilizados pela BRAMON. Todos os vídeos possuem resolução de 720x480 pixels a 30 frames por segundo. A análise foi feita em um monitor de 21 polegadas em tela cheia, ou seja, cada pixel dos vídeos nesta tela possui um tamanho de 0,616mm (0,0616 cm). Cada vídeo será identificado por um código que varia de V1 à V6 junto a data de registro do meteoro.

A análise dos resultados consiste de duas comparações, visual e numérica. Na análise visual, os pontos de centroide do meteoro obtidos manualmente e os obtidos pelo código de seleção manual são plotados sobre uma imagem de rastro do meteoro obtido pelo código gerador de rastro. Esta comparação visual permite avaliar como varia a posição de centroide

em relação a alteração de magnitude e posição do meteoro. Já na análise numérica, são avaliados o quão distante dos pontos de referência (seleção manual) estão os pontos obtidos automaticamente. Neste estudo os pontos obtidos manualmente pela seleção de um operador serão chamados de “pontos manuais”. Já os pontos obtidos através do código proposto serão chamados de “pontos automáticos”. Vale notar que para um mesmo evento de meteoro, as análises manuais podem diferir devido a variáveis como experiencia do operador, tamanho e resolução da tela e qualidade do vídeo.

V3# 2018/04/26 – 03:10:01.576 UTC 0012 S22.8 W45.2 Guaratinguetá | D. Mourão GDOP1/SP BRAMON

O primeiro caso de meteoro analisado neste estudo apresenta características clássicas como pouca variação de brilho e duração de 0,5 segundos. A representação visual abaixo mostra a comparação entre os pontos obtidos manualmente e através do código.

Figura 9: Comparação entre método manual (vermelho) e método automático (azul) para o meteoro V3.

Fonte: autoria própria.

Neste caso é possível notar que os pontos automáticos apresentam um espaçamento relativamente uniforme em relação os pontos manuais. Isto se deve ao fato de que o meteoro

possui pouca variação luminosa e de que não ocorreu saturação no sensor ótico da câmera. Este é um cenário ideal para as operações morfológicas aplicadas na imagem uma vez que devido à baixa variação de luminosidade o evento de meteoro não muda drasticamente de forma, o que desloca a sua posição de centroide. Além disso, a análise automática conseguiu capturar o início e fim do evento com precisão humana.

Na tabela abaixo estão presentes as coordenadas de centroide em pixels, obtidas manualmente (Mx,My) e automaticamente (Cx,Cy).

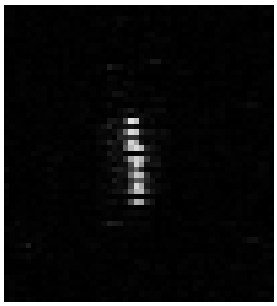
Tabela 3:Relação entre distância D8 dos pontos manuais e automáticos por duração do evento de meteoro V3

Frame	Mx	Cx	Dist x	My	Cy	Dist y	D8(m,c)
30	257	256	1	72	77	-5	5
31	258	257	1	79	84	-5	5
32	258	257	1	88	92	-4	4
33	259	258	1	97	101	-4	4
34	259	259	0	105	110	-5	5
35	260	260	0	113	115	-2	2
36	260	259	1	122	120	2	2
37	261	259	2	127	130	-3	3
38	260	260	0	135	136	-1	1
39	261	261	0	143	147	-4	4
40	262	262	0	152	155	-3	3
41	262	263	-1	161	162	-1	1
42	263	263	0	167	171	-4	4
43	264	264	0	175	178	-3	3
44	265	264	1	183	186	-3	3
45	265	266	-1	188	197	-9	9

Fonte: autoria própria

As colunas “Dist x” e “Dist y” representam a distância em pixels entre as componentes x e y das coordenadas manuais e automáticas. A coluna “D8(m,c)” representa a distância D8 ou distância de Chebyshev entre pontos no mesmo frame. Segundo a tabela, para a componente x em 93,75% do tempo a diferença entre elas é menor ou igual a 1 pixel. Já para a coordenada y, este valor é de apenas 12,5%. O que está acontecendo neste caso é que esta coordenada está consistentemente deslocada de 4 pixels negativos. A imagem da subtração entre frames consecutivos realizada no código de detecção e rastreamento na linha 41 ajuda a explicar a situação.

Figura 10: Imagem binaria do meteoro V3 entre subtração de frames



Fonte: autoria própria

Analisando a imagem é possível notar que se trata de um meteoro com a imagem de sua ponta alongada, o que causa o centroide se deslocar para “cima” ao invés da ponta do meteoro, ocasionando a diferença negativa em pixels.

A tabela abaixo apresenta dados que relacionam a distância D8, em pixels, entre dois pontos em um frame (pontos manuais e pontos automáticos) e a frequência com que essa distância é menor ou igual a determinado número.

Tabela 4 :Relação entre distância D8 dos pontos manuais e automáticos por duração do evento de meteoro V3

Distancia D8 (pixels)	Porcentagem de frames com distancia menor ou igual (%)
0	0
1	12,5
2	25,0
3	50,0
4	75,0
5	93,8
9	100

Fonte: autoria própria

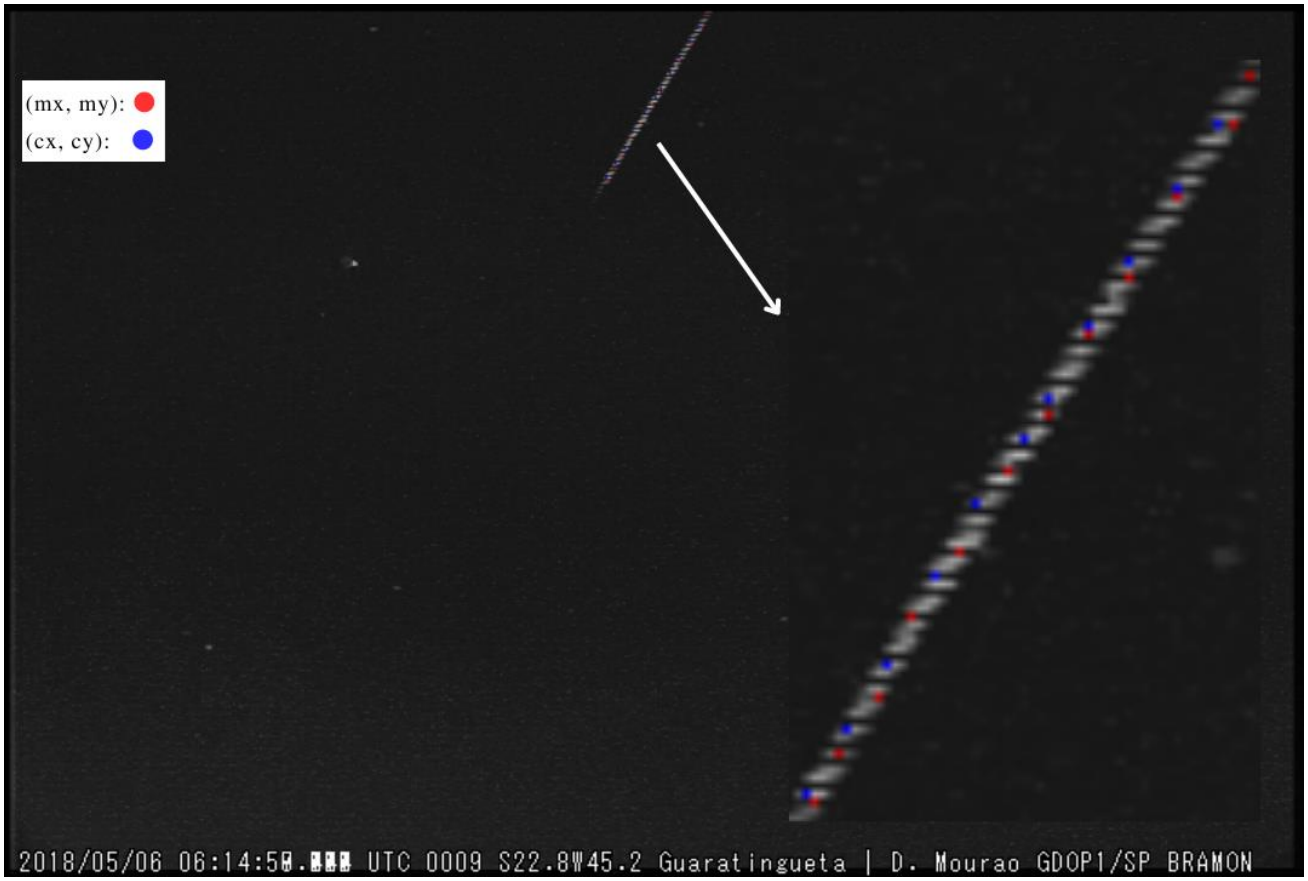
Esta análise é importante para aferir a precisão dos pontos automáticos em relação aos pontos manuais. Apesar do valor máximo de distância entre pontos ser de 9 pixels, ele ocorre apenas uma vez segundo a tabela [3]. Em 93.8% dos frames a distância entre os pontos é menor ou igual a 5 pixels, o que apresenta uma boa aproximação para uma análise automática.

V4# 2018/05/06 – 06:14:50.100 UTC 0012 S22.8 W45.2 Guaratinguetá | D. Mourão GDOP1/SP BRAMON

O meteoro V4 apresenta características muito semelhantes ao anterior, com uma duração de 0,4 segundos e pouca variação em seu brilho e um rastro bastante distinto. A

representação visual abaixo mostra a comparação entre os pontos obtidos manualmente e através do código.

Figura 11: Comparação entre método manual (vermelho) e método automático (azul) para o meteoro V4.



Fonte: autoria própria.

Analisando a duração do meteoro, é possível notar que a análise automática foi capaz de capturar o primeiro instante do evento até o seu último. Os pontos obtidos automaticamente estão consistentemente dentro da linha de traço do meteoro e espaçados de maneira ordenada assim como os pontos manuais, o que é confirmado pela tabela de comparação de coordenadas abaixo. Alguns pontos estão ligeiramente deslocados para trás em relação a direção de movimento do meteoro, o que indica que a mesma explicação de alongamento do caso anterior se aplica aqui.

Tabela 5:Comparação entre os pontos obtidos manualmente (Mx,My) e os pontos obtidos atravez do código proposto (Cx,Cy) para o meteoro V4

Frame	Mx	Cx	Dist x	My	Cy	Dist y	D8(m,c)
30	386	386	0	6	6	0	0
31	384	382	2	12	12	0	2
32	377	377	0	21	20	1	1
33	371	371	0	31	29	2	2
34	366	366	0	38	37	1	1
35	361	361	0	48	46	2	2
36	356	358	-2	55	51	4	4
37	350	352	-2	65	59	6	6
38	344	347	-3	73	68	5	5
39	340	341	-1	83	79	4	4
40	335	336	-1	90	87	3	3
41	332	331	1	96	95	1	1

Fonte: autoria própria

Segundo estes dados, a distância máxima entre os pontos manuais e os pontos automáticos é de 6 pixels, e ocorre apenas uma vez. Um valor de 6 pixels para um observador em uma tela de 21 polegadas com resolução de 720x480 pixels corresponde a 0,3cm. Abaixo é apresentado uma tabela relacionando a porcentagem dos frames em que a distância entre os pontos é menor ou igual a determinado valor.

Tabela 6 :Relação entre distância D8 dos pontos manuais e automáticos por duração do evento de meteoro V4

Distancia D8 (pixels)	Porcentagem de frames com distancia menor ou igual (%)
0	8,3
1	33,3
2	58,3
3	66,7
4	83,3
5	91,7
6	100,0

Fonte: autoria própria

Como era de se esperar a coincidência exata de pontos é rara, e mais da metade dos das detecções estão a 2 ou menos pixels de distância da referência.

V2# 2018/04/26 – 03:10:01.576 UTC 0012 S22.8 W45.2 Guaratinguetá | D. Mourão GDOP1/SP BRAMON

Este evento de meteoro serve como um bom exemplo em favor da análise visual. Nela é possível observar ao lado esquerdo do rastro de meteoro um tracejado mais sutil que se trata

de um artefato de câmera e não de um componente do meteoro. Na análise numérica será possível ver claramente o seu efeito no resultado. O evento tem duração de 0,66 segundos.

Figura 12: Comparação entre método manual (vermelho) e método automático (azul) para o meteoro V2.

Fonte: autoria própria.

O código foi capaz de capturar desde o primeiro até o último instante do meteoro. Ambos momentos não apresentam artefato de vídeo, o que proporcionou uma análise mais precisa dentro do rastro do meteoro. Segundo a imagem, a partir do terceiro frame é possível observar que o ponto automático foi deslocado ligeiramente para fora da linha de rastro do meteoro. Isto se deve ao artefato de vídeo, que foi processado nas operações morfológicas como parte do meteoro e com isso a imagem gerada deste objeto não representava fielmente o meteoro, o que fez o centroide da figura se deslocar para aquela região. Porém mesmo nestas circunstancias os pontos estão consistentemente separados paralelamente ao traçado do meteoro. A partir dos últimos 5 frames com a diminuição do artefato de vídeo os pontos começar a se alinhar novamente. Os dados da tabela abaixo ajudam a corroborar esta explicação.

Tabela 7:Comparação entre os pontos obtidos manualmente (Mx,My) e os pontos obtidos através do código proposto (Cx,Cy) para o meteoro V4

Frame	Mx	Cx	Dist x	My	Cy	Dist y	D8(m,c)

30	267	265	2	6	8	-2	2
31	262	259	3	12	12	0	3
32	259	255	4	16	15	1	4
33	253	249	4	22	22	0	4
34	248	243	5	27	27	0	5
35	242	239	3	33	32	1	3
36	238	234	4	38	38	0	4
37	233	229	4	45	44	1	4
38	228	224	4	50	50	0	4
39	223	220	3	56	55	1	3
40	219	215	4	61	60	1	4
41	214	210	4	66	66	0	4
42	209	205	4	72	72	0	4
43	205	201	4	78	78	0	4
44	201	197	4	83	84	-1	4
45	195	192	3	88	89	-1	3
46	191	189	2	95	96	-1	2
47	186	184	2	101	101	0	2
48	182	182	0	106	104	2	2
49	178	176	2	111	109	2	2

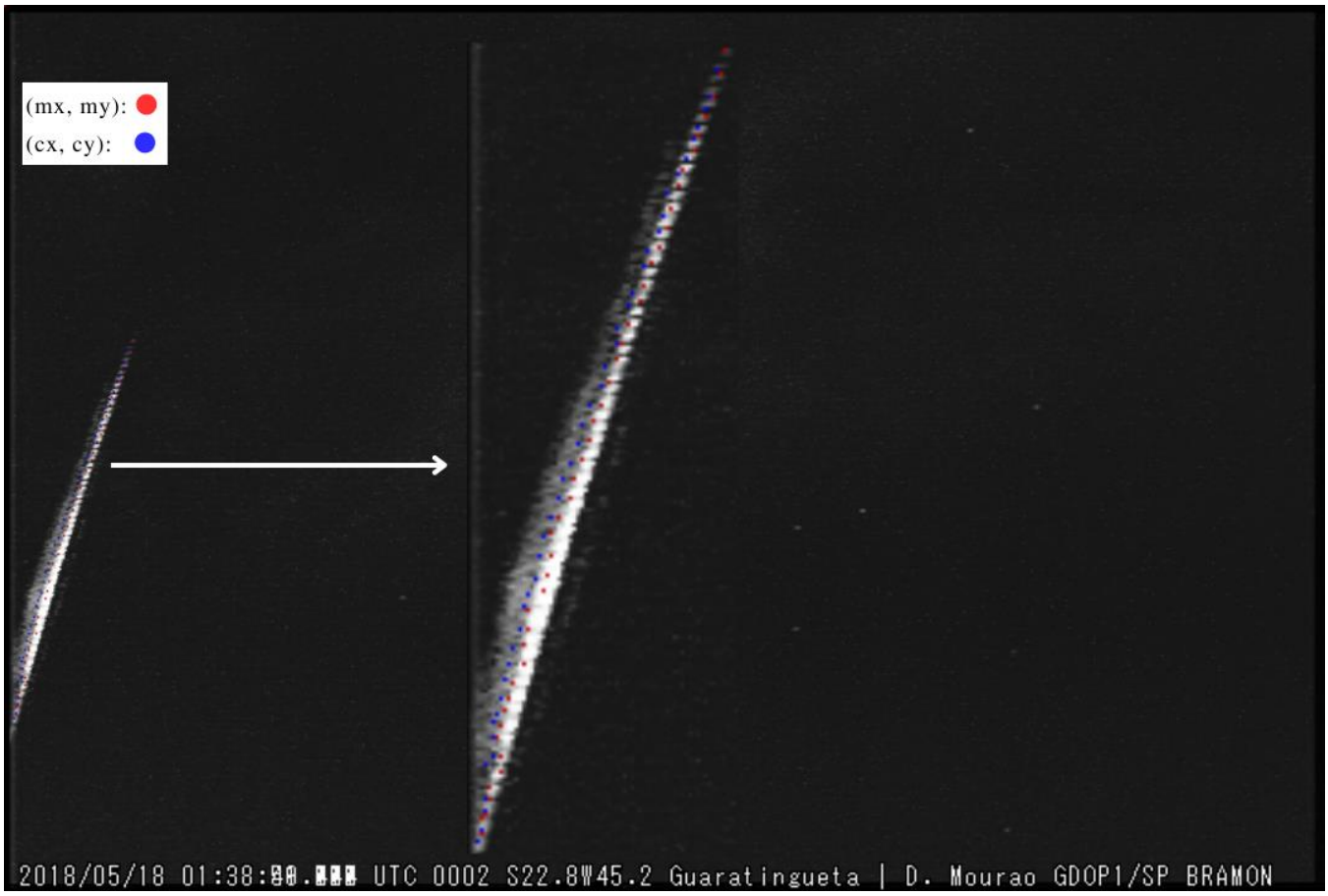
Fonte: autoria própria

A partir do primeiro frame a distância D8 aumenta consistentemente até que começa a diminuir por volta do frame 45, coincidindo com o artefato de vídeo.

V6# 2018/04/26 – 03:10:01.576 UTC 0012 S22.8 W45.2 Guaratinguetá | D. Mourão GDOP1/SP BRAMON

Este meteoro apresenta características similares ao anterior. A imagem abaixo mostra o traço ao longo deu seus 1,5 segundos de duração.

Figura 13: Comparação entre método manual (vermelho) e método automático (azul) para o meteoro V6



Fonte: autoria própria.

Novamente este é um caso de um meteoro altamente assimétrico com um artefato de vídeo ao lado esquerdo do seu traço. Com uma intensidade luminosa com pouca variação, é possível observar que o método proposto consegue capturar com sucesso desde o início até final do evento. Sendo que os pontos dos primeiros seis frames apresentam uma precisão maior em relação restante do vídeo devido a presença do artefato de vídeo a partir deste ponto. Em geral os pontos apresentam um espaçamento uniforme assim como os pontos os manuais.

A tabela abaixo fornece mais informações para completar a compreensão do evento.

Tabela 8:Comparação entre os pontos obtidos manualmente (Mx,My) e os pontos obtidos através do código proposto (Cx,Cy) para o meteoro V6

Frame	Mx	Cx	Dist x	My	Cy	Dist y	D8(m,c)
30	67	67	0	186	186	0	0
31	68	65	3	193	193	0	3
32	66	64	2	193	196	-3	3
33	64	62	2	198	201	-3	3
34	62	61	1	203	204	-1	1
35	61	59	2	207	209	-2	2

36	58	57	1	212	213	-1	1
37	57	54	3	216	218	-2	3
38	55	53	2	222	224	-2	2
39	54	52	2	227	228	-1	2
40	52	49	3	232	233	-1	3
41	50	48	2	236	237	-1	2
42	48	45	3	242	244	-2	3
43	47	44	3	246	247	-1	3
44	44	41	3	252	253	-1	3
45	42	41	1	257	257	0	1
46	41	38	3	261	263	-2	3
47	39	37	2	267	268	-1	2
48	37	34	3	273	273	0	3
49	35	32	3	277	278	-1	3
50	34	31	3	282	283	-1	3
51	32	29	3	287	288	-1	3
52	30	27	3	292	292	0	3
53	29	26	3	297	297	0	3
54	26	24	2	302	302	0	2
55	24	22	2	306	307	-1	2
56	24	21	3	312	312	0	3
57	23	20	3	317	318	-1	3
58	22	18	4	321	322	-1	4
59	18	17	1	326	325	1	1
60	18	16	2	331	331	0	2
61	17	14	3	335	336	-1	3
62	17	13	4	340	340	0	4
63	14	12	2	345	344	1	2
64	13	11	2	349	349	0	2
65	12	10	2	352	353	-1	2
66	11	9	2	356	355	1	2
67	10	9	1	359	359	0	1
68	11	9	2	364	364	0	2
69	11	7	4	368	366	2	4
70	8	8	0	372	372	0	0
71	8	7	1	375	375	0	1
72	7	7	0	379	378	1	1
73	6	6	0	380	383	-3	3
74	6	5	1	384	386	-2	2

Fonte: autoria própria

Como esperado, os oito primeiros frames, apresentam uma distância D8 entre os pontos menor em relação ao meio do vídeo devido a presença do artefato. A distância D8 entre os

pontos volta a diminuir nos últimos doze frames, coincidindo com a ocultação do artefato pelo limite do campo de visão da câmera. Vale notar que, a distorção ou erro de centroide ocorre predominantemente nas coordenadas x. Isto se deve ao fato de que o artefato de vídeo se propagou mais na direção horizontal do vídeo. Outro ponto importante é em relação aos pontos manuais, que segundo inspeção visual apresentam variações que fogem ligeiramente a linha de rastro do meteoro, mostrando a dificuldade e se analisar alguns casos.

A tabela da comparação de distancias mostra que foi atingido uma boa precisão apesar das dificuldades apresentadas. Mais de 90% dos pontos apresentam uma distância menor ou igual a 3 pixels em relação a referência.

Tabela 9 :Relação entre distância D8 dos pontos manuais e automáticos por duração do evento de meteoro V6

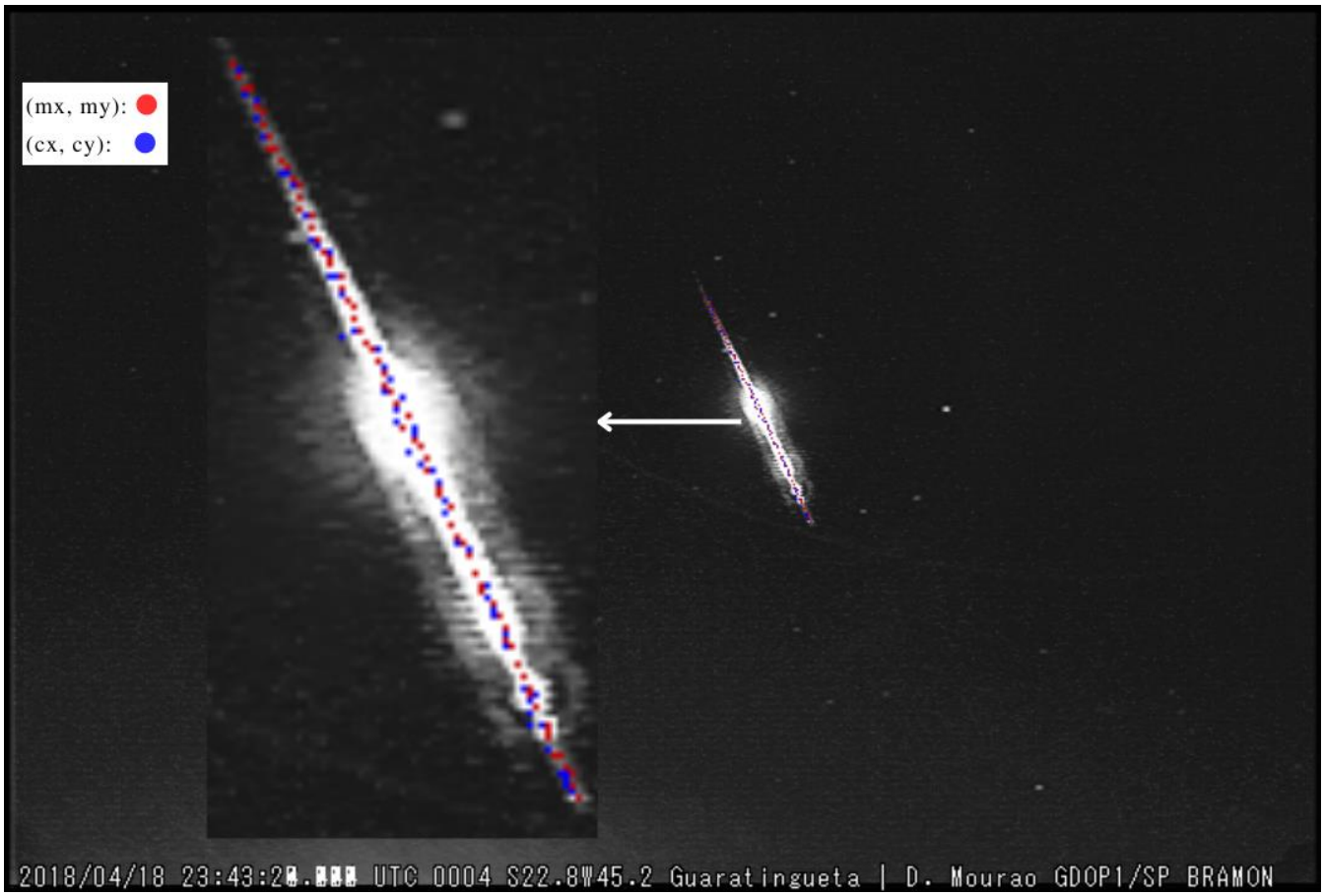
Distancia D8 (pixels)	**Porcentagem de frames com distancia menor ou igual (%)**
0	4,4
1	20,0
2	51,1
3	93,3
4	100,0

Fonte: autoria própria

V1# 2018/04/26 – 03:10:01.576 UTC 0012 S22.8 W45.2 Guaratinguetá | D. Mourão GDOP1/SP BRAMON

Este é o primeiro caso de um bólido analisado pelo programa. Embora o estudo não tenha sido feito com foco em fireballs e sim em meteoros com baixa variação luminosa, os resultados mostram dados positivamente inesperados. O traço do bólido pode ser visto na imagem abaixo, com duração de 2,3 segundos.

Figura 14:: Comparação entre método manual (vermelho) e método automático (azul) para o meteoro V1.



Fonte: autoria própria.

O programa foi capaz de realizar o rastreamento em toda a duração do evento, desde o seu surgimento com baixa intensidade luminosa, passando pela sua fase de alta magnitude até desaparecer novamente.

Todos os pontos se encontram dentro do rastro de meteoro a uma distância consistente entre eles. Diferente dos meteoros com maior intensidade analisados anteriormente, este apresenta simetria em sua forma de traço, em contrapartida ao meteoro V6 e V2, que apresentavam artefatos luminosos apenas de um lado do rastro principal do evento. Essa característica simétrica circular, contribui para que a imagem passada à função de cálculo de centroide devolva dados que melhor correspondem a posição real do meteoro.

Abaixo, os dados numéricos de posição revelam mais sobre o desempenho do código para este caso.

Tabela 10:Comparação entre os pontos obtidos manualmente (Mx,My) e os pontos obtidos através do código proposto (Cx,Cy) para o meteoro V1

Frame	Mx	Cx	Dist x	My	Cy	Dist y	D8(m,c)
29	379	380	-1	158	159	-1	1
30	380	380	0	160	160	0	0
31	381	381	0	162	163	-1	1
32	382	383	-1	162	164	-2	2
33	383	383	0	165	165	0	0
34	384	383	1	166	167	-1	1
35	384	384	0	168	170	-2	2
36	385	385	0	170	170	0	0
37	385	385	0	172	170	2	2
38	386	387	-1	172	174	-2	2
39	387	387	0	174	176	-2	2
40	388	387	1	175	176	-1	1
41	389	389	0	177	178	-1	1
42	390	388	2	178	176	2	2
43	390	389	1	180	178	2	2
44	390	390	0	182	182	0	0
45	392	391	1	183	183	0	1
46	392	392	0	185	185	0	0
47	393	392	1	187	187	0	1
48	394	393	1	189	188	1	1
49	395	395	0	190	189	1	1
50	395	395	0	191	191	0	0
51	397	396	1	193	193	0	1
52	397	395	2	195	193	2	2
53	398	397	1	197	196	1	1
54	399	398	1	198	197	1	1
55	399	399	0	200	202	-2	2
56	400	397	3	202	203	-1	3
57	401	403	-2	204	205	-1	2
58	402	403	-1	205	207	-2	2
59	403	404	-1	207	208	-1	1
60	404	405	-1	209	210	-1	1
61	404	404	0	211	212	-1	1
62	405	407	-2	212	213	-1	2
63	406	406	0	214	215	-1	1
64	408	406	2	216	217	-1	2
65	407	409	-2	218	218	0	2
66	409	409	0	220	219	1	1
67	410	408	2	221	222	-1	2
68	411	410	1	223	224	-1	1

69	411	412	-1	225	225	0	1
70	413	413	0	228	227	1	1
71	413	414	-1	229	230	-1	1
72	415	414	1	231	232	-1	1
73	415	415	0	234	234	0	0
74	416	416	0	236	237	-1	1
75	417	418	-1	237	238	-1	1
76	418	418	0	239	239	0	0
77	419	419	0	242	242	0	0
78	420	421	-1	244	244	0	1
79	420	421	-1	245	246	-1	1
80	422	422	0	247	249	-2	2
81	423	422	1	249	248	1	1
82	424	424	0	251	251	0	0
83	424	424	0	253	252	1	1
84	425	425	0	254	254	0	0
85	426	424	2	257	254	3	3
86	427	428	-1	259	263	-4	4
87	428	429	-1	261	262	-1	1
88	428	427	1	262	261	1	1
89	429	428	1	264	265	-1	1
90	431	428	3	267	267	0	3
91	431	430	1	268	267	1	1
92	431	431	0	269	271	-2	2
93	432	433	-1	272	272	0	1
94	433	433	0	272	275	-3	3
95	434	434	0	274	276	-2	2
96	435	434	1	275	277	-2	2
97	435	434	1	277	275	2	2
98	436	435	1	279	278	1	1

Fonte: autoria própria.

Tabela 11 :Relação entre distância D8 dos pontos manuais e automáticos por duração do evento de meteoro V6

Distancia D8 (pixels)	**Porcentagem de frames com distancia menor ou igual (%)**
0	15,7
1	64,3
2	92,9
3	98,6
4	100,0

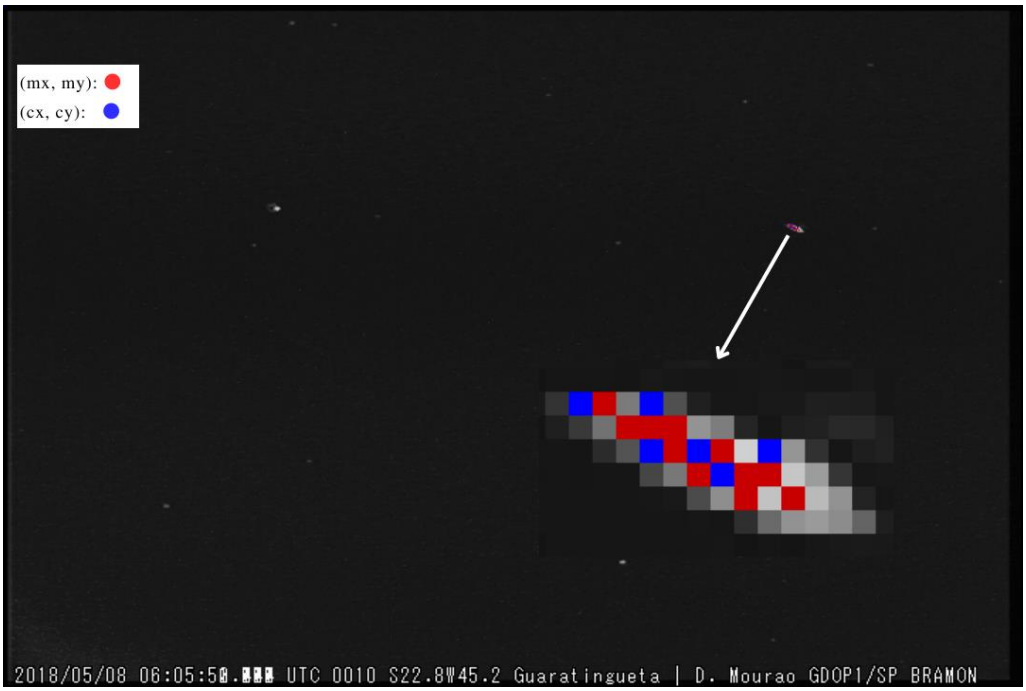
Fonte: autoria própria.

Os números da tabela revelam que até o momento este caso foi que apresentou maior precisão, com quase 16% dos pontos coincidindo, 64 % com distância de até um pixel e um rápido crescimento para 93% com distancias de até dois pixels.

V5# 2018/04/26 – 03:10:01.576 UTC 0012 S22.8 W45.2 Guaratinguetá | D. Mourão GDOP1/SP BRAMON

O meteoro V5 apresenta características únicas em relação aos analisados até agora. Sua duração é de apenas 0,4 segundos e seu rastro é bastante curto, como mostrado na comparação visual dos pontos abaixo.

Figura 15: Comparação entre método manual (vermelho) e método automático (azul) para o meteoro V5.



Fonte: autoria própria.

A imagem informa que a análise automática foi capaz de realizar o rastreamento em toda duração do evento se mantendo dentro do rastro de meteoro. A rapidez e tamanho na tela deste meteoro torna até mesmo a analise manual difícil, como é possível observar pontos vermelhos. Por se tratar de um evento tão rápido e curto, as coordenadas de interesse deste meteoro podem ser de apenas o seu ponto inicial e final. A análise numérica revela mais informações.

Tabela 12:Comparação entre os pontos obtidos manualmente (Mx,My) e os pontos obtidos através do código proposto (Cx,Cy) para o meteoro V5

Frame	Mx	Cx	Dist x	My	Cy	Dist y	D8(m,c)
29	553	552	1	154	154	0	1

30	554	554	0	155	155	0	0
31	554	554	0	155	155	0	0
32	555	555	0	155	155	0	0
33	556	555	1	155	154	1	1
34	556	555	1	155	156	-1	1
35	556	557	-1	156	156	0	1
36	557	557	0	157	156	1	1
37	558	558	0	156	157	-1	1
38	559	559	0	158	157	1	1
39	559	560	-1	157	156	1	1
40	560	560	0	157	157	0	0

Fonte: autoria própria.

Tabela 13 :Relação entre distância D8 dos pontos manuais e automáticos por duração do evento de meteoro V6

Distancia D8 (pixels)	**Porcentagem de frames com distancia menor ou igual (%)**
0	33,3
1	100,0

Fonte: autoria própria.

A imagem mostra que a análise automática foi bem-sucedida, capaz de realizar o rastreamento em toda duração do evento se mantendo dentro do rastro de meteoro.

7. CONCLUSÃO

O código apresentado é um exemplo de como utilizar técnicas de processamento de imagens para detectar objetos em um vídeo e acompanhar seu movimento ao longo do tempo. A abordagem utilizada consiste puramente de operações morfológicas, que em vista de outros métodos de processamento de imagens, é relativamente simples, porém menos intensiva computacionalmente o que possibilitaria a execução do software em um maior número de computadores de mesa. Métodos mais sofisticados de análise de imagem envolvendo aprendizado de máquina, como redes neurais convolucionais e inteligência artificial podem também ser implementados para aumentar a precisão, porém a um maior custo computacional e operacional.

A possibilidade de alcançar ou até superar a precisão humana, eliminando variações que ocorrem pela diferença de operadores, através de métodos morfológicos é um tema que demanda extensos estudos e testes. Tais investigações requerem um significativo investimento de tempo e esforço. Até mesmo o software UFO analyzer, em utilização, apresenta a perda de rastreio do centroide de meteoros, ou regressão espacial na linha de rastro. Em conclusão, o

código apresentado é um bom ponto de partida para se trabalhar com detecção de objetos em vídeos, mas há muitas outras abordagens que podem ser utilizadas para melhorar a precisão e desempenho da detecção de objetos em diferentes cenários.

8. ESTUDOS FUTUROS

8.1. Melhorias em operações morfologias

Embora o código apresentado seja eficaz na detecção de meteoros em vídeos e no armazenamento dos dados de suas posições, existem outras abordagens que poderiam ser utilizadas para aprimorar ainda mais esse processo.

Neste estudo, observou-se que os pontos de detecção de centroide dos meteoros se alinham de maneira muito linear ao rastro do meteoro. Dessa forma, é possível aplicar uma regressão linear nos pontos de centroide do meteoro ao longo de toda a sua duração para obter a melhor linha que passe pelo traço do meteoro. Essa linha que descreve a trajetória do meteoro pode, então, ser utilizada para criar uma máscara sobre o vídeo do evento, isolando todo o resto do vídeo.

Além disso, a espessura da linha e da máscara poderia ser ajustada para aumentar a precisão do rastreamento e mitigar os erros observados no estudo em conjunto com um filtro de velocidade de pixels, afim de eliminar os pontos em que há um regressão na posição espacial do centroide do meteoro.

8.2.Machine Learning

Uma abordagem possível para a detecção de meteoros em vídeo é a utilização de redes neurais convolucionais (CNNs), que são capazes de extrair características relevantes das imagens por meio de camadas de convolução e pooling. Como a BRAMON possui um grande acervo de eventos de meteoro, o treinamento da CNN pode ser realizado com um conjunto de imagens rotuladas de meteoros e imagens com artefatos semelhantes a meteoros com intuito de aprender o que não são meteoros. Isso permite que a CNN aprenda a distinguir características relevantes dos meteoros, como sua forma, centroide e trajetória e para que assim possa ser usada para identificar novos meteoros em vídeos. Feito esta etapa, o modulo de rastreamento deste estudo ainda poderia ser utilizado, bastando passar as posições de centroide encontrados pela CNNs à função tracker.

8.3.Astrometria

Um estudo futuro possível é utilizar as ferramentas da Global Meteor Network (GMN) para realizar a astrometria dos meteoros. A Rede Global de Meteoros (GMN) é uma rede internacional de câmeras automáticas e infraestrutura relacionada, projetada com o objetivo de detectar e rastrear meteoros na atmosfera da Terra. A rede é composta por mais de 300 estações de câmeras distribuídas em mais de 40 países, sendo que cada estação está equipada com uma câmera digital de alta resolução e um software especializado de detecção de meteoros [10]

A organização possui um conjunto de softwares próprios para análise de meteoros, denominado Raspberry Pi Meteor Software (RMS). Um dos módulos desse software, o Skyfit2, é capaz de realizar a astrometria para o apontamento de câmera. Em outras palavras, o programa fornece os parâmetros necessários para converter as coordenadas das imagens de vídeo (obtidas neste estudo) em coordenadas reais.

As instruções para converter coordenadas de imagens em coordenadas equatoriais, utilizando um conjunto de equações e parâmetros, estão descritas no apêndice do artigo "The Global Meteor Network - Methodology and First Result [11]". Além disso, foi realizado um estudo preliminar detalhado sobre o processo de instalação e operação do software Skyfit2 em Português, devido à sua relativa complexidade para usuários finais. Também foi desenvolvido um algoritmo geral para implementação das equações de conversão de coordenadas mencionadas no artigo. O estudo preliminar detalhado pode ser encontrado na seção de apêndices.

9. REFERENCIAS

[1] R. Langhi, “PATRULHAMENTO INVESTIGATIVO DO CÉU POR IMAGEAMENTO AUTOMÁTICO DE METEOROS”, bauru, ago. 2014.

[2] SonataCo, “SonataCo.com”. https://sonotaco.com/soft/e_index.html (acessado 6 de janeiro de 2023).

[3] British Council, “POLÍTICAS PÚBLICAS PARA O ENSINO DE INGLÊS”, São Paulo, 2019. Acessado: 6 de janeiro de 2023. [Online]. Disponível em: https://www.britishcouncil.org.br/sites/default/files/final-publicacao_politicaspublicasingles-compressed.pdf

[4] T. C. Marsola e A. C. Lorena, “METEOR DETECTION USING DEEP CONVOLUTIONAL NEURAL NETWORKS”, em *Anais do 14º Simpósio Brasileiro de Automação Inteligente*, 2019, p. 3–4. doi: 10.17648/sbai-2019-112456.

[5] R. M. Silva, A. C. Lorena, e T. A. Almeida, “Detecting the presence of meteors in images: new collection and results”, em *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional (ENIAC 2018)*, out. 2018, p. 128–139. doi: 10.5753/eniac.2018.4410.

[6] NASA - Solar System Exploration, “What’s the difference between a meteor, meteoroid, and meteorite?”, 15 de dezembro de 2022. https://solarsystem.nasa.gov/asteroids-comets-and-meteors/meteors-and-meteorites/overview/?page=0&per_page=40&order=id+asc&search=&condition_1=meteor_shower%3Abody_type (acessado 7 de janeiro de 2023).

[7] R. Korotev, “EARTH AND PLANETARY SCIENCES - Some Meteorite Information”, *Washington University in St.Louis*, 2021. https://sites.wustl.edu/meteoritesite/items/how-big-are-meteorites/#:~:text=Here%20are%20some%20statistics.,283%20g%20(10%20oz.) (acessado 7 de janeiro de 2023).

[8] R. Lunsford, *Meteors and How to Observe Them*, 2009ª edição. 2008.

[9] K. Pavid, “The Imilac meteorite: a gem as old as the solar system”, *Natural History Museum*, 9 de março de 2021. https://www.nhm.ac.uk/discover/imilac-meteorite-gem-as-old-solar-system.html (acessado 7 de janeiro de 2023).

[10] GMN - Website, “Global Meteor Network Overview”, *Global Meteor Network’s wiki page*, 28 de outubro de 2022. https://globalmeteornetwork.org/wiki/index.php?title=Main_Page#What_is_an_RMS_GMN_station.3F (acessado 7 de janeiro de 2023).

[11] D. Vida *et al.*, “The Global Meteor Network -- Methodology and First Results”, jul. 2021, doi: 10.1093/mnras/stab2008.

[12] Global Meteor Netword, “RMS Windows Installation”, *GMN Website*, 16 de outubro de 2021. https://globalmeteornetwork.org/wiki/index.php?title=Windows_Installation (acessado 7 de janeiro de 2023).

10. APENDICES

A. Astrometria

1.A.1. Skyfit/RMS

A Global Meteor Network (GMN) é uma organização mundial de astrônomos amadores e profissionais, cujo objetivo é observar o céu noturno usando câmeras de vídeo com pouca luz e produzir trajetórias de meteoros de maneira coordenada[10] .Os dados são obtidos por estações de monitoramento do céu, as estações RMS-GMN, onde o software RMS é responsável por realizar as capturas e processar os dados de evento de meteoro. Um dos módulos do software RMS, designado Skyfit2 é responsável por realizar a calibração astrométrica da imagem captada, ou seja, corrigir a distorção atmosférica e de lentes baseada na posição de estrelas presentes na imagem.

Visto que o RMS/Skyfit2 é um software gratuito e que o problema de astrometria é um procedimento extremamente complexo, foi decido realizar a integração do software neste estudo ao invés de realizar uma implementação própria.

1.A.2. Instalação de requisitos

Antes de realizar a instalação do Skyfit2/RMS para Windows 10, alguns requisitos foram satisfeitos segundo o manual de instalação da GMN[12]

1.A.2.1. Instalação das ferramentas de compilação do MS visual Studio

- Foi usado o Visual Studio 2022 Community Edition : https://visualstudio.microsoft.com/downloads/#build-tools-for-visual-studio-2019
- Em seguida foi selecionado "Download gratuito" para instalar a edição Community. Isso fará o download e executará o instalador.
- No painel esquerdo, foi selecionado “Desenvolvimento de desktop com C++”.
- No painel direito, foi selecionado as ferramentas de compilação MSVC v143 - VS 2022 C++ x64/x86 e SDK do Windows 10 estavam selecionadas.

Em seguida foi instalado o software Anaconda para Windows

1.A.2.2. Instalação do anaconda para Windows

- Foi baixado o Anaconda pelo link: https://www.anaconda.com/products/individual
- O arquivo foi executado e foram mantidas as configurações padrão

O software Git para Windows como requisito também foi instalado

1.A.2.3. Instalação do Git para Windows

- O software Git foi baixado em:
- As opções padrão foram selecionadas.

1.A.3. Instalação do RMS/Skyfit

A instalação do Software RMS/Skyfit seguiu os seguintes três passos:

1.A.3.1. Clone do código RMS para o computador

- No menu Iniciar, abra um prompt do Anaconda Powershell e altere o diretório para o local em que deseja manter o código.
- Execute este comando:

```
git clone https://github.com/CroatianMeteorNetwork/RMS.git
```

- Isso criará uma nova pasta "RMS" contendo o código.

Em seguida, foi criado um ambiente virtual Anaconda, necessário

1.A.3.2 Criação de um ambiente virtual Anaconda

- Ainda na janela do Anaconda powershell, digite o seguinte:

```
conda create -n RMS python=3.8
```

- Isso criará um ambiente virtual Python chamado "RMS" contendo Python 3.8

O ultimo passo, foi a realização da instalação dos pacotes Python necessários.

1.A.3.3 Instalação dos pacotes Python necessários

- Ainda na janela do Anaconda powershell, digite o seguinte para ativar o ambiente virtual:

```
conda ativar RMS
```

- O prompt deve mudar para "(RMS) c:\source\" ou algo semelhante
- Altere o diretório para a pasta RMS.
- Agora instale os módulos Python necessários executando estes comandos:

```
pip install -r requisitos.txt
pip install PyQt5
pip install opencv_python
pip install rawpy
```

1.A.4. Configuração inicial

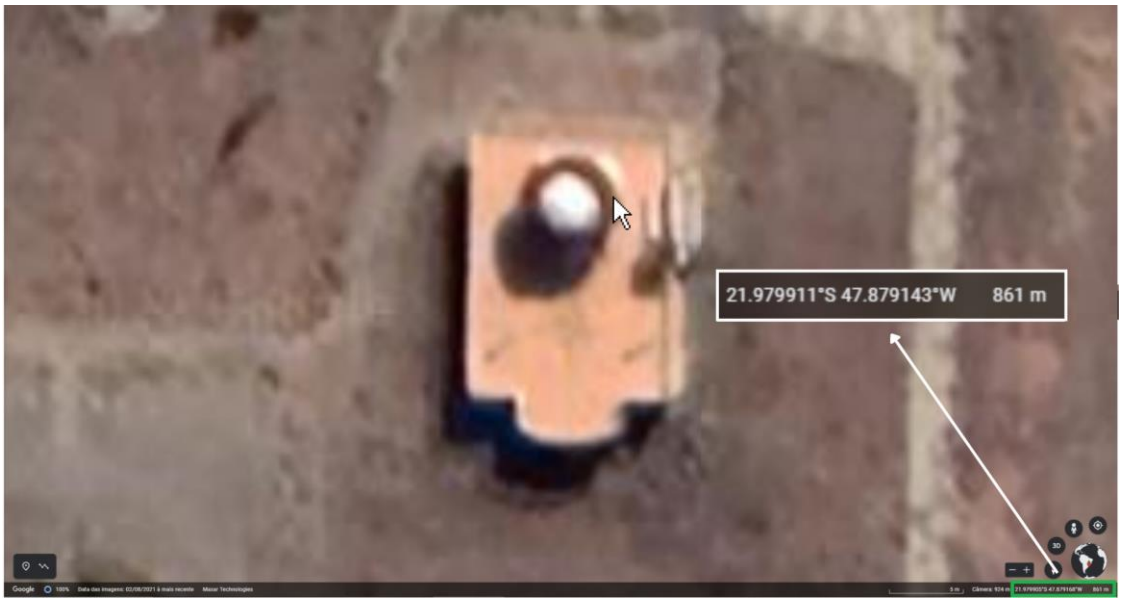
Antes de começar a operar o software Skyfit, é necessário realizar uma configuração inicial no arquivo “.config”, localizado no diretório raiz de instalação do RMS. Nele devem conter os seguintes dados.

- Latitude: latitude da câmera em graus decimais.

- Longitude: longitude da câmera em graus decimais.
- Altura(metros): Altura da estação de captura em relação ao nível do mar.

Estes dados podem ser obtidos com precisão suficiente pelo Google Earth apontando o cursor do mouse na localização de instalação da câmera.

Figura 16:Apontamento e vista de satélite no programa Google Earth



Fonte: Google Earth

Os dados foram inseridos como mostrado abaixo:

Figura 17:Arquivo .config com dados de apontamento

```
; IMPORTANT: There always must be at least one space after the argument value, before the
semicolon in front of the comment.

[System]

stationID: BR0017
latitude: -22.801310 ; WGS84 +N  (degrees )
longitude: -45.190270 ; WGS84 +E (degrees)
elevation: 569 ; mean sea level EGM96 geoidal datum, not WGS84 ellipsoidal (meters)
cams_code: 0 ; Should be set only if full CAMS compatibility is desired
```

Fonte: autoria própria

Em seguida foi necessário inserir os seguintes dados, localizados na seção “Capture” do arquivo “.config”

- fov_w: Largura do campo de visão da câmera em graus
- fov_h: Altura do campo de visão da câmera em grau

Os dados foram inseridos como na figura abaixo:

Figura 18: Dados de resolução e campo de visão no arquivo .config

```
[Capture]

device: rtsp://200.145.13.66:554/user=_password=_channel=1_stream=0.sdp ; device id
force_v4l2: false
width: 1280
height: 720
fps: 25.0 ; frames per second
report_dropped_frames: false

; Region of interest, left limit. -1 to disable
roi_left: -1
; Region of interest, right limit. -1 to disable
roi_right: -1
; Region of interest, upper limit. -1 to disable
roi_up: -1
; Eegion of interest, lower limit. -1 to disable
roi_down: -1

; Bit depth of the camera (e.g. an 8-bit camera)
bit_depth: 8
; Gamma of the camera. Usually 0.45 or 1.0
gamma: 1.0

; Format of files, either 'bin' (CAMS format), or 'fits' (new RMS format)
ff_format: fits

; Approx. horizontal Field-of-view in degrees
fov_w: 96
; Approx. vertical Field-of-view in degrees
fov_h: 54
```

Fonte: autoria própria

Em seguida a criação da máscara é necessária. Uma máscara, em processamento de imagens, nada mais é do que uma imagem utilizada para remover elementos indesejados para a análise. É comum que o campo de visão da câmera contenha elementos que não são de interesse, como casas e árvores.

Para fazer a máscara é preciso uma imagem do campo de visão da câmera no formato .bmp e, carrega-la em um software editor de imagens, e em seguida pintar de preto todos os elementos que não são o céu noturno. Como alguns objetos indesejados podem apresentar movimentos durante a captura, é boa prática pintar os mesmos com uma pequena margem de sobra para além de sua fronteira, afim de garantir que a máscara cubra estas regiões. Para completar a máscara, o céu noturno será preenchido com cor branca, obtendo assim uma imagem binária. Para utilizar a máscara é necessário colocar o arquivo no diretório raiz do RMS. Abaixo é ilustrado o processo de confecção da máscara a partir da imagem original.

Figura 19:Imagem do campo de visão da câmera.

Fonte: Tutorial de operação do skyfit

Figura 20:Máscara do campo de visão da câmera.

Fonte: Autoria própria

1.A.5. Execução do Skyfit

Para iniciar o software foi executado o comando abaixo, onde “path/to/night/directory” é a pasta que contém as capturas do céu e o comando “--config” o arquivo alterado na sessão xxx

```
python -m RMS.Astrometry.SkyFit path/to/night/directory --config
```

Na primeira execução do software, um prompt é exibido para selecionar um “platepar”, abreviação para “plate parameters” que são as transformações de coordenadas entre o céu e a imagem. É necessário selecionar “cancel” e preencher o próximo pronto com os dados de azimute, altitude e rotação

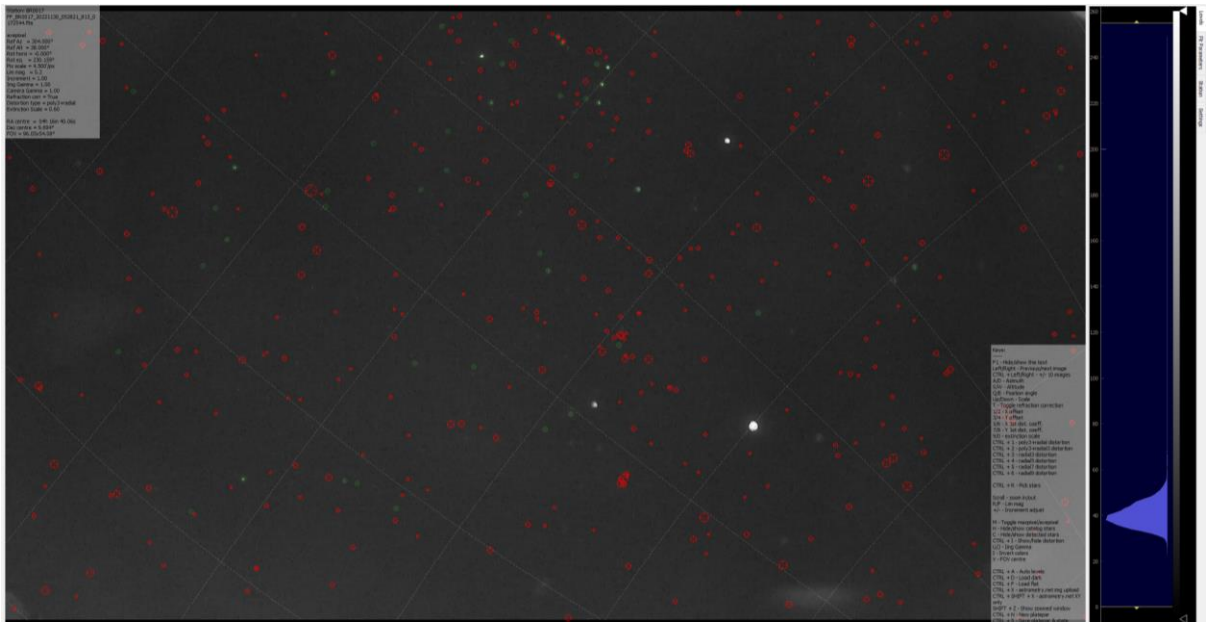
1.A.6. Operação do Skyfit

Uma vez que todas etapas de configuração foram finalizadas, foi dado início a operação do software. O objetivo desta etapa é de realizar um casamento das estrelas detectadas, com as estrelas do catalogo interno do software. Este catalogo nada mais é do que uma representação visual das estrelas que um observador enxergaria ao olhar para céu numa determinada data e localização.

Para iniciar o ajuste de estrelas é necessário encontrar uma imagem limpa do céu , sem nuvens ou outras perturbações. As setas do teclado esquerda (←) e direita (→) podem ser

usadas para visualizar as imagens captadas e escolher a que proporciona a melhor visualização com grande número de estrelas. Círculos verdes representam estrelas detectadas automaticamente pelo algoritmo. Círculos vermelhos representam a projeção de posição de estrelas do catálogo segundo os parâmetros astrométricos fornecidos. A figura abaixo mostra a tela principal de operação do Skyfit com um zoom em estrelas, para melhor visualização.

Figura 21: Modo de seleção de estrelas no Skyfit



Fonte: autoria própria

O objetivo é alterar os parâmetros do software até que as estrelas detectadas e de catálogo se sobreponham. Abaixo são mostrados os comandos utilizados para operação, presentem na secção cinza inferior-direita.

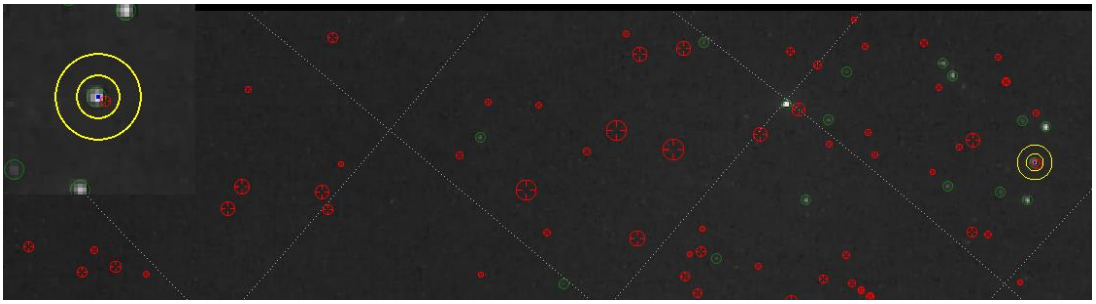
- F1 – Esconder/Mostrar informações
- Seta Esquerda/Direita – Avançar/Retroceder imagem
- A/D – Ajuste de azimute
- S/W – Ajuste de Altitude
- Q/E – Ajuste de Rotação
- Seta para cima/baixo- Ajuste de escala da imagem
- T – Acionar correção de refração
- CTRL+R – Selecionar estrelas
- Roda do mouse – Aumentar/Diminuir zoom

- R/F – Seleciona magnitude limite de estrelas do catalogo
- M – Aciona modo de máxima intensidade de pixel ou pixel médio.
- H – Esconder/Mostra catalogo de estrelas
- C – Esconder/Mostra estrelas detectadas
- CTRL + I – Mostra/Esconder distorção
- U/J – Controle de gama da imagem
- I – Inverter cores
- V – Aciona janela para inserir o centro de visão da câmera
- CTRL+X – Realiza o upload da imagem para o astrometry.net
- CTRL+S – Salva o arquivo Platepar e estado atual.

1.A.7. Ajustes astrométricos

Assim que as estrelas do catálogo estiverem próximas o suficiente das estrelas detectadas, é necessário realizar um ajuste manual mais preciso. Este modo é ativado por "CTRL+R", onde dois círculos concêntricos aparecerão no lugar no ponteiro do mouse. O objetivo agora é selecionar as estrelas do catalogo e as estrelas detectadas de modo que a estrela detectada esteja dentro do círculo interior e a do catalogo esteja no máximo, na região entre os círculos assim como mostrado na figura abaixo.

Figura 22: Zoom em janela de operação do Skyfit



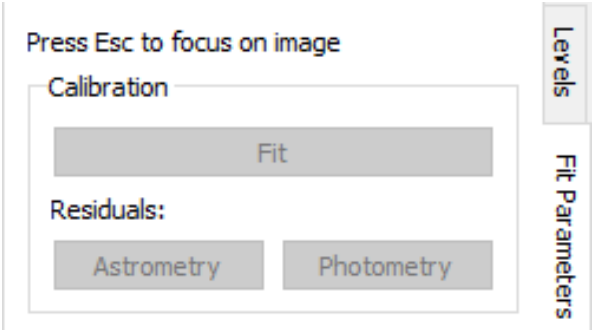
Fonte: autoria própria

Uma vez selecionada basta apertar a tecla "Enter"/"Espaço", e repetir o processo. Um bom ajuste é o que cobre todo o campo de visão da câmera, e com ao menos 15 estrelas. Para executar o ajuste é necessário pressionar "CTRL+Z, isso fará com que as estrelas do catalogo mudem de posição para se aproximar o máximo possível das estralas detectadas.

Verificação do ajuste

Após a calibração manual, afim de verificar a precisão do ajuste, é necessário selecionar a aba no lado direito superior “Fit Parameters”. Ao clicar em “Astrometry” como na figura abaixo, serão gerados dados sobre a astrometria da sessão atual.

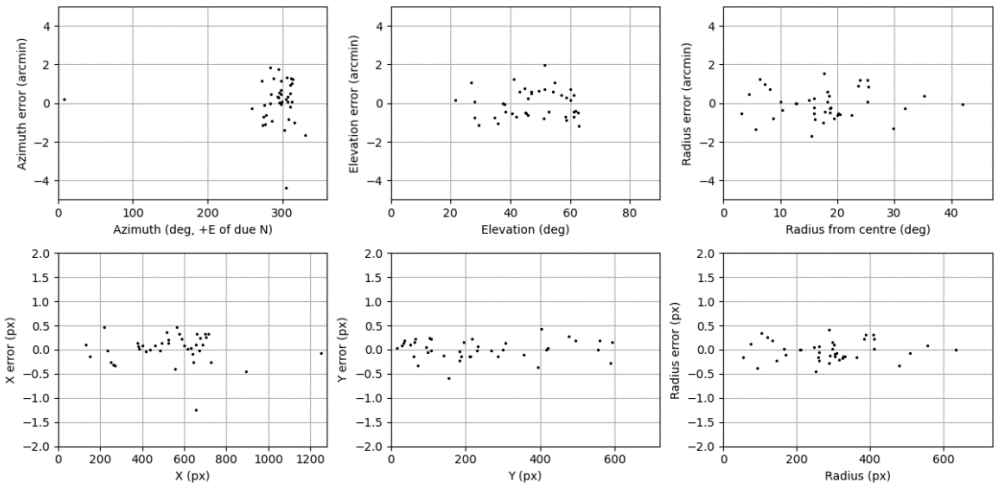
Figura 23:Aba Fit Parameters



Fonte: autoria própria

Os dados gerados servem como um guia para decidir se foi feito uma boa calibração astrométrica. Nos seis gráficos gerados, como na figura abaixo, o objetivo é produzir o menor erro possível, que pode ser observado pelo eixo y dos gráficos. O processo de calibração deve ser repetido até que se obtenha o menor erro possível.

Figura 24:Graficos de erro da calibração astrometrica



Fonte: tutorial de operação do Skyfit

Geração do platepar

Neste último passo, assim que a calibração astrometria desejada for obtida, basta acionar o comando “CTRL+S” para salvar o arquivo “platepar”, abreviação para “Plate Parameters”, que contém os dados astrométricos que serão utilizados para calcular a posição do meteoro no céu.

B. Conversão de coordenadas

A saída do programa Skyfit2 fornece parâmetros para uma equação polinomial de 11 termos que representa a distorção de imagem do apontamento de câmera. Uma vez que este procedimento foi realizado para um apontamento de câmera em determinada ascensão reta e declinação, não é necessário realizar o procedimento novamente se a câmera não for movida de lugar. Abaixo está um exemplo de uma saída do programa Skyfit2, o platepar:


```
{
    "F_scale": 16.562923141418587,
    "Ho": 151.18766026757658,
    "JD": 2459913.7280895025,
    "RA_d": 67.85114071207016,
    "UT_corr": 0,
    "X_res": 1280,
    "Y_res": 720,
    "alt_centre": 41.49675216957533,
    "asymmetry_corr": false,
    "auto_check_fit_refined": false,
    "auto_recalibrated": false,
    "az_centre": 305.942200641628,
    "dec_d": 8.504932976159067,
    "distortion_type": "radial9-odd",
    "distortion_type_list": [
        "poly3+radial",
        "poly3+radial3",
        "poly3+radial5",
        "radial3-all",
        "radial4-all",
        "radial5-all",
        "radial3-odd",
        "radial5-odd",
        "radial7-odd",
        "radial9-odd"
    ],
    "distortion_type_poly_length": [
        12,
        13,
        14,
        7,
        8,
        9,
        6,
        7,
        8,
        9
    ],
    "elev": 569.0,
    "equal_aspect": true,
    "extinction_scale": 0.6,
    "force_distortion_centre": false,
    "fov_h": 83.34899064259895,
```

```
    "fov_v": 44.5161999978445,
    "gamma": 1.0,
    "lat": -22.80131,
    "lon": -45.19027,
    "mag_0": -2.5,
    "mag_lev": 9.78125,
    "mag_lev_stddev": 1.58204675151109,
    "measurement_apparent_to_true_refraction": false,
    "poly_length": 6,
    "pos_angle_ref": 140.65067473327196,
    "refraction": true,
    "rotation_from_horiz": 5.5168611267559635,
    "star_list": [
        [
            2459913.7280895025,
            700.36,
            62.66,
            248.42307692307696,
            84.6865271917247,
            -2.60007908591377,
            3.68681
        ],
        [
            2459913.7280895025,
            713.66,
            66.01,
            1154.384615384612,
            85.1898637404987,
            -1.94323615717053,
            3.89003
        ],
        [
            2459913.7280895025,
            271.68,
            184.53,
            795.9999999999982,
            59.5076333836323,
            -13.5089972976979,
            2.15061
        ],
        [
            2459913.7280895025,
            248.86,
            232.81,
            185.15384615384605,
            56.5358107955592,
            -12.1013329014498,
            3.5379
        ],
        [
            2459913.7280895025,
            262.48,
            269.85,
            313.0384615384609,
            55.8116765700273,
            -9.76018726493585,
            3.13472
        ],
```

```
        [
            2459913.7280895025,
            234.04,
            300.79,
            184.26923076923129,
            53.228430601097,
            -9.45817151519136,
            3.36914
        ],
        [
            2459913.7280895025,
            598.74,
            34.57,
            122.46153846153825,
            80.9867200292966,
            -7.80825086531684,
            3.78899
        ],
        [
            2459913.7280895025,
            588.47,
            62.11,
            276.53846153846087,
            79.4015491096642,
            -6.84444911487671,
            3.49769
        ],
        [
            2459913.7280895025,
            387.61,
            72.02,
            330.1538461538458,
            69.5447447788516,
            -14.3046880319333,
            3.47358
        ],
        [
            2459913.7280895025,
            381.43,
            211.96,
            125.11538461538481,
            63.8082666493037,
            -7.66760265960419,
            4.10706
        ],
        [
            2459913.7280895025,
            659.99,
            36.44,
            607.3461538461528,
            83.8186191685019,
            -5.38969617097692,
            4.93193
        ],
        [
            2459913.7280895025,
            654.38,
            29.61,
```

```
            340.2307692307689,
            83.761181151456,
            -6.00202284150761,
            4.6875
        ],
        [
            2459913.7280895025,
            657.12,
            100.0,
            365.0384615384618,
            81.1197266348244,
            -2.39704114179649,
            4.90372
        ],
        [
            2459913.7280895025,
            633.19,
            285.79,
            285.76923076923117,
            72.8015164881292,
            5.60510777911225,
            3.58933
        ],
        [
            2459913.7280895025,
            643.2,
            306.48,
            479.6923076923084,
            72.4620587388972,
            6.96132903947652,
            2.96543
        ],
        [
            2459913.7280895025,
            401.67,
            555.47,
            249.11538461538373,
            51.2030371503799,
            9.02854893092308,
            3.25672
        ],
        [
            2459913.7280895025,
            416.51,
            557.76,
            113.38461538461496,
            51.7925589840764,
            9.73255529599468,
            3.55472
        ],
        [
            2459913.7280895025,
            462.98,
            587.74,
            234.80769230769215,
            52.7183221375978,
            12.9366714714252,
            3.74526
```

```
        ],
        [
            2459913.7280895025,
            726.66,
            394.58,
            273.61538461538527,
            73.133182838715,
            14.2504058083127,
            3.45122
        ],
        [
            2459913.7280895025,
            639.35,
            494.21,
            270.07692307692344,
            64.9488690738787,
            15.627541217671,
            3.28999
        ],
        [
            2459913.7280895025,
            133.51,
            402.53,
            161.84615384615338,
            44.1072124395409,
            -8.8990871712927,
            3.46408
        ],
        [
            2459913.7280895025,
            219.94,
            591.86,
            53.88461538461519,
            40.8245193063078,
            3.23518879619282,
            3.42881
        ],
        [
            2459913.7280895025,
            522.67,
            187.86,
            181.7307692307695,
            71.3756958262087,
            -3.25471484482801,
            3.93052
        ],
        [
            2459913.7280895025,
            523.79,
            141.44,
            119.92307692307719,
            73.2235416917389,
            -5.4525882778212,
            4.23629
        ],
        [
            2459913.7280895025,
            435.75,
```

```
            155.89,
            168.26923076922992,
            68.5483523498673,
            -8.23133364716624,
            4.21169
        ],
        [
            2459913.7280895025,
            492.25,
            210.51,
            219.38461538461615,
            69.0797619465794,
            -3.35248037366393,
            3.82801
        ],
        [
            2459913.7280895025,
            151.7,
            94.31,
            71.19230769230785,
            56.7112809579941,
            -23.2520034279893,
            4.04396
        ],
        [
            2459913.7280895025,
            675.61,
            194.28,
            140.8461538461537,
            78.3228563889199,
            2.86125487459529,
            4.09506
        ],
        [
            2459913.7280895025,
            554.86,
            420.29,
            84.8076923076924,
            63.8836612934309,
            8.89226065113701,
            4.1826
        ],
        [
            2459913.7280895025,
            1250.19,
            184.32,
            114.8076923076924,
            111.918862350315,
            22.1414416468522,
            5.98887
        ],
        [
            2459913.7280895025,
            891.83,
            356.7,
            408.88461538461627,
            83.0531364098511,
            18.5942170359832,
```

```
            3.02804
        ],
        [
            2459913.7280895025,
            687.56,
            103.25,
            153.11538461538433,
            82.4332614702992,
            -1.0923480839662,
            3.98752
        ],
        [
            2459913.7280895025,
            564.23,
            52.99,
            1755.2307692307663,
            78.6334025005198,
            -8.20406671720238,
            6.58312
        ],
        [
            2459913.7280895025,
            575.42,
            107.89,
            450.38461538461473,
            76.9620121776637,
            -5.08679980559722,
            8.32508
        ],
        [
            2459913.7280895025,
            484.9,
            416.16,
            201.69230769230788,
            60.7891015928781,
            5.98928371900892,
            3.81115
        ],
        [
            2459913.7280895025,
            669.74,
            476.06,
            488.92307692307753,
            67.1442155628193,
            15.9620332625605,
            3.49308
        ],
        [
            2459913.7280895025,
            616.45,
            216.75,
            218.538461538461,
            74.6370901476594,
            1.71398571743095,
            3.9469
        ]
    ],
    "station_code": "BR0017",
```

```
    "version": 2,
    "vignetting_coeff": 0.001,
    "vignetting_fixed": true,
    "x_poly": [
        -0.06197712046701795,
        0.038909484819859834,
        0.07126857952994872,
        0.005832649622015277,
        6.1476818186878114e-06,
        -2.3864372956202397e-05
    ],
    "x_poly_fwd": [
        -0.06197712046701795,
        0.038909484819859834,
        0.07126857952994872,
        0.005832649622015277,
        6.1476818186878114e-06,
        -2.3864372956202397e-05
    ],
    "x_poly_rev": [
        -0.06198793710200802,
        0.038916292446974206,
        0.0712808457837272,
        -0.009338086255248372,
        0.0009025024740258129,
        6.807226883767647e-06
    ],
    "y_poly": [
        0.5,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0
    ],
    "y_poly_fwd": [
        0.5,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0
    ],
    "y_poly_rev": [
        0.5,
        0.0,
        0.0,
        0.0,
        0.0,
        0.0
    ]
}
```


Os valores listados na subsecção "x_poly" e "y_poly" do arquivo platepar representam os valores de a0 até a11 do polinômio de terceira ordem de distorção de imagem. Com estes valores

obtidos nenhum outro software adicional é requerido para realizar a conversão de coordenadas de imagem em coordenadas reais. O procedimento restante é apenas a implementação das equações presentes na secção de apêndices do artigo “The Global Meteor Network -- Methodology and First Results”[11].

Abaixo é apresentando um algoritmo simplificado para implementação de conversão de coordenadas baseada no artigo.

i. Normalização das coordenadas de imagem (x,y) para o centro do quadro.
ii. Aplicação do polinômio de correção de distorção com termos a0-a11, b0-b11 para cada ponto.

iii. Normalização das coordenadas de imagem (x,y) para o centro do quadro.
iv. Cálculo das coordenadas não distorcidas (x', y') aplicando a correção de distorção (usando os coeficientes do polinômio de terceira ordem com 11 termos)
v. Representação das coordenadas desdistorcidas em uma forma polar normalizada pela escala de vídeo (valor de escala F em px/°, presente no platepar).
vi. Calculo da ascensão reta instantânea do centro de projeção, somando a diferença entre os ângulos horários de Greenwich no tempo de referência JD0 (data de realização do platepar) e no tempo de interesse JD(data de avistamento do meteoro) ao valor de referência de ascensão reta alpha_C0 (presente no platepar)
vii. Aplicação de correção de refração atmosférica ao centro de projeção para obter as coordenadas equatoriais aparentes.

Após realizar este procedimento para todos os pontos de detecção de centroide do meteoro (obtidos por este estudo por exemplo) deve-se obter a trajetória completa do meteoro em coordenadas reais pelo céu, possibilitando assim que estes dados sejam entradas de outros programas que realizem analise de trajetória do meteoro pelo sistema solar e estipulação de possíveis locais de queda.