**Arkenstone**

**Last updated: 02272025**

This manual for the Python-based package written for the analysis of in-house high-speed images studying shock-compressed samples has three sections:

* Section-I is the general outline of the package, its capabilities and details on how to set it up and use it.
* Section-II provides more details on a particular processing option: hyperspectral emission analysis.
* Section-III is an incomplete list of common errors and suggested troubleshooting.

# SECTION-I: User Manual for Arkenstone: Particle Emission Analysis Software

## 1. Introduction

Arkenstone is a software package designed to process emissions of particles that exhibit hotspots and react after being shock compressed. It provides analysis using optical and hyperspectral emissions and includes tools for segmentation, classification, and visualization of particle ignition and reaction growth.

## 2. System Requirements & Installation

### 2.1 Installing MiniConda

Arkenstone requires a Python environment. The recommended approach is to install MiniConda:  
1. Download MiniConda for your OS: https://docs.conda.io/en/latest/miniconda.html  
2. Install and open terminal (Anaconda Prompt on Windows)  
3. Create and activate a new environment:

conda create -n arkenstone\_env python=3.8  
conda activate arkenstone\_env

### 2.2 Installing Required Dependencies

Run the following commands to install required libraries by copy pasting following in anaconda prompt of the activate environment you want to run Arkenstone on:

pip install numpy pandas matplotlib scipy opencv-python torch torchvision scikit-image tkinter  
pip install 'git+https://github.com/facebookresearch/segment-anything.git'

### 2.3 Installing an IDE (Optional)

You can use Spyder, VS Code, or any IDE of your choice.  
To install Spyder via Conda, type following in anaconda prompt:

conda install spyder

To launch Spyder type following in anaconda prompt:

spyder

## 3. Capabilities of Arkenstone:

Arkenstone includes two primary approaches for assessing shock-initiated reactions within particles based on emissions. Analysis based on type of emission.

**3.1.** **Optical Emission Analysis (Note: Some aspects need more work):**

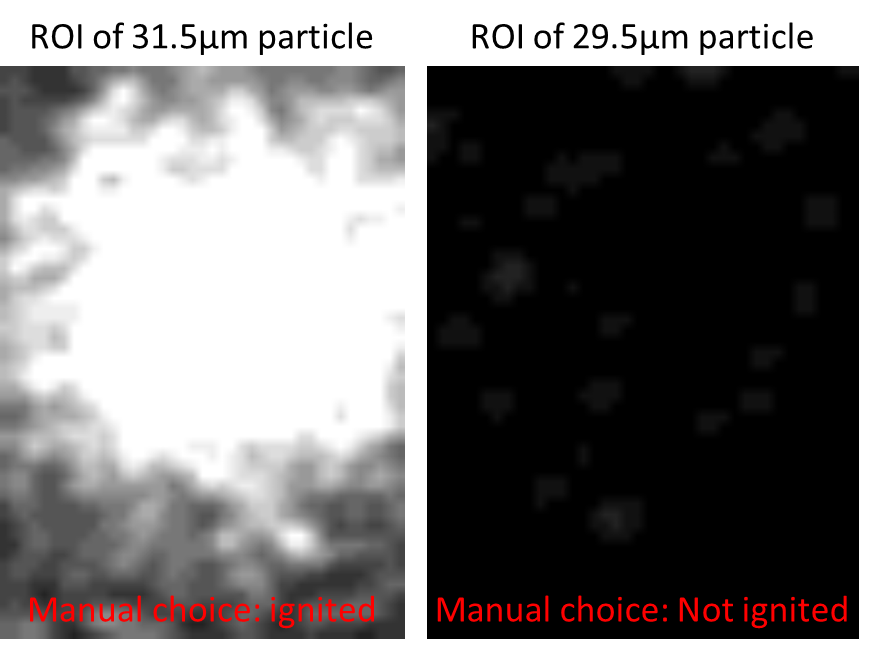
Uses raw emissions from SIMX4/8 (or other cameras) to identify reactions within particles by using stationary images as stencils to draw out spaces to look out for emissions. Currently (02272025) uses manual decision making to decide if the pixel values within can be thought of as emissions associated to hotspots/reactions. Future work will make it automated and draw the same type of inferences as hyperspectral emission analysis such as growth in size of emission within each ROI, change in intensity of emission within each ROI.

* **Approach:**
  + Uses Random Forest Classifier (RFC) (currently -02272025 not ready and needs more training so do not use will throw errors) or Segment Anything Model (SAM) (primary tool as of 02272025) to identify individual spatial regions or region-of-interests (R.O.I) associated with individual particles from stationary images.

A white circle with blue squares

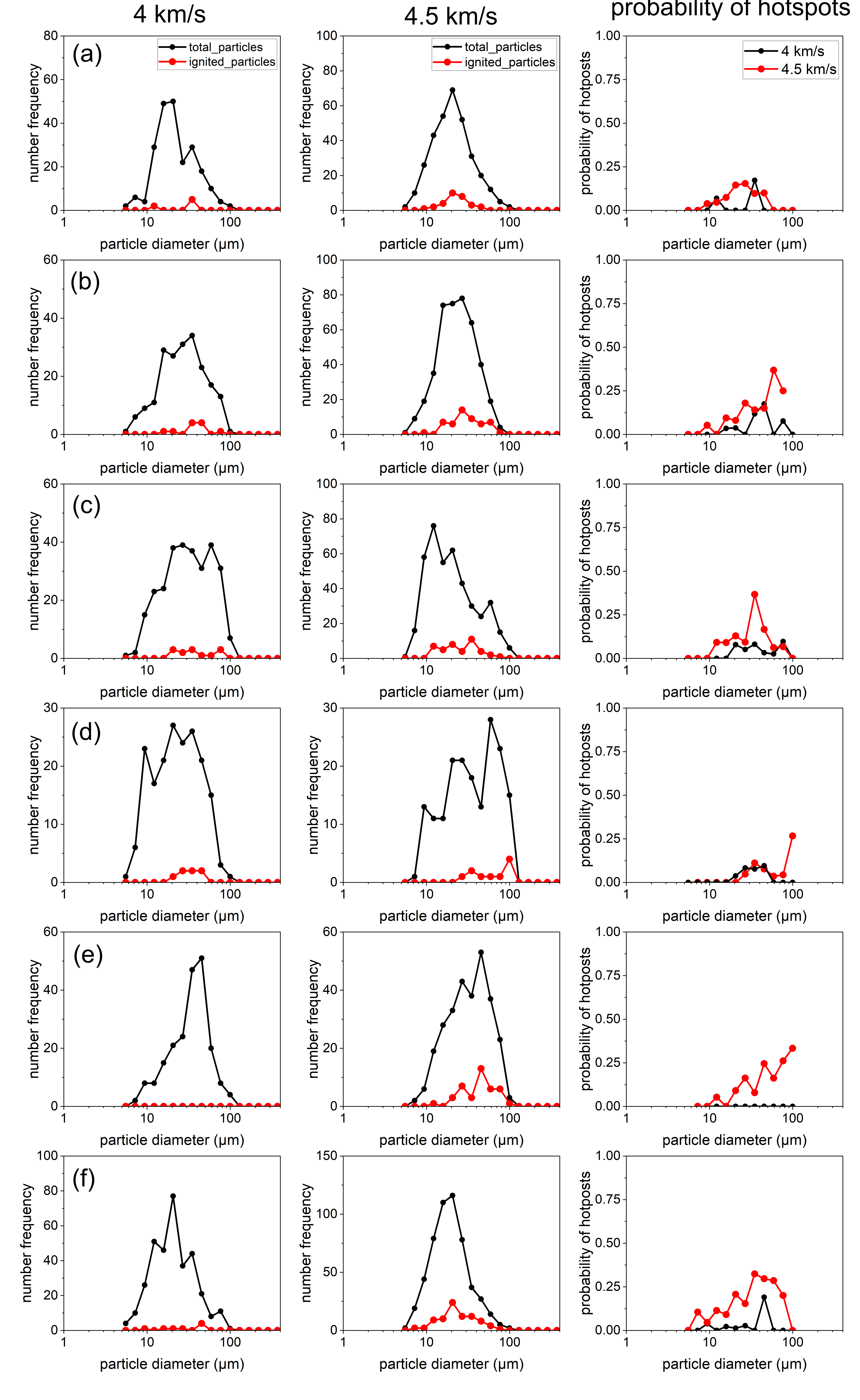
AI-generated content may be incorrect.

* + The ROIs so identified are mapped onto emission frames chosen by the user to consider to manually identify if emissions within each ROI is a hotspot to define if the particle is exhibiting shock-initiated reactivity.



* **Outcome:** Then we bin all initially identified particles and particles deems reacting to form two types of distributions. The ratio of frequencies of the two distributions at each size bin gives us the probability of shock initiation for that size.

Example image shows the data obtained for Al-CuO (Equivalence ratio 4-60 min milling-3 BPR-250 RPM-24mL hexane) sample, when shocked at 4 and 4.5 km/s.

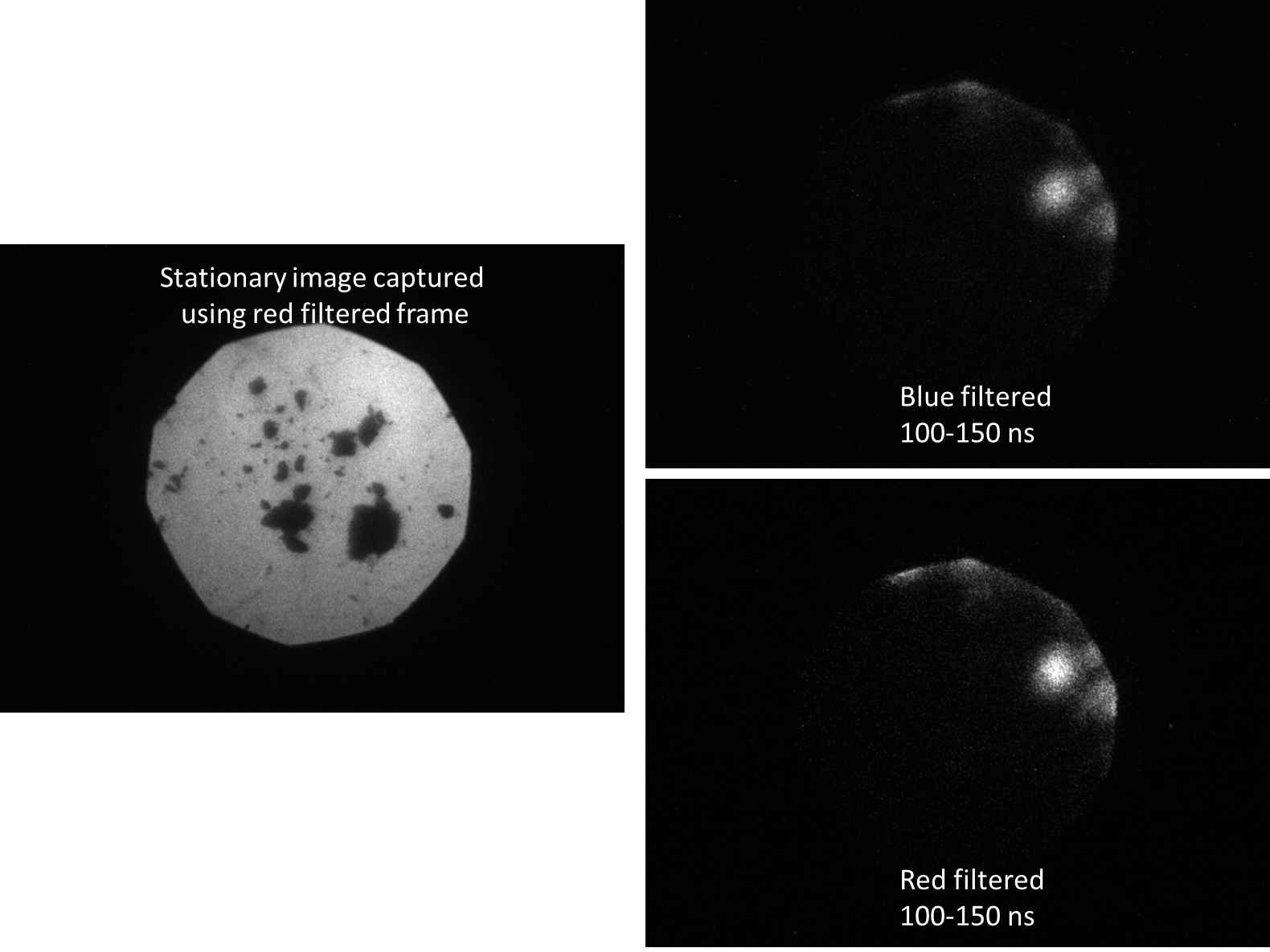


**3.2.** **Hyperspectral Emission Analysis:**

Uses emissions from red and blue filtered frames looking at the same event, at the same duration to get temperatures within individual shock-compressed particles by using stationary pre-experiment images as stencils to draw out spaces associated with particles to look out for emissions. The processing is automated as temperatures values are used to define the threshold where particles can be considered to have ‘reactions’ within.

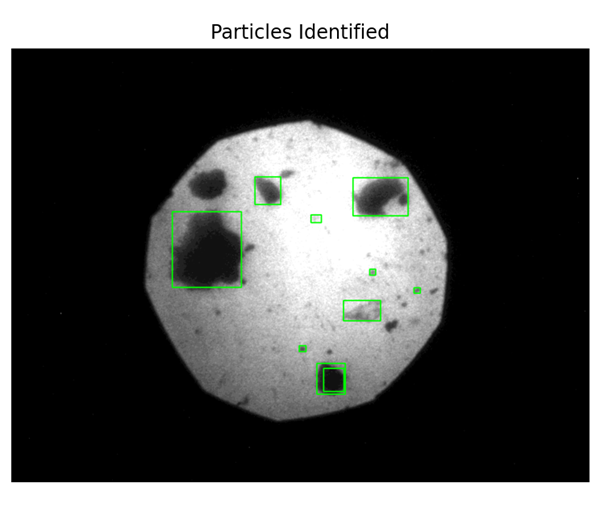
Note: Use long and short band filters on the SIMX cameras and then ensure for each instance of observation, both the long and short band filters have the same delays and exposures.

Example image shows one stationary image and a pair of red-filtered and blue-filtered emissions during 100-150ns from multiple particles belonging to Al-CuO (Equivalence ratio 4-60 min milling-3 BPR-250 RPM-24mL hexane) sample, that were shocked at 4.5 km/s.

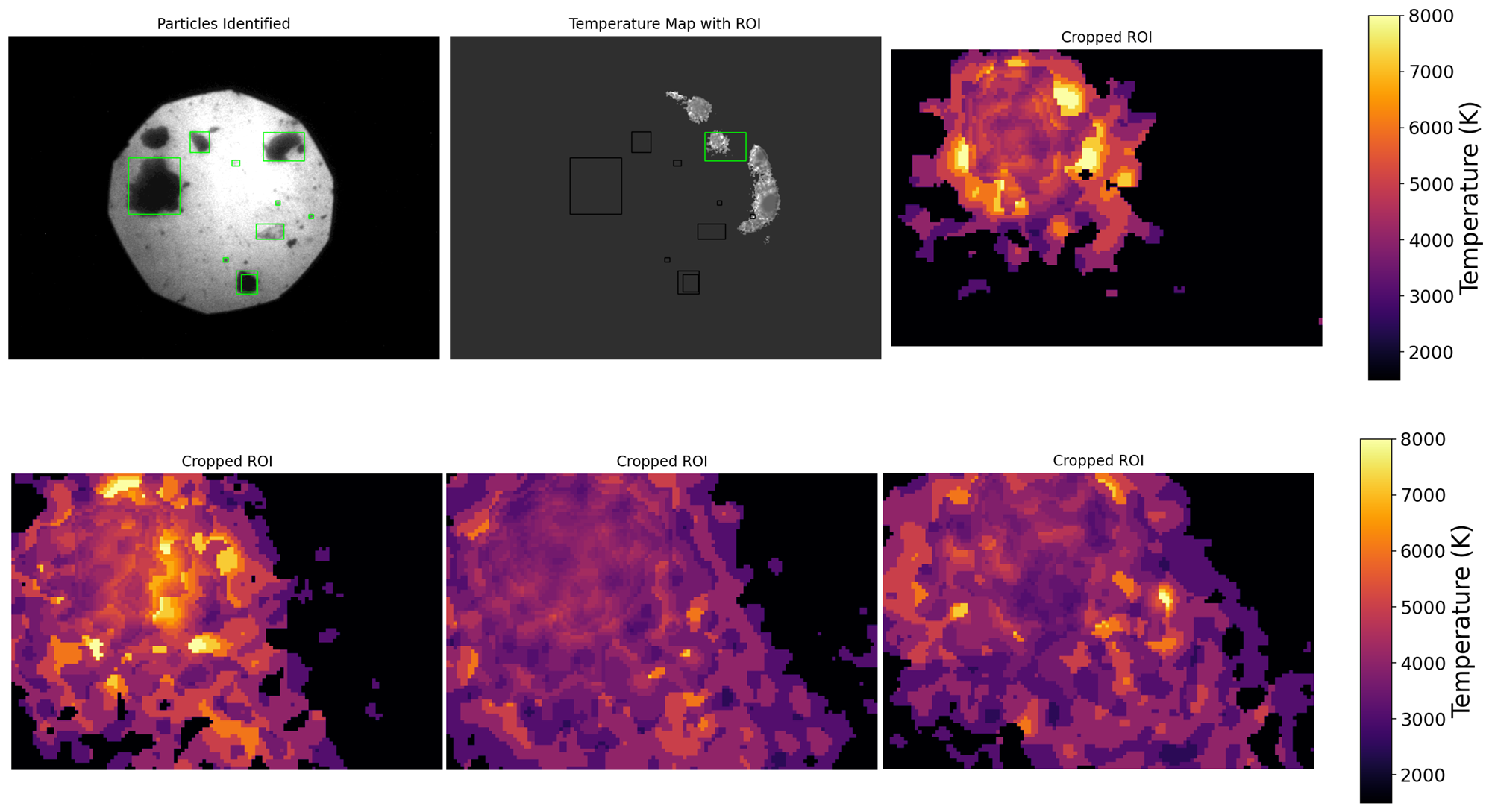


* **Approach:**

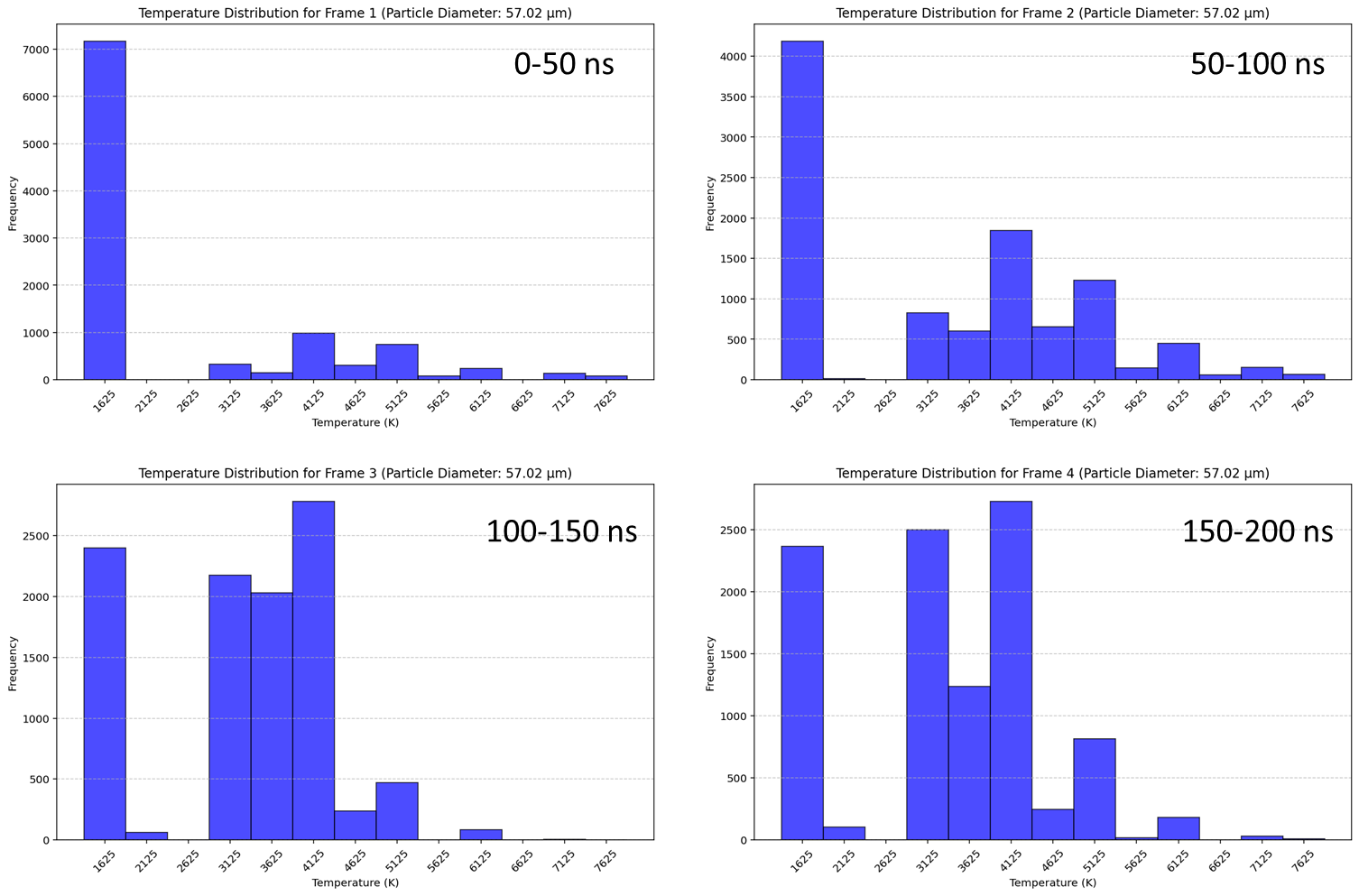
### Relies solely on Segment Anything Model to isolate particles and generate masks for emission frames chosen by user to process.



### The ROIs are identified and temperatures within are stored as histograms from 1500-8000K with 250 K bin width.

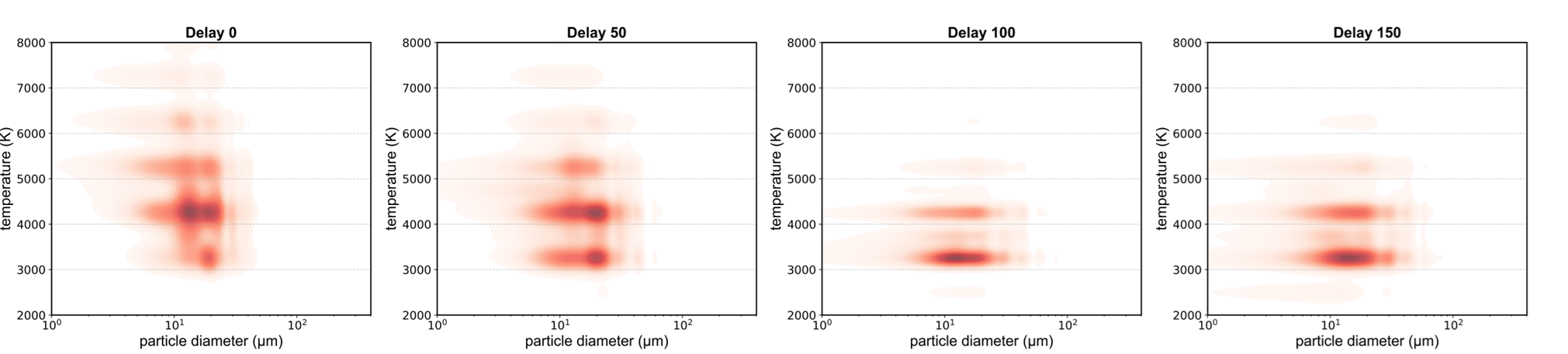


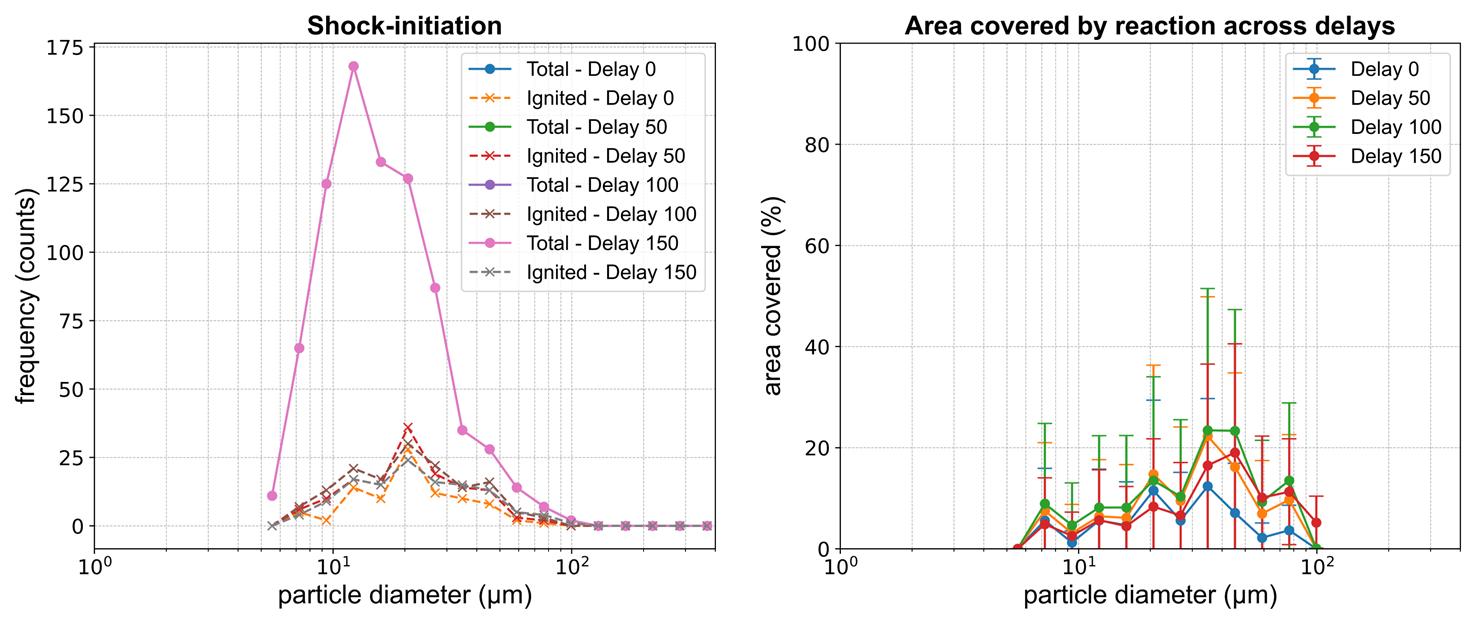
* + The threshold for meaningful reactions within ROI is at least 5 % of the pixels being greater than 2200K (the minimum identifiable temperatures in SIMX CCD sensors confirmed using 32-channel PMTs. Refer to HMX single particle work by Belinda Pacheco-Johnson). The code recognizes ‘background’ and attributes a value of 1500K.  
    The image below provides the histograms of temperatures within an ROI showing significantly non-reacting portions being cooler than 2200K but more than 5% of values being above 2200K signifying clear reaction (over background noise)



### Outcome:

• Hotspot Analysis: Probability of ignition, temperature mapping, and size tracking.  
• Growth Analysis: Tracking temperature and size as a function of time.  
  
Example plots showing analysis of dataset: Al-CuO (Equivalence ratio 4-60 min milling-5 BPR-350 RPM-12mL hexane) several runs of multiple particles suspended in 250 µm deep PDMS wells shock compressed at 4.5 km/s. The SIMX8 frames observe 0-50 ns, 50-100 ns, 100-150 ns and 150-200 ns, where 0 ns is the instance of flyer impacting the wells.





## 4. Using Arkenstone

## 4.1. Required Input File Structure

For Arkenstone to process data correctly, input images and files should be structured as follows:

### 4.1.1 Main Data Folder

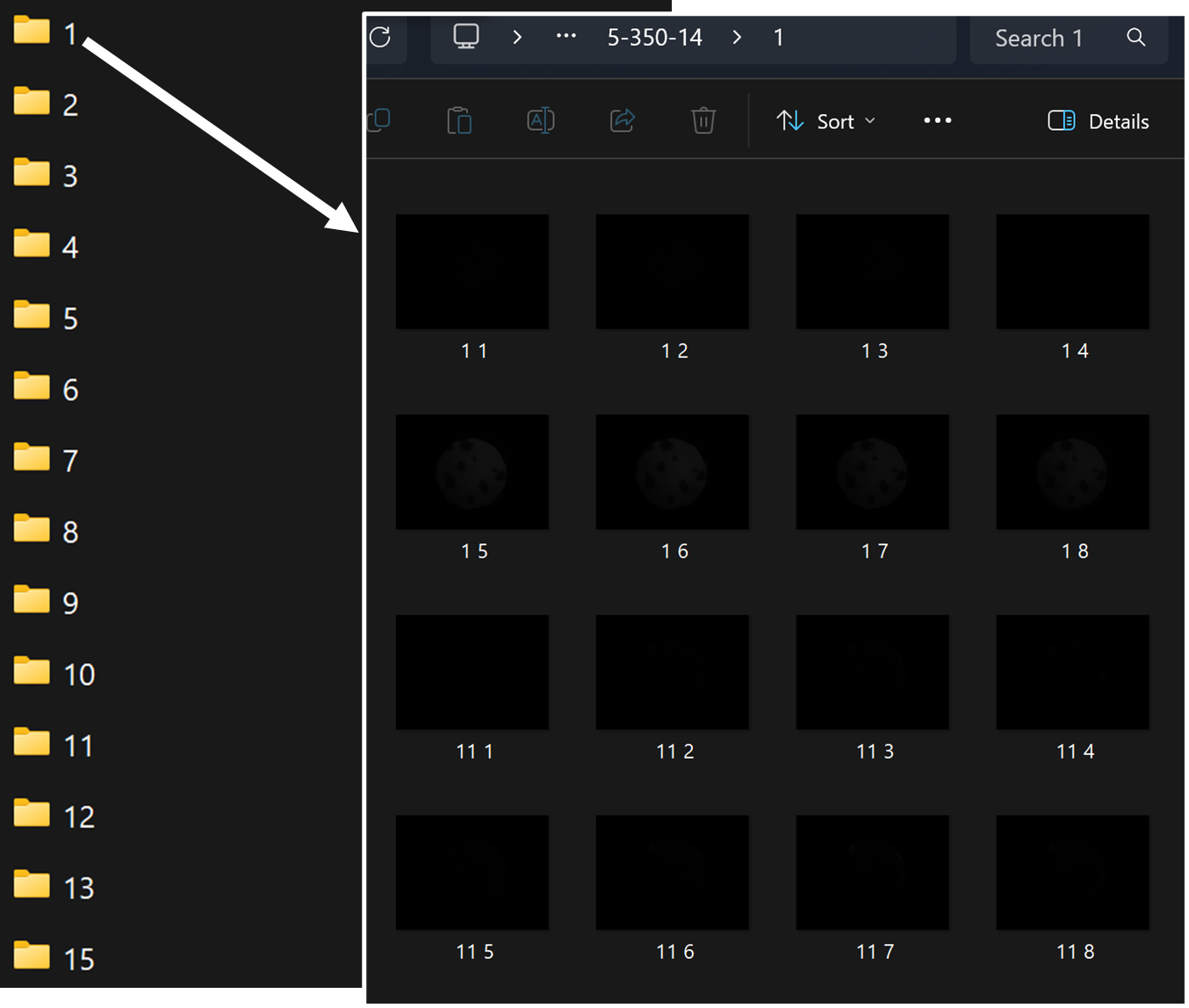
The main folder (will be used as sample name when saving data) should contain subfolders for different image sets.

### 4.1.2 Subfolder Structure

Each dataset should be stored in a separate subfolder, typically named as run number or timestamp. Sequence doesn’t matter.

### 4.1.3 Image Naming Convention

Images should be stored in \*\*.tiff\*\* format and named according to the frame sequence. For example see the image showing the subfolders denoting runs (1,2,…) and each folder containing 16 images(as the example uses SIMX8), 8 stationary ( 1 1, 1 2, …) and 8 emission images (11 1, 11 2,…) saved in sequence. They must be saved in sequence! Naming scheme is non specific.



### 4.1.4 Calibration Data (Optional if using Hyperspectral imaging)

If calibration images are available, they should be placed in a subfolder named `Calibration`. Theres no need to save them in separate run folders.

### 4.1.5 Dark Field Images (Optional if using Hyperspectral imaging)

Dark field images, if used, should be placed in a folder named `DarkField`. This is optional even if you’re running hyperspectral processing.

### 4.2. Launching Arkenstone

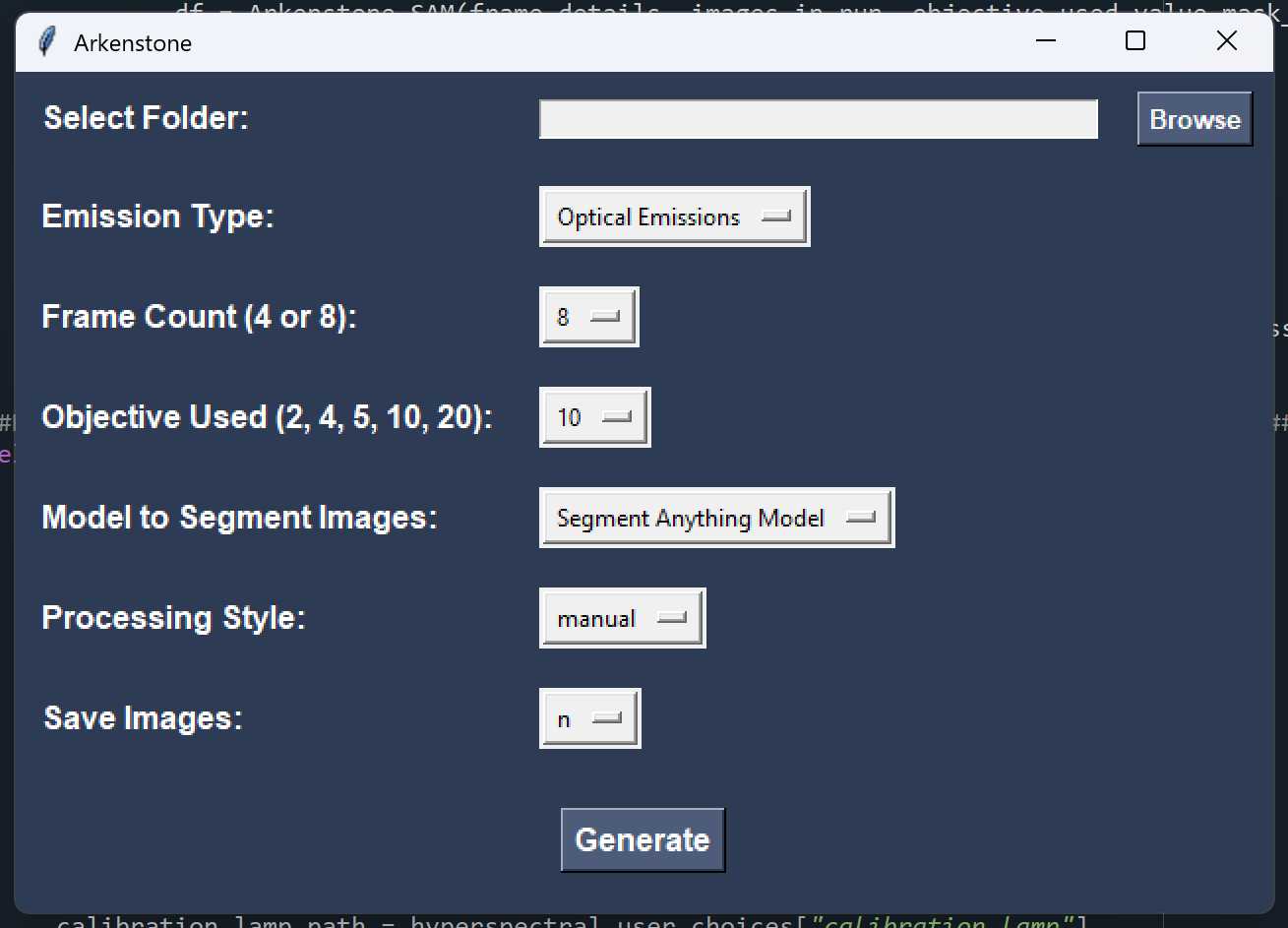
Run the main wrapper script using this (or use IDE user interface to run)

python Arkenstone\_wrapper.py

### 4.3 Program Configuration:

### 4.3.1 Primary GUI

1. Select the folder containing images. **see 4.1 for file structure details**  
2. Choose the emission type (Hyperspectral or Optical): **can handle both**   
3. Select frame count (typically 4 or 8): **can handle both datasets from SIMX4 and SIMX8**4. Specify microscope objective used (2x, 4x, 5x, 10x, 20x): **currently (02272025) only** **10x has accurate pixel to micron scale.**  
5. Choose segmentation model (Random Forest or Segment Anything Model): **currently (02272025) only** **SAM is better tuned.**   
6. Choose processing style (Manual or Automated): **currently only manual for optical and automated for hyperspectral**  
7. Choose whether to save cropped emission images associated with specific particles.

****

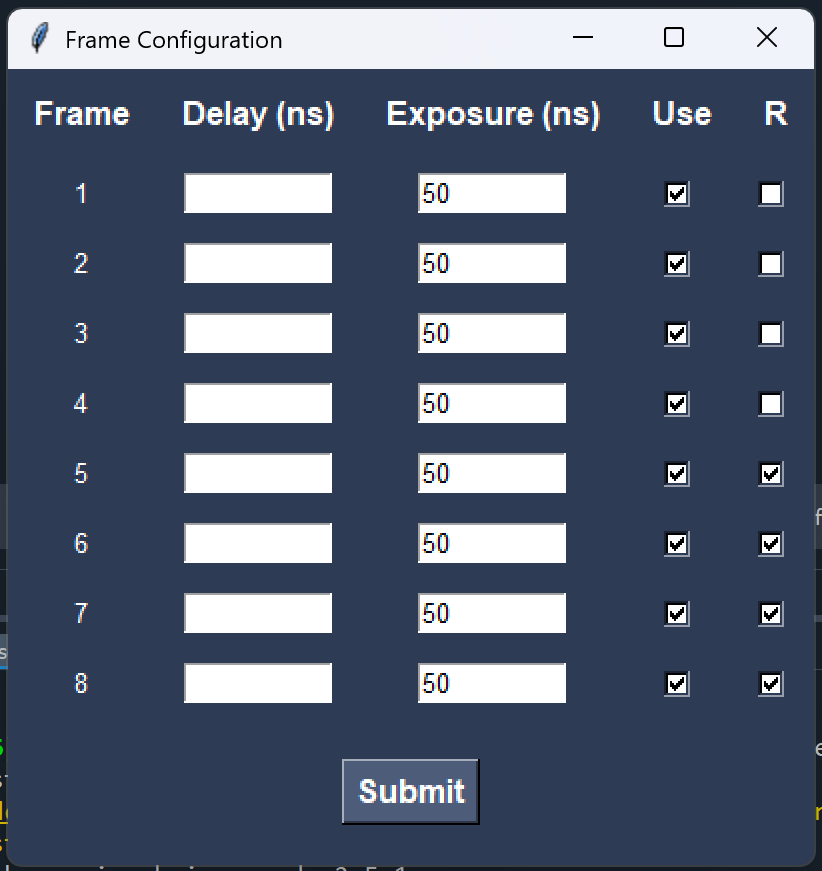
**Click 'Generate' to proceed!**

### 4.3.2 Frame detail GUI (mandatory GUI for both emission type analysis)

1. Enter the delays and exposures for the frames used (4 or 8): **No default values and mandatory entries for frames chosen to be ‘used’**

2. The default settings ‘use’ all 8(4) frames. And all considered frames need entries.

3. For hyperspectral analysis, the default settings consider the last 4(2) as red filtered (labelled ‘R’)



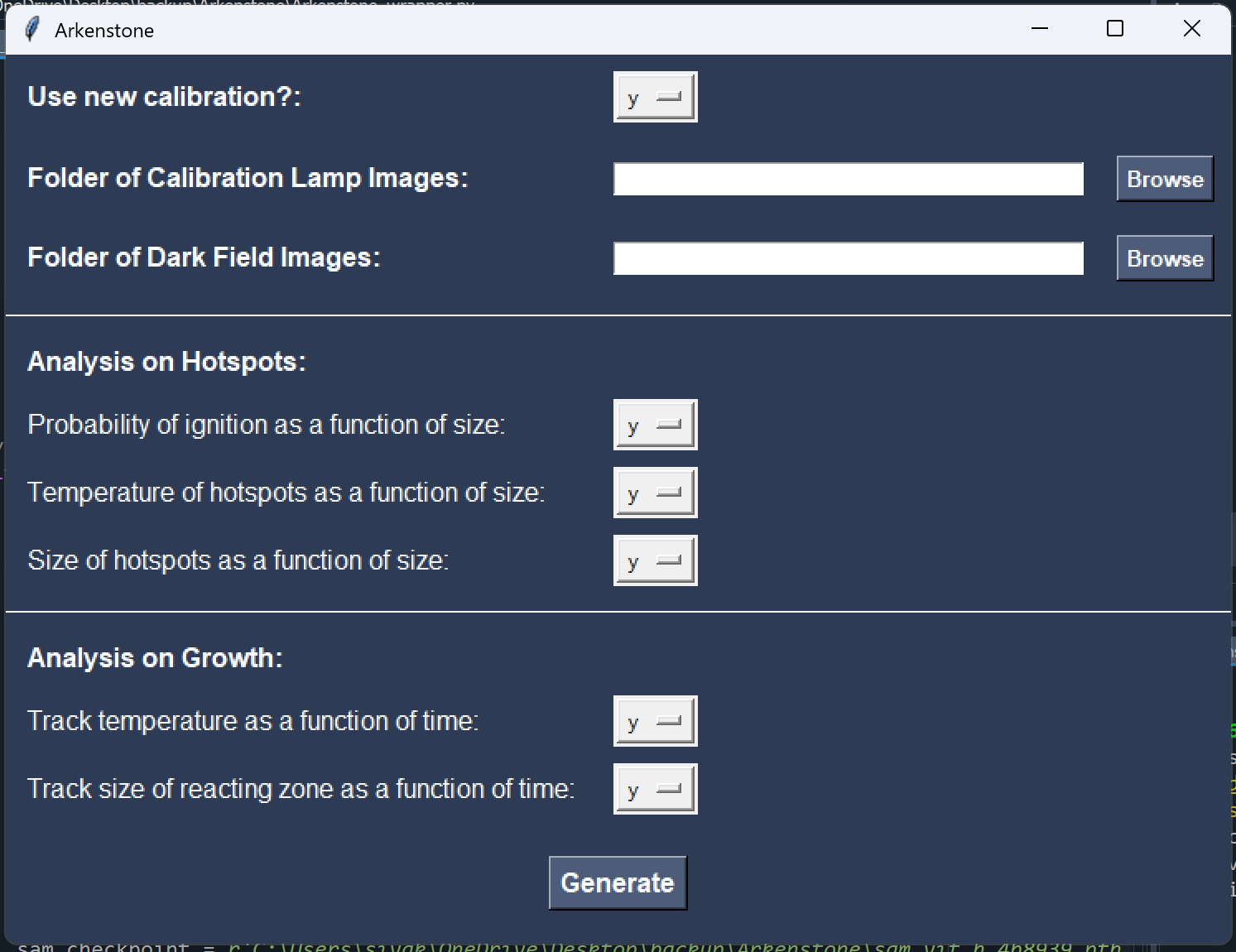
**Click 'Submit' to proceed!**

### 4.3.2 Hyperspectral GUI (optional for hyperspectral emission type analysis)

1. Option to provide fresh images for calibration and background correction: **y/n**

2. Based on choice option to browse and select folders with images.

3. Options to choose specific analysis to perform on temperature maps.



**Click 'Generate' to proceed!**

## 5. Additional Customization: Further customization needed for each file is listed below: • `Arkenstone\_wrapper.py` (Main execution) Optional. None needed. • `Arkenstone\_SAM.py` (Image Analysis for manual Optical Emission analysis) You will need to change address of where the vit\_h flavor pre-trained SAM model can be found in script to run segmentation! • `Arkenstone\_Hyperspectral.py` (Image Analysis for Hyperspectral Emission analysis) You will need to change address of where the vit\_h flavor pre-trained SAM model can be found in script to run segmentation! • `Arkenstone\_analysis.py` (Data Analysis) Optional. None needed.

# SECTION-II: Hyperspectral Imaging: Theory & Implementation in Arkenstone

## 1. Introduction

Hyperspectral imaging (HSI) is a technique that captures detailed spectral information across a wide range of wavelengths. In combustion analysis, it is used to analyze particle emissions and determine temperature distributions by examining wavelength-dependent intensities. Arkenstone implements hyperspectral imaging using its core scripts `Arkenstone\_Hyperspectral.py` and `Arkenstone\_wrapper.py`.

## 2. Theoretical Background

## 2.1 Principles of Hyperspectral Imaging

Hyperspectral imaging collects information from multiple spectral bands. This enables identification of material temperature as detailed below.

## 

Hyperspectral imaging for temperature estimation relies on blackbody radiation theory, governed by Planck’s law:  
  
L(λ, T) = (2hc² / λ⁵) \* [1 / (e^(hc/λkT) - 1)]  
  
where:  
- L(λ, T) = Spectral radiance at wavelength λ and temperature T  
- h = Planck's constant (6.626 × 10⁻³⁴ J·s)  
- c = Speed of light (3.00 × 10⁸ m/s)  
- k = Boltzmann’s constant (1.381 × 10⁻²³ J/K)  
- λ = Wavelength (m)  
- T = Absolute temperature (K)  
  
By analyzing emission intensities at different wavelengths, the temperature of emitting particles can be inferred using known blackbody radiation properties.

## 3. Implementation in Arkenstone

## 3.1 Data Acquisition/instrumentation

Arkenstone captures hyperspectral images using SIMX 8 (or other specialized cameras) that filter emissions using two band filters filtering different range of wavelengths:  
- Red Filter -long pass filter 650-800 nm (central λ ≈ 0.725 μm) - Captures emissions in the red.  
- Blue Filter -short pass filter 300-650 nm (central λ ≈ 0.5 μm) - Captures emissions in the blue.  
  
By computing the intensity ratio of these channels, temperature values are determined.

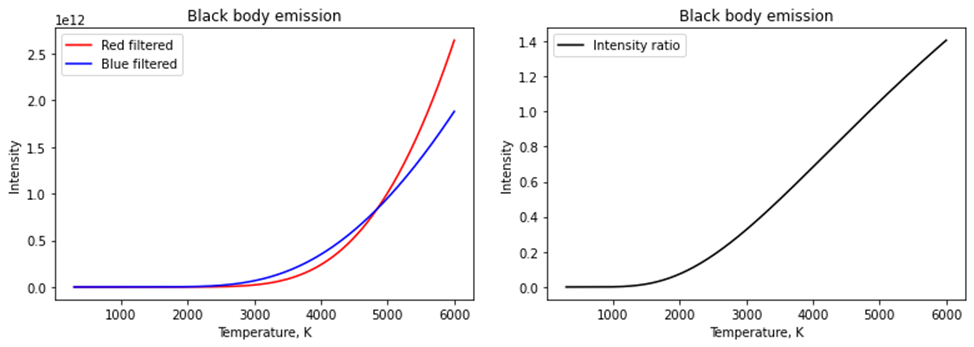
## 3.2 Image Preprocessing

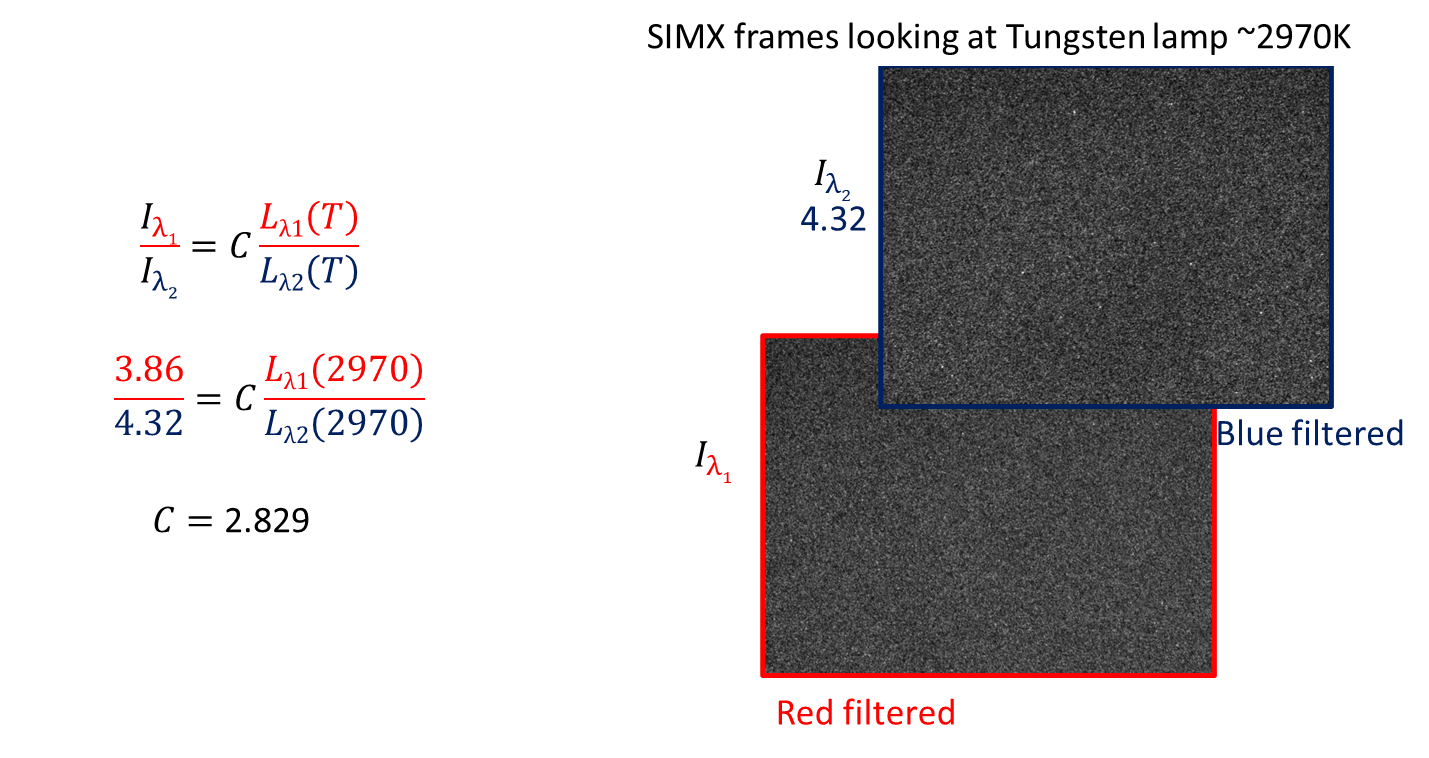
Preprocessing ensures accurate measurements, including:  
- Dark Field Subtraction: Removes background noise. (optional)  
- Normalization: Converts raw intensities to a standardized range.  
- Gaussian Filtering: Reduces noise and smooths images. (we use 5x5 convolutional smoothing)  
kernel = np.ones((5,5),np.float32)/(5\*\*2)  
blurred\_image = cv2.filter2D(emission\_fr,-1,kernel)



## 3.3 Temperature Estimation

## Step 1: Calibration with a Known Blackbody Source

A calibrated tungsten lamp (2970K) establishes the reference intensity ratio of red to blue emissions. The expected blackbody intensity ratio at each temperature is precomputed using Planck’s law. A calibration constant (C) is derived from experimental measurements:

Calibration constant for one red/blue frame combo = (Measured I\_red / Measured I\_blue) / Intensity\_Ratio (at 2970K)  
  
If calibration images are unavailable, a default calibration constant (2.829) is used based on prior calibration. Though use of fresh calibration data is strongly suggested.

## Step 2: Applying the Calibration to Unknown Samples

Compute the ratio of red to blue intensities in the sample:  
Ratio = Image\_red / Image\_blue  
  
Apply the calibration constant:  
Corrected\_Ratio = Ratio / calibration constant  
  
Map corrected ratios to temperature values using blackbody radiation tables:  
T\_image = np.interp(Corrected\_Ratio, Intensity\_Ratio(T), T)

## 3.5 Visualization & Output

Processed data is visualized as:  
- Temperature Maps: Color-coded heatmaps showing temperature distributions.  
- Histograms: Displays frequency distributions of hotspot temperatures.  
- Growth Tracking: Monitors changes in reaction zone size over time.  
- Time-Resolved Data: Tracks temperature variations across multiple frames.  
- Text Data Files: Stores processed temperature data in tab-separated format.

## 4. Challenges & Future Improvements

## 4.1 Sensitivity to Calibration

Temperature accuracy depends on calibration quality. Variations in lamp intensity or sensor characteristics can introduce errors.

## 4.2 Computational Demands

Hyperspectral image processing is computationally expensive. Future improvements could optimize interpolation algorithms and implement GPU acceleration.

## 4.3 Noise & Background Interference

Thermal emissions from background surfaces can interfere with particle detection. Improved background subtraction techniques can help mitigate these issues.

## 5. Conclusion

Hyperspectral imaging in Arkenstone provides a robust method for analyzing temperature distributions of shock-initiated particles. By leveraging spectral intensity ratios and blackbody radiation models, it accurately determines temperature maps. Continued advancements in calibration, noise reduction, and computational methods will further enhance its precision and efficiency.

# SECTION-III: Error Handling & Troubleshooting for Arkenstone (updated on 02272025)

## 1. Introduction

This document outlines common errors encountered while running the Arkenstone software, the built-in error handling mechanisms, and troubleshooting steps for resolving issues.

## 2. Built-in Error Handling Capabilities

Arkenstone incorporates several error handling mechanisms to ensure smooth execution. These include exception handling, data validation, and user feedback through the GUI.

### 2.1 Exception Handling

• The scripts catch and handle common exceptions such as missing files, incorrect inputs, and segmentation failures.  
• Errors are displayed via Tkinter message boxes (GUI) or printed to the console.  
• Logging mechanisms can be added for debugging complex issues.

### 2.2 Data Validation

• Input images and calibration data are checked for existence and format correctness before processing.  
• If required files are missing, the user is prompted to provide them before proceeding.

### 2.3 GPU and Processing Mode Detection

• The script automatically detects if CUDA (GPU) is available and selects the appropriate processing mode.  
• If CUDA is unavailable, the software runs in CPU mode, with performance warnings provided.

## 3. Common Errors & Troubleshooting

### 3.1 CUDA and PyTorch Version Mismatch

If PyTorch does not recognize the installed CUDA version, ensure that your PyTorch installation matches the installed CUDA version. To check CUDA version:

nvcc --version

To check PyTorch CUDA compatibility, run:

python -c 'import torch; print(torch.version.cuda)'

If there is a mismatch, uninstall the incorrect version and reinstall the correct PyTorch version using:

pip uninstall torch torchvision torchaudio

pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118

(Replace 'cu118' with your correct CUDA version, e.g., cu117 for CUDA 11.7)

### 3.2 Missing Dependencies

If you encounter a ModuleNotFoundError, ensure all required packages are installed using:

pip install -r requirements.txt

### 3.3 GUI Freezing or Crashing

If the Tkinter-based GUI crashes or freezes:  
• Ensure your system has the latest Python and Tkinter versions.  
• Try running the script with administrator privileges.  
• Close unnecessary applications to free up memory.

### 3.4 Segmentation Errors

If the segmentation model fails to detect particles correctly:  
• Ensure images are in grayscale format.  
• Try reducing segmentation thresholds in `Arkenstone\_SAM.py`.  
• If using Random Forest, ensure the model is trained on similar data.

### 3.5 Performance Issues

If the software runs slowly:  
• Reduce the image resolution before processing.  
• If running on CPU, consider enabling GPU processing.  
• Reduce the number of segmentation points in `Arkenstone\_SAM.py`.

## 4. Limitations of Arkenstone

While Arkenstone is a powerful tool for analyzing shock-induced emissions, it has certain limitations:

### 4.1 Hardware Dependence

• Running on CPU is significantly slower than GPU.  
• Requires a high-performance GPU for optimal segmentation performance. Future version will improve random forest classifier to ensure an option of lower complexity.

### 4.2 Data Quality Sensitivity

• Poor-quality images can lead to inaccurate segmentation results.  
• Requires well-calibrated hyperspectral data for accurate temperature mapping.

### 4.3 Limited Model Generalization

• The Random Forest model may need retraining for different datasets.  
• SAM-based segmentation may require fine-tuning for specific conditions.

## 5. System Settings & Dependencies

The following system settings and Python dependencies were used to successfully run Arkenstone:  
(Highlighted packages are primary to Arkenstone)  
PyQt5.sip: 12.13.0  
gmpy2: 2.2.1  
importlib\_resources: 6.4.0  
numexpr: 2.10.1  
pure\_eval: 0.2.2  
ipykernel: 6.29.5  
mpmath: 1.3.0  
platformdirs: 3.10.0  
traitlets: 5.14.3  
jedi: 0.19.2  
matplotlib: 3.9.2  
pygments: 2.15.1  
cloudpickle: 3.0.0  
six: 1.16.0  
cycler: 0.11.0  
tornado: 6.4.2  
psutil: 5.9.0  
sip: 6.7.12  
parso: 0.8.4  
decorator: 5.1.1  
colorama: 0.4.6  
torch: 2.5.1  
torchvision: 0.20.1  
cuda version 11.8  
IPython: 8.15.0  
chardet: 4.0.0  
asttokens: 2.0.5  
numpy: 1.26.4  
typing\_extensions: 4.12.2  
spyder\_kernels: 3.0.2  
pandas: 2.2.3  
pyparsing: 3.2.0  
packaging: 24.2  
prompt\_toolkit: 3.0.43  
debugpy: 1.8.11  
stack\_data: 0.2.0  
bottleneck: 1.4.2  
executing: 0.8.3  
pickleshare: 0.7.5  
exceptiongroup: 1.2.0  
jupyter\_client: 8.6.3  
kiwisolver: 1.4.4  
dill: 0.3.8  
sympy: 1.13.3  
comm: 0.2.1  
zipp: 3.21.0  
matplotlib\_inline: 0.1.6  
importlib.metadata: 8.5.0  
jupyter\_core: 5.7.2  
importlib\_metadata: 8.5.0  
segment\_anything: 1.0  
soupsieve: 2.5  
pytz: 2024.1  
backcall: 0.2.0  
defusedxml: 0.7.1  
wcwidth: 0.2.5  
PyQt5: 5.15.10

Hardware used was an Asus Zenbook laptop  
CPU: 13th Gen Intel® Core™ i9-13900H, 32GB RAM and GPU: NVIDIA GeForce RTX 4070 laptop version (GPU memory dedicated 8GB and shared 16GB)  
(20 runs of SIMX8 takes <10 mins to process.)

## 6. Contact & Support

For further assistance, consider reaching out to Siva Kumar Valluri ([svalluri@illinois.edu](mailto:svalluri@illinois.edu)).