arcjetCV: an open-source software to analyze material ablation

Alexandre M. Quintart [*]
Flying Squirrel, Bourg-Saint-Pierre, Valais 1946, Switzerland

Magnus A. Haw[†]
NASA Ames Research Center, Moffett Field, CA 94035, USA

Federico Semeraro[‡]
AMA at NASA Ames Research Center, Moffett Field, CA 94035, USA

arcjetCV is an open-source Python software designed to automate time-resolved measurements of heatshield material ablation and recession rates from arcjet test video footage. This new automated and accessible capability enables two-dimensional (2D) in-situ recession characterization from profile video. arcjetCV automates the video segmentation process using two primary machine learning models: a one-dimensional (1D) Convolutional Neural Network (CNN) to infer the time-window of interest and a 2D CNN for image segmentation. A graphical user interface (GUI) simplifies the user experience and an application programming interface (API) allows users to call the core functions from scripts, enabling batch video processing. arcjetCV's capability to measure time-resolved recession in turn enables characterization of non-linear processes (non-linear recession, shrinkage, swelling, etc.), contributing to higher fidelity validation and improved modeling of heatshield material performance. The source code associated with this article can be found at https://github.com/magnus-haw/arcjetCV.

Keywords: Arcjet; Recession Tracking; Machine Learning; Computer vision; Porous Ablator; Open-source.

Nomenclature

CNN	=	Convolutional Neural Network
HSV	=	Hue, Saturation, Value
$HyMETS$	=	Hypersonic Materials Environmental Test System
IHF	=	Interaction Heating Facility
ROI	=	Region Of Interest

[*]Aerospace Engineer, Flying Squirrel, Switzerland, alex@flying-squirrel.space
[†]Plasma physicist, Thermal Protection Materials Branch, NASA Ames Research Center, MS 234-4, Member AIAA, magnus.haw@nasa.gov
[‡]Aerospace Engineer, Thermal Protection Materials Branch, NASA Ames Research Center, federico.semeraro@nasa.gov

I. Introduction

Thermal Protection Systems (TPS) are critical components for spacecraft, safeguarding them during atmospheric entry from extreme heat and pressure. Ablative TPS materials, which dissipate heat by undergoing controlled degradation, are rigorously tested in ground facilities including arcjets. Arcjets, or plasma wind tunnels, simulate atmospheric entry conditions, exposing materials to high-enthalpy flows with heat fluxes up to 6000 kW/m^2, stagnation pressures exceeding 10 kPa, and partially ionized plasma (~2 eV). These facilities provide invaluable data for assessing TPS performance. Recession rates of TPS materials are typically determined through pre- and post-test measurements of sample height and recession rates are calculated assuming a linear ablation rate. This assumption fails to account for time-dependent effects such as in-situ swelling/shrinkage, non-linear recession, recession occurring during sample insertion/extraction, and any post-test swelling/shrinkage/recession. These effects are magnified for short test dwell times (< 5 s) where the transient effects are significant.

To address these limitations, researchers have explored methods for capturing time-resolved recession. Sherrouse and Carver (1992) developed a laser triangulation-based recession diagnostic to measure surface profiles of flat materials in real-time [1]. Schairer and Heineck (2012) developed a Photogrammetric Recession Measurement (PRM) technique, using synchronized video cameras to capture spatially distributed, time-resolved 3D recession data for hemispherical samples [2]. Callaway et al. (2012), also used photogrammetry to quantify three-dimensional shape changes in low-temperature ablator models during supersonic flow tests [3]. Similarly, Loehle et al. (2014) applied stereoscopic photogrammetry during plasma wind tunnel tests, achieving high-resolution, real-time surface analyses [4]. Winter et al. (2018), combined emission spectroscopy with material seeding techniques to develop a method for spectroscopic remote sensing of material recession [5]. Additional methods can be found in a review of ablative measurement methods by Yang and Koo (2020) [6]. These studies demonstrated the benefits of capturing transient material behaviors but were too involved to be adopted as standard practice for arcjet tests.

Profile video footage is a standard diagnostic in all test facilities and is ostensibly simple to process for recession: track a bright edge against a dark background. However, arcjet videos are unexpectedly difficult to process. This is due to the samples under test undergoing large changes in brightness and color over time, issues with camera and sample vibrations, inconsistent background lighting, and projection errors. Consequently, extracting time-resolved recession via simple edge tracking methods (brightness/color/edge filtering) in arcjet videos has remained a labor-intensive task requiring significant manual tweaking [7].

To address this challenge, *Arcjet Computer Vision (arcjetCV)* was developed as a software application to automate the analysis of arcjet video footage. By leveraging computer vision and machine learning techniques, arcjetCV enables time-resolved tracking of material recession, sting arm motion, and shock-material standoff distance. Unlike traditional methods, arcjetCV processes standard video footage without requiring additional instrumentation, making it highly accessible and scalable. Moreover, its ability to analyze archived video footage from tests performed long ago opens the door to extracting new insights from historical datasets, further enhancing its value for facilities seeking to maximize the utility of their existing records. Initial applications revealed unexpected non-linear material behaviors and provided critical recession rate data, demonstrating the software's potential to improve TPS testing and further development of TPS material modeling.

II. Software Architecture

arcjetCV is developed in Python and leverages PySide6 [8] and matplotlib [9] to construct its Graphical User Interface (GUI). The GUI is built using PySide6, leveraging the Python bindings to the Qt framework for a responsive and intuitive interface. This design enables researchers to adjust settings, execute analyses, and review results seamlessly, making complex image processing tasks accessible to a wide range of users.

At the core of arcjetCV's design is the Model-View (MV) architecture, a paradigm that effectively separates the application's data management (the model) from the user interface (the view), enhancing modularity and maintainability.

The model layer plays a critical role in managing the application's central logic and data. It employs OpenCV [10], a comprehensive open-source library renowned for its wide range of image processing capabilities, including advanced filtering, transformation, and object detection functions. Additionally, arcjetCV integrates PyTorch [11] for its machine learning operations, specifically the implementation of CNNs. This modern deep learning library is selected for its efficiency, flexibility, and extensive support for neural network architectures, making it indispensable for arcjetCV's image analysis tasks.

A. Graphical User Interface

The GUI consists of three main tabs corresponding to the primary stages of analysis provided by arcjetCV: "Calibration" for ensuring accurate conversion of pixel measurements to physical units (figure 1 and 2),"Extract Edges" for video processing (figure 3), and "Analysis" for post-processing the results (figure 4).

- **Calibration tab:** facilitates the setup of accurate scaling between pixel measurements and real-world dimensions. It includes intrinsic calibration, which corrects lens distortions and estimates internal camera properties, as well as extrinsic calibration, which establishes the pixel-to-millimeter ratio and corrects the camera's orientation using calculated rotation vectors to ensure properly aligned measurements.
- **Video processing tab:** identifies and extract the edges of the sample and/or shock from video frames. The profiles obtained from this extraction are saved in a JSON file for subsequent analysis.
- **Post-processing tab:** allows the visualization of these extracted edges and further refines the data to plot material recession over time.

Notably, arcjetCV supports several video formats, including *.mp4, *.avi, *.mov, and *.m4v, ensuring wide applicability across various datasets.

1. Calibration

The calibration process in arcjetCV ensures the accurate conversion of pixel-based measurements into physical units (e.g., millimeters). This is crucial for deriving meaningful metrics, such as material recession rates, shock standoff distances, and other time-resolved measurements. The calibration process involves two primary steps: Intrinsic calibration and metric calibration.

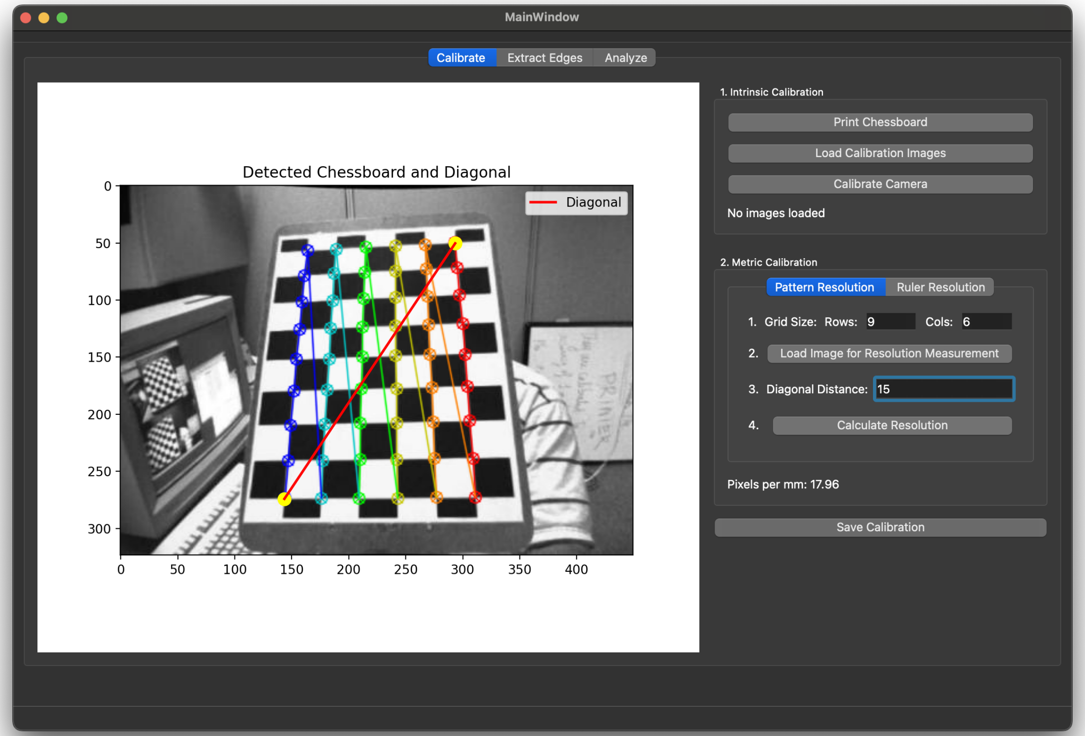
Intrinsic Calibration Intrinsic calibration corrects for distortions introduced by the camera's lens and determines the camera's internal parameters, such as focal length, optical center, and distortion coefficients. This step ensures that the measurements are not affected by lens-induced errors. The procedure includes:

1) Capture Calibration Frames: Calibration requires several images of a calibration pattern with known dimensions, such as a checkerboard or a circular grid, typically captured prior to the test.
2) Detect Calibration Pattern: The calibration pattern in each image is automatically detected. OpenCV's `findChessboardCorners` is used to detect checkerboard patterns, while `findCirclesGrid` is used for circular grids. The program selects the appropriate detection algorithm based on the detected pattern.
3) Compute Camera Matrix and Distortion Coefficients: Using OpenCV's `calibrateCamera` function, the camera's intrinsic parameters and lens distortion coefficients are derived from the detected pattern features.
4) Undistort Frames: The distortion coefficients are then applied to undistort all video frames, ensuring accurate geometric measurements.

The automatic pattern detection ensures flexibility in the calibration process, allowing users to utilize either checkerboard patterns or circular grids without manual selection. The resulting intrinsic calibration parameters are stored in the `VideoMeta` class for use throughout the video processing pipeline.

Metric Calibration Metric calibration establishes the pixel-to-millimeter ratio, enabling the translation of image-based measurements into real-world dimensions. This step uses either a reference size from a calibration pattern (checkerboard or circles grid) or a manually marked reference in the video frames. The process includes:

1) Enter Grid Size: The user inputs the grid dimensions (rows and columns) of the calibration pattern to ensure accurate scaling.
2) Load Image for Resolution Measurement: The user loads an image of the calibration pattern with a known diagonal measurement for reference.
3) Enter Diagonal Distance: The user inputs the known physical dimension of the diagonal (e.g., its length in millimeters).
4) Calculate Pixel-to-Millimeter Ratio: The pixel distance of the diagonal is computed and divided by the known physical distance to determine the pixel-to-millimeter conversion factor.



No images loaded

1. Grid Size: Rows: 9 Cols: 6

3. Diagonal Distance: 15

Pixels per mm: 17.96

Fig. 1 GUI Calibration with the pattern resolution tool that automatically detects the calibration pattern and uses the diagonal distance to set the metric resolution, ensuring precise pixel-to-millimeter scaling for accurate video analysis.

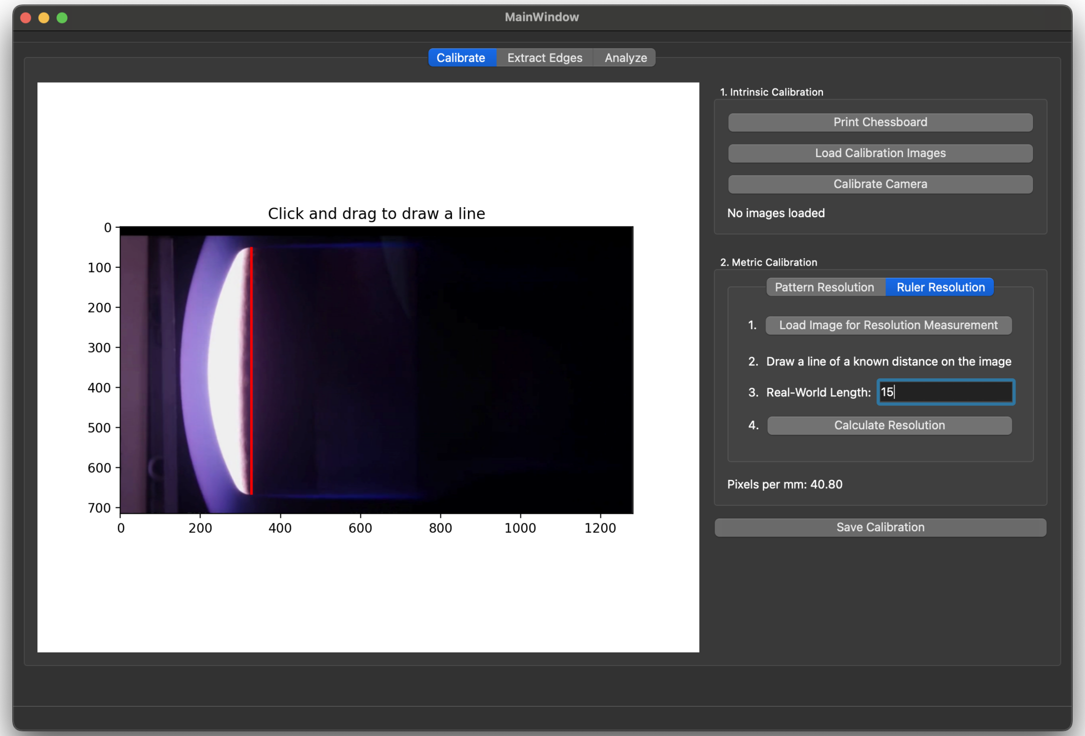
Alternatively, the software provides a Ruler Resolution tab for non-pattern-based calibration. In this mode:

1) Load Calibration Frame: The user loads an image where the reference object or line is visible.

2) Draw Line: The user draws a line on the image representing a known physical length.

3) Enter Real-World Length: The user specifies the real-world length of the drawn line.

4) Calculate Resolution: The software computes the pixel-to-millimeter conversion factor based on the drawn line's length.



No images loaded

Real-World Length: 15

Pixels per mm: 40.80

Fig. 2 GUI Calibration with the ruler resolution tool that allows the user to manually draw a reference line on the calibration image and input the real-world length to set the metric resolution, ensuring accurate pixel-to-millimeter scaling for video analysis.

This calibration step allows the user to choose either automated or manual methods for determining resolution. The resulting scaling factor is stored in the video metadata and applied to subsequent measurements, ensuring that all pixel-based calculations are properly translated into real-world units.

In addition to the calibration steps outlined above, the following outlines recommendations for acquiring calibration images,

- **Ensure alignment:** Calibration patterns used to determine camera orientations should have a mounting system consistent with the samples to be tested.
- **Calibrate under vacuum:** Due to the difference in refractive index between air and vacuum, vibrations, and other operational changes, calibration should occur under conditions as similar to the actual test as possible.
- **Add in-situ fiducials to mounts:** In-situ fiducials visible both in the video and the calibration images are very valuable for isolating different sources of motion (camera vibrations, expansion of sample mounting adhesive, motion of mounting arm etc.)

2. Video Processing

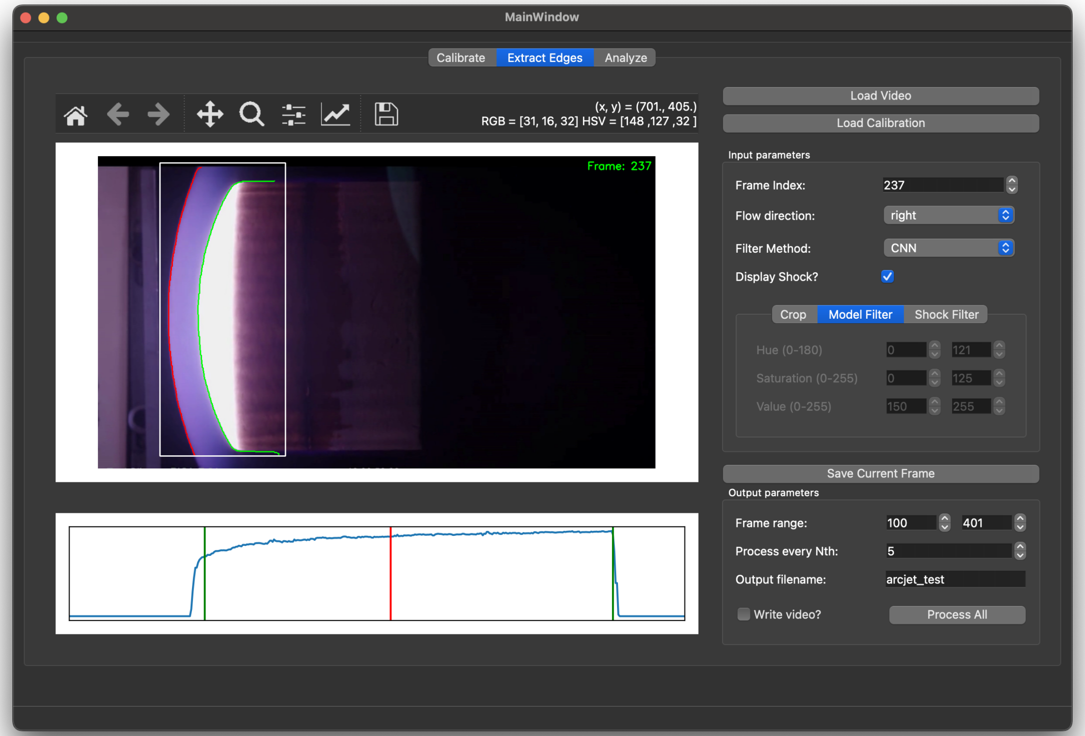
The video processing step is computationally demanding (see Section II.C), potentially taking several minutes to complete, depending on the size of the video. To balance efficiency and accuracy, it is often sufficient to analyze only a subset of frames (e.g., every 10th frame), as material recession is typically slow compared to the video frame rate. For high frame rate videos, fewer frames can be analyzed without compromising temporal resolution, whereas for slower acquisition speeds or more rapid material changes, a higher sampling rate may be necessary. This approach should be adjusted to the specifics of each experiment to ensure that all critical details are captured. The essential steps in the video processing workflow are as follows:

1) Time segmentation of the video (find first and last frames of interest).
2) Define a region of interest (ROI) to minimize processing per frame.
3) Identify the direction of the arcjet flow (left, right, up, down).
4) For each frame, classify ROI's pixels into 4 classes, as described in section III.B.
5) For each frame, extract leading edge from pixels classified as part of the sample.

In addition to these primary steps, the quality of the video plays a crucial role in ensuring the accuracy and reliability of the results and should satisfy quality thresholds. The following provides basic metrics and rules of thumb to record quality video.

- **Brightness:** Oversaturation and underexposure negatively affect the accuracy of edge detection. Saturation causes important features to be lost, while underexposure reduces contrast, making it harder to distinguish the shock and sample edges. For best results, the sample edge brightness should not be saturated and remain three sigma above the background brightness distribution.
- **Blur:** Blur caused by camera motion or incorrect focus will degrade the sharpness of the image and increase the uncertainty of edge position. The system should minimize blur and vibrations.
- **Zoom and Pixel Density:** Changes in zoom typically change the distance per pixel and the focus. This can void pre-test calibrations and it is recommended to keep the zoom fixed for a given video.
- **Focus and Depth of Field:** The edge of the shock and the sample must remain in focus. For typical sample dimensions (1"-6") ensuring the depth of field is > 3 mm is typically sufficient. Maintaining sharp focus on these critical features ensures high-quality extraction of edges, especially in fast-moving or highly detailed scenes.
- **Camera Orientation:** 3D projection effects can skew results so the camera should be centered and the field of view oriented perpendicular to the recession direction (e.g., 8 degrees off of perpendicular gives a 1% projection error).

Following these recommendations will ensure robust and accurate recession measurements. Additionally, real-time feedback and manual adjustments through the GUI allow users to fine-tune settings for improved results.



(x, y) = (701., 405.)
RGB = [31, 16, 32] HSV = [148 ,127 ,32]

Frame Index: 237
Flow direction: right
Filter Method: CNN
Display Shock? ✓

Hue (0-180) 0 121
Saturation (0-255) 0 125
Value (0-255) 150 255

Frame range: 100 401
Process every Nth: 5
Output filename: arcjet_test
Write video?

Fig. 3 GUI Video Processing window that enables users to define the region of interest (ROI), determine the recession direction, and apply segmentation filters such as CNN, HSV, autoHSV or grayscale intensity. These filters extract edges of the sample and shock regions, saving the results in a JSON format for further analysis.

For the segmentation process to function correctly, the pipeline requires various inputs, including the first and last frames, ROI, and the recession direction. arcjetCV automatically deduces these parameters upon loading a video. The selection of the first and last frames is guided by the time-segmentation model highlighted in section III.A. The ROI determination involves classifying the initial and final frames using the CNN model outlined in section III.B, followed by creating a buffered bounding box around the identified sample. The flow (and therefore recession) direction is inferred by tracking the position of the image centroid in a few key frames. Even in videos with minimal recession, the heating of the sample holder and arm typically gives a strong shift in the image centroid over time. Users also have the option to manually adjust these inferred settings (ROI, recession direction, first/last frames) through the GUI or API as needed.

The software offers four segmentation filters for the shock and the sample. An overview of these available segmentation methods is provided below:

- **CNN**: utilizes a convolutional neural network that has been trained on various frames as the core method. While this approach shows strong performance in detecting the edges of shock and sample, it may not perform well on highly underexposed frames [12, 13].
- **AutoHSV**: employs a combination of preset HSV (Hue, Saturation, Value) ranges to identify contours, opting

for the HSV color model over RGB for its simpler and more intuitive thresholding capabilities. This method is effective for segmenting based on color, saturation, and intensity [14].

- **HSV**: identifies contours through the use of adjustable HSV ranges. A desktop interface offers real-time display of RGB, HSV, and pixel position values upon mouse hover, aiding the precise adjustment of these ranges. This is especially helpful for manual processing of frames with low exposure [14].
- **Gray**: detects the leading edge of the sample using a grayscale intensity threshold. This method will only capture a single edge where the other methods will try to capture both sample and shock edges [14].

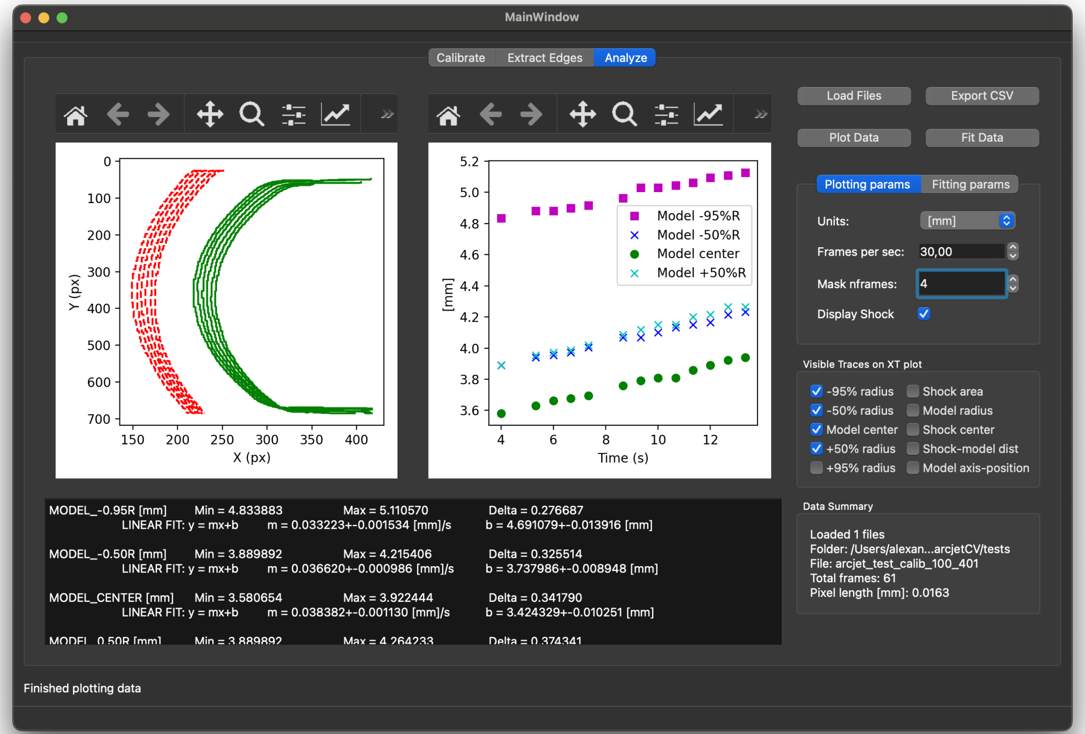
Upon configuring all the necessary input parameters, the targeted video sub-sequence can be processed by clicking the "Process All" button. This action generates a JSON file with the intermediate edge traces, which is then forwarded to the data analysis tab of the GUI, along with an optional video output. A Python API to handle these requests has also been developed (see section II.B).

The CNN filter is the default and works quite well for most videos. If the CNN filter is not performing well on a given frame, resizing the ROI slightly can improve results. If this is insufficient, any important frames (e.g., the first frame which is typically underexposed) can be manually segmented with the HSV filter, exported and then merged into a larger set in the analysis tab. The post-processing phase is designed to accommodate multiple processed sequences, facilitating manual edge case corrections.

3. Data Analysis

The post-processing stage in the pipeline focuses on plotting, cleaning, calibrating, and fitting the edge traces extracted during the video processing phase. The "Analysis" tab of the GUI (Figure 4) loads one or more JSON files from a processed video. These loaded edges can then be visualized with the "Plot Data" button which will display the traces on the two plotting windows. The left plot (XY) shows a subset of the extracted traces and the right plot (TX) displays a range of possible time series data (e.g, recession at discrete positions as function of sample radius, sample area, shock-sample standoff distance, model vertical position, etc.). The pixel values are scaled to real-world units based on the calibration parameters.

Upon displaying the data, users have the option to apply linear fits to the visible time series, with both the fits and corresponding fit parameter error bars showcased in the lower part of the window. This feature is particularly valuable for analyzing test cases characterized by linear recession, facilitating the standard processing workflow. However, a poor linear fit can also indicate when a particular sample has non-linear recession. To conclude the analysis, the plots, time-series and fitting data can be exported to PNG/CSV files.



Units: [mm]
Frames per sec: 30,00
Mask nframes: 4
Display Shock ✓

✓ -95% radius
✓ -50% radius
✓ Model center
✓ +50% radius
+95% radius
Shock area
Model radius
Shock center
Shock-model dist
Model axis-position

```
Loaded 1 files
Folder: /Users/alexan...arcjetCV/tests
File: arcjet_test_calib_100_401
Total frames: 61
Pixel length [mm]: 0.0163
```

```
MODEL_-0.95R [mm]     Min = 4.833883          Max = 5.110570          Delta = 0.276687
          LINEAR FIT: y = mx+b     m = 0.033223+-0.001534 [mm]/s     b = 4.691079+-0.013916 [mm]

MODEL_-0.50R [mm]     Min = 3.889892          Max = 4.215406          Delta = 0.325514
          LINEAR FIT: y = mx+b     m = 0.036620+-0.000986 [mm]/s     b = 3.737986+-0.008948 [mm]

MODEL_CENTER [mm]     Min = 3.580654          Max = 3.922444          Delta = 0.341790
          LINEAR FIT: y = mx+b     m = 0.038382+-0.001130 [mm]/s     b = 3.424329+-0.010251 [mm]

MODEL_0.50R [mm]      Min = 3.889892          Max = 4.264233          Delta = 0.374341
```

Finished plotting data

Fig. 4 GUI Data Analysis window that provides tools to visualize and analyze processed data, including recession metrics and shock standoff distances. Users can interact with graphical outputs, compare results, and export data for further evaluation.

B. Python API

The arcjetCV framework is designed to streamline video processing tasks, particularly focusing on the analysis of arcjet videos. It encapsulates this functionality within three primary classes: `Video`, `VideoMeta`, and `ArcjetProcessor`, each serving a distinct role in the video processing pipeline.

Video Class The foundational layer for manipulating video files, utilizing OpenCV to offer a user-friendly interface for various video operations. It simplifies the complexities of video file handling, providing straightforward access to individual frames, video attributes, and the generation of processed output videos. The class enables easy retrieval of specific video frames for processing, and supports the creation of new video files from processed frames.

VideoMeta Class Complementing the `Video` class, `VideoMeta` specializes in the management of video metadata. It extends Python's dictionary class with JSON-based metadata storage, improving the preservation and accessibility of video processing parameters and analytical data. This class is designed for efficient metadata handling, capturing processing parameters and video analysis results, supporting JSON format for easy metadata saving and retrieval, and allowing for dynamic adjustments to video processing parameters.

ArcjetProcessor Class At the core of arcjetCV's processing capabilities, this class conducts several image processing tasks, including segmentation, edge detection, and feature extraction. Many of its methods are described next.

C. Recommended Computer Architecture and Computational Performance

All parts of arcjetCV can be used on any modern computer but we provide guidelines here for recommended hardware as the machine learning models are optimized for systems with GPUs:

- **Processor:** Apple M1/M2 chip or Intel Core i7 (8th gen or newer)
- **Memory (RAM):** 8 GB or more
- **GPU (optional for non-M1 systems):** NVIDIA GTX 1080 Ti or newer
- **Storage:** SSD with at least 512 GB
- **Operating System:** macOS 12+, Ubuntu 20.04+, or Windows 10/11

Computational Performance: On a tested system (MacBook Air M1, 2020), processing 30 frames at a resolution of 1282 × 716 pixels yielded the following results:

- **Processing Time:** 42 seconds
- **Memory Usage:** 1.4 GB RAM
- **Inference Speed:** 0.7 frames per second

Since the CNN only processes the ROI box, most videos with a suitably constrained ROI (typically much smaller 1282 × 716) will perform better than the values listed above.

III. Methodology

In this section, we describe the methodologies used by arcjetCV to automate video analysis. This encompasses machine learning architectures and training methods, image processing techniques, and post-processing methods designed to consolidate the extracted features into a user-friendly format.

A. Time Segmentation

The process of identifying the time interval of interest is automated through a 1D CNN, which has been trained on synthetic brightness data. The model takes in a normalized time series of video brightness and then provides semantic classification (i.e., continuous probabilistic classification at pixel level) of the time series into 3 classes: *OFF*, *TRANSITION*, and *ON*. The 3 classes were implemented to capture the 3 distinct behaviors in a video: *OFF* corresponds to the pre/post insertion state where the video is typically dark, *TRANSITION* corresponds to the insertion/extraction portions, and *ON* corresponds to the sample dwelling in the flow. This breakdown into 3 classes proved more robust and more generalizable than a simple ON/OFF classification. The model outputs a probability for each class at each time.

Classification of a typical brightness time series is illustrated in Figure 5 where the probability of each class is plotted on top of the input brightness trace. The input brightness curves are normalized to provide a consistent input metric which can be used for arbitrary video formats with the **normalized mean brightness** defined as:

$$\text{Normalized Mean Brightness} = \frac{\sum_{i=1}^{N} I_i}{N \times I_{\max}}$$

where:

- N is the total number of pixels in each frame.
- I_i is the intensity of each pixel i.
- $I_{\max}$ is the maximum intensity value (e.g., 255 for an 8-bit image).

By automatically finding the start/stop times, the model automates an important step that previously required manual inputs in the analysis workflow.

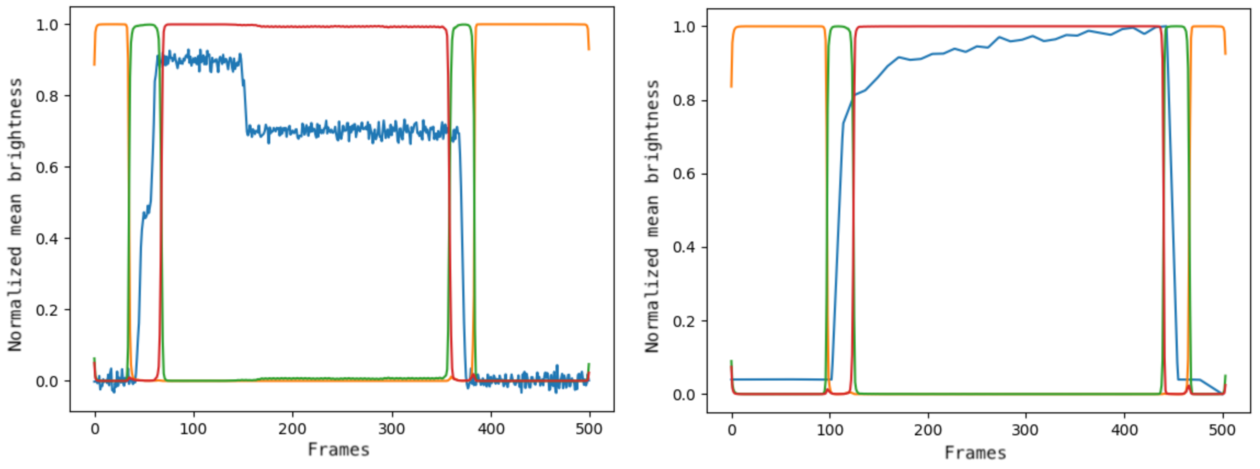
Architecture The Conv1DNet architecture, combines 1D convolutional and transposed convolutional layers. This fully convolutional design effectively captures and reconstructs temporal patterns and provides semantic classification for arbitrary length sequences. The network starts with two convolutional layers, expanding and then compressing the feature space from 1 to 32 and then to 16 channels, respectively. Each layer uses a kernel size of 7, stride of 2, and padding of 3, with 25% dropout rate after each layer to mitigate overfitting. Following convolution, three transposed convolutional layers incrementally reconstruct the time series dimensionality. These layers adjust the feature space from 16 channels back to 3, correlating to the desired segmentation outputs. ReLU activation functions are applied post each convolutional layer to introduce non-linearity, except for the final layer where a softmax function is utilized. The softmax output enables a probabilistic interpretation of the segmentation results, categorizing each time step into one of the three segmentation classes.

Dataset Generation The training dataset needed to replicate a typical brightness trace from an arcjet test where a model is inserted, dwells for a period of time longer than 1 seconds, and is then extracted. Unfortunately, the set of videos with suitable brightness traces was limited (<50) so a synthetic dataset was generated using top hat signals augmented with noise and additional features. Each signal corresponds to three superimposed top hat traces with random noise, height, width, and includes brightness spikes (also with random height). These multi-step traces are designed to reflect the dynamic changes in arcjet video camera settings (in-test changes of exposure, filters, ISO, iris etc.). To further enhance the realism of these artificial signals, a smoothing process is applied using a Hanning filter. Figure 5a provides a visual representation of an artificial brightness trace as well as the labeled classes employed to train the time segmentation model. The dataset was generated within the following ranges:

- **Amplitude:** 0.7 to 0.9,

- **Spike Amplitude:** 0 to 0.3,
- **Noise Amplitude:** 0 to 0.015,
- **Sequence Length:** 500 time steps,
- **Top Hat Width:** 10-300 time steps.

The normalized synthetic dataset covers all brightness traces of interest provided that the interval between input points is greater than 0.1 seconds. Faster time cadences (e.g., from exceptionally high-speed video) can result in video brightness traces with gradual slopes too small for the model to correctly process.

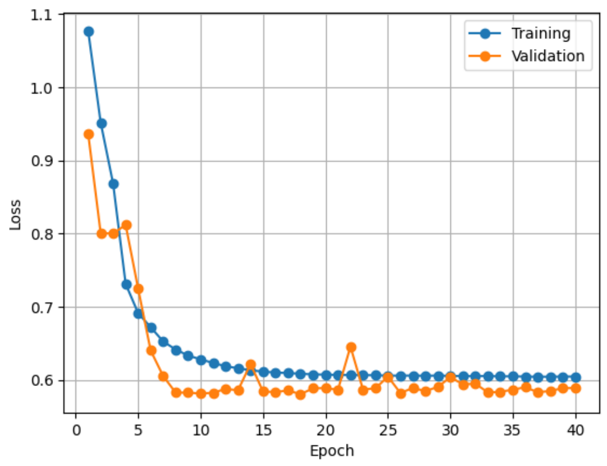


(a) Synthetic data (blue) and classification (orange, green, red) **(b) Real data (blue) and classification (orange, green, red)**

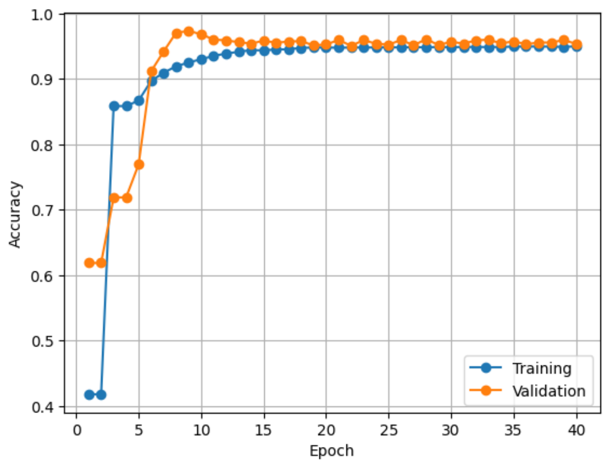
Fig. 5 Classification outputs of 1D-CNN segmentation model. For each plot, the blue trace is the input brightness trace and the other traces represent the output probabilities from the model for each class: orange-*OFF*, green-*TRANSITION*, and red-*ON*.

Training and Results The training was conducted over 40 epochs, as shown in figure 6a, using a dataset of 1000 synthetic samples. Due to the low dimensionality of the model and data, the full 40 epoch training could be done in less than 5 minutes on a personal laptop. This allowed the synthetic dataset and 1DCNN training to be iterated many times to produce a model that performed well on real data. Figure 5b illustrates the 1D CNN accurately predicting phases in a video's normalized brightness signal. At the end of the training the 1D CNN reached an accuracy of 96% on the validation dataset composed of real traces, as shown in figure 6b.

It is expected that this model will generalize well to arbitrary top hat traces with normalized intensity and time cadence since the model performed equivalently (accuracy = 0.96) on an arbitrarily large synthetic dataset (10^6 traces) which effectively constitutes the full range of desired inputs. It has also been successfully ported to classify start/stop times of other arcjet diagnostics (arc current, arc voltage, pressure etc.) and for automating condition segmentation [15].



(a) Loss curve showing the decline in training and validation loss over 40 epochs, indicating model optimization during training.



(b) Accuracy curve showing training and validation accuracy increasing, with the model achieving 96% accuracy on validation data by the end of training.

Fig. 6 1D CNN model training and validation results.

B. Image Segmentation and Edge Detection

A 2D CNN was developed to improve the automatic segmentation of videos, aiming to track the leading edges of material samples and shocks. The model predicts 4 classes: *background*, *sample*, *sample-edge* and *shock* (see figure 8). Pretrained on ImageNet [16], the model was fine-tuned on a manually annotated dataset described below. The trained network demonstrated robust performance across all tested videos and enabling fully automated video processing.

Architecture The model combines a standard UNet architecture with the Xception encoder [17] from the "Segmentation Models Pytorch" package [18]. The frame of the video to be analyzed is first scaled with padding to a 512x512 square, which is then passed to the model for prediction. Structured into three main flows, the model features an entry flow for initial feature extraction, a middle flow dedicated to deeper analysis through the repeated application of depthwise separable convolutions, and an exit flow that compiles and prepares the outputs. UNet was chosen following a comparison of various architectures, including FCN, SegNet, PSPNet.

Dataset Generation For the training of the machine learning models within arcjetCV, it was imperative to construct a comprehensive dataset. This dataset was assembled from 400 frames extracted from 40 distinct arcjet videos (10 frames per video). We noticed that adding too many frames from a given video would result in overfitting (e.g., trained model would not generalize to new videos), likely because subsequent frames are not sufficiently distinct. Each of these frames underwent a manual segmentation process, resulting in the creation of masks that delineate the four classes of interest. To streamline the segmentation process of our dataset, we developed a specialized tool based on the Segment Anything Model (SAM) [19–21], combined with the grabcut function [22] provided by OpenCV. Following the segmentation of the original frames, the dataset was extended using a series of data augmentation techniques provided by the Albumentations library [23] to enhance its diversity. These augmentation strategies included rotating the frames, altering the positions of the segmented elements within the frames, and modifying the color schemes, amounting to a final total of 1778 training frames. Due to the fairly consistent composition and layout of most videos (bright sample, colored shock, dark background), data augmentation was very effective in producing good models from limited input data. For example, we were previously able to get 95% accuracy with only 200 frames (with augmentation) in the training set.

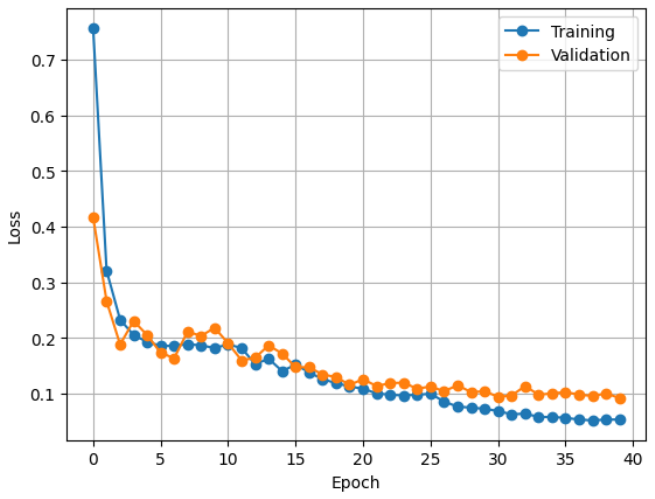
Training and Results Three distinct UNet models were considered, as shown in table 1, with training durations ranging from 15 to 40 epochs. The Xception encoder performed the best and the 25-epoch training regimen was chosen to avoid overfitting. Specifically, monitoring both the training and validation metrics indicated that beyond approximately 25 epochs, the training loss continued to decrease while the validation loss plateaued (see figure 7a). Likewise, the validation accuracy remained stable after 25 epochs, whereas the training accuracy continued to increase (see figure 7b). As expected, the epoch-25 model has since performed well on dozens of new videos from a variety of facilities which indicates sufficiently general performance. As indicated in tables 1 and 2, the Xception model delivers the highest accuracy by pixel and mean Intersection over Union (mIoU) on the test set, as compared to ResNet18 and VGG16, which were predominantly used during past iterations of arcjetCV [24]. In addition, figure 8 showcases a side-by-side view of the original video frame, the ground truth mask, and the prediction produced by the Xception model after it was trained for 25 epochs.

Table 1 Comparison of Best Pixel Accuracy and mIoU Across Different Models.

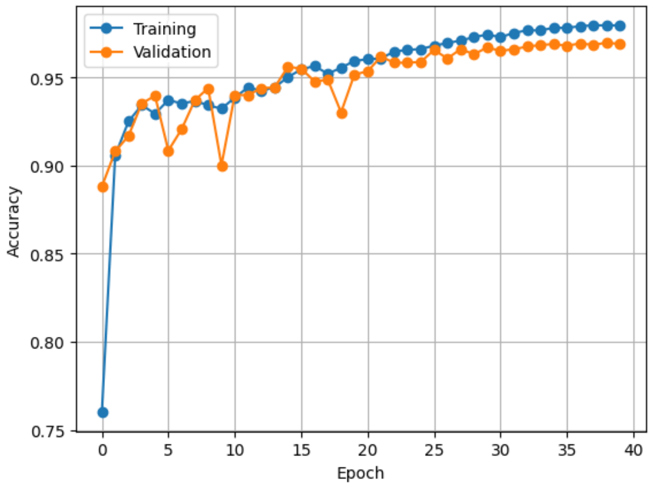
Model	VGG16	ResNet18	Xception
Pixel Accuracy	0.9411	0.9413	0.9687
mIoU	0.7311	0.7543	0.8005

Table 2 Comparison of Pixel Accuracy and mIoU for the Xception Model Across Different Epochs.

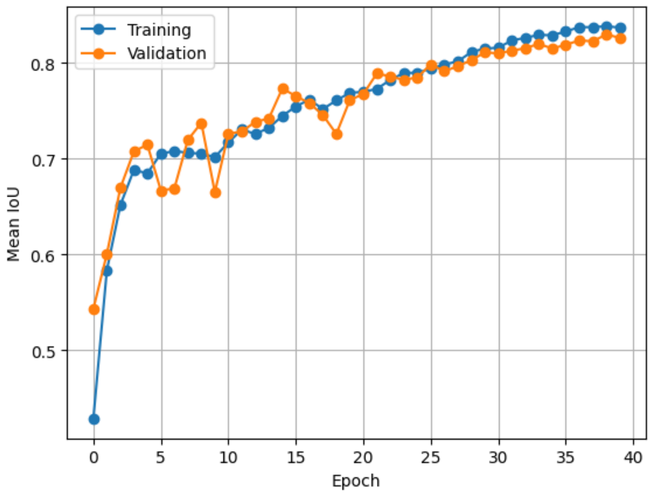
Epochs	Pixel Accuracy	mIoU
15	0.9482	0.7618
20	0.9645	0.7897
25	0.9687	0.8005
30	0.9657	0.7993
40	0.9662	0.8097



(a) Loss curve showing training loss continuing to decrease, while validation loss plateaus after 25 epochs, indicating potential overfitting.



(b) Accuracy curve showing training accuracy rising while validation accuracy stabilizes after 25 epochs.



(c) Mean Intersection over Union (mIoU) providing a quantitative measure of segmentation accuracy, which similarly stabilizes by 25 epochs, confirming this as the optimal training duration.

Fig. 7 Metrics for the Xception model trained up to 40 epochs.

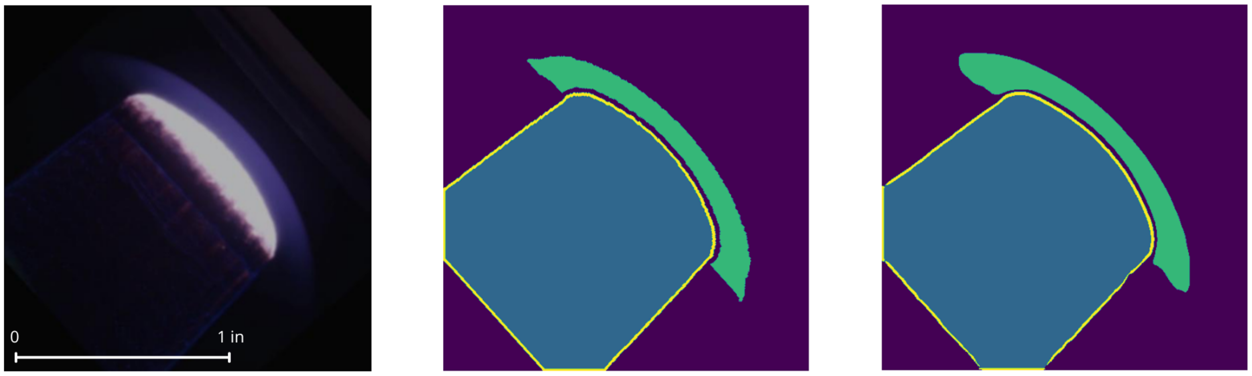


Fig. 8 Challenging video frame (left), ground truth masks (middle), and Xception prediction (right).

C. Local Outlier Factor

Most arcjet videos have one or more frames where the automated edge detection makes a significant error (e.g., a spallation particle disrupts the identified edge). These outliers are removed by utilizing the LOF method from the scikit-learn [25] library. This technique identifies and filters out anomalous frames by assessing their deviation from the local density of their neighbors, effectively pinpointing frames that are outliers due to unusual characteristics. This approach proved necessary to achieve fully automated video processing.

D. Error quantification

As with most measurements, the details and error bars are important. For arcjetCV, there are two primary important sources of error to quantify: projection and edge detection error. The edge detection error is typically small ($< \pm 3$ px) but can be significant if the resolution is low or the video quality is poor. This error can be reduced by zooming in to a smaller region on the sample to increase the resolution. Alignment within a few degrees is typically sufficient for most circumstances as the projection error of a viewing angle θ off from profile scales like:

$$\text{Error fraction} \approx \frac{1}{\cos(\theta)} - 1 \tag{1}$$

However, more severe projection errors can occur if the sample is a flat-face with sharp edges. This is because the leading edge visible in the video changes location from one edge to a point closer to the stagnation point as the edges on these models typically recess faster than the stagnation point.

IV. Enabling New Analysis and Validation Capabilities

The fundamental goal of arcjetCV is to enable higher fidelity analysis and validation for arcjet testing. Unfortunately, the majority of material recession rate data is redacted and not approved for public release. Consequently, with the notable absence of comprehensive material data, this section describes some of the new analyses and validation capabilities that have been enabled by arcjetCV.

A. Non-linear Recession

Although most heatshield materials are thought to undergo linear recession under nominal conditions, all materials exhibit non-linear recession under certain conditions (e.g., non-linear test conditions, testing to failure, multi-layer materials). For these circumstances, acquisition of time-resolved recession is essential for analysis of the test and the material performance.

A typical example where non-linear recession occurs are tests where the sample geometry is not an IsoQ geometry. Due to the non-uniform recession of non-IsoQ geometries, these samples will change shape under test leading to time dependent conditions at the stagnation point and a corresponding change in stagnation recession rate. arcjetCV can measure this time-dependent change in conditions via both the recession rate as well as the shock standoff distance.

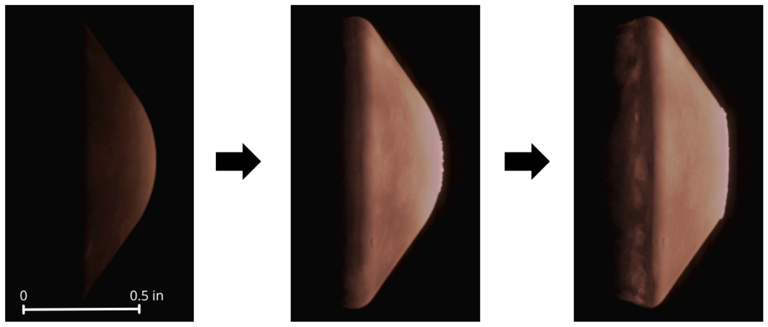
Another example is NASA's Heat shield for Extreme Entry Environment Technology (HEEET) material [26], a 3D woven composite. This material has two distinct layers, a high density ablation layer and a low-density insulating layer that are woven together. Recovering time-resolved recession is necessary to capture the different recession rates of the different layers.

Testing to failure also results in non-linear recession due to runaway effects that occur in edge cases. These include testing of micrometeorite damaged regions, testing of out-of-spec material samples, and intentional over-testing to observe failure modes. In-situ measurements are clearly needed for tests which pass the point of failure as models usually cannot be recovered for post-test analysis.

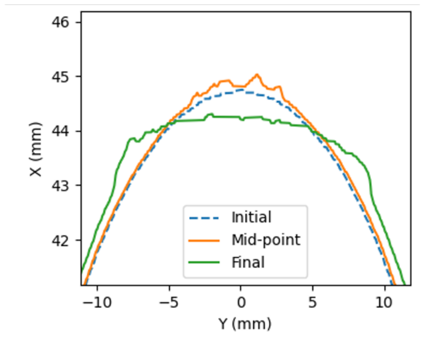
Tests with short dwell times ($<$ 15 sec) or particularly fast recession have measurable differences between the total recession measured by arcjetCV and the total recession measured by pre-test and post-test 3D scans. This is attributed to the finite insertion time for the supporting arm to move into the flow and stabilize (typically 0.3-1 sec). This effect is especially significant for very short dwell times (e.g., 2 sec) where the additional recession occurring during the insertion/extraction is a significant fraction of the total recession. For these cases, it is imperative for material models to use the arcjetCV in-situ recession rate rather than the recession rate inferred by pre/post-test scans assuming a linear recession rate over the dwell duration.

Recent testing of PICA and Avcoat materials also indicate a need for material models which can account for material swelling and shrinkage [27, 28]. These models can only be validated by in-situ tracking during high-temperature tests.

B. Shape Change



(a) Frames from arcjet video.



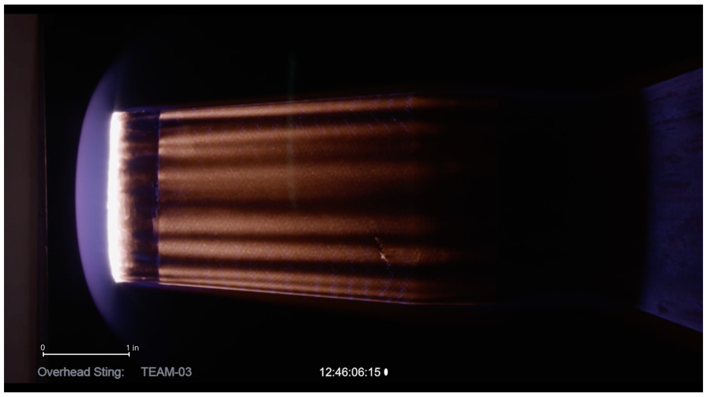
(b) Shape-change over time.

Fig. 9 Non-linear recession with significant shape change for a sample at HyMETS.

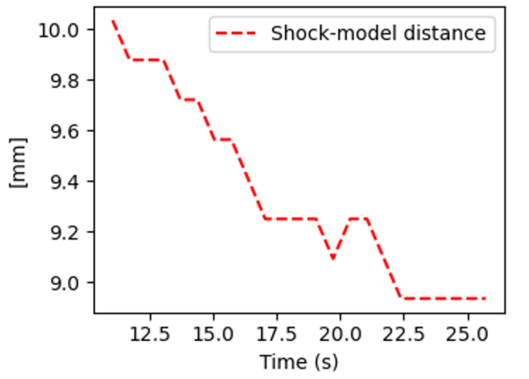
arcjetCV also enables characterization of the changing shape of various samples during tests. As illustrated in figure 9, the plots show the initial and final contours for a sample from a test conducted at NASA Langley Research Center's HyMETS arcjet. The high-fidelity segmentation of the images, achieved with a resolution of 0.1 mm, reveals several notable effects. Firstly, the sample nose experiences flattening, along with the formation of millimeter-sized ripples of a surface coating. Secondly, there's a noticeable overall expansion of the sample, as highlighted by the negative recession at the edges. This example shows several new forms of investigation enabled by arcjetCV: analysis of shape change, tracking melt flows, and sample swelling.

C. Changing Shock Standoff

The shock standoff distance is another valuable form of validation that arcjetCV analysis provides. This distance is a simple validation metric for comparison with computational fluid dynamics (CFD) simulations that are used to interpret the results of arcjet testing as well as scale them to flight. Confirming the standoff distance matches the simulated values is a simple sanity check. More involved analysis that simulates shape change can also be verified by checking the changing shock standoff distance. Furthermore, for in-situ diagnostics including stagnation pressure sensors and calorimeters, confirming the standoff distance is as expected is an excellent metric to detect undesired pressure leaks: calibration models with a smaller standoff than expected will typically have a pressure leak that needs to be repaired.



(a) **Frame from arcjet video with a flat face sample from the IHF.**



(b) **Plot of sample shock-standoff distance over time.**

Fig. 10 Stagnation shock standoff distance over time for an arcjet test with a flat face sample.

V. Conclusion

In this paper, we introduced the open-source software arcjetCV, an application designed to automate analysis of arcjet test video by leveraging machine learning techniques. Initial testing and evaluation has demonstrated good performance in terms of precision, processing speed, and generalization. The application's easy installation, graphical user interface, and streamlined processing pipeline render it accessible to a broad spectrum of users, from technicians to scientists. Its automated segmentation and analysis features significantly save time and resources in video processing enabling extraction of in-situ recession as a standard data product. Furthermore, this new wealth of in-situ time-resolved recession data will be important for future validation of multi-physics simulations of the ablation process. arcjetCV is currently used by the NASA Ames Research Center arcjet facilities in support of several missions, including Mars Sample Return (MSR), Dragonfly, and the Artemis lunar program.

The development of arcjetCV is an ongoing effort. Future work will aim at improving the calibration process, the user interface, as well as 3D reconstructions of the sample's surface through the use of a stereo camera setup.

VI. Acknowledgements

This work was supported by internal research and development (IRAD) funds from the Thermal Protection Materials branch of NASA Ames Research Center. The authors would particularly like to thank Mairead Stackpoole, the Mars Sample Return project, and the arcjet team for providing test videos and for their feedback.

References

[1] Sherrouse, P., and Carver, D., "Demonstrated Real-Time Recession Measurements of Flat Materials During Testing in High-Enthalpy Flows," *Calspan Corporation AEDC Operations, Arnold Air Force Base*, 1992.

[2] Schairer, E. T., and Heineck, J. T., "Photogrammetric Recession Measurement of Ablative Materials During Arc-Jet Testing," *NASA Ames Research Center*, 2012.

[3] Callaway, D. W., Reeder, M. F., and Greendyke, R. B., "Photogrammetric Measurement of Recession Rates of Low-Temperature Ablators in Supersonic Flow," *Air Force Institute of Technology*, 2012.

[4] Loehle, S., Staebler, T., Reimer, T., and Cefalu, A., "Photogrammetric Surface Analysis of Ablation Processes in High Enthalpy Air Plasma Flow," *University of Stuttgart, AIAA*, 2014.

[5] Winter, M., Butler, B. D., Danehy, P. M., and Splinter, S., "Remote Recession Measurements of Wire-Seeded PICA Samples in an Arc-Jet Flow," *University of Kentucky and NASA Langley Research Center*, 2018.

[6] Yang, C., and Koo, J. H., "Review of Sensing Techniques of Ablative Materials," *University of Texas at Austin*, 2020.

[7] Zibitsker, A., Bessire, B., Haskins, J., and Stern, E., "Video processing for evaluation of ablative behavior of meteorite samples tested in the IHF and HyMETS Arc-Jet facilities," *11th Ablation Workshop*, University of Minnesota, USA, 2019.

[8] "Qt for Python - PySide6 Documentation," , 2024. URL `https://doc.qt.io/qtforpython/`, accessed: 2024-12-23.

[9] Hunter, J. D., "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, Vol. 9, No. 3, 2007, pp. 90–95. https://doi.org/10.1109/MCSE.2007.55.

[10] Bradski, G., "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[11] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, Vol. 32, 2019. https://doi.org/10.48550/arXiv.1912.01703.

[12] Paszke, A., and et al., "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *Advances in Neural Information Processing Systems*, 2019.

[13] Chollet, F., "Xception: Deep Learning with Depthwise Separable Convolutions," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258. https://doi.org/10.1109/CVPR.2017.195.

[14] Bradski, G., "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[15] Haw, M. A., MacDonald, M. E., and Colom, S. V., "Big-data Efficient and Automated Science Transfer (BEAST): An open-source software architecture for arc jet data management, modeling, and automation," *AIAA SCITECH 2023 Forum*, 2023, p. 2712.

[16] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "Imagenet: A large-scale hierarchical image database," *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848.

[17] Chollet, F., "Xception: Deep Learning with Depthwise Separable Convolutions," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807. https://doi.org/10.1109/CVPR.2017.195.

[18] Iakubovskii, P., “Segmentation Models Pytorch,” https://github.com/qubvel/segmentation_models.pytorch, 2019.

[19] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., et al., “Segment anything,” *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026. https://doi.org/10.48550/arXiv.2304.02643.

[20] Semeraro, F., and Quintart, A., “A simple GUI for the Segment Anything Model,” , 2023. URL https://github.com/fsemerar/segment-anything-gui.

[21] Semeraro, F., Quintart, A., Izquierdo, S. F., and Ferguson, J. C., “TomoSAM: a 3D Slicer extension using SAM for tomography segmentation,” *arXiv preprint arXiv:2306.08609*, 2023. https://doi.org/10.48550/arXiv.2306.08609.

[22] Rother, C., Kolmogorov, V., and Blake, A., “"GrabCut": interactive foreground extraction using iterated graph cuts,” *ACM Trans. Graph.*, Vol. 23, No. 3, 2004, p. 309–314. https://doi.org/10.1145/1015706.1015720.

[23] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., and Kalinin, A. A., “Albumentations: Fast and Flexible Image Augmentations,” *Information*, Vol. 11, No. 2, 2020. https://doi.org/10.3390/info11020125, URL https://www.mdpi.com/2078-2489/11/2/125.

[24] Quintart, A., and Haw, M. A., “ArcjetCV: a new machine learning application for extracting time-resolved recession measurements from arc jet test videos,” *AIAA SCITECH 2023 Forum*, 2023, p. 1912. https://doi.org/10.2514/6.2023-1912.

[25] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E., “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830. https://doi.org/10.48550/arXiv.1201.0490.

[26] Tavares, F., Buckner, D., Burton, D., McKaig, J., Prem, P., Ravanis, E., Trevino, N., Venkatesan, A., Vance, S. D., Vidaurri, M., et al., “Ethical exploration and the role of planetary protection in disrupting colonial practices,” *arXiv preprint arXiv:2010.08344*, 2020. https://doi.org/10.48550/arXiv.2010.08344.

[27] Ferguson, J. C., Semeraro, F., Thornton, J. M., Panerai, F., Borner, A., and Mansour, N. N., “Update 3.0 to “PuMA: The porous microstructure analysis software”,(PII: S2352711018300281),” *SoftwareX*, Vol. 15, 2021, p. 100775. https://doi.org/10.1016/j.softx.2021.100775.

[28] Lopes, P. C., Vianna, R. S., Sapucaia, V. W., Semeraro, F., Leiderman, R., and Pereira, A. M., “Simulation toolkit for digital material characterization of large image-based microstructures,” *Computational Materials Science*, Vol. 219, 2023, p. 112021. https://doi.org/10.1016/j.commatsci.2023.112021.