GPJet: Physics-Informed Machine Learning of E-jet Printing Dynamics

Athanasios Oikonomou1,6, Theodoros Loutas1, Dixia Fan2,3, Alysia Garmulewicz, George Nounesis and Filippos Tourlomousis4,5,6\*

1Mechanical Engineering, University of Patras, Patras, Greece

2Mechanical and Material Engineering, Queen's University, Kingston, ON, Canada K7L 3N6

3Ingenuity Labs, Queen's University, Kingston, ON, Canada K7L 3N6 3

4The Center for Bits and Atoms, Massachusetts Institute of Technology, Cambridge, MA, USA

5Biomolecular Physics Laboratory, Institute of Nuclear & Radiological Sciences and Technology, Energy & Safety, NCSR Demokritos, Athens, Greece

6SuperLabs, Athens, Greece

Abstract

We present GPJet, a Bayesian learning framework that is capable of …

(revolutionizing Melt Electro-Writing (MEW), an additive manufacturing technology with wide range of applications in biomanufacturing, energy storage technology etc. One of the greatest challenges is accurate real time metrology of features characterizing the jet, and therefore the process. Another is the difficulty in control and optimization of the process due to its multiparametric nature. GPJet addresses both of these issues. It consists of a Computer Vision module capable of performing real time object detection and metrology, and a Bayesian Optimization module to automatically minimize the jet's lag distance by altering collector's speed. Most notably, Bayesian Optimization can contribute in both process and computational optimization. Implementation and adaptation of Bayesian Optimization and machine learning in general, into additive manufacturing techniques, could lead to more efficient printing, by enabling better informed, data-driven parameter tuning.)

# Introduction

The programmable assembly of functional inks in two- and three-dimensions dimensions using computer numerically controlled (CNC) machines coupled with printing technologies has revolutionized the way we design and fabricate physical objects. Scientists, engineers, designers and makers have at their disposal a wide palette of extrudable inks to process. This toolset allows them to provide material solutions with cellular architectures and feature resolutions spanning nm to cm scale1. With these attributes at hand, extrusion-based additive manufacturing (AM) technologies, often referred as direct ink writing or 3D printing, are transforming fields such as healthcare, robotics, electronics and sustainability 2,3.

While the potential of 3D printing is celebrated very often in scientific journals and in the media, there is a “secret” that practitioners and companies of 3D printing, do not stress out. This under-reported reality entails the extensive experimentation and manual labor required to achieve expected end-part properties and reproducible process outcomes. Every time a new ink needs to be processed or a machine is moved to another location with different ambient conditions, trial and error approaches are being followed to optimize the printing process. These practices have led to desktop and industrial machines that are rigid and never “learn”. Only the user learns, leading to the creation of experienced "super users" at the expense of an enormous degree of individual process engineering.



**Figure 1: Electrohydrodynamics Jet Printing Process.** Solution electrospinning (SES) vs. melt electrospinning (MES). The main differentiating feature between the two processes is the extent of the jet instabilities that arise from the electrostatic forces acting at the polymer jet-air interface. For MES, the chaotic jet regime is limited close to the grounded collector plate due to the high viscosity and dielectric properties of the pure polymer melt. b Direct melt electrowriting (MEW) and its operating principle.



**Figure 2: The GPJet Pipeline Framework.** A Physics-informed Bayesian Machine Learning framework comprised by three different modules: a) the Machine Vision module, which takes as an input timeseries video focusing on the polymer jet in the free flow regime and performs extraction of high-fidelity jet features in real-time based on an automated image processing workflow b) the Physics-Based Modeling Module, which and c) the Machine Learning module, which takes as an input high fidelity experimental data from the Machine Vison module and low fidelity modeling data from the and performs a series of data-driven tasks to learn the jet dynamics.

**E-jet printing**. A notable example of such a family of extrusion- AM technologies are the electrohydrodynamics-based additive manufacturing technologies, also known as E-jet printing.

……

**Data-Driven Approaches in Additive Manufacturing.**

…….

**Proposed Solution.**

…….

Overall, the main contributions of this paper are as follows:

# GPJet: The Physics-Informed Machine Learning Pipeline

…….

# Results

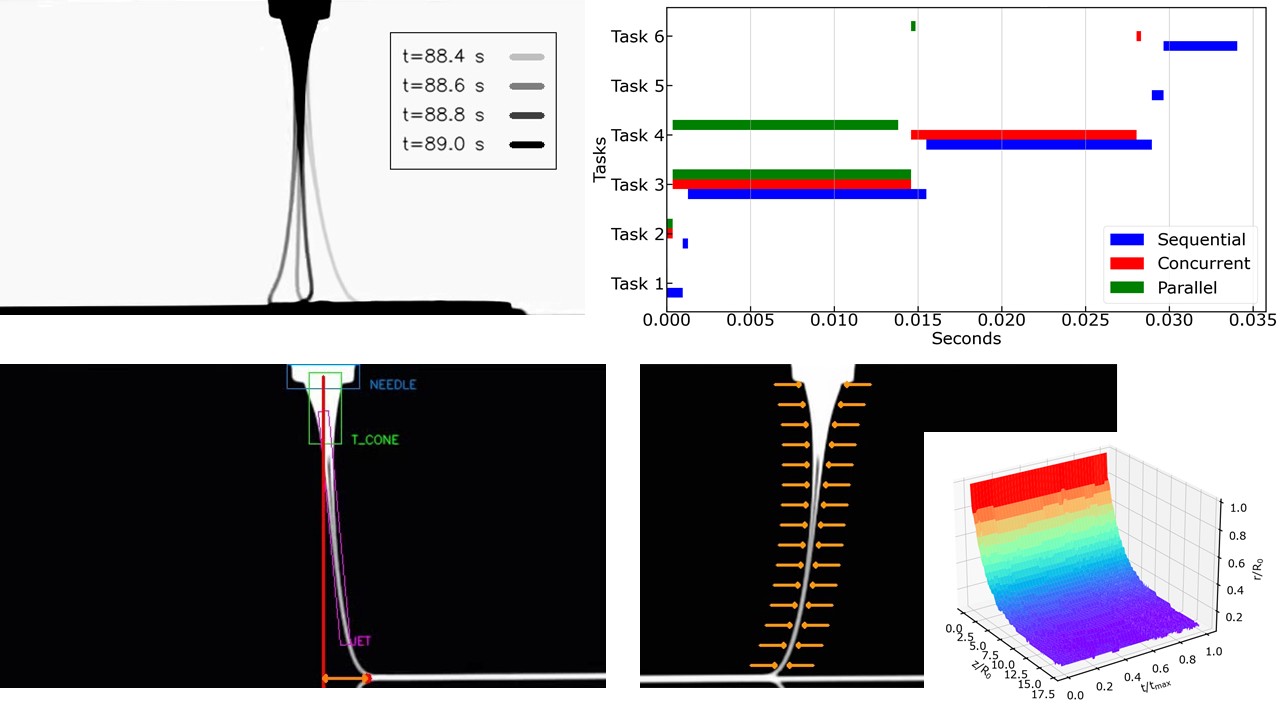
To demonstrate the ability of data-driven self-calibrating EHD Jet printing processes, we first developed a computer vision workflow, hereafter denoted as the Machine Vision module. The Machine Vision module allows us to probe the jet dynamics and is validated by a previously published dataset of time series video data [ADD CITATION]. The dataset is acquired by a camera with 50 fps and a field of view spanning the area between the needle tip and the grounded collector of a melt electrowriting system. MEW constitutes an ideal testbed for demonstrating the capabilities and the flexibility of our GPJet framework. The highly dynamic nature of the process and the multiple user-controlled independent process parameters, pose several challenges. These challenges are reflected to the following fundamental question that we tried to address during the development of GPJet:

Could we come up with a data-driven framework that allows both real time process monitoring and metrology with conventional low-cost cameras? If yes, such a framework would give us the ability to actively learn the dynamics of the process in an online self-calibrating scenario. IN such a scenario the optimization goal we set is two-fold. What is fiber diameter & lag distance for direct writing mode?

## 3.1 Learning Jet Dynamics from Videos

To tackle the challenge of real time monitoring and feature extraction of the jet, we developed a machine Vision module. The key characteristic of the module is the multi-threading and multiprocessing capabilities that have been implanted to achieve real time processing of videos acquired by camera with 50 frames per second (fps). Yet the challenge remains, to perform all these functions in real time. But what is defined as real time? In this case, as real time is defined the capability to run the whole algorithm in times less than 0.02 second, which corresponds to the time between the capture of two consecutive frames.

This is accomplished by dividing the computer vision workflow in specific algorithmic tasks, starting with a sequential code implementation of the workflow, profiling the code, identifying which tasks are the most computationally expensive and then gradually parallelizing to reduce computational processing time. This led to 3 different code implementations of the machine vision module: a) the sequential, b) the concurrent and the c) parallel, with the last one achieving real-time performance. The results are shown in Figure X-b, where all the tasks are plotted along with their respective processing time for the 3 different code implementations. Specifically, the machine vision tasks per frame are the following: Task 1: Read new frame, Task 2: Reverse image, Task 3: Edge-based feature extraction and storage, Task 4: Object-based feature extraction and storage, Task 5: Show processed frame, Task 6: Save video. Profiling the sequential code version reveals that an average time of 0.033 sec. is needed to perform the whole machine vision workflow per frame with the most expensive one being task 3. The first alternative is multithreading strategy (concurrent version). Multithreading is implementing software to perform two or more tasks in a concurrent manner within the same application. Multithreading is spawning multiple threads to perform each task. If thread 1 is busy waiting for I/O to complete, thread 2 uses the processor during this time and then switches back to thread 1 to complete. There is no limitation in the number of threads that can be used [10].



**Figure 3**: bla bla

From the profiling experiments, we learned that multithreading on one hand can reduce processing time of I/O bound tasks almost to zero, but on the other hand does not improve processing time of CPU bound tasks. In our case, CPU bound tasks, such as Task 3 and Task 4, are the most computationally expensive. This challenge was overcome by employing a multiprocessing strategy (parallel version). Multiprocessing systems have multiple processors running at the same time. Therefore, different tasks of an application can be run in different processors in a parallel manner. This capability considerably accelerates program performance. The limitation in this alternative is that the number of processes that can be employed must be less or equal with the number of processors (CPU cores) of the device [10]. Finally, by employing multithreading for I/O bound tasks, such as reading, showing, writing video frames, and multiprocessing for CPU bound tasks, which are the Features extraction and Object segmentation and detection algorithms, processing times up to 0.014 sec can be achieved.

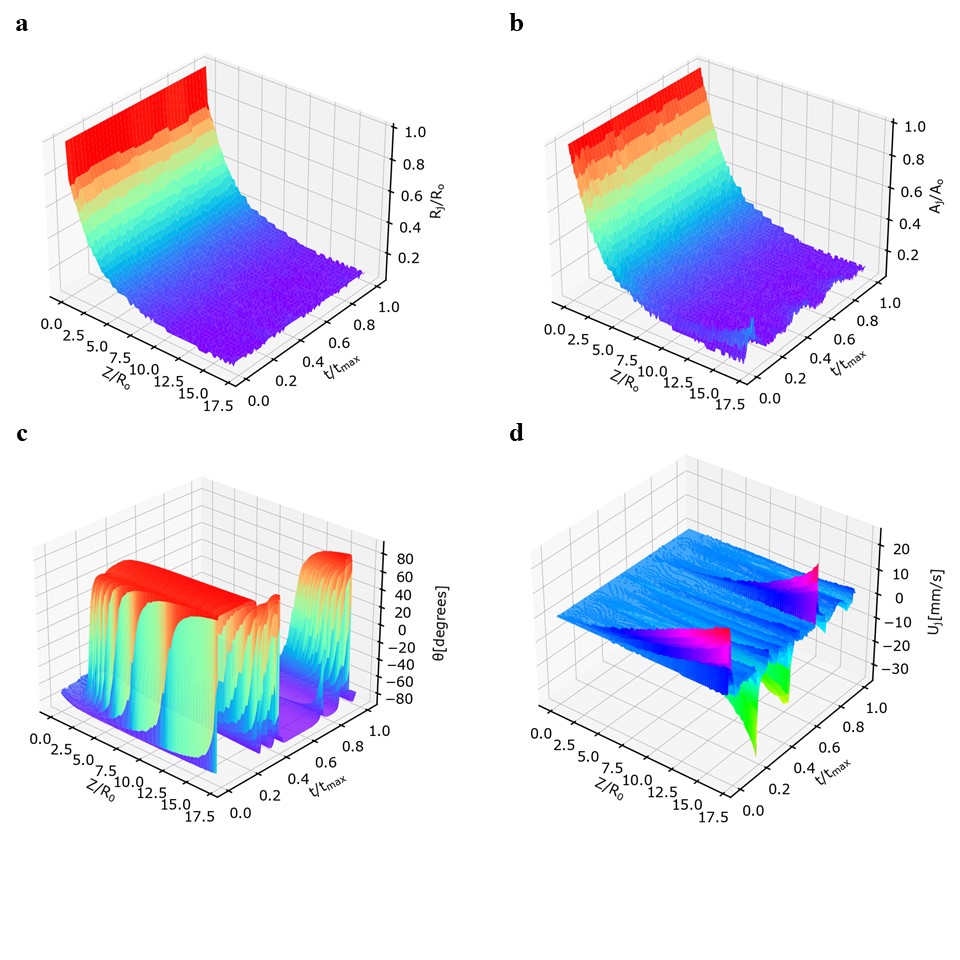
Having the capability to quantify process dynamics-relevant metrics in real time with the machine vision module, enables to set up the basis for mapping the relationship between process dynamics represented by the extracted features of interest and the user-controlled, tunable independent process parameters.

The optimization goal is dual from []: to achieve steady Taylor cone jet in the free flow regime and minimum lag distance to avhieve direct writing. It is important to mention that for the community of MEW it took almost a decade to understand that through trial & error and … (look my job package) Edisonian approach. With such a tool in ourt hands and advancement of data-driven approaches, we can have self calivbrating printers that could learn the mappin between this and that to achivge the previously mentioned objective. IN csuch a self-calivbrating scenario you would have 2 scanarions: a) off learning and b) online leaerning. In both case, the user would need minimum amount of data to minimize computational cost.

To achieve , we use GPs and we describe noe now Figure 4. The 1st question that we aim to address is to what is the minimum amount of observations d needed in order to learn and predict the jet profile radius and learn the mapping between the jet and the speed ratio. Talk more about PSO

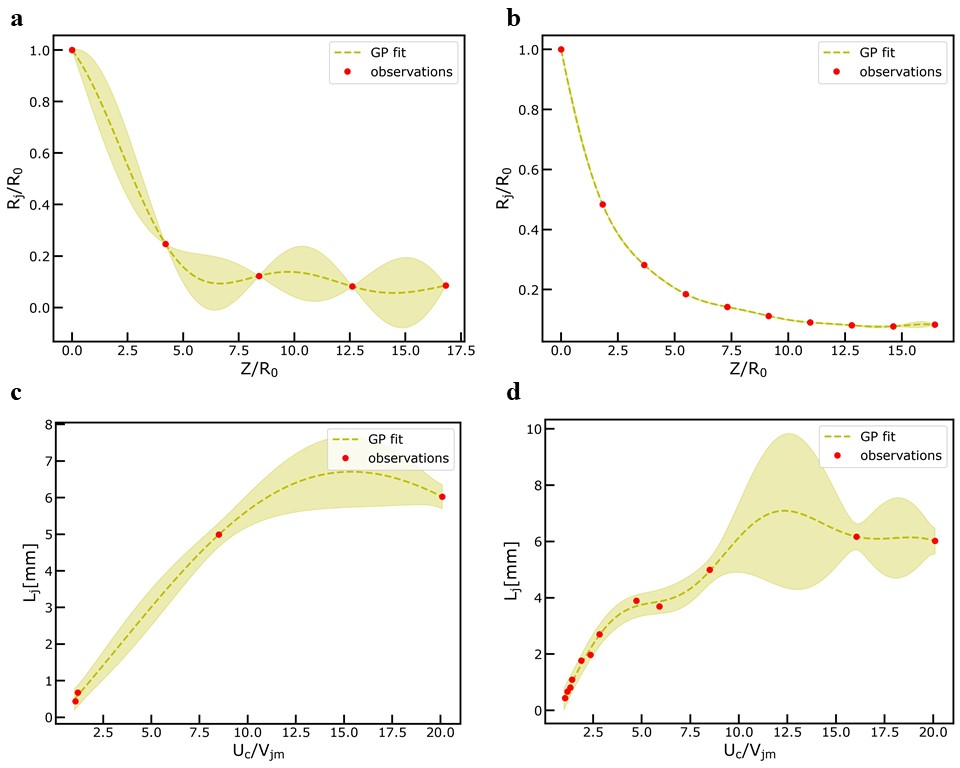
The extracted features of interest by the machine vision module are spatio-temporal data and are presented in Figure 3. : a) the jet diameter, b) the jet area, c) the jet angle, d) the jet velocity in the x-axis and e) the jet lag. These are domain-relevant features with physical significance related to the process physics/.

...



**Figure 3: Features Extracted from Computer Vision Module**. **a**) Normalized jet radius () obtained from the computer vision metrology module of the GPJet framework plotted against the normalized jet length () and the normalized time (). **b**) Normalized jet area () obtained from the computer vision metrology module of the GPJet framework plotted against the normalized jet length () and the normalized time (). **c**) Jet angles () obtained from the computer vision metrology module of the GPJet framework plotted against the normalized jet length () and the normalized time (). **d**) Jet velocities () obtained from the computer vision metrology module of the GPJet framework plotted against the normalized jet length () and the normalized time ().

**Figure 4**: **Results of Gaussian Process Modeling Regression Tasks**. **a**) fitting normalized () jet radius observation data (n=5) obtained from the computer vision metrology module of the GPJet framework at specific z axis coordinates along the normalized jet length (). **b**)fitting normalized jet radius using a higher number of observation data (n=10) compared to the previous case (a). **c**)fitting lag distance () observation data (n=3) obtained from the computer vision metrology module of the GPJet framework for specific speed ratios (). **d**) fitting lag distance using all available observation data (n=12). For non-normalized quantities units are in SI. Filled contours represent uncertainty bounds (95% confidence intervals (CIs)).

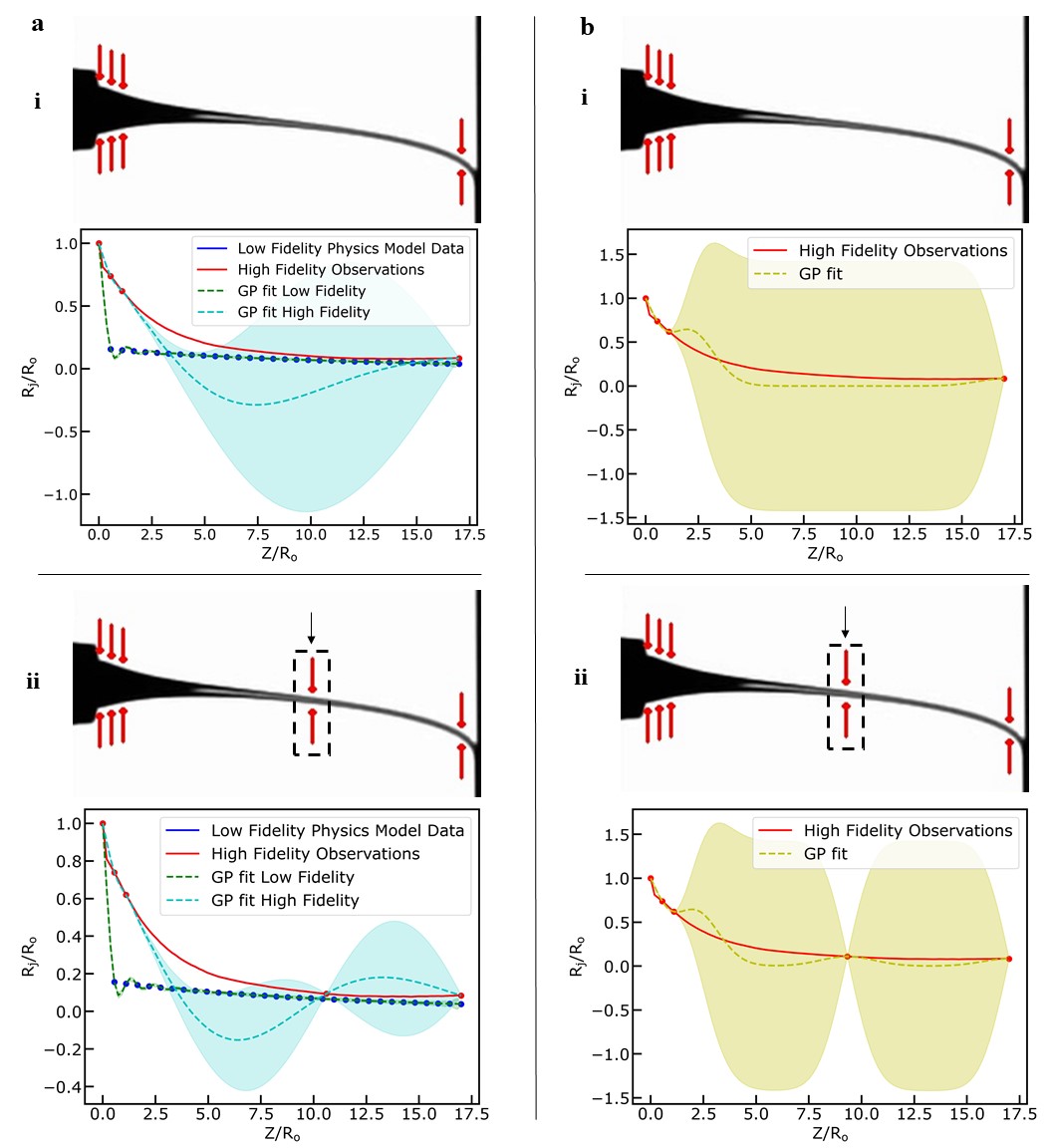


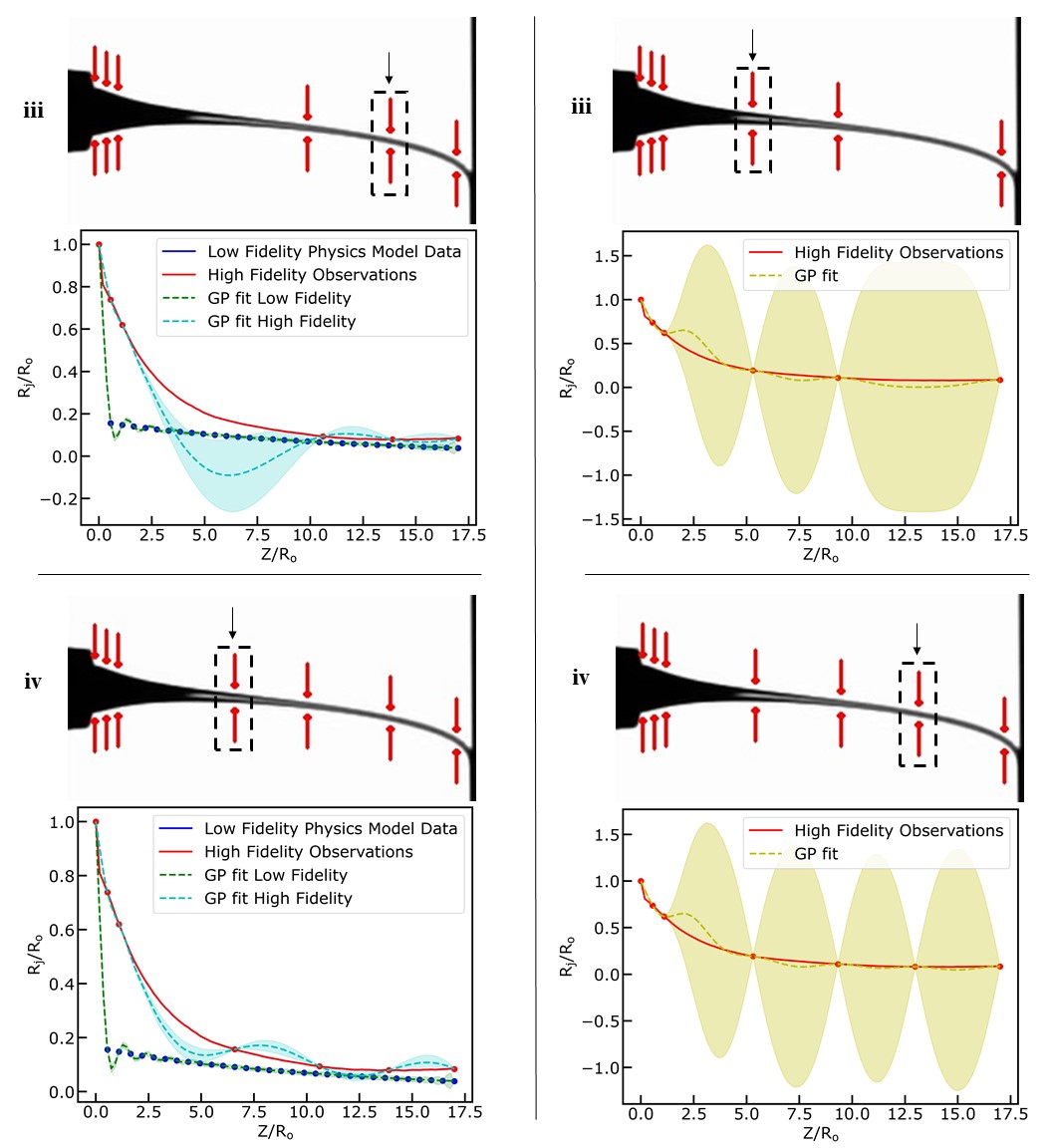
## 3.2 Learning Jet Dynamics from Videos & Physics

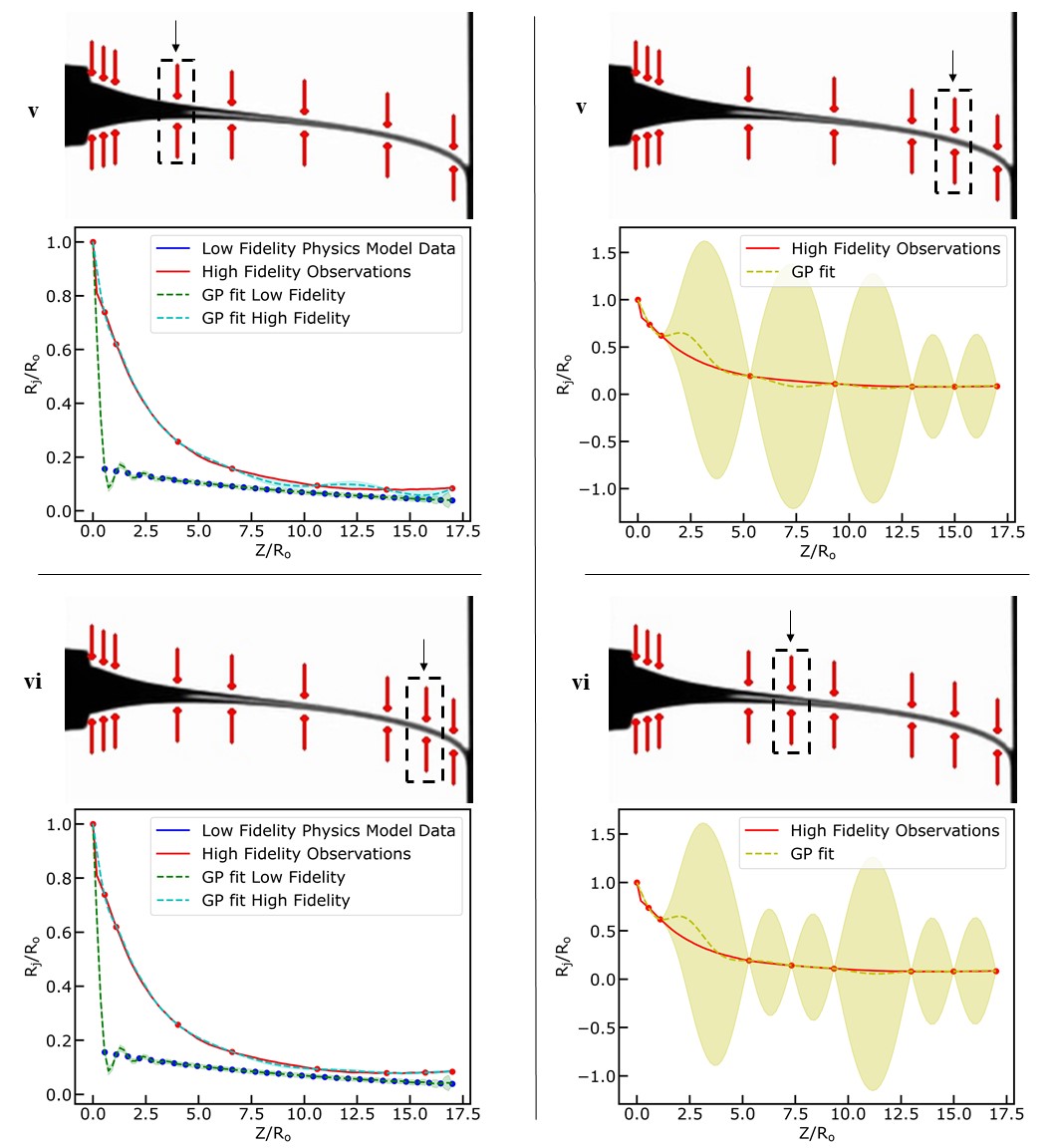


**Figure 5**: **Results of** **Multifidelity Modeling Regression Tasks. a**) fitting normalized high fidelity observation data (n=6, red color) of jet radius () and low fidelity model data obtained from the computer vision metrology module of the GPJet framework and from the multi-physics model, respectively, at specific z axis coordinates along the normalized jet length () and comparing the results with a simple GP fit using the same number of high fidelity observation data. **b**) fitting a higher number of normalized high fidelity observation data (n=7, red color) of jet radius () and low fidelity model data obtained from the computer vision metrology module of the GPJet framework and from the multi-physics model, respectively, at specific z axis coordinates along the normalized jet length () and comparing the results with a simple GP fit using the same number of high fidelity observation data.

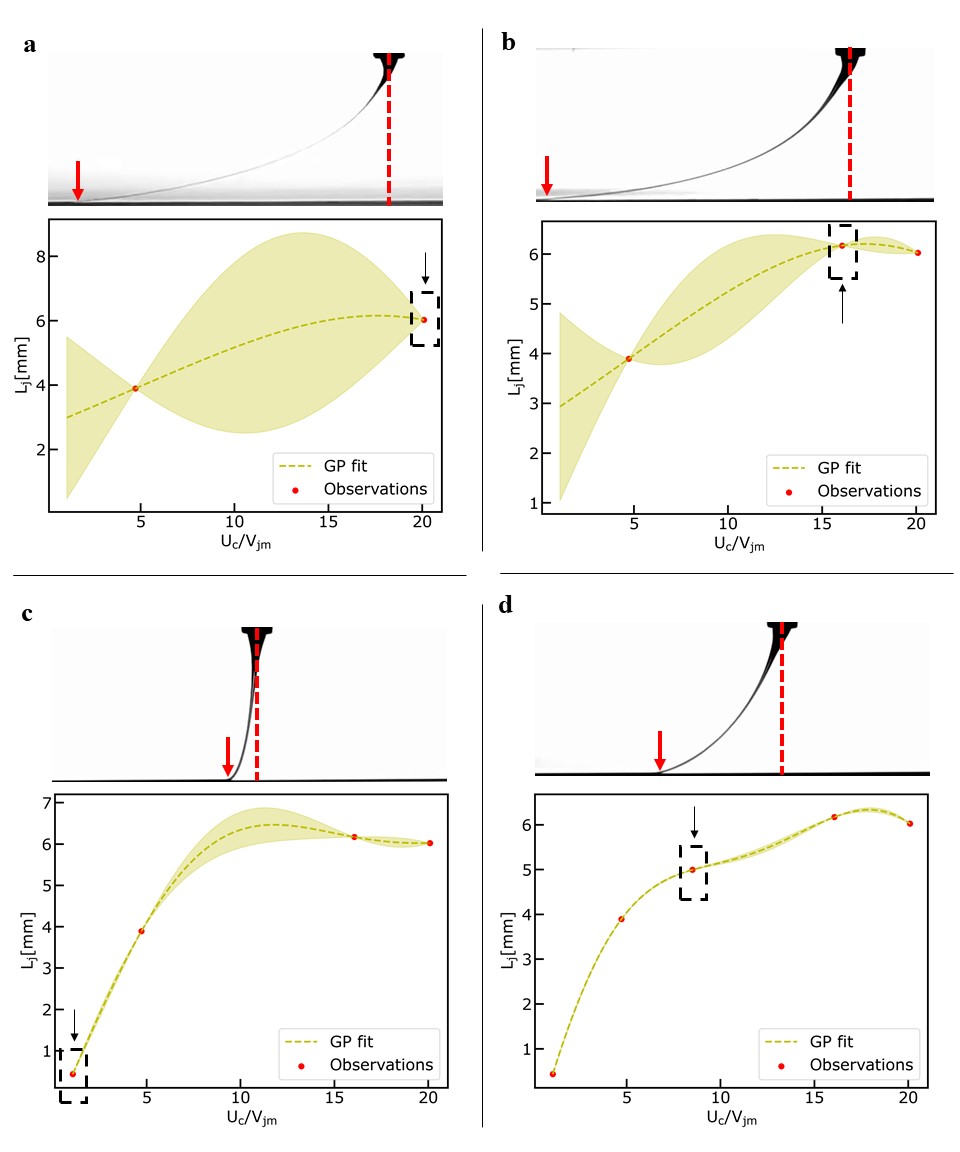
## 3.3 Active Learning of Jet Dynamics



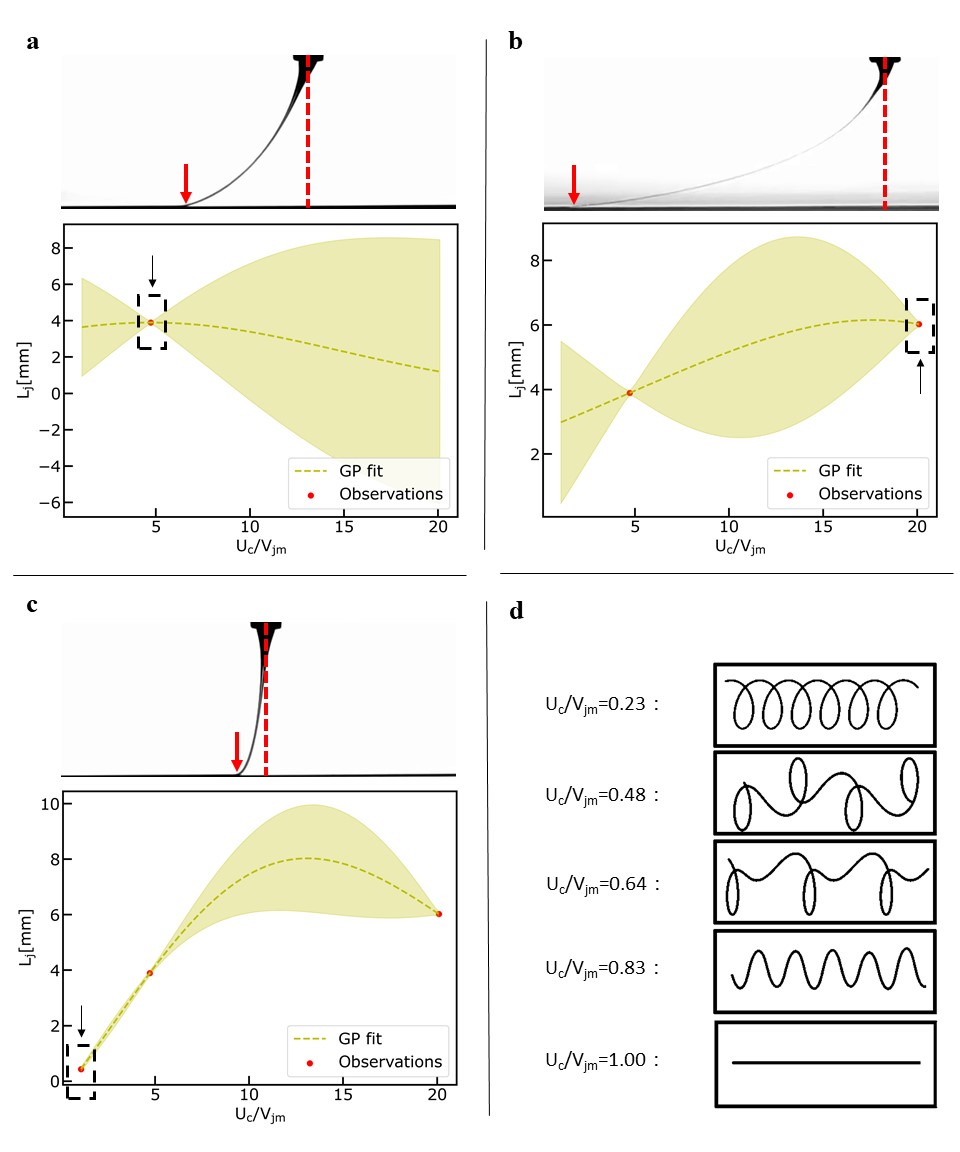




**Figure 6: Results of Active Learning on Multifidelity Modeling Task. a**) exploring the design space using Active Learning to fit a Multifidelity Gaussian Process to normalized high fidelity observation data (red color) of jet radius () and low fidelity model data obtained from the computer vision metrology module of the GPJet framework and from the multi-physics model, respectively, at specific z axis coordinates along the normalized jet length () ( **i – vi** denote the iterations of the active learning algorithm until it meets termination criteria). **b**) exploring the design space using Active Learning to fit a Gaussian Process to normalized high fidelity observation data (red color) of jet radius () obtained from the computer vision metrology module of the GPJet framework at specific z axis coordinates along the normalized jet length () ( **i – vi** denote the iterations of the active learning algorithm until it meets termination criteria).



**Figure 7: Results of Exploring the Design Space Task.** Exploring the design space using active learning tofit a Gaussian Process Model to lag distance () observation data obtained from the computer vision metrology module of the GPJet framework for specific speed ratios (). **a-d)** Iterations of the active learning algorithm until it meets termination criteria.



**Figure 8: Results of Bayesian Optimization Task.** Performing Bayesian Optimization to find the minimum lag-distance () by fitting a Gaussian Process Model to lag distance () observation data obtained from the computer vision metrology module of the GPJet framework for specific speed ratios (). **a-c)** Iterations of the Bayesian optimization algorithm until it meets termination criteria. **d)** For speed ratios less than one ( the process is unstable, no straight line is formed, instead the translated coiling, alternating loops, W patterns and meanders patterns are formed, therefore no lag distance () observation data can be obtained from the computer vision metrology module of the GPJet framework.

...

# Conclusions

...

# Methods

## 2.1 Dataset

Video S1 and Video S2 published by P. Dalton [5] were chosen as the dataset to be used for this paper. As described a Sony Alpha 7 (Sony Corp. Japan) digital camera was used with a Nikon ED 200 mm lens (Nikon Corp. Japan). 1080 p resolution videos of the nozzle, jet and collector were taken at 50 frames per second. Process hyperparameters were set to 8 m s-2 and 500 m s-3 maximal stage acceleration and jerk, a 22G nozzle was used, polymer temperature was set to 87 o C and the voltage to the collector was set to -1.5kV, while the voltage to the nozzle was set to +5.75kV.

For Video S1 air pressure feeding the nozzle was set to 1.2 bar and the distance between nozzle and collector was set to 3.5mm with a standard deviation 0.1mm. Collector’s speeds tested in Video S1 were 191.25 mm s-1, 212.5 mm s-1, 255 mm s-1, 340 mm s-1, 510 mm s-1, 850 mm s-1, 1530 mm s-1 and 2890 mm s-1.

For Video S2 air pressure feeding the nozzle was set to 2.4 bar and the distance between nozzle and collector was set to 4.5mm with a standard deviation 0.1mm. Collector's speeds tested in Video S2 were 292.5 mm s-1, 520 mm s-1, 1300 mm s-1 and 4420 mm s-1.

**Video Preprocessing**

Videos were split according to their collector speed and video frames were cropped, in order to get rid of unnecessary pixels, which lead to increased processing time. For real time video processing the user would need to specify the region of interest in the frame, so that we can crop it and dispose of needless information, as well as the position of the nozzle, the collector, and a factor, which represents the length of the Taylor cone depended on the nozzle's diameter.

## 2.2 Jet Metrology

For the implementation of the Jet Metrology algorithm Python 3.8 was used, along with extensive usage of the OpenCV library, which enables as to read and process video data. Jet Metrology algorithm consists of two sub-algorithms. The first is the Object Segmentation and Detection algorithm. The second is the Features Extraction algorithm.

The first sub-algorithm segments the MEW printing Needle, the Taylor Cone, the Jet and the Deposited Material on the collector. Also attempts to find the jet's deposition point on the collector. Finally, it draws some graphics, so that the user can assess its performance. In order to detect the 'objects' of interest in each video frame we use the very much alike meanshift and camshift algorithms. The meanshift algorithm is actually a statistical concept related to clustering. Like many other clustering algorithms, it attempts to check the whole frame for high concentration of pixels of the same color. The algorithm and its application in computer vision are widely described in [6]. The main difference between the two algorithms is that the camshift algorithm has the capability to adjust, so that the tracking box can change its size and direction, to better correlate to the movements of the tracked object.

The meanshift and camshift algorithms are useful tools to employ for object tracking. Also, unlike neural networks and other machine learning methods of object detection, these algorithms can be immediately implemented and deployed without the need to train a model with numerous labeled images. Instead, the algorithm takes as an input the initial color of the object, that needs to be detected, and then it tracks it throughout the rest of the video. On the other hand, using color as a primary method of identification, neither of the two algorithms is able to identify objects based on particular shapes and features, which makes them significantly less powerful than other methods. Furthermore, objects varying in color on a large scale and complex or noisy backgrounds can make object detection and tracking problematic. So, it's obvious that meanshift and camshift algorithms work best under controlled environments.

So, it's in our interest the fact that the videos are captured in grayscale color-space, and that the nozzle as well as the collector do not move vertically, just horizontally. The first step is to reverse the image so that the objects of interest are white and the background black. The next step is to apply a multi color mask to segment them, and then to change the image color-space from BGR to HSV. Finally, the meanshift algorithm is applied to detect the needle and the Taylor cone, since there is barely any significant movement to them, and the camshift algorithm to detect and track the jet.

In order to find the deposition point the algorithm needs to know the collector's position. Then it creates a window around the collector, crops the region of interest from the frame and processes that instead of the whole frame. The build-in function used to find the deposition point is the *cv2.goodFeaturesToTrack*. This function finds the most prominent corner in our region of interest by calculating eigen-values, as described in [9].

Finally, by subtracting the deposition point from the nozzle's position (center of blue rectangle in Figure 2c), we get the lag distance, which is depicted with a two-ways orange arrow in Figure 2c.

The second sub-algorithm is the one responsible for extracting all those features of the jet, that are important to gain some insight to the physics of the process. Such features are the diameter, areas and angles of the jet as we move along the z-axis. Another important feature is the velocity of each jet's point along the x-axis relatively to the nozzle's position. To get all those features we follow a pretty much straightforward procedure. The algorithm takes three inputs, the first is the current video frame. The second input is the calibration factor (), which is a correlation between distance units (mm) and pixels. The last one is the stride. The stride actually indicates every how many pixels along the z-axis we perform computations. Using a too small stride would lead to more precise calculations but would tremendously rise the computation time. On the other hand, using a too large stride would lead to shorter computation times but there is danger to lose important information.

The first step is to change frame's color-space from BGR to grayscale, so that the Canny edge detection algorithm can be applied. The parameters of the Canny edge detector are [threshold\_1, threshold\_2] and were specified in a semiautomatic way, using trackbars while performing edge detection to other video samples. After this procedure they were set to 150 and 255 respectively. After performing Canny edge detection, we read the first row of pixels in our canny-frame, which now is an array of 0 and 255. If Canny algorithm has been performed correctly when we read this row of pixels from left to right, the first time we encounter a 255 should be the left edge () of our jet. Likewise, the first time we encounter a 255, while reading the row of pixels from right to the left, should be the right edge () of our jet. By subtracting those two pixels' indexes and multiplying with the calibration factor we get the diameter of the jet at this position in the z-axis, which is equal to .

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

Those indexes are also stored in two variables () so that they can be used to calculate the jet angles as we move down the z-axis.

Then we repeat the procedure for every 'stride' rows. After finding the left () and right () edges and calculating the diameter, the area and angles can be calculated as:

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |
|  | (5) |

The are then updated with the values.

After accessing all frame's rows, the algorithm returns arrays containing all Diameters, Areas, Right Boundaries, Angles left and Angles right. The same procedure is applied to all frames. Right Boundaries are important because by subtracting the right edges of two consecutive frames we can calculate the jet's velocity on the x-axis.

**Real Time Video Processing**

The algorithm works perfectly providing us with all those results and visual representation to assess the whole process. Yet the challenge remains, to perform all these functions in real time. But what is defined as real time? In this case, as real time is defined the capability to run the whole algorithm in times less than 0.02 second, which corresponds to the time between the capture of two consecutive frames.

First step is to divide the whole process into Tasks and start profiling, to figure out which tasks are more time consuming, and then address the problem, if any. In Figure 2b, Task 1: Read new frame, Task 2: Reverse image, Task 3: Features extraction and store results, Task 4: Object segmentation and detection, Task 5: Show processed frame, Task 6: Save video.

As depicted in Figure 2b, with sequential programming, more than 0.033 sec. are needed to process each frame. The first alternative is multithreading (concurrent programming). Multithreading is implementing software to perform two or more tasks in a concurrent manner within the same application. Multithreading is spawning multiple threads to perform each task. If thread 1 is busy waiting for I/O to complete, thread 2 uses the processor during this time and then switches back to thread 1 to complete. There is no limitation in the number of threads that can be used [10]. Turns out that multithreading reduces I/O bound tasks almost to zero, but does not help with CPU bound tasks, which in this case are the most time consuming. The second alternative is multiprocessing (parallel programming). Multiprocessing systems have multiple processors running at the same time. Therefore, different tasks of an application can be run in different processors in a parallel manner. This capability considerably accelerates program performance. The limitation in this alternative is that the number of processes that can be employed must be less or equal with the number of processors (CPU cores) of the device [10]. Finally, by employing multithreading for I/O bound tasks, such as reading, showing, writing video frames, and multiprocessing for CPU bound tasks, which are the Features extraction and Object segmentation and detection algorithms, processing times up to 0.014 sec can be achieved.

## 2.3 Multiphysics Model

The importance of accurately extracting jet properties is signified by the numerous studies on predicting the jet stable region diameter, through mathematical modeling. Among those is the model proposed by Zhmayev et al. [11], which is developed by fully coupling the conservation of mass, momentum, charge and energy equations with a constitutive model and the electric field equations at the steady state. Same as most models, this one also utilizes the thin filament approximation to obtain a simpler and more tractable solution. This assumption is possible by appropriately averaging the model variables across the radial direction. In addition, the charge and electric field equations are simplified, under the assumption of low electrical conductivity, as compared to the governing equations for isothermal simulations presented by Carroll and Joo [12].

Also, the conservation of energy relation and a non-isothermal constitutive model were added to extend to non-isothermal situations. The resulting governing equations after being nondimensionalized are:

|  |  |  |
| --- | --- | --- |
| Continuity: |  | (6) |
| Momentum: |  | (7) |
| Charge: |  | (8) |
| Electric field: |  | (9) |
| Energy: |  | (10) |
| Constitutive: |  | (11) |

The system of equations can be reduced to a set of five coupled first order ordinary differential equations (ODEs). Boundary Conditions are required, in order to proceed to numerical analysis.

|  |  |
| --- | --- |
|  | (12) |
|  | (13) |
|  | (14) |
|  | (15) |
|  | (16) |

|  |  |
| --- | --- |
| Property | Value |
| Zero-shear-rate (at ) () |  |
| Relaxation time (at ) () |  |
| Activation energy of flow () |  |
| Density () |  |
| Heat capacity () |  |
| Thermal conductivity () |  |
| Electrical conductivity () |  |
| Surface tension () |  |
| Ratio of solvent to zero-shear-rate viscosity () |  |
| Mobility factor () |  |
| Dielectric constant ratio () |  |

|  |  |
| --- | --- |
| Parameter | Value |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

The model was implemented in Python. While true properties and parameters of the material are not provided the ones used in [13] for PCL were used. As referred in [12], [13] the model slightly underpredicts the jet radius while in the Taylor cone area, but when the jet is stabilized, it accurately predicts it's radius. Knowing this, even if the volumetric flowrate () is not provided with the dataset, a Particle Swarm Optimization (PSO) algorithm was also implemented in order to find the for which the predicted jet's radius better fits the computer vision observations.

## 2.4 Geometrical Model

Lag distance is a highly important parameter regarding the quality of the process outcome. Specifically, for some collector speeds, the jet falls onto the moving collector in a way reminiscent of a sewing machine, generating a rich variety of periodic patterns, such as meanders, W patterns, alternating loops and translated coiling. P. T. Brun et al. [14] proposed a quasistatic geometrical model, consisting of three coupled ordinary differential equations for the radial deflection, the orientation and the curvature of the path of the jet's contact point with the collector, capable of reconstructing the patterns observed experimentally while successfully calculated the bifurcation threshold of different patterns. It was also discovered that the jet/collector velocity ratio () was the key factor for pattern variation.

According to this geometrical model, the deposited trace on the collector is a combination of the obit of the contact point (when collector's speed is equal to zero the jet creates coiling patters with radius and the movement of the collector.

|  |  |
| --- | --- |
|  | (17) |

where is the deposited trace, is the arc-length, is time, is the contact point at time , is the direction of the collector's speed, is the time that the contact point moves together with the collector. Differentiating and moving from Cartesian to Polar coordinates ( denote the poler coordinates of the contact point ), and taking into account the curvature at the bottom of the jet, we get the system of ODEs:

|  |  |
| --- | --- |
|  | (18) |
|  | (19) |
|  | (20) |

This geometrical model was implemented in Python, and varying the dimensionless parameter from 0 to 1 as suggested, the orbit and the deposited trace can be reconstructed. Verifying the results from [14], the critical velocity at which the straight pattern appears is , which means . for speed ratios the process is highly unstable, forming the translated coiling, alternating loops, W patterns and meanders when the speed ratios are 0.25, 0.5, 0.65, 0.85, respectively.

## 2.5 Gaussian Process Regression

Gaussian Process Regression is a nonparametric stochastic process with strong probabilistic fundaments [17][24]. GPR is a supervised machine learning technique, which predicts a probability distribution based on Bayesian theory unlike other machine learning algorithms that give deterministic predictions. The idea behind GPR is that the posterior probability can be modified based on a prior probability, given a new observation. Those characteristics give reliability to the prediction from a probabilistic point of view, while providing more wholesome information regarding the prediction. Assuming there is a dataset available, consisting of input-output pairs of observations that are generated by an unknown model

|  |  |
| --- | --- |
|  | (21) |

can be completely determined by a mean and a covariance function

|  |  |
| --- | --- |
|  | (22) |
|  | (23) |

GPR aims to learn the mapping between the set of input variables and the unknown model given the set of observations . To map this correlation is typically assigned a GP prior.

Gaussian Processes (GPs) are powerful modelling frameworks incorporating a variety of kernels. A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution [17].

|  |  |
| --- | --- |
|  | (24) |

, where is a kernel function with a set of trainable hyperparameters . The kernel defines a symmetric-positive covariance matrix , which reflects the prior available knowledge on the function to be approximated. Furthermore, kernel's eigenvalues define a reproducing kernel Hilbert space, that determines the class of functions within approximation capacity of the predictive GP posterior mean [17].

Hyper-parameters are trained by maximizing the marginal log-likelihood of the model [17].

|  |  |
| --- | --- |
|  | (25) |

Assuming a Gaussian likelihood and using the Sherman-Morrison-Woodbury formula [17] the expression for the posterior distribution is tractable and can be used to perform prediction given a new output for a new input .

|  |  |
| --- | --- |
|  | (26) |
|  | (27) |
|  | (28) |

, where . As referenced before prediction consists of a mean, computed using the posterior mean , and an uncertainty term, computed using the posterior variance .

## 2.6 Multi-fidelity Gaussian Process Regression

The GPR framework, presented above, can be extended to construct probabilistic models able to take into account numerous information sources of different fidelity levels [24]. Supposing that s levels of information source are available, the input, output data pairs can be organized by increasing fidelity as . So, denotes the output of the most accurate and expensive to evaluate model, while denotes the output of the cheapest and least accurate model to evaluate. Assuming that only two models are available, a high-fidelity model and a low fidelity model. Then, the high-fidelity model can be defined as a scaled sum of the low fidelity model plus an error term:

|  |  |
| --- | --- |
|  | (29) |

, where is a scaling constant quantifying the correlation between the two models and denotes another GP which models the bias term for the high-fidelity data.

A numerically efficient recursive inference scheme can then be constructed, by replacing the GP prior with the GP posterior  of the previous inference level, while assuming that the corresponding experimental design sets have a nested structure. This implies that the training inputs of higher fidelity model needs to be a subset of the training inputs of the low fidelity model. This scheme is matching totally the Gaussian posterior distribution predicted by the fully coupled scheme [24], only now the inference problem is essential decoupled into two GPR problems, yielding the Multifidelity posterior distribution with a predictive mean and variance at each level [29].

|  |  |
| --- | --- |
|  | (30) |
|  | (31) |
|  | (32) |
|  | (33) |

, where denote the number of training points from the high and low fidelity models, respectively.

## 2.7 Active Learning

Let's assume again that observations are available where and the next point to be evaluated needs to be considered. The question that arises is if there is a more informed way to pick those points when evaluation is expensive to perform, rather than randomly picking those.

This is achieved through an acquisition function . The role of the acquisition function is to guide the search for the optimum. They are defined in a way such that high acquisition values to correspond to potentially high (if maximization is the optimization goal) or low values (if minimization is the optimization goal) of the unknown model , great uncertainty or even combination of those. Maximizing the acquisition function is used to select the next point to evaluate the function at. So, goal is to sample at .

Every acquisition function depends from or a combination of the both. The scale at which it depends from each one of those defines its exploration, exploitation tradeoff. When exploring, points where the GP variance is large should be chosen. When exploiting, points where the GP mean is closest to optimization goal should be chosen. Many acquisition functions are available, some of them are:

|  |  |  |
| --- | --- | --- |
| Variance |  | Purely exploration, makes sure, that we learn the function  everywhere on x to a similar level of absolute error. |
| Probability of Improvement | is the normal cumulative distribution function | Selects the point most likely to offer an improvement of at least but is extremely sensitive to the choice of the target. |
| Expected Improvement | where  is the normal probability distribution function | Similar to PI but takes into account the magnitude  of the improvement a point can potentially yield as well |
| Lower Confidence Bound |  | Selects points  for evaluation based on the lower uncertainty bound |

After sampling and evaluating , GP regression is performed to fit to the new point as well. Then the process repeats itself until termination criteria are met, such as a maximum number of iterations, a minimum or maximum value is reached or uncertainty is below an allowed value.

References

1. Schaedler, T. A. & Carter, W. B. Architected Cellular Materials. *Annu Rev Mater Res* 46, 1–24 (2015).

2. Truby, R. L. & Lewis, J. A. Printing soft matter in three dimensions. *Nature* 540, 371–378 (2016).

3. Lewis, J. A. & Ahn, B. Y. Three-dimensional printed electronics. *Nature* 518, 42–43 (2015).

-------------------------

[5] Accurate Prediction of Melt Electrowritten Laydown Patterns from Simple Geometrical Considerations \*\*??

[6] Bradski, G.R., "Real time face and object tracking as a component of a perceptual user interface," Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on Applications of Computer Vision, vol., no., pp.214,219, 19-21 Oct 1998

[7] D. Comaniciu, P. Meer: Mean shift: A robust approach toward feature space analysis. [IEEE Transactions on Pattern Analysis and Machine Intelligence](https://fr.wikipedia.org/wiki/IEEE_Transactions_on_Pattern_Analysis_and_Machine_Intelligence) (TPAMI), vol.24, p. 603-619, 2002.

[8] Gary Bradski and Adrian Kaehler: Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly Media, 555 pages, 2008

[9] Jianbo Shi and Carlo Tomasi. Good features to track. InComputer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*,* pages 593–600. IEEE, 1994.

[10] ["Intel Hyper-Threading Technology, Technical User's Guide"](https://web.archive.org/web/20100821074918/http:/cache-www.intel.com/cd/00/00/01/77/17705_htt_user_guide.pdf) . p. 13. Archived from [the original](http://cache-www.intel.com/cd/00/00/01/77/17705_htt_user_guide.pdf) on 2010-08-21.

[11] Zhmayev, E.; Zhou, H.; Joo, Y. L. Modeling of non-isothermal polymer jets in melt electrospinning. Journal of Non-Newtonian Fluid Mechanics 2008, 153, 95-108.

[12] C.P. Carroll, Y.L. Joo, Electrospinning of viscoelastic Boger fluids: modeling and experiments, Phys. Fluids 18 (2006) 053102.

[13] N. Mayadeo, K. Morikawa, M. Naraghi, M. J. Green, J. Polym. Sci., Part B: Polym. Phys. 2017, 55, 1393.

[14] Brun, P.-T.; Audoly, B.; Ribe, N.M.; Eaves, T.S.; Lister, J.R. Liquid ropes: A geometrical model for thin viscous jet instabilities. Phys. Rev. Lett. 2015.

[15] Jawed, M.K., Brun, P.-T.; Reis, P.M. A geometric model for the coiling of an elastic rod deployed onto a moving substrate. J. Appl. Mech. 2015.

[16] Habibi, M.; Najafi, J.; Ribe, N.M. Pattern formation in a thread falling onto a moving belt: An "elastic sewing machine". Phys. Rev. E 2011.

GPs

[17] C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X.

[18] Ren, J., Cai, J. & Li, J. High precision implicit function learning for forecasting supercapacitor state of health based on Gaussian process regression. Sci Rep 11, 12112 (2021). https://doi.org/10.1038/s41598-021-91241-z

[19] GPy. GPy: A gaussian process framework in python (since 2012). <http://github.com/SheffieldML/GPy>

[20] Cheng, L., Ramchandran, S., Vatanen, T. et al. An additive Gaussian process regression model for interpretable non-parametric analysis of longitudinal data. Nat Commun 10, 1798 (2019). <https://doi.org/10.1038/s41467-019-09785-8>.

[21] W.V. Li, Q.-M. Shao, Gaussian processes: Inequalities, small ball probabilities and applications, Handbook of Statistics, Elsevier, Volume 19, 2001, Pages 533-597, ISSN 0169-7161, ISBN 9780444500144, <https://doi.org/10.1016/S0169-7161(01)19019-X>.

[22] Hensman J, Fusi N, Lawrence ND. 2013 Gaussian processes for big data. (<http://arxiv.org/abs/1309.6835>).

[23] Eric Schulz, Maarten Speekenbrink, Andreas Krause, A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions, Journal of Mathematical Psychology, Volume 85, 2018, Pages 1-16, ISSN 0022-2496, <https://doi.org/10.1016/j.jmp.2018.03.001>.

LMF

[24] Kennedy, Marc C., and Anthony O'Hagan. 2000. "Predicting the Output from a Complex Computer Code When Fast Approximations Are Available." Biometrika 87 (1): 1–13. <http://www.jstor.org/stable/2673557>.

[25] Le Gratiet L, Garnier J. 2014 Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. Int. J. Uncertainty Quant. 4, 365–386. (doi:10.1615/Int.J.UncertaintyQuantification.2014006914)

[26] Babaee H, Perdikaris P, Chryssostomidis C, Karniadakis G. 2016 Multi-fidelity modeling of mixed convection based on experimental correlations and numerical simulations. J. Fluid Mech. 809, 895–917. (doi:10.1017/jfm.2016.718)

[27] Lawrence, Neil D., and Andrew J. Moore. 2007. "Hierarchical Gaussian Process Latent Variable Models." In, 481–88.

[28] Babaee H, Perdikaris P, Chryssostomidis C, Karniadakis G. 2016 Multi-fidelity modeling of mixed convection based on experimental correlations and numerical simulations. J. Fluid Mech. 809, 895–917. (doi:10.1017/jfm.2016.718)

[29]Perdikaris P, Raissi M, Damianou A, Lawrence ND, Karniadakis GE. 2017 Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling. Proc. R. Soc. A 473: 20160751. <http://dx.doi.org/10.1098/rspa.2016.0751>.

[30] L. Parussini, D. Venturi, P. Perdikaris, G.E. Karniadakis, Multi-fidelity Gaussian process regression for prediction of random fields, Journal of Computational Physics, Volume 336, 2017, Pages 36-50, ISSN 0021-9991, <https://doi.org/10.1016/j.jcp.2017.01.047>.

AL-BO

[…] Brochu, E., Cora, V.M., & Freitas, N.D. (2010). A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. ArXiv, abs/1012.2599.

[…] Shalloo, R.J., Dann, S.J.D., Gruse, JN. et al. Automation and control of laser wakefield accelerators using Bayesian optimization. Nat Commun 11, 6355 (2020). <https://doi.org/10.1038/s41467-020-20245-6>.

[…] Lookman, T., Balachandran, P.V., Xue, D. et al. Active learning in materials science with emphasis on adaptive sampling using uncertainties for targeted design. npj Comput Mater 5, 21 (2019). <https://doi.org/10.1038/s41524-019-0153-8>.

[…] Sheng, Y., Wu, Y., Yang, J. et al. Active learning for the power factor prediction in diamond-like thermoelectric materials. npj Comput Mater 6, 171 (2020). <https://doi.org/10.1038/s41524-020-00439-8>.

[…] Forrester, Alexander I. J., András Sóbester, and Andy J. Keane. 2008. Engineering Design via Surrogate Modelling: A Practical Guide. wiley. <https://doi.org/10.1002/9780470770801>.

[…] Forrester, Alexander I. J., András Sóbester, and Andy J. Keane. 2008. Engineering Design via Surrogate Modelling: A Practical Guide. wiley. <https://doi.org/10.1002/9780470770801>.

[…]

[…] Lizotte, D. (2008). Practical bayesian optimization.

[…] Luong, P., Gupta, S., Nguyen, D., Rana, S., & Venkatesh, S. (2019). Bayesian Optimization with Discrete Variables. Australasian Conference on Artificial Intelligence.

[…] Jalali, A., Azimi, J., & Fern, X.Z. (2012). Exploration vs Exploitation in Bayesian Optimization. ArXiv, abs/1204.0047.

[…] Kandasamy, K., Schneider, J., & Póczos, B. (2015). Bayesian active learning for posterior estimation.