TUM Data Innovation Lab

Munich Data Science Institute (MDSI)

Technical University of Munich

&

Henkel AG & Co. KGaA

Final report of project:

Visco Vision: Accelerate Adhesives Innovation via Video-Based Viscosity Measurement

Authors	B.Sc Kamil Hlubek, B.Sc Irem Zeynep Alagoz, B.Sc Huajian Zeng, B.Sc Azza Baatout
Mentor(s)	M.Sc Florian Roscheck, PhD. Andrii Kleshchonok
TUM Mentor	Prof. Dr. Massimo Fornasier
Project lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

Acknowledgements

First of all, we would like to thank our Project Sponsor, Henkel, and our Henkel Mentors Florian Roscheck, M.Sc., Andrii Kleshchonok, Ph.D., Andreas Pamvouxoglou, Ph.D., and Giovanna Tezeli, M.Sc., for their support, encouragement, and guidance throughout the project.
Thank you very much for your great personal commitment.
We would also like to extend our gratitude to our Project Lead, Dr. Ricardo Acevedo Cabra, and Prof. Dr. Massimo Fornasier for giving us the opportunity to conduct the project at the Munich Data Science Institute (MDSI).

Abstract

Viscosity measurement of non-Newtonian fluids plays a crucial role in adhesive development, yet conventional testing methods are time-consuming and limit high-throughput experimentation. In this study, we investigate a computer vision-based approach to estimate viscosity from video recordings of fluid flow through a dispensing system. Our focus is on complex geometries comprising a cylindrical section, a rapid diameter reduction, and a smooth convergent section.

To predict key flow characteristics such as pressure drop, velocity distribution, and shear rate, we initially considered using Computational Fluid Dynamics (CFD) tools, including *FEniCS*, *RheoTool* [17], *OpenLB* [11], and *deal.II* [1]. However, due to their high computational cost and setup complexity, we instead adopted analytical and semi-analytical models based on the Navier-Stokes equations in spherical coordinates. While these models provided rapid predictions, they failed to match experimental measurements due to the unknown boundary conditions of the experimental setup, but maybe also because of uncertainties in flow rate estimation and sensitivity to power-law fluid parameters.

Given these challenges, we ultimately focused on deep learning techniques to enhance viscosity prediction, leveraging structured video data, dispensing pressure, and adhesive cartridge geometry. This approach aims to enable fast, contactless viscosity measurements and streamline adhesive formulation in Henkel's automated laboratory.

Contents

1 Introduction

Adhesives, sealants, and functional coatings are critical to countless industrial applications—from automotive manufacturing and aerospace engineering to consumer electronics and packaging. Henkel Adhesive Technologies, recognized as a global leader in this field, continuously pushes the boundaries of innovation to develop products that meet rigorous performance and sustainability standards. In particular, the rapidly evolving landscape of high-throughput formulation requires not only precision in product performance but also efficiency in testing and quality control.

One of the fundamental properties that determine the performance of adhesives is viscosity—a parameter that influences flow, adhesion, and curing behavior. Traditionally, viscosity measurements are obtained using mechanical rheometers, which, while accurate, are inherently time-intensive and limited in throughput. This manual process presents a significant bottleneck in automated laboratory environments, where rapid iteration and immediate feedback are essential for optimizing formulations.

In response to these challenges, our project proposes a novel, computer vision–based approach for estimating the viscosity of non-Newtonian adhesives from video recordings of the dispensing process. By capturing the dynamics of fluid flow using high-speed cameras and analyzing the recorded images with advanced optical flow algorithms and deep learning models, we aim to extract critical flow features that are directly linked to the underlying viscosity. Moreover, by trying to incorporate auxiliary parameters—such as dispensing pressure and cartridge geometry—into our model, we enhance the predictive accuracy and robustness of the viscosity estimation.

This work is built upon an interdisciplinary collaboration that spans expertise in fluid dynamics, deep learning, and industrial process automation. The research leverages cutting-edge cloud infrastructure, including Azure DevOps and Azure Machine Learning, to enable scalable model training and deployment. The resulting system is designed to integrate seamlessly into Henkel's automated laboratory workflow, providing a fast, contactless, and reliable method for real-time viscosity monitoring.

In the following sections, we first outline the theoretical foundations underlying our approach, including the derivation of analytical models based on the Navier–Stokes equations and power-law rheological descriptions. We then describe the data exploration, preprocessing pipelines, and optical flow methodologies used to extract meaningful features from raw video data. Finally, we present our deep learning–based architecture, which combines visual cues with physical flow measurements to achieve high predictive performance across diverse adhesive formulations.

By addressing the limitations of traditional viscosity measurement techniques, our approach not only promises to accelerate the formulation process but also paves the way for further innovations in process monitoring and quality control within the adhesive industry.

2 Mathematical Description of the flowing Fluid

To get an overall understanding of the problem at hand we present the governing Navier Stokes equations. These model the viscosity as a constitutive relation and are used to derive a boundary value problem describing the radial flow through a conical geometry (see Appendix D)

2.1 Navier-Stokes Equations

We describe the fluid flow using the Navier-Stokes equations for incompressible fluids, which embody the conservation of mass and momentum. For our analysis, we focus on the momentum conservation equation:

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v}\right) = -\nabla p + \nabla\cdot\boldsymbol{\tau} + \mathbf{f}, \tag{1}$$

where:

- ρ is the fluid density,
- $\mathbf{v}$ is the velocity field,
- p denotes the pressure,
- $\boldsymbol{\tau}$ is the deviatoric stress tensor,
- $\mathbf{f}$ represents the body force per unit volume (e.g., gravity).

For the flows considered here, the Reynolds numbers are very low—indicating that the inertial forces (represented by the convective term $(\mathbf{v}\cdot\nabla)\mathbf{v}$) are negligible compared to viscous forces. Moreover, the body forces $\mathbf{f}$ can be neglected. With these simplifications, the momentum equation reduces to:

$$0 = -\nabla p + \nabla\cdot\boldsymbol{\tau}.$$

2.2 Rheological Models

In fluid mechanics, rheological models are constitutive equations that describe how a fluid's internal stresses (particularly the deviatoric stress) depend on the rate at which the fluid deforms (characterized by the strain rate tensor). While Newtonian fluids have a constant viscosity regardless of the deformation rate, many real-world fluids exhibit non-Newtonian behavior, meaning that their effective viscosity changes with the local shear rate. The choice of a rheological model depends on the fluid's microstructure and the observed flow behavior. Our overall task to solve was to calculate exactly this function experimentally.

1. Newtonian Model

For a Newtonian fluid, the relationship between the deviatoric stress tensor $\boldsymbol{\tau}$ and the strain rate tensor $\mathbf{D}$ is linear:

$$\boldsymbol{\tau} = 2\eta\mathbf{D} \tag{2}$$

where η is the constant viscosity. This simple relationship implies that the fluid's resistance to deformation does not change with the shear rate.

2. Power-Law Model

Many non-Newtonian fluids, such as polymer solutions or blood, exhibit shear-thinning or shear-thickening behavior. The power-law model is widely used to describe such fluids. In this model, the effective viscosity η_{eff} is given by:

$$\eta_{\text{eff}} = K\,\dot{\gamma}^{n-1}, \tag{3}$$

or equivalently, the deviatoric stress can be expressed as:

$$\boldsymbol{\tau} = 2K\,\dot{\gamma}^{n-1}\mathbf{D},$$

where:

- K is the flow consistency index,
- n is the power-law index ($n < 1$ indicates shear-thinning, $n > 1$ indicates shear-thickening),
- $\dot{\gamma}$ is a measure of the shear rate, typically defined as $\dot{\gamma} = \sqrt{2\,\mathbf{D} : \mathbf{D}}$.

3. Carreau Model

The Carreau model is designed to capture the behavior of fluids that exhibit a transition between two viscosity plateaus: one at low shear rates (zero-shear viscosity, η_0) and one at high shear rates (infinite-shear viscosity, η_∞). Its formulation is:

$$\eta_{\text{eff}} = \eta_\infty + (\eta_0 - \eta_\infty)\left[1 + (\lambda\,\dot{\gamma})^2\right]^{\frac{n-1}{2}}, \tag{4}$$

where:

- η_0 is the viscosity at very low shear rates,
- η_∞ is the viscosity at very high shear rates,
- λ is a time constant related to the fluid's relaxation behavior,
- n is the power-law index governing the high shear rate behavior.

4. Cross Model

Like the Carreau model, the Cross model describes fluids with a gradual transition between two viscosity limits. It is typically written as:

$$\eta_{\text{eff}} = \eta_\infty + \frac{\eta_0 - \eta_\infty}{1 + (\lambda\,\dot{\gamma})^m}, \tag{5}$$

where m is a dimensionless parameter that adjusts the sharpness of the transition between the two regimes.

5. Yield Stress Models (Bingham Plastic and Herschel–Bulkley Models)

Some fluids require finite stress to initiate flow, exhibiting yield stress τ_y below which they behave like a solid. Two common models incorporating yield stress are:

Bingham Plastic Model The fluid flows only when the applied stress exceeds the yield stress, and above that threshold, it behaves like a Newtonian fluid:

$$\boldsymbol{\tau} = \tau_y + 2\eta\mathbf{D} \quad \text{for } \tau > \tau_y, \tag{6}$$

with no flow (or a rigid behavior) when $\tau \leq \tau_y$.

Herschel - Bulkley Model This model generalizes the Bingham plastic by combining a yield stress with a power-law dependence:

$$\boldsymbol{\tau} = \tau_y + K\,\dot{\gamma}^n \quad \text{for } \tau > \tau_y, \tag{7}$$

where K and n describe the flow behavior once the yield stress is overcome.

3 Data Exploration and Preprocessing

3.1 Evaluating the Data from a traditional Rheometer

One part of the data was obtained by employing a traditional rheometer to receive the shear stress as a function of the shear rate. Using this data, we implemented a *Python* class fitting the data to rheological models. The primary goal of this analysis was to evaluate the performance of various rheological models, with a particular focus on the Power Law model, and compare it against other models such as the Carreau, Cross, Yasuda, Bingham, Herschel-Bulkley, and Casson models The value of this comparison is that the Power-Law is the easiest model to deal with in a theoretical investigation.

3.1.1 Data Loading and Filtering

The analysis started with loading and filtering rheological data from a file. A threshold for the maximum shear rate was applied to filter the data because the traditional rheometer operates within a specific range of shear rate values, yielding physical results. This step ensured that only relevant data points were included for further analysis.

The results of the model comparison revealed that the **Carreau model** performed the best in most cases, outperforming the Power Law and Cross models. This outcome was expected, as the Carreau model is a more complex, 5-parameter model, whereas the Power Law model is characterized by two parameters only.

3.1.2 Power Law Model Evaluation

Despite the superior performance of the Carreau model, the analysis also focused on evaluating the suitability of the Power Law model for modeling fluid viscosity. The results (see Fig. 1 for a subset of the data) showed that while the Power Law model had higher MAE values compared to the Carreau model (see Fig. 2 for the same subset), it still provided reasonable predictions for most products.
Now, Tab. 1 summarizes how often each viscosity model performed the best in our evaluation.

Model Name	Count
Carreau	13
Power Law	3
Cross	3
Yasuda	2

Table 1: Counts of models achieving the best performance.

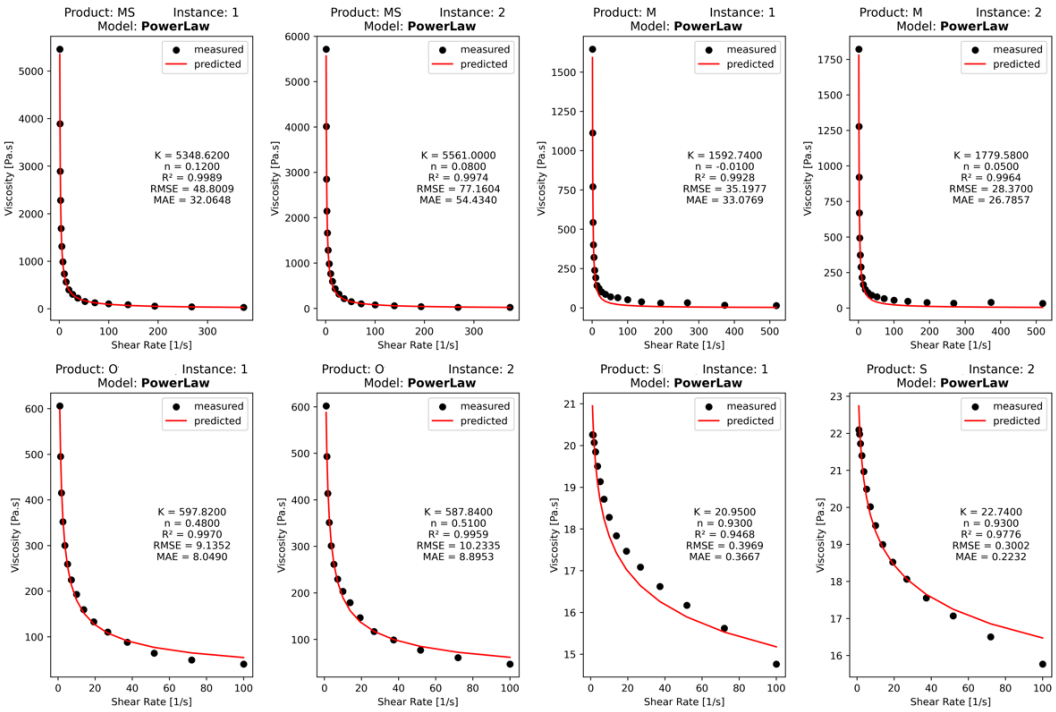


Figure 1: Rheometer Data fitted to the Power Law Model

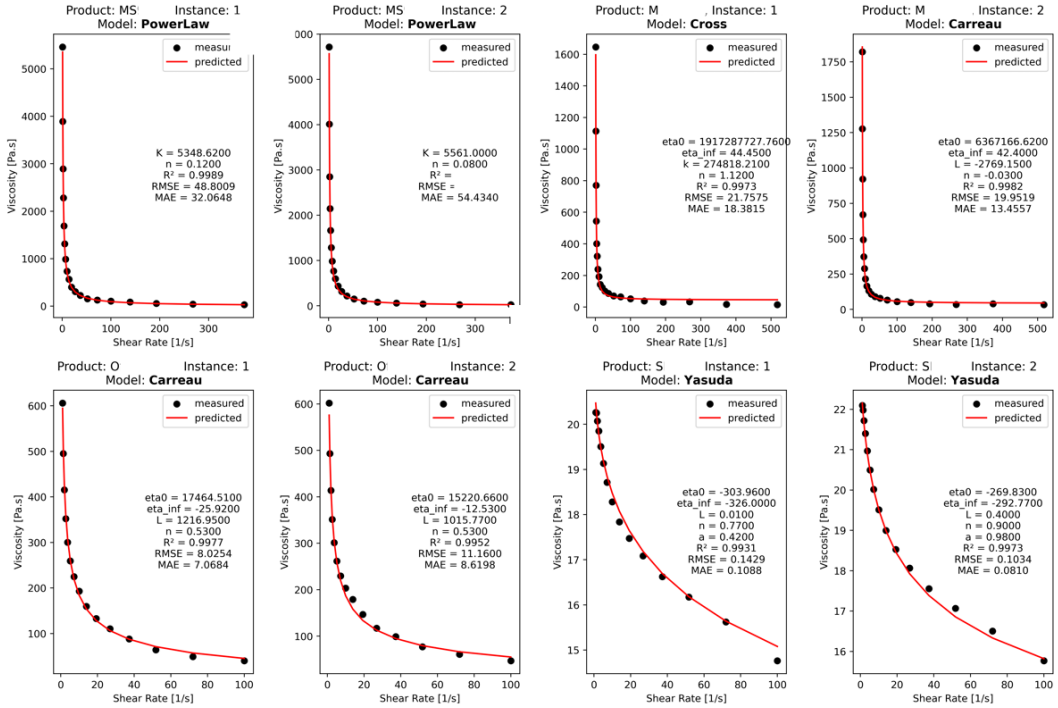


Figure 2: Rheometer Data fitted to the Best Performing Model

3.2 Evaluating the captured Videos of the Fluids

To extract the flow rate and free jet diameter for our models, we developed a structured data pipeline that transforms raw `.bmp` files into both visual and numerical outputs while displaying the result of each step. Like that we were able to use a simple draft implementation in order to validate our analytical formulations. The following steps describe the pipeline´s operation:

1. **Video Conversion:** The pipeline begins by converting raw `.bmp` files into `.mp4` video files. This conversion reduces storage requirements and simplifies processing, while also providing an intuitive visual representation of the flow.

2. **Preprocessing and Cropping:** The generated videos are cropped to isolate the region of interest:

   - **Left Crop:** Excludes the flow tip to ensure measurements start after a set distance.
   - **Top and Bottom Crop:** Refines the vertical region.
   - **Right Crop:** Removes irrelevant sections of the flow field.

   These steps focus the analysis on the most pertinent parts of the fluid motion.

3. **Optical Flow Analysis:** The Lucas-Kanade optical flow algorithm [13] is applied to the cropped videos to track particle movements. Here, the Lucas-Kanade optical flow algorithms were used in order to obtain easy-to-interpret data, but we evaluated other algorithms for their precision, as well (see Appendix B and Sec. 5.1). The

resulting trajectories are visualized in the videos, providing an initial check of the flow behavior. Detailed trajectory data, including particle coordinates and tracking status (active or lost), is saved in `.json` files for further analysis.

4. **Visualization and Metric Computation:**
   - **Final Frames and Overlays:** Final video frames are saved as snapshots and active trajectories are overlaid to emphasize meaningful particle movements.
   - **Quantitative Metrics:** For each trajectory, key metrics (trajectory length, particle velocity, and flow rate) are computed. The velocity is calculated using the trajectory length, tip diameter, and frame rate (see Sec. 4.2 for the conversion approaches used). The results are compiled into a CSV file containing experimental parameters and trajectory statistics.
   - **Statistical Visualization:** Boxplots are generated to display the distributions of trajectory lengths, velocities, and flow rates across experiments. These plots (see Fig. 12) include annotations such as the total number of processed trajectories and flag cases where no trajectory was identified.
5. **Interactive Reporting:** The processed results are summarized in interactive HTML tables that group experimental data and highlight key metrics (e.g., highest velocities in dark red). A comprehensive report consolidates trajectory visualizations, last-frame snapshots, and cropped flow images into a single navigable HTML file (see Fig. 13).

4 Pipeline Architecture & Model Framework

4.1 General Preprocessing Pipeline

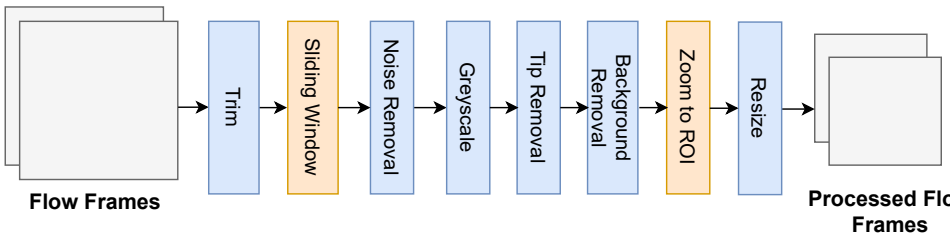
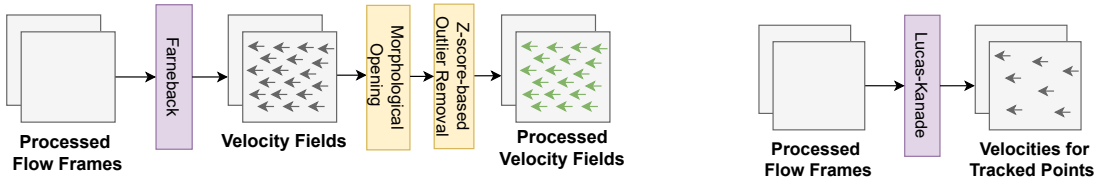


Figure 3: **Preprocessing steps.** Steps highlighted with orange colors are only used in the learning-based approach.

Our preprocessing pipeline is a fundamental component of the architecture. It begins with trimming the video frames by excluding the initial and final frames, as we observed unstable flow states in these regions in certain cases.
In the learning-based approach, we apply a sliding window strategy, which is explained in more detail in Sec. 4.4. To reduce noise in the images, we employ Gaussian blurring and convert RGB videos to grayscale. Gaussian blurring is effective for noise reduction by smoothing the image, which helps in subsequent processing steps.

A common issue in most videos is the presence of a tip with varying sizes. To address this, we incorporate a tip removal step, which ensures more consistent frames and enhances focus on the flow near the tip. Additionally, the background in the images is often nonuniform. We remove it to improve the robustness of the optical flow calculation and prevent the model from learning irrelevant or misleading features. We utilize Canny edge detection [5] and Hough transform implemented in OpenCV [3] for both tip and background removal. These algorithms are fast and do not require training.
For the learning-based approach, we also zoom into the region of interest (ROI) to emphasize flow-related features while avoiding excessive resizing that could distort the input dimensions. This step ensures that the processed flow frames highlight relevant areas while maintaining a suitable input size for downstream tasks.
Finally, we apply resizing to ensure a manageable input size.

4.2 Optical Flow Models



(a) **Dense Optical Flow Pipeline.** (b) **Sparse Optical Flow Pipeline.**

Figure 4: Comparison of Dense and Sparse Optical Flow Pipelines. We apply additional post-processing steps to increase the stability and robustness of the dense algorithm.

We experimented with both dense and sparse optical flow algorithms in our architectures. Dense optical flow algorithms calculate the velocity for every pixel in the image, as detailed in Appendix B.1. For this purpose, we utilized the Farneback algorithm [7] as the primary dense optical flow approach.
Since dense optical flow algorithms are sensitive to noise and may detect spurious or outlier velocities in noisy images, we applied additional post-processing steps to enhance their robustness. As illustrated in Fig. 4 (a), we introduced the following techniques:

- **Morphological Opening**: This operation uses a rectangular kernel to refine the magnitude matrix. It removes small, noisy regions while preserving the primary motion patterns in the image.
- **Z-Score-Based Outlier Removal**: Outlier flow vectors, which have magnitudes significantly larger or smaller than the mean by more than two standard deviations, are removed. This step helps mitigate the impact of extreme noise or artifacts, focusing on realistic motion patterns.

Sparse optical flow algorithms, on the other hand, calculate velocities for only the most salient or feature-rich points in the frames. For this purpose, we employed the Lucas-Kanade algorithm [13], one of the most widely used sparse optical flow methods. Since sparse optical flow inherently operates on a reduced subset of points, we did not apply

post-processing steps after this method. This decision was made to retain the detected feature points and avoid reducing the number of tracked velocities.
We present and discuss the comparative performance of these two optical flow strategies in Sec. 5.1.

4.3 Analytical Approach

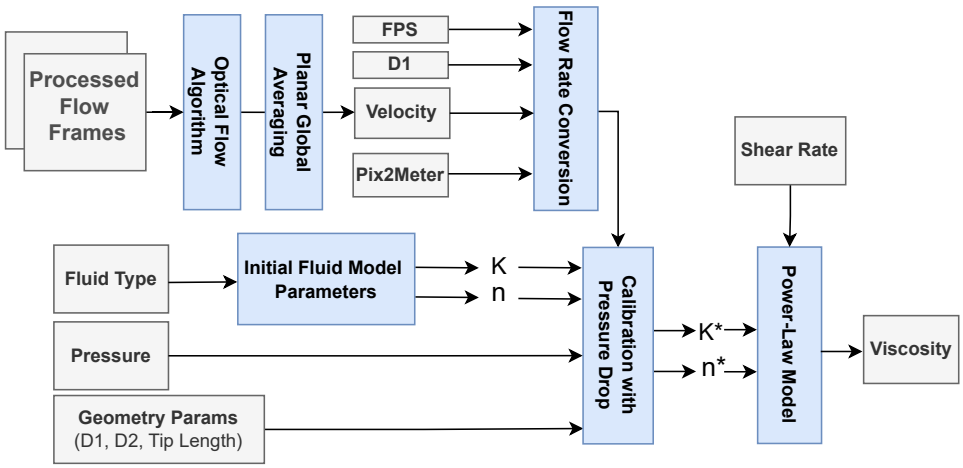


Figure 5: **Analytical Approach Architecture**

We utilized an analytical approach leveraging the physical properties of fluids. Specifically, the Power-Law Fluid Model (detailed in Sec. 2.2) forms the basis of our viscosity prediction method. The overall structure of the proposed analytical approach is depicted in Fig. 5.
The process begins by estimating the velocity field, $\mathbf{v}$, from a fluid video using an optical flow algorithm. The velocity profile of the fluid is highly relevant to its viscosity properties. The planar velocity, v_y, is computed as the average pixel velocity along a single y-plane, evaluated across multiple planes near the tip to minimize measurement errors. A global velocity, $\bar{v}_y$, is obtained as the temporal average of v_y across all frames and planes.
The global velocity is converted into a flow rate using the frame rate (f) and a pixel-to-meter conversion factor (pix2m), estimated from the tip length in the frame. The flow rate Q is given by:

$$Q = \pi \left(\frac{d_1}{2}\right)^2 \bar{v}_y f \cdot \text{pix2m}, \quad \bar{v}_y = \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} v_{i,j},$$

where N is the number of frames, M the number of pixels in the y-plane, $v_{i,j}$ the velocity at pixel j in frame i, and d_1 the tip diameter.
The calculated flow rate, alongside the pressure drop, tip diameter (d_1), opening diameter (d_2), and tip length, is used to calibrate the flow consistency index (k) and flow behavior index (n) in the Power-Law Fluid Model.
This approach provides a physically grounded methodology for estimating fluid-specific power-law parameters, ensuring efficiency and interpretability. The results and two formulated approaches are investigated further in Sec. 5.2

4.4 Learning-Based Approach

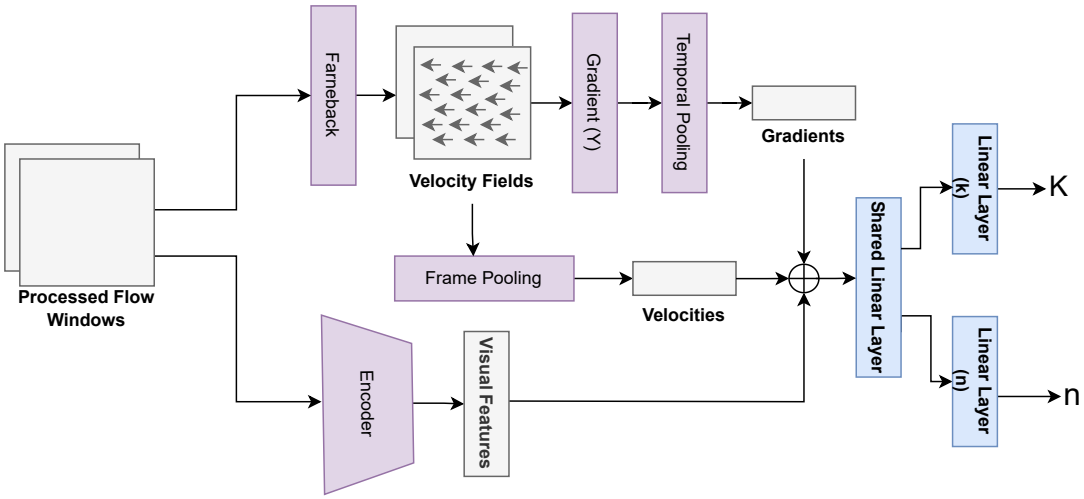


Figure 6: **Learning-based Approach Architecture.** We combine the advantages of deep learning architectures with optical flow algorithms and fluid dynamics models.

To overcome the limitations of purely analytical approaches, we propose a learning-based architecture that leverages additional visual features to model the non-linear relationship between flow rates, visual cues, and power-law parameters. Khayat et al. [10] show that the width of a free-surface jet for shear-thinning power-law fluids at moderate Reynolds numbers, with a fully developed Poiseuille flow upstream, strongly depends on the power-law index n. Higher shear-thinning effects ($n < 1$) lead to a wider jet, whereas for higher values of n, the fluid contracts. For sufficiently low values of n, this contraction effect is significantly reduced or absent. A learning-based system can capture these patterns using flow videos. The primary objective is to develop a model that adheres to physical constraints, remains explainable, is robust to noise, and effectively captures the complex relationship between flow rate and viscosity.

The proposed architecture follows a dual-branch design. The first branch estimates the velocity field using the dense optical flow algorithm by Farneback [7]. Planar velocity calculations are performed near the jet tip, followed by frame-wise global pooling of velocity magnitudes to capture temporal dynamics. Additionally, we compute the velocity gradient in the y-direction (orthogonal to the primary flow direction) and apply temporal pooling to extract average velocity changes across the y-axis. These features provide a comprehensive representation of flow behavior.

The second branch employs an encoder to extract visual features from the video. We applied normalization and augmentation to the images to accelerate training and mitigate overfitting. Normalization is performed by computing the mean and standard deviation over the training dataset. Specifically, we used brightness augmentation to account for lighting variations and small rotational augmentations to counteract the effects of camera movements, following the work of Park et al. [16].

A 3D CNN was chosen as the backbone due to its effectiveness in capturing spatial relationships in 2D data and spatiotemporal patterns in video data through convolutional

operations [23]. Previous studies [16, 24] have also demonstrated the effectiveness of CNN-based encoders in predicting fluid viscosities. Each layer incorporates batch normalization to improve convergence and prevent overfitting, with the Rectified Linear Unit (ReLU) serving as the activation function. Additionally, strided convolution is employed for efficient downsampling.
The outputs from both branches are concatenated and passed through a shared linear layer to generate a single feature vector representing the fluid video. Separate fully connected layers predict the power-law parameters, K (flow consistency index) and n (flow behavior index). Linear normalization is applied after each linear layer, and dropout is introduced before the output layer to mitigate overfitting. ReLU activation is used after all linear layers except the output layer. The n prediction is mapped using the tanh activation function to constrain its range to $[-1, 1]$, aligning with the theoretical boundaries of the flow behavior index. Since K exhibits a large and non-uniform range, we apply logarithmic normalization to stabilize the model training.
We intentionally exclude geometrical and experimental setup information, as these factors vary significantly across experiments for different products. Such variations can lead to overfitting on the experimental configuration rather than learning fluid characteristics. We also evaluated the effect of optical flow on the model's performance by removing it and retaining only the visual branch while keeping the rest of the architecture unchanged. The results are discussed in Sec. 5.3.

4.5 Fluid Video Autoencoder

We utilized the 3D CNN encoder described in Sec. 4.4 to develop an autoencoder for video sequences. The primary motivation for training the autoencoder was to initialize the encoder in the learning-based architecture with robust weights. This process also evaluates the encoder's ability to capture patterns in fluid flows, a critical requirement for downstream tasks.
The objective of the autoencoder is to reconstruct the input video sequence at the output. The autoencoder employs a 3D convolutional neural network as the encoder backbone and a 3D deconvolutional neural network as the decoder. Every convolutional layer other than the last layer of the decoder is followed by batch normalization and reLu activation function. The last layer of the decoder is finalized with Sigmoid activation to obtain values between 0 and 1. Similarly, Linear layers other than the output layers are followed by normalization and activation functions. Similar to our previous approach, we used a sliding window method to augment the video dataset. In addition, we applied rotation and brightness augmentations. To enhance feature extraction, we adopted ROI (region of interest) zooming instead of simple resizing to reduce frame dimensions, as it provided better feature representations. The overall preprocessing pipeline is illustrated in Fig. 3, and the results, including our hyperparameter fine-tuning, are presented in Appendix G.

4.6 Fluid Dynamics Categorization

We also explored a categorization task to assess the encoder's ability to learn visual patterns relevant to viscosity properties. Fluids were classified into four categories based on their fitted Power-Law parameters, with emphasis on the flow consistency index (K).

For multi-class prediction, we appended linear layers to the encoder, applying softmax in the final layer. Further details on fluid categories are provided in Appendix E.

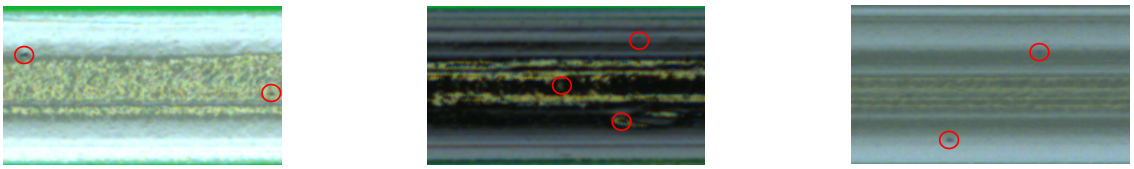
5 Results

5.1 Optical Flow Algorithms

As discussed in Sec. 4.2, to estimate flow velocity, we experimented with both dense (Farneback [7]) and sparse (Lucas-Kanade [13]) optical flow algorithms. To determine the most suitable approach for our task, we conducted a comprehensive evaluation across multiple criteria. Notably, for the dense optical flow method, we downscaled the resolution to 25% of the original image size to enhance computational efficiency.

Accuracy of Flow Speed Estimation

- **Objective:** Identify the algorithm that yields the most accurate flow speed estimation.
- **Evaluation Metric:**
  - **Absolute Error (%)**: The absolute deviation between the estimated flow speed and the ground truth. Lower values indicate higher accuracy.



(a) **Product 1** (b) **Product 2** (c) **Product 3**

Figure 7: **Manual Annotation**. We manually annotated products by tracking distinct particles across frames.

To facilitate this evaluation, we manually annotated a subset of the dataset to establish ground truth flow speed values. We tracked distinct particles, as shown in Fig. 7, for the first 100 frames, skipping every 10 frames to capture the average velocity. These annotations served as the reference for computing absolute error and assessing the accuracy of each method. Specifically, we annotated three different products under two distinct pressure conditions. The selected products exhibited significant differences in appearance, ensuring a diverse and representative evaluation set.
As shown in Tab. 2, the Dense Optical Flow method consistently outperforms the Sparse method in accuracy, exhibiting a lower average absolute error. The accuracy of the Dense method slightly decreases when the resolution is reduced to 25% of the original size. Despite this minor increase in error, the Dense method remains the more reliable choice for precise flow estimation.

Table 2: Comparison of absolute error (%) for Dense and Sparse Optical Flow

Method	**Avg ↓**	**Min ↓**	**Max ↓**	**25% ↓**	**75% ↓**
Dense	13%	1%	48%	3.0%	12.75%
Sparse	23%	7%	70%	13.75%	16.0%
Dense (Scale 0.25)	22%	0%	63%	8.25%	29.5%

Uncertainty in Flow Speed Estimation

- **Objective:** Assess the consistency of the flow speed estimation across different frames.
- **Evaluation Metric:**
  - **Variance:** Measures the fluctuations in flow speed estimates. A lower variance indicates more consistent predictions.

Due to the absence of ground truth velocity values for the entire dataset, we also conducted an unsupervised evaluation of the algorithms. A reliable optical flow algorithm should estimate consistent velocities for the same product under identical pressure conditions. Tab. 3 highlights the variance in flow speed estimation, indicating that the Dense Optical Flow method produces more consistent predictions than the Sparse method. Notably, when the input resolution is downscaled to 25%, the variance further decreases, demonstrating that a lower-resolution Dense method not only accelerates computation but also enhances stability in flow predictions.

Table 3: Variance comparison of flow speed estimation

Method	**Avg ↓**	**Min ↓**	**Max ↓**	**25% / 75% ↓**
Dense	0.377	0.001	4.243	0.005 / 0.383
Dense (Scale 0.25)	0.097	0.00008	1.388	0.002 / 0.077
Sparse	1.321	0.0007	4.860	0.108 / 2.356

Computational Efficiency of Flow Speed Estimation

- **Objective:** Evaluate the computational efficiency of different flow estimation methods.
- **Evaluation Metric:**
  - **Processing Time per 10 Frames (s)**: Measures the average time required to compute flow speed over ten frames. Lower values indicate higher efficiency.

Since our primary objective is to develop a pipeline for enhancing product control in the laboratory, processing speed is another key evaluation metric. According to Tab. 4, the Dense method is computationally more expensive than the Sparse method. However, by downscaling the resolution to 25%, the Dense method becomes significantly faster, even surpassing the Sparse method in processing speed. This demonstrates that reducing resolution is an effective way to achieve an optimal balance between computational efficiency and accuracy.

Table 4: Comparison of processing time per 10 frames (in seconds)

Method	**Avg ↓**	**Min ↓**	**Max ↓**	**25% / 75% ↓**
Dense	14.33	13.38	15.51	14.12 / 14.52
Dense (Scale 0.25)	2.75	2.64	3.20	2.69 / 2.82
Sparse	4.47	3.52	5.01	4.28 / 4.75

Summary of Findings

- The Dense Optical Flow method outperforms the Sparse method in accuracy and consistency, making it the preferred choice for our application.
- However, it is computationally more expensive. To balance performance and efficiency, downscaling the input resolution to 25% provides an effective trade-off, maintaining relatively high accuracy while significantly reducing computational cost.
- For our viscosity prediction task, we decided to use the Dense Optical Flow method at 25% resolution as it provides the best balance between accuracy, stability, and speed.

5.2 Analytical Approaches

For this evaluation, two products were selected where our simplified approach to measuring the flow rate proved effective. It is important to note that our preprocessing algorithm—based on the assumption of a constant jet width—is applicable only for fluids with low flow behavior indices (low n values). Also, we found that by repeating the experiment, we get different flow rates for the flowing fluid, which is a consistency problem for the measurement technique as it is at the moment in general. You can observe that already by eye (see also Fig. 12).

5.2.1 Boles' Formula for Conical Extrusion Pressure Drop

Boles' formula offers an analytical estimate for the viscous pressure drop in conical polymer processing flows. Based on a generalized Newtonian (power-law) fluid model and derived via a radial momentum balance in spherical coordinates with an assumed velocity profile, the formula is given by:

$$\Delta P_{\text{con}} = \frac{2k}{3n \sin\alpha} \left[\frac{3Q(3n+1)\sin\alpha}{4\pi n(1-\cos\alpha)^2(1+2\cos\alpha)} \right]^n \left[\frac{1}{R_2^{3n}} - \frac{1}{R_1^{3n}} \right]$$

where α is the cone half-angle, R_1 and R_2 are the inlet and outlet radii, Q is the volumetric flow rate, and k and n are the power-law fluid parameters.
Notably, for the same geometrical parameters, the pressure drop can be expressed as

$$P = C \cdot k \cdot Q^n$$

where C is a constant that encapsulates the geometry-dependent factors. This compact representation emphasizes the direct proportionality of the pressure drop to the consistency index k and the volumetric flow rate Q raised to the power n.

Despite comprehensive testing, no functional dependency was found between the calculated pressure drop values and the pressure parameter used during initialization (see Fig. 8).

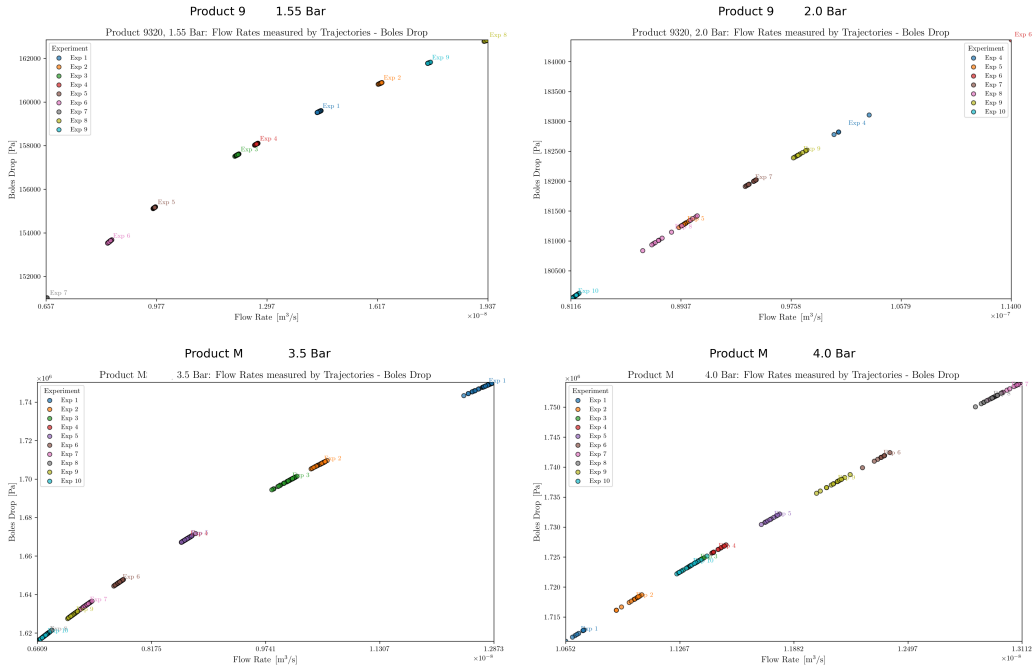


Figure 8: We see the measured flow rates and its corresponding pressure drops for the Boles Formulation for two products at two dispensing pressure paramters.

5.2.2 Navier-Stokes Derived Pressure Model and Power-Law Constants

An alternative approach derives the pressure formulation directly from the Navier-Stokes equations by converting them into a boundary value problem. This method integrates the full momentum balance with the power-law description of viscosity, yielding a pressure expression that more comprehensively captures the flow dynamics—including scenarios where viscoelastic effects are significant. Although this model is inherently more complex, it offers improved predictive capabilities over a wider range of flow conditions. For a detailed derivation and discussion, please refer to Appendix D.
Again, the plots of flow rate versus calculated pressure drop revealed no functional dependency between the predicted pressure drop values and the pressure parameter used during initialization (see Fig. 9).

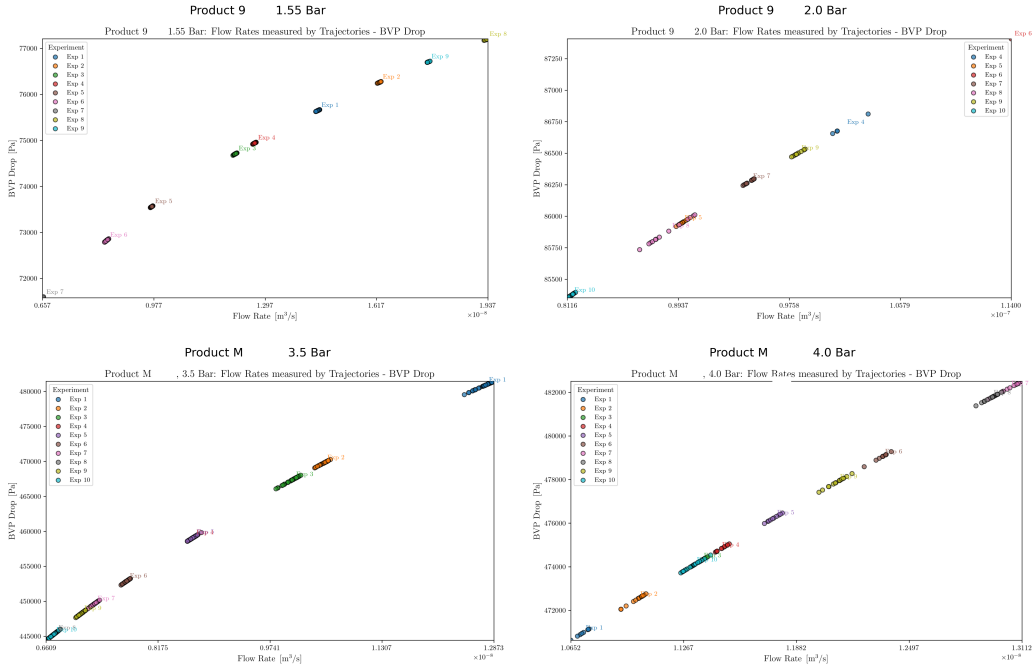


Figure 9: We see the measured flow rates and its corresponding pressure drops for the BVP Formulation

5.2.3 Summary of Findings

In this evaluation, two analytical approaches—the Boles' formula and a Navier-Stokes derived pressure model—were compared for predicting pressure drops in conical processing flows. Both models rely on a power-law description of viscosity but differ in their derivation and complexity. Boles' formula provides a straightforward analytical estimate, while the Navier-Stokes approach, formulated via a boundary value problem, captures additional flow dynamics and viscoelastic effects.
Experimental investigations using two products (each under two distinct pressure parameter settings and repeated over ten measurements) showed that there is no functional dependency between the calculated pressure drop values and the initial pressure parameter. Furthermore, the experimental results underscore a limitation of our preprocessing algorithm: it assumes a constant jet width, an assumption valid only for low n fluids.
Additional challenges associated with these approaches include:

- **Velocity Variations:** For fluids with higher n values, the jet is likely to experience changes in velocity, making a reasonably precise calculation of the flow rate more difficult.
- **Advanced Image Processing Requirements:** The complexities introduced by these velocity changes demand the applicability of an optical flow algorithm to accurately track and quantify the jet behavior.

Overall, while both analytical methods offer valuable insights into pressure drop estimation, the findings indicate that further refinements—especially addressing the challenges of velocity variations and the need for advanced optical flow techniques for high n fluids—are necessary to improve the predictive accuracy of the models.

5.3 Learning-Based Approach

We conducted experiments on the learning-based approach by measuring its performance on **Fluid Consistency Index Prediction** (k), **Fluid Behavior Index Prediction** (n), and **Target Power-Law Viscosity Estimation**. Additionally, we report performance on **Real Viscosity Estimation**, which represents the primary task. The ground truth parameters are obtained by fitting k and n values using Rheometer results, with the fitting process described in Sec. 3. The **target viscosity** is computed using the power-law model, whereas the **real viscosity** is directly measured by the Rheometer.
The dataset is split into training, validation, and test sets with a ratio of 0.8, 0.1, and 0.1, respectively. Product types are uniformly distributed across the splits to prevent bias toward specific products. Additionally, certain products are entirely excluded from the training set to evaluate out-of-domain performance.

5.3.1 Ablation Study

Table 5: Performance on Target Power-Law Viscosity Estimation

Model	MAE (↓)	RMSE (↓)	Weighted MAE (↓)	RMAE (↓)	R^2 (↑)
VisOF	25.7517	56.7659	3.8364	0.0902	0.9959
VisOF-L	26.4737	55.7240	3.5831	0.0843	0.9959
VisOF-FT-AutoE	24.1188	57.9488	4.1947	0.0986	0.9956
VisOF-FT-LAutoE	35.1624	90.1017	3.9588	0.0931	0.9873
Vis	**5.2251**	**17.2681**	**0.7892**	**0.0186**	**0.9996**
Vis-L	27.7798	57.8654	3.4414	0.0809	0.9957
Vis-FT-AutoE	37.7006	92.6426	4.6808	0.1101	0.9873
Vis-FT-LAutoE	38.9736	86.0386	7.0866	0.1666	0.9894
Vis-Freeze-AutoE	59.5921	153.6619	7.4546	0.1753	0.9640
Vis-FT-Classifier	13.0992	29.2855	1.9161	0.0451	0.9989

We conducted ablation studies to investigate the impact of:

- Usage of optical flow algorithm on performance,
- Different latent sizes for visual features,
- Pre-trained weights from the autoencoder and classifier, considering both freezing and fine-tuning options.

Naming Conventions in Tables The following naming conventions are used in the tables:

- **VisOF**: Visual and optical flow features. Details are explained in Sec. 4.4
- **Vis**: Only visual features. The same architecture with VisOF after dropping optical flow calculation.

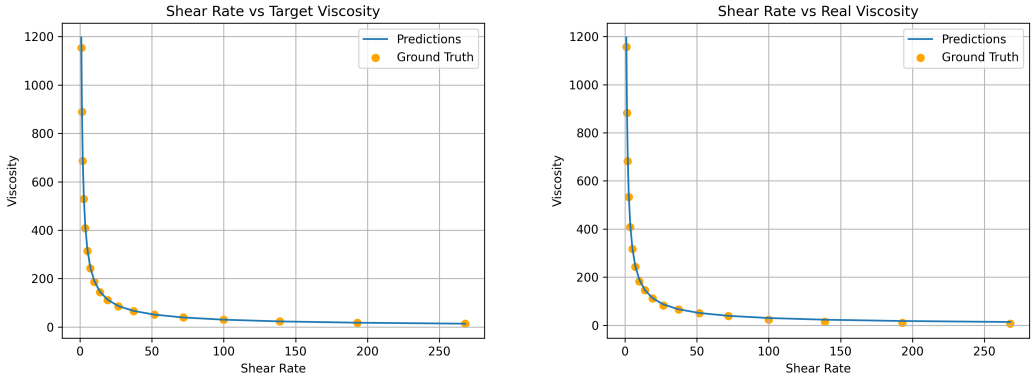
- **L**: The latent dimension for visual features is 512 (default: 256).
- **FT**: Fine-tuning.
- **AutoE**: Using encoder weights from the autoencoder task.
- **Classifier**: Using encoder weights from the categorization task.

Summary of Findings As observed from the experiments in Tab. 5, increasing the latent size does not yield better results in any setting. Fine-tuning weights instead of freezing significantly improved performance.
While the mean error is lower for using autoencoder weights for VisOF, its R^2 and weighted error metrics—which account for the disproportionate effect of larger viscosity values—are worse compared to end-to-end training. Training architecture with visual features end-to-end performed better than all pre-training settings. This suggests that direct training is more effective for this task.
We also evaluated the effectiveness of using encoder weights from the categorization task. It outperformed autoencoder-based pretraining, which suggests categorization training led to learning more relevant features for primary down-stream tasks. However, it still failed to match the performance of end-to-end training.
Lastly, we examined the impact of incorporating optical flow features in our model. Surprisingly, results indicate that the model performs better when relying solely on visual features. This may be attributed to the high variance in flow rates within our dataset, which hinders the model's ability to learn consistent patterns.
More detailed results from the experiments including the performance on the Fluid Consistency Index Prediction k, Fluid Behavior Index Prediction n, and Real Viscosity Estimation can be found in Appendix H

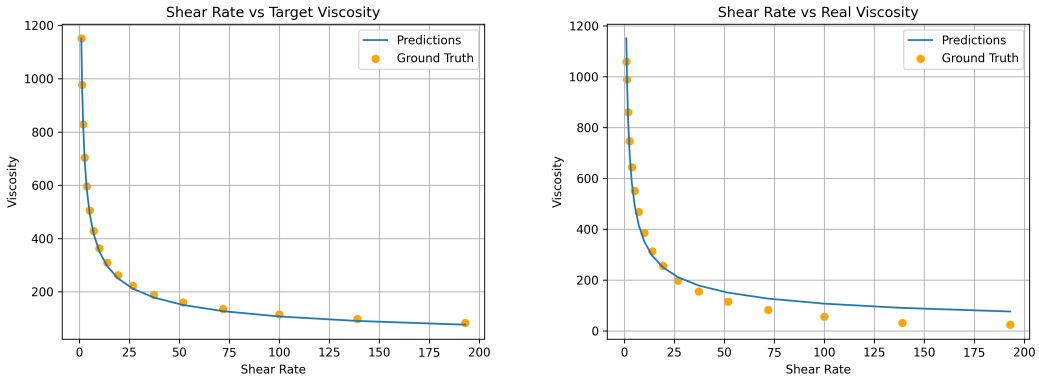
5.3.2 Model Performance on Seen and Unseen Products

We evaluated the models on both seen and unseen fluid types by holding out certain product types from the training set. We present the final performance of models using only visual features and those incorporating flow rate information in addition to visual features. Although using only visual features fits the dataset better than incorporating optical flow, we emphasize the contribution of flow rate, particularly in unseen data, which is critical for the main objective.

Evaluation on Seen Products Fig. 10 presents viscosity predictions for two products, showing strong agreement between model predictions (smooth blue line) and ground truth (orange dots). This indicates a **good model fit**, effectively capturing the shear rate-viscosity relationship, and a **low residual error**, with minimal discrepancies.



(a) Shear Rate vs Target and Real Viscosity - Product A



(b) Shear Rate vs Target and Real Viscosity - Product G

Figure 10: Shear rate vs viscosity prediction performance across two different products.

The results align with shear-thinning behavior: viscosity **decreases rapidly at low shear rates** before stabilizing at higher values. At **low shear rates** (< 20), high viscosity is attributed to molecular interactions, while at **high shear rates** (> 100), it reaches a limiting value. More plots are provided in Appendix H.

Table 6: Comparison of Visual Model and Visual with Optical Flow Model Performance on Real vs Target Viscosity Estimation

Model	**Approach**	**MAE** (↓)	**Weighted MAE** (↓)	**RMAE** (↓)	R^2 (↑)
Target	Vis	5.2251	0.7892	0.0186	0.9996
	VisOF	25.7517	3.8364	0.0902	0.9959
Real	Vis	24.7291	9.5911	0.2071	0.9977
	VisOF	37.2969	11.5314	0.2490	0.9940

It is also worth noting that Fig. 10 (a) illustrates Product 1 aligns more closely with real viscosity values compared to Product 2 as illustrated in Fig. 10 (b) . The difference in real viscosity estimation performance, despite both models aligning strongly with target viscosity, can be attributed to the limitations of the power-law model. Tab. 7 and Tab. 8 quantifies this discrepancy, showing the significant difference between estimating target

viscosity (computed using the power-law model) and real viscosity (measured using the Rheometer), especially for the visual model, which fits the dataset better.

Table 7: Visual Model

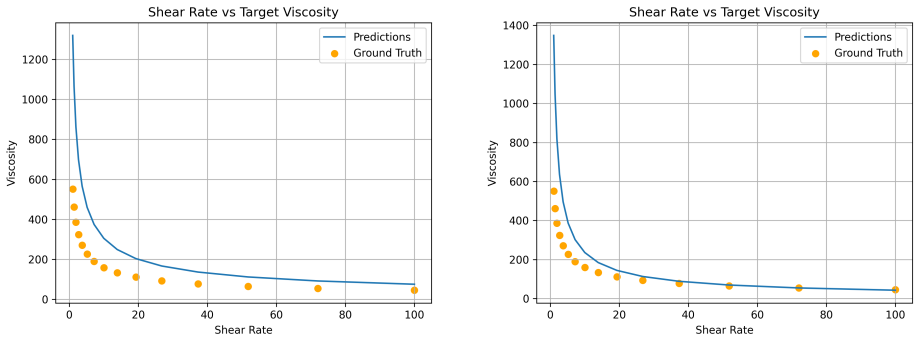
Product	MAE	RMAE
Product A	7.9753	0.0172
Product B	6.8101	0.0180
Product C	1.9959	0.0075
Product D	0.8284	0.0136
Product E	12.8468	0.0538
Product F	0.2782	0.0146
Product G	9.7161	0.0392
Product H	5.4929	0.0088

Table 8: Visual Model with Optical Flow

Product	MAE	RMAE
Product A	6.3942	0.0506
Product B	22.4934	0.0905
Product C	27.3991	0.1141
Product D	3.4914	0.0327
Product E	34.2064	0.1799
Product F	2.1450	0.1103
Product G	29.7146	0.0643
Product H	40.6927	0.0508

We also observed that performance varies depending on the product type, as shown in Tab. 7 and Tab. 8. Additionally, we analyzed the model's uncertainty by calculating the standard deviation in the predictions of K and n parameters for the same fluid across different experiments and sliding windows (Appendix H). Despite performance variations across different products, the overall results are promising, suggesting that the proposed architectures could be further refined for improved viscosity prediction.

Evaluation on Unseen Products We compared the performance of using only visual features versus incorporating optical flow features to enhance visual feature representation. Fig. 11 shows that both approaches capture the viscosity-shear rate relationship, exhibiting a well-defined plateau pattern. However, the precision of the predictions is noticeably lower for unseen products compared to seen products.
Although the model using only visual features performed well on seen fluid types, it struggled on unseen products, performing worse than the model that incorporated optical flow information. This suggests that while visual features effectively capture patterns and shapes of fluids within the dataset, integrating flow rate measurements enables a more robust architecture, improving generalization to unseen fluids.



(a) Only visual features (b) Visual features with optical flow

Figure 11: Shear rate vs viscosity prediction performance on unseen fluid types

6 Conclusions

In this study, we explored a computer vision-based approach for estimating the viscosity of non-Newtonian fluids from video recordings of the dispensing process. While initial investigations using analytical and semi-analytical models based on the Navier-Stokes equations provided theoretical insights, they failed to match experimental measurements due to uncertainties in flow rate estimation and the complex physics introduced by the power-law model. Additionally, Computational Fluid Dynamics (CFD) tools were considered but ultimately disregarded due to their computational expense and complexity.
Our findings highlight the limitations of classical modeling approaches in practical viscosity estimation, particularly for shear-thinning fluids. The observed discrepancies in jet width and velocity distribution motivated us to shift toward deep learning techniques, leveraging structured video data and dispensing parameters. We observed that learning additional visual features with deep learning significantly improves the model's performance. The integration of an optical flow algorithm is particularly important for obtaining a generalizable model for unseen products. This transition aligns with the need for a fast, contactless, and scalable viscosity measurement method to seamlessly integrate into Henkel's automated laboratory workflow.
We would also like to highlight that our findings have been integrated into a user interface that the chemistry team is ready to use. Our focus was on creating a functional and easily implementable design, allowing for seamless adaptation of the interface based on our findings from one sprint to the next.
Future work could focus on refining the deep learning approach, optimizing feature extraction, and validating model performance across a broader range of adhesives. Better performance and evaluation could be achieved by constructing a consistent dataset with controlled velocity variations, diverse product types, and a standardized experimental setup. Additionally, the integration of pressure and tip geometries as input could be further explored with appropriately designed dataset settings. Additionally, simplifying the geometry or capturing footage of the fluid interacting with the floor could enhance the automation pipeline. By addressing these challenges, the aim is to develop a robust system capable of accelerating viscosity characterization in high-throughput adhesive formulation.

References

[1] Pasquale C. Africa et al. "The deal.II library, Version 9.6". In: *Journal of Numerical Mathematics* 32.4 (2024), pp. 369–380. DOI: 10.1515/jnma-2024-0137.

[2] Paul Bourke. *Auto-regression Analysis (AR)*. Online. URL: http://paulbourke.net/miscellaneous/ar/.

[3] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[4] David J. Butler et al. "A Naturalistic Open Source Movie for Optical Flow Evaluation". In: *European Conference on Computer Vision (ECCV)*. 2012, pp. 611–625.

[5] John Canny. "A Computational Approach to Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[6] Alexey Dosovitskiy et al. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 2758–2766.

[7] Gunnar Farnebäck. "Two-frame motion estimation based on polynomial expansion". In: *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer. 2003, pp. 363–370.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361.

[9] R. E. Khayat. "Free-surface jet flow of a shear-thinning power-law fluid near the channel exit". In: *Journal of Fluid Mechanics* 748 (2014), pp. 580–617. DOI: 10.1017/jfm.2014.141.

[10] Roger E. Khayat. "Free-surface jet flow of a shear-thinning power-law fluid near the channel exit". In: *Journal of Fluid Mechanics* 748 (2014), pp. 580–617. DOI: 10.1017/jfm.2014.141.

[11] A. Kummerländer et al. *OpenLB Release 1.7: Open Source Lattice Boltzmann Code*. Version 1.7, Feb. 2024. 2024. DOI: 10.5281/zenodo.10684609. URL: http://doi.org/10.5281/zenodo.10684609.

[12] T. H. Kwon, S. F. Shen, and K. K. Wang. "Pressure drop of polymeric melts in conical converging flow: Experiments and predictions". In: *Polymer Engineering & Science* 26.3 (1986), pp. 214–224. DOI: 10.1002/pen.760260306. URL: https://doi.org/10.1002/pen.760260306.

[13] Bruce D Lucas and Takeo Kanade. "An iterative image registration technique with an application to stereo vision". In: *IJCAI'81: 7th international joint conference on Artificial intelligence*. Vol. 2. 1981, pp. 674–679.

[14] Robert Nau. *ARIMA models for timeseries forecasting*. Online. URL: http://people.duke.edu/~rnau/411arim.htm.

[15] *Optic flow*. URL: https://www.bi.mpg.de/opticflow.

[16] Jae Hyeon Park et al. "Fluid Property Prediction Leveraging AI and Robotics". In: *arXiv preprint* arXiv:2308.02715 (2023). URL: https://arxiv.org/abs/2308.02715.

[17] F. Pimenta and M.A. Alves. *rheoTool*. https://github.com/fppimenta/rheoTool. 2016.

[18] Alessio Pricci, Marco D. de Tullio, and Gianluca Percoco. "Semi-analytical models for non-Newtonian fluids in tapered and cylindrical ducts, applied to the extrusion-based additive manufacturing". In: *Materials & Design* 223 (2022), p. 111168. DOI: 10.1016/j.matdes.2022.111168. URL: https://www.sciencedirect.com/science/article/pii/S0264127522007900.

[19] Shai Rahimi, David Durban, and Savely Khosid. "Wall friction effects and viscosity reduction of gel propellants in conical extrusion". In: *Journal of Non-Newtonian Fluid Mechanics* 165.13–14 (2010), pp. 782–792. DOI: 10.1016/j.jnnfm.2010.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0377025710001126.

[20] Anurag Ranjan and Michael J. Black. "Optical Flow Estimation using a Spatial Pyramid Network". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 4161–4170.

[21] Tobias Senst, Volker Eiselein, and Thomas Sikora. "Robust local optical flow for feature tracking". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.9 (2012), pp. 1377–1387.

[22] Deqing Sun, Stefan Roth, and Michael J. Black. "Secrets of Optical Flow Estimation and Their Principles". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 2432–2439.

[23] Du Tran et al. "Learning Spatiotemporal Features with 3D Convolutional Networks". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 4489–4497. DOI: 10.1109/ICCV.2015.510.

[24] M. Walker et al. "Go with the Flow: Deep Learning Methods for Autonomous Viscosity Estimations". In: *Digital Discovery* 2.5 (2023), pp. 1540–1547.

[25] Philippe Weinzaepfel et al. "DeepFlow: Large Displacement Optical Flow with Deep Matching". In: *IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 1385–1392.

A Data Analysis Pipeline Example

We provide here the outputs for one walkthrough of our experimental data analysis pipeline:

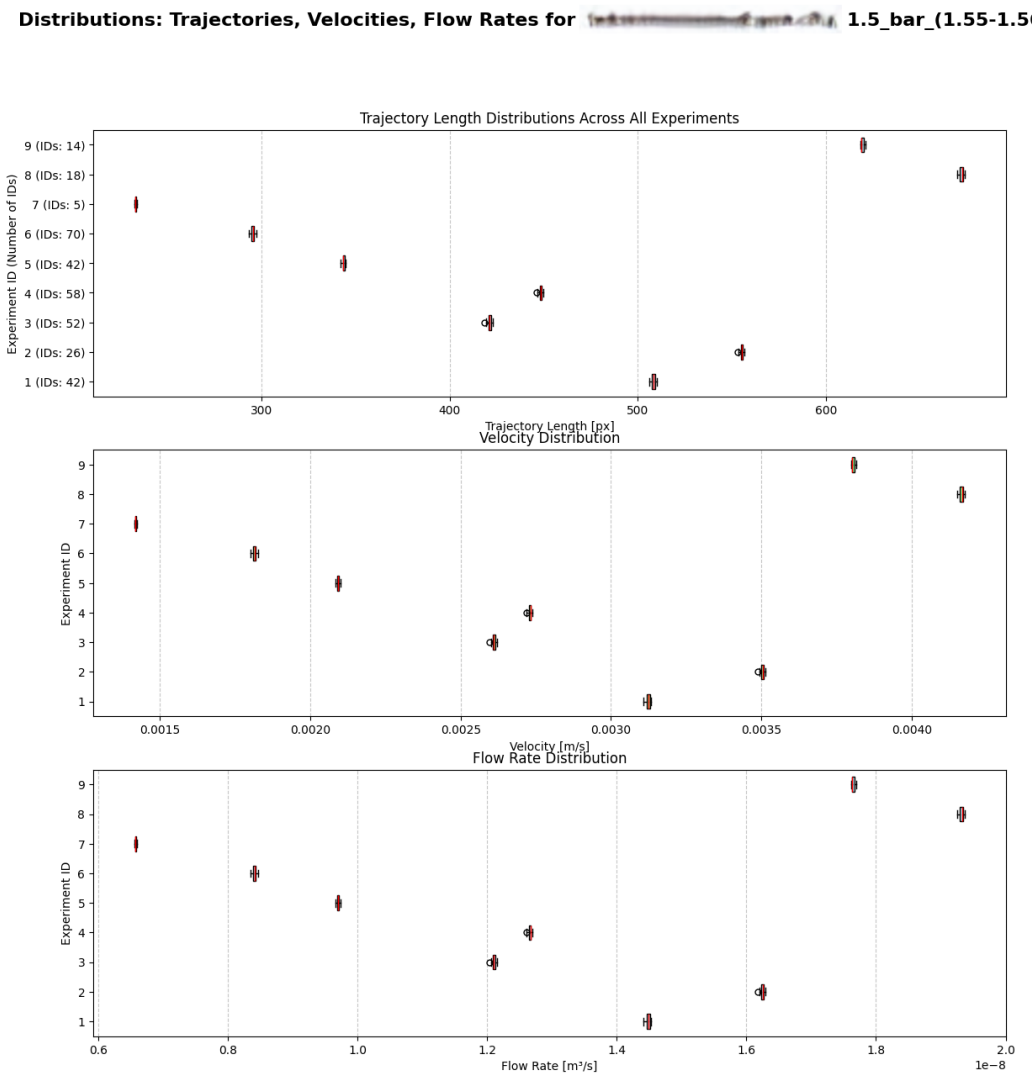


Figure 12: **Velocity and Flow Rates.** Calculated velocities and flow rates for each experiment.

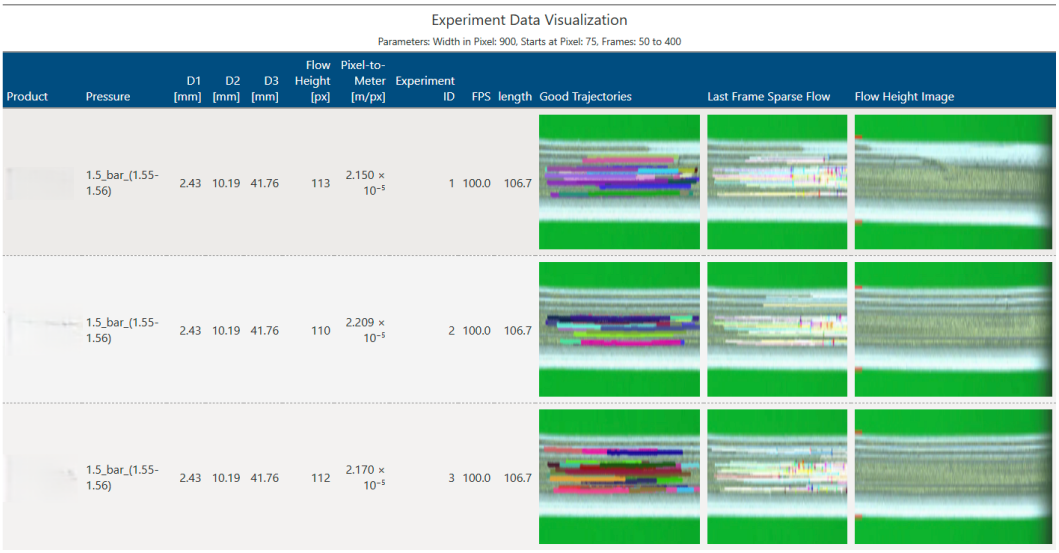


Figure 13: **Comprehensive Report.** Consolidated trajectory visualizations, last-frame snapshots, and cropped flow images.

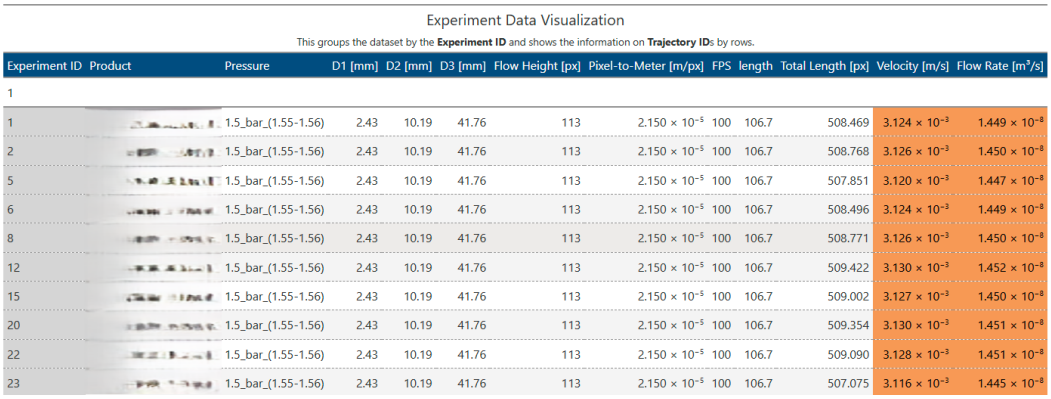


Figure 14: **Grouped Experimental Data.** Calculated metrics grouped by experiment number.

B Optical Flow Algorithms

Optical flow is the pattern of apparent motion of objects, surfaces, and edges in a visual scene caused by the relative motion between an observer and the scene[15]. It represents the distribution of apparent velocities of movement of brightness patterns in an image [15].

Optical flow quantifies the motion of objects between consecutive frames captured by a camera, attempting to capture the apparent motion of brightness patterns. This concept was introduced by psychologist James J. Gibson in the 1940s to describe the visual stimulus provided to animals moving through the world [15].

The optical flow can be mathematically expressed through the optical flow equation:

$$I_x v_x + I_y v_y + I_t = 0 \tag{8}$$

where I_x and I_y are spatial gradients, I_t is the temporal gradient, and v_x and v_y are flow velocities in the x and y directions.
Optical flow estimation is broadly categorized into:

- **Machine Learning-Based Models**: Machine Learning-based optical flow models learn motion patterns from data rather than relying on handcrafted constraints like brightness constancy or smoothness. These models typically use deep neural networks trained on large optical flow datasets such as FlyingChairs [6], Sintel [4], and KITTI [8].
- **Classical Models**: These methods rely purely on mathematical formulations, physics-based constraints, and variational principles without using machine learning. They typically solve an optimization problem based on brightness constancy and spatial smoothness assumptions.
- **Hybrid Models**: Hybrid optical flow models combine classical techniques (e.g., Lucas-Kanade, Farneback) with machine learning to enhance flow estimation. These models either precompute classical flow and use it as a feature for ML models or refine classical flow estimates using deep learning. Additionally, they can incorporate physical constraints like brightness constancy and smoothness to stabilize the predictions. Examples include DeepFlow [25], SpyNet [20], and learning-based refinements for Horn-Schunck [22].

In this study, we investigated classical and hybrid models to analyze their distinct operational principles and explore how these insights could benefit our counterparts in developing a proof of concept for their applications.

B.1 Sparse Optical Flow

Sparse optical flow is a technique used to estimate motion between consecutive frames in a video sequence by analyzing a limited number of feature points. This method focuses on tracking the movement of specific, interesting pixels within an image, typically edges or corners, rather than processing the entire frame [**<empty citation>**].
In this section, we will conduct an in-depth exploration of sparse optical flow methods, specifically the Lucas-Kanade and RLOF algorithms, as they are relevant to our work.

Lucas-Kanade Method The Lucas-Kanade method is one of the most widely used classical optical flow algorithms, primarily for small-motion tracking in image sequences [13]. It assumes that the optical flow is constant in a local window around each pixel, and it estimates the flow by solving the optical flow equations over small regions (usually 3x3 or 5x5 pixel windows) [13]. The method works under the assumption that:

- **Brightness constancy**: The intensity of a pixel in one frame is the same as in the next (i.e., no illumination changes).

- **Small motion**: The movement of objects between consecutive frames is small enough for a linear approximation.

Lucas-Kanade derives the optical flow by solving the following system of equations using the least-squares method:

$$I_x u + I_y v = -I_t \tag{9}$$

Where:

- I_x, I_y, I_t are the image gradients in the x, y, and time directions, respectively.
- u, v are the components of the optical flow (motion) in the x and y directions.

This equation states that the rate of change in image intensity is zero when considering the motion of an object between two frames.

RLOF Robust Local Optical Flow (RLOF) [21] is an extension of the classical Lucas-Kanade method that enhances robustness and accuracy in computing optical flow locally. It focuses on estimating motion in a sparse or dense manner while improving resilience to noise, illumination changes, and occlusions.
RLOF is designed to address the limitations of traditional local methods by incorporating:

- Adaptive window selection
- Robust feature matching
- Error minimization strategies
- Multi-scale and pyramid-based refinements

Given an image sequence $I(x, y, t)$, optical flow assumes that the intensity of pixels remains constant over time:

$$I(x, y, t) = I(x + u, y + v, t + \Delta t) \tag{10}$$

where:

- (x, y) is the pixel position,
- (u, v) is the flow vector (velocity) at that pixel,
- t represents the time dimension.

Applying the Taylor Series Expansion and ignoring higher-order terms gives the Optical Flow Constraint Equation (OFCE):

$$I_x u + I_y v + I_t = 0$$

where:

- I_x, I_y are spatial gradients,
- I_t is the temporal gradient,
- (u, v) is the unknown optical flow.

Since a single equation cannot solve for two unknowns (u, v), local optical flow methods like Lucas-Kanade introduce additional constraints by assuming that motion is constant in a small local window around each pixel.
The standard Lucas-Kanade method solves this by minimizing:

$$\sum_{i \in \Omega} w_i \left(I_{x_i} u + I_{y_i} v + I_{t_i}\right)^2 \tag{11}$$

where w_i is a weight function emphasizing central pixels.
RLOF builds on this approach by introducing robust statistical techniques and adaptive windowing to improve flow estimation.

B.2 Dense Optical Flow

Dense optical flow is a computer vision technique that estimates the motion of every pixel between two consecutive frames in a video sequence [**<empty citation>**]. Unlike sparse optical flow methods that track only specific features, dense optical flow calculates motion vectors for all pixels, providing a comprehensive representation of movement within an image [**<empty citation>**].
In this section, we will conduct an in-depth exploration of dense optical flow methods, specifically the Farnebaeck and Horn-Schunck algorithms, as they are relevant to our work.

Farnebaeck Method Gunnar Farnebaeck introduced his method for two-frame motion estimation based on polynomial expansion in 2003 [7]. The core idea of this algorithm is to approximate the local neighborhoods of both frames using quadratic polynomials and then estimate the displacement field by observing how these polynomials transform between frames [7].
Each small image patch is represented as a quadratic polynomial using a Taylor series expansion:

$$f(x) = x^T A x + b^T x + c \tag{12}$$

where:

- A is a symmetric matrix that captures second-order variations (curvature).
- b is a vector representing first-order changes (gradient).
- c is a scalar constant term.

This method is particularly useful for estimating motion when there is no temporal consistency between frames, such as in cases where a camera is affected by high-frequency vibrations. The algorithm has found applications in various fields, including computer vision, medical imaging, and video processing.

Horn-Schunck Method The Horn-Schunck method is a foundational variational approach for optical flow estimation, introduced to address motion tracking between consecutive image frames by combining brightness constancy with spatial smoothness constraints. Given two consecutive frames $I(x, y, t)$ and $I(x, y, t + 1)$, the optical flow (u, v) can be estimated by solving the following optimization problem:

$$\min_{u,v} \int \int \left((I_x u + I_y v + I_t)^2 + \alpha^2 \left((\nabla u)^2 + (\nabla v)^2 \right) \right) dx\, dy$$

Where:

- I_x, I_y, and I_t are the partial derivatives of the image I with respect to x, y, and t, respectively.
- α is a regularization parameter that controls the smoothness term.
- ∇^2 is the Laplacian operator applied to the flow components u and v.

The Horn-Schunck method is a key technique for estimating motion between video frames. Combining the assumption that pixel brightness stays the same and the idea that motion is smooth, it provides a reliable way to estimate how pixels move. While it has some limitations, it has influenced many modern optical flow methods and is still widely used in computer vision.

C Overview of involved Tensors

Before proceeding to the detailed analysis of the Navier-Stokes equations incorporating the power-law model, we first provide a theoretical explanation of the fundamental quantities involved in the formulation. In the following sections, we review the following topics:

- **Background on the Stress Tensor:** An explanation of the stress tensor, its decomposition into hydrostatic and deviatoric components, and its role in describing internal forces within the fluid.
- **Background on the Strain Rate Tensor:** Discuss the strain rate tensor, its definition as the symmetric part of the velocity gradient, and its significance in quantifying the deformation rate.

This theoretical framework lays the foundation for our subsequent analysis, where we reintroduce the Navier-Stokes equations and apply the appropriate rheological model (specifically, the power-law formulation) to capture the non-Newtonian characteristics of the fluid flow.

C.1 Background on the Stress Tensor

In continuum mechanics, the stress state at a point within a material is described by the stress tensor, $\boldsymbol{\sigma}$. For a three-dimensional body, the stress tensor is a symmetric 3×3 matrix:

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_{33} \end{pmatrix}.$$

The diagonal components σ_{11}, σ_{22}, and σ_{33} represent normal stresses acting perpendicular to the coordinate surfaces, while the off-diagonal components represent shear stresses. In many physical situations, especially when dealing with fluids or ductile solids, it is useful to distinguish between the part of the stress that tends to change the volume of a material (volumetric or hydrostatic stress) and the part that tends to distort its shape (deviatoric stress).

C.1.1 Decomposition into Hydrostatic and Deviatoric Components

The total stress tensor $\boldsymbol{\sigma}$ can be split into two distinct parts:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}^{\text{hydrostatic}} + \boldsymbol{\sigma}^{\text{deviatoric}}.$$

Hydrostatic Stress. The hydrostatic (or spherical) component represents the isotropic pressure acting uniformly in all directions. It is defined as:

$$\boldsymbol{\sigma}^{\text{hydrostatic}} = -p\,\mathbf{I},$$

where:

- p is the pressure, defined in terms of the trace of the stress tensor by

$$p = -\frac{1}{3}\operatorname{tr}(\boldsymbol{\sigma}) = -\frac{1}{3}(\sigma_{11} + \sigma_{22} + \sigma_{33}),$$

- $\mathbf{I}$ is the identity tensor.

Deviatoric Stress. The deviatoric stress tensor, denoted by $\boldsymbol{\tau}$, is obtained by subtracting the hydrostatic part from the total stress:

$$\boldsymbol{\tau} = \boldsymbol{\sigma} - \boldsymbol{\sigma}^{\text{hydrostatic}} = \boldsymbol{\sigma} + p\,\mathbf{I}.$$

In component form, this is written as:

$$\tau_{ij} = \sigma_{ij} - \frac{1}{3}\sigma_{kk}\delta_{ij},$$

where δ_{ij} is the Kronecker delta. An important property of the deviatoric stress tensor is that its trace is zero:

$$\operatorname{tr}(\boldsymbol{\tau}) = \tau_{11} + \tau_{22} + \tau_{33} = 0.$$

This confirms that $\boldsymbol{\tau}$ represents only the distortional (shape-changing) aspects of the stress state.

C.1.2 Physical Interpretation of the Deviatoric Stress Tensor

The decomposition of the stress tensor into hydrostatic and deviatoric components allows us to understand different aspects of the material's response under load:

Volumetric Changes versus Distortional Effects.

- The *hydrostatic stress* (or pressure) is associated with changes in volume. When a material is subjected solely to hydrostatic stress, it experiences uniform expansion or compression without any change in shape.
- The *deviatoric stress* is responsible for shear and distortion. It drives the change in the shape of a material element while keeping its volume constant. This is particularly important in plasticity and yield criteria, where the material's response depends on its ability to undergo shear deformations.

Application to Low Reynolds Number Flows. In the context of the Navier-Stokes equations discussed earlier, the deviatoric stress tensor $\boldsymbol{\tau}$ encapsulates the viscous stresses acting on the fluid. For flows characterized by low Reynolds numbers, inertial forces are negligible compared to viscous forces, and the flow behavior is primarily governed by the balance between pressure gradients and the divergence of the deviatoric stress tensor.

C.2 Background on the Strain Rate Tensor

In continuum mechanics, the strain rate tensor measures the rate at which a material element deforms over time. While the stress tensor describes the internal forces acting within a material, the strain rate tensor quantifies the material's change in deformation (or strain). This tensor is fundamental in fluid mechanics and the study of viscous flows, where the deformation rate directly influences the fluid's dissipative forces.

Definition. For a fluid with a velocity field $\mathbf{v}$, the strain rate tensor, commonly denoted as $\mathbf{D}$, is defined as the symmetric part of the velocity gradient:

$$\mathbf{D} = \frac{1}{2}\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^\top\right). \tag{13}$$

Here, $\nabla\mathbf{v}$ is the gradient of the velocity field and $(\nabla\mathbf{v})^\top$ is its transpose. This definition ensures that $\mathbf{D}$ is symmetric, meaning that its off-diagonal elements are equal. The symmetry of $\mathbf{D}$ guarantees that it represents only the rate of deformation (i.e., stretching and shearing) and does not include contributions from rigid body rotations.

Physical Interpretation. The strain rate tensor captures two main types of deformation:

- **Normal Deformation:** The diagonal components of $\mathbf{D}$ represent the rates of stretching or compression along the principal coordinate directions. These components describe how the length of a material element changes over time.

- **Shear Deformation:** The off-diagonal components of **D** represent the rates of shear deformation, which change the shape of a material element without necessarily changing its volume.

Because **D** is derived from the velocity gradients, it provides insight into the local flow behavior. In viscous fluids, for instance, the magnitude of the strain rate tensor directly influences the viscous stresses, as seen in constitutive equations that relate stress to strain rate.

Mathematical Properties.

- **Symmetry:** By construction, **D** is symmetric. This property simplifies the deformation analysis by decoupling pure deformation from rigid body rotation.
- **Invariance:** The strain rate tensor is invariant under coordinate transformations, meaning its physical interpretation does not depend on the chosen coordinate system.
- **Relation to Material Behavior:** In Newtonian fluids, the viscous (deviatoric) stress is directly proportional to the strain rate tensor, reflecting that viscous forces oppose the deformation rate. For non-Newtonian fluids, the relationship may be more complex, often involving an effective viscosity that depends on the magnitude of the strain rate.

In summary, the strain rate tensor is a fundamental quantity in both fluid and solid mechanics that quantifies how the shape of a material changes over time. Its symmetric nature allows for a clear distinction between deformation and rigid body motions, making it essential for developing constitutive models and analyzing material behavior under flow.

D From Navier-Stokes EquPower-Law Constants

In this section, we briefly outline the derivation of the boundary value problem governing the axially symmetric, quasi-static radial flow of the Fluid and present the associated pressure formulation and connection to power-law rheological constants [19]. Although the complete derivation involves several technical steps, the key ideas are as follows:

1. Constitutive Model for the Fluid The Fluid is modeled by a power-law constitutive relation that captures both the coaxiality between the stress and strain rate tensors and strain rate hardening:

$$\boldsymbol{S} = \sqrt{\frac{2}{3}}\,\sigma_e\,\frac{\boldsymbol{D}}{\sqrt{\boldsymbol{D}\cdot\boldsymbol{D}}}, \qquad \sigma_e = \sigma_0\left(\frac{2}{3}\,\boldsymbol{D}\cdots\boldsymbol{D}\right)^{\frac{1}{2n}},$$

where:

- $\boldsymbol{S}$ is the stress deviator tensor,
- $\sigma_e = \left(\frac{3}{2}\,\boldsymbol{S}\cdot\boldsymbol{S}\right)^{1/2}$ is the Mises effective stress,

- $\boldsymbol{D}$ is the Eulerian strain rate tensor (the symmetric part of the velocity gradient),
- σ_0 and n are material parameters (with $n = 1$ corresponding to a Newtonian fluid and $n \to \infty$ to a perfectly plastic solid).

(Notice that the power index n here is the inverse of n_0 often used in fluid mechanics literature.)

2. Assumptions on the Flow Field The flow is assumed to be radially directed toward a virtual apex O. Placing the origin of a spherical coordinate system (r, θ, φ) at O with unit vectors $\boldsymbol{e}_r$, $\boldsymbol{e}_\theta$, and $\boldsymbol{e}_\varphi$, the velocity field is taken as

$$\boldsymbol{V} = V\,\boldsymbol{e}_r, \qquad \text{with} \quad V = -\frac{Q\,f(\theta)}{r^2},$$

where Q is the steady-state volumetric flow rate and $f(\theta)$ is an unknown angular function (normalized appropriately). In the following, the prime (e.g. $f'(\theta)$) denotes differentiation with respect to θ.

3. Formulation of the Strain Rate Tensor Given the radial flow, the Eulerian strain rate tensor is

$$\boldsymbol{D} = Q\left[\frac{f(\theta)}{r^3}\Big(2\,\boldsymbol{e}_r\boldsymbol{e}_r - \boldsymbol{e}_\theta\boldsymbol{e}_\theta - \boldsymbol{e}_\varphi\boldsymbol{e}_\varphi\Big) - \frac{f'(\theta)}{2r^3}\Big(\boldsymbol{e}_r\boldsymbol{e}_\theta + \boldsymbol{e}_\theta\boldsymbol{e}_r\Big)\right].$$

4. Derivation of Normalized Stress Components Substituting $\boldsymbol{D}$ into the constitutive relation and normalizing with respect to $\frac{2^{1/n}\sigma_0}{\sqrt{3}}$ leads to the scalar relations:

$$\Sigma_r - \Sigma_\theta = \sqrt{3}\,Q^{1/n}\,F\,r^{-3/n},$$
$$\Sigma_{r\theta} = -\beta\,Q^{1/n}\,r^{-3/n},$$

where the normalized stress components are Σ_r, Σ_θ, and $\Sigma_{r\theta}$, and

$$F = f^{1/n}\,\Delta^{\frac{1-n}{n}}, \qquad \Delta = \sqrt{1+\beta^2}, \qquad \beta = \left(\frac{\sqrt{3}}{6}\right)\frac{f'(\theta)}{f(\theta)}.$$

The normalized Mises stress is then

$$\Sigma = \frac{\sigma_e}{\frac{2^{1/n}\sigma_0}{\sqrt{3}}} = \sqrt{3}\,Q^{1/n}\,\Delta\,F\,r^{-3/n}.$$

5. Equilibrium and Integration In the absence of inertia, the radial equilibrium equation

$$r\,\frac{\partial\Sigma_r}{\partial r} + \frac{\partial\Sigma_{r\theta}}{\partial\theta} + 2\,(\Sigma_r - \Sigma_\theta) + \Sigma_{r\theta}\cot\theta = 0$$

(with a similar equation in the circumferential direction) leads—after integrating over r—to the expressions

$$\Sigma_r = -Q^{1/n}\left(\frac{n}{3}\right)\Big[(\beta F)' + (\beta F)\cot\theta - 2\sqrt{3}\,F\Big]\,r^{-3/n} + H(\theta),$$

$$\Sigma_\theta = -Q^{1/n}\left\{\left(\frac{n}{3}\right)\left[(\beta F)' + (\beta F)\cot\theta\right] - \left(\frac{2n-3}{\sqrt{3}}\right)F\right\} r^{-3/n} + H(\theta),$$

where $H(\theta)$ is determined to be a constant C from the equilibrium conditions.

6. Final Boundary Value Problem and Pressure Formulation The derivation reduces to the following boundary value problem for $f(\theta)$:

$$\left\{\left(\frac{n}{3}\right)\left[(\beta F)' + (\beta F)\cot\theta\right] - \left(\frac{2n-3}{\sqrt{3}}\right)F\right\}' + \frac{3(n-1)}{n}(\beta F) = 0,$$

with

$$\beta = \left(\frac{\sqrt{3}}{6}\right)\frac{f'(\theta)}{f(\theta)}, \quad F = f^{1/n}\,\Delta^{\frac{1-n}{n}}, \quad \Delta = \sqrt{1+\beta^2}.$$

The boundary conditions are:

- $\beta = 0$ at $\theta = 0$ (ensuring vanishing shear stress on the axis),
- $\beta = -m\,(1-m^2)^{-1/2}$ at $\theta = \alpha$ (incorporating wall slip with friction factor m),
- Normalization of the velocity profile:

$$\int_0^\alpha f(\theta)\sin\theta\,d\theta = \frac{1}{2\pi}.$$

The integration constant $H(\theta) = C$ is determined from the inlet/outlet pressure data. The normalized hydrostatic pressure (with $\Sigma_p = \Sigma_\theta$) is given by

$$P = Q^{1/n}\left\{\frac{n}{3}\left[(\beta F)' + (\beta F)\cot\theta\right] - \frac{2(n-1)}{\sqrt{3}}F\right\} r^{-3/n} - C,$$

and its cross-sectional average over a spherical surface is

$$\bar{P} = -Q^{1/n}\,\bar{p}(\alpha, m, n)\,r^{-3/n} - C,$$

with

$$\bar{p}(\alpha, m, n) = \frac{-\frac{n}{3}\,(\beta F)_{\theta=\alpha}\sin\alpha + \frac{2(n-1)}{\sqrt{3}}\int_0^\alpha F\sin\theta\,d\theta}{1-\cos\alpha}.$$

This concise formulation, which encompasses the boundary value problem and the pressure equations, forms the basis for our numerical solution of the flow behavior.

7. Mapping to Power-Law Parameters To ensure consistency with the experimentally measured one-dimensional power-law behavior, we relate the exponents and the stress scale as follows:

$$n = \frac{1}{n_0}, \quad \sigma_0 = 3^{\frac{n_0+1}{2}} K$$

These relations convert the rheological parameters obtained from shear flow experiments (using, for instance, cone-and-plate or parallel-plate geometries) into the parameters used in our tensorial formulation. In this way, the three-dimensional constitutive model accurately reflects the shear-thinning or shear-thickening behavior observed experimentally and provides a robust basis for modeling the complex flow behavior of Fluids.
This framework forms the basis for our further investigations, where we will analyze the flow behavior by solving the boundary value problem and incorporating the power-law rheology into the Navier–Stokes equations.

E Clustering

Here, we present the clustering with the corresponding product.

Product	**K_value**	**n_value**	**Cluster**
Product 1	1,592.74	-0.01	Cluster 1
Product 2	1,306.39	0.14	Cluster 1
Product 3	892.35	0.06	Cluster 1
Product 4	5,348.62	0.12	Cluster 1
Product 5	2,034.91	0.01	Cluster 1
Product 6	1,153.53	0.21	Cluster 2
Product 7	350.21	0.30	Cluster 2
Product 8	464.64	0.24	Cluster 2
Product 9	1,151.07	0.50	Cluster 3
Product 10	562.51	0.44	Cluster 3
Product 11	551.19	0.46	Cluster 3
Product 12	597.82	0.48	Cluster 3
Product 13	907.53	0.75	Cluster 4
Product 14	20.95	0.93	Cluster 4

Table 9: Clustering of Products by *n_value*

The products are grouped into four clusters based on their *n_value*:

- **Cluster 1** (Very Low/Negative *n_value*): Includes products with *n_value* ≤ 0.14.
- **Cluster 2** (Low *n_value*): Includes products with *n_value* between 0.21 and 0.30.
- **Cluster 3** (Medium *n_value*): Includes products with *n_value* between 0.44 and 0.51.
- **Cluster 4** (High *n_value*): Includes products with *n_value* ≥ 0.72.

This clustering helps identify patterns in product performance based on the *n_value* metric.

F Technical Implementation

In this project, we developed a viscosity prediction interface with a backend written in Python and a frontend built using Streamlit. The goal was to create an intuitive and functional tool that would predict the viscosity of adhesives, focusing on delivering an effective user experience for laboratory settings. Given the constraints of the lab environment, where the computer is cut off from the internet and operates under strict conditions, we had to carefully design the entire system to ensure its proper functionality under these constraints.
Throughout the project, we utilized an iterative approach, breaking the development process into multiple sprints, each focused on implementing a set of features. The entire lifecycle spanned three different software versions, validated and refined in collaboration with the team.

Frameworks and Tools Used

- **Backend: Python**
  Python was chosen for the backend due to its extensive libraries and ease of integration with scientific tools and data processing algorithms. Libraries for numerical and statistical analysis (e.g., NumPy, SciPy) were crucial in building the viscosity prediction models.

- **Frontend: Streamlit**
  Streamlit was selected for the frontend due to its simplicity and ability to create interactive web interfaces quickly. Streamlit's lightweight nature allowed us to develop the interface without requiring extensive web development resources, making it ideal for lab settings where simplicity and efficiency are paramount.

- **Lab Computer Constraints**
  The lab computer being cut off from the internet required us to develop the solution so that all necessary dependencies and functionalities were self-contained within the system. This meant the application had to be fully offline, with all libraries preinstalled and no external API calls or internet connections used. The operating system and hardware were older, requiring extra attention to system compatibility, ensuring that Python and Streamlit ran smoothly in the limited environment.

Development Approach: Agile Methodology with Iterative Sprints

- **Initial Version (Core Functionality)**

- Built a Python-based backend for viscosity predictions and a basic Streamlit interface with input fields and a prediction button.

- Validated the algorithm's functionality with the chemist team.

- **Feature Expansion (Enhanced UI)**
- Improved the user interface with an export function and added more input fields for adhesive parameters.
- Refined the backend with advanced libraries to enhance prediction accuracy.
- Validated usability and performance through team feedback.
- **Final Version (Polished and Validated)**
- Polished the interface based on feedback, optimized the prediction algorithm with a learning-based approach, and finalized features for seamless data upload and fast predictions.
- Conducted rigorous validation tests to ensure accuracy and alignment with lab requirements.

Software Environment The primary libraries used in our pipeline include:

- **Optical Flow:** OpenCV, providing Farneback and Lucas-Kanade algorithms.
- **Deep Learning:** PyTorch for CNN-based models and training routines.
- **Data Processing:** NumPy, SciPy, and other standard scientific computing libraries.

Every feature we developed got integrated through a CI/CD Azure pipelines in Azure DevOps. The Azure Pipelines YAML defines a CI/CD pipeline to package a Python application using **Poetry** and **cxFreeze**. The key steps are:

1. **Trigger:** Runs on the `main` branch.
2. **Setup:** Uses *Windows-latest*, installs Python 3.11, Poetry, and dependencies.
3. **Build:** Activates a virtual environment, installs dependencies, and locks versions.
4. **Packaging:** Uses **cxFreeze** to bundle the application and copies required files (`app`, `config`, `.streamlit`).
5. **Artifact Publishing:** Uploads the packaged application as a build artifact for deployment.

G Autoencoder Hyperparameter Tuning

To ensure the quality of extracted visual features, we fine-tuned different autoencoder configurations and evaluated their performance using the following metrics:

- **Structural Similarity Index Measure (SSIM)**: Measures the similarity between two images by considering luminance.
- **Peak Signal-to-Noise Ratio (PSNR)**: Quantifies the ratio between the maximum possible power of a signal and the power of corrupting noise.
- **Mean Squared Error (MSE)**: The average squared difference between the original and reconstructed images.

We trained three models that differed in terms of network depth and latent feature dimensionality. The evaluation results are presented in Fig. 15. Based on these findings, we selected the autoencoder with the following training settings for downstream tasks: a batch size 16, a training time of 500 epochs, a learning rate of 0.001, a 256-dimensional latent space, and 3 network layers.

H Viscosity Prediction Results

Table 10: Performance on Fluid Consistency Index Prediction k

Model	MAE (↓)	RMSE (↓)	Weighted MAE (↓)	RMAE (↓)	R^2 (↑)
VisOF	120.7428	166.8652	26.5374	0.0737	0.9921
VisOF-L	116.7576	156.6515	24.3545	0.0676	0.9928
VisOF-FT-AutoE	110.1824	164.9900	34.3255	0.0953	0.9919
VisOF-FT-LAutoE	156.8333	254.1645	27.5309	0.0765	0.9771
Vis	**20.9847**	**48.4447**	**6.0065**	**0.0167**	**0.9993**
Vis-L	126.2796	169.1786	25.1202	0.0698	0.9920
Vis-FT-AutoE	155.8631	251.4273	21.7197	0.0603	0.9787
Vis-FT-LAutoE	178.4680	258.2319	29.6797	0.0824	0.9785
Vis-Freeze-AutoE	299.2371	467.1099	56.3451	0.1565	0.9249
Vis-FT-Classifier	56.5229	82.0041	13.6864	0.0380	0.9980

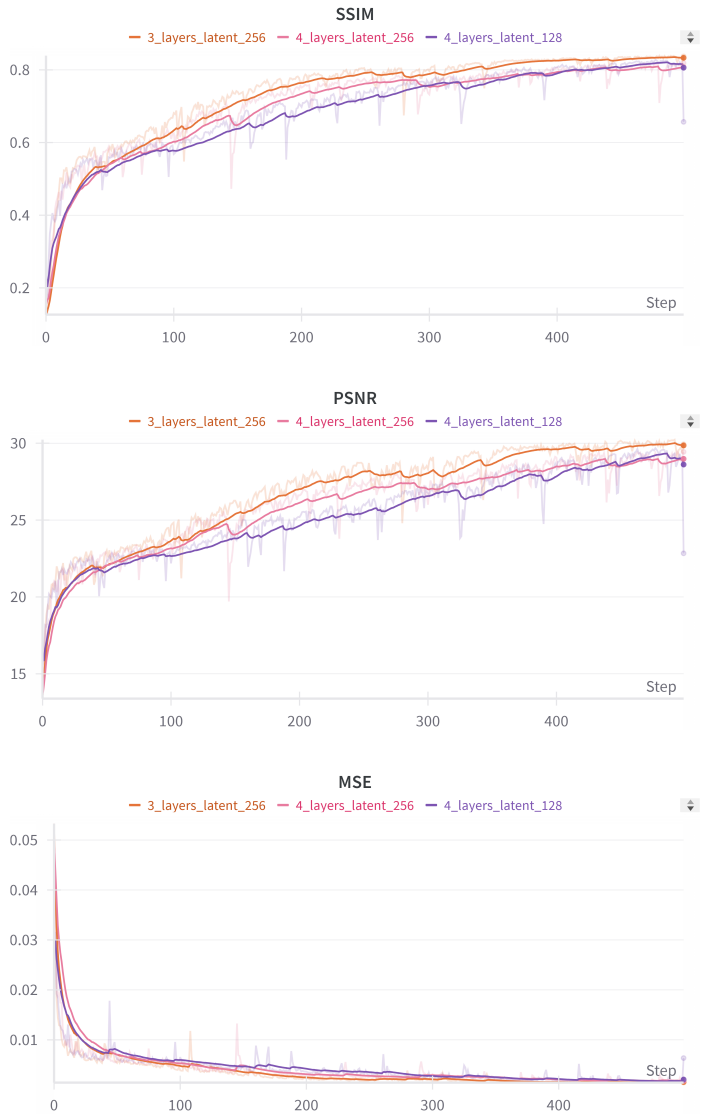
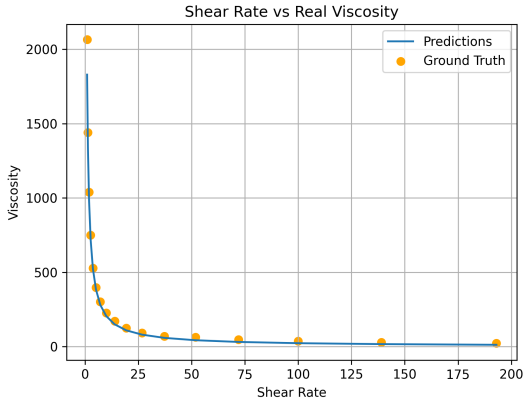
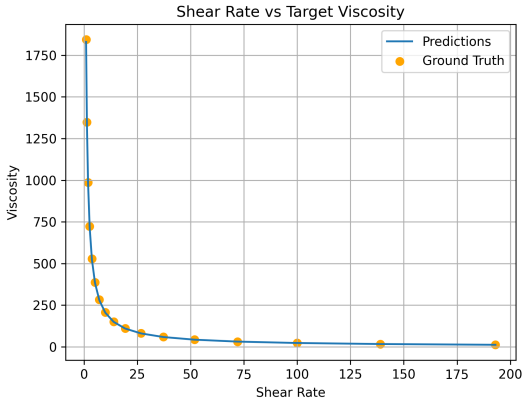


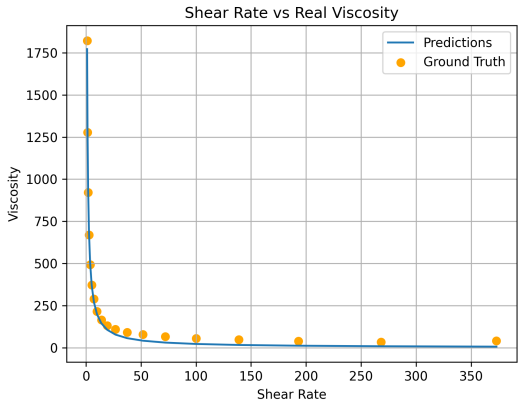
Figure 15: The fine-tune evaluation of the autoencoder. The model with 3 layers is optimal across all metrics.



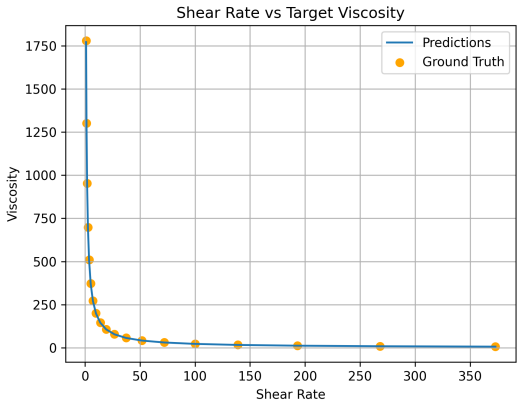
(a) Measured real viscosity for product B



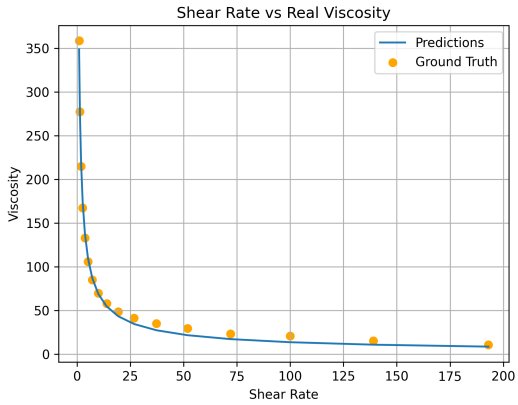
(b) Measured target viscosity for product B



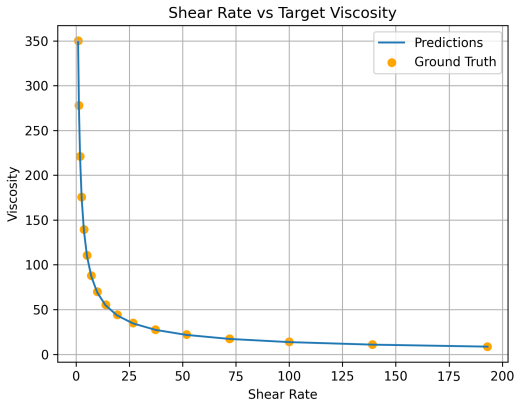
(c) Measured real viscosity for product C



(d) Measured target viscosity for product C

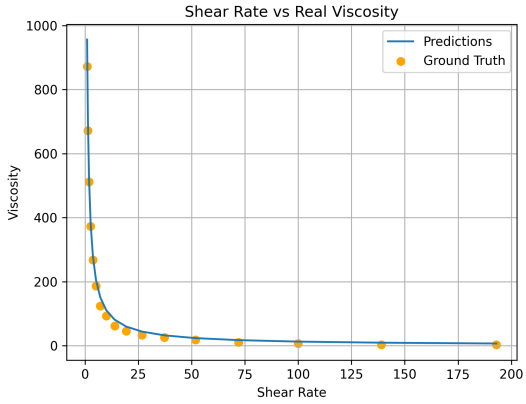


(e) Measured real viscosity for product D

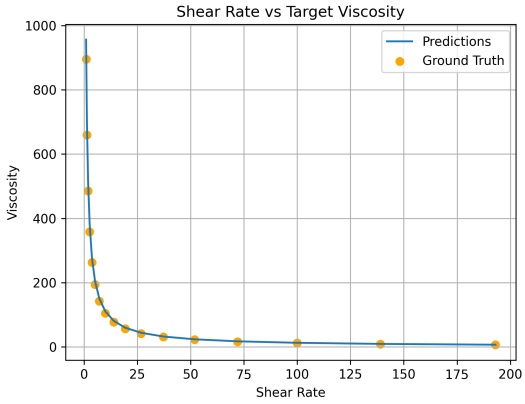


(f) Measured target viscosity for product D

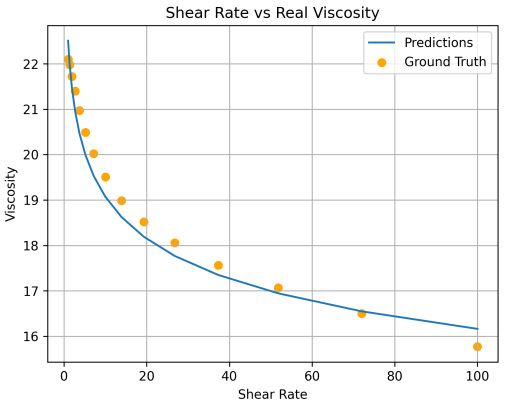
Figure 16: Visual Model Comparison of Shear Rate vs. Viscosity Part 1



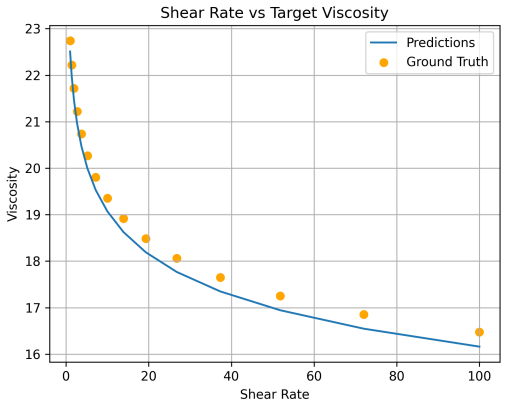
(a) Measured real viscosity for product E



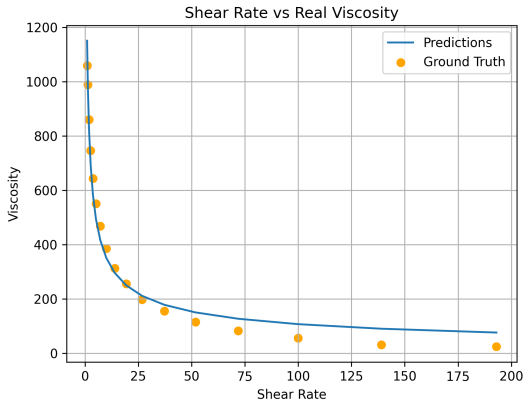
(b) Measured target viscosity for product E



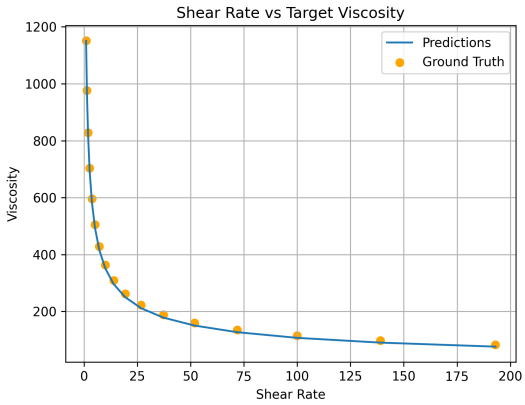
(c) Measured real viscosity for product F



(d) Measured target viscosity for product F

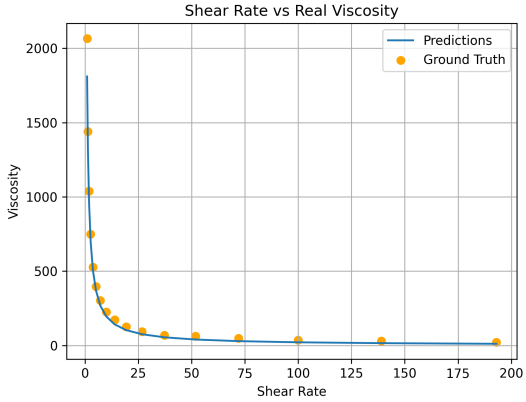


(e) Measured real viscosity for product G

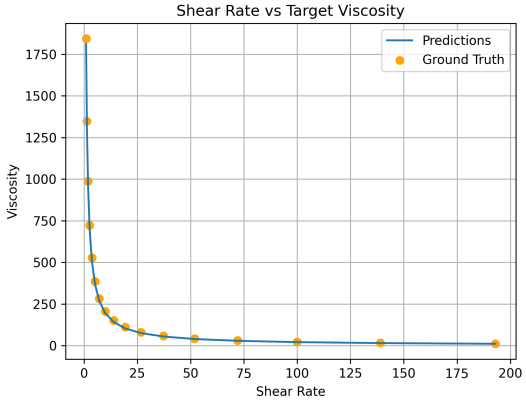


(f) Measured target viscosity for product G

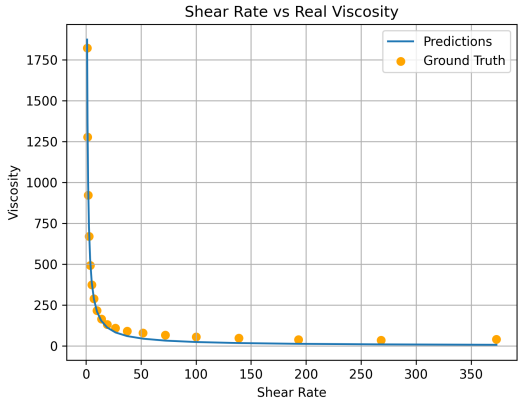
Figure 17: Visual Model Comparison of Shear Rate vs. Viscosity - Part 2



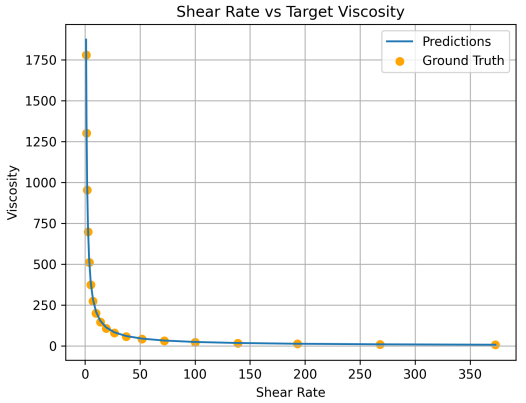
(a) Measured real viscosity for product B



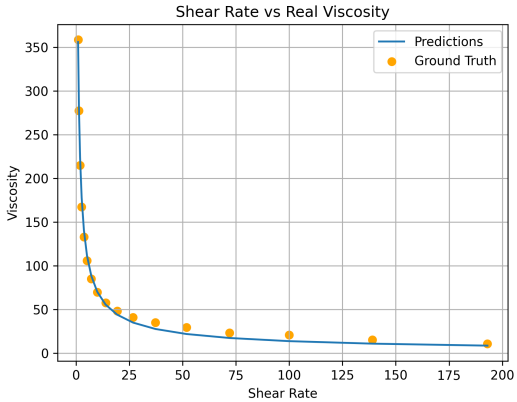
(b) Measured target viscosity for product B



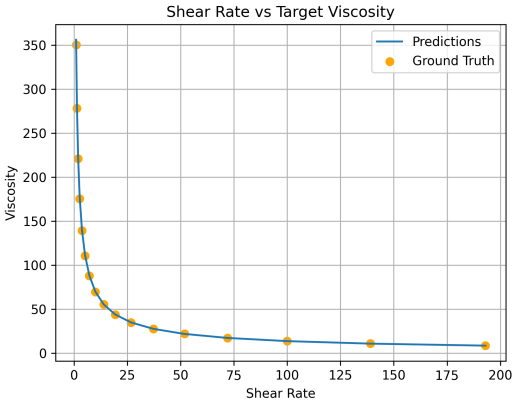
(c) Measured real viscosity for product C



(d) Measured target viscosity for product C

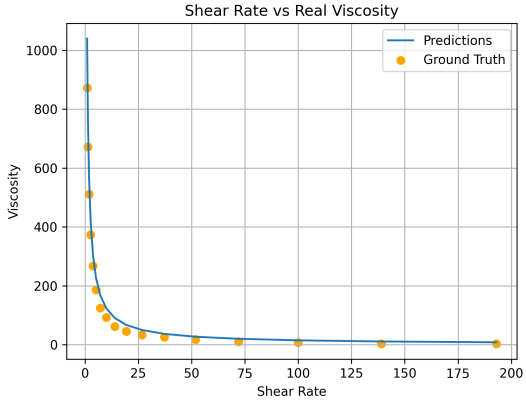


(e) Measured real viscosity for product D

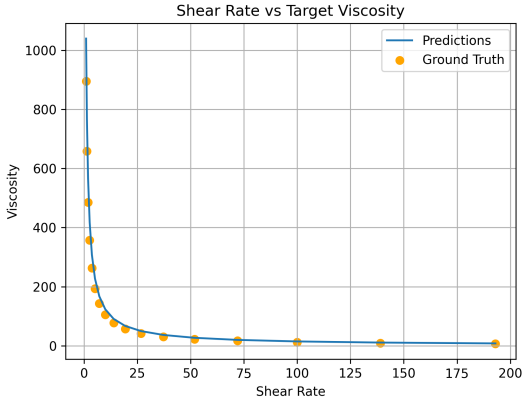


(f) Measured target viscosity for product D

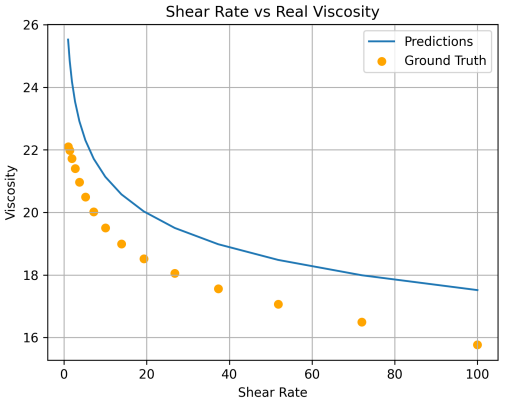
Figure 18: Visual with Optical Flow Model Comparison of Shear Rate vs. Viscosity - Part 1



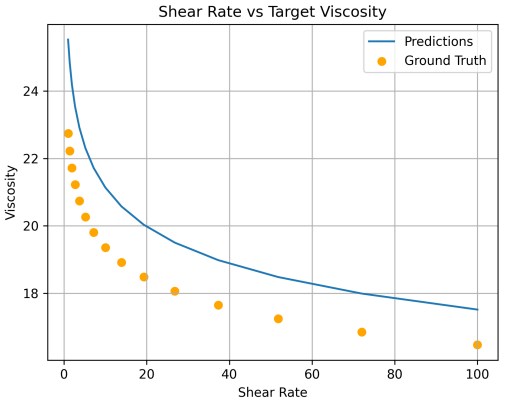
(a) Measured real viscosity for product E



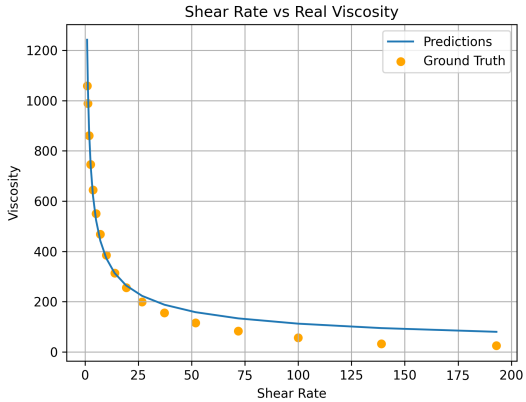
(b) Measured target viscosity for product E



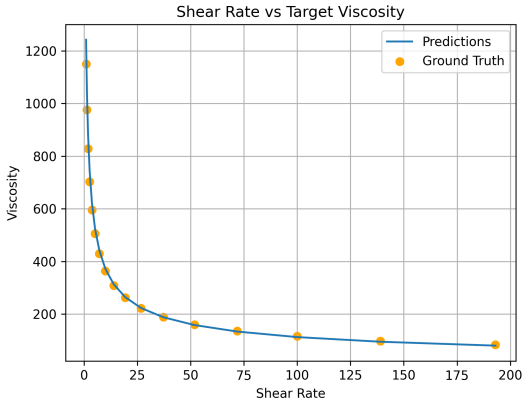
(c) Measured real viscosity for product F



(d) Measured target viscosity for product F



(e) Measured real viscosity for product G



(f) Measured target viscosity for product G

Figure 19: Visual with Optical Flow Model Comparison of Shear Rate vs. Viscosity - Part 2

Table 11: Performance on Fluid Behaviour Index Prediction n

Model	MAE (↓)	RMSE (↓)	Weighted MAE (↓)	RMAE (↓)	R^2 (↑)
VisOF	0.0200	0.0275	0.0240	0.2902	0.9805
VisOF-L	0.0143	0.0204	0.0159	0.1917	0.9890
VisOF-FT-AutoE	0.0113	0.0144	0.0116	0.1407	0.9950
VisOF-FT-LAutoE	0.0128	0.0226	0.0133	0.1606	0.9872
Vis	**0.0027**	**0.0042**	**0.0021**	**0.0254**	**0.9996**
Vis-L	0.0208	0.0321	0.0173	0.2096	0.9730
Vis-FT-AutoE	0.0243	0.0348	0.0273	0.3298	0.9705
Vis-FT-LAutoE	0.0455	0.0739	0.0495	0.5979	0.8584
Vis-Freeze-AutoE	0.0316	0.0417	0.0335	0.4051	0.9588
Vis-FT-Classifier	0.0105	0.0156	0.0107	0.1295	0.9935

Table 12: Performance on Real Viscosity Estimation

Model	MAE (↓)	RMSE (↓)	Weighted MAE (↓)	RMAE (↓)	R^2 (↑)
VisOF	37.2969	68.7504	11.5314	0.2490	0.9940
VisOF-L	37.7963	70.9316	11.2892	0.2438	0.9933
VVisOF-FT-AutoE	34.8666	66.7378	12.5968	0.2720	0.9942
VisOF-FT-LAutoE	47.2416	101.1618	11.0760	0.2392	0.9840
Vis	**24.7291**	**41.3181**	**9.5911**	**0.2071**	**0.9977**
Vis-L	38.8037	70.8926	11.2105	0.2421	0.9936
Vis-FT-AutoE	48.2093	100.5171	12.7140	0.2746	0.9851
Vis-FT-LAutoE	48.0404	95.7139	16.9062	0.3651	0.9869
Vis-Freeze-AutoE	66.6220	156.4552	14.1836	0.3063	0.9627
Vis-FT-Classifier	25.7112	47.1944	10.0479	0.2170	0.9971

Table 13: Visual model per-product performance metrics for k and n values

Product	**Parameter**	**MAE**	**RMAE**	**Mean (Pred)**	**STD (Pred)**
Product A	k	42.7772	0.0371	1196.3076	12.6652
	n	0.0079	0.0376	0.2021	0.0009
Product B	k	30.3008	0.0164	1829.8286	37.0505
	n	0.0009	0.0186	0.0497	0.0009
Product C	k	9.0289	0.0051	1771.7388	11.5210
	n	0.0009	0.0180	0.0491	0.0003
Product D	k	1.5477	0.0044	348.6952	1.0936
	n	0.0036	0.0119	0.2964	0.0004
Product E	k	62.9348	0.0703	956.0381	96.3198
	n	0.0055	0.0793	0.0648	0.0062
Product F	k	0.2313	0.0102	22.5087	0.0058
	n	0.0019	0.0021	0.9281	0.0015
Product G	k	8.9714	0.0077	1150.4581	8.5090
	n	0.0063	0.0129	0.4850	0.0058
Product H	k	17.5346	0.0033	5332.9660	22.0903
	n	0.0020	0.0165	0.1180	0.0012

Table 14: Visual model with optical flow per-product performance metrics for k and n values

Product	**Parameter**	**MAE**	**RMAE**	**Mean (Pred)**	**STD (Pred)**
Product A	k	30.4873	0.0264	1184.0177	5.4906
	n	0.0258	0.1227	0.1842	0.0061
Product B	k	78.2835	0.0425	1810.7653	97.4072
	n	0.0235	0.4705	0.0320	0.0219
Product C	k	144.4046	0.0811	1873.0692	145.0608
	n	0.0363	0.7269	0.0474	0.0444
Product D	k	13.9020	0.0397	356.0596	16.7518
	n	0.0078	0.0261	0.2950	0.0068
Product E	k	150.8897	0.1685	1039.9020	130.7430
	n	0.0068	0.0970	0.0751	0.0064
Product F	k	2.7846	0.1225	25.5246	1.5859
	n	0.0126	0.0135	0.9174	0.0110
Product G	k	111.7175	0.0964	1242.5641	93.6237
	n	0.0260	0.0531	0.4776	0.0291
Product H	k	202.6903	0.0379	5509.1406	176.8085
	n	0.0163	0.1355	0.1061	0.0135