Deep Learning Based Object Tracking in Walking Droplet and Granular Intruder Experiments
Supplemental Document

Erdi Kara[*,†], George Zhang[‡], Joseph J. Williams[†], Gonzalo Ferrandez-Quinto[†,§], Leviticus J. Rhoden[¶], Maximilian Kim[‖], J. Nathan Kutz[†,**], Aminur Rahman[†,††]

This document provides additional supporting material for the research presented in the main body of our paper. The codes, datasets, and results discussed in this study are available in our GitHub repository [1].

1 Additional Details Regarding for Model Training

1.1 Object Detection via YOLO

YOLO frames the object detection problem as a regression task. As a supervised machine learning framework, YOLO must be optimized on a training dataset. In this case, the training data is composed of images and bounding box coordinates of the object classes present in each image. Each input image is divided into $S \times S$ regular rectangular grids. For each grid, the model predicts B bounding boxes represented by the box center coordinates (relative to the cell) and the width and height (relative to the entire image). A confidence score CS and conditional class probability $Pr(C_i)$ are also part of the model output. Confidence score CS is calculated as

$$CS = Pr(Object) \cdot IOU \tag{1}$$

where $Pr(Object)$ indicates the probability that the grid contains an object belonging to any of the predefined classes. While values close to 0 indicate the absence of an object in that particular grid, values close to 1 point to the presence of an object. The *intersection-over-union* (IOU) is one of key metrics in object detection problems that measures the degree of overlap (0 being no overlap, 1 being complete overlap) between the bounding box prediction and the ground truth bounding box. In model training or testing stages, one can set certain IOU threshold values to decrease the number of false positive identifications. The conditional class probability is computed as

$$Pr(C_i) = Pr(C_i|Object), \text{ for } i = 1, 2, ..N \tag{2}$$

where N is the number of classes. $Pr(C_i)$ is a direct measure of the probability of the object belonging to the class C_i. The model outputs one set of probabilities per grid. Note that if the grid does not

[*]Corresponding Author, erdikara@spelman.edu
[†]Department of Mathematics, Spelman College
[‡]Department of Applied Mathematics, University of Washington
[§]Department of Physics, University of Washington
[¶]Department of Mechanical Engineering, University of Washington
[‖]Department of Civil and Environmental Engineering, University of Washington
[**]Department of Electrical Engineering, University of Washington
[††]AI Institute in Dynamic Systems, University of Washington

contain any object, $Pr(C_i)$ must be zero. As a result, for a single image, YOLO produces a tensor of size $S \times S \times (5B + N)$, which is the prediction of the output layer of the YOLO network. Finally, this output is passed through a Non-Maximal suppression algorithm, eliminating the bounding boxes with low confidence scores. Once the output is obtained, YOLO computes the error via a *multi-component* loss function which is composed of localization error for bounding box predictions and the classification error for conditional class probabilities [2]. The confidence score associated with each box is readily available during training phase. In testing time, it is a predicted quantity calculated by the network. For droplets and intruders, the number of classes $N = 1$. Thus, whenever the object is predicted present in the image, the confidence score reduces to IOU.

We lastly discuss the concept of *mean average precision (mAP)* which is one of the most common metrics used in object detection problems. This is a single number between 0 and 1 measuring the accuracy of the network across the dataset in consideration. To compute mAP, we first calculate precision (P) and recall (R) using the following formula

$$P = \frac{TP}{TP + FP} \quad , \quad R = \frac{TP}{TP + FN} \tag{3}$$

In (3), TP represents the number of true positives, i.e, droplet is present and detected while TN indicates the number of true negatives, i.e, droplet is not present and not detected. FN is the number of false negatives, i.e, the droplet is present but not detected while FP is the number of false positives, i.e. the droplet is not present but detected. Notice that to classify a detection as TP or FP, a threshold value must be imposed. In the object detection problem, it is very common to choose an IOU to set this threshold. In this work, we report the mAP metric with the threshold value $IOU = 0.5$. In other words, whenever $IOU \geq 0.5$, we classify this detection as a true positive. Once we set the threshold value, we obtain precision and recall values and consider precision as a function of recall. Formally, the average precision (AP) is defined as

$$AP = \int_0^1 P(r)dr \tag{4}$$

where mAP is calculated among all predefined object classes in the dataset. This metric is known as *mAP@.5* when the threshold is set to $IOU = 0.5$. Another common metric is *mAP@[.5:.95]*. In this case, IOU threshold is varied from 0.5 to 0.95 and mAPs registered for each separate threshold are averaged out to obtain a single mAP@[.5:.95] score.

As outlined in the main body of the paper, we trained two different models for walking droplet and granular intruder experiments. We reported the training and testing scores associated with each experiment in Table-5 therein. Fig-1 demonstrates the evolution of mAP@.5 and mAP@[.5:.95] over the course of model training. In both cases, we can observe that the mAP@.5 values gradually increase and saturate around 1 which points to a near perfect performance.

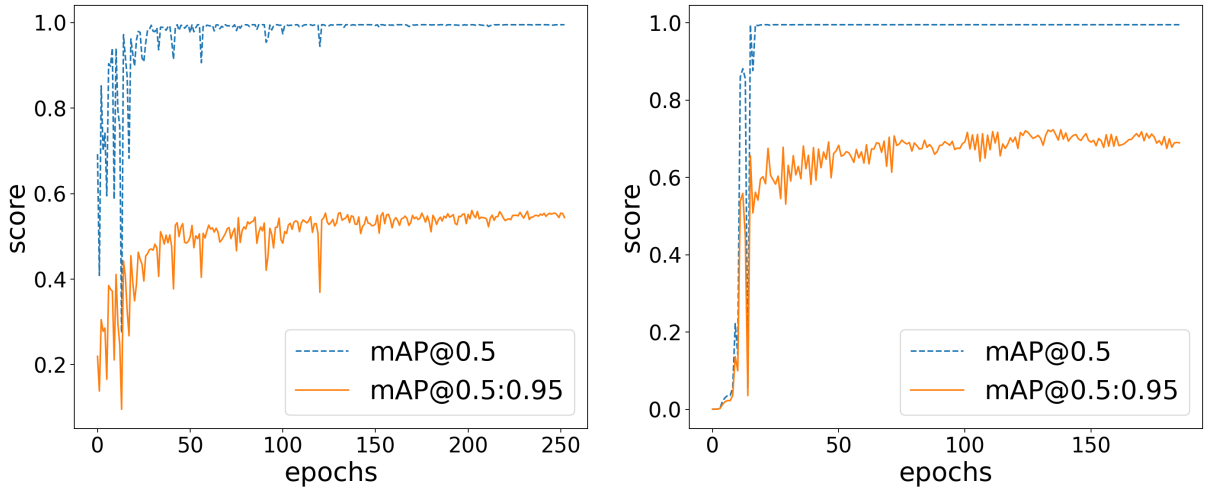

Figure 1: Training performance of the droplet model **(a)** and granular flow model **(b)**. The yellow solid curve (bottom) represents the performance of mAP@0.5:0.95 and the blue dashed curve (top) represents the performance of mAP@0.5.

1.2 Distribution of Missed Frames

In Table-5 of the main paper, we discussed the high detection rates of YOLOv8 model for our experiments. We can visualize the distribution of those missed frames based on the minimum distance of the between droplets or intruders and the time stamp of the processed frame(detected or undetected). For missed frames, we can approximate to the minimum distance between particles using the closest detected frame before the corresponding missed frame. We can observe that frames that do not pass our criteria are scattered across the experiment and usually occur during the close interaction of particles.

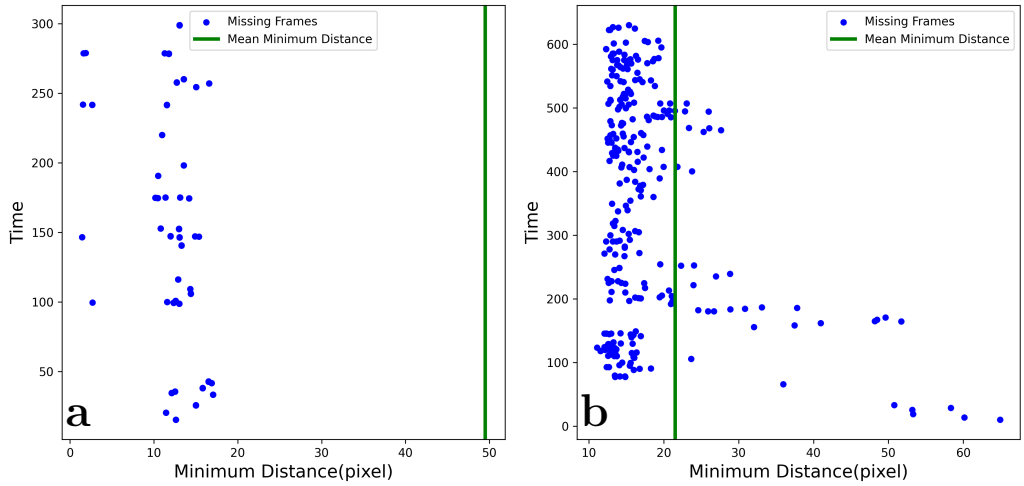

Figure 2: Distribution of those missed frames based on the minimum distance between particles and the time stamp of the processed undetected frame. For missed frames, we approximate to the minimum distance between particles using the closest detected frame before the corresponding missed frame. Three Droplet **(a)** and 2white2black-10m **(b)** experiments. We can observe that frames that do not pass our criteria are scattered across the experiment and usually occur during the close interaction of particles.

1.3 Droplet Tracking with Hungarian Algorithm and 5 SOTA Models

In the main body of our paper, we provided an extensive comparison between Hungarian Algorithm and some of the SOTA models StrongSORT, OS-Sort, Deep OC-SORT, BoT-SORT, and ByteTrack for the Three Droplet experiment. We provide the full summary in Fig-3 below where green indicating successful tracking without any ID switch and red highlighting instances where an ID switch occurred during the tracking process. As we can see, all of these models fail in case of multiple droplet experiments and similarly most of them suffer from ID switches in granular intruder experiments.

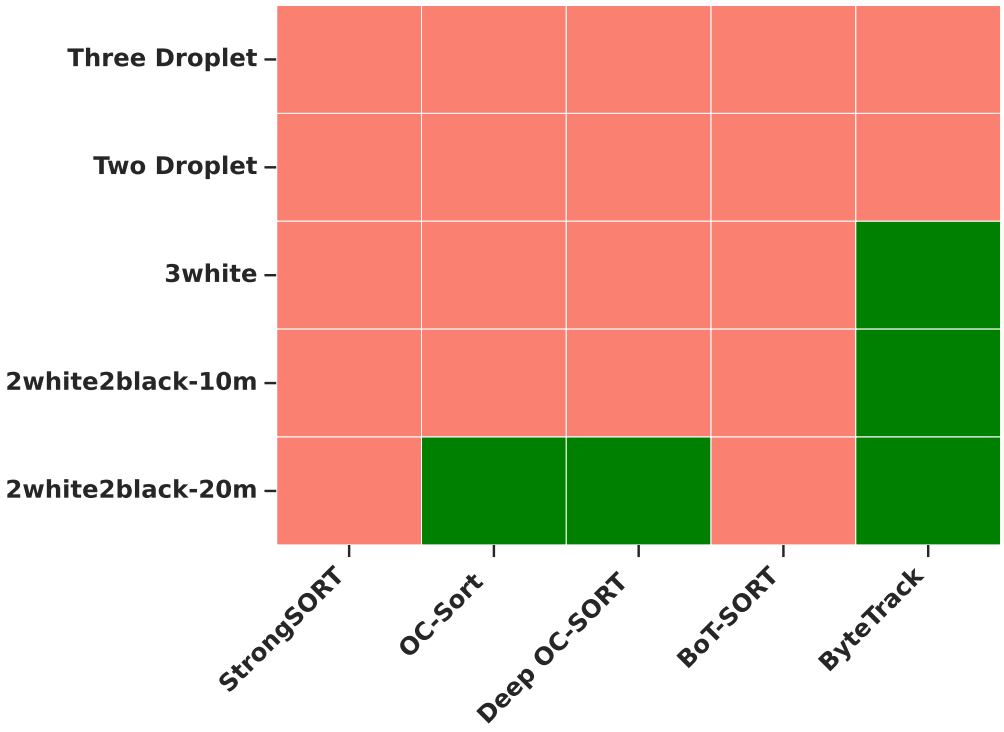

Figure 3: Comparison of tracking results using different SOTA tracking methods. Green boxes indicate successful tracking without any ID switches, and red boxes represent instances of ID switches during the tracking process.

Here, we also provide a similar results for Two Droplet experiment as well. Real-time tracking videos for each experiment can be downloaded from our repository mentioned above.

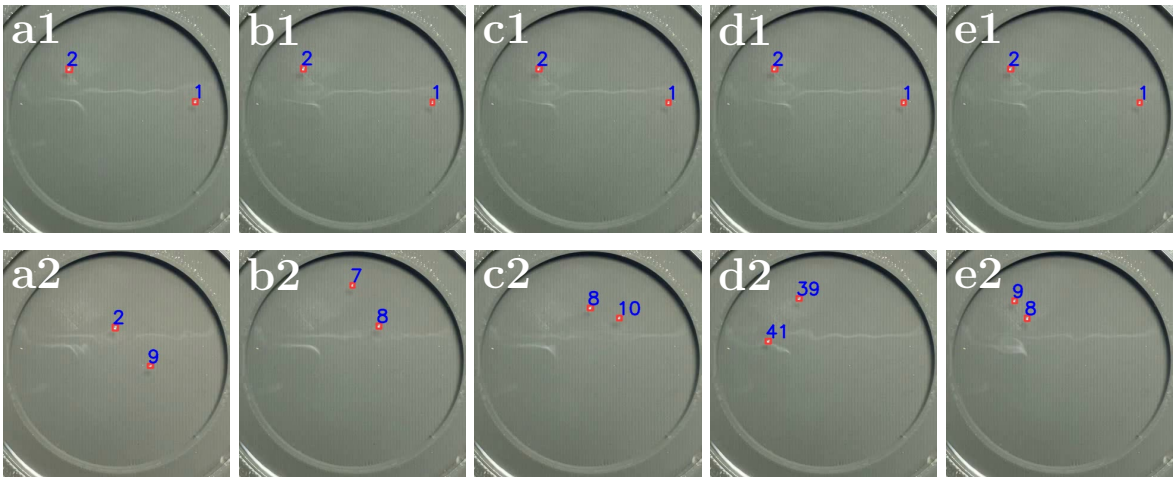

Figure 4: Initial ID assignments(top row) and ID assignments in a later frame(bottom row). StrongSORT **(a1, a2)**, OS-Sort **(b1, b2)**, Deep OC-SORT **(c1, c2)**, BoT-SORT **(d1, d2)**, and ByteTrack **(e1, e2)**. We can see that SOTA models not only fail to maintain the initial ID assignments but also create new tracks.

1.4 Walking Droplet Simulation with Large Number of Walkers

In this section, we develop a particle simulation framework to replicate the conditions of a real walking droplet simulation based on existing ground truth trajectories. Our aim is to test the capacity of our model to accurately track a large number of particles. The simulation framework is designed to capture the dynamics of particle motion, including gravitational interactions and collision avoidance. While this simulation cannot fully replicate the intricacies of the real walking droplet experiment, by faithfully capturing the dynamics of particle motion through the principles of Newtonian mechanics, the simulation provides a valuable tool to test our tracking method.

To begin, we have particle positions and timestamps from experimental data obtained from a series of frames in the walking droplet simulations we carried out in this paper. The simulation starts by initializing the particle positions and velocities based on the first frame of the experimental data. Each particle was represented as a disc shape with a defined radius and mass. In terms of color, we tried to replicate the original color a walking droplet and use one of the Control experiment as the background image. As the simulation progresses frame by frame, the particle positions and velocities are updated to incorporate gravitational forces between particles and to avoid collisions. The simulation utilizes the pair-wise forces acting between particles. These forces were determined based on the masses of the particles(which we set to 1 for simplicity) and their separation distances. Additionally, a collision avoidance mechanism was implemented to repel particles when they approached each other too closely, ensuring a minimum separation distance was maintained. This approach essentially allows us to create a simulation with large number of particles by combining the ground truth trajectories coming from different experiments. We created two such simulation with 7 droplets and 10 droplets each being approximately 4 minutes long.

By adopting the methodology described in the paper, we proceeded to train our YOLOv8 model utilizing 57 training images derived from a 10-droplet simulation. Subsequently, we conducted tests on both 7-droplet and 10-droplet simulations. Our model is again capable of tracking droplets without any ID-switch in both simulations. In Fig-5, the initial ID assignments in the corresponding experiments are depicted in the first column. The subsequent frames at two different time points are presented in the middle and right figures. The color-coded ID assignments demonstrate the model's consistent and accurate tracking, without any instances of ID-switch. Reader are encouraged to watch the actual tracking videos

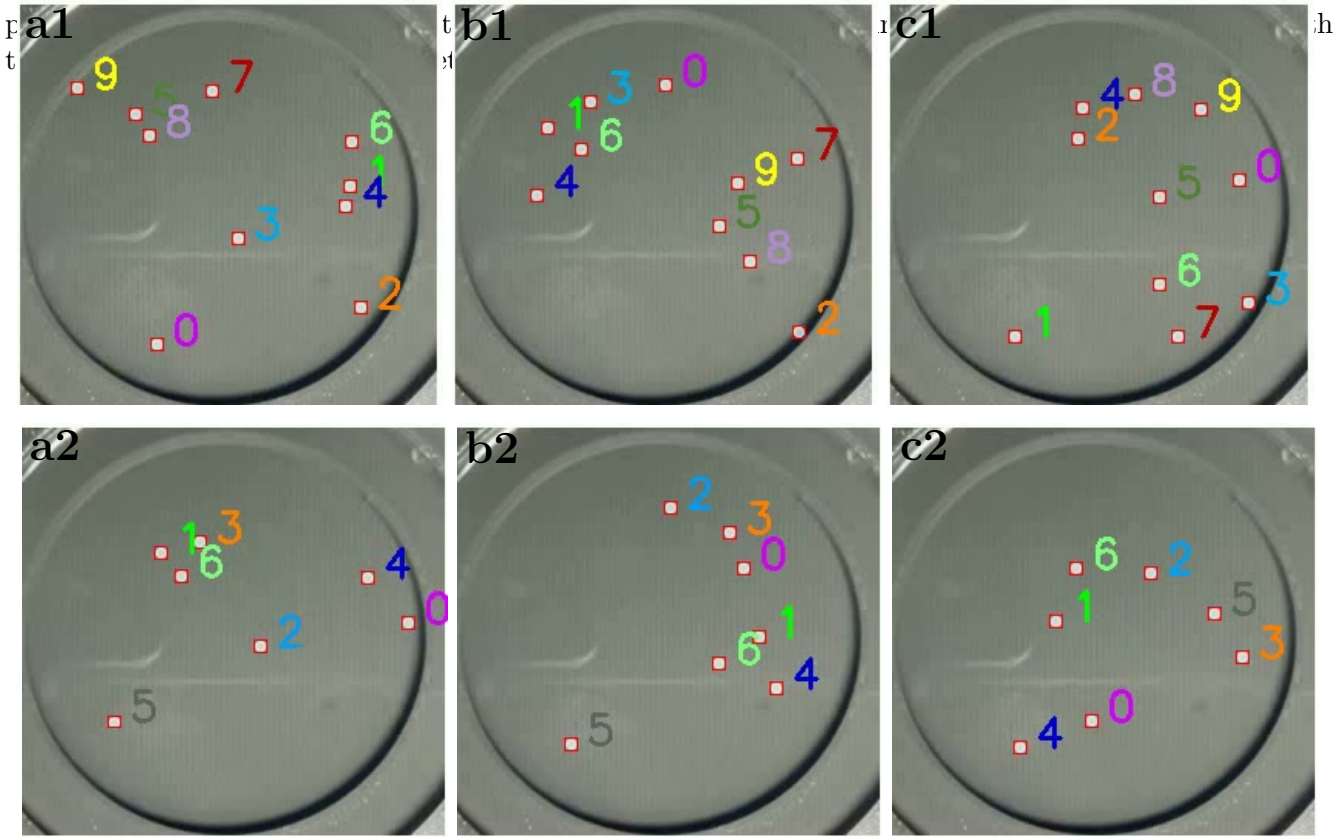

Figure 5: Tracking samples with 10-droplets(top-row) and 7-droplet (bottom row). The left figures in each row indicate the initial ID assignments while the middle and the right figures are two random frames in a later time.

2 Analysis of Droplet and Intruder Trajectories

The main body of our paper details the extraction of trajectories of object of interests in walking droplet and granular intruder experiments. Here, we present the complete set of results obtained from each experiment.

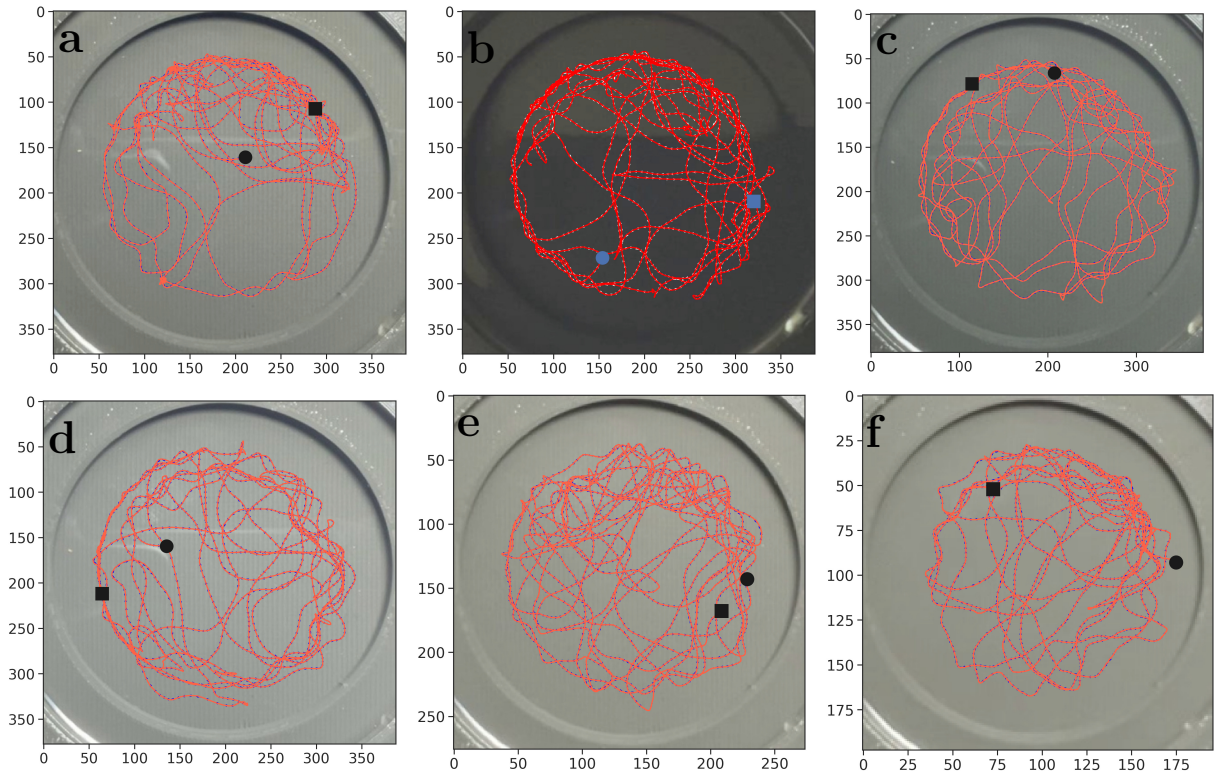

Figure 6: Flow map for single droplet experiments, "Control" **(a)**, "Lights Off" **(b**, "Lights Low" **(c**, "Lights High" **(d)**, "Res Mid" **(e)**,and "Res Low" **(f)** droplet experiments. The square and disc (black) markers represents the initial and terminal points of the motion.

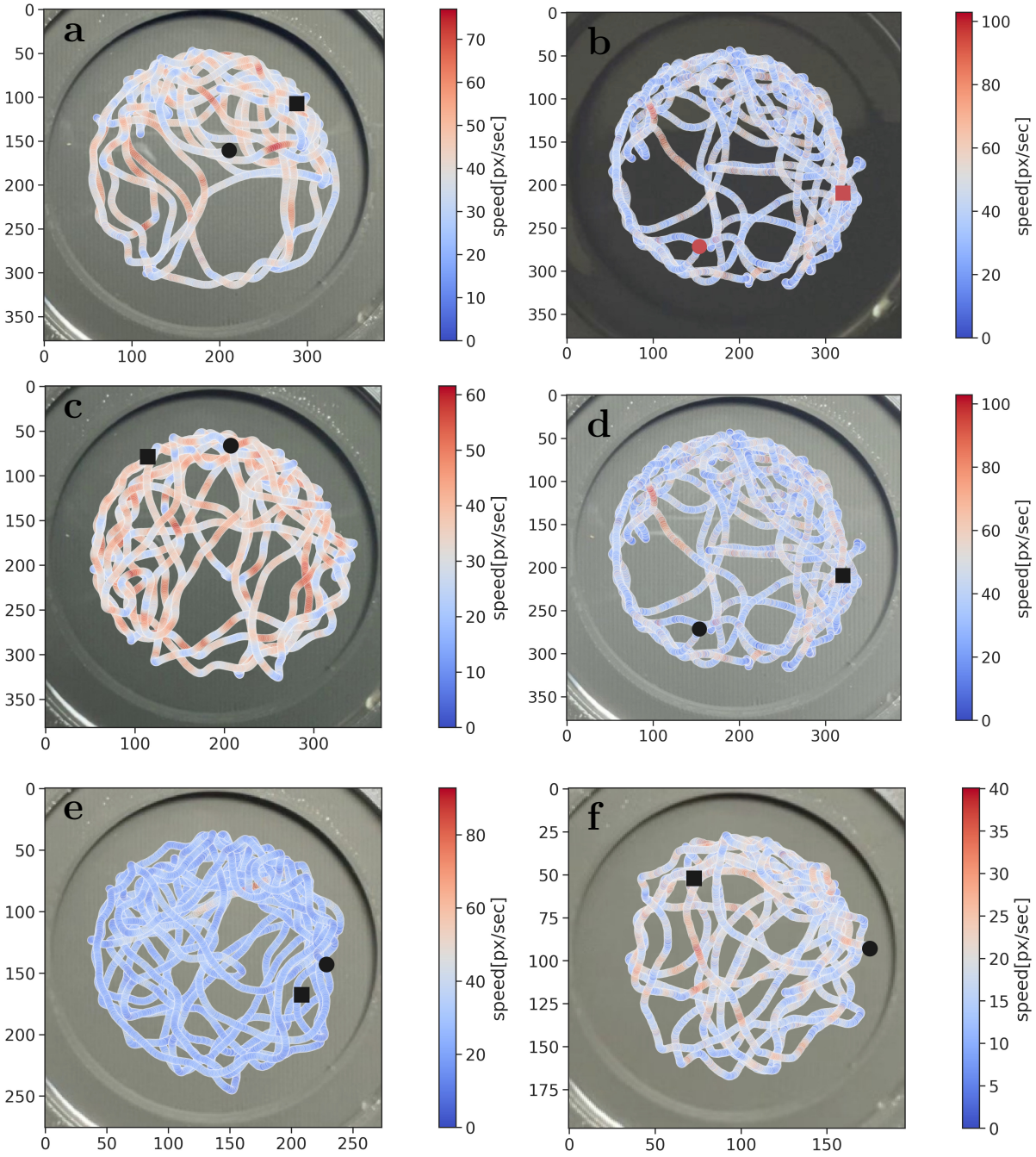

Figure 7: Speed map for single droplet experiments, "Control" **(a)**, "Lights Off" **(b**, "Lights Low" **(c**, "Lights High" **(d)**, "Res Mid" **(e)**,and "Res Low" **(f)** droplet experiments. The speed of the droplet at each location (shown in Fig. 6) is represented by the color bars. The square and disc (black) markers represents the initial and terminal points of the motion.

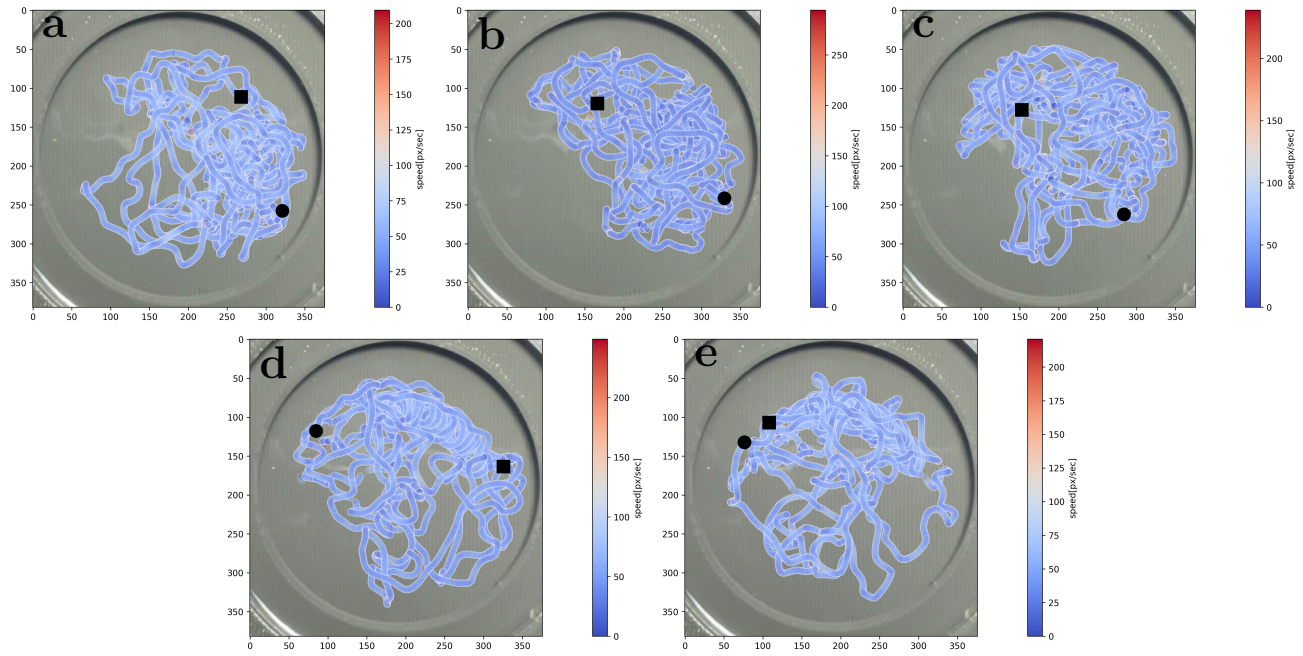

Figure 8: Speed map for multiple droplet experiments. Top row indicates the individual droplet trajectories for "Three Droplets" experiment while the bottom row are the speed maps for the "Two Droplets" experiment. The speed of the droplet at each location is represented by the color bars. The square and disc markers represents the initial and terminal points of the motion.

In the Faraday experiment which was carried out above the Faraday threshold, one can visually confirm that the droplet motion occurs in the vicinity of its initial location. As seen in Fig-9, the walker bounces back and forth between certain locations. Relatively sparse and discrete red points surrounded by blue points indicate sharp speed changes (potentially sharp turns) at these locations.

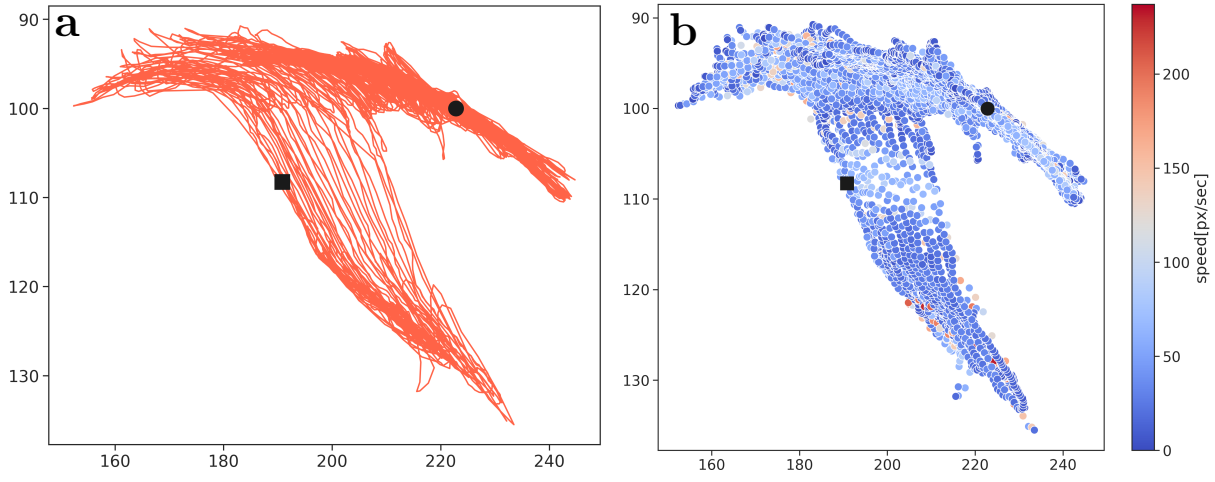

Figure 9: Location history **(a)** and speed map **(b)** for the above-threshold Faraday droplet experiment. Square and disc (black) markers represents the initial and terminal points of the motion.

Similarly, we can create inspect the speed map for the individual granular intruders.

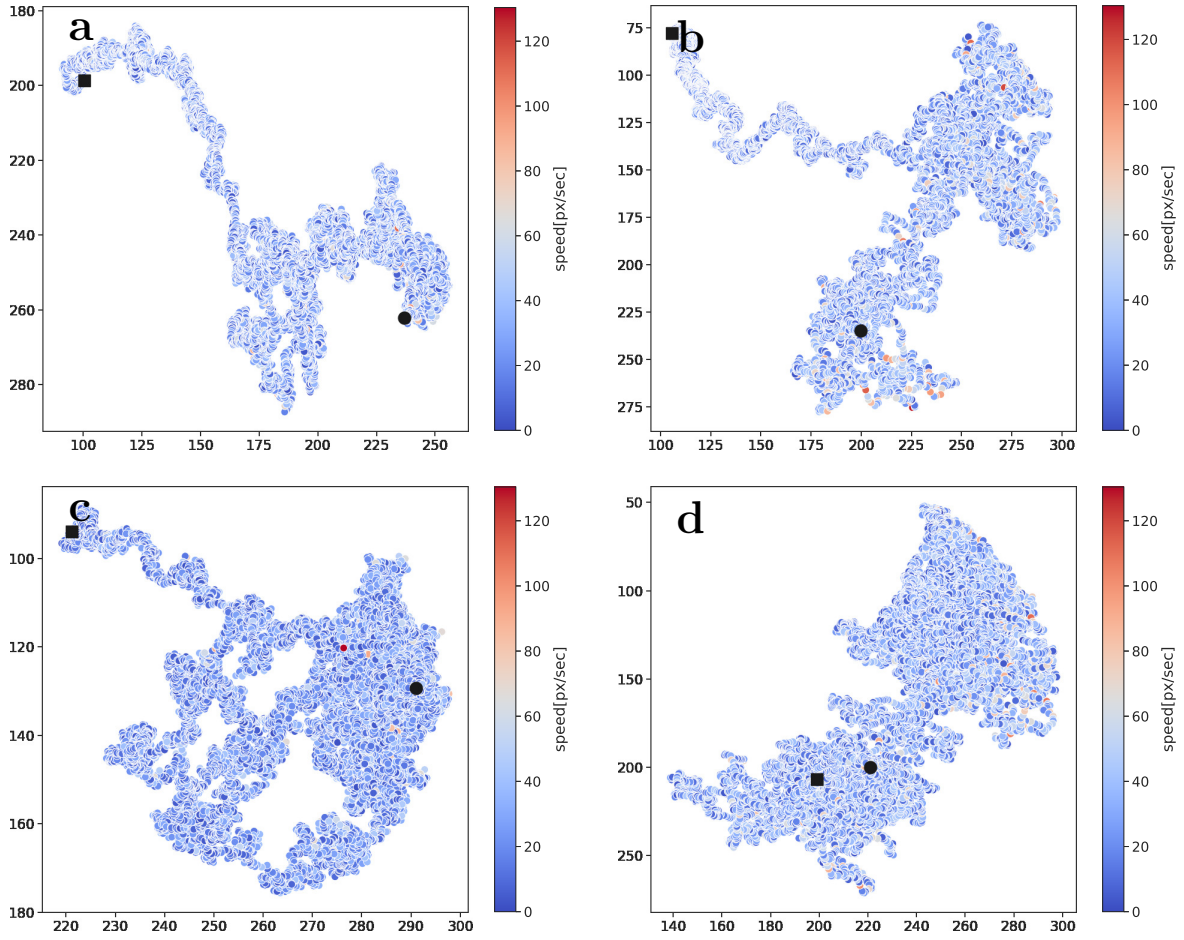

Figure 10: Speed map for "2white2black-long" granular experiment. The speed of the intruder at each location (shown in Fig. 6) is represented by the color bars. The square and disc (black) markers represents the initial and terminal points of the motion.

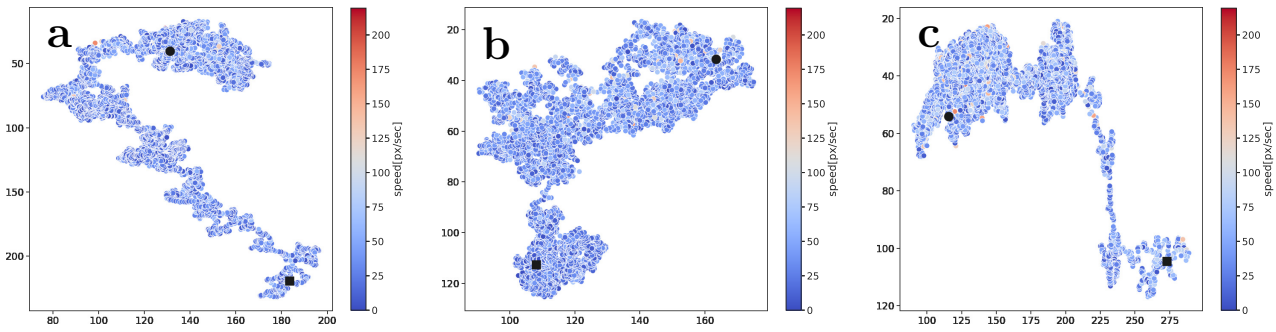

Figure 11: Speed map for "3white" granular experiment. The speed of the intruder at each location (shown in Fig. 6) is represented by the color bars. The square and disc (black) markers represents the initial and terminal points of the motion.

3 Conventional Detection Methods

In this section we detail the results of alternative detection methods performed on the walking droplet experiments. Specifically, we compare and contrast the performance of image processing, background subtraction, and the Viola-Jones methods under various experimental conditions. All of these methods, while more simplistic in nature than YOLOv8, come with drawbacks and can fail in some conditions.

A.1 Image Processing

Image processing algorithms apply operations or filters to images in order to extract meaningful information. In general, image processing algorithms do not need to be trained but do require a few parameters to be tuned for each experiment. Their simplicity comes at a cost, however. Most of the image processing methods suffer heavily from noise in the image and experimental setup. Among the methods we tested, the combination of edge detection and Hough transform, after parameter tuning, yields high precision and recall on all videos except Res Mid and Faraday. This inconsistency is the trade off made when applying image processing algorithms in exchange for ease of use.

A.1.1 Hough Transform

The simplest of the image processing algorithms we applied is arguably the Hough transform. In Hough transform, shapes in the pixels of an image are identified through a voting procedure. This technique has been shown to detect circles within an image [3, 4, 5]. As our droplets appear mostly circular in the videos we captured, we will apply the Hough transform as a last step to all our image processing methods. We implement the Hough transform for circle detection by utilizing the built-in *imfindcircles* function in MATLAB. Using parameters *RadiusRange = [2, 5]*, *Sensitivity = 0.95*, and *EdgeThreshold = 0.15* on the Control video, we obtained a recall of 0.98 and a precision of 0.48 within the corral region out of 50 frames throughout the video. We ignore false positives outside the corral region since we can safely assume droplets never escape the corral region. The parameters we chose maximized recall but introduced many false positives and therefore a low precision. Using parameters *RadiusRange = [2, 5]*, *Sensitivity = 0.96*, *EdgeThreshold = 0.1* on the Lights Off video, we obtained a recall of 1.00 and a precision of 0.93, which is an improvement over the control setup as much of the noise and imperfections in the background are no longer visible. However, there still remain a signficant number of false positives. Having additional droplets do not seem to decrease the performance of the Hough Transform. In contrast, both lower resolution droplets and a white corral background decimate the performance. Perhaps unsurprisingly, the Hough Transform completely fails when detecting droplets in the Faraday video as the Faraday waves in the background produce too many false positives regardless of the parameters used. It is also important to note that in all of the cases, the region outside of the corral produced many false positives due to imperfections in the 3D printed corral which were not counted. Frames showing detections in various experimental conditions can be seen in Figure 12. Table 1 lists the results from the Hough Transform experiment.

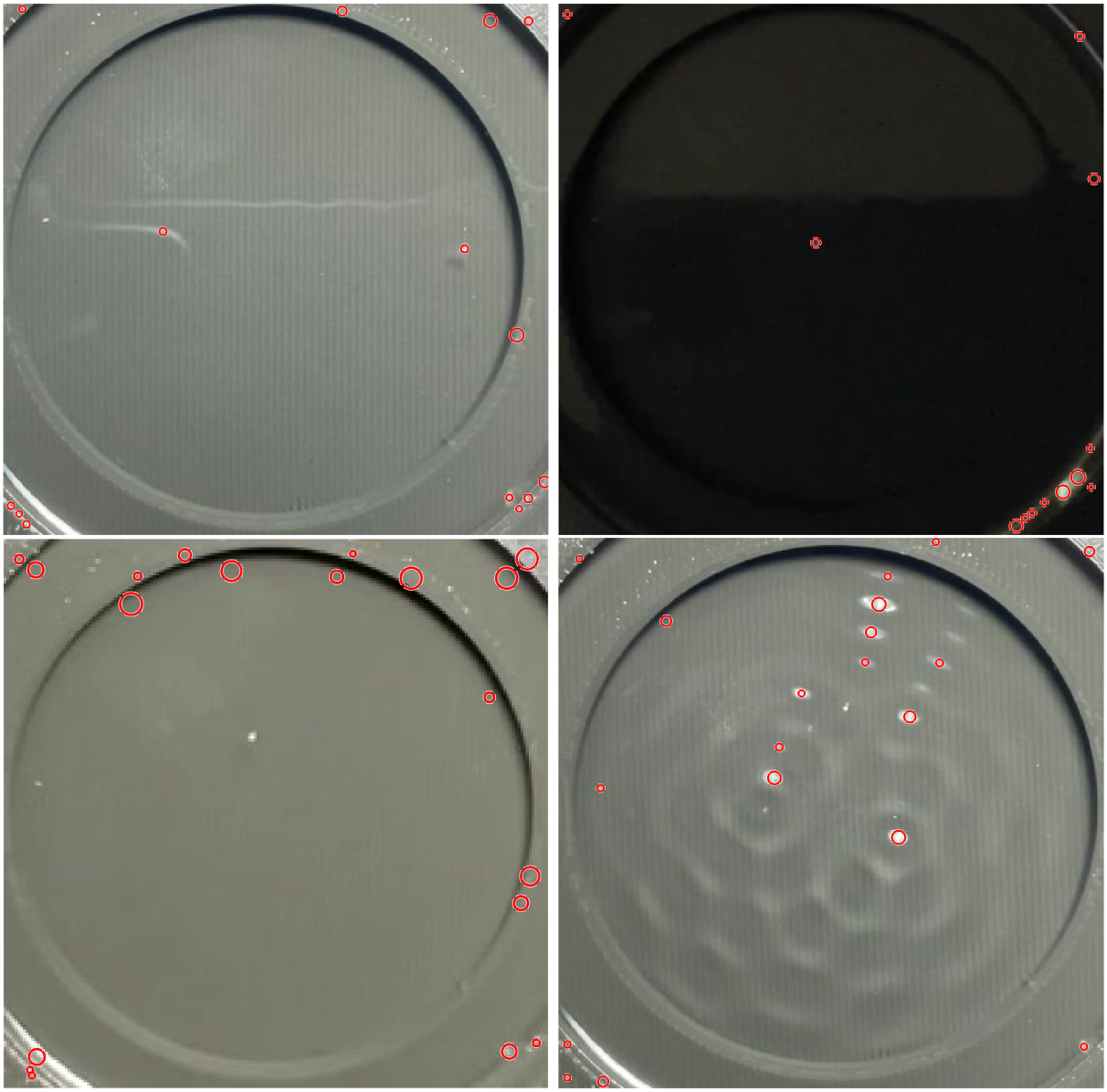

Figure 12: The upper left image shows a frame in the control video. In this frame, the droplet to the right of the corral is correctly detected but there is another false positive detection made within the corral region. Additionally, the imperfection to the left of the corral is at times mistaken for a droplet as well. The upper right image shows the Lights Off video, the lower left image shows the Res Low video, and the lower right image shows the Faraday video.

A.1.2 Top-Hat Transform

One additional step we included to improve image processing performance was carrying out morphological operations prior to applying the Hough Transform. We applied the top-hat transform to reduce uneven lighting and to enhance the droplet shapes against the background [6]. The top-hat transform computes and subtracts the morphological opening of an image from itself. Figure 13 shows the results of the transform. The performance of the top-hat transform can be seen in Table 2. Overall, the inclusion of the top-hat transform significantly improved precision while keeping recall about the same (and in the case of

Table 1: Hough Transform Results

Experiment	RadiusRange	Sensitivity	EdgeThreshold	Recall	Precision
Control	[2 5]	0.95	0.15	0.98	0.48
Lights Off	[2 5]	0.96	0.1	1.00	0.93
Lights Low	[2 5]	0.95	0.15	0.98	0.74
Lights High	[2 5]	0.95	0.15	0.96	0.40
Two Droplets	[2 5]	0.95	0.15	0.98	0.66
Three Droplets	[2 5]	0.95	0.15	0.97	0.87
Res Low	[1 4]	0.95	0.12	0.58	0.17
Res Mid	[2 5]	0.955	0.1	0.88	0.43
Faraday	X	X	X	X	X
Corral White	[2 5]	0.95	0.25	0.46	0.49

the Res Low video both recall and precision were improved). However, the detection is still not perfect, failing when applied to the Faraday video and performing poorly on the Corral White video. Additionally, precision is generally low under extreme lighting (Lights Low and Lights High) and lower resolutions (Res Low and Res Mid).

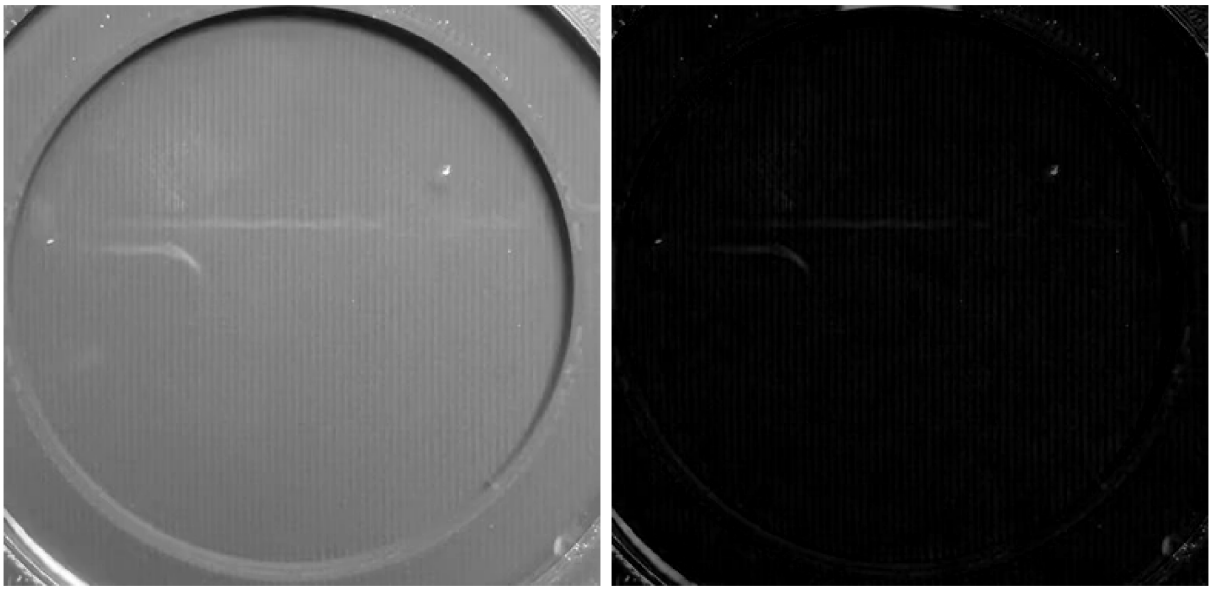

Figure 13: Before (left) and after (right) top-hat transform

A.1.3 Edge Detection

We also experimented with applying edge detection before using *imfindcircles*. MATLAB already maintains a built-in *edge* function that we utilized [7]. After experimenting with available methods, we noted that the Canny method generally outperformed others at ignoring noise and imperfections within the image (see Figure 14). The Canny edge detection method is a multi-stage process that incorporates filters and thresholding to make accurate and reliable edge detections [8]. Again, the performance of edge detection is listed below in Table 3. Incorporating edge detection resulted in a improved performances compared to using the top-hat transform. This procedure only had trouble with the Res Mid and Corral

Table 2: Top-Hat Filtering Results

Experiment	THRadius	RadiusRange	Sensitivity	EdgeThreshold	Recall	Precision
Control	5	[2 5]	0.95	0.2	0.94	0.92
Lights Off	5	[2 5]	0.96	0.2	1.00	1.00
Lights Low	5	[2 5]	0.95	0.2	0.98	0.82
Lights High	5	[2 5]	0.95	0.2	0.94	0.87
Two Droplets	5	[2 5]	0.95	0.2	0.95	0.92
Three Droplets	5	[2 5]	0.95	0.2	0.95	0.96
Res Low	5	[1 4]	0.96	0.2	0.96	0.76
Res Mid	5	[2 5]	0.955	0.2	0.88	0.86
Faraday	X	X	X	X	X	X
Corral White	5	[2 5]	0.94	0.2	0.38	0.40

White videos. Note that having a score of 1.00 does not mean perfect detections as only 50 frames were considered in calculating the recall and precision. For example, the Three Droplets vdeo produced false negatives despite having a 1.00 recall. Unfortunately, the Canny method still falls short of making accurate detections on the Faraday video as it is difficult to distinguish Faraday waves from droplets by simply examining edges and circles. While a potentially viable method in controlled experimental conditions, using edge detection may not be the most robust as it dips in performance for the Res Mid video but reported good performances for higher and lower resolutions.

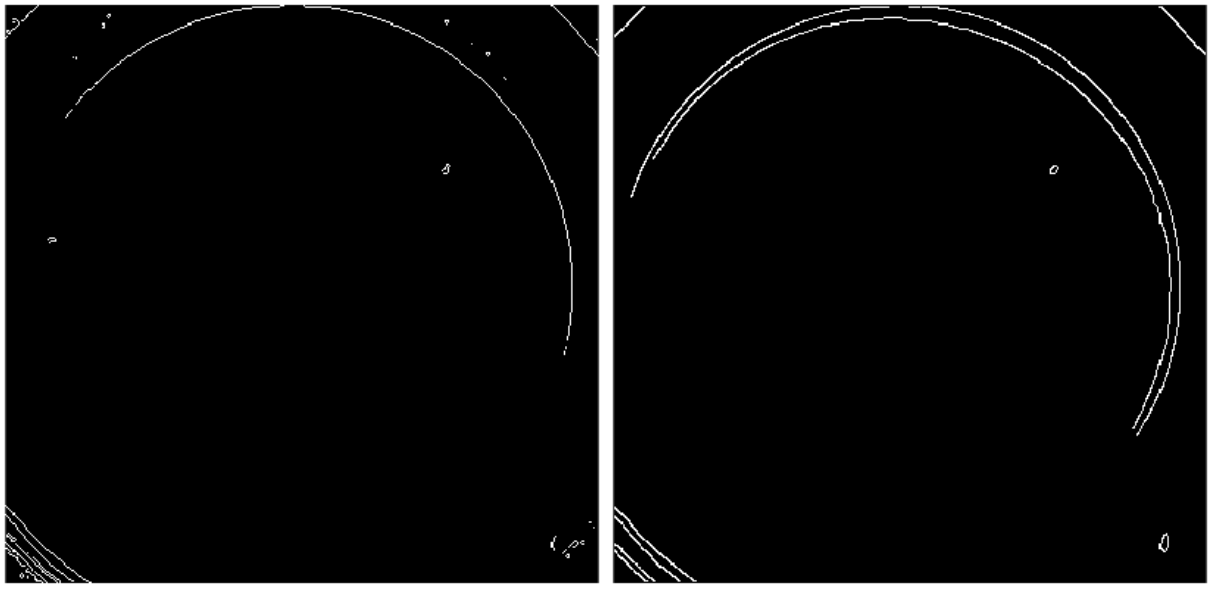

Figure 14: The default Sobel method in MATLAB (left) picks up on the corral imperfections while the Canny method (right) is able to filter out such imperfections.

A.1.4 Watershed Transform

The last method we implemented was the watershed transform using MATLAB's built-in function *watershed*. The watershed transform segments an image into separate regions by treating the image as a topographical map [9]. Readers can see Table 4 for the results. The watershed transform produced high

Table 3: Edge Detection Results

Experiment	FudgeFactor	RadiusRange	Sensitivity	EdgeThreshold	Recall	Precision
Control	8	[2 5]	0.93	0.5	1.00	0.98
Lights Off	5	[2 5]	0.95	0.5	1.00	1.00
Lights Low	9	[2 5]	0.92	0.5	0.94	1.00
Lights High	7	[2 5]	0.92	0.5	1.00	1.00
Two Droplets	8	[2 5]	0.93	0.5	1.00	1.00
Three Droplets	8	[2 5]	0.93	0.5	1.00	1.00
Res Low	5	[1 4]	0.94	0.5	1.00	1.00
Res Mid	6	[2 5]	0.94	0.4	0.94	0.89
Faraday	X	X	X	X	X	X
Corral White	14	[2 5]	0.93	0.5	0.42	0.35

precision but was the most inconsistent algorithm so far. There is generally a low recall that varied drastically depending on the lighting level. Additionally, the watershed transform was also the only algorithm that failed completely on Res Low and Res Mid videos in addition to Faraday and Corral White videos. Figure 15 demonstrates the watershed transform in action.

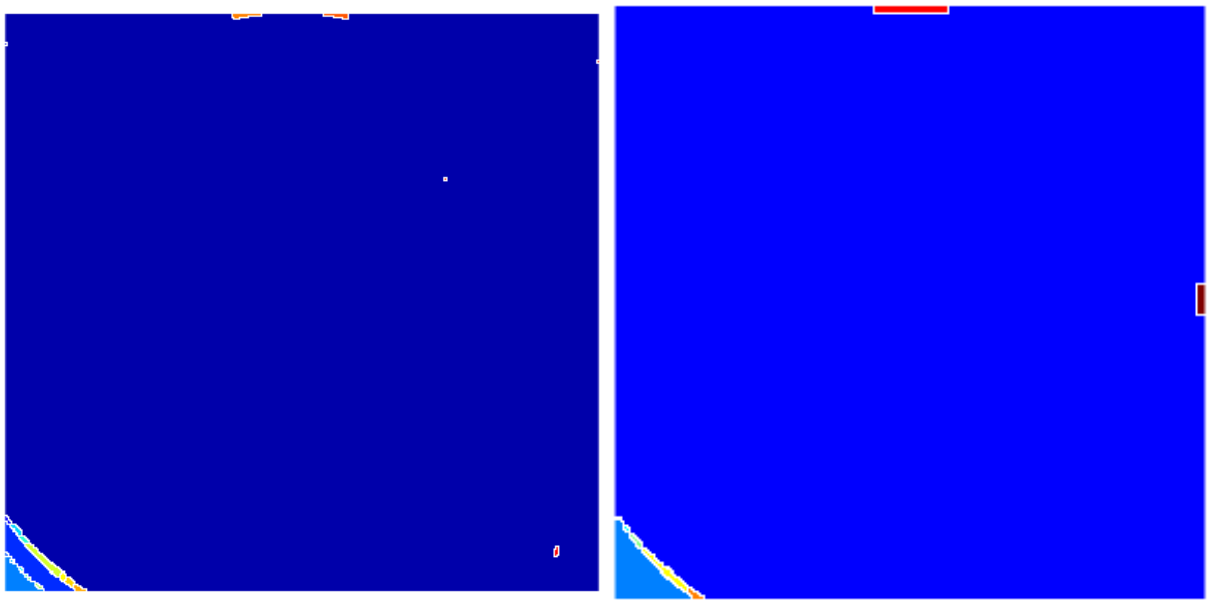

Figure 15: The watershed transform applied to the Control video (left) and the Res Mid video (right).

A.2 Background Subtraction

Background subtraction algorithms build up a model of the static background to extract moving foreground objects in the form of a black and white foreground mask. We compared and contrasted 3 different background subtraction techniques based on Gaussian mixture models (GMM) to detect and segment the moving droplets from the similarly-colored background [10, 11]. The method MOG [12, 13] is implemented in MATLAB and MOG2 [14] and KNN [15] are both implemented in Python using the OpenCV library. For both of the Python algorithms, we first apply a threshold operation to remove any shadows from the foreground mask. Prior to detecting the droplets, we also utilize an erode then a dilate operation to

Table 4: Watershed Transform Results

Experiment	THRadius	RadiusRange	Sensitivity	EdgeThreshold	Recall	Precision
Control	5	[2 5]	0.97	0.1	0.68	1.00
Lights Off	5	[2 5]	0.97	0.1	0.3	1.00
Lights Low	5	[2 5]	0.97	0.1	0.98	1.00
Lights High	5	[2 5]	0.97	0.1	0.76	0.90
Two Droplets	5	[2 5]	0.97	0.1	0.74	1.00
Three Droplets	5	[2 5]	0.97	0.1	0.63	1.00
Res Low	X	X	X	X	X	X
Res Mid	X	X	X	X	X	X
Faraday	X	X	X	X	X	X
Corral White	X	X	X	X	X	X

remove noise from the mask (except for MOG2 and KNN when applied to the Res Low video, KNN when applied to the Res Mid video, and all algorithms when applied to the Corral White video). For the MOG algorithm implemented in MATLAB, we used 5 training frames and a training rate of 0.0001 while only detecting blobs with a minimum area of 10 (3 for Res Low and 5 for Res Mid).

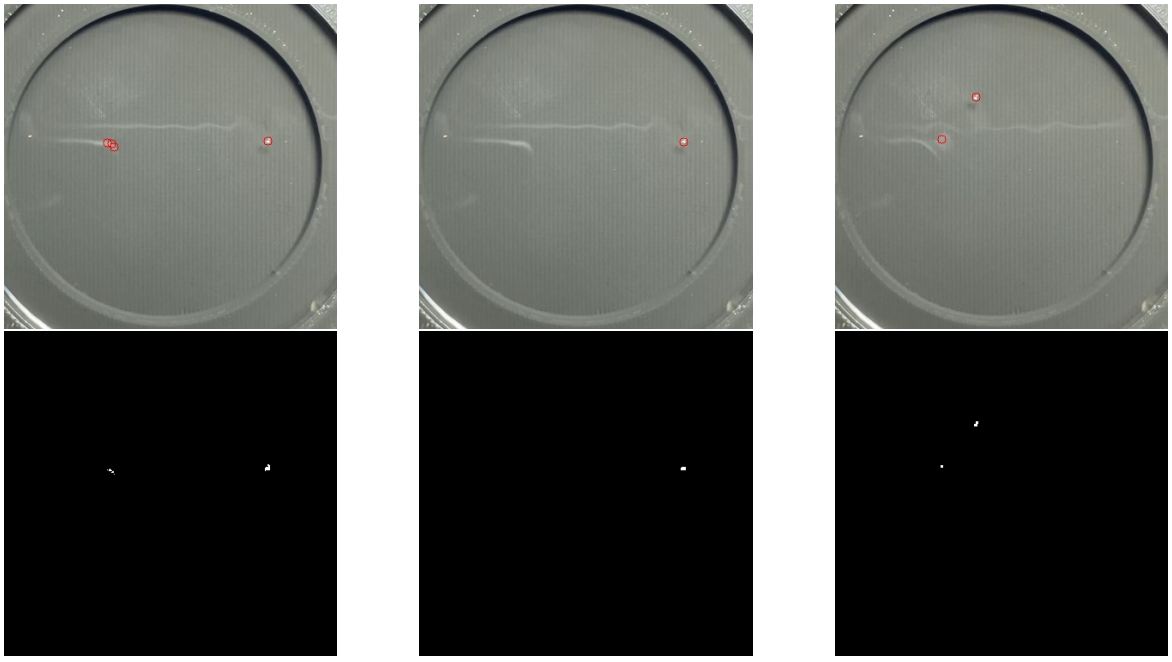

Figure 16: From left to right: frame 1 before applying erosion and dilation, frame 1 after applying erosion and dilation, and frame 2. Top images show the video post droplet detection and the bottom images show the foreground mask.

Without the erosion and dilation operations, noise within the videos translates to noise in the foreground masks which then induce false positives. We can see this in one frame (frame 1) when the KNN algorithm is applied to the Control video (as shown to the left of Figure 16). Applying the erosion and dilation operations removes most noise within the foreground mask (as shown in the middle of Figure 16). However, some noise have a large enough area such that the morphological operations are unable to remove them from the foreground mask, which then leads to false positive detections. This can be

seen in another frame (frame 2) of the same experiment (as shown to the right of Figure 16). We can potentially eliminate these false positives by increasing the number of iterations that we apply the erosion and dilation operations, but doing so will drastically decrease the recall since droplets are often erased from the foreground mask as well.

Table 5: Background Subtraction Results

Experiment	MOG (MATLAB)		MOG2 (Python)		KNN (Python)	
	Recall	Precision	Recall	Precision	Recall	Precision
Control	0.98	1.00	0.96	0.70	0.98	1.00
Lights Off	0.74	1.00	0.98	1.00	0.98	1.00
Lights Low	1.00	1.00	0.98	0.88	1.00	1.00
Lights High	0.98	0.98	1.00	0.65	1.00	1.00
Two Droplets	0.98	1.00	0.98	0.87	0.99	1.00
Three Droplets	0.99	0.99	0.97	0.88	0.99	1.00
Res Low	0.98	1.00	1.00	0.96	1.00	1.00
Res Mid	0.98	1.00	0.96	0.8	1.00	1.00
Faraday	X	X	X	X	X	X
Corral White	0.24	1.00	0.26	0.93	0.10	1.00

As expected, the motion of the Faraday waves caused all background subtraction algorithms to fail for the Faraday video. Movement in the background makes it impossible to extract the foreground based on motion. Otherwise, the KNN algorithm outperformed both MOG and MOG2 with a high recall and precision for all videos except Corral White. Again, note that having a score of 1.00 does not mean perfect detections as only 50 frames were considered in calculating the recall and precision. For example, the KNN algorithm had occasional false positive detections despite a precision score of 1.00 on most videos. While all algorithms failed to detect droplets occasionally, MOG also failed to detect droplets in a large proportion of frames for the Lights Off video. MOG2 was imprecise in general and made a significant amount of false detections in all but the Lights Off and Res Low videos. Compared to image processing techniques, background subtraction methods require less parameter tuning and do not produce false positives on 3D printing imperfections of the corral. The KNN algorithm, while not perfect, generally offers more consistency compared to image processing methods and can be viable in controlled settings. However, background subtraction algorithms do require the camera and setup to remain perfectly fixed and undisturbed throughout the experiment. Additionally, the frames of a video must be processed in chronological order.

A.3 Viola-Jones Algorithm

The Viola-Jones Algorithm is a supervised machine learning algorithm used for object detection [16]. Supervised machine learning algorithms detect objects by training on a set of labelled training images in which rectangular bounding boxes are marked around each object of interest, and in our case droplet. We implemented the Viola-Jones Algorithm through MATLAB's *CascadeObjectDetector* and used roughly 125 labelled training images per video and 6899 negative images [17] (although only about 250 negative samples per droplet were used by the detector at each training stage). The parameters used during training were *NumCascadeStages*, *FalseAlarmRate*, *ObjectTrainingSize*, and *FeatureType*. We kept *TruePositiveRate=0.999* the same for all videos. The results below in Table 6 were obtained by only keeping bounding boxes below an area of 200.

The Viola-Jones algorithm was designed primarily for face detection. Therefore, the low performance

Table 6: Viola-Jones Results

Experiment	Stages	FalseAlarm	TrainingSize	FeatureType	Recall	Precision
Control	6	0.02	[10 10]	LBP	0.44	1.00
Lights Off	4	0.02	[10 10]	LBP	0.42	1.00
Lights Low	4	0.01	[10 10]	LBP	0.28	0.88
Lights High	5	0.25	[10 10]	Haar	0.28	0.88
Two Droplets	4	0.2	[10 10]	Haar	0.2	0.80
Three Droplets	4	0.1	[10 10]	Haar	0.21	1.00
Res Low	X	X	X	X	X	X
Res Mid	4	0.2	[10 10]	Haar	0.28	1.00
Faraday	X	X	X	X	X	X
Corral White	X	X	X	X	X	X

is perhaps not a surprise when it comes to droplet detection. Ultimately, the minuscule, low resolution, and minimally textured droplets stand in stark contrast to highly textured faces. We obtained a low recall using the Viola-Jones algorithm for all videos and could not achieve successful detections on the Res Low, Faraday, and Corral White videos. For videos with multiple droplets, only one detection was able to be made on each frame. Although the Viola-Jones algorithm trains much faster than YOLOv8, its low performance makes it unviable as an option for droplet detection.

4 Trajectory Tracking Videos

As noted above, the codes, datasets, and results discussed in this study are available in our GitHub repository [1]. We highly encourage reader to examine the materials presented therein. All the experiment videos, annotations used for model training, real-time tracking videos achieved by the methodology described in the paper can be downloaded from this repository.

Here, we provide results only for the Three Droplets Experiment part of the supplementary material. All files,including the current document, are zipped into a single file named *"supp-material.zip"*. Inside this zip file, *three-droplet-experiment* folder includes the real-time tracking results of the corresponding experiment. There are three types of results associated with each experiment. In this case, *three_droplet_trace.avi* displays the evolution of the trajectory of the object of interests in real time while *three_droplet_bb.avi* shows only the bounding boxes with object IDs. *three_droplet.csv* is the extracted trajectories. For a single object, it has the following structure.

frame_id	time	detected	x1	y1	c1
1	0.00	1	64.38	211.90	0.81
2	0.03	1	64.83	213.62	0.80
3	0.07	1	64.79	215.57	0.80
4	0.10	1	65.27	217.29	0.82
5	0.13	1	65.77	219.31	0.81
6	0.17	1	66.22	220.92	0.83
7	0.20	1	66.86	222.63	0.79
8	0.23	0	Nan	Nan	Nan
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Explanation of the columns is as follows;

frame_id : ID of the processed frame.

time : time stamp of the frame associated with frame_id.

detected : Boolean value indicating if the processed frame passes our acceptance criteria.

x1 : x coordinate of the center of the bounding box.

y1 : y coordinate of the center of the bounding box.

c1 : confidence score associated with this detection.

In the case of experiments involving multiple droplets or intruders, we include separate columns associated with each object. It is important to note that we only consider a detection valid if the number of detections exactly matches the number of objects present in the experiment and if all confidence scores associated with these detections are above a certain threshold. Consequently, each frame that meets these criteria contains all the necessary information regarding the object(s) of interest. For further analyis and diagnosis, we also include the *frame_id* and *time* information for the frames that do not meet this criteria. *detected* flag is set to 0 for those frames.

Acknowledgements

The authors acknowledge support from the National Science Foundation AI Institute in Dynamic Systems (grant number 2112085). JNK further acknowledges support from the Air Force Office of Scientific Research (FA9550-19-1-0011).

References

[1] Erdi Kara. Real-time droplet tracking with YOLOv8.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[3] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, jan 1972.

[4] HK Yuen, J Princen, J Illingworth, and J Kittler. Comparative study of hough transform methods for circle finding. *Image and Vision Computing*, 8(1):71–77, 1990.

[5] T.J. Atherton and D.J. Kerbyson. Size invariant circle detection. *Image and Vision Computing*, 17(11):795–803, 1999.

[6] Suman Thapar and Shevani Garg. Study and implementation of various morphology based image contrast enhancement techniques. *Int. J. Comput. Bus. Res*, 128:2229–6166, 2012.

[7] MathWorks. Find edges in 2-D grayscale image - MATLAB edge, 2011.

[8] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.

[9] Fernand Meyer. Topographic distance and watershed lines. *Signal Processing*, 38(1):113–125, 1994. Mathematical Morphology and its Applications to Signal Processing.

[10] Nir Friedman and Stuart Russell. Image segmentation in video sequences: A probabilistic approach, 2013.

[11] Imane Benraya and Nadjia Benblidia. Comparison of background subtraction methods. In *2018 International Conference on Applied Smart Systems (ICASS)*, pages 1–5, 2018.

[12] C. Stauffer and W. Eric L. Grimson. Adaptive background mixture models for real-time tracking. *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 2:246–252 Vol. 2, 1999.

[13] Pakorn KaewTrakulPong and R. Bowden. An improved adaptive background mixture model for realtime tracking with shadow detection. 2002.

[14] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 2, pages 28–31 Vol.2, 2004.

[15] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.

[16] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[17] Prasun Roy, Subhankar Ghosh, Saumik Bhattacharya, and Umapada Pal. Effects of degradations on deep neural network architectures. *arXiv preprint arXiv:1807.10108*, 2018.