Vision-based anomaly detection to optimize logistic processes

by

Valentina Sanchez

Bachelor Thesis in Intelligent Mobile Systems

Submission: August 31st, 2022

Supervisor: Prof. Francesco Maurelli
Arturo Gomez Chavez

Jacobs University Bremen | Department of Intelligent Mobile Systems

Statutory Declaration

Family Name, Given/First Name	Sanchez, Valentina
Matriculation number	30002598
What kind of thesis are you submitting: Bachelor-, Master- or PhD-Thesis	Bachelor

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

German: Erklärung der Autorenschaft (Urheberschaft)

Ich erkläre hiermit, dass die vorliegende Arbeit ohne fremde Hilfe ausschließlich von mir erstellt und geschrieben worden ist. Jedwede verwendeten Quellen, direkter oder indirekter Art, sind als solche kenntlich gemacht worden. Mir ist die Tatsache bewusst, dass der Inhalt der Thesis in digitaler Form geprüft werden kann im Hinblick darauf, ob es sich ganz oder in Teilen um ein Plagiat handelt. Ich bin damit einverstanden, dass meine Arbeit in einer Datenbank eingegeben werden kann, um mit bereits bestehenden Quellen verglichen zu werden und dort auch verbleibt, um mit zukünftigen Arbeiten verglichen werden zu können. Dies berechtigt jedoch nicht zur Verwendung oder Vervielfältigung.

Diese Arbeit wurde noch keiner anderen Prüfungsbehörde vorgelegt noch wurde sie bisher veröffentlicht.

August 29th, 2022 Valentina Sanchez

……

Date, Signature

Acknowledgements

Family Name, Given/First Name	Sanchez, Valentina
Matriculation number	30002598
Kind of thesis submitted	Bachelor Thesis

Words cannot express my gratitude for Francesco Maurelli, my supervisor and chair of the Robotics and Intelligence Systems at Jacobs University. Not only for his support but also for the person who inspired me to start my path as a Computer Vision Engineer and also being so available and caring for all his students.
I am also grateful to my second supervisor, Arturo Gomez Chavez for his invaluable support and mentorship all throughout my thesis. I could not have had this opportunity at WasteAnt if it wasn't for him and his generosity to bring forth the opportunity to the students. For all his efforts to lead me onto the right path and taking his time to have multiple meetings with me with advice. I have felt inspired to continue improving myself in this field and to continue learning.
Last but not least, to my family and my loved one who have supported me since the very beginning. Their unconditional love and support has allowed me to grow and held me up through the hardest of times. If it wasn't for them I wouldn't be here and be the person I am today.

Abstract

Waste-to-Energy plants usually receive high volumes of waste, 15 tons of waste per delivery truck, which makes it very hard to inspect all elements manually. Even for automated systems this case would be very hard because all objects are in a highly cluttered environment, so object detection becomes very difficult. Objects can be occluded by others or fall at very high speeds which can cause motion blur or image distortions. For this reason, we decided to analyze objects' velocities (optical flow) and analyze its patterns via the LSTM(Long Short Term Memory) classifier. From WasteAnt datasets, we train a temporal LSTM model and then check for anomalies that could be hazardous to the waste management plant. My hypothesis is that anomalies in the way waste falls could be a more robust indicator of an anomaly in the waste batch than a classical object detector.
Key Words: Anomaly Detection, Machine learning, Highly Cluttered Environment

Contents

1 Introduction

The field we focused on in this paper is Computer Vision, a field of Artificial Intelligence that trains computers to interpret and understand the visual world through the use of cameras and videos. In Waste-to-Energy plants, having multiple or single objects that are not expected in our waste are subject to being hazardous for the plant. In this thesis, we aim to tackle this issue by detecting these unknown objects in live videos. We hope that this model can be used not only by the tested finely milled waste but with all the other types of trash that come into the plant.

2 Literature review

Over the years researchers have accomplished low-level and high-level computer vision tasks from the development of deep convolutional neural networks to current deep learning techniques.

2.1 Object Detection

One of the fundamental problems of computer vision, object detection detects instances of visual objects of a certain class. It is the foundation to many other tasks, such as obstacle avoidance, object tracking, segmentation, etc[7].
The initial topic dealt with action recognition and we used these same ideas to our approach in anomaly detection. In a supervised environment, a more common approach would be to do object detection, however because of the randomness and unexpected nature of anomalies in our waste it was decided that going in the unsupervised route was best.

2.2 What is anomaly detection?

Anomalies are patterns that are unpredictable or uncertain, both in supervised and unsupervised scenarios[2]. In this paper we focus on unsupervised scenarios, that because of our highly cluttered environment and unexpected anomalies, we do not have samples that are labeled. Even though we could build data from observing and hand-picking anomalies from every video, we opted to find other methods where anomalies are unknown to our model(unsupervised learning).

2.3 Challenges of Anomaly Detection

We will make note of some challenges that apply to video-based anomaly detection.
(i)Illumination - some methods mentioned, such as those of surveillance cameras, need to keep in mind the illumination changes in the scene. Most methods, including our own, expect to have constant illumination in order for anomalies to be best detected.
(ii)Perspective - camera angles could have an impact on how our model is able to detect anomalies. Challenges of detecting the same object in different perspectives may arise, in our case, the camera is stationary and we expect to have a good angle of the incoming flow of waste into the plant.
(iii)Sparse or Dense Environment - scene based methods could generate many false negatives, in our case we are dealing with a highly cluttered landscape where tons of trash is falling down fast. The need to build an unsupervised model was with this factor in mind, a robust model to this dense and highly paced environment had to be established[3].

2.4 Density Estimation

Similar to object detection, density estimation requires image data in order to train to detect the object that in our case will be an anomaly. Unsupervised learning is usually preferred when it comes to having a large number of unlabeled data at hand. By creating a probability density from the normal images or what is expected to be seen, a new image is then presented and compared against the density distribution[5]. If the features of the image are outside the expected density or its likelihood probability, then this image will be labeled as having an anomaly.
The downside, being the need of a large amount of training data in order to build an adequate density model that could give reasonable estimations for our case. Due to our environment, a stationary camera and common waste falling in the same direction, this method could work. However, because of our high-dimension data, the images, deep generative models proposed prove to be not robust enough or stable. It is our objective that we take not multiple features but less and more meaningful to our pattern creation.

2.5 Temporal Spatial

Initially, the proposed topic of this thesis leaned toward the topic of action recognition, however, currently we shifted to a more important topic for Waste-Ant and that was the detection of anomalies in waste. This did not mean that we completely disregarded action recognition, but in fact made use of important techniques.
Actions can be recognized by the scene alone, like object detection, looking at a single frame alone is enough to give results. If a picture of a child riding a bike is portrayed then the action can be identified by the detection of both the child and the bike. However, in many cases a single frame is not enough to recognize the action.
This is where temporal information comes into play, in other words more than a single frame is needed to know the context of the action. The basis architecture for this type of problems is known as Two-Stream Convolutional Networks[4], not only is the temporal information considered but also the spatial, or what is identified in a single image. As the human visual cortex, the ventral stream, which performs object recognition, and the dorsal stream, which recognises motion, all work together to make us recognise an action. By combining both streams, computers have a higher chance of pinpointing the action. If we only look at an image of a video of a person picking up an object, then the computer could interpret it as holding that object, picking it up, or putting it down. Even though this is not our path to anomaly detection, we considered relevant work on other topics to our advantage.

2.6 Feature Modeling

Transitioning from the image space, we take a look at the feature space for anomaly detection. Handcrafted features, properties of the image itself such as corners and edges, are used to build an effective representation of local regions in our image. A downside could be that hand-crafted features may be unable to adapt to or learn unexpected features effectively.
Anomalies are then detected when dividing the image into grids and identifying the anomalies in the image in a grid level. Models such as the Gaussian mixture model, assumes all data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters, and the K means clustering can be used to create the feature probability distributions of the normal images[5]. Similar to the previously discussed density estimation method, anomalies are those that are less probable or go outside our defined distribution.
Due to dividing the image into patches during our process, though effective because of its

patch-level perspective it unfortunately has downsides. One, being too time consuming during the training and testing phase. Second, the evaluation of each local region independently could disregard anomalies where the global context of the image is needed. Multi-scale strategies proposed could serve as a solution to this issue, however the relationship between features in different scales is ignored. This fault is of great importance when determining or identifying the anomaly in the scene.

2.7 Dictionaries

Another example of a work that deals with separating the image into patches is the use of dictionaries. The sparse coding algorithm is employed to create a dictionary from a learn video, the reconstruction of the weight vector that is learned for each query is then used to compute the normality measure[6]. Updating the dictionary and using the sparse coding to detect the anomalies. This algorithm has proven effective and robust for anomaly detection, dictionaries are prone to noisy elements with its constant update process.

3 Methodology

In the field of Healthcare and Security, anomaly detection has been a relevant task to deal with the unknown and uncertain information under the open and dynamic environment. Anomaly detection, depending on its environment and context, could be dealt with in many ways. For this research unsupervised and pixel level work is what will be discussed and handled. Unsupervised, in the sense that it is a long and tedious task to collect image samples of anomalies which we cannot predict or have priori of due to its randomness in our scenario. Pixel level, because of the type of features we observe in order to create a normal pattern. Therefore, we must train our model to collect patterns of motion in order to discern the rare events from the crowd. If the detected velocity goes beyond our defined threshold, then we know for sure that there is an anomaly present in our waste. As previously mentioned, it was decided to take ideas from action recognition in our methodology. Instead of looking at a single frame we considered multiple frames that could give more context to our pattern. We obtained this pattern by the optical flow of the waste, by looking at the velocity of the waste we created a threshold on regularity score that would indicate whether there is an anomaly present or not.

3.1 ROS

It is a powerful framework known as Robot Operating System that creates a web of interconnected paths within a robot. By the use of nodes and topics, each process of the robot or machine is able to communicate and connect in a more simple manner without having to code multiple files to manage the complex software. WasteAnt uses this technology at their startup, they keep videos or the data used by keeping them in rosbags or the file format in ROS for storing ROS message data.
On the python script a subscriber and a publisher is needed to access files. The subscriber that will take the topic

`/stereo\_ueye\_cam\_1/left/image\_raw/compressed`

takes the rosbags that are being played in order to pass it through the script or code. It is then that it publishes the final optical flow output through a topic called

`optical\_flow\_output`

```python
def __init__(self):
    # initialize ros publisher, ros subscriber

    self.publisher = rospy.Publisher("optical_flow_output", Image, queue_size=10)

    self.bridge = CvBridge()

    self.subscriber = rospy.Subscriber("/stereo_ueye_cam_1/left/image_raw/compressed", CompressedImage, self.callback, queue_size=1)
```

Figure 1: Code snippet of ROS subscriber and publisher

3.2 Optical Flow

Optical Flow is the task of per-pixel motion estimation between every two consecutive frames in one video. The pattern of apparent motion of objects, surfaces or edges relative to the camera. It is useful as a motion representation of a video[1]. It can be applied to image segmentation, object classification, visual odometry, and driver assistance. This task has some assumptions that must be met. One, the motion is small, meaning we could look at the next image expecting that same pixel to move close to where it was before. The second assumption is that the brightness remains constant at its appearance from *t* to *t+1* will not change. There is an equation that describes this brightness constancy constraint.

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

Let us explain this with an example, *I(x,y)* are the current coordinates of a pixel in our image at time t. Then *(dx,dy)* is the displacement of the pixel in the next frame at time *dt*. Taking the Taylor Series of the equation above and removing the common terms and dividing by *dt*:

$$fxu + fyv + ft = 0$$

We obtain the *fx* and *fy*, which are the image gradients where *(u,v)* are unknown, and with many unknowns we try to find ways to solve them.
There are two methods used depending on the task, sparse optical flow and dense optical flow. In sparse optical flow, we track some points in a video and use those same points throughout the video or in every frame and track their motion. Even if the feature points disappear in an image, the optical flow finds those similar to it and continues its work of tracking the motion. This shows that optical flow is robust for tracking.

However, in our case we do not keep track of a few selected points in our image. In fact, we keep track of all points in our image, this is known as dense optical flow. Dense optical flow, though more computationally expensive, proves to be more robust and accurate. To overcome the computational expense disadvantage, we take an ROI (Region of Interest) that is significant to us. Since we are using the optical flow for anomaly detection, it is important to keep track of all points in the image in order to create a pattern of the waste that is falling. The stationary camera and the single entry point of the waste, allows for an easy identification of what is the ROI in every video in our data. In this thesis the The Gunnar Farneback Method technique was used, to estimate the motion between the two frames and atoning for the background motion, resulting in the magnitude and direction of the optical flow from an array of the flow vectors. For instance, in a surveillance scene of cars, we do not want the motion of pedestrians to be as prominent as that of the cars. The idea of the Gunnar Farneback Method is the use of polynomial expansion to approximate some neighborhood of each pixel with a polynomial. Then, analyze what occurs to the polynomial as an ideal translation is performed.

3.3 Visualization

After deciding whether we would use sparse or dense optical flow, the next phase was extracting the velocities in order to build a visualization. Initially there was a trial and error phase of finding the right visualization technique for optical flow. Visualizing the motion of the waste is important because we want to know if our data has noise or if parameters need to be changed in order to have the best representation of our motion for later use.

3.4 HSV

The most standard visualization of optical flow is the Hue Saturation Value method. The u and v motion component matrices are converted into polar coordinates where we can get the angle and magnitude matrices of the motion vectors. The angle corresponds to the Hue channel and the magnitude to the Saturation and the Value is set to the maxima. Unfortunately, during our implementation some noise was present during the time when no trash was falling towards the end of the video. Other pre-processing techniques such as blurring and resizing of the frame was done to try to solve this issue. However, even changing the parameters of the Optical Flow Farneback function, part of OpenCV, did not help the problem. When trying different HSV implementations to our code, in a trial video of a traffic surveillance camera, the motion of the objects were clear and had almost no noise. On the other hand, any other video from the waste would be full of noise, i.e. even after the waste was finished there would be colors present in the picture.

This is when we looked at the implementation of the HSV, there we found that the normalization component could be the issue. Due to the fact that as the video progresses, the image is normalized every time and values that are extremely small are put within this normalization range and therefore appear as noise. In comparison to the car video, our video of waste stopped at a certain point or slowed down when the trash was finished being discarded. Although beautiful in its representation, the HSV implementation did not work to our needs, therefore the need of finding a new representation of our optical flow was needed.

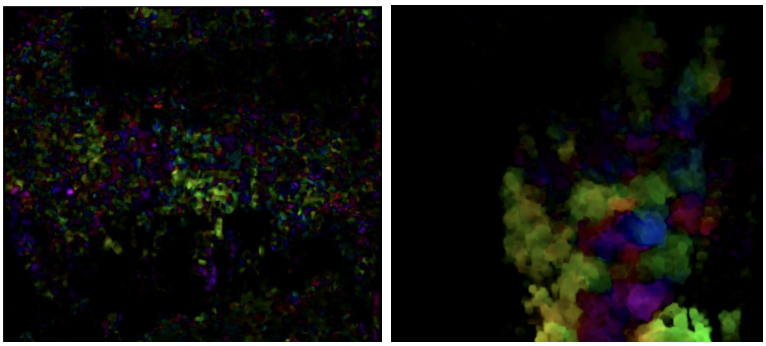

Figure 2: *Left:* Noise of no trash falling *Right:* Normal and expected flow

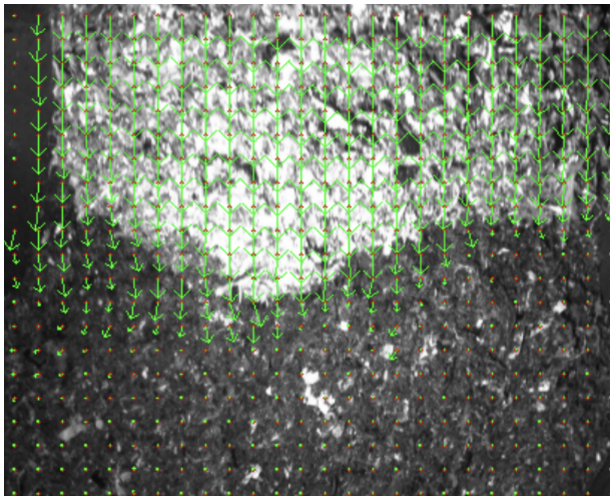

Figure 3: Optical Flow of ROI with arrows

3.5 Drawing arrows

The second most known method of visualizing Optical Flow was drawing arrows. By creating a grid of the ROI, represented by small dots. Then with the optical flow, every ten frames draw an arrow using these dots as starting points and the arrows as the displacement or movement in the trash present going in the downward direction. This proved to be a more simple and intuitive approach, where noise came from the way optical flow was being calculated rather than the way we were visualizing it. It was important not only to have an ROI established, but also to resize our video or images to a smaller size that was less computationally expensive but did not affect the quality of our data.

3.6 cv.calcOpticalFlowFarneback()

```
cv.calcOpticalFlowFarneback(prev,
                            next,
                            pyr_scale,
                            levels,
                            winsize,
                            iterations,
                            poly_n,
                            poly_sigma,
                            flags,
                            [flow])
```

Both the previous and next are the images we input into the function in order to differentiate the intensity changes from an image of the past to a current one. Since we are working with a video, we have to keep it within a constant loop. The pyramid scale specifies the image scale to build pyramids for each image. For instance, if the scale is 0.5(*scale* less than 1), then every new layer added is halved to the previous. The number of levels is what says the number of pyramid layers we will have, including the initial image. The window size on average the larger it is the more robust to noise it will be and vice versa, in our case this proved to be one of the most important parameters to change. Then there is the number of iterations done to each level or layer of the pyramid. The *poly_n* is to find the polynomial expansion between the

pixels by determining the size of the pixel neighborhood. There's the polynomial sigma or the standard deviation of the Gaussian to smoothen the derivatives as the basis of the polynomial expansion. We decided to ignore the flags and flow as it was not of use to our experimentation and were not required.
The window size being changed to 30 made it very robust that even when trash was falling at a slower pace or with a shorter displacement, no arrows would be drawn. At times it drew arrows when no trash was falling either.

On the other hand, when the window size is small, or in this case 5, it is too sensitive to any movement or intensity change. The arrows are drawn going in all different directions and not in the usual downward motion it is supposed to go.

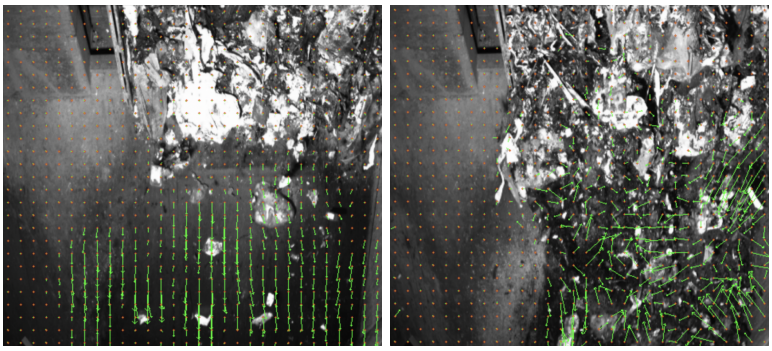

Figure 4: *Left:* 30 window size *Right:* 5 window size

4 Conclusions

4.1 Preprocessing

The image was 1080x1440 and had to be rescaled down to 434x480, then take the ROI from coordinates (100,434),(70,480). In order to go into the optical flow function the image, or in this case the ROI, has to be converted into grayscale.
In order to input the csv into our LSTM we created two columns called Past Velocities and Future Velocities. We concatenated both X and Y velocities to both columns and made the Future column every time one step forward to the Past Future in order to prepare it for training. Creating a function that splits our input into sequences that we can manage the number of steps that go in (past steps) and the number of steps that go out (future steps).
The number of steps we take every time we do a new prediction is five, and the number of steps we expect is one.
In order to identify the threshold a function that returns the upper and lower limit of the training data was created. Obtaining the mean we create the cut off two standard deviations away from the mean. Another function takes the previously found upper and lower limit and the new data to test, then finds the new mean from the new data. If the minimum velocity goes below or the maximum velocity above our data, then there is an anomaly present.
In the case of our testing, we presented a video of a different type of trash that delivers large waste. Then presented a video of the same type of trash that is expected and delivered from the videos we used for training.

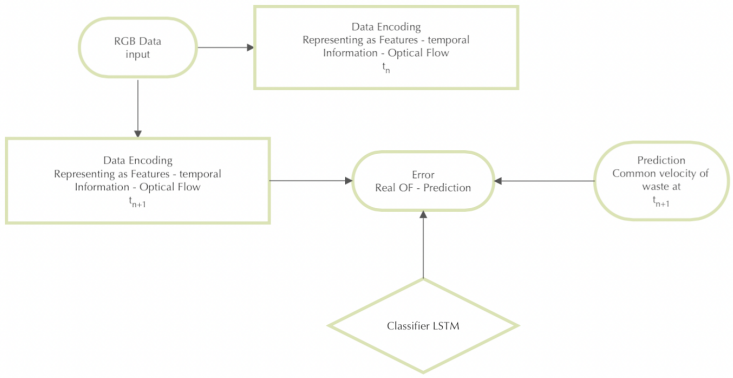

Figure 5: Graph depicting what we want to achieve.

4.2 LSTM

LSTM is a type of recurrent neural network, what makes it so special is how it is able to keep past information of inputs, hence short term memory. Not only constitutes the transformation into an output sequence but also keeps contextual information. The LSTM layers have recurrently connected blocks that act as memory blocks. It can be thought of having input, output, and forget gates, i.e. that write, read and request operations for cells. Hold the promise of sequential processing tasks such as language modeling and speech recognition. Its learning of long range temporal data makes it suitable for our case, where every image is needed to tell a story. Anything less could create gaps and not be good enough for context to build a pattern.

4.3 Models

Several models were built throughout to improve our train and testing accuracy. Since the input of our model were not pictures but rather graphs of the changes in velocity, a Convolutional LSTM was not possible. Making the model simple was one thing, but knowing what layers to have and the right parameters to implement was the trickiest. Using a single LSTM layer and two Dense layers was not giving accurate results, especially when it came to testing.

```
model = Sequential()
model.add(LSTM(10, activation='relu',
        input_shape=(n_steps_in, n_features)))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(Dense(n_features))
```

We used extra dummy data to test our anomaly detection. One of the videos was anomaly free and had never been trained or touched and another had anomalies or rather the video was of different trash. As it can be seen from the figure below, the training accuracy was relatively high, on the other hand the test and the dummy accuracy were fickle.

```
Train Accuracy: 0.842
Test Accuracy: 0.636
Dummy 1 Accuracy: 0.618
Dummy 2 Accuracy: 0.513
```

Another layer that was tried was the TimeDistributed layer, a wrapper that applies a layer to every temporal slice of an input. Unfortunately, this layer performs well when the input is an input and in conjunction with convolution. Though it seems to have improved our overall accuracy, it was still not enough to have consistency.

```
model = Sequential()
model.add(LSTM(10, activation='relu', return_sequences=True,
        input_shape=(n_steps_in, n_features)))
model.add(Dropout(0.2))
model.add(Dense(10))
model.add(TimeDistributed(Dense(n_features)))
model.add(Dense(n_features))
```

When using a higher number of epochs, the TimeDistributed layer gave better numbers. It is then when I delve into more research on what other moves I could do other than changing the epochs, batches, and number of neurons.

```
Train Accuracy: 0.781
Test Accuracy: 0.833
Dummy 1 Accuracy: 0.702
Dummy 2 Accuracy: 0.537
```

The use of stacking other LSTM layers is said to make the model deeper and therefore more accurate. Though I was afraid of increasing the model's complexity, it was the best approach so far. In all models that were tried, the Dropout layer was used because it is one of the methods used to prevent the neural networks from over-fitting.

```
model = Sequential()
model.add(LSTM(10, activation='relu', return_sequences=True,
        input_shape=(n_steps_in, n_features)))
model.add(Dropout(0.2))
model.add(LSTM(10, activation='relu'))
model.add(Dense(n_features))
```

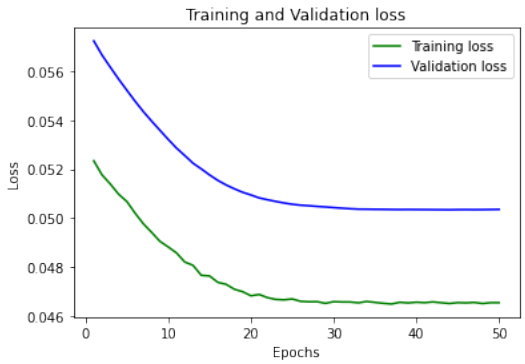

Figure 6: Graph depicting training and validation loss.

4.4 Analysis

This research aimed to identify an alternative approach to anomaly detection in highly cluttered environments, more specifically in the waste industry. Based on the lack of knowledge and ambiguity existing with knowing what is an anomaly. By using the velocities and the optical flow to create a pattern and the collection of temporal information. Which is important in order to create context and create a robust pattern. The results indicate that there is a long way to go, in making the model robust and receptive of other trash we want patterns from. Another method I would consider trying in the future is collecting patterns of what the trash looks like from the features in the picture, and comparing to see whether the new picture does not follow the pattern of the previous.

5 References

References

[1] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433–466, sep 1995.

[2] Diego Carrera, Giacomo Boracchi, Alessandro Foi, and Brendt Wohlberg. Detecting anomalous structures by convolutional sparse models. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

[3] K. K. Santhosh, D. P. Dogra, and P. P. Roy. Anomaly detection in road traffic using visual surveillance. *ACM Computing Surveys*, 53(6):1–26, nov 2021.

[4] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos, 2014.

[5] Jie Yang, Ruijie Xu, Zhiquan Qi, and Yong Shi. Visual anomaly detection for images: A survey, 2021.

[6] Bin Zhao, Li Fei-Fei, and Eric P. Xing. Online detection of unusual events in videos via dynamic sparse coding. In *CVPR 2011*, pages 3313–3320, 2011.

[7] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey, 2019.