Faculty of Engineering

BEng in Engineering Design

YEAR 4 PROJECT

Investigating Computer Vision for Production Line Health Estimation

Daniel Rodrigues

Project thesis submitted in support of the degree of
Bachelor of Engineering

Project Advisor: Jeff Barrie, Department of Civil Engineering

May 2021

Acknowledgements

I would like to thank Stephen Gilmore, Tim Quigg, Leo Green, and Nikita Alexander from Crux Product Design for supporting this project with their time and expertise, and for providing valuable insight into industry trends and regulations. At the University of Bristol, I would like to thank Jeff Barrie, for attempting to supervise the project in as turbulent a time as the present. I would also like to thank Paul Harper for his assistance across the year, both technical and pastoral, as well as Mike McCann for his assistance with scoping the project from the outset. I am additionally thankful to my project team (Ollie Fogg, James Irvin, and Jake Hodges) for their sustained effort throughout the year, resulting in a supportive, productive and exciting project time span, even amidst external challenges. Further gratitude is extended to the Open Source community for the development of numerous tools and libraries that have been used throughout this project, and have been central to the modelling activity. This is also particularly extended to the team at GitHub, who have facilitated easy and robust version control which has consistently saved the project from an assortment of weird and wacky "technical issues".

Declaration

The accompanying research project report entitled: "Investigating Computer Vision for Production Line Health Estimation" is submitted in the fourth year of study towards an application for the degree Bachelor of Engineering in Engineering Design at the University of Bristol. The report is based upon independent work by the candidate. All contributions from others have been acknowledged above. The views expressed within the report are those of the author and not of the University of Bristol.

I hereby declare that the above statements are true.

Signed (author):

Full Name:
Daniel Anthony Rodrigues

Date:
28 May 2021

Executive Summary

The medical industry demands a high volume and sterility of borosilicate glass vials, necessitating the use of carefully controlled production lines. The two key contributors to line stoppages are tipped vials (46.6%) and crushed glass (15.6%), which can reduce product quality and operational efficiency, leading to major cost penalties. Companies are increasingly looking to new solutions to assist root cause analysis and provide a greater insight into the cause of these failures.

This body of work is part of a wider group project which aims to develop novel analytical processes to help improve performance monitoring and analysis. This specific report considers the development of an image processing pipeline for visual health estimation.

A literature review highlighted a industry-wide trend towards increasingly automated, flexible, and data-driven monitoring techniques, for which image processing techniques are becoming increasingly popular. Existing systems were found to be either too specific or high-level in approach, highlighting an opportunity for an intermediary modular solution to address this commercial demand.

A lack of real-world data was identified and mitigated through the development of synthetic data generation utilities. Blender was used to create a virtual vial, which was subsequently transferred into Unity and assigned a pseudo-glass appearance to be used across the required environments. These consisted of a rendering environment for detector development, and a simulation environment for the development of the tracker and analysis layer. These tools proved to be highly effective in progressing the project in the absence of supporting data.

Several promising detection and tracking techniques were identified during the literature review, and were subsequently implemented and benchmarked to identify the best-performing methods. The optimal pair of techniques was shown to be the Scaled-YOLOv4 object detector and centroid tracker, demonstrating the greatest speed, accuracy and robustness when applied to the test data.

A custom analysis layer was developed to address the common failure modes, comprising of a localised damage assessment procedure and a macroscopic toppling risk assessment process. Although uncalibrated, it demonstrated great potential for commercial applicability, providing rich insights into product flow behaviour, sample-wise trauma, and void location and severity as an indicator of pack instability.

The main limitation of the image processing pipeline is its execution time, which is shown to exceed the 30 FPS performance budget by a considerable margin. Frame sampling at 10 FPS and reducing the region of interest are suggested to improve feasibility while retaining most of the proposed functionality, however the suitability of each modification is dependent on the intended use case.

Further avenues for development were identified, primarily relating to system calibration and performance optimisation. The supporting work packages considered failure characterisation, bulk handling understanding, and strength distribution modelling, which could help tune and further expand the proposed visual health estimation procedures. The second year of the project will aim to integrate these systems into a joint offering, demonstrating the value of the combined system for applied line analytics.

Contents

List of Figures

1 Introduction

1.1 Project Background

The medical industry requires 15-20 billion borosilicate glass vials in an average year [1], with current demand surging far beyond this due to the ongoing Coronavirus pandemic and resulting vaccination efforts. Borosilicate glass is favoured for medical containers due to its preferable properties, the most important of which being its high chemical stability. This reduces the risk of degradation and contamination of the liquid medicament, which is especially important given the already compromised state of the recipient. However, while glass is advantageous for reducing chemical contamination, its high brittleness provides an increased risk of chips, cracks, and catastrophic brittle failure, ultimately reducing the sterility of a production line through an increased risk of glass contamination [2]. A breach of sterility is of great concern, as the presence of foreign bodies in intravenous medication can be highly damaging to the end user, contributing to an increased risk of arterial embolism, granuloma formation and septicaemia [3]. This necessitates urgent action in response to faulty samples, whether it be a product recall to recover defective samples from the market, or ideally a line stoppage to prevent the release of these faulty candidates in the first place. Tipped vials are responsible for the majority of line stoppages (46.6%), accounting for three times as many line stoppages as the more catastrophic crushed glass event (15.6%) [4]. Neither event is ideal, reducing product quality and operational efficiency, and subsequently incurring large cost penalties and further adverse consequences for the consumer if left unaddressed.

In an effort to ensure a consistent quality of medical goods, there are a strict set of regulatory requirements that must be adhered to, with the Food and Drug Administration (FDA) being one of the main governing bodies for these medical standards. The FDA's auditing process largely focuses on Corrective and Preventative Actions (CAPA), ensuring the presence of systems for continuous compliance through timely and accurate performance monitoring, and non-compliance reporting and intervention [5]. CAPA helps to ensure a high quality output, however it can be difficult to achieve the scale, accuracy and level of detail of observations required, calling for the implementation of more sophisticated infrastructure and operational practises to facilitate a closed-loop CAPA process [6].

The "Industry 4.0" paradigm concerns the adoption of data-driven solutions and has great potential for improving quality management systems. Smart monitoring solutions promise augmented insights into line performance, with benefits pertaining to product quality, operational efficiency and targeted root cause analysis [7]. However, while data-driven solutions can provide substantial benefit, they often require large quantities of annotated data which presents a barrier to entry that ultimately hinders its adoption during this transition period to digital solutions. This indicates the need for an interim solution, providing tangible benefits in the immediate future, while establishing a foundation upon which more sophisticated data-driven solutions can build upon in the future.

1.2 Group Project Aim

The client for this project is Crux Product Design [8], who are a Bristol-based design consultancy with experience in the medical and fast moving consumer goods sectors. Crux are repeatedly tasked with understanding the cause of failure during bulk material transfer events, however the complexity

of these scenarios make them difficult to analyse solely by traditional methods. This highlighted the need for developing an alternative analysis approach, with clear commercial value in expanding Crux's industrialisation solution portfolio. Discussion with the client assisted the formation of the following problem statement:

> *"To develop a heuristic-driven approach to production line performance monitoring through in silico failure characterisation and bulk transport analysis".*

The overall project aims to provide a greater insight into the cause of failure events and to use this to augment passive monitoring techniques with targeted insights. The breadth of expertise required to deliver on this objective motivated the subdivision of the project into a set of more focused research areas which were identified following preliminary analysis as a group. These include the following:

1. Development of a computer vision system for data acquisition and line health analysis
2. Development of a mathematical model for estimating failures caused by processing steps
3. Investigation into catastrophic collision events using finite element analysis
4. Investigation into the interactions and behaviours within bulk material flow

The interaction between these subsystems is presented in Figure 1, with a cohesive integration of system elements allowing for the macroscopic challenge to be addressed appropriately.

This report considers the first of the research themes, namely the development of a computer vision approach for production line monitoring and health estimation. The research phase involved investigating the state of sensing and monitoring techniques within academia and industry, which was used to inform the techniques selected during the modelling stage. The outcome of this was an end-to-end analysis platform for production line footage, providing actionable intelligence through the provision of domain-specific health descriptors at individual and macroscopic scales. There exists scope for further development of the proposed pipeline with regards to feature extension and integration with the supporting systems. This will be addressed in the second year of the project.

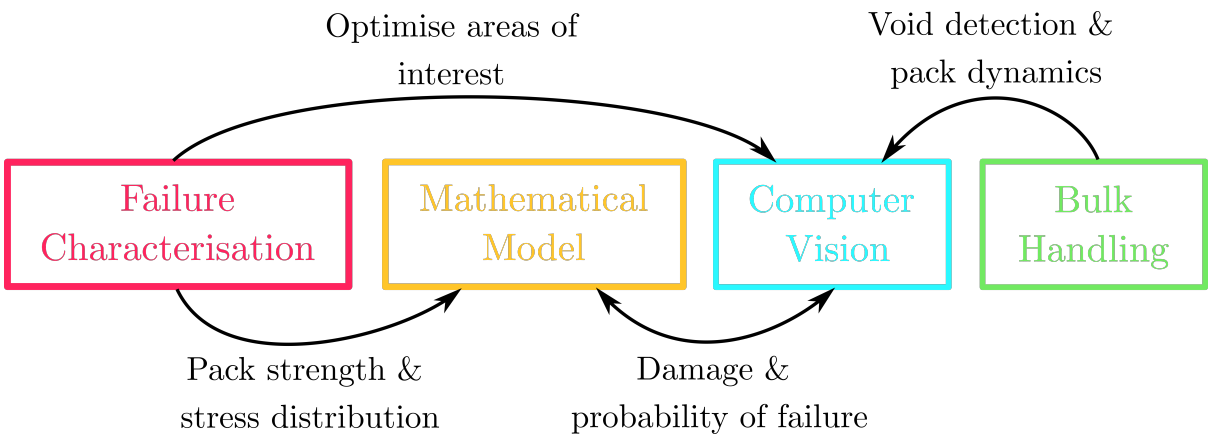


Figure 1: System diagram of group project components

1.3 Project Objectives

The project's core objective is to design an image processing pipeline for production line health monitoring. To help make this more achievable, a list of sub-objectives was produced to provide greater structure and better clarify the scope of the research project. These objectives are presented below:

1. Gain a baseline understanding of the state of monitoring techniques through a review of academic papers and commercial solutions. This is important to streamline solution synthesis, allowing for existing solutions to be applied and modified to help solve the domain-specific challenges associated with the project.

2. Build a data generation utility to provide a source of rich data for developing and validating the computer vision pipeline. This is particularly important given the absence of physical samples and real-world production line footage, allowing for development work to proceed despite these limitations. This will also facilitate the provision of accompanying ground-truth data with little further effort, unlike a traditional data collection exercise.

3. Implement detection functionality, allowing for individual packs to be localised in the scene and used for tracking efforts and to quantify spatial relations between packs. The detector should be capable of near real-time operation and should operate at a high level of accuracy for most relative and informative insights.

4. Implement tracking functionality, allowing packs to be consistently identified along the line, and thereby providing motion history information for filtering high-risk packs. This should be capable of near real-time performance, and should be robust, allowing for more prolonged tracking and acquisition of data. It should also be able to track candidates with similar or identical appearance.

5. Design and implement domain-specific health assessment procedures to quantify line health at a sample and population level. This will provide an insight into the formation of failure events, and facilitate automated detection and extraction of these events with further work. These processes should be highly performant (near real-time) to remain relevant and should be designed to capture the key failure modes on the line (tipping and crushing).

6. Evaluate the system components to quantify the level of performance than can be expected from the proposed methods, with a comparison against ground truth measures and real-time performance standards to identify an optimal technique where multiple approaches are presented. An overall evaluation of the pipeline will also be conducted to understand the overall capabilities of the presented system.

7. Evaluate the findings from the research and modelling work, considering the key outcomes and commercial applicability of the activities. Scope for further development will be considered, as well as outlining the influence of this research project on the development activities in the second year of the design project.

2 Literature Review

2.1 Current State of the Industry

Production systems are complex entities and require sophisticated monitoring and inspection processes to maximise product quality and yield. Traditional production tracking was highly manual, with a process manager logging system-wide performance information on a central production board. While providing some centralised monitoring capabilities, these manual methods were highly flawed, with the need for a human operator resulting in limited consistency, reliability, and scalability.

Automation of these inspection and monitoring processes followed through the adoption of electronic record-keeping and sensor-driven monitoring [9]. This provided round-the-clock operation, improved quality and consistency of logged information, and improved data synchronisation over the manual counterpart. However, one of the largest challenges associated with sensors is the reduced flexibility of operation over a human operator, requiring careful selection of appropriate sensors and processing algorithms to capture the same information, especially where a more abstract measurement is required.

The most common type of sensor used in production lines environments has traditionally been the photoelectric sensor [10], with applications in product counting and occupancy detection due to their high precision, high-speed operation, and suitability for sensing transparent surfaces. The main drawbacks to these simple sensors are their operational limitations, only being suitable for single-file monitoring applications and producing spatially-discrete scalar measurements, thereby requiring numerous sensors to monitor a larger region.

Radio Frequency Identification (RFID) tags have also been widely used within industrial monitoring applications [11], taking a different approach to device monitoring, with on-device integration as opposed to external sensing. This allows them to store device-specific information, such as quality control or inspection data (when coupled with other sensors), and allows devices to be uniquely identified for improved traceability and factory-wide tracking [12]. RFID technology is largely limited by the need for the addition of an RFID tag per device, which can be difficult to integrate with the product, especially for a vial, and made more complicated by the need for sterility which this could jeopardise. Further limitations arise from the additional infrastructure required to interface with and read from the RFID tags, contributing to an additional expense, as well as spatially and temporally discretising the data, with transmission only possible at an RFID read station.

SmartSkin [13] is a commercial offering specifically developed for production line applications, consisting of a wireless drone pack equipped with sensors to continuously log and report operational conditions throughout the line, including position, speed, and pressure readings. The integration of all these sensors provides a unified approach to monitoring, as well as continuous visibility throughout the line, improving upon the simpler RFID monitoring approach. However, this technology is similarly not without fault, with the drone pack being a separate device which adds logistical challenges regarding the separation of this device from the market-ready flow. Further to this, there is a reliance on using enough drones to receive an adequate representation of the line characteristics, which itself is hindered through the high price of these devices.

Image and video processing techniques are an alternative approach to monitoring [14], and one that is gaining increasing popularity due to the richness of the visual data, and the close approximation to the analogous vision-based analysis that would be expected from a human operator. Additionally, these approaches help to overcome the limitations associated with other sensing approaches, namely the single-file restriction of photoelectric sensors (camera systems provide a larger rectangular field of view), and the coupling between product flow and sensor count associated with RFID tags and drone packs (camera systems operate remotely and so can monitor multiple devices). Computer vision techniques have traditionally been used for simple inspection purposes, such as for checking occupancy, orientation, or basic indicators of product quality [15]. While capable of assessing a lot of different measures, traditional imaging systems have needed precise and consistent conditions and entirely hand-crafted detection pipelines, resulting in expensive setups, and long and labour-intensive development processes with tight operating windows.

Recent developments in machine learning have vastly improved the capability and robustness of imaging systems [16], reducing the need for the environment to be controlled to the same extent, and thereby reducing the cost and barrier to adoption. At the same time, data-driven approaches have allowed for more sophisticated algorithms, such as enabling the detection of complex product imperfections [17] with greater robustness than before. While highly attractive developments, the main drawback to these data-driven techniques is the need for lots of data, which is time-consuming to collect and annotate manually, and in some instances, may not be possible without further insight. Commercial offerings such as Roboflow [18] and Amazon Mechanical Turk [19] have emerged to help address this challenge by expediting the annotation process through user-friendly interfaces, distributed workloads and automated utilities. While relatively simple, these tools have garnered a lot of interest and uptake, with Roboflow recently receiving a $2.1M investment [20], helping to illustrate the increasing demand in computer vision systems, and hinting at the vast potential if further developed to tackle specific industrial challenges.

Despite all the advancements in image processing, most solutions currently offered are either overly specific and narrow in scope (e.g. per-device crack detection systems), or excessively abstract and lacking in task-specific analytical capabilities (e.g. generic object detection systems). This highlights a gap in the market for an intermediary system which provides the high-level large-scale detection capabilities expected for macroscopic operation, in addition to a supplementary analysis layer for domain-specific health insights. Such a system would retain a high degree of modularity and flexibility, with ease of implementing further health measures as appropriate, presenting an attractive business case both in the immediate and distant future. This forms the focus of the research project, with the remainder of the literature review considering the techniques and technology required to realise such a system.

2.2 Methodology Scoping

Visual data is typically provided in the form of images or videos, which would be ultimately sourced from a camera device. Typical cameras operate at a resolution of 1080p (or 1920x1080 pixels) and a frame rate of 30 FPS (frames per second) [21]. Camera focal length can be varied based on application, with a shorter focal length providing a wider field of view, but greater distortion, and vice versa. A focal length of approximately 50 mm is deemed similar to the human eye, and can be used to achieve a life-like

representation [22]. In a typical data collection exercise, a large volume of images would be captured with the camera, and subsequently annotated with bounding box locations to denote the location of objects within the image [23]. This is highly time-consuming in both acquiring the raw data and annotating it, necessitating improvements to data collection to reduce the barrier to entry.

Data augmentation is a technique used to help reduce the volume of raw data collection required, which works by extending the dataset with slightly modified versions of the input data (such as a modified brightness, colour or image scale), thereby artificially increasing the amount of data available for training with [24]. While improving the efficiency of the data collection exercise, it still requires real-world data, which can prove difficult to obtain in large quantities due to logistical issues or time constraints. Synthetic data is a popular solution to this, using general-purpose tools (e.g. Blender [25], Unity [26][27] and Unreal Engine [28][29]) or more industry-specific offerings (e.g. Nvidia Clara for healthcare [30], Nvidia Drive for autonomous vehicles [31] and Nvidia Isaac Sim for industrial robots [32]) to produce a digital proxy for the real environment. This helps to speed up the collection process and provides additional control over environmental parameters, allowing for a larger and more varied dataset to be collected, which is demonstrated to improve performance of deep learning techniques.

Object detection refers to the act of localising (and classifying) objects within an image, often through the use of a rectangular bounding box. Traditional approaches used hand-crafted features for detection, such as matching colours [33], shape [34], or even a reference template image [35], which a rectangular bounding box could subsequently be fitted to. These methods are simple, resource efficient, and relatively fast, however their lack of flexibility and need for manual tuning results in a need for consistent imaging conditions and extended development effort. Deep learning approaches have found much success within the domain of computer vision, achieving state-of-the-art performance across a range of detection benchmarks [16]. Early data-driven approaches were relatively slow, requiring multiple passes on the image to extract likely regions of interest, classify these, and then predict predict a suitable bounding box [36]. Speed improvements were achieved by merging these operations into a single pass, leading to the development of the single-stage object detector network. YOLO (You Only Look Once) [37] was one of the first detection models capable of real-time (30 FPS) operation, albeit suffering an accuracy penalty as a by-product of the performance optimisations. Further developments have improved the accuracy of these methods and their ability to detect small objects, with the Scaled-YOLOv4 network [38] being the state-of-the-art in object detection tasks at the time of writing. Deep learning approaches have largely superseded traditional image processing techniques due to their increased robustness and flexibility, however they require large datasets to train effectively, are less interpretable than their manual counterpart, and have increased computational requirements. Further developments in deep learning architectures have led to systems which are capable of detecting 6 degree-of-freedom orientation of objects [39], and even detect transparent mediums (such as glass) [40][41]. This further highlights the potential for data-driven detection methods, however become increasingly complicated and less performant, and so will be not be of interest for this scope of work, with an emphasis on near real-time performance.

Object tracking techniques address the problem of tracking and consistently identifying a set of objects within a video sequence, and has also attracted a lot of research interest. In the classical tracking-by-detection approach, a set of detections in two adjacent frames are compared using similarity metrics such

as spatial positioning or appearance, which can be used to assign IDs in such a way that the cost of the similarity metrics is minimised. Popular multiple object tracking techniques include SORT (Simple Online Realtime Tracking) [42], IOU (Intersection-Over-Union) tracking [43] and centroid tracking [44], using simple distance and overlap measures to compute similarity and solve the assignment problem. The simplicity of these methods results in a high performance and ease of implementation, however they can struggle with extended occlusions and other complicated tracking scenarios. Deep learning techniques have shown promise to resolve this issue, with Deep SORT [45] adding a data-driven appearance metric to help with consistent ID allocation. While the contribution of an appearance metric is useful for tasks with objects of different appearance, this is not applicable for the intended use case in the factory (with similar appearances expected), and so the extra complexity is deemed inappropriate. A promising alternative looks at combining the detection and tracking steps into one simultaneous prediction task [46], predicting object locations, as well as motion through the use of anchor tubes (a means of characterising object trajectory). While this method is shown to perform well, it adds excessive complexity through the integration of the detection and tracking stages into one model, thereby complicating the subsequent analysis procedures, and further complicating dataset creation due to its specific requirements.

The proposed system would also require a means of analysing the risk of crushed glass and pack tipping events. Cracks within a glass container can eventually lead to catastrophic failure or glass particle generation, which should be avoided. Bump check cracks are a type of crack caused by mechanical bumps and contact between devices [47]. Collision detection techniques would be required to identify these bump check events. The majority of visual collision detection approaches use trajectory-based methods to forecast and avoid collisions [48][49], motivated by their use in autonomous vehicle applications. Acceleration-based approaches to collision detection are also shown to be successful [50], and would be better suited to the intended application of detecting and analysing collision events that actually manifest. Toppling detection has been the focus of research for human fall detection, using motion cues and classification techniques to characterise the event [51]. While successful for its intended domain, it may not be applicable for a fast and continuously moving scene, such as a production line environment. Orientation detection systems [39] similarly show some potential for detecting toppling events, however remain restrictive based on data requirements and computational performance. Density estimation is an attractive alternative method that could be used to characterise the risk of falling within a crowded scene. These techniques typically characterise an image based on the number of instances present per unit area over the whole frame or region of interest [52][53], thereby lacking the level of spatial information that is desired. A Voronoi tessellation approach to spatial distribution analysis [54] is better suited, capable of producing a detailed density map with a more direct applicability to localised toppling detection.

2.3 Approach to Implementation

Informed by the literature review, this project will adopt a synthetic approach to the development and tuning of a visual analysis pipeline, benefiting from the increased data availability this provides. A core detection and tracking capability will be central to the approach, which will be selected on the basis of an empirical comparison of techniques where high performance and accuracy are favoured. A subsequent analysis layer will be developed to assess the risk of crushed glass and toppling vials through an acceleration-based collision detection method and detailed density estimation technique respectively.

3 Research Methodology

3.1 Methodology Overview

The aim of this project was to develop a computer vision system to extract health metrics from production line video footage. This could be broken into four distinct modelling stages, as follows:

1. Synthetic generation of production line footage to form the target for analysis
2. Object detection of vial instances to provide spatial information about line state
3. Object tracking across frames to provide temporal information about object motion
4. Extraction of metrics and analysis of line health using visual spatio-temporal measures

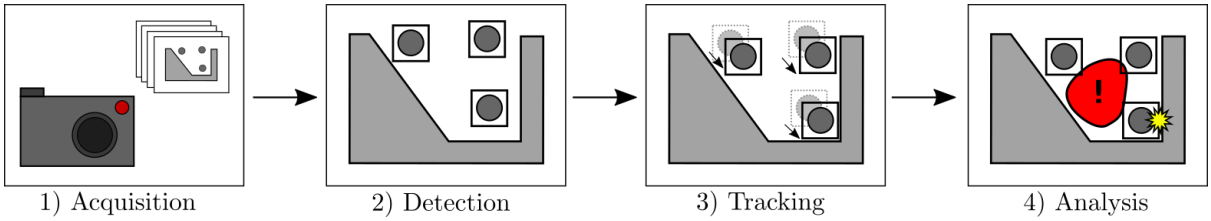


Figure 2: Diagrammatic overview of research methodology

Figure 2 provides a graphical illustration of the project methodology, with each stage sequentially built upon the last to provide the required functionality. This thereby defined the order of modelling activities, starting with the creation of data, and later concerning the extraction of metrics and insights from this.

3.2 Selection of Modelling Tools

In order to achieve the outlined project objectives, an appropriate selection of modelling tools was required. Ubuntu [55] was selected for the operating system due to it's low overhead, free licensing model, and high degree of compatibility with various software development libraries and hardware platforms. Python [56] was chosen as the primary programming language as a result of its versatility, accessibility and wide assortment of libraries, with notable examples including OpenCV [57] for image processing functionality, and Numpy [58] for fast numeric computation. C# was also selected for use in the project, particularly for scripting the functionality of the virtual environment. The Unity game engine [59] was selected for this, being highly customisable, and having a competent integrated physics engine and graphical rendering pipeline.

3.3 Data Generation

Synthetic data generation was acknowledged as the preferred form of data acquisition during the early stages of the project. This was both to help address the difficulty in accessing real-world footage due to the inherent sensitivity of this data and logistical issues surrounding the COVID-19 pandemic, as well as to provide the additional benefits with regards to data efficiency, as outlined in the literature review. The Unity game engine was chosen for creating the synthetic environment, which would be imaged using

a virtual camera. This would provide a video output similar to that expected from a real-world camera, with the additional benefit of full environmental control through user-scripted behaviours and round-the-clock availability, unlike a physical line visit. Two virtual environments were created, one producing static images for detector development, and the other generating video footage for tracker development and overall assessment of system functionality.

3.3.1 Rendering Environment

The rendering environment was created in order to facilitate the acquisition of ground truth data for training and evaluating detection algorithms. This included: producing a representative virtual vial geometry; curating a realistic visual appearance within the scene; implementing a suitable spawn algorithm for randomised vial locations; and enabling the output of ground truth annotations in a standardised format for subsequent detection operations. It was decided that detection efforts would be conducted in a top-down approach in order to provide full visibility of the bulk material flow with minimal perspective distortion, and so the synthetic environments were built with this constraint in mind for added simplicity.

3.3.1.1 Vial Geometry Creation

While all commercial vials are largely unique in geometry, they share several common features such as a wide base and narrow neck, ensuring they remain recognisable even with varying shapes and sizes. To this extent, a representative 3D model of a vial was created, drawing reference from existing vials with respect to their overall geometry and proportions to achieve relative familiarity, however not fixating on dimensional accuracy to achieve a brand and device-agnostic outcome (see Figure 3). Blender was used to create the 3D model due to its powerful polygonal modelling capabilities and its native support within the Unity game engine, allowing game assets to be automatically updated upon modification. This streamlined the development process, as well as enabling rapid exploration of different geometric variants, thereby expanding the usefulness of the data generation utilities.

Figure 3: 3D model of vial created in Blender

3.3.1.2 Visual Appearance Curation

Once imported within the Unity game engine, the appearance of the vial asset had to be specified. A high level of graphical fidelity was required in order to accurately assess the feasibility of the proposed workflow, which excessive abstraction of the visual appearance would serve to over-simplify.

A pseudo-glass material was created in the Unity editor to capture the general visual properties of a vial, utilising the High Definition Render Pipeline (HDRP) for greater realism. This included the allocation of a transparent surface type with alpha blending (62%), back-then-front rendering, transparent depth prepass, and geometric specular anti-aliasing. A sphere refraction model was used with an index of refraction of 1.032 in order to capture the subtle refraction observed with a real glass vial.

This rendering approach has its shortcomings, with a key limitation of the HDRP transparency model being the inability to render multiple transparent surfaces directly behind one another. Raytracing provides a means of overcoming the transparency limitation, however incurs large performance penalties without specialist hardware, so was omitted from the present modelling exercise. The top-down imaging approach that was adopted was deemed to help limit the impact of the transparency limitations without the performance penalty, providing a reduced ability for transparent objects to occlude each other, and hence reducing the number of cases in which these discrepancies would be observed.

Aside from directly tuning the vial material, wider HDRP settings were also configured in order to improve the overall visual fidelity and realism of the environment. The virtual camera was positioned 60 cm above the ground plane, which was deemed to be an acceptable height for a camera rig, and it was configured with a 50 mm focal length and field of view equivalent to that of the 65 mm Alexa sensor. This provides physically plausible imaging properties, akin to using the aforementioned camera, and reduces the likeliness of imaging artefacts that would not be replicable in the real world. In order to reduce the cleanliness of the image, additional optical phenomena were introduced into the virtual image capture process, namely motion blur, depth of field, and chromatic aberration, reproducing some of the "messiness" that would be expected from an imperfect real world camera. The combined result of the pseudo-glass material and HDRP setting adjustments can be seen in Figure 4.

(a) Perspective view (b) Top-down view

Figure 4: Psuedo-glass vial within Unity engine

3.3.1.3 Spawn Algorithm Design

In order to produce an representative and diverse object detection dataset, it was important to get good coverage of different spatial arrangements of objects (known as "GameObjects" in Unity) within the frame. In order to facilitate this, a parametric packing routine was implemented to randomise vial placement. A combination of user-specified starting position, instance offset, and column-wise shift amount were used to defined either a square or hexagonal packing arrangement of instantiation points for the creation of GameObjects. Further randomisation was offered through the addition of a random offset in the range [-0.1, 0.1] to translate the pattern by up to a whole vial diameter, thereby allowing for full translational coverage in the image plane for additional spatial variation. Figure 5 provides a graphical illustration of the algorithm.

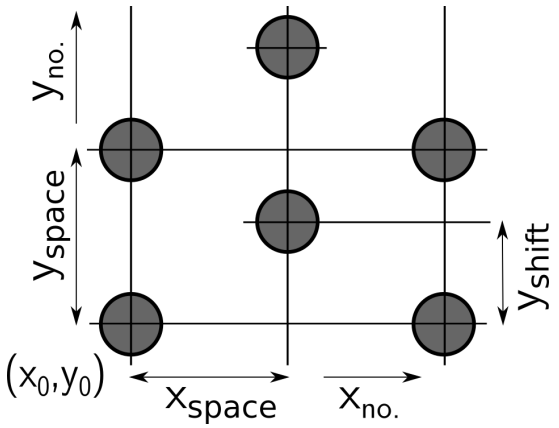


Figure 5: Parametric packing algorithm overview

In order to capture a range of packing densities and object counts, different subsets of these instantiation points were selected using reservoir sampling [60], in which k items (user-specified) are randomly selected from a population of size n in a single pass. This was achieved by iterating through all the points and instantiating a GameObject at a point if $Random(0,1) < \frac{\text{Number of remaining items to be selected}}{\text{Total number of points remaining}}$. This could be repeated several times and for different values of k to add further variety to the dataset.

3.3.1.4 Output Generation

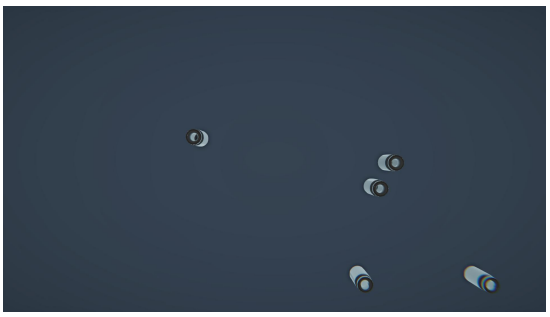
The raw 3D information was incompatible with the planned image processing techniques, and so a 2D output had to be produced. This was achieved by rendering the virtual camera view to a render texture which could be dumped to a 2D image file, thereby translating the scene into a more standard 2D image representation. In addition to saving out an image file, a corresponding label file was to be generated with the ground truth bounding box information of all instances, thereby allowing for training and testing of detection algorithms. The ground truth bounding boxes were tightly fitted about the neck of the vial, with the top of the vial selected as the most robust feature to detect given its consistent appearance and resistance to occlusion (self or otherwise), unlike the base of the vial that would be hidden upon dense vial packing.

A script was written to retrieve the ground truth bounding box information. This worked by iterating through all the vial instances present within the frame, extracting the centre of each ob-

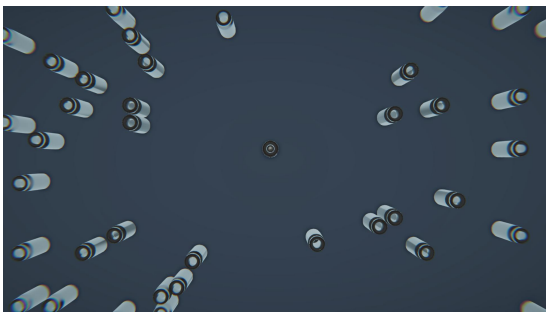
ject and its render bounds (or 3D bounding box dimensions). This information could be used to locate the corner points of a bounding rectangle for the top of the vial in 3D space, corresponding to $(x_{centre} \pm x_{extents}, y_{centre} + y_{extents}, z_{centre} \pm z_{extents})$. These points were subsequently transformed from 3D world coordinates to screen space coordinates for the virtual camera by using the `WorldToScreenPoint()` method within Unity, corresponding to the pixel locations of the bounding boxes within the image. Subsequent conversion of the bounding box information to the YOLO-style notation was achieved by first converting the 2-point corner specification (x1,y1,x2,y2) to a centre and size definition (x,y,w,h), and then normalising these by dividing by the respective dimension of the image. An additional class label (constant value 0 for the single vial class) was prepended to this data, producing a finalised data line in the following format:

{class_no., normalised_*x_{centre}, normalised_*y_{centre}*, normalised_width, normalised_height}*

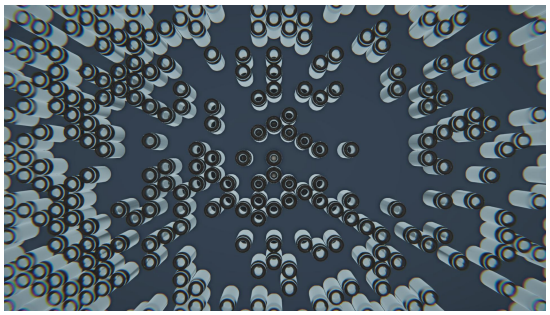
A single spatial arrangement was deemed insufficient for producing a representative dataset, and so the driving script was extended to automatically repeat the data generation process for different instance counts and in different quantities. The final script was configured to output 500 samples with 5 objects, 300 samples with 50 objects, and 200 samples with 500 objects, which were automatically divided into a 70-30 train-test split for subsequent detector development. Example data is presented in Figure 6.



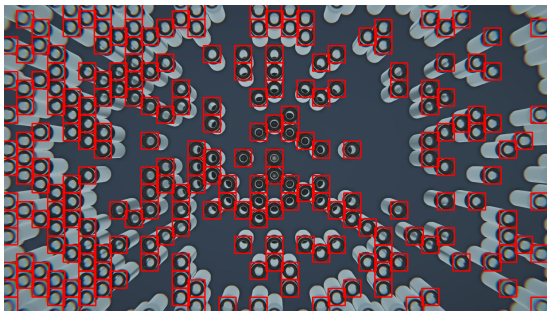
(a) 5 objects



(b) 50 objects



(c) 500 objects



(d) 500 objects (bounding box overlay visualised)

```
0 0.009839386 0.9827353 0.03169644 0.05634946
0 0.009839386 0.9260229 0.03169644 0.05634958
0 0.009839386 0.8693106 0.03169644 0.05634958
```

(e) 500 objects (bounding box data excerpt)

Figure 6: Render environment outputs

3.3.2 Simulation Environment

The simulation environment was created in order to provide a source of video footage for demonstrating tracking capabilities, overall system performance, and the applicability of domain-specific metrics. The simulation environment was largely an extension of the existing rendering environment, albeit with the addition of game physics to simulate approximate pack interactions and macroscopic behaviours. Development of the simulation environment required consideration of the physics setup, integration of an appropriate spawn algorithm, and suitable ground truth output logging.

3.3.2.1 Physics Setup

Unity's integrated Nvidia PhysX engine [61] was used to provide 3D physics simulation capabilities. The physics engine prioritises stability and computational speed over accuracy, with a primary use case being video games as opposed to physically-accurate simulations. This reduced physical realism was deemed acceptable for the purposes of the modelling exercise, with the intended health estimation workflow being scenario-agnostic and hence not being overly reliant on any specific interactions or observations that may otherwise be impacted.

A custom accumulator scenario was created (as depicted in Figure 7), with a bulk-monitoring environment selected in order to fully realise the benefits of a computer vision pipeline over a classical sensing approach. A funnel-shaped guide profile was used within the scene to encourage a dense packing of objects, resembling a crowded accumulator. A different funnel angle was used at either side in order to achieve different velocity profiles for incoming packs and to try and further randomise packing behaviour throughout the frame.

Figure 7: Simulation environment

Regarding simulation settings, a 0.01 second timestep was used within the solver (half the default value). This was settled upon in order to improve the accuracy of the simulation and minimise inter-frame penetration resulting from the discrete collision formulation for high speed objects. A rigid body representation was used for all simulated objects, which was justified by the relatively high stiffness (and resulting low capacity for deformation) of the concerned materials. The scenario was assumed to be gravity-fed at an inclination of 20° about the X-axis, and hence directional conveying was achieved through a gravitational load with components resolved in the Y and Z directions. This provided a

lightweight approximation to the gravity-fed load case, and avoided simulation artefacts resulting from improper contact computation. To this extent, a small clearance was also added between the vials and the ground plane, with the vertical translational degree-of-freedom subsequently removed from the vials in order to avoid ground contact and the problematic simulation results that would follow. All rotational degrees-of-freedom were also locked, limiting the movement of the vials to pure XZ translation. Although this limited the physical accuracy through a reduced ability for organic toppling events, it was deemed appropriate in order to negate the deficiencies of the game physics, with improper contact computation otherwise leading to excessively unstable objects when compared to a real-world case. This was not a major limitation as far as the scope of the project was concerned, with the spatial toppling formulation inspired by the literature review remaining relevant even without the possibility of directly observing toppling events in the simulation itself.

3.3.2.2 Spawn Algorithm Design

The physics simulation needed a means of introducing new vial instances at a regular interval, in order to emulate the batch-wise loading that could be expected on a production line. This was achieved by using Unity's `InvokeRepeating()` function with a custom spawn method to provide this periodic object instantiation. The parametric packing algorithm from the rendering environment was not reused for this purpose, opting for a 2D implementation of the Poisson Disk Sampling algorithm [62] which packs points closely together, but not within a specified minimum distance. The algorithm works by creating a background grid to match the size of a spawn area. A random point is inserted into the grid, after which k further points are selected from the surrounding annular disk. These are tested for validity by checking the distance between the candidate point and those in surrounding cells, thereby accelerating the search process. This is repeated until a specified limit of success points or rejected candidate points is reached, after which the list is iterated over and a GameObject is instantiated at each point. The main advantages of using this spawn algorithm in the simulation environment included the simpler control over the algorithm due to a reduced parameter set, and the increased randomness in packing arrangement which was ideal for capturing the way that variations in conveying behaviour manifest within the accumulator inlet. A graphical overview of the algorithm is presented in Figure 8.

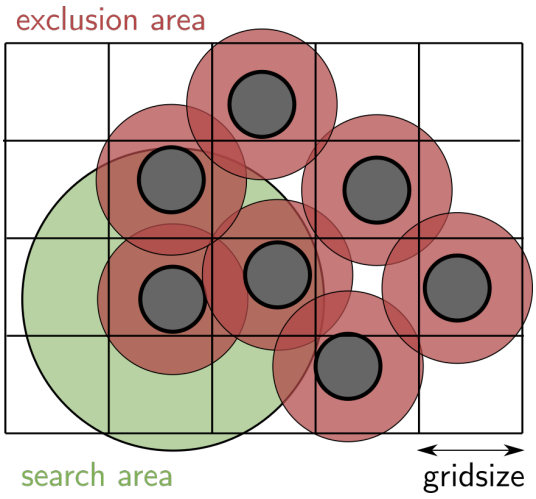
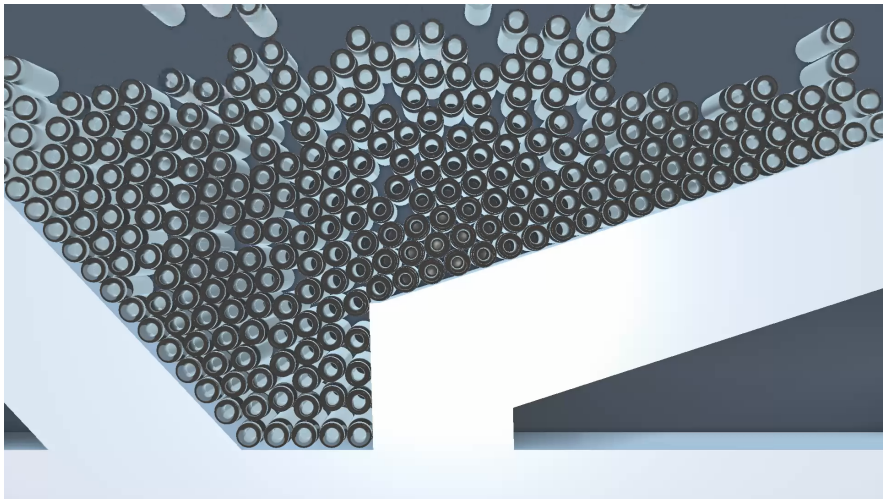


Figure 8: Poisson disk sampling algorithm overview

3.3.2.3 Output Generation

The simulation environment required two outputs, being video footage of the simulation taken from the virtual camera, and a ground truth data file for logging the number of vials visible per timestep. The video footage was produced using the Unity Recorder plugin, exporting the footage as a WebM video file and specifying a resolution of 1080p (1920x1080 px) and frame rate of 30 FPS, in line with most commercial camera solutions. The video was recorded from the main camera's field of view for a total of 407 frames, after which the scene was completely filled with vials, thereby allowing for a more comprehensive assessment of the scalability of the proposed methods. Instance counting was achieved by testing visibility via the overlap of each GameObject with the main camera's viewing frustum. If the two volumes overlapped, then the object was deemed to be visible from the camera, and so the instance was added to a list to keep track of these objects. After iterating through all the GameObjects within the scene, the length of the visible object list was queried, providing a count for the number of objects visible for that timestep. This was written as a new line within the data file and was repeated for the duration of the simulation, thereby providing a sequential listing of ground truth object counts for each frame in the video sequence. Example data is presented in Figure 9.



(a) Simulation footage (frame excerpt)

```
344 245
345 246
346 246
347 246
348 246
349 246
350 246
351 247
352 250
353 250
354 251
355 253
356 255
357 256
358 259
359 261
360 263
361 264
362 266
```

(b) Object count

Figure 9: Simulation environment outputs

3.4 Detection

Several detection techniques were explored in order to localise the vials within the frame. These included traditional methods such as colour detection, Hough circle detection, and template matching, as well as the data-driven Scaled-YOLOv4 object detection method. Detector tuning was performed using the training dataset, in conjunction with supporting scripts to streamline parameter selection, template generation and generic model training. All detection outputs were returned as both centroids and bounding boxes in the 2-point format (x1,y1,x2,y2), in order to ensure consistent formatting of detector results.

3.4.1 Colour (HSV) Detection

The colour detection approach began by blurring the input image using the `cv2.GaussianBlur()` function with a 5x5 kernel in order to reduce noise and excessive detail. This was subsequently converted to the HSV colour space using the `cv2.cvtColor()` function, in order to provide greater separability of regions based on colour and brightness when compared to the standard RGB colour space. Target regions were extracted by comparing the pixel information at each point with a user-specified lower and upper HSV threshold, defined as (0, 0, 56) and (179, 94, 255) respectively, where pixels satisfying the specified range appeared as white, and those that did not were coloured black. Connected components analysis was then performed on this binary image using the `cv2.connectedComponentsWithStats()` function with square-neighbourhood (adjacent) connectivity. Each connected component was then iterated over, extracting the position, dimensions, and area, which could be used to further filter anomalous detections. This was achieved by enforcing an area constraint between 100 and 1500 pixels, with conforming regions treated as positive detections and having their bounding boxes and centroids logged.

3.4.2 Hough Circle Detection

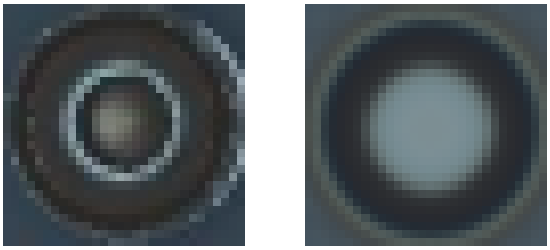
The Hough circle detection approach worked by first converting the image to grayscale using the `cv2.cvtColor()` function. The `cv2.HoughCircles()` function was subsequently called, which worked by performing a Canny edge detection [63] on the image that is then converted into Hough space in order to propose circle candidates. The function additionally filtered these candidates based on additional geometric information, such as the minimum distance between detections and the size of the radius to help minimise anomalous detections. The standard Hough Detection method was used with an accumulator size 1.5 times smaller than the image size. Minimum distance between circles was set as 20 pixels, with a minimum radius of 12 pixels and maximum radius of 15 pixels. The maximum threshold for the Canny detection was set to 27 and the circle accumulator threshold was set to 19. Centroids and bounding boxes were logged for positive detections.

3.4.3 Template Matching

The template matching approach took an input template image and searched the frame for matching instances. This matching was achieved by using the `cv2.matchTemplate()` function with the zero-normalised cross-correlation similarity measure. The result of this comparison was a grayscale image or similarity map, where pixels with greater value correspond to a greater match with the convolved template image. A user-specified threshold could therefore be used to select matching regions, corresponding to pixel values with a value exceeding the threshold. Non-maximum suppression was included with an overlap threshold of 0.4 in order to combine detected bounding boxes where appropriate and avoid duplicate or redundant detections. The bounding boxes and centroids were logged for positive detections.

Two template images were investigated, one extracted manually from the training data, and the other synthesised automatically through a custom script. The manual (or raw) template image was extracted from the centre of the frame to capture the top-down appearance of a vial with minimal influence of perspective. On the contrary, the synthetic (or averaged) image was generated by averaging the appearance within all of the ground truth bounding boxes in the training images, thereby producing an

averaged or generalised representation of the detection target. Both images were cropped for a tight bounding box, with the template images presented in Figure 10. The raw template image used a match threshold of 0.3, while the averaged template image used a match threshold of 0.5.



(a) Raw template (b) Averaged template

Figure 10: Template matching images

3.4.4 Scaled-YOLOv4

This detection approach used the Scaled-YOLOv4 object detection network to detect vials within the image. A YAML file was produced for the custom training scenario, using a single class with "vial" as the class name. As the data generator's outputs were designed with the YOLO format in mind, this data was directly compatible with the standard training procedure without further preprocessing, and so the train, validation and test directories were simply pointed to the appropriate locations. Different splits of data were experimented with, and optimal results were achieved when using all the training images for the situations with 5, 50 and 500 instances. The YOLOv4-CSP model configuration file was used to define the neural network architecture, which was then trained on an Nvidia Geforce GTX 1080Ti graphics card with 11GB VRAM. A batch size of 16 was used, with images dynamically resized to 416x416 px. The neural network was trained for 50 epochs using the default hyperparameters obtained from training on the COCO dataset, taking approximately 1 hour to train to completion.

Detection was facilitated by calling a forward pass of the neural network with gradient computation disabled. An image is used as the input to the model, which is resized to the target size of 416x416 px and normalised prior to inference. Bounding box predictions and corresponding confidence values are returned, which are then used to filter detections through non-maximum suppression, using a confidence threshold of 0.3 and IOU threshold of 0.1 to filter out the detection candidates. The final detections were then rescaled to the input image dimensions, returning the bounding boxes and centroids as outputs.

3.5 Tracking

Several tracker methods were explored in order to identify the optimal technique for robust, consistent and performant object tracking. These methods were implemented using the Motrackers library [64], using detected bounding box results from the Scaled-YOLOv4 detection model as the tracker inputs. The Motrackers library was used in order to expedite the development process and in an attempt to standardise the level of optimisation for each method, justifying its use for benchmarking purposes. The Motrackers library implemented the following tracker algorithms:

- Centroid tracker (distance-based matching)
- Kalman-filtered centroid tracker (distance-based matching, noise-aware, constant-velocity model)
- IOU tracker (overlap-based matching)
- SORT (overlap-based matching, noise-aware, constant-velocity model)

Each tracker was instantiated using default values, except from a reduced tracker persistence, in order to limit tracker lifetime to 3 consecutive frames before dropping an object. This value was settled upon in order to minimise the number of redundant trackers for a given frame, with the lack of environmental occlusions meaning that dropped trackers would likely reassociate incorrectly if retained for too long. This was especially true given the speed of the moving objects in this instance.

3.6 Line Health Analysis

Having already addressed a means of acquiring position and motion data through detection and tracking methods, the next step was to develop procedures to characterise production line health. Two levels of analysis were considered to target the different failure modes, being an individualistic scale where trauma-induced damage was of interest, and a macroscopic scale where toppling and pack instability was the main focus.

3.6.1 Damage Assessment Procedure

The damage assessment procedure was intended to capture the level of trauma a pack had experienced throughout its time in the production line, with increased trauma likely corresponding to a greater probability of catastrophic failure. While there are many contributors towards the fracture of a glass vial, only two of these were considered for the purposes of formulating a simplified trauma measure, namely cumulative loading and high-energy collisions.

Cumulative loading refers to the successive application of forces on the packs throughout their lifetime, which can eventually cause device failure. A simplified characterisation was used in order to achieve better computational performance, solely characterising the cumulative damage contribution in terms of the residence time (the time within the system) for a given pack. The rationale was that a pack with a higher residence time would have a greater level or likelihood of damage, given the increased number of interactions that would be expected (assuming a fixed probability of interactions occurring per unit time, and uniform damage contribution).

The cumulative loading contribution was practically implemented as a frame-wise computation, providing an incremental contribution that could then be applied across all tracked instances to help simplify the procedure and reduce the memory footprint over per-instance computations. The contribution can be expressed mathematically as:

$$\delta D_{cum} = \alpha \delta t \tag{1}$$

where δD_{cum} is the incremental cumulative loading damage contribution, δt is the time increment since

the last processed frame, and α is a scalar value.

High energy impacts refer to those that occur beyond a critical impact speed, such that the fracture energy of the medium is exceeded and catastrophic failure occurs. A simplified formulation of the collision contribution was defined in terms of the impact speed between two objects given a successful collision, with higher speed and thereby higher energy collisions expected to have a greater contribution.

An acceleration-based collision detection system was selected, inspired by the literature review. This required historic motion data per tracked object, which was achieved using a deque (double-ended queue) data structure of length 5. Positional data for the most recent 5 frames was stored within this, with old information continually removed and replaced with new data to maintain a consistent buffer size.

A collision can be interpreted as a large acceleration change, and so can be identified by using a finite-difference approach to calculate the accelerations and subsequently the mean magnitude of acceleration from the positional data buffer. This is illustrated in the following equation:

$$|\bar{a}| = \left\| \frac{\overline{\Delta \mathbf{v}}}{\Delta t} \right\|_2 = \left\| \frac{\overline{\Delta(\Delta \mathbf{s})}}{\Delta t^2} \right\|_2 \tag{2}$$

where $|\bar{a}|$ is the mean magnitude of acceleration, $\overline{\Delta \mathbf{v}}$ is the mean change in velocity, Δt is the time interval between frames, and $\Delta \mathbf{s}$ is the change in position(s).

Practical implementation of the finite-difference method was achieved by twice applying the `numpy.diff()` command along the 0^{th} axis of the positional deque to first compute the displacement differences and subsequently the velocity differences (in frame-time). This could then be converted to accelerations by dividing by the square of the time increment (0.033 seconds for a standard 30 FPS camera), thereby producing three values for acceleration with the positional buffer of length 5. The mean of these values was then computed, followed by calculating the euclidean norm in order to obtain an average magnitude of acceleration. This was compared against a user-specified threshold to designate what was deemed a sufficient acceleration change to warrant a collision, with a value of 10 offering the most robust identification of collision events when compared to the number of false detections.

If a tracker satisfied the collision detection procedure, then a mean impact speed could be computed using the finite-difference method over a subset of the position deque (excluding the final item assumed to occur after the collision). This was subsequently scaled by a user-specified coefficient to provide an incremental collision damage contribution, as shown by:

$$\delta D_{col} = \beta \left\| \frac{\overline{\Delta \mathbf{s}}}{\Delta t} \right\|_2 \tag{3}$$

where δD_{col} is the incremental collision damage contribution, $\overline{\Delta \mathbf{s}}$ is the mean change in position, δt is the time increment since the last processed frame, and β is a scalar value.

The collision detection procedure only identifies a collision event for a moving object, and so a static

object involved in a collision would not be recognised by this. According to Newton's third law, "every action has an equal and opposite reaction", and so the same incremental collision damage contribution was applied to all other tracked objects within a search radius of 40 pixels (or 4/3 the imaged vial diameter). The intuition behind this was to find all instances that were likely in contact and propagate the damage to these, being likely candidates for the collision interaction.

With the cumulative loading damage contribution and all the collision damage contributions computed, the combined damage increment could then be calculated as a sum of damage contributions. This could then be used to update the damage state for each tracked object, allowing for persistent monitoring of trauma levels across the population of vials. This calculation can be expressed as follows:

$$\delta D_{comb,i} = \sum_{j=1}^{n} \delta D_{col,i,j} + \delta D_{cum} \tag{4}$$

where $\delta D_{comb,i}$ is the combined damage value for the i$^{\text{th}}$ object, $\sum_{j=1}^{n} \delta D_{col,i,j}$ is the combined collision damage contribution for the i$^{\text{th}}$ object, and δD_{cum} is the cumulative loading damage contribution.

3.6.2 Toppling Risk Assessment Procedure

The toppling risk assessment procedure was intended to capture pack instability throughout the line. Given the orientation lock within the simulation environment, it was not possible to directly observe a tilt angle or toppling event, and so an indirect measure was required to characterise stability. Informed by the literature review, a spatial density approach was selected as a proxy measure for pack instability, with a lower spatial density contributing to an increased risk of toppling through insufficient support from surrounding pack members. A Voronoi tessellation approach is shown to be effective at evaluating occupancy [54], however the focus of this partitioning effort is placed on the objects themselves as opposed to the gaps between them. To this extent, the Delaunay triangulation was selected, being the dual graph of the Voronoi diagram, and hence maintaining validity in approach while shifting the focus to the vacant regions (or voids), which are more relevant for identifying toppling sites.

In order to apply the Delaunay triangulation, a set of triangulation points had to be specified, corresponding to entities providing resistance to toppling, including vials and external support geometry, such as guide rails. Vial locations were directly propagated from the detection stage to the analysis layer as object centroids, while the support geometry was provided through the use of a custom utility. This was achieved by specifying a straight line contour to encompass the unoccupiable region of the frame, the boundary of which could be interpreted as the external support site. This contour was specified coarsely, and so was subsequently subdivided into line segments of length 30 px (equivalent to a vial diameter) for a more uniform point distribution and higher spatial resolution, emulating the distribution achieved from a dense object packing. These boundary points and centroids were collectively used for the Delaunay triangulation, spatially partitioning the convex hull formed by the supplied points. As a result of the concave shape of the accumulator's support boundary, some triangular regions were formed outside the occupiable region in the frame. These were filtered by computing the centroid of each triangle and testing for inclusion within the exclusion contour, rejecting the facets that met this criterion in order to produce a density map that conformed to the occupiable region of the frame.

Having produced the triangulation, this could then be analysed to indicate toppling risk. Triangle skew and size were deemed the most appropriate measures for analysing the triangular facets, with a healthy and stable pack arrangement assumed to be one where the vials were densely and regularly packed, implying the presence of small equilateral triangles. In contrast, larger or more skewed triangles would likely be an indicator of poor packing, and hence higher instability and toppling risk. Area was used as the measure of size, while the standard deviation of edge length was used to quantify skew. The size and skew calculations were performed per triangle, which was then used to classify the facet as one of 3 danger levels (healthy, warning or critical), which could be assigned a corresponding primary colour for visualisation purposes (green, blue and red respectively).

While this the existing triangulation and classification approach met the needs for the stability analysis in terms of identifying regions of greater toppling risk, it was recognised that the information could be captured more succinctly and appropriately by grouping similar facets together into meta-regions of identical health rating. This was achieved by repeating the facet colourisation onto a blank image matching the input size of the video source, which could subsequently be segmented by splitting the image into its individual red, green and blue colour channels, thereby isolating each void class. An automated Otsu thresholding operation was then applied to each channel, separating the foreground (containing the appropriate void regions) from the background. This served to merge adjacent regions of the same class due to their single common colour and lack of distinguishable boundaries, forming the reduced set of meta-regions from the dense triangulation. The contours were subsequently extracted per channel and analysed to extract the number disconnected void regions per class, as well as additional position and size information, providing an indication of macroscopic line health with regards to toppling risk which could be reported. The complete toppling assessment procedure is summarised visually in Figure 11 for clarity.

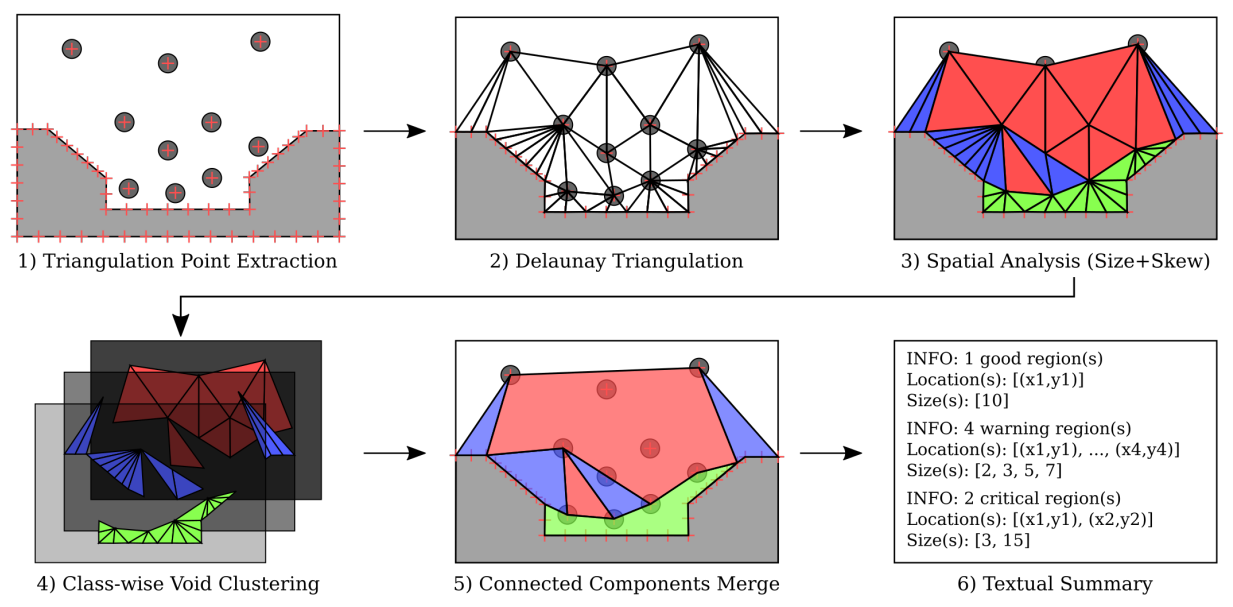


Figure 11: Diagrammatic overview of toppling risk assessment procedure

4 Results and Discussion

Having completed the experimental phase of the project, a detailed evaluation of the designed pipeline required. This was conducted at both an individual component level (including the detector, tracker, and health analysis components) and combined system level, using appropriate measures to compare the different methods and overall effectiveness.

4.1 Detector Performance

Accuracy and detection speed were the most import measures of performance for the detection algorithms. Detection speed was measured directly as the time to process an image (in milliseconds), with a lower detection time preferred. Average Precision (or AP) was used to score detection accuracy, being the standard metric for object detection tasks, where a higher value is better.

In order to calculate the AP, first the Intersection-Over-Union (IOU) of the detected and ground truth bounding box was computed as follows:

$$\text{IOU} = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \tag{5}$$

where B_p is the predicted bounding box and B_{gt} is the ground truth bounding box.

The IOU evaluates the level of agreement between the two bounding boxes, and so the detections could then be classified as either true positives (IOU ≥ 0.1) or false negatives (IOU < 0.1). A low IOU threshold was used in order to reduce the need for matching bounding box dimensions, given the variability in proposed bounding box shapes for the different methods, and the limited impact this would have on the remainder of the analysis pipeline. The detection results were subsequently sorted by the confidence of the detection algorithm (dynamically computed by the Scaled-YOLOv4 algorithm and statically assigned for the other methods), which were then used to calculate the cumulative precision and recall values, as illustrated by Equations 6 and 7.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} = \frac{\text{True Positives}}{\text{All Detections}} \tag{6}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \frac{\text{True Positives}}{\text{All Ground Truths}} \tag{7}$$

The 11-point average precision [65] could then be calculated by finding the average precision of the recall at 0.1 intervals between 0 and 1, as shown by:

$$\text{AP} = \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} \max_{\tilde{r}: \tilde{r} \geq r} \rho(\tilde{r}) \tag{8}$$

where ρ is the precision, and $\tilde{r}$ is the (sampled) recall.

Detection speed and accuracy was tested for each method across images with 5, 50, and 500 objects, in order to assess their scalability. The results are presented in Figure 12.

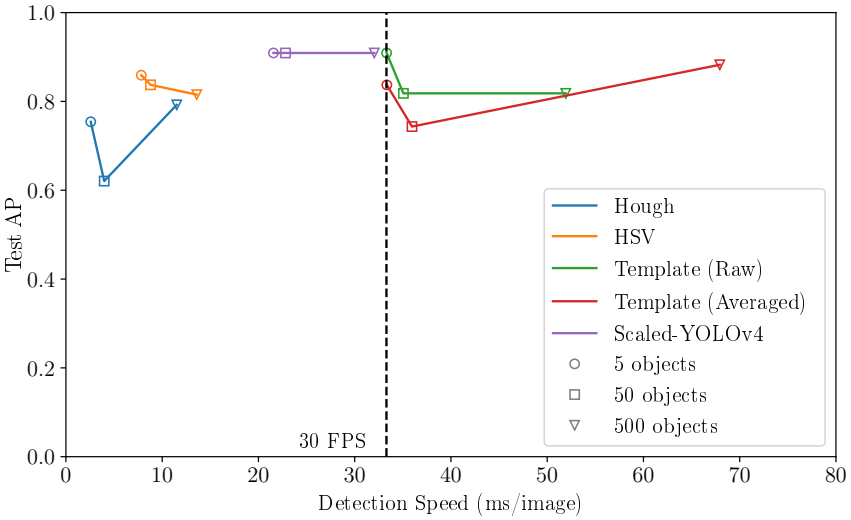


Figure 12: Detector accuracy and speed for increasing object count

Detection speed is shown to decrease with an increasing object count for all methods. This was caused by the need to iterate over all detection candidates to generate bounding box proposals, forming a $\mathcal{O}(n)$ operation at best, which scales linearly with input size.

Object count appears to have varying effects on the detection accuracy. The HSV and Raw template matching approach are shown to decrease in accuracy with increasing object count. The HSV approach performs worse at larger numbers due to the less distinct visual boundary between adjacent packs at a high packing density which reduces detection accuracy, whereas the raw template matching approach suffers because of the increased deviation from the template image. The raw template image depicted an isolated vial at the centre of the frame, thereby ignoring the visual influence of surrounding packs and the illumination of the side wall that occurs when off-centre, which dominate the image at high packing densities. The results from the averaged template matching approach supported this idea, showing greatest performance with 500 objects, due to the inclusion of the aforementioned artefacts through an averaged template. Worse accuracy is observed at lower object counts, where the vial appearance is better approximated by the raw template due to the lack of contact. In practise, the averaged approach would likely work better, with close-packing observed with any number of vials due to an accumulator's tendency to group objects together.

The Hough detection approach is shown to achieve its greatest accuracy with 500 objects. This was caused by its inability to differentiate the base and neck of the vial, detecting both features and thereby accounting for many false detections. This was only an issue at sparser configurations (with fewer objects), with the base of the vials occluded in the 500 object scenario, and hence resulting in a higher accuracy overall through a reduction in false detections.

The Scaled-YOLOv4 approach achieved consistent detection accuracy across all scenarios. This was likely a direct result of the quality of the training data, providing sufficient training examples across each

detection scale and thereby adequately teaching the algorithm how to consistently detect the vials with high accuracy. This is in contrast to the other approaches, which exhibited a bias towards a particular scale or configuration demonstrated by an inconsistent AP.

An optimal detection algorithm for the intended use case requires good accuracy and detection speed across all the test scenarios. While the template matching approaches demonstrated decent accuracy and superior ease of setup (requiring a single reference image), they exhibited poor detection speed at under 30 FPS throughout, and high sensitivity to the target appearance demonstrated by a varied preference in template, rendering them unsuitable. The HSV approach was fairly fast and accurate, however required extensive tuning with reference to the colour thresholds and size constraints, which suggests high sensitivity to the imaging conditions and difficulty in adapting to new packs, especially where colour may be less consistent or separable. The Hough circle detection approach is the fastest and achieves a decent accuracy, however is comparatively the least accurate of the methods explored, as well as being overly reliant on a circular shape and requiring extensive parameter tuning, which can introduce challenges if wanting to apply the pipeline to a different product. The Scaled-YOLOv4 detector demonstrates real-time performance and superior accuracy across all scenarios, however requires GPU-accelerated compute to achieve such performance, as well as a large set of training data, which can be expensive to produce. To this extent, synthetic data was demonstrated as a viable and preferable substitute for real-world data, justified by the success achieved with the synthetically-trained detector model in this body of work. The results achieved with the Scaled-YOLOv4 model are idealised through the use of the same synthetic environment for generating the training and test set, however the insights remain valid, as such an approach is analogous to forming a highly realistic synthetic environment in which differences between the virtual and real-world appearance are minimised.

Overall, the Scaled-YOLOv4 detection approach appears best-suited for the processing pipeline, especially if augmented with a synthetic data generation pipeline. It demonstrates good speed and accuracy, as well as high adaptability provided sufficient quality and variety is achieved in the training data. This also extends to different products, either as single-class or multi-class problem, which the Scaled-YOLOv4 algorithm natively supports with minimal setup and regardless of appearance, highlighting its increased versatility over the other methods which require extensive tuning and suffer algorithmic detection constraints. The high resource requirements associated with this detection approach remains its main limitation, especially with regards to deployment on an embedded system. Quantisation and network pruning are techniques that can help reduce the complexity and memory footprint for greater compatibility and higher processing speeds, however a performance penalty is often incurred in the process, and a GPU remains necessary. The HSV and Hough detection approaches are suitable alternatives where a GPU is not accessible, with the most appropriate method selected based on the use case, particularly mindful of the shape and colour of the object.

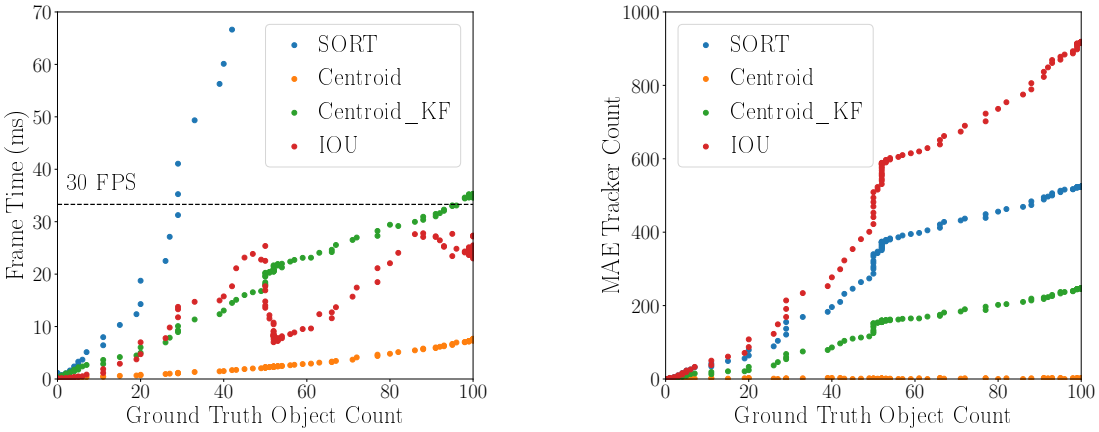
4.2 Tracker Performance

The tracker methods were compared based on the speed and robustness of the different methods when applied to the footage of the simulation environment. The tracker update time (in milliseconds) was used to assess the speed, for which a lower value was better. The robustness was assessed by calculating

the Mean Absolute Error (MAE) in the cumulative tracker count compared to the ground truth object count, as shown by:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{9}$$

where y_i is the predicted value and $\hat{y}_i$ is the true value. A lower MAE indicated better robustness, corresponding to a smaller discrepancy between the counts which is minimised by avoiding tracker dissociation and the subsequent initialisation of new trackers.



(a) Frame time for different trackers

(b) MAE tracker count for different trackers

Figure 13: Performance comparison of different trackers throughout the benchmark

Figure 13a demonstrates an increasing frame time with increasing object count across all trackers, which is caused by a greater number of calculations required for the matching process. The SORT tracker performed the worst with regards to frame time, exhibiting a steep linear or even quadratic relationship and thereby reaching the 30 FPS real-time limit at only 27 objects. This was likely caused by the complexity of the SORT algorithm, utilising a relatively costly overlap-based matching schema in addition to uncertainty estimation, which form a significant contribution to frame time, especially when processing a large number of items. In contrast, the centroid tracker was the fastest approach, remaining below 10 ms per frame, even with 100 objects to track. This can be attributed to the relative simplicity of the method, performing a distance-based linear sum assignment, which is inherently simpler and more efficient to compute. This is reflected by a roughly linear relationship between object count and frame time in the graph. The Kalman-filtered centroid tracker is functionally similar to the standard centroid tracker, albeit with the addition of uncertainty estimation, similar to the SORT algorithm. This increased complexity manifests as a higher frame time for a given object count, resulting in a linear relationship with a higher proportionality constant, although not to the extent of the SORT algorithm through the simpler matching schema. The IOU tracker's frame time demonstrated the most complex relationship, following an upward sawtooth pattern over the observed duration, with an overall upward trend observed albeit with the inclusion of an abrupt drop in frame time at 50 objects. While the overall increase is a common feature among the trackers caused by the increasing cost associated with additional tracker targets, the sharp decline appears more method-specific. Lost or dissociated trackers are the probable cause for the frame time reduction, with a dense packing event of vials within the accumulator basin occurring at

the ground truth object count of 50 items. As the IOU tracker relies on a track-by-overlap framework, the denser tessellation of objects increases the difficulty of association through an increase in spurious bounding box overlaps, which reduces the matching accuracy and subsequent tracker dissociation. This reduction in tracked objects results in fewer computations, thereby decreasing the frame time.

Figure 13b demonstrates an increasing MAE with increasing object count for all methods. Of the methods that were explored, the Kalman-filtered centroid tracker, IOU tracker and SORT approaches appeared to follow a similar pattern albeit with different vertical scaling, exhibiting a marked increase in MAE at around 27 objects and 50 objects. These instances correspond to when the newly introduced vials in the simulation collided with the accumulator and its contents, and were subsequently brought to rest. Abrupt velocity changes occurred during the collision event, further exacerbated by the high initial pack velocity resulting from constant acceleration and the omission of a terminal velocity. This interfered with the constant-velocity assumption for the SORT and Kalman-filtered centroid tracker approaches, providing a larger change upon collision than these trackers expected, and resulting in subsequent failed associations. SORT and the IOU tracker were also disadvantaged by the velocity of the vials, with the large magnitude reducing the bounding box overlap of a given object in subsequent frames, which was required for successful matching. The IOU tracker was further penalised by the subsequent tessellation of vials within the accumulator, causing a global increase in bounding box overlaps which complicated the association process. The standard centroid tracker did not exhibit the same sharp increase in MAE, which was likely a direct result of the simplicity of its distance-based assignment process which is inherently more resistant to the challenges faced by the other methods. This is further demonstrated by its superior MAE at the end of the benchmark, having a final error of under 10, which is an order of magnitude better than the other approaches.

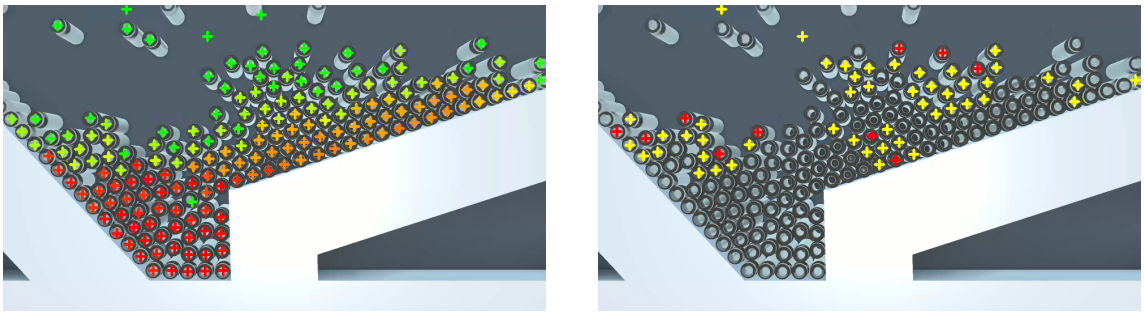
Considering the combined information from Figure 13, the centroid tracker appears to be optimal tracking approach given the high speed and robustness demonstrated when compared with the other approaches. While the standard centroid tracker performs the best in the given benchmark, it is acknowledged that these outcomes may have been somewhat influenced by the simulation design, with the excessive and unrealistic speeds and resulting implications implying worse robustness than would otherwise be expected. Further to this, the lack of occlusions artificially favoured the standard centroid tracker, failing to address the sharp decrease in robustness that would result from an obstruction due to the methods entirely positional operation and omission of motion understanding. An uncertainty-based method would be more appropriate in this case, with the Kalman-filtered centroid tracker being the ideal candidate considering its slight reduction in speed over the standard method, while providing an increased robustness through the Kalman filter's motion prediction capabilities.

4.3 Evaluation of Health Assessment Procedures

Two health assessment procedures were developed in the project for assessing damage and toppling risk respectively. A quantitative evaluation of methods was deemed inappropriate due to the absence of real-world data, and subsequent lack of system calibration. Instead, the techniques were evaluated qualitatively with regards to the capabilities and commercial suitability demonstrated within the virtual environment.

4.3.1 Damage Assessment Procedure Evaluation

The damage assessment procedure provided an overall indication of vial damage through a weighted combination of cumulative damage (time-based) and collision damage (acceleration-based). Realistic weight allocation was hindered by the lack of experimental data, reducing the value of a combined evaluation and therefore considering the individual contributions instead, as presented in Figure 14.



(a) Cumulative damage contribution visualisation (b) Collision damage contribution visualisation

Figure 14: Damage assessment procedure (individual contributions)

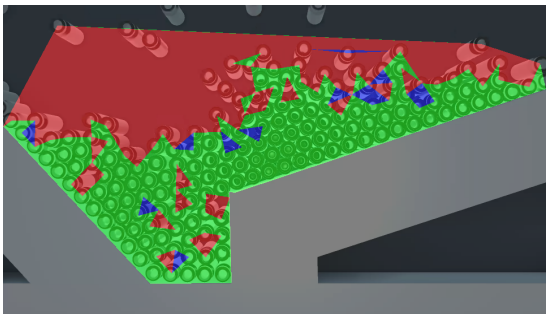
The cumulative damage contribution is visualised in Figure 14a using a colour gradient from green to red to denote increasing residence time. Residence time extraction appears to be implemented effectively, presenting an accurate depiction of the historic behaviour of the system through residence time colourisation. The red accumulator basin suggests an initial migration towards this site, followed by an overspill of packs onto the right shoulder upon saturation (orange annotations), and a subsequent introduction of newer packs towards the top of the image (green annotations). These insights largely match the events of the simulation, supporting the functionality of this assessment stage, as well as indicating wider applicability of this technique. Aside from the contribution to the damage formulation, the residence time metric could be used to help visualise and understand product flow, with particular emphasis on identifying stagnation zones and regions of recirculation or turbulence, the presence of which may otherwise limit the throughput of the production line, or exaggerate accumulated damage.

The collision damage contribution is visualised in Figure 14b using red for primary collision targets (sources of abrupt acceleration change) and yellow for secondary targets (recipients of close-proximity damage propagation). Collision proposals are largely sensible, identifying sites that visually appear to be undergoing a collision event, such as initial impacts of introduced packs at the top of the image, and the collisions deeper within the accumulator likely resulting from a morphing pack arrangement. While this approach appears successful, it is noted that small inter-frame changes could provide a challenge due to the discrete nature of video feeds and subsequent rounding issues that affect these small scale impacts, which in themselves likely form the majority of collision events.

The damage assessment pipeline demonstrates clear commercial value through the analytical capabilities offered at the constituent level, and the potential benefit of the combined formulation. The reliance on robust detection and tracking suggests rigorous tuning requirements for persistent damage and prevention of anomalous analysis results, such as those caused by false detections at the top of Figure 14a and 14b.

4.3.2 Toppling Risk Assessment Procedure Evaluation

The toppling risk assessment procedure analysed pack instability based on the size (area) and shape (regularity) of void regions. The output of a frame of analysis consisted of a colour-coded density map and textual summary, as shown in Figure 15.



(a) Toppling risk assessment density map

```
INFO: 4 good regions
Location(s): [[425, 272], [433, 97], [629, 137], [430, 169]]
Size(s): [107150, 373, 439, 360]

INFO: 11 warning regions
Location(s): [[613, 80], [668, 128], [738, 159], [648,
158], [592, 158], [48, 224], [442, 223], [399, 242], [325,
242], [253, 353], [309, 445]]
Size(s): [241, 468, 1058, 1076, 471, 517, 538, 577, 508,
552, 510]

INFO: 7 critical regions
Location(s): [[366, 132], [505, 152], [315, 324], [376,
350], [279, 379], [357, 415], [290, 445]]
Size(s): [107247, 680, 1228, 1289, 686, 1189, 628]
```

(b) Textual summary of toppling risk assessment

Figure 15: Toppling risk assessment procedure outputs

Figure 15a demonstrates the method's ability to reliably identify voids and assess their relative stability. A clear difference in packing density is evident between the green (good) and red (critical) regions, corresponding to densely packed (stable) and more sparsely distributed (unstable) arrangements respectively. Several blue (warning) regions are also observed throughout the image, having little value at present due to the lack of calibration, however having the potential to identify early-stage void emergence if tuned accordingly. The critical and warning regions located within the stable zone are of particular interest, being more representative of a real-world scenario. These highlighted regions clearly show pack separation, demonstrating the procedure's effectiveness for automatically extracting these formations from within a crowded and complex frame. The subsequent summarised output (shown in Figure 15b) successfully reduces the informational complexity and provides a more succinct indicator of line health, and one that is more readily interpretable by IT systems for improved compatibility and commercial feasibility.

4.4 Combined System Performance Evaluation

A benchmark was performed to assess system performance, as presented in Figure 16. The benchmark spanned 407 frames, with vials introduced at frame 55, followed by a periodic addition of batches of vials up to a maximum of 296 objects by the end of the benchmark. This is reflected in the performance plot, with a sharp increase at frame 55 followed by a periodic increase thereafter, resulting from an increased number of computations with additional objects.

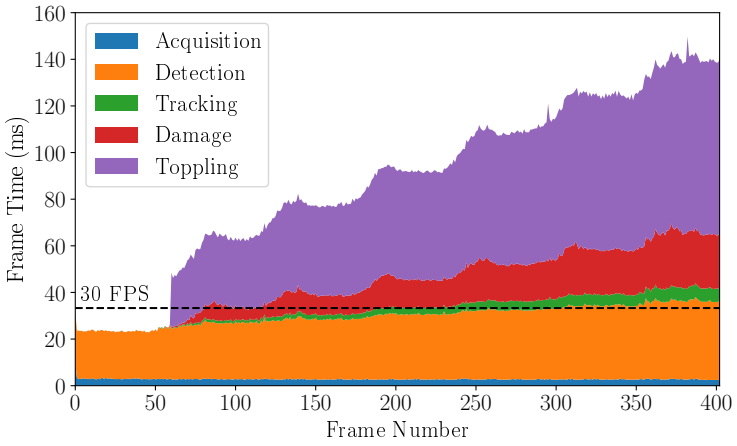


Figure 16: Performance profile for image processing pipeline

The overall system is shown to quickly exceed the 30 FPS real-time performance budget, with the toppling analysis procedure contributing 20 ms of execution time with only 3 objects, and increasing to an 80 ms contribution by the end of the benchmark, easily saturating the performance budget, and highlighting its poor suitability for online operation. While the damage analysis contribution appears to start small, this too demonstrates poor scalability as the benchmark progresses, increasing considerably to a 25 ms execution time by the end of the benchmark, once again consuming the entire performance budget when accounting for the constant 4 ms contribution from the acquisition stage. While algorithmic optimisation could help improve these frame times, a 33 ms performance target remains infeasible, as demonstrated by the contribution provided by the core detection and tracking framework, which while under the 30 FPS limit at the start, eventually exceed this at frame 200 (140 objects), making any further analysis and processing infeasible, especially given their relative contributions at these object counts.

These results appear to suggest that the proposed end-to-end processing pipeline would be infeasible for 30 FPS operation, with the core framework unable to meet this goal at large numbers, excluding the further penalty attributed to the addition of the analysis layer. To this extent, the video feed could be sampled at a lower frame rate such as 10 FPS, providing a more lenient performance budget which would easily support the detection and tracking functions throughout the entire benchmark. The remaining performance budget would be sufficient to support the damage analysis functionality across the whole frame range or the complete analysis procedure up to frame 230 (150 objects), and even further if performed concurrently. Suitability of a lower frame rate is intrinsically related to the speed of operations being monitored, with a reduction in frame rate exacerbating the frame-wise disparity, especially for fast-moving objects, which could negatively effect the distance-based tracking procedure.

An alternative modification to improve feasibility would be to reduce the region of interest through a narrower field of view camera or closer placement to the line. This would thereby reduce the number of objects that could fit within the frame and hence the maximum frame time given its linear relationship with visible object count. Potential drawbacks would include a need for more cameras for line-wide monitoring given the smaller observable region, as well as greater positional disparity if achieved by

moving the camera. While non-negligible, the shortcomings of such mitigating actions would be easily justified by the commercial appeal of the combined pipeline, with its ability to monitor the major failure modes (toppling and crushing) providing valuable insights in diagnosing production line faults, which are otherwise associated with high inflicted costs.

5 Conclusions

Computer vision techniques show great potential for industrial monitoring applications, demonstrating superior versatility and assessment capabilities when compared with other approaches. This project considered the development of an image processing pipeline for assessing line health in the context of glass vial production.

Synthetic data generation techniques were shown to be effective in facilitating pipeline development in spite of an absence of real-world data. A range of detection and tracking techniques were evaluated using the synthetic data, from which the Scaled-YOLOv4 detection algorithm and centroid tracker algorithm were identified as the optimal methods with regards to speed, accuracy and robustness. A novel analysis layer was developed to characterise toppling risk and damage, and was shown to demonstrate a high level of capability and wider commercial applicability through the richness of the provided insights.

The insufficient performance of the combined pipeline ultimately restricts its feasibility for real-time analysis in its current form. Frame sampling and region of interest reduction were proposed to help improve feasibility, given the potential value of such a system for complex production line analytics.

6 Recommendations for Further Work

Further work should prioritise the calibration and validation of these systems in a real-world context, demonstrating the commercial applicability and feasibility of the system. This is especially important given the novelty of the analysis layer, requiring further calibration against experimental data to provide a physically-based quantification of the risk of toppling and crushing.

Improvements to the proposed pipeline would primarily revolve around software optimisation, such as reimplementation in a lower level language (such as C++), and pruning and quantisation of the object detection network. These modifications would reduce overhead and memory footprint, improving the feasibility of embedded system deployment, and hence providing greater commercial appeal.

Integration with the supporting research streams would provide added value and relevance to the proposed image processing pipeline. The outputs from the damage modelling and bulk flow understanding work packages could help contextualise the visual health metrics and inform the tuning procedures of the damage and toppling risk assessment approaches respectively. The mathematical modelling research could be integrated within the visual damage estimation process, providing a more relevant strength-based expression of health, which could be allocated from a distribution on the basis of the scalar damage quantity per vial. The second year of the project will realise this system integration exercise, demonstrating the value proposition through a scale-model deployment of the integrated analysis procedure.

Bibliography

[1] M Blamont and L Burger. Exclusive: Bottlenecks? glass vial makers prepare for covid-19 vaccine, 2020. URL https://www.reuters.com/article/us-health-coronavirus-schott-exclusive-idUSKBN23J0SN. [Accessed 28th April 2021].

[2] Advances in vial processing in pharmaceutical primary packaging to reduce risk of vial damage and particle contamination - pharmaceutical processing world. URL https://www.pharmaceuticalprocessingworld.com/advances-in-vial-processing-in-pharmaceutical-primary-packaging-to-reduce-risk-of-vial-damage-and-particle-contamination/. [Accessed 28th April 2021].

[3] NJ Shaw and EG Lyall. Hazards of glass ampoules. *BMJ*, 291(6506):1390–1390, 1985. doi: 10.1136/bmj.291.6506.1390.

[4] S Karlén. Reduce machine stops to limit human intervention in aseptic production, 2020. URL https://lup.lub.lu.se/student-papers/search/publication/9029267. Student Paper.

[5] Corrective and preventive actions (capa), 2014. URL https://www.fda.gov/corrective-and-preventive-actions-capa. [Accessed 28th April 2021].

[6] Corrective and preventive action: The closed-loop system. Technical Report 9.23.13, The British Standards Institution., 2013. URL https://www.bsigroup.com/LocalFiles/en-US/Whitepapers/Quality%20Management/bsi-whitepaper-capa-closed-loop-system.pdf. [Accessed 28th April 2021].

[7] K Miller. The impact of industry 4.0 on quality management systems, 2018. URL https://www.pilgrimquality.com/blog/smart-quality-management-impact-industry/. [Accessed 28th April 2021].

[8] Crux product design. URL http://www.cruxproductdesign.com/. [Accessed 28th April 2021].

[9] Industry 4.0: Challenges and solutions for the digital transformation and use of exponential technologies. Technical report, Deloitte., 2015. URL https://www2.deloitte.com/content/dam/Deloitte/ch/Documents/manufacturing/ch-en-manufacturing-industry-4-0-24102014.pdf. [Accessed 28th April 2021].

[10] G Knutson. Plant engineering | using photoelectric sensors on the production line, 2008. URL https://www.plantengineering.com/articles/using-photoelectric-sensors-on-the-production-line/. [Accessed 28th April 2021].

[11] Control Engineering Staff. Control engineering | rfid on the production line, 2005. URL https://www.controleng.com/articles/rfid-on-the-production-line/. [Accessed 28th April 2021].

[12] J Brusey and DC McFarlane. Effective rfid-based object tracking for manufacturing. *International Journal of Computer Integrated Manufacturing*, 22(7):638–647, 2009.

[13] Reduce glass breakage, bruising, microfractures with schott & smartskin | schott ag. URL https://www.schott.com/pharma_consulting/english/smartskin/index.html?WTC=EN_Pharma-Consulting_Smartskin_First. [Accessed 28th April 2021].

[14] R Ciora and C Simion. Industrial applications of image processing. *ACTA Universitatis Cibiniensis*, 64, 12 2014. doi: 10.2478/aucts-2014-0004.

[15] JDD Cabral and SA de Araújo. An intelligent vision system for detecting defects in glass products for packaging and domestic use. *The International Journal of Advanced Manufacturing Technology*, 77(1-4):485–494, 2015.

[16] N O'Mahony, S Campbell, A Carvalho, S Harapanahalli, GV Hernandez, L Krpalkova, D Riordan, and J Walsh. Deep learning vs. traditional computer vision. In *Science and Information Conference*, pages 128–144. Springer, 2019.

[17] PC Lien and Q Zhao. Product surface defect detection based on deep learning. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pages 250–255. IEEE, 2018.

[18] Roboflow: Everything you need to start building computer vision into your applications. URL https://roboflow.com/. [Accessed 28th April 2021].

[19] Amazon mechanical turk. URL https://www.mturk.com/. [Accessed 28th April 2021].

[20] F Lardinois. Roboflow raises $2.1m for its end-to-end computer vision platform | techcrunch, 2021. URL https://techcrunch.com/2021/01/12/roboflow-raises-2-1m-for-its-end-to-end-computer-vision-platform/. [Accessed 28th April 2021].

[21] H Liu, S Chen, and N Kubota. Intelligent video systems and analytics: A survey. *IEEE Transactions on Industrial Informatics*, 9(3):1222–1233, 2013.

[22] WT Neale, D Hessel, and T Terpstra. Photogrammetric measurement error associated with lens distortion. Technical report, SAE Technical Paper, 2011.

[23] A Hanbury. A survey of methods for image annotation. *Journal of Visual Languages & Computing*, 19(5):617–627, 2008.

[24] C Shorten and TM Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[25] R Sieczka and M Pańczyk. Blender as a tool for generating synthetic data. *Journal of Computer Sciences Institute*, 16:227–232, 2020.

[26] J Tremblay, A Prakash, D Acuna, M Brophy, V Jampani, C Anil, T To, E Cameracci, S Boochoon, and S Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977, 2018.

[27] Q Wang, J Gao, W Lin, and Y Yuan. Pixel-wise crowd understanding via synthetic data. *International Journal of Computer Vision*, 129(1):225–245, 2021.

[28] W Qiu and A Yuille. Unrealcv: Connecting computer vision to unreal engine. In *European Conference on Computer Vision*, pages 909–916. Springer, 2016.

[29] Q Wang, J Gao, W Lin, and Y Yuan. Learning from synthetic data for crowd counting in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8198–8207, 2019.

[30] Nvidia clara | nvidia developer. URL `https://developer.nvidia.com/clara`. [Accessed 28th April 2021].

[31] Nvidia drive - autonomous vehicle development platforms | nvidia developer. URL `https://developer.nvidia.com/drive`. [Accessed 28th April 2021].

[32] Nvidia isaac sim | nvidia developer. URL `https://developer.nvidia.com/isaac-sim`. [Accessed 28th April 2021].

[33] E Hamuda, B Mc Ginley, M Glavin, and E Jones. Automatic crop detection under field conditions using the hsv colour space and morphological operations. *Computers and electronics in agriculture*, 133:97–107, 2017.

[34] VF Leavers. *Shape detection in computer vision using the Hough transform*, volume 1. Springer, 1992.

[35] K Briechle and UD Hanebeck. Template matching using fast normalized cross correlation. In *Optical Pattern Recognition XII*, volume 4387, pages 95–102. International Society for Optics and Photonics, 2001.

[36] S Ren, K He, R Girshick, and J Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[37] J Redmon, S Divvala, R Girshick, and A 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.

[38] C Wang, A Bochkovskiy, and HM Liao. Scaled-yolov4: Scaling cross stage partial network, 2021.

[39] G Pavlakos, X Zhou, A Chan, KG Derpanis, and K Daniilidis. 6-dof object pose from semantic keypoints. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2011–2018. IEEE, 2017.

[40] P Lai and C Fuh. Transparent object detection using regions with convolutional neural network. In *IPPR Conference on Computer Vision, Graphics, and Image Processing*, volume 2, 2015.

[41] MP Khaing and M Masayuki. Transparent object detection using convolutional neural network. In *International Conference on Big Data Analysis and Deep Learning Applications*, pages 86–93. Springer, 2018.

[42] A Bewley, Z Ge, L Ott, F Ramos, and B Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[43] E Bochinski, V Eiselein, and T Sikora. High-speed tracking-by-detection without using image information. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2017.

[44] JC Nascimento, AJ Abrantes, and JS Marques. An algorithm for centroid-based tracking of moving objects. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99*, volume 6, pages 3305–3308. IEEE Computer Society, 1999.

[45] N Wojke, A Bewley, and D Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[46] S Sun, N Akhtar, X Song, H Song, A Mian, and M Shah. Simultaneous detection and tracking with motion modelling for multiple object tracking. In *European Conference on Computer Vision*, pages 626–643. Springer, 2020.

[47] RA Schaut, KC Hoff, SE Demartino, WK Denson, and RL Verkleeren. Enhancing patient safety through the use of a pharmaceutical glass designed to prevent cracked containers. *PDA journal of pharmaceutical science and technology*, 71(6):511–528, 2017.

[48] H Veeraraghavan, O Masoud, and NP Papanikolopoulos. Computer vision algorithms for intersection monitoring. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):78–89, 2003.

[49] J Lai, L Mejias, and JJ Ford. Airborne vision-based collision-detection system. *Journal of Field Robotics*, 28(2):137–157, 2011.

[50] P Speleers and M Ebner. Acceleration based collision detection with a mobile robot. In *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pages 437–438. IEEE, 2019.

[51] C Rougier, J Meunier, A St-Arnaud, and J Rousseau. Fall detection from human shape and motion history using video surveillance. In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, volume 2, pages 875–880. IEEE, 2007.

[52] SA Velastin, JH Yin, AC Davies, MA Vicencio-Silva, RE Allsop, and A Penn. Automated measurement of crowd density and motion using image processing. In *Seventh International Conference on Road Traffic Monitoring and Control, 1994.*, pages 127–132. IET, 1994.

[53] N Abbas, M Tayyab, and MT Qadri. Real time traffic density count using image processing. *International Journal of Computer Applications*, 83(9):16–19, 2013.

[54] C Duyckaerts and G Godefroy. Voronoi tessellation to study the numerical density and the spatial distribution of neurones. *Journal of chemical neuroanatomy*, 20(1):83–92, 2000.

[55] Enterprise open source and linux | ubuntu. URL `https://ubuntu.com/`. [Accessed 28th April 2021].

[56] Welcome to python.org. URL `https://www.python.org/`. [Accessed 28th April 2021].

[57] Opencv. URL https://opencv.org/. [Accessed 28th April 2021].

[58] Numpy. URL https://numpy.org/. [Accessed 28th April 2021].

[59] Unity real-time development platform | 3d, 2d vr & ar engine. URL https://unity.com/. [Accessed 28th April 2021].

[60] JS Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

[61] Gameworks physx overview | nvidia developer. URL https://developer.nvidia.com/gameworks-physx-overview. [Accessed 28th April 2021].

[62] R Bridson. Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches*, 10:1, 2007.

[63] J Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[64] AM Deshpande. Multi-object trackers in python, 2020. URL https://github.com/adipandas/multi-object-tracker. [Accessed 28th April 2021].

[65] M Everingham, L Van Gool, CKI Williams, J Winn, and A Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.