Report Criteria

* Report of research into methodologies that would be attempted given more time.
* Report should cover possible techniques, pros/cons of each technique.
* Cost of implementation
* Potential issues, methods could face when running in production.

**Introduction**

Description of the recorded video. Video shows a top-down view of a lab technician filling several stacks of petri dishes with a chemical. There are a few equipment’s placed on the sides and the work appears to be carried out in an enclosed environment. The required task for the provided video is listed below in order of priority.

* Identify the different objects in the video.
* Determine the interactions between the objects in scenes in the video.
* Track the count of filled dishes
* Identify any additional vision-based insights in the scene.

The task presented can be described as an object detection & tracking problem, for which the objects in the video are to be detected and tracked and the interactions between the objects in the video should be logged. Components of the task have been broken down into segments listed below.

* Object detection
* Tracking
* Colour based detection
* Event Logging.

**Summary of approach taken**

Towards addressing the task listed above, classical computer vision techniques (HoughCircles & colour-based thresholding) have been applied to identify the different objects in the video (petri-dish, gloves worn). Traditional techniques were first considered as it seemed, there are many features (geometric features, colours, crafted features) within the video that could be easily detected by traditional detectors with little effort and cost. The petri-dish are circular with a consistent radius (with a reasonable margin for error) so could be picked up by OpenCV’s HoughCircles detector. The gloves worn by the lab technician are distinctive throughout the video and could be detected by looking for a particular range of colour within each frame. Effectively detecting the chemical bottle in the video in the video would require a deep learning approach as the traditional methods would require more tuning and more effort than required. Results using HoughCircles and Colour based detection are shown below in figure 1 & experimenting with sift feature extraction & matching are shown below in figure 2.

|  |  |
| --- | --- |
| **A group of plastic cups  AI-generated content may be incorrect.** | **A screenshot of a computer  AI-generated content may be incorrect.** |
| **A screenshot of a computer  AI-generated content may be incorrect.** | |

Figure 1. Top left shows results using houghcircle detector. Top right shows result using colour thresholding and bottom show result detecting of composing both detectors onto a single frame.

|  |  |
| --- | --- |
| **A person holding a red and white object  AI-generated content may be incorrect.** | **A close-up of a container  AI-generated content may be incorrect.** |
| **A group of plastic cups  AI-generated content may be incorrect.** | **A person in gloves pouring liquid into a glass  AI-generated content may be incorrect.** |

Figure 2. Applying SIFT feature extraction and matching. With the extracted features from a target image (top left) matched against features extracted from subsequent frames in the video (remaining frames).

Towards detecting when the petri-dish has been filled, colour-based thresholding was further applied on the detected HoughCircles. The idea being that we can detect the presence of the chemical on the petri dish by looking for a narrow range of colours within each detected circle.

Main benefit of resulting method is that it runs fairly fast and outperforms object detection models (experimented with yolo and faster rcnn) in terms of inference speed without any optimization and access to GPU. Downside is it provides very noisy detections with occasional false detections (shown in figure 1). Sources of false detections include reflections from surface while using colour-based thresholding, for HoughCircles, objects with similar geometric features to the petris dish were detected. Another downside with this approach is that it required fine tuning to the video and is possibly not transferable to other video without further adjustments. Towards addressing some of the limitations with the traditional method. The noisy detections could be smoothed out with a Kalman filter tracker, with strict conditions for track creation and deletion and the noise in the detection accounted for with the filters measurement covariance.

With regards to applying a deep learning-based solution, a pretrained model from PyTorch was adapted and fine-tuned to detect the petri dish and gloves within the video. This approach required some effort to generate a full dataset of images and labels in the COCO format. A custom PyTorch dataset class was implemented to load and return the features and labels in the required format. frames from the original video were sampled at every 30 frames (resulting in 300 images with automated annotations) and the OpenCV detectors were used to generate labels. The individual annotations were then cleaned to remove any false positive detections, but no effort was made to add any missed detections. The model was then trained on the cleaned annotation on Google’s colab.

The benefit of this approach is that it offers a more robust solution to detect the desired objects in the video. A possible con with this approach would be the reduced inference speed without additional optimisation or with limited access to GPU. Also as the model was only trained on data gathered from a single video, it would require fewer diverse training sample, but it could also easily overfit to features specific to the video and would have limited generalization beyond the provided video.

**Given more time methodologies that would have been attempted.**

**Object detection (self-supervised learning)**

Contrastive learning-based methods such as SimCLR’s contrastive loss could be used to pre-train models on pseudo labels generated from the original video. The pre-trained model could then be used to generate labels to further fine-tune the model for a supervised detection task. The main benefit with this approach would be the reduced dependency and cost of labelling data, although the cost of labelling data is effectively traded for high computational cost with having multiple training stage.

**Implementing a Kalman filter tracker**

As mentioned above, implementing a Kalman filter tracker would address the issues with the noisy detections, by smoothing over the noise, and it provides a means for identifying and assigning an id to each detected object. While the filter is useful in assigning a unique id to each detected object (which helps with counting the number of petri dish filled with chemical), it still could be limited by detector the filter is paired with, for example if the detector does not reliably detect the target object or picks up multiple false positive. Other tracking solutions such as Deep Sort and optical flow could also be explored.

Main benefits with the tracking solutions mentioned above is it offers tracking solution in event of occlusion and Deep Sort would be able to re-id the tracked object with the same id. Possible disadvantages with the tracking solutions are, I suspect it might fail when the petri dishes are stacked over each other or moved from one stack to another. As the tracked object from the tracker’s perspective would appear to remain in the same position despite being moved from one stack to another, possible solutions to counter this issue could be effective colour based detection and clever manipulation of each track state.

With regards to cost of implementation, that depends on the approach chosen. There are existing python packages to pair deep sort with deep learning object detection model, so there could be low implementation cost, but effectively utilizing it would require GPU access. The filter parameters might also need to be tuned to model the motion of the objects in the video.

**Determining interactions between objects**

Naïve solutions to determine interactions between objects in the scene, could include using colour-based thresholding checks on detected petri-dish to determine the pouring of the chemical into the petri dish. Similarly, intersection over union (IOU) could be used to determine when two different bounding boxes intersect with each other. This solution should be relatively simple to implement and would require little effort but a good detector to accurately detect each tracked object and to have each tracked object assigned to a label. So, an example would be if a track associated with the glove has a good iou score with the track associated with the chemical bottle, we can at least discern that both objects are close to each other and using further information about their velocities we can determine if the lab technician is currently holding the bottle.

**Testing and General improvements**

* Following test driven development (TDD) practices would have been good to follow from the start, but this was skipped in favour of more quicker development time. Adding tests to the written functions and class would be useful. As commentary on this, it seemed some of the OpenCV functions are actually using y,x and not xy format having test to confirm and document this would have been very useful.
* General linting
* With regards to model training, applying object detection metrics (mAP, IOU, Precision & recall) during model training/ testing would be useful beyond using training loss as an indication of model performance
* Instantiating a CVAT server to label the video would have been useful. Commentary on this, I did try to instantiate a local CVAT server but had some issues with starting up the server.
* Optimizing the fine-tuned model for improved inference speed. Could explore using OpenVino or converting into different formats or exploring other ml frameworks.

**Reference**

* <https://pyimagesearch.com/2014/07/21/detecting-circles-images-using-opencv-hough-circles/>
* <https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html>
* <https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html>
* <https://medium.com/@amit25173/opencv-object-tracking-project-29954e0f0418>
* <https://docs.opencv.org/4.x/df/d0c/tutorial_py_fast.html>
* <https://docs.opencv.org/4.x/d1/de0/tutorial_py_feature_homography.html>
* <https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html>
* <https://arxiv.org/abs/2002.05709>