Research Report

Problem overview

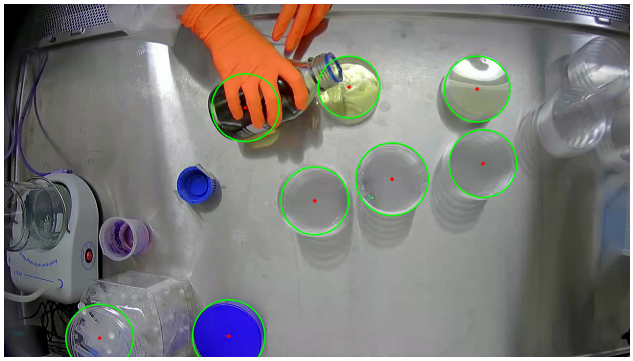
The task requires an application that can do object detection on a number of objects and from this determine some interactions between them. Given the time and resources available, a reasonable approach would be to: use a combination of classical and learnt detection methods, something to monitor the “world state” and keep track of objects even when detection fails, and some kind of heuristic to determine interactions.

Initial Classical Methods

Colour Segmentation
The hands are the only orange thing in the frame, so these can be reasonably detected with colour space thresholding. Converting the image to HSV and then thresholding on min and max orange HSV values produces a mask, from which hands can be detected by finding the largest contiguous regions.

This is a comparatively cheap method although it does require making a whole image copy. Execution time is still in the order of a few milliseconds. However, this method cannot discern between left and right hands and is not robust to certain types of occlusion resulting in non-contiguous regions - i.e. holding the bottle palm up separates the wrist from the fingertips.

Hough Circles
Given that the petri dishes are a fairly fixed size and are circular, the circle Hough Transform can be used to identify them in the image. From a region of interest pixel sample about the centre of the circles, the petri dish state can be classified as well as filtering out false positives based on the colour. Low mean saturation corresponds to an empty dish whereas yellow hues of different saturations and intensities can be used to classify dishes that are being filled or are filled with a lid on. The main advantage of this method is that it’s comparatively computationally cheap provided that the search space is appropriately constrained and it is reasonably robust to partial occlusion. Execution time on a M1 Macbook Pro is approximately 10ms.



Despite this, this method was still not as robust as initially hoped - colour space filtering is not robust to different positions in the hood and objects such as the lip of the plastic jar in the bottom right of the frame were often classified as an empty petri dish.

YOLO detector

Initially the plan was to use a pre-trined YOLO model to detect the bottle as this class is part of the MS COCO dataset. However, give the failure modes of the classical methods, it seemed prudent to try fine-tuning a YOLO model that could detect everything.

200 images were extracted from the sample video and labelled using Label Studio[1] with the labels:
1. bottle
2. bottle cap
3. petri dish empty
4. petri dish filled
5. petri dish filled with lid
6. left hand,
7. right hand

Then using the Ultralytics package, a YOLOv11[2] nano model was fine tuned with these new custom classes in a Google Colab notebook[3] for 60 epochs. Given the number of classes and constrained environment, the smallest “nano” model was sufficient to achieve robust detection while keeping execution time low, achieving real time speeds on a CPU.

This model, while it works well on the sample video, is very likely overfit to this scene. It would probably perform worse on a completely new video.

Interaction Heuristics

Events and interactions in the scene can be detected from heuristics based on the object detections. For example, counting new instances of the “petri dish filled” class can be used to keep track of how many petri dishes have been filled. Simple intersection area checks on detected object bounding boxes can be used to infer whether a hand is touching an object or whether a bottle has its cap removed or not.

These methods work to an extent but they have obvious shortcomings. Simple intersection area checks won’t account for hands moving over the top of other objects for example. More complex interactions such as topping up an already partially filled petri dish will also cause simple counting heuristics to fail. In order for more robust action recognition, a learned solution is likely.

Further work

Detection
YOLO v11 has an AGPL license and it would be preferable not to have to pay for an enterprise license. The Ultralytics package was used here for its ease of use in getting a proof-of-concept model trained in a short time period. There exist other implementations online of older YOLO versions with more permissive licenses that would likely work well enough for this application[4,5].

---

[1] https://labelstud.io/

[2] https://docs.ultralytics.com/models/yolo11/

[3] https://colab.research.google.com/drive/1coXG7L99l5mkGVLame3yv7-ncjFHYRH8?usp=sharing

[4] https://github.com/Lornatang/YOLOv4-PyTorch

[5] https://github.com/MultimediaTechLab/YOLO

Depending on future requirements, it might be prudent to produce our own network, whether that be a YOLO network or something DETR-like[6]

The network trained for this task is likely overfit to this specific video sequence and so further work would include creating a larger more diverse dataset from other videos taken from a lab setting. The Ultralytics train script does some image augmentation as part of its training procedure, though whether more sophisticated and varied augmentations could be applied ought to be investigated.

Action Recognition
Heuristic methods have their obvious shortcomings for detecting interactions and doing action recognition. A learned a approach should be explored. A good place to start would be looking at winners of the EPIC Kitchens Action Recognition competition as this is a particularly challenging dataset. Both AVION[7] and OpenTAD[8] look promising. However dataset curation for such tasks is time consuming and training can be expensive (although AVION claims SOTA despite training on consumer hardware). More research needs to be done in to what would be needed to train an action recognition model to determine whether it's worthwhile over attempting to craft more sophisticated heuristics.

---

[6] Zhao et al. "DETRs Beat YOLOs on Real-time Object Detection", 2024 IEEE/CVF Conference on Computer Vision, 2024

[7] Yue Zhao & Philipp Krähenbühl, "Training a Large Video Model on a Single Machine in a Day", CVPR 2023 https://github.com/zhaoyue-zephyrus/AVION

[8] Liu S et al. "Harnessing Temporal Causality for Advanced Temporal Action Detection", CVPR 2024 https://github.com/sming256/OpenTAD/