Computer Vision Poka-Yoke

Nuno Fernandes

Objective

The objective of this challenge was to develop a system to analyze a “Poka-Yoke” proofing system. Therefore, my aim was to develop a system that could be adapted to real-time video analysis. Additionally, throughout the task, I intended to use a wide range of computer vision techniques, such as movement tracking, edge and line detection, frame differencing, and deep learning. I delivered a video analysis of the four phases of the task: the pick-up movement, the probe test, the pen scratch, and when the person places the piece in the box. The video displays the number of operations, and for each operation, the number of successful probe tests and pen marks. It also presents the current and average duration in seconds for the entire operation, as well as the success ratio, expressing the percentage of operations in which 2 probes and 2 pen marks were performed. However, there is a wide range of potential improvements for this system, as it shows both false positives and negatives. Therefore, I will briefly explain my reasoning and techniques for the 4 phases, as well as the system's limitations and possible future improvements.

Preprocessing

Still images were extracted from the video at 30 fps. Then, I chose to downsize the data and extract one image every 5 frames. This decision sacrificed the potential cost of lower latency for the ease of testing different methods, as this was an initial approach. For implementation in real-time systems, I would not opt for this sacrifice.

Hand Coordinates Annotation - YOLO

Next, the YOLO model was trained to detect the position of each hand. For this, the provided training database with the corresponding coordinates was used. The model was then utilized to predict the annotations of each hand's position in the image.

Dominant Hand Detection

The system was designed to accommodate both right-handed and left-handed users. However, this is the only part of the system that requires an initial training phase for the user. The dominant/non-dominant hand was determined by the number of times it was not detected (non-dominant hand) when placing the piece in the box. The system stores information about whether the hand is more often on the left or right and calculates for each frame which hand is dominant/non-dominant, as this information is important in detecting operations. An alternative would be a classifier to detect which hand holds the pen in each frame.

Pick-Up

In this phase, the objective was to detect the movement pattern based on hand coordinates in relation to the previous frame and establish a classifier algorithm based on these criteria. A decision had to be made regarding which movement to consider—whether the moment when the operator grabs the piece but it's still on the table, e.g., to drag other pieces, or the moment when the piece is effectively lifted. I chose to consider the movement when the piece is lifted from the table. For this, I defined conditions with a high threshold for the non-dominant hand to perform an initial upward movement significantly greater than that observed in other phases of the operation. Upon detecting this movement, a contraction in the hand's height was expected, followed by a small movement. However, specifying the threshold for detection became quite challenging without three-dimensional data. More refined alternatives to this approach could involve, for example, using time series analysis techniques or even network science techniques to model this movement. A limitation of these techniques is that we don't know if the user actually picked up the piece or, for example, if they dropped it and returned to the initial position, but this could be easily overcome with a classifier.

PEN/PROBE

In this phase, different detection techniques were tested and utilized, as a false positive in the probe test incurs a high cost. During preprocessing, the images from the provided training database were converted to grayscale and resized to 224x224. A Canny filter was applied for edge detection, and the images were then converted back to RGB format for training with the VGG-19 architecture using a transfer learning methodology.

VGG-19

Various parameter tuning tests were conducted, as the Canny filter is very sensitive to configurations, especially aperture size. Additionally, for probe/pen detection purposes, the images were cropped around the hand area. The base layers of VGG-19 were frozen, and only the last layers added for our problem were trained for 30 epochs. Data generation techniques were employed. The classifier was trained to evaluate the pick-up movement, the probe, and the pen. However, it was only trained with the pick-up phase to reduce the rate of false positives when encountering a new pattern. The best model showed an AUC of 0.98 on the validation set. However, for example, in the case of drawing, the model detected when the operator touched the pen but not necessarily the line, as it is not visible in most of the holes closest to the body.

Frame Differencing – Hough Lines

The goal here was to detect the line and its length, as well as the depth of the probe, which could be crucial, based on the movement of the pen/hand relative to the previous frame.

Figure 1. Invisible line

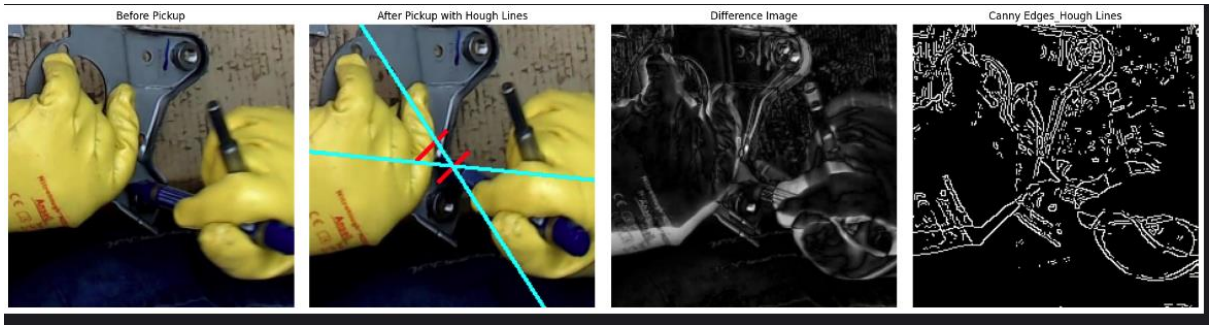


Figure 2. Distant line.

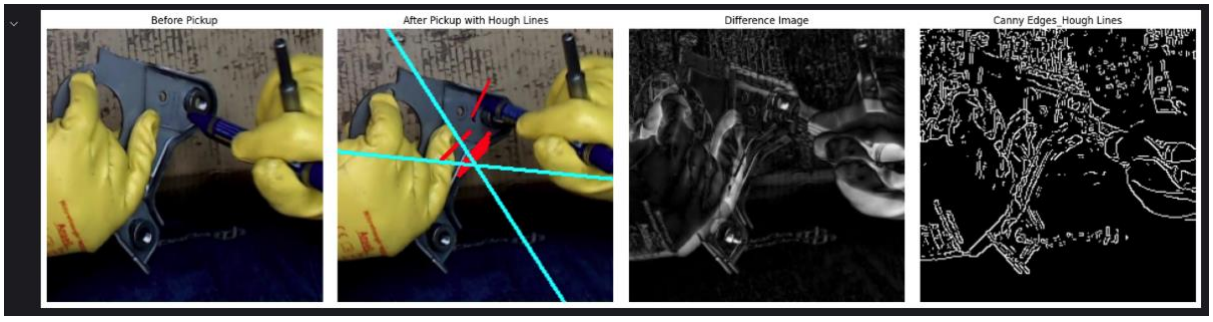


Figure 3. Probe

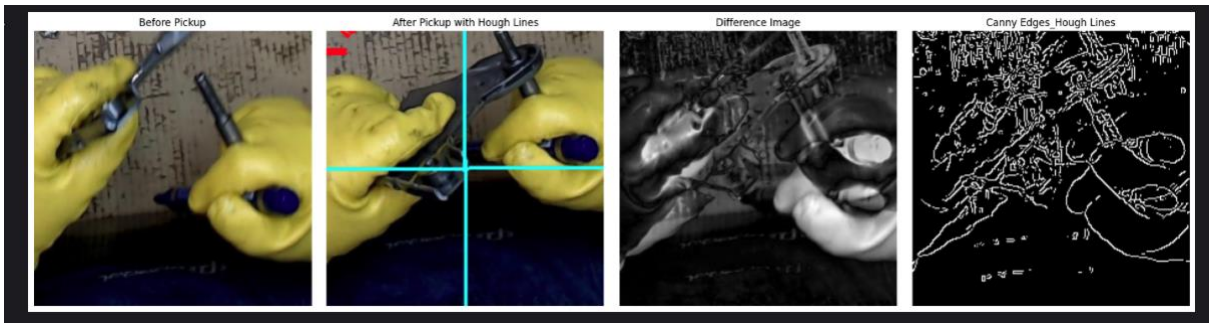
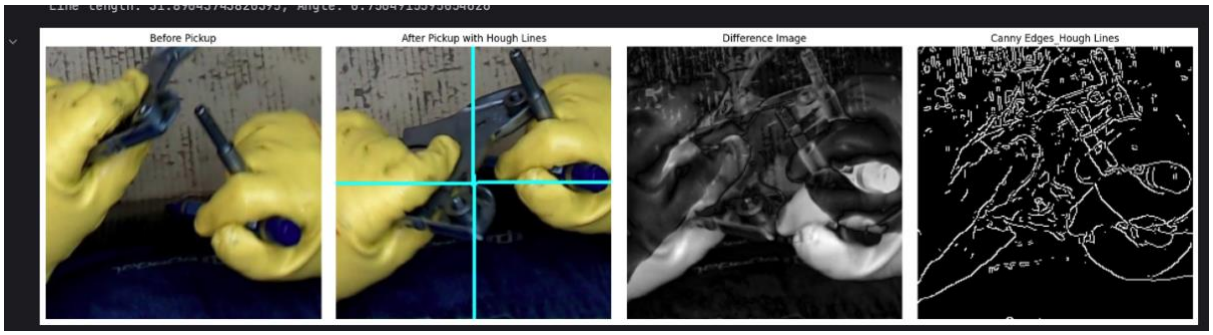


Figure 4. Miss



Pipeline Construction and Analysis

A pipeline was constructed with the objective of detecting whether one of these movements was initiated using VGG-19, followed by line detection based on size and orientation. In the case of the pen, this approach did not work as expected, requiring better tuning of thresholds to reduce the number of false positives. For the probe, I expected the system to await for 2 similar movements within the defined parameters (distance and orientation) in the last 3 previous trials, identifying the moment of hit and removal. In this way, I expected to increase recall while reducing the likelihood of incorrectly identifying the same movement. However, this was not well implemented. A possible implementation could involve detecting whether a probe was successfully performed, as well as the size of the perforation. Furthermore, this approach, combined with unsupervised techniques, could prove quite useful for training Machine Learning models to distinguish between a successful probe (Figure 3) and a failure (Figure 4). To determine whether the draw was in the same location, I compared the distance of the hand annotations relative to the previous hit. However, this approach is somewhat

rudimentary given the YOLO predictions' spacial accuracy, and machine learning algorithms based on the mentioned preprocessing, such as PCA, may prove to be quite useful.

End Phase

This movement was detected when the operator's non-dominant hand disappeared from the image (near the body, as YOLO does not detect in some operations at the top corner when retrieving pieces). A delay of one frame was considered, as this is approximately the time it takes for the hand to completely disappear from the screen.

Insights

As mentioned earlier, there is significant room for improvement, and the exploration of other techniques could lead to simpler implementations with better performance. Among these are techniques such as object and motion tracking.

Additional ideas for this system include, as mentioned, measuring the size of the scratch, the depth of the probe, and detecting whether the operator is wearing safety gloves, the angle of hit, and how it could predict a miss to enhance the operator's performance. Additional analyses could be performed with this system, such as monitoring performance and fatigue throughout the day. By analyzing time series, it could be determined if the number of misses, incomplete operations, operation time, and break period (the interval between placing the piece and starting a new operation) worsen throughout the day or during specific periods. Ideally, with 3D cameras or gloves with sensors, a better analysis could be conducted to improve human performance.