Computer Vision Engineer Assessment Task: Report

Objective: To detect and classify objects from a live camera feed using both classical and deep learning-based computer vision techniques, and to analyze the impact of camera properties on detection performance.The project implemented two approaches for object detection in live video: a classical computer vision pipeline for circular shape detection and a deep learning method using YOLOv11.

Methodologies:

1. Image Acquisition & Initial Camera Properties:

- Real-time video was captured using the default camera (`cv2.VideoCapture(0)`).
- Initial camera properties (resolution, frame rate) were obtained.
- Brightness was programmatically set to **95**, and exposure to **1**, chosen experimentally for good image visibility under test lighting.
- Intrinsic parameters were initially estimated (focal length = `frame_width` , principal point = center, zero distortion). These were then simulatedly modified: focal length to **600.0**, principal point to **(330, 250)**, and distortion to zero, to observe the potential impact of camera optics on detection.
- While camera calibration (using checkerboards and online tools) was explored for accurate intrinsic parameters, time and resource constraints led to the use of estimated and simulated values, limiting the accuracy of size and position measurements. With proper calibration, the accuracy of these measurements and the overall shape analysis could be significantly improved.

2. Object Detection Using Classical Computer Vision:

- Detect circular, round, and oval shaped objects in live video through color filtering in HSV space, morphological operations, Canny edge detection, and contour analysis.
- **Explored Techniques:**

- **HSV Color Filtering:** Preferred over RGB as it separates color (Hue) from luminance (Value) and saturation. Initial broad range [0,0,0]-[180,255,255] caused significant noise; narrower range [0,50,50]-[180,255,255] effectively filtered dark/unsaturated regions. Success depends heavily on HSV range calibration based on lighting and object color.
- **Adaptive Thresholding:** Considered for handling varying illumination using local thresholds (block sizes 3-11, C values -2 to 5). Rejected as less direct for colored circular detection compared to combined color and shape analysis.
- **Sobel Filtering:** Explored to highlight intensity gradients (kernel sizes 3,5,7) but required additional processing (thresholding, non-maximum suppression) for clean contours. Canny was preferred since it incorporates Sobel with these additional steps.
- **Bilateral Filtering:** Evaluated as preprocessing to reduce noise while preserving edges. Could potentially improve robustness of subsequent color filtering or edge detection, but not implemented in final solution.
- **Kalman Filtering:** Considered for tracking moving objects by predicting future states and smoothing detections. Identified as future improvement requiring state vector, matrices, and noise covariance definition.
- **Hough Circle Transform:** Explored specifically for perfect circles with parameters for detection method, resolution ratio, minimum distance, and radius ranges. Too rigid compared to contour analysis with tunable circularity threshold for near-circular shapes.
- **Contour Analysis:** Central to final approach with critical parameters:
  - Area filtering (min_area = 500, max_area = 50000) based on expected object sizes
  - Circularity threshold (0.7) balanced actual circular objects with near-circular shapes
  - Polygonal approximation with epsilon of 0.02 × perimeter affected circularity calculations

- **Successful Pipeline:**

    - HSV color filtering ([0,50,50]-[180,255,255])
    - Morphological operations (erosion, dilation, closing) to reduce noise and fill gaps
    - Canny edge detection to highlight edges
    - Contour finding
    - Area-based filtering (500-50000 pixels)
    - Circularity calculation ($4\pi \times \text{Area}/\text{Perimeter}^2$)
    - Circularity-based filtering (threshold ≥ 0.7)
    - Drawing bounding boxes and enclosing circles with circularity scores
- The pipeline processed each frame, and the results (original frame with detections and intermediate processing steps) were displayed.

3. Object Detection Using Deep Learning:

- A deep learning-based object detection system was implemented in `deep_learning_detection.py` using the pre-trained **YOLOv11s** model from the `ultralytics` library.
- The model, trained on the COCO dataset, was used for inference on the live camera feed.
- **Techniques Used:**
    - **Pre-trained Object Detection Model:** YOLOv11s, a state-of-the-art real-time object detection model known for its speed and accuracy, was loaded directly using `YOLO("yolo11s.pt")`.
    - **Class Filtering:** Detection was focused on a predefined list of classes (easily available to test on live input): `['person', 'book', 'scissors', 'teddy bear', 'hair drier', 'remote' , 'cell phone' , 'bottle' ,'cup', 'spoon']`.
    - **Confidence Thresholding:** Detections with a confidence score below `0.7` were filtered out.
    - **Bounding Boxes and Labels:** Detected objects within the specified classes and above the confidence threshold were annotated with bounding boxes and class labels along with their confidence scores.

    - **Screenshot Capture:** When specific combinations of detected objects were present in the frame for the first time or after a 5-second interval, a screenshot was saved to the `screenshots` directory, with the filename indicating the detected object classes.

4. Performance Analysis & Camera Impact:

- **Classical CV:** The exploration highlighted the importance of parameter tuning in classical CV techniques. Color filtering is sensitive to the chosen HSV range, and the circularity threshold directly impacts the types of shapes detected. Canny edge detection proved more robust than basic Sobel filtering for this task. While other techniques like adaptive thresholding and Hough Transform were considered, the combination of color filtering and contour analysis with circularity proved most effective for the specific goal. The lack of camera calibration limits the accuracy of size and position estimations. The classical approach demonstrated the ability to detect circular objects based on color, shape, and size. However, it is sensitive to:
    - **Lighting Conditions:** Changes in brightness and shadows (which can create circular-like edges) significantly impacted the detection accuracy and could lead to false positives.
    - **Camera Properties:** While the code programmatically modified brightness and exposure, the direct impact on the classical detection was observed visually. Lower brightness could make color-based filtering less effective, and overexposure could wash out features needed for edge detection. The simulated intrinsic parameter changes were not directly incorporated into the classical CV pipeline and thus their impact wasn't explicitly analyzed within this approach.
- **Deep Learning:** The YOLOv11 model showed robust object detection across varying lighting conditions encountered during the live feed.
    - **Lighting Conditions:** The pre-trained model exhibited a degree of invariance to changes in brightness and exposure.
    - **Camera Properties:** Similar to the classical approach, the direct impact of simulated intrinsic parameter changes was not explicitly evaluated within the deep learning pipeline in this implementation. However, it is known that

significant deviations in intrinsic parameters (if they were actually changed in the hardware) could affect the accuracy of bounding box predictions.

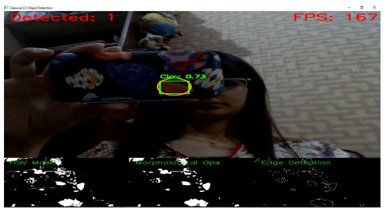
- **Comparison:** The deep learning-based approach (YOLOv11) significantly outperformed the classical CV method in terms of the variety of detectable objects and robustness to environmental changes. The classical method was limited to detecting circular objects and was more susceptible to noise and lighting variations.

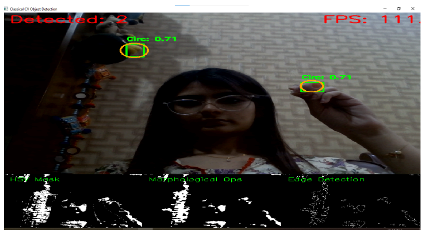
5. Code Structure

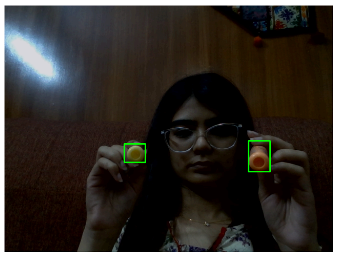
- code/

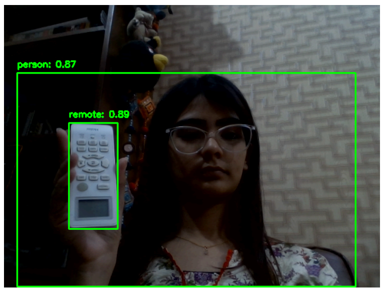
```
│
├── model/
│   ├── yolo11s.pt
├── screenshots/
├── main.py
├── camera_utils.py (Part 01)
├── classical_cv.py  (Part 02)
├── deep_learning_detection.py (Part 03)
└── requirements.txt
```

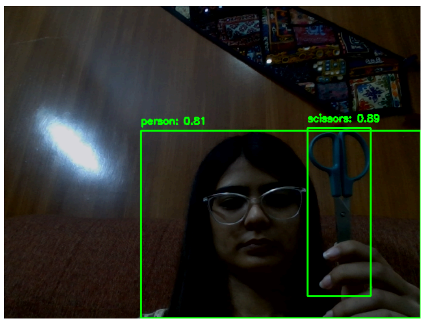
- **Screenshots**

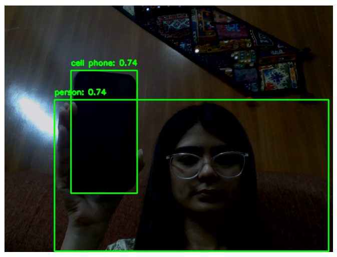












Based on the computer vision assessment, YOLOv11 deep learning significantly outperformed classical CV approaches in object detection due to its versatility across object classes, robustness to environmental variations, and superior accuracy in complex scenarios. The classical circular object detection method, while simpler and potentially useful in constrained environments with circular targets, suffered from sensitivity to lighting conditions and limited object type recognition. Without ground truth data, evaluation relied on qualitative visual inspection rather than quantitative metrics. Future improvements could include exploring advanced classical feature detectors, fine-tuning deep learning models on custom datasets, implementing tracking algorithms, developing proper evaluation metrics through annotated datasets, and creating a more systematic approach to analyze camera parameter impacts. A more user-friendly interface with comprehensive error handling would also enhance the system's overall functionality.