Automating People Counting with Video Analytics

Vedant Misra

The Challenge of Manual People Counting

- Manual counting is time-consuming and labor-intensive.
- It's prone to errors due to fatigue or distractions.
- Inconsistent counting methods can lead to inaccurate data.
- Real-time data collection is not feasible with manual methods.

Leveraging YOLO and Deep Learning

Experimentation

- YOLO (You Only Look Once) is a real-time object detection algorithm.
- Deep Learning models can identify and track people in video footage.
- Experimentation allows us to fine-tune the model for specific needs.

My Implementation with Live Video Input

We implemented a system that:

- Takes live video input from a CCTV camera.
- Detects and tracks people using a YOLO-based deep learning model.
- Counts the number of people entering and exiting predefined zones.
- Periodically logs the count data to a text file.

Code Structure and Logs

Modules:

- Main Script: Handles program execution, initialization, and main loop.
- Frame Processing Function (Optional): Processes video frames, detects objects, and updates counts.
- Logging Function: Logs entry/exit counts at specific intervals.

Logs:

- Record entry/exit counts at specific time intervals.
- Example structure: "HH:MM:SS: Enter: {count}, Exit: {count}".
- Processing interval determines logging frequency.

```python
# Variables for counting, logging, and time tracking
total_enter_count = 0
total_exit_count = 0
frame_count = 0
last_processed_frame = 0  # Track frame for periodic processing
processing_interval = 5  # Process frames and save results every 5 seconds
log_file_name = "people_count_log.txt"  # Adjust filename if desired

def update_log_file(enter_count, exit_count, timestamp):
    """
    Appends people count data to the log file.
    """
    with open(log_file_name, "a") as log_file:  # Open in append mode
        log_file.write(f"{timestamp}: Enter: {enter_count}, Exit: {exit_count}\n")

def get_current_timestamp():
    """
    Returns the current timestamp in a human-readable format.
    """
    return time.strftime("%H:%M:%S")

# Object tracker (assuming tracker.py defines the Tracker class)
tracker = Tracker()

# Live CCTV input (replace with appropriate code for your camera)
cap = cv2.VideoCapture(0)  # Change 0 to your camera ID if using a webcam

while True:
    ret, frame = cap.read()
    if not ret:
        break

    # Resize frame (optional)
    frame = cv2.resize(frame, (1028, 500))

    # Background subtraction for motion detection
    mask = bg_subtractor.apply(frame)
    _, mask = cv2.threshold(mask, 245, 255, cv2.THRESH_BINARY)
    contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    # Track and classify objects
    object_list = []
    for cnt in contours:
```

```python
for cnt in contours:
    area = cv2.contourArea(cnt)
    if area > 1500:
        object_list.append(cv2.boundingRect(cnt))
bbox_idx = tracker.update(object_list)

# Process objects and update counts
enter_count = 0
exit_count = 0
for bbox in bbox_idx:
    x1, y1, x2, y2, id = bbox
    cx = int((x1 + x2) / 2)
    cy = int((y1 + y2) / 2)

    # Check for entry and exit based on region intersections
    result_enter = cv2.pointPolygonTest(np.array(area1, np.int32), (cx, cy), False)
    result_exit = cv2.pointPolygonTest(np.array(area2, np.int32), (cx, cy), False)

    if result_enter >= 0:
        enter_count += 1

    if result_exit >= 0:
        exit_count += 1

    # Draw bounding box, ID, and update tracker
    cv2.rectangle(frame, (x1, y1), (x2 + x1, y2 + y1), (0, 255, 0) if id in tracker.tracked_objects else (0, 0, 255), 3)
    cvzone.putTextRect(frame, f"{id}", (cx, cy), 2, 2)
    tracker.update_status(id, (cx, cy))

# Draw entry and exit regions
cv2.polylines(frame,[np.array(area1,np.int32)],True,(0,0,255),2)
cv2.polylines(frame,[np.array(area2,np.int32)],True,(0,0,255),2)

Enter=len(counter1)
Exit=len(counter2)
cvzone.putTextRect(frame,f'ENTER:-{Enter}',(50,60),2,2)
cvzone.putTextRect(frame,f'EXIT:-{Exit}',(50,130),2,2)


print(er)
cv2.imshow('RGB', frame)
```

```python
    print(er)
    cv2.imshow('RGB', frame)
    if cv2.waitKey(1) & 0xFF == 27:  # Press 'Esc' to exit
        break

# Release the video capture and close windows
video_capture.release()
cv2.destroyAllWindows()
```

Snapshots

File Edit View Run Tools Help

trained_YOLO_line_v4.py | test.py | tracker.py

```
                if counter1.count(id)==0:
                    counter1.append(id)

        result2=cv2.pointPolygonTest(np.array(area2,np.int32),((cx,cy)),False)
        if result2>=0:

            ex[id]=(cx,cy)
        if id in ex:
            result3=cv2.pointPolygonTes
            if result3>=0:
                cv2.rectangle(frame, (x1
                cvzone.putTextRect(frame
                cv2.circle(frame,(cx,cy)
                if counter2.count(id)==0
                    counter2.append(id)




    cv2.polylines(frame,[np.array(area
    cv2.polylines(frame,[np.array(area

    Enter=len(counter1)
    Exit=len(counter2)
    cvzone.putTextRect(frame,f'ENTER:-
    cvzone.putTextRect(frame,f'EXIT:-{

    cv2.imshow('RGB', frame)
#    time.sleep(0.1)
    if cv2.waitKey(0) & 0xFF == 27:  #
        break

# Release the video capture and close windows
video_capture.release()
cv2.destroyAllWindows()
```

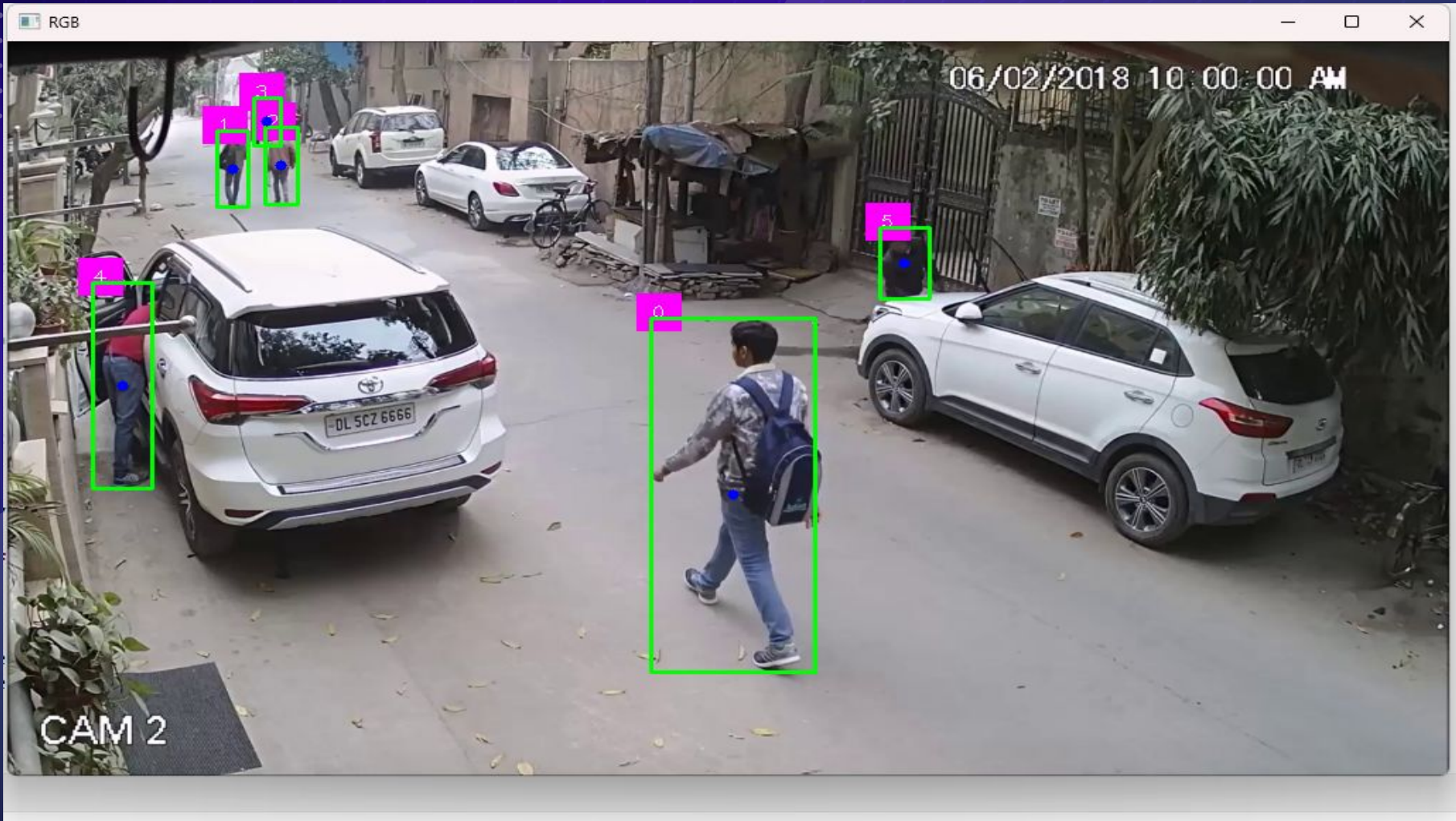
RGB

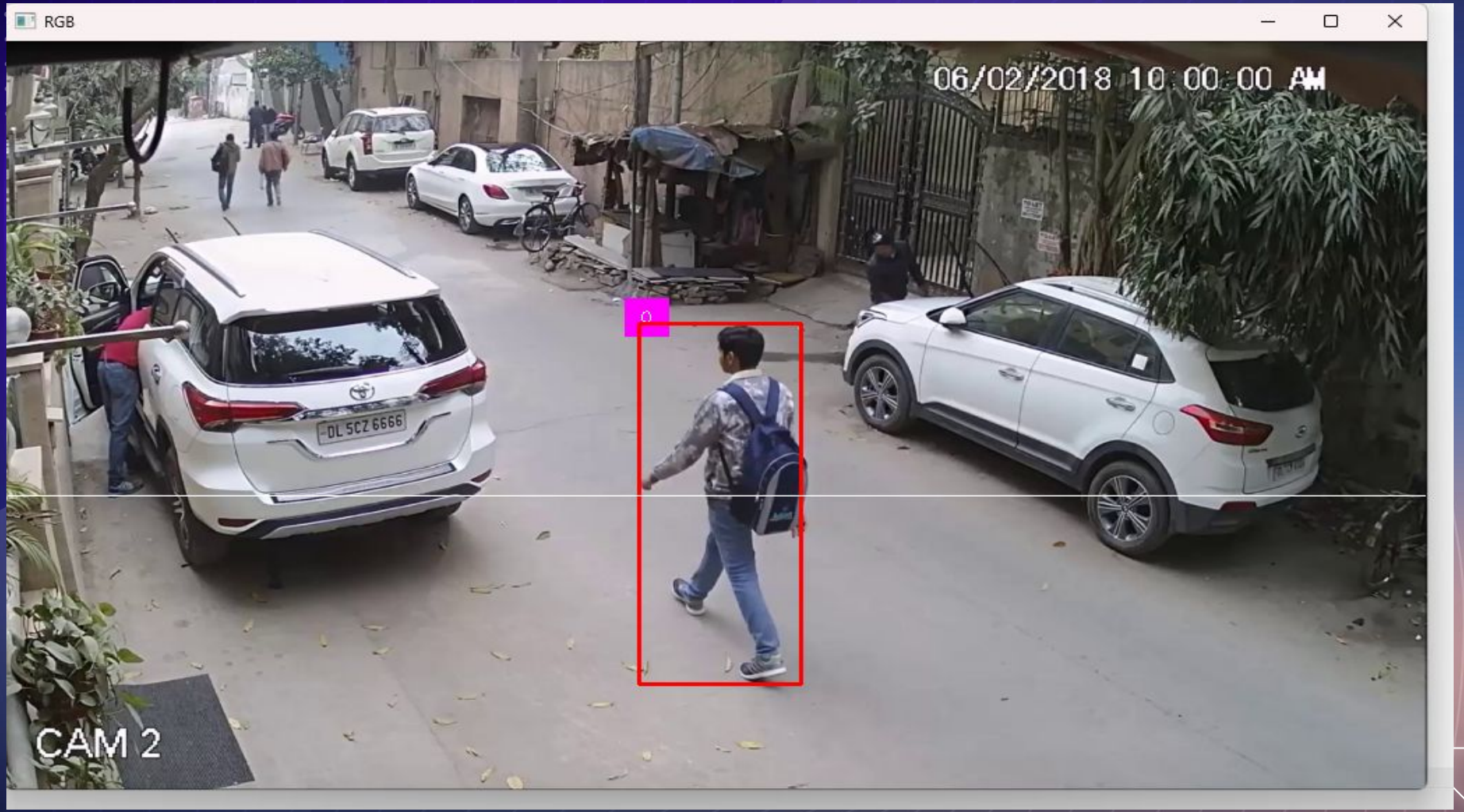
Assistant

Shell

```
[625, 233]
[626, 233]
[627, 233]
[628, 233]
[629, 233]
[629, 233]
```


RGB
ENTER:-3
EXIT:-1
Camera 01mar


RGB
06/02/2018 10:00:00 AM
CAM 2
DL 5CZ 6666


RGB
06/02/2018 10:00:00 AM
0
DL 5CZ 6666
CAM 2


RGB
People:-4
06/02/2018 10:00:13 AM
DL 5CZ 6666
CAM 2

Benefits of Automated People Counting

- Increased Accuracy: Deep learning models can provide more accurate counts compared to manual methods.
  → Observable differences after training a YOLO model
- Real-time Insights: Automated systems allow for real-time data collection and analysis.
- Reduced Labor Costs: Automation eliminates the need for dedicated personnel for manual counting.
- Improved Data Analysis: Logs provide valuable data for analyzing foot traffic patterns and trends.

Conclusion

- Automates people counting, saving time and resources.
- Provides real-time data for better decision making.
- Can be integrated with other systems for further analysis.
- Accuracy can be improved through continued experimentation.