Library Seat Occupancy Detection

Abhinav Sethi
Department of Data Science & Engineering
IISER Bhopal
Bhopal, India
abhinav19@iiserb.ac.in

Sumit Dangi
Department of Data Science & Engineering
IISER Bhopal
Bhopal, India
sumit19@iiserb.ac.in

Sushil Vemuri
Department of Data Science & Engineering
IISER Bhopal
Bhopal, India
sushil19@iiserb.ac.in

***Abstract* - This paper explains a method of seat occupancy detection in the library, using Computer Vision techniques on the images of the library to map which seats are available while leveraging existing surveillance camera infrastructure.**

1. INTRODUCTION

Most IISERB community members visit the library for a peaceful session to learn. People walk long distances to reach the library. This consumes a lot of time in their already busy schedule. As the final exam season approaches, many students prefer to study in the library and seats are usually in short supply. With students dispersed throughout the maze of levels, finding a vacant seat can become a difficult task. Students will also frequently leave their belongings in a seat to indicate that it is occupied. This significantly reduces the usability of available seats. Knowledge of the number and locations of vacant seats can be very helpful for the students.

1.1 PROBLEM DEFINITION

The challenge was to build an AI system that uses the library's CCTV footage to output the seat status of every seat of the library in almost real-time. Then to update the seat status on a website for all IISERB members to access.

1.2. OBJECTIVE

The objective was to be able to accurately pinpoint the vacancy of the seats in the library, and make this available to all the community members. This would save a lot of time for all the students during exam weeks and allow them to plan their visits to the library. This would also help the library to take note of the seats which are “ON HOLD” for a long period of time and clarify the status of these seats.

2. CONTRIBUTIONS

- Chia-Hung Lin made a seat occupancy detection project [1] in his college library. They had detected objects in real-time, but since we took images from the footage, we cropped out each ROI and detected images in these ROIs, this approach had not been tested yet and has proved to be more accurate than previously used techniques, especially when using pre-trained models for object detection.

- Taking inspiration from the vehicle seat vacancy identification [2] our AI processed the images by using deep neural networks and object detection technology to detect objects in the defined ROIs.

- Adapting from the techniques applied in image-based seat belt detection in cars [3] we, based on the defined ROIs, tried adjusting the confidence values for each of the ROIs to ensure that the objects are

detected with the most accuracy, this approach had not been applied yet and we were curious as to what this could entail for the future of computer vision technologies in surveillance. But due to the presence of less data we weren't able to conclude whether ROI-specific confidence values could be a viable approach that can be applied in practice.

- Computer Vision had not been extensively explored in the domain of seat occupancy detection and thus, we hope that our attempt encourages future research in this field with novel approaches.

3. BACKGROUND

- Traditionally, seat occupancy detection in libraries has been done using pressure sensors and IoT techniques. These sensors are expensive and difficult to replace.
- Computer Vision techniques have not been successfully applied yet in the context of libraries, as most cameras are not placed in favorable locations.

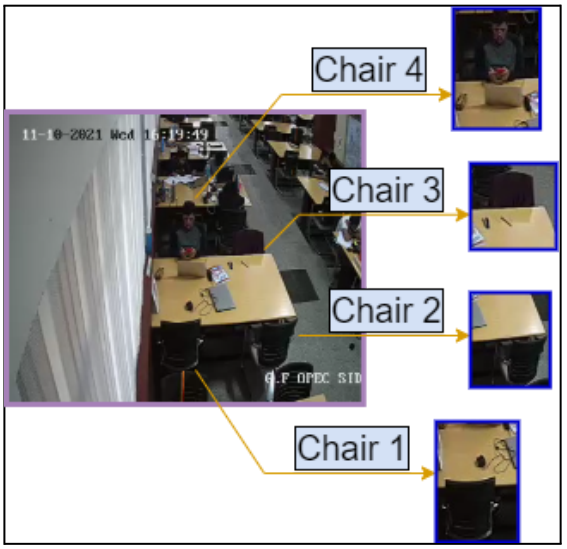
4. MATERIAL & METHODS

There are 4 major steps:

- **Data collection**:
  The CCTV data from the IISERB library was obtained at 4 different instances of time. Based on the availability and quality of data we decided to go ahead with the ground floor of the library. Data from 5 cameras on the ground floor.

- **ROI (Region of Interest) definition & cropping:**
  ROI is an area of an image defined for further analysis.
  For 3 of the cameras we chose to focus on one table each, and from the rest 2, we focused on two tables each, on the basis of visibility, lack of collinearity, ease of detection, etc.

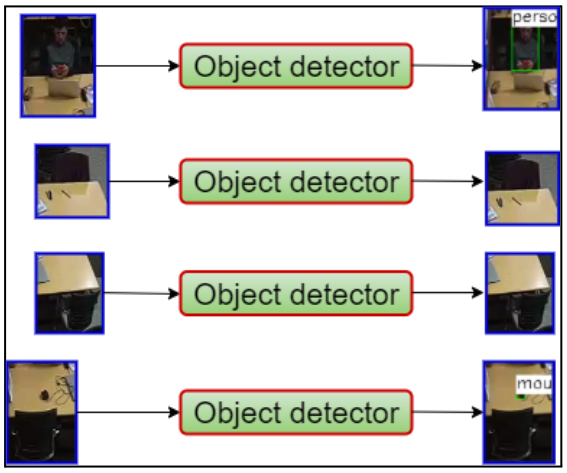
Then, using Matplotlib, we defined 20 ROIS, in total, across all 5 cameras. We chose Matplotlib to define the ROIs due to its ease of use and various functionalities which made it convenient to find the relative coordinates of the ROIS in the image.
For each of these 20 ROIS, we defined one smaller ROI each. Each smaller ROI solely focused on the section of the table belonging to that particular ROI or seat.



The figure above shows the defined ROIS being cropped (Camera 2).

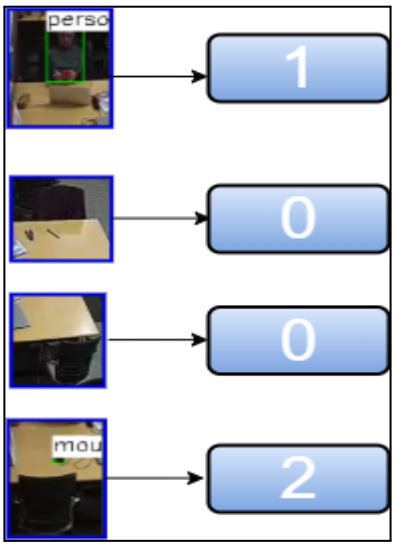
- **Data Preprocessing:**
  We converted the bitmap images to jpeg because the bitmap images are stored differently in OpenCV, and we wanted the project to be extended onto general libraries..
  OpenCV's *imread* function was used to load the image and convert the images into NumPy arrays.
  The ROIS were defined on images of dimension (352, 288). Thus, all images passed to the program were resized to ensure consistency.
  We cropped the ROIS from the images following the conventional NumPy array slicing.

- **Object Detection:**
  We used a pre-trained [4] object detection model called the YOLOv4-p6. YOLOv4 from [5] is a real-time object detection model published in April 2020 that achieved state-of-the-art performance on the COCO dataset. It works by breaking the object detection task into two pieces, regression to identify object positioning via bounding boxes and classification to determine the object's class. YOLO stands for You Only Look Once. It is a one-stage object detection model based on Convolutional Neural Networks(CNN). The COCO dataset consists of 80 different types of objects including books, people, and chairs.
  The model was implemented using OpenCVs deep neural network(DNN) module.
  Each ROI was passed into the object detection function with a confidence threshold and NMS frequency.
  The function then returned a *pandas* dataframe of the class id, Confidence level, and location of every detected object in the given image.



The figure above shows the objects being detected in the defined ROIS.

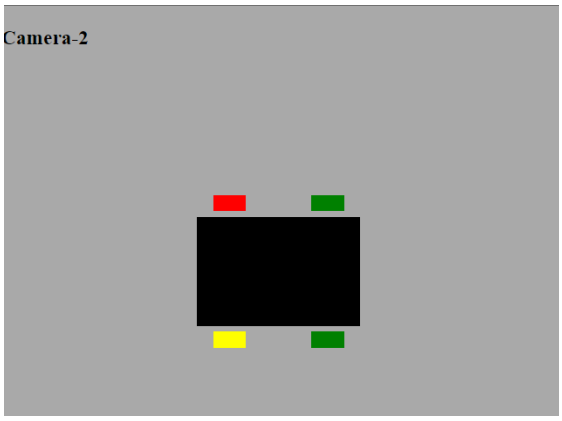
- **Seat status indication:**
  Each resized ROI was passed on to the object detection phase.
  If the object detection phase returned a dataframe with an entry 1 (“person”) as one of the multiple objects detected, then the seat was marked as “OCCUPIED” (corresponds to the status value of 1).
  If the object detection returned an empty dataframe, then we concluded that the seat was not “OCCUPIED”, for certain, and thus it could be either “ON HOLD” (corresponds to the status value of 2) or “EMPTY” (corresponds to the status value of 0). Now, the smaller ROIS were passed to the object detection phase. If, once again, the dataframe returned was empty, then it was concluded that the seat was “EMPTY”. If any object was detected within the smaller ROI, then the seat was marked as “ON HOLD”. The status was stored in a dataframe and this was stored in a CSV file (*seat_status.csv*).



The figure above shows ROIS being mapped to their respective seat status.
(0- Empty; 1- Occupied; 2- On hold)

- **Layout**
  In the file named *final.html*, using HTML and CSS, a layout for the ground floor was designed. This layout was divided into six equal sections. Five sections showed five camera views with tables and chairs. One section showed ‘keys’ to explain what each color of the

seat meant. The color-coding of the seats was done using JavaScript. The *seat_status.csv* was read using the fetch() function, and the data was converted into arrays. A function *cc()* was defined which takes a chair and its status as parameters and changes the seat's color accordingly. This function was called on all the seats. Running this script gave us the final layout with color-coded seats.

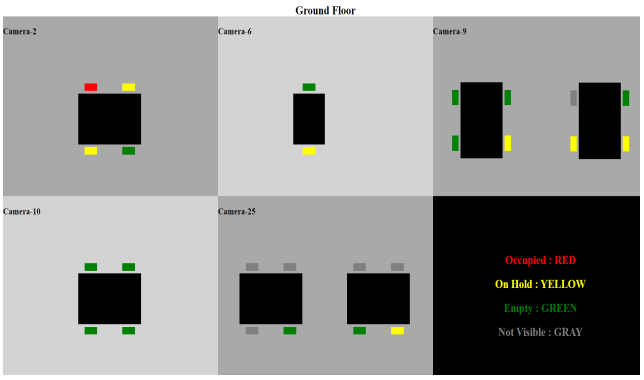


The figure above shows the section corresponding to Camera-2.

6. RESULTS

The model accurately predicted the seat status of 76.25% of the selected seats in the library, on average. This, roughly, translates to 15 seats out of 20 seats per instance of data.

Output layout:



The figure above shows the final output of our project, which is the layout of the tables.

7. DISCUSSIONS

When experimenting with ROIs, we had to select the ROIS in a manner that minimized the collinearity between two ROIS. As it was difficult to detect humans in the ROIs and thus, the model misinterpreted a seat to be occupied when someone was sitting in a collinear seat. The YOLOv4 model is one of the most accurate Computer Vision models, even so, its accuracy is around 70% and thus might result in incorrect detections of objects in very crowded images.

7.1 LIMITATIONS

We initially targeted more seats, but eventually had to drop out a few of the ROIS due to 2 major reasons:

- The model could not detect any object in the ROI due to either the ROI being too small or due to external factors such as the lighting conditions which caused shadows or reflections, which in turn affected the model's performance.

- The ROIS were too particularly picked for the training data and thus, the model was not able to, with sufficient confidence, detect any other objects within the ROI.

7.2 SCOPE

- In the current state of the project, it would not be practical to deploy it, as it does not cover a major portion of the ground floor of the library itself.

- Given more time and more data, it would definitely be possible to make a model that achieves high accuracy in relatively less time.

7.3 FUTURE RECOMMENDATIONS

- **ROI specific confidence values:**
  The confidence threshold could act as a hyperparameter and thus for each ROI, we could map a specific confidence level

which would help us fine-tune the object detection.

- **Template Matching:**
  OpenCV has multiple image processing techniques, and a function that could prove to be very powerful to the library seat occupancy detection use-case is *matchTemplate*. It is the template matching technique inbuilt in OpenCV which could be used to classify a seat as being empty or not.
  Template matching is done when you compare a reference image to another image and check how close the two are. The idea is to use the smaller ROIS, defined already, and store 20 reference images of empty tables, and check if the current image being processed has the same smaller empty ROI inside it, with say 80% accuracy. If this is true then that seat can instantly be marked as empty, and thus could help reduce time complexity drastically. It is also preferred as the computational load of template matching is minimal.

- Libraries are generally very big and it may not be possible to get a good view of every table in it. So, it is recommended to use IOT sensors for those places where CCTV cameras aren't viable.

- The COCO dataset enables object detection of 80 different objects but this wide domain of objects is generally not required in the library. So, training our own object detection model with a smaller domain but more practically found objects may have been beneficial.

- A model which is accurate enough in human and chair detection would greatly help in generalizing this program because the need for manually defining ROIs would be eliminated.

REFERENCES

[1] (2018) Chia-Hung Lin, Morgan Hobson, and Ruiqing Yin, "Seat Status Detection in Library ". Available: https://github.com/RexxarCHL/library-seat-detection

[2] Thongchai Yooyativong, Janewit Wittayaprapakorn, "Vehicle Seat Vacancy Identification using Image Processing Technique", 2017 International Conference on Digital Arts, Media and Technology (ICDAMT). Available: https://ieeexplore.ieee.org/document/7904953

[3] Huiwen Guo, Hui Lin, Shaohua Zhang, Shutao Li, "Image-based seat belt detection". Proceedings of 2011 IEEE International Conference on Vehicular Electronics and Safety. Available: https://www.researchgate.net/publication/252029744_Image-based_seat_belt_detection

[4] Bochkovskiy, Alexey & Wang, Chien-Yao & Liao, Hong-yuan. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. Available: https://arxiv.org/pdf/2004.10934.pdf

[5] AlexeyAB. (2020). Yolo v4 pre-release on Github. [Online]. Available: https://github.com/AlexeyAB/darknet