**T.C.**

**BAHÇEŞEHİR UNIVERSITY**



**FACULTY OF ENGINEERING AND NATURAL SCIENCES**

**CAPSTONE FINAL REPORT**

**CROWD-TRACK: REAL-TIME MONITORING AND HEATMAP GENERATION SYSTEM FOR CONFERENCE HALLS**

**Artificial Intelligence Engineering:**

* **Ayah Kouider**
* **Hala Hussein Eltayeb Elmahi**
* **Sandra Hesham Elzeini**

**Electrical and Electronic Engineering**

* **Necati Berkan Şafak**
* **İbrahim Yılmaz**

**Advisors:**

**Prof. Ahmet Arif Ergin, Electrical and Electronic Engineering**

**Prof. Fatih Kahraman, Artificial Intelligence Engineering**

**ISTANBUL, May 2025**

**STUDENT DECLARATION**

By submitting this report, as partial fulfillment of the requirements of the Capstone course, the students promise on penalty of failure of the course that

* they have given credit to and declared (by citation), any work that is not their own (e.g. parts of the report that is copied/pasted from the Internet, design or construction performed by another person, etc.);
* they have not received unpermitted aid for the project design, construction, report or presentation;
* they have not falsely assigned credit for work to another student in the group, and not take credit for work done by another student in the group.

# TABLE OF CONTENTS

ABSTRACT iii

TABLE OF CONTENTS iv

LIST OF TABLES v

LIST OF FIGURES v

LIST OF ABBREVIATIONS vi

1. OVERVIEW 7

1.1. Identification of the need 7

1.2. Definition of the problem 7

1.3. Conceptual solutions 9

1.4. Physical architecture 11

2. WORK PLAN 14

2.1. Work Breakdown Structure (WBS) 14

2.2. Responsibility Matrix (RM) 15

2.3. Project Network (PN) 16

2.4. Gantt chart 17

2.5. Costs 20

2.6. Risk assessment 21

3. SUB-SYSTEMS 24

3.1. Electric subsystem 24

3.2. Ai subsystem 27

4. INTEGRATION AND EVALUATION 31

4.1. Integration 31

4.2. Evaluation 31

5. SUMMARY AND CONCLUSION 33

ACKNOWLEDGEMENTS 34

REFERENCES 35

APPENDIX A 36

APPENDIX B 37

# LIST OF TABLES

[Table 1. Comparison of the three conceptual solutions.](#_17dp8vu) 11

[Table 2. Responsibility Matrix for the team](#_2jxsxqh) 16

[Table 4. Costs](#_2xcytpi) 20

[Table 5. Risk matrix](#_3whwml4) 21

[Table 6. Risk assessment](#_2bn6wsx) 22

# LIST OF FIGURES

Figure 1. Avigilon camera 10

Figure 2. axis camera 11

Figure 3. physical architecture 12

Figure 4. software architecture 12

Figure 5. Work breakdown structure15

Figure 6. The project network. 17

Figure 7. Gantt chart 19

Figure 8.electric architecture 26

Figure 9. Heatmap generation output 39

Figure 10. First Approach 42

Figure 11. Final Approach 42

### Abstract

This project presents crowd-track, a real-time monitoring and heatmap generation system designed for conference halls and similar indoor environments. The system uses object detection to identify and track individuals through surveillance camera feeds. After evaluating multiple deep learning models, YOLOv5 was selected for its high accuracy, fast inference speed, and suitability for human detection in standard indoor scenes. Detected positions are used to generate dynamic heatmaps and analyze crowd movement patterns. The system is optimized for deployment on edge devices such as the NVIDIA Jetson Nano and integrates with tracking modules and a web-based visualization dashboard. This end-to-end pipeline supports effective occupancy management and safety monitoring.

# 1. OVERVIEW

The primary goal of our project is to design and implement a real-time room occupancy monitoring system that tracks the presence and movement of individuals and generates heatmaps for visual analysis. By integrating advanced artificial intelligence techniques with sensor-based detection, the system aims to enhance space utilization and support efficient resource and energy management in environments such as conference rooms, offices, and classrooms.

For the object detection part, we tested several models and finally chose YOLOv5 because it was fast, accurate, and easy to use for detecting people in regular camera footage. It works well with real-time processing on the NVIDIA Jetson Nano, which is important for our setup. The model helps us figure out where people are in the room so we can use that information to create heatmaps that show how crowded different areas get.

By combining expertise in electronics and AI, the team has developed a functional and scalable system designed to improve spatial awareness, occupancy monitoring, and safety.

## 1.1. Identification of the need

The purpose of this project is to create a crowd tracking system for monitoring crowd densities in the conference halls and classrooms of universities. There is little knowledge on how those spaces should be utilized which leads to issues such as inefficient room usage, safety hazards due to overcrowding, and offers data for future references.

The product that will service this problem is an advanced real-time monitoring system which will utilize optical sensors and cameras with fisheye lenses to track individuals entering and exiting the premises. To detect people in the camera footage, an object detection model trained on fisheye images is used. This information is then used to generate a heatmap to visualize crowd density, which will be processed by the NVIDIA Jetson Nano. A web-based application will be developed to view the results and provide data that can be used for future analysis.

## 1.2. Definition of the problem

Managing large spaces such as conference halls and classrooms is difficult without accurate data on occupancy and crowd movement. Existing methods that are currently used include manual monitoring, which is time-consuming and more prone to error, and lacks real-time updates.

This can result in overbooking or underutilizing spaces, which affects users and the experience. Additionally, in emergency situations, not knowing how crowded a space is can imply huge safety risks. To address these issues, our system uses real-time object detection with cameras to automatically track and count people, providing accurate and instant occupancy data.

### 1.2.1. Functional requirements

The system must use optical sensors and fisheye cameras to accurately detect and count individuals entering and exiting conference halls in real time, as well as identify the entrance and exit points in the room. The object detection model, trained on fisheye images, will process the camera feed to recognize people and track their movement. It should provide live occupancy data through an interactive web application interface. To give valuable insights, the system must also generate heatmaps to visualize crowd density and movement patterns in the monitored hall.

The sensors, cameras, and NVIDIA Jetson Nano for data preprocessing must all work together smoothly, enabling the system to function automatically without human assistance. The web application must allow authorized users to view historical trends and analytics, in addition to providing real-time occupancy levels and heatmaps.

### 1.2.2. Performance requirements

The system should be able to reliably detect and count individuals under a variety of lighting and environmental conditions, ensuring consistent performance regardless of the situation. It must generate heatmaps that accurately reflect the current crowd density, providing useful visual insights into how the space is being used. For tracking individuals entering and exiting, the system should process the data in real time with minimal delay, so that occupancy information is always up to date.

The hardware and processing units used in the system should be able to handle the workload efficiently to maintain smooth and continuous operation without causing delays or interruptions. Additionally, the web application should offer a smooth user experience with high availability, minimizing any interruptions to access or data updates. This will allow authorized users to rely on the system for both real-time monitoring and reviewing historical trends.

### 1.2.3. Constraints

To ensure the system can be deployed on a larger scale at colleges and universities without exceeding budget limits, it must be designed to be reasonably priced and use physical components such as the NVIDIA Jetson Nano, optical sensors, and cameras. To reduce environmental impact and maintain reliability in various indoor conditions—such as changing lighting or positioning—it should also be energy efficient.

Additionally, the system must have a user-friendly and non-intrusive design while strictly following GDPR and KVKK regulations to prioritize data security and privacy. The system should be capable of processing data from multiple sensors and cameras quickly and effectively, while being flexible enough to accommodate rooms of different sizes, layouts, and uses.

All hardware and software components should comply with IEEE standards for quality and reliability. Agile development methods should guide the project to allow frequent updates and maintain flexibility throughout the process.

## 1.3. Conceptual solutions

The solutions we developed for creating a crowd tracking system that counts the number of people in a room can be grouped into three main approaches.

The first approach uses a standard setup where the electrical engineering team customizes optical sensors to detect motion and count people entering and exiting. They also install a wide-angle camera at the entrances and a fisheye camera on the ceiling to monitor the room. The sensors and cameras are connected to a microcontroller like an Arduino, which sends the data to the NVIDIA Jetson Nano for processing. On the AI side, the team uses pre-trained models like YOLOv5 for object detection. Using OpenCV, they write simple code to count people and track their movements, and generate basic heatmaps from the fisheye camera data.

The second approach involves the electrical engineering team creating better-customized sensors and adding more cameras to cover the entire room. The data from these sensors and cameras is combined on a small computer like a Raspberry Pi before being sent to the Jetson Nano. The AI team initially tested several advanced models—including YOLOv5, YOLOv8, EfficientDet, and rotated object detection models like R3Det using the MMRotate framework. Although R3Det showed good results on fisheye images with a high mean Average Precision (mAP) of 0.909, the team faced difficulties managing the dataset and integrating it into the system. After overcoming initial challenges with dataset handling in YOLOv5, they chose to proceed with YOLOv5 for its better compatibility and easier implementation. The AI team also uses advanced tracking methods such as the Kalman filter to follow people and generates detailed heatmaps to analyze crowd density.

The third approach uses wireless sensors along with rotating and zoom-capable cameras (PTZ cameras) to focus on busy areas. The data is sent to a central hub. The AI team would process the data using cloud servers such as AWS alongside the Jetson Nano to prevent overheating. They would also add predictive features to estimate peak crowd times.

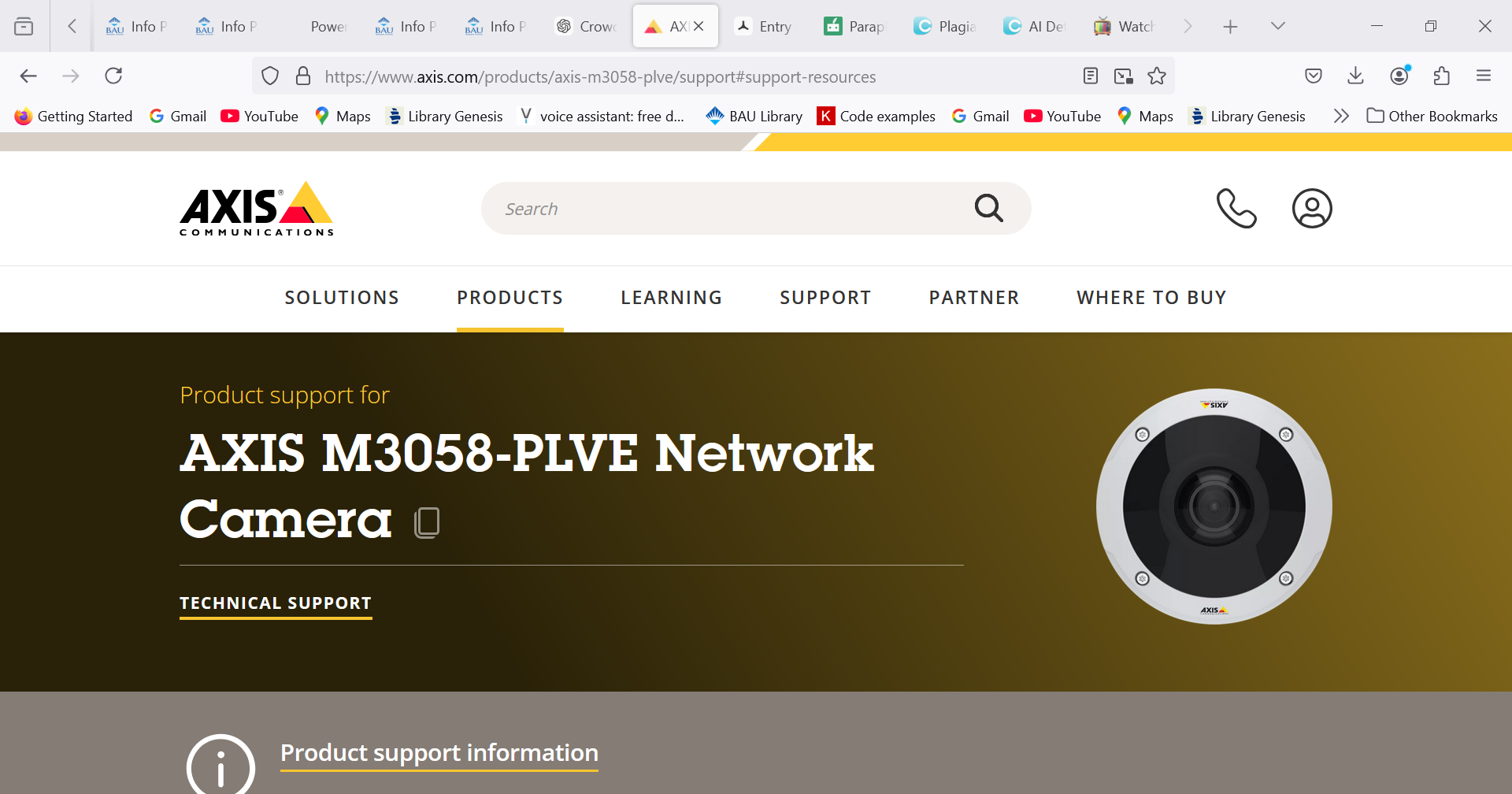
### 1.3.1. Literature Review

A growing number of industries are using real-time occupancy monitoring systems to effectively manage spaces and guarantee safety. Although the market's current solutions offer cutting-edge technologies, they frequently have drawbacks in terms of price, scalability, and flexibility. An outline of current products, pertinent technologies, and how the suggested concept differs are provided below**.**

The Axis M3058-PLVE Network Camera is a cutting edge fisheye camera developed for crowd

tracking and heatmap generation.that offers 360degree coverage,infrared night vision and preloaded Ai analytics, making it the perfect camera for conference halls,retail establishments and airports.However its high cost and limited customizability restricts accessibility for small scale projects.[1]

Similarly, Avigilon Occupancy Management offers AI driven video analytics that provides us with real-time information on crowd patterns and room capacity[2].Our proposed system aims to address these gaps by utilizing open source tool like TensorFlow,PyTorch and OpenCV,in addition NIVIDIA Jetson Nano for edge computing this cost effective approach includes a web based interface ,live tracking and heatmap generation while complying with GDPR and KVKK regulations, ensuring user privacy.

  .

**Figure 1:Avigilon camera Figure 2:Axis camera**

**⦁ 1.3.2. Concepts**

The conceptual solutions we came up with are different but share many of the same components, with some adjustments. They all meet the main requirements of the project and can achieve the end goal.

The first solution is straightforward and simple to implement, which increases our chances of success without running into many complications. However, it may not be very accurate because it uses pre-trained models that are not customized specifically for our dataset or environment.

The second solution is cost-effective compared to its potential results and uses a more customized AI model. It performs well, but there is a risk that the NVIDIA Jetson Nano could overheat during heavy processing. Although it requires more advanced tracking methods, this solution strikes a good balance between accuracy and complexity.

The last solution is the most complex and expensive. It offers the best performance and includes all required features, but it also uses more resources than necessary to achieve accuracy.

After considering all options, we decided to go with the second solution. It shows the most potential to reach our goals in an efficient and realistic way. While it may require regular system checks to avoid issues like overheating, this is a small trade-off for a cost-effective and achievable design.

**Table 1. Comparison of the three conceptual solutions.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Concept 1 | Concept 2 | Concept 3 |
| Cost | low | low | high |
| Complexity | low | medium | high |
| Performance | low | medium | high |
| Features | high | high | medium |

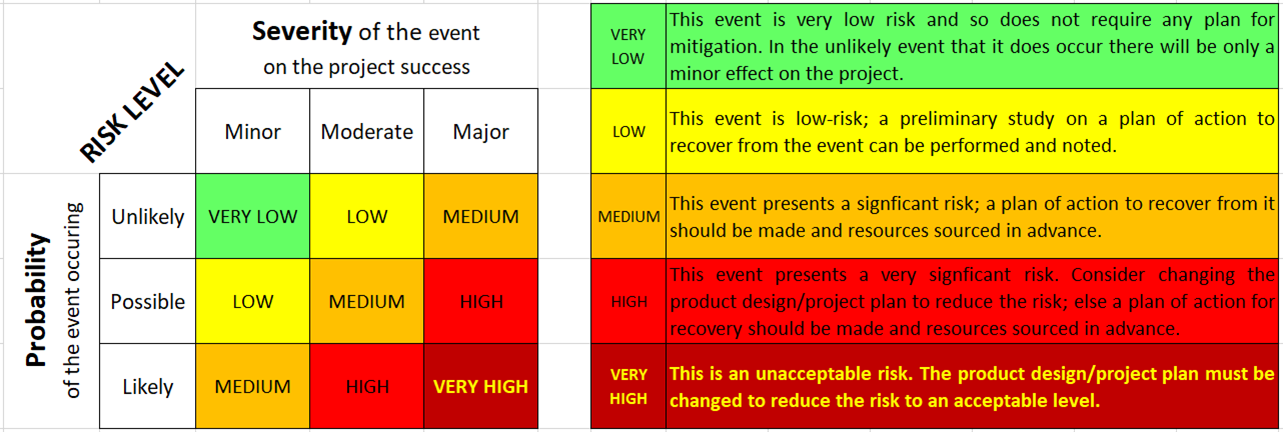
Table 1 compares different conceptual solutions with respect to the four most important requirements; Concept 2 is chosen for this project due to its low cost for the end result it may give us, with not much complexity to make it feasible.

## 2.6. Risk assessment

Risks of our project are very important to identify and find solutions to avoid them in the future. It gives the chance to plan ahead and be able to work more efficiently. The project that is needed to be worked on is very timely and has a deadline that we must meet, due to that the ability to find the problems that we may face helps reduce the time we need to spend on it. Analyzing the risks helps give a better understanding of the capability of your components and their advantages and disadvantages.

To mitigate these risks, we have to come up with plans on how to solve them. For the hardware components they will have to be tested individually for functionality. In addition, with the software as they have to cohesively work together. Also the software parts need a lot of test trials and feedback sessions. By actively working together and solving the problems beforehand, we will be able to achieve our goal with the least amount of issues.

**Table 5. Risk matrix**

******

**Table 6. Risk assessment**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Failure event | Probability | Severity | Risk level | Plan of action |
| Sensors does not operate accurately under different lighting | Likely  The sensors work according to the work done on them | Moderate  Requires planning for and adjustment | HIGH | Apply test trials on the sensors under different lightings, mainly dim ones. Adjust the hardware and software components as needed. |
| NIVIDIA Jetson Nano overheats | Possible  The component is unreliable and it could happen with lack of attention | Major  It will affect the whole system from operating | HIGH | The addition of a cooling system or addition of another system to take some weight off the Jetson Nano. |
| The setup of the camera fails to reveal the whole premises | Likely  The positioning of the camera is vital and may not be placed properly first time | Minor  It is easily fixed and can be changed based on our liking | MEDIUM | The group will have to guide each other in order to find the best placement for the camera and with a couple of testing it will work. |
| AI is unable to perform object detection in crowded areas | Possible  AI should be programmed to function under such states and in the case it is not it will not operate as required | Major  This could affect the project outcomes as identifying the people in the room is a requirement | HIGH | Train the AI to operate under crowded area repeatedly until it runs properly. |
| Bad internet connection | Likely  Such occurrences will happen and preparation is important | Moderate  With the right planning it may not cause as much damage to the final result | MEDIUM | Design the product to function offline and be able to store the data until reconnected to the wifi |
| Collected data face privacy problems due to GDPR/KVKK violations | Possible  The rules are placed to be implemented and if not done it will be violated | Major  Not meeting the laws appointed may cause harm to those creating the project | HIGH | Hide the private data that you collect by encrypting it to meet the privacy regulations |
| Difficulties using the web application | Possible  The web application needs to be studied in order for the person utilizing it can work on it | Minor  The issue is solvable and attainable | LOW | Create tests with user and based on their feedbacks adjust the interface |
| Heatmap does not generate quickly | Possible  The algorithms need a lot of testing before reaching your goal | Moderate  It could affect the end result and make them inaccurate | HIGH | Develop faster algorithms to achieve a quicker generation of heatmap of your environment |
| Goes beyond budget | Unlikely  As the products we need for the project are unexpensive | Moderate  of the chance we go over budget we will not reach the required features needed | LOW | Keep track of the products that we are using and the money spent on them. Make sure to take care of the components as not to have to repurchase and add to the costs |



## 3.2. Ai crowd tracking application

This sub-system is responsible for detecting and counting individuals as well as identifying entry and exit points to track crowd movement. It combines input from sensors and cameras to produce accurate live data, including crowd density and heatmap generation. The output provides insights and analytics on room usage and crowd behavior through a web-based interface. Additionally, the system integrates hardware components with AI models for object detection and tracking to ensure smooth and efficient operation.

### 3.2.1. Requirements

The application must be capable of handling various challenging scenarios, including crowded halls, multiple entry and exit points, as well as changing weather and lighting conditions, without causing significant drops in performance. It should operate smoothly and integrate seamlessly with the NVIDIA Jetson Nano platform and the accompanying web application. A key function of the system is to provide accurate and reliable object detection to identify and count individuals in real time. The system should also effectively track movement patterns to support crowd management. Additionally, it must generate heatmaps that visualize crowd density and update these regularly to help with analyzing space utilization. Overall, the system should deliver timely and consistent live data with minimal delays, ensuring the application remains responsive and informative under different conditions.

### 3.2.2. Technologies and methods

The sub-system uses the NVIDIA Jetson Nano as the main processing unit to perform real-time analysis and computation. Wide-angle and fisheye lens cameras, along with optical and electronic sensors, are employed for tracking individuals and providing comprehensive spatial coverage.

For object detection, the system uses YOLOv5, which was chosen after extensive testing for its effective performance with our dataset. DeepSORT with an improved ReID method is utilized for tracking individuals across frames. Image processing tasks such as frame handling, preprocessing, and heatmap overlay are performed using OpenCV.

To generate heatmaps, the system uses Scikit-learn, Matplotlib, and OpenCV to create real-time grid-based and density-based visualizations.

Finally, the web interface is developed using React.js for the frontend, providing an intuitive platform for live monitoring and visualization. FastAPI is used for the backend to manage API communications and data handling, ensuring a robust and real-time implementation of the system.

### 3.2.3. Conceptualization

Our high-level design breaks the system into 3 conceptual modules: inference, visualization, and integration each optimized for accuracy, latency, cost and maintainability.

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Purpose** | **Final choice** | **Justification** |
| **object detection** | Detect individuals in real-time using camera feeds | YOLOv5 | Easy setup,High accuracy, and speed on Jetson hardware |
| **object tracking** | Track individuals and Maintain IDs for each person across frames | DeepSORT with Re-ID | minimizes ID switches and preserves identity through motion and occlusions |
| **heatmap generation** | Visualize crowd density in real time | Track centroids → accumulate 2D grid → Gaussian blur | Smooth, interpretable visual of traffic patterns |
| **web interface** | Display tracking results and heatmaps to users |  |  |
| **deployment** | Ensure system runs reliably |  |  |

### 3.2.4. Physical architecture

The physical architecture is the material that will be used for the AI to program the system.

|  |  |
| --- | --- |
| **Components** | **Purpose** |
| **NIVIDIA JetsonNano** | Collects video and sensor data in real time. Sends data to the PC for inference. Receives and displays results. |
| **PC or Laptop** | Runs the tracking and heatmap generation system ,processes data from jetson and sends the results. |
| **Fisheye camera** | centrally mounted on the ceiling; providing a 360° view of the room |
| **Sensors** | Assist in directional movement detection and initial counting. |
| **Keyboard, Mouse, Monitor** | Connected to the Jetson Nano for local control, debugging, and result visualization |

The materials of this project are very valuable in achieving the desired outcome. The AI team is responsible for ensuring that these components feed data effectively into the AI models. Coordination with the Electrical Engineering team is crucial to ensure hardware-software integration. This collaboration enables the system to function accurately and efficiently in real time, especially under dynamic environmental conditions and reach the desired outcome of this project.

### 3.2.5. Materialization

The first phase in the AI subsystem materialization was data preprocessing. We began by normalizing and resizing images to ensure consistency across the dataset. We also incorporated a model that could reconstruct frames back into video format for testing purposes. Early in this process, we encountered issues with label formatting, which caused the initial YOLOv5 model to perform poorly, failing to detect individuals effectively.

For object tracking, we implemented DeepSORT; however, initial tests showed inconsistent tracking and random shifts between frames, primarily due to inaccurate detections from the YOLOv5 model.

To improve detection accuracy, we initially transitioned to MMRotate’s R3Det model, designed for precise rotated object detection under fisheye distortion. This required extensive dataset preparation, including converting YOLO-style annotations to MMRotate’s format — which uses radians for angle measurements and normalized bounding boxes. We organized the dataset into a structured directory and developed a custom dataset class within MMRotate to handle our fisheye-specific data, updating the \_\_init\_\_.py for dataset registration and modifying configuration files to include custom paths and pipelines. Additionally, anchor sizes were optimized, hyperparameters fine-tuned, and training pipelines adjusted to improve localization accuracy. These efforts resulted in better bounding box alignment, reducing identity drift and improving overall detection performance.

However, after thorough evaluation of several models, we determined that YOLOv5, once adapted to handle rotated labels by converting them to an axis-aligned format, achieved the best balance of speed, accuracy, and integration ease.

|  |  |  |
| --- | --- | --- |
| **Models Used** | **Summary of Results** | **Outcome** |
| **YOLOv5** | We picked this first because it’s fast and light. It struggled with rotated labels on fisheye images at first. | Didn’t work well at first due to misaligned boxes, but after fixing labels, it got the best accuracy (mAP 0.995) and we used it in the end. |
| **YOLOv8** | This was a little better at detecting and tracking than YOLOv5. | Still had problems with label formats and fisheye distortions, so it wasn’t much better overall. |
| **EfficientDet** | Balanced accuracy and design or normal images | Didn’t do well on fisheye images and was too slow for real-time on the Jetson Nano. |
| **MMDetection (Rotated Anchors)** | Had cool features for rotated objects. | Couldn’t make pretrained weights work well with our fisheye data, so training was unstable. |
| **Custom CNN Backbones** | We built our own models for full control (like Darknet-53). | Overfitting happened because our dataset was small, so accuracy was too low for real use. |
| **YOLOv8-OBB (Ultralytics)** | Came with support for rotated boxes right away. | Needed polygon labels and YAML files that didn’t fit our training setup, making it hard to use. |
| **Fast R-CNN** | Traditional detection model.   |  | | --- | |  | | Training didn’t work on our data — the loss went crazy and the model failed. |
| **MMRotate – R3Det** | Great at detecting rotated boxes and worked well on fisheye images. | Worked well but was slower and harder to use than YOLOv5, so we switched to YOLOv5 in the end. |

A) B)



**Figure 9. YOLOv5 detection performance under varying lighting conditions.**

(a) Well-lit environment: High-confidence detections (p > 0.85) with precise bounding boxes.

(b) Low-light environment: Reduced confidence scores (p = 0.65–0.80) but maintained spatial accuracy.

**Live Tracking:**

To deploy Live tracking we experimented with 3 tracking pipelines.Each setup was evaluated based on identity consistency,visual clarity and ReID effectiveness.

1. **1.Approach : DeepSORT Default configuration for tracking**

Easy to set up but resulted in jittery inaccurate tracks and Frequent ID changes due to detection noise and inadequate ReID capabilities.

1. **Approach: DeepSORT + supervision:**

This approach yielded better results with cleaner tracking and better layout however the bounding box and text sizes were often distracting and it struggled with ReID.

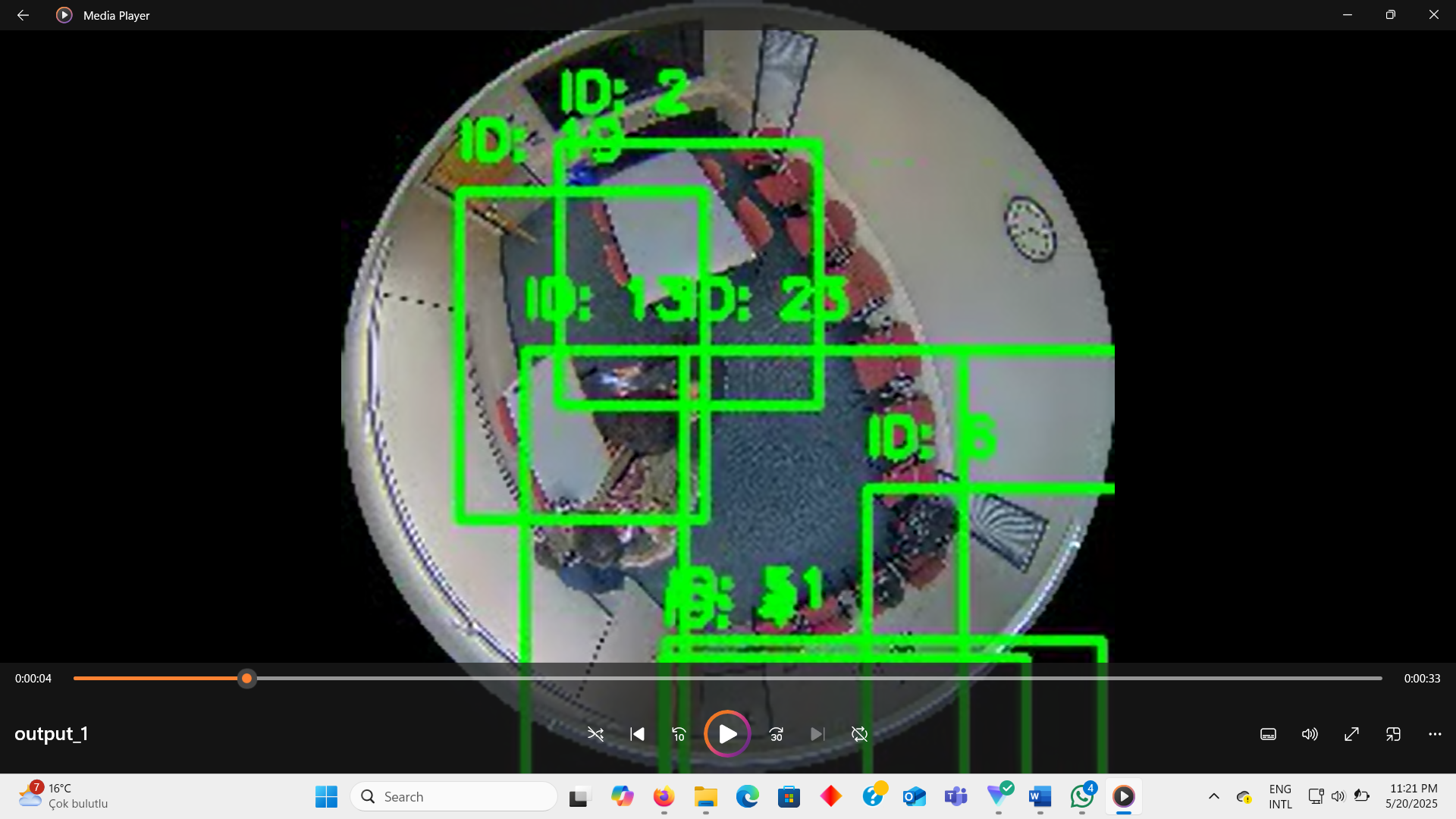
1. **Approach: ByteTrack + supervision:**

It was the hardest to set up.Additionally, we tried a different layout using supervision. This layout was much cleaner and solved the issue of the distracting IDs and boxes.While this approach has better ReID it struggled with accurate detection.

**Final Approach:DeepSORT + ReID + supervision:**

Finally we decided to use DeepSORT enhanced with a ReID model to improve accuracy and supervision for layout.This approach enables best ID preservation out of the 4 approaches , and flexible layout customization. It offers the best tradeoff between simplicity and ReID performance.

**Figure 10: Approach 1 Figure 11: Final Approach**

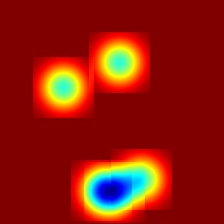
**Heatmap Generation For Crowd density:**

While Continuing to improve our models accuracy we simultaneously began developing a real time heatmap generation function.This function was built using a custom YOLOv5 for person detection and was based on an open source implementation from the repository CrowdMap[11] which was modified for our case.

The development process starts with frame by frame detections using YOLOv5 where detection and bounding boxes are extracted followed by generating a 2D gaussian kernel centered on the detected persons location on a grid. To enhance interoperability the grid or map is colored and normalized using OpenCVs COLORMAP\_JET.This colored heatmap is blended with the original frame to produce an overlay visual that combines raw scenes with density insights.

* heatmap.jpg
* overlay.jpg
* boundingbox.jpg
* An optional video showing the heatmap stream.

**Figure 10: heatmap generation output model 1**

### 3.2.6. Evaluation

The initial YOLOv5 model faced challenges with bounding box shifts caused by object rotation, which led to inaccurate detections and tracking instability. To solve this issue we tested several alternatives including an MMRotate model which achieved mAP of 0.90.However, the final refined YOLOv5 model outperformed MMRotate in detection accuracy and offered a simpler, easier-to-implement setup.

By transforming the label format from rotated to axis aligned. The YOLOv5 model produces more accurate detections with minimal position drift.This enhances tracking stability and reduces jittering and ReID switches.The system supports live tracking and heatmap generation where blue regions correspond to higher crowd density.Additionally, users have the option to save the tracking and heatmap results for further analysis.

# 4. INTEGRATION AND EVALUATION

## 4.1. Integration

The Electrical and AI subsystems are combined during the integration phase to form a unified system for real-time room occupancy monitoring. The AI subsystem, initially intended to run on an NVIDIA Jetson Nano, will communicate with the Electrical subsystem which includes cameras, sensors, and an Arduino.The Electrical subsystem is responsible for setting up the cameras and sensors for recording room entry and exit events.while the AI subsystem processes the data and runs object detection, live heatmap generation and tracking.e to compatibility issues and limited memory on the Jetson Nano, all AI processing will be performed on a PC. The jetson will send live feeds to the PC and receive and display the results on the jetson monitor. This will be done through FastAPI/FLASK app and a uvicorn server. During integration, we first attempted to install and run our entire Python/OpenCV/YOLOv5 inference stack directly on the NVIDIA Jetson Nano, only to discover that its legacy Linux distribution lacked compatible library versions and package repositories; unable to resolve dependency conflicts, we then tried linking the Nano to our development laptops via SSH and USB-over-IP, but persistent driver mismatches and OTG compatibility issues prevented reliable remote execution. In a third effort, we connected a keyboard, mouse, and HDMI display to the Nano and convened a setup session in our university lab, where local firewall and proxy restrictions blocked the device’s internet access and thwarted online package installation. We also experimented with MMRotate’s R3Det rotated-object detection implementation on the Nano, its training would run for roughly 40 minutes before exhausting memory and crashing, so we abandoned that route as well. Undeterred, we transferred the complete application folder from our Windows environment onto the Nano using a USB flash drive, yet filesystem permission errors and unresolved runtime dependencies again led to build failures. After exhausting these trials, we decided to offload all AI inference processing to a dedicated PC-based server, where dependency management proved straightforward, while retaining the Jetson Nano exclusively for raw data capture, sensor interfacing, and video streaming.

## 4.2. Evaluation

The NVIDIA Jetson Nano was successfully set up and integrated into the system. The sensors developed by the Electrical team function as expected, providing reliable data for occupancy monitoring.

The YOLOv5 object detection model demonstrated excellent performance, achieving a high training mAP of 0.995, indicating both accuracy and speed suitable for real-time applications.

Live object tracking was effectively implemented using DeepSORT with Re-identification (ReID) and the Supervision framework, resulting in stable and consistent tracking of individuals across frames.

Heatmap generation was also successfully integrated, with blue regions clearly indicating crowded areas, providing an intuitive visualization of occupancy density.

Overall, the system meets the design goals for accuracy, speed, and usability in real-time room occupancy monitoring.

# 5. SUMMARY AND CONCLUSION

1. This project integrated the Electrical and AI subsystems to create a real-time room occupancy monitoring system.The Electrical team successfully developed and setup the sensors and cameras, while the AI subsystem trained a YOLOv5 model for fast and accurate object detection. By combining YOLOv5 with DeepSORT tracking and heatmap visualization, the system can reliably detect, track, and visualize occupancy density in real time. Despite initial challenges with label format and after trying multiple alternatives.YOLOv5 was successfully trained (mAP 0.995).Another challenge was processing limitation and compatibility issues on the NIVIDIA jetson.solution such as offloading AI processing to a PC ensured stable and accurate tracking performance. To um up,the project met its objectives of delivering an effective, efficient, and user-friendly occupancy monitoring solution. Future work could focus on running the system only on jetson and improving ReID in extremely crowded settings.

# ACKNOWLEDGEMENTS

We wish to thank our advisers prof. Fatih kahraman and prof.Arif Ergin . Also acknowledge any other help/support from friends, technicians and other staff etc...

This work was partly/wholly funded by Bahçeşehir University

We wish to express our sincere gratitude to our advisers for their invaluable guidance, cand insightful feedback throughout the course of this project. We acknowledge **Bahçeşehir University** for providing the resources and facilities that enabled us to complete this project successfully.

# REFERENCES

1. Axis Communications, “AXIS M3058-PLVE Network Camera - Support Resources,” Accessed: Nov. 21, 2024. [Online]. Available: <https://www.axis.com/products/axis-m3058-plve/support#support-resources>
2. Avigilon, “Avigilon Security Cameras,” Accessed: Nov. 21, 2024. [Online]. Available: <https://www.avigilon.com/security-cameras>
3. K. Sun, X. Wang, T. Xing, S. Liu, and Q. Zhao, "High-accuracy occupancy counting at crowded entrances for smart buildings, <https://www.sciencedirect.com/science/article/pii/S037877882400625X>
4. Kim, J., Choi, A., Moon, H. J., Moon, J. W., & Sung, M. (2023). Occupancy estimation using IoT sensors and machine learning: Incorporating ventilation system operating state and preprocessed differential pressure data. <https://www.sciencedirect.com/science/article/pii/S0360132323010065?via%3Dihub>
5. J. Yun, D. Kim, D.M. Kim, T. Song, J. Woo, Gan-based sensor data augmentation: application for counting moving people and detecting directions using pir sensors, Eng. Appl. Artif. Intell. 117 (2023) 105508. <https://www.sciencedirect.com/science/article/pii/S0952197622004985>
6. Atlassian, “Heatmaps: The complete guide to heatmaps,” *Atlassian*, Accessed: Nov. 21, 2024.

<https://www.atlassian.com/data/charts/heatmap-complete-guide>

1. Viso.ai, “Object tracking: Complete guide to deep learning object tracking,” *Viso.ai*, Accessed: Nov. 21, 2024.

<https://viso.ai/deep-learning/object-tracking/>

1. S. Li, M.O. Tezcan, P. Ishwar, and J. Konrad, “Supervised people counting using an overhead fisheye camera”, *IEEE International Conference on Advanced Visual and Signal-Based Surveillance (AVSS)*, Sept. 2019
2. MMRotate Contributors, “MMRotate: Rotated Object Detection Based on PyTorch and MMDetection,” *OpenMMLab Technical Report*, 2023. [Online]. Available:<https://mmrotate.readthedocs.io/en/latest/intro.html>
3. **Ultralytics**, *YOLOv5 Documentation*. 2023. [Online]. Available: <https://docs.ultralytics.com/models/yolov5/>. [Accessed: April 15, 2025].
4. K. Somani, *CrowdMap*, GitHub repository, [Online]. Available <https://github.com/kr1shnasomani/CrowdMap>,

Use the **IEEE style** when listing references. Try to add MINIMUM 10 references.

*A good guide can be found here:* <http://libguides.murdoch.edu.au/IEEE/>,

and many examples here: <https://libguides.murdoch.edu.au/IEEE/all>

# APPENDIX A

*Information that does not fit naturally into the main body of the report can be put into an appendix. Typically this would be long sections of software code, product user manuals, large tables of validation results, etc.*

*An example of providing source code is shown in this appendix. Display the source code in a* monospace (fixed-width) *font and single-spaced.*

*Alternatively give a link to an online code repository.*

Code for packing (and unpacking) an occupancy value [0,1] into an unsigned char.

// maximum error = +-0.25% (standard dev. = 0.5/sqrt(12)

// 0% and 100% occupancies are stored exactly.

unsigned char occByte;

if (occ==0.0) { occByte = 254; } // 0% occupancy stored exactly

else if (occ==1.0) { occByte = 255; } // 100% occupancy stored exactly

else { occByte = int(occ\*200.0); }

// unpack occupancy

float occFlot;

if (occByte==254) { occFlot = 0.0; }

else if (occByte==255) { occFlot = 1.0; }

else { occFlot = (occByte+0.5)/200.0; }

# APPENDIX B

**Figure B.X:** YOLOv5s Training Log Output

*Epoch-wise training and validation loss values captured during 50-epoch training of YOLOv5s model using Google Colab with CUDA support.*