TED UNIVERSITY

CMPE492

Computer Engineering

Library Occupancy Detector

Low-Level Design Report

12.10.2024

Team Members

Burak Koç

Buse Öner

Furkan Safa Altunyuva

Table of Contents

Introduction

A real-time computer vision project called the Library Occupancy Detector System was created to monitor how often TED University's seats and tables are used. The system determines whether library seats are occupied by humans or "held" by personal possessions (such as cell phones) by utilizing OpenCV and the YOLOv10 object detection method.This technology optimizes the utilization of space and enhances the learning environment by providing real-time occupancy statistics to students and library staff. This is designed to let students see the current occupancy rate prior to going to the library and help them make informed choices.
The library administration can analyze the usage patterns and even detect chairs that are not occupied for extended periods, thereby improving space utilization.
It focuses on correctness, efficiency, confidentiality, ease of use, object detection, machine learning, and video analysis. The programming language used in the system is Python; moreover, the recognition of objects has been done using the YOLOv10 model for its excellent performance in recognizing multiple objects within a single frame in real time.

Object Design Trade-offs

Cameras: By weighing the balance of cost and functionality, we used two webcams to detect real-time occupancy. A decision like this allows a variety of effective coverage in the library without more sophisticated camera systems. On the other hand, this will undermine the accuracy when there are limited cameras in larger or more complex environments, but this is just the first step into the project.
Storage: Storing raw images for potential privacy issues, we decided to go with processing video streams in real time. This removes the possibility of any potential violations of privacy, while limiting our ability to do post-event analysis or enhancing the model using past data.
Performance: The YOLOv10 object detection model deployed gives way for the effective detection of objects with low latency. In this trade-off, rapid feedback enhances user experience but may, over time, compromise the sustained learning ability of the system.

Privacy: We anonymized all data we collected, so we don't keep any personally identifiable information. Of course, this compromise ensures that no privacy rules such as KVKK are violated; however, it limits the future opportunity for data-driven insights that would improve the system's performance.

Interface Documentation Guidelines

The interfaces between Data Collection, Image Processing, and Backend Service follow RESTful API standards, utilizing FastAPI. Communication happens over HTTP, with JSON responses ensuring seamless data transfer. All API endpoints are documented in OpenAPI format, and authentication uses JWT tokens to secure access.

- **Authentication:** JWT tokens for secure requests.
- **API Documentation:** OpenAPI format, detailing request/response schemas.

Engineering standard

The project adheres to the following engineering standards:

- **UML Diagrams:** Employed for system, class, and interaction designs.
- **IEEE 1471-2000:** Followed for creating architectural documentation.
- **ISO/IEC 27001:** Applied to guarantee data security and comply with KVKK (Personal Data Protection Law).

Definitions, acronyms, and abbreviations

- KVKK: Personal Data Protection Law, the Turkish legislation governing the protection of personal data, in accordance with Kişisel Verilerin Korunması Kanunu.
- myTEDUPortal: The website where students, faculty, and staff can access academic and administrative services, such as course registration, grades, and schedules.
- TeduAPP: The "Tedu App" is an application designed and developed by Tedu students, aiming to guide students about school activities, class schedules, and communities within the school under the slogan of "by students, for students.
- Occupancy Rate: The ratio of occupied seats to the total seating capacity in the library, basically representing the percentage of students currently utilizing seats.
- Hold Status: A situation where a seat or table is considered full due to the presence of personal items left by a student.
- YOLOv10: You Only Look Once, is an object identification method known for its high accuracy and real-time detection capabilities.
- Real-Time Detection: The ability of the system to process/analyze and show data almost instantly.

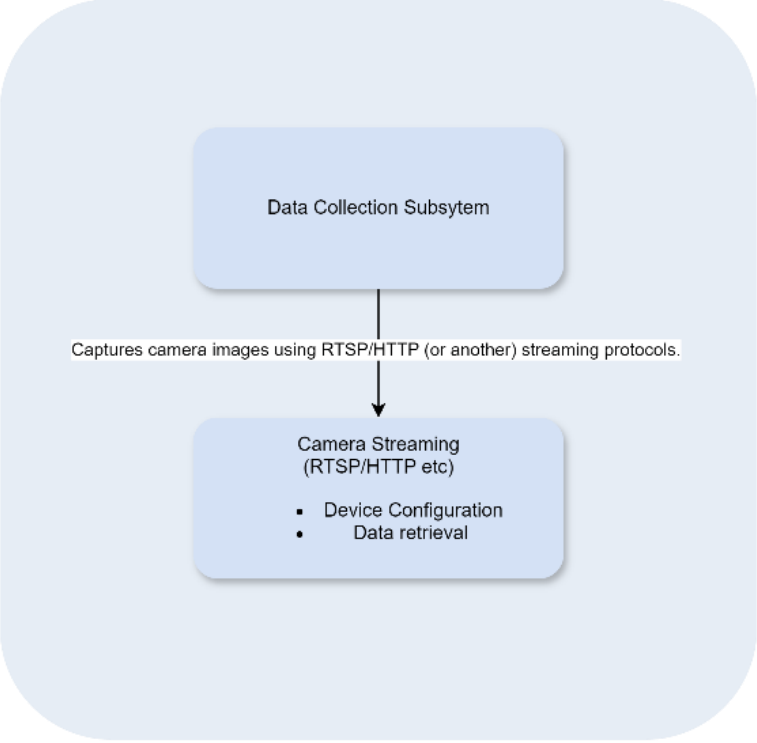
Packages

There is no existing software architecture related to the current project within TED University. Existing software architecture specifically addressing the library occupancy using image processing for real-time calculation of occupancy rates, without data storage for security reasons, have not been able to be identified. We are doing this project from scratch to fill this gap and satisfy the students' needs.

Overview

Our system has four main subsystems which are Data Collection, Image Analysis, Backend Service and GUI.

Data Collection

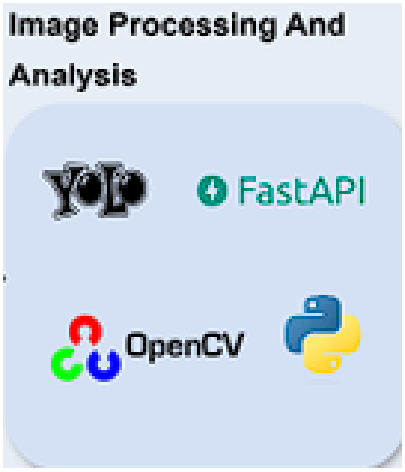


This subsystem is responsible for capturing camera images from cameras by implementing a streaming protocol (e.g., RTSP or HTTP). It will ensure proper device configuration and data retrieval based on our university's camera system. But in the first

version of our system, we are going to use two webcams to gather camera images. This subsystem captures live videos from a webcam. The StableCam class is responsible for initializing the webcam and setting the resolution.

- **Attributes**:
  - stream_url: Webcam stream URL.
- **Methods**:
  - start_stream(): Initializes the camera stream and sets the resolution.
  - capture_frame(): Captures individual frames for object detection.
  - set_roi(): Defines monitor specific ares where occupancy needs to be detected.

Image Processing



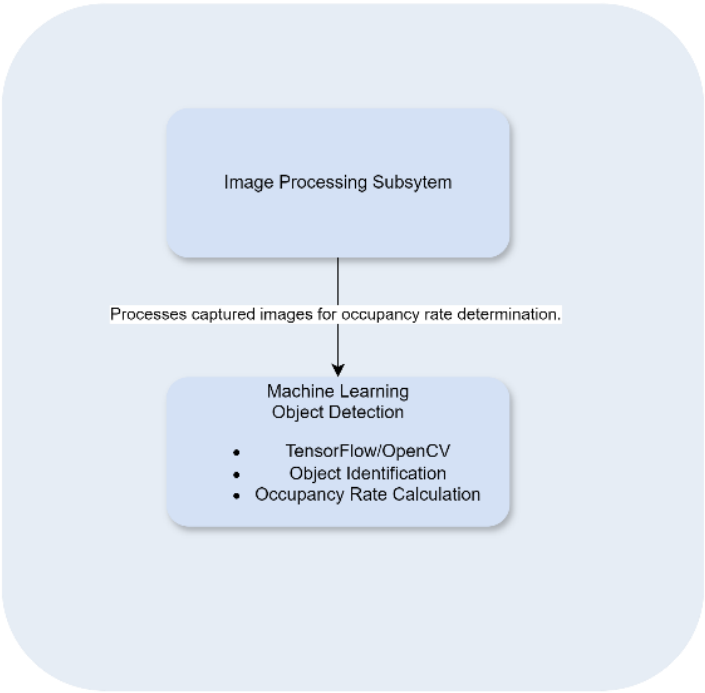
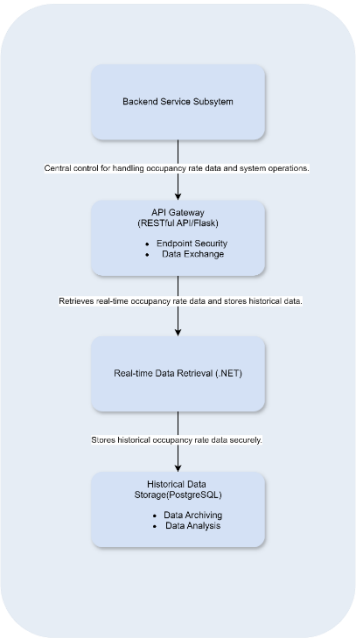


Image Processing subsystem is responsible for processing the captured images to determine the occupancy rate. This subsystem includes machine learning models (e.g., TensorFlow or OpenCV) for object detection to identify tables (hold, empty, occupied) and occupancy rate. Algorithms are implemented to calculate the occupancy rate based on the number of occupied tables versus total tables, ensuring real-time accuracy.

- **Attributes:**
  - model: YOLOv10 model for detecting objects.
  - chair_roi: Coordinates for each chair's region of interest (ROI).

- **Functions:**
  - run_yolov10_model(frame): Processes the frame using YOLOv8 and returns a list of detected objects.
  - count_objects(detections): Filters the detection results to count the number of people in the frame.

Backend Service



Backend Service is like the central control room for handling everything related to occupancy rates. It's designed to make sure our system runs smoothly and keeps track of how many people there are in the libraries.

- **API Gateway:** Utilizes technologies like RESTful API endpoints with Flask or FastAPI to facilitate seamless interaction with other subsystems. This allows for efficient and fast data exchange and retrieval of occupancy rate data.

  Functions:

  - send_data_to_api(formatted_data): Sends the formatted occupancy data to the API.

- **Real-time Data Retrieval:** The .NET-based backend to retrieve real-time occupancy rate data from the Image Processing subsystem. We ensure timely access to fresh data for display on the user interface or further analysis.

- **Historical Data Storage:** A PostgreSQL database for secure storage of historical occupancy rate data. This enables functionalities such as:

  - Storing occupancy rate data over time to analyze trends, occupancy and identify usage (e.g., Occupancy in midterm and final weeks).
  - Manipulating data for specific periods or locations (libraries).
  - Providing historical data for the user interface.

Class Interfaces

Since the code provided is not organized into formal classes but instead uses functional programming, the interfaces can be represented as functional modules:

1. Webcam Interface

Functionality:

- Start Webcam: Initializes and configures the webcam for image capture.
    - Method:
        - **initialize ()**: Prepares the webcam for capturing video.
        - **release ()**: Releases the webcam resources when no longer needed.

2. Image Capture Interface

Functionality:

- Capture Frame: Captures a frame from the webcam.
    - Method:
        - **capture_frame ():** Retrieves a single frame from the video feed.

3. Object Detection Interface

Functionality:

- Load Model: Loads the YOLOv10 model for object detection.
    - Method:
        - **load_model ()**: Loads the YOLOv10 model from a specified path.
        - **detect_objects ()**: Analyzes a captured frame to identify objects.

4. Data Processing Interface

Functionality:

- Determine Chair Occupancy: Checks if a detected object is occupying a specific chair.
    - Method:
        - **check_occupancy ()**: Determines if detected objects occupy specified regions.

5. Display Interface

Functionality:

- Draw Bounding Boxes: Visualizes detected objects and occupancy status on the captured frame.
    - Method:
        - **draw_bounding_boxes ()**: Renders bounding boxes around detected objects.
        - **update_display ()**: Updates the visual output with current status information.

6. Cleanup Interface

Functionality:

- Release Resources: Cleans up the webcam and closes windows when the application ends.
    - Method:
        - **cleanup ()**: Handles the release of resources and the closing of display windows.

References

1. YOLOv10 Algorithm: YOLOv10
2. Software Architecture for Computer Vision: ResearchGate - Software architecture for computer vision: Beyond pipes and filters
3. Deep Learning with OpenCV: OpenCV Blog - Deep Learning with Computer Vision
4. Proposed ICPS Real-Time Monitoring System: ResearchGate - Architecture of the proposed ICPS real-time monitoring system
5. diagrams.net [For other consumer use]. diagrams.net
6. Real-Time Architecture Considerations: Upsolver Blog - Building a Real-Time Architecture: 8 Key Considerations