Distancing & Occupancy Validator

By Mayur Ryali and Kashyap Panda

CS179J: Project in Computer Architecture and Embedded Systems
Winter 2022

Table of Contents

Introduction

As this planet grows ever more interconnected, controlling and reducing the spread of contagious diseases is very important. This has a variety of benefits, some of which include reduced transmission and a decreased strain on the healthcare systems of the world. Slowing the transmission of a disease also delays its epidemic peak, which is the point in time when the most people are infected.

Along with its general importance, there are also several different ways to actually incorporate this topic, each with its own sets of benefits and drawbacks. Out of all of the options, it is generally agreed upon that the act of physically distancing is the most cost effective method to reduce transmissions and spread. This is due to its independence from monetary restrictions and pharmaceutical resources, which is something in common with other methods such as sanitization and protective equipment.

Distancing is useful in many settings where it is easy for disease to spread. These are all situations where there are many humans tightly packed into one room such as movie theaters, sporting events, classrooms, parties, and more. For these areas, the typical distancing requirement is about 6 feet apart.

Along with distancing, another factor that is closely related to it is room occupancy. Many buildings already have this for safety reasons, and it has also been adapted to help with distancing. This is because limiting the amount of people in a room can aid in spreading them apart.

While maintaining a set distance and limiting room occupancy can be effective, it is often difficult to do so. When there are a lot of people, there is no natural way to monitor the distance between bodies, and counting room occupancy is almost impossible unless somebody stands at a high point and counts one by one.

In order to aid in this situation, we have created a device that acts as a distancing and occupancy validation system, which quickly and accurately counts the number of people in a room and checks if they are adequately distanced. It does this using a two-tiered validation system that consists of a camera feed and a LiDAR sensor. The rest of the report explores the design of this project and future plans.

Design & Operation

This project has three main components. The first is a camera that is used to locate bodies and count them. The second is a LiDAR sensor that detects objects in a 360-degree field-of-view. Lastly, both the camera and LiDAR sensor are connected to a PYNQ Z2 which analyzes the data. The following steps through how our device operates and the different components in play.

Camera

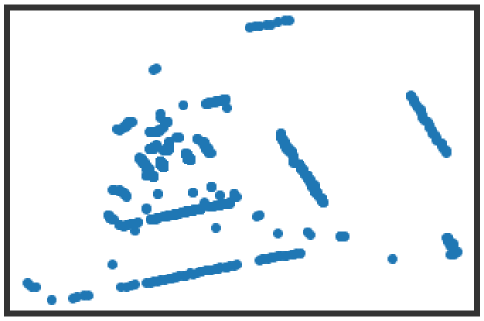
The camera records a live video of the operating space, in 640x480 resolution. As the video is being recorded, the program uses the open-source library OpenCV to modify and analyze the images. First, it converts the image to grayscale to reduce its complexity. Next, it uses a cascade classifier built-in to the library to get the locations of different people in frame and count them.

The cascade classifier is a pre-trained model built into OpenCV. The classifier was trained with several hundred “positive” and “negative” images. A “positive” image is one that is an image of the object the model is trying to classify whereas a “negative” image is one that is not. For example, if we build a dog classifier, a “positive” image is a picture of a dog and a “negative” image is one that is not a dog like a picture of a cat. Once the classifier is trained it can be applied to regions of an input image to begin classification [1][3].

When we apply the cascade classifier on our camera image, OpenCV splits the image into multiple sections. The classifier runs through each section and determines whether or not it thinks that section is part of a “positive” image. After running through all of the sections, the classifier identifies all of the humans in the camera frame. Our program then stores the positions of the human found in the camera frame. The positions are stored as x-y coordinates if more than one person is identified. While this is happening, the program also counts the number of positions detected and stores it as the room occupancy [1][3].

LiDAR

The LiDAR sensor, specifically a Slamtec RPLIDAR A1, is used to validate the data the program gets from the camera. The sensor performs a 360-degree scan of the space. The data is returned as a list of detected objects in polar coordinates. The returned list contains the distance, in millimeters, of the object from the sensor as well as the angle at which the object was detected. The image below shows a sample scan of a room with various objects scattered around it. Starting from the top and going clockwise, the borders correspond to the 0, 90, 180, and 270 degree positions.



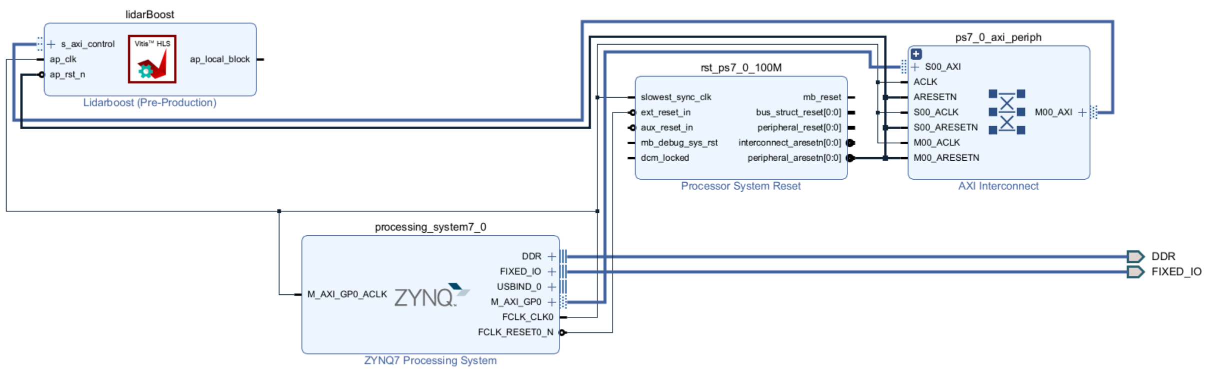
If multiple people are detected, our program takes the data from the camera and maps the position values to LiDAR angles. This mapping converts an x-coordinate from the image position into a possible angle for the LiDAR sensor to check. For example, a position detected at (0, 220) could correspond to a LiDAR angle of 45 degrees. This method of mapping is hard-coded, so it must be changed and recalibrated for each unique room in use. Once all of the human positions from the camera feed have been mapped, the program checks if there is any LiDAR data at the mapped angle. If there is, the presence of such data likely means that the detected object is the human being in question. Once two such presences are detected, the program calculates the distance between the two bodies.

The distance calculation is done by constructing a triangle between the two positions. Once the triangle has been constructed, we use the distance formula. In order to preserve accuracy, the program calculates the distance five times and extracts the average. Once we have our final distance between the people detected, the program checks for social distancing and max room occupancy. The program prints the number of people in the room to the console. If the distance calculated is below six feet, the program also prints an alert to distance otherwise it displays a message saying that social distancing is in effect.

While the program was calculating distances, there were sometimes outliers. This is mainly due to the nature of random human movement, both within the video frame and out of frame. In order to deal with the outliers, an algorithm was used that calculates the interquartile range to remove these outliers.

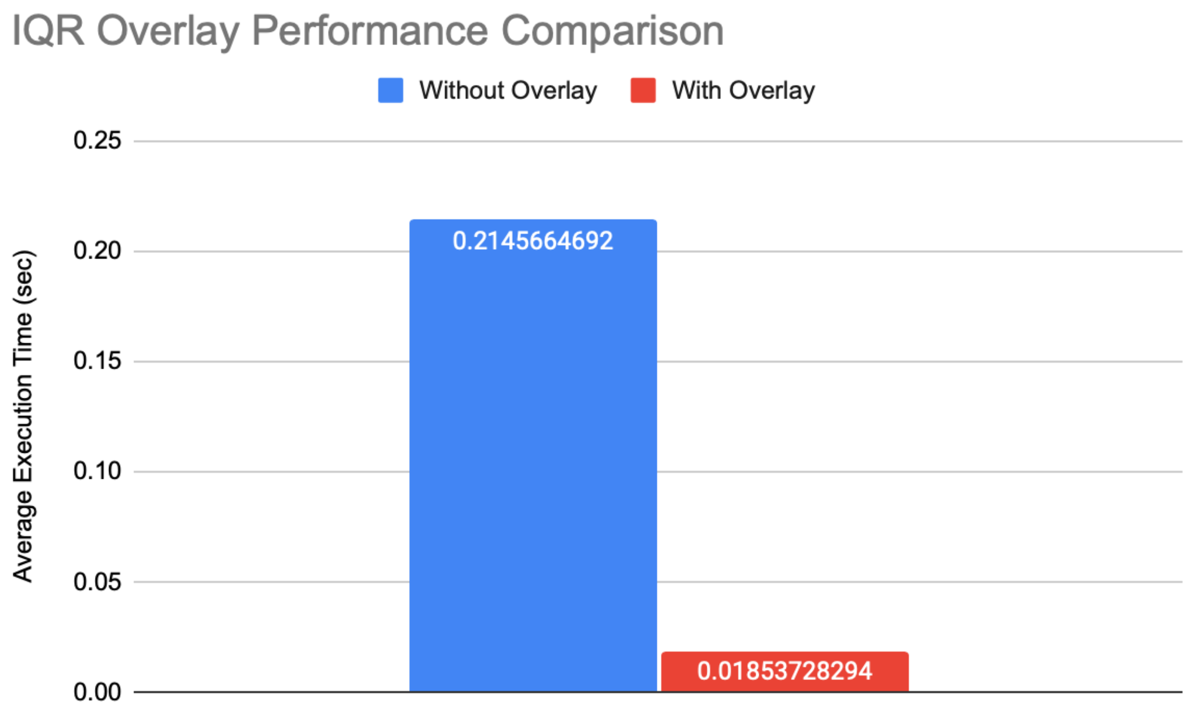
PYNQ Overlay

The PYNQ board was used throughout the distance and interquartile range calculations to reduce the computation time in addition to the overall time it took to print a result in the console. This was done using a custom PYNQ overlay that implemented a wide range of functions. These functions included basic arithmetic operations (addition, subtraction, multiplication, division), LiDAR angle mapping, and interquartile range.



The image above shows the block diagram for the custom overlay, which was created using the Vivado Design Suite by Xilinx. The functions were implemented in a C++ RTL IP using Vitis HLS, which were then applied as part of the lidarBoost block in the diagram. The overlay was primarily configured using the built-in block automation in Vivado in order to connect the HLS IP to the base ZYNQ7 IP block.

The overlay is used in the program with similar syntax as a library function. It is imported before program execution, and the required data is passed into the overlay function whenever the use of one is required [2]. The graph below compares the execution time of the interquartile range function with the overlay and without the overlay.



As seen above, the execution time of the version with the overlay is about 10 times faster than the version without the overlay. This is due to the custom overlay, which optimizes the PYNQ board to speed up calculation and return the data faster.

Technical Problems

We faced a variety of different issues and problems in almost every part of the creation of this project.

One of the initial problems was with integration of the LiDAR device with our program. The Python package provided to us by Adafruit, the seller, was outdated and did not function properly with the version of Python in use with the PYNQ board. This problem was resolved by modifying the provided Python package with another one on GitHub to work with newer Python versions [4].

The biggest problem was with the overlay capabilities. The first issue was with the write_reg() function which is used to write to registers only allowing for integer inputs. The problem with this is most of the data we get from our peripherals is in double or floating point. As a result, we needed to cast a majority of the values to integers which may have cost us some precision in our calculations.

Another issue with the overlays was with the number of overlays we could load at a time. One potential feature we couldn’t implement was flashing the LEDs. The LEDs are under the base overlay while our arithmetic boosting was under our custom overlay. We had to find a way to merge the overlays but couldn’t do so in the time we had for our project.

Conclusion and Future Work

Conclusion

Due to the ongoing COVID-19 pandemic, it is clear that there is a need for better and more effective ways to track physical distancing and building/room occupancy. To fill this need, our project is a device that tracks the number of people in a room and checks if they are properly distanced or not.

This is done by analyzing a camera feed for the locations of different people, and then calculating their distances using additional data from a LiDAR sensor. In order to increase performance, the calculations were done using custom overlays on a PYNQ Z2 board.

We ran into some pretty considerable problems as we completed our project. However, we were able to overcome them pretty well considering the time and resources we had. Overall, this project was an excellent introduction to FPGA programming and its benefits in the world of embedded systems.

Future Work

This project can be further expanded to improve efficiency and usage.

One such usage improvement is with user-facing features. Currently, the device only prints a single alert message with the occupancy count and social-distancing indicator to the console. This could be further improved by instead sending alert notifications to an external screen or smartphone. The device could also have more visual effects such as flashing LEDs if social distancing or max occupancy restrictions are not satisfied.

In addition to user-facing features, we could improve the performance and scope of the device. Currently, the device checks for proper distancing by calculating the distance between two people at a time. While this lets the project function properly, the speed and accuracy can be further increased by checking the distance between multiple people at the same time.

More improvements can also be done to the overlays. Currently we only use the overlay to boost basic integer arithmetic. We could look into incorporating decimal and floating point arithmetic. Furthemore, we could find a way to merge overlays so that we can use two simultaneously. For example, we could use our custom overlay with the base overlay to allow for LEDs to flash.

Github Repository

The Python program, overlay hardware handoff file, and the overlay bitfile can be found in the following GitHub repository:

https://github.com/CS179J-W22/DistanceValidator

References

1. https://github.com/Limerin555/Real-time_Upper_Body_Detection
2. https://pynq.readthedocs.io/en/latest/
3. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
4. https://github.com/Roboticia/RPLidar