TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Microcontroller based Object Tracking for Room Monitoring

Chenyuan Zhao

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

Master's Thesis in Informatics

Microcontroller based Object Tracking for Room Monitoring

Objekt Tracking auf einem Mikrocontroller zur Raumüberwachung

Author:	Chenyuan Zhao
Supervisor:	Prof. Dr. Michael Gerndt
Advisor:	Prof. Dr. Michael Gerndt
Submission Date:	15.10.2021

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Ich versichere, dass ich diese Master's Thesis selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Munich, 15.10.2021 Chenyuan Zhao

Acknowledgments

A thesis may include an acknowledgments section where the author may want to thank certain people, groups, institutions, etc. who provided support.

Abstract

This is the abstract. It is a short summary of your work, consisting of roughly one to three paragraphs. It should give the main ideas of your paper, i.e., the posed problem, a motivation for solving it, your solution method, and your results. Keep it understandable for a general audience. Do not include references.

Contents

1 Introduction

1.1 Motivation

Occupancy estimation of a given space is an attractive use case of IoT (Internet of Things) devices. Knowing an approximate number of people may help by optimization of HVAC (Heating, Ventilation and Air-condition) consumption in a smart building. At home, a continuous monitoring of dwell time in each room may improve the safety of elder residents. Since the Corona pandemic breaks out, there has also been a strong demand for monitoring the people number inside a building for social distance controlling.

Many different sensors could be used to sense the number of people in a given space. Teixeira et al. [1] has classified the signals of human presenting into three categories and given their corresponding sensors. Static traits are the intrinsic attributes of a human, and are detectable even if the human stays still. Examples in this category are weight (measured by pressure sensor), reflectivity (camera) or emissivity (thermal sensor). Dynamic traits are the activities of a human, such as body movement detected by a ultrasonic sensor [2] or speech detected by a microphone [3]. Finally, external traits are the traits of objects carried by a person. A common case is a wearable RFID card [4]. Due to the broad use of mobile phones nowadays, Wifi connection activity could also be a good indicator of human occupancy [5]. Beyond the three categories, a coarse estimation of a large amount of people could also be derived from the power consumption, number of lights switched on or the CO2 concentration in a building [6].

Among all sensors, the camera based method is chosen because its capability to detect human in both static/ dynamic scenarios, and the capability to handle complex scenarios with multiple human co-occurrence. An human count estimation by cameras could be obtained either by absolute counting or relative counting. The former method manages to overwatch the whole region of interest by one camera or by merging information from several cameras. The human count at a given time is determined by one single frame using object detection. Therefore, the counting algorithm is independent of camera frame rate as well as number of people entering/ leaving at the same time. The relative counting method only monitors the entrance of a room. The final count value is obtained by accumulating the number of entering/ exiting events. A tracking algorithm is required, therefore camera frame rate is critical. And for each frame, the tracking must be done strictly within the time window before the next frame is captured. Besides, when multiple people enter/ exit at the same time the algorithm should still be capable to track correctly.

Though absolute counting method may be effective for small rooms, a single camera cannot cover larger rooms. Our approach based on a relative counting method because we want to design a general device that is independent of the possible room space. Such a device is sometimes called a doorway turnstile [7].

However, how to monitor the people by a camera in a public space without intruding their privacy has poses a challenge to the device design. Due to concerns about privacy violation, RGB-cameras are usually excluded from home surveillance designs [8]. Monitoring through a low-resolution infrared camera may solve the challenge fundamentally. The most remarkable feature of this kind of cameras is that they only have several pixels in vertical and horizontal directions respectively. Sometimes these cameras are referred as thermopile sensors. Thanks to the nature of low-resolution, a person could be detected but the identification of the person is impossible. Without the suspicion of leaking privacy, the monitor device is ideal for an application in public space, but also at home. Moreover, despite the fact that infrared cameras are usually much more expensive than common RGB-cameras, a low-resolution infrared camera is still affordable at the cost of low accuracy and scarce pixels, therefore it is suitable for mass deployment.

1.2 Contribution

This thesis proposes a human tracking algorithm which based on traditional CV (computer vision) methods of blob tracking. The algorithm takes top-view frame sequence of a infrared camera as input and outputs the relative count of the event. The absolute number of people in a room is obtained by accumulating the relative counts. A highlight of the algorithm is its capability to tracking multiple human instances, even when their blob representations merge or split. Several examples are shown in Figure 1.1.

1.3 Thesis Overview

In the next chapter, related works will be reviewed. In chapter 3, objective of the design will be described in detail and certain choices of hardware is explained. In chapter 4, the communication protocol between the end node device and the data centralization platform is introduced. The main idea of the purposed algorithm is introduced in

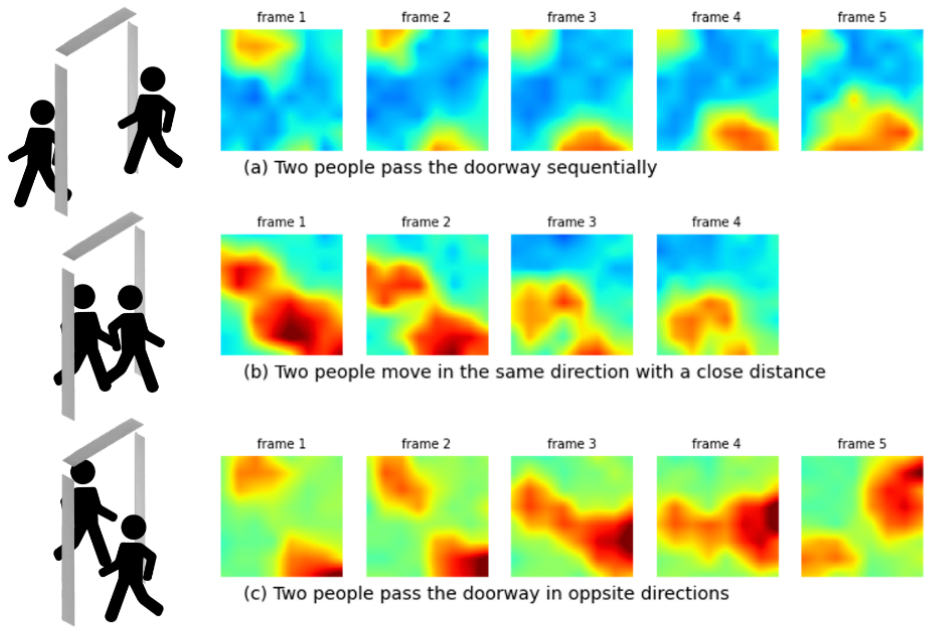

frame 1
frame 2
frame 3
frame 4
frame 5
(a) Two people pass the doorway sequentially
frame 1
frame 2
frame 3
frame 4
(b) Two people move in the same direction with a close distance
frame 1
frame 2
frame 3
frame 4
frame 5
(c) Two people pass the doorway in oppsite directions

Figure 1.1: several corner cases that the proposed algorithm is capable to handle

chapter 5. And chapter 6 shows the final accuracy of the purposed algorithm based on a two-weeks in-situation test.

2 Literature Review

Low-resolution thermopile-array sensors (LR-TAS) have been widely used in health-caring and indoor monitoring because of the feature of preserving privacy. According to the final objective, the related literature could be roughly classified into three categories: indoor localization and gesture estimation [9, 10, 11], room surveillance [12, 13, 14, 15] or human object tracking and event counting [16, 17, 18, 19]. Most of the literature purposed a two-layers algorithm structure, which consists of a object detection layer and a activity classification layer. Regardless of the different final goal of the researches, several similar ideas are found in the object detection algorithms, which has inspired our design. The later data processing procedure is more task-specific and only the literature in the last category are highly related to our design.

Before we dive into the related literature, it is worth mentioning that there are already several commercial thermal sensor products that could track and count human at a high accuracy [20, 21]. Unfortunately, either because the source code of these products are not released or research papers are not published, we cannot include them in this section.

2.1 Detection Algorithms

In the active pixel detection phase, the algorithm needs to distinguish human from the background, namely to answer the question of "Which pixels are occupied by human?"

Mashiyama et al. [22] rearrange the pixels by their reading values in descending order, and take a heuristic that the first *n* pixels could be human, having a prior knowledge that a human usually takes up *n* pixels in a scene. They also compare the average temperature of the first *n* pixels with the rest pixels, only when the difference is large enough the first *n* pixels are considered a human. This method has several drawbacks. The most obvious issue is that it can only detect one person. Moreover, the algorithm depends heavily on the prior knowledge threshold *n*, so it lacks the flexibility to detect different people with different figures. Finally, Trofimova et al. [23] has re-conducted the research and pointed out that it is very hard to determine the threshold.

Trofimova et al. [23] has also proposed their detection algorithm, which makes use of the on-board thermistor to estimate the environment temperature. They found that there is an approximately linear relation between the IR array temperature and the thermistor temperature, which indicates that any higher reading that breaks this linear relation is probably caused by a human. Since the sensor they use has a reading deviation up to $\pm 2.5°C$, a larger deviation from the thermistor temperature is considered a activated pixel. This method is able to detect activated pixels individually. But from temperature readings alone, an IR-TAS cannot distinguish human from other heat emitters such as computer or heater.

Another way to separate the human object from the background is to fully exploit the whole image frame. If human is the only heat source in a thermal image, it is almost safe to assume that the body temperature is higher than the ambient surrounding. Therefore, the human body and the background are naturally separated by temperature. Otsu's method could be applied to each frame individually to find out a threshold, those pixels has temperature higher than the threshold are considered occupied by human [17].

Otsu's method provides good segmentation result but its low speed may be a drawback when real-time performance is required. The average temperature and standard deviation of a thermal image could also be used to calculate the threshold between fore- and backgrounds. Qu et al. [19] have purposed a simple algorithm based on average temperature: any pixels with a temperature in the range of average temperature $\pm 3.0°C$ are regarded as potential human objects and the rest pixels in the same frame are regarded as background. They further calculate the average temperature of fore- and background respectively, denoted as T_o (object temperature) and T_a (ambient). Next, a measurement temperature (T_m) is calculated by the following formula Equation 2.1, where S_o is the size of object and S_p is the total pixel number of one frame. A pixel that is $3°C$ higher than T_m is regarded as non-human heat source and will be excluded from object list.

$$T_m = T_a + (T_o - T_a) * \frac{S_o}{S_p} \tag{2.1}$$

Similarly, Jeong et al. [11] purpose a global adaptive threshold using information of a single image frame. They exploit the fact that a too low standard deviation of an thermal image suggests it contains noises only. The purposed threshold is shown in Equation 2.2, where Max and Mean are the maximum and mean value in one frame, and SD is the standard deviation.

$$Threshold = (Max - Mean) * \left(0.025 + \frac{0.85}{Max - Mean + 1}\right) + Mean - 0.7 \times SD \tag{2.2}$$

Beside the aforementioned methods, a large group of detection algorithms fall into background substraction [24], see Figure 2.1. This method maintains a background model in the processor's memory. A pixel is regarded active if its reading deviates

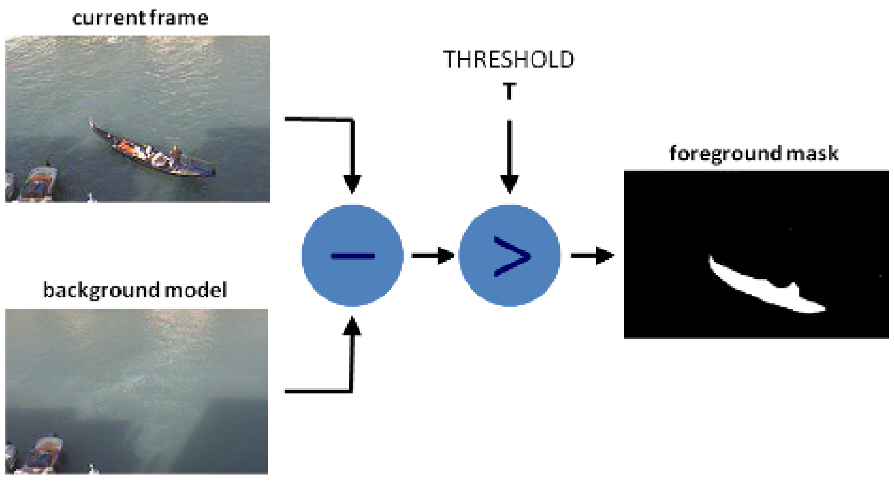


Figure 2.1: A general idea of background substraction. Credit: [25]

from the stored background significantly. Instead of a global threshold, background substraction allows different regions of an camera FOV having different thresholds, which may be useful when there is a known heat source that should be ignored. Though background substraction technique is more commonly used in RGB image data, transferring it to thermal image data also shows impressive results.

A simple implementation of background substraction is comparing the input image with a fix background. Basu and Rowe [14] collect more than 100 sample frames without human being, and use the average temperature as the background reference. Mohammadmoradi et al. [17] measure the background temperature only once when the device starts up, and take the assumption that the thermal background remains stable in a long term afterwards.

A fixed background may give acceptable result when the ambient temperature in the test environment does not vary. Under most circumstances, a background model that could adapt to the varying environment is desired. This could be done by a continuous update of the background pixels. After the fore- and background segmentation of each image frame, those pixels that do not exceed the background temperature will be merged into the background [18, 15], while those pixels that are considered as object do not contribute to the background model. The related formula is shown in Equation 2.3:

$$\begin{aligned} \mu_n[\mathbf{x}] = &\mathbf{1}\left(T_n[\mathbf{x}] \in B\right)\left[\alpha T_n[\mathbf{x}] + (1-\alpha)\mu_{n-1}[\mathbf{x}]\right] \\ &+ \mathbf{1}\left(T_n[\mathbf{x}] \in F\right)\mu_n[\mathbf{x}] \end{aligned} \tag{2.3}$$

where fore- and background are denoted as F and B respectively, $\mathbf{1}(\cdot)$ is the indicator function, and $\alpha \in (0,1)$ is the update rate.

The dynamic background learning method will dissolve an object that does not move for a long time into the background, which may be an advantage and drawback at the same time. When there is an unexpected non-human heat source in the camera view, the unanimated object could be ignored and do not influence detection judgements. On the other hand, a human will also be undesirably ignored if he stays still, which may brings about difficulties to the following algorithms. In order to detect human only, Beltran et al. [13] attached a PIR (passive infrared) sensor to their project in addition to the IR-TAS. PIR sensors are based on a principle of pyroelectric that is different from thermopile and can only detect heat movements. The authors assume that human will not keep still for 15 minutes. The thermal camera will only capture and update the background if the PIR sensor does not sense a movement in such a period.

Finally, some researchers have explored machine learning (ML) methods for human object detection [9, 10]. But these methods will not be elaborated in this section because we want to concentrate on traditional CV methods.

2.2 Tracking Algorithms

After the human objects are detected and located, the task of a tracking algorithm is to assign a consistent label to the same object throughout the video image sequences [26]. Following a bottom-up approach, four questions need to be answered to define an object tracker: what's the suitable (shape) representation of the object, what are the other features (color, texture, etc.) used to represent the object, how is the object detected, and how is it tracked [27]. According to our discussion in the last section, the first and the third questions are already answered. To be more specific, we use pixel active/ inactive states to indicate whether a pixel is occupied by the human, which corresponds to the silhouette representation in Figure 2.2 (i). However, this blob representation could be simplified to other simpler representation such as point representation after a feature extraction procedure, and is not necessary the final input of the tracker.

Mohammadmoradi et al. [17] abstract the detected blob into a single point which has the largest distance to the background, and use this point as human body location. By tracking, a same label is assigned to a new blob if its distance to a previously seen blob is less than 10% of frame width. In addition to the spatial constraint, the authors use a temperature feature to mark a blob, two blobs in sequential frames are considered the same person if their temperature difference is less than 1°C.

Maaspuro [16] also use the blob centroid to represent human body location. Because the original resolution of the IR-TAS they used is too coarse (only 8 by 8), they need

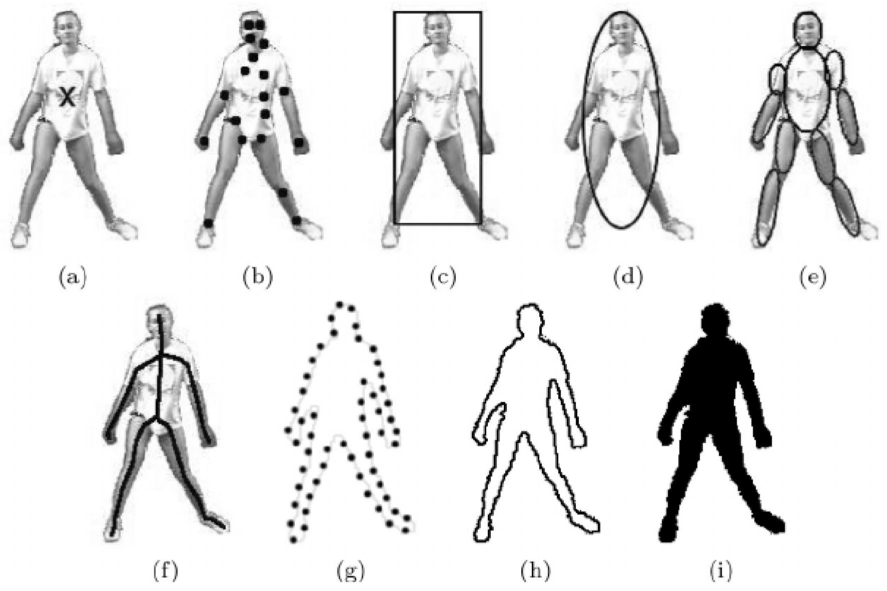


Figure 2.2: Different representations of objects. Credit: [27]

to interpolate the image to 71 $\times$ 71 for tracking). Note that for a camera with 8 $\times$ 8 resolution, a spacial difference threshold of 10% the frame width is less than 1 pixel, which is impossible. Instead of matching the current position of a human body with a previously seen position directly, the authors smoothen the location updating with a Kalman filter [28]. Similar tracking algorithms that based on nearest neighbour association and Kalman filter can also be found in [18, 19].

By multiple blob association in one scene, a common challenge is to track merging/ spliting blobs correctly. When two people occur in the same frame, it is required that their blob representations have at least one pixel distance, otherwise the two blobs will be regarded as one large blob. Due to the low-resolution nature of IR-TAS, a gap of one pixel in the final image typically means some tens of centimeters between two people [16]. This issue is very hard to deal with in the tracking layer, which forbids tracking more than two people (because two people already amount to more than half of the camera view), or tracking any scenarios that two people may stand close, even for a short period of time.

An attempt to solve the issue is proposed by [17]. If one blob is large enough to be two merged blobs (larger than 30% of frame's area), the background temperature threshold will be increased continuously at a step of 0.25°C until a gap between two blobs emerges or the blob size shrinks to less than 10% of frame area.

Qu et al. [19] try to solve the issue in the tracking layer. When two blobs merge,

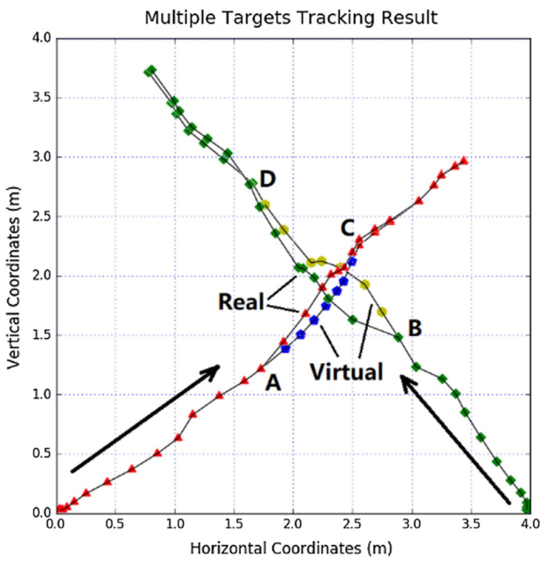


Figure 2.3: Multiple objects tracking result using virtual trajectory. Credit: [19]

virtual trajectories will be created for both object with their previous positions and velocity. The merged human objects will be further propagate with their last seen velocity until two blobs split. When the actual location points for both blobs are available again, the object position will be associated with a object label if it appears on the expected position given by a virtual trajectory. Since the positions in a virtual trajectory may deviate from the real ones largely, the virtual trajectory will be replaced by a smoother back traced trajectory which connects the last actual position before merging and the first position after splitting. Figure 2.3 demonstrates the virtual trajectory tracking process.

Overall, we notice that few related works have solved the aforementioned issue. However, we argue that two people moving in opposite direction through a doorway is a common scenario. Though people tend to keep a social distance of several tens of centimeters in a public space, at a narrow doorway they usually tilt the body to pass simultaneously rather than waiting the other to pass. Inevitably, the result of such a scenario will be a merged blob, which should be taken into consideration by the tracker's initial design.

3 Problem Statement and Hardware Selecting

We want to develop a low-cost, easy to install doorway human counter device for room occupancy estimation. The main component of the device is a low-resolution thermal camera to guarantee privacy protection as well as detection of static human being. Relative counting is done by tracking the human object passing through the doorway and determine whether it is an entering or exiting event. Final occupancy estimation of a room could be obtained by accumulating all relative counts of a device, or several devices, in case of a room with multiple entrances. Moreover, the camera should be installed on the top of a doorframe or on the ceiling of a corridor, facing towards the ground. By this way the size of the monitored person is invariant with respect to the position in the frame, which cuts down the complexity of the tracking procedure.

The captured image frames should be sent to a microprocessor where the detector and tracker algorithms are implemented. When a frame sequence terminates and the relative count is obtained, this count value should be transferred to a data centralization platform for further processing and long term storage. The data transfer should be wireless to ensure easy installation of the device. We want to keep the device design as simple as possible with minimal components. A Wifi-MCU (micro-controller unit) would be an ideal choice because no other radio components are needed.

3.1 Infrared Camera

For the IR-camera we have considered two candidates: the MLX90640 from melexis and the Amg8833 Grideye from Panasonic. The main attributes of both cameras are listed in Table 3.1. Among all the technical features, a sufficiently high frame rate is essential so that the whole traversing event could be captured. When installed at a height of 2 meters, both cameras cover about 2.3m along the moving direction. If we subtract the border where a human is partially seen, the valid length left for tracking is merely about 1.7m. Assuming human walking speed is $1.4m/s$, a normal traverse event will last about 1 second. And at least 3 frames should be captured for a simple single-human event. More frames are required for a complex scenario. Overall a minimal frame rate of 8Hz should be reached.

Table 3.1: Comparison of the MLX90640 and AMG8833

	MLX90640	AMG8833
resolution	32×24	8×8
frame rate	0.5∼64 Hz	1 or 10 Hz
FOV	$55^\circ \times 35^\circ$	$60^\circ \times 60^\circ$
measuring accuracy	$\pm 1^\circ C$	$\pm 2.5^\circ C$
interface	I2C	I2C
cost	€60	€20

The MLX90640 outperforms AMG8833 in accuracy and resolution. Though we are aware that a better camera may simplify the algorithm design drastically, the AMG8833 is chosen because of its low cost.

The hardware limits of the AMG8833 have been discussed by many researchers.

- high sensor noise: the product datasheet states that the worst case temperature difference between two pixels is $5^\circ C$ even if the camera faces towards a homogenous temperature surface [29]. This significant noise level is further confirmed by a experiment [17].
- short detection range: IR waves emitted by human body characterize an amplitude drop as the distance increases. But the reading of AMG8833 drops more faster than other high-end alternatives. At a distance of 120cm, the measurement of a human body is around $18^\circ C$, which is approximate to the room temperature [17].
- radial distortion: the pixel detection areas are not evenly distributed as a 8×8 grid, see Figure 3.1. The barrel distortion could be fixed by applying Brown's lens correction (Equation 3.1), where r_c and r_u are corrected and uncorrected distance to the optical axis. The reported radial coefficients are $K_1 = 7.4 \times 10^{-3}$ and $K_2 = 0.17 \times 10^{-3}$ [12].

$$r_c = r_u + K_1 r_u^3 + K_2 r_u^5 \tag{3.1}$$

3.2 Temperature Sensor

The AMG8833 comes along with a on-board thermistor which measures its working temperature. Trofimova et al. [23] estimate the ambient temperature with the thermistor reading because they have a linear dependence. Nevertheless, an external temperature sensor that reads the accurate room temperature is useful, especially in case the IR module malfunctions.

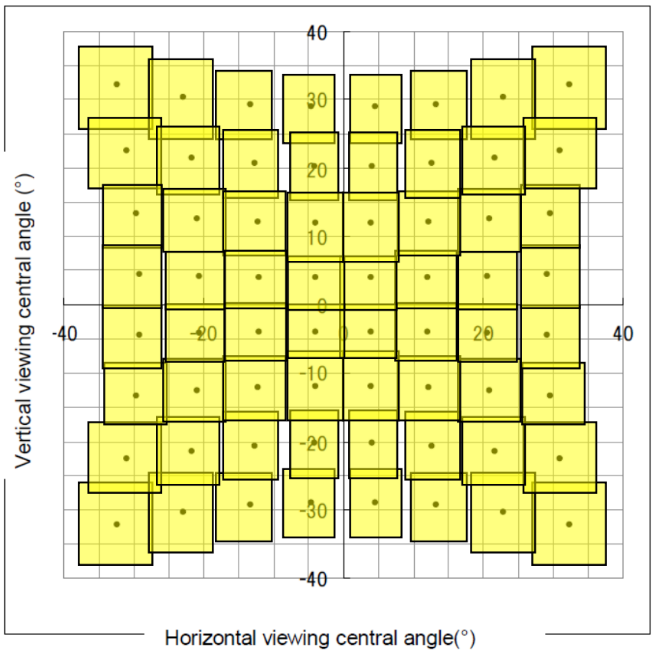


Figure 3.1: Detection area of each pixels of the Grideye sensor. Credit: [29]

We have chosen a DHT11 temperature and humidity sensor [30] for this purpose. It measures the surrounding temperature at 1°C precision.

3.3 Main Processor

The microprocessor should have one I2C interface to retrieve data from the IR camera. Since a new frame must be read every 0.1 second otherwise it will be covered by following frames, a realtime task support is required. The MCU should have a on-board radio component to send out data without an edge device. Besides, for debug purpose we want to store the frame sequences and let the tracker replay them frame by frame, an UART interface is required to simulate the camera frame input.

Our choice for the main processor is a ESP32-WROVER module from Espressif [31]. The specifications are shown in Table 3.2.

The ESP32 board contains not merely a microprocessor, but also a set of out-of-box development firmware such as Wifi and bluetooth protocol, see Figure 3.2. The board could be run as bare metal, however, the ESP-IDF (Espressif Internet-of-things Development Framework) empowers it to run on a adapted version of FreeRTOS [32]. In a RTOS (real-time operating system), each logically independent code snippet could

	ESP32-WROVER	Required
I2C interface	2	1
UART interface	3	1
GPIO pins	>10	1
realtime task	FreeRTOS	yes
radio component	Wifi, bluetooth	yes
flash	4MB	-
RAM	520KB	-

Table 3.2: Specifications of the ESP32-WROVER module

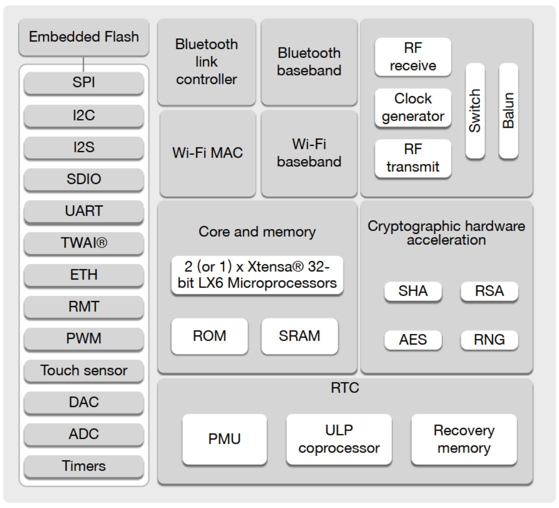


Figure 3.2: Functional block diagram of ESP32 chip

be encapsulated as a task. The RTOS task scheduler divides CPU process time to tiny time slices (1 millisecond), and always assign a time slice to the task with the highest priority in the ready list at that moment. A task will be blocked if it requires a resource that is not available, and it will not participate in task scheduling until the required resource is available again. By this way, a strong realtime response performance is largely guaranteed, unless the processor's maximum computation power is exceeded.

The ESP32-WROVER module features a dual-core architecture. It contains two Xtensa CPU [33] with a frequency up to 240MHz. Wireless communication through Wifi or BLE is inherently handled by the first core, while user applications could be attached to either core. The powerful computational capacity yet a relative cost (€10) made it our first choice for the main processor.

The total cost of our design is €35, including an AMG8833 Grideye (€20), an ESP32 board (€10) and a DHT11 temperature sensor (€5).

4 Data Transmitting and Platforms

The statistic data collected by a doorway counter is not of much use until it is analyzed. For a quick acquisition of data and a frequently updated overview of room occupancy, the count value should be sent to a remote platform at every time for further visualization and analyse.

4.1 MQTT protocol

We use MQTT (message queuing telemetry transport) to transfer data because it is supported by the ESP-IDF and widely accepted in IoT use cases. MQTT is an application layer protocol built on top of Wifi and TCP/IP. It is a light-weight, publish-subscribe network protocol, its structure is shown in Figure 4.1. When a publisher publishes, instead of sending to the receiver directly, the message is sent to a intermediate server called broker. Each message is published to a specific topic, a client would filter the messages by the subscribed topic and could only receive the message if it is listening to the same topic.

The light-weight usage of MQTT is mainly reflected by its transparent topic mechanism. A client does not need to create a topic before it publish or subscribe to it. A topic keeps alive when there is at least one publisher or subscriber. When a topic has at least one client on both pub/sub side, the data transmission becomes valid almost immediately. The topics in MQTT decouple the rigid connection between clients, reducing the complexity of connection drastically regardless of the scale (up to tens of thousands per MQTT server).

The main drawback of MQTT is no message buffering. A QoS (Quality of Service) setting higher than 0 on the client side only confirms that the message is received by the broker. If a message is published to a topic without listeners, that message is lost forever. Latest versions of MQTT supports retained messages, which stores one *last known good value* in the broker. However, the retained messages aim to serve as a *initialization default value* for those subscriber that join the topic between two publish intervals, storing more than one message is impossible. Secondly, MQTT is a queuing system instead of streaming, it delivers messages to a single consumer. When there are multiple consumers subscribing to a topic, they will consume the messages in a round-robin manner.

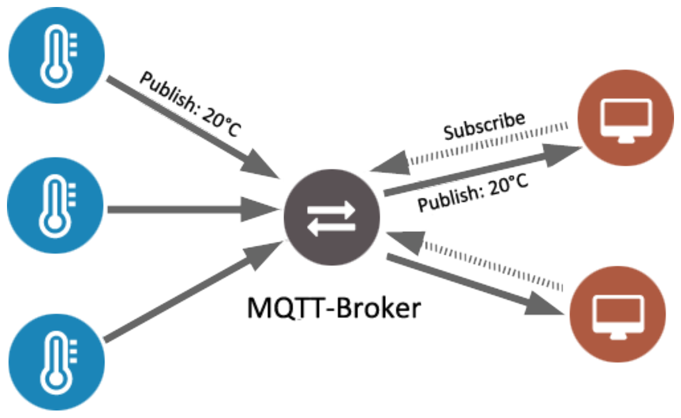


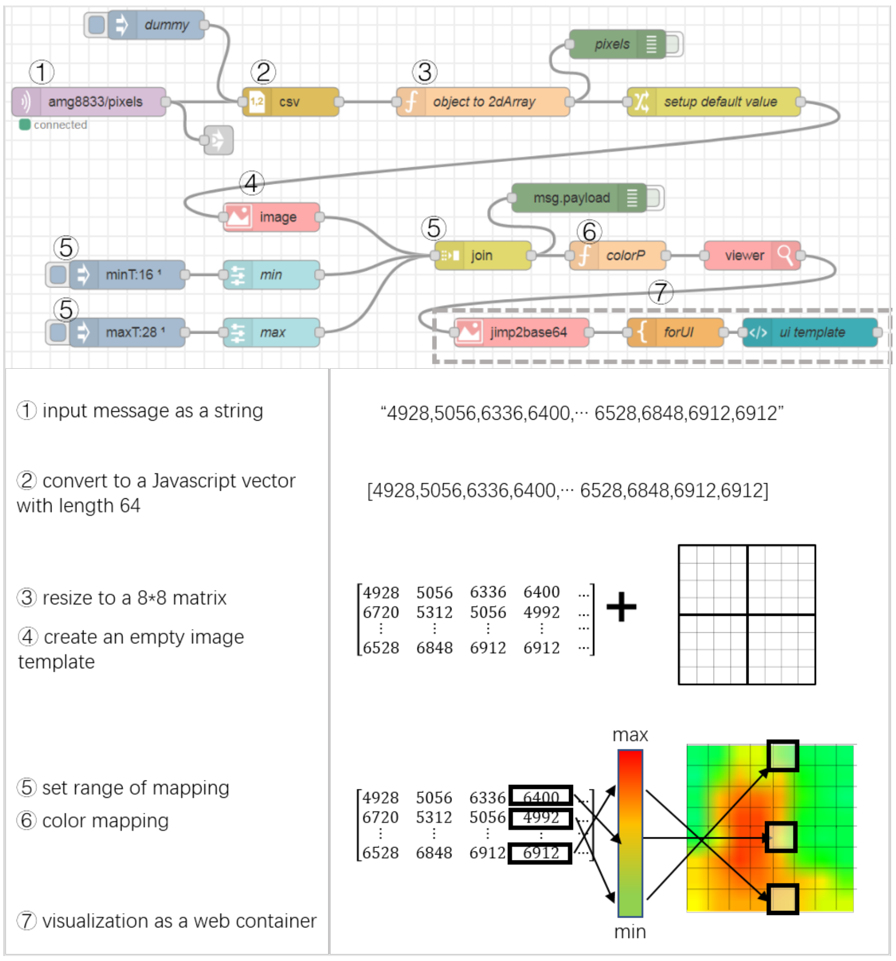
Figure 4.1: MQTT network structure

4.2 Node-Red

Visualization of an image frame sequence on websites often requires extensive programming. Fortunately, Node-Red [34] relieves us from irrelevant affairs so that we can concentrate on the doorway counter development. Node-Red is a web-based visual programming editor that connects IoT devices or acts as an end-consumer. It supports message input from a MQTT broker, as well as http, websocket and serial. The smallest execution unit is called a node. A complete data process procedure consists of multiple nodes and is called a "flow", an example is shown in Figure 4.2. Messages are propagated from flow start to flow end, and could be modified, truncated, merged with other flows, or splitted.

Node-Red has integrated a dashboard which could be used for data visualization. Data in simple format like an integer number or a string, could be visualized with offered presets with no effort. Complicated data like images could be visualized with custom javascript scripts and Angular framework. Figure 4.3 shows the dashboard used in our project. The dashboard contains three subsections: region 1 on top-left shows the actual camera view, grayscale pixel values of the IR camera are converted to a heat map, where a redder pixels represents a hotter temperature; region 2 shows the last updated room temperature given by the temperature sensor, which is used for camera calibration, and this value could also be manually assigned; region 3 shows the process time of a single frame and the actual count number, the former is crucial to check whether our algorithm could meet the frame rate constraint, the latter is useful to get an immediate feedback whether the algorithm works properly when we witness an traverse event in region 1.

Besides, Node-Red has a "file" node which terminates the flow and stores messages



$$\begin{bmatrix} 4928 & 5056 & 6336 & 6400 & \cdots \\ 6720 & 5312 & 5056 & 4992 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \cdots \\ 6528 & 6848 & 6912 & 6912 & \cdots \end{bmatrix} +$$

$$\begin{bmatrix} 4928 & 5056 & 6336 & \boxed{6400} & \cdots \\ 6720 & 5312 & 5056 & \boxed{4992} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \cdots \\ 6528 & 6848 & 6912 & \boxed{6912} & \cdots \end{bmatrix}$$

Figure 4.2: Our Node-Red flow used for 8 × 8 pixels image visualization

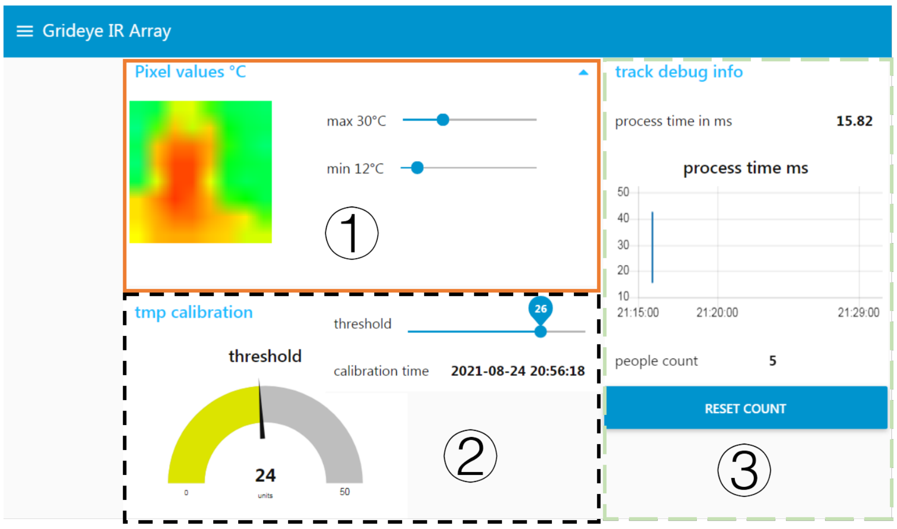


Figure 4.3: Our NodeRed visualization panel is divided into three sections: (1)raw image frame, (2)temperature sensor reading, (3)debug info

on local storage. We store image files by this way to bypass the MQTT's buffering issue. We don't store images in a data bank because the generated image file size is relative large, about 70Mb every day.

4.3 ElasticSearch and Kibana

To evaluate whether our algorithm outputs the expected relative count, a long-term storage of the count value is needed. For this purpose a data bank based on ElasticSearch and Kibana is used. In ElasticSearch, an individual message is called a "document", and a series documents that are similar or generated by a same source are gathered in a group called "index". Kibana is a visualization tool combined with ElasticSeearch that can visualize data by indexes. Developed by TUM researchers, the data bank we used has an MQTT interface which receives time-series data, and the accepted message format is one float number and a timestamp.

In our use case, whenever an entering/ exiting event is detected, a pair of messages, the actual timestamp and the count value at that time will be published and recorded in the data bank. Figure 4.4 shows the recorded data in one afternoon, where the green line is the output of the counting algorithm and the red dots are the ground truth, annotated manually by watching the stored image sequence. Details about ground truth annotation and the evaluation result will be elaborated in chapter 6.

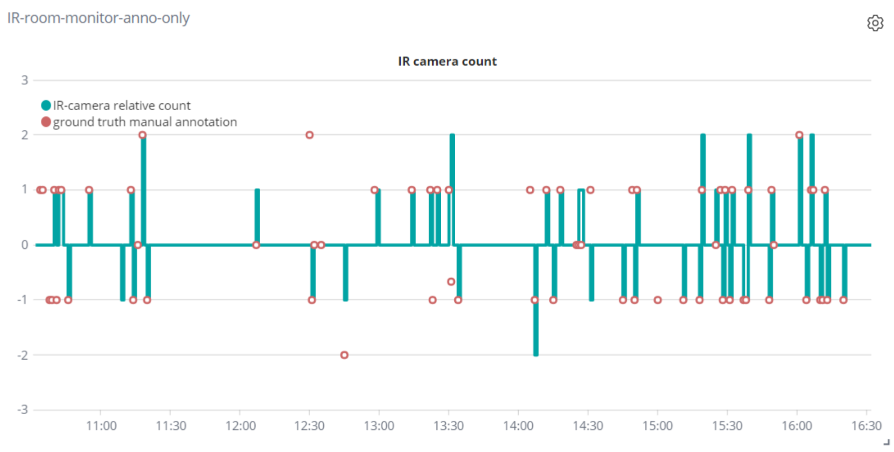


Figure 4.4: A fraction of the recorded data. Green line shows the output of the algorithm and red dots are the ground truth

5 Proposed Algorithm

The objective of the human counting algorithm is clarified as following.

The detector should be capable of detecting human objects in a relative large range of temperature, because the detected temperature of human body depends heavily on the wearing clothes and individual body condition. The detector should be sensitive to detect a human that is slightly hotter than ambient environment. Meanwhile, fluctuation of the environment temperature should be taken as a reference carefully. A pixel value that is just above the ambient temperature in a cool morning would be regarded as active, but a same value would probably be a noise reading in the afternoon of the same day.

The human body tracker should work properly in single human events, as well as more complicated events when multiple people interact with the door, including but not limited to: two people traversing parallel, traversing sequential but with a narrow distance, traversing in opposite direction, or one person standing still and another pass by. The tracker should not assume that the detected pixel blobs are separated perfectly, and should be able to track two once separate blobs when they merge, as well as assign the correct labels when they split.

And finally, the time consumption of one single image processing must be constraint within 100ms because the camera runs at 10fps. Considering that other tasks, such a data reading from camera register and publishing MQTT messages, also takes CPU process time, the time window left for detecting and tracking is merely around 50ms.

5.1 Blob Detection

5.1.1 active pixel detection

Before the actual fore- and background segmentation, an early detection process is conducted on the 8×8 coarse image to see whether there is a potential heat source in the camera view. We check if there are at least three pixels having a value $1.5°C$ higher than the room temperature. The parameters are chosen based on the following calculation. When installed at a height of $2.5m$, the Grideye sensor covers an area of about $8m^2$, which is much larger than a human body. However, when a human enters the camera view, the projection of his body on the image is caused by the horizontal

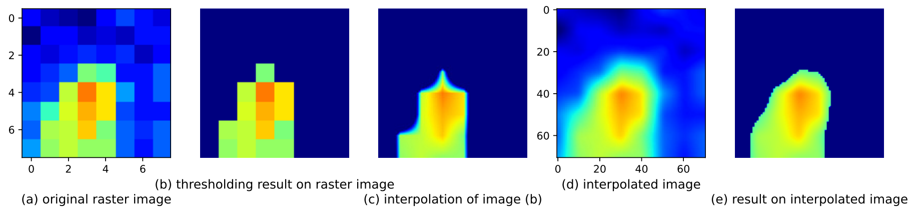


Figure 5.1: Result given by Equation 5.1, the interpolated image preserves more details after thresholding.

plane of the shoulders, which is much higher than the ground and closer to the camera. Assuming the average shoulder height is $1.4m$, the real coverage of the camera is $4m^2$, only half of the ground area that it can cover. We assume the horizontal plane of a human body is no less than $2dm^2$, which amounts to at least three pixels in the image frame. The threshold $1.5°C$ above the room temperature is chosen to filter out the sensor noise, we denote this threshold as T_{lb} (low boundary).

5.1.2 the baseline formula

To separate human objects from the background, we follow the idea of calculating a global threshold from image statistic values, which is inspired by [19] and [11]. The threshold T_{th} is calculated by the following formula (Equation 5.1), where Max is the maximum pixel value among all 64 pixels and SD is the standard deviation. By thresholding, the original raster image is linearly interpolated to a resolution of 71×71 by inserting nine pixels between each pixel to avoid numeric issues in the following segmentation. The same resolution is also used by [19] and [16].

$$T_{th} = max\{Max - 4 \times SD, T_{lb}\} \tag{5.1}$$

The formula is simple but effective. The result is shown in Figure 5.1. Comparing image (c) and (e), it is apparent that the interpolation step is necessary to preserve more information from the original image and obtain a smooth shape of the object.

5.1.3 a spacial filter

We notice that a single global threshold cannot separate two close objects very well. When two human stand close to each other, the small region between them will have a temperature higher than the room temperature because the pixel value results from a

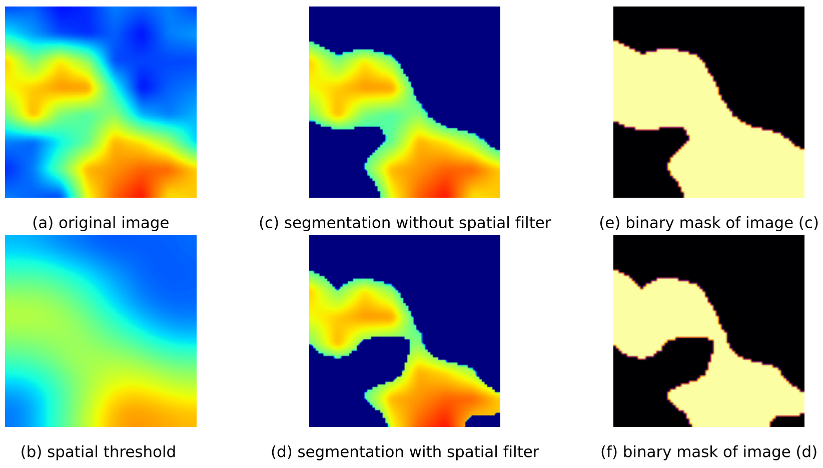


Figure 5.2: After introducing a spatial filter, the region between two objects are no longer marked as occupied.

size-weighted average of both objects and background. This effect creates fake active pixels in the binary mask, see the first row in Figure 5.2.

Therefore, pixels higher than the threshold but with hotter pixels nearby should be ignored because they origins from those hotter pixels and do not represent the true heat distribution. Reversely speaking, only those pixels that are hotter than its surrounding, namely local maxima pixels, contain valid information. We use the average filtered image as a spatial filter. After experiments on the collected image frames, a kernel size of 36 is chosen (half of the frame width). The value of every pixel in the filter is calculated by averaging a 36×36 neighbourhood surrounding that pixel location in the original image. Afterwards, the original image is compared with the filter pixel-wise, and only those pixels with a value higher than its filter counterpart will be kept. Figure 5.3 shows the process in a surface plot, it shows the same image frame from Figure 5.2. The threshold calculated from Equation 5.1 is applied to the local maxima image. The result is shown in the second row of Figure 5.2, it is clear that the bridge connecting two objects becomes much narrower, which is beneficial for recognition that there are two objects in the frame, instead of one single large object. Moreover, it is not necessary to eliminate the bridge completely as we have a feature extraction step, which will be discussed in the next section.

Moreover, this method outperforms simply increasing the global threshold because

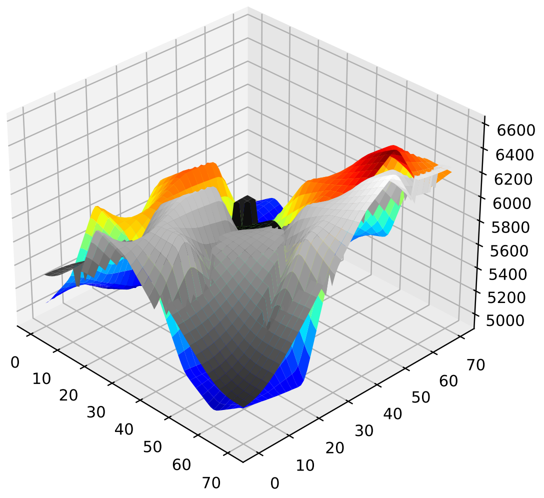


Figure 5.3: Only those pixels hotter than region average is kept. The gray layer is the average filtered image and the colored layer is the original image.

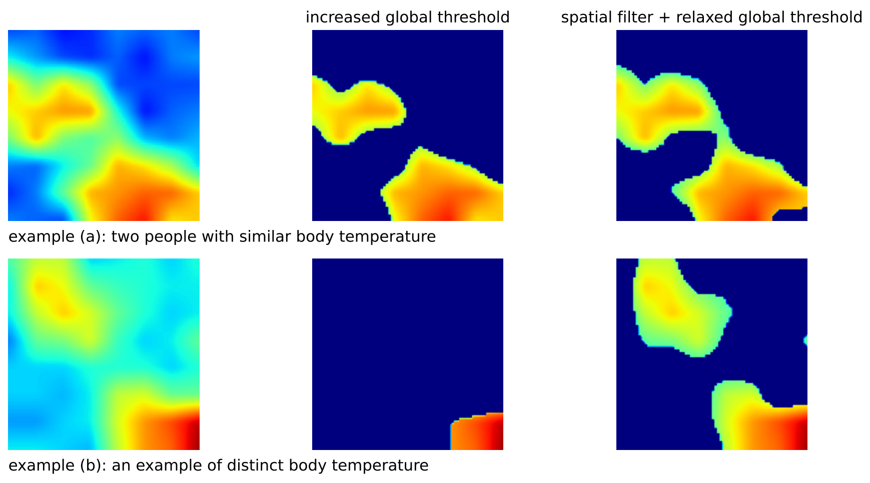
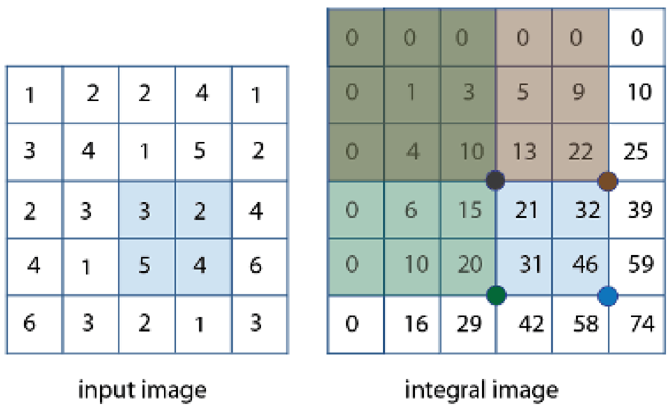


Figure 5.4: The middle column shows the segmentation result by a global threshold of $Max - 1.5 \times SD$, the right column shows the result by the proposed method. Both objects are detected whether they have distinct body temperature.

it has a better generalization ability for different body temperature. Though the aforementioned pixel bridge could also be eliminated by increasing the global threshold, we find that the threshold must be set very high to obtain a similar result. Figure 5.4 shows the result of both method in two real scenarios. The first row shows the same frame in Figure 5.2, where a tight global threshold of $Max - 1.5 \times SD$ turns out to be even better than our proposed method. However, when two human with distinct temperature occur in the same frame, the hotter object raises the global threshold with a higher Max pixel value. The threshold will be too high for the cooler object, and only the hotter object is detected.

By implementation, a convolution kernel with size 36×36 is too slow to meet the demand of realtime performance. For a $M \times N$ image and a $K \times K$ kernel, the time complexity of average polling is $M \cdot N \cdot K^2$. Though the time complexity could be reduce to $M \cdot N \cdot K$ by separating the 2-dimensional kernel to two 36×1 1-dimensional kernels, the time consumption is still not satisfying. Our experiment shows that on a ESP32 processor with 160MHz frequency, it takes seconds to convolve with a 2d kernel and about 70 milliseconds to convolve with two 1d kernels. We boost the processing performance by replacing the average polling algorithm with a summed-area table [35], which reduce the time complexity to $M \cdot N$.



1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

0	0	0	0	0	0
0	1	3	5	9	10
0	4	10	13	22	25
0	6	15	21	32	39
0	10	20	31	46	59
0	16	29	42	58	74

Figure 5.5: The basic idea of a summed-area table. Credit: [36]

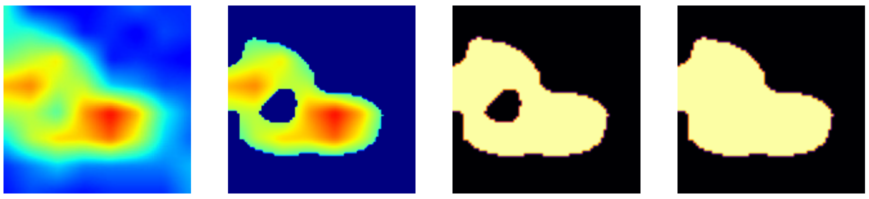
A summed-area table, or sometimes called an integral image, is an image with the same size of the original image, where every pixel's value is the summation of all the pixels on the top-left of it. A summed-area table could be calculated efficiently from the original image in a recursive way, because the value of each pixel only depends on three pixels above, on the left and on the left-top of it. To calculate the average value of a sub-region in the original image, only four corner points of that region in the summed-area table are needed. Figure 5.5 demonstrates how a summed-area table is calculated and used.

For comparison, the summed-area table based average filter takes only around 2ms for a single frame with the same processor configuration.

5.1.4 filling holes

By the segmentation, another common issue is caused by the hair isolation of the head. From a top view, the human body is typically represented as a ellipse containing the head, two shoulders and partial upper torso. Ideally, the blob representation should have the highest temperature at the center and have a descent gradient to the margin. Unfortunately, a lower temperature is often observed in the blob center because a person's hair decays the heat emissivity. This will create a hole in the final binary blob and cause confusion, see Figure 5.6.

Since our detection method does not include any prior knowledge about the blob shape, any pixel that is cooler than the threshold will be regarded as background. There is no way to distinguish whether a pixel is truly a background pixel or is occupied but isolated. Moreover, because the hair is closer to the camera and has a large proportion in the blob presentation, the isolated region could be very large and even deteriorate


(a) hair isolation creates a hole in the blob center

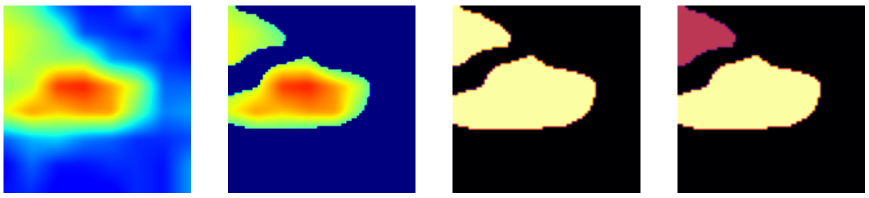

(b) hair isolation tears the blob into two parts

Figure 5.6: Two incorrect detection caused by hair isolation. The former could be fixed in detection layer, but the latter is left for tracking layer.

the detection result.

If the hair isolation only creates a hole and the binary blob is still a connected domain, the hole could be filled with flood-fill algorithm. Starting from the top-left pixel of the image, every connected background pixel is marked as occupied until there is no connected background pixel anymore. When there is a hole in the blob, the hole pixels are the only background left because they are surrounded by occupied pixels and are not reachable from a border background pixel. Then the processed binary mask will be logically reversed and merged with the original image. The final result is a blob representation without holes.

Nevertheless, if the hair isolation region is too large, the original blob will be separate to two parts, see the second row in Figure 5.6. In this situation, even a human observer cannot distinguish whether it is a large person or two standing close smaller people by watching a single frame. A conclusion can be drawn by combination of previous frames. We decide not to solve this issue in the detection layer and leave it for the tracking layer.

5.2 Feature Extraction

The binary blob mask of an object contains too much redundant information for the tracking algorithm. As reviewed in chapter 2, a binary blob could be represented as a feature vector of the centroid and the size, denoted as Equation 5.2, where C is the 2D coordinate of the geometry center and S is the total pixel number of the blob.

$$B = \{C, S\} \tag{5.2}$$

This basic representation is sufficient for a single object tracking because the centroid is a unique feature for a blob. However, when two objects overlap and their blob masks merge together, they will be regarded as one single blob and therefore have only one shared centroid. The uniqueness of the centroid no longer holds, and the one-to-one matching relation between an object and its corresponding blob mask is broken. Figure 5.7 shows an example demonstrating how the merged blobs impede the feature extraction. In the first two frames where two blobs are not merged, their centroids remain unique and transit a small shift from the last frame. The size of the blobs remain similar as the last frame as well. In the third frame, the blobs are merged, their centroids disappear and is replaced with a single centroid at the center of two blobs. Viewing the extracted feature only, we may only conclude that there is one blob with twice the size as before. Furthermore, the location of the centroid jumps abruptly, not close to either centroids in the last frame. In the fifth frame, the blobs are again separated. An abrupt change in centroid position takes place again.

Since the tracker match objects with its nearest neighbour in the last frame, a distance that is too long will break the matching. Even if the blobs are successfully matched, a shared centroid will bring ambiguity to the label assignment. Labels may be exchanged between two objects after splitting.

We follow the research of [37], adding “central points” as addition feature for blob representation. Central points of a blob are those points that are far away from the background, they can be found out by picking local maxima in a distance transform of the blob mask.

5.2.1 introduction of distance transform

A distance transform of a binary image labels every pixel of the image by the distance between that pixel and the nearest background pixel [38], which could be denoted as Equation 5.3.

$$I_d(x,y) = \begin{cases} 0 & I(x,y) \in \{Bg\} \\ \min\left(\|x - x_0, y - y_0\|\right), \forall I(x_0,y_0) \in Bg & I(x,y) \in \{Ob\} \end{cases} \tag{5.3}$$

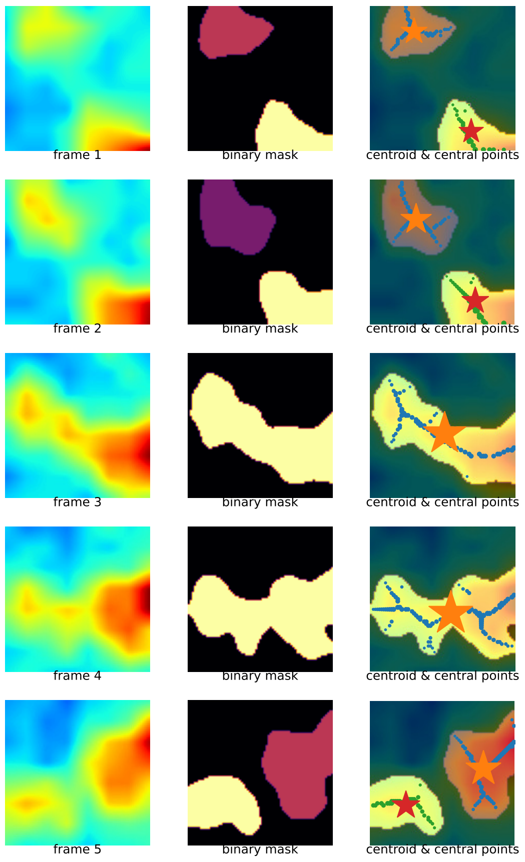

frame 1
binary mask
centroid & central points
frame 2
binary mask
centroid & central points
frame 3
binary mask
centroid & central points
frame 4
binary mask
centroid & central points
frame 5
binary mask
centroid & central points

Figure 5.7: An example frame sequence showing the blob merging issue. Left column is the interpolated image, middle column is the blob masks and right column shows the extracted feature. The star symbol represents the centroid location and the size of the start represents the size of the blob. Other round dots scattered inside the blob represent central points.

The $\|\cdot\|$ symbol is a distance definition, could be L_1, L_2, L_{max} or any other distance metric. To our concern, the distance is measured in Euclidean metric, namely $\|x, y\| = \sqrt{x^2 + y^2}$.

A precise computation of the Euclidean distance transform is complex. A commonly used approximation is the chamfer distance transform [39]. The chamfer distance transform exploits the fact that the distance between of each pixel to its nearest background pixel depends on the previously computed neighbour. The whole distance transform image could be obtained in two pass, one propagating from the upper-left corner to the lower-right corner and the other reversely. Mathematically, the two-pass algorithm with a 3×3 window could be written as the following Equation 5.4, where r, c denote the row and column coordinate of a pixel respectively, x_- is the distance transform value after the first pass and x_+ is the final value after the second pass.

At the first pass, a pixel's distance to the background is propagated from three pixels above it and the one on the left of it. Two vertical and horizontal connected pixel contribute to the transform value by an distance increment of a, while two diagonal connected pixel contribute and increment by b. The value a and b depend on the desired approximation and usually $a \leq b$. The transform result of the first pass is the minimal value among four candidates. After the first pass, those pixels at the bottom or at the lower-right corner will have the largest distance value (to the upper-right background pixel). The second pass is analogue to the first pass. Except for four neighbour pixel below and on the right, the previous result from the first pass becomes the fifth candidate. At the beginning of the second pass, the distance transform of those pixels at the bottom and lower-right corner will be iteratively replaced by values propagated from their neighbourhood because they are closer to the lower-right background pixel. After passing the object's geometry center, the results from the first pass will dominate the final result.

There are various choices for value a and b. We use the taxicab distance $a = 1$ and $b = 2$. The result of distance transform is shown in Figure 5.8.

$$x_-(r,c) = \min \left\{ \begin{pmatrix} x_-(r-1,c-1) & x_-(r-1,c) & x_-(r-1,c+1) \\ x_-(r,c-1) & & \end{pmatrix} + \begin{pmatrix} b & a & b \\ a & - & - \\ - & - & - \end{pmatrix} \right\} \tag{5.4}$$

$$x_+(r,c) = \min \left\{ \begin{pmatrix} & x_-(r,c) & x_+(r,c+1) \\ x_+(r+1,c-1) & x_+(r+1,c) & x_+(r+1,c+1) \end{pmatrix} + \begin{pmatrix} - & - & - \\ - & 0 & a \\ b & a & b \end{pmatrix} \right\} \tag{5.5}$$

5.2.2 local maxima in a distance transform

By the definition of distance transform, a distance transform of a binary blob mask is essentially a local shape descriptor. The distance value of a pixel only depends on the size of its *local convex region*. Another concatenated blob has little influence to the distance transform result except the overlapped pixels.

However, not all of the distance values are equally interesting. Ideally, only two points representing the location of the merged blobs are needed to track two merged blobs. One way to find local region centers is filter the distance values by whether they are close to the maximum distance value in the distance transform. The fourth column in Figure 5.8 shows the result. This method assumes that the size of two merged blobs are similar, otherwise most of the detected points will located within the larger object or the small object is eventually not detected at all. The advantage of this method is that all the detected points do locate in the blob center.

Sharma [37] determine the central points by applying a 3×3 filter to extract all the local maxima, and filtering out the local maxima that is smaller than 50% of the maximum distance value. However, their work focus on side-view RGB images and cannot be directly transferred to our use case.

We extract the local maxima by an edge detector inspired by Laplacian filter. A Laplacian filter is commonly used in grayscale images to find out the points where pixel intensity changes fiercely. In the discrete world, a Laplacian filter could be a 3×3 convolution kernel written as the following Equation 5.6. Intuitively speaking, a Laplacian filter compares the intensity of a pixel with its four neighbours, multiplying the intensity of the center pixel by 4 and subtracting the intensity of surrounding pixels. In a homogenous region when all five pixels have the same intensity, the computation result would be zero. At the border of a image, half of the kernel locate inside the object, having a brighter pixel, and half of the kernel fall into the background (darker therefore the pixel value is lower). The result of Equation 5.6 would be greater than zero and that pixel would be marked as a boarder pixel.

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{5.6}$$

Similarly, the second column in Figure 5.8 shows that the local maxima of our top-view images looks like a "border" that separates the entire blob into two symmetric parts. Continue using the idea of a edge detector, we propose the filter shown in

Equation 5.7 and the results are shown in the third column of Figure 5.8.

$$\begin{pmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{pmatrix} \tag{5.7}$$

This method ensures that every convex part of the blob obtains a point representation, so it will not miss out any part of a blob. However, many redundant points are also included in the final blob representation, which may cause a match failure in the tracking stage. Imagine the following common scenario: when a blob moves at a constant speed and several frames are captured. In the second frame, comparing with the true geometry center, the local maxima points near the blob edge will always be closer to the blob's previous location. Therefore the matched object location is lagged behind its actual position. Then in the third frame, the matching is made between a previous boarder point and a current border point. If the human moves a little bit faster, the distance between the previous border point and any current point would exceed the nearest neighbour threshold. A matching is lost.

Considering the robust point representation it can offer, we still choose this method for feature extraction, and solve the possible matching failure in the tracking layer.

5.2.3 central points - an informative and stable representation

Central points preserve most information about the original blob. If the distance value to the background of every central point is preserved, the original blob mask could be approximately recovered from the central points, see Figure 5.9. In contrary, the shape information is lost when using the centroid representation. Certainly, multiple central points requires more memory space than a single centroid, but the central points representation compresses information efficiently. Based on our observation, the ratio of cental points and blob size is around 4% $\sim$ 6%: a blob with size 1200 has around 70 central points and a blob with size 500 has around 30 pixels. Therefore, we have reduced the memory usage for every blob by 95% while still kept most information.

The above analyse shows that the central point representation is a stand-alone feature describer, and is independent of the centroid representation. However, because we cannot reduce the number of central points to a low level, the tracking algorithm must find the optimum candidate among all central points, which brings about extra computational cost and waste when blobs are not merged. Therefore, we describe the blobs by two representations (see Equation 5.8, P is the coordinate of a central point and D is its distance to the background). And would only switch to central points matching if the centroid matching fails. Details about matching would be elaborated in

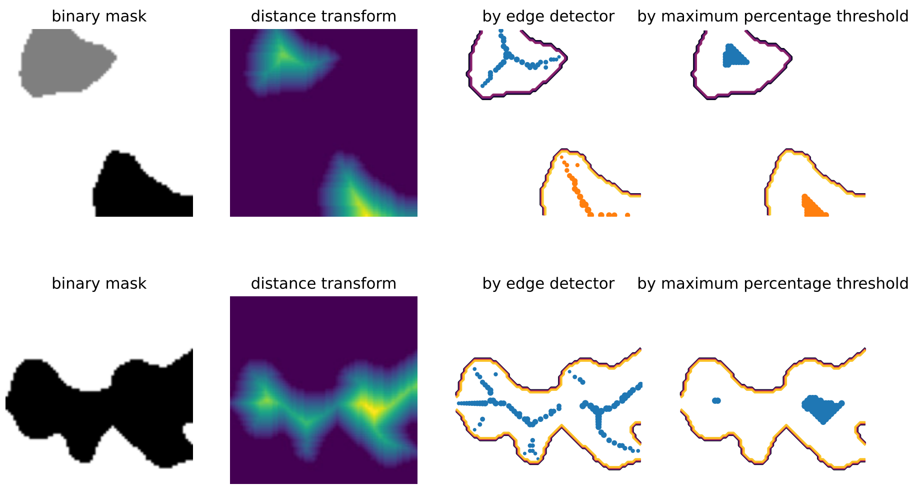


Figure 5.8: The first two columns show the original mask and its distance transform result. In the second column, a brighter pixel indicates a higher distance to the background. The third column shows the central points extracted by a edge detector. The fourth column shows central points found by a percentage filter ($> 80\%$) of the highest distance value.

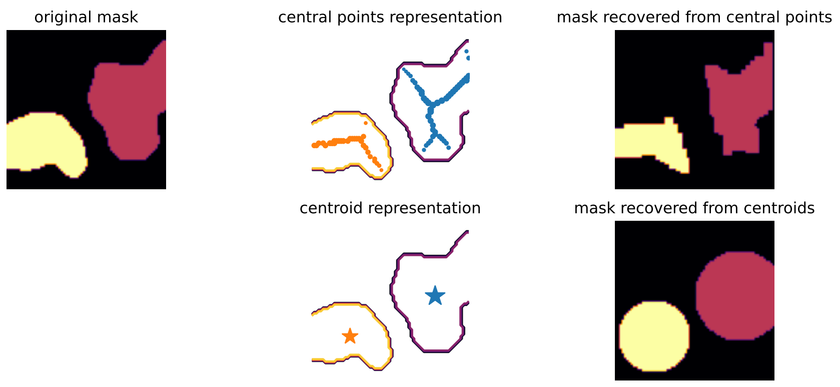

original mask
central points representation
mask recovered from central points
centroid representation
mask recovered from centroids

Figure 5.9: Mask recovery from the central points shows most of the shape information is preserved

section 5.3.

$$\begin{aligned} B =& \{C, S\}; \\ & \{(P_1, D_1), (P_2, D_2), \cdots, (P_n, D_n)\} \end{aligned} \tag{5.8}$$

The cluster of central points looks like the "spine" of the original blob. This is not surprising because we visually examine the shape of the local maxima points in Figure 5.8. Actually, the set of local maxima points in a distance transform are an approximation to the medial axis of a binary mask [40], see Figure 5.10. A medial axis of a R^2 shape is the set of points that have more than one closest point on the object's contour line. Attali et al. [41] have proved that for shape in R^2, fluctuation of the shape contour only effect part of the branches of the medial axis. This partially explains why the central points stay stable by blob merging/ splitting.

5.3 Human Object Tracking

After abstracting the blob masks into feature vector representation shown in Equation 5.8, the tracking algorithm finally comes into use. To handle the one-to-many relation between blobs and actual objects, we introduce an abstract layer called object layer above the observed blob layer. The tracking algorithm maintains a list of known objects and matches an object with its optimum pair in the blob list given by the feature extractor. Every object consists of the attributes listed in Table 5.1.

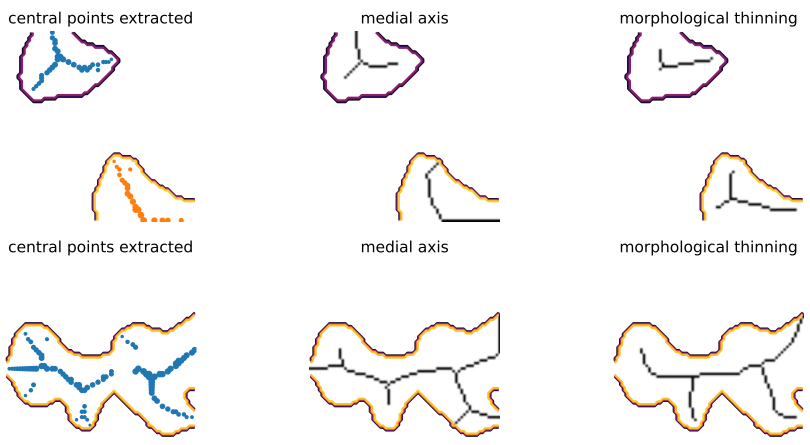


Figure 5.10: The first column shows the set of central points, the second column is the medial axis of the same binary mask. The third column shows result of another thinning algorithm by [42].

Overall speaking, the objective of the tracking layer is to:

1. track the blobs by their centroid as much as possible
2. if step 1 fails, try matching existing object with one of the central points
3. in step 2, choose the most likely candidate for matching among all central points, considering the previous object location, motion direction as well as speed, as mentioned in section 5.2
4. determine if two blobs actually belong to one previous object, as mentioned in section 5.1
5. determine if it is an entering/ exiting event by an object's first position and last seen position, as well as modify the room occupancy count

The above listed tracking steps are adapted from [37] for a top-view IR image sequence, and could be further divided to two phases.

In the pairing phase, the location of an object will be updated to the paired blob location with a filter. If an object cannot find a suitable blob pair, it will not be eliminated

i	unique label
$\vec{x}$	current position
$\dot{\vec{x}}$	current velocity
$\vec{x}_0$	position of first appearance
t_v	number of virtual track propagation

Table 5.1: Object attributes for tracking

immediately but continue propagated a few frames with the previous velocity to see if there could be a match on the forward trajectory. This is exactly the virtual track method used by [19].

If there is at least one unmatched blob left in the blob list, the spawn phase is activated. The tracking algorithm first check if the matched blob could be a fraction of a splitted blob. If it is the case, the blob will be merged to the nearby existing object and contribute to the object location update. Finally, if the blob is unlikely to have any relation with existing objects, a new object will be spawned from the blob center.

5.3.1 Matching step

The objects and blobs are matched by a distance score based on nearest neighbour rule. By matching, a time disparity between the objects and blobs should be considered. Because the blobs are observed at a time step t, but the object location are updated in the last time step $t - 1$. By the time of matching, the objects should have already moved forward. Therefore, the blobs should be matched with the *predicted position* of the objects.

A Kalman filter could be used to store the object states and predict its next state by state transition [16, 19]. Since the states in our use case only contains two components (position and velocity), and the second one is the derivative of the first one, an Alpha-Beta filter could be used to simplify the filtering process and cutdown unnecessary computational overhead.

An Alpha-Beta filter resembles Kalman filter in its nature, combining state transition and observation for state update, but does not vary coefficients dynamically. Denote $\hat{x}$ as the estimated value, x^p as the predicted value and x^m as the measured value, the system state is obtained by the following Equation 5.9.

$$\begin{cases} \hat{x}_n = x_n{}^p + \alpha \left(x_n{}^m - x_n{}^p \right) \\ \hat{\dot{x}}_n = \dot{x}_n^p + \dfrac{\beta}{T} \left(x_n{}^m - x_n{}^p \right) \\ x_{n+1}{}^p = \hat{x}_n + T\hat{\dot{x}}_n \\ \dot{x}_{n+1}^p = \dot{x}_n \end{cases} \tag{5.9}$$

In the first two lines of Equation 5.9, α and β are two fixed coefficients that are chosen experimentally. For an camera installed at a height of 2.5m with a FOV of $60°C$, we choose $\alpha = 0.75$, $\beta = 0.8$. Figure 5.11 demonstrates the smoothen effect of the filter, the estimated trajectory becomes more stable compared to the measured trajectory. More importantly, even though the filter uses fixed coefficients, the prediction usually approximate the actual measurement very well. An exception is when the human moves close to the border, because the human body is only partially detected, the centroid of that partial body stays almost still. The estimated object position will converge quickly to the actual (but wrong) measurement. This behaviour is designed intended so that the tracking could be continued.

During the middle stem of the trajectory when the human is fully observed, the predicted position given by the filter largely simplifies the tracking algorithm. The distance threshold between a blob observation and an object no longer depends on the object velocity, but merely on the prediction deviation and measurement error. Therefore, the distance threshold could be reduced to a low value, which potentially avoids false matching because of a large searching radius.

Moreover, an (relatively) accurate prediction could solve the nearest neighbour issue during central point matching mentioned in section 5.2. By central points extraction, several points on the child branch would be inevitably picked to be possible candidates, while the actual blob central must locate on the main branch. Since those points on the child branch are closer to the border therefore closer to the last estimated object location, they will have a higher chance in paring if we directly match the last estimated object location with the measurement. However, with one time step forward prediction, the predicted object location will locate approximately on the main branch of the central point representation, and the tracking algorithm will choose a candidate that is closer to the actual object position.

5.3.2 Virtual track

5.3.3 Blob registration

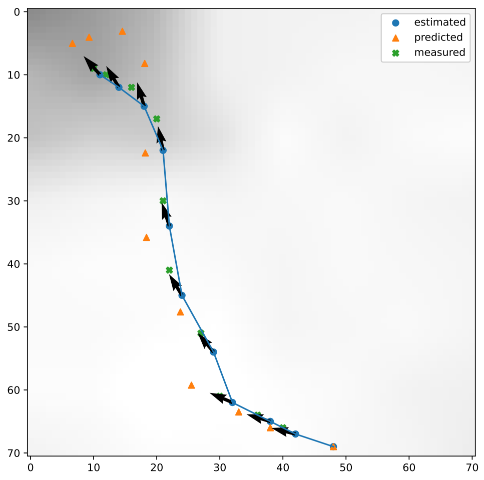


Figure 5.11: A track example. The blue dot shows the estimated position, the quiver from the blue dot points the velocity direction of the object and the yellow triangle shows the predicted position one time step forward.

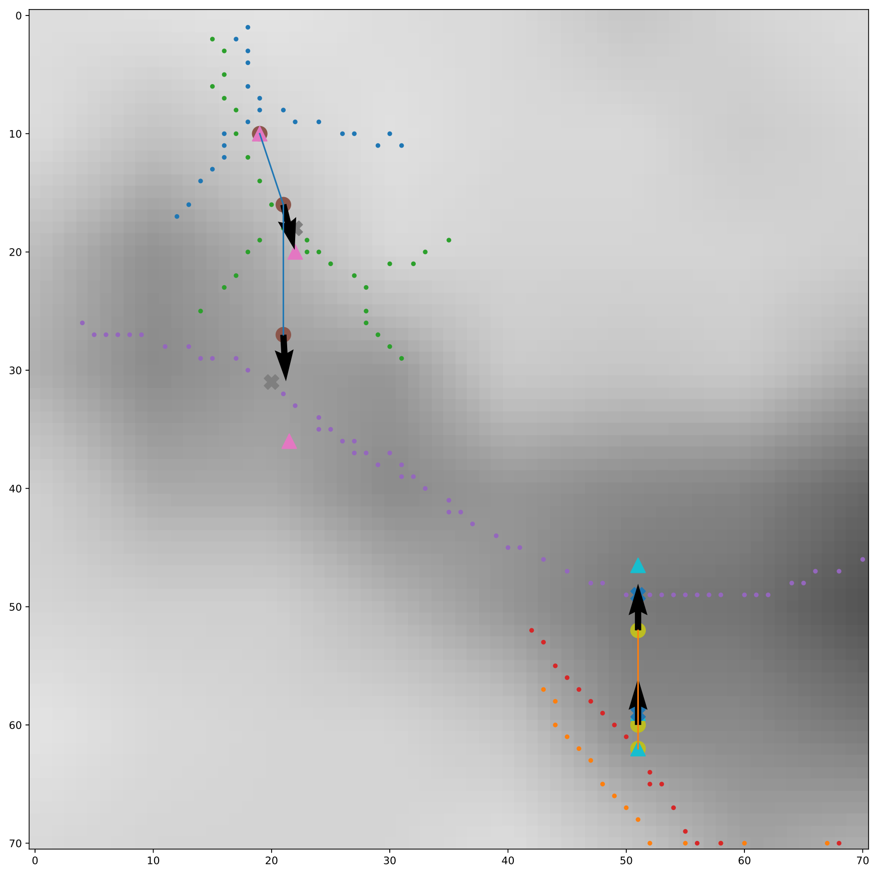


Figure 5.12: A track example (same sequence as Figure 5.7) until two blobs merged and the central point matching is activated. The pixel dots represent the central points. The round markers are the estimated object location, triangle markers are prediction, and cross markers are measurements. Blob contour is omitted for a clear demonstration.

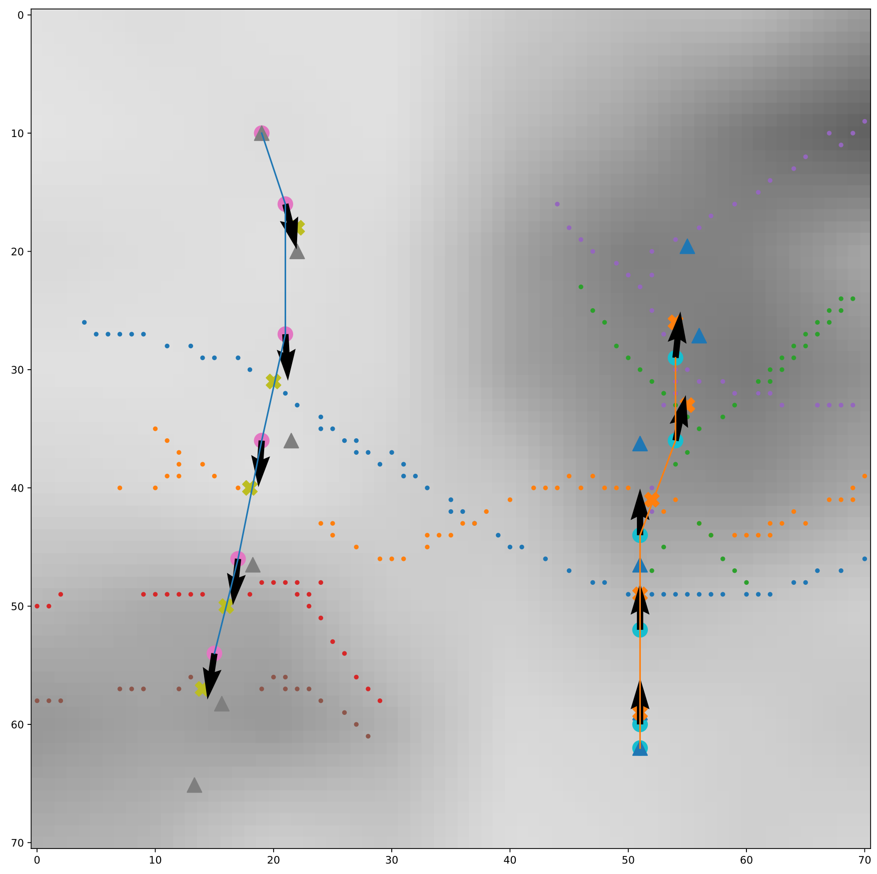


Figure 5.13: A track example (same sequence as Figure 5.7, continued from Figure 5.12) after the merge event. The tracking algorithm successfully handle the merging blobs correctly and continue tracking with centroids.

6 Evaluation

List of Figures

List of Tables

Bibliography

[1] T. Teixeira et al. "A survey of human-sensing: Methods for detecting presence, count, location, track, and identity." In: *ACM Computing Surveys* 5.1 (2010), pp. 59–69.

[2] T. W. Hnat et al. "Doorjamb: unobtrusive room-level tracking of people in homes using doorway sensors." In: *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*. 2012, pp. 309–322.

[3] J. M. Sim et al. "Acoustic sensor based recognition of human activity in everyday life for smart home services." In: *International Journal of Distributed Sensor Networks* 11.9 (2015), p. 679123.

[4] N. Li et al. "Measuring and monitoring occupancy with an RFID based system for demand-driven HVAC operations." In: *Automation in construction* 24 (2012), pp. 89–99.

[5] B. Balaji et al. "Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings." In: *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. 2013, pp. 1–14.

[6] S. Wilhelm et al. "Human Presence Detection by monitoring the indoor CO2 concentration." In: Sept. 2020.

[7] E. Griffiths et al. "An empirical design space analysis of doorway tracking systems for real-world environments." In: *ACM Transactions on Sensor Networks (TOSN)* 13.4 (2017), pp. 1–34.

[8] P. Klasnja et al. "Exploring privacy concerns about personal sensing." In: *International Conference on Pervasive Computing*. Springer. 2009, pp. 176–183.

[9] C. Kowalski et al. "Multi Low-resolution Infrared Sensor Setup for Privacy-preserving Unobtrusive Indoor Localization." In: *ICT4AWE*. 2019, pp. 183–188.

[10] Y. Karayaneva et al. "Use of low-resolution infrared pixel array for passive human motion movement and recognition." In: *Proceedings of the 32nd International BCS Human Computer Interaction Conference 32*. 2018, pp. 1–2.

[11] Y. Jeong et al. "Probabilistic method to determine human subjects for low-resolution thermal imaging sensor." In: *2014 IEEE Sensors Applications Symposium (SAS)*. IEEE. 2014, pp. 97–102.

[12] L. I. L. Gonzalez et al. "Using a thermopile matrix sensor to recognize energy-related activities in offices." In: *Procedia Computer Science* 19 (2013), pp. 678–685.

[13] A. Beltran et al. "Thermosense: Occupancy thermal based sensing for hvac control." In: *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*. 2013, pp. 1–8.

[14] C. Basu and A. Rowe. "Tracking motion and proxemics using thermal-sensor array." In: *arXiv preprint arXiv:1511.08166* (2015).

[15] A. Tyndall et al. "Occupancy estimation using a low-pixel count thermal imager." In: *IEEE Sensors Journal* 16.10 (2016), pp. 3784–3791.

[16] M. Maaspuro. "Low-Resolution IR-Array as a Doorway Occupancy Counter in a Smart Building." In: *International Journal of Online & Biomedical Engineering* 16.6 (2020).

[17] H. Mohammadmoradi et al. "Measuring people-flow through doorways using easy-to-install ir array sensors." In: *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE. 2017, pp. 35–43.

[18] M. Cokbas et al. "Low-resolution overhead thermal tripwire for occupancy estimation." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 88–89.

[19] D. Qu et al. "Indoor multiple human targets localization and tracking using thermopile sensor." In: *Infrared Physics & Technology* 97 (2019), pp. 349–359.

[20] I. LLC. *Irisys True Occupancy*. 2021. URL: `https://www.irisys.net/products` (visited on 08/21/2021).

[21] T. FLIR. *End-To-End Solutions for Large Enterprises*. 2021. URL: `https://www.flir.eu/security/large-enterprises` (visited on 08/21/2021).

[22] S. Mashiyama et al. "Activity recognition using low resolution infrared array sensor." In: *2015 IEEE International Conference on Communications (ICC)*. IEEE. 2015, pp. 495–500.

[23] A. A. Trofimova et al. "Indoor human detection based on thermal array sensor data and adaptive background estimation." In: *Journal of Computer and Communications* 5.04 (2017), p. 16.

[24] R. B. Devi et al. "A survey on different background subtraction method for moving object detection." In: *International journal for research in emerging science and technology* 3.10 (2016).

[25] O. Tutorial. *How to Use Background Subtraction Methods*. 2021. URL: `https://docs.opencv.org/3.4/d1/dc5/tutorial_background_subtraction.html` (visited on 08/23/2021).

[26] S. L. Tang et al. "Hybrid blob and particle filter tracking approach for robust object tracking." In: *Procedia Computer Science* 1.1 (2010), pp. 2549–2557.

[27] A. Yilmaz et al. "Object tracking: A survey." In: *Acm computing surveys (CSUR)* 38.4 (2006), 13–es.

[28] G. Welch, G. Bishop, et al. "An introduction to the Kalman filter." In: (1995).

[29] Panasonic. *White paper: Grid-eye state of the art thermal imaging solution*. 2016 (accessed 2021-04).

[30] Adafruit. *DHT11 basic temperature-humidity sensor + extras*. 2021. URL: `https://www.adafruit.com/product/386` (visited on 08/24/2021).

[31] LILYGO. *LILYGO® TTGO T7 V1.4 Mini32 ESP32-WROVER-B PSRAM Wi-Fi Bluetooth Module Development Board*. 2021. URL: `http://www.lilygo.cn/prod_view.aspx?TypeId=50033&Id=978&FId=t3:50033:3` (visited on 08/24/2021).

[32] Espressif. *ESP32 FreeRTOS API Reference*. 2021. URL: `https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html` (visited on 08/24/2021).

[33] I. Cadence Design Systems. *Tensilica Customizable Processors*. 2021. URL: `https://ip.cadence.com/ipportfolio/tensilica-ip/xtensa-customizable` (visited on 08/24/2021).

[34] N.-R. Committee. *Node-RED Low-code programming for event-driven applications*. 2021. URL: `https://nodered.org/` (visited on 08/24/2021).

[35] A. Kasagi et al. "Fast algorithm using summed area tables with unified layer performing convolution and average pooling." In: *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2017, pp. 1–6.

[36] MATLAB. *Calculate 2-D integral image*. 2021. URL: `https://www.mathworks.com/help/images/ref/integralimage.html` (visited on 08/28/2021).

[37] V. Sharma. "A blob representation for tracking robust to merging and fragmentation." In: *2012 IEEE Workshop on the Applications of Computer Vision (WACV)*. IEEE. 2012, pp. 161–168.

[38] D. G. Bailey. "An efficient euclidean distance transform." In: *International workshop on combinatorial image analysis*. Springer. 2004, pp. 394–408.

[39] M. A. Butt and P. Maragos. "Optimum design of chamfer distance transforms." In: *IEEE Transactions on Image Processing* 7.10 (1998), pp. 1477–1484.

[40] Y.-H. Lee and S.-J. Horng. "The chessboard distance transform and the medial axis transform are interchangeable." In: *Proceedings of International Conference on Parallel Processing*. IEEE. 1996, pp. 424–428.

[41] D. Attali et al. "Stability and Computation of Medial Axes: a State-of-the-Art Report." In: *Mathematical Foundations of Scientific Visualization, Computer Graphics, and Massive Data Exploration* (Jan. 2009).

[42] T. Y. Zhang and C. Y. Suen. "A fast parallel algorithm for thinning digital patterns." In: *Communications of the ACM* 27.3 (1984), pp. 236–239.