2021 S.T. Yau High School Science Award (Asia)

Research Report

The Team

Registration Number: Comp-044

Name of team member: Leong Chi Io
School: Pui Ching Middle School, Macau
City, Country: Macau, China

Name of team member: Wu Chan In
School: Pui Ching Middle School, Macau
City, Country: Macau, China

Name of team member: Ye Chon Hou
School: Pui Ching Middle School, Macau
City, Country: Macau, China

Name of supervising teacher: Leong Chon Kit
Job Title: Mathematics Teacher
School/Institution: Pui Ching Middle school, Macau
City, Country: Macau, China

Title of Research Report

Person Follower Robot Based on Person Re-identification and YOLO for Elderly Caring

Date

30 August 2021

Person Follower Robot Based on Person Re-identification and YOLO for Elderly Caring

Leong Chi Io, Wu Chan In, Ye Chon Hou

Abstract

In 2021, elderlies made up 15.2% of the global population, and 13.5% of the elderlies (worldwide) who are aged above 60 live alone. For these elderlies, everything is dangerous: the sharp corners of a table, woods on the floor, a street full of pedestrians, etc. As nobody might aid them when they are in trouble, an assistant is important to these solo elderlies.

In this paper, we present a system we have developed that tracks and follows an elderly in order to keep an eye on his/her status. We've employed a technique known as person re-identification, which has always been used in multi-camera tracking systems, combined with mapping and prediction of trajectory trends.

The system first detects people from the screen with YOLO person detection, then person re-identification is used to recognize who our master elderly is among all detected ones. Lastly, the robot will follow the elderly for monitoring. Mapping and trajectory trend prediction will be automatically activated when the elderly's existence state has changed (e.g. disappears from the camera).

In addition, due to the low signal strength and accuracy of using GPS indoors, most systems nowadays can only be used outdoors. Since our robot uses other technologies including trajectory trend prediction and person re-identification, we can overcome this problem easily.

As a result, our robot could follow the elderly both at home and outside while monitoring his/her status. When our target is in a crowded location, the robot can still behave well. Even if there are obstacles in front of the robot, it can constantly track and monitor the target. With the above functions achieved, the robot could stick to the elderly and follow it consistently.

There are other person following robots developed in the past, but most of them used continuous tracking methods to 'track' the target person. Tracking methods update its model continually, which makes their accuracy decrease as time progresses (in the long term). Instead, our project aims to 'recognize' the elderly, the target, in order to keep following it, so this method's accuracy won't be bothered by time.

Keywords: EMachine Learning, Object Detection, Person Re-Identification, Mapping, Neural Networks, CNN, AutoEncoder

Acknowledgement

During the writing process of this article, we have gotten a lot of support from teachers and other institutions. We would like to express our thanks to our teachers, Mr. Lam Kin Un, Mr. Lao Guan Hua, and Mr. Leong Chon Kit. We are also very grateful for The Science and Technology Development Fund (FDCT)'s support.

Mr. Lam Kin Un and Mr. Lao Guan Hua have been teaching us for years. In this project, they helped us a lot in programming. Furthermore, while we were planning the experiments, we encountered many difficulties, and they would always discuss with us, trying to improve the experiments.

Mr. Leong Chon Kit is our supervising teacher. Since we have rarely read any paper before, he taught us many precautions for writing essays at the beginning. During the process of writing, Mr. Leong Chon Kit has given us many recommendations, and he has also helped us check our paper.

Moreover, we thank The Science and Technology Development Fund(FDCT) a lot. FDCT has supported us with many types of equipment, such as 3D printers, laser cutting machines, etc. without the support of DCT, we could hardly build up the robot.

We once again sincerely thank the teachers and institutions who have helped us!

Commitments on Academic Honesty and Integrity

We hereby declare that we

1. are fully committed to the principle of honesty, integrity and fair play throughout the competition.
2. actually perform the research work ourselves and thus truly understand the content of the work.
3. observe the common standard of academic integrity adopted by most journals and degree theses.
4. have declared all the assistance and contribution we have received from any personnel, agency, institution, etc. for the research work.
5. undertake to avoid getting in touch with assessment panel members in a way that may lead to direct or indirect conflict of interest.
6. undertake to avoid any interaction with assessment panel members that would undermine the neutrality of the panel member and fairness of the assessment process.
7. observe the safety regulations of the laboratory(ies) where we conduct the experiment(s), if applicable.
8. observe all rules and regulations of the competition.
9. agree that the decision of YHSA(Asia) is final in all matters related to the competition.

We understand and agree that failure to honour the above commitments may lead to disqualification from the competition and/or removal of reward, if applicable; that any unethical deeds, if found, will be disclosed to the school principal of team member(s) and relevant parties if deemed necessary; and that the decision of YHSA(Asia) is final and no appeal will be accepted.

(Signatures of full team below)

X 梁梓堯

Name of team member: 梁梓堯

X 吳粲然

Name of team member: 吳粲然

X 葉俊濠

Name of team member: 葉俊濠

X 梁駿杰

Name of supervising teacher: 梁駿杰

Noted and endorsed by

(signature) 高錦雄

PUI CHING MIDDLE SCHOOL 澳門 培正中學 MACAU

Name of school principal: 高錦雄

Table Of Contents

1. Background

1.1 Current solutions

Person-Following using Active Target Search[1]

This method will first apply leg detection with a random tree classifier with the laser scanner, then track it with an extended Kalman Filter with a constant velocity assumption.

After the human leg was tracked, OpenPose 3d pose estimation will be applied on that corresponding person to compute the distance to the person from a robot by using the average coordinates of the recognized body parts, and clustering the point clouds of that average. With OpenPose, the system could find out the upper body position of the target person and use cloth detection based on histogram template matching.

Finally, using a Histograms of Oriented Gradients (HOG) to detect faces and face landmark estimation to extract face features. Then, the extracted features are used to train a Deep Convolutional Neural Network to recognize faces to identify the identity of the target person.

As a bonus, the robot will use SVR based trajectory prediction to find the person in case it loses from the view.

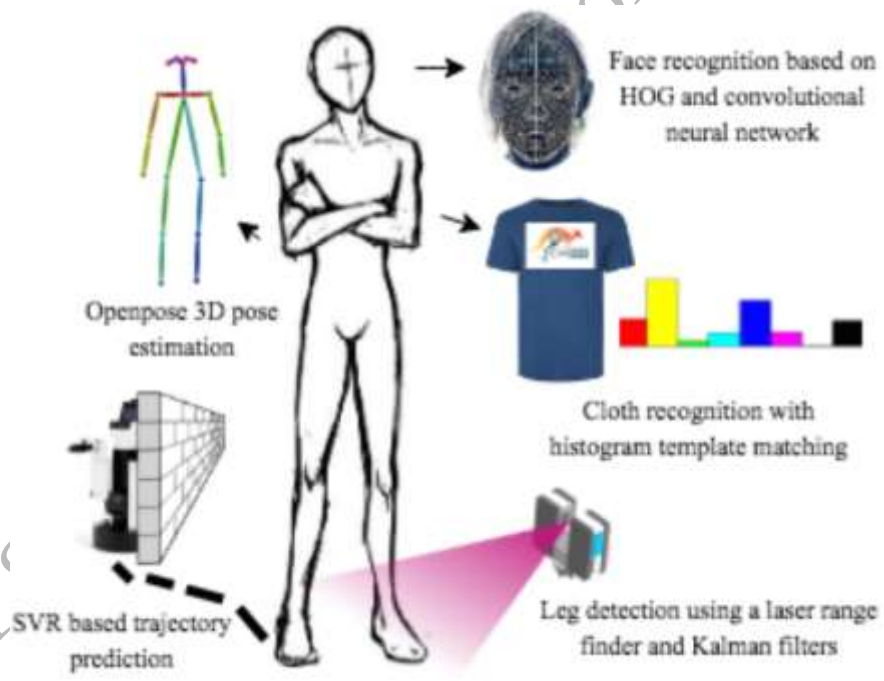
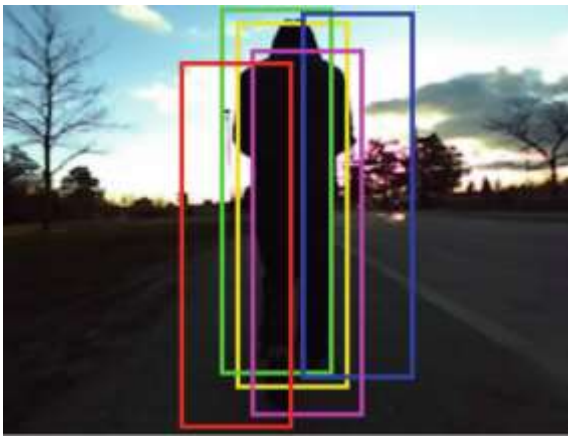


Fig 1.1.1 The Algorithm process of this method

Integrating Stereo Vision with a CNN Tracker for a Person-Following Robot[2]

This method introduced an actively self-updated CNN based tracker to follow the target person. The system will first require the target person to send inside a bounding box placed at the center of the camera view in order to activate the CNN tracker. Then, the model will get patches from the images in the blue bounding box as the target class and images outside as the non-target class. The system will train the model instantly with these datas.

After the model was initialized, the system searched the test patches in a local image region. The system also restricts the search space with respect to the depth. If the patches in the image do not have the depth within previous depth ± α, the system does not consider them, where α is the search region in depth direction (we use α = 0.25 m). If the extracted patch was smaller than the original patch size's 70%, it will be discarded. After patches are extracted, it will be sent to the model to recognize.



During the following process, the system will keep training the model with new data as to update the CNN tracker to duel with different environments and also cloth changing situations.

As same as the first method, this method also applies trajectory prediction when the target was lost after 0.5s

← Fig 1.1.2 Extracting random patches from image

1.2 Gathering up

We could easily see the weakness from the first method: backwards. This method hardly relies on face recognition to recognize the target, so if the target turns around, the only option for it to recognize will be the cloth color detection by histogram template matching. If somebody wears the same clothes and walks by, the robot will follow the wrong person without seeing it's face.

As for the second method, the CNN Tracker which was introduced in that method did a great job beating most methods on following. But there's still problems. As the target patch was extracted out by depth distance ± α, it actually doesn't truly detect whether it's a human or not. So if we give a dog at the beginning of the initialization, the method will follow the dog. Also, the model trained instantly has claimed by the author that it has strong over-fitting problems, as the author claimed that the only way to handle this strong over-fitting was that the target pose and appearance should not change dramatically in the first 50 frames.

To sum up, the first method was trying to 'recognize' the target, but was over-relying on face recognition. The second method did a great job overall shown in the test video, which uses trajectory prediction to supply the weakness of the CNN Tracker. But there's still one problem: The method doesn't even know what a human is, it may follow something which wasn't a human. For this, extracting patches from the image just by depth was very dangerous.

Viewing the weaknesses in these projects, we decided to create a method in which the robot is truly recognizes the person and knows what a person is. If it was succeeded, this could deal with a lot of problems that currently tracking methods may occur.

2. Person Following System

2.1 System Building

Our system is divided into 3 parts: the *Initializer*, the *Main Application* and the *RobotHandler*. The following actions occur in order:

1. The system first registers an initial descriptor with the target person's front and back body image cropped out by YOLO's human detection. At this point, the person must stand inside a blue bounding box in the center of the robot's view.
2. After registration is done, the main application continues to track people with YOLO and capture the descriptor of everybody detected with the image cropped out.
3. It compares them with the initial descriptor (the target's) with *cosine similarity*. The descriptor with the highest similarity would be recognized as our target.
4. The robot will follow the target person using the x-coordinate in the image and the distance from the target person by inputting them into the PID Controller, which is handled by the RobotHandler for the robot to move.

When moving, the robot will keep collecting the target person's coordinates on the map and save them as waypoints for trajectory trend prediction, and this part is handled by the *Waypoint Recorder Application*.

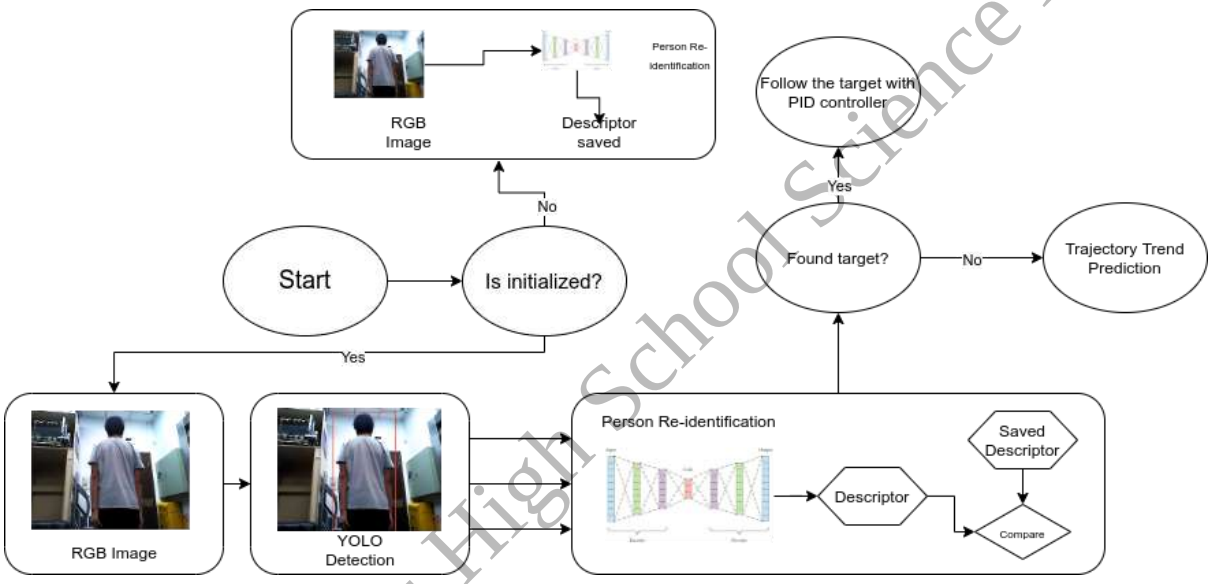


Fig 2.1.1 The state graph of our system

If the Main Application loses the target person from the camera view (for instance, another person suddenly crosses between the target and the robot), the robot will first enter **SEARCHING** mode. The robot will follow the closest person to the main target person's last existing point for 1.5 seconds. If the duration has passed, and the robot still couldn't spot the main target person from the view, then the robot will enter **LOST** mode. It will predict the trajectory trend of the target with the help of previous waypoints recorded by the waypoint recorder. After that, the predicted trajectory will be sent to the AMCL planner to plan a path that can reach the predicted position of the target. With the map, AMCL could also help us avoid obstacles when approaching the predicted position.

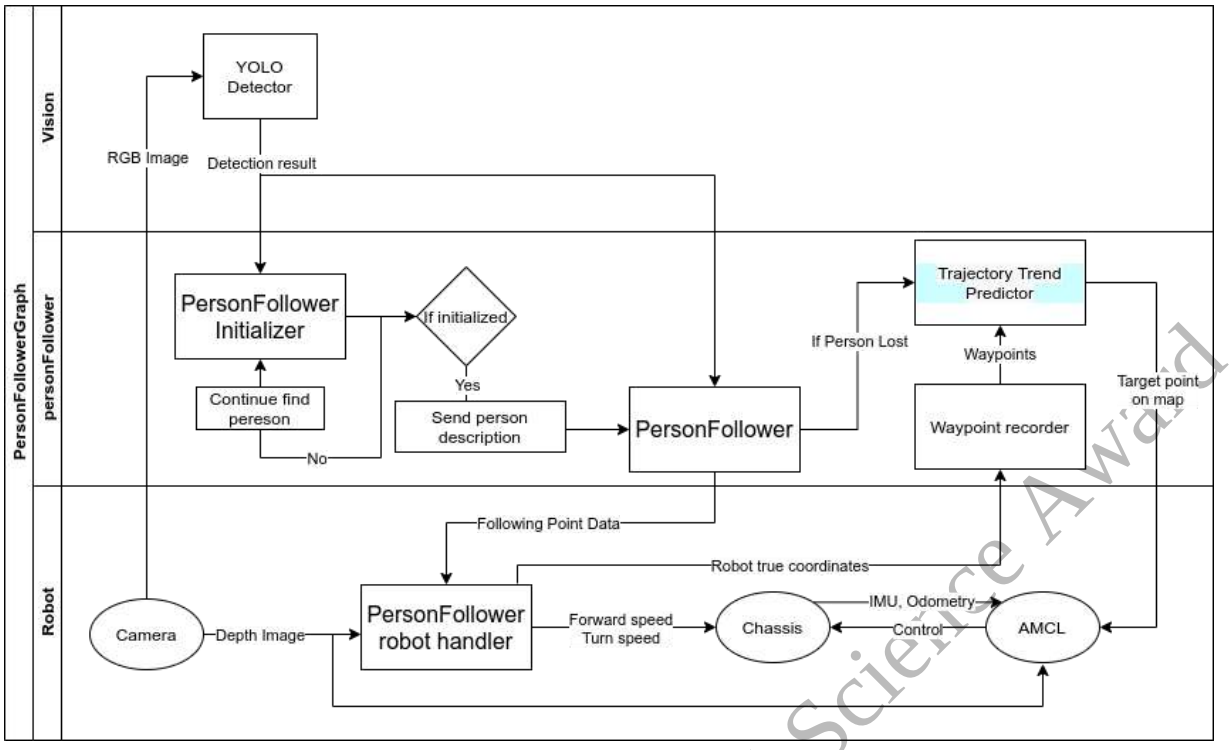


Fig 2.1.2 A brief graph of the system

2.2 Feature Extraction

2.2.1 Extracting Persons

Our robot relies heavily on its computer vision. It makes use of a camera to "look" at the outside world and passes this information to a series of image processing and (re)identification models so that it can follow the main target. The first part of our model, the *YOLO real-time object detection system*[4], involves detecting all individuals who appear in front of the robot.

Humans have the innate ability to immediately recognize a person in an image thanks to thousands of years of evolution, and making machines doing so has been a major topic for researchers. Older object detection systems first trained a classifier that was able to recognize objects (such as human in our case) and ran it on several locations on the image to be able to detect every object, some approaches with R-CNN (*Region-based Convolutional Neural Network*) [3] first computes several bounding boxes and runs a separate classifier on each box. These systems are comparatively slower due to the number of computations, so the YOLO system employs a "*you-only-look-once*" principle and trains a single convolutional network to detect all objects with their corresponding bounding boxes in one go.

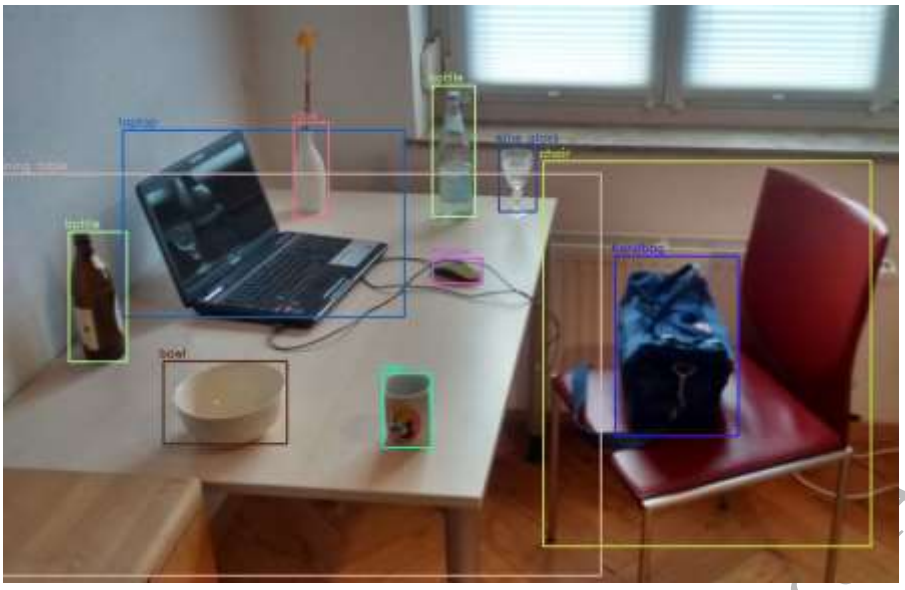


Fig 2.1.1 An example of YOLO object detection in a room

In essence, YOLO is given the RGB image from our camera on the robot and it figures out all humans and "draws" a bounding box for them. YOLO has the advantage of bringing real-time human detection. Each bounding box created by YOLO has 5 parameters, $(x,\ y,\ w,\ h,\ confidence)$, where the $(x,\ y)$ coordinates denotes the position of the center of each box, and $(w,\ h)$ is the width and the height of the bounding box respectively. The $x,\ y,\ w,\ h$ values are normalized so that $x,\ y,\ h,\ w \in [0,\ 1]$. The *confidence* value depicts how "sure" the system is about its prediction. This information (vectors of objects) is later passed to the second stage of our system, the person re-identification part, for further processing.

The YOLO system divides the image taken from the RGB-D camera into an $S \times S$ grid. Mathematically, $confidence := \mathbb{P}(Obj) * IOU_{pred}^{truth}$, where *IOU* is the *intersection-over-union* of the predicted box (generated by the system) and the ground truth (prelabeled information). During training, when labeled pictures are passed to the system, $\mathbb{P}(Obj)$ is 1 if and only if an object is presented in the cell, and 0 otherwise; the *IOU* value is repeatedly calculated based on both the ground truth and the prediction made. The *leaky ReLU activation function*

$$\phi(x) = \begin{cases} x & \text{if } x > 0 \\ 0.1x & \text{if } x \leq 0 \end{cases}$$

is used for all layers.[4]

We've tested other methods including SSD[5] and R-FCN[6], as the results are shown in ***Fig 2.1.2***.

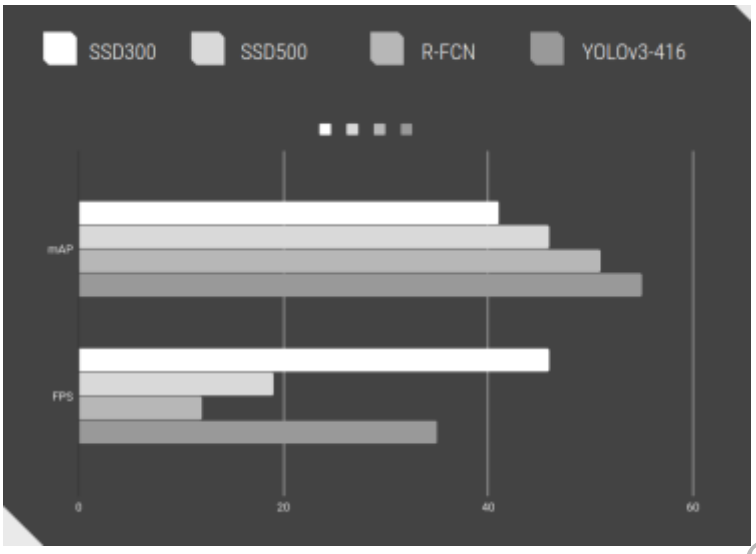


Fig 2.1.2 The comparison mAP (mean average precision) and FPS (frame per second) between tested methods

As we need to keep the video stream live to spot the target person, FPS is an important criterion. The results yield that SSD300 gains the highest speed in FPS, while YOLO has mediocre speed (about 30 ~ 35 FPS, close to real time), but YOLO has the highest mAP score which others couldn't achieve, so at last YOLO was selected to be our detection model.

2.2.2 Extract Features from Persons

After extracting people from the image, we would need to recognize the master target among all people in order to follow him/her.

Before person re-identification is used, we have tested using object trackers to archive the re-identification process. As the trackers yield high accuracy results.

Fig 2.2.2.1 Two cars tracked by MedianFlow tracker

Here is a list of trackers we've tested, description made by Dr.Adrian Rosebrock[15].

Tracker	Backend	Description
BOOSTING Tracker	Adaboost[7]	It's over a decade old. This tracker is slow and doesn't work very well.
MIL Tracker	Adaboost	Better accuracy than BOOSTING tracker

KCF Tracker	Kernelized Correlation Filters[8]	Faster than BOOSTING and MIL. Similar to MIL and KCF, it does not handle full occlusion well.
CSRT Tracker	Discriminative Correlation Filter[9]	Tends to be more accurate than KCF but slightly slower.
MedianFlow Tracker	Itself	If there is too large of a jump in motion, such as fast-moving objects, or objects that change quickly in their appearance, the model will fail.
TLD Tracker	Itself	Is incredibly prone to false-positives
MOSSE Tracker	Itself	Very fast but not as accurate as CSRT or KCF
GOTURN Tracker	Deep learning	Requires additional model files to run, reportedly handles view changes well
Dlib Correlation Layer Tracker	Correlation Layer[10]	Better than everything above, but accuracy lowers the longer it runs
YOLO + Deepsort[11]	Kalman Filter	A method that depends on the kalman filter, performs good at tracking balls experiment

The trackers require a piece of image of the target object to do the tracking operation, which means the object trackers could track any objects. It is not limited to humans, cars in the figure do work.

Testing the trackers

As for testing the listed trackers, we've set up a situation with the following figure to test if the trackers do behave as how they are in the example videos.
In the tests, there is a camera stayed at a fixed location and a fixed angle, as in ***Fig 2.2.2.2***:

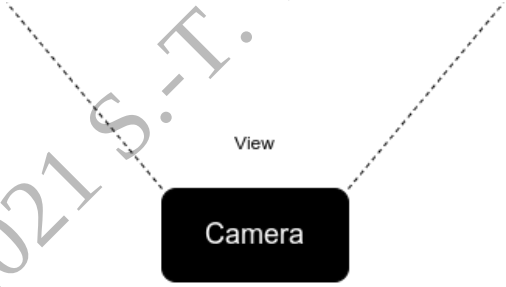


Fig 2.2.2.2 The camera position environment

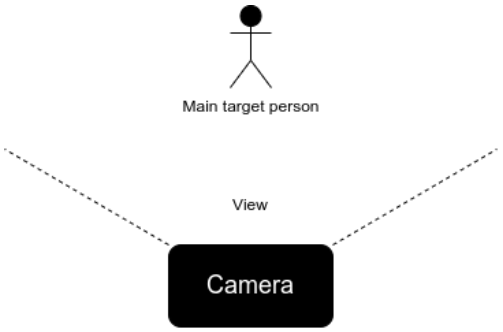


Fig 2.2.2.3 The main person

The environment will be fixed so the lighting conditions would never change throughout the tests. The target person that the tracker should recognize, will lay in the center.

There would be two levels of tests performed. Each level includes 2 stages. In the first stage of the first level, the main target person only slightly swings without others appearing in front of the robot,

and the second stage involves having two non-target people (‘*annoying people*’) slowly following a specific trajectory, trying to steal the focus of the main target person from our robot.

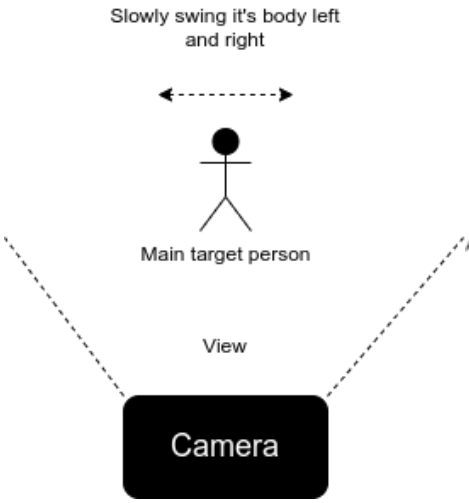


Fig 2.2.2.4 The first stage in the first level

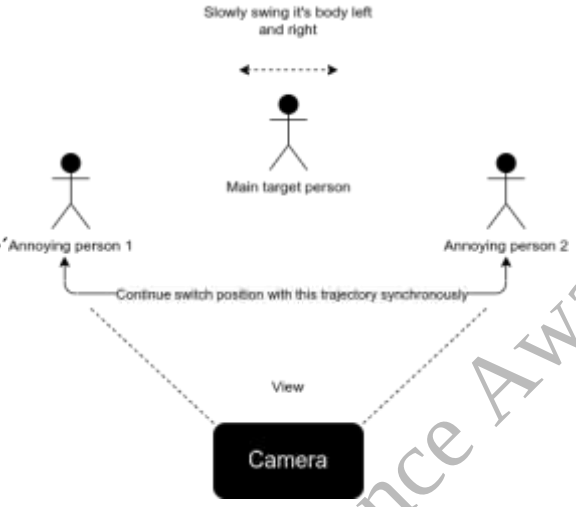


Fig 2.2.2.5 The second stage in the first level

The second level of the test has the same environment and also 2 stages (just as level 1), but the main person would move in a specific trajectory as shown in ***Fig 2.2.2.6*** and ***Fig2.2.2.7***. This increases the difficulty for the tracker, especially when the ‘annoying people’ are present in the second stage.

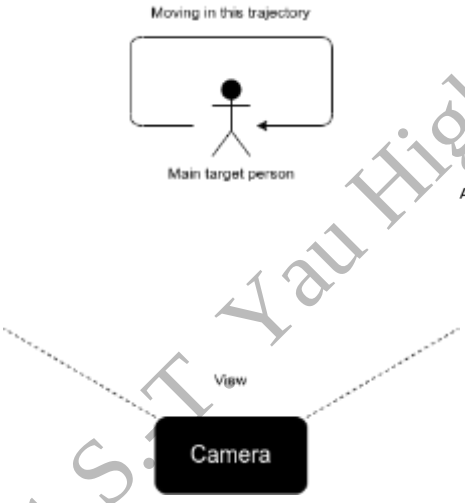


Fig 2.2.2.6 The first stage in the second level

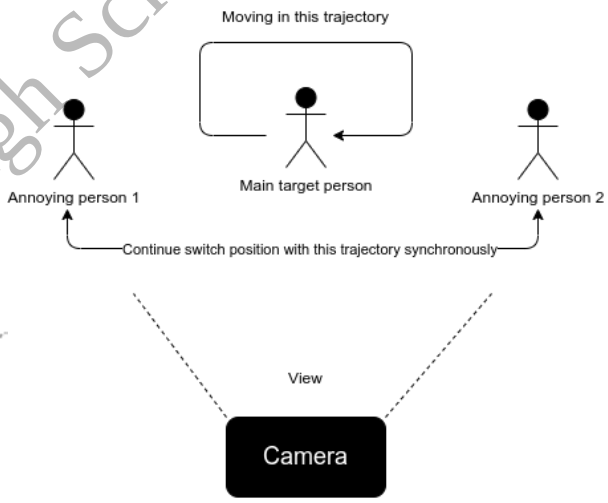


Fig 2.2.2.7 The second stage in the second level

Each tracker is tested 3 times in a stage, and each stage lasts for 3 minutes. Due to the trackers won’t lose the box when it missed the target (for instance, Dlib correlation layer tracker’s box will continue to increase its size until the box is as large as the image if it lost the target), If the centroid of the box wasn’t on the main target person, the tracker will be estimated as it lost the target and the test will stop immediately.

Here are the test results in level 1:

Level 1	Averagely last for in 3 times	Averagely last for in 3 times
Tracker	First Stage	Second Stage
BOOSTING Tracker	3 minutes	5 seconds
MIL Tracker	3 minutes	9 seconds
KCF Tracker	3 minutes	12 seconds
CSRT Tracker	3 minutes	12 seconds
MedianFlow Tracker	1 minute	7.3 seconds
TLD Tracker	50 seconds	3 seconds
MOSSE Tracker	30 seconds	4 seconds
GOTURN Tracker	2 minutes	5.356 seconds
Dlib Correlation Layer Tracker	3 minutes	20 seconds
YOLO + Deepsort	3 minutes	2.5 minutes

Most trackers could pass the first stage without losing the main target person in 3 minutes. But in the second stage, some tracker's tracking box starts to lose the target when annoying persons start to do their job. Some frequently let the annoying person steal the main target person's position. However, YOLO + Deepsort was ok even in both stages, YOLO + Deepsort does a great job in level 1.

Here are the test results in level 2

Level 2	Averagely last for in 3 times	Averagely last for in 3 times
Tracker	First Stage	Second Stage
BOOSTING Tracker	20 seconds	1 seconds
MIL Tracker	2 minutes	1 seconds
KCF Tracker	1 minute	6 seconds
CSRT Tracker	1.4 minute	7 seconds
MedianFlow Tracker	10 seconds	2 seconds
TLD Tracker	20 seconds	5 seconds
MOSSE Tracker	5.4 seconds	6 seconds

GOTURN Tracker	22 seconds	5.356 seconds
Dlib Correlation Layer Tracker	2 minutes	13 seconds
YOLO + Deepsort	3 minutes	1 minute

In level 1, most of the trackers could track the main target person for as long as 3 minutes at stage 1, but it immediately decreased into seconds in stage 2. Moving seems to give all trackers a trouble to finish their job at the first stage, not to mention in the second stage, as even the Dlib correlation layer tracker failed in averagely 13 seconds, and the phenomenon of the main target person's position being stolen becomes more often in level 2.

In conclusion, although YOLO + Deepsort tracking method did last for an average of 1 minute at the last test, the following was not a timed task, it wasn't timed, So YOLO + Deepsort couldn't be used in our project. not to mention the others, like the older following solutions mentioned. Moreover, some trackers, including YOLO + Deepsort, failed in stage tests due to it following the annoying person instead of the main target person.

For our project, we have to find a way that could 'recognize' our main target person, at least not bothered by time problems.

Person Re-identification

Due to these consequences, we have decided to use a new technology, which other papers had never used: person re-identification.

The person re-identification model, pre-trained by Intel developers, is a model based on OmniScaleNet that can achieve person re-identification. It is based on RMNet[12] and is an autoencoder structure based model. This part works by inputting the people detected with YOLO to the autoencoder, the model would then return a 1×256 feature vector representing each person's image.

This lets us achieve the comparison using cosine similarity. Let $V = \{\vec{v_1},\ \vec{v_2},\ \ldots,\ \vec{v_k}\}$ be the set of vectors where each vector represents a single person. Each time, we pick $\vec{v_i}$ to be our target if and only if it is the "closest" vector to $\vec{v_0}$, the vector of our master target registered at the beginning. Formally, we select $\vec{v_i}$ s.t.

$$\cos\langle \vec{v_i},\ \vec{v_0} \rangle = \frac{\vec{v_i} \cdot \vec{v_0}}{|\vec{v_i}||\vec{v_0}|}$$

is the greatest. We opted for *cosine similarity* to calculate how close these vectors are since compared to Euclidean distance, it has the advantage of already normalized to be in $[-1,1]$ and is thus easier to tackle.

Fig 2.2.2.2 A testing image provided by Intel development team.
The black bordered image was the initial image. The right images with green and red borders are images to be compared with. Images with red boxes are estimated that the person inside isn't the same person as the initial one, and green has the opposite meaning.

Testing with all levels as the same, combined with YOLO person detection:

Level 1	Averagely last for in 3 times	Averagely last for in 3 times
Tech	First Stage	Second Stage
YOLO + Person Re-identification	3 minutes	3 minutes

Level 2	Averagely last for in 3 times	Averagely last for in 3 times
Tech	First Stage	Second Stage
YOLO + Person Re-identification	3 minutes	3 minutes

The test results promoted that person re-identification could pass all tests without losing the main target person in 3 minutes. Theoretically, if the test continues without a time limit, it could recognize the main target person forever. This method's accuracy wouldn't be decreased as time progresses due to the fact that person re-identification focuses on '*recognizing*' the target instead of 'tracking' it. Here's an example: if a person re-identification model has recognizes me, even another non-target person blocks the view, no matter how long time has passed (1 minutes, 1 hours, 1 day, even a year), as long as the environment doesn't change substantially, once the other person unblocks the view, the system could still recognize the target. This is a thing other trackers couldn't do, since they're not trying to 'recognize' the target, but to track. (Dlib Correlation Layer Tracker is an exception, though if the target keeps moving before another person approaches, it would still lose the target). With these characteristics, the person re-identification model would never follow the wrong person. Moreover, in reality our robot will be moving, but object trackers were often designed for fixed spot cameras. As such, our robot performs comparatively better than its alternatives.

Fig 2.2.2.3 One of the images captured from the test
Green bounding box represents the main target person

OpenVINO

OpenVINO™ toolkit is a comprehensive toolkit for quickly developing applications and solutions that solve a variety of tasks including emulation of human vision, automatic speech recognition, natural language processing, recommendation systems, and many others. Based on latest generations of artificial neural networks, including Convolutional Neural Networks (CNNs), recurrent and attention-based networks, the toolkit extends computer vision and non-vision workloads across Intel® hardware, maximizing performance. It accelerates applications with high-performance, AI and deep learning inference deployed from edge to cloud.

Normally, neural networks require a graphic processor unit to gain fast evaluation, not to mention training the model. Especially for object detection models, these models require fast evaluation to gain real time speed of detection. The OpenVINO toolkit introduces a model optimizer, which could make models as lightweight as possible. The converted model could even run on a CPU, though the use of GPU can still results in gain of speed.[13]

This framework helps us gain high speed evaluation although running over 2 models at the same time. Our tests reported that YOLO would still be slow even when optimized by OpenVINO, so YOLO would be running on the GPU. As YOLO has already consumed it, person re-identification would instead be evaluated by OpenVINO.

Fig 2.2.2.4 The scheme above illustrates the typical workflow
for deploying a trained deep learning model

2.3 Robot Moving

2.3.1 Keeping Same Distance

With the help of our computer vision system, our robot can now capture the state of our target person, the last part of the job is to keep the same distance with our master target as to follow him/her.
Proportional-integral-derivative control, or *PID control*, is an industrial standard method of offering both steady-state and short-term response. It achieves the goals by computing the *error value* and using it to give adequate actions (or more formally *feedback*) to correct the error. Due to its simplicity and efficiency, it has been employed widely.
The PID controller is a closed loop controller that consists of three parts - proportional controller (P), integral controller (I) and derivative controller (D).

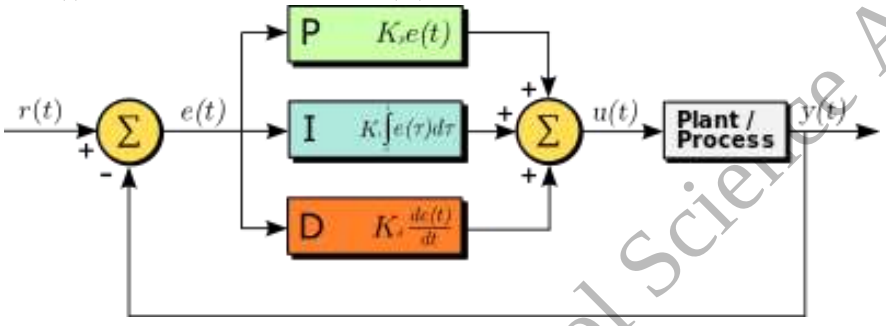


Fig 2.3.1.1 A diagram of the PID Controller

The first part of our control system, the proportional controller, simply takes the error value and multiplies it by the *proportional gain* (K_p) constant. Formally, let $u(t)$ denote the control output function with time being its input, and $e(t)$ as the error value. As such, $u(t) := K_p e(t)$, where $u(t)$ can be viewed as a direct control parameter for our person follower robot. In essence, the greater the error is, the more our robot needs to correct its movement so as to compensate for the error. The proportional controller gives a general corrective response.
The "P" term alone isn't enough for us to have consistent motion. On some occasions, our robot might suffer from sudden changes in the environment, for example, a pedestrian suddenly appearing in front of our robot. In such cases, $e(t)$ changes suddenly, but the proportional controller alone isn't enough to cope with such changes since it performs better in steady-state situations. For that case, we add the *derivative term* to our control function, leading to $u(t) := K_p e(t) + K_d \frac{de(t)}{dt}$, where K_d denotes the *derivative gain* constant and the slope encapsulates the idea of sudden changes. When there's no sudden (and significant) changes in the environment, the slope should be 0; when there is, the derivative term would be large and help us correct such differences in a short period of time.
With the aid of the first two terms, our robot can now tackle lots of circumstances, yet our robot might encounter "long-lasting" errors. For example, an improperly constructed robot that's imbalanced may lead to non-straight line movements, or a robot moving along a slope would constantly be shifted (due to gravity) to a certain direction.
Hence for these persistent errors, the *integral* term is required to compensate them (such as constantly applying correction to the side the robot is incorrectly shifting to) so that the robot can take into account accumulated error. With the integral gain (K_i) constant, we can apply this term into the control output function. As such, the final expression is as follows:

$$u(t) := K_p e(t) + K_d \frac{de(t)}{dt} + K_i \int_0^t e(s)ds$$

.

When there are persistent errors due to any reason, the integral term would constantly be non-0. Thus the term would help the robot regulate its movements.
The three "gain" constants ($K_p,\ K_d,\ K_i \in \mathbb{R}$) are the ones that can be tuned to achieve optimal results. In conclusion, the PID controller gives our robot the ability to effectively deliver safe and accurate motion based on various types of input described above.

2.3.2 On Target Lost

After the basic features of our system, we have added some bonus features to deal with situations when he/she (the target person) is temporarily lost from view. Trajectory trend prediction is employed to deal with it.
If the main target person is lost from the view, the robot will enter **SEARCHING** mode - the robot would follow the person closest to the main target person's last existing point for 1.5 seconds. If this duration has passed and the robot still couldn't spot the main target person from its view, the robot will enter **LOST** mode, in which trajectory trend detection will be activated for the robot to actively search for the target.

True Positions & Waypoints

As to spot the target's position on map for trajectory prediction and AMCL, we've got the target's real coordinates ($\mathrm{R_x},\ \mathrm{R_y}, \mathrm{R_z}$) in the image view instead of pixel unit coordinates (x', y').
We could get the pixel unit coordinates (x', y') directly through the RGB image, also the distance d from the depth image. We also know the FOV_H of our astra $\angle\beta$ and FOV_W $\angle\alpha$. Lastly, the image width and height in pixels w' and h'. We could use the following formula to convert it to real coordinates:

$$x = \frac{x'}{w'} \cdot 2 \cdot d \cdot \tan\frac{\beta}{2},\ y = \frac{y'}{h'} \cdot 2 \cdot d \cdot \tan\frac{\alpha}{2}$$

where $R_w = 2d\tan\frac{\beta}{2}$ is the true width of the camera in mm and $R_h = 2d\tan\frac{\alpha}{2}$ is the true height of the camera in mm
But in actual concern, our camera will lift up $\angle$C as the astra camera was not enough to view the upper body of the person. Also, the d which was received was actually the vertical distance between the object and the camera. In order to get the true distance z, we will have to correct it to ($\mathrm{R_x},\ \mathrm{R_y}, \mathrm{R_z}$):

$$R_x = x - \frac{R_w}{2}$$

$$R_y = z\sin\angle C + \cos\angle C \cdot \left(\frac{1}{2}R_h - y\right)$$

$$R_z = z\sin\angle C - \tan\angle C \cdot \cos\angle C \cdot \left(\frac{1}{2}R_h - y\right)$$

With these algorithms, we could get the true position of the person and mark it as waypoints on the map to record the trajectory of the main target person.

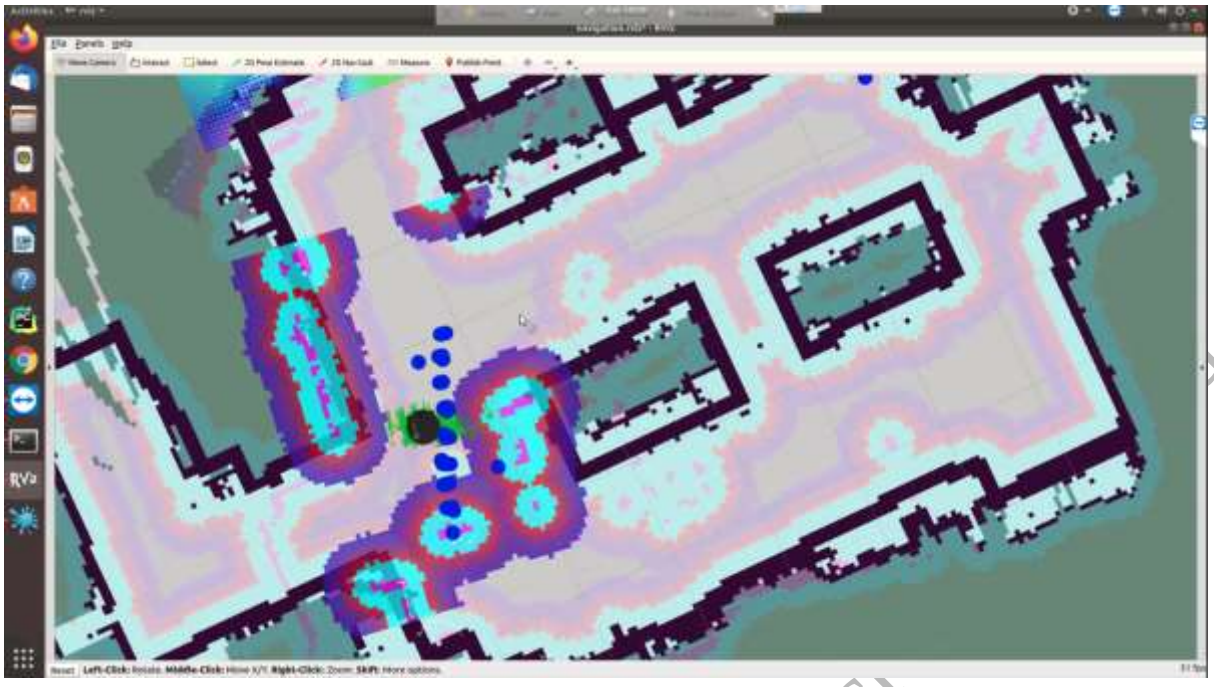


Fig 2.2.2.5 Blue dots are the waypoints as the black object was the robot

Trajectory Trend Prediction

When **LOST** mode is activated, the robot will start using trajectory trend prediction. The robot would first use AMCL to approach the last point where the main target person existed. Then, the robot would predict the average of all waypoints' relative x-coordinates to the robot and calculate if the person was going left or right. Lastly, the robot will turn to that direction and try to spot the main target person again.

2.4 ROS

ROS, which stands for *Robot Operating System*, is an open-source, meta-operating system for your robot. It provides the services you would expect from anrgen operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks', which transports ROS messages in and out of a real-time process. ROS also has seamless integration with the Orocos Real-time Toolkit. As for ROS's characteristics, it is a decent cross-platform tool for multiprocessing.[14]

ROS declares each program as '*nodes*' running asynchronously. Technically, multiprocessing is easy to achieve in most modern programming languages, for instance, Python has a library named *threading* that handles it. However, ROS introduced a communication platform for 'nodes' to communicate with each other through 'topics', on which each 'node' could 'publish' a message through a '*topic*' and another 'node' could '*subscribe*' that. This streaming communication tool provides ROS with the ability to 'glue' programs from different platforms, such as a 'node' written in C++ could easily communicate with a 'node' programmed in Python.

Another characteristic of ROS is the fact that it supports multiple-machine communications through the *TCP protocol*. Practically, one can create a 'node' on a computer and communicate with a 'node' on another machine easily by ROS. This characteristic plays an important role in our system's

structure. As you saw in the brief graph, our system involves a large amount of programs running asynchronously and requires communication between each other.

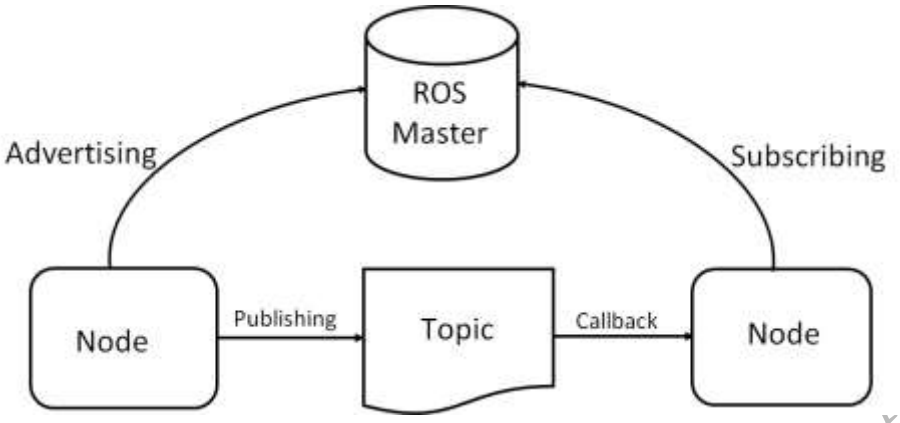


Fig 2.3.1 A typical ROS Model: System components.

2.5 AMCL

After the trajectory of the robot is predicted, it still needs to know the surrounding environment to correctly move. The robot *localization* problem involves determining the position and orientation of a robot based on partial information, such as from sensor data, without an in-built map already stored in the robot.

Particle filter localization, also known as *Monte Carlo localization* (abbreviated as *MCL*), generates some *particles* that represent the hypothesis of the robot's current position. They are first instantiated according to some distribution, then at each round, each particle (guessed location) is moved to the new location (*resamples*) and those which are inconsistent with sensor input are removed and the system generates more particles closer to those who are consistent for the following iteration. After several iterations, the place particles converge are expected to be the actual position. *Adaptive particle filter localization*, the method used, samples a dynamic number of particles. Instead of having a large $|\chi|$, the number of particles is recalculated in each iteration since when most particles converge the number of computations can be reduced by having a more appropriate sample size.

Given AMCL, our robot is able to apply sensor input to the map planner to construct a partial map in case the target suddenly disappears. This works even when it's positioned in an unfamiliar environment.

3. Experiment

3.1 Stage Setup

3.1.1. Hardware

First, we used *Solidworks* (a 3D CAD modeling software) to design our robot's body. The body is mainly divided into three layers, from top to bottom.

The first layer is equipped with a touch screen, a microphone, and a RGB-D lens. The role of this layer is to help and understand the target and to provide real-time communication with the outside world.

The target can use the microphone and the screen to command the robot or give instructions, and the robot can also understand the real-time situation of the target with the camera.
The second layer is the *storage layer*, where the target can place heavy objects. The current load on this layer is 8 kg.
The third layer is the *body layer*: an additional camera is installed here; all the computers, batteries, and the chassis are installed on this layer, which promotes the progress and operation of the robot.

Fig 3.1.1 The robot design in solidworks

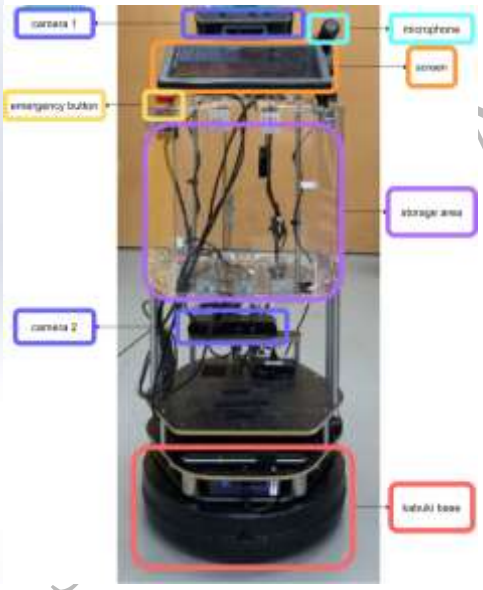


Fig 3.1.2 The robot in reality

Camera

Two cameras are used throughout the whole robot. The top camera (camera 1) is used for person following and the bottom one (camera 2) is used by the AMCL planner to achieve localization and path planning.
We used an *RGB-D camera* (RGB with depth)(camera 1) to collect colorful image information in order to recognize the target, and the depth part is for distance measuring. Combining these two parts can provide the robot with the ability to track and follow our target.
In addition, trajectory prediction requires data of past locations of the target (as waypoints) to predict his/her path, so the RGB-D camera is required to record the position of the target and generate waypoints. As for the position of the top astra camera and the specs, the upper astra camera couldn't totally view the whole target's body except only the upper part of the body.

Parameter	Sensor			
	ORBBEC ASTRA S	MICROSOFT KINECT II	INTEL SR300	INTEL D435
Range	0.4m-2m	0.5m-4.5m	0.3m-2m	0.2m -10m
RGB FOV	60°(H) × 49.5°(V) × 73°(D)	70.6°(H) × 60°(V)	41.5°(H) × 68°(V) × 75.2°(D)	69.4°(H) × 42.5°(V)) × 77°(D)
Depth FOV	60°(H) × 49.5°(V) × 73°(D)	70.6°(H) × 60°(V)	55°(H) × 71.55°(V) × 88°(D)	91.2°(H) × 65.5°(V)) × 100.6°(D)
Frame rate	30 fps	30 fps	30, 60 fps	30, 60, 90 fps
RGB resolution	640×480 pixel	1920 × 1080 pixel	1920 × 1080 pixel	1920 × 1080 pixel 1280×720 pixel 848×480 pixel 640×480 pixel
Depth resolution	640×480 pixel	512 × 424 pixel	640 × 480 pixel	1280 × 720 pixel 848×480 pixel 640×480 pixel
Weight	300 gr	966gr	300gr	100gr
Size	165mm × 30mm × 40mm	255mm × 66mm × 67mm	110mm × 12.6mm × 4.1mm	90mm × 25mm × 25mm
Power supply	USB 2.0	power supply + USB 3.0	USB 3.0	USB 3.0
Power consumption	< 2.4 W	~ 15 W	650-1800 mW	618 -1978 mW
Operating system	Android, Linux Windows 7/8/10	Linux Windows 8/10	Linux Windows 8/10	Linux Window 8/10
SDK	Astra SDK OpenNI2 3rd party S	Kinect V2 SDK libfreenect2	Intel©RealSense™ SDK librealsense sdk [6]	Intel©RealSense™ SDK librealsense SDK[5] hand and face tracking

Fig 3.1.1 A list of Depth cameras data

We used the ORBBEC Astra S camera as it has the lowest price of 1,000 RMB and its resolution is enough for our person-following system.

Storage area

The storage area is an essential hardware section in the design of our robot. We could help our targetcarry objects such as trash. When designing this storage area, the aim was to create a place as large as possible. As we don't want our robot to be too heavy, we picked transparent plastic for our storage area design.

Center Control System

Fig 3.3.1 Intel NUC

Fig 3.3.2 Nvidia jetson Nano

We've opted for the Intel NUC as the main computer to control all components on the robot. However, Intel NUC lacks a graphic processor which evaluates the models for human detection. Moreover, YOLO optimized by OpenVINO was already too much for its 4-core processor to load and process, so it caused a decrease in speed. Nvidia Jetson tx2 and Nano were also tested but yielded the

same results as NUC, as there are over 2 models and various large RAM-eating programs are running at the same time.
To solve this problem, we have used two computers connected through the internet. One of them is the Intel NUC, and the other is a gaming laptop equipped with a Nvidia Geforce GTX 1060 graphic processor. ROS has the ability to communicate across the internet along with the host computer and the clients. So with the help of 5G technologies, the gaming laptop is processing all models, evaluating tasks and sharing the result through the internet to the robot. This has successfully decreased the runtime of the evaluation system.
Summing things up, the robot itself still has to run 3 models, YOLO and person re-identification frequently and (waiting to be filled) when the target is lost. Since runtime speed of YOLO (optimized by OpenVINO) is the only model on the robot that would decrease CPU speed, YOLO is the only model that requires the use of GPU.

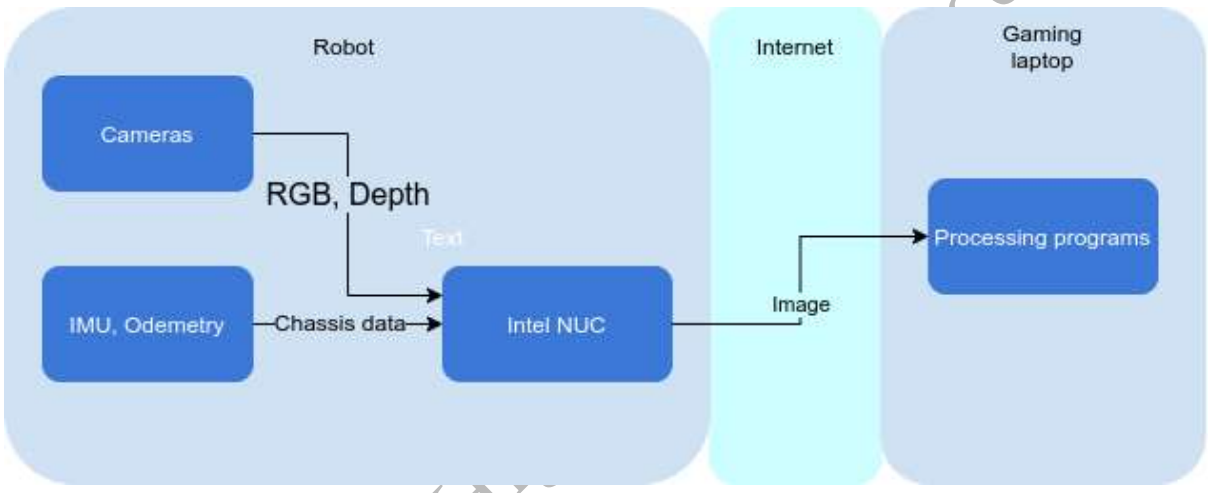


Fig 3.3.3: When there's a situation which computer vision processing is required, Intel NUC will send required information (such as image, or other details) to the main gaming laptop which act as a server, the Gaming Laptop's programs will return the result to the robot after processing

3.1.2 Environment

As to test our robot, we've set up 2 stages to test our following method. The first stage will be focusing on testing the robot's reaction to annoying persons who try to steal the main target person's position; the second stage will be focusing on testing the person re-identification reacting in the dark.

Stage 1

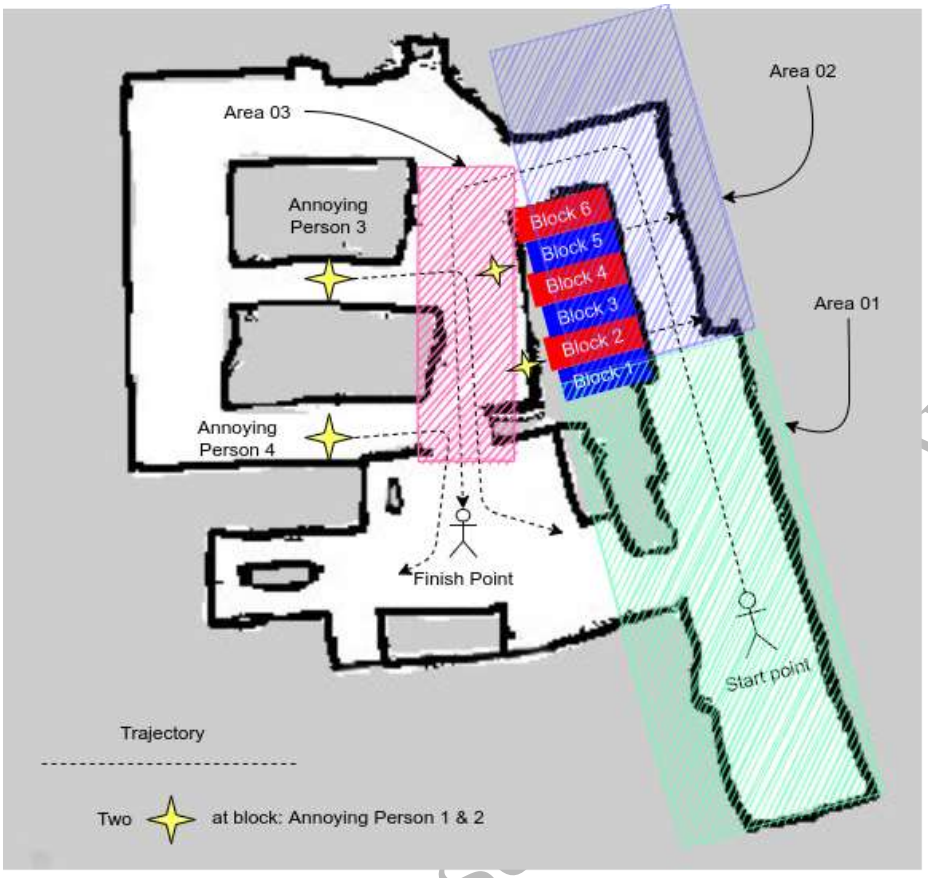


Fig 3.1.2.1 Stage 1 testing environment

As shown in the figure, Stage 1 is divided into 3 areas: Area 01: Non-blocking area; Area 02: Blocking area; Area 03: Normal conversation area. The robot is tested for its performance of following the target consistently while some non-target pedestrians occasionally pass by. Note that all lights are on throughout Stage 1.

In Area 01, the robot starts following the main target person at the Start point. Before entering Area 02, there will be no other people trying to appear before the robot, testing if its 'attention' would be drifted away; in other words, only the robot and the main target person being followed were present.

In Area 02, two testers known as the 'annoying people' try to do their job in this narrow lane by crossing between the target and the robot. The starting point of annoying person 1 & 2 will be at block m & n, generated randomly at the beginning of the test.

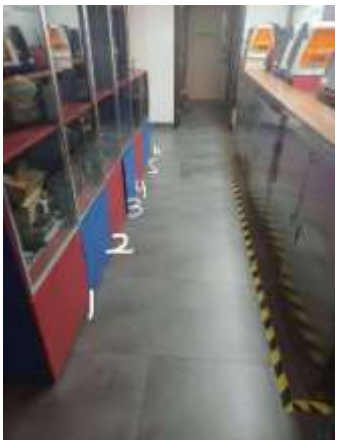


Fig 3.1.2.2 The blocks

When the main target experimenter passes by, the interference experimenters will walk between the robot and the main target experimenter. The interference experimenters will stop for 0.5 second while the main target experimenter keeps walking, blocking the view of the robot temporarily.
In Area 03, the annoying person 3 first comes in between the robot and the main target person and walks with the main target person on the left. Annoying person 4 does the same shortly after and Stage 1 is finished.
In Stage 1, we've tested the robot for 10 times and recorded if the robot successfully passed the areas. Each test lasted for approximately 1m08s. Technically, our method could recognize both sides of the main target person so they could watch the robot following them. But in this test, we hope that the main target person could let it's back open for the robot to follow them. Moreover, the system gains more accuracy for the main target person to look at it when it's following. So, if the robot requires the main target person to turn back to let the robot find it back for more than once, that area will be estimated as failed.
However, there's still a special case that our main target person could turn back to let the robot follow them: the big u turn between Area 02 and Area 03. As our robot doesn't have the obstacle avoiding system, the main target person could check on the robot when doing that u turn but not considered as one.

Stage 2

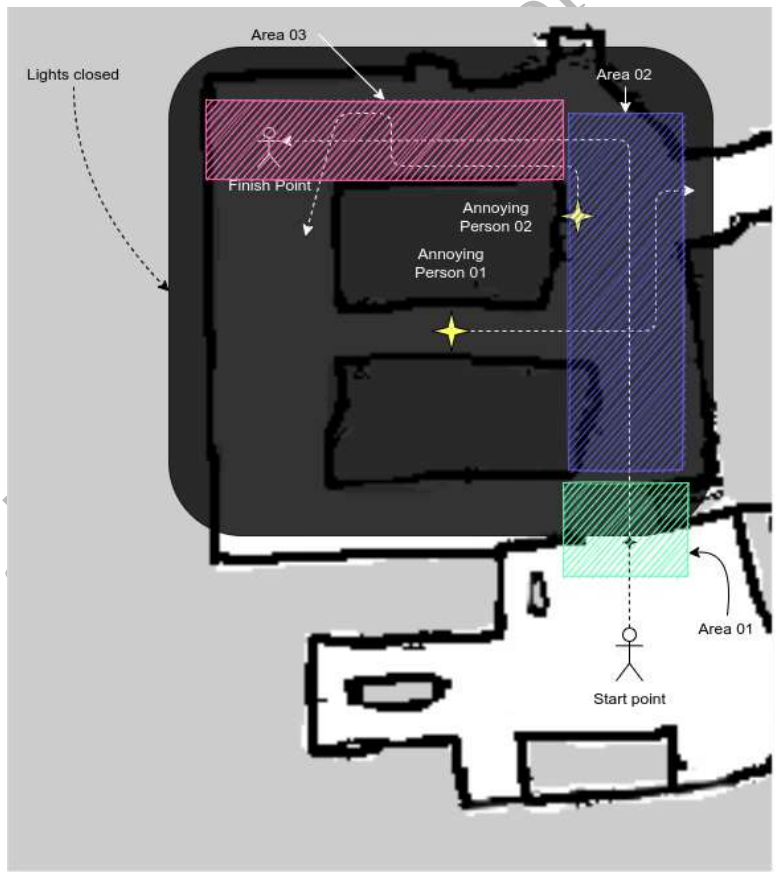


Fig 3.1.2.3 Stage 2 environment

Like Stage 1, there will still be 3 areas. In Stage 2, also the same map but without the alley. As where the finish point in Stage 1, becomes the start point in Stage 2. In the room of the Start point, the lights

were on but were closed in areas 02 & 03. By the way, area 02 will be the medium level of darkness, and area 03 will be the darkest area in the whole stage.
There will still be 10 tests and records if the robot successfully passed the areas with the same rules of failing in Stage 1. Also, the duration will be shortened into 48 seconds each as the path was shorter.

3.2 Experiment Result

Here are the test results of Stage 1. As Annoying person 1 & 2's stand point was random, the standing position displays each test's stand point

Stage 1					
No	standing position	Area 01	Area 02	Area 03	remark
1	5 6	✓	✓	✓	
2	6 3	✓	✓	✓	
3	4 1	✓	✓	✓	
4	1 6	✓	✗	✓	The robot misses at last
5	5 2	✗	✓	✓	Poor light
6	3 5	✓	✓	✓	
7	3 4	✓	✓	✓	
8	2 1	✗	✓	✓	Poor light
9	5 4	✓	✓	✓	
10	4 6	✓	✓	✓	

As shown in the table, the robot failed 2 times in Area 01 more than the others. This is because there's a large light at the top front in the robot's view, which causes the main target person to be blacked out by that. As you could see in the fig below: Our main target person was marked red because person re-identification can't recognize him.

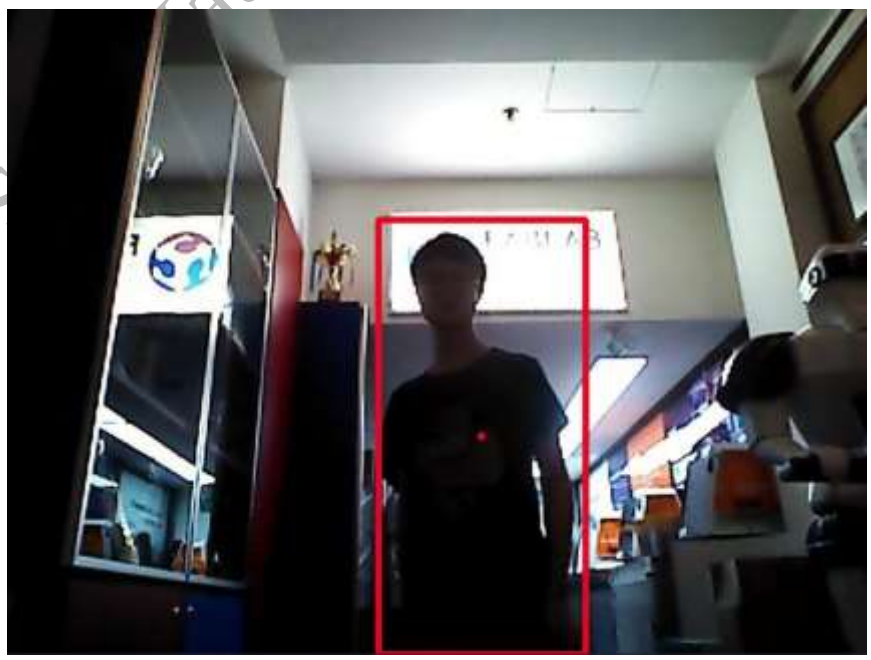


Fig 3.2.1 The poor light situation

As a result, the robot still figures out some situations that it can't deal with properly. But seeing the test result in a big picture, the robot could still deal with persons blocking easily and could keep following the main target person. From the fig below, taken from Area 03, we could see the robot could recognize both the main target person and the interference person with green and red bounding boxes even if they are walking together, proving that this method is actually 'recognizing' the target.

Fig 3.2.2 Shot in area 03 at the end

After Stage 1, here are the test results from Stage 2

Stage 2				
No	Area 01	Area 02	Area 03	Remark
1	✓	✓	✕	Can't recognize at last
2	✓	✕	✓	Dark
3	✓	✓	✓	
★4	✓	✓	✓	★： Testers will stand for a chat at Area 03
5	✓	✓	✕	Can't recognize at last
6	✓	✓	✓	
7	✓	✓	✕	Can't recognize at last
8	✓	✕	✕	
9	✓	✓	☆✓	☆： The tester has to turn around 1 time at last seconds
10	✓	✓	☆✓	☆： The tester has to turn around 1 time at last seconds

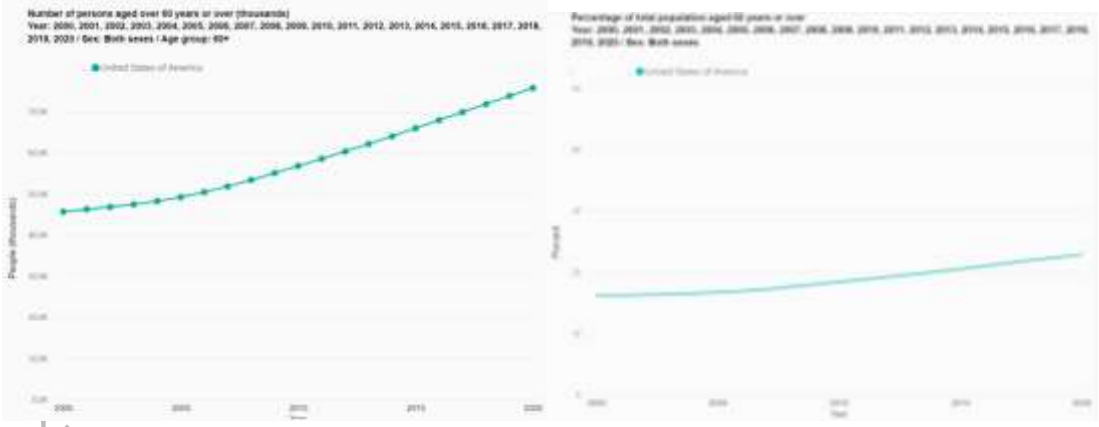
As the main target person was registered in full light conditions, environmental changes seems to have affected person re-identification performance, as there are more failures compared to Stage 1.
In conclusion, after these experiments, they claim that although our person followers may sometimes lose the target due to environmental reasons, our method will never follow the wrong person compared to the trackers and other methods, even if a long time has passed. Moreover, our method won't be bothered if another pedestrian blocks the view.

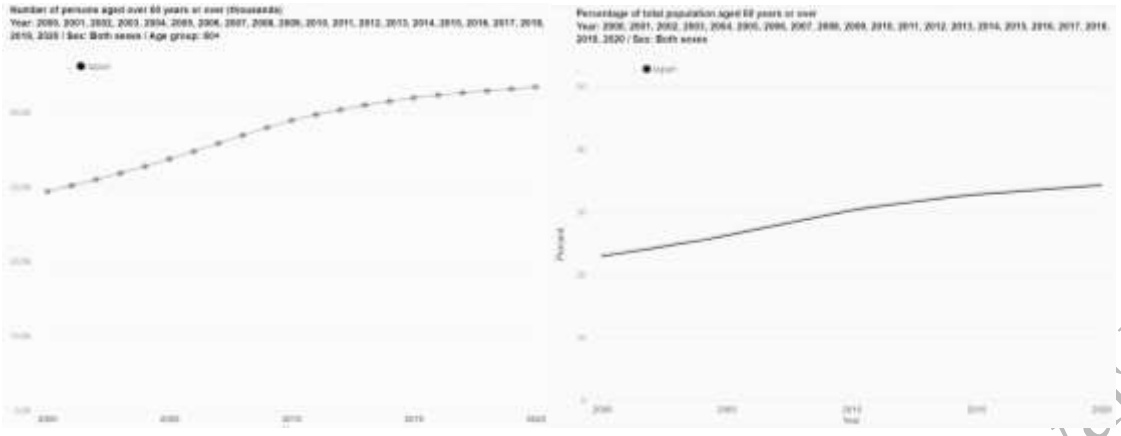
4. Conclusion

From the result of the experiment, it turns out that person re-identification technology has the ability to track the target pedestrian continuously, as results have low possibilities to miss the target.

4.1 Use case

"Aging population" is a controversial issue in recent times. For the past few years, the problem of the aging population has appeared all over the world. According to the World Health Organization[17], in 2000, the number of total elderly in the United States of America was 45652 thousand, which constituted 16.2% of the total population. Yet the number reached 75718 thousand in 2020 with the percentage increased to 22.9%. Moreover, Japan's total number of elderly in 2000 was 29382 thousand, accounting for 23% of the total population; in 2020, however, the total number of elderly rose to 43412 thousand, making up 34.3% of the total population.
Based on the "Fifth National Population Census"[18], we can also see that in 2000, there were 88110 thousand elderly in China and it was 6.96% of the total population. Whereas the "Seventh National Population Census" [19] shows that in 2020, there were about 264018 thousand elderly in China and the percentage of it was 18.7%, which nearly tripled. According to the above information, it is obvious that the number and percentage of the population proportion of the elderly will be much.





[17]

The aging population brings a lot of problems to an elderly and their families. An elderly may need to face more danger and they may get lost in certain places. As an example, during 2016, around 500 thousand elderlies got lost in China. On the other hand, their offspring might need to work so they can't take care of their parents all the time.

If we could keep track of the elderly outside on the street, then also monitor their physical status by using pose recognition, combining with alarm systems and the IOT, we could help these elderlies and their families to take care of them.

4.2 Future Improvements

The very main problem of our current system will be the poor night situation.

As the person was totally dark in this situation, person re-identification was hard to extract usable information from this blacked out person.

We currently found a method called Zero-DCE (Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement)[15], which can increase the brightness of the image to normal.

Fig 4.1 Zero-DCE used on the RAW image to increase brightness

Currently, we've tested this method and found it very compatible with our situation. But there is only one problem of Zero-DCE was: if we input a normal image to it, it will still increase the brightness and make it become worse.

Fig 4.2 An normal image brightness improved by Zero-DCE

As with this characteristic, we've to find a way to determine whether the image was in a poor light situation.

5. Videos

Comp_044_attachment_1.mp4

This video contains a test with two people. The stream shown on the screen was the robot's camera view. In the video, green bounding boxes represent the target estimated by this method, while red boxes are non-targets estimated by this method.

Comp_044_attachment_2.mp4

This video contains another test but with two differences: This video was shooted from third-person view. But it still provides the robot's camera view on the left (The window in the middle). Moreover, the annoying person in this test has the same clothing as the target person.

The box color rules were the same as Comp_044_attachment_1.

Reference

[1] Kim, M., Arduengo, M., Walker, N., Jiang, Y., Hart, J.W., Stone, P., & Sentis, L. (2018). An Architecture for Person-Following using Active Target Search. ArXiv, abs/1809.08793.
[2] Chen, B., Sahdev, R., & Tsotsos, J.K. (2017). Integrating Stereo Vision with a CNN Tracker for a Person-Following Robot. ICVS.
[3] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, & UC Berkeley (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)*
[4] Redmon, J., Divvala, S., Girshick, R.B., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788.
[5] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. ECCV.
[6] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object Detection via Region-based Fully Convolutional Networks. ArXiv, abs/1605.06409.

[7] Freund, Y., & Schapire, R. (1999). A Short Introduction to Boosting.
[8] Henriques, J.F., Caseiro, R., Martins, P., & Batista, J. (2015). High-Speed Tracking with Kernelized Correlation Filters. IEEE Transactions on Pattern Analysis and Machine Intelligence, 37, 583-596.
[9] Lukežič, A., Vojír, T., Zajc, L.Č., Matas, J., & Kristan, M. (2017). Discriminative Correlation Filter with Channel and Spatial Reliability. ArXiv, abs/1611.08461.
[10] Danelljan, M., Häger, G., Khan, F., & Felsberg, M. (2015). Convolutional Features for Correlation Filter Based Visual Tracking. 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), 621-629.
[11] Host, K., Ivasic-Kos, M., & Pobar, M. (2020). Tracking Handball Players with the DeepSORT Algorithm. ICPRAM.
[12] Izutov, E. (2018). Fast and Accurate Person Re-Identification with RMNet. ArXiv, abs/1812.02465.
[13] OpenVINO™ Documentation. (2021). https://docs.openvinotoolkit.org/latest/index.html (April 18. 2020)
[14] Quigley, M. (2009). ROS: an open-source Robot Operating System. ICRA 2009.
[15] PyImageSearch. (2018). Retrieved from https://www.pyimagesearch.com/2018/07/30/opencv-object-tracking/ (February 12, 2020)
[16] Guo, C., Li, C., Guo, J., Loy, C.C., Hou, J., Kwong, S., & Cong, R. (2020). Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1777-1786.
[17] World Health Organization. (2020). Retrieved from https://www.who.int/data/maternal-newborn-child-adolescent-ageing/indicator-explorer-new/mca/number-of-persons-aged-over-60-years-or-over-(thousands) (April 12, 2021)
[18] 2000 年第五次全国人口普查. (2000). Retrieved from http://www.gov.cn/gongbao/content/2001/content_60740.htm (April 12, 2021)
[19] 2021 年第七次全国人口普查. (2021). Retrieved from http://www.stats.gov.cn/tjsj/tjgb/rkpcgb/qgrkpcgb/202106/t20210628_1818824.html (April 12 2021)
[20] Islam, M., Hong, J., & Sattar, J. (2019). Person-following by autonomous robots: A categorical overview. The International Journal of Robotics Research, 38, 1581 - 1618.
[21] Dr. Adrian Rosebrock (2017). *Deep Learning For Computer Vision*, Philadelphia, Pennsylvania, United States : PyImageSearch
[22] François Chollet, (2018), *Deep Learning with Python*, Shelter Island, NY : Manning Publications Co.
[23] 陳孟元, (2017), *移動機器人 SLAM、目標跟及路徑規劃*, 北京 : 北京航空航天大學出版社
[24] 高翔, 張濤, 劉毅, 顏沁睿, (2019), 視覺 SL4AM 十四講，從理論到實踐, 北京: 電子工業出版社