**System Overview:**

The following diagram shows the basic three physical components of this robot, they are camera, the development board, and the car.

figure [num] The overall diagram of the system.

The camera is responsible for taking photos and passing the image data to the development board. It could be a digital camera with USB port, or an android phone with camera functionality and Wi-Fi connection.

The development board is doing image analysis and motion planning, and it is the brain of the whole system. It use XBEE device to communicate with the Car, and use Wi-Fi to communicate with the camera.

The car carries out the command given by the development board, like going forward, rotate degree, to have the next image (photo) closer to the expected one.

The user interface is a cell phone with android OS installed, running an APP written by us, it’s a bridge between the user and the robot, so that the user could know what the robot is doing, and retrieve the photo when necessary.

More detailed description of these components would be presented in the following chapter.

**Workflow of the robot**



Figure [num] workflow of the robot

Description of the workflow:

Firstly, the program will initialize and test all the devices and port, and show every possible warning, or maybe exit the program if any of the devices are missing.

Then, it will wait for user’s command. User may select different options to customize the photo to be taken.

The robot then will start to execute the pre-programmed algorithm to find a good position to take a photo for the user.

If the robot thinks it has finished position adjusting, it will inform the user and check whether user has any specific command. If the user has confirmed this picture is good enough, then the User Interface will retrieve the photo and store it in the cellphone.

**Image Analysis subsystem**

The Image Analysis subsystem is responsible for analyzing the Image taken by the camera. It’s using the very famous OpenCV[i] library’s Image processing module and object detection module to try to detect possible human object in the image, for the motion planning system to plan the following actions. However, using the default detector of the OpenCV library does not provide satisfying results (there are usually error in detection), so that we have implemented a filter to take away some unlikely results. Unfortunately, error still exists, because human detection is still a not finished research topic, and the open source detector of OpenCV cannot be as good as those commercial ones. But the robot still works as long as there are no too many distractions.

Roughly speaking, its functionality can be categorized into two kinds, one is detection, and the other is filtering.

**Detection:**

The main objective of detection is to find human object in the image. To detect a human, we can either try to detect the face of a human, or the body structure of a human. However, after some testing, either of the detection of face of detection of the body can work as accurate as we expect, so we decide to do both detections at the same time to get more information of the image, which is helpful for the filter to do filtering and a simple data fusion to finalize the detection result. And this method gives out a quite good result in terms of eliminating misdetection.



Figure [num] How the Image Analysis subsystem works

**Face Detection**

The OpenCV library provides two famous classifiers for face detection, one is Haar Cascade Classifier[ii-iv], and the other is the Local Binary Pattern Cascade Classifier. Originally we are using LBP classifier because it’s about 10% faster, however the detection comes with more errors, so we have tried Haar cascade classifier, and find it more accurate, and it’s not significantly slower than LBP. The Haar Cascade Classfier is now at use, but still LBP is reserved as an option.

**Body Detection**

The implementation of body detection employs the famous HOG (Histogram of oriented gradient) method[v] implemented by the OpenCV library. This HOG algorithm consumes a lot of computation power, so in order to have the program run faster, it’s used only when a full length portrait is required by the user.

**Filtering**

Filtering is a very important step in this Image Analysis subsystem, because the previous detection results are usually not as perfect as we have expected. Body detection may mistake a straight object as the human body, and the face detection may also make some mistakes when some regions’ haar-feature is close to the haar-feature of a face. So, the following methods are deployed to increase the accuracy of human detection.

1. Usually for a human, he/she will definitely have a face together with a body, so the first step is to filter out all bodies that are detected without a face in its body region.
2. The skin color is a very good indicator for face region, so for each face detected, check its YCrCb color representation, and determine whether it can be really classifier as a face, and discard any faces that are not in the expected YCrCb color space.



Figure [num] The raw image taken from the Camera module

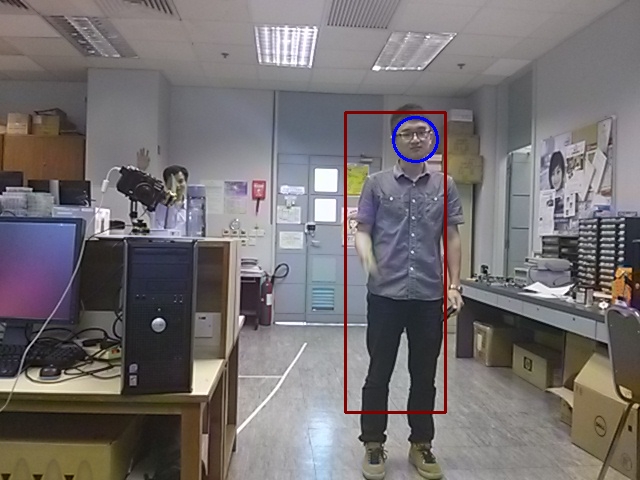


Figure [num] The image after the basic analysis

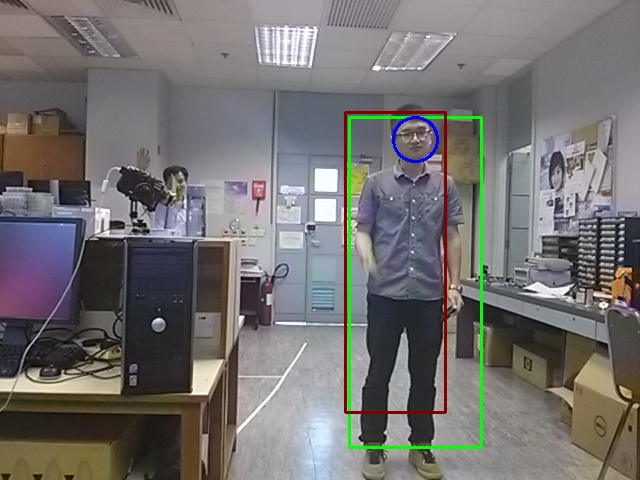


Figure [num] The image after the filtering

**Motion Planning System**

The Motion planning system is the core system that translates the image analysis result, together with the current status of the robot, into different actions commands and sends the command to the motion system (i.e. the car).



Figure [num] Motion Planning System’s input and output

The Motion Planning system is mainly responsible for adjusting the position and maybe orientation of the car to give an initial lock-on on the target, and adjusting the position of the camera after the target is detected to have the target placed in the expected position inside the photo.

In the Motion Planning system, we do estimation and computation based on the detection result from the Image Analysis system. And we have made some assumptions, like the body detected is always 170cm tall, and the face detected is always 230cm long. These assumptions provide great convenience to our robot and algorithm design, while it does not results in malfunctioning of the robot.

**Find Target**

In order to determine the initial position of the target, we have design an algorithm to utilize compass to compute what direction the target is with respect to the robot. After the initial rotation, the target should be in front of the robot. What’s more, the data from the compass is also used to filter out unlikely target.

**Adjusting Position**

We have designed the algorithm to adjust the camera position based on three different stages, centering, zooming and adjusting.

In the centering stage, the robot will adjust its position so that it can place the target at the center of the photo taken, it’s crucial for the next zooming stage, because an angle between the camera shot-line and the target will result in irregular change of the portion of the target in the image if the car is trying to zoom in or out by moving forward or backward.

After a target is at the center, the Motion Planning System will determine whether it needs to move forward or backward to make the target bigger or smaller in the scope.

Finally, when the previous stages have passed, the Motion Planning system will execute the final stage to try to place the target in the expected position, by both moving left or right, and raising or lowering the camera.



Figure [num] flow diagram of how Motion Planning Works when adjusting position for photo

[i]OpenCV is released under a BSD license and hence it’s free for both academic and commercial use, website: <http://opencv.org/>

[i] Lienhart, R.; Maydt, J., "An extended set of Haar-like features for rapid object detection," Image Processing. 2002. Proceedings. 2002 International Conference on , vol.1, no., pp.I-900,I-903 vol.1, 2002

[ii]Paul Viola and Michael J. Jones., *“*Rapid Object Detection using a Boosted Cascade of Simple Features” IEEE CVPR, 2001

[iii]Cascade Classification — OpenCV 2.4.9.0 documentation, available at <http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html>

[iv] Dalal, N.; Triggs, B., "Histograms of oriented gradients for human detection," Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on , vol.1, no., pp.886,893 vol. 1, 25-25 June 2005