CHALMERS

Stereo visual odometry using a supervised detector

Master's thesis in Systems, Control and Mechatronics & Communication Engineering

WEIMING LI
YUMENG JIANG

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2018

MASTER'S THESIS IN SYSTEMS, CONTROL AND MECHATRONICS & COMMUNICATION ENGINEERING

Stereo visual odometry using a supervised detector

WEIMING LI
YUMENG JIANG

Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2018

Stereo visual odometry using a supervised detector
WEIMING LI
YUMENG JIANG

Master's thesis 2018:09
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Visualizing the visual odometry

Chalmers Reproservice
Göteborg, Sweden 2018

Stereo visual odometry using a supervised detector
Master's thesis in Systems, Control and Mechatronics & Communication Engineering
WEIMING LI
YUMENG JIANG
Department of Mechanics and Maritime Sciences
Division of Vehicle Engineering and Autonomous Systems
Chalmers University of Technology

ABSTRACT

Stereo visual odometry, which is widely used in agent navigation because of its fast execution time and relatively good accuracy, is a process that computes the position and orientation of an agent using a stereo camera. Additionally, the method can build up a map which can be used for path planning. However, traditional stereo visual odometry only provides a content-less point cloud, making it difficult for path planning under some circumstances such as crossroads. To give content to the map, a *convolutional neural network* (CNN) classifier for detecting specific objects is created, and has been applied on the keypoints detected by *oriented FAST and rotated BRIEF* (ORB) detector. Thereafter, the labeled keypoints are inputted to a simplified keypoint-based visual odometry to build up the map. Our CNN classifier is trained on data provided by *2018 Chalmers formula student driveless* (CFSD18) and achieved 98.97% accuracy in test data. Our stereo visual odometry algorithm is tested on an elliptic track and the mapping is compared to a raw GPS sensor data as ground truth. The average processing time of the whole approach is 68.6ms per frame with a CPU-based multi-threaded algorithm. Since our keypoint detector is a combination of ORB detector and CNN classifier, which is based on supervised learning, we have called it a supervised detector.

Keywords: stereo visual odometry, convolutional neural network, ORB detector, supervised detector

ACKNOWLEDGEMENTS

First, we would like to thank our examiner Ola Benderius from the Department of Mechanics and Maritime Sciences for giving us the freedom to decide the topic of the master thesis, as well as his valuable advice and helpful support. We would also thank our supervisor Björnborg Nguyen from the Department of Mechanics and Maritime Sciences for his coaching, guidance during the whole thesis. The programming and thesis writing skill he taught us are useful for our life.

Secondly, we would specially thank Marcus Andersson from the program of Systems, Control and Mechanics and Martin Baerveldt from the program of Complex Adaptive System for their valuable data as reference for our thesis. Thanks to their open-mindedness and wisdom, it helped us better understand the limitations of our thesis. We would also want to thank all CFSD18 members for helping us set up the tests and data collections.

Finally, we would like to thank our friends and families for their care, which helped us overcome difficulties, relieve pressure, and gain courage.

Examiner: Ola Benderius, Department of Mechanics and Maritime Sciences
Supervisor: Björnborg Nguyen, Department of Mechanics and Maritime Sciences

Contents

1 Introduction

Autonomous driving is being one of the most popular topics in the world because of its potential benefits such as reduced traffic accidents and related cost, increased driving satisfaction, improved traffic system, etc. Because of the complicated circumstances, a fully automated self-driving car, which can handle all roadways and environmental conditions without human intervention, is not yet available. Most self-driving cars in the world are working under special circumstances such as highways, parking lots and traffic jams. One of the technical hurdles for the self-driving car is the software part, which consists of three main components: perception, planning and control. The perception system transforms the raw data obtained by sensors into useful information. The planning system makes decisions based on predefined policies and knowledge, as well as information provided by the perception system. The control system converts the decisions into actions. Vehicle behavior gives feedback to the perception system and forms a closed loop.

The robustness of the self-driving car relies a lot on the perception system because it directly interacts with the complicated environment. Most accidents caused by self-driving cars are due to perceptive mistakes. Thus it is crucial to create a robust and safe perception system. A variety of perceptive sensors, including *light detection and ranging* (LiDAR), radar, sonar, camera, wheel speed sensor, *inertial measurement unit* (IMU), *global positioning system* (GPS), etc., together build up a complex perception system to find paths, detect obstacles, identify traffic lights, etc. Among these sensors, LiDARs and cameras are the most important ones. LiDARs collect data in the 3D world at high speed, regardless of the lighting condition. But they are more expensive than other sensors and might have trouble in some weather conditions like heavy raining, snowing and fogging. In addition, the detecting range and resolution are limited and they have problems in detecting blocked objects. Cameras, on the contrary, are much cheaper and able to see a long distance in the daytime. With cameras, complicated things from the scene can be identified more easily than with LiDARs because of the high resolution and the capability of recognizing colors. However, the lighting variation greatly impacts the perception capability of the cameras thus high-level computer vision knowledge and intensive computing units are required. LiDARs and cameras are usually used together but in some cases only cameras are used to reduce cost.

Among the various technical hurdles in perception, "where I am" is the first question to answer. *Simultaneous localization and mapping* (SLAM) describes a situation in which both the positions of the agent and the environment are unknown, and tries to build a map and update it while keeping track of the agent's moving trajectory within it. *Visual Simultaneous localization and mapping* (vSLAM) is a SLAM solution that is based on visual information from monocular cameras, stereo cameras or RGB-D cameras. Since no expensive sensors are involved, vSLAM is very popular in low-cost applications. The drawback of the current vSLAM is that the points that make up the map, which are called landmarks, can only provide geometry information but no semantic information, which means that they may be points on the trees, grounds, buildings or other objects. This kind of semantic information may be useful for path planning algorithms to increase their performance, for example, distinguishing traffic lights to decide which direction to go.

Semantic information may be obtained by using image recognition, object detection or semantic segmentation. These methods are based on *convolutional neural network* (CNN). CNN is a multi-layer neural network that is usually used to analysis visual imagery [53]. The major advantage of CNN is that it learns the relationship between a group of pixels automatically and achieves similar performance as a combination of different filters. Thus all kinds of objects can be detected using the same method and procedure. This is very difficult or impossible to achieve by traditional computer vision techniques. CNN has shown great potential value in providing semantic information for classical vSLAM problem, but the relationship between CNN and vSLAM needs to be established.

1.1 Background

1.1.1 Visual simultaneous localization and mapping

vSLAM is inspired by basic SLAM and robot vision. Early SLAM solutions were based on data provided by odometry or laser sensors until the 1990s when robot vision started to boom. Some important topics such as *visual odometry* (VO) and *structure from motion* (SFM) were introduced meanwhile the idea of combining robot vision and SLAM became available. The first benchmark work of vSLAM was A. Davison's Mono-SLAM in the early 2000s [10]. Mono-SLAM uses keypoints to represent landmarks in the map and keeps track of them through successive camera frames to triangulate their 3D positions. This frame-to-frame matching is also used to estimate the camera pose and update the map within an *extended Kalman filter* (EKF) framework [26]. A keypoint usually contains the 2D position of an image feature and

other information like scale and orientation. It should be detected stably regardless of different conditions like viewing angles, light conditions, etc. An image feature is a small image patch that is used to compute similarities between images. Most vSLAM implementations had adopted the framework that Davison created until graph SLAM was introduced [51]. In graph SLAM, keyframes and map points are symbolized as nodes of a graph while their constraints are symbolized as edges. Because of the sparsity of the graph, graph SLAM is more computationally efficient than Mono-SLAM [14]. Thanks to loop closing and efficient map representation and refinement, large scale mapping becomes possible. However, loop closure detection is still a major research topic in vSLAM.

1.1.2 Stereo visual odometry

vSLAM has shown great value in autonomous driving. However, the high complexity and computation load make it hard to apply in projects with limited computing resources. In this sense, VO might be a good alternative. VO is the key solution to the egomotion estimation in vSLAM. It is a process that estimates the position and orientation of a camera by analyzing successive image frames. Compared to traditional odometry methods like wheel encoders and laser-based iterative closest point, VO has some advantages such as high resolution, long-range detection capability, low power consumption, good robustness and so on. Depending on the camera types, VO can be categorized as mono visual odometry which uses a mono camera, and stereo visual odometry which uses stereo camera systems. Since one mono camera cannot determine the object depth from one frame, mono visual odometry has scale ambiguity issue, resulting in complicated initialization procedure, drifting and high frequency noise. Stereo visual odometry can determine the object depth from a pair of overlapped synchronous images taken by stereo cameras, thus it is more welcomed. Depending on the visual information acquisition method, VO can be categorized as feature based method and direct method. Feature based method extracts keypoints in the image and keeps track of them in the image sequence while direct method directly uses the pixel intensity in the image sequence [56]. All VO algorithms have drifting problems because of estimation errors accumulation. vSLAM has less drifting issue since estimation errors are minimized during loop closing.

1.1.3 Convolutional neural network

CNN was inspired by biological processes in the 1960s [23]. A CNN usually contains an input layer, an output layer and multiple hidden layers, each of which usually contains convolutional layers, pooling layers, fully connected layers and activation functions [53]. It is end-to-end learning which means that the network weights are calculated by considering the input and output directly. Thus it learns the object features without much engineering knowledge, pre-processing and post-processing. CNN is widely used in image recognition, video analysis, natural language processing, etc. [53],and has shown great performance in autonomous driving. Image recognition, object detection and semantic segmentation are the major applications of CNN in perception. Image recognition is a process of identifying what is presented in the image and using a label to represent it. AlexNet [30] is the first deep learning model that outperformed the previous best image recognition algorithm in *ImageNet large scale visual recognition challenge* (ILSVRC) [25]. From then on, the champions of ImageNet challenge were occupied by deep learning based algorithms. Some famous network architecture like VGG, Inception, Resnet are still the base frameworks of the state-of-art object detection and semantic segmentation algorithms. The drawback of image recognition is that it does not localize the objects, which means that some localization method needs to be applied first to find image patches.

Object detection is a process of classifying objects with labels and localizing them with bounding boxes in the image. It is widely used to detect vehicles, pedestrians and traffic lights in perception. Since object detection needs to estimate the coordinates of the bounding boxes, it is more computation-intensive and more difficult to implement than image recognition algorithms. Besides, the annotation process is more time-consuming.

Semantic segmentation is a process of labeling every pixel in the image [55].*Fully convolutional network* (FCN) is the first deep learning algorithm that trains CNN end-to-end for semantic segmentation and is regarded as the cornerstone in this field. FCN replaces the fully connected layers of existing CNN models with convolutional layers and outputs spatial maps [17]. These maps are up-sampled to produce dense pixel-wise outputs. Most state-of-art semantic segmentation algorithms adopted this idea and developed more techniques to improve performance. Semantic segmentation is even more computation-intensive and more difficult to implement than object detection but it gives more semantic information to the perception. The segmentation result can be easily associated with the geometric information to create a semantic map.

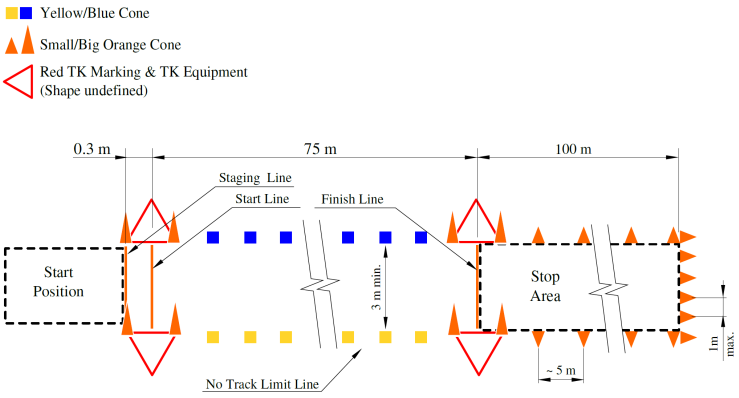
The computation loads of object detection and semantic segmentation are huge thus when the computation resource is limited, image recognition together with some localization method might be the only option to provide semantic information. As known to us, the first step of VO is to detect keypoints which propose possible object positions in the image. This gives us a hint to use CNN to classify keypoints detected by some keypoint detector like *scale invariant feature transform* (SIFT) [50], *speed up robust feature* (SURF) [1] or *oriented FAST and rotated BRIEF* (ORB) [45], to provide semantic information in VO.

1.1.4 Formula student Germany driverless competition

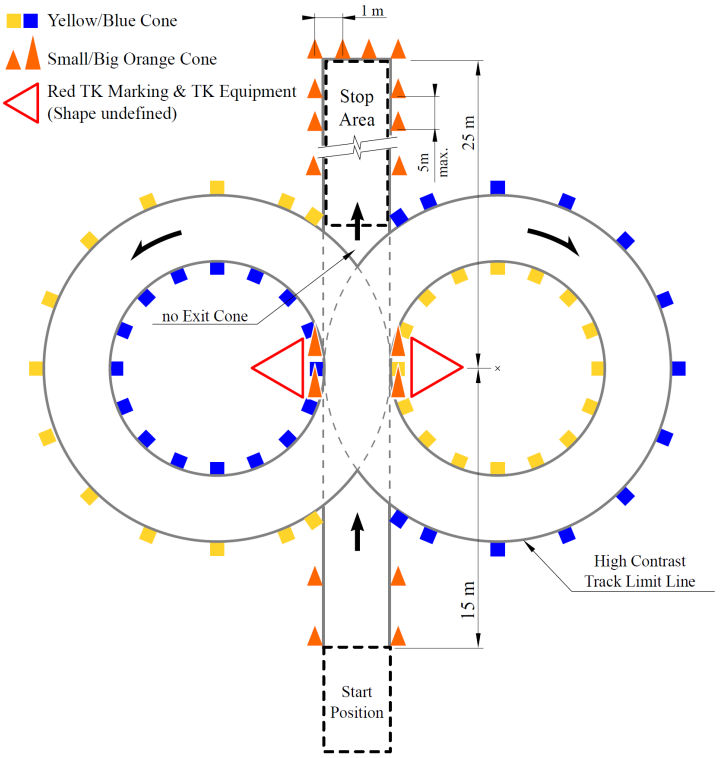
The potential benefits of autonomous driving attract automotive companies, universities and research institutes to get involved. *Formula student germany* (FSG), the largest formula student competition on the world-wide scale, even created a new branch for autonomous driving in 2017. In the same year, in order to participate in the FSG driverless 2018 competition, Chalmers University of Technology decided to assemble a new formula student driverless team, *2018 Chalmers formula student driveless* (CFSD18) team. The main task for FSG driverless competition is to develop an autonomous system to drive the vehicle automatically in paths bounded by two curvatures of cones, which are of the same shape but with different colors and sizes as shown in Figure 1.1. Blue cones specify the left side of the track while yellow cones specify the right side. Small orange cones indicate the starting and ending area while big orange cones are used for lap counting. Teams lose points if their cars hit the cones and all points are invalid if the car goes out of the track.

Figure 1.1: *Cone specification in FSG competition.*

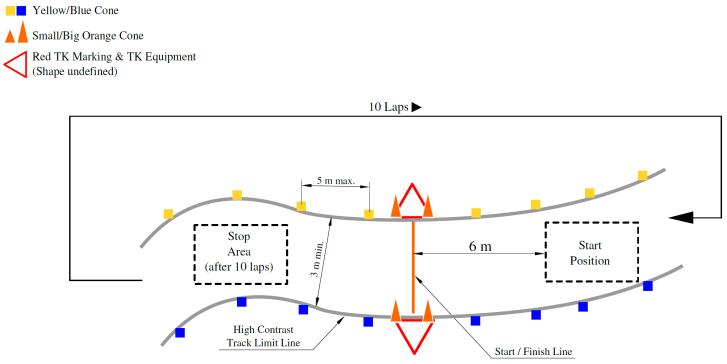
There are three different tracks corresponding to three different competition events: acceleration, skidpad and trackdrive. The acceleration track is a 75m straight track as shown in Figure 1.2a. The skidpad track consists of two pairs of concentric circles in a figure of eight pattern as shown in Figure 1.2b. The trackdrive layout is a closed loop circuit with turning and straight sequences as shown in Figure 1.2c. The team gets the highest score if the least time is used and fewest cones are hit during the event.



(a) *Track layout for acceleration event.*



(b) *Track layout for skidpad event.*



(c) *Track layout for trackdrive event.*

Figure 1.2: *Track layouts for different competition events.* [16]

1.2 Aim

Since different kinds of cones have different usages in the FSG driverless competition, it is important to know not only the cone positions but also the cone types. To ensure the car staying in the track, a localization and semantic mapping solution is preferred. The purpose of this project is to propose a stable, efficient and accurate solution to the classical localization and mapping problem and to build a map with cone positions and colors. Our research questions are listed below:

(1) How can the algorithm be designed to deal with the challenges of different background, weather conditions, lighting conditions, etc.?

(2) How to achieve good execution time relevant for live systems? The speed of the algorithm should be faster than 10 *Frames Per Second* (FPS).

(3) How to achieve a high classification accuracy? More specifically, all classes' precision, recall and F_1 score should be higher than 95% in test data.

1.3 Limitations

Although all kinds of sensors like LiDAR, wheel speed sensor, GPS and IMU could be used in our thesis, but to make use of those sensors together with the stereo camera, sensor fusion would be required. It is outside the scope of this thesis, therefore only a stereo camera system is considered in our work.

Since a high computation load is expected in CNN classification, *graphics processing unit* (GPU) is recommended because it is designed for intensive computing and much faster than general CPUs. However, GPU has not been integrated in our work at the moment and may become future work.

2 Method

Generally, our approach can be divided into three main parts: stereo vision, supervised detector and visual odometry. Stereo vision converts the 2D image information into the 3D geometry information for the supervised detector and visual odometry. The supervised detector detects and classifies keypoints which are then used to build the semantic map and estimate the camera's position and orientation by visual odometry. In this chapter, basic knowledge and technical algorithms that are covered in our approach will be given. Firstly, a brief introduction to the design flowchart of the whole visual system is shown in section 2.1. Then the basic camera geometry is introduced in section 2.2, followed by stereo vision in section 2.3, supervised detector in section 2.4 and visual odometry in section 2.5.

2.1 Design flowchart

The design flowchart shown in Figure 2.1 illustrates the workflow of the whole visual system. The input to the system is a flow of stereo images, which contains two images that are captured by two camera sensors at the same time. The first step of the workflow is to split the stereo image into left and right images, which are referred to as a *stereo pair*. Then the stereo pair is rectified in order to simplify the calculation of the disparity map, which will be illustrated in section 2.3.3. Afterwards, *block matching* (BM) algorithm [49] is used to compute the disparity map. Then a depth map is computed based on the principle of triangulation. So far, the bridge between the 2D coordinate system and the 3D coordinate system is built. ORB keypoint detector is then used to detect keypoints in the rectified image. After some post-processing, keypoints are divided into different groups in the 3D coordinate system meanwhile non-of-interest keypoints are filtered out. These keypoint groups infer possible locations of the cones. To further classify the keypoint groups, the center point of each keypoint group is computed and an image patch centered at it is extracted and classified by a CNN classifier. The procedure of the supervised detector is from the ORB keypoint detection, post-processing to CNN classification. So far the keypoints of one frame are detected and classified. These steps are repeated for the next frame. Keypoints between two frames are matched using *brute force* (BF) matcher and the transformation matrix between them are estimated. This

transformation matrix is used to build and fuse landmarks. Landmarks are the labelled points that represent the map. Finally, *bundle adjustment* (BA) is used to reduce estimation error between frames. The output of the visual system is the camera position and local map for the current frame. The camera trajectory and the global map are built incrementally by these steps.

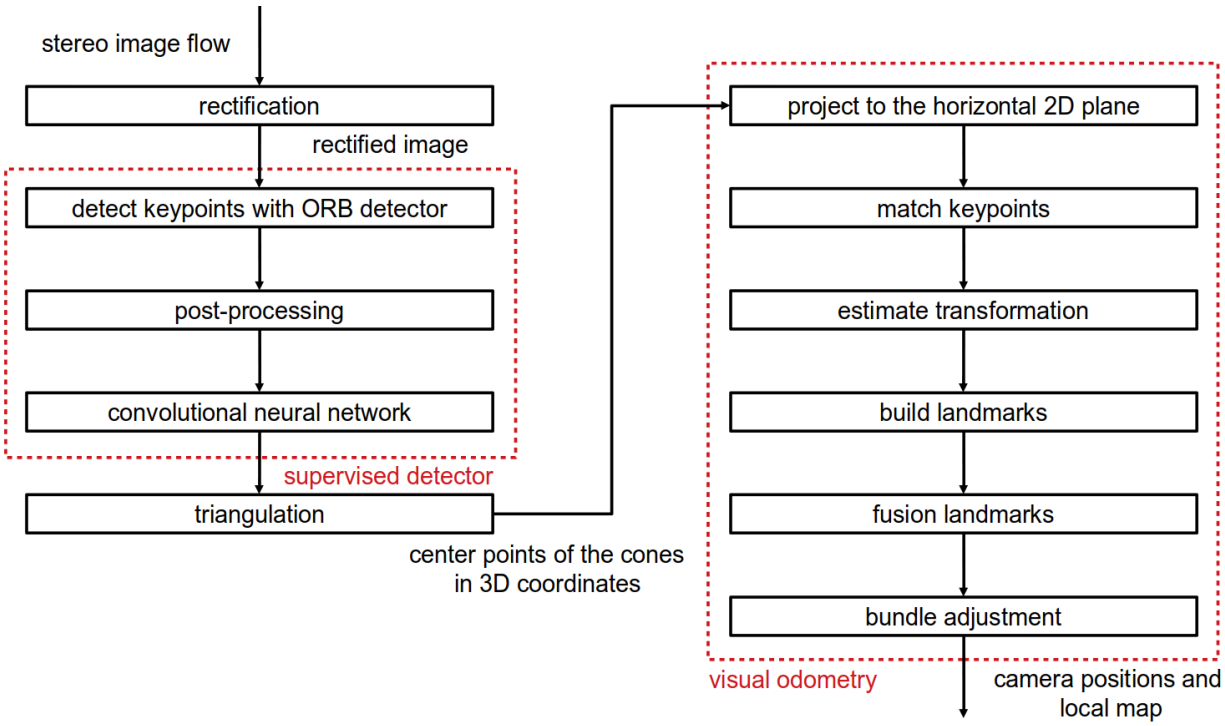


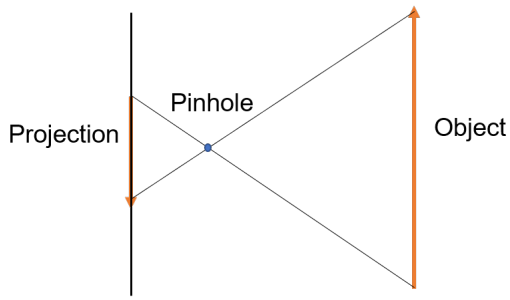
Figure 2.1: *Flowchart of the whole visual system. The input to the system is stereo image flow, the output is the camera position and local map for the current frame. The whole flowchart can be divided into three main parts: stereo vision, supervised detector, and visual odometry. The supervised detector and visual odometry are marked as red dotted lines in the flowchart. Those that are not marked with lines belong to stereo vision category.*

2.2 Basic camera geometry

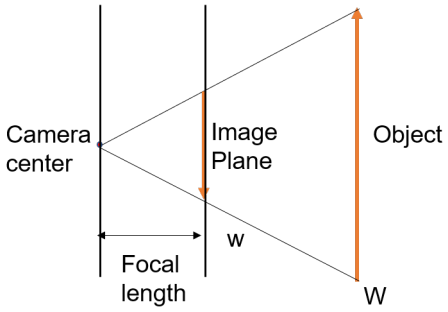
In order to better understand the knowledge of stereo vision, supervised detector and visual odometry, some fundamental concepts in camera geometry theory such as the pinhole camera model, three different coordinate systems and the relationships between them will be introduced in this section.

2.2.1 Pinhole camera model and coordinate system

When the light goes through a pinhole into a dark room, it will project an upside-down image of the outside on the wall. Similarly, the pinhole camera model is the mathematical relationship between a point in the 3D coordinate system and its projection on the image plane through a pinhole. Figure 2.2a visualizes the basic principle of the pinhole camera model.



(a) *This figure shows the principle of pinhole camera model in the real world. When the light goes through a pinhole, it will project an upside-down image on the camera plane.*



(b) *This figure shows the principle of the practical pinhole camera model where the image plane is in front of the pinhole.*

Figure 2.2: *Real pinhole camera model and practical pinhole camera model.*

However, it is more practical to place the image plane in front of the pinhole to simplify the calculation and analysis. As shown in Figure 2.2b, the distance between the camera center and the image plane is called the focal length. W is defined as a vector from the camera center to the object and w is defined as a vector from the camera center to the projection of the object in the image plane. The relationship between w and W is

$$\lambda w = W, \quad \lambda > 0 \tag{2.1}$$

Here λ is the depth of W. In order to compute the image coordinate of w, it is necessary to establish the relationship between three coordinate systems: camera coordinate system, image coordinate system and global coordinate system.

2.2.2 Relationship between the image coordinate system and camera coordinate system

To transfer the coordinate between image coordinate system and camera coordinate system, the relationship between these two coordinate systems needs to be established. Assuming there is a point in the 3D coordinate system $P(X, Y, Z)$ and a point on the image plane $p(x, y)$. Assuming the global coordinate system and the camera coordinate system coincide, the origin of camera coordinate system is the camera center, x-axis and y-axis are parallel to the image plane while z-axis is perpendicular to the image plane.

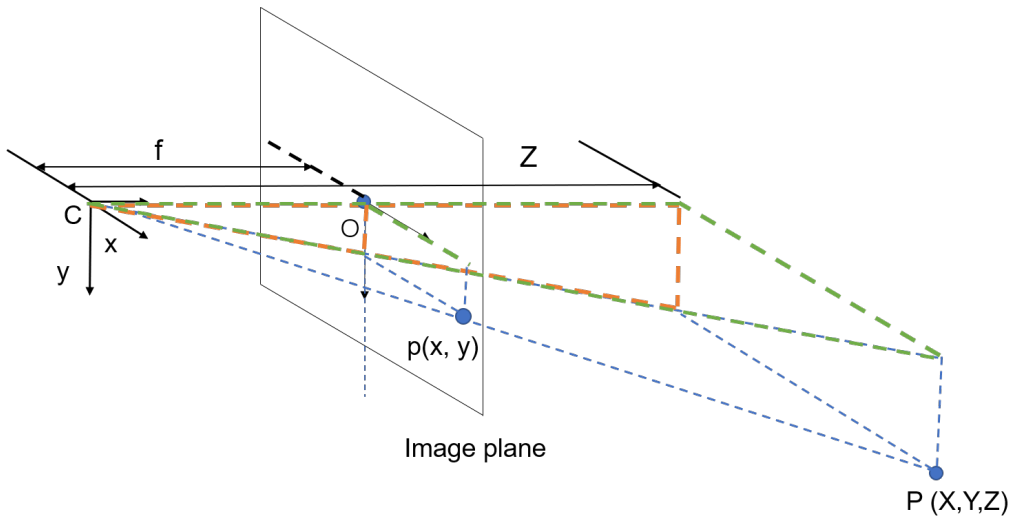


Figure 2.3: *When the camera coordinate system and global coordinate system coincide, the 3D point P and its projection p are the corresponding vertexes of two similar triangles, which infers the calibration matrix.*

As shown in Figure 2.3, Z is the perpendicular distance between P and the camera center C. O is the projection of the

camera center in the image plane. It is in the middle of the image plane and is defined as the principal point. Noticing that triangle POC and triangle $PO'C$ are similar triangles, which implies that[21]:

$$\frac{f}{Z} = \frac{x}{X} = \frac{y}{Y} \tag{2.2}$$

$$x = \frac{fX}{Z}, \;\; y = \frac{fY}{Z} \tag{2.3}$$

Here the principal point is assumed as the origin point of the camera coordinate system. But according to the image processing regulation, the origin point is assumed to be the top-left corner of the image. So a coordinate translation is required. Defining the pixel coordinate of the principal point as (c_x, c_y), then the image coordinate of p is:

$$x = \frac{fX}{Z} + c_x, \;\; y = \frac{fY}{Z} + c_y \tag{2.4}$$

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{2.5}$$

$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$ is defined as the calibration matrix[21], where f, c_x and c_y are often provided by the camera manufacturers. Equation 2.5 is considered as the bridge between the image coordinate system and the camera coordinate system.

2.2.3 Relationship between the global coordinate system and camera coordinate system

The global coordinate system of this project is defined as the initial camera coordinate system and is not changed when the camera is moving. The global map is built on the global coordinate system, i.e., the initial camera coordinate system. Landmarks starting from the second frames are projected back to the first frame to update the global map. To transfer coordinates between the camera coordinate system and the global coordinate system, the relationship between two coordinate systems needs to be established. As shown in Figure 2.4, the rotation matrix between the global coordinate system and the camera coordinate system is R. The definition of R is mentioned in section 2.5.3. The translation vector between the two coordinate systems is T. Assuming the global coordinate of a point is U, then its camera coordinate is $\begin{bmatrix} R & T \end{bmatrix} \cdot U$. $\begin{bmatrix} R & T \end{bmatrix}$ is defined as the transformation matrix between the camera coordinate system and the global coordinate system.

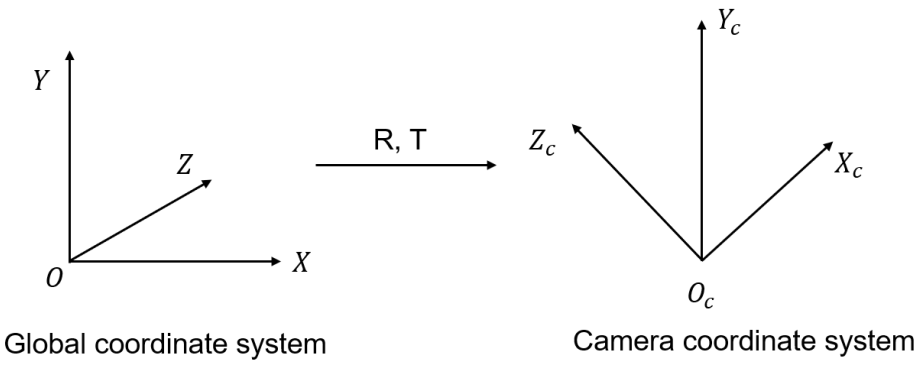


Figure 2.4: *The relationship between the camera coordinate system and the global coordinate system.*

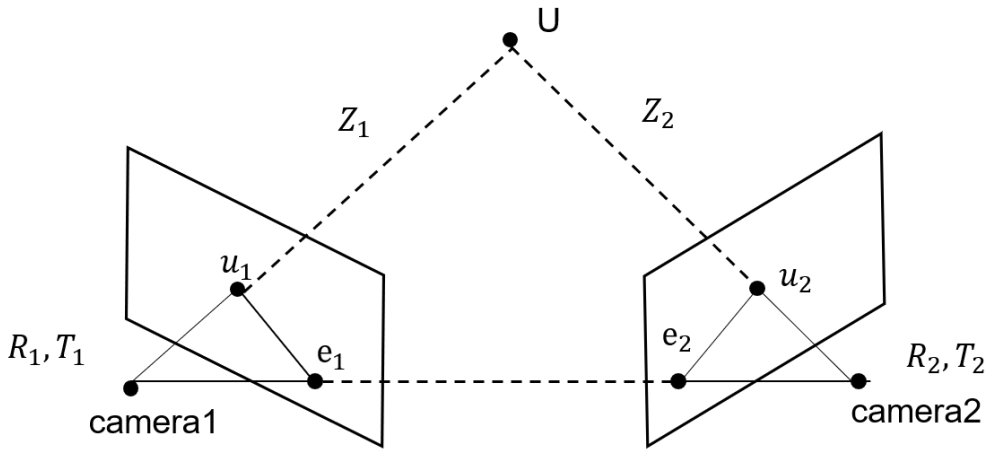


Figure 2.5: *Epipolar geometry is used to derive the transformation matrix between two camera coordinate systems*

The transformation matrix between two arbitrary camera coordinate systems can be derived from epipolar geometry [21]. As shown in Figure 2.5, u_1 is the projection of U in camera 1 and u_2 is the projection of U in camera 2. The perpendicular distances between U and two camera centers are Z_1 and Z_2. Assuming both cameras have the same calibration matrix K. The transformation matrix between the global coordinate system and two camera coordinate systems are $\begin{bmatrix} R_1 & T_1 \end{bmatrix}$ and $\begin{bmatrix} R_2 & T_2 \end{bmatrix}$, then[21]

$$Z_1 u_1 = K(R_1 U + T_1) \tag{2.6}$$

$$Z_2 u_2 = K(R_2 U + T_2) \tag{2.7}$$

Assuming $R_1 = I,\ \ T_1 = 0,\ \ R = \frac{R_2 \cdot Z_1}{Z_2},\ t = \frac{T_2}{Z_2}$, then it can be concluded that

$$u_2{}' t \times R u_1{}' = 0 \tag{2.8}$$

Equation 2.8 is called epipolar constraint. $E = t \times R$ is defined as the essential matrix.

$$u_2{}' E u_1{}' = 0 \tag{2.9}$$

More details about the derivation of the essential matrix can be found in appendix B. Given an essential matrix, the position and orientation of the camera can be calculated by methods such as five-point algorithm [38].

2.3 Stereo vision

A stereo camera contains two independent image sensors on the base and captures two images simultaneously, which are called the stereo pair. Figure 2.6 shows an example stereo image that is taken by the stereo camera used in this project. Stereo vision is the process of determining the depth from a stereo pair. Since the depth can be calculated from a single frame, stereo visual odometry does not have the scale ambiguity issue that exists in mono visual odometry.

Rectification, disparity map calculation and triangulation are the basic steps in stereo vision. Triangulation is the fundamental concept in stereo vision thus it will be discussed first. Disparity map calculation and rectification are introduced afterwards.

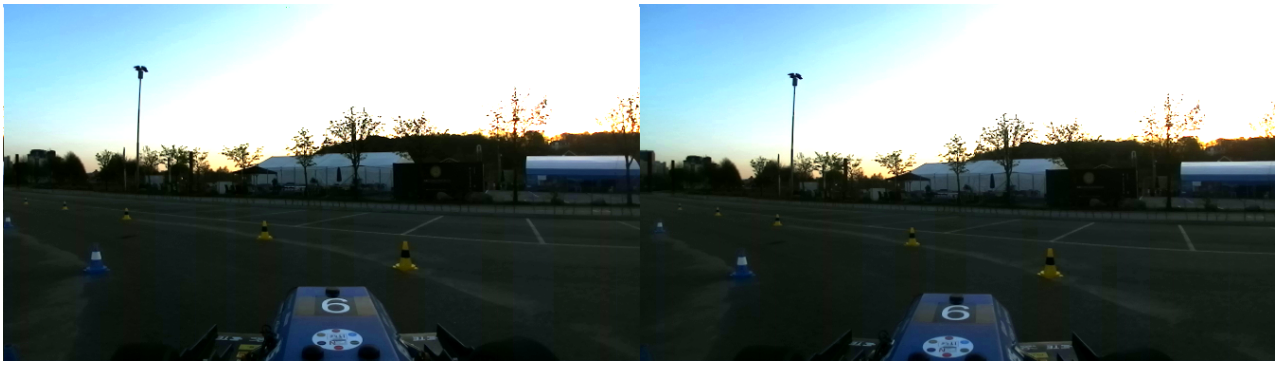


Figure 2.6: *A stereo image contains left and right images that are captured by two image sensors at the same time.*

2.3.1 Triangulation

Triangulation is the process of determining the position of a 3D point from its projections in a stereo pair. Disparity map calculation and rectification are based on this principle. Figure 2.7 shows the basic idea of triangulation.

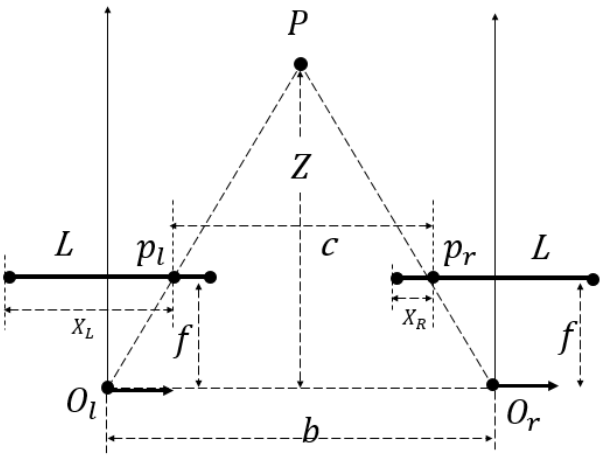


Figure 2.7: *The relationship between a 3D point P and its projections P_l and P_r in a stereo pair. The depth Z can be calculated by triangulation.*

Assuming O_l is the center point of the left camera and O_r is the center point of the right camera. The projection from a 3D point P to two image sensors are defined as P_l and P_r. Z is the depth of P. b is the distance between two camera sensors and is called the baseline. c is the distance between P_l and P_r. f is the focal length. L is the width of the image. X_L is the image coordinate of P_l and X_R is the image coordinate of P_r. Assuming P_l and P_r are at the same row in the image, then the coordinate distance between P_l and P_r is $d = X_R - X_L$. Here d is a scalar called disparity. Noticing that triangle PP_lP_r and triangle PO_lO_r are similar, so that

$$\frac{c}{b} = \frac{Z-f}{Z} = 1 - \frac{f}{Z} \tag{2.10}$$

It can be concluded that

$$Z = \frac{fb}{b-c} = \frac{fb}{X_R - \frac{L}{2} + \frac{L}{2} - X_L} = \frac{bf}{X_L - X_R} = \frac{bf}{d} \tag{2.11}$$

Equation 2.11 is called the triangulation function where b and f are known from the camera manufacturer. Given the disparity d, Z is determined by the triangulation function.

2.3.2 Disparity map calculation

There are a lot of methods for calculating the disparity map such as *semi-global block matching* (SGBM) and *block matching* (BM). Grant has compared the performance of BM and SGBM and concluded that BM is much faster than SGBM while their accuracy are similar [31], thus BM is used in our project. Figure 2.8 shows the principle idea of BM. Given an image patch in the right image of a stereo pair, BM tries to find out the best match in the left image by comparing the absolute image intensity difference in grayscale. Assuming the stereo pair is already rectified, BM algorithm only needs to search for the image patches on the same row and the disparity is the column pixel difference between the centers of the matched image patches.

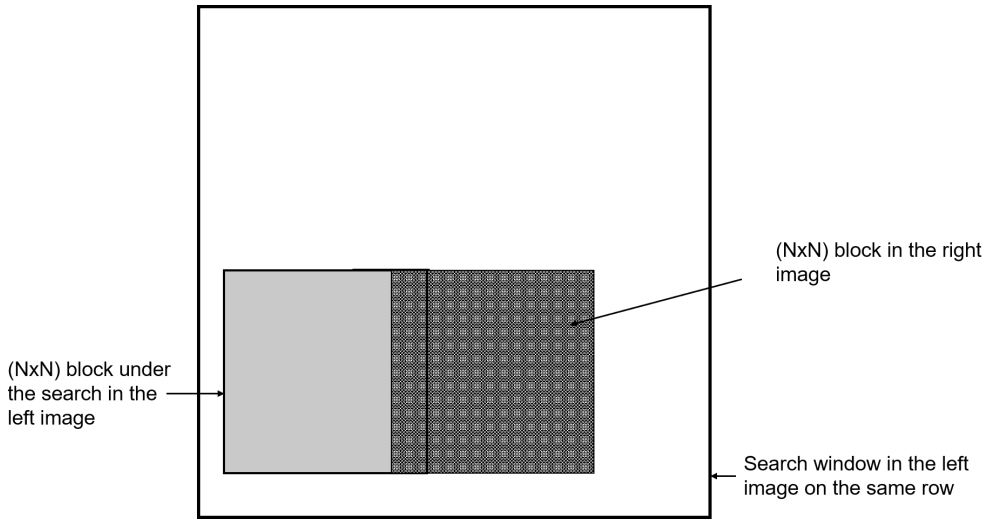


Figure 2.8: *This figure shows how BM finds the best matches in a stereo pair in order to calculate the disparity.*

2.3.3 Rectification

To compute the disparity of one pixel, its corresponding pixel in the other image needs to be found in a line called epipolar line. If the images are coplanar, all pixels' epipolar lines are parallel to the horizontal axis. Then the disparity is the horizontal offset between two pixels in the stereo pair. When images are not coplanar, a transformation process is required. Image rectification is a transformation process that projects images to one common image plane, so that the corresponding points in the images have identical row coordinates [54]. This can simplify the computation process of the disparity map because the searching area is reduced to one dimension. Figure 2.9 shows the principle concept of rectification. The projection from the 3D point U to the stereo pair are defined as x_1 and x_2. The intersection points between the baseline and two image planes are called epipoles, they are defined as e_1 and e_2 in Figure 2.9. l_1 and l_2 are the epipolar lines. Assuming the image pair are not coplanar, the epipolar lines are not parallel. But after rectification, l_1 and l_2 are parallel to the horizontal axis. Besides, e_1 and e_2 are mapped to the infinite points $e_1^{'}$ and $e_2^{'}$.

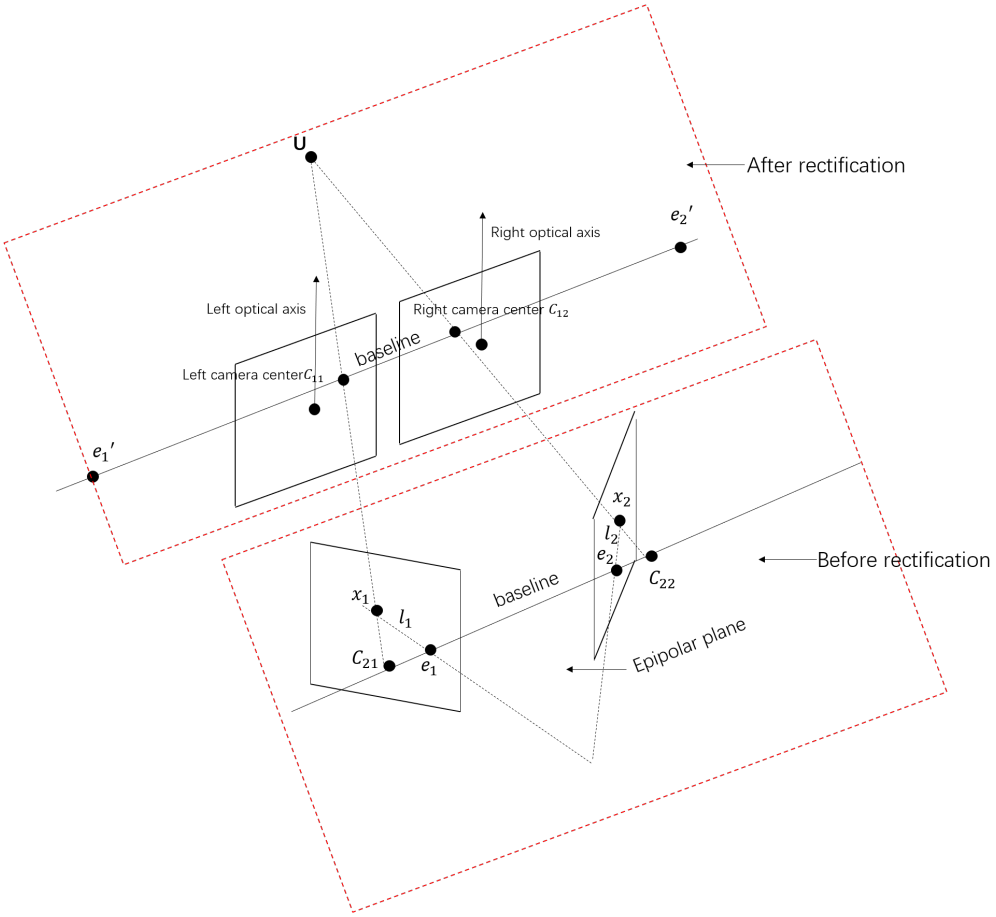


Figure 2.9: *Basic concept of the rectification. Epipolar lines l_1 and l_2 are not parallel before rectification. But after rectification, l_1 and l_2 are parallel to the horizontal axis. e_1 and e_2 are mapped to the infinite points $e_1^{'}$ and $e_2^{'}$*

Figure 2.10 shows the rectification result of a stereo pair. It can be seen that the corresponding points in the stereo pair have identical row coordinates.

Figure 2.10: *After rectification, the corresponding points in a stereo pair have identical row coordinates.*

2.4 Supervised detector

The general workflow of the supervised detector is shown in Figure 2.11. Firstly, keypoints in a rectified RGB image are detected by ORB detector. Afterwards, non-of-interest keypoints are filtered out by their 3D coordinate. Then keypoints are separated into different groups by their 3D distance. Finally, the image patches centered at the center points of the groups are extracted and classified by a CNN classifier. The output of the supervised detector is the image coordinates and labels of the center points.

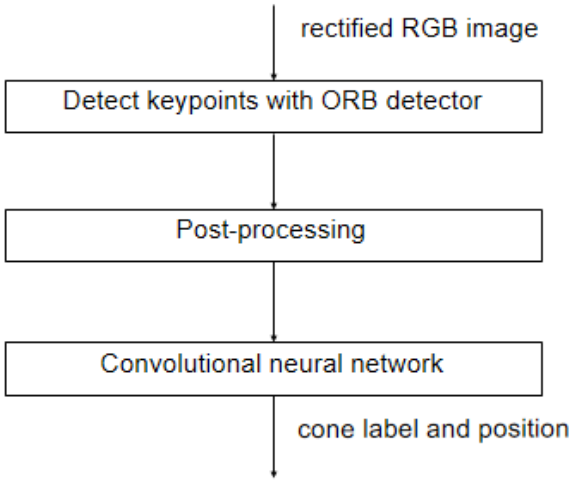


Figure 2.11: *Flowchart of the supervised detector. The input to the supervised detector is a rectified RGB image, the output is the image coordinates and labels of the cones.*

2.4.1 ORB detector

Ebrahim *et al* compared the performance of three keypoint detectors (SIFT, SURF and ORB) for different kinds of transformations and deformations in the images. In the end, they concluded that ORB is the fastest keypoint detector while SIFT performs the best in most scenarios [28]. They also found that the keypoints detected by ORB are mostly centred at the objects while the others are more distributed over the whole image [28]. Because of these various advantages of ORB, it is chosen as the keypoint detector in this project.

ORB is a fusion of *features for accelerated segment test* (FAST) keypoint detector [44] and *binary robust independent elementary features* (BRIEF) descriptor [7] with some modifications. The main idea of FAST detector is to compare the selected point with its surrounding points and if most of its surrounding points' intensity differ a lot from it, it is considered to be a keypoint. The drawback of FAST is that it is orientation variant, but ORB finds a way to define the orientation of the keypoints by calculating the intensity centroid [45]. Besides, ORB uses Harris corner measure to find top N keypoints [20] to filter out the noise and save time for keypoint matching. The main idea of BRIEF descriptor is to select n pairs of points around the keypoint in a unique way [7] and compare the pixel intensity of each pair. The comparison results form an n-dimensional bitstring, which is the descriptor of the keypoint. Similar to FAST, BRIEF also has rotation issue. To solve this, ORB computes a rotation matrix using the orientation of the patch and lets the BRIEF descriptors rotate according to it [28].

Since cones are usually quite different from the background, ORB detector is able to find keypoints inside the cones. These keypoints propose possible positions of the cones. So the CNN classifier can only work on these proposed regions, which are called *regions of interest* (ROI), to save computation resource. Figure 2.12 shows the ORB detection result of an example frame in the testing track. The white spots are the keypoints detected by ORB detector and they are almost all inside the cones. Only a few keypoints hit the ground and they can be filtered out by post-processing and the CNN classifier. Besides, keypoints are almost symmetrically located on the cones, so it is reasonable to take the mean coordinate of the keypoints to represent the center point of the cone. To save computation load, ORB detector only detects

the middle part of the image (the area between two red lines in Figure 2.12) where most cones are located.

Figure 2.12: *ORB detection result of an example frame in the testing track. The white spots in the image are keypoints detected by ORB. The detection is limited within the region of interest (the area between the two red lines), where most cones are located.*

2.4.2 Post-processing

From Figure 2.12, it is clear to see that there are a lot of keypoints on the same cone, it is unnecessary to run the CNN classifier on all of them. So some post-processing algorithms have been applied to filtered out non-of-interest keypoints and then the representing point of each cone are found. The flowchart of the post-processing is shown in Figure 2.13. The inputs to the post-processing are the keypoints detected by ORB detector. Since the heights of the cones are within a certain range, those points that are outside the range can be considered as outliers and be filtered out.

After this simple filtering, there are still a lot of keypoints belonging to different cones. So the next step is to separate keypoints into different groups. K-d tree nearest neighbor search algorithm [2] is applied to speed up the grouping process. The basic idea of k-d tree is to give a special index to each keypoint so that it can find its closest keypoints faster. The complexity for k-d tree is $O\left(log\, n\right)$, where n is the total number of keypoints [2]. It is better to apply k-d tree in the 3D coordinate because cones can be overlapped in the image but they are at least 1m away from each other in the 3D coordinate. To project the 2D image coordinate to the 3D coordinate, Equation 2.5 is used. The grouping process is as follows: First, keypoints are projected to the 3D coordinate. Then for each point, search for its neighbor points within a certain radius. If it doesn't have any neighbor points or all its neighbor points have already been grouped, then mark it with a new group id. Otherwise, all its neighbor points that have not been grouped are marked with a new group id. When looking for the next point, only those points that have not been grouped are of interest.

After grouping points, the mean of the 3D coordinates is calculated and referred to the center point of each group. The 3D coordinate of the center points is then projected back to the image using Equation 2.5. The last step is to extract image patches centered on the center points, resize to the same size and run the CNN classifier on them. The challenge is how to decide the size of the image patch. The image patch should cover the whole cone in the image. But because of the perspective transform, cones closer to the camera are larger in the image. However, since all cones have similar size in the 3D world, the image patch size can be computed from the 3D size and the 3D position of the cone using Equation 2.5.

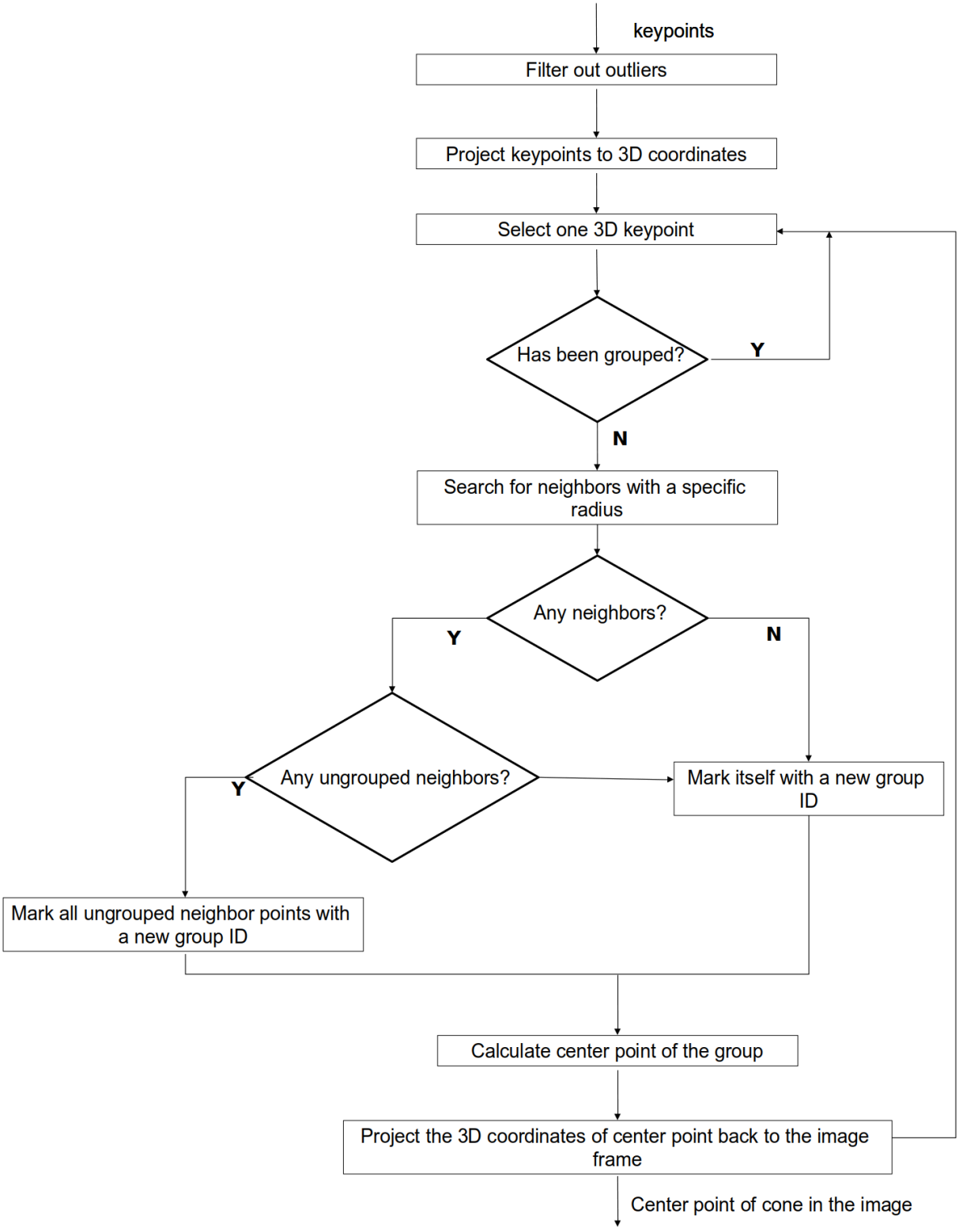


Figure 2.13: *The flowchart of post-processing. The input is keypoints detected by ORB detector, the output is center points of cones in the image.*

2.4.3 CNN classifier

A CNN architecture is created and it takes an image patch as input and outputs a recognition score that describes what kind of cone it is. As shown in Table 2.1, this is a *visual geometry group* (VGG)-style [47] neural network architecture. The network uses convolutional layers with stride 2 to reduce the dimensionality of the image. Every convolutional layer follows by a tanh activation function [27] to introduce non-linearity to the network. Dropout layer [48] is used after the tanh activation function to avoid overfitting [12]. Convolutional layer, tanh activation function and dropout layer together form a visual geometry group. After four these groups is a fully connected layer with a leaky-relu [36] activation function. In the end, a fully connected layer follows by a softmax activation function [22] is used to output a recognition score. Since this is a multi-class recognition problem, the loss function is categorical crossentropy [13]. The optimization algorithm is *adaptive moment estimation* (Adam)[29] which combines the advantages of two other stochastic gradient descent methods [4], *adaptive gradient algorithm* (AdaGrad) [46] and *root mean square propagation* (RMSProp) [24].

Table 2.1: Convolutional neural network architecture, where conv stands for convolutional layer; tanh, tanh activation function; dropout, dropout layer; fc, fully connected layer; leaky relu, leaky relu activation function; softmax, softmax activation function.

input: $64 \times 64 \times 3$
conv+tanh: $3 \times 3 \times 16$, stride=2
dropout
conv+tanh: $3 \times 3 \times 16$, stride=2
dropout
conv+tanh: $3 \times 3 \times 32$, stride=2
dropout
conv+tanh: $3 \times 3 \times 32$, stride=2
dropout
fc+leaky relu: 128
fc+softmax: 5

Precision, recall and F_1 score [42] are used as metrics of the recognition accuracy. The precision of a class (let's say class A) is $\frac{|T \cap P|}{|P|}$, recall is $\frac{|T \cap P|}{|T|}$, and F_1 score is their harmonic mean, which is $\frac{2 \times Precision \times Recall}{Precision + Recall}$ [6]. Here T stands for the true set, whose elements are from class A. $|T|$ is the total number of true set. P stands for the positive set, whose elements are classified as class A. $T \cap P$ stands for the true-positive set, whose elements are correctly classified as A. Precision and recall are usually used together to give a proper comparison of the accuracy. F_1 score can give an overall comparison of the accuracy for each class, thus it is more convenience.

2.5 Visual odometry

Figure 2.14 shows the pipeline of visual odometry. Keypoints detected by the supervised detector are first projected from the 3D coordinate to the 2D horizontal plane due to essential decomposition, which is discussed in subsection 2.5.1. Then keypoints from two frames are matched and a rigid transformation matrix between frames is estimated. This transformation matrix is used to build and fuse landmarks. Finally, bundle adjustment is used to reduce estimation error between frames.

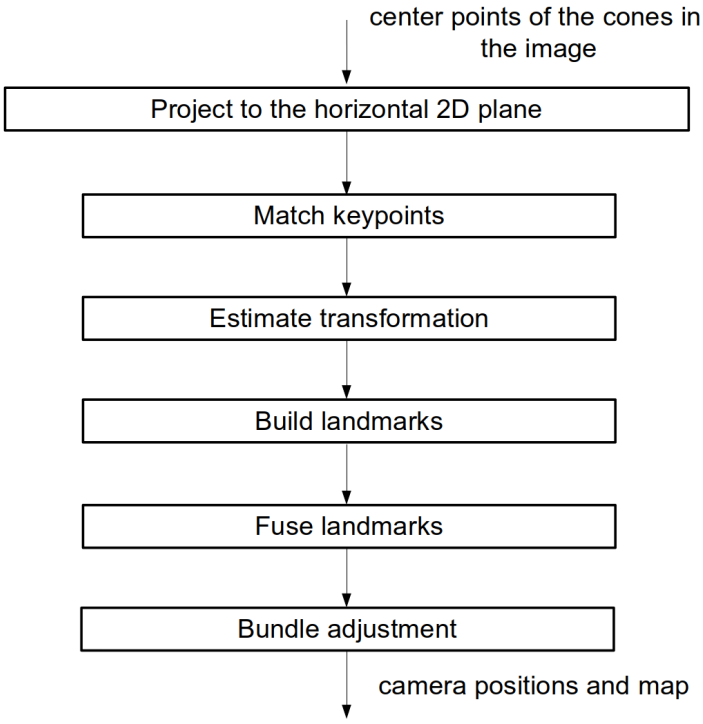


Figure 2.14: *Flowchart for visual odometry. The input is keypoints in the 3D coordinate, the output is camera positions and map.*

2.5.1 Coordinate projection

Traditional stereo visual odometry calculates the essential matrix using Equation 2.9. According to five-point algorithm [38], five matched keypoint pairs are used to decompose the essential matrix and compute the transformation matrix between two image frames. However, the essential decomposition will degrade, which means that the correct rotation and translation matrix cannot be obtained, if most keypoints are on the same plane as shown in Figure 2.15.

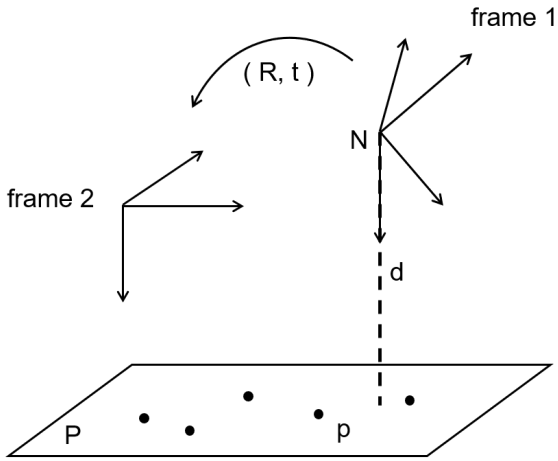


Figure 2.15: *This figure shows a special case that essential decomposition will degrade when most of the keypoints are on the same plane.*

In this situation, keypoints in frame 1 can be transformed to frame 2 following Equation 2.12, H is the corresponding 2×3 transformation matrix. x_1, x_2 is a matched keypoint pair.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \underbrace{\begin{bmatrix} R & T \end{bmatrix}}_{\text{H}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.12}$$

$$x_2 = Hx_1 \tag{2.13}$$

According to the property of the cross product

$$x_2 \times x_2 = 0 \rightarrow x_2 \times Hx_1 = 0 \tag{2.14}$$

By definition of the cross product, $u \times x_2$ must be perpendicular to x_2, here u is an arbitrary three dimensional vector, then

$$(u \times x_2) \perp Hx_1 \tag{2.15}$$

$$\Rightarrow (u \times x_2)^T Hx_1 = 0 \tag{2.16}$$

$$\Rightarrow (-x_2 \times u) Hx_1 = 0 \tag{2.17}$$

$$\Rightarrow x_2 \widehat{u} Hx_1 = 0 \tag{2.18}$$

According to Equation 2.9, $E = \widehat{u}H$, which means that any three dimensional vector u satisfies Equation 2.9. The essential matrix has infinite solutions, thus the unique solution can not be found. Since cones are on the ground and the ground is roughly a plane, the heights of the cones are almost the same, which means the cone centers are almost on the same plane. Thus the essential matrix has infinite solutions. To solve this problem, the 3D cone position is projected to the horizontal plane, as shown in Figure 2.16. In the 2D plane, the position and orientation of the camera can be determined by 3 variables, they are the rotation angle θ and 2D translation t_x and t_y. These variables can be obtained by solving a system of linear equations, which is discussed in subsection 2.5.3.

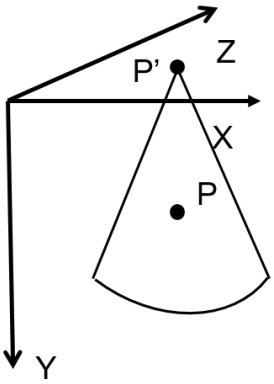


Figure 2.16: *This is a cone model in the 3D coordinate system. P is a keypoint in the 3D coordinate, $P^{'}$ is the horizontal projection of P. The X-axis and Y-axis follow the same axis direction in the image coordinate system, where X-axis points to the right and Y-axis points to the bottom. Due to the right-hand rule, Z-axis points inside.*

2.5.2 Match keypoints

Keypoints in two frames need to be matched to compute the transformation between frames. BF matcher is used to match keypoints because of its simpleness. The idea of BF matcher is to compare the Euclidean distance between one keypoint's descriptor in one frame and all keypoints' descriptors in the other frame. The keypoint descriptor is the description of the visual features of a keypoint and is used to compare the similarity between keypoints . The keypoint with the shortest Euclidean distance is the corresponding matched one. The descriptor of our supervised detector is the 2D horizontal coordinate together with a cone label. Before comparing the coordinate distance by BF matcher, a cone label check is applied first.

2.5.3 Rigid transformation

Since only the movement in the horizontal planes is considered, the transformation between frames is just a combination of rotation and translation. This kind of transformation is called the rigid transformation [3]. As shown in Figure 2.17, p_1 and p_2 are keypoints in frame 1, p_1' and p_2' are their matched keypoints in frame 2. R is the rotation matrix, t is the translation vector.

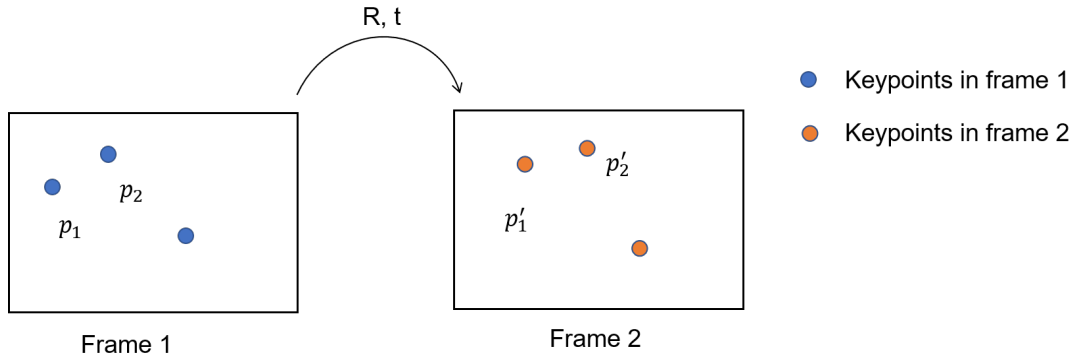


Figure 2.17: *In 2D horizontal plane, keypoints in frame 1 are transformed to frame 2 by the rigid transformation.*

Given two keypoints $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ in frame 1 and their corresponding matched keypoints $p_1{}'(x_1{}', y_1{}')$ and $p_2{}'(x_2{}', y_2{}')$, the relationship between two matched pairs is,

$$R \cdot p_1 + t = p_1{}' \tag{2.19}$$

$$R \cdot p_2 + t = p_2{}' \tag{2.20}$$

As the rotation angle α is the only variable in the rotation matrix and the translation is a two-dimensional vector, the Equation 2.19 can be rewritten as

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1\cos\alpha - y_1\sin\alpha + t_x \\ x_1\sin\alpha + y_1\cos\alpha + t_y \end{bmatrix} = \begin{bmatrix} x_1{}' \\ y_1{}' \end{bmatrix} \tag{2.21}$$

Same to p_2,

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} x_2\cos\alpha - y_2\sin\alpha + t_x \\ x_2\sin\alpha + y_2\cos\alpha + t_y \end{bmatrix} = \begin{bmatrix} x_2{}' \\ y_2{}' \end{bmatrix} \tag{2.22}$$

Let

$$a = (x_2 - x_1)^2 + (y_2 - y_1)^2 \tag{2.23}$$

$$b = -2(x_2{}' - x_1{}')(y_1 - y_2) \tag{2.24}$$

$$c = (x_2{}' - x_1{}')^2 - (x_2 - x_1)^2 \tag{2.25}$$

According to Equation 2.21 and Equation 2.22, it can be concluded that

$$\sin\alpha = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{2.26}$$

$$\cos\alpha = \pm\sqrt{1 - \sin^2\alpha} \tag{2.27}$$

$$t_x = x_1{}' - x_1\cos\alpha + y_1\sin\alpha \tag{2.28}$$

$$t_y = y_1{}' - x_1\sin\alpha - y_1\cos\alpha \tag{2.29}$$

Details can be found in appendix C.

According to Equation 2.26, $\sin\alpha$ has two solutions. During a short moment (it depends on the frequency of the system), the rotation angle cannot be too big. Thus it is reasonable to set a limit to α to filter out unreasonable estimations. The total residuals between different solutions is compared in order to produce a good estimation of the rigid transformation. Assuming p_2 is a keypoint in frame 2, p_1 is its matched keypoint in frame 1, H is the rigid transformation from frame 1 to frame 2, p_3 is the projection of p_2 in frame 1, so that $p_3 = H^{-1}p_2$. The residual between p_1 and p_3 is defined as,

$$r_{residual} = |p_1 - p_3| = \left|p_1 - H^{-1}p_2\right| \tag{2.30}$$

For all solutions in Equation 2.26 and Equation 2.27, all keypoint pairs' residuals are summed up, which produces the total residual of each solution. The one that results in the lowest total residual is the best solution and then the rigid transformation matrix is computed. The position of the camera is calculated using the rigid transformation between frames. Assuming the initial camera position is $p = \begin{bmatrix} x & y \end{bmatrix}^T$ and the rigid transformation from frame 1 to frame k is $H_1^k = H_1^2 H_2^3 \cdots H_{k-1}^k$. Then the position in frame k is $H_1^k p = H_1^2 H_2^3 \cdots H_{k-1}^k \begin{bmatrix} x & y \end{bmatrix}^T$.

2.5.4 Landmark building and landmark fusion

Landmarks are points that make up the point cloud. The process of landmark building is based on the global coordinate, which is usually referred to the initial camera coordinate. All matched keypoints are projected to the global coordinate in order to build up the landmarks. Book-keeping list is a list to keep track of the matching relationship of keypoints and landmarks. Table 2.2 shows an example book-keeping list. There are two rows which correspond to two continuous frames, let's say frame i and frame $i+1$. The j^{th} column means the j^{th} keypoint in that frame. The value in the table corresponds to the index of the landmark. -1 means the keypoint is not matched to any landmarks. Let's say that there are five keypoints in frame i and they are matched to landmark 0, 1, 2, 3 and 5 correspondingly. The 2^{nd} keypoint in frame i is matched to the 1^{st} keypoint in frame $i+1$ and they are matched to the same landmark 1, etc. A landmark should match keypoints in more than two frames under the assumption that the camera frequency is high and the image processing speed is fast enough. Those landmarks only match to keypoints from two frames are considered as noise and removed from the book-keeping list.

i	0	1	2	3	5
i+1	1	3	-1	5	

Table 2.2: This is a book-keeping list that shows the result of landmarks building.

Landmark fusion is the process to check whether to add new landmarks for a new coming frame. The rule of landmark fusion is that, for a matched keypoint pair, if one of them has been assigned to a landmark, the other one is assigned to the same landmark; if none of them are assigned to any landmarks, they are assigned to a new landmark. Continue with Table 2.2, if a new frame $i+2$ is added to the book-keeping list. Let's say 1^{st} keypoint in frame $i+2$ is matched to the 2^{nd} keypoint in frame $i+1$. According to the rule of landmark fusion, since the 2^{nd} keypoint in frame $i+1$ has been assigned to landmark 3, so the value of the 1^{st} element in row $i+2$ is 3. If 2^{nd} keypoint in frame $i+2$ is matched to the 3^{rd} keypoint in frame $i+1$, since 3^{rd} keypoint in frame $i+1$ has not been assigned to any landmark (marked as -1), a new landmark 6 is assigned to these two keypoints. Table 2.3 shows the result of the landmarks fusion.

i	0	1	2	3	5
i+1	1	3	6	5	
i+2	3	6			

Table 2.3: This is a book-keeping list that shows the result of landmarks fusion.

2.5.5 Bundle adjustment

The rigid transformation estimation introduces errors due to incorrect matching or inaccurate keypoint positions. Therefore, when projecting landmarks back to each frame, there is an error between the projection and its corresponding keypoint. This error is called the reprojection error. Estimated camera positions drift with the accumulation of reprojection errors, which is the major drawback of visual odometry. Bundle adjustment, being an optimization method, is one of the key methods to mitigate the reprojection error. The mathematical model of bundle adjustment is based on the graph as shown

in Figure 2.18. This graph shows the relationship between landmarks x_j and frames P_i. If x_j has a projection in frame P_i, then add a link between x_j and P_i.

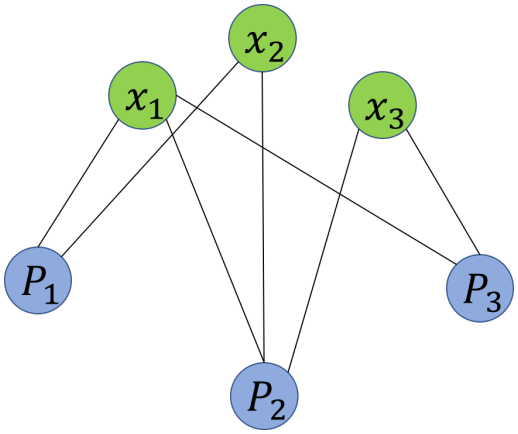


Figure 2.18: *This graph shows the relationship between landmarks x_j and frames P_i. If x_j has a projection in frame P_i, then add a link between x_j and P_i.*

Assuming the image coordinate of landmark x_j in frame P_i is u_{ij}, the image coordinate of its reprojected point is v_{ij}. Then the reprojection error is defined as

$$e_{ij} = \|u_{ij} - v_{ij}\| \tag{2.31}$$

The principle of bundle adjustment is to minimize the sum of all reprojection errors, which is defined as,

$$\min_{R_i,t_i,X_j} \sum_{i,j} \alpha_{i,j} \|u_{ij} - v_{ij}\| = \min_{R_i,t_i,X_j} f \tag{2.32}$$

$\alpha_{i,j} = 1$ when landmark x_j has a projection in frame P_i, otherwise $\alpha_{i,j} = 0$. The process of finding the minimum of f is the same as finding its fastest descent direction iteratively. According to Newton's method, the descent direction for x_k is

$$p_k = -\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k) \tag{2.33}$$

However, $\nabla^2 f(x_k)^{-1}$ is not always positive, so a diagonal matrix γI is added to make sure that $\nabla^2 f(x_k)^{-1} + \gamma I$ is positive definite[34]. As a result, the descent direction for x_k is,

$$p_k = -\left[\nabla^2 f(x_k) + \gamma I\right]^{-1} \nabla f(x_k) \tag{2.34}$$

3 Results

The result chapter first introduces the hardware and software that are used in this project. Details about how data is collected, annotated and trained will be explained. After that, results on stability, efficiency and accuracy will be shown.

3.1 Hardware

The major hardware used in this project is a ZED camera and an eight-cored computer with processor AMD Ryzen 7 1800x, frequency at 3.6GHz. GPU was not available in our autonomous system at the moment.

ZED camera is a popular stereo camera with the maximum resolution 4416x1242, throughput between 15 and 100 FPS, 110° field of view and 120mm baseline. Moreover, it is compact, low-distorted and well-synchronized. Figure 3.1 shows the ZED camera mounted under the main hoop of the vehicle.

Figure 3.1: *This is a stereo camera mounting under the main hoop of the vehicle.*

Table 3.1 shows the specification of the zed stereo camera[58]. It can support maximum 4416×1242 resolution but the throughput is only 15 FPS. Resolution at 2560×720 with throughput at 15 FPS makes a good balance between efficiency and accuracy and thus is used in this project.

Table 3.1: Specification of the stereo camera.

Video Mode	Frames per second	Output Resolution (side by side)
2.2K	15	4416×1242
1080p	30	3840×1080
720p	60	2560×720
WVGA	100	1344×376

3.2 Software

All code is implemented in C++ in Linux operating system to achieve high efficiency. OpenCV library is the major library that is used in this project. ORB detector and most functions in stereo vision such as rectification, BM algorithm and triangulation have OpenCV implementation. As for deep learning library, an efficient C++ library called Tiny-dnn [40] has been used. Tiny-dnn is just a header file and it is created for computing resource limited project. It supports a lot of multi-threading techniques such as *threading building blocks* (TBB), *open multi-processing* (OpenMP), etc., to make use of the CPU. The code of our approach, related data and results are available in github [57].

3.3 Data collection

Data about three competition layouts are collected by ZED camera to test the whole system. Besides, data in different weather conditions (snowy, sunny, rainy, cloudy) are also collected to test the robustness of the supervised detector. Sample images are shown in Figure 3.2. ZED camera was mounted under the main hoop of the vehicle and connected to a laptop when data was collecting. The vehicle was pushed manually to provide a stable low vehicle speed. The stereo images were saved into the disk frame by frame, they are used to annotate training data and test our approach.

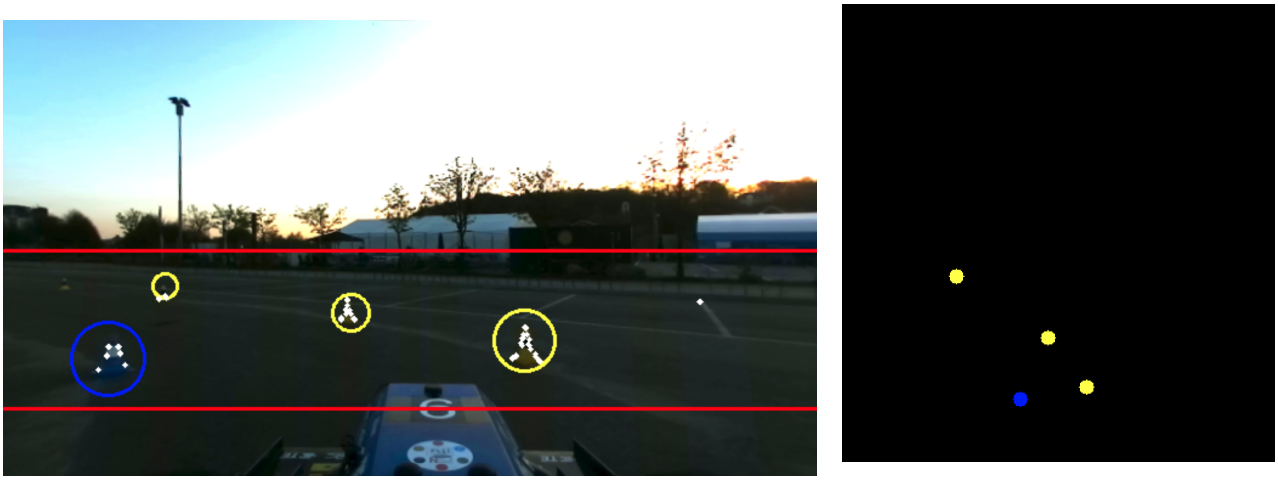
Figure 3.2: *Data sample collected by ZED camera.*

3.4 Annotation and training

Our annotation covers five different types of data including background, blue cone, yellow cone, small orange cone and big orange cone, in different background, scale, weather and lighting conditions. All types of data are in similar amount of numbers. An efficient annotation procedure using a supervised detector has been created. First, the supervised detector was applied on all images to get keypoints. Then image patches centered on the keypoints were extracted and put on one dataset. The CNN classifier was then applied to this dataset and classified the image patches into five different subfolders, each of which contained image patches being classified as the same class. After that, all image patches were verified manually and those that were misclassified were moved to the correct subfolder manually. At the end, the dataset is randomly splited into 70% training dataset and 30% validation dataset. Data augmentation was used to expand the training dataset [52] and overcome the overfitting. Finally, the training data was normalized between 0 and 1 and fed to the network. 11451 training data were used to train the network. The hyperparameter setting was as follows: learning rate = 0.005, epochs = 300, batch size = 64.

3.5 Results for stability

Stability of the whole system is the ability of handling different situations like background, weather and lighting variance. These variance affects most on the supervised detector and secondly on the stereo visual odometry. Therefore, the supervised detector was tested with images under different conditions. Figure 3.3a shows the result of an image captured at sunset. The lighting condition is bad and even hard for human eyes to distinguish the cones from the image. The detection results are marked with colored circles. The blue circles stand for blue cones, the yellow circles stand for yellow cones. The diameter of the circle corresponds to the length of the image patch. It can be seen that the image patches are correctly localized and classified. Figure 3.3b shows the center points of the image patches in the 3D coordinate system, they are calculated by triangulation (in a bird's eye view).



(a) *The result of the supervised detector in an image captured at sunset. Colored circles stand for different cone recognition results.*

(b) *The center points of the cones are projected to the 3D coordinate system by triangulation (in a bird's eye view).*

Figure 3.3: *The result of the supervised detector in an image captured at sunset.*

Figure 3.4 shows another example frame in the same lighting condition. The yellow circles stand for yellow cones, the orange circles stand for small orange cones. Although there are some cones overlapped in the image, they are still detected correctly by the supervised detector.

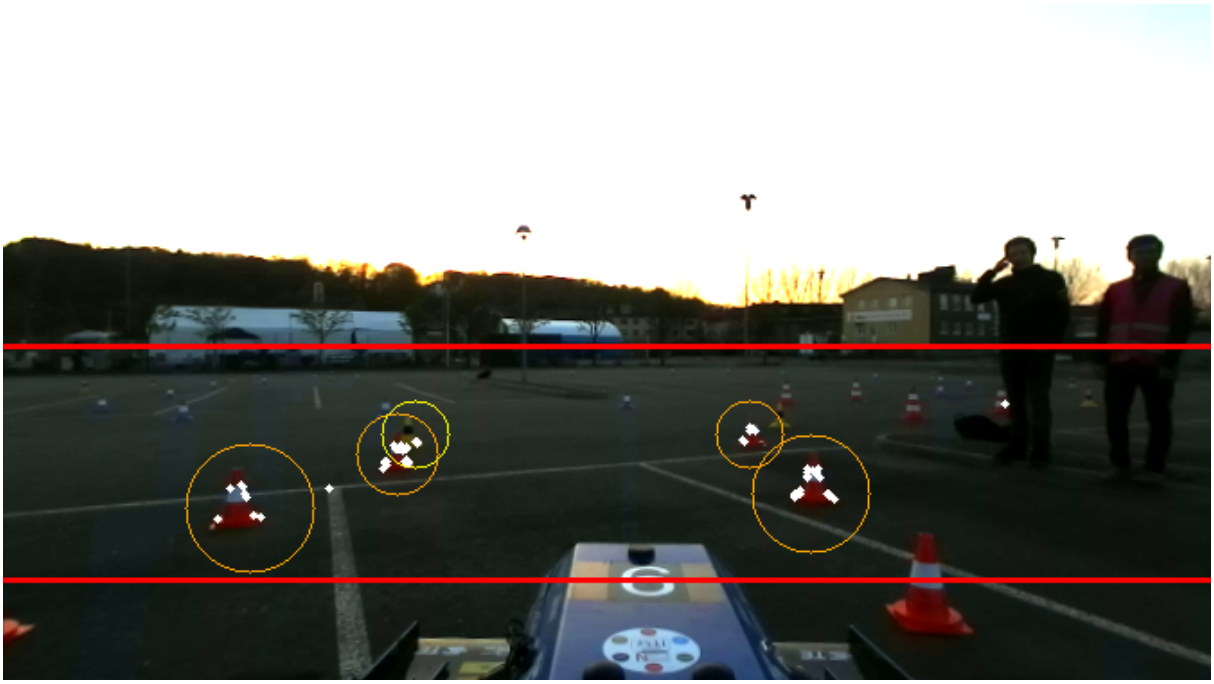
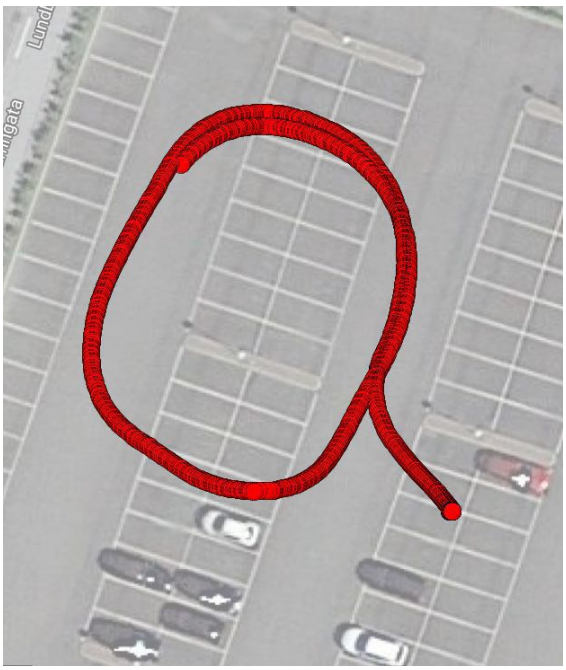


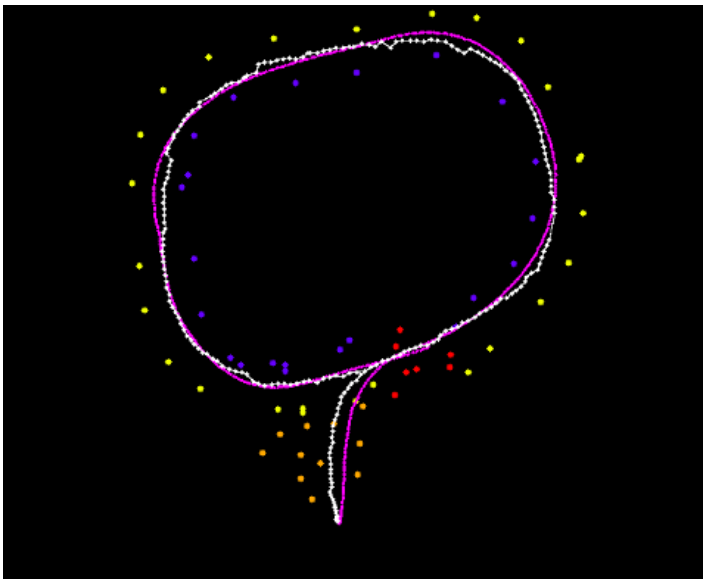
Figure 3.4: *The result of the supervised detector in another image captured at sunset.*

Figure 3.5 shows another example frame in a sunny winter morning, the ground is covered with snow, cones are much brighter. There are much more cones in the course than the last two examples. Although some keypoints hit the ground, they are successfully filtered out by post-processing and the CNN classifier. All cones are correctly localized and classified. More detection results (with different backgrounds, weathers and lighting conditions) are available in appendix D.

Figure 3.5: *The result of the supervised detector in an image captured on a sunny winter morning.*



(a) *Vehicle's GPS data shown in Google Maps as ground truth.*



(b) *The result of the stereo visual odometry using a supervised detector in the testing track. The purple curve shows the GPS data of the vehicle. The white curve shows the camera trajectory calculated by our approach. The blue, yellow, orange and red points show the cones' positions and colors in the track.*

Figure 3.6: *Localization and mapping using our approach with well-tuned parameters.*

To evaluate the performance of the whole system, a simple ellipse track in a parking lot was set up as the testing track. The results in efficiency and accuracy sections are based on this testing track. Figure 3.6a shows the vehicle's GPS data in Google Maps while Figure 3.6b shows the performance of the whole system. The purple curve shows the GPS data of the vehicle trajectory, which can be regarded as the ground truth. The white curve shows the camera trajectory calculated by our approach. The blue, yellow, orange and red points show the cones' positions and colors in the track. The camera

curve has a similar shape as the ground truth and very close to the ground truth. Besides, both the camera curve and the vehicle curve are inside the track, indicating good reconstruction result. Since mapping error is summing up, the starting cones and the ending cones are not coincide but they are very close. However, the ground truth is not measured with a more accurate sensor and thus the mapping error is not calculated. Figure 3.7 shows the performance of the stereo visual odometry using slightly different parameters. It can be seen that some cones are not positioned on the map and the localization has a large drift compared to the ground truth.

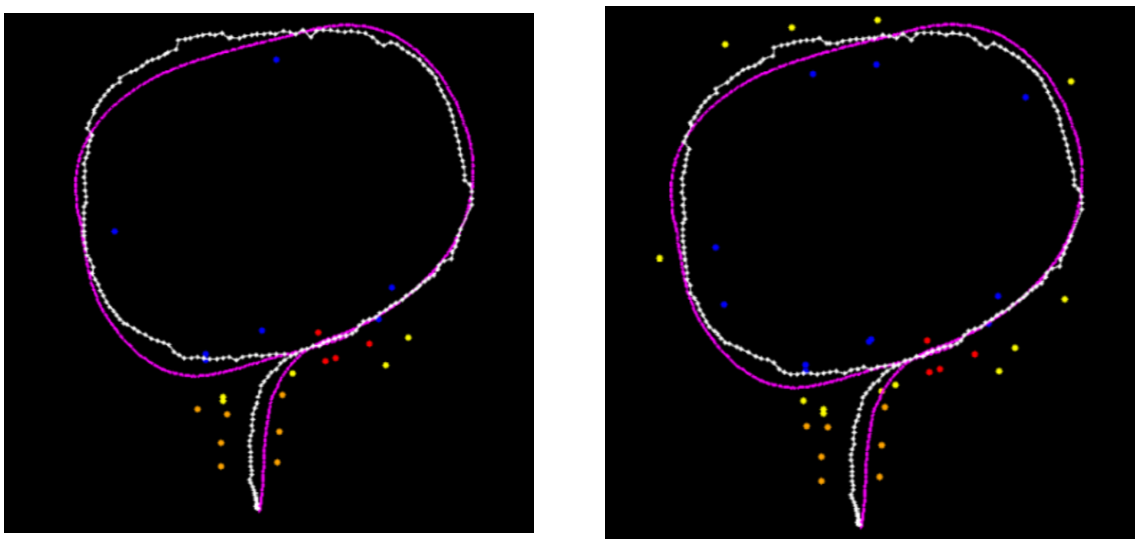
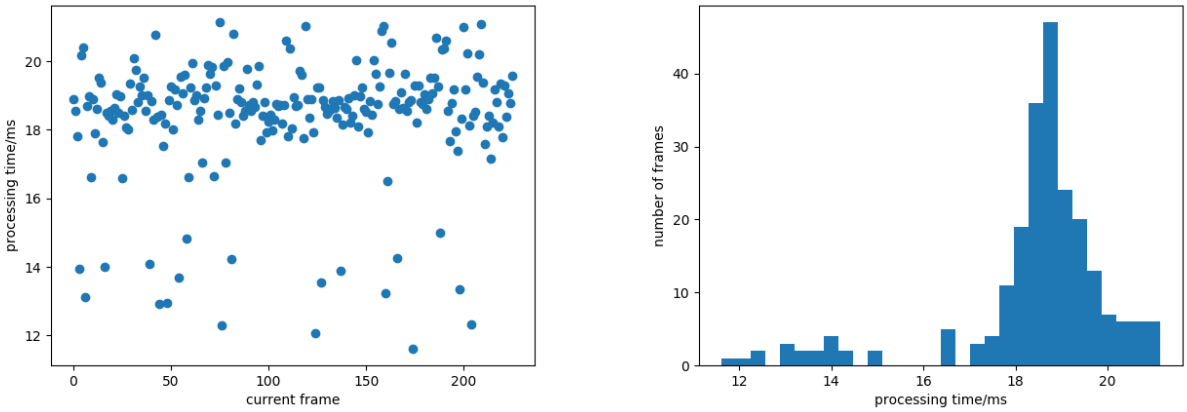


Figure 3.7: *Localization and mapping using our approach with bad parameters.*

3.6 Results for efficiency

The efficiency of the whole system is measured by the execution time. The major time-consuming modules in our system are stereoBM, ORB detector and CNN classifier.

The processing time of stereoBM in the testing track is shown in Figure 3.8a. It can be clearly seen that the average processing time is 15.8ms on 672x130 image with standard deviation 1.64ms. Figure 3.8b shows the processing time histogram of the stereoBM.



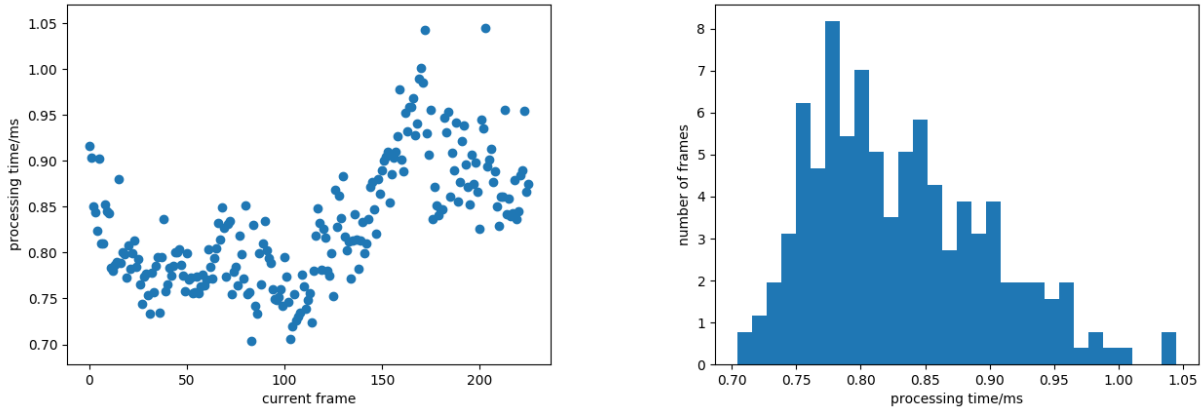
(a) *The processing time of the stereoBM.*

(b) *The processing time histogram of the stereoBM.*

Figure 3.8: *The processing time of the stereoBM in the testing track.*

Figure 3.9a illustrates the processing time of ORB in the testing track. According to this figure, the average processing

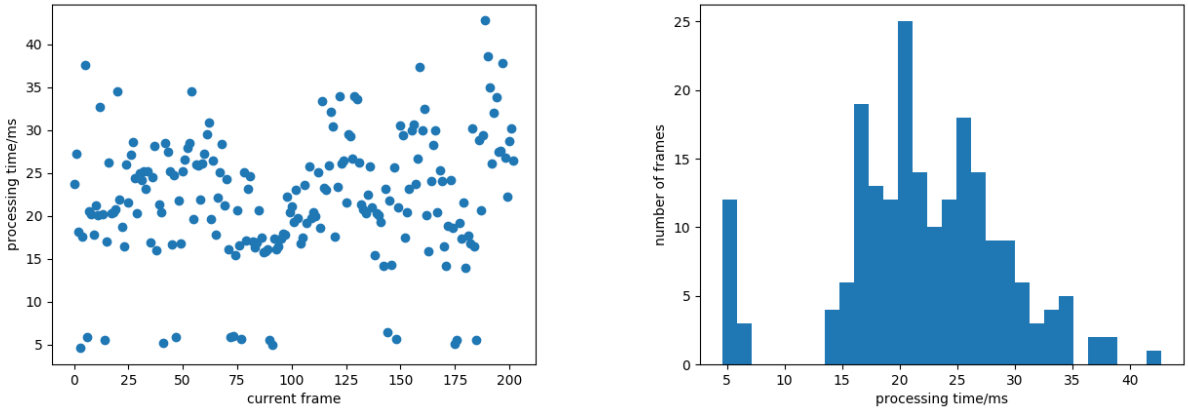
time is 0.8ms on 672x130 image with standard deviation 0.06ms. The processing time histogram of the ORB is shown in Figure 3.9b.



(a) *The processing time of the ORB.* (b) *The processing time histogram of the ORB.*

Figure 3.9: *The processing time of the ORB in the testing track.*

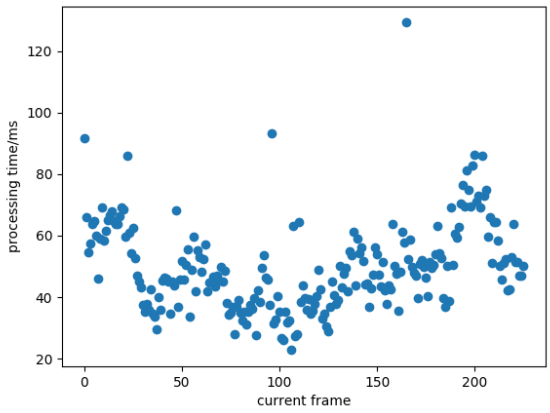
Figure 3.10a indicates the processing time of the CNN classifier in the testing track. As shown in this figure, the processing time varies a lot from time to time. The average processing time is 22.1ms with TBB multi-threading technique and the standard deviation is 7.2ms. The processing time histogram of the CNN classifier is shown in Figure 3.10b.



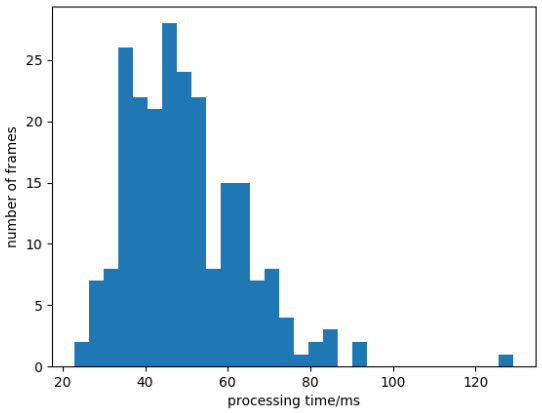
(a) *The processing time of the CNN classifier.* (b) *The processing time histogram of the CNN classifier.*

Figure 3.10: *The processing time of the CNN classifier in the testing track.*

As shown in Figure 3.11a, the processing time of the supervised detector in the testing track varies a lot from time to time. The standard deviation reaches 14.3m with TBB multi-threading techniques while the average processing time is 49.8ms. The processing time histogram of the supervised detector is shown in Figure 3.11b.



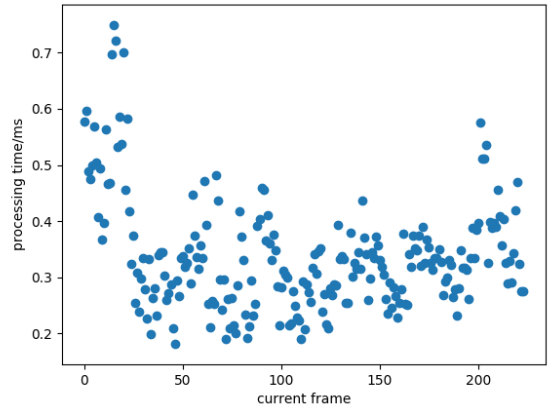
(a) *The processing time of the supervised detector.*



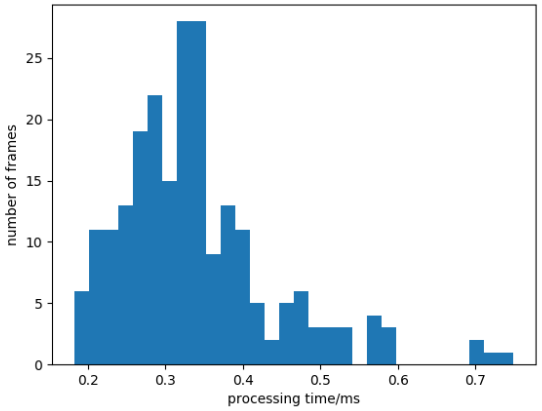
(b) *The processing time histogram of the supervised detector.*

Figure 3.11: *The processing time of the supervised detector in the testing track.*

Figure 3.12a demonstrates the processing time of SVO in the testing track, which averages out at 0.34ms with the standard deviation of 0.1ms. The processing time histogram of SVO is shown in Figure 3.12b.



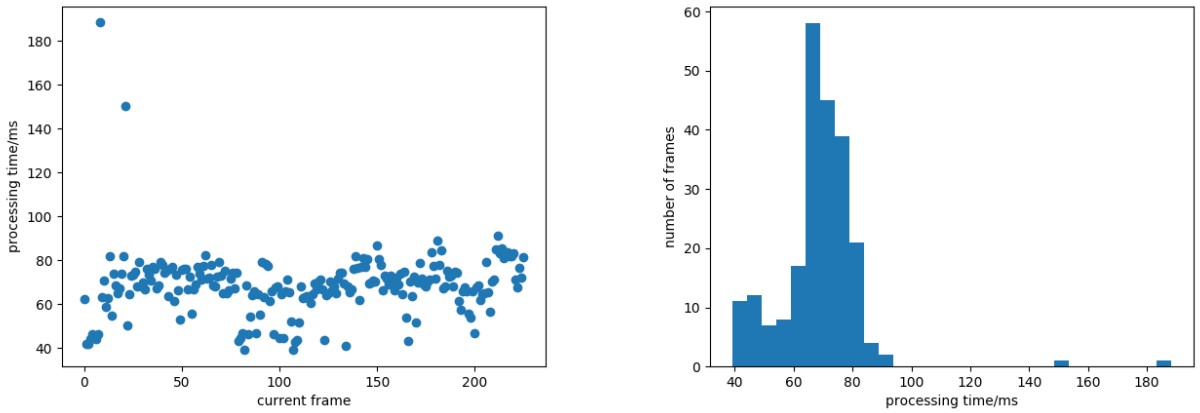
(a) *The processing time of the SVO.*



(b) *The processing time histogram of the SVO.*

Figure 3.12: *The processing time of the SVO in the testing track.*

The scatter diagram and histogram of the processing time of the whole system in the testing track are shown in Figure 3.13b and Figure 3.13a respectively. They clearly indicates that the processing time doesn't increase when the map is building gradually. The average processing time is 68.6ms with the standard deviation of 14.5ms, faster than our expectation (100 ms).



(a) *The processing time of the system.*

(b) *The processing time histogram of the system.*

Figure 3.13: *The processing time of the system in the testing track.*

3.7 Results for accuracy

The accuracy of the whole system is measured by the recognition accuracy and the mapping error. However the ground truth is not measured and the mapping error is not calculated, so in this section only the recognition accuracy is covered.

5062 validation data are used to validate the CNN training result. Table 3.2 shows the validation accuracy result, the column of the table is $True$ data, the row of the table is $Positive$ data. The first column means class 0 contains 1404 data, 1374 of them are correctly classified, 14 of them are misclassified as class 1, etc. The first row means 1388 data are classified as class 0, 1374 of them are correctly classified, 5 of them are actually from class 1, etc. So the precision of class 0 is $\frac{1374}{1388} = 98.99\%$, recall of class 0 is $\frac{1372}{1404} = 97.79\%$. F_1 score for class 0 is $\frac{2 \times P_{precision} \times P_{recall}}{P_{precision} + P_{recall}} = \frac{2 \times 98.99\% \times 97.79\%}{98.99\% + 97.79\%} = 98.48.4\%$. Table 3.3 shows the precision, recall and F_1 score of all classes. From the table, it can be seen that all classes' precision, recall and F_1 score are higher than 95% in validation data. The average validation accuracy is 98.81%.

Table 3.2: Convolutional neural network validation accuracy.

	background	blue	yellow	small orange	big orange
background	1374	5	3	3	3
blue	14	1271	0	0	0
yellow	5	0	1072	0	0
small orange	9	0	1	790	5
big orange	2	0	0	10	495

Table 3.3: Precision, recall and F_1 score of all classes in validation data.

	background	blue	yellow	small orange	big orange
precision	99.0%	98.9%	99.5%	96.9%	97.6%
recall	97.8%	99.6%	99.5%	98.4%	98.4%
F_1	98.4%	99.2%	99.5%	97.6%	98.0%

All classes' precision, recall and F_1 score of all classes in testing data is shown in Figure 3.4. All classes' precisions are higher than 99% excepts for the background precision, which is less than 50%. All classes' recall are higher than 96%. The F_1 score of the background is only 42.9% because of its low precision, other classes' F_1 scores are higher than 98%. The overall test accuracy is 98.97%.

Table 3.4: Precision, recall and F_1 score of all classes in testing data.

	background	blue	yellow	small orange	bid orange
precision	44.3%	99.9%	99.9%	100%	98.6%
recall	96.9%	98.6%	99.7%	96.6%	97.4%
F_1	60.8%	99.2%	99.8%	98.3%	98.0%

4 Discussion

In this chapter, results in stability, efficiency and accuracy will be discussed separately. Appropriate algorithms, skills, choices that contribute to the results will be highlighted. Inappropriate algorithms, solutions that lead to problems will be investigated. Possible solutions to the issues will also be investigated. At the end of this chapter, the applications of our approach will be discussed.

4.1 Stability

The stability of the algorithm influences the performance, robustness and function requirements in embedded systems. If the algorithm gets errors suddenly, the autonomous car would stop, get out of the track or have unexpected behaviors, resulting in *did not finish* (DNF) in the competition or even cause accidents. The stability of the algorithm will be discussed in two directions: the supervised detector and stereo visual odometry.

From Figure 3.3a, 3.4, 3.5 and appendix D, it can be seen that our supervised detector localized and classified most cones in the images correctly despite of various backgrounds, weathers and lighting conditions. There are mainly two contributors, ORB detector and CNN classifier. On the one hand, since cones have obvious contours, ORB detector finds most keypoints on the cones regardless of various conditions. Even though some keypoints hit on the ground, they can be filtered out by the 3D positions. Besides, most keypoints are symmetrically detected on the cones, thus the mean coordinate of the keypoints is close to the cone centroid in the image. In addition, overlapped cones are separated correctly as shown in Figure 3.5 since our supervised detector groups cones in the 3D coordinate where all cones are at least 1m away from each other. On the other hand, stable keypoint positions provide RoIs for the CNN classifier which reduces the possibility of misrecognition. Besides, data in different backgrounds, weathers and lighting conditions are annotated and trained with the CNN classifier so that this variance can be learnt by the CNN classifier. However, sometimes yellow cones might be classified as blue cones when the light is too strong and makes the shadow of the cones in blue. To avoid this mistake, the recognition labels in three continuous frames are checked and those cones that change their labels are filtered out. In conclusion, our supervised detector has good stability.

As can be seen from Figure 3.6b and 3.7, our stereo visual odometry has bad stability. The performance of the visual odometry relies a lot on the parameters. The major parameter to be tuned is the threshold of Euclidean distance, which is used in BF matcher. Other parameters of our approach are listed in appendix A. The threshold of Euclidean distance is quite sensitive to the vehicle speed and track layout especially when the vehicle is turning at the corner. There are basically four reasons.

First of all, there are too few cones in the image especially when the car is turning, so that the keypoint matcher might make mistakes. Besides, too few keypoints might not give a good estimation of the rigid transformation. One misclassified keypoint or incorrect position may result in a big error in the rigid transformation estimation, making the system drifting or crash. One possible solution is to make use of all the keypoints detected by ORB to match keypoints and estimate rigid transformation. More specifically, after recognition, just assign the labels to the keypoints that belong to the same group. Then each keypoint has its position and label. Furthermore, the descriptor of the keypoint could be calculated and a better keypoint matcher like K-nearest neighbour algorithm [9] could be used. When more keypoints and a better keypoint matcher is used, the transformation estimation would be more accurate and robust.

For another, although ORB detector finds keypoints mostly on the cones, it still detects keypoints on the ground even though keypoints have been filtered out by their height. This is because the ground is uneven, thus the heights of the cones are different. This noise would result in wrong recognition, wrong keypoint matching and introduce errors when calculating the cone center. One possible solution is to use *random sample consensus* (RANSAC)[15] to estimate the

ground and remove those keypoints that are close to the ground. RANSAC is the algorithm to iteratively find the optimal parameter model using a set of data including inliers and outliers.

On top of that, the stereo visual odometry has been simplified too much. For example, keypoints are projected into the 2D horizontal plane, this introduces errors when the ground is uneven. If all the keypoints detected by ORB detector are used to estimate the transformation matrix, the essential decomposition issue does not exist and thus the 3D coordinate could be used to estimate the transformation matrix, so that the projection error can be reduced.

Last and most importantly, the positions of the cones are inaccurate. Inaccurate cone positions lead to imprecise transformation estimation and further affect the accuracy of the localization and mapping. More discussions will be covered in section 4.3.

4.2 Efficiency

The speed of the autonomous car is usually limited by the processing time of the algorithm. The expected maximum average image processing time is 100ms per frame and the following decisions are made to reach the aim. First of all, the camera resolution is set to $2560 \times 720 \times 3$ to give a decent image quality and high throughput (15 FPS). Afterwards the image is resized to $1344 \times 376 \times 3$ in order to reduce the computation load of the supervised detector. Secondly, CNN classifier is used to classify small image patches instead of using object detection to run on the whole image. Since only CPU is available, the computation load of the object detection would be huge. As shown in Figure 3.10a and 3.10b, the processing time varies a lot throughout the test because the number of cones varies between frames. Besides, ORB detector is used to find image patches because it is much faster than other keypoint detectors. As shown in Figure 3.9a and 3.9b, the processing time of ORB detector increased at the end might because that there are more cones or noise. But the processing time difference is just 0.35ms. Irrelevant keypoints detected by ORB are filtered out or classified as the background and not considered to be used in visual odometry, this speeds up the keypoints matching, landmarks building and landmarks fusion. After that, the keypoint searching area is limited to the middle part of the image where most cones are located. Those cones that are outside of the searching area are usually above the detecting range of the disparity map, which means their depths are invalid. Thus it is safe to ignore them. Last but not the least, C++ implementation helped improve the execution efficiency and finally the average processing time of the whole system in test data is 68.6ms per frame. As shown in Figure 3.13a and 3.13b, those peaks that are higher than 100ms might be system delay.

To reduce CPU load and increase speed, it would be nice to use GPU. GPU is the microprocessor designed to run graphics calculations on personal computers, workstations, game consoles, and some mobile devices (tablets, smartphones, etc.).GPU is particular good at operational analysis, deep learning, and machine learning algorithms[41]. With GPU, the computation can run 10 to 100 times faster than the same computation with CPU.

The efficiency of the algorithm is usually bounded with the accuracy of the algorithm. Although the efficiency aim is reached, some accuracy is sacrificed. This will be discussed in section 4.3. Since the traditional visual odometry has been simplified, the algorithm is somehow unstable.

4.3 Accuracy

The accuracy of a SLAM algorithm is very important since it directly affects the path planning that relies on it, results in choosing wrong directions, hitting cones or getting out of the track. The accuracy of the algorithm will be discussed in two directions: recognition accuracy and mapping accuracy.

As shown in Table 3.4, all classes' precisions on the test data are higher than 99% excepts for the background precision, which is less than 50%. This is because there are very few patches that are actually background in the test data (64 out of 7760) after ORB detection and post-processing. Thus, this precision cannot reflect the real performance of the CNN classifier, and less practical than recall in this case. The recall of the background is 96.9%, much higher than the precision, this means that cones are misclassified as the background and this is probably because it was sunset during data collection, the light was too dark and made some cones indistinguishable from the background.

In section 4.1, one of the reasons for bad stability is the inaccurate cone positions. The positions of the cones are inaccurate due to several reasons. In the first place, the resolution of the disparity map is in low resolution, i.e., the

boundary between the cone and the background is not very clear. Increasing image resolution could help but meanwhile the processing time would increase. In order to higher the resolution of the disparity map, considering only the depth information of the keypoints is needed, one could only compute disparity for the keypoints instead of the whole image. However, ORB detector is needed to be run on both left and right images, also need to match them. Though ORB detector is very fast, the overall processing time might be slower and could not meet our requirement. Thus this solution needs further investigations.

Additionally, the maximum detecting range of the stereo vision is only 8.1234m, that introduces big errors to the system. The detecting range is limited by the baseline of the stereo camera. The longer baseline, the longer detecting range. This is because according to the triangulation Equation 2.11, $Z_{max} = \frac{Bf}{d_{min}}$. Assuming f, d_{min} are the same, if the baseline B is longer, Z_{max} is larger. It is necessary to choose another stereo camera with longer baseline or use two or more mono cameras to build a stereo visual system.

Apart from that, the cone center is computed by averaging the 3D coordinates of the keypoints that are considered to belong to the same cone, which is based on the assumption that keypoints are symmetrically located in the cone and within a certain range in the 3D coordinate. This assumption is not always true and noise might be introduced into the calculation and produce errors.

Last but not the least, the camera's position driftings because errors accumulate gradually. Big errors in some frames have a great effect on the latter localization and mapping. It is suggested to use vSLAM instead of visual odometry because it keeps track of the key frames and closes the loop when the agent reaches the same key frame again, which is called loop closure. Errors in all previous frames are minimized during the loop closure, that's why vSLAM has fewer drifting problems. Besides, filters like extended Kalman filter and particle filter [19] can be used to fuse sensors like GPS, IMU or wheel speed sensor with the camera, to provide better localization.

4.4 Comparison between the supervised detector and object detection algorithms

There are similarities and differences between the supervised detector and object detection algorithms. The supervised detector can localize and classify objects as object detection algorithms, but since the localization and classification are separated, it is not an end-to-end method. The supervised detector is designed for detecting specific objects (i.e. cones in this project) and based on object characteristics (i.e. cone height and distance between cones), thus compared with object detection algorithms, more fine-tuning is required to be tweaked to detect other objects. But since it is rule-based, it reduces the probability of wrong detection and unnecessary region proposals. Besides, it is easier to annotate data and train the network.

Faster R-CNN, *you only look once* (YOLO) [43] and *single shot multibox detector* (SSD) [32] are probably the most well-known object detection algorithms that are applied in autonomous driving owing to their real-time performance and sufficient accuracy. Faster R-CNN achieves the highest accuracy among them, with 73.2 *mean average precision* (mAP) in *PASCAL visual object classes 2017* (VOC2017). But the efficiency is low, just around 7 *frames per second* (FPS) with GPU. YOLO achieves 155 FPS with GPU, faster than the other two algorithms, but the accuracy is quite low (52.7 mAP). SSD makes a good balance between efficiency and accuracy, 22-46 FPS with GPU and 72.3 mAP. Our supervised detector achieves speed at 20 FPS with TBB multi-threading technique, which is similar to SSD. It would be much faster with GPU. Since the dataset between VOC and this project are different, we did not compare the mAP.

4.5 Comparison between stereo visual odometry and vSLAM

One of the most popular vSLAM algorithm is ORB-SLAM[35]. Compared to stereo visual odometry, ORB-SLAM directly uses ORB detector to extract features and has one more optimization procedure called loop closing. Loop closing is the task to decide whether or not the vehicle is passing the same place or frame that it has been to before[37] after the vehicle has been driving for a certain distance. The loop closing is composed of loop detection and loop correction. When loop detection correctly detects a similar frame, the correlation of current data with all historical data is provided. After that, loop correction uses this correlation to re-localize the vehicle and help eliminate the drifting. With loop closing, the

localization of the vehicle is more stable and accurate than stereo visual odometry.

According to the report written by Raul Mur-Artal[35], ORB-SLAM takes 405 ms per frame after feature extraction, while stereo visual odometry we used only takes 0.34 ms which is much faster than ORB-SLAM.

5 Conclusion

SLAM, being a fundamental problem in robotics and navigation, has been developed for many years and is getting more and more popular in autonomous driving. Vehicle manufacturers are looking for a robust, efficient, accurate but low-cost solution. LiDAR is one of the most common sensors in SLAM solutions but the price is so high. Thus a lot of advanced algorithms such as vSLAM and stereo visual odometry are based on camera, which is much cheaper and contains more potential information. The common drawback of visual-based SLAM solutions is that the landmarks that make up the map are content-less and could not provide semantic information for path planning algorithms to increase their performance. CNN has shown great value in image recognition recently and may prove useful for semantic mapping. Thus one natural idea is to integrate CNN into SLAM solutions.

This thesis aims to construct an algorithm of simultaneous localization and semantic mapping. The approach is evaluated on the data driven by the FSG driverless 2018 competition, in which hundreds of teams all over the world worked hard to build up their autonomous Formula-style cars and competed with each other. The main task for FSG driverless competition is to drive the vehicle automatically in paths bounded by two curvatures of cones, which are of the same shape but have different colors, sizes and usages. To achieve a good score in the competition, a stable, efficient and accurate SLAM solution is expected.

In this project, we combined ORB detector and a CNN classifier to create a keypoint detector called supervised detector and applied it to stereo visual odometry to localize the camera and build up a semantic map simultaneously. Our approach achieved 98.97% test accuracy in recognition and created a map that looked similar to the GPS data. All the percentages of classes' precision, recall and F_1 score are higher than 95%, except for one class' precision. The supervised detector stably detects almost all the cones in the image regardless of different backgrounds, weathers, lighting conditions, etc. It also attaches labels to the keypoints, meta information to the path planner. Besides, with the help of BM algorithm, ORB detector, k-d tree, Tiny-dnn library, TBB multithreading technique and C++ implementation, the average processing time of the whole system is 68.6ms with standard deviation 14.5ms, this is a really fast and quite stable speed. Overall, our supervised detector is a stable, fast and accurate keypoint detector and classifier, it also speeds up the visual odometry procedure, achieving good execution time relevant for live systems. Besides, it can be tweaked to detect other objects such as vehicles and pedestrians. Compared with the state-of-art object detection algorithms, supervised detector reduces the probability of wrong detection and unnecessary region proposals but requires more engineering knowledge and fine-tuning. Besides, it is easier to implement with lower computation load required. Stereo visual odometry is more efficient than the well-known vSLAM solutions but has drifting issue and lower stability.

The major disadvantage of our approach is that the stereo visual odometry is not stable. With proper parameters, stereo visual odometry will show promising performance. However, minor changes of parameters may lead to large drift or even system crash, and do take time and effort to tune them. Since same parameters may not apply to a new map, they can only be used offline. We did not measure the ground truth with high accuracy sensor and calculate the mapping error due to the limited time. Our approach is only tested on one track offline, thus the online performance is unknown. As a result, there are still a lot of work that we can do in the future.

With the supervised detector, stereo visual odometry can build up a map with content and filter out useless landmarks, which are of great value for agent navigation. It can help improve the accuracy of mapping and localization. Besides, the supervised detector can also be used in other keypoint based methods like vSLAM and SfM. Also, combining stereo visual odometry with the supervised detector can promote the development of the applications in robotics fieldlike logistics robots, home robots and medical robots. The idea of giving labels to a point cloud is valuable, but how to combine the localization and mapping method as well as the object detection method still needs more investigations.

5.1 Future work

This project explores a possible solution to simultaneous localization and semantic mapping. However, there are some ideas deserving further investigation but we don't have time to implement. In the first place, the major disadvantage of our approach is that the stereo visual odometry is not stable due to wrong keypoint matching and inaccurate transformation estimation. The fundamental reason is too few keypoints are used to calculate the transformation matrix. Possible solution is to make use of all the keypoints detected by ORB to match keypoints and estimate rigid transformation and assign labels to them afterwards. Additionally, in order to increase the localization accuracy, more sensors like GPS, IMU or wheel speed sensor could be fused with the camera using filters like extended Kalman filter and particle filter. Furthermore, the supervised detector could be applied to vSLAM solutions since the fundamental parts between the stereo visual odometry and vSLAM are the same. The potential benefits include better stability, lower drifting error and higher accuracy, but the increased computation load could be an issue.

Appendices

A Parameters

Name	Description	Value
inputWidth	width of the stereo image	2560
inputHeight	height of the stereo image	720
width	width of the rectified image	672
height	height of the rectified image	376
patchSize	size of the image patch	64
cnnThreshold	threshold of being classified as cones	0.9
blockSizeBM	parameter in opencv StereoBM	19
numDisparitiesBM	parameter in opencv StereoBM	32
uniquenessRatioBM	parameter in opencv StereoBM	15
blockSizeSGBM	parameter in opencv StereoSGBM	9
numDisparitiesSGBM	parameter in opencv StereoSGBM	32
uniquenessRatioSGBM	parameter in opencv StereoSGBM	10
P1	parameter in opencv StereoSGBM	1620
P2	parameter in opencv StereoSGBM	12960
groupRadius	searching radius in kdTree	0.5m
nFeature	number of features in ORB	100
matchThreshold	the distance between matched keypoints should be lower than this threshold	0.2m
skippingFrames	maximal number of frames to be skipped if the matching error is too big	1
angleThreshold	the rotation angle between two frames should be lower this threshold	1 radian

Table A.1: A list of parameters used to produce results.

B Essential Matrix derivation

Define a 3D point U in the global coordinate system and its projection in camera 1 and camera 2 as u_1 and u_2. The distances between U and two camera centers are Z_1 and Z_2. The calibration matrix of two cameras is the same K. The transformation matrices and translation vectors between the global coordinate system and two camera coordinate systems are $\begin{bmatrix} R_1 & T_1 \end{bmatrix}$ and $\begin{bmatrix} R_2 & T_2 \end{bmatrix}$, then

$$Z_1 u_1 = K(R_1 U + T_1) \tag{B.1}$$

$$Z_2 u_2 = K(R_2 U + T_2) \tag{B.2}$$

Assume $|K| \neq 0$, K is invertible.

$$Z_1 K^{-1} u_1 = R_1 U + T_1 \tag{B.3}$$

$$Z_2 K^{-1} u_2 = R_2 U + T_2 \tag{B.4}$$

Replace $K^{-1}u_1$, $K^{-1}u_2$ with ${u_1}'$ and ${u_2}'$.

$$Z_1 {u_1}' = R_1 U + T_1 \tag{B.5}$$

$$Z_2 {u_2}' = R_2 U + T_2 \tag{B.6}$$

Define the global coordinate system as the first camera coordinate system, then $R_1 = I$, $T_1 = 0$,

$$Z_1 {u_1}' = U \tag{B.7}$$

$$Z_2 {u_2}' = R_2 U + T_2 \tag{B.8}$$

From Equation B.7 and Equation B.8,

$$Z_2 {u_2}' = Z_1 {u_1}' R_2 + T_2 \tag{B.9}$$

$${u_2}' = \frac{R_2 \cdot Z_1}{Z_2} {u_1}' + \frac{T_2}{Z_2} \tag{B.10}$$

Define $R = \frac{R_2 \cdot Z_1}{Z_2}$, $t = \frac{T_2}{Z_2}$.

$$u_2{}' = Ru_1{}' + t \tag{B.11}$$

Apply cross product t to both sides of the equation,

$$t \times u_2{}' = t \times Ru_1{}' + t \times t \tag{B.12}$$

Here $t \times t = 0$, after multiplying with $u_2{}'$ on both sides, Equation B.12 becomes

$$u_2{}^{'}(t \times u_2{}^{'}) = u_2{}' t \times Ru_1{}' \tag{B.13}$$

Since $u_2^{'T}$ is perpendicular to the plane $t \times u_2{}^{'}$, so the left side of the equation equals to 0, i.e.,

$$u_2{}' t \times Ru_1{}' = 0 \tag{B.14}$$

This is called epipolar constraint. $t \times R$ is defined the as essential matrix E.

$$u_2{}' E u_1{}' = 0 \tag{B.15}$$

C The calculation of rotation and translation in the rigid transformation

Given two pairs of matched keypoints $p_1(x_1, y_1)$, $p_1{}'(x_1{}', y_1{}')$, and $p_2(x_2, y_2)$, $p_2{}'(x_2{}', y_2{}')$, R is the rotation matrix and T is the translation vector, the relationship between two matched pairs can be derived as

$$R \cdot p_1 + T = p_1{}^{'} \tag{C.1}$$

$$\begin{bmatrix} R & T \end{bmatrix} \cdot p_1 = p_1{}^{'} \tag{C.2}$$

$$R \cdot p_2 + T = p_2{}^{'} \tag{C.3}$$

$$\begin{bmatrix} R & T \end{bmatrix} \cdot p_2 = p_2{}^{'} \tag{C.4}$$

In our case, the rotation angle α is the only variable in the rotation matrix while the translation is a two-dimensional vector, then Equation C.2 and Equation C.4 can be rewritten as

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1\cos\alpha - y_1\sin\alpha + t_x \\ x_1\sin\alpha + y_1\cos\alpha + t_y \end{bmatrix} = \begin{bmatrix} x_1{}' \\ y_1{}' \end{bmatrix} \tag{C.5}$$

$$\begin{bmatrix} \cos\alpha & -\sin\alpha & t_x \\ \sin\alpha & \cos\alpha & t_y \end{bmatrix} \cdot \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} x_2\cos\alpha - y_2\sin\alpha + t_x \\ x_2\sin\alpha + y_2\cos\alpha + t_y \end{bmatrix} = \begin{bmatrix} x_2{}' \\ y_2{}' \end{bmatrix} \tag{C.6}$$

$$x_1\cos\alpha - y_1\sin\alpha + t_x = x_1{}' \tag{C.7}$$

$$x_1\sin\alpha + y_1\cos\alpha + t_y = y_1{}' \tag{C.8}$$

$$x_2\cos\alpha - y_2\sin\alpha + t_x = x_2{}' \tag{C.9}$$

$$x_2\sin\alpha + y_2\cos\alpha + t_y = y_2{}' \tag{C.10}$$

then

$$\cos\alpha(x_2 - x_1) + \sin\alpha(y_1 - y_2) = x_2{}' - x_1{}' \tag{C.11}$$

$$\cos\alpha = \frac{x_2{}' - x_1{}' - \sin\alpha(y_1 - y_2)}{x_2 - x_1} = \sqrt{1 - \sin^2\alpha} \tag{C.12}$$

$$\frac{[x_2{}' - x_1{}' - \sin\alpha(y_1 - y_2)]^2}{(x_2 - x_1)^2} = 1 - \sin^2\alpha \tag{C.13}$$

$$\left[(x_2 - x_1)^2 + (y_2 - y_1)^2\right]\sin^2\alpha - 2(x_2{}' - x_1{}')(y_1 - y_2)\sin\alpha + (x_2{}' - x_1{}')^2 - (x_2 - x_1)^2 = 0 \tag{C.14}$$

Let

$$a = (x_2 - x_1)^2 + (y_2 - y_1)^2 \tag{C.15}$$

$$b = -2(x_2{}' - x_1{}')(y_1 - y_2) \tag{C.16}$$

$$c = (x_2{}' - x_1{}')^2 - (x_2 - x_1)^2 \tag{C.17}$$

It can be concluded that

$$\sin\alpha = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{C.18}$$

$$\alpha = \arcsin(\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}) \tag{C.19}$$

$$t_x = x_1{}' - x_1 \cos\alpha + y_1 \sin\alpha \tag{C.20}$$

$$t_y = y_1{}' - x_1 \sin\alpha - y_1 \cos\alpha \tag{C.21}$$

There are four possible solutions to α in the range of $[-\pi, \pi]$. The logic to determine the correct solution is discussed in section 2.5.3.

D More CNN detection results

The network is also tested in different weather conditions to evaluate the robustness. Figure D.1, D.2 and D.3 show some detection results in sunny, cloudy and rainy day correspondingly. Despite of the weather changing, cones are correctly classified, indicating good robustness.

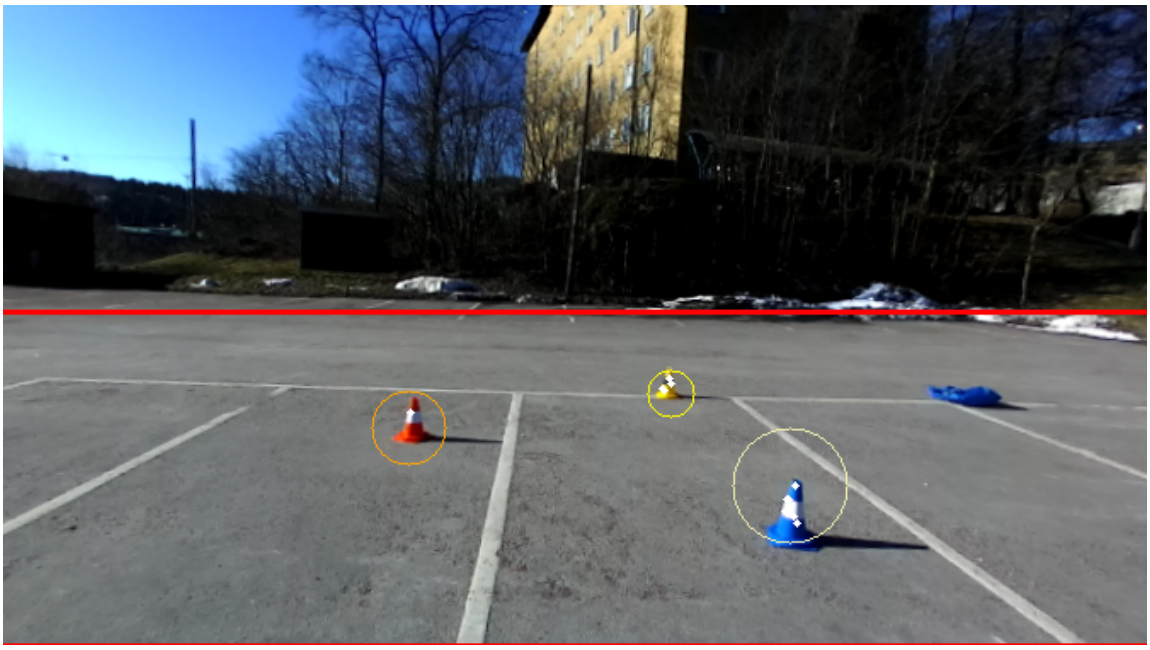


Figure D.1: *CNN detection result on a sunny day.*

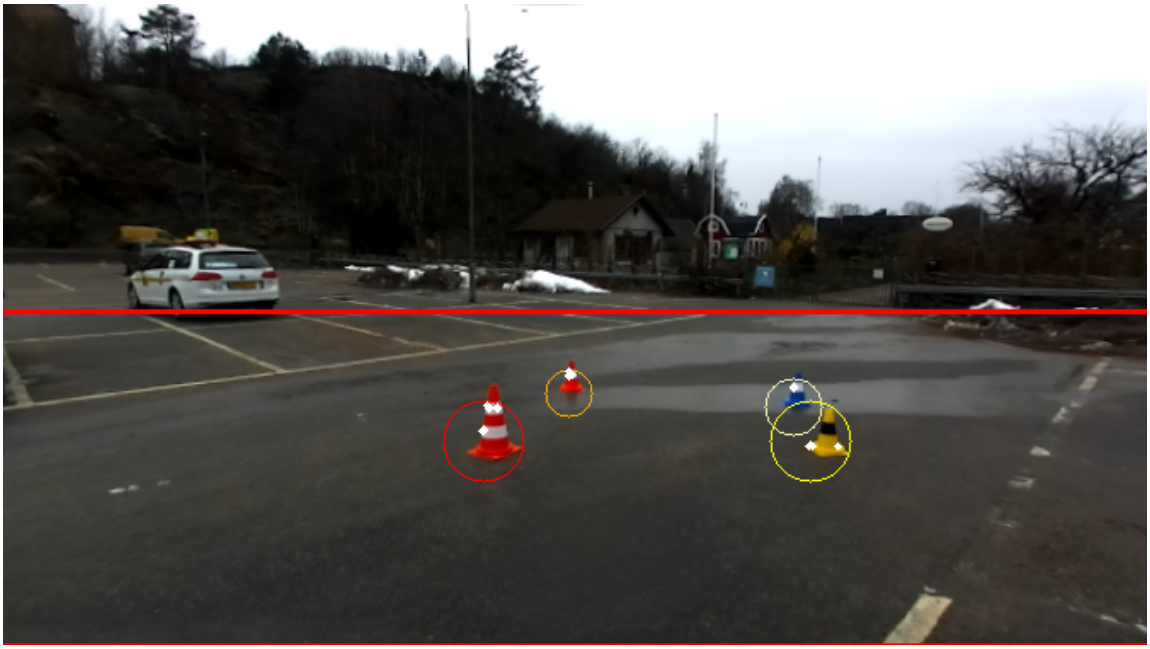


Figure D.2: *CNN detection result on a cloudy day.*

Figure D.3: *CNN detection result on a rainy day.*

References

[1] H. Bay, T. Tuytelaars, and L. V. Gool. *SURF: Speeded Up Robust Features*. Springer Berlin Heidelberg, 2006, pp. 404–417.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM* **18**.9 (1975), 509–517.

[3] O. Bottema and B. Roth. Theoretical kinematics. *NORTH-HOLLAND PUBL. CO., N. Y., 1979, 558* (1979).

[4] L. Bottou. "Large-scale machine learning with stochastic gradient descent". *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[5] J.-Y. Bouguet. *Camera Calibration Toolbox for Matlab*. 2018. URL: http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/ref.html.

[6] P. S. Bullen, D. S. Mitrinovic, and M Vasic. *Means and their Inequalities*. Vol. 31. Springer Science & Business Media, 2013.

[7] M. Calonder et al. "Brief: Binary robust independent elementary features". *European conference on computer vision*. Springer. 2010, pp. 778–792.

[8] R. Chatila and J.-P. Laumond. "Position referencing and consistent world modeling for mobile robots". *Robotics and Automation. Proceedings. 1985 IEEE International Conference on*. Vol. 2. IEEE. 1985, pp. 138–145.

[9] T. Cover and P. Hart. Nearest neighbor pattern recognition. *IEEE transactions on information theory* **13**.1 (1967), 21–27.

[10] A. J. Davison et al. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (2007), 1052–1067.

[11] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-Supervised keypoint Detection and Description. *arXiv preprint arXiv:1712.07629* (2017).

[12] T. Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)* **27**.3 (1995), 326–327.

[13] R. DiPietro. *A Friendly Introduction to Cross-Entropy Loss*. https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/. May 2008.

[14] Z. Fan. *Brief Review on Visual SLAM: A Historical Perspective*. 2016. URL: https://fzheng.me/2016/05/30/slam-review/.

[15] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Readings in Computer Vision* (1987), 726–740.

[16] FSG19. *Rules for FSG 2019*. 2018. URL: https://www.formulastudent.de/fsg/rules/.

[17] A. Garcia-Garcia et al. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857* (2017).

[18] D. T. Girshick Ross Donahue Jeff and J. Malik. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 580 –587.

[19] F. Gustafsson et al. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing* **50**.2 (2002), 425–437.

[20] C. Harris and M. Stephens. "A combined corner and edge detector." *Alvey vision conference*. Vol. 15. 50. Citeseer. 1988, pp. 10–5244.

[21] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[22] G. E. Hinton and R. R. Salakhutdinov. "Replicated softmax: an undirected topic model". *Advances in neural information processing systems*. 2009, pp. 1607–1614.

[23] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology* **195**.1 (1968), 215–243.

[24] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International journal of forecasting* **22**.4 (2006), 679–688.

[25] ILSVRC. *Large Scale Visual Recognition Challenge (ILSVRC)*. 2019. URL: http://www.image-net.org/challenges/LSVRC/.

[26] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE* **92**.3 (2004), 401–422.

[27] B. L. Kalman and S. C. Kwasny. "Why tanh: choosing a sigmoidal function". *Neural Networks, 1992. IJCNN., International Joint Conference on*. Vol. 4. IEEE. 1992, pp. 578–581.

[28] E. Karami, S. Prasad, and M. Shehata. Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. *arXiv preprint arXiv:1710.02726* (2017).

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". *Advances in neural information processing systems*. 2012, pp. 1097–1105.
[31] G. Latham. *Evaluation of Stereo Vision Algorithms*. 2019. URL: http://inside.mines.edu/~whoff/courses/EENG510/projects/2014/Latham.pdf.
[32] W. Liu et al. "Ssd: Single shot multibox detector". *European conference on computer vision*. Springer. 2016, pp. 21–37.
[33] A. L. Maas, A. Y. Hannun, and A. Y. Ng. "Rectifier nonlinearities improve neural network acoustic models". *Proc. icml*. Vol. 30. 1. 2013, p. 3.
[34] J. J. Moré. "The Levenberg-Marquardt algorithm: implementation and theory". *Numerical analysis*. Springer, 1978, pp. 105–116.
[35] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics* **31**.5 (2015), 1147–1163.
[36] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.
[37] P. Newman and K. Ho. "SLAM-loop closing with visually salient features". *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE. 2005, pp. 635–642.
[38] D. Nister. An efficient solution to the five-point relative pose problem. English. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**.6 (2004), 756–770.
[39] D. Nistér, O. Naroditsky, and J. Bergen. "Visual odometry". *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 1. Ieee. 2004, pp. I–I.
[40] T. Nomi. *tiny-dnn documentations*. https://tiny-dnn.readthedocs.io/en/latest/. 2016.
[41] J. D. Owens et al. GPU computing. *Proceedings of the IEEE* **96**.5 (2008), 879–899.
[42] D. M. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation (2011).
[43] J. Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". English. IEEE, 2016, pp. 779–788.
[44] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection". *European conference on computer vision*. Springer. 2006, pp. 430–443.
[45] E. Rublee et al. "ORB: An efficient alternative to SIFT or SURF". *Computer Vision (ICCV), 2011 IEEE international conference on*. IEEE. 2011, pp. 2564–2571.
[46] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).
[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[48] N. Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* **15**.1 (2014), 1929–1958.
[49] K. T and O. M. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16** (1994), 920 –932.
[50] A. P. Tafti et al. A comparative study on the application of SIFT, SURF, BRIEF and ORB for 3D surface reconstruction of electron microscopy images. English. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging Visualization* (2016), 1–14.
[51] S. Thrun. Probabilistic robotics. *Communications of the ACM* **45**.3 (2002), 52–57.
[52] J. Wang and L. Perez. *The effectiveness of data augmentation in image recognition using deep learning*. Tech. rep. Technical report, 2017.
[53] wikipedia. *Convolutional neural network*. 2019. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network.
[54] wikipedia. *Image rectification*. 2018. URL: https://en.wikipedia.org/wiki/Image_rectification.
[55] wikipedia. *Image segmentation*. 2019. URL: https://en.wikipedia.org/wiki/Image_segmentation.
[56] wikipedia. *Visual odometry*. 2019. URL: https://en.wikipedia.org/wiki/Visual_odometry#Feature_Based_and_Direct_Method.
[57] W. L. Yumeng Jiang. *all the codes of the thesis*. https://github.com/liweim/supervised-detector. Accessed November 11, 2018.
[58] ZED. *ZED stereo camera*. 2019. URL: https://www.stereolabs.com/zed/.