Dissertation

Doctor of Philosophy

A STUDY ON AN OBSTACLES DETECTION SYSTEM EMPLOYING A CAR-MOUNTED CAMERA

by

Shaohua Qian

Supervised by
Associate Professor Joo Kooi Tan
Professor Seiji Ishikawa

Graduate School of Engineering
Department of Mechanical and Control Engineering
Kyushu Institute of Technology
JAPAN

2014

博士論文

車載カメラを用いた 障害物検出システムの研究

Dissertation
Doctor of Philosophy

A Study on an Obstacles Detection System Employing a Car-mounted Camera

銭　少華
学生番号　11584207

指導教員　　タン　ジュークイ　准教授
　　　　　　石川　聖二　教授

九州工業大学　大学院　工学府
機械知能工学専攻　知能制御工学コース
平成 26 年度

Abstract

In recent years, autonomous collision avoidance systems have been researched and developed for realizing safe driving using cameras and sensors. These systems are designed to warn the drivers the presence of obstacles on the road and help them take a necessary action in advance. In these systems, the ability to detect obstacles is essential.

Although various methods of obstacles detection have already been reported, these existing obstacles detection methods have some inadequacies: Some of them can be only used to detect moving obstacles; Some of them cannot extract the shape of obstacles, and they only use a rectangular frame that surrounds an obstacle to represent a detected obstacle; Some of them can only be used to detect one kind of specific object, such as pedestrian detection or vehicle detection.

In order to make up for the inadequacies of the existing obstacles detection method, in this thesis, a method is proposed for detecting obstacles on a road by the employment of the background modeling and the road region detection.

In obstacles detection, true obstacles are defined as arbitrary objects which protrude from the ground plane in the road region, including static and moving objects. Road marks in the road region and objects outside the road region are considered as false obstacles. The output of this obstacles detection method is based on the obstacles’ shape.

In this thesis, we also propose a method of classifying 2D objects and 3D objects. The results of 2D objects and 3D objects classification can be used in the resultant image of obstacles detection to delete 2D objects (such as road marks) and improve the accuracy of obstacles detection.

The originalities of this thesis are as follows:

In the first place, the proposed method can detect arbitrary objects including both static objects and moving objects. This is helpful because static objects such as boxes fallen on the road from a car are dangerous for drivers.

In the second place, the output of the proposed method is the shape of obstacles. Extraction of the shape of an obstacle is important for obstacles recognition. If the detected obstacle is recognized as a pedestrian from its shape, we can foresee his/her next motion.

In the third place, the proposed method can distinguish which objects are 3D objects, and which objects are 2D objects in a pile of objects using a monocular camera. It is useful in the obstacles detection and other applications, such as navigation of walking robots.

In the performed experiments, it is shown that the proposed obstacles detection method is able to extract the shape of both static and moving obstacles in a frontal view from a car.

Acknowledgements

It gives me great pleasure in expressing my gratitude to all those people who have supported me and had their contributions in making this thesis possible.

First and foremost, I would like to express my deepest gratitude and gratefulness to my academic supervisors, Professor Seiji Ishikawa and Associate Professor Joo Kooi Tan, who have done a great favor to my thesis. From the guiding of the research to the revision of the thesis, I have benefited greatly from their patience, encouragement and excellent guidance. What's more, I am deeply moved by their serious attitude towards academic work.

I express my sincere acknowledgements to Professor Hyoungseop Kim and Professor Takashi Morie for their valuable suggestions as the co-supervisors of my thesis.

I would like to show my thankfulness to my current and past lab members for their kind co-operation and helpfulness in accomplishing my experiments and make my university life smooth. This four and a half years' experience of studying in Japan means a lot to me. I would like to thank all people I met here; you gave me an unforgettable memory.

I would also like to thank my respectable teacher Associate Professor Ling Gu in Yangzhou University, my best friends in china, especially Han Zhou, LinLin Xie, XiaoDong Sun, Dawei Zhang, MinMin Huang, JingJing Lu, Jing Liu, Yan Ma, for helping me get through the difficult times, and for all the emotional support and entertainment.

Last, but by no means least, I give my special gratitude to my father Jinde Qian and my mother Fanhong Meng for always believing me and encouraging me to follow my dreams. They bore me, raised me, supported me, taught me, and loved me. To them I dedicate this thesis.

Contents

Chapter 1　Introduction

1.1　Background

Vehicle has brought great convenience to people’s lives as one of the main tools of transport in modern society. However, because of substantial increase of the vehicles in recent years, the phenomenon of traffic congestion has become more and more serious, and traffic pollution and accidents have caused social general attention. To solve these difficulties, Intelligent Transportation Systems (ITS) has been developing rapidly in recent years.

The Intelligent Transport Systems [1] are new transport systems which are comprised of an advanced information and telecommunication network for users, roads and vehicles. It contributes to solving problems such as traffic accidents and congestions. Through the implementation of ITS, we can improve transportation system performance, including reducing traffic congestion, traffic accidents, energy consumption and environmental pollution. Therefore, ITS have been considered as the important development direction of future transportation systems. Japan has the most wide application of ITS in the world: Actually ‘The Vehicle Information and Communication System’ (VICS) is already quite mature in Japan. ITS are the complex integrated systems, and can be separated into nine development areas [2]; advances in navigation systems, electronic toll collection systems, assistance for safe driving, optimization of traffic management, increasing efficiency in road management, support for public transport, increasing efficiency in commercial vehicle operations, support for pedestrians, and support for emergency vehicle operations. Among them, the purpose of assistance for a safe driving system is to develop various vehicle control techniques, such as safe driving and autonomous navigation techniques. Because the traditional vehicles cannot meet the needs of ITS, Intelligent Vehicle (IV) technologies have aroused at this historic moment.

At present, the studies on intelligent vehicles are mainly devoted to a safe driving system for the reason of improving the security and comfort. One important indicator of improving security is reducing traffic accidents. According to the report published by ministry of transport of Japan national police agency [3], 629,021 traffic accidents happened in 2013, and casualties reached 785,867 people, including 4,373 deaths. The analysis of traffic accidents shows that, 80% traffic accidents were due to not prompt reaction and improper handling. According to the statistics of American highway driving safety management committee [4], if a reminder is provided to the drivers 0.5 seconds earlier, the incidence of traffic accidents can be reduced by 60%. So we see if the drivers can get a reminder of the danger information, they may be able to react quickly and make the corresponding actions accurately, and then many of those accidents could be avoided or alleviated.

In recent years, the autonomous collision avoidance systems have been developed rapidly for realizing safe driving to prevent car accidents [5-6]. These systems use camera, radar or other sensors (sometimes a combination of several sensors) to detect an imminent collision, then either provide a warning to the driver or take action autonomously (by braking or steering or both) without any driver’s action.

Recently, Subaru’s Eyesight system and Volvo’s Pedestrian and Cyclist Detection with Full Auto Break system are two most famous commercial collision avoidance systems. Subaru’s Eyesight system can detect pedestrian, bicycles and vehicles on the road when a vehicle equipped with this system drives in the speed less than 30km/h. This Eyesight system uses a stereo camera, and uses a rectangular frame that surrounds an obstacle to represent a detected obstacle. Volvo’s Pedestrian and Cyclist Detection with Full Auto Break system uses a camera and a radar to detect pedestrians and cyclists in certain situations, such as one swerving out in front of the car. This system also uses a rectangular frame that surrounds an obstacle to represent a detected obstacle.

We know that safe driving of a car depends heavily on vision. This is why so many accidents happen at night or when the visibility is reduced by weather conditions such as fog. The vision of a driver can be improved by the systems that give information on the environment around the vehicle that cannot be seen or hardly seen by human eyes. Therefore, a vision-based obstacles detection system is the mainstream of current researchers. Detection of obstacles in video sequences is a basic task in this system. Accurate obstacles detection will improve the performance of obstacles tracking, recognition, classification and their motion analysis.

1.2 Previous Work

The existing obstacles detection methods can be separated into three categories [7]:

1) The first method uses a monocular static camera. This method detects obstacles based on the optical flows [8-10]. It is often separated into three steps. In the first place, the optical flow is calculated using two adjacent frames in image sequences. In the second place, the main movement direction of vehicles is estimated. Finally, the obstacles are decided based on the optical flows which are inconsistent with the main movement direction of vehicles. This method needs huge calculation and it is sensitive to vehicle motion. It cannot detect static obstacles. It can only be used to

detect moving obstacles.

2) The second method uses a monocular moving camera. This method detects obstacles based on features. It is often used when the obstacles are defined as a specific kind of objects. Because these obstacles have some specific and obvious features, this detection can be based on searching for these features, such as shape [11-13] or symmetry [14]. This method can only be used to detect one kind of specific object, such as pedestrian detection [15-16] or vehicle detection [17-18].
3) The third method is based on stereo vision [19-20]. Scene images are captured using two or more cameras from different angles simultaneously, and then the obstacles are detected through matching. This method needs huge amount of calculation and sensitive to vehicle motion.

These existing methods have some inadequacies:

In the first place, although the third method uses two or more cameras, it is generally accepted that the method which uses a monocular camera is much better because of economic aspect and of processing time. Actually the method using a monocular camera is easier to achieve real-time processing.

In the second place, unlike the first method which detects only moving obstacles, a method which can detect both moving and static objects simultaneously is necessary. It is because static objects such as boxes fallen on the road from a car are also dangerous for drivers. It is, however, noted that a 2D static object such as a paper on the road is not dangerous for driving.

In the third place, most of the existing methods cannot extract the shape of obstacles. They only use a rectangular frame that surrounds an obstacle to represent a detected obstacle.

1.3 Objective of the Thesis

In order to make up for the inadequacies of the existing obstacles detection method, in this thesis, we propose an obstacles detection method using a vehicle-mounted monocular camera. This camera records the road environment in front of a vehicle when the vehicle is moving, and the computer begins to categorize these captured images in order to differentiate obstacles from ordinary objects. The output of this method is based on the obstacles' shape. After having obtained the obstacle information, the drivers can react quickly and make corresponding actions accurately to prevent car accidents. Here *true obstacles* are defined as arbitrary objects which protrude from the ground plane in the road region, including static and moving objects. Road marks in the road region (e.g., zebra crossings) and objects outside the road region are considered as *false obstacles*.

The most significant differences between the proposed method and the existing obstacles detection methods are as follows:

In the first place, the proposed method detects arbitrary objects, irrespective of static objects or moving objects, which may pose a threat to safe driving on the road, not just specific objects which have been detected. This is helpful because even the objects which have fallen on the road from a car are dangerous for drivers. The existent methods, however, concentrate only on detecting moving objects such as pedestrians, bicycles and cars.

In the second place, the output of the proposed method is the shape of obstacles. Currently, existing obstacles detection methods cannot extract the shape of obstacles. They only use a rectangular frame that surrounds an obstacle to represent a detected obstacle. This shape information is important for obstacles recognition and classification. If the detected obstacles are judged as a pedestrian, we can use the shape to carry out his/her motion recognition.

In the third place, the proposed method can be applied for the speed up to 45 km/h which is usually the speed limit within the city.

1.4 Organization of the Thesis

The organization of the thesis is as follows:

In Chapter 2, we propose a method of automatic obstacles detection for detecting obstacles on a road by use of background modeling and road region detection.

In Chapter 3, we propose a method of classifying 2D objects and 3D objects. The results of 2D objects and 3D objects classification can be used in the resultant image of obstacles detection in Chapter 2 to delete 2D objects and improve the accuracy of obstacles detection.

In Chapter 4, we summarize the obstacles detection method which we have proposed in Chapter 2 and Chapter 3. In order to prove the effectiveness of the proposed methods, we also introduce a comparative obstacles detection method. We carry out the comparative experiment using the same experimental videos to discuss the effectiveness of the proposed obstacles detection method.

Finally, the thesis is concluded in Chapter 5.

Chapter 2 Obstacles Detection

In this chapter, we propose an obstacles detection method using a vehicle-mounted monocular camera. This camera records the road environment in front of a vehicle when the vehicle is moving, and the computer begins to categorize these captured images in order to differentiate obstacles from ordinary objects. The output of this method is based on the obstacles' shape [21]. After having obtained the obstacle information, the drivers can react quickly and make corresponding actions accurately to prevent car accidents. Here correct obstacles are defined as arbitrary objects which protrude from the ground plane in the road region, including static and moving objects. Road marks in the road region (e.g. zebra crossings) and objects outside the road region are considered as noises, they are incorrect obstacles.

2.1 Outline of the Proposed Method

When a car is moving forward, stationary objects in a frontal scene are considered as the background, and the foreground can be obtained based on the background model. Because the road has almost no texture, the road can be considered to be static in the frontal video image and is regarded as the background. In this condition, the foreground image which is obtained from the background model contains obstacles on the road (including static and moving objects), road marks in the road region and the objects outside the road region. In order to extract the shape of the obstacles in the foreground image, the following operations are employed. First, the road region is detected using Support Vector Machine (Section 2.3). Second, non-road region in the result of the road region detection is classified as noise region and obstacles region (Section 2.4.2). After the region classification, we have three kinds of regions, noise region, obstacles region and road region. All the objects inside the noise region and the road region in the foreground image are considered as noises and deleted using the result of region classification. Finally the shape of the obstacles (e.g., pedestrians, boxes, etc.) in the foreground image is extracted. Figure 2.1 shows the flowchart of the proposed obstacles detection method.

2.2 Background Modeling

Background modeling is a method of background reconstruction which is often used to detect moving objects in computer vision, with applications to several fields, such as video surveillance [22-23] and target tracking [24].

In the ideal situation, the background image can be simply acquired when the scene doesn’t include any moving object. However, in the realistic situation, the scene is always changing, such as illumination change, objects introduced or removed from the scene. Many background modeling methods have been developed to deal with these problems. These background modeling methods can be classified into following categories [25]: Basic Background Modeling [26], Statistical Background Modeling [27-28], Fuzzy Background Modeling [29-31] and Background Estimation [32-34].

Among these background modeling methods, the Gaussian Mixture Model (GMM) is the most used model which was proposed by Stauffer and Grimson [24].

Pfinder [27] used a single Gaussian distribution to model the values of a particular pixel and to get the background model. When this method is applied in an indoor scene, the output is good; but not good for outdoor scenes. Rather than modeling the values of one

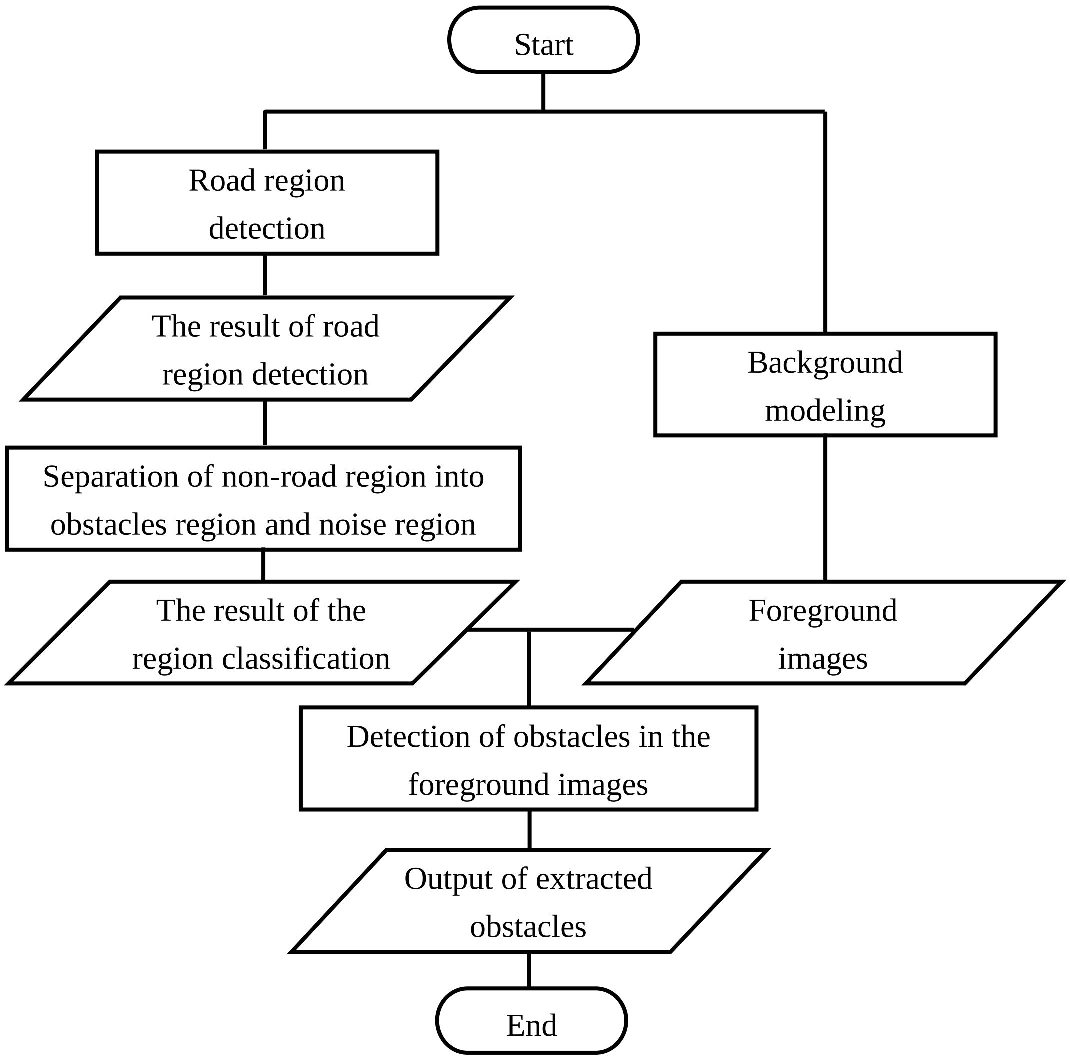

Start
Road region detection
The result of road region detection
Separation of non-road region into obstacles region and noise region
The result of the region classification
Background modeling
Foreground images
Detection of obstacles in the foreground images
Output of extracted obstacles
End

Fig. 2.1 Flowchart of the proposed obstacles detection method.

pixel as one Gaussian distribution, the GMM modeled the values of a particular pixel as mixture of Gaussian distributions. Different Gaussians are assumed to represent different gray values. Based on the mean value and the variance of each Gaussian in the mixture, they determine which Gaussians may correspond to the present background. Pixels that do not fit the background distributions are considered as foreground. To allow the model to adapt to changes in illumination and run in real-time, an update algorithm was applied. It is based upon selective updating. This method is capable of dealing with lighting changes, repetitive motions of scene elements, tracking through cluttered regions, slow-moving objects, and introducing or removing objects from the scene.

The GMM is robust when employed in a fixed camera case. But, in this research, the employed camera is moving since it is mounted on a car. When this car is moving on the road, the employed camera is as well moving. In order to employ the GMM in a moving camera case, we need to construct a virtual scene based on a real scene. We then employ the GMM in this virtual scene in reconstructing the background.

2.2.1 Virtual Scene Construction

In this research, a camera is mounted on a vehicle, when the vehicle is moving, the camera is moving as well. This is the real situation.

But when we see the frontal scene in the frontal video image, the camera can be considered to be static, and then buildings, the road and static objects are moving according to the relative motion. Moreover, since the road has almost no texture, we can assume that the road is static in the frontal video image. Thus, the virtual scene will be defined as the frontal scene (in the videotaped image) with the assumption of the road being static. In this virtual scene, the camera is static; the road area which is classified as the background is static; objects (including static and moving objects) and pedestrians on the road, buildings, road marks and zebra crossings which are classified as the foreground are moving. Then we employ the GMM to reconstruct the background in this moving camera case.

2.2.2 Gaussian Mixture Model

The sequence of a particular pixel is a time series of pixel values, i.e. the pixel values of a particular pixel in an image sequence over time. At any time t (t=1,2,…,T), the sequence of a particular pixel (x_0, y_0) is given by

$$\{X_1,..., X_T\} = \{I(x_0, y_0, t): 1 \le t \le T\} \tag{2.1}$$

where *I* is the gray value of pixel (x_0, y_0).

For the sequence of a particular pixel, $\{X_1, \ldots X_T\}$, *K* Gaussian distributions are used to model these pixel values. The probability of the pixel value X_t is

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \tag{2.2}$$

where *K* is the number of Gaussian distributions in the mixture and it is determined by the available memory and computational power. Currently, *K* = 3 to 5 are used. $\omega_{i,t}$ is the weight of the i^{th} Gaussian distribution at time *t*; $\mu_{i,t}$ is the mean value of the i^{th} Gaussian distribution at time *t*; $\Sigma_{i,t}$ is the covariance matrix of the i^{th} Gaussian distribution at time *t*, and η is a Gaussian probability density function defined by

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{1}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu)^T \Sigma^{-1} (X_t - \mu)} \tag{2.3}$$

For computational reasons, Stauffer and Grimson assumed that the red, green, and blue pixel values are independent and have the same variances. This assumption can avoid a costly matrix inversion at the expense of some accuracy. So the covariance matrix is assumed to be of the form:

$$\Sigma_{i,t} = \sigma_{i,t}^2 I \tag{2.4}$$

where *I* is a unit matrix; $\sigma_{i,t}^2$ is the variance value of the i^{th} Gaussian distribution at time *t*.

Thus, each pixel is characterized by a mixture of *K* Gaussian distributions. Once the background model is defined, the different parameters of the Gaussian Mixture Model must be initialized. The parameters of the GMM are the number of Gaussians *K*, the weight $\omega_{i,t}$ associated to the i^{th} Gaussian at time *t*, the mean value $\mu_{i,t}$ and the variance value $\sigma_{i,t}^2$.

Once the parameters initialization is done, the first foreground detection is performed and then the parameters are updated.

2.2.3 Foreground Detection

In order to reconstruct the background, a method for deciding what portion of the mixture model represents the background is needed.

We are interested in the Gaussian distributions which have high weight and low variance. Because, when a static persistent object is visible, the background distributions have high weights and the relatively low variances. In contrast, when a new object occludes the background, it will not match any of the existing distributions, resulting in either increasing a new distribution or replacing the existing distribution with a new distribution having low weight and high variance. Also, the variance of the moving object is expected to remain larger than a background pixel until the moving object stops.

Firstly, the existing Gaussians are ordered by the value of ω / σ. This value increases when the weight increases and the variance decreases. This ordering supposes that a background pixel corresponds to a high weight with a low variance due to the fact that the background is more present than moving objects. This ordering of the model gives an effective ordered list, where the most likely background distributions remain on the top and the less probable transient background distributions are on the bottom and are eventually replaced by new distributions.

The first *B* Gaussian distributions which exceed certain threshold *T* are considered to represent a background distribution:

$$B = \arg\min_b \left(\sum_{i=1}^{b} \omega_{i,t} > T \right) \tag{2.5}$$

Other distributions are considered to represent foreground distributions.

Then, a match test is carried out for each pixel of the new frame taken at time *t*+1. Every new pixel value, X_{t+1}, is checked against *K* Gaussian distributions to find if it matches one of those distributions. The match (see Fig. 2.2) is defined by

$$\frac{\left| X_{t+1} - \mu_{i,t} \right|}{\sigma_{i,t}} < T_{gauss} \tag{2.6}$$

Then, two cases can occur:

Case 1: A match is found with one of the *K* Gaussians. In this case, if the matched Gaussian distribution represent a background distribution, the current pixel X_{t+1} is classified as background else the current pixel X_{t+1} is classified as foreground.

Case 2: If none of the *K* Gaussians match the current pixel X_{t+1}, the current pixel X_{t+1} is classified as foreground.

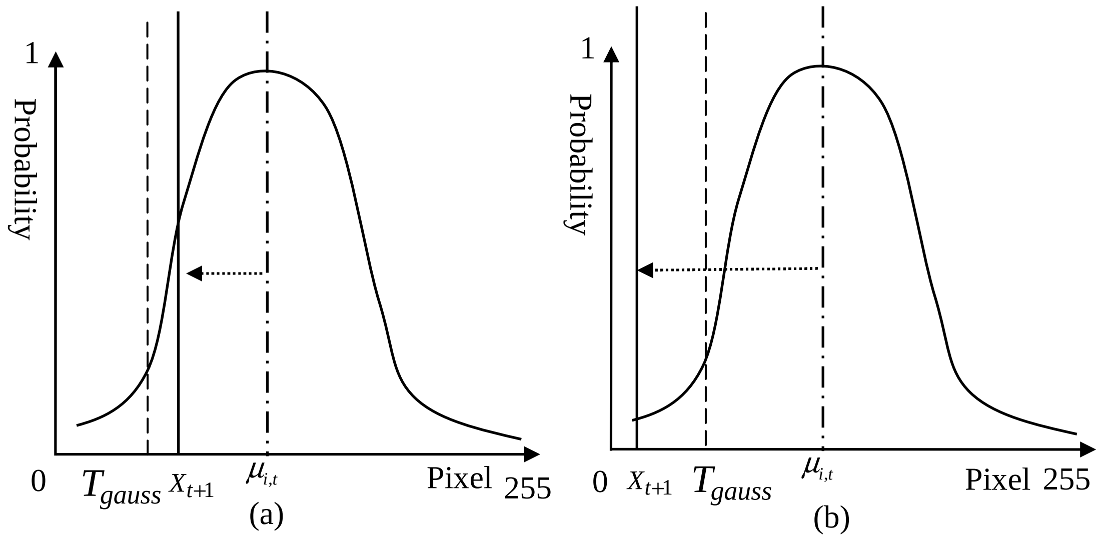


Fig. 2.2 The matching relationship between a pixel and the background model. (a) Pixel X_t matches the distribution, (b) the unmatched case.

2.2.4 Updating of Model Parameters

The updating algorithms of GMM are different for two cases.

Case 1: A match is found with one of the *K* Gaussians.

For the matched Gaussian distribution, the update goes as follows:

$$\omega_{i,t+1} = (1-\alpha)\omega_{i,t} + \alpha \tag{2.7}$$

where α is a constant learning rate.

$$\mu_{i,t+1} = (1-\rho)\mu_{i,t} + \rho X_{t+1} \tag{2.8}$$

$$\sigma_{i,t+1}^2 = (1-\rho)\sigma_{i,t}^2 + \rho\left(X_{t+1} - \mu_{i,t+1}\right)^T \left(X_{t+1} - \mu_{i,t+1}\right) \tag{2.9}$$

where

$$\rho = \alpha\eta\left(X_{t+1}, \mu_i, \Sigma_i\right) \tag{2.10}$$

For the unmatched Gaussian distributions, the parameters are unchanged, only the weight is replaced by:

$$\omega_{i,t+1} = (1-\alpha)\omega_{i,t} \tag{2.11}$$

Case 2: No match is found with any of the *K* Gaussians.

In this case, the least probable distribution is replaced by a new distribution with the parameters:

$$\omega_{i,t+1} = Low \quad Prior \quad Weight \tag{2.12}$$

$$\mu_{i,t+1} = X_{t+1} \tag{2.13}$$

$$\sigma^2_{i,t+1} = High \quad Initial \quad Variance \tag{2.14}$$

Once the parameters are updated as above, the foreground detection can be carried out using the next new frame taken at time *t*+2.

This updating algorithm has one significant advantage. It need not destroy the existing model of the background when a moving object stops move and is stationary long enough to become a part of the background. The original background color remains in the mixture model until it becomes the last probable distribution and is replaced by a new model. Therefore, if this object moves again, the distribution describing the previous background still exists with the same μ and σ^2, but a lower ω will be quickly re-incorporated into the background.

2.3 Road Region Detection

Because the camera is moving, the foreground which is obtained from the background modeling often contains a lot of noises. These noises are mostly caused by the objects outside the road region. In order to delete these noises, we need to detect the road region.

In advanced driving assistance systems, it is important to be able to detect the region covered by the road in images. To extract the road region in a general road scene, a method using stereo cameras has been proposed [35]. This method assumes that the stereo cameras are calibrated, and it estimates only those parameters relating to the 3D road plane. By using calibrated stereo cameras, the road region can be estimated in a stable manner. For practical use, however, monocular camera systems are preferred to stereo cameras systems, because monocular systems have advantages in terms of reduced costs. On the other hand, some methods that make use of a monocular camera have also been proposed [36]. For stable estimation of the road region, they approximate the vehicle ego-motion by considering a reduced number of motion parameters. However, when a camera is mounted on a vehicle, other motion parameters are not negligible. Moreover, these methods warp one of the images by using an approximated optical flow model with regard to incremental motion. When the distance between the two images is not close, the accuracy of this model is not sufficient. These issues may cause poor estimation of the road region when using images taken by a vehicle-mounted camera.

In the following sections, we propose a road region detection method using the Support Vector Machine (SVM). This method includes two steps: training and test.

Figure 2.3 shows an outline of the SVM classifier training. In the first frame of the input video, a small sample of pixels is labeled by a human supervisor as a road class or a non-road class (described in Section 2.3.3). For each pixel in this sample, we extract a feature vector by feature extraction (described in section 2.3.2). These feature vectors are considered as the training data of SVM. Then, these training data are used to train a SVM classifier.

Figure 2.4 gives an outline of the testing (classification) using the SVM classifier. In other frames of the input video, a feature vector is extracted for each pixel using the same feature extraction method. These feature vectors are considered as the test data. Then, each pixel in the input video is classified as a road pixel or a non-road pixel using the trained SVM classifier and the test data.

2.3.1 Support Vector Machine

The original Support Vector Machine (SVM) algorithm was invented by Boser, Guyon, and Vapnik in COLT-92. On the other hand, the current standard SVM was proposed by Corinna Cortes and VladimirVapnik in 1995 [37].

The SVM is a useful classification tool that uses a machine learning theory to maximize predictive accuracy while automatically avoiding over-fit to the data. A classification task usually involves with training and test data which consist of some data instances. Each instance in the training data contains one target values and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the test data which are given only attributes.

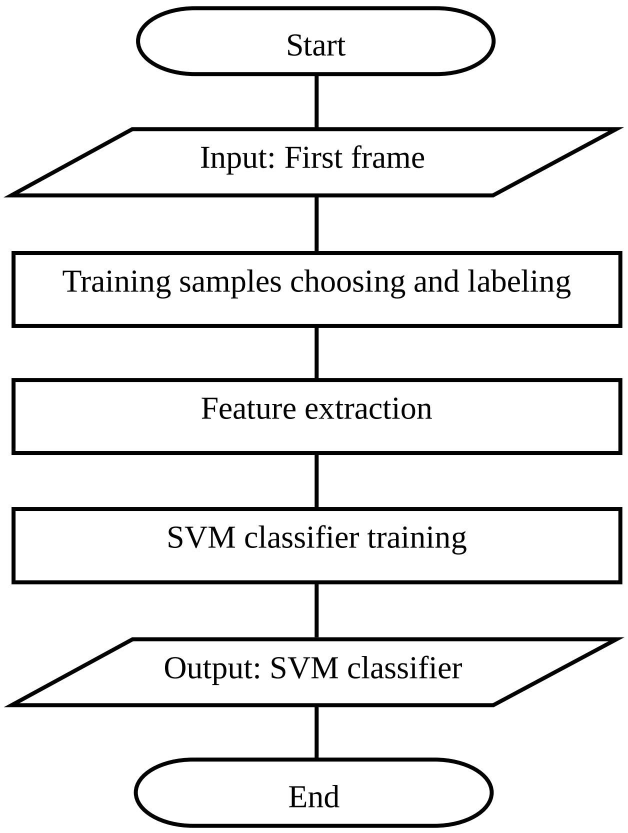


Fig. 2.3 Outline of the SVM classifier training.

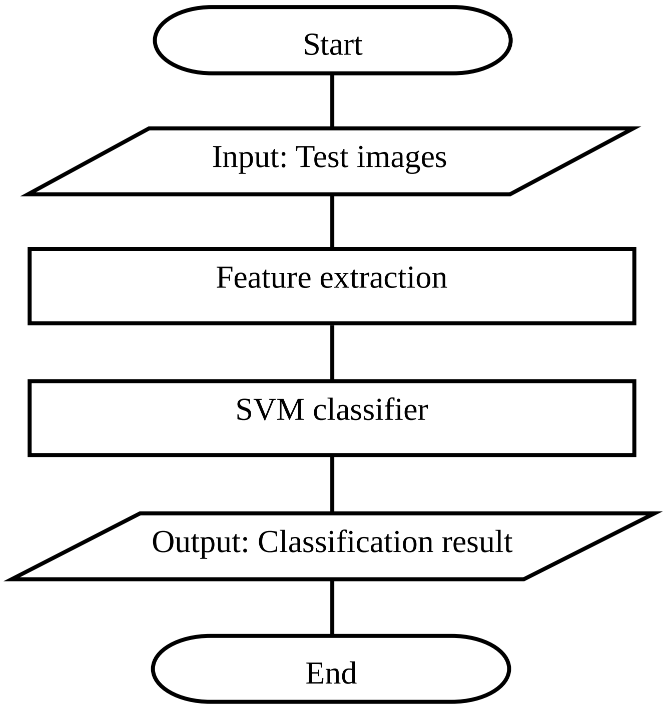


Fig. 2.4 Outline of the testing.

2.3.2 Feature Extraction

In this research, the features we use are color features and texture features. For color features, three features in HSV color space are used. For texture features, five Haralick statistical features [38] are used as follows:

$$Energy = \sum_{u=1}^{N_g}\sum_{v=1}^{N_g}\{p(u,v)\}^2 \tag{2.15}$$

$$Entropy = -\sum_{u=1}^{N_g}\sum_{v=1}^{N_g} p(u,v)\log\{p(u,v)\} \tag{2.16}$$

$$Contrast = \sum_{n=0}^{N_g-1} n^2 \left\{\sum_{u=1}^{N_g}\sum_{v=1}^{N_g} p(u,v)\right\}, |u-v| = n \tag{2.17}$$

$$Inverse \quad difference \quad moment = \sum_{u=1}^{N_g}\sum_{v=1}^{N_g}\frac{1}{1+(u-v)^2}\,p(u,v) \tag{2.18}$$

$$Correlation = \frac{\sum_{u=1}^{N_g}\sum_{v=1}^{N_g}(uv)p(u,v) - \mu_x\mu_y}{\sigma_x\sigma_y} \tag{2.19}$$

where $p(u,v)$ is an element of a Gray Level Co-occurrence Matrix (GLCM). N_g is the size of the GLCM. μ_x, μ_y and σ_x, σ_y are the mean values and variance values calculated from GLCM, respectively.

The algorithm of texture features calculation goes as follows:

1) The gray levels of original images are 256. We reduce the number of gray levels from 256 to 8.
2) The pixels in the small square window of the size 5*5 centered at the current pixel are used to calculate a GLCM. The size of the GLCM is the same as the number of gray levels of the image. Therefore, in the proposed method, the GLCM is an 8*8 matrix. The definition of the GLCM is given below:

   The GLCM is a tabulation of how often different combinations of pixel brightness values (gray levels) occur in an image. GLCM texture considers the relation between two pixels at a time, called the reference and the neighbor pixel. $P_\delta(i, j)$ (i, j = 0,1,2,⋯,7) is the frequency of the reference pixel with the value i and the neighbor pixel with the value j which satisfy a given offset $\delta(D_X, D_Y)$ within the window. $P_\delta(i, j)$ is considered as the value of element (i, j) in GLCM.

3) According to Eqs. (2.15) - (2.19), we calculate the five texture features.

These five texture features and three color features are combined to form an eight-element feature vector as follows:

$$F_{i,j} = [\, f_{t1(i,j)},\, f_{t2(i,j)},\, f_{t3(i,j)},\, f_{t4(i,j)},\, f_{t5(i,j)},\, f_{c1(i,j)},\, f_{c2(i,j)},\, f_{c3(i,j)}\,]$$

$$(i = 1, \cdots, 240 \quad j = 1, \cdots, 320) \tag{2.20}$$

where $f_{t_n(i,j)}$ is the n^{th} Haralick statistical feature at the point (i, j) and $f_{c_n(i,j)}$ is the n^{th} color feature at the point (i, j) in the HSV color space.

2.3.3 Training Data Initialization

The first frame in the input video is used as a training image; the other frames in the input video are used as test images.

In the first frame, the training data is selected and labeled by a human. Two rectangle windows are used by a supervisor to select the training data on the image as shown in Fig. 2.5. A green window is placed in the road region. The pixels located in this green window are labeled as positive samples. A red window is placed outside the road region. The pixels located in this red window are labeled as negative samples. These two samples constitute the training data.

2.4 Detection of Obstacles in the Foreground Images

In this section, we will extract the shape of obstacles in the foreground images which have been detected in Section 2.2 using the road region which have been detected in Section 2.3.

In order to extract the shape of obstacles in the foreground images, we need to delete two kinds of things: road marks in the road region and objects outside the road region in the foreground images. In the following sections, we describe the method of deleting these two kinds of things (noises).

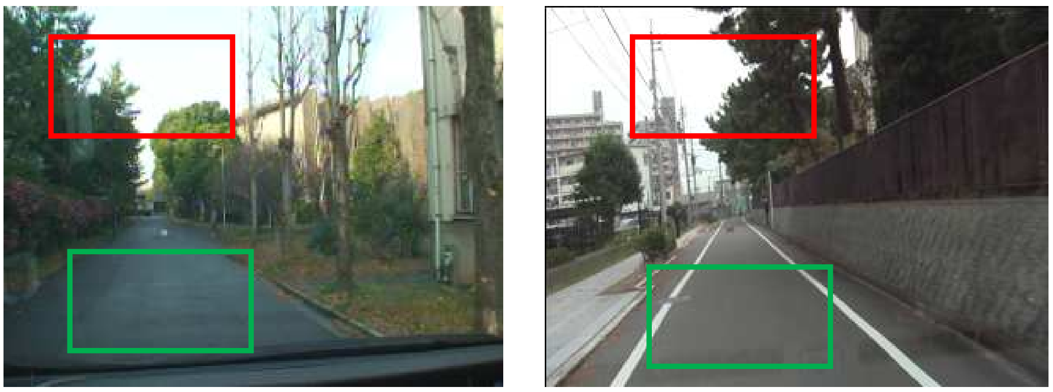


Fig. 2.5 Selection of training data.

2.4.1 Problem Description

Here we will delete road marks in the road region and objects outside the road region in the foreground images.

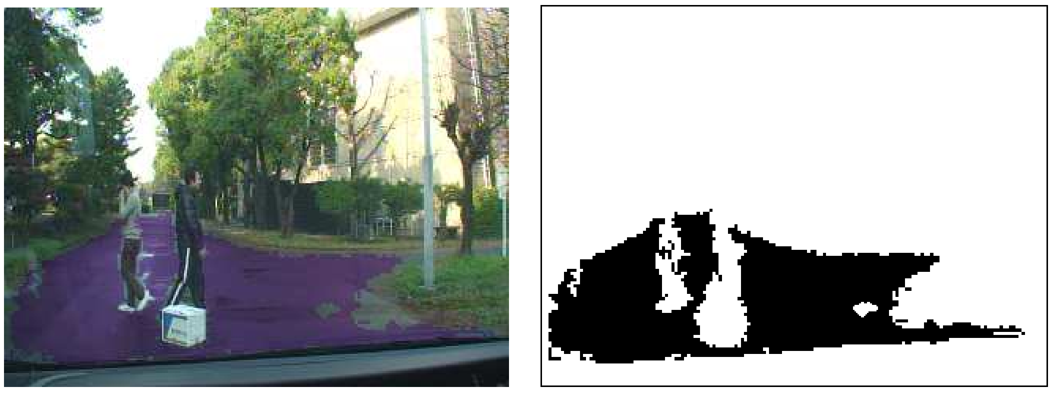
For road marks in the road region, we can use the result of the road region detection to delete them.

For objects outside the road region, if we use the non-road region to delete these objects, it also deletes the obstacles in the road. In order to solve this problem (reserving the obstacles in the road and deleting the objects outside the road), we need to divide the non-road region into obstacles region and noise region (Section 2.4.2). Then we use this noise region to delete the objects outside the road in the foreground images.

2.4.2 Region Classification

Here we will divide the non-road region into obstacles region and noise region. Figure 2.6 (a) and (b) show the result of road region detection and the corresponding road region template image, respectively.

In this road region template image, black pixels are road pixels, whereas white pixels are non-road pixels. If we check the pixels of one particular row in this image, we can get a curve to describe the distribution of these pixel values in this row. Figure 2.7 shows the pixel values distribution of the 140th row in the road region image. Based on this curve, we consider white regions (high values) which have two adjacent black regions (low values) both on the left and right sides as the obstacles region. Because, in this research, the obstacles which we want to detect are defined as the 3D objects located on the road, other 3D objects locate outside the road are no danger to driving (classified as noise outside the road). Based on this definition, we know the obstacles must be located in the road region (surrounded by a road region) or have adjacent road regions on both left and right sides. We check each row in the road region template image to carry out region classification. Figure 2.8 shows the result of the region classification. The obstacle region is indicated by gray pixels, the road region is indicated by black pixels, and the noise region is indicated by white pixels.



(a) (b)

Fig. 2.6 Road region detection: (a) The result of road region detection, (b) road region template image.

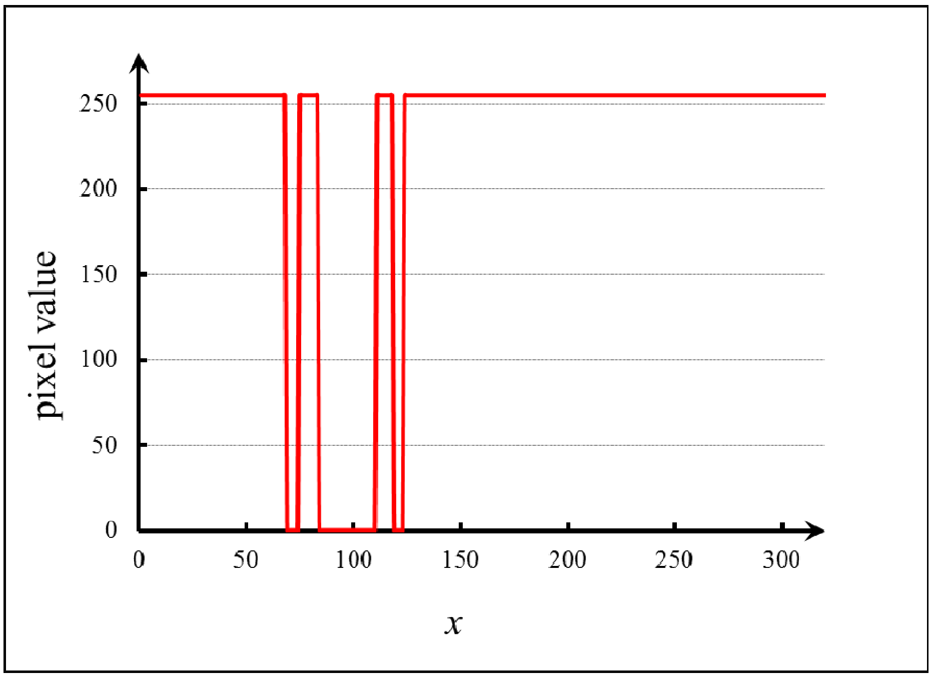


Fig. 2.7 Pixel values distribution of the 140th row in the road region image.

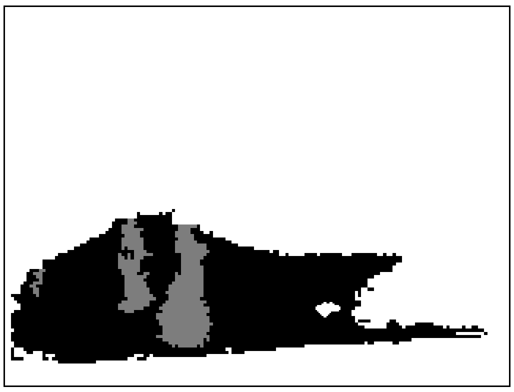


Fig. 2.8 The result of region classification.

2.4.3 Classification of Foreground Objects

In the foreground image, black pixels mean the pixels of foreground objects. These black pixels contain the pixels of obstacles, the pixels of road marks and the pixels of objects outside the road. In order to extract the shape of obstacles, we should change the black pixels which represent the noise to white. We check each black pixel's position in the result of region classification (shown in Fig. 2.8). If the current black pixel is located in the noise region (white region), this black pixel is considered as the object outside the road and is changed to white in the foreground image. If the current black pixel is located in the road region (black region), this black pixel is considered as the noise inside the road and is changed to white in the foreground image. If the current black pixel exists in the obstacle region, it is left unchanged. By this operation, noises are deleted. Then we carry out erosion operation and regional expansion as the post-processing.

2.5 Experimental Results

2.5.1 Experimental Environment

In order to obtain the videos for obstacles detection, we install a camera as shown in Fig. 2.9. The camera, which is fixed in front of the vice driver's seat, records the road conditions in front of the car when the car moves forward.

In this section, we examine the proposed obstacle detection method using three videos. Video 1 is captured in the artificial scene inside the campus; this scene includes two pedestrians and one box on the road. Two pedestrians cross the road in the opposite direction when the car is moving forward at normal speed; the box is located between pedestrians and a car. Video 2 is captured in the real scene: the car is moving forward to the crossroad, a pedestrian crosses the road before this car stops completely. Video 3 is captured in the artificial scene outside the campus; this scene includes two pedestrians and one box on the road. Two pedestrians cross the road in the opposite direction when the car is moving forward at normal speed; the box is located between pedestrians and a car.

In the proposed obstacles detection method, obstacles are defined as arbitrary objects that protrude from the ground plane in the road region, including static and moving objects. Road marks in the road region (e.g., zebra crossings) are considered as false obstacles and objects outside the road region is noise. According to this definition, correct objects in video 1 are two pedestrians and a box; a correct object in video 2 is a pedestrian; correct objects in video 3 are two pedestrians and a box.

The configurations of the PC used in the experiments are shown in Table 2.1.

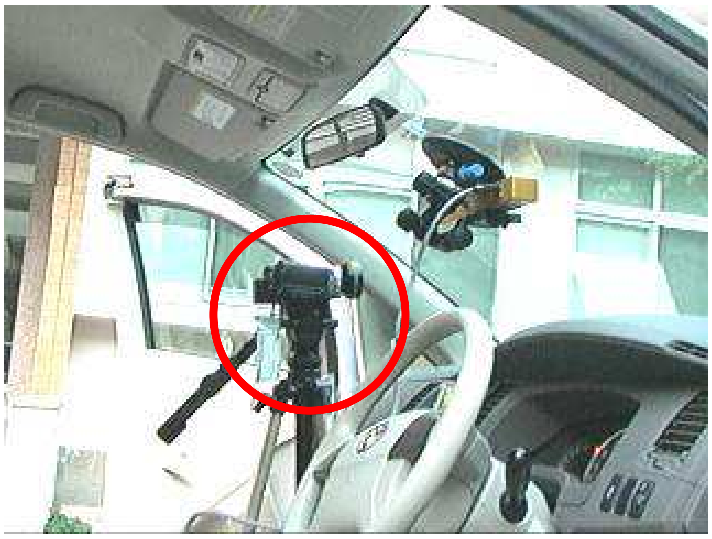


Fig. 2.9 The camera installation.

Table 2.1 Configurations of the PC used in the experiments.

OS	Microsoft Windows 7 Professional 64bit
CPU	Intel(R) Core(TM) i7-2600 3.40GHz
Memory	8.0 GB
Software Tool	Microsoft Visual Studio 2008

2.5.2 Detection of Foreground Images

In the first place, we reconstruct the background model using a Gaussian mixture model in the input images. The parameters of the Gaussian mixture model are shown in Table 2.2. After the background modeling, we obtain the background images and corresponding foreground images. Figure 2.10 – Fig. 2.12 shows the input images and the corresponding background and foreground images.

Because the camera is moving, the foreground images as shown in Fig. 2.10 – Fig. 2.12 contain a lot of noises. These noises are mostly caused by the objects outside the road region and shadows inside the road region. In order to delete these noises, we need to detect the road region.

Table 2.2 The parameters of the Gaussian mixture model.

Method	**Parameters**	**Values**
Gaussian mixture model	Learning rate α	0.05
	Threshold T	0.2
	Threshold T_{gauss}	1.0
	The number of distributions K	3

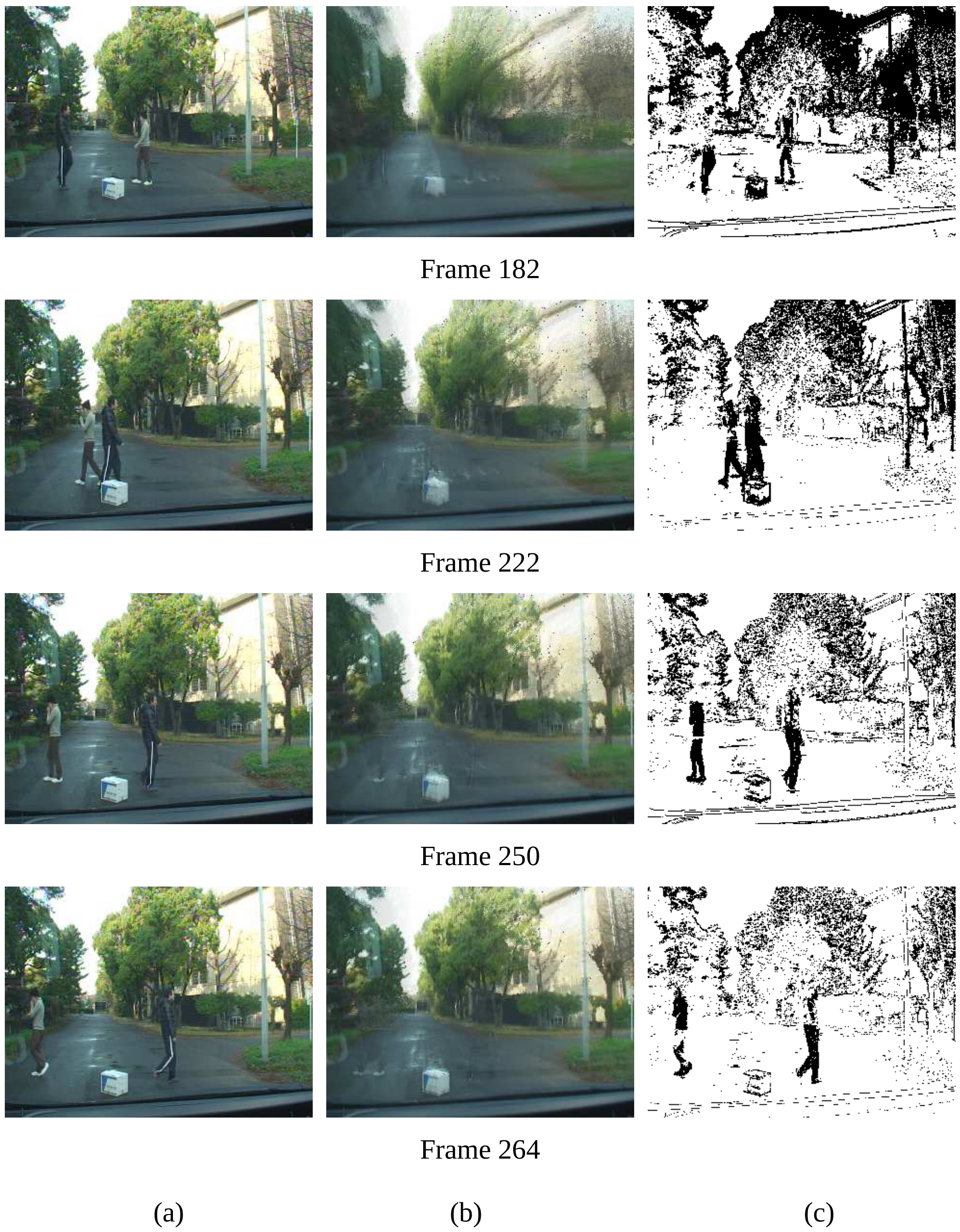
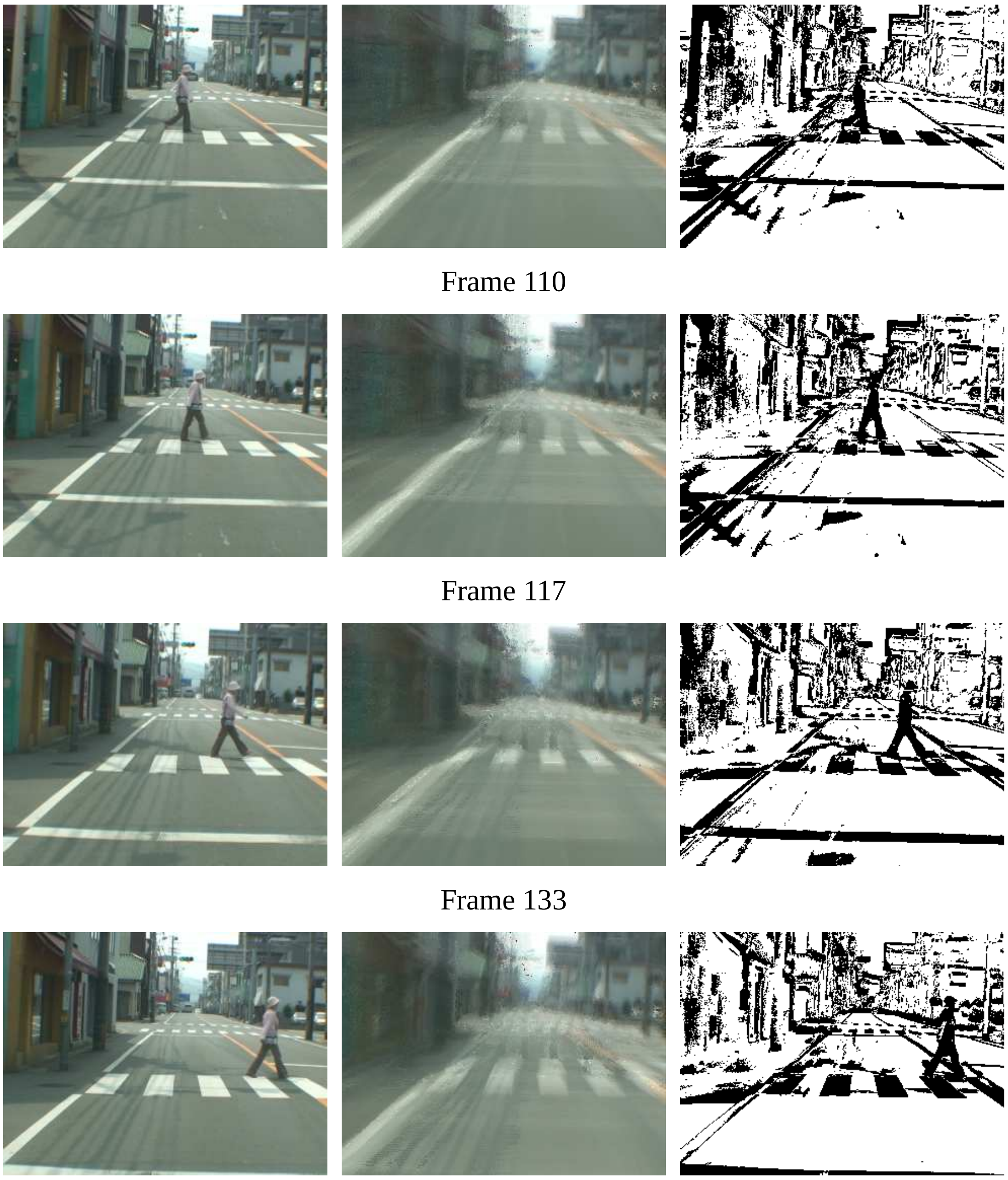


Fig. 2.10 The result of background modeling (Video 1). (a) Input images, (b) background images, (c) foreground images.



(a) (b) (c)

Fig. 2.11 The result of background modeling (Video 2). (a) Input images, (b) background images, (c) foreground images.

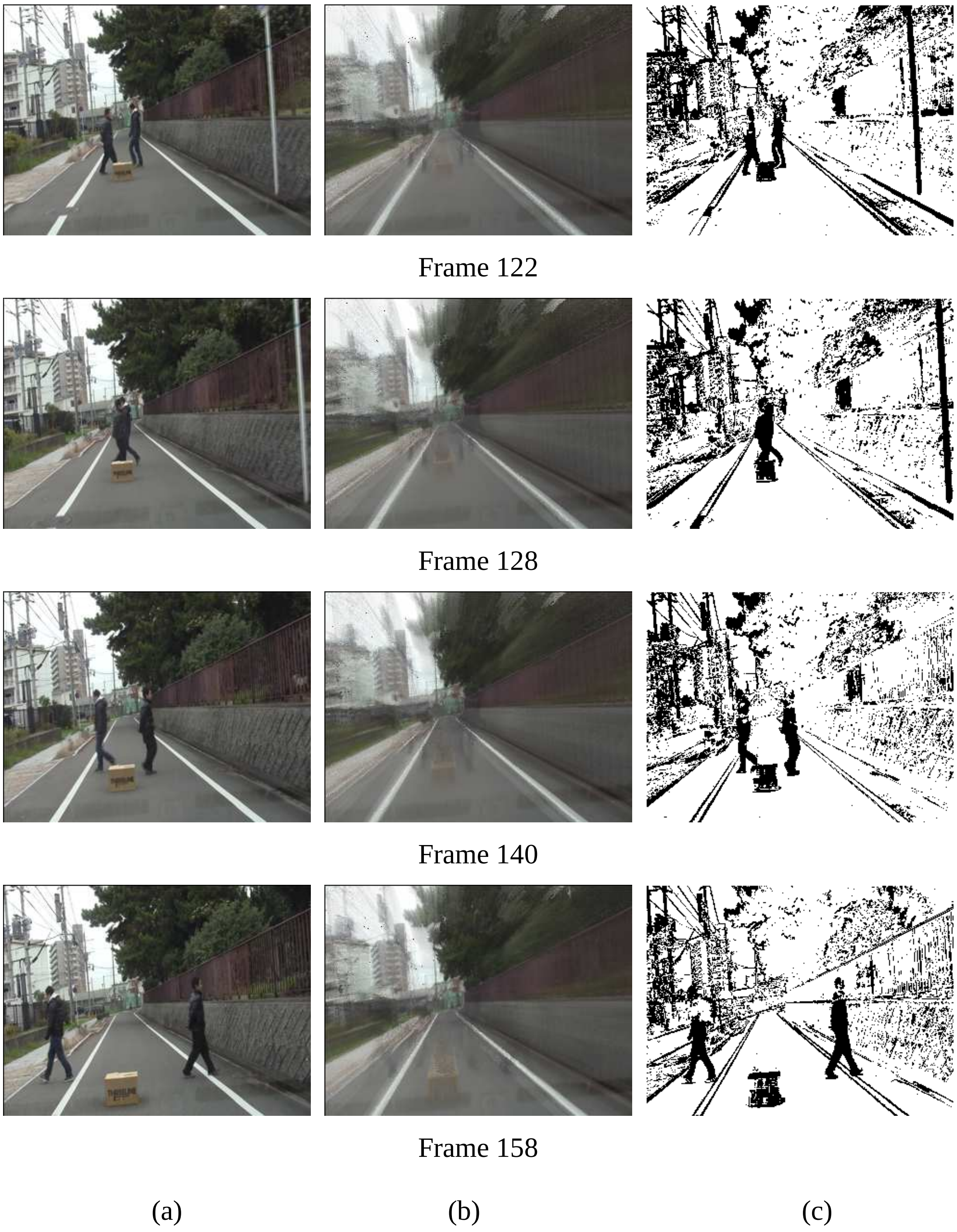


Fig. 2.12 The result of background modeling (Video 3). (a) Input images, (b) background images, (c) foreground images.

2.5.3 Detection of Road Region

In the second place, we detect the road region in the input images using the Support Vector Machine.

The result of road region detection depends on the selection of training data in the training step of road region detection (described in section 2.3.3).

Figure 2.13 shows two different experiments of road region detection.

In Fig. 2.13, (a) and (b) are one experiment; (a) shows the section of training data, (b) shows the corresponding result of road region detection. In Fig. 2.13 (a), the positive sample of the training data (pixels located in the green box) contains the pixels of road marks. In the corresponding result of road region detection (as shown in Fig. 2.13 (b)), the road marks are classified as a part of the road region (purple region means road region).

In Fig. 2.13, (c) and (d) are another experiment; (c) shows the section of training data, (d) shows the corresponding result of road region detection. In Fig. 2.13 (c), the positive sample of the training data (pixels located in the green box) does not contain the pixels of road marks. In the corresponding result of road region detection (as shown in Fig. 2.13 (d)), the road marks are classified as the non-road region (purple region means road region, and other regions mean non-road region).

Road marks such as zebra crossings on the road are not dangerous to driving. According to the definition of obstacles in this research, these road marks are considered as false obstacles. We should avoid detecting these false obstacles, because they will reduce the accuracy of detection. In order to delete road marks in the foreground images, it seems the result of road region detection shown in Fig. 2.13 (b) is much better.

Figure 2.14 – Fig. 2.16 show the results of road region detection. In this detection, the training data contain the pixels of road marks. In these resultant images, the purple color region means the road region.

2.5.4 Detection of Obstacles

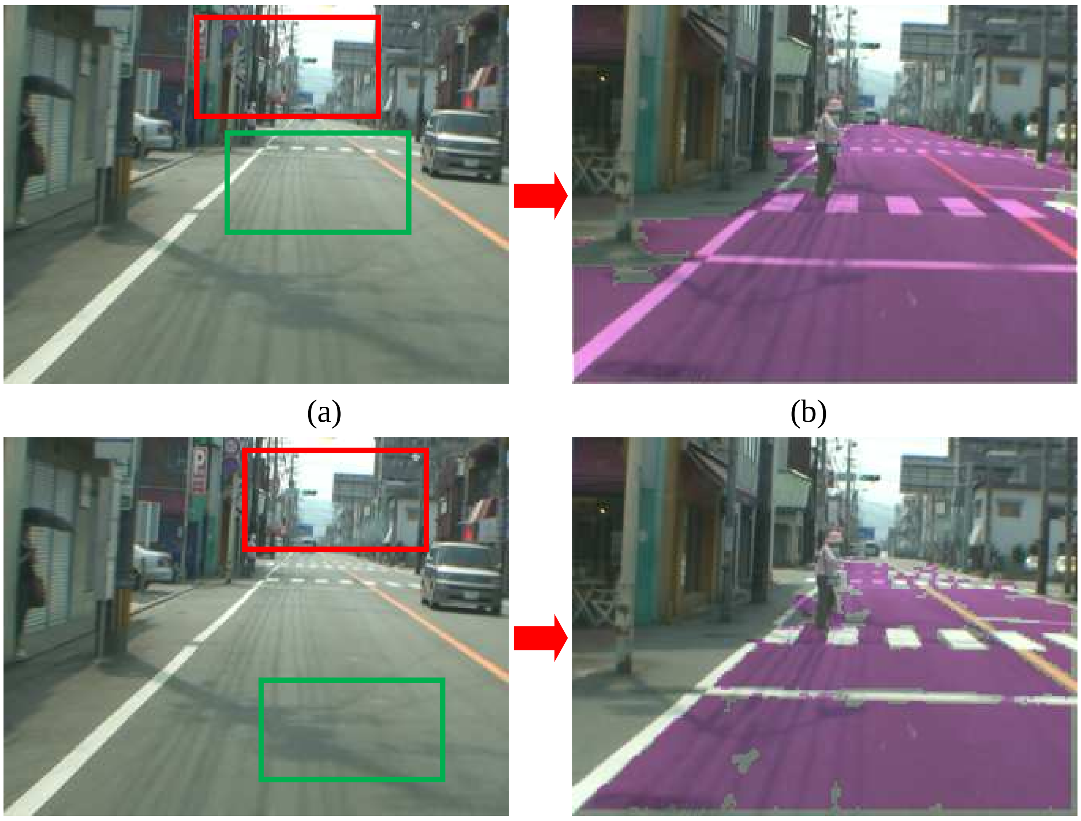
In the third place, we carry out region classification in the road region template images which have been obtained in Section 2.5.3 using the method explained in Section 2.4.2. The results of region classification are shown in Fig. 2.17 (b), Fig. 2.18 (b) and Fig. 2.19 (b). In these regional template images: black region means road region; white region means noise region; gray region means obstacles region.

Then, by deleting the noises in the foreground images (as shown in Fig. 2.10 (c), Fig. 2.11 (c) and Fig. 2.12 (c)) using the result of the region classification, we obtain the results of obstacle detection as shown in Fig. 2.17 (c), Fig. 2.18 (c) and Fig. 2.19 (c).

2.5.5 Comparative Experiment

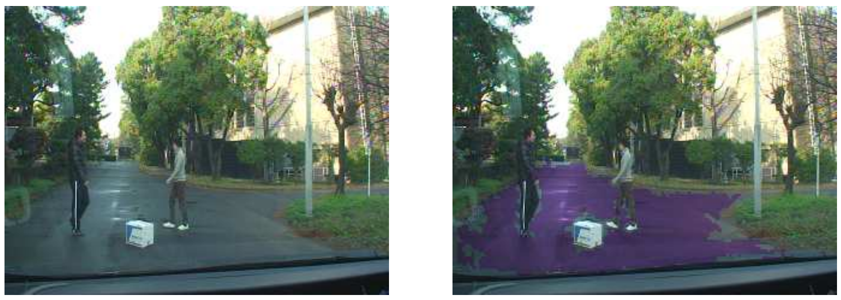
However, in a general case, the first frame of the detected video may not contain the road marks. So in the training step of the road region detection, the training data doesn't contain the pixels of road marks. We detect the road regions using this training data. Then we carry out the obstacles detection described in Section 2.5.4. This experiment is considered as the comparative experiment.

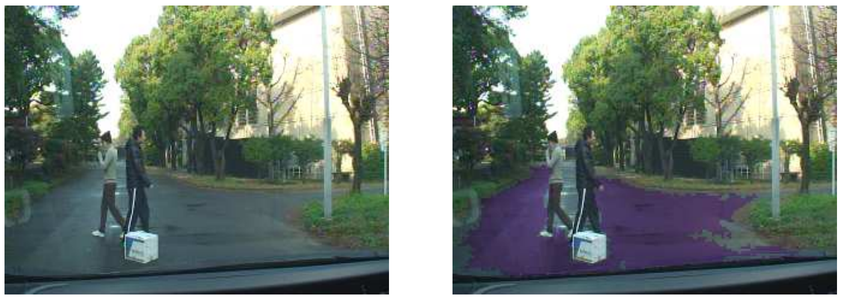
Figure 2.20 – Fig. 2.21 show the results of the comparative experiment.

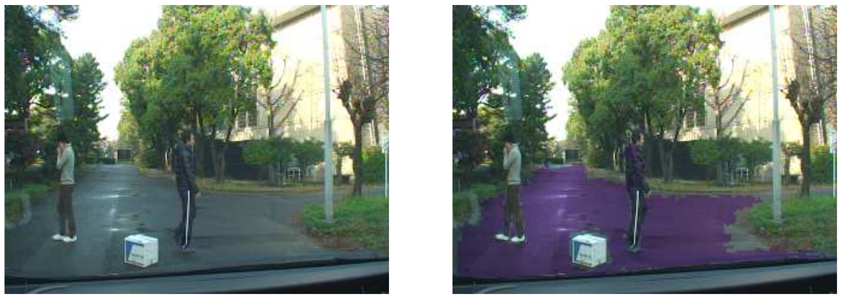


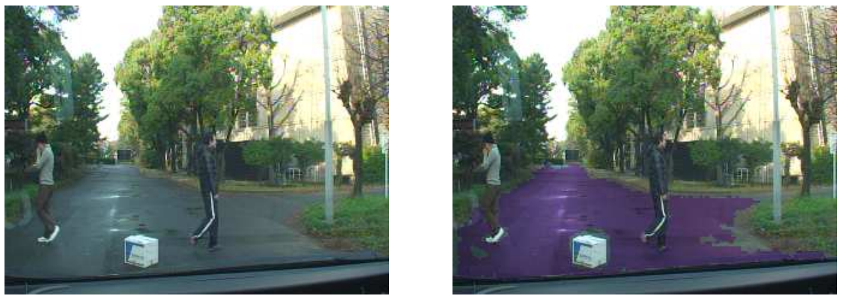
(a) (b) (c) (d)

Fig. 2.13 The experiments of road region detection. (a) Training data selection, (b) the result of road region detection, (c) training data selection, (d) the result of road region detection.


Frame 182


Frame 222


Frame 250


Frame 264

(a) (b)

Fig. 2.14 The results of road region detection (Video 1). (a) Input images, (b) the results of road region detection.

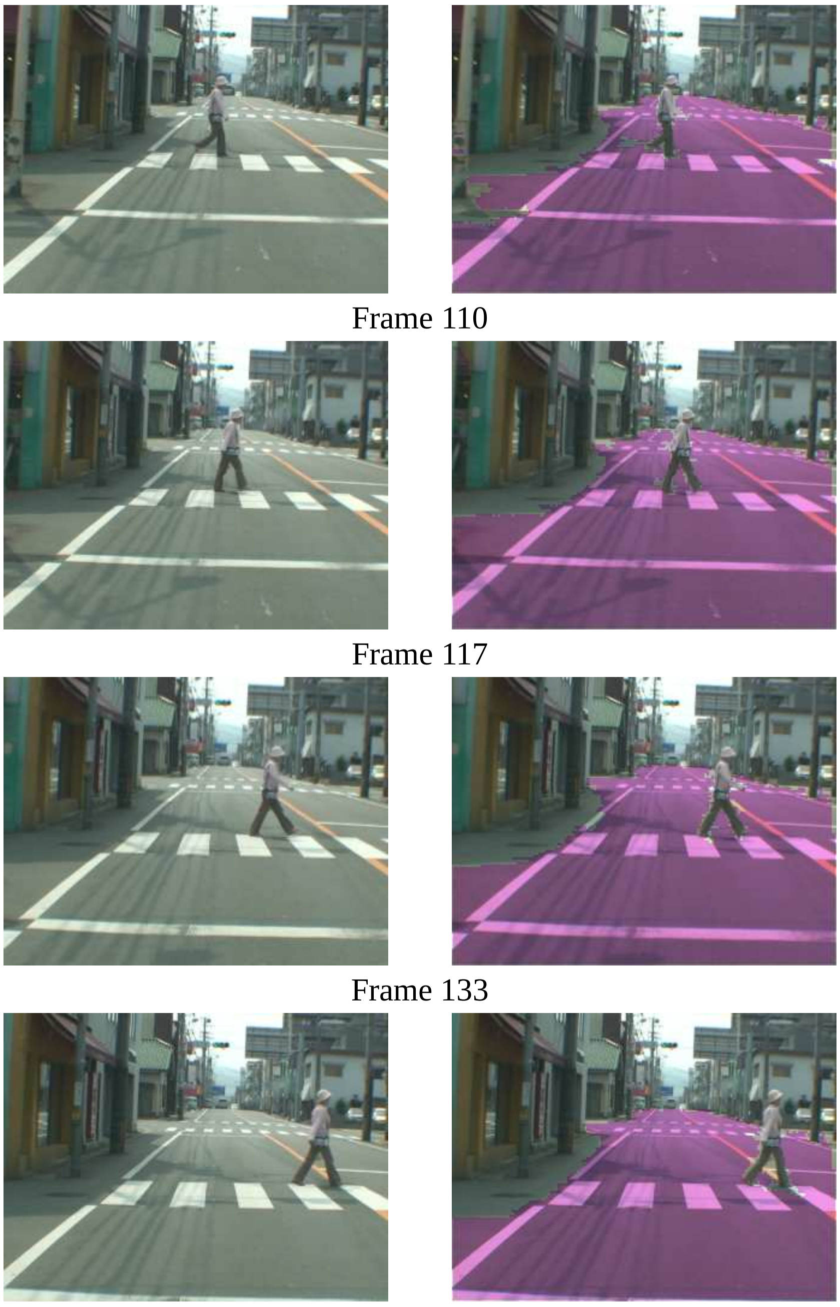


Fig. 2.15 The results of road region detection (Video 2). (a) Input images, (b) the results of road region detection.

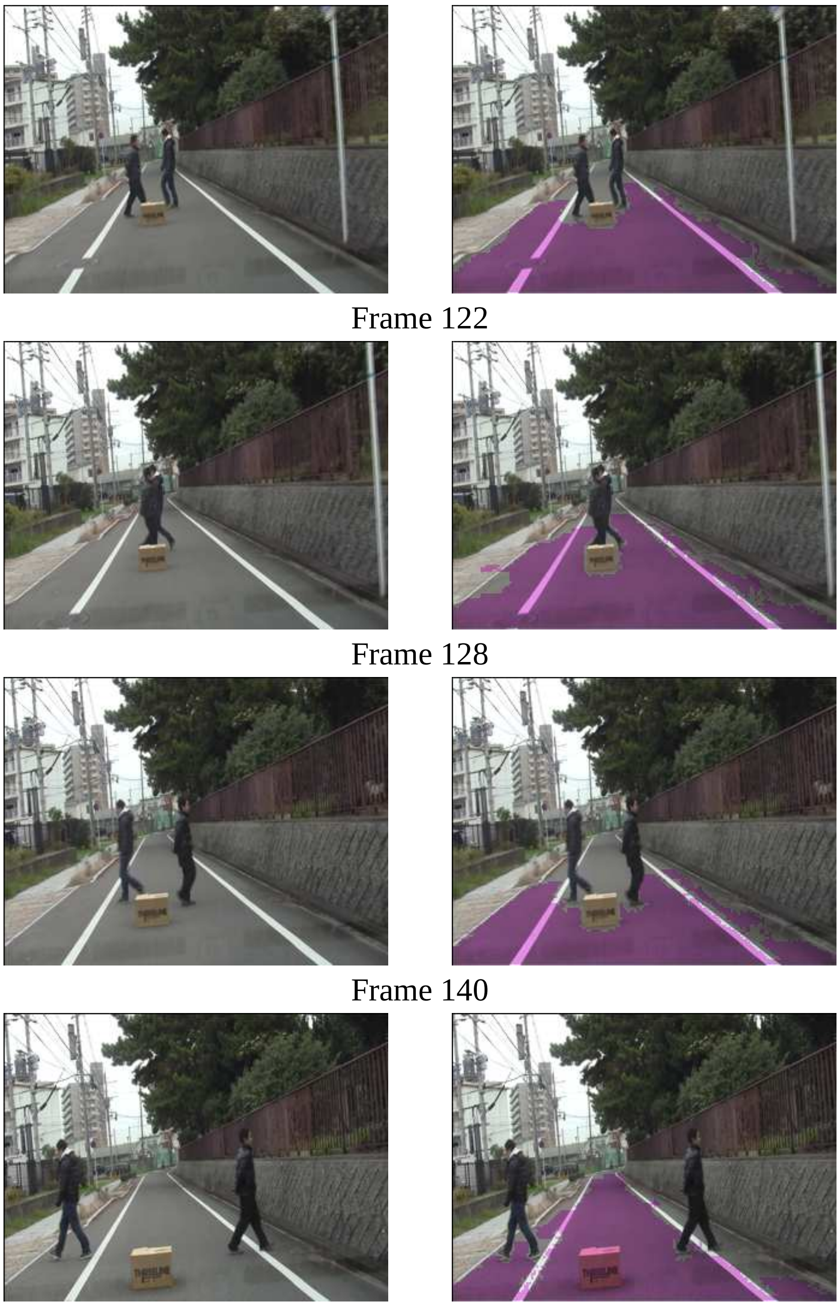
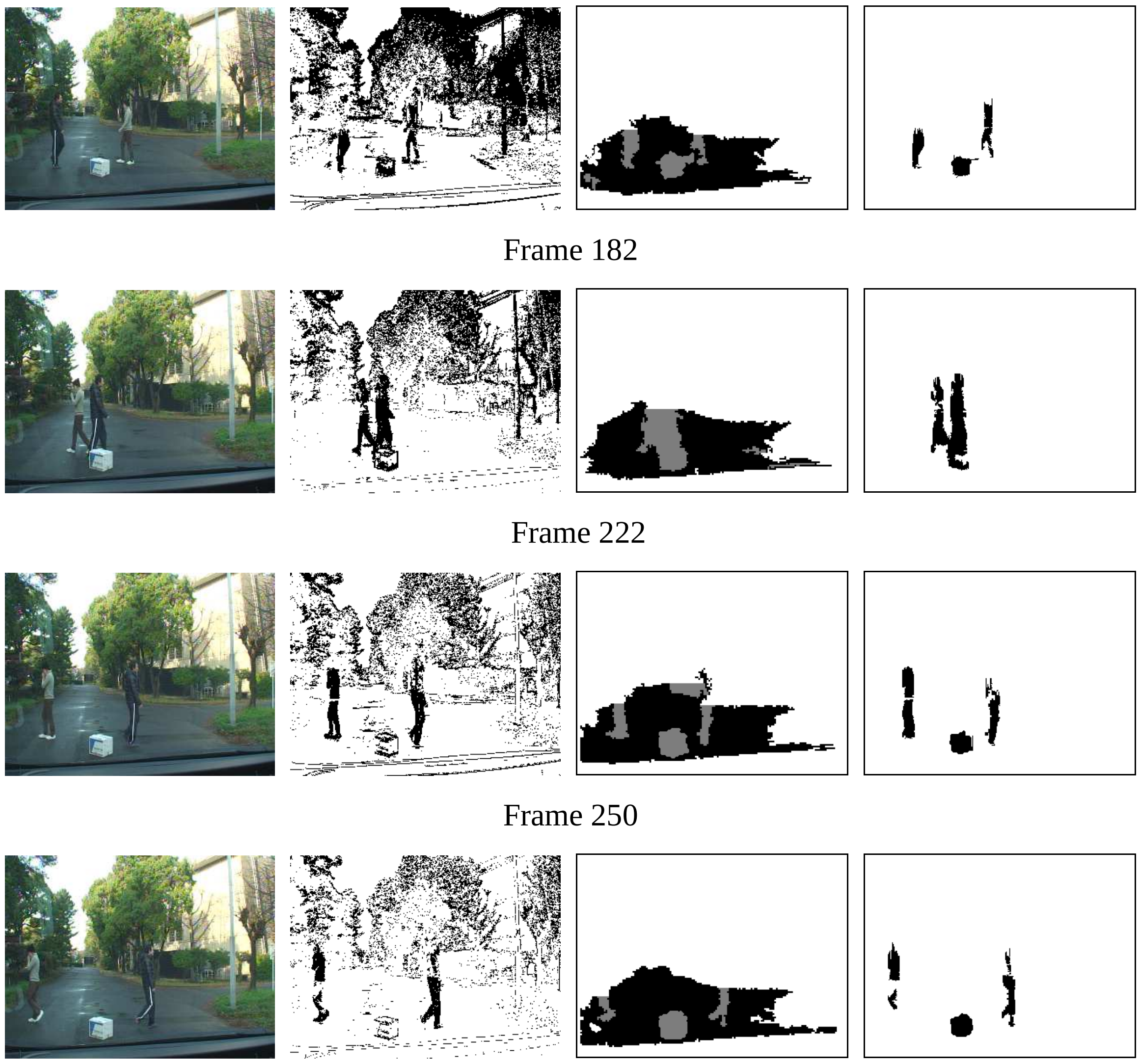


Fig. 2.16 The results of road region detection (Video 3). (a) Input images, (b) the results of road region detection.



(a) (b) (c) (d)

Fig. 2.17 Obstacle Extracting (Video 1). (a) Input images, (b) foreground images, (c) regional template images, (d) the result of obstacle detection.

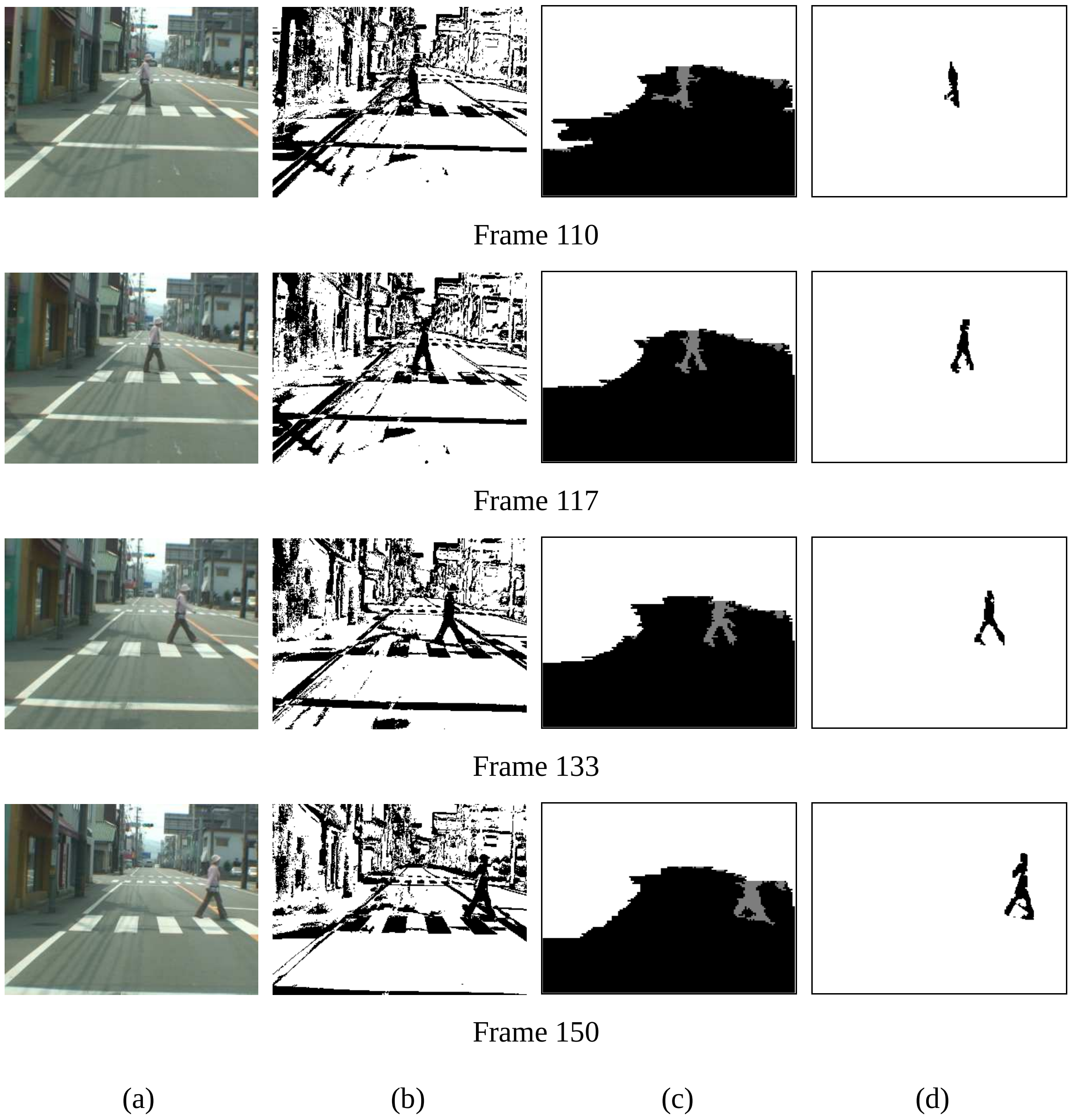
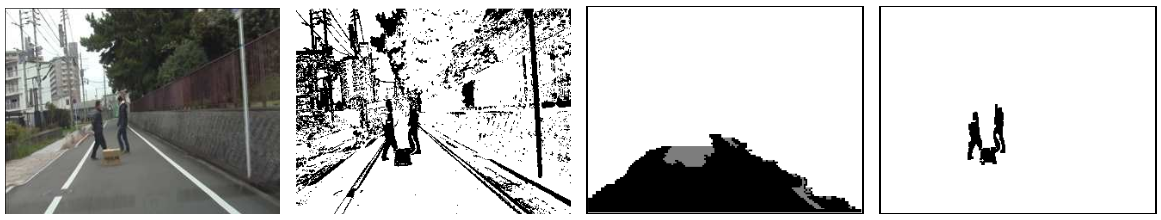
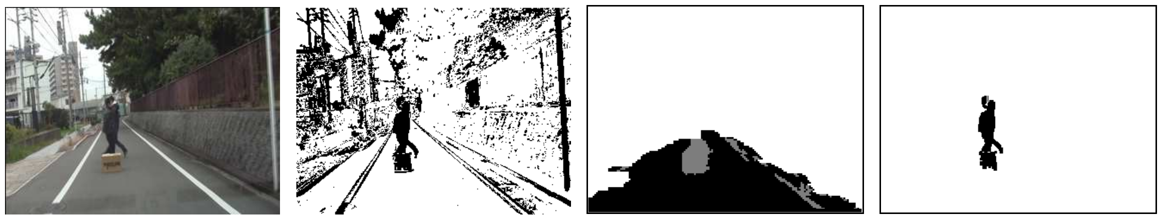
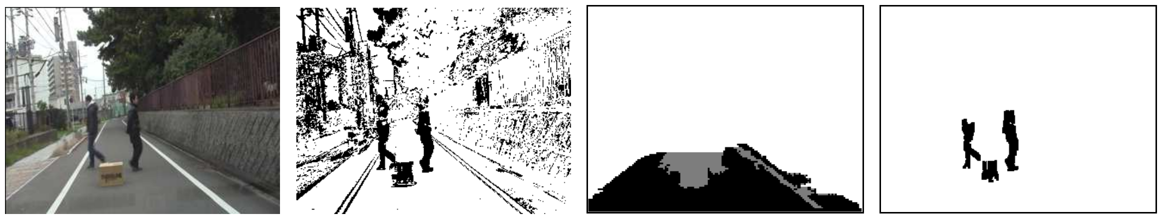
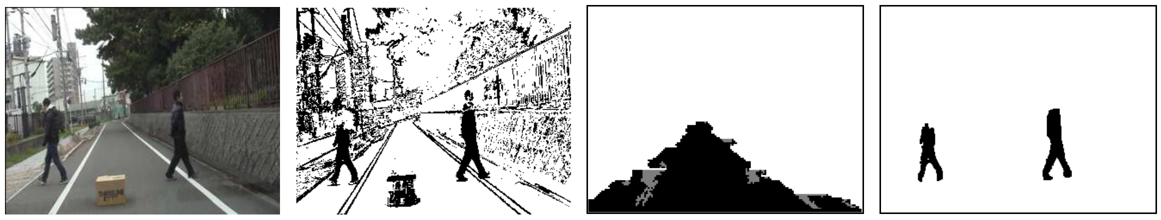


Fig. 2.18 Obstacle Extracting (Video 2). (a) Input images, (b) foreground images, (c) regional template images, (d) the result of obstacle detection.


Frame 122


Frame 128


Frame 140


Frame 158

(a) (b) (c) (d)

Fig. 2.19 Obstacle Extracting (Video 3). (a) Input images, (b) foreground images, (c) regional template images, (d) the result of obstacle detection.

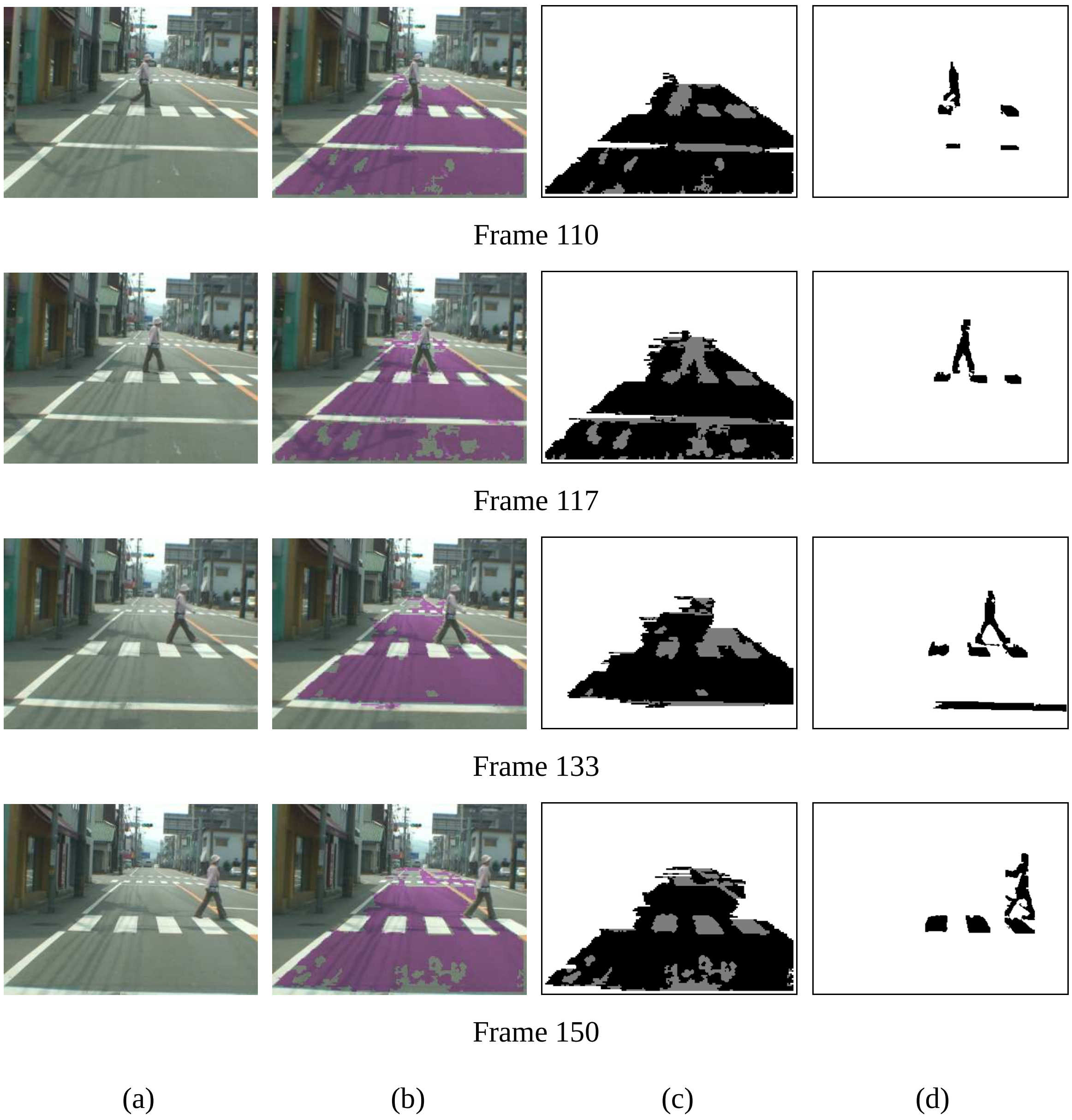


Fig. 2.20 The results of comparative experiment (Video 2). (a) Input images, (b) the results of road region detection. (c) regional template images, (d) the result of obstacle detection.

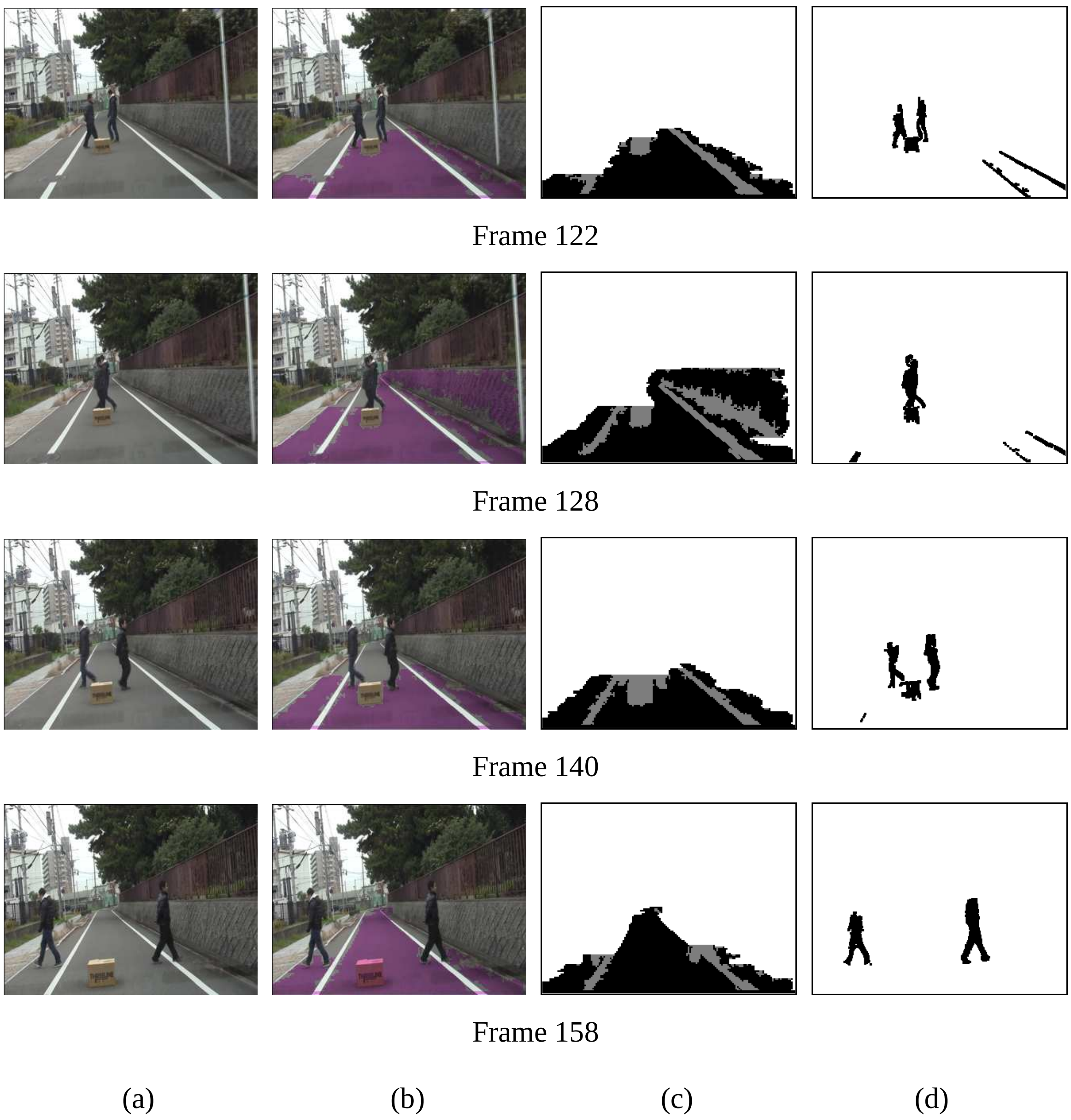


Fig. 2.21 The results of comparative experiment (Video 3). (a) Input images, (b) the results of road region detection. (c) regional template images, (d) the result of obstacle detection.

2.6 Evaluation

In order to evaluate the effectiveness of the proposed obstacles detection method, we compare the result of obstacles detection with the *Ground Truth* (shown in Fig. 2.23 (a) and (b)). In the resultant image of comparison (shown in Fig. 2.23 (c)), the red area means the overlap part of (a) and (b), and this part is called *True Positive*; Blue means the part which is included in (b) but not in (a), and this part is called *False Positive*; Green means the part which is included in (a) but not in (b), and this part is called *False Negative*. We calculate *recall* using the following formula:

$$recall = \frac{TP}{GT} \times 100[\%] \tag{2.21}$$

Here *TP* is the number of pixels in the *True Positive* area; *GT* is the number of black pixels in the *Ground Truth* image.

If *recall* is larger than 0.5, we consider this object has been extracted. Then we calculate *Recall* and *Precision* using the following formulas:

$$Recall = \frac{N_{TP}}{N_{GT}} \times 100\ [\%] \tag{2.22}$$

$$Precision = \frac{N_{TP}}{N_{TP} + N_{FP}} \times 100\ [\%] \tag{2.23}$$

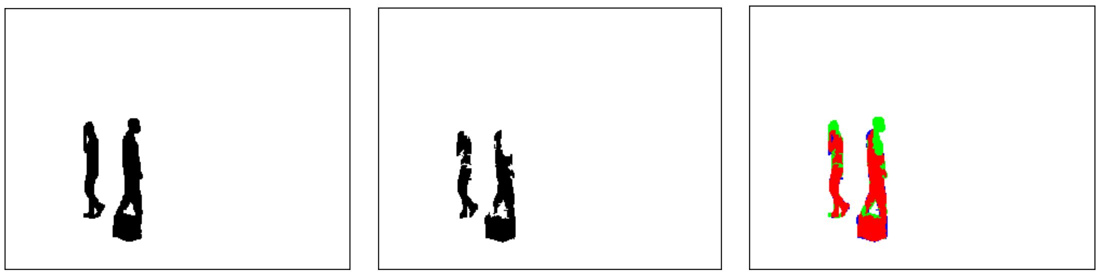
where N_{TP} is the number of correct objects in the resultant images; N_{GT} is the number of objects in the ground truth images; N_{FP} is the number of incorrect objects in the resultant images.

Precision can be seen as a measure of exactness or fidelity, whereas *Recall* is a measure of completeness. We also define *FPR* as a measure of inaccuracy; the formula is given as follows:

$$FPR = 100 - Precision = \frac{N_{FP}}{N_{TP} + N_{FP}} \times 100\ [\%] \tag{2.24}$$

The result of evaluation is composed of three values: *Precision*, *Recall* and *FPR*. The result of evaluation with Video 1, Video 2 and Video 3 using the proposed obstacle detection method (the results of obstacles detection are shown in Fig. 2.17 – Fig. 2.19) are shown in Table 2.3.

Table 2.4 – Table 2.5 show the results of evaluation with Video 2 and Video 3 using two different detection methods (the proposed obstacle detection method and the method explained in comparative experiment).



(a) Ground Truth (b) Obstacles (c) Resultant Image

Fig. 2.23 Images employed for evaluation.

Table 2.3 The result of evaluation of three different videos using the proposed obstacle detection method.

Videos	**Evaluation Values**		
	***Precision*[%]**	***Recall*[%]**	***FPR*[%]**
Video 1	97.5	79.7	2.5
Video 2	94.5	80.0	5.5
Video 3	96.8	73.9	3.2

Table 2.4 The result of evaluation of Video 2 using two different detection methods.

Methods	**Evaluation Values**		
	***Precision*[%]**	***Recall*[%]**	***FPR*[%]**
The Proposed Method	94.5	80.0	5.5
Comparative Experiment	47.7	78.2	52.3

Table 2.5 The result of evaluation of Video 3 using two different detection methods.

Method	**Evaluation Values**		
	***Precision*[%]**	***Recall*[%]**	***FPR*[%]**
The Proposed Method	96.8	73.9	3.2
Comparative Experiment	80.7	72.5	19.3

2.7 Discussion and Conclusion

In this chapter, we proposed an obstacles detection method using a video taken by a vehicle-mounted monocular camera.

In the part of background modeling, we applied GMM to a moving camera scene. The GMM is an effective background modeling method normally used in a static camera case. But we expanded it so that it can be applied to a moving camera case. This expansion is important to industrial applications of an obstacle detection system based on a vehicle-mounted camera.

In the road region detection method using the SVM, we extracted a feature vector for each pixel in the input image. This feature vector is combined by five texture features and three color features. According to experiments, this composite feature vector contributed sufficiently to detecting a road region from a video. It is noted that the composite feature vector was better than the feature vector which only uses texture features or only uses color features.

In Fig. 2.19, the input image of frame 158 includes three obstacles: two pedestrians and a box, but the resultant image of obstacles detection doesn’t include the box. At frame 158, since the car is close to the box, its reflection may have given influence to the gray value change with the box in the image and may have been judged as part of a road. That’s why the result of obstacles detection of frame 158 in Fig. 2.19 doesn’t include the box.

The same reason is applied to the result of obstacles detection of frame 158 in Fig. 2.21 which doesn’t include the box.

The proposed method has some advantages over the existing obstacles detection methods. In the first place, the proposed method uses a monocular camera. This realizes an economic system and smaller computation time. It is also advantageous for achieving real-time processing. In the second place, the proposed method can detect arbitrary objects including both static objects and moving objects. To the best of our knowledge, no researches have ever proposed a method which detects both static and moving objects simultaneously. This is helpful because static objects such as boxes fallen on the road from a car are dangerous for drivers. Most of the existent methods concentrate only on detecting moving objects such as pedestrians, bicycles and cars. In the third place, the output of the proposed method is the shape of obstacles. Most of the existing obstacles detection methods only indicate the location of an obstacle by a rectangular frame which surrounds it and they do not extract the shape of obstacles. Extraction of the shape of an obstacle is important for obstacles recognition. If the detected obstacle is recognized as a pedestrian from its shape, we can foresee his/her next action.

In the results of comparative experiment (shown in Fig. 2.20 – Fig. 2.21), road marks such as zebra crossings are detected as obstacles. Since these 2D objects are not dangerous to driving, they will reduce the accuracy of detection if they are detected as obstacles. From Table 2.4 and Table 2.5, we see the *Precision* of comparative experiment is much lower than the proposed method. These 2D objects are considered as *false obstacles*. We should avoid detecting these false obstacles. In order not to detect these 2D objects, we need to develop a method of classifying 2D objects and 3D objects. Then we can use the result of the classification to delete these false obstacles. In the next chapter, we will propose a method for classifying 2D objects and 3D objects in the resultant images of obstacles detection.

Chapter 3 2D and 3D Objects Classification

In Chapter 2, we have proposed an obstacles detection method based on background modeling. But this method detects 2D and 3D objects simultaneously. Since these 2D objects are not dangerous to driving, they will reduce the accuracy of detection if they are detected as obstacles. In order not to detect these 2D objects, in this chapter, we propose a method for classifying 2D objects and 3D objects in the resultant images of obstacles detection which have been obtained in Chapter 2.

3.1 Outline of the Proposed Method

The proposed 2D and 3D objects classification method consists of four steps; camera motion estimation, 3D coordinate estimation, road plane estimation, and 2D and 3D classification. Figure 3.1 shows the flowchart of the 2D and 3D objects classification.

The proposed method first estimates camera motion parameters from the correspondences of feature points between two successive images (Section 3.2). Then we calculate the 3D positions of the feature points on a detected object in the world coordinate system using triangulation (Section 3.3). Then we estimate the parameters of the road plane using 3D positions of those feature points (Section 3.4). Finally we calculate the distances from the 3D positions of the feature points to the road plane (Section 3.5). Based on these distances, we classify 2D objects and 3D objects.

3.2 Camera Motion Estimation

This section estimates the camera motion parameters from the correspondences of feature points between two successive images. The camera motion parameters consist of a 3×3 rotation matrix R and a 3×1 translation vector T [39]. In this method, two successive images are used at any time, i.e., the image I_t taken at time *t* and the image I_{t+1} taken at time t+1 are used to calculate camera motion parameters from time *t* to time t+1.

3.2.1 Feature Points Detection

In this section, we want to detect feature points in the first image I_t taken at time *t*.

A corner is a point whose brightness changes dramatically in the image or which has the maximum curvature in the edge curve of an image. One of the most known corner detection methods is the Harris corner detector [40].

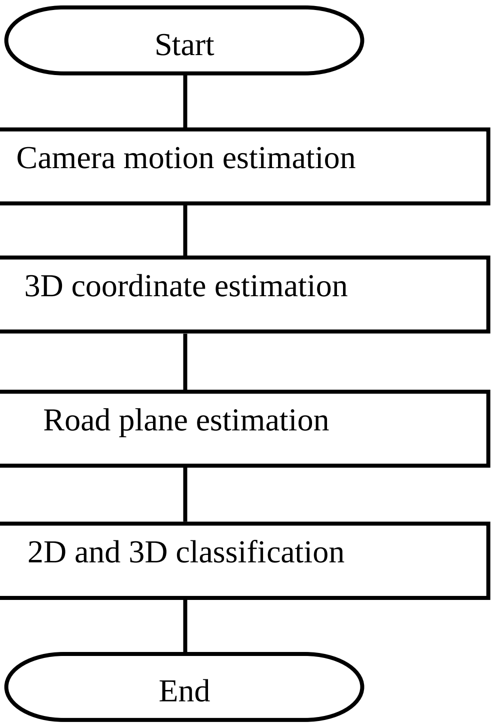


Fig. 3.1 Flowchart of the 2D and 3D objects classification.

The Harris corner detector is based on the local auto-correlation function of a signal. The local auto-correlation function is the change of intensity for a shift (x, y), it is defined as:

$$E(x,y) = \sum_{u,v} w(u,v)\left[I(u+x, v+y) - I(u,v)\right]^2 \tag{3.1}$$

Here, $w(u,v)$ is a Gaussian function window which is centered at (u,v); $I(u,v)$ is the intensity at (u,v); $I(u+x,v+y)$ is the intensity at the moved position $(u+x,v+y)$.

The intensity at the moved position, $I(u+x,v+y)$, can be approximated by a Taylor expansion:

$$I(u+x,v+y) \approx I(u,v) + \begin{bmatrix} I_x & I_y \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} \tag{3.2}$$

Here I_x and I_y are the first-order partial derivatives in x and y directions.

Substituting Eq.(3.2) into Eq.(3.1), we have

$$\begin{aligned}
E(x,y) &= \sum_{u,v} w(u,v)[I(u+x,v+y)-I(u,v)]^2 \\
&\approx \sum_{u,v} w(u,v)\left(I(u,v)+[I_x \quad I_y]\begin{bmatrix} x \\ y \end{bmatrix} - I(u,v) \right)^2 \\
&= \sum_{u,v} w(u,v)\left([I_x \quad I_y]\begin{bmatrix} x \\ y \end{bmatrix} \right)^2 \\
&= [x \quad y]\sum_{u,v} w(u,v)\begin{bmatrix} I_x{}^2 & I_xI_y \\ I_xI_y & I_y{}^2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}
\end{aligned} \tag{3.3}$$

Let us denote

$$M = \sum_{u,v} w(u,v)\begin{bmatrix} I_x{}^2 & I_xI_y \\ I_xI_y & I_y{}^2 \end{bmatrix} \tag{3.4}$$

Then the change of intensity for a shift (x, y) (Eq. (3.1)) can be concisely written as

$$E(x,y) \approx [x \quad y]M\begin{bmatrix} x \\ y \end{bmatrix} \tag{3.5}$$

Note that *E* is closely related to the local auto-correlation function, *M* describing the shape of this auto-correlation function at the origin point.

Let λ_1, λ_2 be the eigenvalues of matrix *M*. λ_1, λ_2 are proportional to the principal curvatures of the local auto-correlation function, and they form a rotationally invariant description. We can decide the flat region, corner and edge through checking the values of λ_1, λ_2. There are three cases to be considered (shown in Fig. 3.2):

1) If both λ_1, λ_2 are small, so that the local auto-correlation function is flat(little change in $E(x, y)$ in any directions), the small window has approximately constant intensities.
2) If one eigenvalue is high and the other is low, so the local auto-correlation function is ridge shaped, local shifts along the ridge cause little change in $E(x, y)$ and local shifts

in the orthogonal direction cause significant change; this means an edge.

3) If both eigenvalues are high, so the local auto-correlation function is peak shaped, local shifts in any directions will cause a dramatic change; this means a corner.

3.2.2 Feature Points Tracking

In this section, we want to detect the corresponding feature points $\boldsymbol{m}_{t+1}$ in the second image I_{t+1} using the above detected Harris feature points $\boldsymbol{m}_t$ in the first image I_t. We can use feature tracking methods to find the location of points on the second image. One of the most known methods is the Lucas-Kanade feature tracking algorithm [41].

Let us consider two successive images (gray images), I_t and I_{t+1}, and a feature point $\boldsymbol{u} = [u_x \quad u_y]^T$ in the first image I_t. The goal of feature tracking is to find the location $\boldsymbol{v}$=$\boldsymbol{u}$+$\boldsymbol{d}$$= [u_x + d_x \quad u_y + d_y]^T$ in the second image I_{t+1}, which should satisfy that $I_t(\boldsymbol{u})$ and $I_{t+1}(\boldsymbol{v})$ are "similar". Here, the vector $\boldsymbol{d} = [d_x \quad d_y]^T$ is the optical flow at the point $\boldsymbol{u}$. The similarity between $I_t(\boldsymbol{u})$ and $I_{t+1}(\boldsymbol{v})$ is analyzed in a small window. Two integers w_x and w_y are defined as the size of the window. We define the optical flow $\boldsymbol{d}$ as being the vector that minimizes the residual function ε as follows:

$$\begin{aligned}\varepsilon(d) &= \varepsilon(d_x, d_y) \\ &= \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I_t(x,y) - I_{t+1}(x+d_x, y+d_y))^2 \end{aligned} \tag{3.6}$$

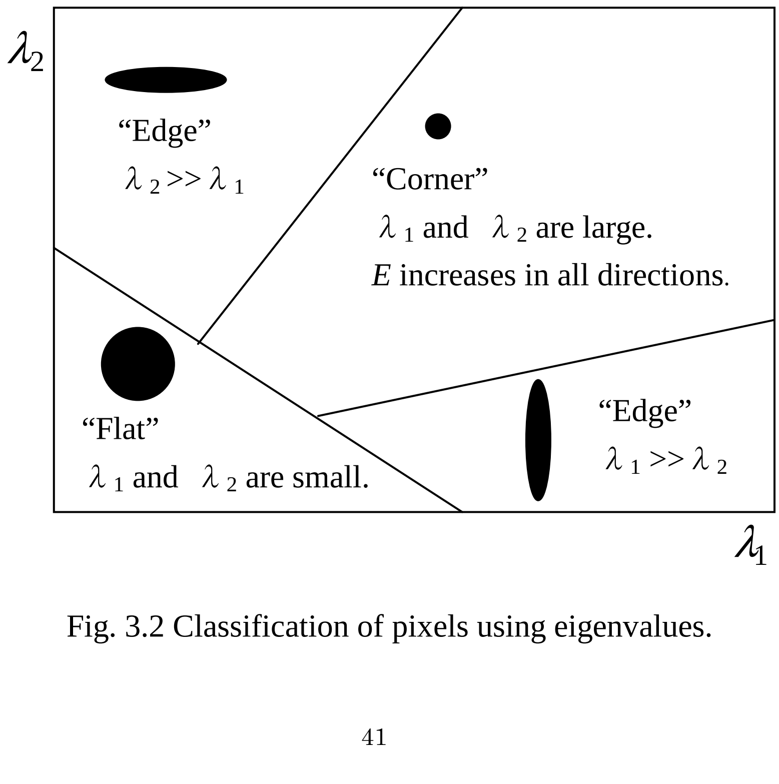

λ_2
"Edge"
$\lambda_2 >> \lambda_1$
"Corner"
λ_1 and λ_2 are large.
E increases in all directions.
"Flat"
λ_1 and λ_2 are small.
"Edge"
$\lambda_1 >> \lambda_2$
λ_1

Fig. 3.2 Classification of pixels using eigenvalues.

The residual function is measured in a small window of size $(2w_x+1)\times(2w_y+1)$. This window often has the size between 2 and 7. In order to choose a good window size, two important aspects, accuracy and robustness, should be considered. If we choose a small size, it will increase the accuracy, because it carries out the calculation only using the nearest neighbors. But it will decrease the robustness, since large vectors are ignored. If we choose a large size, it will decrease the accuracy and increase the robustness. In an idealistic situation, we should have $d_x \le w_x$ and $d_y \le w_y$ to cover all the possible motions.

According to Bouguet [42], a solution to this problem is a pyramidal implementation of the classical Lucas-Kanade algorithm. An iterative implementation of the Lucas-Kanade optical flow computation provides sufficient local tracking accuracy.

First let us see the classical Lucas-Kanade algorithm, and then the pyramidal implementation of the classical Lucas-Kanade algorithm is explained.

The Classical Lucas-Kanade Algorithm

For clarity purposes, let us change the names, $A(x,y)$ is the first image; $B(x,y)$ is the second image; $\bar{\boldsymbol{v}}=\begin{bmatrix}v_x & v_y\end{bmatrix}^T$ is the displacement vector; $\boldsymbol{p}=\begin{bmatrix}p_x & p_y\end{bmatrix}^T$ is the image position vector. Following the new notation, the goal of the classical Lucas-Kanade algorithm is to find the vector $\bar{\boldsymbol{v}}=\begin{bmatrix}v_x & v_y\end{bmatrix}^T$ that minimizes the residual function

$$\begin{aligned}\varepsilon(\bar{v})&=\varepsilon(v_x,v_y)\\ &=\sum_{x=p_x-w_x}^{p_x+w_x}\sum_{y=p_y-w_y}^{p_y+w_y}\left(A(x,y)-B(x+v_x,y+v_y)\right)^2\end{aligned} \tag{3.7}$$

At an optimum point, the first derivate of this function with respect to $\bar{\boldsymbol{v}}$ is zero:

$$\left.\frac{\partial\varepsilon(\bar{v})}{\partial\bar{v}}\right|_{\bar{v}=\bar{v}_{opt}}=[0 \quad 0] \tag{3.8}$$

After expanding the derivative and replacing $B(x+v_x,y+v_y)$ by its first order Taylor expansion, we obtain:

$$\frac{\partial\varepsilon(\bar{v})}{\partial\bar{v}}\approx -2\sum_{x=p_x-w_x}^{p_x+w_x}\sum_{y=p_y-w_y}^{p_y+w_y}\left(A(x,y)-B(x,y)-\begin{bmatrix}\frac{\partial B}{\partial x} & \frac{\partial B}{\partial y}\end{bmatrix}\bar{v}\right)\cdot\begin{bmatrix}\frac{\partial B}{\partial x} & \frac{\partial B}{\partial y}\end{bmatrix} \tag{3.9}$$

In order to simplify Eq.(3.9), we can make some replacements as follows:

$$\delta I(x,y) \doteq A(x,y) - B(x,y) \tag{3.10}$$

$$\nabla I = \begin{bmatrix} I_x \\ I_y \end{bmatrix} \doteq \begin{bmatrix} \frac{\partial B}{\partial x} & \frac{\partial B}{\partial y} \end{bmatrix}^T \tag{3.11}$$

$$I_x(x,y) = \frac{\partial A(x,y)}{\partial x} = \frac{A(x+1,y) - A(x-1,y)}{2} \tag{3.12}$$

$$I_y(x,y) = \frac{\partial A(x,y)}{\partial y} = \frac{A(x,y+1) - A(x,y-1)}{2} \tag{3.13}$$

After doing these modifications, we can rewrite Eq. (3.9) as follows:

$$\frac{1}{2}\frac{\partial \varepsilon(\bar{v})}{\partial \bar{v}} \approx \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} (\nabla I^T \bar{v} - \delta I)\nabla I^T \tag{3.14}$$

$$\frac{1}{2}\left[\frac{\partial \varepsilon(\bar{v})}{\partial \bar{v}}\right]^T \approx \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \bar{v} - \begin{bmatrix} \delta I \cdot I_x \\ \delta I \cdot I_y \end{bmatrix} \right) \tag{3.15}$$

We can denote the matrix by ***G*** and vector $\bar{b}$ in the following way;

$$G \doteq \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{3.16}$$

$$\bar{b} \doteq \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \begin{bmatrix} \delta I \cdot I_x \\ \delta I \cdot I_y \end{bmatrix} \tag{3.17}$$

Therefore, Eq. (3.15) is written as

$$\frac{1}{2}\left[\frac{\partial \varepsilon(\bar{v})}{\partial \bar{v}}\right]^T \approx G\bar{v} - \bar{b} \tag{3.18}$$

Since the first derivative at an optimum point is zero (Eq. 3.8), in the end, the optimum optical flow is calculated by

$$\bar{v}_{opt} = G^{-1}\bar{b} \tag{3.19}$$

This is the classical Lucas-Kanade optical flow equation.

Pyramidal Implementation of the Classical Lucas-Kanade Algorithm

First, we establish an image pyramid.

Considering an image I of size $n_x \times n_y$, the pyramidal representation is constructed as follows:

Let $I^0 = I$ be the "zero^th" level image. This image is the original image. The image width and height at this level are defined as $n_x^0 = n_x$ and $n_y^0 = n_y$.

The pyramidal representation is then built in a recursive way: I^1 is computed based on I^0, I^2 is computed based on I^1, and so on. In a general view, I^L is computed based on I^{L-1}, where L is the pyramidal level. The image I^L is defined as follows:

$$\begin{aligned} I^L(x,y) = {} & \frac{1}{4} I^{L-1}(2x,2y) + \frac{1}{8}(I^{L-1}(2x-1,2y) + \\ & I^{L-1}(2x+1,2y) + I^{L-1}(2x,2y-1) + I^{L-1}(2x,2y+1)) \\ & + \frac{1}{16}(I^{L-1}(2x-1,2y-1) + I^{L-1}(2x+1,2y-1) + \\ & I^{L-1}(2x-1,2y+1) + I^{L-1}(2x+1,2y+1)) \end{aligned} \tag{3.20}$$

The corresponding coordinates of the point $\boldsymbol{u} = [u_x \quad u_y]^T$ in the pyramidal image I^L is defined as $\boldsymbol{u}^L = [u_x^L \quad u_y^L]^T$. The vector $\boldsymbol{u}^L$ is computed as follows:

$$\boldsymbol{u}^L = \boldsymbol{u}/2^L \tag{3.21}$$

The overall pyramidal tracking algorithm proceeds as follows:

1) The optical flow, $\boldsymbol{d}^{Lm} = [d_x^{L_m} \quad d_y^{L_m}]^T$, is computed at the deepest pyramid level L_m using the classical Lucas-Kanade algorithm. The initial guess $\boldsymbol{g}^{Lm}$ is $[0 \quad 0]^T$.
2) The result of that computation, $\boldsymbol{d}^{Lm} = [d_x^{L_m} \quad d_y^{L_m}]^T$, is propagated to the upper level L_{m-1}. The initial guess $\boldsymbol{g}^{Lm\text{-}1} = [g_x^{L_{m-1}} \quad g_y^{L_{m-1}}]^T$ is computed as follows: $\boldsymbol{g}^{Lm\text{-}1} = 2(\boldsymbol{g}^{Lm} +$

$\boldsymbol{d}^{Lm}$).

3) Given that initial guess, the refined optical flow, $\boldsymbol{d}^{Lm\text{-}1} = \begin{bmatrix} d_x^{L_{m-1}} & d_y^{L_{m-1}} \end{bmatrix}^T$, is computed at the pyramid level L_{m-1} using the classical Lucas-Kanade algorithm.
4) The same procedures are performed until we reach the level 0 (the original image).

Let us describe the recursive operation between two levels L+1 and L in more details.

We assume that an initial guess for optical flow computation at level L, $\boldsymbol{g}^L = \begin{bmatrix} g_x^L & g_y^L \end{bmatrix}^T$, is obtained from the computation done at level L+1. Then, in order to compute the optical flow at level L, it needs to find the residual pixel displacement vector $\boldsymbol{d}^L = \begin{bmatrix} d_x^L & d_y^L \end{bmatrix}^T$ that minimizes the new image residual function ε^L of the form

$$\varepsilon^L(d^L) = \varepsilon^L(d_x^L, d_y^L) = \sum_{x=u_x^L - w_x}^{u_x^L + w_x} \sum_{y=u_y^L - w_y}^{u_y^L + w_y} (I_t^L(x,y) - I_{t+1}^L(x + g_x^L + d_x^L, y + g_y^L + d_y^L))^2 \tag{3.22}$$

The initial guess flow vector $\boldsymbol{g}^L$ is used to pre-translate the image patch in the second image I_{t+1}. We use the classical Lucas-Kanade optical flow algorithm to compute the optical flow $\boldsymbol{d}^L$. Then, the result of this computation is propagated to the next level L-1 and gets the new initial guess $\boldsymbol{g}^{L\text{-}1}$:

$$\boldsymbol{g}^{L\text{-}1} = 2(\boldsymbol{g}^L + \boldsymbol{d}^L) \tag{3.23}$$

The next level optical flow residual vector $\boldsymbol{d}^{L\text{-}1}$ is then computed using the same procedure. The algorithm finishes when the level 0 (the highest pyramidal level) is reached.

The final optical flow solution $\boldsymbol{d}$ is obtained by

$$\boldsymbol{d} = \boldsymbol{g}^0 + \boldsymbol{d}^0 \tag{3.24}$$

3.2.3 RANSAC

After feature points detection and feature points tracking, we get a set of corresponding feature points $\{\boldsymbol{m}_t \leftrightarrow \boldsymbol{m}_{t+1}\}$. $\boldsymbol{m}_t = \begin{bmatrix} x_t & y_t \end{bmatrix}^T$ is the point in the first image, $\boldsymbol{m}_{t+1} = \begin{bmatrix} x_{t+1} & y_{t+1} \end{bmatrix}^T$ is the corresponding point in the second image. But in the practical situation, $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t+1}$ might be mismatched. We often call these mismatched points as *outliers* and matched points as *inliers*. These outliers will severely reduce the detection accuracy. So we must take some measures to delete these outliers. One of the most known methods for this purpose is the RANdom SAmple Consensus algorithm (RANSAC) [43].

RANSAC Algorithm [43-44]:

The data set $\boldsymbol{S}$ is a set of feature point pairs $\{\boldsymbol{m}_t \leftrightarrow \boldsymbol{m}_{t+1}\}$; model is a fundamental matrix F.

1) Randomly select a sample of 8 pairs of feature points from $\boldsymbol{S}$ using 8-points algorithm to calculate fundamental matrix F. (For specific algorithm, see Section 3.2.4)
2) For each pair of feature points in the data set $\boldsymbol{S}$, calculate the equations of corresponding epipolar line $\boldsymbol{l}_t$ and $\boldsymbol{l}_{t+1}$.

$$\boldsymbol{l}_t = F^T \tilde{\boldsymbol{m}}_{t+1} \tag{3.25}$$

$$\boldsymbol{l}_{t+1} = F \tilde{\boldsymbol{m}}_t \tag{3.26}$$

Here, $\boldsymbol{l}_t = [a_t \quad b_t \quad c_t]^T$; $\boldsymbol{l}_{t+1} = [a_{t+1} \quad b_{t+1} \quad c_{t+1}]^T$;
$\tilde{\boldsymbol{m}}_t = [x_t \quad y_t \quad 1]^T$; $\tilde{\boldsymbol{m}}_{t+1} = [x_{t+1} \quad y_{t+1} \quad 1]^T$.

For example, if current point pair $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t+1}$ are inliers (matched), they should satisfy the epipolar constraint: For $\boldsymbol{m}_t$ in the first image, the corresponding epipolar line is $\boldsymbol{l}_{t+1}$. Similarly, $\boldsymbol{l}_t$ represents the epipolar line corresponding to $\boldsymbol{m}_{t+1}$ in the second image. $\boldsymbol{m}_{t+1}$ lies on the epipolar line $\boldsymbol{l}_{t+1}$: $\boldsymbol{m}_{t+1}^{\mathrm{T}} \boldsymbol{l}_{t+1} = 0$ .Similarly, $\boldsymbol{m}_t$ lies on the epipolar line $\boldsymbol{l}_t$: $\boldsymbol{m}_t^{\mathrm{T}} \boldsymbol{l}_t = 0$.

3) Calculate the distance d_t from point $\boldsymbol{m}_t$ to the epipolar line $\boldsymbol{l}_t$. Calculate the distance d_{t+1} from point $\boldsymbol{m}_{t+1}$ to the epipolar line $\boldsymbol{l}_{t+1}$.

$$d_t = \frac{|a_t x_t + b_t y_t + c_t|}{\sqrt{a_t^2 + b_t^2}} \tag{3.27}$$

$$d_{t+1} = \frac{|a_{t+1} x_{t+1} + b_{t+1} y_{t+1} + c_{t+1}|}{\sqrt{a_{t+1}^2 + b_{t+1}^2}} \tag{3.28}$$

4) If $\max(d_t, d_{t+1})$ is smaller than the preset threshold, the current point pair $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t+1}$ are classified as inliers; otherwise, outliers.
5) Iterate steps 2), 3) and 4) until all point pairs in the data set $\boldsymbol{S}$ are classified as inliers

or outliers. Record the number of inliers.

6) If the number of inliers is greater than some threshold *T*, re-estimate the fundamental matrix *F* using all inliers and terminate.
7) If the number of inliers is less than *T*, select a new subset and repeat the above process.
8) After *N* trials, using the results corresponding to the largest number of inliers, re-estimate the fundamental matrix *F*.

3.2.4 Fundamental Matrix [44]

The epipolar geometry is the intrinsic projection geometry between two views. It is independent of a scene structure, and only depends on the camera's internal parameters and the relative pose. The fundamental matrix *F* is the algebraic representation of epipolar geometry.

Definition: Suppose we have two images acquired by cameras, then the fundamental matrix *F* is a unique 3×3 rank 2 homogeneous matrix which satisfies:

$$\tilde{\boldsymbol{m}}_{t+1}{}^{\mathrm{T}} F \tilde{\boldsymbol{m}}_t = 0 \tag{3.29}$$

for all corresponding points $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t+1}$.

Here,

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \qquad \tilde{\boldsymbol{m}}_t = \begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} \qquad \tilde{\boldsymbol{m}}_{t+1} = \begin{bmatrix} x_{t+1} \\ y_{t+1} \\ 1 \end{bmatrix}$$

In this paper, we use the 8-points algorithm to compute the fundamental matrix.

8-points Algorithm [45]**:**

Equation (3.29) can be easily written in terms of the known coordinates $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t+1}$.

$$[x_t x_{t+1} \quad y_t x_{t+1} \quad x_{t+1} \quad x_t y_{t+1} \quad y_t y_{t+1} \quad y_{t+1} \quad x_t \quad y_t \quad 1] \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{3.30}$$

From a set of *n* point matches, we obtain a set of linear equations of the form

$$Mf = \begin{bmatrix} x_{t,1}x_{t+1,1} & y_{t,1}x_{t+1,1} & x_{t+1,1} & x_{t,1}y_{t+1,1} & y_{t,1}y_{t+1,1} & y_{t+1,1} & x_{t,1} & y_{t,1} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{t,n}x_{t+1,n} & y_{t,n}x_{t+1,n} & x_{t+1,n} & x_{t,n}y_{t+1,n} & y_{t,n}y_{t+1,n} & y_{t+1,n} & x_{t,n} & y_{t,n} & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.31}$$

With respect to ***Mf*** = **0**, in order to derive a solution except for the trivial solution ***f*** = **0**, the solution ***f*** is calculated which makes $\|\boldsymbol{Mf}\|$ the minimum under the condition that $\|\boldsymbol{f}\| = 1$. The following theorem holds:

Theorem [44]**:** Let *A* be a matrix having *m* rows and *n* columns and ***x*** be a column vector having *n* components. The vector ***x*** which realizes $\|A\boldsymbol{x}\| \rightarrow \min$ under the condition that $\|\boldsymbol{x}\| = 1$ is given by the eigenvector corresponding to the minimum eigenvalue of the matrix $A^{\mathrm{T}}A$.

From the above theorem, the eigenvector ***f*** corresponding to the minimum eigenvalue of the matrix $M^{\mathrm{T}}M$ makes $\|\boldsymbol{Mf}\|$ the minimum. This ***f*** apparently gives the *F* matrix which satisfies Eq.(3.31) and hence Eq. (3.29), but it doesn't satisfy $rank(F) = 2$.

In order to enforce this constraint ($rank(F) = 2$), the following steps are adopted:

By applying the singular value decomposition to the matrix *F* obtained from above, we have

$$F = U\Sigma V^{\mathrm{T}} = U\begin{pmatrix} d_{11} & & \\ & d_{22} & \\ & & d_{33} \end{pmatrix}V^{\mathrm{T}} \tag{3.32}$$

Let us make $d_{33} = 0$. Then we have

$$F' \equiv U\begin{pmatrix} d_{11} & & \\ & d_{22} & \\ & & 0 \end{pmatrix}V^{\mathrm{T}} \tag{3.33}$$

This matix F' satisfies $rank(F') = 2$ and it is employed finally as the F matrix.

The key to success with the 8-points algorithm is proper careful normalization of the input data before constructing the equations to solve. In the case of the 8-points algorithm, the suggested normalization is a translation and scaling of each image so that the centroid of the reference points is at the origin of the coordinates and the distance of the points from the origin is equal to or less than $\sqrt{2}$.

Fundamental Matrix Algorithm:

1) Normalization: Transform the image coordinates according to $\hat{\boldsymbol{m}}_t = T_t \boldsymbol{m}_t$ and $\hat{\boldsymbol{m}}_{t+1} = T_{t+1}\boldsymbol{m}_{t+1}$, where T_t and T_{t+1} are normalizing transformations consisting of a translation and scaling.
2) Calculate the fundamental matrix $\hat{F}$ corresponding to the matches $\hat{\boldsymbol{m}}_t \leftrightarrow \hat{\boldsymbol{m}}_{t+1}$ using 8-points algorithm.
3) Denormalization: Set $F = T_{t+1}^{\mathrm{T}}\hat{F}T_t$. Matrix F is the fundamental matrix corresponding to the original data $\boldsymbol{m}_t \leftrightarrow \boldsymbol{m}_{t+1}$.

3.2.5 Camera Motion Parameters

The camera motion parameters consist of a rotation matrix R and a translation vector $\boldsymbol{T}$ as follows:

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad \boldsymbol{T} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

In this section, we want to calculate these camera motion parameters using the epipolar geometry between two views.

First, the essential matrix E is calculated using the following formula:

$$E = K^{\mathrm{T}} F K \tag{3.34}$$

Here, K is a camera inner parameters matrix (obtained from camera calibration), F is a fundamental matrix which has been calculated in section 3.2.4.

Next, the essential matrix E can be represented by motion parameters of a camera between two images, the rotation matrix R and the translation vector $\boldsymbol{T}$.

$$E = [\boldsymbol{T}]_X R \tag{3.35}$$

Here,

$$[\boldsymbol{T}]_X = \begin{bmatrix} 0 & -t_Z & t_Y \\ t_Z & 0 & -t_X \\ -t_Y & t_X & 0 \end{bmatrix} \tag{3.36}$$

is the corresponding skew-symmetric matrix of the translation vector $\boldsymbol{T}$.

Finally, we obtain the matrix R and translation vector $\boldsymbol{T}$ by applying the singular value decomposition to the essential matrix E.

The singular value decomposition of the essential matrix E is as follows:

$$E = U \Sigma V^T \tag{3.37}$$

Using the results of the singular value decomposition (Eq.(3.37)), we can calculate the rotation matrix R and the translation vector $\boldsymbol{T}$:

$$R = UWV^{\mathrm{T}} \quad \text{or} \quad R = UW^{\mathrm{T}}V^{\mathrm{T}} \tag{3.38}$$

$$[\boldsymbol{T}]_X = U\Sigma W U^{\mathrm{T}} \quad \text{or} \quad [\boldsymbol{T}]_X = U\Sigma W^{\mathrm{T}} U^{\mathrm{T}} \tag{3.39}$$

Here,

$$W = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

There are four possible choices of the camera motion parameters, based on the two possible choices of R and two possible choices of $\boldsymbol{T}$.

The selection method of camera motion parameters is, using four combinations, to do the 3D coordinate estimation and then to check which combination's compensation image is correct.

3.3 3D Coordinate Estimation

In this section, we estimate the coordinates of 3D points in the world coordinate system using two corresponding feature points in two successive images.

3.3.1 The World Coordinate System and the Camera Coordinate System

In this 3D coordinate estimation, two successive images are used at any time, i.e., the image I_t taken at time t and I_{t+1} taken at time t+1. Because a camera is moving in this research, a camera is in different locations in these two moments. The camera locations and coordinates are shown in Fig. 3.3.

Here, two blue points are camera lens at time t and time t+1, respectively; (X_t, Y_t, Z_t) and $(X_{t+1}, Y_{t+1}, Z_{t+1})$ are camera coordinates at time t and time t+1, respectively; (X_W, Y_W, Z_W) is the world coordinate which coincides with the camera coordinate at time t; M is one 3D point in the world coordinate system; $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t+1}$ are the 2D points in the virtual image planes which correspond to the 3D point M; matrix R and vector $\boldsymbol{T}$ are camera motion parameters from time t to time t+1 which have been calculated in Section 3.2.

3.3.2 A Camera Model

Figure 3.4 shows camera geometry in which (X_W, Y_W, Z_W) is the world coordinate system; (X, Y, Z) is a camera coordinate system; M is the 3D point; X_c is the coordinate of M in the camera coordinate system; X is the coordinate of M in the world coordinate system; $\boldsymbol{x} = (x, y)$ is the 2D point coordinate with the unit of millimeter in the virtual image plane, and $\boldsymbol{m} = (u, v)$ is the 2D point coordinate with the unit of pixel in the virtual image plane.

The relationship between $\boldsymbol{m}$ and $\boldsymbol{x}$ are given by

$$\tilde{m} = A\tilde{x} \tag{3.40}$$

where A is a 3×3 matrix, this parameter is decided by a camera; $\tilde{m} = (u, v, 1)$; $\tilde{x} = (x, y, 1)$.

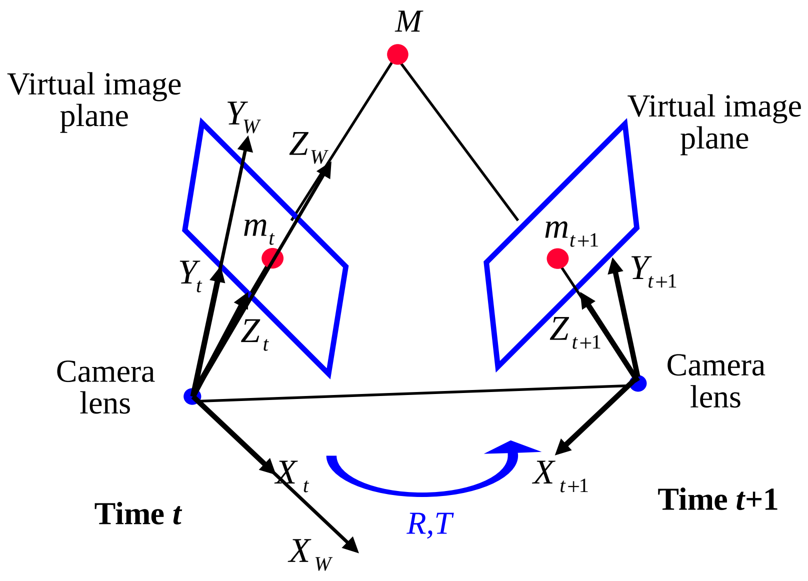


Fig. 3.3 Camera locations and coordinates.

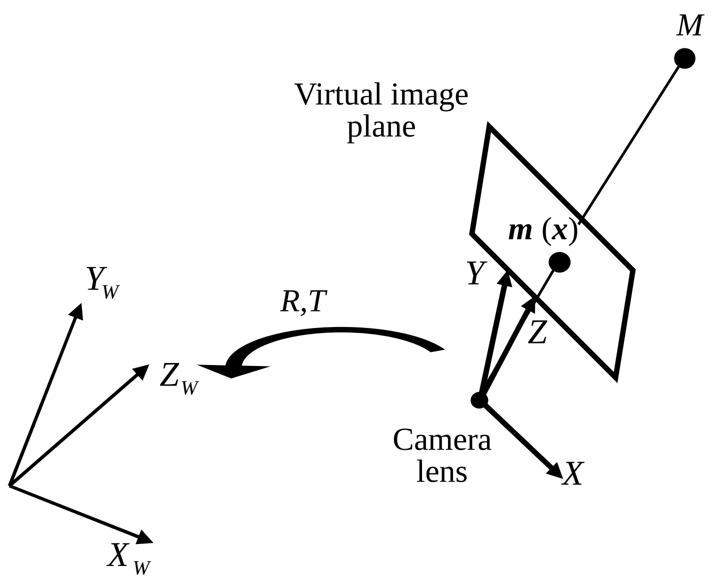


Fig. 3.4 Camera geometry.

The relationship between the camera coordinate system and the virtual image plane is given by

$$\lambda\tilde{x} = P_f \tilde{X}_c \tag{3.41}$$

where λ is a constant and P_f is a 3×4 matrix. This parameter is decided by a camera.

Putting Eq. (3.41) together with Eq. (3.40) leads to the formula

$$\lambda\tilde{\boldsymbol{m}} = AP_f \tilde{X}_c \tag{3.42}$$

The world coordinate and the camera coordinate are related via a rotation and a translation. The relationship is given as

$$\tilde{X}_c = M\tilde{X} = \begin{bmatrix} R & \boldsymbol{T} \\ 0 & 1 \end{bmatrix}\tilde{X} \tag{3.43}$$

where the parameters of R and $\boldsymbol{T}$ which relate the camera orientation and position to the world coordinate system are called the external parameters.

Putting Eq. (3.43) together with Eq. (3.42) leads to the formula

$$\lambda\tilde{\boldsymbol{m}} = AP_f M\tilde{X} = K[R \quad \boldsymbol{T}]\tilde{X} \tag{3.44}$$

where K is the calibration matrix of the camera (internal camera parameters), which can be obtained from camera calibration.

Finally, we have the projection equation, Eq. (3.44). This equation indicates the relationship between 2D points in the image plane and 3D points in the world coordinate system.

3.3.3 3D Coordinate

In Fig. 3.3, if we just consider the moment *t*, we can get the camera geometry at time *t* (shown in Fig. 3.5). At this moment, the world coordinate system coincides with the camera coordinate system. According to this, rotation matrix R is equal to a unit matrix I; translation matrix T is equal to zero vector $\boldsymbol{0}$. Projection equation Eq. (3.44) is rewritten as

$$\lambda_t\tilde{\boldsymbol{m}}_t = K[I \quad \boldsymbol{0}]\tilde{X} \tag{3.45}$$

At the moment *t*+1, the camera geometry is shown in Fig. 3.6. At this moment, the world coordinate system and the camera coordinate system are in different locations. Because the world coordinate system coincides with the camera coordinate system at time *t*, and the motion parameters between two camera coordinate systems have been calculated in Section 3.2, rotation matrix *R* and translation vector ***T*** is known. Projection equation Eq. (3.44) is rewritten as

$$\lambda_{t+1}\tilde{\boldsymbol{m}}_{t+1} = K[R \quad \boldsymbol{T}]\tilde{X} \tag{3.46}$$

According to the forms of the following matrices (Eq. (3.47) and Eq. (3.48)),

$$K = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \tag{3.47}$$

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \tag{3.48}$$

Eq. (3.45) and Eq. (3.46) are easily rewritten in terms of the known coordinates $\boldsymbol{m}_t$ and $\boldsymbol{m}_{t+1}$ as follows:

$$\lambda_t \begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.49}$$

$$\lambda_{t+1} \begin{bmatrix} u_{t+1} \\ v_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.50}$$

Based on Eq. (3.49) and Eq. (3.50), we can calculate the 3D coordinates of point *M*.

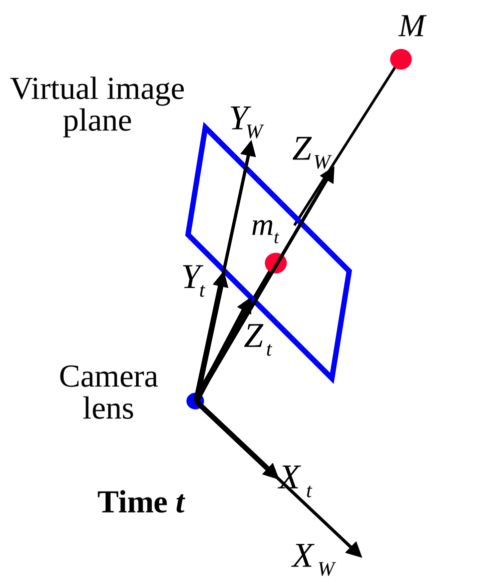


Fig. 3.5 Camera geometry at time t.

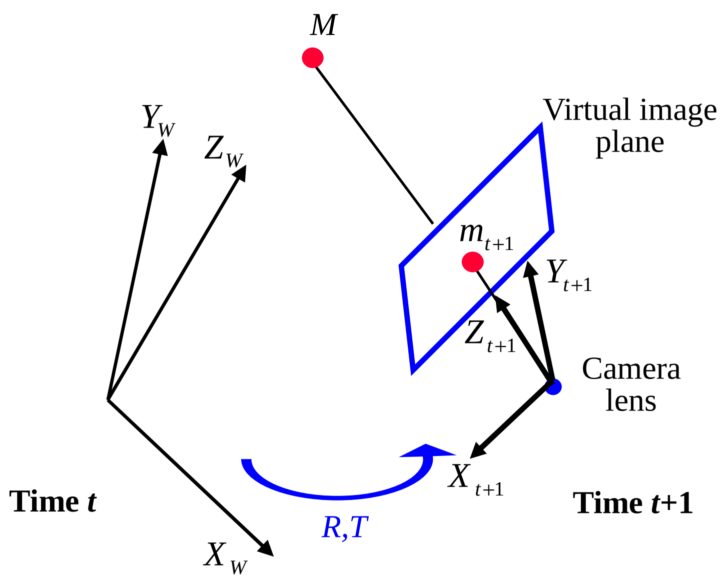


Fig. 3.6 Camera geometry at time *t*+1.

3.4 Road Plane Estimation

In this section, we estimate the parameters of the road plane using the 3D coordinates of feature points which locate in the road region.

The equation of a road plane is given by

$$aX + bY + cZ + d = 0 \tag{3.51}$$

Since the norm of the normal vector of the road plane is 1, we can calculate the parameters of the road plane using three 3D points.

However, because of the false detection in the road region detection, some points which are identified as being in the road region in the previous frame and some points which have false 3D positions due to false correspondences may not actually exist on the road. To escape from these difficulties, the LMedS (Least Median of Squares) estimator [46] is used in this paper for estimating the parameters of the road plane.

3.5 2D and 3D Classification

Once we obtain the equation of a road plane, we can calculate the distances from the 3D positions of the feature points to the road plane. Based on the distance, we determine whether the 3D position of a feature point is located in the road plane.

For a feature point (x, y), (X, Y, Z) is the corresponding 3D position of this feature point. We calculate the distance from (X, Y, Z) to the road plane (Eq. (3.51)) as follows:

$$d = \frac{|aX + bY + cZ + d|}{\sqrt{a^2 + b^2 + c^2}} \tag{3.52}$$

If the distance *d* is smaller than the threshold T_{dis}, it means this 3D position of the feature point is located in the road plane. Then this feature point (x, y) is classified as a 2D point; otherwise, classified as a 3D point.

For all feature points of an object, we carry out this classification operation to classify them as 2D points or 3D points. Then, we count the number of 2D points and the number of 3D points. If the number of 2D points is larger than the number of 3D points, then this object is considered as a 2D object. Otherwise, this object is considered as a 3D object.

3.6 Experimental Results

First, we construct a set of feature points S_t located in the road region using the result of road region detection in the first image I_t taken at time *t*.

Second, we construct a set of corresponding feature points S_{t+1} using the

Lucas-Kanade Tracker in the second image I_{t+1} taken at time *t*+1.

Third, we construct a set of 3D candidate points on the road by calculating 3D coordinates of the feature points sets S_t and S_{t+1} using the method described in Section 3.3.

Fourth, the equation of the road plane is estimated from the 3D coordinates of these candidate points.

Finally, 2D and 3D objects are classified based on this road plane in the resultant image of obstacles detection.

Figure 3.7 and Fig. 3.8 show experimental results of 2D and 3D objects classification of video 2 and video 3. In Fig. 3.7 and Fig. 3.8, (a) shows the input images and (b) is the result of 2D and 3D points classification. In Fig. 3.7 (b) and Fig. 3.8 (b), black regions mean the shapes of the obstacles which have been detected in Chapter 2; red points mean 2D points; blue points mean 3D points. Then based on the number of 2D and 3D points, we obtain the result of 2D and 3D objects classification (shown in Fig. 3.7 (c) and Fig. 3.8 (c)); blue regions mean 3D objects, whereas red regions mean 2D objects.

3.7 Evaluation

In order to evaluate the effectiveness of the proposed 2D and 3D objects classification method, we calculate *Precision* using the following formula:

$$Precision = \frac{N_{correct}}{N_{all}} \times 100[\%] \tag{3.53}$$

Here $N_{correct}$ is the number of areas classified correctly and N_{all} is the number of total areas. The results of evaluation are shown in Table 3.1.

Table 3.1 The result of evaluation of two different videos using the proposed 2D and 3D objects classification method.

Videos	***Precision*[%]**
Video 2	88.7
Video 3	85.4

3.8 Discussion and Conclusion

In this chapter, we proposed a method for classifying 2D objects and 3D objects in the resultant images of obstacles detection described in Chapter 2.

In the part of camera motion estimation, in order to improve the calculation accuracy of the fundamental matrix, we normalized the coordinates of input data before constructing the equations to calculate the fundamental matrix F using the 8-point algorithm.

In the part of road plane estimation, we use the points that are contained in the road region detected in Chapter 2 to estimate the parameters of the road plane. In this estimation, because there are some errors in the calculation, we use the LMedS estimator.

In the obstacles detection method which we proposed in Chapter 2, if there is a paper or other 2D objects on the road, they are detected as obstacles. But these 2D objects are false obstacles. The method proposed in Chapter 3 can classify the 2D objects and 3D objects in the resultant image of the obstacles detection and delete these 2D objects using the result of object classification successfully. In this way, the obstacle detection method proposed in Chapter 2 has been improved and gained better accuracy.

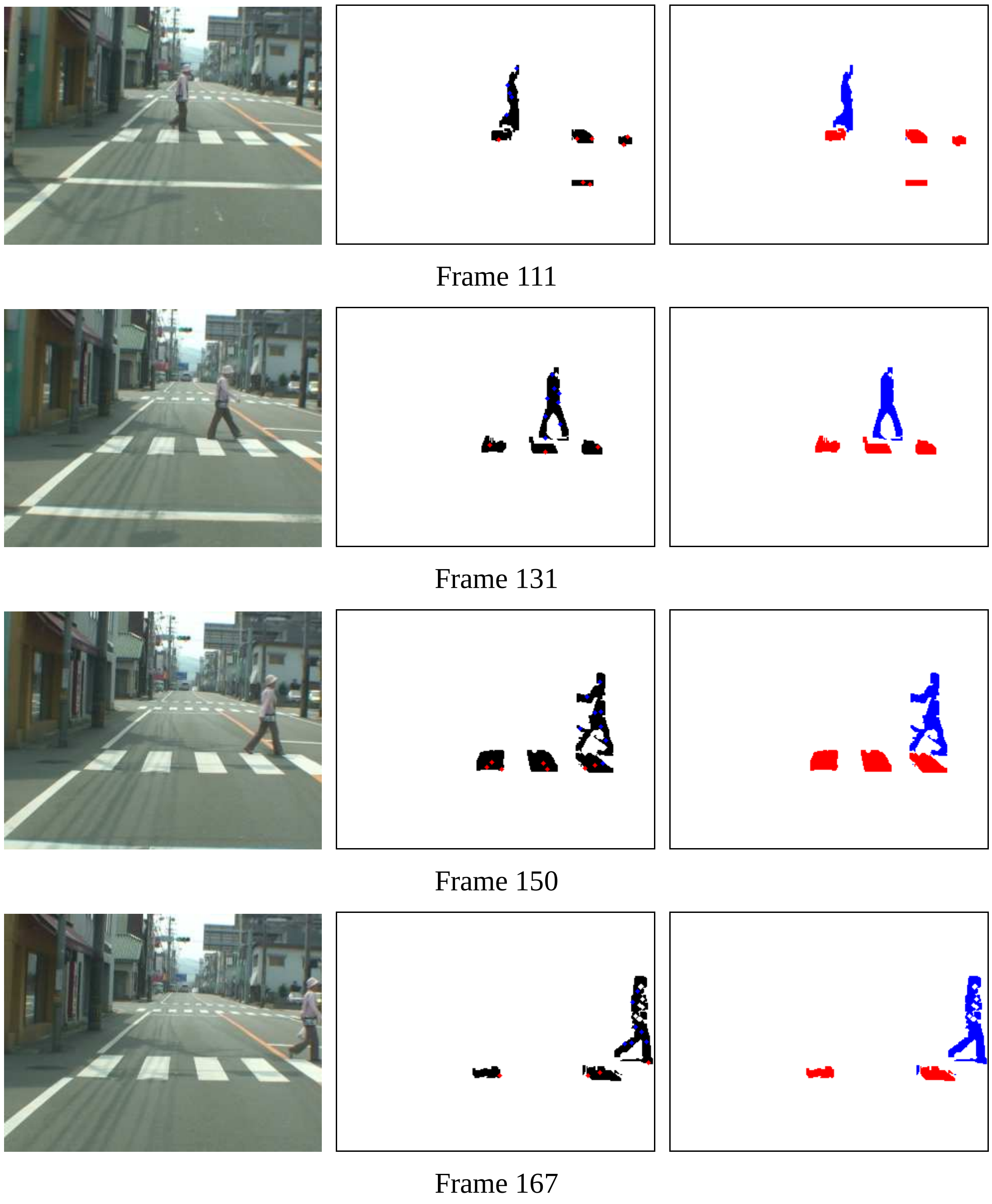


Fig. 3.7 Experimental results of 2D and 3D objects classification (Video 2). (a) Input images, (b) the result of 2D and 3D points classification, (c) the result of 2D and 3D objects classification.

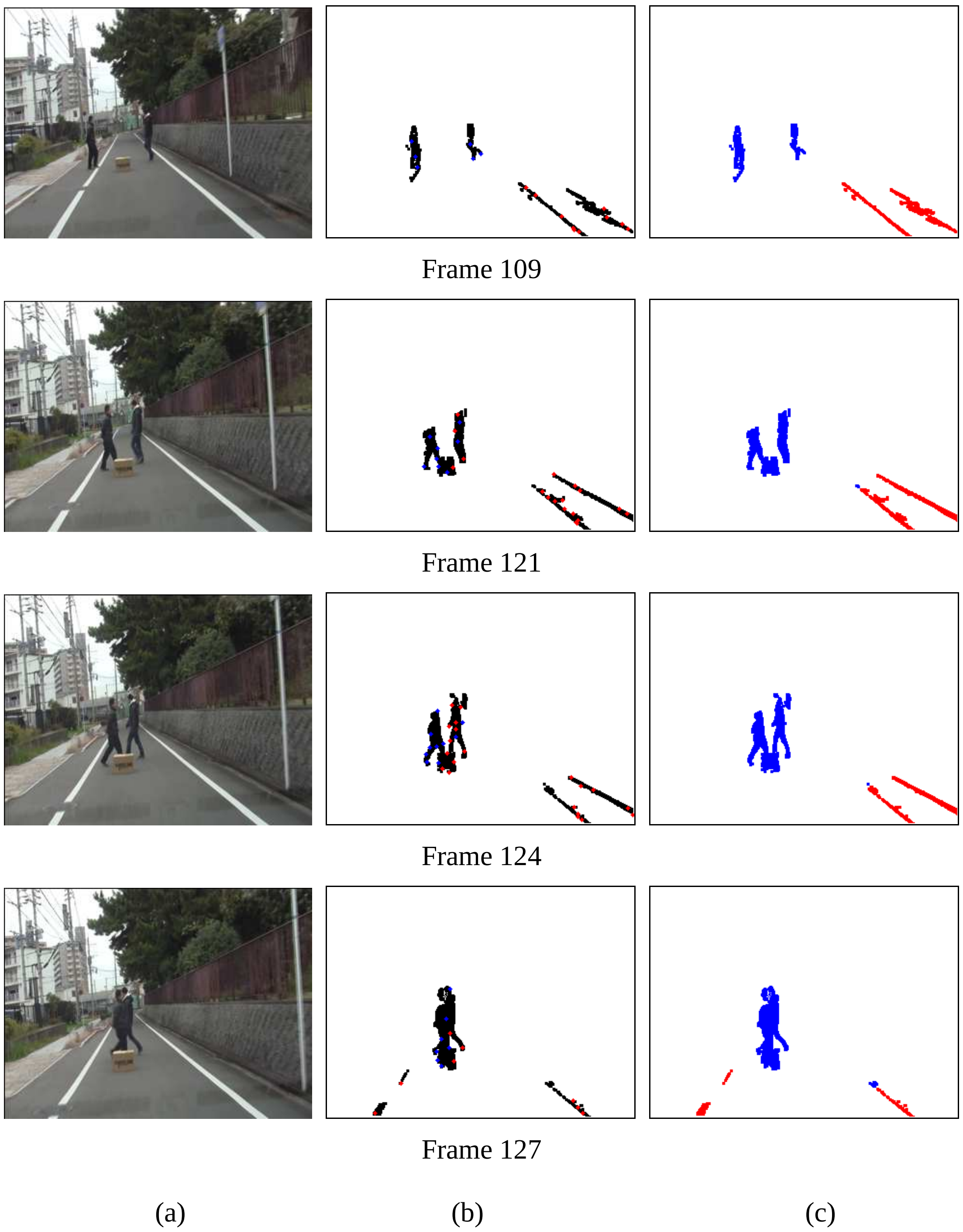


Fig. 3.8 Experimental results of 2D and 3D objects classification (Video 3). (a) Input images, (b) the result of 2D and 3D points classification, (c) the result of 2D and 3D objects classification.

Chapter 4　Final Experimental Results and Evaluation

In this chapter, we use the results of 2D and 3D objects classification to optimize the results of obstacles detection. We also carry out a comparative experiment using the same experimental videos to discuss the effectiveness of the proposed obstacles detection method.

4.1　The Proposed Obstacles Detection Methods

In this section, we summarize two obstacles detection methods which we have proposed in Chapter 2 and Chapter 3.

The Proposed Method 1:

STEP1: Background modeling (described in Section 2.2).

STEP2: Road region detection (described in Section 2.3).

In this road region detection, the training data contains the pixels of road marks. In the result of this road region detection, the road marks are classified as a part of the road region (shown in Fig. 2.15 – Fig. 2.16).

STEP3: Region classification (described in Section 2.4.2).

STEP4: Classification of foreground objects (described in Section 2.4.3).

The results of obstacles detection using the proposed method 1 are shown in Fig. 2.17 – Fig. 2.19.

The Proposed Method 2:

STEP1: Background modeling (described in Section 2.2).

STEP2: Road region detection (described in Section 2.3).

In this road region detection, the training data doesn’t contain the pixels of road marks. In the result of this road region detection, the road marks are classified as the non-road region (shown in Fig. 2.20 (b) and Fig. 2.21 (b)).

STEP3: Region classification (described in Section 2.4.2).

STEP4: Classification of foreground objects (described in Section 2.4.3).

The results of operations from STEP 1 to STEP 4 are shown in Fig. 2.20 – Fig. 2.21.

STEP5: 2D and 3D objects classification (described in Chapter 3).

The results of 2D and 3D objects classification are shown in Fig. 3.7 – Fig.

3.8.

STEP6: Deleting 2D objects in the resultant images of STEP 4.

Figure 4.1 and Fig. 4.2 show the experimental results of detecting 2D objects, these results are the results of obstacles detection using the proposed method 2.

In Fig. 4.1 and Fig. 4.2, (a) shows the resultant images of STEP 4 (also shown in Fig. 2.20 (d) and Fig. 2.21 (d)). (b) shows the results of 2D and 3D objects classification (also shown in Fig.3.7 (c) and Fig. 3.8 (c)); blue regions mean 3D objects, red regions mean 2D objects. We can get the final results of obstacles detection after deleting 2D objects in (a) using the results shown in (b) and deleting the small noise. The results of obstacles detection using the proposed method 2 are shown in Fig. 4.1 (c) and Fig. 4.2 (c).

4.2 The Method of Comparative Experiment

In this section, we introduce a comparative method.

The Comparative Method:

STEP1: Background modeling (described in Section 2.2).

STEP2: Road region detection.

In this road region detection, we detect the road regions using another method. This road region detection method has been proposed in [47-48], which is called the road region detection method using motion compensation.

The road region detection method using motion compensation assumes all the 3D points in the world coordinate system which correspond to the pixels in the first image are on the road plane. Based on this assumption, using the camera motion parameters and epipolar geometry, it warps the first image and gets the warped image. Then the second image is compared with the warped image, the different region between these two images are caused by 3D points which are not on the road plane and this region is the non-road region; the similar region in these two images is the road region.

STEP3: Region classification (described in Section 2.4.2).

STEP4: Classification of foreground objects (described in Section 2.4.3).

The results of obstacles detection using the comparative method are shown in Fig. 4.3 – Fig. 4.5.

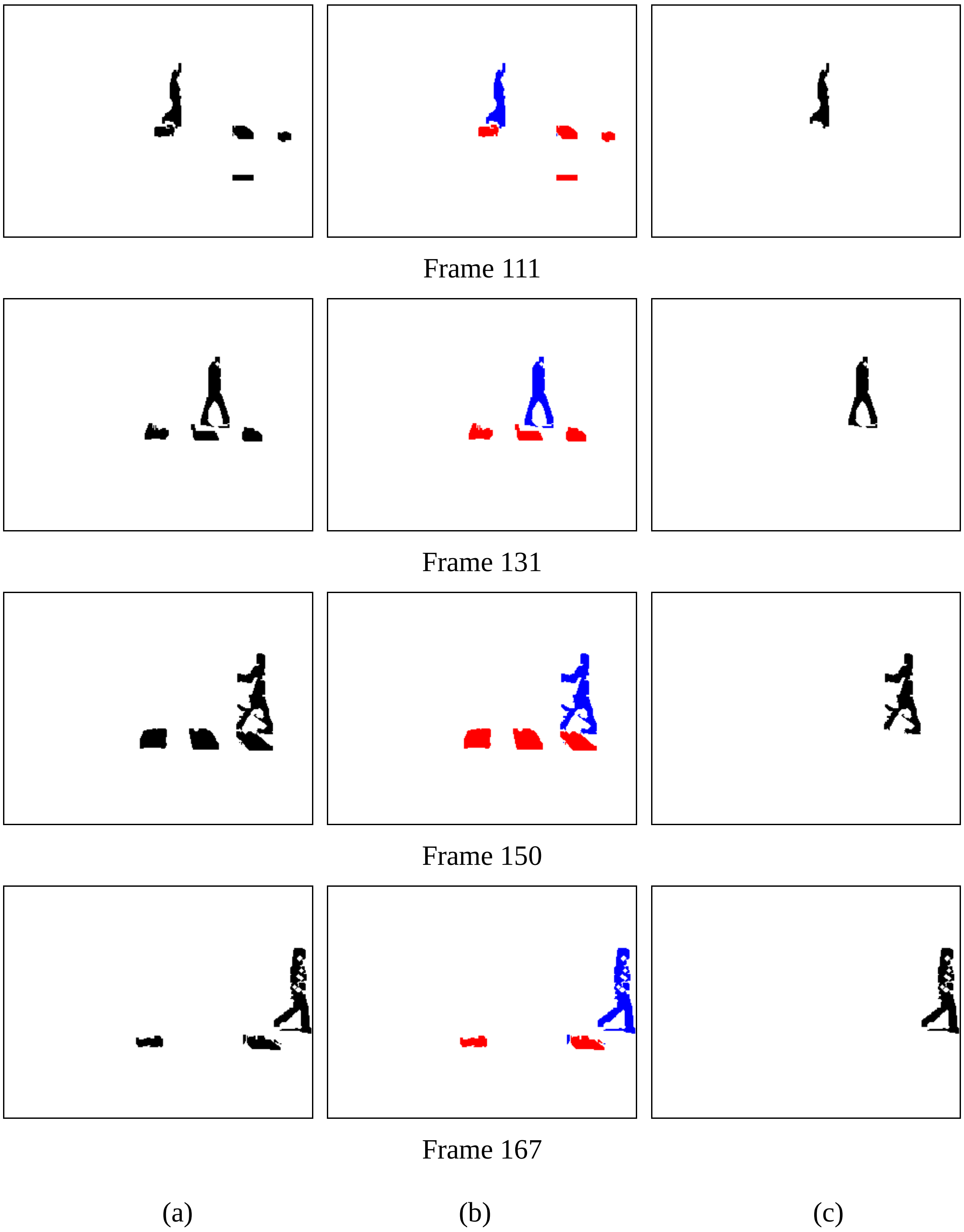


Fig. 4.1 The results of obstacles detection using the proposed method 2 (Video 2). (a) The resultant images of STEP 4, (b) the results of 2D and 3D objects classification, (c) the results of obstacles detection using the proposed method 2.

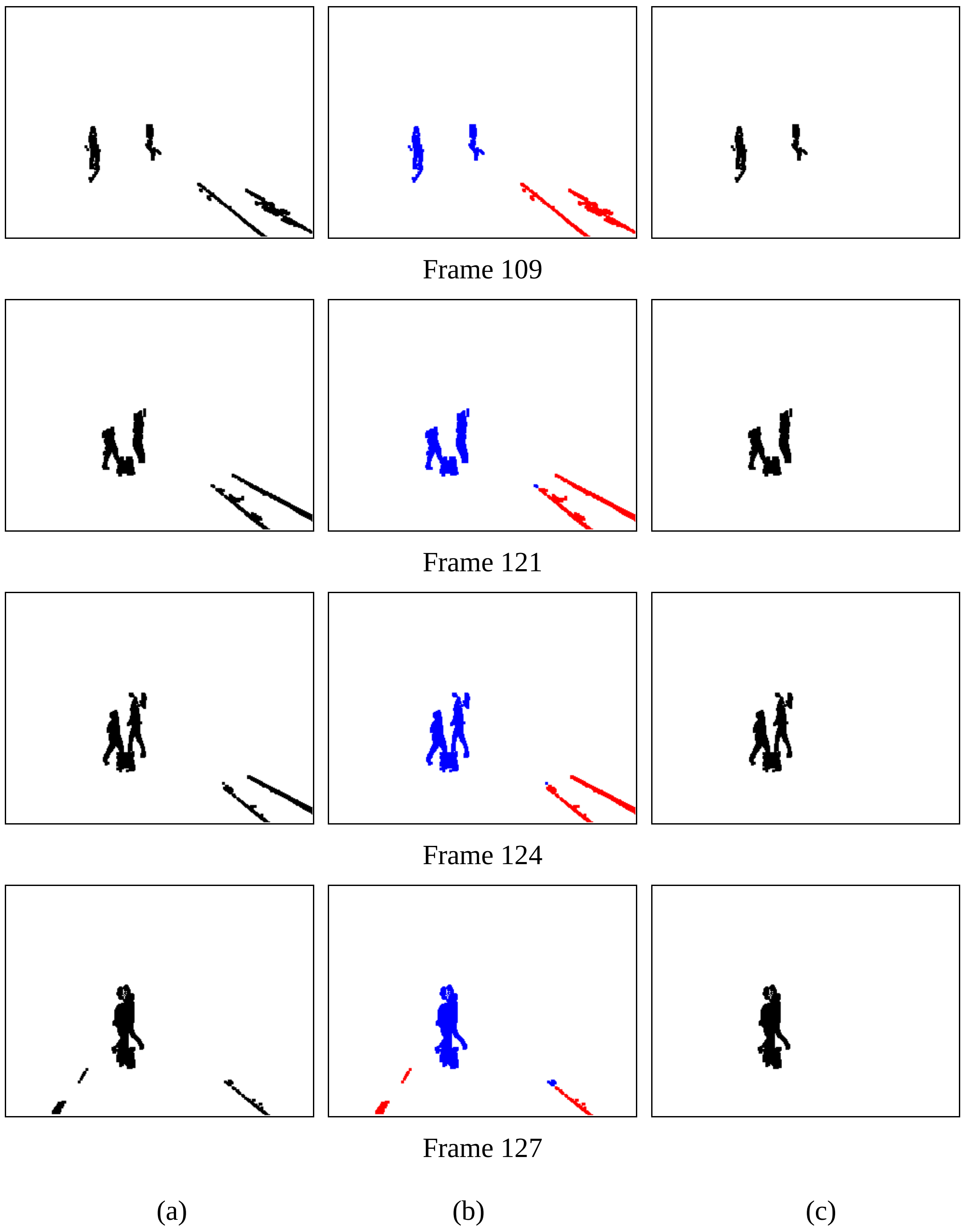


Fig. 4.2 The results of obstacles detection using the proposed method 2 (Video 3). (a) The resultant images of STEP 4, (b) the results of 2D and 3D objects classification, (c) the results of obstacles detection using the proposed method 2.

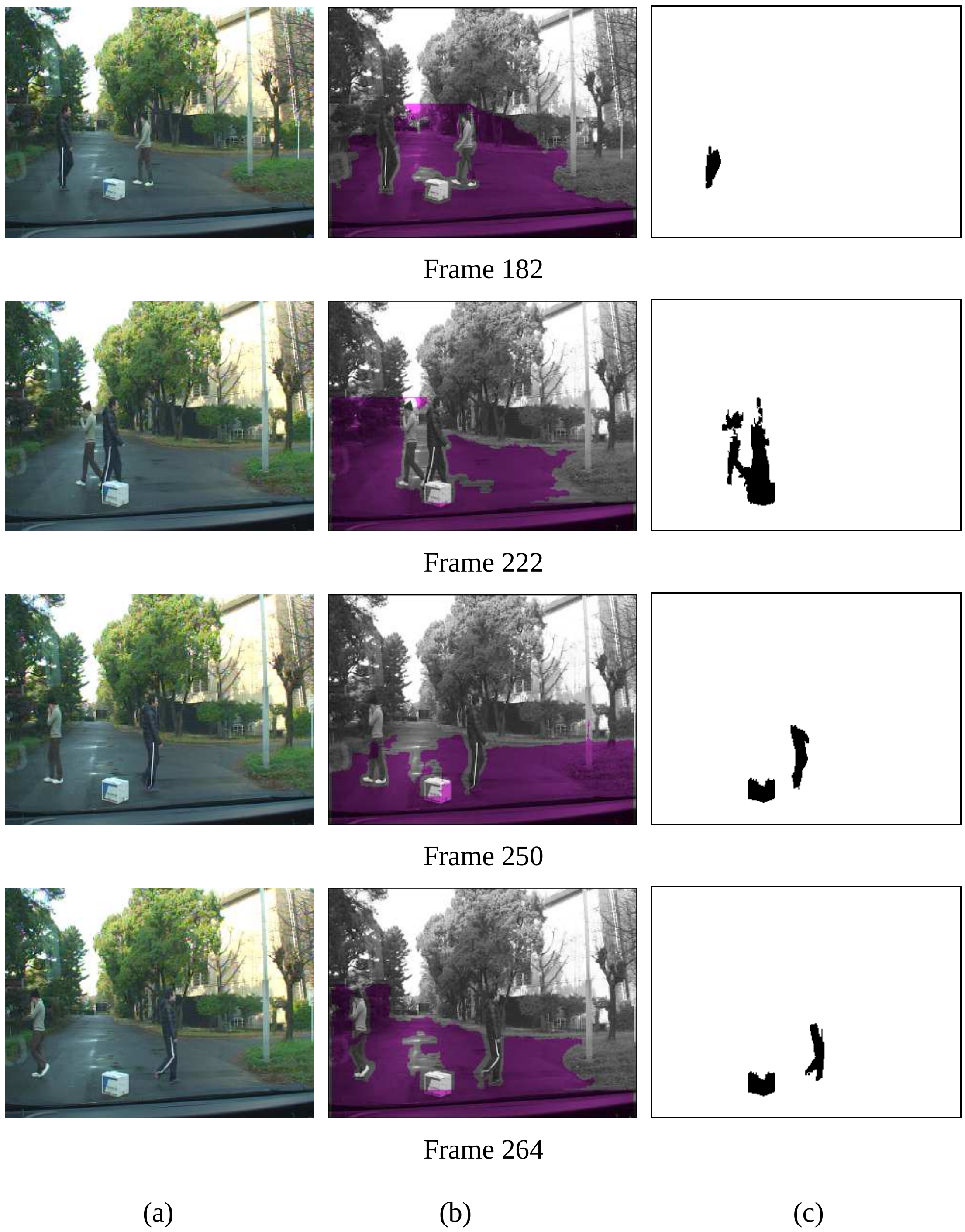


Fig. 4.3 The results of obstacles detection using the comparative method (Video 1). (a) Input images, (b) the results of road region detection using motion compensation, (c) the results of obstacles detection of comparative experiment.

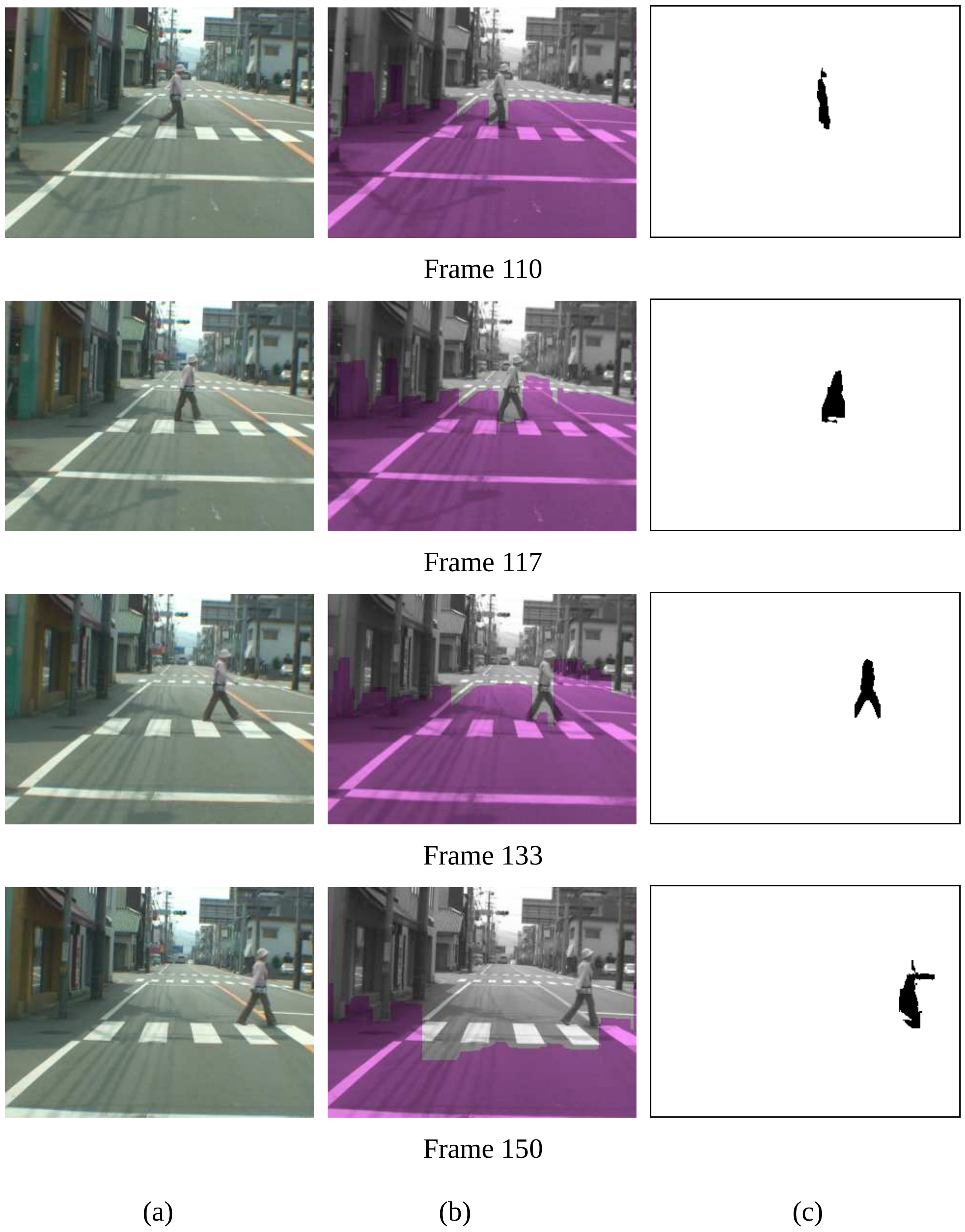


Fig. 4.4 The results of obstacles detection using the comparative method (Video 2). (a) Input images, (b) the results of road region detection using motion compensation, (c) the results of obstacles detection of comparative experiment.

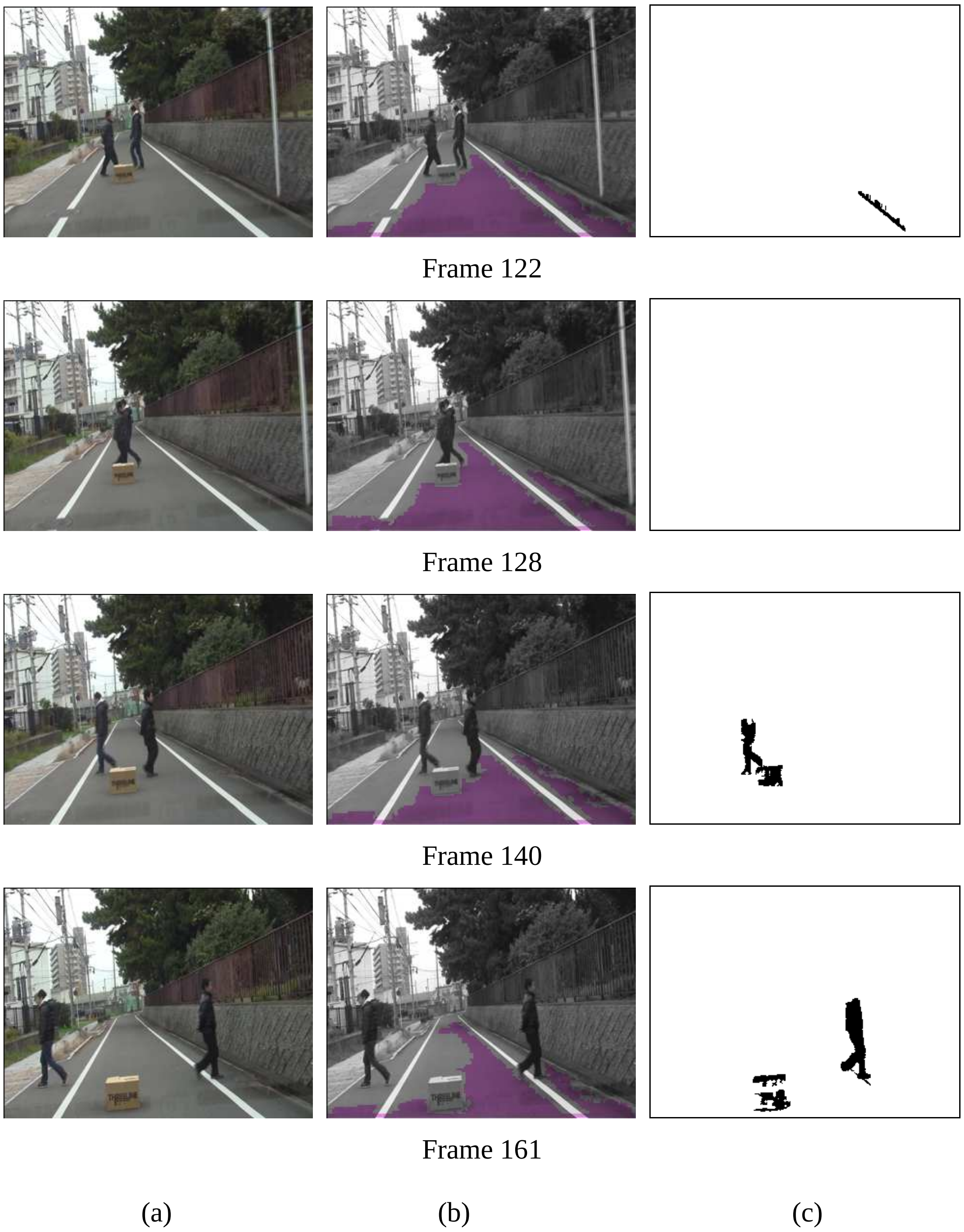


Fig. 4.5 The results of obstacles detection using the comparative method (Video 3). (a) Input images, (b) the results of road region detection using motion compensation, (c) the results of obstacles detection of comparative experiment.

4.3 Evaluation

In this section, we evaluate the effectiveness of the proposed method 1, the proposed method 2 and the comparative method using the evaluation method described in Section 2.6. The result of evaluation is composed of three values: *Precision*, *Recall* and *FPR*. *Precision* is a measure of exactness or fidelity; *Recall* is a measure of completeness; *FPR* is a measure of inaccuracy.

Table 4.1 shows the results of evaluation with Video 1 using two different detection methods (the proposed method 1 and the comparative method).

Table 4.2 – Table 4.3 show the results of evaluation with Video 2 and Video 3 using three different detection methods (the proposed method 1, the proposed method 2 and the comparative method).

Table 4.1 The result of evaluation of Video 1 using two different detection methods.

Methods	**Evaluation Values**		
	***Precision*[%]**	***Recall*[%]**	***FPR*[%]**
The Proposed Method 1	97.5	79.7	2.5
The Comparative Method	94.9	74.6	5.1

Table 4.2 The result of evaluation of Video 2 using three different detection methods.

Methods	**Evaluation Values**		
	***Precision*[%]**	***Recall*[%]**	***FPR*[%]**
The Proposed Method 1	94.5	80.0	5.5
The Proposed Method 2	94.0	80.0	6.0
The Comparative Method	92.9	80.0	7.1

Table 4.3 The result of evaluation of Video 3 using three different detection methods.

Methods	**Evaluation Values**		
	***Precision*[%]**	***Recall*[%]**	***FPR*[%]**
The Proposed Method 1	96.8	73.9	3.2
The Proposed Method 2	95.2	73.5	4.8
The Comparative Method	93.4	55.7	6.6

4.4 Discussion

In this chapter, we summarized two proposed obstacles detection methods which have been described in Chapter 2 and Chapter 3. We also introduced a comparative obstacles detection method, and then compared these three obstacles detection methods and evaluated the effectiveness of them.

According to the results of evaluation (shown in Table 4.1 – Table 4.3), the proposed method 1 and the proposed method 2 work well than the comparative method. It is because, in the part of road region detection, the comparative method detected the road region using the method based on motion compensation. This road region detection method using motion compensation needs camera calibration [49] and must calculate the motion parameters of the car. Due to the error of these calculations, the road region detection results are worse than the road region detection method using SVM. This also leads to the low precision of the final obstacles detection.

The performance of the proposed obstacles detection method depends on the size of an obstacle in an image or the distance between the obstacle and the car. If the size of an obstacle in an image is too small, it is recognized as noise and deleted. According to the performed experiments, the maximum feasible distance of detecting a pedestrian is about 70m.

The distance that a car moves during the period from the detection system starts detecting obstacles until the car stops is defined as a stop distance. When this stop distance is larger than the maximum feasible distance, the obstacle detection makes no sense as the car crashes against the obstacle. We calculate the stop distance using the following formula:

$$d = vt_f + vt_r + d_v \tag{4.1}$$

where v is the speed of a car; t_f is the processing time of each frame; t_r is the driver reaction time; d_v is the distance needed to stop the car from the instance of brake application begins. The term vt_f means the distance that the car moves during the process of the detection; vt_r is the distance that the car moves during the driver reaction time. The processing time of the detection method t_f is 695ms/f. The parameters t_r and d_v can be found in [50]. Then the stop distance can be calculated and the results are shown in Table 4.4.

Because the stop distance must be smaller than the maximum feasible distance, the proposed method can be applied to a vehicle driving up to 45 km/h. This obstacles detection method can be applied to the real driving condition of a vehicle in the city.

Table 4.4 Stop distance.

Speed of a car (km/h)	20	30	40	50	60
Stop distance (m)	24	41	58	75	97

Chapter 5 Conclusion

5.1 Conclusion

In this thesis, we proposed an obstacles detection method using a video taken by a vehicle-mounted monocular camera.

In Chapter 2, a method of automatic obstacles detection is proposed for detecting obstacles on a road, even if they are moving or static, by the use of background modeling and road region detection.

Background modeling is often used to detect moving objects when a camera is static. In the proposed method, we apply it to a moving camera case in order to obtain foreground images.

In the part of road region detection, we carried out two experiments: the training data contains the pixels of road marks; the training data doesn’t contain the pixels of road marks. In the experiment of the training data which contains the pixels of road marks, it can delete 2D objects (which are considered as false obstacles) in the foreground images; whereas the experiment of the training data which doesn’t contain the pixels of road marks cannot delete 2D objects.

In the performed experiments, it is shown that the experiment of the training data which contains the pixels of road marks works much better than the experiment of the training data which doesn’t contain the pixels of road marks.

In Chapter 3, we proposed a method of classifying 2D objects and 3D objects. This classification method is based on the estimation of a road plane. According to the relationship between the feature points and the road plane, these feature points are classified as 2D points or 3D points. Finally, one object is considered as a 2D object or a 3D object based on the number of 2D points and the number of 3D points on the object.

This classification method can be used to delete 2D objects (considered as false obstacles) in the resultant images of obstacles detection (obtained from the method described in Chapter 2). After deleting these 2D objects, it will improve the accuracy of obstacles detection. In addition to the obstacles detection, the proposed 2D and 3D objects classification method also can be used in other applications (e.g., walking robots).

In Chapter 4, we summarized the obstacles detection methods which we have proposed in Chapter 2 and Chapter 3. In order to prove the effectiveness of the proposed methods, we also introduced a comparative method. In the road region detection part of the comparative method, it detected the road regions using another method (the road region detection method using motion compensation).

This road region detection method using motion compensation needs camera calibration and it must calculate the motion parameters of the car. Due to the error of these calculations, the proposed method 1 and the proposed method 2 work better than the comparative method (shown in the results of evaluation Table 4.1 – Table 4.3).

The proposed obstacles detection method has some advantages over the existing obstacles detection methods:

In the first place, the proposed method uses a monocular camera. This realizes an economic system and smaller computation time. It is also advantageous for achieving real-time processing.

In the second place, in the proposed method, the video which is used to reconstruct the background is captured by a vehicle-mounted camera, and this car is driving at normal speed. The existing background modeling methods are often used in the static camera occasions or in the slow-moving camera occasions (e.g., when a car is driving near the crossroad). This characteristic of the proposed method is important to the industrial applications of a vehicle-mounted camera based obstacles detection system.

The originalities of this thesis are as follows:

In the first place, the proposed method can detect arbitrary objects including both static objects and moving objects. To the best of our knowledge, no researches have ever proposed a method which detects both static and moving objects simultaneously. This is helpful because static objects such as boxes fallen on the road from a car are dangerous for drivers. The existent methods concentrate only on detecting moving objects such as pedestrians, bicycles and cars.

In the second place, the output of the proposed method is the shape of obstacles. Most of the existing obstacles detection methods only indicate the location of an obstacle by a rectangular frame which surrounds it and they do not extract the shape of obstacles. Extraction of the shape of an obstacle is important for obstacles recognition. If the detected obstacle is recognized as a pedestrian from its shape, we can foresee his/her next motion.

In the third place, the proposed 2D and 3D objects classification method can distinguish which objects are 3D objects, and which objects are 2D objects in a pile of objects using a monocular camera. To the best of our knowledge, no researches have ever proposed a method which classified 2D and 3D obstacles on the road. The proposed 2D and 3D objects classification method can be used to delete 2D objects in the resultant images of obstacles detection and improve the accuracy of obstacles detection. It is useful in the obstacles detection and other applications, such as navigation of walking robots.

5.2 Future Work

The proposed obstacles detection method also has disadvantages, and these are our future work.

In the first place, the proposed method is weak on a rainy day since the windscreen wipers hinder visibility of background images.

In the second place, the proposed method is also not effective on hilly roads as the method assumes the road is an even plane while detecting the road region.

In the third place, the proposed method works well in a straight road. But when it is applied to a curved road, the result of detection is not very well compared to the result when applied to a straight road. The proposed method is now under improvement so that it may be applicable to a slightly curved road.

References

[1] S. Ezell: "Intelligent transportation systems", The Information Technology & Innovation Foundation, January 2010.

[2] U.S. Department of Transportation, Federal Highway Administration: "Safety applications of intelligent transportation systems in Europe and Japan", International Technology Scanning Program, January 2006.

[3] Ministry of Transport of Japan National Police Agency: "Occurrence of traffic accidents of 2013", 27th February 2014.

[4] J. C. Becker, A. Simon: "Sensor and navigation data fusion for an autonomous vehicle", Proc. of the IEEE Intelligent Vehicles Symposium, pp. 156-161, 2000.

[5] D. F. Llorca, V. Milanes, I. P. Alonso, M. Gavilan, I. G. Daza, J. Perez and M. A. Sotelo: "Autonomous pedestrian collision avoidance using a fuzzy steering controller", IEEE Transactions on Intelligent Transportation Systems, Vol. 12, No. 2, pp. 390-401, June 2011.

[6] R. Hayashi, J. Isogai, P. Raksincharoensak and M. Nagai: "Autonomous collision avoidance system by combined control of steering and braking using geometrically optimised vehicular trajectory", Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility, Vol. 50, Sup. 1, pp. 151-168, 2012.

[7] C. Demonceaux and D. Kachi-Akkouche: "Robust obstacle detection with monocular vision based on motion analysis", Intelligent Vehicles Symposium 2004 IEEE, pp. 527-532, 2004.

[8] W. Kruger, W. Enkelmann and S. Rossle: "Real-time estimation and tracking of optical flow vectors for obstacle detection", Proc. of the IEEE Intelligent Vehicles Symposium 1995, pp.304-309, 1995.

[9] G. Lefaix, E. Marchand and P. Bouthemy: "Motion-based obstacle detection and tracking for car driving assistance", Proc. of 16th Int. Conf. on Pattern Recognition, Vol.4, pp. 74-77, 2002.

[10] J. Lalonde, R. Laganiere and L. Martel: "Single-view obstacle detection for smart back-up camera systems", IEEE Computer Society Conference on CVPRW, pp. 1-8, 2012.

[11] A. Broggi, M. Bertozzi, A. Fascioli and M. Sechi: "Shape-based pedestrian detection", Proc. of the IEEE Intelligent Vehicles Symposium, pp.215-220, 2000.

[12] M. Bertozzi, A. Broggi, R. Chapuis, F. Chausse, A. Fascioli and A. Tibaldi:

“Shape-based pedestrian detection and localization”, Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems, pp. 328-333, 2003.

[13] M. Lutzeler and E. D. Dickmanns: “Road recognition with MarVEye”, Proc. of the IEEE Int. Conf. on Intelligent Vehicles, Vol.2, pp. 341-346, 1998.

[14] A. Kuehnle: “Symmetry-based vehicle location for AHS”, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 2902, pp. 19-27, 1997.

[15] D. M. Gavrila: “Pedestrian detection from a moving vehicle”, Proc. of European Conference on Computer Vision (ECCV), pp.37-49, 2000.

[16] M. Enzweiler and D. M. Gavrila: “Monocular pedestrian detection: Survey and experiments”, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 2179-2195, Vol. 31, Iss. 12, 2009.

[17] A. Bensrhair, M. Bertozzi, A. Broggi, A. Fascioli, S. Mousset and G. Toulminet: “Stereo vision-based feature extraction for vehicle detection”, Proc. of the IEEE Intelligent Vehicles Symposium 2002, Vol. 2, pp.465-470, 2002.

[18] A. Jazayeri, H. Cai, J. Zheng and M. Tuceryan: “Vehicle detection and tracking in car video based on motion model”, IEEE Transactions on Intelligent Transportation Systems, Vol. 12, Iss. 2, pp.583-595, 2011.

[19] M. Bertozzi and A. Broggi: “Gold: a parallel real-time stereo vision system for generic obstacle and lane detection”, IEEE Transactions on Image Processing, Vol.7, Iss. 1, pp. 62-81, 1998.

[20] R. Labayrade and D. Aubert: “Robust and fast stereovision based obstacles detection for driving safety assistance”, IEICE Transactions on information and systems, Vol. 87, No. 1, pp. 80-88, 2004.

[21] S. Qian, J. K. Tan, H. Kim, S. Ishikawa, T. Morie and T. Shinomiya: “Road Region Estimation and Obstacles Extraction Using a Monocular Camera”, International Journal of Innovative Computing, Information and Control, Vol. 9, No. 9, pp.3561-3572, September, 2013.

[22] A. Elgammal, R. Duraiswami, D. Harwood and L. S. Davis: “Background and foreground modeling using nonparametric kernel density estimation for visual surveillance”, Proc. of the IEEE, Vol. 90, Iss. 7, pp. 1151-1163, 2002.

[23] I. Haritaoglu, D. Harwood and L. S. Davis: “A fast background scene modeling and maintenance for outdoor surveillance”, Proc. of the IEEE Int. Conf. on Pattern Recognition, Vol. 4, pp. 179-183, 2000.

[24] C. Stauffer and W. E. L. Grimson: “Adaptive background mixture models for real-time tracking”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2, pp.246-252, 1999.

[25] T. Bouwmans: “Recent advanced statistical background modeling for foreground detection-a systematic survey”, Recent Patents on Computer Science, Vol. 4, No. 3, pp. 147-176, 2011.

[26] N. J. B. McFarlane and C. P. Schofield: “Segmentation and tracking of piglets in images”, Machine Vision and Applications, Vol. 8, Iss. 3, pp. 187-193, 1995.

[27] C. B. Wren, A. Azarbayejani, T. Darrell and A. P. Pentland: “Pfinder: real-time tracking of the human body”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, Iss. 7, pp. 780-785, 1997.

[28] A. Elgammal, D. Harwood and L. Davis: “Non-parametric model for background subtraction”, Proc. of European Conference on Computer Vision (ECCV), pp. 751-767, 2000.

[29] H. Zhang and D. Xu: “Fusing color and texture features for background model”, Proc. of Int. Conf. on Fuzzy Systems and Knowledge Discovery, pp. 887-893, 2006.

[30] F. E. Baf, T. Bouwmans and B. Vachon: “Fuzzy integral for moving object detection”, IEEE Int. Conf. on Fuzzy Systems, pp. 1729-1736, 2008.

[31] M. H. Sigari, N. Mozayani and H. R. Pourreza: “Fuzzy running average and fuzzy background subtraction: concepts and application”, International Journal of Computer Science and Network Security, Vol. 8, No. 2, pp. 138-143, 2008.

[32] K. Toyama, J. Krumm, B. Brumitt and B. Meyers: “Wallflower: Principles and practice of background maintenance”, Proc. of the 7th IEEE Int. Conf. on Computer Vision, Vol. 1, pp. 255-261, 1999.

[33] S. Messelodi, C. M. Modena, N. Segata and M. Zanin: “A Kalman filter based background updating algorithm robust to sharp illumination changes”, Proc. of Int. Conf. on Image Analysis and Processing, pp. 163-170, 2005.

[34] R. Chang, T. Gandhi and M. M. Trivedi: “Vision modules for a multi-sensory bridge monitoring approach”, Proc. of the 7th Int. IEEE Conf. on Intelligent Transportation Systems, pp. 971-976, 2004.

[35] M. Okutomi, K. Nakano, J. Maruyama, T. Hara: “Robust estimation of planar regions for visual navigation using sequential stereo images”, Proc. of IEEE Int. Conf. on Robotics and Automation, Vol. 4, pp. 3321-3327, 2002.

[36] K. Yamaguchi, A. Watanabe, T. Naito and Y. Ninomiya: “Road region estimation using a sequence of monocular images”, Int. Conf. on Pattern Recognition, pp. 1-4, 2008.

[37] C. Cortes and V. Vapnik: “Support-vector networks”, Machine Learning, Vol. 20, Iss. 3, pp. 273-297, 1995.

[38] R. M. Haralick: “Statistical and structural approaches to texture”, Proc. of the IEEE,

Vol. 67, Iss. 5, pp. 786-804, 1979.

[39] K. Yamaguchi, T. Kato and Y. Ninomiya: “Ego-motion estimation using a vehicle mounted monocular camera”, IEEJ Transactions on Electronics, Information and Systems, Vol. 129, Iss.12, pp. 2213-2221, 2009.

[40] C. Harris and M. Stephens: “A combined corner and edge detector”, Proc. of 4th Alvey Vision Conference, pp.147-151, 1988.

[41] B. D. Lucas and T. Kanade: “An iterative image registration technique with an application to stereo vision”, Proc. of the 7th Int. Joint Conf. on Artificial Intelligence, Vol. 2, pp.674-679, 1981.

[42] J. Y. Bouguet: “Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm”, Intel Corporation, Microprocessor Research Labs.

[43] M. A. Fischler and R. C. Bolles: “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography”, Communications of the ACM, Vol. 24, Iss. 6, pp.381-395, 1981.

[44] R. Hartley and A. Zisserman: “Multiple view geometry in computer vision”, Cambridge University Press, 2nd edition, 2004.

[45] R. I. Hartley: “In defense of the eight-point algorithm”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, Iss. 6, pp. 580-593, 1997.

[46] P. J. Rousseeuw: “Least median of squares regression”, Journal of the American Statistical Association, Vol. 79, pp. 871-880, 1984.

[47] S. Qian, J. K. Tan, H. Kim, S. Ishikawa, T. Morie: “Obstacles extraction from a video taken by a moving camera”, Proc. of Int. Conf. on Connected Vehicles and Expo, pp.268-273, 2012.

[48] S. Qian, J. K. Tan, H. Kim, S. Ishikawa, T. Morie and T. Shinomiya: “Videotaped Obstacle Extraction From a Moving Camera”, International Journal of Innovative Computing, Information and Control, Vol. 10, No. 2, pp.717-728, April, 2014.

[49] Z. Zhang: “A flexible new technique for camera calibration”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, Iss. 11, pp. 1330-1334, 2000.

[50] AASHTO: “A policy on geometric design of highways and streets”, 6th edition, 2011.