

This is a repository copy of *Line-Circle-Square (LCS):A multilayered geometric filter for edge-based detection*.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/175061/>

Version: Accepted Version

---

**Article:**

Tafrishi, Seyed Amir, Dai, Xiaotian [orcid.org/0000-0002-6669-5234](https://orcid.org/0000-0002-6669-5234) and Kandjani, Vahid Esmailzadeh (2021) Line-Circle-Square (LCS):A multilayered geometric filter for edge-based detection. *Robotics and Autonomous Systems*. 103732. ISSN: 0921-8890

<https://doi.org/10.1016/j.robot.2021.103732>

---

**Reuse**

This article is distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivs (CC BY-NC-ND) licence. This licence only allows you to download this work and share it with others as long as you credit the authors, but you can't change the article in any way or use it commercially. More information and the full terms of the licence here: <https://creativecommons.org/licenses/>

**Takedown**

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing [eprints@whiterose.ac.uk](mailto:eprints@whiterose.ac.uk) including the URL of the record and the reason for the withdrawal request.

# Line-Circle-Square (LCS): A Multilayered Geometric Filter for Edge-Based Detection

Seyed Amir Tafrishi<sup>a,\*</sup>, Xiaotian Dai<sup>b</sup> and Vahid Esmaeilzadeh Kandjani<sup>c</sup>

<sup>a</sup>Department of Mechanical Engineering, Kyushu University, Kyushu, Japan

<sup>b</sup>Department of Computer Science, University of York, York, UK.

<sup>c</sup>Department of Computer Engineering, University of Tabriz, Aras Free-Zone Campus, Iran

## ARTICLE INFO

### Keywords:

Edge detection

Motion field

Geometric filter

Vision

## ABSTRACT

This paper presents a state-of-the-art filter that reduces the complexity in object detection, tracking and mapping applications. Existing edge detection and tracking methods are proposed to create suitable autonomy for mobile robots, however, many of them face overconfidence and large computations at the entrance to scenarios with an immense number of landmarks. The method in this work, the Line-Circle-Square (LCS) filter, claims that mobile robots without a large database for object recognition and highly advanced prediction methods can deal with incoming objects that the camera captures in real-time. The proposed filter applies detection, tracking and learning to each defined expert to extract higher level information for judging scenes without over-calculation. The interactive learning feed between each expert increases the consistency of detected landmarks that works against overwhelming detected features in crowded scenes. Our experts are dependent on trust factors' covariance under the geometric definitions to ignore, emerge and compare detected landmarks. The experiment validates the effectiveness of the proposed filter in terms of detection precision and resource usage in both experimental and real-world scenarios.

## 1. Introduction


Object detection and tracking are among the challenging tasks in the field of robotics and computer vision [43]. One of the research challenges is to have satisfactory real-time performance and run-time efficiency. In recent years, a considerable large number of detection and tracking algorithms have been proposed [11, 20, 46], some of which can reduce computation complexity and/or increase the rate of object detection [15, 16]. These detectors use real-time data to process during their activation states. For autonomous mobile robots including autonomous driving cars, it is vital to assure fast responses during their motion. Additionally, due to size and energy constraint of embedded computing platforms, it is an urge to reduce resource usage and the number of sensors during active times [16].

To address these issues, a multi-layer filter is constructed in this paper, namely the *Line-Circle-Square (LCS) filter*, for object detection using information from a two-dimensional image sensor, e.g., a camera. The LCS filter is a multilayered geometric filter for edge-based detection. In our previous work, we did a preliminary study on the *Line-Circle filter* [36], the predecessor of the LCS filter. The algorithm of the LC filter was run offline with limited performance evaluation in the scene. Compared with the LC filter [36], we have optimized computation flow, restructured the algorithmic conditions and improved the code for real-time applications in all of the experts. The LCS filter is advanced with more inter-

active information transfer between each expert. Also, the proposed filter has an extra layer — the *Square expert*, in contrast to the Line-Circle (LC) filter. By utilizing inertial measurement units (IMUs) in parallel to the real-time captured images, an estimated scene is built that predicts possible objects that are passing or moving towards the robot. The proposed filter optimizes the data collection from existing edge detection methods [30, 31] or event-based camera [17] to ignore/concentrate on particular landmarks by using geometric and kinematics conditions. Besides these capabilities, with inspiration from certain estimators, e.g., the *Extended Kalman Filter (EKF)* [7, 10], errors are minimized to prevent any misclassification or integrated uncertainties of detected landmarks.

Compared with other related filters in the literature, the proposed LCS filter has the following unique advantages: (1) it reduces computation demand; (2) it has the ability to minimize the problem of overconfidence during detection; (3) real-time process for detecting abnormal behaviors at outside world such as partial detection of incoming objects toward the camera/moving vehicle; (4) primary detection with geometrical computation which creates a different level of information, i.e., low (edges) to high (layers) for mapping and localization; (5) the multi-layer nature makes it suitable for real-time processing with potential to be executed in parallel. In contrast to other learning methods, e.g., deep learning [39, 44], the LCS filter is an online learning algorithm, without the need for a large pool of training images and data. For implementation, this geometric filter only requires IMU and camera for working; Hence, it can work in battery-powered and memory-constrained systems including autonomous vehicles. Also, we have made the source code of LCS filter public for the benefit of the industrial and research commu-

\*Corresponding author

 amir@ce.mech.kyushu-u.ac.jp (S.A. Tafrishi);

xiaotian.dai@york.ac.uk (X. Dai); v.esmaeilzadeh@gmail.com (V. Esmaeilzadeh Kandjani)

ORCID(s): 0000-0001-9829-3144 (S.A. Tafrishi);  
0000-0002-6669-5234 (X. Dai)

nities<sup>1</sup>.

The proposed LCS filter works by collecting the detected edges (features) from the monocular camera. Next, Line expert groups and removes the detected features depending on the updates that come from Circle and Square experts. This operation decreases the computation size while preparing the features for higher accuracy of matching depending on the number of captured landmarks in different parts of the frame. Then, the Circle expert estimate features and do kinematic grouping for finding layers (grouped features with the same properties). Also, this expert predicts the features and layers through the motion field kinematics by using the angular rotation and velocity of the vehicle's IMU sensor. Square expert finally does the geometric and kinematic matches of the circles to construct the highest level of information as the layer of the objects in the scene. These kinematic classifications in the Circle and Square help us to find independently moving objects in the scene and have more consistent and accurate feature detection.

The paper is organized as follows: the related work is first given in Section 2. An overview of the LCS filter is introduced in Section 3. Then, the LCS filter is decomposed into three different subsections: line, circle, and square in Section 4. Evaluations in both simple and real-world scenarios are given in Section 5, followed by discussions and conclusions in Section 6.

## 2. Related Work

Object detection and tracking methods have been widely studied [11, 20, 46] to create accurate and fast analysis over the surrounding environment. In this section, we give a brief discussion for existing methods and highlight the difference in contrast to the LCS filter.

### 2.1. Object Detection

The primary goal of object detection is to localize the existing objects in the captured scene from input images. Note that objects can correspond to many combined class of objects or to a single one. To determine the features of surrounding, edge detection methods [31, 37] have been the most common studies. Also, there have been many alternative methods. As a classic approach for understanding the environment, the stereo-based model was firstly designed by Murray and Little for SLAM [27]. Other detection methods were studied related to the point analysis [34] and image contouring [1]. Next, a semi-dense monocular SLAM with the integration of color was applied to determine the surrounding objects at MIT [28]. An overall review was presented by Sun et al. about the vision-based localization methods for other vehicles on the road [35]. There is another survey that summarized different approaches based on edge detection for object/place recognition [40]. This study showed that image-to-image matching, based on the appearance in the surrounding, has better scaling. In particular, when the

environment is large, the map-to-map or image-to-map approaches face limitations. However, if a overwhelming number of landmarks is detected, the loop closer and effective recognition in these SLAM problems will be impractical for its applications [24, 26, 40].

### 2.2. Corner Detection and Applications

Rosten and Drummond proposed a machine learning corner detection method [31] as an efficient feature detection of the environment in real-time. Next, the monocular SLAM studies were taken place by applying different approaches for mapping based on feature detection [5, 7, 8]. Monocular SLAM with corner detection, as the most practical and advanced method, was analyzed to have better solutions for overconfidence and dealing with high computations related to the detected landmark. Lui and Drummond proposed a new system for constant time monocular SLAM that uses only 2D measurement and takes the image graph with sparse pairwise geometries. There were some improvements such as no global consistency and bundle adjustments but the system was based on a multi-camera perspective and it was also dependent on a reference image [19]. In the research field of edge detection, a fast event-based corner detection method is also proposed in [23]. This detection method is based on a novel event-based camera that only responds to local changes in brightness, and can detect edges directly from the camera. The other work is related to Kalman filter vision [18], which has low complexity but can still be overwhelmed by increasing the number of landmarks.

### 2.3. Detection and Tracking Filters

The Kalman filter reduction was applied for SLAM problems for using bundle adjustment by sparse matrix and double windowing method by Gamage and Drummond [10]. They tried to decrease the dimension of the covariance matrices of the camera and landmark position. Although it minimized some non-linearities that create inconsistency in extended Kalman filter (EKF) [7], it was not able to deal with overwhelming landmarks. A requirement for the solely image-based filter as a superior method rather than a classic Kalman filter has been remained unsatisfactory. Vision dependent applications require a filter that is able to do real-time computation reduction within diverse machine learning evaluation layers. Also, KF and EKF will reach overconfidence and might take the noise as an actually existed object.

On the other hand, the Correlation Filter trains a linear template that can discriminate between images and their translations, which is formulated in the Fourier domain that provides a fast solution for object tracking [13, 38]. A Kernelized Correlation Filter (KCF) is derived in [13] that has the exact same complexity as its linear counterpart for kernel regression. In [38], fully-convolutional Siamese networks are used which enables learning deep features that are tightly coupled to the Correlation Filter. However, the correlation filter only works with a small amount of data in the training process, and it could not work if the environment changes stochastically, for example, if pedestrians pass in front of

<sup>1</sup>GitHub Source: [https://github.com/SeyedAmirTafrishi/LCS\\_Filter/tree/LCS\\_RAS](https://github.com/SeyedAmirTafrishi/LCS_Filter/tree/LCS_RAS).

an autonomous car. In the literature of convolutionary neural network for machine vision, various techniques existed in complexity reduction for visual detection [4]. Some of the representative work include model compression [3, 45], quantization including binary neural networks (BNNs) [29], network pruning [14] and approximated computation [21]. However, these works are limited to the context of CNNs, and cannot be applied to more general cases.

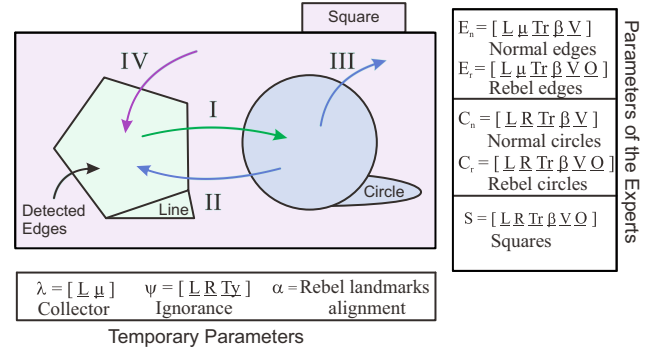
The Line-Circle-Square (LCS) introduced in this work uses geometrical information to filter overwhelmed data points which makes it computational efficient. The edges are filtered and only the edges that are important or new need to be processed in a way that significant computation can be saved. This has fundamental differences to other detection works as LCS focuses on reducing complexity and thus computational and memory cost. These make it an ideal front end of other object detection methods. The LCS filter extends the in-hand information level from low (edges) to high (layers). This gives more flexibility to address SLAM problems. A various of recognition algorithms can run faster because we are able to create kinematic labels for the chosen desired objects. Finally, the LCS filter is a multilayered approach that helps us to utilize parallel processing for each dedicated expert (line, circle and square).

### 3. Line-Circle-Square (LCS) Filter

A filter is proposed in this work of which the mainframe system has its evaluation from the surroundings in real-time. The algorithm detects objects in layers with using the past and current data from camera and IMU sensor which is obtained collectively. This filter runs without any pool of images or information except the initialization in the first frame. The novelty of this approach stands for flexible multi-level analysis on captured edge from the corner detection method [30, 31]. In other words, the LCS filter carries its feature analysis with filtering the data in each state of Line, Circle, Square experts. These experts update the data for future incoming frames besides correcting counterpart experts; Hence, each expert provides high-level geometrical detection from the environment. Thus, the collected edges from the camera are transferred between experts for detection, learning and tracking in each stage as Figure 1.

#### 3.1. Overview of the Filter

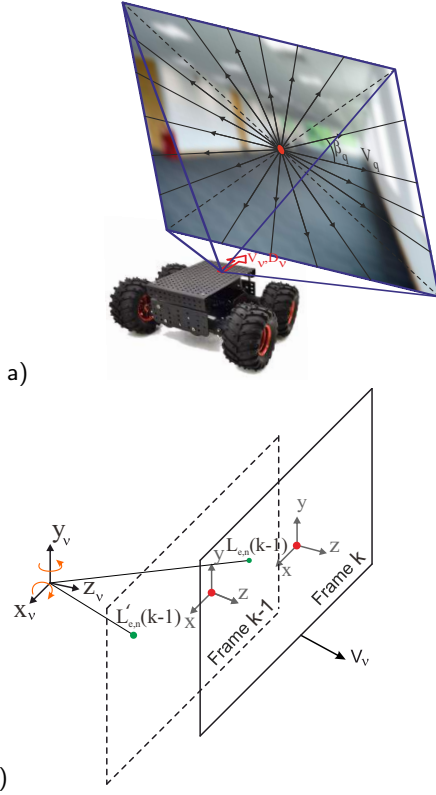
The parameters of the LCS filter are divided into two groups i.e., experts and temporary parameters [see Figure 1]. In the temporary parameters, the collector  $\lambda$  is responsible for grouping the edges with the location of the center  $L_\lambda$  and the detection radius size  $\mu_\lambda$ . The ignorance parameter  $\psi$  is for  $\lambda$  updates in which it takes the information from Circle and Square experts to create ignorance regions via  $L_\psi$ ,  $R_\psi$  and  $Ty_\psi$  where they are the location of the center, the radius of the region and a flag for geometry type, respectively. The ignorance parameter  $\psi$  removes unnecessary edges during the collection of edges by  $\lambda$ . This helps the filter to process faster and concentrate more on the essential locations.



**Figure 1:** An overview of the geometric LCS filter with data flow and filter parameters.

Also,  $\alpha$  is for detecting certain landmarks that don't follow the main direction of the vector field (along with the robot motion, see Figure 2 (a) as an example). Next, parameters of the experts carry information from previous frames for each of them. Each expert parameter has a collection series of information about the location  $L$ , size/radius of the region  $R/\mu$ , trust factor  $Tr$ , angle respect to origin  $\beta$ , the velocity of the corresponding landmark/layer  $V$ . Also,  $O = [O_x, O_y]$  stands for the origin of the landmark/layer of the expert after the first detection. Note that we define layer as a group of landmarks that have similar kinematics such as velocity, the direction of motion, or being in the same region on the frame. These layers not only can stand for a certain region of objects but also create an abstract presentation for better evaluation of the scene by experts.  $E_n$  and  $E_r$  are the parameters responsible for estimated edges to contain their properties. Besides these parameters in the Circle expert,  $C_n$  and  $C_r$  are used to assist the system to group the edges with relevant kinematic properties. Finally,  $S$  is the Square expert where it keeps the highest level of information about existing objects in the layer form. This final expert matches the objects based on defined geometric and kinematics conditions.

We have four transitions in this online learning algorithm. Each transition carries detection, learning, and tracking operations. Experts of the filter work in integration with the dynamic learning without sole reliance on a pyramidal supervisory [2]. At first, the camera detects the edges and saves it in  $\chi$ , and the Line Expert groups/removes them by  $(\lambda, \psi)$  with feedback from Circle and Square experts. At the transition I, collected edges  $\chi$  are transferred to the Circle expert. Circle expert estimates its layers of the landmarks. Next, there is a transition II that updates the ignorance regions  $\psi$ . Also, the updated Circles (layered landmarks) are transferred to Square expert in transition III. The Square expert geometrically matches the exiting objects in the environment with data from the estimated circles and its predicted data. Finally, the ignorance region  $\psi$  for the next edge collection (incoming frame) is updated by the Square expert.



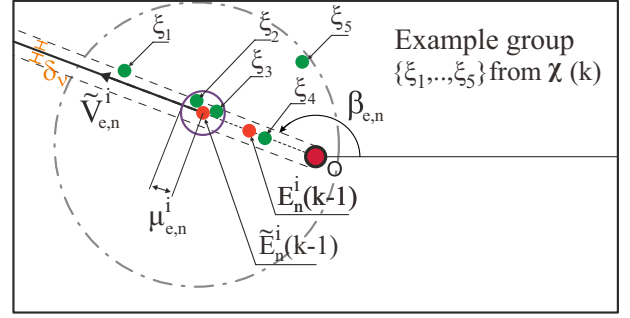
**Figure 2:** Kinematic analysis of locomotion with orientated vector field in a frame. a) The general view of created linear vector field due to robot motion along  $z$  axis with velocity  $V_v$ . b) Kinematic analysis of edges motion with respect to the frame and corresponding object.

### 3.2. Kinematics

The kinematics of a moving robot is sketched as Figure 2, where a holonomic robot with velocity of  $V_v$  travels  $D_v$  distance in the direction of  $z$  axis. In this work, we consider a motion field model rather than a linear rotational models that was used in our previous study [36]. The rotation is getting updated by the motion field kinematics where the translation is decoupled from the rotation [6]. The location of landmarks/layers  $\mathbf{L}(k-1) = [L_x, L_y]$  from  $k-1$  step including the origin points  $\mathbf{O}$  (rebel landmarks/layers have their own origin [36]) are updated by

$$\begin{aligned} L_x(k-1) &= L'_x(k-1) + O_{I,x} + f\omega_y + L'_y(k-1)\omega_z \\ &+ \frac{1}{f} \left[ L'_x(k-1)L'_y(k-1)\omega_x - L'^2_x(k-1)\omega_y \right] \\ L_y(k-1) &= L'_y(k-1) + O_{I,y} + f\omega_x + L'_x(k-1)\omega_z \\ &+ \frac{1}{f} \left[ L'_x(k-1)L'_y(k-1)\omega_y - L'^2_y(k-1)\omega_x \right] \end{aligned} \quad (1)$$

where  $\mathbf{L}'(k-1) = [L'_x(k-1), L'_y(k-1)]$ ,  $f$  and  $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]$  are the location before the rotational update with respect to the frame origin  $\mathbf{O}_I = [O_{I,x}, O_{I,y}]$ , camera's focal length and angular velocities of the moving vehicle that comes from the IMU. Additionally, the angular direction of



**Figure 3:** The edge's classifications with current information of  $\chi$ .

the features/layers  $\beta(k-1)$  is calculated based on  $\mathbf{L}(k-1)$  and  $\mathbf{O}(k-1)$ . It is important to note that we have assumed the rotation changes of the robot (it is independent of depth) with small values in each frame while the decomposed translation motion is estimated using our filter. One can obtain an accurate model for large rotations by utilizing a depth estimation of a Stereo camera [12] or applying advanced models of the motion field. As the camera moves in an aligned direction, a vector field can be expressed for the normally distributed locomotion of landmarks on the frame [see Figure 2 (b)]. This linear outward vector field will help us to develop a way to distinguish fixed or low-velocity objects that moves along the field from the abnormal landmarks, namely "rebel" landmarks/layers. These rebel landmarks can be an object that comes toward the camera or be an independently moving object that does not follow the vector field of the vehicle motion.

### 3.3. Trust Factor

The trust factor as an important part of the LCS filter is a variable that evaluates the reliability of the detected edge or layers. The trust variable  $Tr \in \mathcal{N}$  is the main factor in the learning part of the filter that is parameters with three ranks:

$Tr_s$ : The standard trust means the trust factor  $Tr$  with equal and greater than the value of  $Tr_s$  is landmarks/layers that are highly accurate with minor errors. Also, a high accuracy here means kinematic approximation matches with collected data. Thus, the trust factor of the parameters that reaches this value is possible to be tracked about their accuracy level respect to existing data from the landmark/layer.

$Tr_c$ : If the trust factor of edges/layers  $Tr$  hit the values less than the critical value  $Tr_c$ , they get deleted. A decrease in trust value to below  $Tr_c$  means these landmarks belong to objects that already passed or the wrong estimation happened in the frame.

$Tr_m$ : The maximum value in trust  $Tr$  that limits the parameters in the filter to a certain value for preventing over-confidence. Also, it helps in evaluations of the ignorance parameter  $\boldsymbol{\psi}$ .

Each of these ranked trusts lets the filter properly eliminate, re-coordinate, or combine the edges and layers of the Line, Circle and Square experts. Note that every newly created normal landmarks/layers i.e., normal edges  $\mathbf{E}_n$  and  $\mathbf{C}_n$

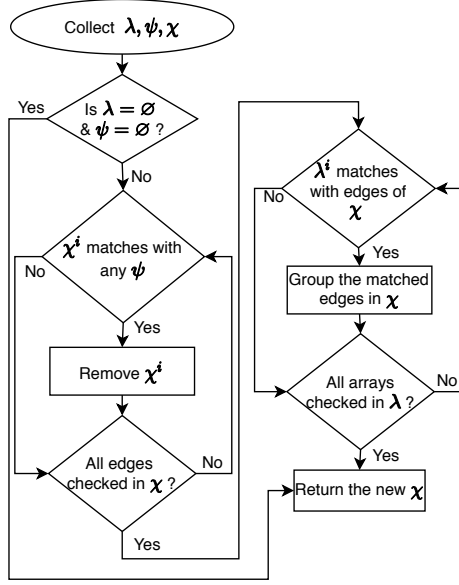


Figure 4: Flowchart of the Line expert.

circles, have their initial trust factor as  $Tr(0) = \frac{1}{2}[Tr_c + Tr_s]$ . A new candidate in the Square expert  $S$  has the initial  $Tr(0) = Tr_s$  trust factor. Also, because tracking the rebel landmarks/layers i.e.,  $E_r$  and  $C_r$ , are harder, they will have an initial trust factor larger than  $Tr(0) > Tr_s$  in contrast to normal ones.

## 4. Experts of the LCS Filter

### 4.1. Line Expert

The Line expert evaluates the landmarks that are collected from raw images of camera. This expert has two tasks: the first is removing unnecessary/repetitive information; The second is grouping the landmarks for faster evaluations in the next expert.

The flowchart in Figure 4 shows the computation of the Line expert. At first, the landmarks are reduced in  $\chi$  with using the feed of ignorance regions  $\psi$  since the high-level experts (the Circle and Square) grantee that the landmarks in  $\psi$  region are unnecessary to be updated for certain frame numbers. We will explain how the trust factor  $Tr$  will help the expert evaluation to predict certain regions with decreasing computations in the next steps. Next,  $\lambda$  is grouping the landmarks in  $\chi$  with given variables  $(L_\lambda, \mu_\lambda)$  in each frame. Also, newly appeared unmatched edges with  $\lambda$  are added as a single landmark in  $\chi$ . Grouping temporary parameter  $\lambda$  detects candidate edges in  $\chi$  by their distance  $\|L_\lambda - L_\chi\|$  from distance radius  $\mu_\lambda$  [36]. Next, the ignorance parameter  $\psi$  removes the edges via including the expressed regional constraints for different geometric flag  $Ty$  as

$$\begin{cases} \pi R_\psi^2, & (L_{\psi,x}, L_{\psi,y}) & Ty = 1 \\ R_{\psi,x} \cdot R_{\psi,y}, & (L_{\psi,x}, L_{\psi,y}) & Ty = 2 \end{cases}$$

where  $Ty = 1$  and  $Ty = 2$  flags are presenting the circular

and rectangle areas. Finally, the flowchart in Figure 4 returns new grouped  $\chi$ .

### 4.2. Circle Expert

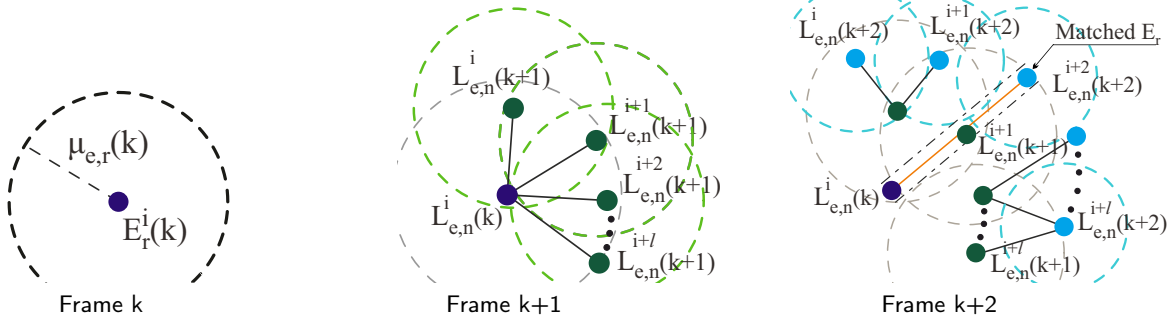
The Circle expert combines the landmarks with kinematic comparison at the grouped data. This stage transforms individual landmarks into grouped dynamic patterns. This expert increases the level of information to determine partially recognized layers in the scene by the next expert.

#### 4.2.1. Edge Classifications

The Circle expert classifies the detected landmarks  $\chi(k)$  in the direction of the moving robot that creates the vector field (normal edges) with respect to their predicted  $\tilde{E}_n(k-1)$  landmarks as shown in Figure 3. By considering the center of the captured image aligned with  $z$  axis of the robot motion as Figure 2, the detected landmarks are classified with five basic scenarios  $\{\xi_1, \dots, \xi_5\}$  depending on their location with respect to predicted landmarks  $\tilde{E}_n(k-1)$  on the image. Note that the properties of  $V_{e,n}$  and  $\beta_{e,n}$  (edge angle) are for the edges that are mostly passed objects as called “normal edges”. This is why the velocity is aligned with the center of frame  $O_I$ . The motion flow of normal edges vectors is linear since we assume robot holonomic motion is always aligned with  $z$  axis. However, the center for the rebel edges  $O_{e,r}$  are different depending on the coming origin from  $\alpha$ . The dashed double lines express the angular error span  $\delta_v$  that is dependent on the model error and vehicle velocity (obtained by IMU sensor). Also, the normal edge boundary radius  $\mu_{e,n}$  is getting updated proportional to the numbers of obtained landmarks and their kinematics.

Before giving the prediction definitions, the trust factor changes in each frame depends on the obtained evaluations through proposed classifications presented in Fig 3. This tells us how far the approximated landmarks are reliable for estimations by the captured edges. Not only these classifications let us determine new and low-accuracy landmarks but also filter can recognize rebel and normal moving edges from each other. Assuming the  $i$ -th landmark candidate in  $E_{e,n}^i$ , the estimations are classified in five cases:

- $\xi_1$ : This detected landmark in  $\chi$  is out of the  $\mu_{e,n}$  radius of the predicted edge  $\tilde{E}_n^i(k-1)$  but exists in the error span  $\delta_v$ . This detection will be added as an extra normal edge inside the included circle but the trust value  $Tr$  of estimated edge  $\hat{E}_n^i(k)$  will be reduced by 1.
- $\xi_2$ : This edge is successfully within the area of  $\mu_{e,n}$  and the error span  $\delta_v$ ; Hence, they are matched for estimation with approximated edge  $\tilde{E}_n^i(k-1)$ . The trust  $Tr$  of the corresponding edge is added with 1.
- $\xi_3$ : This edge is in the area of  $\mu_{e,n}$  but it fails from approximated location  $\tilde{L}_{e,n}(k-1)$  and velocity  $\tilde{V}_{e,n}(k-1)$  of the chosen edge  $E_{e,n}^i(k-1)$ . To include the approximation error for incoming estimation, the trust value  $Tr$  for estimated edge will be reduced by 1.
- $\xi_4$ : Despite the exclusion from approximated edge radius  $\mu_{e,n}$ , it is in the error span of  $\delta_v$ . Thus, we call it a



**Figure 6:** Rebel landmarks detection through within the three frames. The dashed gray lines are the previous step radius size, the dot-dashed lines are the error span variation's space  $\delta_v$ . The remaining colorized dashed circles are corresponding  $\mu_{e,r}$  radius size.

candidate for being a rebel landmark. Thus, the trust of estimated normal edge  $\mathbf{E}_{e,n}^i(k-1)$  will be decreased by 1 but the information will be carried to  $\alpha$  for the possible existence of rebel edges  $\mathbf{E}_r$ .

•  $\xi_5$ : This case fails all the predictions of velocity  $\tilde{V}_{e,n}(k-1)$  and location  $\tilde{\mathbf{L}}_{e,n}(k-1)$  as well as regional constraints  $\mu_{e,n}$  and  $\delta_v$ ; Hence, it is added as a rebel edge candidate at  $\alpha$  or new normal edge based on their failed categories. The trust value of estimated edge  $\mathbf{E}_{e,n}^i(k-1)$  will decrease by 1. In here, the condition for satisfying the error span  $\delta_v$  is defined by the similar formula in our previous study [36] respect to the image center  $\mathbf{O}_I$  and the detected landmark location  $\mathbf{L}_\chi$ .

The flowchart in Figure 5 shows how the Circle expert evaluates and detects the edges. At the initiation of  $\mathbf{E}_n$  matrix, landmarks velocities are equal to the vehicle velocity  $V_{e,n}(0) = V_v$ , the angle  $\beta_{e,n}(0)$  is defined depending on the edge location  $\mathbf{L}_{e,n}(0)$  respected to frame center  $\mathbf{O}_I$ . Also, the radius of detection  $\mu_{e,n}(0)$  is considered with a constant  $\mu_0$  value where it will be updated relative to the grouped landmarks  $\chi$  later on. We define the estimations of certain variables e.g.,  $\mathbf{L}$ ,  $\beta$  and  $R$ , in Figure 1 at  $k$ -th frame as follows

$$\hat{\mathbf{P}}(k) = \frac{(Tr(k-1) - Tr_c)\mathbf{P}' + \mathbf{P}(k)}{(Tr(k-1) - Tr_c) + 1} \quad (2)$$

where  $\mathbf{P}(k)$  and  $\mathbf{P}'$  are two arbitrary vectors. For example,  $i$ -th normal edge in  $\mathbf{E}_n$  at  $k$ -th frame is estimated for its location  $\mathbf{L}_{e,n}$ , velocity  $V_{e,n}$  and detection radius  $\mu_{e,n}$  and the angular direction  $\beta_{e,n}$  as follows

$$\begin{aligned} \hat{\mathbf{L}}_{e,n}^i(k) &= \frac{(Tr_{e,n}(k-1) - Tr_c)\tilde{\mathbf{L}}_{e,n}(k-1) + \mathbf{L}_\chi(k)}{(Tr_{e,n}(k-1) - Tr_c) + 1}, \\ \hat{V}_{e,n}^i(k) &= \left| V_v(k) \pm \frac{\|\tilde{\mathbf{L}}_{e,n}(k-1) - \mathbf{L}_\chi(k)\|}{t_f} \right|, \\ \hat{\mu}_{e,n}^i(k) &= \frac{1}{2} \left[ \frac{|V_v(k) - \tilde{V}_{e,n}(k-1)|}{Cor(\sum \chi)} + \mu_{e,n}(k-1) \right], \\ \hat{\beta}_{e,n}^i(k) &= \angle(\mathbf{L}_{e,n}^i(k), \mathbf{O}_I) \end{aligned} \quad (3)$$

where  $t_f$ ,  $Cor(\cdot)$ ,  $\sum \chi$  and  $\angle(\cdot, \cdot)$  are the sampling time, the correlation operation, group of captured edges in  $\chi$  matched

with the predicted landmark  $\tilde{\mathbf{E}}_{e,n}^i(k-1)$  and the angle of point  $\mathbf{L}_{e,n}^i(k)$  with respect to frame origin  $\mathbf{O}_I$ , in the given order. It is assumed that the robot recovers the angular error  $\delta_v$ , due to camera vibrations and IMU estimation errors, in each step. Thus, the motion angle of the edge  $\beta_{e,n}$  is kept the same. However,  $\beta_{e,n}$  should be predicted in early steps to find the corresponding edge classification [see Figure 3] where the predicted velocity is  $\tilde{V}_{e,n} = \frac{1}{2}[V_{e,n}(k-1) + V_v]$ .

For detecting the rebel edges  $\mathbf{E}_r$ , we propose a line tracking model that tries to find these edges within  $N$ -frame steps [see Figure 6]. By presenting the minimum  $N = 3$  frame, we include the landmarks into the matrix of the rebel landmarks alignment  $\alpha$  that satisfy  $\xi_4$  and  $\xi_5$  classifications within the  $N$  frame. We apply the detection with considering the underneath specifications:

- Due to unexpected motion in the rebellious landmarks for consecutive frames, tracking will be a non-linear motion with a certain deviation in each frame.
- The number of frames to evaluate the reliability of existing detection to rebel landmarks will be  $N = 3$ .
- The relation of connected landmarks is deleted after success/fail in every  $N$  frame analysis of  $\alpha$ .

The matrix of rebel landmarks alignment  $\alpha$  collects the locations of edges in the following way

$$\alpha = \begin{bmatrix} \mathbf{L}_{e,n}^i(k) & \mathbf{L}_{e,n}^i(k+1) & \mathbf{L}_{e,n}^i(k+2) \\ \mathbf{L}_{e,n}^{i+1}(k) & \mathbf{L}_{e,n}^{i+1}(k+1) & \mathbf{L}_{e,n}^{i+1}(k+2) \\ \mathbf{L}_{e,n}^{i+2}(k) & \mathbf{L}_{e,n}^{i+2}(k+1) & \mathbf{L}_{e,n}^{i+2}(k+2) \\ \vdots & \vdots & \vdots \\ \mathbf{L}_{e,n}^{i+l}(k) & \mathbf{L}_{e,n}^{i+l}(k+1) & \mathbf{L}_{e,n}^{i+l}(k+2) \end{bmatrix} \quad (4)$$

where, as an example,  $\mathbf{L}_{e,n}^i(k)$  is the location of  $i$ -th edge candidate at  $k$ -th frame. In this case, Figure 6 shows the successfully matched rebel edge that is obtained in the third row of the collection. Note that failed candidates get removed from matrix  $\alpha$  every frame. The initial values for constructed rebel edges at  $\mathbf{E}_r$  are defined by

$$\begin{aligned} \mathbf{O}_{e,r}(0) &= \mathbf{L}_{e,n}^i(k), \\ V_{e,r}(0) &= \frac{\|\mathbf{L}_{e,n}^{i+2}(k+2) - \mathbf{L}_{e,n}^{i+1}(k+1)\|}{t_f}, \end{aligned}$$

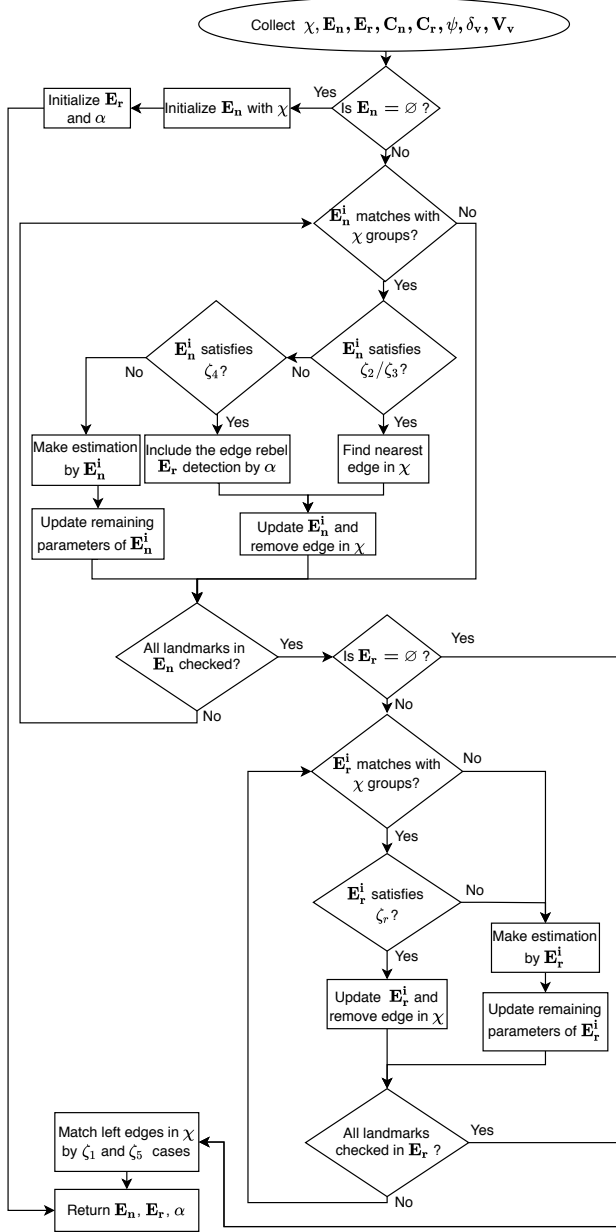


Figure 5: Flowchart of the Circle Expert for edge estimation.

$$\begin{aligned}
 \beta_{e,r}(0) &= \angle \left( \mathbf{L}_{e,n}^{i+2}(k+2), \mathbf{L}_{e,n}^i(k) \right), \\
 \mu_{e,r}(0) &= \angle \left( \mathbf{L}_{e,n}^{i+2}(k+2), \mathbf{L}_{e,n}^i(k) \right) \\
 &\quad - \angle \left( \mathbf{L}_{e,n}^{i+1}(k+1), \mathbf{L}_{e,n}^i(k) \right)
 \end{aligned} \quad (5)$$

where  $\mu_{e,r}$  is defined as the deviation angle here which is different from the detection radius of the normal edge  $\mu_{e,n}$ . These rebel landmarks do not follow the normal vector field of the vehicle motion; Hence, their kinematics are defined by detected location information in  $\alpha$ . For the landmarks that already exist in the rebel edge matrix  $\mathbf{E}_r$ , their location  $\mathbf{L}_{e,r}$  and  $\mathbf{V}_{e,r}$  velocity are estimated with same equations in equations (2) and (3), respectively. However, the deviation angle of  $\mu_{e,r}$  is updated by  $\hat{\mu}_{e,r}(k) = |\hat{\beta}_{e,r}(k) - \hat{\beta}_{e,r}(k-1)|$ , where

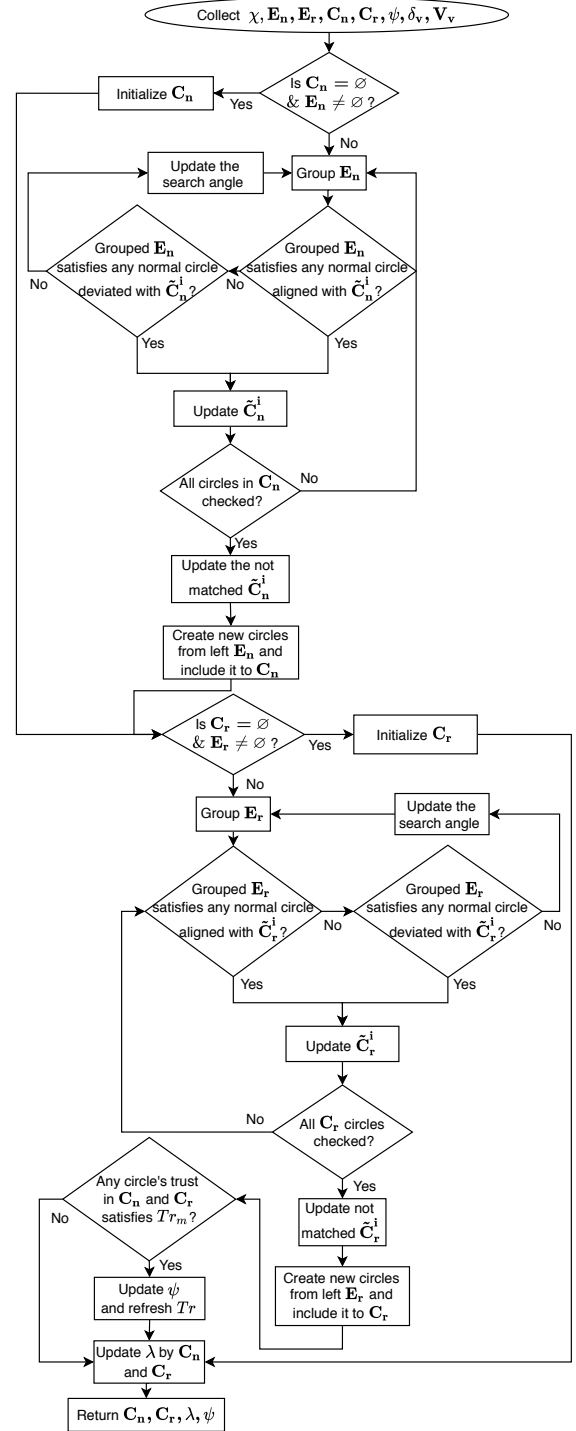


Figure 7: Flowchart of the Circle expert for circle estimation part.

the predicted angle  $\hat{\beta}_{e,r}(k)$  is calculated from the last detected edge respected to rebel edge's origin  $\mathbf{O}_{e,r}$  [36]. These kinematic formulations make the detection and learning to be smooth and have a similar appliance of negative and positive classified examples [15] in here. In contrast, gradual learning is used with a disappearing pattern via the trust factor definitions. Note that  $\zeta_r$  classification for the rebel edges

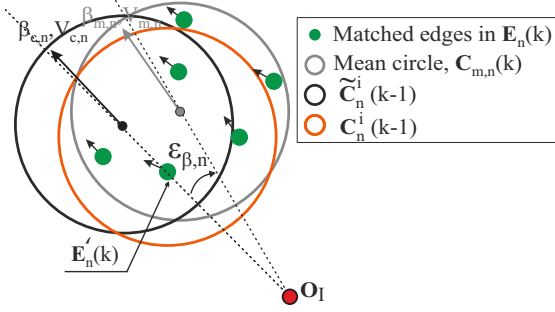


Figure 8: Circle matching in normal landmarks.

is similar to  $\zeta_2$  while the detected edge is inside both  $\delta_v$  and  $\mu_{e,r}$  constraints.

The computation flow of the given flowchart in Figure 7 is to prepare the normal and rebel circles as the layers of the detected objects in the frame. This designed expert is divided into three main parts. The first part was about detecting and comparing the collected group edges  $\chi$  with  $E_n$  as explained. Next, after elimination, remaining unmatched landmarks are carried to the second part for the rebel edges  $E_r$  study which was based on flowchart in Figure 5. Then,  $E_n$  and  $E_r$  matrices are compared under certain kinematic comparisons to develop the latest circles as layers [as shown in Figure 7].

#### 4.2.2. Normal Edge Circling

In the normal circles, collected landmarks are following the direction of the vector field [see Figure 2] and they are categorized depending on their directions of the motion, velocities and, locations on the image.

Figure 8 shows how the edges with the alike kinematics are matched in the expert. In this matching, the estimated circle  $\hat{C}_n^i(k)$  in the  $k$ -th frame is determined by previously predicted normal circle  $\tilde{C}_n^i(k-1)$  of the frame  $k-1$  and the grouped new circle  $C_{n,m}(k)$  with estimated edges in the  $k$ -th frame. At first, the chosen candidate normal edge  $E'_n$  is matched with other landmarks depending on their angle  $\beta'_{e,n}$  and velocity  $V'_{e,n}$  to construct  $C_{n,m}(k)$  as follows

$$\begin{cases} (\beta'_{e,n} - \epsilon_{\beta,n}) < \beta_{e,n}^i < (\beta'_{e,n} + \epsilon_{\beta,n}) \\ |V'_{e,n}| \leq |V_{e,n}^i| + \epsilon_{v,n} V_v \end{cases} \quad 1 < i \leq l \quad (6)$$

where  $\epsilon_{\beta,n}$  and  $\epsilon_{v,n}$  are the accuracy constants of angle and velocity for the normal circle. Next, to compare the predicted circle  $\tilde{C}_n^i(k-1)$  with the mean circle of the grouped edges  $C_{m,n}$ , the following condition is used:

$$[(\beta_{m,n} - \epsilon_{\beta,n}) < \beta_{c,n}^i < (\beta_{m,n} + \epsilon_{\beta,n})] \& [V_{m,n} \leq \epsilon_v V_{c,n}^i] \quad (7)$$

where

$$\beta_{m,n} = \frac{1}{l} \sum_{q=1}^l \beta_{e,n}^q, \quad V_{m,n} = \frac{1}{l} \sum_{q=1}^l |V_{e,n}^q|$$

Together with matching conditions in (7), there is a weighting function that finds out the percentage of the overlap. The percentage of overlap finds the ratio of the located edges  $q \in [1, l]$  in the mean circle of the grouped edges  $C_{m,n}$  are in the predicted circle  $\tilde{C}_n^i(k-1)$  as

$$Per \left[ \left\| \mathbf{L}_{e,n}^q(k) - \tilde{\mathbf{L}}_{c,n}(k-1) \right\| < R_{c,n}^q(k-1) \right] < \rho_c \quad (8)$$

where  $Per[\cdot]$  and  $\rho_c$  are the percentage operation that calculates the ratio of edges in  $C_{m,n}$  which are located inside of  $\tilde{C}_n^i$  and the minimum overlap percentage, respectively.

The estimated circles are updated with certain similar functions as landmarks. For example, the location  $\hat{\mathbf{L}}_{c,n}(k)$ , the radius  $\hat{R}_{c,n}(k)$  and direction angle  $\hat{\beta}_{c,n}(k)$  are estimated by equation (2). While the predicted circle's angle  $\tilde{\beta}_{c,n}^i(k-1)$  is aligned with averaged group of edges in  $C_{m,n}$  within the accuracy constant  $\epsilon_{\beta,n}$ , the estimated circle  $\hat{C}_n^i(k)$  is upgraded with +1 trust factor. If the angle has deviation with respect to the estimated one despite major inclusion of edges in the circle, the trust factor is updated with -1.

#### 4.2.3. Rebel Edge Circling

Certain objects in the scene will have inconsistent motion in the frame. These objects normally will be the ones that have potential to come toward the robot or are moving objects in an arbitrary directions (such as side passing cars). Therefore, detected rebellious edges will have the most importance when it comes to SLAM or object avoidance applications which is one of the strength of the proposed filter. The rebel landmarks in the worst-case scenario can be appeared within the normal edge circles [see Figure 9]. To detect the rebel edges, previously available similar edges are considered with specific dedication to the  $\beta_{c,r}$  and  $V_{c,r}$  similar to condition (6) where it has accuracy constants of the angle  $\epsilon_{\beta,r}$  and  $\epsilon_{v,r}$  velocity in grouping rebel edges. It must be said that these edges are the hardest ones since they are not following vectors field flow ( $\epsilon_{v,r} > \epsilon_{v,n}$  and  $\epsilon_{\beta,r} > \epsilon_{\beta,n}$ ) when they are having a nonlinear displacement (unpredictable motion) in frame.

Based on the characteristics of the rebel edges, the velocity condition is largely dependent on rebel edges estimated velocities and the mean circle of grouped rebel edges. Thus, a condition is developed for detecting the right grouped rebel edges  $C_{m,r}(k)$  with the predicted  $i$ -th rebel circle  $\tilde{C}_r^i(k-1)$  as follows

$$\begin{aligned} & [(\beta_{m,r} - \epsilon_{\beta,r}) < \beta_{c,r}^i < (\beta_{m,r} + \epsilon_{\beta,r})] \\ & \& [V_{m,r} \leq (V_{c,r}^i + \epsilon_{v,r} V_v)] \end{aligned} \quad (9)$$

where

$$\beta_{m,r} = \frac{1}{l} \sum_{q=1}^l \beta_{e,r}^q + \mu_{e,r}^q, \quad V_{m,r} = \frac{1}{l} \sum_{q=1}^l |V_{e,r}^q|$$

The same overlap condition (8) for locating rebel circles are taken place as normal circles. The location  $\hat{\mathbf{L}}_{c,r}(k)$  and radius  $\hat{R}_{c,r}(k)$  and trust factor  $Tr_{c,r}$  are updated similar to the

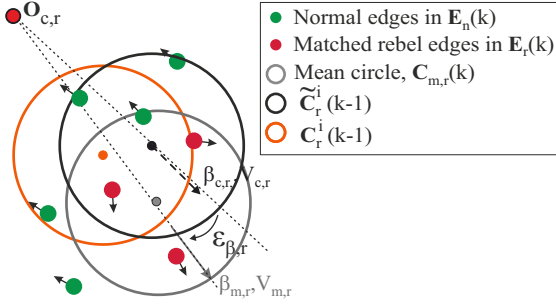


Figure 9: Circle matching in the rebel landmarks.

normal circle except in the estimation of angle  $\hat{\beta}_{c,r}(k)$ , we have

$$\hat{\beta}_{c,r} = \frac{1}{l} \sum [\beta_{e,r} + \mu_{e,r}]$$

Note that overall filter is constructed with the same analyzed flow as Figure 7.

### 4.3. Square Expert

The Square expert geometrically combines circles from the Circle expert to find physical layers. These layers stand for the objects' regions or the object itself. Note that our filter does not contain any recognition source for different objects; Hence, the objects can be packed in layers due to equal distances. This grouping happens by kinematic properties that are captured by the inertial sensors mounted on the moving vehicle. This grants fast evaluation of the incoming scene to group each region with corresponding geometry where it can be either passing or incoming object toward the camera. Note that we here mention each candidate as a "Square" for simplicity but the detected layers can geometrically be any geometry of the rectangle form.

The flow chart in Figure 10 shows the workflow of the Square expert. The Square expert mainly consists of two primary parts: in the first part, it collects the circles with their certain geometries and velocity properties. In the second part, this expert matches the grouped circles with the previous step's predicted square. In this expert, after combining the rebel and normal circles, the algorithm tries to find the furthest couple of circles that can correspond to a complete object or part of it. Then, it will cancel out the couple and update the distance if there exist other objects that do not relate to the same layer (distance) of it. The grouped circles (minor circles) with included couples are then transformed into a rectangle geometry  $S_m$ . Then, the mean square candidate  $S_m$  is estimated with the predicted  $i$ -th square  $\tilde{S}^i(k-1)$  in  $k-1$  frame. The process continues till all the circles are looped in the frame  $k$ .

#### 4.3.1. Geometric Layering

In this procedure, a circle is firstly selected as  $C_a$  from  $\{C_n, C_r\}$ , and then the algorithm searches for two cases of geometric matches. The first case (Case 1) is when two circles are matched with  $\pm 90^\circ$  angle from each other [see Figure 10 as an example]. Case 2 matches circles that are ap-

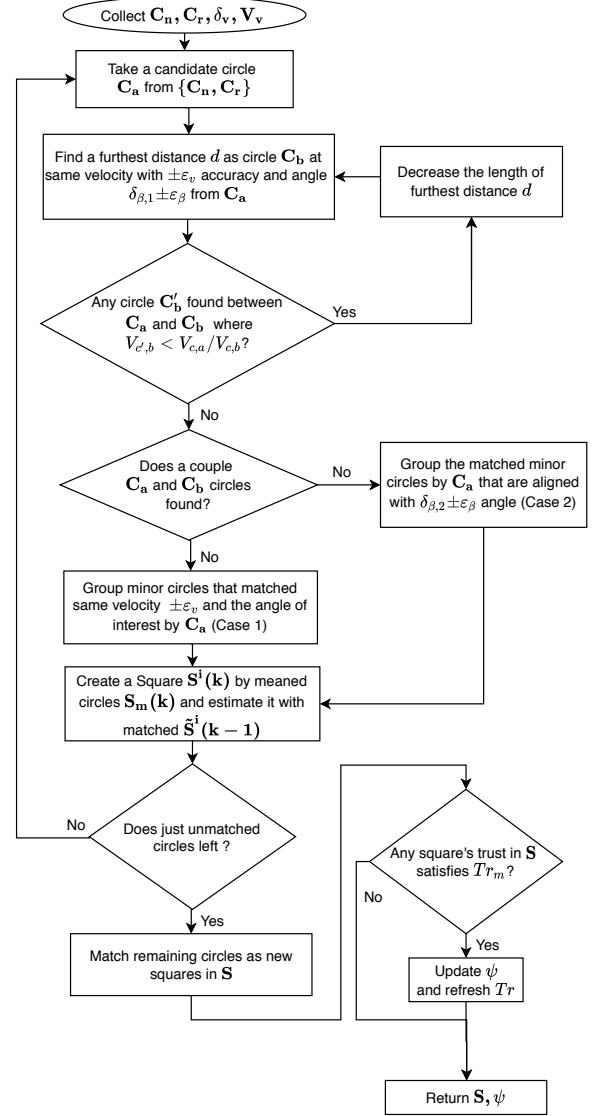


Figure 10: Flowchart of the Square Expert.

proximately aligned in the same direction and with the same velocity. Note that, here, we choose simple extending square corners and aligned groups matching as layers in the frame. One can make more complex structures, for example, hexagon or smooth closed convex graphs which we think is a separate study in itself.

In case 1, the selected initial circle  $C_a$  is matched with its couple circle  $C_b$  with the following velocity  $(V_{c,a}, V_{c,b})$ , angle  $(\beta_{c,a}, \beta_{c,b})$  and distance  $(d)$  conditions:

$$\begin{aligned} & [V_{c,b} - \epsilon_v \leq V_{c,a} \leq V_{c,b} + \epsilon_v] \\ & \& [\beta_{c,b} + \delta_{\beta,1} - \epsilon_\beta \leq \beta_{c,a} \leq \beta_{c,b} + \delta_{\beta,1} + \epsilon_\beta] \& [d < d_t] \\ & || [V_{c,b} - \epsilon_v \leq V_{c,a} \leq V_{c,b} + \epsilon_v] \\ & \& [\beta_{c,b} - \delta_{\beta,1} - \epsilon_\beta \leq \beta_{c,a} \leq \beta_{c,b} - \delta_{\beta,1} + \epsilon_\beta] \& [d < d_t] \end{aligned} \quad (10)$$

where  $\delta_{\beta,1}$ ,  $d$  and  $d_t$  are the angle difference for the candidate corner circle  $C_b$  in the first case, the distance between couple of circles and the maximum temporary distance between circles  $\{C_a, C_b\}$ . Initial term in condition (10) matches the similar velocities between the circles. Next, the angle of the counterpart circle  $C_b$  should be satisfied which we consider as  $\pm 90^\circ$  here. As mentioned before, more complicated geometries can be assumed but this will complicate the iterations due to the grouping stage for other circles. Next, the distance of the two circles  $d = \|\mathbf{L}_{c,a} - \mathbf{L}_{c,b}\|$  has to be always smaller than the maximum temporary distance  $d_t$  where  $\mathbf{L}_{c,a}$  and  $\mathbf{L}_{c,b}$  are the locations of circle candidates  $C_a$  and  $C_b$ , respectively.

A condition has to be constructed to update the temporary distance of  $d_t$ . Also, the algorithm should break a couple of circles and let the expert search for a smaller region to find the right couple of circles as in Figure 11. The main reason is the existence of the far objects as  $C'_b$  that there is located between/inside the objects (they can be in same distance), here are the matched  $\{C_a, C_b\}$ . Thus, if the filter combines the two aligned objects, it will create the misinterpretation in object detection and maybe the furthest object would be ignored. Based on the obtained insight, the condition for decreasing the maximum temporary distance  $d_t$  is determined

$$[V_{c',b} \leq V_{c,a}] \ \& \ [|\beta_{tr}| \leq |\beta_{tl}| \leq |\beta_m|] \ \& \ [d < d_t] \quad (11)$$

where  $V_{c',b}$  is the candidate circle velocity and

$$\begin{aligned} \beta_{tl} &= \sin^{-1} \left( \frac{\|\mathbf{L}_{tl} - \mathbf{L}_{c,b}\|}{\|\mathbf{L}_{c,a} - \mathbf{L}_{c,b}\|} \right), \\ \beta_{tr} &= \sin^{-1} \left( \frac{\|\mathbf{L}_{tr} - \mathbf{L}_{c,b}\|}{\|\mathbf{L}_{c,a} - \mathbf{L}_{c,b}\|} \right), \\ \beta_m &= \cos^{-1} \left( \frac{d_{ab}^2 + d_{ab'}^2 - d_{bb'}^2}{2d_{ab}d_{ab'}} \right) \end{aligned}$$

with

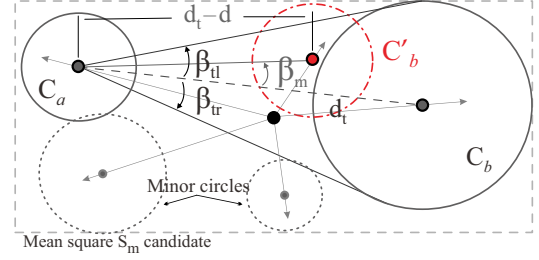
$$\begin{aligned} d_{ab} &= \|\mathbf{L}_{c,a} - \mathbf{L}_{c,b}\|, \ d_{ab'} = \|\mathbf{L}_{c,a} - \mathbf{L}_{c',b}\|, \\ d_{bb'} &= \|\mathbf{L}_{c,b} - \mathbf{L}_{c',b}\| \end{aligned}$$

where  $\mathbf{L}_{c',b}$ ,  $\mathbf{L}_{tl}$  and  $\mathbf{L}_{tr}$  are the locations of the potential circle  $C_{b'}$ , the left and right tangent points on  $C_b$  circle. If the condition (11) is satisfied, the temporary maximum distance  $d_t$  is updated by  $d_t = d_t - d_{bb'}$ .

As searching loop continues for matching the circles, the algorithm checks for case 2 matches. These matches are circles that are aligned with approximately similar velocities and direction of motion where they satisfy following condition:

$$\begin{aligned} [V_{c,b} - \varepsilon_v \leq V_{c,a} \leq V_{c,b} + \varepsilon_v] \\ \& \ [(\beta_{c,b} - \delta_{\beta,2} - \varepsilon_\beta) \leq \beta_{c,a} \leq (\beta_{c,b} + \delta_{\beta,2} + \varepsilon_\beta)] \end{aligned} \quad (12)$$

where  $\delta_{\beta,2}$  is the angle difference for satisfying case two.



**Figure 11:** An example case showing how the maximum temporary distance  $d_t$  decreases during the collection of a circle couple  $\{C_a, C_b\}$ .

#### 4.3.2. Mean Square

After a successful match for the couple of circles  $\{C_a, C_b\}$ , the algorithm includes the matching other minor circles into  $C_b$  [see Figure 11 as an example]. The constructed mean square  $S_m$  from matched circles with included minor circles has following parameters

$$\mathbf{O}_m = \frac{1}{2} \left( \mathbf{O}_{c,a} + \frac{1}{l} \sum_{i=1}^l \mathbf{O}_{c,b}^i \right), \quad (13)$$

$$\beta_m = \angle(\mathbf{L}_{c',b}, \mathbf{O}_{s,m}), \quad (14)$$

$$V_m = \frac{1}{2} \left( V_{c,a} + \frac{1}{l} \sum_{i=1}^l V_{c,b}^i \right), \quad (15)$$

$$\mathbf{R}_m = (\mathbf{L}_m^+ - \mathbf{L}_m^-)/2, \ \mathbf{L}_m = (\mathbf{L}_m^+ + \mathbf{L}_m^-)/2 \quad (16)$$

where

$$\mathbf{L}_m^+ = \max\{\mathbf{L}_{c,a}, \mathbf{L}_{c,b}\}, \ \mathbf{L}_m^- = \min\{\mathbf{L}_{c,a}, \mathbf{L}_{c,b}\}$$

Additionally, the algorithm finds the matches of the circles for a minor circle and the circle couple  $\{C_a, C_b\}$  is

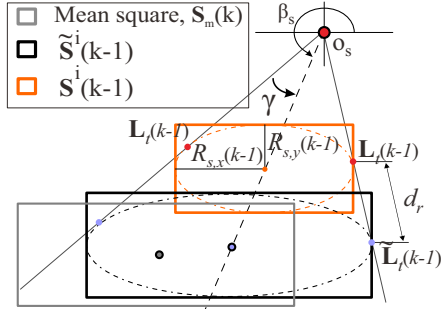
$$\begin{aligned} [(V_m - \varepsilon_v) \leq V_{c',b} \leq (V_m + \varepsilon_v)] \ \& \\ [(\beta_m - \varepsilon_\beta) \leq \beta_{c',b} \leq (\beta_m + \varepsilon_\beta)] \end{aligned} \quad (17)$$

By succeeding the condition (17), if minor circles are located inside the constructed square  $\mathbf{L}_{c'} \in \{\mathbf{L}_m^+, \mathbf{L}_m^-\}$  by  $\{C_a, C_b\}$ , they are added to the  $C_b$ . Note that our  $C_a$  is a single candidate but  $C_b$  might have more than one circle. This reduces the computation of the search in the algorithm. Also, if maximum temporary distance  $d_t$  is required to be reduced in any case, the excluded circles (they are in  $d > d_t$  region) are removed from  $C_b$ .

#### 4.3.3. Square Estimation

After collecting circles that matched with proposed cases, the constructed temporary square  $S_m$  by (16) has to be estimated with previous step's prediction that exists at  $S$ ; Hence, we predict  $k$ -th step's square from  $k-1$  step and compare it with constructed temporary one.

To calculate the predicted squares  $\tilde{S}$ , the tangent lines are firstly found on the trapped ellipse in each rectangle (called as "square")  $S^i(k-1)$  [see Figure 12] that are derived by



**Figure 12:** Matching squares with geometric parameters.

solving ellipse and tangent lines' equations for obtaining the couple of points  $(L_{t,x}, L_{t,y})$ ,

$$(L_{t,y} - L_{s,y})^2/R_{s,y}^2 + (L_{t,x} - L_{s,x})^2/R_{s,x}^2 = 1 \quad (18)$$

$$\begin{aligned}
& R_{s,x}^2(O_{s,y} - L_{t,y})(L_{t,y} - L_{s,y}) \\
& + R_{s,y}^2(O_{s,x} - L_{t,x})(L_{t,x} - L_{s,x}) = 0
\end{aligned} \tag{19}$$

Note that these equations have two solutions, so  $(L_{t,x}, L_{t,y})$  is chosen arbitrarily from the couple for simplicity. Equation (19) is derived by differentiation equation (18) as

$$dL_y(L_{t,y} - L_{s,y})/R_{s,y}^2 + dL_x(L_{t,x} - L_{s,x})/R_{s,x}^2 = 0 \quad (20)$$

Next, the tangent line slope relation between  $(O_{s,x}, O_{s,y})$  and  $(L_{t,x}, L_{t,y})$  points is defined by

$$dL_y/dL_x = (O_{s,y} - L_{t,y})/(L_{t,x} - L_{s,x}), \quad (21)$$

which substituting (21) into (20) results in the equation (19).

Before determining the predicted location of the tangent points  $\tilde{\mathbf{I}}_t(k-1)$  at frame  $k$ , the traveled distance  $d_r$  is found through trigonometric relations as Figure 12:

$$d_r = \begin{cases} (||\mathbf{O}_s - \mathbf{L}_s|| + \Delta r) \cos \gamma - ||\mathbf{O}_s - \mathbf{L}_t||, & \frac{||\mathbf{O}_s - \mathbf{L}_s||}{||\mathbf{O}_s - \mathbf{L}_t||} > 0 \\ \frac{||\mathbf{O}_s - \mathbf{L}_s|| + \Delta r}{\cos \gamma} - ||\mathbf{O}_s - \mathbf{L}_t|| \cos \gamma, & \frac{||\mathbf{O}_s - \mathbf{L}_s||}{||\mathbf{O}_s - \mathbf{L}_t||} \leq 0 \end{cases} \quad (22)$$

where  $\Delta r = ||L_s(k-1) - \tilde{L}_s(k-1)||$  is the distance difference between the location of square in the frame  $k-1$  and the predicted location in  $k$ -th step that is calculated by solving the vehicle kinematics, and  $\gamma$  is

$$\gamma = \cos^{-1} \begin{cases} \frac{\|\mathbf{O}_s - \mathbf{L}_t\|}{\|\mathbf{O}_s - \mathbf{L}_s\|}, & \|\mathbf{O}_s - \mathbf{L}_s\| > \|\mathbf{O}_s - \mathbf{L}_t\| \\ \frac{\|\mathbf{O}_s - \mathbf{L}_s\|}{\|\mathbf{O}_s - \mathbf{L}_t\|}, & \|\mathbf{O}_s - \mathbf{L}_s\| \leq \|\mathbf{O}_s - \mathbf{L}_t\| \end{cases} \quad (23)$$

Note that all the parameters in these formulas are related to  $k - 1$  frame such as  $\mathbf{L}_s = \mathbf{L}_s(k - 1)$  and  $\mathbf{L}_t = \mathbf{L}_t(k - 1)$ . We have wrote them this way for simplicity. By substituting equation (23) to equation (22), the traveled distance  $d_r$  is simplified to

$$d_r = \begin{cases} (||\mathbf{O}_s - \mathbf{L}_s|| + \Delta r) \frac{||\mathbf{O}_s - \mathbf{L}_t||}{||\mathbf{O}_s - \mathbf{L}_s||} - ||\mathbf{O}_s - \mathbf{L}_t||, & \frac{||\mathbf{O}_s - \mathbf{L}_s||}{||\mathbf{O}_s - \mathbf{L}_t||} > 0 \\ (||\mathbf{O}_s - \mathbf{L}_s|| + \Delta r) \frac{||\mathbf{O}_s - \mathbf{L}_s||}{||\mathbf{O}_s - \mathbf{L}_t||} - ||\mathbf{O}_s - \mathbf{L}_s||, & \frac{||\mathbf{O}_s - \mathbf{L}_s||}{||\mathbf{O}_s - \mathbf{L}_t||} \leq 0 \end{cases} \quad (24)$$

Next, the slope for these couple of points are determined by

$$m_t = (L_{t,x}(k-1) - O_{s,x}) / (L_{t,y}(k-1) - O_{s,y}) \quad (25)$$

Now, the predicted location of the tangent couple of points  $\tilde{\mathbf{L}}_t$  at the frame  $k$  are

$$\begin{aligned}\tilde{L}_{t,y}(k-1) &= L_{t,y}(k-1) + \left( \frac{d_r^2}{m_t^2 + 1} \right)^{\frac{1}{2}}, \\ \tilde{L}_{t,x}(k-1) &= L_{t,x}(k-1) + m_t[L_{t,y}(k) - \tilde{L}_{t,y}(k-1)]\end{aligned}\quad (26)$$

The same equation (26) is used to predict the location of the considered square candidate  $\mathbf{L}_s(k)$  in  $k$ -th frame. Finally, the new minor and major radii of the ellipse as the sides of the geometric rectangle (the square) are found by solving following algebraic equations:

$$\begin{aligned}
\tilde{R}_{s,y}(k-1) &= \left| (O_{s,x} - \tilde{L}_{t,x}(k-1))(\tilde{L}_{t,y}(k-1) - \tilde{L}_{s,y}(k-1))^2 \right. \\
&\quad - (O_{s,y} - \tilde{L}_{t,y}(k-1))(\tilde{L}_{t,x}(k-1) - \tilde{L}_{s,x}(k-1))(\tilde{L}_{t,y}(k-1) \\
&\quad \left. - \tilde{L}_{s,y}(k-1)) / [O_{s,x} - \tilde{L}_{t,x}(k-1)] \right|^{\frac{1}{2}} \\
\tilde{R}_{s,x}(k-1) &= \left| \tilde{R}_{s,y}^2(k-1)(O_{s,x} - \tilde{L}_{t,x}(k-1))(\tilde{L}_{t,x}(k-1) \right. \\
&\quad \left. - \tilde{L}_{s,x}(k-1)) / [(O_{s,y} - \tilde{L}_{t,y}(k-1))(\tilde{L}_{t,y}(k-1) - \tilde{L}_{s,y}(k-1))] \right|^{\frac{1}{2}}
\end{aligned} \tag{27}$$

The velocity of the predicted  $i$ -th square  $\tilde{\mathbf{S}}^i(k-1)$  with the known  $\tilde{\mathbf{L}}_s^i(k-1)$  and  $\tilde{\mathbf{R}}_s^i(k-1)$  is  $\tilde{\mathbf{V}}_s^i(k-1) = \mathbf{V}_s^i(k-1)$ . Note that velocities of the Circle and Square experts from previous frame ( $k-1$ ) are always updated with new vehicle velocity  $\mathbf{V}_v$  if the mobile robot is not moving in constant speed. Now, the predicted square  $\tilde{\mathbf{S}}^i(k-1)$  has to be compared with constructed square  $\mathbf{S}_m(k)$  from the collected circles

$$\begin{aligned} & [\tilde{V}_s^i(k-1) - \varepsilon_{v,s} \leq V_{s,m}(k) \leq \tilde{V}_s^i(k-1) + \varepsilon_{v,s}] \\ & \& [\tilde{\beta}_s^i(k-1) - \varepsilon_{\beta,s}^i \leq \beta_{s,m}(k) \leq \tilde{\beta}_s^i(k-1) + \varepsilon_{\beta,s}] \quad (28) \\ & \& [\rho_s > \rho_c] \end{aligned}$$

where  $\rho_s$ ,  $\rho_c$ ,  $\varepsilon_{\beta,s}$  and  $\varepsilon_{v,s}$  are the overlap percentage between predicted and constructed squares, minimum overlap constant, the accuracies of the angle and velocity, respectively. The overlap percentage  $\rho_s$  between the predicted and constructed squares is

$$\rho_s = \frac{100 \cdot S_{oa}}{4 \cdot \min\{\tilde{R}_{s,x}(k-1) \cdot \tilde{R}_{s,y}(k-1), R_{m,x}(k) \cdot R_{m,y}(k)\}} \quad (29)$$

where the overlap area  $S_{oa}$  is the constructed rectangle geometry between predicted rectangle  $\tilde{\mathbf{S}}(k-1)$  and the constructed rectangle by circles in  $\mathbf{S}_m$  is defined by

$$S_{oa} = L_{oa,x} \cdot L_{oa,y} \quad (30)$$

where

$$\begin{aligned}
L_{oa,x} &= \left| \min\{\tilde{L}_{s,x}(k-1) + \tilde{R}_{s,x}(k-1), L_{m,x}(k) + R_{m,x}(k)\} \right. \\
&\quad \left. - \max\{\tilde{L}_{s,x}(k-1) - \tilde{R}_{s,x}(k-1), L_{m,x}(k) - R_{m,x}(k)\} \right|, \\
L_{oa,y} &= \left| \max\{\tilde{L}_{s,y}(k-1) - \tilde{R}_{s,y}(k-1), L_{m,y}(k) - R_{m,y}(k)\} \right. \\
&\quad \left. - \min\{\tilde{L}_{s,y}(k-1) + \tilde{R}_{s,y}(k-1), L_{m,y}(k) + R_{m,y}(k)\} \right|
\end{aligned}$$

After satisfying the condition (28), the parameters of the square filter are estimated by using equation (2). Please note that for the initialization all squares are constructed by the collected mean squares in equations (14)-(16). Also, the trust factor is updated similarly to the normal circle definitions.

## 5. Evaluation

In order to evaluate the performance and the properties of the proposed LCS filter, we first explain the filter in a simple scenario in Section 5.1, and then we check how individual expert works. Also, memory overhead and computational efficiency are evaluated. Next, we apply the filter for autonomous vehicles in Section 5.2. Finally, the filter's performance is checked with real-world scenarios in autonomous cars on a sunny day and a rainy day at night with a high threshold for corner detection.

### 5.1. Filter Behavior in Simple Scenario

The experiment is set up as follows: an IMU sensor and a high-definition camera (with a resolution of  $640 \times 480$  pixels) are attached to a mobile robot platform to collect image and motion data which are fed to the LCS algorithm. Our study takes place with the worst-case scenario where the frame rate is at 1 frame/s. Although our geometric filter demonstrates its best performance with at least 3 frame/s, due to low velocity ( $V_v = 3$  cm/s) and minor acceleration  $a_v = 0.05$  cm/s<sup>2</sup> of the robot. A corner detector, *FAST9* [31, 32], is used as the detection back-end, with 20 point threshold utilized. All the filter parameters are given in Table 1. It is assumed that there are minor deviations ( $\delta_v = 9^\circ$ ) in the robot motion. We have chosen some of the values for the parameters (for example the accuracy parameter  $\epsilon$ ) with respect to the environment geometric complexity, object detection accuracy, and processing speed. For instance, if the environment is very complex, the angular and velocity accuracy of the Circle ( $\epsilon_{\beta,n}, \epsilon_{\beta,r}, \epsilon_{v,n}, \epsilon_{v,r}$ ) and the Square experts ( $\epsilon_{\beta,1}, \epsilon_{\beta,2}, \epsilon_{\beta,s}, \epsilon_{v,s}$ ) variables should be re-tuned (certain increase) to prevent over-computation. By increasing accuracy (lower  $\epsilon$ ), we force the filter to separate landmarks with a higher possible degree which means more memory usage. Also, the trust factors are chosen in given values to let the square candidates stay longer than circles due to their abstract and high-level object detection.

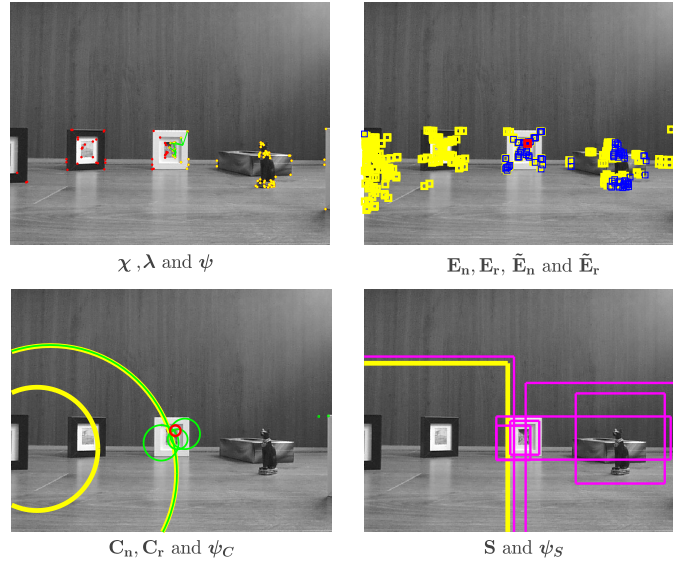
The environment is chosen crowded with an average of 75-200 landmarks in each frame. The linear kinematics of the vehicle is solved using MATLAB *ode45* solver to predict the location of landmarks and experts (circle and square) as

well as the  $\Delta r$  in equation (22). Please note that the superscripts  $c$  and  $s$  in trust factors  $Tr$  in Table 1 stand for the Circle (edges and circles) and Square experts.

At first, we show an example frame as in Figure 13 to explain how the results are interpreted in each expert and its geometric presentation.

We have collected a total of 37 frames during the motion of the robot. The full video is available in our open-source code. We have located certain objects (statue) with different forms as well as frame objects that contain complex graphical presentations as shown in Figure 13. Each frame consists of four identical images and each shows specific parameters of the LCS filter. At first, the top-left hand-side frame presents the operation of the Line expert [see Figure 4 for the flowchart of the Line expert] with detected landmarks  $\chi$  in yellow dots where they were obtained by *FAST9* edge detection algorithm. Red dots are the ones that our filter ignores by  $\psi$  which is updated through the Circle and Square experts. Also,  $\lambda$  is the grouping parameter that Line expert does to decrease the computation and find the detection radius of the edges  $\mu$  when  $\chi$  is transferred to the Circle expert. Then, top-right and bottom-left figures in Figure 13 is for the Circle expert, the Circle expert computation flow is in Figure 5 and Figure 7. The blue and red small squares are the final estimated location of the normal  $E_n$  and rebel  $E_r$  edges by the expert where they are calculated by equation (2). Note that these red landmarks do not follow the normal vector field flow which means the object either coming toward the camera or is an independently moving object in the frame. The yellow small squares are presenting the predicted squares ( $\tilde{E}_n, \tilde{E}_r$ ) of frame  $k-1$  before doing estimation. Next, the circling operation takes place where green and red circles are presenting the normal  $C_n$  and rebel  $C_r$  circles [see bottom-left image]. These geometries are tracking the layers that have similar relative velocity, location, and angle. The yellow circle stands for the layer that reaches maximum trust  $Tr_m^c$  in the frame  $k-1$  where the ignorance (Circle expert ignorance update  $\psi_C$ ) is applied by the update in  $\psi$  at current frame ( $k=9$ ). Finally, the bottom-right figure shows the Square expert operation based on the flowchart in Figure 10. Similar to the Circle expert, magenta and yellow rectangle geometries are presenting the estimated squares  $S$  and ignored  $\psi_S$  (which get updated from the previous frame) ones. Note that the Square expert combines the layers of the circles to determine more compact objects based on their matching kinematics geometrically. Thus, after every frame passes, the accuracy of estimation gets higher and information becomes more compact and summarized (high-level features) [see Figure 1 for the overall filter]. This helps to create a continuous feed to minimize the amount of collected data, e.g., edges.

We check the behavior of the LCS filter with some collected frames ( $k = 4, 14$  and  $25$ ) as shown in Figure 14. It is clear that the location of the landmarks in the first row of images is estimated with accuracy where they are shown as small blue squares. However, because we have chosen our study as 1 frame/s in the worst-case scenario, some mo-



**Figure 13:** Example frame ( $k = 9$ ) with filter experts. Top left: red and yellow dots are showing detected  $\chi$  and ignored edges  $\psi$ , line presents the line spacing  $\lambda$ ; Top right: the yellow and blue squares are approximated edge  $\tilde{E}_n, \tilde{E}_r$  and estimated normal edge  $E_n$  and red square is rebel edges  $E_r$ ; Bottom left: green, red and yellow colors are for normal circles  $C_n$ , rebel circles  $C_r$  and ignored circles  $\psi_C$ ; Bottom right: magenta and yellow rectangle are the squares  $S$  and ignored ones  $\psi_S$ .

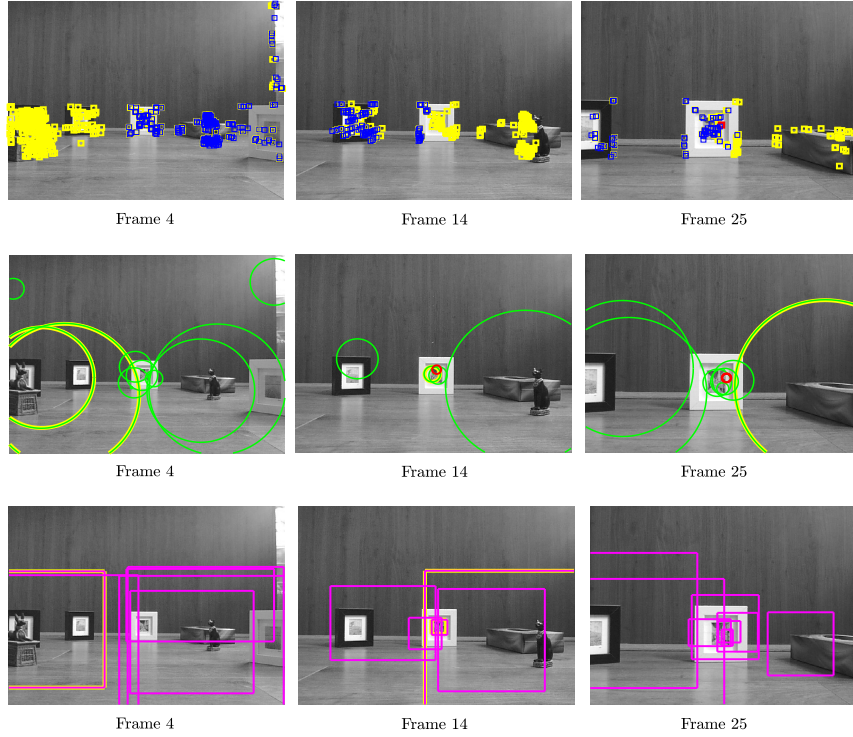
**Table 1**  
Parameters for the LCS filter used in the experiment.

Variable	Value	Variable	Value	Variable	Value
$O_{I,x}$	320	$Tr_c^c$	3	$\epsilon_{\beta,1}$	15
$O_{I,y}$	240	$Tr_c^s$	2	$\epsilon_{\beta,2}$	35
$V_v$	3 cm/s	$Tr_m^c$	5	$\epsilon_{\beta,s}$	20
$a_v$	0.05 cm/s <sup>2</sup>	$Tr_s^s$	5	$\epsilon_\beta$	20
$\delta_v$	9°	$Tr_c^s$	3	$\epsilon_{v,r}$	100
$\delta_\beta$	90°	$Tr_m^s$	7	$\epsilon_{v,s}$	0.7
$\mu_0$	25	$\epsilon_{\beta,n}$	20	$\epsilon_{v,n}$	40
$\rho_c$	40	$\epsilon_{\beta,r}$	50	$\epsilon_v$	0.7

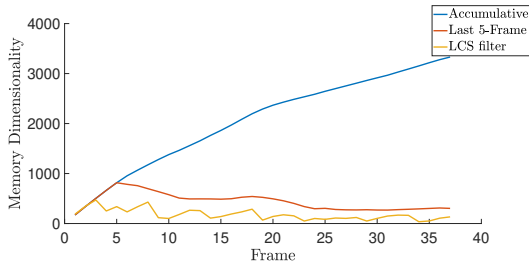
tion errors create a deviation in predicted landmarks (yellow squares). This can be minimized by having more accuracy in the velocity/angular displacement coming from IMU and increasing the number of frames that are sampled. As a strength of the LCS filter, it is able to detect the rebel edges that stand for incoming objects in the scene. The next row of images is for circles with expressing layers with similar kinematics. As the robot moves closer to the objects, they get distinguished better from each other [see Frame 14 and 25] rather than being in a compact form [see Frame 4]. Next, the Square expert can locate the objects especially the one that is in the middle. Please note that one can increase the trust factor of the Square expert to keep the coming data. We think increasing the trust factors ( $Tr_s^s$  and  $Tr_m^s$ ) and decreasing the critical trust ( $Tr_c^s$ ) are beneficial for a higher accuracy. However, in that case, the sensory data (IMU) and the robot kinematics model should be more accurate to prevent over-confidence in the filter. It is important to note that when a certain number of objects are too far away, their velocity/location are unified. Thus, this can be an interesting

future work to determine the distance sensitivity of this proposed geometric filter. Also, the algorithm determines the objects with  $\delta_\beta = 90^\circ$  angular matches here in an arbitrary direction. We plan to see, as future work, how we can transform the square data to detection for more smooth curves on the complex objects, more in their natural curvature (such as the status in the scene). However, the designed filter can be very beneficial for applications including mapping (Monocular SLAM) [25] and motion control [41, 42] that utilize the image features (edge detection) in their planning model or control law. It can be used as a data reduction filter for the scene to speed up the computation while it is increasing the accuracy of detected landmarks. Also, with specifying properties, e.g. velocity, of certain objects in the frame, we can use the filter to apply accurate object recognition methods [40, 9, 33] in only certain regions of the frame for recognizing what they are.

Next, we check the performance of our filter with respect to the memory dimensionality and computation complexity. In here, the memory dimensionality means the matrix

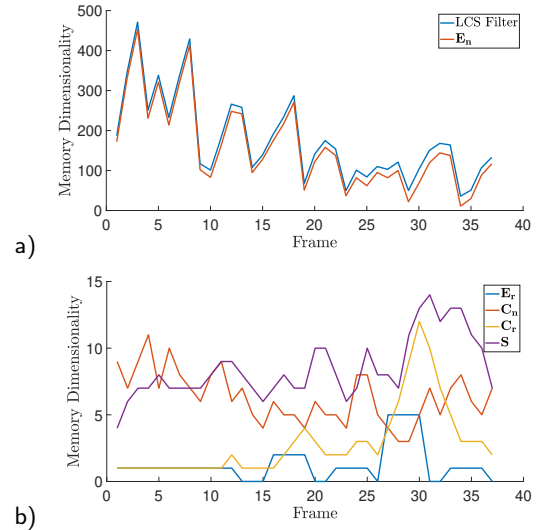


**Figure 14:** Comparison between sampled frames for understanding the Circle and Square experts behavior.



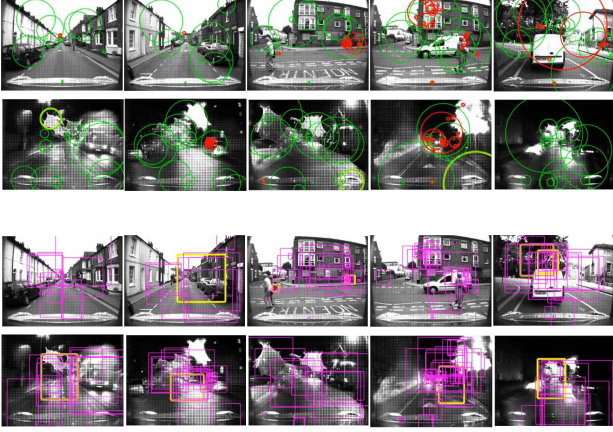
**Figure 15:** Result of the memory dimensionality versus the frame index.

size. The dimensionality can be looked at as the number of candidates (landmark or layer) each parameter has in the filter (the parameters are in Figure 1) every frame. Figure 15 shows that the LCS filter memory usage based on the dimensionality. We compared our filter with two cases of memory saving: An accumulative data collection that normally takes place in mapping problem and a 5-frame constraint data collection. We can observe that the total size of our filter (including all the expert parameters) is drastically smaller than both of the other cases. Even for the case that the last five frames are saved in memory, our filter still has a half size of its counterpart. Interestingly, as the LCS filter obtains more data in each frame, the dimensionality of it decreases to around 100. In order to have a more detailed analysis of the data flow of each expert in the filter, Figure 16 is presented. The most of data logically are restored in the normal edge  $E_n$  matrix since most of the objects are pass-



**Figure 16:** Detailed dimensionality analysis of the LCS filter parameters: a) The occupied size of normal edges  $E_n$  in comparison to overall LCS filter, b) The memory occupation for the parameters of the experts in the LCS filter.

ing with the following vector field of the vehicle [see Figure 2-a] motion as it moves with the high number of detected edges. Consequently, the rebel edges  $E_r$  and  $C_r$  circles occupy the least memory. However, there is a considerable increase in the number of detected rebel circles  $C_r$  as the camera reaches closer to the incoming frame object at the end (when  $k = [25 - 35]$ ). Additionally, the normal circles  $C_n$



**Figure 17:** Example captured frames (day/night) used in the autonomous car example.

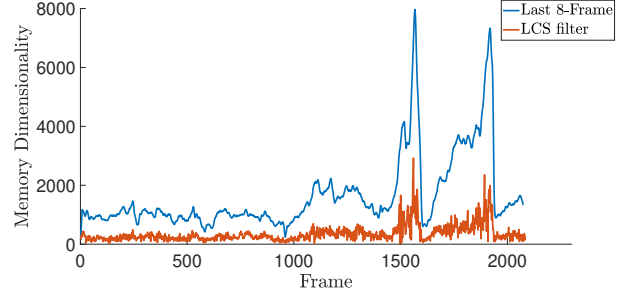
and squares  $\mathbf{S}$  have more dynamic fluctuation in their memory size but the overall trend is approximately constant.

In addition to the reduced memory footprint, the computational complexity of the LCS filter is relatively low. The time complexity of the circle expert (see the flowchart in Figure 7) is correlated to the number of input edges,  $n$ . The complexity of the circle expert is  $\mathcal{O}(2n \cdot \log(n))$  as two main search loops used in matching are involved. A similar result applies to the Square Expert (see Figure 10), in which the time complexity is based on the number of normal and rebel circles in total. As the filter is designed in three different layers, it is possible to run the filter in a pipeline fashion which allows for a higher degree of parallelism, e.g., by utilizing multi-cores or general-purpose graphic processing unit (GPGPU).

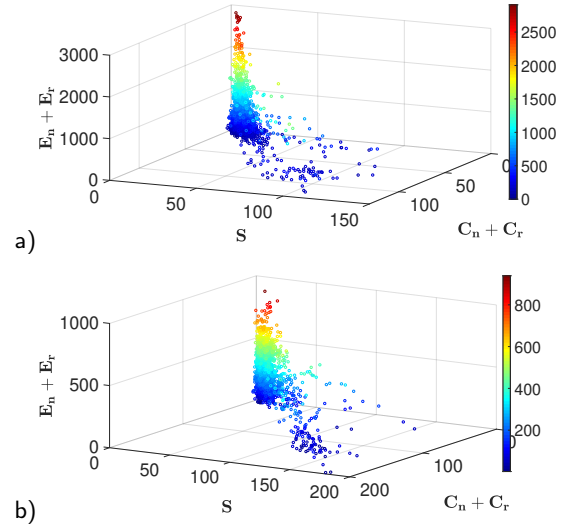
## 5.2. Experiment Studies on the Autonomous Cars

In this part, we applied our filter in a real-world scenario with autonomous driving cars. The Oxford RoboCar dataset [22] is chosen for this study.

Within the dataset, we have chosen scenes that are much more crowded and complex with over 700 detected edges in each frame. The algorithm trust factors are re-tuned as  $Tr_c^c = 1$ ,  $Tr_s^c = 4$  and  $Tr_m^c = 7$  for the Circle expert and for the Square expert  $Tr_c^s = 2$ ,  $Tr_s^s = 5$  and  $Tr_m^s = 8$  to have better performance. Other accuracy variables have minor changes where reader can check the open-source code for the details. We have experimented with the following two cases: 1) a scene during day light and 2) a rainy scene at night. Both cases are experimented with over 2200 frames. Figure 17 depicts some sampled frames from our study. The Circle expert in our filter is able to detect the independently moving objects (rebel landmarks) as pedestrians and passing-by cars as well as normal landmarks. This is true even when it is night with many distorted images due to rain. For example, the filter detects a parking car that moves toward the camera and then parks, see the last image before the end in Figure 17. Also, the Square expert can group the layers of the objects in both cases. The long-version videos are included in our



**Figure 18:** Dimensionality versus the frame index from the autonomous car experiment.



**Figure 19:** The dimensionality (memory size) comparison of the filter in each frame. a) It is the scene at day light with no disturbance on camera. b) It is the night scene with raining disturbance at camera lens.

source code repository for further check.

In order to verify our findings in simple scene analysis (dimensionality study) in the previous study (lab environment), the dimensionality of the filter is presented in Figure 18. The filter outperforms the last 8-frame with an average of 360 overall sizes of which shows how effectively our filter can decrease the memory size even when there is overwhelmingly large numbers of landmarks are detected. Note that we have excluded accumulative memory save since the size of the landmarks rises incredibly large (can be understood from the last 8-frame case). Also, the consistency of detected edges can be understood from the steady value of the dimensionality. Next, Figure 19 illustrates the behavior of the proposed filter in general. We see that the filter has a consistent behavior despite the change in the number of landmarks in  $\mathbf{E}_n + \mathbf{E}_r$ . The pattern kept almost the same in the worst-case scenario as Figure 19-b. However, the droplets that disturbed the motion field prediction through the filter have deviations which that is why the number of circles and squares are more. Overall, we think the filter is one of its kind that can operate at an acceptable level in highly dis-

turbed images that were not normally considered in previous studies.

## 6. Conclusion

In this paper, we present a novel geometric Line-Circle-Square (LCS) filter that is the successor of the original Line-Circle (LC) filter [36]. This filter has three layers, with an additional square layer compared with the LC filter, each of which consists of experts that have dedicated purposes. The first expert is the Line expert to collect, ignore and group the edges depending on the incoming feed from other experts. Next expert, the Circle expert, estimates the edges with defined kinematics, and then it groups the edges as a layer (circles) depending on their locations, angle, and velocity in the frame. Finally, the Square expert collects and matches the circles with geometrical and kinematic conditions. The Square expert with the highest information level tries to detect the objects partially or fully. Also, there is information feed between experts to keep them updated and minimize the level of overconfidence. With the obtained filter, the data of the landmarks becomes more accurate, compact, and dynamic as more frames are collected from the scene.

The main advantage of the LCS filter is to provide accurate features of the environment at different levels from low (edges) to high (layers). Also, it reduces the computation with its defined learning feed when there are overwhelming features in the environments. It is important to note that this filter is not meant to recognize objects with high accuracy. However, the proposed LCS filter can be utilized as the supervisor to feed only the specific regions of the frame that conventional recognition approaches will determine the objects. On the other hand, this filter without having any pool of images or reference images is able to detect and track objects fully or partially (layers) depending on their kinematics and amount of information that comes from the captured images.

For future work, we will investigate parallel execution of the experts of the filter with utilizing multi-core processors to speed-up the computation. Additionally, we plan to apply the error estimation matrix to all experts for operating the filter faster and more accurately in our incoming works. We think that the proposed filter will be a high potential candidate after the integration of error estimation to the decision parameters. This will help us to automate the choice of the best trust factors in the air within highly dynamic scenes.

## References

- [1] Bibby, C., Reid, I., 2008. Robust real-time visual tracking using pixel-wise posteriors, in: *Proceedings of European Conference on Computer Vision*.
- [2] Bouguet, J.Y., 2000. Pyramidal implementation of the lucas kanade feature tracker. Technical report, Microprocessor Research Labs .
- [3] Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y., 2015. Compressing convolutional neural networks. *arXiv preprint arXiv:1506.04449* .
- [4] Cheng, Y., Wang, D., Zhou, P., Zhang, T., 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282* .
- [5] Civera, J., Davison, A.J., Montiel, J.M., 2008. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics* 24, 932–945.
- [6] Collins, R., 2007. Introduction to Computer Vision course notes for CSE/EE486.
- [7] Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O., 2007. Monoslam: Real-time single camera slam. *IEEE transactions on pattern analysis and machine intelligence* 29, 1052–1067.
- [8] Eade, E., Drummond, T., 2006. Scalable monocular slam, in: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, pp. 469–476.
- [9] Gálvez-López, D., Tardos, J.D., 2012. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics* 28, 1188–1197.
- [10] Gamage, D., Drummond, T., 2015. Reduced dimensionality extended kalman filter for slam in a relative formulation, in: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1365–1372.
- [11] Geronimo, D., Lopez, A.M., Sappa, A.D., Graf, T., 2009. Survey of pedestrian detection for advanced driver assistance systems. *IEEE transactions on pattern analysis and machine intelligence* 32, 1239–1258.
- [12] Hartley, R., Zisserman, A., 2003. Multiple view geometry in computer vision. Cambridge university press.
- [13] Henriques, J.F., Caseiro, R., Martins, P., Batista, J., 2014. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence* 37, 583–596.
- [14] Hu, H., Peng, R., Tai, Y.W., Tang, C.K., 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* .
- [15] Kalal, Z., Mikolajczyk, K., Matas, J., 2012. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 1409–1422.
- [16] Latgahn, H., Stiller, C., 2014. Vision-only localization. *IEEE Transactions on Intelligent Transportation Systems* 15, 1246–1257.
- [17] Lichtsteiner, P., Posch, C., Delbruck, T., 2008. A 128x128 120 dB 15  $\mu$ s latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits* 43, 566–576.
- [18] Liu, L., Guo, R., Wu, J., 2018. A collision-free motion planning method by integrating complexity-reduction slam and learning-based artificial force design. *Robotics and Autonomous Systems* 100, 132–149.
- [19] Lui, V., Drummond, T., 2015. Image based optimisation without global consistency for constant time monocular visual slam, in: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5799–5806.
- [20] Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Zhao, X., Kim, T.K., 2014. Multiple object tracking: A literature review. *arXiv preprint arXiv:1409.7618* .
- [21] Ma, D., Jiao, X., 2020. Axby: Approximate computation bypass for data-intensive applications, in: *2020 23rd Euromicro Conference on Digital System Design (DSD)*, IEEE. pp. 332–339.
- [22] Madder, W., Pascoe, G., Linegar, C., Newman, P., 2017. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)* 36, 3–15.
- [23] Mueggler, E., Bartolozzi, C., Scaramuzza, D., 2017. Fast event-based corner detection., in: *BMVC*.
- [24] Mur-Artal, R., Montiel, J.M.M., Tardos, J.D., 2015. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics* 31, 1147–1163.
- [25] Mur-Artal, R., Tardós, J.D., 2017a. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics* 33, 1255–1262.
- [26] Mur-Artal, R., Tardós, J.D., 2017b. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters* 2, 796–803.
- [27] Murray, D., Little, J.J., 2000. Using real-time stereo vision for mobile

- robot navigation. *Autonomous Robots* 8, 161–171.
- [28] Pillai, S., Leonard, J., 2015. Monocular slam supported object recognition, in: *Proceedings of Robotics: Science and Systems*, Rome, Italy.
  - [29] Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A., 2016. Xnor-net: Imagenet classification using binary convolutional neural networks, in: *European conference on computer vision*, pp. 525–542.
  - [30] Rosten, E., Drummond, T., 2005. Fusing points and lines for high performance tracking, in: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pp. 1508–1515.
  - [31] Rosten, E., Drummond, T., 2006. Machine learning for high-speed corner detection, in: *European Conference on Computer Vision*, Berlin, Heidelberg. pp. 430–443.
  - [32] Rosten, E., Porter, R., Drummond, T., 2010. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 105–119.
  - [33] Rublee, E., Rabaud, V., Konolige, K., Bradski, G., 2011. Orb: An efficient alternative to sift or surf, in: *2011 International conference on computer vision*, pp. 2564–2571.
  - [34] Sand, P., Teller, S., 2008. Particle video: Long-range motion estimation using point trajectories. *International Journal of Computer Vision* 80, 72.
  - [35] Sun, Z., Bebis, G., Miller, R., 2006. On-road vehicle detection: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 694–711.
  - [36] Tafrishi, S.A., Kandjani, V.E., 2017. Line-circle: A geometric filter for single camera edge-based object detection, in: *2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pp. 588–594.
  - [37] Tuytelaars, T., Mikolajczyk, K., 2008. Local invariant feature detectors: a survey. Now Publishers Inc.
  - [38] Valmadre, J., Bertinetto, L., Henriques, J., Vedaldi, A., Torr, P.H., 2017. End-to-end representation learning for correlation filter based tracking, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2805–2813.
  - [39] Wang, S., Clark, R., Wen, H., Trigoni, N., 2017. Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks, in: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2043–2050.
  - [40] Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., Tardós, J., 2009. A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems* 57, 1188–1197.
  - [41] Wisth, D., Camurri, M., Fallon, M., 2019a. Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry. *arXiv preprint arXiv:1910.09875*.
  - [42] Wisth, D., Camurri, M., Fallon, M., 2019b. Robust legged robot state estimation using factor graph optimization. *IEEE Robotics and Automation Letters* 4, 4507–4514.
  - [43] Wu, Y., Lim, J., Yang, M.H., 2015. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 1834–1848.
  - [44] Zhan, H., Garg, R., Saroj Weerasekera, C., Li, K., Agarwal, H., Reid, I., 2018. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 340–349.
  - [45] Zhang, S., Wu, G., Gu, J., Han, J., 2020. Pruning convolutional neural networks with an attention mechanism for remote sensing image classification. *Electronics* 9, 1209.
  - [46] Zhao, Z.Q., Zheng, P., Xu, S.t., Wu, X., 2019. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems* 30, 3212–3232.