



A fine tuned tracking of vehicles under different video degradations

Mohamed Maher Ata¹ · Mohamed El-Darieby² · Mustafa Abd El-nabi¹

Received: 24 March 2017 / Accepted: 10 April 2018 / Published online: 20 April 2018
© Bharati Vidyapeeth's Institute of Computer Applications and Management 2018

Abstract Estimating the tracking efficacy of vehicles in traffic videos is one of the most desirable analysis specially in the presence of a challenging weather conditions. In this paper, a fine-tuning Kalman filter based tracking system has been proposed so that it will work robustly on the traffic videos. Such tracking efficacy has been tested and tuned by integrating a system that calculates the distance between adjacent vehicles as a case study. The integrated system could provide some sort of traffic warning system according to the allowable traffic safety standards. Analysis has been utilized in two phases; phase (1): by changing the performance indices of Kalman filter parameters (initial estimation error, motion noise, and measurement noise). We have measured both average number of assigned tracks and processing time of interest in order to acquire best tuning decision. From observations, changing values of initial estimation error has no effect on the performance of the tracking efficacy however increasing both motion noise and measurement noise has an adverse impact on the tracking performance. Phase (2) by applying the integrated system on a degraded version of a captured urban traffic video to measure performance of the tracking procedure according to the existence of salt and pepper, Gaussian, and

Speckle video degradations. Such video disturbances could perform an evaluation for some sort of challenging weather conditions (e.g., rain, fog, and reduced light conditions). It is obviously that average number of assigned tracks has been degraded in the presence of video disturbance with respect to percentage of occurrence and the appropriate statistical features (mean, and variance) of such degradation. Twelve different types of filtering mask have been applied in order to measure average number of assigned tracks (correct predictions) after the cleaning process. We have measured the deviation between both the no noise and the with noise traffic video to study effect of each filter mask within each noise type of video disturbance. Such deviation measurements introduce a decision making criteria for best tuning that increases the efficacy of the vehicular tracking.

Keywords Intelligent transportation system (ITS) · Blob analysis · Kalman filter · Assigned tracks · Video degradation · Salt and pepper · Gaussian noise · Speckle noise · Filter mask

1 Introduction

Intelligent transportation systems (ITS) refers to a variety of tools, software, hardware, and communication technologies that could be applied in an integrated fashion to improve the efficiency and safety of vehicular traffic [1, 2]. ITS provides support to enhance operation of transportation services, transit management and information to travelers [3, 4]. Research in ITS is targeting improvements in safety of future transportation systems through integrating safety enhancing functions within vehicles. Technologies such as Radar/Lidar, loop detectors and traffic video analysis have

✉ Mohamed Maher Ata
mmaher844@yahoo.com

Mohamed El-Darieby
mohamed.el-darieby@uregina.ca;
http://uregina.ca/~eldariem/

Mustafa Abd El-nabi
mnaby45@gmail.com

¹ Electrical Communication and Electronics Department,
Faculty of Engineering, Tanta University, Tanta, Egypt

² Faculty of Engineering, Regina University, Regina, Canada

been used to provide such safety features. We discuss these technologies in Sect. 2 below.

The on-road automated vehicular detection and tracking has been considered as one of the most valuable research point over the past decades [5, 6]. Such point of interest plays a vital role in the evolution of intelligent transportation systems (ITS). Many available techniques have been grown up for the on-road vehicular detection. Those techniques can be classified into software based computer vision technique and hard ware active sensors based Millimeter radar and lidar techniques. Computer vision methodology introduces a good point of view in the state of the art of analyzing traffic videos. Vehicle interaction, automated traffic warning system, traffic rule violation, and congestion are good examples which can be targeted using surveillance on-road installed cameras. Foreground estimation, background estimation, and motion tracking are classical visual techniques for detecting and classifying vehicles on highways of interest. Video analysis of urban areas are still more challenging because of its dependency on some sort of road parameters such as traffic density, variation of road users, and the degree of occlusion [7]. In order to performing a comparison study between the proposed algorithms, it would be more difficult to perform such study as there is no standardized benchmark dataset to be used [8].

ITS provide the opportunity to establish functions in the infrastructure and/or vehicle to mitigate these deficiencies. For example, sensors in the main line highway could provide an advance warning about an oncoming vehicle to side street traffic on a stop-controlled intersection, to compensate for any sight distance deficiencies for the side street traffic. In-vehicle sensors could provide advance warning to inattentive or drowsy drivers before hitting another vehicle or object, or before running off the road. The possibilities to improve the safety of our transportation system are endless [9].

In this paper, video analysis has been proposed in order to consider vehicular tracking operations under different weather conditions (e.g., rain, fog, and reduced light conditions). accordingly, such analysis has been rarely performed in spite of being highly desirable. The proposed traffic video analysis setup a Kalman filter in the presence of weather conditions. The paper uses three different types of video degradation noises (salt and pepper, speckle, and Gaussian) with different levels of occurrence to analyze traffic videos. In addition, a system of filters has been applied to the degraded version of the test video producing a new record with respect to filter masking system. We calculate distance between two vehicles as an example application of vehicular tracking. Calculating such distance enhances safety function in order to provide an automated warning system in order to increase safety. This may result

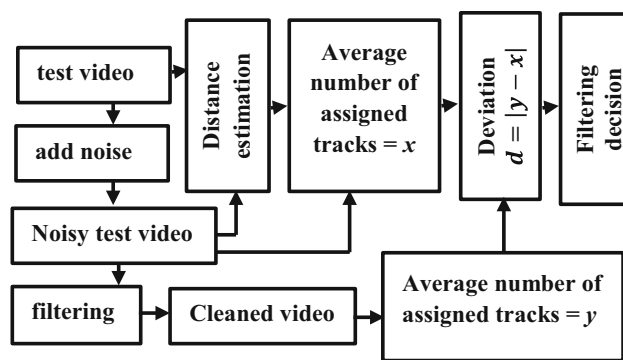


Fig. 1 Proposed analysis block diagram

in reducing aggressive driving behavior as well as providing drivers with more time to react to road events. Figure 1 shows the main block diagram of the introduced study.

The rest of this article has been organized as follow: Sect. 2 ITS vehicular tracking technologies; Sect. 3 distance estimation and tracking algorithm; Sect. 4 introduces code setup; Sect. 5 introduces experimental results; Sect. 6 conclusion; references.

2 ITS vehicular tracking technologies

In this section, an overview has been introduced for the following ITS technologies: radar/Lidar and computer vision. One of the most widely used techniques that serves the process of detecting vehicles is the Millimeter radar active sensor. Typically, a continuous waveform signal which is frequency modulated will be emitted. Once receiving the demodulated wave form, the frequency content will be analyzed. The distance between the active sensor and the appropriate vehicle of interest will be easily calculated according to the frequency shift between the transmitted and the received signal. Tracking the detected objects will be performed according to motion characteristic of interest [10].

Vehicular detection and tracking based active millimeter radar works fairly in a challenging weather conditions (rain, fog, and darkness). In addition, in case of noisy measurements, a cleaning process would be extensively required. Millimeter radar detects and tracks all moving objects. A classification process would be necessary to classify those objects as vehicles according to the appropriate relative acceleration, motion, and size of interest. Furthermore, detection of stopped vehicles would be fairly performed [10, 11]. As millimeter radar, lidar detects and tracks all moving objects. A classification process would also be necessary to classify those objects as vehicles according to the appropriate relative acceleration, motion,

and size of interest too. However, lidar provides cleaner measurements and more sensitivity to precipitation than radar. Lidar use a rotating hexagonal mirrors which split the laser beam [11]. The upper three beams are used for detecting vehicles and appropriate obstacles however the lower three beams are used for detecting road features and land marks [12]. Lidar cost remains a challenging issue. Vehicular detection and tracking based computer vision uses a system of installed surveillance cameras. Acquisition system based cameras provide a wide range view which allowing the vehicular detection and tracking across multiples lanes [13]. The appropriate imaging system contains a lens and a charged coupled device (CCD). By using means of computer vision, sophisticated computations would be required due to the presence of large amount of homogeneous mapped pixels in the digital video frames of interest. Successive video frames provide researchers with rich visual information source for manual inspection. In addition, there will be no traffic disruption for installation and maintenance. By using means of computer vision, recognizing objects as vehicles would be easier than active lidar and radar. In addition, there will be no need for any classification processes [14].

One of the most foundational and valuable technique based computer vision is the Kalman filter (KF). Kalman filter is an estimator which infers the appropriate parameters of interest from inaccurate, indirect, and uncertain perceptions [15]. Its filtering functionality based on a linear mean square error estimation. The main target of its filtering functionality is to minimize the estimated mean error covariance according to some sort of presumed conditions. Kalman filter produces good results due to optimality and structure in addition, KF offers a convenient form for online real time processing. However, KF also does not just clean up the data observations, but also projects these observations into an enhanced version of measurements [16]. The basic mathematical model for the KF involves a discrete-time nonlinear dynamic enhanced system as follow:

$$x_{k+1} = F(x_k, v_k), \quad (1)$$

$$y_k = H(x_k, n_k), \quad (2)$$

where x_k represents the un-observed state of the system and y_k represents the only observed state of the system, v_k represents the process based noise which drives the dynamic system, and n_k represents the observation noise [15]. The dynamic model based system F and H has been assumed to be known. In state-estimation, the KF is the best predictor which has the ability to achieve a recursive maximum likelihood estimation of the state [16]. KF model could be expressed as follow:

$$\widehat{X}_k = K_K \cdot Z_k + (1 - K_K) \cdot \widehat{X}_{K-1}, \quad (3)$$

where \widehat{X}_k is the current estimation, K_K is the Kalman filter gain with discrete values (K_1, K_2, K_3, \dots), Z_k is the measured value, and \widehat{X}_{K-1} is the previous estimation.

Kalman filter has five performance indices which affect directly on the accuracy of the tracking methodology:

1. *Motion model* The KF can follow one of two motion models. The first model is the constant velocity based model where the velocity of the moving object is assumed to be constant. The second model is the model based constant acceleration where the object is assumed to be accelerating at a constant rate.
2. *Initial location* For the KF to be able to track a certain object, its initial location must be known. This parameter is given in X and Y coordinates.
3. *Initial estimation error* This parameter expresses the amount of error in the X and Y directions which the KF should accept before deeming a certain track as “unacceptable” and dropping it altogether. It only affects the accuracy of the first few predictions since the KF adapts and creates its own estimation error values later on based on the previous results. Increasing this value will make it possible for the KF to adapt faster but it might also make the first few predictions inaccurate.
4. *Motion noise* This parameter represents the acceptable deviation from the set Motion Model because it might not fit the object’s velocity or acceleration perfectly. Increasing this parameter might be good for the reason of making the KF more suitable for the object’s movement but it might also cause some inaccuracy.
5. *Measurement noise* This value is given as a scalar. Increasing it causes the KF to adhere more to the set Motion Model.

3 Distance estimation and tracking algorithm

Figure 2 introduces the main algorithm for estimating the distance between two adjacent vehicles after the detection procedure for the vehicle in a certain video and how this vehicle is kept tracked through that video according to their centroids and how any other moving objects being eliminated according to specifying a minimum threshold size of interest. The following steps summarize the algorithm:

Step 1 traffic video based multiple moving vehicles would be read according to Algorithm 1:

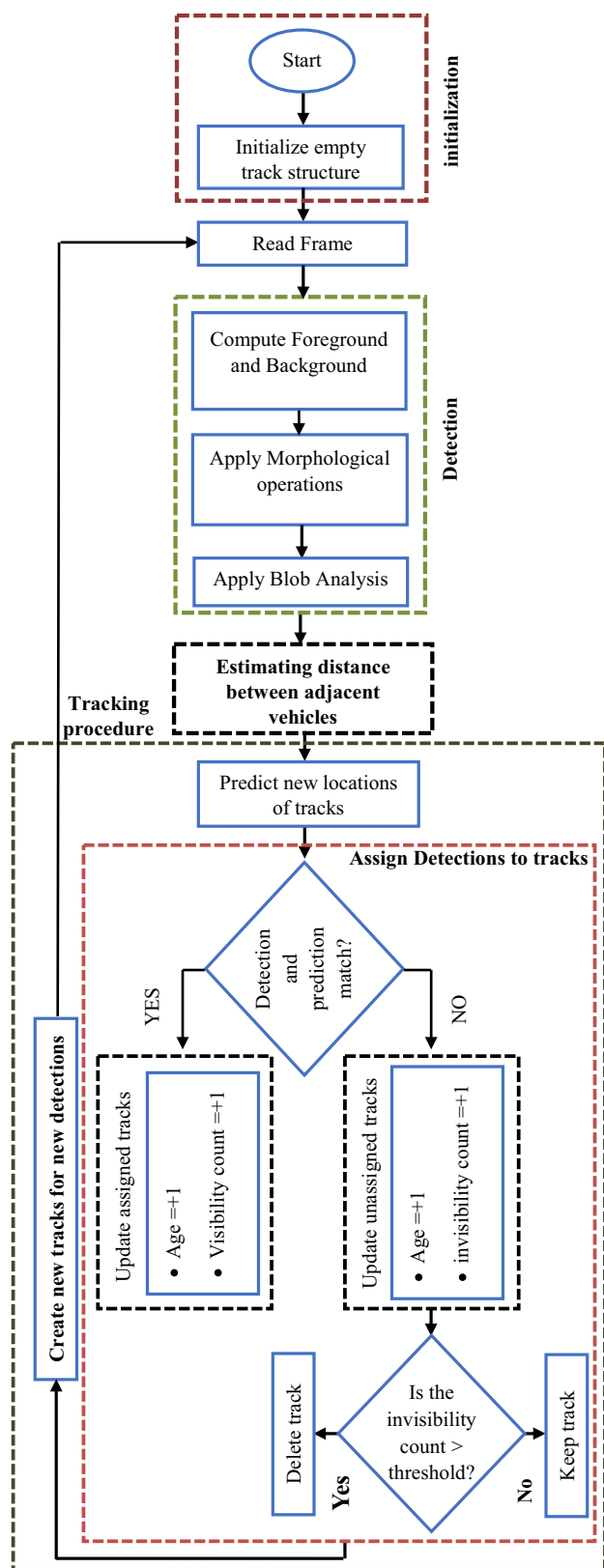


Fig. 2 Shows the flow chart for detection, estimating distance and tracking procedure in the proposed system

Algorithm1: read traffic video contains multiple moving vehicles

```

// Load traffic video using a video reader object
x= vision.videofilereader('test video.extension')
// Create video player object to play traffic video file
// Read and play traffic video frames
• While ~isDone(x)
  x=video_frame
  Video_player(x)
• End
// Release both video reader and player objects
Release (x)
Release (video player)
  
```

Step 2 Separate the background from the foreground (vehicles). A number of consecutive frames have been taken and the pixels have been divided into static and dynamic pixels. Foreground detector uses means of background subtraction technique. The appropriate methodology has been utilized as follow:

- Specify a reference frame which represent the appropriate back ground of interest. Now, the background model based frame would be initialized.
- Estimate the appropriate threshold value in order to satisfy the required detection rate of interest. The selection of the threshold plays a vital role in the subtraction operation.
- Identify and classify the type of given pixel with respect to degree of both brightness and chromaticity compared with pixels in the background frame. The four pixel classes would be summarized as follow:

Class 1: moving foreground based pixels According to both chromaticity and brightness, they would be different from the expected values in the background frame.

Class 2: shaded background based pixels According to both chromaticity and brightness, chromaticity would be similar to those in the background frame however brightness would but lower.

Class 3: ordinary background based pixels According to both chromaticity and brightness, they would be similar to those in the background frame of interest.

Class 4: highlighted background based pixels According to both chromaticity and brightness, chromaticity would be similar to those in the background frame however brightness would but higher.

A binary frame has been resulted where black represents the background and white represents the moving as shown in Figs. 3 and 4. The mathematical representation for the background subtraction was as follow:

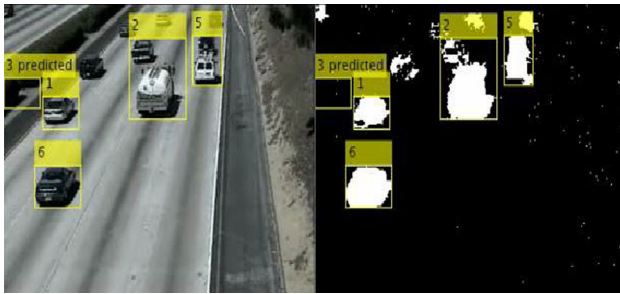


Fig. 3 Shows detected vehicles



Fig. 4 Shows the next frame of interest

$$x_t(s) = \begin{cases} 1 & d(I_{s,t}, B_s) > \tau \\ 0 & \text{elsewhere} \end{cases}, \quad (4)$$

where I: observed video sequence, $I_{s,t}$: foreground model at pixel s within time t, B_s : background model at pixel s, B: static background, τ : threshold, x_t : motion mask, d: the distance between $I_{s,t}$ and B_s . The occurrence probability of color I at pixel s is given by:

$$p(I_{s,t}) = \sum_{i=1}^k w_{i,s,t} \cdot N(\mu_{i,s,t}, \Sigma_{i,s,t}), \quad (5)$$

where $N(\mu_{i,s,t}, \Sigma_{i,s,t})$: is the ith Gaussian model and $w_{i,s,t}$ is the appropriate weights.

Step 3 Apply morphological operations to preprocess the appropriate test video and remove undesirable objects. Two phases have been introduced for such analysis; first phase is to get rid of undesirable objects such as smaller (i.e. birds) and larger (i.e. pedestrians) moving objects compared with the size of the desired moving vehicles of interest [17, 18]. This phase would be concerned in the state of the art of adaptive thresholding. Algorithm 2 introduces phase 1 based morphological operations of interest.

Algorithm2: phase 1 based morphological operations

```

// Read successive frames
While ~isDone(x)
  x=video_frame
//partition each frame for (for examples) four
  equally partitioned regions. Assume 256×256
  frame based test video
  p1=x(:,1:64)
  p2=x(:,65:128)
  p3=x(:,129:192)
  p4=x(:,193:256)
//Thresholding each region of interest
  Array_g1=frame_thresholding(p1)
  Array_g2= frame_thresholding(p2)
  Array_g3= frame_thresholding(p3)
  Array_g4= frame_thresholding(p4)
//Concatenate all partitions based frame
  concatenated_array= [g1 g2 g3 g4]
end
    
```

Second phase based morphological analysis guarantee the process of filling undetectable pixels in vehicle window. The appropriate filling would be concerned by using means of vehicle **closing**. This mathematical morphology based technique has been derived by applying dilation process cascaded by erosion process. The closing process working in the state of the art of enlarging the appropriate bright boundaries of the foreground objects (vehicles) in each frame and shrinking the appropriate background holes in such vehicles regions. Algorithm 3 introduces phase 2 of interest [19].

Algorithm3: phase 2 based morphological operations

```

// perform Algorithm2
// Assume a frame to have the following binary values
  y = array_thresholded_binary_frame
  set y=[1 0 0 0 1 1 1 0 1 1]
//assume structing element z containing only 1's
  set z=[1 1 1]
// frame dilation
  If z touches y_foreground (1's)
    //write a "1" at the origin of the structuring element
    Output1= [1 0 1 1 1 1 1 1]
  end
// frame erosion
  input= [1 0 1 1 1 1 1 1]
  If z touches y_background (0's)
    //write a "0" at the origin of the structuring element
    Output2= [0 0 1 1 1 1]
  end
    
```

The proposed system has been utilized regardless the possibility of object losses due to the conditions that change the objects appearance.

Step 4 Apply blob analysis in order to isolate the blobs (vehicles) in each binary frame. A blob consists of a group of connected pixels which represent each vehicle of interest. Blob analysis methodology has the ability of extracting the most salient statistical features; area, perimeter, centroids, bounding box [20]. All these features would be used in order to classify blobs (vehicles) in order to ease the decision making of that if they hold the objects which we are concerned about or not. In the introduced paper, we have concerned with the calculation and the assignment of the following properties to each detected vehicle; Area of the objects, Bounding box of the object, and X and Y coordinates of the blob’s centroid [24]. Figures 3 and 4 show the output from the proposed system for two successive frames after applying blob analysis according to calculating centroids of the moving objects [21, 22]. The state of the art of blob detector is based on normalized Laplacian of Gaussian (LOG)_{norm}:

$$L(x, y, t) = g(x, y, t) \times f(x, y, t), \tag{6}$$

where $g(x, y, t) = \frac{1}{2\pi t^2} e^{-\frac{x^2+y^2}{2t^2}}$ is the Gaussian kernel and $f(x,y,t)$ is the video frame of interest. The (LOG)_{norm} can be estimated as follow:

$$\nabla_{norm}^2 L = t(L_{xx} + L_{yy}). \tag{7}$$

Step 5 For each detected blob, perform the following:

- Assign an ID number that identifies the vehicle throughout the duration of its appearance in the video as shown in Figs. 3 and 4.
- Apply KF to actually track the appropriate vehicle and associate its detections throughout the video to a single track.
- Calculate duration of how long has a particular object been detected.
- Estimate total Visible Count which indicates how many consecutive frames has the particular object been detected.
- Estimate consecutive invisible count to indicate how many consecutive frames has the object been undetectable for.

Step 6 Estimate distance between two adjacent vehicles according to spatial coordinates of the centroid of each blob (calculated in step 4). The distance between these two centroids (x_1, y_1) and (x_2, y_2) of two blobs is calculated using the Euclidean equation:

$$d = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]}. \tag{8}$$

Step 7 For each of the following frames, the KF will predict new locations of blobs and places a bounding box around it [21].The reason of using KF instead of any other object tracking [i.e. Hidden Markov Model (HMM)] [22, 23] is that KF introduces some facilities; its ability to Predict moving vehicles in future locations, its ability to reduce the appropriate noise that introduced by inaccurate detections, KF provides some sort of Facilitating the process of association of multiple objects to their tracks, finally, KF introduces multiple moving vehicles tracking with lower processing time.

In addition, the reason of using the concept of Euclidean distance instead of any other distance estimation techniques [(i.e. K-nearest neighbor’s algorithm (k-NN)] [24, 25] is that this method offers an acceptable processing time and simplicity in computations [26, 27]. however, K-NN is a learning methodology with more sophisticated analysis due to its main applications as a classifier detector and predictor [28–31]. In the proposed analysis, a simple way for calculating such distance is required due to the continuous variation of the centroid values for each detected based in each frame.

Step 8 As a particular vehicle is predicted, a possibility of error is generally expected. To ensures tracking vehicles in spite of changing position, speed and acceleration, we calculate the distance of centroids of each blob calculated in two consecutive frames using the Euclidean Eq. (1). If this difference is found to be less than a specified threshold value, then this prediction is deemed “accurate” and the track’s confidence level is incremented. If the difference value is greater than the threshold value, then the prediction is deemed ‘inaccurate’ and the track’s confidence level is decremented. In order to illustrating this mechanism, a cost matrix, shown in Fig. 5, is created. This matrix consists of M rows and N columns. M is the number of tracks (predictions) and N is the number of detections. Each

cost of matching prediction number 1 to detection number 1

cost =	58.0362	44.3617	12.0783	
	18.5849	20.3701	26.3975	
	16.4591	12.0572	41.7484	
	12.0181	17.8909	58.0351	
cost =	55.4527	41.5394	12.0506	
	18.8696	20.1081	24.5344	
	16.3658	12.0500	40.3170	
	12.0092	17.3203	55.5829	
cost =	91.1889	52.2626	38.0291	12.0793
	18.7253	19.2707	20.0922	23.2953
	41.4337	16.4548	11.9995	39.1052
	22.9523	11.9906	16.2250	52.3244

Fig. 5 Shows cost matrix for detection and prediction

element in this matrix represents the cost of matching the Mth prediction to the Nth detection. This cost is calculated via the Euclidean equation for distance calculation. If this cost is low, then a match between the prediction and detection is achieved, otherwise the match does not happen. Another parameter that goes into the process of deciding whether the track is to be assigned to the detection or not is the “Cost of non-assignment” which represents the cost of not assigning a prediction or a detection. The higher this parameter is the more likely for most detections and predictions to be matched. Figure 5 show the cost matrix from the proposed system.

Step 9 Based on the values regarding “accurate” predictions obtained from steps 7 and 8, perform the following:

- Update the bounding box of the object to the current one instead of the previously predicted one.
- Adds [1] to the age of the track.
- Adds [1] to the visibility count for the track.
- Sets the invisibility count to [0].

Step 10 Based on the values regarding “inaccurate” predictions obtained from step 7 and 8, perform the following:

- Adds [1] to the age of the track.
- Adds [1] to its invisibility count.

Step 11 Delete tracks with frequent inaccurate predictions (it stays invisible for a certain number of consecutive frames).

Step 12 Create new tracks for every new vehicle that enters the camera’s scope, assign a corresponding track structure to start tracking by the KF.

4 Code setup

The Software of the proposed system has been developed using Matlab 2015 release (a) with a PC that have 4 GB RAM and 2.5 GHZ dual core processor. According to the following procedure. A graphical user interface has been integrated for estimating the required observations of the two appropriate phases based computer vision analysis.

➤ Set up objects

1. Create a video file reader.
2. Create a video player to display the original video.
3. Create a video player to display the distance.
4. Create an object from the Foreground detector.
5. Create an object from the Blob Analyser.

➤ Initialize Tracks

Create an empty array of tracks with the following parameters:

- ✓ ID
- ✓ Bounding Box
- ✓ Instance from the KF
- ✓ Age
- ✓ Total visible count
- ✓ Consecutive invisibility count

➤ Read appropriate Frame

➤ Detect objects

1. Use the Foreground detector object to get the mask for the frame.
2. Apply morphological operations on the frame.
3. Identify the blobs and their characteristics (Centroids and Bounding Boxes) using the Blob analyser object.
4. Calculate the distance between any two adjacent vehicles according to the following steps:
 - ✓ The coordinates of the centroids of all detected vehicles will be stored in a matrix called Centroids.
 - ✓ The distance between each of the centroids will be calculated.
 - ✓ If the distance between two vehicles goes below a certain value, a danger indication will appear.

This could be utilized as follow:

```

for (each element in the centroid array)
{
    calculate distance
    if distance < threshold
    {
        display distance
    }
    end
    if distance < safe value
    {
        initiate warning alarm
        display warning message
    }
    end
}
End
}

```

➤ Assign a track structure to each detected blob.

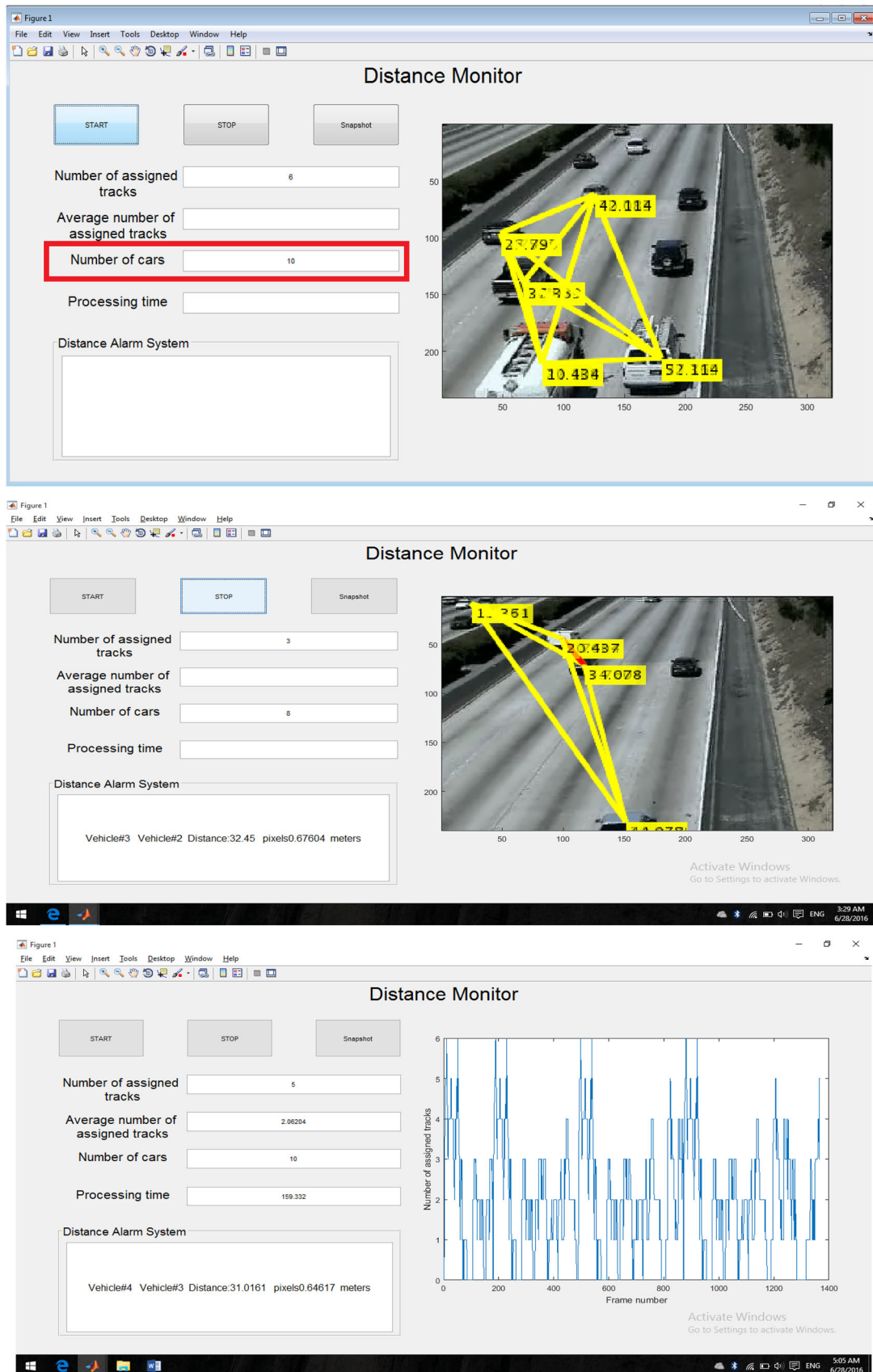


Fig. 6 Shows the interface of phase 1 system topology

- **Predict New Locations of Tracks**
 Predict the location of the blob in the next frame using the KF.
- **Detection to track assignment**
 1. Create cost matrix
 2. Assign a Cost of Non Assignment value
 3. Call the Assign Detections to tracks function which based on the cost matrix will decide which predictions are acceptable and which aren't. (As in which predictions match the detections and which ones don't match any detections).
- **Update Assigned Tracks**
 1. Increase the age of each matched track by 1.
 2. Set the invisibility count of that track to zero.
 3. Set the centroid and bounding box of the track to the currently detected values instead of the previously predicted ones to be used for the next prediction.
- **Update Unassigned tracks**
 1. Increase the invisibility count for unassigned tracks by 1.
- **Delete Lost Tracks**
 1. Delete a track if one of two conditions is satisfied:
 - ✓ The invisibility count surpasses the acceptable invisibility count threshold.
 - ✓ The age is less than 8 and less than 0.6 of that age was visible.
- **Create New Tracks**
 1. Create new tracks for unassigned detections.
 2. Add them to the track array

5 Experimental results

The objectives are to calculate distance between adjacent vehicles as an application of vehicular tracking. The main procedure was to measure efficiency of the tracking by

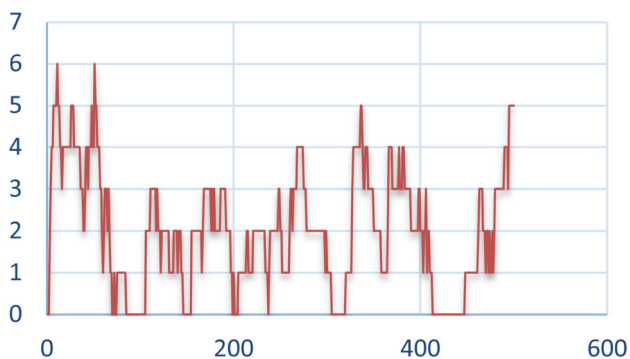
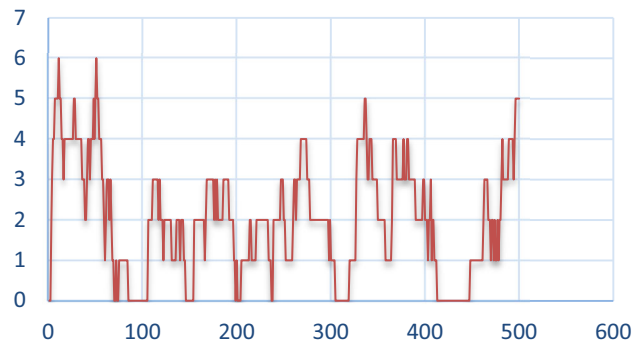
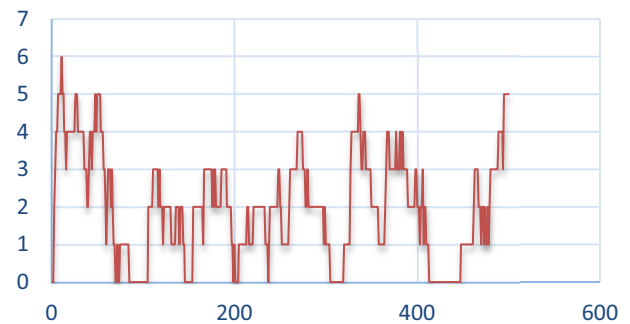


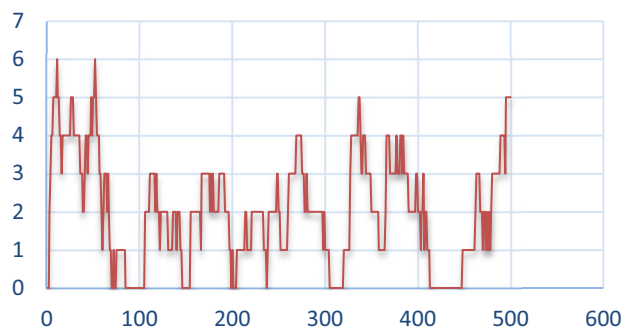
Fig. 7 Initial estimation error representation for all cases with no change



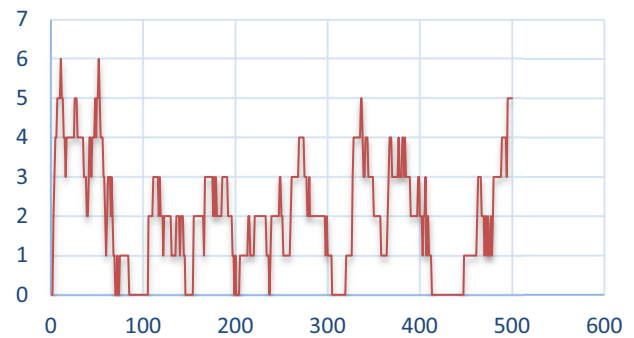
Case1: input (10,10) and input (100,25)



Case2: (100,100)



Case3: (150,150)



Case4: from input (170,170) to input (400,400)

Fig. 8 Motion noise representation with each case study

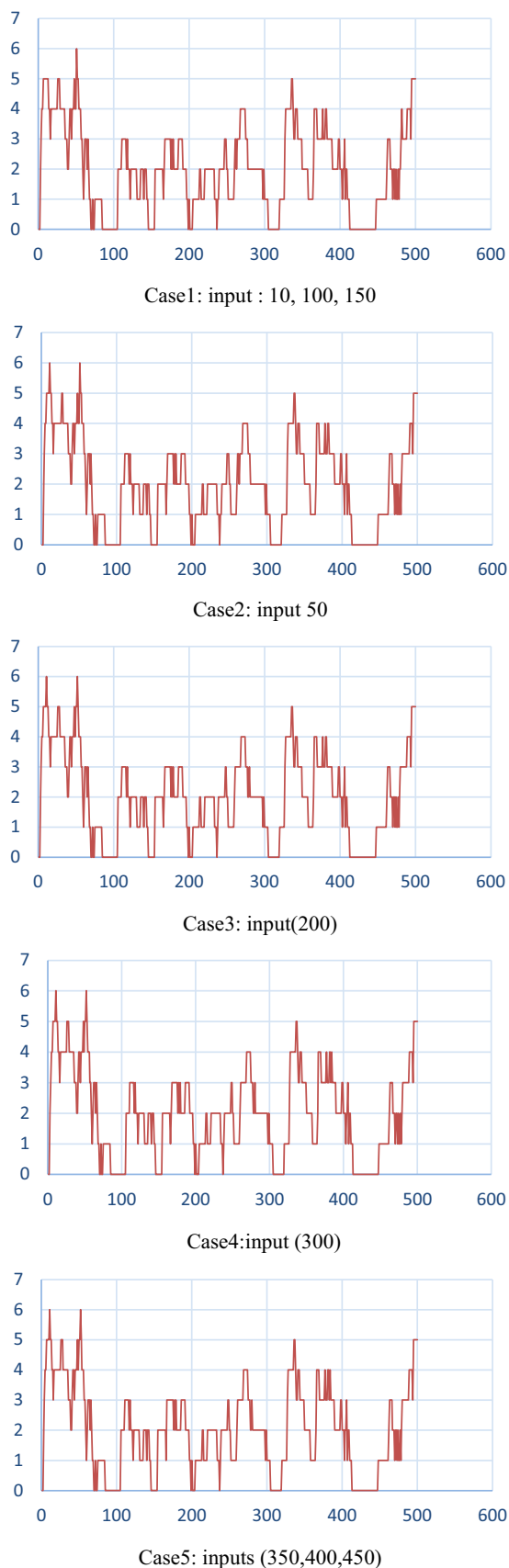


Fig. 9 Shows the measurement noise with each case study

analyze a raw test video with ideal weather conditions (i.e., no noise) and a simulated challengeable weather conditions (i.e., adding noise). A complete GUI interface has been utilized with the following indicators: number of assigned tracks, average number of assigned tracks, number of cars, distance alarm indicator, and processing time indicator. Figure 6 show screen shots from the interface of interest.

5.1 The no-noise case

The number of assigned tracks (correct predictions) has been measured in each frame according to the test video in response to varying the following KF parameters: initial estimation error, motion noise, and measurement noise. Motion model has been set to a constant velocity model and initial location has been set to the coordinates of the centroids. Figures 7, 8, and 9 represent the relation between number of frames (along horizontal axis) and number of assigned tracks in each appropriate frame (along vertical axis).

In Fig. 7, the initial estimation error is varied along the $[X Y]$ direction which the KF should accept before deeming a certain track as “unacceptable” and dropping it altogether. Table 1 represents all case studies of initial estimation error as a response for applying a gradual variation in its coordinate value according to a 500 frames test video. In each case, the average number of assigned tracks have been calculated in all frames as well as processing time. It is observed that the value of the initial estimation error does not have a significant effect on the number of assigned tracks (correct predictions) and all case studies remain unchanged. This is due to the fact that the KF adapts and changes its estimation error value based on input data.

Figure 8 is concerned with the change of motion noise according to four categories of observations. Table 2 represents all case studies of interest of motion noise. From the obtained results, it is observed that as for the motion noise, choosing a value that is above $[150, 150]$ would be unsuitable. This is due to the fact that increasing its value could lead into a great deviation from the motion model set which causes inaccuracy that appears as a drop in the values of the average number of assigned tracks. This drop causes some sort of wrong predictions which leads to inaccurate tracking.

Figure 9 is concerned with the changing of the measurement noise according to five categories of observations. Table 3 represents all case studies of interest. As for the Measurement Noise, increasing its value causes inaccuracy measurements for estimating the average number of assigned tracks. The number of assigned tracks is indeed increasing with the increase in its value. But, this increase could still have an adverse impact on the long run. From all previously discusses observations, a first stage fine-tuned

Table 1 Initial estimation error with [X Y] direction

Initial estimation error	Average number of assigned tracks	Processing time per 500 frame test video
10, 5	1.996	14.998908
20, 10	1.996	14.774024
30, 15	1.996	14.726030
40, 20	1.996	14.933885
50, 25	1.996	14.983851
60, 30	1.996	14.906489
70, 35	1.996	15.829720
80, 40	1.996	15.328098
90, 50	1.996	15.580287
120, 70	1.996	14.847166

Table 2 Motion noise

Motion noise	Average number of assigned tracks	Processing time per 500 frame test video
10, 10	1.996	15.270699
100, 25	1.996	15.233302
100, 100	2.002	14.852864
150, 150	2.006	14.961544
170, 170	2.004	15.242621
200, 200	2.004	15.400802
250, 250	2.004	15.255876
270, 270	2.004	15.074271
300, 300	2.004	14.801201
350, 350	2.004	15.286121
370, 370	2.004	15.198801
400, 400	2.004	14.815989

Table 3 Measurement noise

Measurement noise	Average number of assigned tracks	Processing time per 500 frame test video
10	1.996	15.171940
50	1.994	15.079669
100	1.996	15.043541
150	1.996	15.300675
200	2.002	15.121708
300	2.004	15.324122
350	2.006	14.982728
400	2.006	15.218703
450	2.006	14.905973

tracking criteria has been concluded as follow; by adjusting the performance indices of KF (for a 500 frames test video) as follow:

- Choose initial estimation error to have any coordinates below 500.
- Choose motion noise to have any coordinates below 150.
- Choose measurement noise to have any value below 150.

5.2 The with-noise case

Now, the analysis of non-ideal test video would be started by adding noise into test video. In addition, the previously discusses conclusions have been considered for the required modifications in the proposed GUI system. The appropriate modifications are needed to acquire the analysis of test video with different disturbances. In addition, the tracking accuracy has been utilized and tested again.

Table 4 Effect of salt and pepper degraded video

Salt and pepper noise percentage (%)	Number of assigned tracks	Processing time per 500 frame (s)
Zero noise	1.996	15.533471
5	2.056	21.989983
10	4.922	33.693454
20	8.858	55.756604
30	5.936	33.686794
40	5.438	33.280617
50	6.57	39.827585
60	5.896	46.375300
70	3.402	32.767757
80	0.996	26.726041
90	0.996	28.016424
100	0.996	28.807787

Table 5 Effect of Gaussian noise degraded video

Gaussian mean, variance	Number of assigned tracks	Processing time per 500 frame (s)
Zero noise	1.996	15.533471
0, 0.01	1.228	16.762822
0, 0.05	0.028	13.653209
0, 0.1	0.022	13.832700
0.5, 0.01	1.398	17.116468
0.5, 0.05	0.9	16.964905
0.5, 0.1	0.578	16.586056
1, 0.01	0.202	14.436344
1, 0.05	0.928	16.274630
1, 0.1	1.26	17.218930

Table 6 Effect of speckle noise degraded video

Speckle variance	Number of assigned tracks	Processing time per 500 frame (s)
Zero noise	1.996	15.533471
0.04	0.974	41.912283
0.05	0.976	19.138182
0.06	0.77	18.296054
0.07	0.61	18.330546
0.08	0.484	17.760295
0.09	0.306	17.024124
0.1	0.17	16.633673

The main purpose for that phase is to create simple simulation criteria for some of the most challenging weather conditions. Modifications has been utilized by adding salt and pepper noise, Gaussian noise, and speckle noise as three types of video disturbances with different levels of occurrence percentages. In addition, system of filters has been added to the interface in order to measure the required performance (correct tracking) after the cleaning processes. Many distinguishing filtering systems have been applied for each type of noise; average, maximum, minimum, wiener, disk, Laplacian, of Gaussian

(LoG), motion, sobel, prewitt, median, and Gaussian filters. Observations has been recorded as follow: In each case of salt and pepper noise, by increasing percentage of occurrence, observations have been targeted that the values of assigned tracks have been deviated from the ideal case value (zero noise). In addition, by adding a Gaussian noise with a changeable mean and variance values, the number of assigned tracks has been degraded than the ideal case (zero noise) however, these degradations could be assumed non

Fig. 10 Shows two snapshots for the interface of phase 2

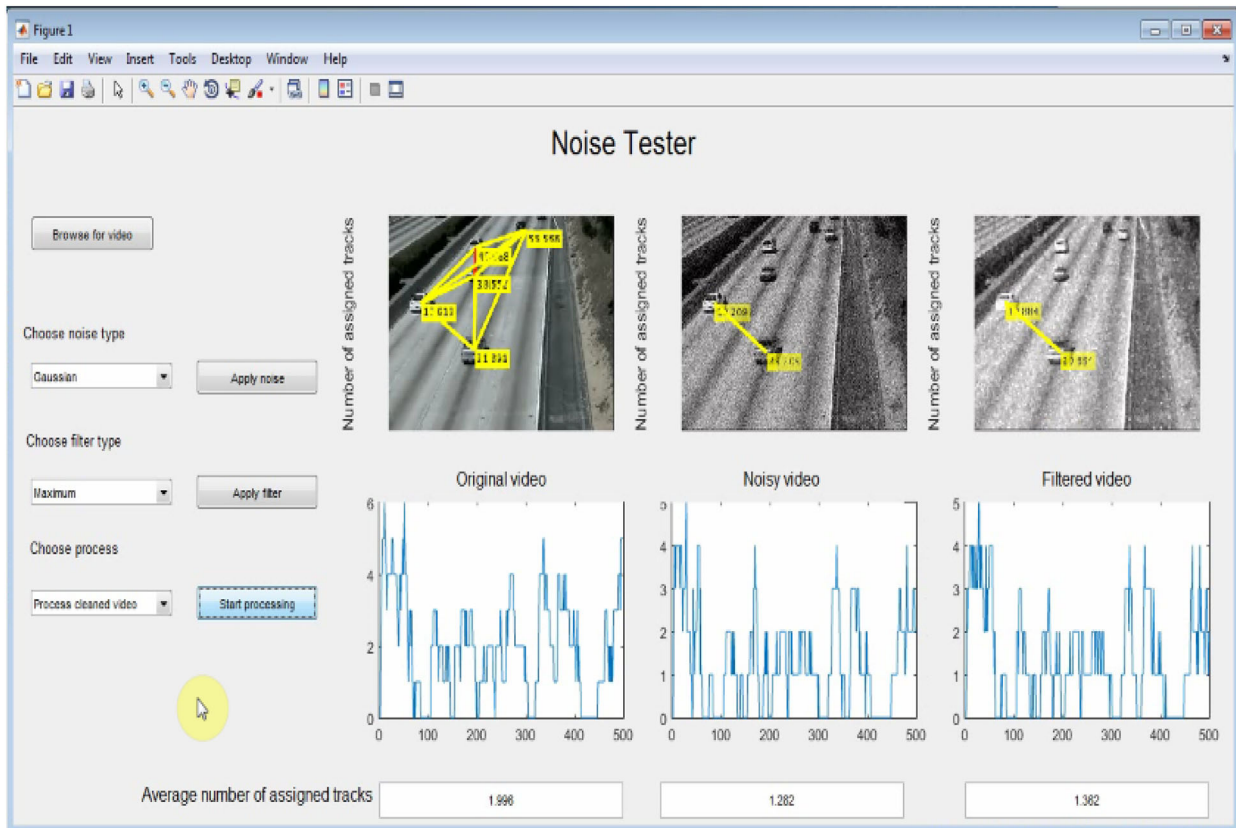


Table 7 The deviation from original value after cleaning

	Speckle	Gaussian	Salt and pepper
Average	1.399	1.022	0.368
Maximum	1.646	1.281	0.381
Minimum	1.798	1.107	0.526
Wiener	1.0419	0.701	0.065
Disk	1.021	0.775	0.056
Laplacian	1.487	1.217	1.708
LoG	1.229	1.065	1.408
Motion	1.355	0.867	0.276
Sobel	1.553	1.197	1.534
Prewitt	1.59	1.014	1.092
Median	0.661	0.469	0.276
Gaussian	1.5988	1.242	0.696

Table 10 Average number of assigned tracks after adding Speckle noise

Filter type	Variance						
	0.04	0.05	0.06	0.07	0.08	0.09	0.1
Average	1.636	1.632	1.605	1.592	1.552	1.51	1.494
Maximum	1.634	1.624	1.592	1.518	1.508	1.542	1.502
Minimum	1.436	1.45	1.464	1.392	1.382	1.356	1.284
Wiener	1.992	1.978	1.974	1.936	1.918	1.83	1.818
Disk	1.94	1.96	1.954	1.92	1.926	1.896	1.928
Laplacian	0.766	0.74	2.438	3.742	4.702	5.236	5.042
LoG	4.596	4.672	3.664	3.192	2.852	2.844	2.692
Motion	1.72	1.742	1.712	1.682	1.686	1.632	1.59
Sobel	1.73	1.76	2.882	3.818	4.666	5.156	5.228
Prewit	0.616	0.596	0.62	0.63	0.726	0.87	1.172
Median	1.738	1.706	1.688	1.688	1.684	1.62	1.636
Gaussian	1.48	1.474	1.376	1.316	1.204	1.162	1.088




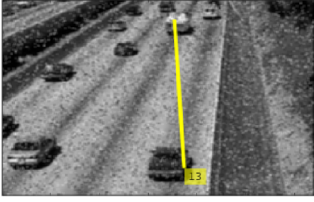
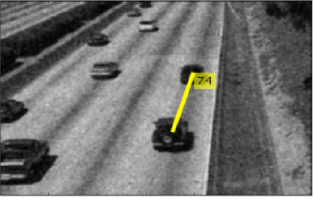

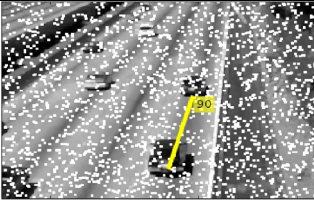


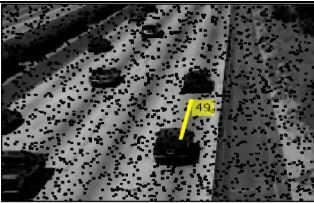
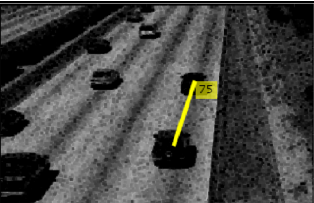
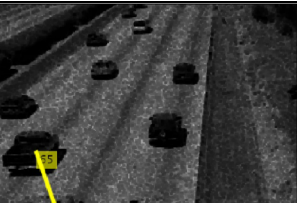
Table 8 Average number of assigned tracks after adding salt and pepper noise

Filter type	% of occurrence										
	0.05	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Average	1.606	1.518	1.096	0.684	0.236	0.022	0	0	0	0	0
Maximum	1.4	0.742	0.034	0.018	0	0	0	0	0	0	0
Minimum	0.232	0.114	0.018	0.012	0	0	0	0	0	0	0
Wiener	1.83	1.83	1.83	1.83	1.83	1.83	1.83	1.83	1.83	1.83	1.83
Disk	1.952	1.884	1.736	1.49	1.188	0.858	0.454	0.132	0	0	0
Laplacian	2.992	5.344	7.498	0.996	0.996	0.996	0.996	0.996	0.996	0.996	0.996
LoG	3.148	3.338	6.252	0.996	0.996	0.996	0.996	0.996	0.996	0.996	0.996
Motion	1.742	1.62	1.218	0.762	0.312	0.034	0	0	0	0	0
Sobel	11.3	1.662	0.996	0.996	0.996	0.996	0.996	0.996	0.996	0.996	0.996
Prewit	6.668	7.64	1.232	0.996	0.996	0.996	0.996	0.996	0.996	0.996	0.996
Median	1.706	1.706	1.692	1.664	1.626	1.616	1.586	1.474	1.264	4.338	0.032
Gaussian	1.604	1.58	0.558	0.29	0.088	0.052	0.046	0.054	0.038	0.03	0.03

Table 9 average number of assigned tracks after adding Gaussian noise

Filter type	Mean, variance								
	0, 0.01	0, 0.05	0, 0.1	0.5, 0.01	0.5, 0.05	0.5, 0.1	1, 0.01	1, 0.05	1, 0.1
Average	0.56	1.212	0.742	1.68	1.406	1.11	0.012	0.42	0.552
Maximum	1.697	0.788	0.726	0.96	0.404	0.016	0	0	0
Minimum	1.338	0.05	0.022	1.522	0.82	0.22	0.594	1.348	0.762
Wiener	1.972	1.738	1.404	1.842	1.84	1.766	0.006	0.398	0.58
Disk	0.616	1.81	1.62	1.902	1.848	1.758	0.006	0.442	0.66
Laplacian	1.228	0.028	0.022	1.398	0.9	0.578	0.202	0.928	1.26
LoG	2.784	0.996	0.996	2.648	5.168	1.022	0.594	1.788	3.594
Motion	1.784	1.3	0.856	1.796	1.518	1.224	0.02	0.452	0.61
Sobel	1.232	2.096	1.002	2.506	5.956	4.84	0.78	1.912	3.896
Prewit	0.706	2.926	5.172	2.216	1.87	4.654	0.632	1.704	2.114
Median	1.706	1.706	1.692	1.664	1.626	1.616	1.586	1.474	1.264
Gaussian	1.556	0.576	0.164	1.554	1.22	0.888	0.004	0.306	0.518

Table 11 Some snapshots from the system output in each case study

	Salt and pepper noise	Gaussian noise	Speckle noise
No filters			
average			
maximum			
minimum			

effective in cases of small mean value. Finally, in case of speckle noise, we can see that these degradations will be very small at large variance. Tables 4, 5 and 6 represents the effect of video degradation on tracking accuracy.

By adjusting the GUI interface in filtering video mode as shown in Fig. 10. New observations have been recorded after applying the cleaning methodology of each filter mask on each type of noise with its different percentage of occurrence. These observations have been listed in Tables 8, 9, and 10. In addition, the main target was to measure the efficiency of each filter mask with each type of noise disturbance with respect to different levels of occurrences. This could be accomplished by measuring the amount of deviation of the average number of assigned tracks between the zero noise test video (value was recorded approximately to be 1.996) and the noisy version. Those deviations have been summarized in Table 7 according to comparing the observations listed in Tables 8, 9, and 10 (calculations after cleaning) with the ideal.

Observations were as follow: in case of Speckle noise: The Wiener and Disk filters scored the least deviation from the original values, thus it appears that they are the most accurate and the processing times are approximately the same. In case of Gaussian noise: The Median filter showed the least deviation from the original values, thus it's the most suitable filter for this type of noise. However, its processing time was high. Finally, in case of Salt and pepper noise: The Median filter scored the least deviation from the original values, thus it's the most suitable one however its processing time was high. Table 11 represents snapshots from the system output for the applied three types of video noises after the cleaning process. From all previously discusses observations, a secondly stage fine-tuned tracking criteria has been recommended to use such median, wiener, and disk filter masks to discriminate between video degradations.

Table 11 continued

Gaussian			
laplacian			
LoG			
median			
sobel			
prewit			
motion			
disk			

6 Conclusion

Measuring and enhancing the tracking efficiency for moving vehicles in an urban video is an important challenging research points especially under abnormal weather conditions. A new system has been developed for calculating distance between adjacent vehicles as an application of vehicular tracking. Calculating such distance enhances safety function in order to provide an automated warning system for the drivers. Two phases based analysis have been observed in order to establish such system in both ideal and challenging weather conditions. Phase 1 is responsible for adjusting performance indices of KF parameters in order to achieve the best tracking in case of no noise test video. We recommend the values of initial estimation error, motion noise, and measurement noise to be below **one quarter** the total number of frames for the video of interest (in our case, we have used 500 frames test video and performance indices were below 150 for best tracking). For a noisy test video, the first procedure is to set KF parameters as discussed before. Then we recommend the use of wiener, disk, and median filters according to the type of disturbance of interest (use wiener and disk filter masking in case of speckle noise, use median filter in case of Gaussian noise, and use median filter in case of salt and pepper noise). Future work should be directed toward cascading filters with a focus on realistic conditions during evaluation taking in considerations the level of complexity. In addition, a comparison study would be targeted in the state of the art of using new algorithms for detection and tracking moving vehicles such as Otsu method, k-nearest neighbor's algorithm (k-NN), and Hidden Markov Model (HMM).

References

1. An S, Lee BH, Shin DR (2011) A survey of intelligent transportation systems. In: Proceedings of 2011 Third International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN). pp 332–337
2. Wang F (2010) Parallel control and management for intelligent transportation systems: concepts, architectures, and applications. *IEEE Trans Intell Transp Syst* 11(3):630
3. G Yan, W Yang, W Yang (2011) A secure and intelligent parking system. *IEEE Intell Transp Mag* 18
4. J Zaldivar, T Calafate, J Cano, P Manzoni (2011) Providing accident detection in vehicular networks through OBD-II devices and android-based smartphones. 5th IEEE workshop on user mobility and vehicular networks, on-move, pp 813
5. Zhang J, Wang F, Wang K, Lin W, Xu X, Chen C (2011) Data-driven intelligent transportation systems. *IEEE Trans Intell Transp Syst* 12(4):1624
6. Faouzi N, Leung H, Kurian A (2011) Data fusion in intelligent transportation systems progress and challenges—a survey. *Inf Fus* 4–10:5
7. Buch N, Velastin SA, Orwel J (2011) A review of computer vision techniques for the analysis of urban Traffic. *IEEE Trans Intell Transp Syst* 12(3):920–939
8. Sivaraman S, Trivedi MM (2013) Looking at vehicles on the road: a survey of vision based vehicle detection, tracking, and behavior analysis. *IEEE Trans Intell Transp Syst* 14(4):1773–1795
9. Sivaraman S, Trivedi M (2013) Integrated lane and vehicle detection, localization, and tracking: a synergistic approach. *IEEE Trans Intell Transp Syst* 14(2):907–917
10. S Sato, M Hashimoto, M Takita, K Takagi, T Ogawa (2010) Multilayer lidar-based pedestrian tracking in urban environments. In *proc. IEEE IV*, pp 849–854
11. Garcia F, Cerri P, Broggi A, Armingo JM, de la Escalera A (2009) Vehicle detection based on laser radar. Springer Verlag, New York
12. Xue L, Jiang C, Chang H, Yang Y, Qin W, Yuan W (2012) A novel Kalman filter for combining outputs of MEMS gyroscope array. Elsevier, New York, pp 745–746
13. Guo J, Williams BM, Huang W (2014) Adaptive Kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transp Res Part C Emerg Technol* 43(Part 1):50–51
14. Ko CN, Lee CM (2012) Short-term load forecasting using SVR (support vector regression)-based radial basis function neural network with dual extended Kalman filter. Elsevier, New York, p 413
15. Y Tian, Y Wang, X Chen, Z Tan (2014) Kalman-filter-based state estimation for system information exchange in a multi-bus islanded microgrid. Proceedings of the 7th IET international conference on power electronics, p 1
16. PL Houtekamer, X Deng, HL Mitchell, SJ Beak, N Gagnon (2014) Higher resolution in an operational ensemble Kalman filter. *IEEE*, p 1143
17. AH Cherif, K Boussetta, G Diaz, D Fedoua (2016) A probabilistic convergecast protocol for vehicle to infrastructure communication in ITS architecture. 13th IEEE annual consumer communications and networking conference (CCNC), pp 776–777
18. Y Yao, G Xiong, K Wang, Fe Zhu, FY Wang (2013) Vehicle detection method based on active basis model and symmetry in ITS. Conference on intelligent transportation systems, IEEE, 16th international, ITSC, pp 614–618
19. T Edwards, J Moore, M Loukadaki, P Jaworski (2011) A network assisted vehicle for ADAS and ITS testing. 14th international IEEE conference on intelligent transportation systems, pp 681–685
20. Sofia Janet R, Bagyamani J (2015) Traffic analysis on highways based on image processing. *Int J Comput Intell Inf* 5(1):14–23
21. Sivaraman S, Manubhai M (2013) Integrated lane and vehicle detection localization, and tracking: a synergistic approach. *IEEE Trans Intell Transp Syst IEEE* 14(2):906
22. Kamal MS, Chowdhury L, Khan MI, Ashour AS, Manuel J, Tavares RS, Dey N (2017) Hidden Markov model and Chapman Kolmogrov for protein structures prediction from images. *Comput Biol Chem* 68:231–244
23. MS Kamal, S Parvin, AS Ashour, F Shi, N Dey (2017) De-Brujin graph with MapReduce framework towards metagenomic data classification. *Int J Inf Technol* 9:59–75. <http://link.springer.com/article/10.1007/s41870-017-0005-z>. Accessed Mar 2017
24. Kamal MS, Dey N, Ashour AS, Ripon SH, Balas VE, Kaysar MS (2017) FbMapping: an automated system for monitoring facebook data. *Neural Netw World* 27:27–57
25. Kamal MdS, Nimmy SF, Parvin S (2016) Performance evaluation comparison for detecting DNA structural break through big data analysis. *Comput Syst Sci Eng* 31:275–289

26. Kamal MS, Dey N, Nimmy SF, Ripon SH, Ali NY, Ashour AS, Karaa WA, Shi F (2016) Evolutionary framework for coding area selection from cancer data. *Neural Comput Appl*. <https://doi.org/10.1007/s00521-016-2513-3>
27. MS Kamal, MI Khan, K Deb, L Chowdhury, N Dey (2016) An optimized graph based metagenomic gene classification approach: metagenomic gene analysis, chap 12. In: Dey N, Ashour A (eds) *Classification and clustering in biomedical signal processing*. IGI Global, *Advances in bioinformatics and biomedical engineering (ABBE) book series*. pp 290–314. <https://doi.org/10.4018/978-1-5225-0140-4>
28. S Ripon, MS Kamal, S Hossain, N Dey (2016) Theoretical analysis of different classifiers under reduction rough data set: a brief proposal. *Int J Rough Sets Data Anal (IJRSDA)* 5(1)
29. Kamal MS, Khan MI (2014) Performance evaluation of Warshall algorithm and dynamic programming for Markov chain in local sequence alignment. *Interdiscip Sci Comput Life Sci* 7(1):78–81
30. Lamba A, Kumar D (2016) Optimization of KNN with Firefly algorithm. *BIJIT BVICAM's Int J Inf Technol* 8(2):997–1003
31. Soundarya K (2014) Video denoising based on stationary wavelet transform and center weighted median filter. *BIJIT BVICAM's Int J Inf Technol* 6(1):722–726