International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:12, 2012

A Study on Algorithm Fusion for Recognition and Tracking of Moving Robot

Jungho Choi, and Youngwan Cho

***Abstract*—**This paper presents an algorithm for the recognition and tracking of moving objects, 1/10 scale model car is used to verify performance of the algorithm. Presented algorithm for the recognition and tracking of moving objects in the paper is as follows. SURF algorithm is merged with Lucas-Kanade algorithm. SURF algorithm has strong performance on contrast, size, rotation changes and it recognizes objects but it is slow due to many computational complexities. Processing speed of Lucas-Kanade algorithm is fast but the recognition of objects is impossible. Its optical flow compares the previous and current frames so that can track the movement of a pixel. The fusion algorithm is created in order to solve problems which occurred using the Kalman Filter to estimate the position and the accumulated error compensation algorithm was implemented. Kalman filter is used to create presented algorithm to complement problems that is occurred when fusion two algorithms. Kalman filter is used to estimate next location, compensate for the accumulated error. The resolution of the camera (Vision Sensor) is fixed to be 640x480. To verify the performance of the fusion algorithm, test is compared to SURF algorithm under three situations, driving straight, curve, and recognizing cars behind the obstacles. Situation similar to the actual is possible using a model vehicle. Proposed fusion algorithm showed superior performance and accuracy than the existing object recognition and tracking algorithms. We will improve the performance of the algorithm, so that you can experiment with the images of the actual road environment.

***Keywords*—**SURF, Optical Flow Lucas-Kanade, Kalman Filter, object recognition, object tracking.

I. Introduction

BECAUSE intelligent cars are spread a very wide range of applications in the technical aspects and merchantability, it is a key technology for the future of the high value-added automotive industry. Intelligent Safety Car is a smart car to maximize driving efficiency and safety. It collaborates with infrastructure such as transportation operating system by using electrical and electronic systems that includes sensors and actuators. These smart cars can be divided into two technical fields. First field is smart-safety-vehicle-technology to make vehicle itself smart. Second is a technology that can enhance the safety. It obtains the risk of information around the vehicle using vehicle-to-vehicle communications (V2V) or vehicles and infrastructure communication (V2I). Electronics control, IT technologies are required to implement on vehicles but cognitive skills should be preceded at the first step. As shown in Fig. 1, future safety cars are expected to recognize within 200m radius of the vehicle's surroundings, driver's condition and road condition or traffic accidents occurred within few kilometers. Recently, technologies are actively being commercialized to improve driver's safety and convenience. For example, lane-departure warning system and lane-keeping system are being commercialized in the transverse safety system. Besides, for longitudinal safety, forward collision warning system, forward collision mitigation system and more have been commercialized by a variety of names. Smart Safety Vehicle Technology has been developed from passive system such as pedestrian recognition, Collision Warning, Lane Departure Warning, drowsiness warning system to active system such as collision avoidance and damage reduction. To develop these safety systems, recognize lane or the vehicle in front the vehicle by using sensors installed on the vehicle and then evolve with technology that recognizes the vehicle in the rear. To recognize the vehicle, Rader, Vision sensor, Lidar, Ultrasound, Infrared sensors are used. Among them, vision sensors are sensitive to illumination changes and the weather changes however distance or transverse direction accuracy is excellent and the price is cheap, it's being fusion and studied with several types of sensors [6], [7].

In this paper, in a situation where traffic congestion, recognizing the particular car driving low-speed and keep track to verify image-processing algorithm for limited autonomous driving. Heavy traffic road environment is assumed to simulate desired auto drive by tracking specific vehicle and recognize accurately. 1/10 scale model car was used for the safety. In this paper, to develop an algorithm, simulation is performed under restrict circumstance that except illumination and weather changes because vision sensors are sensitive to illumination changes and the weather changes.

Jungho Choi is with the Seokyeong University, Jeongneung 4-dong, Seongbuk-gu, Seoul, Korea (e-mail: popmantion@skuniv.ac.kr).

Youngwan Cho is with the Seokyeong University, Jeongneung 4-dong, Seongbuk-gu, Seoul, Korea (corresponding author, e-mail: ywcho@skuniv.ac.kr).

II. Explanation SURF, LK and KF Algorithm

A. SURF Algorithm

SURF and SIFT algorithms are almost the same recognition, but SURF algorithm is faster about operations processing speed than SIFT.

The first of the proposed method to speed up is using Integral Image. The integral image in Fig. 1 relies to reduce the computation time and we therefore call it the Fast-Hessian detector. For example, if you want to know about the cumulative value of the brightness of the S region, you can just calculate quickly about A-B-C+D region.

$$\prod(x, y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(i, j) \quad (1)$$

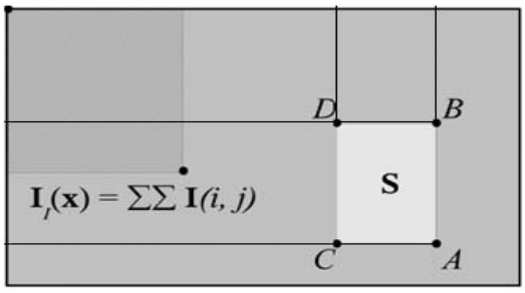


Fig. 1 Integral Image

The second, Detector and Descriptor are used simplification. The box filters in Fig. 2 are approximations for Gaussian second order derivatives with and represent lowest scale.

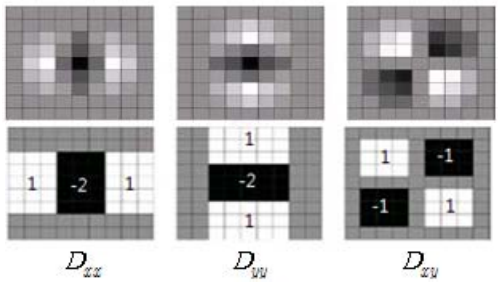


Fig. 2 The Gaussian Second order derivative filter and the approximate box filter

The Gaussian second order partial derivatives in x-direction, y-direction and xy-direction, and approximations thereof using box filters. The grey regions are equal to zero.

In order to be invariant to image rotation, we identify a reproducible orientation for the interest points. For that purpose, we first calculate the Haar-wavelet responses in x and y direction, shown in Fig. 3, and this in a circular neighborhood of radius 6s around the interest point, with s the scale at which the interest point was detected. Also the sampling step is scale dependent and chosen to be s. In keeping with the rest, also the wavelet responses are computed at that current scale s. Accordingly, at high scales the size of the wavelets is big. Therefore, we use again integral images for fast filtering. The dominant orientation is estimated by calculating the sum of all responses within a sliding orientation window covering an angle of. The horizontal and vertical responses within the window are summed. The two summed responses then yield a new vector. The longest such vector lends its orientation to the interest point. The size of the sliding window is a parameter, which has been chosen experimentally. Small sizes fire on single dominating wavelet responses, large sizes yield maxima in vector length that are not outspoken. Both result in an unstable orientation of the interest region [1], [3], [4].

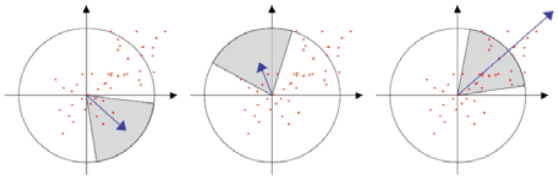


Fig. 3 SURF Descriptor to Invariant Rotation

B. LK Method

The basic idea of the Optical Flow's LK (Lucas-Kanade) algorithm rests on three assumptions.

The first is Brightness constancy. This means we assume that the brightness of a pixel does not change as it is tracked from frame to frame. It means that our tracked pixel intensity exhibits no change over time.

$$f(x,t) \equiv I(x(t),t) = I(x(t+dx),t+dt) \quad (2)$$

given by, $\frac{\partial f(x)}{\partial t} = 0$

Second, is temporal persistence. This means the temporal increments are fast enough relative to the scale of motion in the image that the object does not move much from frame to frame. Let's call the y component of velocity v and the x component of velocity u, then we have:

$$I_x u + I_y v + I = 0 \quad (3)$$

For this single equation there are two unknowns for any given pixel. This means that measurements at the single-pixel level are under constrained and cannot be used to obtain a unique solution for the two-dimensional motion at that point. Instead, we can only solve for the motion component that is perpendicular or "normal" to the line described by our flow equation. Fig. 4 presents the mathematical and geometric details.

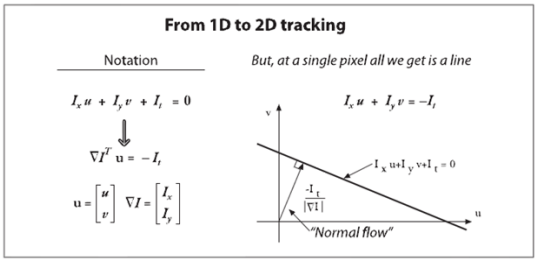


Fig. 4 Normal Flow

Normal optical flow results from the aperture problem.

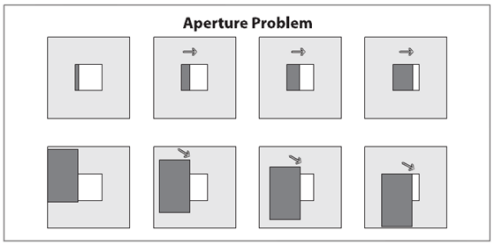


Fig. 5 Aperture problem

In Fig. 5, through the aperture window (upper row) we see an edge moving to the right but cannot detect the downward part of the motion (lower row). We turn to the last optical flow assumption for help. If a local patch of pixels moves coherently, then we can easily solve for the motion of the central pixel by using the surrounding pixels to set up a system of equations. For example, if we use a window of brightness values around the current pixel to compute its motion, we can the set up 25 equations as follows.

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} \quad (4)$$

To solve for this system, we set up a least-squares minimization of the equation, whereby $\min \| Ad - b \|^2$ is solved in standard from as:

$$(A^T A)d = A^T b \quad (5)$$

given by, $d = \begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b$

When $(A^T A)$ is invertible, we can solve and get u and v [2], [5].

C. Kalman Filter

In order to use a Kalman filter to remove noise from a signal, the process that we are measuring must be able to be described by a linear system. Many physical processes, such as a vehicle driving along a road, a satellite orbiting the earth, a motor shaft driven by winding currents, or a sinusoidal radio-frequency carrier signal, can be approximated as linear systems. A linear system is simply a process that can be described by the following two equations:

State equation:
$x_{k+1} = Ax_k + Bu_k + w_k$

Output equation:
$y_k = Cx_k + z_k$

In the above equations A, B, and C are matrices; k is the time index; x is called the state of the system; u is a known input to the system; y is the measured output; and w and z are the noise. The variable w is called the process noise, and z is called the measurement noise. Each of these quantities are (in general) vectors and therefore contain more than one element. The vector x contains all of the information about the present state of the system, but we cannot measure x directly. Instead we measure y, which is a function of x that is corrupted by the noise z. We can use y to help us obtain an estimate of x, but we cannot necessarily take the information from y at face value because it is corrupted by noise. The measurement is like a politician. We can use the information that it presents to a certain extent, but we cannot afford to grant it our total trust. For example, suppose we want to model a vehicle going in a straight line. We can say that the state consists of the vehicle position p and velocity v. The input u is the commanded acceleration and the output y is the measured position. Let's say that we are able to change the acceleration and measure the position every T seconds. In this case, elementary laws of physics say that the velocity v will be governed by the following equation:

$$v_{k+1} = v_k + Tu_k \quad (6)$$

That is, the velocity one time-step from now (T seconds from now) will be equal to the present velocity plus the commanded acceleration multiplied by T. But the previous equation does not give a precise value for vk+1. Instead, the velocity will be perturbed by noise due to gusts of wind, potholes, and other unfortunate realities. The velocity noise is a random variable that changes with time. So a more realistic equation for v would be:

$$v_{k+1} = v_k + Tu_k + \tilde{v_k} \quad (7)$$

Where $\tilde{v_k}$ is the velocity noise. A similar equation can be derived for the position p:

$$p_{k+1} = p_k + Tv_k + \frac{1}{2}T^2 u_k + \tilde{p_k} \quad (8)$$

Where $\tilde{p_k}$ is the position noise. Now we can define a state vector x that consists of position and velocity:

$$x_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix} \quad (9)$$

Finally, knowing that the measured output is equal to the position, we can write our linear system equations as follows:

$$x_{k+1} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} T^2/2 \\ T \end{bmatrix} u_k + w_k \quad (10)$$

z_k is the measurement noise due to such things as instrumentation errors. If we want to control the vehicle with some sort of feedback system, we need an accurate estimate of the position p and the velocity v. In other words, we need a way to estimate the state x. This is where the Kalman filter comes in.

III. Implementation Using Parallel Processing of SURF, LK and KF Algorithm

However, SURF to find the desired object, shows the excellent effect, it is impossible to tracking of object because of too many volume of operations in real-time. In contrast, however, it is impossible to recognize the object, LK to track

found object, shows the excellent effect. If you need to implement for use as both algorithms by Single-Thread, you may perform LK randomly repeated after object recognition by SURF. If the method was used, it is impossible to track object until performed SURF when missing object during tracking. Also the number of execution of SURF raises to improve recognition during the same time. We recommend Multi-Thread to complement this part so that it can be increased recognition more the number executions of SURF than Single-Thread. At the same time, it can be tracking of object by performing a combined LK in real-time.

In MFC application, Thread is separated into two types and it is used in two ways. One of the two is Worker Thread and the rest is User Interface Thread. When you perform tasks in the background, Worker-Thread is used because there's no message loop so it doesn't need user's input. On the contrary, User Interface Thread is used with user's input because there's message loop.

User Interface Thread is used for perform the merge of SURF and LK algorithm.

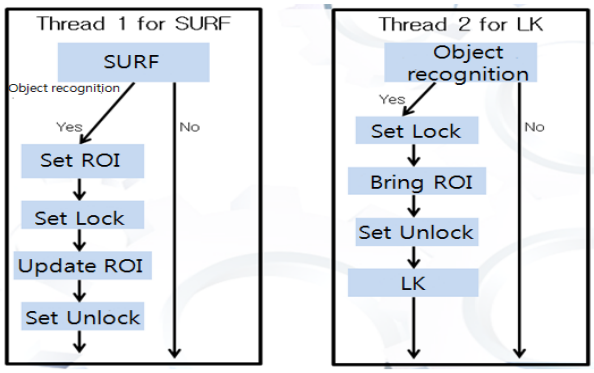


Fig. 6 Flow Chart of Multi-Thread

Thread 1 is used for implementation of SURF and Thread 2 is used for implementation of LK in Fig. 6. The first, Thread 1 is performed to recognition of object. If object were not recognized, Thread 1 is performed again until recognition of object, and Thread 2 is waiting. When object was found by SURF of Thread 1, set a ROI (Region Of Interest). Then coordinates of ROI are updated to share with LK of Thread 2. LK of Thread 2 is tracking from using updated coordinates after recognition of object in SURF of Thread 1 [8], [9].

However, when object is moved fast, it does not determine the exact location of moving object position.

Kalman Filter cannot recognize the position of object in the image. Kalman Filter can estimate the exact location from input the position of the object by image processing techniques. First, SURF can find the position of the moving object.

Kalman Filter usually uses for remove noise or not measure the value by estimation. The role of the Kalman Filter is the same for tracking of moving object. Kalman Filter is used to estimate the movement speed and remove the location errors from obtaining SURF and LK algorithm. Position-velocity model should be extended to the two-dimensional for object tracking on a plane. State variables are derived by horizontal direction (x-axis) position-velocity and vertical direction (y-axis) position-velocity.

$$x = \begin{bmatrix} position_x \\ velocity_x \\ position_y \\ velocity_y \end{bmatrix} \tag{11}$$

Oreder of the state variables is not important from the equation (11). If we define as state variables equation (11), system model is given by equation (12).

$$x_{k+1} = Ax_k + w_k \tag{12}$$

$$z_k = Hx_k + v_k \tag{13}$$

$$A = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{14}$$

The matrix A is substituted to equation (14), Direction of the x-axis and y-axis can be expressed as the determinant of each of the following equation (15).

$$\begin{bmatrix} position_{k+1} \\ velocity_{k+1} \end{bmatrix} = \begin{bmatrix} position_k + velocity_k \times \Delta t \\ velocity_k \end{bmatrix} + w_k \tag{15}$$

And to sum up, matrix H is assignment to equation (13), it means measurement of the position and the direction of the x-axis and y-axis. The system model of Kalman Filter to track moving object is given by equation (15). This model is an extension of a two-dimensional plane of the relationship between the position-velocity, and only the position of the moving object from SURF and LK algorithm is input Kalman Filter. However, the fusion of SURF and LK algorithm with multi-thread has a problem that the cumulative error caused by LK part makes it difficult to track fast-moving object. We used the error estimation function of Kalman Filter in order to overcome this problem. The proposed structure of Kalman Filter fusion of SURF and LK is as shown following Fig. 7.

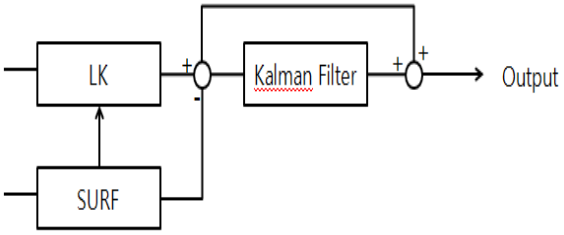


Fig. 7 Applying Kalman Filter

SURF and LK will perform the same using existing multi-thread algorithm. However, it is difficult to exact object tracking using moving average, when tracking moving object through LK algorithm. So the cumulative errors of LK algorithm are compensated through Kalman Filter.

Coordinates of the center is update from recognize object through SURF, LK algorithms tracks moving object by the coordinates of the center of object. Kalman Filter works like LK algorithm and coordinates of the center is corrected by new coordinates of the center through SURF.

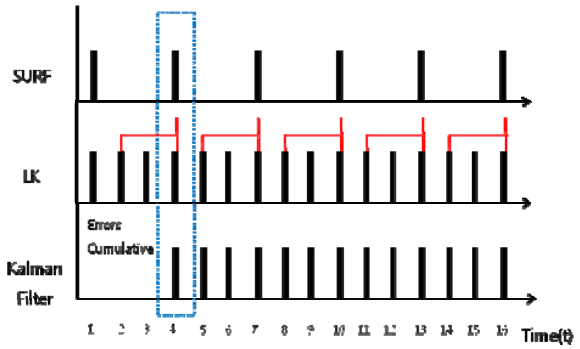


Fig. 8 Synchronization of SURF, LK and Kalman Filter

LK algorithm is performed three times while SURF algorithm is performed one time, because processing speed of SURF algorithm is too slow. When Time(t) is 1, if we update the coordinates of the center of the object through the SURF, LK algorithm is tracking the position until next perform SURF. However when Time(t) 2, 3, 4 during tracking object causes cumulative errors by LK algorithm, so we update again by SURF when time 4. This paper suggest that the coordinates of the center of the object is update by SURF when Time(t) 4, LK algorithm is tracking the position and error during tracking by LK is corrected by Kalman Filter [10].

IV. Experimental Results

To determine the speed of the fusion algorithm, SURF algorithm is compared. The time it takes to process the input images (500 frames, 640x480 resolutions), was used for performance evaluation.

TABLE I
Comparison of Algorithm Execution Time and Testing Count

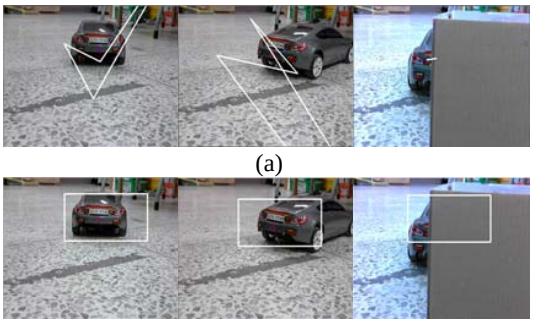
Algorithm	Execution time	Testing count
SURF	84.166	500
Fusion algorithm	16.716	500

When two algorithms are compared with the same resolution and the number of images, Fusion algorithms is approximately 5 times faster. SURF algorithm is approximately 4 ~ 6 fps (fps: frame per second) performance. Fusion algorithm shows that the performance of about 26 ~ 27fps. In other words, the convergence of real-time moving object recognition and tracking algorithms are available. Experimental results in a real-time input video, because of the slow processing speed of SURF algorithm, moving object image resolution is compromised. Due to the damaged resolution, object recognition is impossible.

To verify the performance of the fusion algorithm, test is compared to SURF algorithm under three situations, driving straight, curve, and recognizing cars behind the obstacles with 1/10 scale model car. SURF algorithm performs excellent on recognition of objects through the creation of a 64-dimensional representation of characters.ORB (2011) or BRISK (2011) algorithm represents with the binary bits so it is faster, but the recognition rate is slightly less than SURF algorithm. To compare the recognition and tracking accuracy, the experiment was in a particular situation except for the speed. Model vehicle for the experiment is shown in Fig. 9.

Fig. 9 Car Model

Fig. 10 shows a comparison of the performance in the following three situations. From left to right, photo show driving straight, curve driving, car behind the obstacles.



(a)

(b)

Fig. 10 Performance comparison for three different situations

Fig. 10 (a) is the result of implemented SURF algorithm, Fig. 10 (b) is the result of the implementation of the fusion algorithm. As shown in the Fig. 10, SURF algorithm does not recognize when the vehicle close to the model moves away. Fusion algorithm can keep track of the object through LK and KF even though it cannot recognize through SURF algorithm. Also based on z-axis when the vehicle was driving curve, SURF algorithm cannot recognize whereas the fusion algorithm track continuously. When the vehicle is behind the obstacle, it can be confirmed that fusion algorithm success on object tracking.

V. Conclusion and Future Work

We have this paper presents an algorithm for the identification and tracking of the vehicle in front, and to verify the performance. Recognize and track moving objects has been

actively studied in many areas, they are being commercialized. But there are many difficulties to recognize and track a specific object Because of the surrounding environment or the speed of a moving object. In order to solve this problem, recognition of moving objects and Tracking of speed with How to increase the accuracy is presented by proved algorithm. SURF algorithm is used for the recognition of objects. SURF algorithm, the processing speed is slow but it is strong in Rotation, contrast, size changing outstanding performance in the recognition of specific objects. But because of computational complexity, it has disadvantage that is slow processing speed. For tracking objects, disadvantage of SURF algorithm has been complemented with LK algorithm. It is impossible for LK algorithm to recognize a particular object. Its optical flow compares the previous and current frames so that can track the movement of a pixel. Also to solve the error accumulation when fusion SURF and LK algorithm, estimate next position of an object using the Kalman filter and through the correction of the error, Algorithm is designed to track the object more accurately. In order to verify the performance of fusion algorithm, SURF algorithm that has great performance on specific object recognition is compared. Speed and performance of both SURF algorithm and fusion algorithm is confirmed under the experiment that three major situations which can be happened on real road, using the 1/10 scale model car. There are three situations of the vehicle in front, driving straight, curve, and recognizing cars behind the obstacles. The resolution of the camera (Vision Sensor) used for the experiments is 640x480. Algorithm is easy to apply on using a model vehicle for the experiment. To verify the performance, there are no problem for ensuring the safety of the experiment, and the experimental space. In this paper, to recognize and track the vehicle in front, fusion SURF, LK, KF algorithm are used for object recognition and tracking. Fusion algorithm has better performance compared to the existing algorithms but, 100% reliable result and more experiment should be performed. In future research, to enhance performance of recognition and tracking algorithms, experiment the performance by using the images of the actual road environment should be better.

Acknowledgment

This work (Grants No. C0034062) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2012.

References

[1] H. Bay, E. Andreas, T. Tuytelaars and L. V. Gool, "Speeded-up robust features", Computer Vision and Image Understanding, Vol 110, Issue 3, pp 346-359, June 2008.
[2] Gary Bradski and Adrian Kaehler, “Learning OpenCV” O’REILLY, pp. 322-329, 2009.
[3] Lindeberg, “Feature detection with automatic scale selection,” IJCV. 1998.
[4] D. G. Lowe, "Distinctive image features from scale invariant keypoints", International Journal of Computer Vision, Vol. 60, No. 2 pp. 91-110, 2004.
[5] Parah H. Batavia, Dean A. Pomerleau and Chuck Thorpe, “Evertaking Vehicle Detection using Implicit Optical Flow”, Proceedings of the IEEE Transportation Systems Conference, pp. 729-734, 1997.
[6] Konstantinos G. Derpainis, “The Harris Corner Detector”, Cot, 2004.
[7] E. Bublee, C. Rabaud, K. Konolige, and G.Bradski, “ORB: an efficient alternative to SIFT or SURF”, International Conference on Computer Vision, Nov, 2011.
[8] John G. Allen, Richard Y. D. Xu and Jesse S. Jin, “Object tracking using CamShift algorithm and multiple quantized feature spaces” In ACM International Conference Proceeding Series; Vol 100, pp.3~7, 2004.
[9] U. C. Jung, S. H. Jin, X. D. Pham, J. W. Jeon, J. E. Byun, H. Kang, “A real-time object tracking system using a particle filter”, 2006 IEEE/ RSJ Int. Conf. vol. 9, pp. 2822~2827, 2006. 10.
[10] H. W. Sorenson, “Least-square estimation:from Gauss to Kalman”, IEEE Spectrum, vol. 7. pp. 63~68, July 1970.