International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:2, 2012

Performance Improvement of Moving Object Recognition and Tracking Algorithm using Parallel Processing of SURF and Optical Flow

Jungho Choi, Youngwan Cho

***Abstract*—**The paper proposes a way of parallel processing of SURF and Optical Flow for moving object recognition and tracking. The object recognition and tracking is one of the most important task in computer vision, however disadvantage are many operations cause processing speed slower so that it can't do real-time object recognition and tracking. The proposed method uses a typical way of feature extraction SURF and moving object Optical Flow for reduce disadvantage and real-time moving object recognition and tracking, and parallel processing techniques for speed improvement. First analyse that an image from DB and acquired through the camera using SURF for compared to the same object recognition then set ROI (Region of Interest) for tracking movement of feature points using Optical Flow. Secondly, using Multi-Thread is for improved processing speed and recognition by parallel processing. Finally, performance is evaluated and verified efficiency of algorithm throughout the experiment.

***Keywords*—**moving object recognition, moving object tracking, SURF, Optical Flow, Multi-Thread.

I. Introduction

In latest, many sensors and camera are used for object around recognition and tracking in robot and automotive sector. Among them, the importance of computer vision research is emphasized because people getting more than 99% of information depend on the eyes. if people can get accurate information of the acquired image for recognition object, it can be replaced with many sensors or enhance the utilization.

So far object recognition and tracking algorithms have been developed and used in computer vision sector. There are algorithms for object recognition, such as SIFT and SURF, for object tracking, such as Optical Flow and camshift. However, there's many problems for real-time object recognition and tracking by the algorithms. Priority, object recognition algorithm to exact recognize is required on many operations because of it is extracted feature points and created descriptor. It causes slow processing speed and limitations of real-time object recognition.

First of all, SURF algorithm was used for object recognition in this paper. However, SURF and SIFT features of the algorithm are very similar to the exact object recognition by feature points matching through a camera, SIFT is slow by too many operations processing, it makes disadvantage for real-time object tracking.

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

Youngwan Choi is with the Seo Kyeong University, corresponding author, Seo-Gyeong Univ., Jeongneung 4-dong, Seongbuk-gu, Seoul, Korea (e-mail: ywcho@skuniv.ac.kr)

In contrast, SURF has been improved by optimizing the operations processing. Nevertheless, there's a limit to process real-time object tracking. Optical Flow's LK(Lucas-Kanade) algorithm has been implemented in conjunction in this paper. LK is a way to keep track of an image motion between the previous frame and current frame. there's a problem that tracking object recognized by the SURF using LK. So I recommend that using Multi Thread for parallel processing with SURF and SIFT so that you can recognize and track object with no problem.

The paper is organized as follows: Chapter 2 is about explanation SURF and LK algorithm. Chapter 3 is about implementation using parallel processing of SURF and LK algorithm. Chapter 4 is about experimental results and Chapter 5 is conclusions.

II. Explanation SURF and LK 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.

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

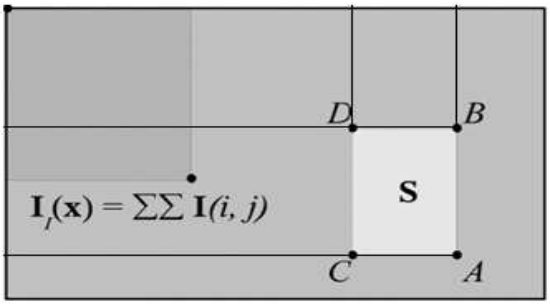

Fig. 1 Integral Image

The second, Detector and Descriptor are used simplification. The 9*9 box filters in Fig. 2 are approximations for Gaussian second order derivatives with and represent lowest scale.

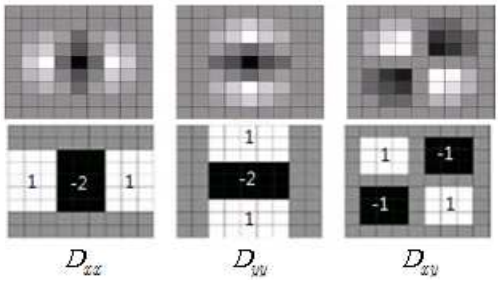

Fig. 2 The gaussian Second order derivative filter and The approximative box filter

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 rotation, identification 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 neighbourhood 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 $\frac{\pi}{3}$. 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.

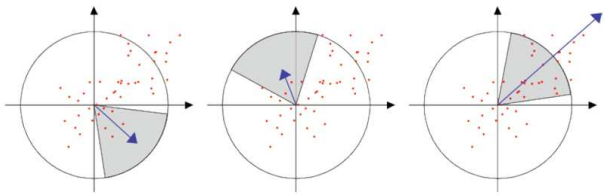

Fig. 3 SURF descriptor to invariant rotation

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.

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

given by, $$\frac{\partial f(x)}{\partial t} = 0 \quad (2)$$

The 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 underconstrained 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.

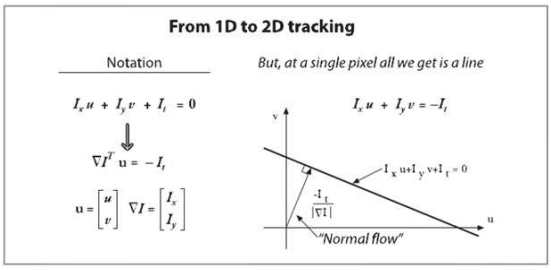

Fig. 4 Normal Flow

Normal optical flow results from the aperture problem

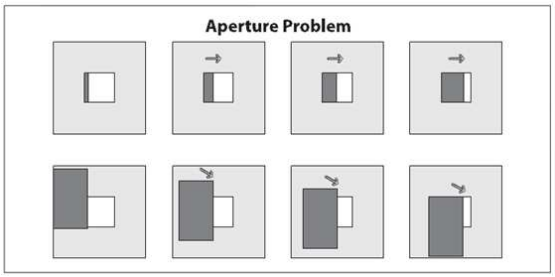

Fig. 5 Aperture problem

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

$$(A^T A)d = A^T b$$

given by, $d = \begin{bmatrix} u \\ v \end{bmatrix} = (A^T A)^{-1} A^T b \quad (5)$

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

III. Implementation Using Parallel Processing of SURF and LK 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 raise 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 way. 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.

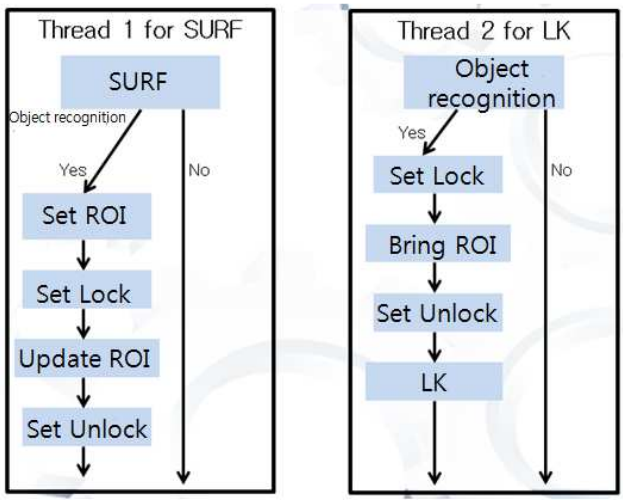

Fig. 6 Flow chart of Multi-Thread

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.

IV. Experimental Results

Table I represents the number of performed and execution time of SURF and LK algorithm when using the same video (16 second, 492 frame, 640×480 resolution)

TABLE I
The Number of Performed and Execution Time(second) of SURF and LK

Algorithm	Execution Time	The Number of Performed
SURF	86.229	492
LK	15.335	492

Through the above table, execution time takes approximately 0.175 second and 0.031 second each SURF and LK algorithm.

Table. 2 is described how much execution time and the number of performed take time in Single-Thread and Multi-Thread. After SURF is performed once, LK is performed 10 times repeatedly in Single-Thread to compare Single-Thread and Multi Thread.

TABLE II
THE NUMBER OF PERFORMED AND EXECUTION TIME(SECOND) OF SURF AND LK IN SINGLE-THREAD AND MULTI-THREAD

Thread	Algorithm	Execution Time	The Number of Performed
Single Thread	SURF	21.98	45
	LK		447
Multi Thread	SURF	14.778	62
	LK		385

Execution time is different between in Single-Thread and Multi-Thread. There's a lot of difference when the number of performed is compared between SURF in Single-Thread and Multi-Thread. In Single-Thread, SURF is performed once on approximately 0.49 seconds. But in Multi-Thread, SURF is performed once on approximately 0.24 seconds. It means performed of SURF in Multi-Thread more than 2 times in Single-Thread. On the other hand, LK is performed almost same in Single-Thread and Multi-Thread. It is impossible to recognize and track the object in real-time in Single-Thread. It took more time in Single-Thread, because there're too many load to execute, whereas it took less time in Multi-Thread. It means that object recognition and tracking is possible without problems in Multi-Thread.

Fig. 7 Result Images of Object Recognition and Tracking

It is the result images of object recognition and tracking in real time by using Multi Thread technique. It is impossible to recognize object using SURF because of inaccurate resolution when the object in image is moving fast, however, it is tracking object exactly by LK.

V. CONCLUSIONS AND FUTURE WORK

We have proposed method of object recognition and tracking in real-time using Multi-Thread with SURF and LK. We can recognize and track moving object in real time by using parallel processing of SURF and LK. The disadvantage of SURF that slowness of processing speed of object recognition has been solved with using Multi Thread and LK together so the speed of recognition object got faster and the accuracy has been improved.

For future work, we will further investigate generality of the proposed system. We plan to extend more complex environment like moving camera with background.

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] J. H. Duncan, and T. C. Chou, "Temporal edges: The detection of motion and the computation of optical flow," in Proc. IEEE 2nd int. Conf. Computer Vision, Florida, USA, Dec. 1988, pp.374-382
[7] S. Denman, V. Chandran, and S. Sridharan, "Adaptive Optical Flow for Person Tracking," in Proc. Digital Image Computing: Techniques and Applications, Caims, Australia, Dec. 2005, pp. 44-50