Fast and Effective Multiple Moving Targets Tracking Method for Mobile Robots

Jiyoon Chung and Hyun S. Yang†
Center for Artificial Intelligence Research, Dept. of Computer Science, KAIST
373-1 Kusong-dong, Yusong-ku, Taejon, 305-701, KOREA

Abstract

In this paper we describe a tracker that provides real-time visual feedback using on-board low-cost processors. The proposed tracker is based on the Two Stage Visual Tracking Method (TSVTM) which consists of real-time kernel, image saver, database, and vision module. Real-time kernel based on the Earlist-Deadline-First scheduling policy provides the capability of processing tasks with time constraints within the deadline. Image saver takes the responsibility for keeping all the incoming images until they can be processed. Database keeps both the estimated and the predictive location, velocity, intensity, etc of each region that makes up the target.

Vision module consists of two modules: the First-Stage Vision Module (FSVM) and the Second-Stage Vision Module (SSVM). The FSVM processes the whole image to initially recognize targets using the sophisticated vision algorithms while the SSVM can easily find and track them using the focus-of-attention strategy based on Kalman filter since the SSVM knows the approximated location and useful features of the targets. Combining the above four mechanisms effectively, TSVTM can track targets every one thirtieth of a second.

1 Introduction

Visual information is one of the most powerful sources of sensory information available to autonomous mobile robots. However existing visual perception systems are potentially too slow for real-world domains. When a robot wants to accomplish vision tasks, it lacks computational ability since the requirements of the vision task increase rapidly in proportion to the performance improvement of a computer. There are several solutions to make up for this computational shortage. One of the most popular solutions is for a robot to have special purpose hardwares designed to accomplish a few specific vision tasks. The following are two examples of this. HERMIES III[5] equipped with an NCUBE hypercube computer has been used in experiment to clean a simulated chemical spill. It weighs 820kg plus the weight of the batteries which is about 410kg. Extasy-II[6] has a *Vision Processor* (VP) supported by 16 CPUs to track the fixed red landmark to make the correspondence problem easier.

In recent research, many robots have been designed to efficiently use their resources by integrating vision sensors and actuators with a visual perception system[7]. This system processes only the interesting parts of an image and reduces the size of data to be processed. As a result a robot has fewer computational requirements and its cost and size are pared down. This method has an inherently large variation of response times according to the content of the information available at that time.

Some robots were designed to operate without any special purpose hardwares. However, they can not usually use a vision sensor as an absolute sensor due to the lack of the computation speed. Instead they use ultrasound or infrared sensors as a relative sensor, since these sensors do not require a massive computation. Such robots can not get the correct position due to the unbounded accumulation of errors. Therefore, many researchers must have assumptions[3] or active beacons[9] to obtain the information about the environment. Unfortunately, they are either very slow or inaccurate[8], or even unreliable[10]. In order to navigate autonomously in an unknown space, a robot must have a vision capability to obtain its absolute position. However, it is difficult for all robots to be equipped with a computer fast enough to process vision tasks since we must consider their cost, weight, size, and so on. For these reasons, we have studied a method that robots can perform vision tasks without having to employ special purpose hardwares.

Research in computer vision has traditionally em-

†To whom the correspondence should be sent

IEEE International Conference on Robotics and Automation
0-7803-1965-6/95 $4.00 ©1995 IEEE

phasized the paradigm of image understanding. Many researchs in the field of robotics have also studied to reconstruct the scene, by which a robot can reduce its positional uncertainty. This approach requires the precise measurement of the position so vision systems are characteristically viewed as computationally expensive, error-prone, and difficult to calibrate. However, if vision is used to measure error which is the visual distance between a target and a pre-defined position with respect to the visual coordinate, then the system is much less dependent on calibration. Hence it is extremely robust in calibrating error. By tracking multiple moving targets and observing their errors a robot can easily know its position. Some works have been reported and conducted on the use of vision information for tracking in the dynamic feedback loop [4], [11]. These works are focused on the control of a robot, rather than on the visual perception. This paper mainly describes the tracking method and briefly the control of a robot.

1.1 System overview

Since robots we are considering don't have any special purpose hardwares, they must use the method of lightening the load. In other words, they must reduce the search space, by using the information collected from the previous executions. The contents of the information affect computational need of a robot that is necessary to accomplish the vision tasks. Therefore, We can reduce this needs by dividing the vision task into two stages according to the information available. The first vision module will be computationally more expensive and complex since it has to create a lot of information to recognize the targets. However, the second vision module may easily find targets. This is because the previous executions provide the information about the approximate location and useful features of the targets such as the shape and intensity of each region that compose a target. The computational requirement in our method is shown in Figure 1. The distance between the dotted lines in the figure indicates robot's ability. The computational requirement of the first run for a vision task is several times higher than that of the others. Since it may exceed the robot's ability, it might not meet the certain deadline. If a robot adopting our method can sacrifice the responsiveness of the vision tasks and if the large variation of the response time can be treated reasonably, then a robot can accomplish a vision task at a real-time rate.

When we assume that the vision system of a robot operates as in Figure 1. the second and third runs can-

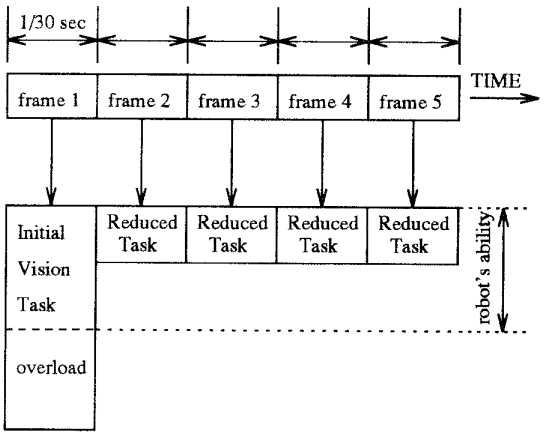

Figure 1: Diagram showing the computational requirement for vision tasks

not be executed on time since the first run of the task uses more computation than allowed. The images that are grabbed while the first image is being processed will be lost unless the images are kept in the system until the first one is finished. So our method has the *image saver* which keeps the images until they can be processed. Several tasks achieving behaviors are involved in the vision task. They are used to inquire the results and to issue the instructions. Consider a situation in which a robot uses the vision system as an absolute sensor to correct its position. The robot must wait for the results of the vision task to become available unless the variation of the response time is carefully treated. Therefore, we have developed a real-time kernel to execute multiple tasks within its specified time constraint. With the real-time kernel several tasks such as the *image saver* and a *motor control loop* are scheduled and the mechanism to communicate with each other is also provided.

2 The two stage visual tracking method (TSVTM)

The TSVTM consists of real-time kernel, image saver, database and vision module. The vision module is divided into two modules according to the information available. They are the *first-stage vision module* (FSVM) and the *second-stage vision module* (SSVM). The FSVM has to process the whole image to recognize targets, so it needs a lot of computation time. However, the SSVM can easily find and track them by focusing on interesting parts of an image, which have

already been obtained from the previous execution of either the FSVM or the SSVM. Since the SSVM knows the approximate location and useful features of the targets, it can achieve a fast response time. Therefore, the overall computational requirement of the TSVTM becomes less strict. Since the FSVM needs more computation time than given, the incoming images during the execution will be lost. Because of this we designed the *image saver* to take responsibility for keeping all the incoming images until they can be processed. The database keeps both the estimated and the predictive location, velocity, intensity, etc. of each region that makes up the target. Using the information in the database the SSVM verifies its result by using the predefined constraints. Whenever a fault is discovered, it sends a signal to the real-time kernel to invoke the FSVM.

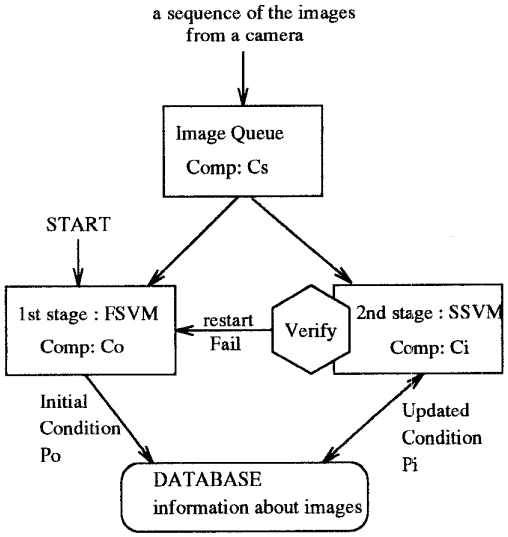

Figure 2: Overview of Two Stage Visual Tracking Method

The tasks, including the TSVTM, are scheduled with their own time constraints. The *image saver* has a very strict time constraint since it has to put an incoming image from the camera into the queue every one thirtieth of a second. The FSVM has no time constraint at all. However, it should be as fast as possible, because it affects the response time of the TSVTM. The SSVM has a time constraint, but violation of the deadline does not make the overall system fail since the *image saver* will save the incoming images.

Figure 2 shows an overview of the TSVTM. There are three computing modules. Each of them has its own computational requirements: C_0, C_i and C_s denote the computation time of FSVM, SSVM, and the *image saver*, respectively. C_f denotes the computation time equivalent to the time that it takes to grab an image, one thirtieth of a second.

There are two constraints in the TSVTM.

$$C_i + C_s < C_f \tag{1}$$

$$\left\lceil \frac{C_0 - (C_f - C_s)}{C_f - C_s} \right\rceil < \frac{S_m}{I_s} \tag{2}$$

Eq. 1 shows the constraint on the computation time of the SSVM and the *image saver*. The summation of them must be less than C_f, or it will cause the length of the queue in the *image saver* to grow unboundly. Eq. 2 shows the constraint on memory which is necessary for the *image saver* to store the images into the queue. The denominator on the left side of Eq. 2 is the given computation time for the FSVM and the numerator is the excess of the computation time over given. Thus, the *image saver* must have the same number of image buffers as the left side of the equation. We assume that the size of an image buffer is I_s, so the minimum memory space of the TSVTM should be larger than S_m.

$$\frac{C_0 - (C_f - C_s)}{C_f - (C_s + C_i)} \tag{3}$$

The TSVTM will have the fastest response time only after the SSVM has been invoked as many times as in Eq. 3 from the last invocation of the FSVM. Therefore, the TSVTM is well suited for the applications which need to track targets in real-time and endure a slightly longer initial response time.

3 An implementation of the TSVTM

Existing trackers can be divided into two groups, depending upon the measurement data used. One group is based on the optical flow field, while the other is based on the correspondence of discrete features such as points, lines, and contours. The optical flow field technique is usually time-consuming and it is difficult to recover the structure of the scene when both the camera and the target move independently. Much of the earlier work involved using two or three frames of images, which is nonrobust and numerically unstable[2]. In this section, we will describe the implementation of the *tracker* as an example of the TSVTM. The *tracker* partly alleviates these problems by the use of a continuous sequence of image frames. Most

tracking tasks are divided into two portions: acquisition and tracking itself. Therefore, they correspond exactly with the FSVM and SSVM in the TSVTM.

The *tracker* trys to track multiple targets such as the one in Figure 3 by using two big eyes, an open mouth and two horns. Tracking is accomplished by the recognition of a target, rather than the correspondence of discreate features. Therefore, the correspondence problem is removed so the tracker is reliable and robust.

3.1 The EDF-based real-time kernel

The *tracker*'s performance depends not only on the logical results of the computation, but also on the time in which the results are produced. For example, the *image saver* must be performed within one thirtieth of a second. Otherwise, the image in the frame grabber will be lost. In addition, a robot has many intrinsic periodic activities such as low-level servo-loop, trajectory planning, etc. They must be executed within strict time constraints in order to guarantee the stability of the system. A robot also has aperiodic activities that must be completed within their deadlines. To support both the *tracker* and the other tasks, we have designed a real-time kernel based on the *Earlist-Deadline-First* (EDF) scheduling policy. This means that, at all times, a processor is assigned to the task whose deadline is the closest. The kernel has three priority classes, $HARD$, $SOFT$ and NRT (Non Real-Time). A $HARD$ task has the highest priority and is either a periodic or a sporadic process with a critical deadline. A $SOFT$ task has fewer strict time constraints and is either periodic or sporadic. A NRT has no time constraints at all and is the lowest priority. The kernel works as follows: as long as there are runnable processes in higher priority class, just each task has been scheduled with EDF fashion for its own *maximum estimated execution time* (MET). The kernel never bothers with lower priority classes.

3.1.1 Process communication

To provide communication facilities among processes, we developed two mechanisms: *shared port* and *shared queue*. *Shared port* was designed for applications interested in the latest message only, while *shared queue* was designed for the one needed to store unread messages. *Shared port* is fast and asynchronous. Its message is non consumable and can be overwritten. It always has the latest messages in its buffer. It is useful in many tasks such as a control loop and a sensory acquisition processes. *Shared queue* is the same as a normal queue except that it has several pointers for deleting an element in the queue. A task that wants to access *shared queue* must acquire access permission with a pointer. A pointer is necessary in deleting an element from the queue. If a task with access permission does not delete elements, the length of the queue will grow unboundly. To prevent this, we have designed it to automatically delete an out-dated element whenever an overflow of the queue occurs.

3.2 The First Stage Vision Module

To recognize a target, an image is segmented into a set of regions. To do this we use one of the conventional region-based segmentation algorithms called *Partition-Mode-Test* (PMT) [1]. The PMT algorithm is quite fast because it performs image segmentation by scanning the whole image one time. Features can then be extracted from each region. These include the centroid, mean intensity, area, aspect ratio, minimal boundary rectangle and shape of the region. We can reduce the number of regions by limiting their feature values. For example, we can assume that an area of a region that makes up a target must exceed 40 pixels. The remaining regions are then labeled by heuristics. Even though many remarkable methods in image labeling have been developed, we do not adopt these methods of labeling since we prefer the fast response time of the system as opposed to the quality. The procedure listed below shows the FSVM of a tracker.

1. An image is segmented into a set of regions by the PMT.
2. Features of each region are extracted.
3. Regions with unqualified feature values are filtered out.
4. Candidates are made by grouping regions in accordance to their geometrical relation.
5. The target can be found by matching the feature values of each candidate to the predefined ones.

After the target has been found, we must write the information necessary in the database for the SSVM to track it. The information includes the position of the object in the image plane, the mean intensity, size and aspect ratio of each region that composes the object.

3.3 The Second Stage Vision Module

Using the information acquired from the FSVM or from the previous run of the SSVM, the SSVM can

limit what a robot has to look for and where it has to look for it. Some region-based segmentation algorithms such as the PMT [1] treat the whole image, so it is not suitable in the second stage. Another method called *region growing* can segment only interesting areas. From the given seed points, it can grow regions by appending each seed point to those neighboring pixels which have a similar intensity. It has difficulty forming of a stopping rule and choosing a set of seed points. Fortunately, these problems are easy for us to solve, since we have enough clues to choose the seed points and to make the stopping rules. Since the database has kept the estimated and predictive information, we can easily determine them. The predictive centroid of each region serves as the seed point while the mean intensity and size of the region as the stopping rule. Only one candidate is made by the SSVM and it must be tested by matching the predefined feature values. Passing the test does not guarantee a proper tracking. Since the SSVM sometimes tracks false targets, the target must be periodically verified. Whenever a fault is discovered, it sends a signal to the real-time kernel to invoke the FSVM.

3.3.1 The prediction of seed points

Since the *region growing* make different results depending on their seed points, we have to choose the seed points that are the closest to the centroid of the region to which it belongs. Each seed point is in the 2-D position on the image plane. Its velocity and acceleration are unknown and can not be measured since it moves by different control system such as another robot or a human body. To describe it, we use the Brownian motion model, since, from the robot's point of view, the target object moves around randomly. As a result the velocity in the state may be affected by a white noise $u(t)$: $\dot{v} = u(t), u(0) = 0, u \sim (0, O)$. The Kalman filter problem is formulated in terms of two equations 4 and 5. In a linear system, this takes the form:

$$x_{k+1} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_k + w_k \tag{4}$$

$$y_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x_k + n_k \tag{5}$$

where the matrix in Eq. 4 is called the state transition matrix denoted A, and the one in Eq. 5 is called the measurement matrix denoted C. The state is $x = [p_x\ p_y\ v_x\ v_y]^T$, with $p_x(t)$ and $p_y(t)$ the positions, $v_x(t)$ and $v_y(t)$ the velocities. T is the sample period, one thirtieth of a second. This system has no control input. $w(t)$ is a state noise and $n(t)$ is a measurement noise with

$$w(t) \sim (0, Q), n(t) \sim (0, R), \tag{6}$$

$$Q = diag\{0.001T, 0.001T, 3T, 3T\}, R = 0.1/T. \tag{7}$$

The initial position covariance is taken in the appropriate units as 1. Then $x(t) \sim (\bar{x}(0), P_0)$,

$$P_0 = diag\{1, 1, 1/T^2, 1/T^2\}, \tag{8}$$

where $\bar{x}(0) = [p_x(0)\ p_y(0)\ 0\ 0]^T$, $p_x(0)$ *and* $p_y(0)$ are the position of the object given by the FSVM. Initial velocities may be taken as zeros since we can not measure them.

Now, we can predict the object as a set of seed points. The system modeling in the Eq. 4 is used to predict the next state as a seed point. Using the state at time sample k, the predictive state at time $k+1$ denoted $\hat{x}_{k+1}^-$ is predicted:

$$\hat{x}_{k+1}^- = A\hat{x}_k. \tag{9}$$

The predicted error covariance matrix can also be computed at this stage:

$$P_{k+1}^- = AP_kA^T + Q. \tag{10}$$

We can start the region growing with $\hat{x}_{k+1}^-$. If the segmented regions are proven to the object whose position on the image plane is $y(t) = [x_i, y_i]^T$, then the measurement update will be made as follows:

$$P_{k+1} = P_{k+1}^- - P_{k+1}^- C^T (CP_{k+1}^- C^T + R)^{-1} CP_{k+1}^- \tag{11}$$

$$\hat{x}_{k+1} = \hat{x}_{k+1}^- + P_{k+1}C^T R^{-1}(y_{k+1} - C\hat{x}_{k+1}^-) \tag{12}$$

4 Summary and Discussion

We have implemented and tested several applications of the TSVTM on the intelligent mobile robot CAIR-2, as seen in Figure 3. Some experimental results of visual tracking will be in the **video proceedings** of this conference.

In this paper, we have proposed a real-time visual tracking method and introduced the intelligent mobile robot CAIR-2. Even though the tracker was designed to recognize and track several moving targets simultaneously, fast special purpose hardwares are not necessary to accomplish it. Therefore, we can make a robot

Figure 3: CAIR-2 following a target using the proposed visual tracker

with the real-time visual tracker smaller and cheaper by employing this method.

To overcome its inherently large variation of response time coming from the two stage strategy, a real-time kernel and the image saver were developed to manage the resources efficiently. The image saver takes responsibility for keeping all the incoming images until they can be processed, therefore this method is able to track targets every one thirtieth of a second. Many applications are not required to meet a deadline in a whole time. In many industrial applications, when an object is about to appear in view of the camera, the robot will have an opportunity to find it and track it before it disappears from view. Therefore, the proposed visual tracker can play an important role in many fields where initial response time does not really matter.

In addition, the tracker was designed to cooperate with the other tasks. Whenever it does not find a target in the camera's view, it requests a certain task to change the view. For example, imagine that a robot is looking in the opposite direction of the targets. Even though the targets are near by, it can not see them. When this happens, the tracker cooperates with the task that is designed to reason the locations of targets.

We have developed several application programs based on the TSVTM. Some of them were exhibited outdoors for three months for real world demonstrations during world EXPO '93 held in Taejon, Korea, 1993.

References

[1] M. S. Suk and S. M. Chung, "A New Image Segmentation Technique Based on Partition Mode Test," *Pattern Recognition*, Vol. 16, No. 5, pp. 469-480, 1983

[2] J. K. Aggarwal, "Motion and time-varying imagery - An overview," in *Proc. IEEE Workshop on Motion: Representation and Analysis*, Kiawah Island, SC, May 1986

[3] Roman Kuc and M. W. Siegel, "Physically based Simulation Model for Acoustic Sensor Robot Navigation," *IEEE trans. on PAMI*, Vol. PAMI-9, No. 6, Nov. 1987

[4] J. T. Feddema and C.S.G. Lee, "Adaptive image feature prediction and control for visual tracking with a hand-eye coordinated camera," *IEEE Trans. on Syst. Man Cybern.*, vol 20, no. 5, pp. 1172-1183, 1990

[5] D. B. Reister et. al., "DEMO 89 - The initial experiment with the HERMIES-III robot," *IEEE conf. on Robotics and Automation*, pp. 2562-2567, 1991

[6] I. S. Kweon, et. al., "Behavior-Based Mobile Robot using Multiple sensors," *First Korea-Japan Joint Conf. on Computer Vision*, pp. 260-267, 1991, Seoul

[7] Douglas A. Reece and S. Shafer, "Using Active Vision to Simplify Perception for Robot Driving," Technical Report, CMU-CS-91-199, Nov., 1991

[8] I. J. Cox, "Blanche - An Experiment in Guidance and Navigation of an Autonomous Robot Vehicle," *IEEE Trans. on Robotics and Automation*, vol.7, no. 2, pp. 193-204.

[9] Lindsay Kleeman, "Optimal Estimation of Position and Heading for Mobile Robots Using Ultrasonic Beacons and Dead-reckoning," *IEEE conf. on Robotics and Automation*, pp. 2582-2587, May 1992, Nice, France

[10] I. Congdon, "CARMEL Versus FLAKEY - A Comparison of Two Winners," *AI Magazine*, Winnter, 1992, pp. 49-56

[11] G. D. Hager, W. C. Chang and A. S. Morse, "Robot Feedback Control Based on Stereo Vision: Towards Calibration-Free Hand-Eye Coordination," *IEEE conf. on Robotics and Automation*, pp. 2850-2856, 1994, San Diego