MVA '98 IAPR Workshop on Machine Vision Applications, Nov. 17-19, 1998, Makuhari, Chiba, Japan

13—27 Design and Implementation of Real Time System for Object Detection and Classification on Parallel Virtual Machine

Tati L. Mengko, Trio Adiono, Handoko Setyawan, Rini Setiadarma, Donny A. Hudiansyah
EE Eng. Department, Bandung Institute Of Technology
Ganesha 10, Bandung 40132, Indonesia ; tel : 62 22 2509173 ext 3229
e-mail: tmengko@ibm.net, tadiono@elka-gtw.ee.itb.ac.id,
{handoko,rini,donny}@students.itb.ac.id

abstract–This paper addresses the problem of implementing an object detection and classification system on a parallel virtual machine. Detection process was implemented using deformable template algorithm. Canny edge algorithm used to provide edge information need in template matching. The system gives descriptions about the shape, size, and orientation of the objects. To be implemented in industrial process, the system needs to meet certain time constrain. Previous research proved, the system's speed-performance is proportional to the number of object detected and model-used. Use of parallel virtual machine makes the system run 3.5 times faster compared to previous system using single processor.

1. Introduction

Automation has long been a technology used in many industrial processes. Not only does it make the overall process faster, but it also results in more accurate operation, especially in processes where accuracy is of concern. Many industries adopted this technology by creating human-like machines, that is machines capable to mimics human abilities.

Human vision plays an important role in enabling human to interact with their environment. It acts as a sensor to gather information from their surrounding. There has been many research devoted to implement this capability into machines. In this paper, we attempt to design a visual sensor to be implemented in many industrial processes such as pick and place application and automatic fuel docking for spacecraft. As a sensor, this system is able to segment and identify objects of interest from the stationary complex background.

Template matching is the most commonly used method to accomplish this task. However, it is not robust against changes of intensity and object's shape deformation. One of more general method is deformable template algorithm, which is used in our system as the basic algorithm. Robustness to noise and changes of intensity become the advantages of this algorithm that enable the system to precisely define the shape, size, position, and orientation of multiple objects.

The foremost process to do is generating templates based on the system's database and extract edge information from the image. These templates will be used to detect object existence in the image. Instead of searching objects in every location possible in the image, we first specify searching areas, that is locations within the image where objects are most probably found. This early process proved to reduce the number of computation greatly. Morphological and bug following algorithms are chosen to locate the coordinates of the object's center of gravity and also detect how many objects there are in the system.

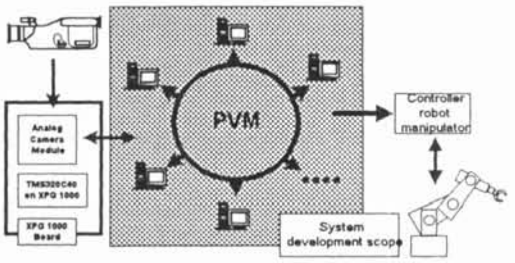

Figure 1 : Future system Implementation

The number of objects detected in an image affect the system's speed performance. The more object detected, the more computation need to be done by the system. Solely reducing the number of computation did not make the system meets real-time criteria, especially if there are too many objects in one image. The fact that identification process involves many sub-processes that are not related to each other gives us a possibility to solve this real-time problem by performing parallel computation on several processor or computers. The objective is to minimize the process time required. It should be noted that this paper emphasizes on optimizing parallel processing algorithm used in the system.

The organization of the paper is as follows: section 2 describes deformable template algorithm, the basic algorithm in the system, while section 3 describes Parallel Virtual Machine. The brief description of process performed in object detection and classification system will be described in section 4. Included in this section are the algorithms used to detect the existence of the objects and to define search area for each object. The algorithms used to match the template into each object detected previously are also described in section 4. System implementation in parallel virtual machine will be described in section 5. Representative result of applying the system to various objects are given in section 6. Images used as the data are 256x256 pixels gray-level images. Section 7 describes directions for future work.

2. Deformable template

Deformable template is a parametrical template that is designed to represent an object shape and used in segmentation and identification process. In order for the template to fit the object, it should be flexible enough to change its shape, size and orientation according to the object. Prior knowledge of object's shape is required to represent the object as a template. Deformable template algorithms consists of two factors [1]:

a. ***Prior probability density function***
This function represents how close the deformed template matches the ideal template based on a priori knowledge of the object.

b. ***Likelihood probability density function***
This function represents how close the deformed template matches the real object on the image based on object edge information.

Few assumptions are made on this system. The first one is the system uses static camera, so that the object will be seen from one direction. The second assumption is, since the system will be implemented in industrial process where the object's shape has been previously described and not much deformed, prior probability density function can be neglected so that likelihood probability density function will be the only factor concerned.

The probability density function use in this system is defined with eq. 3-7 [1].

$$U_E(\varphi_k, Z) = \frac{\sum_{(i,j)\in\varphi_k} \left| \|\nabla I\|(i,j) x h\left(\vec{\varphi}_k . \vec{\nabla} I(i,j)\right)\right|}{\sqrt{\sum_{(i,j)\in\varphi_k} \|\nabla I\|^2(i,j)} \sqrt{\sum_{(i,j)\in\varphi_k} h^2\left(\vec{\varphi}_k . \vec{\nabla} I(i,j)\right)}} \qquad (1)$$

$$h(x) = \begin{cases} 1-|x| & -1 \le x \le 1 \\ 0 & else \end{cases}$$

$$\|\nabla I\|(i,j) = \sqrt{\nabla I_x^2(i,j) + \nabla I_y^2(i,j)} \qquad (2)$$

$$\vec{\nabla} I(i,j) = \frac{1}{\|\nabla I\|(i,j)} \left(\nabla I_x(i,j), \nabla I_x(i,j)\right)^t \qquad (3)$$

$$\|\varphi_k\| = \sqrt{(X_{k+1} - X_k)^2 + (Y_{k+1} - Y_k)^2} \qquad (4)$$

$$\vec{\varphi}_k = \frac{1}{\|\varphi_k\|} \left(X_{k+1} - X_k, Y_{k+1} - Y_k\right)^t \qquad (5)$$

U_E	=	Likelihood energy
$\|\nabla I\|(i,j)$	=	Image gradient magnitude
φ_k	=	Model segment
$\vec{\nabla} I(i,j)$	=	Image gradient unity vector
Z	=	Image

3. Parallel Virtual Machine

Parallel processing, the method of having many small tasks solve one large problem, has emerged as a key enabling technology in modern computing. One method of parallel processing which have low cost, high performance, and sustained productivity is distributed computing. Distributed computing is a process whereby a set of computers connected by a network, are used collectively to solve a single large problem. PVM is a tool which is issued by MIT that unite several low-performance computers to create kind of virtual super computer. PVM is made to solve a complicated computational problem into small and simple computations, such as image processing operation (convolution, optimization, etc)

The PVM software provides a unified framework within which parallel programs can be developed in an efficient and straightforward manner using existing hardware. PVM enables a collection of heterogeneous computer systems to be viewed as a single parallel virtual machine.

The PVM computing model is based on the notion that an application consists of several tasks. Each task is responsible for a part of the application's computational workload. Sometimes an application is parallelized along its functions; that is, each task performs a different function, for example, input, problem setup, solution, output, and display. This process is often called functional parallelism. A more common method of parallelizing an application is called data parallelism. In this method all the tasks are the same, but each one only knows and solves a small part of the data. This is also referred to as the SPMD (single-program multiple-data) model of computing. PVM supports either or a mixture of these methods.

4. Description of methods

The method consists of three sub-processes. First, detecting edge information of the image, second, locating the search area and third, fitting the template in the right shape and orientation over each object found by the system.

4.1. Edge detection

The algorithm used in this process is Canny edge algorithm. The reason to use this algorithm is because not only does it produce edge magnitude of the image, it also produces edge gradient. The system needs both information to avoid detection-error caused by object deformation.

4.2. Search area localization

This process starts by subtracting input image with the background image to eliminate the background from the image. It will produce an image that has low-intensity objects and high-intensity background. The produced image is referred to as clean image.

The manipulation of the template is centered on its center of gravity, which is matched to the objects' centers of gravity. Therefore, if the system could define the center of gravity for each object, it would only need to manipulate the template around the predicted area instead of having to do the manipulation across the entire image area.

In order to remove unnecessary information that still exist, the "clean image" need to be adaptively threshold to produce a binary image, followed by dilating the image once, and subtracting it with the original binary image to obtain objects boundaries. The system use bug following algorithm to trace the boundaries and find the center of gravity of each object.

We define 3x3 pixels area around the center of gravity of an object to be the template searching area. This area will act as starting point of object detection process using deformable template. This process will reduce computation needed from 78643200 (without using searching area) to 10800 computations, and the overall process become 7280 times faster.

It can be seen that edge detection and search area localization are two independent processes. Although both using the same image source, they don't change original image and process the image in two different ways. Thus, in this research we split the processes to be executed in separate processor to speed up the overall performance.

4.3. Object detection process

Object detection process was performed by comparing templates recorded on database to the objects. The objects are classified based on matched template. This process performed three basic template manipulations, namely shifting, scaling and rotating. First the template placed over the image, where its center of gravity closed to the first pixel which defined previously as a member of first object's searching area.

Template manipulation process begins with scaling process. The system also rotating every template produced from scaling process. The increment of scaling and rotating depends on system requirement. Lower the increment level means higher system's precision. After scaling and rotating processes finish, at that location, the system will shift template, based on its center of gravity, to the next search-area point, and repeated until it covers all search area of the object. This process is repeated for each template in the database.

The result of this process is the description of the shape, size, position and orientation of the best-matched template for this object, which will represent the properties of the real object. Matching process described above will be done for each object detected on the image.

It can be seen that all the process done in object detection process could be divided in many ways, to be executed in parallel processors. It could be done by dedicating each processor to handle one object, dedicating each processor to run one template for every objects detected in the image, or using other configurations. Using this way, as long as we could add the number of processors used, we can always maintain the speed performance of the system even in the case of many objects.

5. Implementing system on PVM

The entire process sequence of the system consist of 3 sub-processes:

1. Detecting edge information
2. Search area localization
3. Object Detection

As mentioned above, edge detection and search area localization could be executed separately. The third sub-process, object detection, could only be executed after the system has finished both first and second processes. The third process itself could be executed in parallel in many configurations, based on fact that the computation is repeated for every object using every template recorded.

After examined several configurations for parallel system implementation, we decided that dedicating each processor to do detection process for one object using every template recorded, as the best configuration for the system. This configuration was chosen based on the fact that there were more objects than templates in the system

Topology of this system is star connectivity, which means slaves is only connected to master, and message passing is done between master and slaves, and vice versa, so there won't be message passing among slaves.

The system was tested with a set of images with different condition, such as number of objects, objects placement, and image background. The maximum number of object used is 4 objects, and the number of template used is 2 templates. Computers used in this experiment are UNIX based PC, using Intel Pentium 100 MHz - Intel Pentium Pro 200 MHz processor speed. We divide the computers into two groups, master and slave. Master computer is a computer that has the responsibility to decide how many slave computers used and the task done by each of them. Slave computer is a computer used to do computation needed by the system depend on master's plan. Basically each slave is designed to have the capability to perform same function and computation.

There must be one master for the whole system, and there will be several slaves depending on system configuration. In this case, each computer is designed to identify only one object, so there will only be one slave process in one computer. The number of computer used depends on the number of the object found in the image. If master finds N objects in input image, it will spawn N-1 slaves in N-1 computers for object identification, and will identify the last object by itself. The reason to use this configuration is that the number of template used in this experiment smaller than the number of object found in the image.

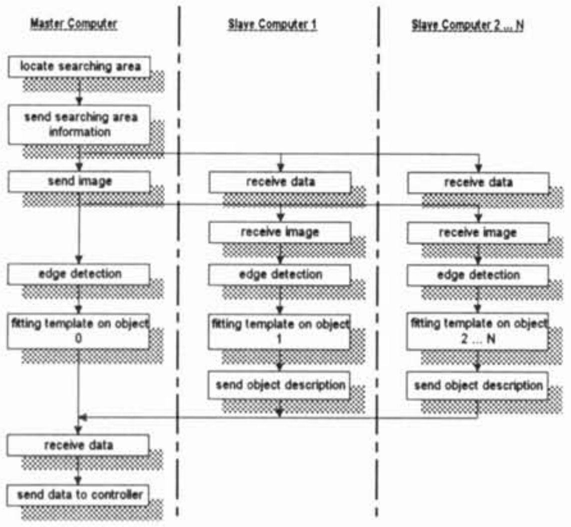

Figure 3 System Implementation on PVM

Edge detection process was executed next on every slave activated previously. We design all slaves to execute this process instead of dedicating one processor to execute the process because of transmission overhead.

By the fact that the time consumed by search area localization process is very small compared to the time consumed by edge detection process, we decide to use this approach to avoid transmission problem.

6. Experimental results

This system was tested on a set of 40 images which represented various object size, shape, and orientation. They also represent various numbers of objects lie on the image. The results of this test were used to measure the system's persistence and speed.

Current experiments have focused on detecting simple geometrical objects with less than 8 vertex.

The more object to be identified, the more time consumed by system. On sequential system, adding more object will add 2700 computations to be done sequentially, which is the same as about 3 seconds. Implementing that system into PVM will decrease time consumed by system processes, by the way using more computer to solve those computations.

Experimental result shows adding one object will increase 0.21 second consumed by system processes, which is 15 times as fast as previous system. This increasing is merely caused by image transmission from master to slave.

Total time consumed by system in identifying 4 objects is 5.84 seconds, which is 3.5 times as fast as previous system.

7. Future works

Having identified what appears to be a fairly robust result, our current major priority is creating a more general system that is able to detect objects undergoing more complex deformation. Thing also considered is implementing this system in real-life application such as automatic robot.

8. Conclusions

Use of Parallel Virtual Machine proved to speed up the overall identification process in the system by distributing unrelated computations into several processors. System's speed mostly depends on the number of objects found in the image. Using this method, the system is capable to maintain its speed performance in confronting great number of objects by adding the number of processor or computers used.

From the experimental result, it could also be concluded that the problems occur in implementing the system were data transmission period between computer used, and waiting periods between each process.

REFERENCES

[1] Marie-Pierre, S Lakshmanan, Anil K. Jain, "Vehicle Segmentation and Classification Using Deformable Templates," Vol. 18 No.3 March 1996, IEEE Trans. on Pattern Analysis and Machine Intelligence

[2] J.R. Parker, "Algorithms for Image Processing and Computer Vision," John Wiley & Sons, Inc., 1997

[3] Al Geist,"PVM : A Users' Guide and Tutorial for Networked Parallel Computing," The MIT Press Cambridge, Massachusetts London, England, 1994

[4] T. Mengko, T. Adiono, H. Setyawan, R. Setiadama, "Design and Implementation of Object Detection and Classification System Based on Deformable Template Algorithm", to be presented in IEEE APCCAS'98, Chiangmai, Thailand.

[5] H. Setyawan, Sistem Identifikasi objek berbasis deformable template (*Object Identification based on Deformable Template)*, unpublished thesis, ITB, 1998.